PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

03/15/2007 MKAYPAGH 00000146 11662704

01 FC:2631	150.00 OP
02 FC:2642	200.00 OP
03 FC:2633	100.00 OP
04 FC:2681	125.00 OP
05 FC:2615	475.00 OP
06 FC:2614	300.00 OP

PTO-1556
(5/87)

U.S. Government Printing Office: 2002 — 459-267/60011

IAP15 Rec'd PCT/PTO 14 MAR 2007

PTO-1390 (Rev. 07-2005)
Approved for use through 03/31/2007. OMB 0651-0021
U. S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A SUBMISSION UNDER 35 U.S.C. 371	ATTORNEY'S DOCKET NUMBER M0025.0369/P369
	U.S. APPLICATION NO. (if known, see 37 CFR 1.5) 11/662704

INTERNATIONAL APPLICATION NO.	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/GB2005/003554	14 September 2005	14 September 2004

TITLE OF INVENTION
A SPEECH RECOGNITION CIRCUIT AND METHOD

APPLICANT(S) FOR DO/EO/US
Guy Larri et al.

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. [x] This is a **FIRST** submission of items concerning a submission under 35 U.S.C. 371.
2. [ ] This is a **SECOND** or **SUBSEQUENT** submission of items concerning a submission under 35 U.S.C. 371.
3. [x] This is an express request to begin national examination procedures (35 U.S.C. 371(f)). The submission must include items (5), (6), (9) and (21) indicated below.
4. [x] The US has been elected (Article 31).
5. [x] A copy of the International Application as filed (35 U.S.C. 371 (c)(2))
   a. [x] is attached hereto (required only if not communicated by the International Bureau).
   b. [x] has been communicated by the International Bureau.
   c. [ ] is not required, as the application was filed in the United States Receiving Office (RO/US).
6. [ ] An English language translation of the International Application as filed (35 U.S.C. 371(c)(2)).
   a. [ ] is attached hereto.
   b. [ ] has been previously submitted under 35 U.S.C. 154(d)(4).
7. [ ] Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371(c)(3))
   a. [ ] are attached hereto (required only if not communicated by the International Bureau).
   b. [ ] have been communicated by the International Bureau.
   c. [ ] have not been made; however, the time limit for making such amendments has NOT expired.
   d. [ ] have not been made and will not be made.
8. [ ] An English language translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
9. [x] An oath or declaration of the inventor(s) (35 U.S.C. 371(c)(4)).
10. [ ] An English language translation of the annexes of the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).

Items 11 to 20 below concern document(s) or information included:

11. [x] An Information Disclosure Statement under 37 CFR 1.97 and 1.98.
12. [x] An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.
13. [x] A preliminary amendment.
14. [ ] An Application Data Sheet under 37 CFR 1.76.
15. [ ] A substitute specification.
16. [ ] A power of attorney and/or change of address letter.
17. [ ] A computer-readable form of the sequence listing in accordance with PCT Rule 13*ter*.2 and 37 CFR 1.821 – 1.825.
18. [ ] A second copy of the published International Application under 35 U.S.C. 154(d)(4).
19. [ ] A second copy of the English language translation of the international application under 35 U.S.C. 154(d)(4).

Amazon / Zentian Limited
Exhibit 1002
Page 2

IAP8 Rec'd PCT/PTO 14 MAR 2007

PTO-1390 (Rev. 07-2005)
Approved for use through 03/31/2007. OMB 0651-0021
U. S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

U.S. APPLICATION NO. (if known, see 37 CFR 1.5)	INTERNATIONAL APPLICATION NO.	ATTORNEY'S DOCKET NUMBER
11/662704	PCT/GB2005/003554	M0025.0369/P369

20. [x] Other items or information: WO 2006/030214, Notification Concerning Submission of Priority Document, Intl. Search Report

The following fees have been submitted	CALCULATIONS	PTO USEONLY
21. [x] Basic national fee (37 CFR 1.492(a)) $300	$ 300.00	
22. [x] Examination fee (37 CFR 1.492(c)) If the written opinion prepared by ISA/US or the international preliminary examination report prepared by IPEA/US indicates all claims satisfy provisions of PCT Article 33(1)-(4) ..... $0 All other situations $200	$ 200.00	
23. [x] Search fee (37 CFR 1.492(b)) If the written opinion of the ISA/US or the international preliminary examination report prepared by IPEA/US indicates all claims satisfy provisions of PCT Article 33(1)-(4) ..... $0 Search fee (37 CFR 1.445(a)(2)) has been paid on the international application to the USPTO as an International Searching Authority ..... $100 International Search Report prepared by an ISA other than the US and provided to the Office or previously communicated to the US by the IB ..... $400 All other situations $500	$ 400.00	
TOTAL OF 21, 22 and 23 =	$ 900.00	
[x] Additional fee for specification and drawings filed in paper over 100 sheets (excluding sequence listing in compliance with 37 CFR 1.821(c) or (e) or computer program listing in an electronic medium) (37 CFR 1.492(j)). The fee is **$250** for each additional 50 sheets of paper or fraction thereof.		
Total Sheets: 104 - 100 =	Extra Sheets: 4 /50 =	Number of each additional 50 or fraction thereof (round up to a whole number): 1	RATE: x $250.00	$ 250.00	
Surcharge of **$130** for furnishing any of the search fee, examination fee, or the oath or declaration after the date of commencement of the national stage (37 CFR 1.492(h)).		$	

CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE		
Total claims	39 - 20 =	19	x 50.00	950.00	
Independent claims	6 - 3 =	3	x 200.00	600.00	
MULTIPLE DEPENDENT CLAIM(S) (if applicable)			+		
			TOTAL OF ABOVE CALCULATIONS =	$ 2,700.00	
[x] Applicant claims small entity status. See 37 CFR 1.27. Fees above are reduced by ½.				1,350.00	
			SUBTOTAL =	$ 1,350.00	
Processing fee of **$130.00** for furnishing the English translation later than 30 months from the earliest claimed priority date (37 CFR 1.492(i)).				$	
			TOTAL NATIONAL FEE =	$ 1,350.00	
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31). **$40.00** per property +				$ 40.00	
				$	
			TOTAL FEES ENCLOSED =	$ 1,390.00	
			Amount to be refunded:	$	
			Amount to be charged	$	

DSMDB-2229206v01

Amazon / Zentian Limited
Exhibit 1002
Page 3

PTO-1390 (Rev. 07-2005)
Approved for use through 03/31/2007. OMB 0651-0021
U. S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

a. [ ] A check in the amount of $ _____ to cover the above fees is enclosed.

b. [ ] Please charge my Deposit Account No. 04-1073 in the amount of $ _____ to cover the above fees. A duplicate copy of this sheet is enclosed.

c. [x] The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 04-1073 . A duplicate copy of this sheet is enclosed.

d. [x] Fees are to be charged to a credit card. **WARNING:** Information on this form may become public. **Credit card information should not be included on this form.** Provide credit card information and authorization on PTO-2038

NOTE: Where an appropriate time limit under 37 CFR 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed an granted to restore the International Application to pending status.

SEND ALL CORRESPONDENCE TO:

Stephen A. Soffen
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
(202) 420-4879

CUSTOMER NUMBER: 24998

Stephen A. Soffen
SIGNATURE

Stephen A. Soffen
NAME

31,063
REGISTRATION NUMBER

DSMDB-2229206v01

11/662704
IAP8 Rec'd PCT/PTO 14 MAR 2007

A Speech Recognition Circuit and Method

The present invention relates to speech recognition circuits and methods. These circuits and methods have wide applicability, particularly for devices such as mobile electronic devices.

There is growing consumer demand for embedded speech recognition in mobile electronic devices, such as mobile phones, dictation machines, PDAs (personal digital assistants), mobile games consoles, etc. For example, email and text message dictation, note taking, form filling, and command and control applications are all potential applications of embedded speech recognition.

However, when a medium to large vocabulary is required, effective speech recognition for mobile electronic devices has many difficulties not associated with speech recognition systems in hardware systems such as personal computers or workstations. Firstly, the available power in mobile systems is often supplied by battery, and may be severely limited. Secondly, mobile electronic devices are frequently designed to be as small as practically possible. Thus, the memory and resources of such mobile embedded systems tends to be very limited, due to power and space restrictions. The cost of providing extra memory and resources in a mobile electronic device is typically much higher than that for a less portable device without this space restriction. Thirdly, the mobile hardware may be typically used in a noisier environment than that of a fixed computer, e.g. on public transport, near a busy road, etc. Thus, a more complex speech model and more intensive computation may be required to obtain adequate speech recognition results.

These restrictions have made it difficult to implement effective speech recognition in mobile devices, other than with very limited vocabularies.

Some prior art schemes have been proposed to increase the efficiency of speech recognition systems, in an attempt to make them more suitable for use in mobile technology.

In an article entitled "A low-power accelerator for the SPHINX 3 speech recognition system", in University of Utah, International conference on Compilers, Architectures and Synthesis for Embedded Systems, Nov 2003, Davis et al have proposed the idea of using a special purpose co-processor for up-front calculation of the computationally expensive Gaussian output probabilities of audio frames corresponding to particular states in the acoustic model.

In an article entitled "Hardware Speech Recognition in Low Cost, Low Power Devices", University of California, Berkeley, CS252 Class Project, Spring 2003, Sukun Kim et al describe using special purpose processing elements for each of the nodes in the network to be searched. This effectively implies having a single processing element for each phone in the network. An alternative suggested by Sukun Kim et al is to provide a processor for each state in the network.

In an article entitled "Dynamic Programming Search for Continuous Speech Recognition" in IEEE Signal Processing Magazine, Sept 1999, Ney et al discuss language model lookahead. Language model lookahead involves computation of a language model factor for each node (i.e. phone) in the lexical tree. This technique is also known as smearing. Each phone instance in the search network can be given a language model factor when it is used in the lexical tree search. Ney et al show that for an example bigram language model, the average number of states per 10 ms frame can be reduced from around 168,000 states with no language model lookahead to around 8,000 states when language model lookahead is used. They also show that bigram language model lookahead requires about a quarter of the states compared with unigram language model lookahead.

Although these prior art documents provide improvements to speech recognition in embedded mobile technology, further improvement is still needed to provide a larger vocabulary and better accuracy.

One aspect of the present invention provides a speech recognition circuit including a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to

said state identifiers. The lexical tree includes a model of words. The speech recognition circuit also has a memory structure for receiving and storing state identifiers identified by a node identifier identifying nodes or groups of adjacent nodes, the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores. An accumulator is provided for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores. A selector circuit is used for selecting at least one node or group of nodes of the lexical tree according to said scores.

One suitable type of hardware for the memory structure includes a content addressable memory (CAM). A CAM is a memory unit which stores a series of data items using a series of addresses. However, the memory is accessed by specifying a data item, such that the CAM returns the corresponding address. This contrasts with a random access memory (RAM) in which the memory is accessed by specifying an address, such that the RAM returns the corresponding data item.

However, the memory structure is not limited to including a CAM. Other types of hardware are also possible, to provide this functionality. For example, a single chip which operates in the same way as a CAM and RAM may be used instead.

Embodiments of the present invention provide a solution to the problem of how to map a lexical tree search to a CAM system architecture. The realisation by the present inventors that certain speech recognition data structures can be mapped into the CAMs allows a lexical tree search to be performed using a CAM system architecture.

Further embodiments of the invention include a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

The node identifier may comprise a direct reference to the lexical tree. However, in some embodiments, the node identifier for at least some of the states includes a pointer to a node identifier for another state. For example, a state corresponding to the furthest part of the search path in the lexical tree may be referenced by a node identifier which directly links to a particular node or group of nodes in the lexical tree. In a lexical tree comprising phones, using a state model of triphones, the node identifier may indicate the position of a triphone in the lexical tree.

However, in this example, for states occuring further back in the search path, instead of supplying a node identifier linking directly to the lexical tree, instead a pointer to a node identifier of another state may be supplied. E.g. a triphone instance may have a pointer to another triphone instance, which has a pointer to another triphone instance, which has a pointer to a node or group of nodes in the lexical tree. Chains of reference may be set up in this way, where only the last state in the chain has a direct pointer to the lexical tree.

There may not be a one-to-one correspondence between the nodes of the lexical tree and the node identifiers. This will occur for a branched lexical tree, where the nodes represent monophones, but the acoustic model states represent triphones, i.e. groups of three adjacent monophones. Then, paths of three monophones will have unique identifiers to be stored in the memory structure, rather than single monophones having unique identifiers.

Phone instance numbers may be generated, and used to uniquely label each phone instance. They can be generated sequentially, using a counter. The phone instance numbers may be used as pointers between phone instances to assist in node identification. It is thus not essential to provide a direct node identifier for each phone instance to directly indicate a location in the lexical tree. The dynamic network of phone instances provided in the memory structure may thus include both direct and relative references to the lexical tree.

The memory structure may be divided into one part which stores phone instance identifiers and direct references to the lexical tree, and a second part which stores phone

instance identifiers and corresponding states. This can speed up the processing, by only storing the phone instances which are furthest on in the lexical tree in the first part of the memory structure.

The memory structure may also be divided into separately accessable units, to reduce the amount to data in each unit, thereby decreasing the chance of finding the same two states identifiers in different phone instances in any single memory unit, and increasing the chance of some state identifiers being completely absent from any single memory unit. This makes it easier to deal with the situation when the same two state identifiers are found, because a spare time slot is available for processing when a state identifier is not present.

A further aspect of the invention provides a distance calculation engine within a speech recognition system. The distance calculation engine may be included within an accelerator. The accelerator may include logic to interface with other parts of a speech recognition circuit, in addition to the distance engine, although this is not essential. For example, the accelerator may include one or more results memories for storing distances calculated by the distance calculation engine. The accelerator may also include at least one of a memory for storing one or more acoustic models, a decompressor for decompressing acoustic data that has been stored in a compressed format, a memory for storing feature vectors, a checksum or data signature calculation means, buffers for data storage, and data registers. The accelerator may be implemented in software or in hardware, or in a combination. It may be physically separate to the rest of the speech recognition circuit, although this is not essential.

The distance calculation engine may calculate one or more of a wide range of distance metrics and probability distributions. The distances may represent the likely correspondance of feature vectors to states in an acoustic model. In other words, the distances can indicate the similarity of an audio data frame to each possible state in an acoustic model

There are a wide variety of probability distributions that can be used for the distance calculation stage of a speech recogniser, and a wide variety of distance metrics used.

These are widely documented in the literature. A point is a simple example of a probability distribution.

A common choice is to use Gaussian Distributions and correspondingly the Mahalanobis Distance metric. The Gaussian probability distribution is then defined by a mean vector, which defines centre point in the N-dimensional space, and a Covariance matrix which defines the shape of the probability distribution. It is common to restrict the Covariance matrix to be a diagonal matrix (only N non-zero values along the diagonal of the NxN matrix) which significantly lowers the implementation cost by reducing the number of arithmetic operations.

In particular embodiments, the distance calculated is a Mahalanobis distance. Particular examples of this are described later in the specification.

In one embodiment, the distance engine autonomously computes all of the distances associated with a given feature vector. This may comprise computing distances for every state in the lexicon. The distance engine may operate in a pipelined manner with other stages of the recognition process. In this context a distance is an indication of the probability or likelihood that a feature vector corresponds to a particular state. An important class of distance computation in speech recognition is the calculation of output state probabilities in recognisers using Hidden Markov Models. Another possible use is in recognisers using Neural Networks.

The distance engine reads data from the acoustic models to use as parameters in the calculation. The acoustic models may be optionally stored in a compressed format. The distance engine may read and de-compress the acoustic models one (or more) times for each feature vector processed. Each reading of the acoustic models may require reading the entire acoustic model, or various optimisations may be implemented to avoid reading parts of the acoustic model that are not required to be used for calculations with the current feature vector. The distance engine may use a de-compression method where the de-compression is sign or zero extension or may otherwise convert data of narrow or variable width to a wider data format. The distance engine may use a de-compression method where the de-compression is sign or zero extension or may otherwise convert

data of narrow or variable width to IEEE standard single or double precision floating point format. The distance engine may use a decompression method where decompression is a codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data. The distance engine may use a decompression method where the decompression is decompression of a Huffman or Lempel-Ziv compressed stream. The distance engine may use a decompression method where decompression is decompression of run length encoded data. The distance engine may use a decompression method where decompression is decompression of difference encoded data. The distance engine may use a decompression method using any well known lossy or lossless compression scheme. The distance engine may use a decompression method using subspace distribution clustering. The distance engine may use a decompression method comprising any combination of the above described decompression types.

The distance engine may read the acoustic models from a dedicated on-chip memory. The distance engine may read the acoustic models from a dedicated off-chip memory. The distance engine may read the acoustic models from a shared on-chip memory. The distance engine may read the acoustic models from a shared off-chip memory. Any of these acoustic models may be compressed.

The distance engine may compute a CRC or checksum or similar signature as it reads in the acoustic model and compares this to a stored CRC, checksum, or signature, in order to check that the acoustic model has not been corrupted, and signals an error condition if such corruption is detected. The stored CRC, Checksum, or signature may had been pre-computed and stored in the model data, or it may be computed at the time the model data is loaded into the Acoustic Model Memory. It may be held in Acoustic Model Memory, or it may be loaded into a register or another memory from where it can be accessed and compared when the CRC/checksum/signature is computed each time the Acoustic Model is loaded.

The distance engine may support the pass-through of data from the front-end to the search-stage. The data to be passed through will be supplied to the distance engine as an adjunct to the feature vector, and the distance engine will pass it to the search stage as

an adjunct to the distance results. This provides a simple mechanism for passing frame-specific data that is not involved in the distance calculation through to the search stage, and keeping it associated with the correct frame, which may otherwise be complex in pipelined systems with multiple processors. The data passed through may be for any purpose. Examples might include silence detected, end-of-audio-stream detected, a frame number, information that some intervening frames have been dropped, or data from another input device such as a button or keyboard in a multi-modal interface.

The distance engine may be implemented in hardware, software, or a combination. Other stages may be implemented in hardware, software, or a combination. The distance engine may be implemented with any number representation format including fixed point or floating point arithmetic, or any mixture of number representation formats.

In particular, the other stages may be implemented on a CPU, or on a DSP and CPU. The "DSP" and "CPU" may each be implemented as software programmable devices.

The distance engine may implement one or more additional pipeline stages to overcome delays introduced by low bandwidth, high latency, or conflicted bus interfaces. The distance engine may also implement additional pipeline stages to maintain the same throughput while allowing more time for each distance calculation. Particular embodiments of the invention may include one or more of the above aspects.

A further aspect of the invention comprises a speech recognition circuit, comprising: an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame; calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words; a buffer memory between the calculating circuit and the search stage, for receiving data passing from the calculating circuit to the search stage, wherein a processor in the search stage has higher bandwidth and/or lower latency access to the buffer compared to the

bandwidth and/or latency of direct transfer between the calculating circuit and the search stage. The data transfer from the calculating circuit to the buffer memory and/or from the buffer memory to the search stage may be performed as one or more sequential bursts. The data transfer to the buffer memory may be performed in parallel with data transfer to the calculating circuit and/or in parallel with data transfer to the search stage. A second buffer memory may be provided between the audio front end and the calculating circuit.

A further aspect of the invention comprises a speech recognition circuit, comprising: an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame; calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words; comprising an elastic buffer between at least one of the front end and calculating circuit, or the calculating circuit and search stage, and/or for buffering said audio signal.

A further aspect of the invention comprises an accelerator for a speech recognition circuit, the accelerator comprising: calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; means for comparing a first version of a stored checksum of data representing said acoustic model and a second version of said stored checksum, wherein the second version is obtained from an updated measurement and calculation of the checksum; and means for indicating an error status if the checksums do not match.

A further aspect of the invention comprises an accelerator for a speech recognition circuit, the accelerator comprising: calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; wherein said

accelerator is configured to autonomously compute distances for every acoustic state defined by the acoustic model.

A further aspect of the invention comprises an accelerator for calculating distances for a speech recognition circuit, the accelerator comprising: calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; first and second storage circuit, which may be referred to as result memories, each for storing calculated distances for at least one said audio time frame, and for making said stored distances available for use by another part of the speech recognition circuit; control circuit for controlling read and write access to the first and second storage circuit, said control means being configured to allow writing to one said storage means while the other said storage means is available for reading, to allow first calculated distances for one audio time frame to be written to one said storage means while second calculated distances for an earlier audio time frame are made available for reading from the other said storage means.

Embodiments of the invention may comprise means for generating a checksum or computed signature for the acoustic model data stored in the memory, and means for comparing checksums or computed signatures that have been calculated at different times, to indicate an error status if the checksums do not match, one possible cause of such mismatch being that the acoustic model data has been overwritten by said other data and said error status being used to indicate that the acoustic model should be re-loaded into the said memory.

A further aspect of the invention comprises a speech recognition circuit comprising: lexical memory containing lexical data for word recognition, said lexical data comprising a lexical tree data structure comprising a model of words; means for accessing a state model corresponding to each phone or each group of phones in the lexical tree; a content addressable memory for storing content addressable data for each phone or group of phones, including states corresponding to said phone or group of phones, and for storing an address value for each said phone or group of phones; a RAM configured to store accumulated scores for each said phone or group of phones,

the accumulated scores being addressable by said address value for each said phone or group of phones; means to obtain scores that each of a plurality of frames of an audio signals corresponds to each of a plurality of said states; a counter to sequentially search for each said state in the content addressable memory, to obtain the corresponding address value if the state is found in the content addressable memory; means to use said address value to access an accumulated likelihood and an accumulator to add said likelihood to the accumulated likelihood; means to use the phones or groups of phones with the highest accumulated scores to obtain a plurality of next phones from the lexical tree which correspond to the next phone; and output means for outputting a lexical tree path of highest likelihood.

A further aspect of the invention comprises speech recognition apparatus comprising: a lexical tree having a corresponding state model; means for obtaining scores of an audio input corresponding to each of a plurality of states in said state model; a content addressable memory for storing a marker indicating a part of the lexical tree, and one or more states associated with said part of the lexical tree; a random access memory addressable by the CAM output, to output accumulated scores for states corresponding to said parts of the lexical tree; adder means for adding likelihood to said accumulated likelihood, to be stored back in the RAM.

A further aspect of the invention comprises speech recognition apparatus, comprising: a CAM-RAM arrangement for storing records including pointers to a lexical tree, and accumulative scores for states within the lexical tree; input means for obtaining scores that an audio frame corresponds to a particular state in the lexical tree; an accumulator for calculating the updated scores and modifying the records in the CAM-RAM accordingly; output means for outputting a path of highest likelihood in the lexical tree.

A further aspect of the invention comprises a speech recognition method comprising: storing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and scores corresponding to said state identifiers in a memory structure, the lexical tree comprising a model of words and the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the

memory structure after modification of the scores; repeating the following sequence of steps for each of a plurality of incoming frames of an audio signal; obtaining score updates corresponding to the likelihoods that said frame of the audio signal corresponds to each of a plurality of said states; accessing said memory structure to obtain scores, updating the scores by adding score updates to the scores, and writing back the updated scores to the memory structure; determining if scores for states furthest on in the lexical tree correspond to a significant likelihood, and if so, then accessing the lexical tree to determine the next set of possible states; and storing the next set of possible states and said scores of significant likelihood in the memory structure.

A further aspect of the invention comprises speech recognition circuit comprising: a circuit for providing state identifiers which identify states corresponding to phones or groups of adjacent phones in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers and phone instance identifiers uniquely identifying instances of phones or groups of phones in the lexical tree; said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one phone instance identifier according to said scores.

A speech recognition apparatus according to the invention may be embedded in or included in a mobile electronic device such as a mobile telephone, PDA (personal digital assistant), etc.

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of the system architecture for a speech recognition apparatus according to an embodiment of the invention;

Figure 2 is a block diagram showing the main data structures used in the speech recognition apparatus of figure 1;

Figure 3 shows a block diagram of the Single Phone Instance Network Engine (PINE) of figure 1;

Figure 4 is a block diagram showing the Wave Front Manager architecture;

Figure 5 shows the use of link nodes to permit path merging;

Figure 6 shows a block diagram of a Path Merge Unit in the Phone Level;

Figure 7 shows the Phone Book architecture;

Figure 8 shows Lexical tree storage;

Figure 9 shows the data structure for the word link record;

Figure 10 shows a block diagram of Word Engine;

Figure 11 shows a Language Model interface architecture;

Figure 12 shows an example of a simple Loop Grammar;

Figure 13 shows a block diagram of grammar engine;

Figure 14 shows a single frame of the Audio Codec link protocol; and

Figure 15 illustrates the system timing which may be used in the embodiment of figure 1;

Figure 16 is a block diagram showing a speech recognition circuit according to an embodiment of the invention, and illustrating data flow between parts of the circuit;

Figure 17 is a block diagram showing a distance calculation engine according to an embodiment of the invention;

Figure 18 shows ideal data flow over several time steps within a speech recognition circuit having separate front end and search stages, in an embodiment of the invention;

Figure 19 shows an example of actual data flow over several time steps within a speech recognition circuit having separate front end and search stages, in a further embodiment of the invention;

Figure 20 shows ideal data flow over several time steps within a speech recognition circuit having a shared processor for front end and search stages, in an embodiment of the invention;

Figure 21 shows an example of actual data flow over several time steps within a speech recognition circuit having a shared processor for front end and search stages, in an embodiment of the invention;

Figure 22 shows ideal data flow in a speech accelerator pipeline with two data transfer stages, according to an embodiment of the invention; and

Figure 23 shows an example of actual data flow in a speech accelerator pipeline with two data transfer stages according to an embodiment of the invention.

Figure 1 shows a speech recognition system architecture for an embodiment of the invention. The speech recognition system is divided into several levels of operation: a state level, a phone level, a word level, a grammar level, and a system level. Each level is shown in figure 1 as a grey shaded box, interconnected to other levels of the system.

The information passed between different levels or subcircuits of figure 1 is indicated on the figure 1 using the following identifiers.

Identifier	Definition
grammar_id	Identifies which grammar model is currently being used
grammar_node_id	Identifies a node in the grammar model
word_id	Uniquely identifies a word
wordlat_node_id	Identifies a node in the word lattice
LT_id	Lexical tree identifier. Identifies the lexical tree to be used
LT_branch_id	Identifies a unique branch in a lexical tree. A branch is defined as all of the monophones subsequent to and including the first monophone in a tree.
LT_node_id	Identifies a node in a lexical tree
monophone_id	Identifies a monophone
transvec_id	Identifies which transition vector to use
state_id	Identifies the state
phone_no	Unique phone instance identifier
prev_phone_no	Phone number of previous phone in network
triphone_id	Triphone identifier
LML_factor	Language model lookahead factor. The smeared language model score that must be added

The system level includes a general purpose processor and cache, a system controller, a DMA (direct memory access) controller, clocks, PLL (phased locked loop), GPIO (general purpose I/O interface), timer watch dog, memory interfaces to interface to main memory and non volatile storage, serial, host, debug and display interfaces. The system level also includes a model adaptation engine, a language model factor engine, a speech recognition system controller, a language model interface, and a recognition output interface.

The processor organises the transfer of the various models from the non volatile storage to the various model memories. It handles the language model factor requests. Language model factors are used to by the state level to apply the language model score. The language model (LM) interface provides the mechanism for obtaining LM factors as and when they are needed. The processor keeps a copy of the wave front in memory which is updated simultaneously with that in the wave front manager. Using its copy of the lexical tree, the processor can determine which language model scores may be needed in the coming frames. Once calculated, the language model factors can be kept in memory for use in subsequent frames. The system level is also used for forming speech model adaptation, and acting as an overflow for the various buffers used by the architecture.

The system level includes the audio codec interface and the audio front end interface, connected to the processor via a APB (advanced peripheral bus) to AHB (advanced high performance bus) bridge.

The grammar level includes a grammar engine and grammar model(s). The grammar level sets out the rules of the particular recognition task at the highest level. During system initialisation a number of grammar models might be loaded, for example a command grammar and a dictation grammar. Command mode grammar is a restricted grammar which includes only a very limited lexical tree with command specific words. Some branches of this lexical tree may include multiple words together, where a command has such a structure, e.g. "open file menu". Typically, only a single instance of the lexical tree will be needed. Command mode is used because it can obtain a higher accuracy, due to the lexical tree constraint. In dictation mode grammar, a full lexical tree structure is used.

The grammar may be selected by a user during the recognition process. For example, the command grammar may be selected when the user presses a command button. The architecture supports fast swapping of grammar models. For example, a user may be in the middle of dictating a message when a "punctuation button" is pressed and the phrase "question mark" uttered. At this point, a punctuation grammar could be loaded, in order to increase recognition accuracy.

The word level comprises a word engine, which includes a word lattice and a word graph generator. The word level has two main functions, of keeping a record of the search path at the word level using a word lattice, and interfacing to the language model via a cache. The word lattice is a record of all of the words that have been searched up to the current frame. As new words are searched, they are appended into the lattice at the appropriate place. After the final frame in the current sentence, the n-best word lattice node identifiers from the n-best scoring tokens are passed back to the word lattice. This enables the n-best sentences to be traced back through the lattice. The word lattice can also be post processed to produce a word graph which is essentially a compact version of the word lattice.

At the word level, a record is kept of all tokens that have passed through a word end. This results in a lattice of words. When the final feature vector has been processed, the best scoring token is identified and used to trace back the search path back to the start of the search.

The phone level includes a "phone book" for generating new triphone instances, a wave front manager for sending requests to the phone book to generate new phone instances, a phone request arbiter, and a path merge unit.

The phone book includes a lexical tree, which comprises a vocabulary of words, sounds and/or phrases organised according to the phones or monophones making up the sound. A lexical tree is commonly used in speech recognition rather than a linear lexicon, because it greatly reduces the amount of storage space required to store a word vocabulary, by avoiding duplication of common word beginnings. It is also easier to search when the word beginning is known.

Although it is possible to search a lexical tree based on monophones, it is well known that better results can be achieved in speech recognition by basing the lexical tree search on groups of more than one monophone, since the sound of one syllable commonly affects the pronunciation of the next syllable in the same word, or the first syllable in the next word. In this embodiment, groups of three monophones are used, and these are known as triphones. It is alternatively possible to use a different number of monophones for each group. The phone generator in the "phone book" uses data from the lexical tree to generate triphones.

In one embodiment of the invention, approximately 45 different cued mono-phones are considered. This gives of order 60,000 tri-phones. Each triphone in this embodiment is modelled by a Hidden Markov Model (HMM) with three states. There is no correspondence between individual states and individual phones of the triphone. Alternatively, it is possible to model each triphone by a single state, or by any other number of states.

The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree. The search space can be envisaged as a wave moving through an acousto-linguistic search space. In order that the wave can be dynamically created, a record of the wave front phone instances must be maintained. This is done in a wave front phone CAM/RAM structure, within the wave front manager. When a token occurs in the final state of a phone instance in the PINE array at the front of the wave, a new phone instance must be generated at the phone level and passed to the PINE array.

The phone instance includes a phone_no, which is a phone instance identifier to identify the phone uniquely in the current search. The phone instance also includes a word lattice node identifier (labelled "word_lat_node_id") to identify the phone's position in the word lattice, state identifiers (labelled "state_ids") for the HMM, storage to hold the tokens as they move through the HMM and LM_factor which is used to by the PINE to apply the language model score. To handle the connectivity between phone instances, a phone instance points to a previous phone instance using the phone_no of the previous phone. This allows tokens to be passed between phones. A "token" is a hypothesised search path, represented by a non-zero score at a particular state, and tokens are propagated through the HMM with each time frame.

The phone instance network can be visualised as a wave with new phone instances being added to the front of the wave, and empty phone instances being removed from the back of the wave. In some circumstances, the back of the wave may be in the same position as the wave front. The back of the wave moves on through the lexical tree as tokens are either moved on or pruned.

Tokens, each representing a particular hypothesis, hop through the states in the HMMs, each hop consuming a feature vector. Each token explores all possible next states which typically involves placing a token in the next state and keeping one in the current state. When there is more than one token in a state, the least likely token can be removed because both have the same word history. When a token reaches the last state of a phone instance at the from of the wave, the next phone instance(s) are requested from the phone level. The phone level uses the lexical tree to create the new phones. As well as

Amazon / Zentian Limited
Exhibit 1002
Page 22

looking up the state identification numbers for each HMM state, it also incorporates the correct language model factor.

The language model factor engine is used to smear the language model scores across the lexical tree. This operates in two ways depending on whether the factor is needed for a word start phone(s) or word internal phones. For word internal phones, LM factors can be computed as needed, provided that the LM scores are available in cache. This is because in general the storage requirements for the LM scores are less than the storage of the LM factors for all of the factors in the branch. However when a token is in a word end phone there may a significant number of word start phones e.g. up to $(\text{monophones})^2$ requiring a large amount of computation. To avoid this the LM engine will pre compute word start LM factors when it seems highly likely that a token will be passed to the word starts. These word start LM factors will be held in the LM factors cache. The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree.

The state level shown in figure 1 includes a phone instance network engine (PINE) array, a MHD (Mahalanobis distance) engine, an audio front end, a feature vector buffer for model adaptation, and compressed acoustic models. The state level is the lowest level, in which a network of phones is dynamically created.

The audio input for speech recognition is input to the front end unit of the state level. This audio input may be digital audio, or it may be analog audio which is converted to digital audio using an analog to digital converter. The audio input is divided into time frames, each time frame typically being of the order of 10ms.

For each audio input time frame, the audio signal is converted into a feature vector. This may be done by splitting the audio signal into spectral components. For example, in the present embodiment, the audio input signal is split into 12 frequency bands and one energy band, and a feature vector is constructed using these 13 components, plus their first and second derivatives, a total of 39 components.

The feature vector is passed to the MHD (Mahalanobis distance) engine. This computes all possible state output likelihoods for the current frame as the state mean and covariance data is read from the acoustic model memory.

The likelihood of each state is determined by the distance between the feature vector and each state. To reduce the number of states, a technique known as "state tying" may be used. Multiple states are grouped together in 39D space, according to their proximity to one another. An average or central value from each group is used to represent the entire group. Thus, the effective number of states may be reduced from 40,000 to 8,000. The accuracy may be improved using a language model in addition to the word recognition lexical tree. A language model is based on the probability of sequential words. The above figure of 8,000 states gives best recognition results when used with a language model.

The MHD (Mahalanobis Distance) is a distance between two N-dimensional points, scaled by the statistical variation in each component. In embodiments of the invention, the Mahalanobis distance between the feature vector and each state is calculated, to determine similarity of the feature vector to each state.

The Feature Vector represents a point in an N-dimensional space. Different embodiments of the invention may use different values of N. Generally we see values of N in the range of 20 to 39, although values outside this range are not excluded.

The distance calculator stage of the recognition process computes a probability or likelihood that a feature vector corresponds to a particular state. Each state has an associated probability distribution in the N-dimensional space. The distance calculator computes the distance in the N-dimensional space from the Feature Vector to the probability distribution for each state. In some embodiments, the calculation is performed only when the search stage algorithms determine that the distance for that particular state is needed by the search stage. In some embodiments, the distances for all of the states are calculated and made available to the search stage, which allows pipelined operation. Embodiments that do the calculations on demand are preferable.

The Mahalanobis Distance (MHD) is extensively described in the literature. It may be calculated using the formula

MHD value = ln(P(x)) = -0.5[N*ln(2*pi)+ln|Cov|)] + (-0.5*[(x-u)'*inverse(Cov)*(x-u)])

where

N= number of dimensions in the space
x=feature vector, a point in the N dimensional space
u=mean vector
Cov=covariance matrix
|Cov|=determinant of covariance matrix
(a)'= transpose of matrix a
Different embodiments may use variations on this equation, for example the base for the logarithm is not always the natural base e that is used in the above example. The log of the probability is preferably calculated, rather than the probability, as it simplifies the implementation. Not all embodiments do this. In alternative embodiments, the likelihoods rather than probabilities are computed as it leads to simpler implementations of computations in the search stage.

The feature vector is used to calculate 8,000 MHD distances for each time frame, i.e. one distance for each of the 8,000 states. It is beneficial, although not essential, to perform the MHD calculations separately in a pre-processor, known as a MHD engine, because MHD calculation is a computationally intensive process.

Due to the 10ms frame length, a feature vector arrives at the MHD engine of the state level every 10ms. The MHD engine calculates the likelihood of each feature vector corresponding to each of a number of states derived from an acoustic model. Each state is also a 39 dimensional vector, having the same spectral components as the feature vector. The MHD engine calculates the difference in 39-D space between the feature vector and each state to obtain the likelihood for that state. A parallel MHD or a single MHD arrangement may be used.

The feature vectors may be sent to a feature vector buffer for model adaptation, which determines whether the observed features fit well with the acoustic model, and may initiate an adaptation of the model if the fit is not sufficiently good.

Scores indicating the state likelihoods or probabilities are output from the MHD engine to the PINE array. It is preferable to use likelihoods rather than probabilities, because likelihoods can be added together to give a total likelihood. Probabilities must be multiplied, which can give rise to rounding errors. In this embodiment, eight separate pines are used, each of 4 kilobytes, thus storing 4000 phones per pine.

During recognition, the Phone Instance Network Engine (PINE) Array will contain a network of phone instances. Tokens representing search hypotheses in the form of a probability(likelihood) will exist in some of the HMM states of the phone instances. The tokens are propagated into all available states and updated with the corresponding state output probabilities. Maximum model, word end and beam pruning are applied to the tokens to ensure the search space is kept bounded. When a token is in the final state of a phone model and has avoided being pruned, the PINE makes a next phone request to the Phone level. The PINE array may receive a number of new phone models from the phone level which must be appended to the appropriate existing phone.

Figure 2 shows the data structures stored in the PINE array, and other main data structures used during the acoustic search. Each PINE includes a phone instance CAM and a phone instance RAM. The phone instance CAM stores the phone instance number (labelled "phone_no") which uniquely identifies each phone instance. The phone instance CAM also stores the state IDs of the three HMM states used to model the phone, and flags indicating the position of the phone. The "back" flag indicates if the phone is at the back of the wavefront. The "front" flag indicates if the phone is at the front of the wavefront. The "empty" flag indicates if there are no tokens in any of the three states for that phone. The "end" flag indicates if the phone is at the end of a word.

The phone instance RAM stores the token scores for each of the three states corresponding to each phone. The st_dum field indicates a dummy state, which is used

to represent a state prior to the first state of the first phone. The "tmp" field is used for passing a token from a previous phone in the lexical tree to the current phone. This will be explained in more detail later. The prev_phone_no field is for storing a pointer to the previous phone instance in the network. The wordlat_node_id identifies the position of the phone instance on the word lattice node, and the LM_factor stores the language model factor for that phone instance.

At the phone level a record of all phones on the wave front must be maintained. If a phone instance is on the wave front, the phone_no stored in the phone instance CAM will point to the details in the wave front phone CAM and RAM.

The wave front phone CAM and RAM keeps pointers into the grammar and lexical tree. When a token in a wave front phone reaches the last state, the PINE array makes a request to the phone level by passing back the phone_id. The phone_id is used as a key to determining the token's grammer_node_id and LT_node_id within the wave front CAM/RAM structure. If the token is at a word end the grammar model is checked. A grammar identifier points to the type of grammar.

At the word level, figure 2 shows an example word lattice structure, with silence having a word_id of 0, and a wordlat_node_id of 6. This is shown as connected to two word nodes – "which" has a word_id of 238 and a wordlat_node_id of 4, and "neither" has a word_id of 3556 and a wordlat_node_id of 5.

At the highest level, i.e. the grammar level, a simple grammar model is shown which will allow the phrase 'silence which house silence' or 'silence neither house silence' to be spoken. Whenever a word end is reached during the search, the grammar model must be consulted to determine the next possible word(s) or lexical tree(s). This allows the word lattice at the word level, consisting of a network of words, to be assembled. Whenever a word end is reached, the word is added to the lattice and the wordlat_node_id updated accordingly.

Figure 3 shows a single phone instance network engine (PINE). At system power up the state transition RAM will be loaded with the appropriate state transition vector but the

CAM and RAM memories will be empty. When the system is initialised an initial set of phone model(s) will be instantiated with a token in the dummy state and a path score of zero. The phone models are received from the phone level.

When the audio data is processed, a stream of all possible state output probabilities will be received from the MHD engine. A counter that is synchronous to the stream of state output probabilities from the MHD produces a concurrent stream of state_ids. The job of the CAM is to perform a search to see if the state_id is instantiated anywhere in the CAM state_id fields. If it exists and the location doesn't have its empty flag set then this indicates that a token may need to be updated. The match signal from the CAM must be encoded into an address for the RAM. The corresponding RAM address can then be read. Any token that will end up in the identified state can then have it's path score updated and including any transition vector likelihood.

It is possible that a single state_id generates multiple CAM matches. To handle this situation the CAM and RAM are effectively decoupled by means of a match fifo and repeat count fifo. The state_id and corresponding state_output_probability are stored in the repeat count fifo. During a subsequent frame when there is no match, this state_id can be revisited, ignoring the matches already dealt with.

All searched phone instances corresponding to words in the current possible lexical tree are stored in the CAM-RAM. This includes phone instances at the beginning of the word, middle of the word and end of the word. These phone instances are connected by the "previous phone instance" field, which allows traceback through an entire word within the CAM-RAM.

A mechanism is thus required to pass tokens from one phone instance to the next. This mechanism is provided by the tmp field in the RAM.

The tmp field in the RAM holds tokens during token passing.

At the start of each time frame, the audio input is converted into a feature vector, the MHD values calculated, and the MHD values used to update the token scores. After

update of all tokens, 2 passes are made through the CAM-RAM, firstly for pruning, and then for token passing. The pruning in this example removes tokens of below a threshold score, although other types of pruning are also possible. The pruning may involve removing a predetermined proportion of the tokens, chosen as those of lowest likelihood. A third possible method of pruning is removing all tokens with scores below a first predetermined value, but at word ends removing all tokens with scores below a second predetermined value, the second predetermined value being greater than the first predetermined value.

The token passing works as follows. Each phone instance points back to a previous phone, due to the prev_phone_no field stored in the RAM. During the token passing, the third token of a previous phone is grabbed and put into the tmp field of the current phone.

In other words, if a token is on the third state, and it is not pruned, it must enter the next state of the next phone. To do this, the tokens that lie in the third state are moved across to the tmp field.

For example, phone no. 345 needs to look at phone no. 320 to see if any tokens are in the third state. If yes, it takes these tokens and puts them in the tmp field.

One field in the CAM is labelled "front". This indicates whether the phone is on the wave front. If the phone is not on the wave front, then another phone will have picked up the token in the third state. However, if the phone is on the wave front, then no other phone will have picked up the token on the third state. Therefore, if there is a token in the third state, and the phone is on the wavefront, it is necessary to generate a next phone request.

The current phone number, word lattice identifier and word end flag are sent to the wave front manager, to request a next phone. If the word end flag indicates a word end, then the score is also passed to the wave front manager. The score can be passed in any case, but it is only required at a word end.

The phone level generates the next phone or phones. A new phone instance starts, and all of state1, state2 and state3 have scores pre-set to zero. The old score is provided in the tmp field.

The old phone is still within the wave, but it is unlikely to be at the back of the wave at this stage. When it gets to the back of the wave, it is deleted. Until it gets there, it remains in the CAM-RAM as before. The third token remains where it was, since tokens can either stay in the third state or move on.

Note that it is possible to end up with more that one token in each state. If this occurs, then the token with the poorer score is deleted. This does not affect the result of the lexical tree search, because both tokens on the same state have exactly the same word history.

Figure 4 shows a block diagram of the wave front manager. The wave front manager is responsible for maintaining a record of the set of phone models that exist at the front of the acoustic search space. This set of phone models is referred to as the "wave front." The key requirement for this block as to minimise the time between receiving a "next phone request" and generating the appropriate responses. The wave front manager receives the next phone requests from the PINE array and generates appropriate responses for the Phone Book, Word Engine, Path Merge Unit and Grammar engine The wave front manager also receives new wave front phone models from the Phone Book and updates the wave front.

The wave front manager receives the next phone request from the PINE array, via the phone request arbiter. The next phone request consists of a phone_no, word_end_flag and wordlat_node_id. If this phone is at the end of a word, the word_end_flag will be set.

The wave front manager uses the phone_no as input to the wave front phone CAM. As each phone_no is unique, it is guaranteed that only a single match will be generated. The wave front manager uses the CAM generated address to access the RAM. It then

sends retrieved data from the RAM to the phone book, grammar engine, Path Merge Unit and word engine.

The wave front manager removes phone instance from CAM and RAM by writing 0's into CAM and RAM, and increments the empty count by one.

Sometime later, the wave front manager receives new phone instances from the phone book and place them in a fifo. The phone instances are taken from the fifo and written into the CAM and RAM at the next_empty_address. The empty_count is decremented by one. A match is run on the "occupy" flag, and the address is placed into next_empty_address register.

Both the phone level and the state level make use of a path merge unit to reduce the number of possible paths, by path merging. Figure 5 shows the use of link nodes to permit path merging. In the present embodiment, all word end phone instances are linked to word start triphones with a link node. During each frame the path merge unit looks for *word end* tokens that can be recombined. This is done by looking for word end phones that have the same word_id and word history. When they are found a link is added using the path merge unit in the PINE array.

Figure 6 shows a diagram of the architecture at the Phone level, showing the path merge unit. The path merge unit includes a path merge CAM and a path merge RAM. The path merge CAM stores word_id, word history for two words, and an "occupy" field. The path merge RAM stores a link_node_id. A CAM RAM occupancy manager keeps track of the empty addresses in the CAM and RAM. The path merge unit is only passed information from the wave front manager if the phone is a *word end phone*. For each frame, there is a check to see if there is an identical word with the same history, and if there is, the link_node_id is passed to the path merge unit. If there isn't an identical word with the same history, a new record is written into the Path merge CAM-RAM.

Figure 7 is a block diagram showing the Phone Book architecture. The Phone Book is tasked with generating the next phone models when given a pointer into the lexical tree. The key requirement for this block as to minimise the time between receiving a "lexical

tree address" and generating the appropriate responses, in particular the next phone model instance(s).

The lexical tree is used to build the next triphones based on the current context.

Current triphone	Next triphone	comments
Left context = a Central context = b Right context = c	discarded Left context = b Central context = c **Right context = ?**	**next_leftmph_id stored in wave front manager current LT_node_id stored in wave front manager must be looked up. Large (< 45) for new word start**

The LT_node_id, next_left_mph_id and prev_phone_no are received at the phone generator from the wavefront manager. The LT_node_id is then used to address the lexical tree store and return the central monophone_id. One or more further accesses to memory are performed using the microcoded instructions, each memory access returning the right monophone_id.

A two stage lookup is accessed using the three monophone_ids (giving a triphone), the associated state_ids. The phone model is assembled, and returned to the PINE array. The wavefront manager is then updated by sending it relevant parts of the phone model.

The lexical tree store is a micro-coded ROM with both instruction and data fields. The lexical tree store holds a number of lexical trees. Each lexical tree may hold single words or multiple words. Each lexical tree comprises lexical tree elements, and each element has three fields. For now, the fields will be integers and will be converted to std_logic_vector later. The following is an example of the lexical tree store, with the LT data in word_id order.

```
ZERO   z iy r ow        19  8 12 11  0  0  0  0        sp  0
ONE    w ah n           18  2 10  0  0  0  0  0        sil 1
TWO    t uw             14 16  0  0  0  0  0  0        ah  2
THREE  th r iy          15 12  8  0  0  0  0  0        ay  3
FOUR   f ow r           6  11 12  0  0  0  0  0        eh  4
FIVE   f ay v           6   3 17  0  0  0  0  0        ey  5
SIX    s ih k s         13  7  9 13  0  0  0  0        f   6
SEVEN  s eh v ih n      13  4 17  7 10  0  0  0        ih  7
EIGHT  ey t             5  14  0  0  0  0  0  0        iy  8
NINE   n ay n           10  3 10  0  0  0  0  0        k   9
OH     ow               11  0  0  0  0  0  0  0        n   10
```

The following shows the LT data in MP (monophone) start order.

```
EIGHT  ey t          5  14  0  0  0  0  0  0
FIVE   f ay v        6   3 17  0  0  0  0  0
FOUR   f ow r        6  11 12  0  0  0  0  0
NINE   n ay n       10   3 10  0  0  0  0  0
OH     ow           11   0  0  0  0  0  0  0
SEVEN  s eh v ih n  13   4 17  7 10  0  0  0
SIX    s ih k s     13   7  9 13  0  0  0  0
TWO    t uw         14  16  0  0  0  0  0  0
THREE  th r iy      15  12  8  0  0  0  0  0
ONE    w ah n       18   2 10  0  0  0  0  0
ZERO   z iy r ow    19   8 12 11  0  0  0  0
```

```
ow 11
r 12
s 13
t 14
th 15
uw 16
v 17
w 18
z 19
```

The following is an example of the three fields of a lexical tree element. The first field is an instruction field, and the second and third fields are data fields.

```
Fields:     1st    2nd    3rd
            instr  data   data
            2b     6b     6b

            00     MP     MP      - raw data field.
            01     br     br      - branch address.
            10     br     br      - branch address.
            11     word_id       - double field.
```

Note that there might be no difference between "01" and "10"

To build the "ey" tree, we only have to consider "EIGHT", because this is the only word in the lexical tree which starts with "ey". EIGHT is represented as "ey t", with phone_nos 5 14 0 0 0 0 0 0. The first lexical tree element stores the raw data specifying the phone_nos, the second lexical tree element is an end field, and the subsequent lexical tree element is set to zero.

```
0  00  5 14   - raw data
1  11  8 0    - end field
2  00  0 0
```

To build the "f" tree, we have:

```
FIVE   f ay v      6  3 17  0  0  0  0  0
FOUR   f ow r      6 11 12  0  0  0  0  0

0  01  4  0  -- branch forward four places as well
1  00  6  3  -- as the implied other brach of next (one).
```

```
2  00  3 17
3  11  5  0  -- all addresses are relative offsets.
4  00  6 11
5  00 11 12
6  11  4  0
7  00  0  0
```

To build the "n" tree, we have:

```
NINE          n ay n          10  3 10  0  0  0  0  0
```

```
0  00 10  3
1  00  3 10
2  11  9  0
3  00  0  0
```

To build the "ow" tree, we have "ow" in both the first and second data fields, as no tree is going to be a single monophone long.

```
OH            ow              11  0  0  0  0  0  0  0
```

```
0  11 11  0
1  00  0  0
```

To build the "s" tree, we have:

```
SEVEN         s eh v ih n     13  4 17  7 10  0  0  0
SIX           s ih k s        13  7  9 13  0  0  0  0
```

```
0  01  6  0  -- branch forward six places.
1  00 13  4
2  00  4 17
3  00 17  7
4  00  7 10
5  11  7  0
6  00 13  7
7  00  7  9
8  00  9 13
9  11  6  0
10  00  0  0
```

To build the "t" tree, we have:

```
TWO   t uw        14 16  0  0  0  0  0  0
```

```
0  00 14 16
1  11  2  0
2  00  0  0
```

To build the "th" tree, we have:

```
THREE          th r iy      15 12  8  0  0  0  0  0

0  00 15 12
1  00 12  8
2  11  3  0
3  00  0  0
```

To build the "w" tree, we have:

```
ONE    w ah n       18  2 10  0  0  0  0  0

0  00 18 12
1  00 12 10
2  11  1  0
3  00  0  0
```

To build the "z" tree, we have:

```
ZERO   z iy r ow     19  8 12 11  0  0  0  0

0  00 19  8
1  00  8 12
2  00 12 11
3  11  0  0
4  00  0  0
```

Once all of the branches have been built, the next stage is to concatenate the tree. It appears that four elements make a page, and each element is two bytes.

```
                                                  page
ey:     00  5 14   11  8  0   00  0  0   00  0  0    5
 f:     01  4  0   00  6  3   00  3 17   11  5  0    6
        00  6 11   00 11 12   11  4  0   00  0  0    7
 n:     00 10  3   00  3 10   11  9  0   00  0  0    8
ow:     11 11  0   00  0  0   00  0  0   00  0  0    9
 s:     01  6  0   00 13  4   00  4 17   00 17  7   10
        00  7 10   11  7  0   00 13  7   00  7  9   11
        00  9 13   11  6  0   00  0  0   00  0  0   12
 t:     00 14 16   11  2  0   00  0  0   00  0  0   13
th:     00 15 12   00 12  8   11  3  0   00  0  0   14
 w:     00 18 12   00 12 10   11  1  0   00  0  0   15
 z:     00 19  8   00  8 12   00 12 11   11  0  0   16
```

Then, the start_lookup header is defined. Each row (page)is 8 bytes and hence the header takes 3 rows. The end_lookup header is two rows. Therefore, the lexical tree store proper starts at the 6th (#5) page.

```
sp sil  ah  ay  eh  ey   f  ih          - MP
 0   1   2   3   4   5   6   7          - MP_id
 3   0   0   0   5   0   6   0          - data stored.

iy   k   n  ow   r   s   t  th
 8   9  10  11  12  13  14  15
 0   0   8   9   0  10  13  14

uw   v   w   z   .   .   .   .
16  17  18  19   .   .   .   .
 0   0  15  16   0   0   0   0
```

The end_lookup header is defined. 20 is out of bounds.

```
ey   f   n  ow   s   t  th   w
 5   6  10  11  13  14  15  18

z   .   .   .   .   .   .   .
19  20  20  20  20  20  20  20
```

To put the above information all together:

```
START       sp:ih     3 0    0 0      5 0    6 0     0
LOOKUP      iy:th     0 0    8 9      0 10   13 14    1
            uw: v      0 0   15 16    0 0    0 0     2

END         ey:w       5 6   10 11    13 14  15 18    3
LOOKUP      z        19 20   20 20    20 20  20 20    4

LEXICAL     ey:     00 5 14  11 8 0  00 0 0  00 0 0    5
TREE        f:      01 4 0   00 6 3  00 3 17  11 5 0   6
STORE               00 6 11  00 11 12  11 4 0  00 0 0   7
            n:      00 10 3  00 3 10  11 9 0  00 0 0    8
            ow:     11 11 0  00 0 0  00 0 0  00 0 0     9
            s:      01 6 0  00 13 4  00 4 17  00 17 7   10
                    00 7 10  11 7 0  00 13 7  00 7 9    11
                    00 9 13  11 6 0  00 0 0  00 0 0     12
            t:      00 14 16  11 2 0  00 0 0  00 0 0    13
            th:     00 15 12  00 12 8  11 3 0  00 0 0   14
            w:      00 18 12  00 12 10  11 1 0  00 0 0  15
            z:      00 19 8  00 8 12  00 12 11  11 0 0  16


  3 0     0 0     5 0     6 0
  0 0     8 9     0 10    13 14
  0 0    15 16    0 0     0 0
  5 6    10 11    13 14   15 18
 19 20   20 20    20 20   20 20
00 5 14  11 8 0  00 0 0  00 0 0
01 4 0   00 6 3  00 3 17  11 5 0
00 6 11  00 11 12  11 4 0  00 0 0
00 10 3  00 3 10  11 9 0  00 0 0
11 11 0  00 0 0  00 0 0  00 0 0
01 6 0   00 13 4  00 4 17  00 17 7
```

```
00  7 10   11  7  0   00 13  7   00  7  9
00  9 13   11  6  0   00  0  0   00  0  0
00 14 16   11  2  0   00  0  0   00  0  0
00 15 12   00 12  8   11  3  0   00  0  0
00 18 12   00 12 10   11  1  0   00  0  0
00 19  8   00  8 12   00 12 11   11  0  0
```

This information is converted to hexidecimal. The first five are byte fields.
The rest are:
2bit - uCode.
6bit - data.
2bit - null.
6bit - data

```
03 00    00 00    05 00    06 00
00 00    08 09    00 0A    0D 0E
00 00    0F 10    00 00    00 00
05 06    0A 0B    0D 0E    0F 12
13 14    14 14    14 14    14 14
05 0E    C8 00    00 00    00 00
84 00    06 03    03 11    C5 00
06 0B    0B 0C    C4 00    00 00
0A 03    03 0A    C9 09    00 00
CB 00    00 00    00 00    00 00
86 00    0D 04    04 11    11 07
07 0A    C7 00    0D 07    07 09
00 0D    C6 00    00 00    00 00
0E 10    C2 00    00 00    00 00
0F 0C    0C 08    C3 00    00 00
12 0C    0C 0A    C1 00    00 00
13 08    08 0C    0C 0B    C0 00
```

The above is the micro code for the digit lexical tree. It allows branching, and it holds the word_id. It knows which lexical trees it contains and where they are housed. It also stores the possible end monophone list.

Figure 8 shows a further example of the Lexical Tree RAM, giving examples of different types of branching in the lexical tree, and how these are represented in the RAM.

Figure 9 is a block diagram showing the data structure that is supported for the word link record, to enable word n-best decoding. The word engine maintains a record of the words that have been currently searched by the PINE array. It is used to determine the word history and allow trace back during or at the end of search. The word history is needed to determine the language model score and whether particular paths can be recombined or merged.

Figure 10 shows a block diagram of the word engine. This is designed to allow fast access to the word history (thus allowing rapid phone creation at the phone level), and to allow reasonably fast trace back. The Word Engine is primarily a memory store that fills incrementally. The decoupling of the wordlat_node_id and the physical address will allow the possibility of being able to keep the WLRs for a given word in contiguous addresses, thus avoiding the need for the pointer field.

The language model interface is designed to allow the Phone Book to quickly obtain the appropriate LM factor for the phone model that it is currently constructing. Based on the phones in the wave front, all possible next LM factors can be determined. This will be done in software together with the language model factor engine. The ARM 9 will maintain its own record of the wavefront using the same next phone requests received by the wave front manager and its own copy of the lexical tree.

Figure 11 shows the architecture of the LM interface. This allows the Phone book to quickly access the LM factor for the phone that is currently being generated. Prior to the phone generation phase LM_factors are computed for all possible phones that may be generated in the next frame.

The grammar engine is designed to give the Phone Book a quick response to requests for the next grammar node(s). In the case of a task grammar where the lexicon may be limited to a few words(e.g. yes, no), it may be desirable to use the same lexical tree. In this case it will be necessary for the grammar, to provide lexical tree pointers such that the phone generator can jump to the appropriate part of the lexical tree.

For a speech recognition system it is also necessary to specify the grammar. Again we will use HTK's Standard Lattice Format(SLF) to specify the grammar. In practice it will be convenient to use HTK's HParse format to specify the grammar using a high level grammar notation. This can then be converted into using the HParse tool. Consider having to specify a grammar for a digit recognition task. The HParse format for this is as follows:

$digit = one | two | three | four | five | six | seven | eight | nine | zero

$digit_sp = $digit sp

$noise = lipsmack | breath | sil

([sil] < $digit_sp | $noise > [sil])

where | denotes alternatives, < > denotes one or more repetitions, { } denotes zero or more repetitions, [] denotes that item is optional, and () contains the grammar.

Figure 12 shows the network that is defined by this grammar. The network starts and ends with a "sil" phone, representing silence. Between the start and end, a number of possible routes correspond to further silence, a breath, a lipsmack sound, or a detected word.

Figure 13 shows the grammar engine architecture. When a next phone request arrives, the next grammar node(s) must be retrieved from memory. In the case of a task grammar, the phone book should be given the lexical tree memory addresses to allow the phone to be determined.

Figure 14 shows a single frame of the Audio Codec (AC) link protocol used in some embodiments of the invention. The Advanced Audio CODEC Interface (AACI) is an ARM Primecell that provides an interface to off-chip AC97 audio codecs. AC97 is Intel's audio codec specification, initially created for the PC market. The AACI is an AMBA slave block that connects to the Advanced Peripheral bus (APB). The AACI connects to the off-chip codec using the AC-link protocol.

The AC-link protocol is a bi-directional, fixed clock rate, serial digital stream. AC-link handles multiple input and output PCM audio streams as well as register control accesses. It employs a time division multiplexed (TDM) scheme that divides each audio frame into 12 outgoing and 12 incoming data streams each with up to 20 bit sample resolution. The tag slots are used to indicate which of the subsequent slots contain valid data. Add and dat are the control register address and data slots for read and writes. The adc slots contain the outputs from the analog to digital converters. Assuming that the

ADC sample rate is set to 16 kHz then the AC-link controller will receive 16 bits of data every third codec frame.

The AACI has 4 channels that can be operated simultaneously. They are accessed through DMA.

Figure 15 shows the system timing. Each frame is divided into three sections:

- Token update (MHD wash)
- First pass - inter phone token passing
- Second pass – garbage collection, max model pruning

The horizontal direction of the figure indicates elapsing time, and various subcircuits and processes are indicated along the vertical direction of the figure. The top part of the diagram shows frame count, and push to dictate button, and the lower part of the diagram shows various subcircuits and processes, within a single time frame, whilst the push to dictate button is engaged. These subcircuits and processes are language model interface, word engine, phone book, PINE array, MHD engine, mean & covar reg, and feature vector reg.

At the start of the time frame, the feature vector is generated, and the MHD values are calculated. The MHD values are then passed to the PINE array, which engages in beam pruning and token score updating. The phone book receives next phone requests and returns next phone responses. The word engine adds word link records, and the language model interface supplies language model factors.

In the second part of the frame, pruning thresholds are calculated for the PINE array, for the next frame, and the PINE array engages in token passing. The language model interface transfers language model factors for the next frame.

Whilst the "push to dictate" button remains activated, the above process repeats in a sequence of frames.

The embodiments of the invention which are now described relate to a distance engine for calculation of Mahalanobis Distances.

Figure 16 shows a block diagram of a speech recognition system according to an embodiment of the invention. The system includes a microphone 100, which outputs an audio signal to a combined AGC (automatic gain control) and ADC (analogue to digital converter) 101. In alternative embodiments, the AGC and ADC may be separate. In this embodiment, the incoming data comes from a microphone, and is then converted to digital format. However, it is also possible that the digital audio data may be received from a network, or may be read from storage means such as a memory, an optical disk, or a hard disk drive.

The digital audio data output from the ADC is input to an audio data buffer 102, and may be held here until required. The data is then output from the audio data buffer 102 to a front end circuit 103. The incoming audio data is divided into frames, as described above. In cases where the system is extremely busy, it is possible to drop some audio frames, to allow the system to catch up with any backlog.

In the audio front end circuit 103, the audio is split into spectral components to generate a feature vector, as described above. For example, the feature vector may have a number of frequency bands, and an energy band, plus their first and second derivatives. The feature vector is then output from the front end 103 to the distance calculation engine 104.

The distance calculation engine 104 converts feature vectors into Mahalanobis distances . The acoustic model may be chosen from a plurality of acoustic models, e.g. according to the language used, and may be stored in a storage means 105 such as a memory, a disk drive, etc. The distances calculated by the distance calculation engine are then transferred to the search stage 106 of the speech processing circuit, which uses models such as one or more word models and/or language models to generate and output recognised text. Such word and language models may be stored in a separate storage means 107.

In one embodiment, the distance calculation engine uses Mahalanobis Distance (MHD) calculations to determine distance values for output to the search stage 106. In further embodiments, different types of distance calculations may be used, e.g as described above.

In some embodiments of the invention, the distance calculation engine is designed as a speech accelerator, to operate as a loosely bound co-processor for a CPU running speech recognition software. This has the advantage of reducing computational load on the CPU, and reducing memory bandwidth load for the CPU.

In some embodiments, the accelerator interface is memory-mapped. Other types of interface are also possible to support different system configurations, for example removable accelerator cards with SDIO (secure digital input/output), CardBus, or USB interfaces.

The speech recognition apparatus may also operate with a CPU and DSP (digital signal processor), where the DSP implements the "front-end" signal processing to produce a feature vector, and the CPU implements the search stage, reading the distance results from the distance calculation engine.

The system may be implemented in any combination of hardware or software.

In some embodiments of the invention the search stage 106 provides the distance calculation engine 104 with an active state list 108 that indicates the subset of states for which the search stage requires distances to be calculated by the distance calculation engine 104. This is an optimization that may reduce computation time and/or power consumption.

In preferred embodiments of the invention, the front end 103, distance calculation engine 104 and the search stage 106 operate in a pipelined manner. When operating in a pipelined manner it is unlikely that the search stage 106 will be able to provide the active state list 108 early enough for the distance calculation engine 104 to implement the optimization of computing only the distances that will be required by the search

stage 106. The distance calculation circuit 104 may calculate the MHD values for every state in the lexicon, per frame, whether it is subsequently required by the search stage or not. This allows the accelerator and software system to operate in a concurrent pipelined manner, which maximizes the system throughput.

The distance calculation engine 104 may have a plurality of memory areas for storing feature vectors, to allow a feature vector for one frame to be transferred to one memory area in the distance calculation engine 104, while another feature vector for another frame is being read from a second memory area, for calculation of a distance. The memory areas used for reading and writing may alternate with subsequent frames.

Figure 17 shows a block diagram of a distance calculation accelerator 200, used in an embodiment of the invention. A feature vector is received at input 212 of the accelerator 200. The feature vector is loaded into feature vector (FV) memory A 208 via a bus 211. In reality, there may be more than a single bus, but for simplicity, only one bus 211 is shown in the figure.

The registers 203 are discussed further at the end of this description. The registers 203 may include, for example, an accelerator status register, for indicating whether the acoustic model data is correctly stored in memory, as described in more detail below.

The FV registers 209 hold the feature vector whose distances are currently bring computed by the distance calculation engine 204. If the distance calculation engine 204 is implemented in software, the FV registers might be implemented as memory attached to the processor running the software. If the distance calculation engine 204 is implemented in hardware the FV registers might be implemented as logic or as a memory block. The FV registers 209 are loaded from the FV memory 208 when the Distance Calculation Engine 204 has completed distance calculations for the current feature vector and a subsequent feature vector is available in the FV memory 208. In some embodiments there is an optional FIFO associated with FV registers 209 so that the FIFO is loaded from the FV memory 208 when there is a subsequent feature vector in the FV memory 208 and there is a space in the FIFO. In these embodiments the FV Registers 209 are loaded from the associated optional FIFO 209 when the Distance

Calculation Engine 204 has completed calculations for the current feature vector and there is a subsequent feature vector in the associated optional FIFO 209. The presence of the FIFO 209 provides for a looser coupling between the Front End 103 and the accelerator 200 which improves performance in some system configurations.

Each feature vector is transferred to a distance calculation engine circuit 204, to obtain distances for each state of the acoustic model. The acoustic model is stored in an acoustic model memory 205, which is connected to the bus 211, for control of reading/writing to the memory. In some embodiments, this also allows acoustic models to be changed or updated via the bus. The acoustic model may be stored in compressed form in the acoustic model memory 205. A decompressor 206 is provided on the data output of the acoustic model memory. The decompressor 206 can decompress the results as they are needed, and thus allows a much smaller size of acoustic memory to be used, than with uncompressed data. From the decompressor 206, the decompressed acoustic model states are sent to the distance calculation engine 204.

The distance calculation engine 204 converts feature vectors and acoustic model states to distances. The accelerator 200 includes two separate results memories for storing these calculated distances, labelled as "result memory A" 201 and "result memory B" 202.

The control, sequencing, and address generation 210 is logic, state machines, and/or software programs that provide overall control of the other elements of the accelerator, taking inputs from the registers 203, and all the other elements and interfaces to determine when to start and stop the Distance Calculation Engine 204. The control 210 also produces addresses and control signals to control the reading and writing of the memories 201, 202, 205 and 208, to control the flow of data between the various elements of the accelerator 200, the operation of the Decompressor 206, CRC check 207 and Distance Calculation Engine 204, and to control the setting or clearing or status bits in the Registers 203 and to control signalling to the front end 103 and search stage 106. The control 210 is implemented using any of the widely known and used methods of controlling and sequencing in digital systems including logic gates, state machines, and embedded software systems. The control 210 might be a centralized control, or it might

be distributed in some or all of the other elements of the accelerator 200, or some combination of these.

Each time a feature vector is loaded into the accelerator, the accelerator computes the distances for all states for that feature vector, and stores the results alternately in the A or B Results Memory. The results are written to one of the two memories 201, 202, while the other memory is available for reading. Thus, while the distance calculation engine 204 is storing the distances calculated for one audio frame, the other result memory may make available the distances calculated for an earlier audio frame, to the search stage of the speech recognition system. For non-real time processing, "earlier" audio frame may refer to an audio frame that has been processed earlier, rather than being limited to an audio frame that corresponds to speech at an earlier time.

Thus, the search stage software reads results alternatively from the A or B result memory, while the accelerator writes results into the other result memory, allowing concurrent pipelined operation of the accelerator and the CPU [and the DSP if present].

In some implementations the A and B results memories may be exposed to the software interface as separate memories appearing at different locations in the CPU and/or DSP memory map. Another implementation is to map each memory alternatively to the same range of addresses in the CPU and/or DSP memory map, so that the software always accesses results at the same location, and the memory containing the set of results the software currently needs to access is the one mapped there.

The accelerator may autonomously load and decompress an acoustic model, which is stored in a compressed format in the acoustic model memory 205. The Acoustic Model may be loaded into the Acoustic Model Memory by software running on the CPU prior to the first use of the Accelerator. The Acoustic Model Memory may be an on-chip memory, to maximize bandwidth and minimize power consumption of reading in the entire Acoustic Model every frame, which means 50 to 100 times per second, depending on the system design.

The Acoustic Model Memory may be a dedicated memory attached to, or incorporated within, the accelerator. Alternatively the Acoustic Model Memory may be a shared resource that is used by other system functions or applications when speech recognition is not running. To allow detection of situations where the acoustic model has been overwritten by another application, a pre-calculated CRC (cyclic redundancy check) signature may be stored with the acoustic model. Each time the acoustic model is read into the accelerator, a CRC may be computed and compared with the signature. If the comparison fails, an error bit is set in the accelerator status register.

Some embodiments of the invention relate to speech recognition systems comprising separate front end 103 and search stage 106 processors. These separate processors perform the front-end and search stages of the recognition process, with the accelerator 200 performing the distance calculations. Typically the front-end processing would be performed on a DSP and search stage would be performed on a CPU, although this description holds for any type of processors, even dedicated hardware implementations. As shown in figure 17, the front end 103 supplies the distance accelerator with feature vectors, and the search stage 106 uses distances calculated by the accelerator.

The accelerator is designed to operate in a concurrent pipelined manner with the other processors, allowing a high level of parallelism in the system.

Figure 18 shows how this pipelined operation works in an embodiment of the invention with ideal data flow, i.e. where each of the three recognition stages require exactly the same time to process each data frame, and the processors are never delayed, stalled, or diverted to other tasks.

The time axis is divided into equal "pipeline step times". If the step time is the same as a frame time then the system is processing audio in real time. If the step time is short than a frame time, the system is operating faster than real time, which may be valuable for batch processing of audio data, or catching up after some type of delay. If the step time is loner than a frame time, the system is operating slower than real time, which may be acceptable in some applications, for example where the person dictating pauses occasionally allowing the system to "catch up".

Once the pipeline has filled (which takes two step times), the DSP, the Accelerator, and the CPU are all operating in parallel. In this idealized example, the total time available for processing a frame is three step times – one step time in the DSP, one step time in the accelerator, and one step time in the search stage. There is a latency of three step times from the time the audio data enters the front end to the completion of the search stage for that frame, and the processing of one frame finishes each step time. If the system were not pipelined, then the processing of all stages for one frame would have to complete in one step time in order to achieve the same overall throughput of frames per second. Thus pipelining in this way can achieve three times the throughput with the same processing elements, compared to a non-pipelined system. A non-pipelined system would be one in which the processing of frame $n+1$ would not start until all stages of the processing of frame n was already complete.

If, for example, in a system running in real time with a frame time and step time of $1/100^{th}$ of a second, which is 10ms, each of the three processing elements has the entire 10ms available to complete the processing of one stage of the recognition task, giving 30ms of processing time overall for each frame. If the system were not pipelined, real time operation would require that all three processing elements would be complete in 10ms, allowing only 3.3ms for each of the three processing elements.

At the end of each pipeline step time, the DSP provides a feature vector to the Accelerator, and the accelerator provides a set of distance results to the CPU. When the CPU is performing the search stage for frame n, the Accelerator is computing the distances for frame $n+1$, and the DSP is performing the front-end stage for frame $n+2$. During each pipeline step, the front end processing consumes one frame of digital audio data (although this is not indicated explicitly on the diagram).

Real world systems are more complex that the idealized system in many ways. The processing time for a frame is often highly variable and data dependent, especially in the search stage, and the processing time for each processing stage is likely to be quite different. The DSP and CPU are likely from time to time to be diverted to run other software tasks, either to service interrupts, or under control of an operating system

scheduling regime. Communications between the processing elements may be stalled due to competing bus traffic, for example for DMA transfers.

Recognition software may implement "frame dropping" and other advanced techniques, and as a result not all frames are processed through all of the recognition stages. Processing elements cannot respond instantly to the availability of data from the preceding processing stage, resulting in delays.

Finite buffer space for information storage between stages means that an earlier stage may be stalled waiting for a space to store its output data when following processing stages are "falling behind".

The accelerator has been carefully designed to mitigate these problems. Figure 19 shows a pipelined system with real world complications added. The following text describes the interactions shown in the diagram. The time for the processing of various stages has been varied to show difference examples of buffering of data between stages and stalling of processing elements. The numbering corresponds to the numbered bubbles on the diagram. The text also makes reference to memories used to pass information into and out of the accelerator, which are shown in figure 17.

The front end has computed the feature vector for the first frame, FV1. It is loaded into the accelerator, causing the accelerator to commence computation of the distances for the frame.

The feature vector for the second frame, FV2, is passed to the accelerator. The accelerator is still busy computing the distances for the first frame, so this feature vector is held in the accelerators FV memory.

The front end has computed the feature vector for the third frame, FV3, but it cannot be delivered to the accelerator as the FV memory is still full, holding FV2. The time period labelled "stall delivering FV3" is the period between FV3 being computed and when it can first be delivered. Of course the DSP could be diverted to some other useful task, or it could buffer FV3 in its own memory and commence computation of FV4.

The arrow labelled "FV3 earliest" shows the earliest time that the feature vector FV3 can be delivered to the accelerator, which is immediately after the accelerator loads FV2 into its internal registers and frees the memory.

The arrow labelled "FV3 latest" shows the latest time that the feature vector FV3 can be delivered to the accelerator without causing the accelerator to stall for lack of a feature vector to work on. This example shows that there is a wide tolerance in the system to when FV3 can be delivered without affecting system performance – an entire accelerator frame processing time. The accelerator provides an interrupt and a status bit that can be polled to indicate the state of the FV memory. Regardless of which control method the DSP uses, the system tolerates long latency without reducing performance.

The accelerator signals to the search stage that the distances for frame 1 are available in result memory A. The search stage commences processing for frame 1, reading distances from result memory A.

The search stage has finished its processing for frame 1 before the accelerator has completed computing the distances for frame 2, so the search stage stalls waiting for the accelerator. Of course the CPU could be diverted to other useful tasks, including any search stage tasks that can be completed before distances are available. Again, an interrupt and status bits are provided by the accelerator to control this interaction between the accelerator and the CPU. .

The accelerator completes the distance calculation for frame 2, and signals to the CPU that it can commence reading the distances from result memory B.

The accelerator has completed the computation of distances for frame 3 before the search stage has finished processing frame 2. As soon as the search stage signals to the accelerator that it has completed frame 2 by releasing memory B, the accelerator signals that the distances for frame 3 are available, allowing the search stage to immediately move on to processing stage 3.

The accelerator is stalled waiting for feature vector FV4, due to the long processing time in the front end. In real systems this may be caused by a number of factors, including the DSP being diverted to another task for a period of time, or simply due to the DSP performance being insufficient to keep up with the accelerator's throughput.

The accelerator restarts as soon as FV4 is delivered by the front end.

The search stage is stalled waiting for distances for frame 4, a follow on consequence of the late delivery of FV4 to the accelerator.

FV5 is delivered to the accelerator while it is still computing the distances for the previous frame, allowing it to move onto processing frame 5 without stalling when it completes frame 4.

The accelerator signals that the distances for frame 4 are available and the search stage commences processing frame 4.

FV6 is delivered to the accelerator while it is still computing the distances for frame 5.

The availability of distances for frame 5 is signalled by the accelerator to the CPU.

Even though the accelerator has already received FV6, it is stalled waiting for a free result memory it can write the distances into.

The search stage releases result memory B at the end of its processing of frame 4. This allows the accelerator to start computing distances for frame 6.

The accelerator completes computing distances for frame 6 and signals to the CPU that the results are ready.

Further embodiments of the invention relate to systems with a Front End and Search Stage on a single processor. This single processor performs the front-end and search

stages of the recognition process, with the accelerator performing the distance calculations. This configuration is likely to be the most common configuration where the accelerator is added to an embedded system such as a PDA or mobile phone that has a high-performance embedded application processor (but no DSP) available to run speech recognition software.

In this configuration the accelerator still operates in a concurrent pipelined manner with the front-end and search stages; it is just that the front-end and search stages are running on a single processor.

CPUs in embedded systems are generally relatively simple compared to desktop PCs and servers. Embedded CPUs in mobile phones, PDAs, in-car navigation and telematics systems, and other consumer electronics devices are typically "simple uni-processors" by which we mean not multi-processors and not supporting hardware multi-threading support – i.e. they can only execute a single thread of instructions (although they can be diverted to a different thread under the control of interrupts, exceptions, or an operating system). For this discussion, except where explicitly stated otherwise, we are talking about embedded processors that are "simple uni-processors". We are also using the term "CPU" and "processor" interchangeably.

Figure 20 shows how the system can be pipelined with a single embedded processor working with the accelerator, in an idealized system. In this idealized system, the time taken for the processor to perform the front-end and search stage processing is identical to the time the accelerator takes to perform the distance calculations, and the processor is never delayed, stalled, or diverted to other tasks.

The time axis is divided into equal "pipeline step" times. If the step time is the same as a frame time then the system is processing audio in real time. If the step time is short than a frame time, the system is operating faster than real time, which may be valuable for batch processing of audio data, or catching up after some type of delay. If the step time is loner than a frame time, the system is operating slower than real time, which may be acceptable in some applications, for example where the person dictating pauses occasionally allowing the system to "catch up".

The first two pipeline step times are used to fill the pipeline. By the third pipeline step time, the pipeline is full, and all three stages of the recognition process are processed every pipeline step time, with each stage working on the processing of a different data frame.

During the first pipeline step time, the CPU completes the front-end processing for frame 1, and the feature vector passed to the accelerator so that distance calculations can start in the accelerator.

During the second pipeline step time, the front-end processing for frame 2 is completed by the CPU, in parallel with the accelerator computing the distances for frame 1.

During the third pipeline step time, the CPU runs the search stage for frame 1 reading distance results from the accelerator, and then runs the front-end processing for frame 3. The accelerator runs in parallel computing the distances for frame 2 using the feature vectors computed in the previous pipeline step.

Each subsequent pipeline step time is similar to the third pipeline step, with the frame numbers incremented. So for pipeline step $n = 4, 5, 6$, etc. the CPU runs the search stage for frame $n-2$ and then runs the front-end for frame n. In parallel the accelerator computes the distances for frame $n-1$.

Note that with this situation the production of feature vectors and the consumptions of results run in lock step - one feature vector is produced every time the search stage consumes one frame of distance results. This considerably reduces the need for elastic buffering between the distance accelerator and the software system.

Again, real world systems are more complex than the idealized system in many ways. For example, the processing time for a frame is often highly variable and data dependent, especially in the search stage, and the processing time for each processing stage is likely to be quite different. The CPU is likely from time to time to be diverted to run other software tasks, either to service interrupts, or under control of an operating

system scheduling régime. Communications between the CPU and accelerator may be stalled due to competing bus traffic, for example for DMA transfers.

Recognition software may implement "frame dropping" and other advanced techniques, and as a result not all frames are processed through all of the recognition stages. Processing elements cannot respond instantly to the availability of data from the preceding processing stage, resulting in delays

Finite buffer space for information storage between stages means that an earlier stage may be stalled waiting for a space to store its output data when following processing stages are "falling behind".

The accelerator has been carefully designed to mitigate these problems. Figure 21 shows a pipelined system with real world complications added. The following text describes the interactions shown in the diagram. The time for the processing of various stages has been varied to show difference examples of buffering of data between stages and stalling of processing elements. The numbering corresponds to the numbered bubbles on the diagram. The text also makes reference to memories used to pass information into and out of the accelerator, which are shown in figure 21.

At the end of front-end processing for frame 1, the CPU passes the feature vectors for frame 1 to the accelerator, allowing distance calculations to start.

To fill the pipeline, the CPU immediately performs the front-end processing for frame 2 and passes the feature vectors to the accelerator. They are stored in the accelerator so that the accelerator can load them into its internal engine for processing as soon as the distance calculations for frame 1 are complete.

The CPU is now ready to start the search stage for frame 1, however the distance computation in the accelerator has not completed. The box labeled "Prep search stage 1" indicates a period where the CPU could do any preparatory search stage processing that does not require distances. Alternatively the CPU could be diverted to another task, or it can stall waiting for the distance results.

The accelerator signals that the distance results for frame 1 are now available. The search stage can commence on the CPU, reading distance results from the accelerator's Result Memory A. The accelerator automatically starts processing the feature vector for frame 2 that was delivered earlier.

The accelerator signals to the CPU that it has completed the distance calculations for frame 2. The CPU is still busy running the search stage for frame 1, and has still to run the front end for frame 3, so the completion signal is ignored until the CPU has need of it, when it starts running the search stage for frame 2. As the accelerator has not received the feature vector for frame 3, it has no task it can perform and so it stalls.

The CPU delivers the feature vector for frame 3, restarting the accelerator, which writes results into Result Memory A. Note that Result Memory A was previously "released" at the end of the processing of search stage 1, informing the accelerator that it can overwrite the old results held in that memory with new results. The CPU then runs the search stage for frame 2, reading distance results from the accelerator's Result Memory B.

The accelerator signals to the CPU that the distance results for frame 3 are complete in Result Memory A. Again, the CPU ignores this signal until it is ready to commence the search stage for frame 3, and the accelerator stalls waiting for the next feature vector.

The CPU delivers the feature vector for frame 4, restarting the accelerator. It them starts search stage 3 reading distances from the accelerator's result memory A.

Search stage 3 and front end 5 were completed quickly, so that the feature vector for frame 5 is delivered to the accelerator while it is still running. The CPU performs all the preparatory work it can for the search stage, and then stalls waiting for distances to be available.

The accelerator signals that the distances for frame 4 are available. The CPU starts search stage 4, reading the distances from the accelerator's result memory B. The

accelerator automatically starts computing distances for frame 5 using the feature vector delivered earlier.

Note that with the pipelining scheme just presented which alternates front-end and search stage processing, the accelerator can never stall waiting for a result memory to be released because the feature vector the accelerator needs as an input is not produced until after the memory has been cleared. The CPU task schedule could be described with the shorthand F1 F2 S1 F3 S2 F4 S3 F5 S4.

It can seen readily from figure 21 and the accompanying description above, that with this CPU task schedule, that the operation of the CPU and accelerator are tightly coupled and whenever the processing time for a pipeline step on the CPU is unequal to the processing time on the accelerator, then it quickly results in one of them stalling.

In a system where either the CPU or the accelerator is always (or at least in most instances) slower than the other in completing its processing task, stalling is inevitable and throughput may not be improved by adding additional buffering (e.g. additional result memories or FIFO depth on the feature vector interface). [Unless there are other difficulties to mitigate such as very low-bandwidth buses linking the CPU to the accelerator, or competing bus traffic such as DMA transfers that can significantly reduce performance.

If the processing time for a pipeline step is highly variable on either the CPU or Accelerator, such that there are regular changes in which one of them takes the longer time, then significant enhancements in throughput may be obtained by adding additional buffering between them. This may have the disadvantage of adding additional latency through the system. This buffering may be implemented by adding additional depth to the Feature Vector FIFO, and/or adding additional Result Memories. Alternatively it may be by implementing elastic buffers in software.

If the accelerator has sufficient buffering on the feature vector interface, e.g. by including the optional FIFO shown in figure 21, then increasing the pipeline depth by computing extra front end frames up front will improve throughput at a cost of extra

latency. This will result in a CPU task schedule of F1 F2 F3 S1 F4 S2 F5 S3 F6 S4 or F1 F2 F3 F4 S1 F5 S2 F6 S3 F7 S4, etc.

Of course it would also be possible to dynamically schedule whether the CPU should run the front-end or search stage code based on the availability, or not, of distance results and/or the space available to store more feature vectors and distance results. Given the fundamental relationship that one set of distance results is produced by the accelerator for every feature vector it consumes, and the restriction stated at the start of this section that we are discussing single-threaded CPUs, dynamic scheduling is likely to add little performance improvement while being complex to implement.

On a CPU supporting multi-threaded operation (which is a technique for improving performance in the presence of high memory system latency), or on a multi-processor, the highest system performance is likely to be achieved by running the front-end and search stages concurrently as separate threads, rather than alternating between the front-end and search stages. The system operation is then very similar to that described above, if each thread is thought of as a separate virtual processor. In some embodiments of the invention, elastic buffers may be used between stages, to accommodate varying time delays between the processing of frames at each stage. Elastic buffers in the interfaces between the three recognition stages may also significantly enhance performance of such a system.

Operation of the accelerator and the software system is not constrained to a rigid repetitive frame time. Instead the timing of the system is controlled dynamically by the availability of feature vectors into the accelerator, and the consumption of results out of the accelerator.

This allows the recognition process to fall behind real-time operation and to "catch up" again. It also allows the system to be robust in complex systems where interrupts, DMA bus traffic, memory caches, operating system task switching, and competing real-time software tasks all make guaranteeing rigid timing constraints impractical.

Elastic buffers at the major data transfer points between processing elements maximize system performance by allowing one element to continue useful processing whenever it can, regardless of whether other processing elements have been stalled or diverted to other tasks. Elastic buffers are useful for the following applications:

(1) For buffering raw audio data arriving in real time from the microphone(s) The data would typically be moved into the buffer by a DMA and/or high priority interrupt driven software process to ensure that hard real time constraints are met.

(2) For buffering feature vectors output from the front end. This is especially true if the front-end is implemented on a DSP or other processing element independent of the search stage.

(3) For buffering distance results from the accelerator. Again this is especially true if the front-end and search stages are implemented on independent processing elements. This buffering is also especially valuable if the interface between the search stage processor and the accelerator is very slow, has high latency, or is subject to competing bus traffic.

The Feature Vector and Results interfaces on the accelerator already provide a level of elasticity that should be sufficient in systems where the Front End and Search stages are running on the same CPU, and the accelerator has a high-bandwidth interface to the CPU with low level of competition from other bus traffic such as DMA.

Other embodiments may have deeper FIFOs. In one embodiment, the two result memories are an implementation of a FIFO where the implementation is exposed across the interface. It would be possible to have additional memories and use them in a round-robin fashion, or to use another implementation of a FIFO that hides the FIFO depth by only exposing one result memory at a time to the interface. An example of this would be to map each memory in turn to a memory range that is always used to read results, so that at any one time the "current" result memory is mapped to that location. This would hide the FIFO depth behind the interface so that the pipeline depth can be changed without changing the software (or hardware) that accesses the interface.

Slow or Conflicted Bus Interfaces may occur. In the system configurations described above, the search stage reads distances directly from the Accelerator's Result Memories. The result memories allow the search stage to randomly access the distance results, as they are needed. This will only give a high performance system if the interface or bus between the CPU and Accelerator provides the CPU with low latency, high bandwidth access to the Accelerator Result Memories.

In some systems it is not practical to provide such a low latency, high bandwidth interface between the CPU and the Accelerator. This may be for many reasons, including:

- The interface is a bus shared with other devices and/or processors that generate bus traffic that keeps the bus busy such that the CPU access to the accelerator is delayed waiting for the bus to be free. DMA transfers are one example of conflicting bus traffic.
- The interface is implemented as a serial bus, or narrow parallel bus, for cost or compatibility reasons, yielding low bandwidth and high latency
- The interface crosses an industry standard interface such as PCI, USB, SDIO, Cardbus, Firewire, PCIExpress etc. that introduces protocol overheads and/or latency as would be the case where the accelerator is in a plug-in peripheral to a system containing a CPU (or CPU and DSP).

In these situations it is possible to add another pipeline stage during which the result data is copied from the Accelerator into another memory (which we will call a "buffer memory") to which the CPU does have high bandwidth, low latency access. This copying can be performed by a DMA controller, or by software running as an interrupt handler, or a separate software thread on the CPU, or a software thread on another processor, or by other commonly known means. The device controlling the transfer may be on the CPU side of the interface or on the accelerator side of the interface, depending on the capabilities of the interface. This extra pipeline stage then allows an entire pipeline step time to copy the data from the accelerator to the other memory. The data copy can optionally be performed as a sequential burst, or a number of sequential bursts, to optimize transfers across the interface between the Accelerator and the buffer

memory. The search stage, running on the CPU, can then randomly access the results in the buffer memory with low latency.

Similarly an extra pipeline stage can be added for transferring the Feature Vectors to the Accelerator, although given that the feature vectors are a much smaller object than the distance results, and that they are not accessed randomly across the interface but delivered once to the Accelerator for each frame, there are fewer situations where the performance benefit from this extra pipeline stage is as significant.

Figures 22 and 23 show idealized systems with the two extra pipeline stages for transfers added. The first shows a system with a CPU, DSP, and Accelerator with the two extra pipeline stages. The second shows a system with a CPU and DSP with the two pipeline stages added.

Other embodiments may add additional pipeline stages so that the distance computations are performed over two or more pipeline stages rather than being restricted to a single pipeline stage. This allows more time for the computation of the distance results for each frame, while maintaining the same throughput of distance results. This is similar in concept to the well-known pipelining techniques used to implement RISC microprocessors.

The Acoustic Models may occupy many megabytes of storage space. In embedded systems they will typically held in a non-volatile memory such as FLASH memory, or other types of non-volatile memory as is the appropriate currently economic technology. The Accelerator may read the acoustic model directly from the non-volatile memory, or more likely the acoustic model will be copied from the non-volatile memory into a higher bandwidth RAM memory of some type (e.g. SRAM or DRAM, SDRAM, etc.). The RAM memory may be part of the same integrated circuit as the Accelerator (i.e. "on chip"), or attached to the integrated circuit containing the Accelerator. ("off chip"). The having the RAM memory on-chip is particularly advantageous, as this configuration would give the lowest power consumption and highest bandwidth. An alternative solution would be for the acoustic model to be stored on a disk drive, or an another device accessible via a wired or wireless bus or network, and for the acoustic

model to be loaded into the RAM memory from the disk drive, or over the wired or wireless bus or network.

On-chip RAM memory is however likely to come at a cost-premium to off-chip commodity memory devices. Many embedded systems face extreme cost pressures in the market place, especially high volume devices supplied by many manufacturers such as mobile phones. For a speech recognizer to be incorporated into such systems, the overall recognizer solution much be delivered at very low cost. There is therefore a significant economic benefit from reducing the cost of the RAM memory and the non-volatile memory. One way to reduce the cost of these memories is to compress the acoustic models and store the compressed model in the memory.

The Accelerator computes all of the distances associated with each Feature Vector that is delivered to it by the front end. This means that the accelerator must use the data from the entire Acoustic Model each time a new Feature Vector is presented to the Accelerator (with the exception of some specific optimizations that may determine that some data values need not be used in the distance computations). It is therefore possible to arrange for the accelerator to read through the acoustic data sequentially once for each Feature Vector, using each data value as it read from the Acoustic Model.

Reading through the entire Acoustic Model sequentially is ideally suited to placing a decompressor between the distance calculation engine and the RAM memory, because many compression schemes depend on the decompressor reading sequentially through the compressed data, and do not readily support random access into the compressed data.

Thus this configuration allows storing compressed data in the RAM memory (and the non-volatile memory), which can therefore be smaller and lower cost. There is no requirement for a memory that is large enough to hold the entire decompressed acoustic model, as the decompressed data is consumed as it is decompressed.

The decompression scheme could be any combination of:

- sign or zero extension or otherwise conversion of narrow or variable width data to a wider data format.
- sign or zero extension or otherwise conversion of narrow or variable width data to IEEE standard single or double precision floating point format
- codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data
- decompression of a Huffman or Lempel-Ziv compressed stream
- decompression of run length encoded data
- decompression of difference encoded data
- any well known compression scheme.

Additionally the model size can be reduced by using *subspace distribution clustering* and the decompressor and distance calculation engine designed to operate with such models.

The following description provides further information on the visible registers in some embodiments of the invention.

The register bank is designed to allow simple connection with an 8-bit data bus. Accesses to the register bank should be made with byte-read and byte-write accesses. To facilitate this, only the least significant 8 bits of the data bus are used.

The address decoder should respond to byte accesses to byte 0 or byte 3 within the designated word. This way the same code can be used on an ARM processor regardless of whether the processor is configured for little endian or big endian operation.

REGBASE=Base address of register bank

Register 0 Product Version

8-bit Product version ID

This register value is changed when the functionality of the hardware charges in a way that changes the hardware/software interface.

Address = REGBASE (and REGBASE+3 for big endian)

Read only. Reads with value 0x01

Writes ignored

Register 1 Implementation Version

8-bit Implementation version ID

This register value is changed for each implementation to facilitate bug tracking and version control.

Address = REGBASE+4 (and REGBASE+4+3 for big endian)

Read only. Reads with value 0x01

Writes ignored

Register 2 Interrupt Enable

8-bit Interrupt Enable register

Address = REGBASE+4*2 (and REGBASE+4*2+3 for big endian)

Readable and writeable.

Value following reset is 0 for bits 0,1,2,3

Bit 0 I 1 = Enable FV memory empty interrupt on Int0 output.

Bit 1 J 1 = Enable Results complete interrupt on Int0 output.

Bit 2 K 1 = Enable FV memory empty interrupt on Int1 output.

Bit 3 L 1 = Enable Results complete interrupt on Int1 output.

Bit 4 Future use. Read value is not defined. Writes ignored.

Bit 5 Future use. Read value is not defined. Writes ignored.

Bit 6 Future use. Read value is not defined. Writes ignored.

Bit 7 Future use. Read value is not defined. Writes ignored.

Software should use a read-modify-write code sequence when setting this register so that future use bits are written with the value read from them.

Register 3 Status

Address = REGBASE+4*3 (and REGBASE+4*3+3 for big endian)

Read-only register. Writes are ignored.

Value following reset is 0 for bits 0,1,2,3

Bit 0 FVMB Feature Vector Memory is Busy

0 = OK to write new value to FV memory (FV memory is empty)

1 = Don't write new value to FV memory (FV memory is busy)

Bit 1 CRM Current Result Memory

0 = Result memory A

1 = Result memory B

This bit toggles every time the Release Result Memory (RRM) command is issued to the accelerator (register 3). It indicates which memory software should read results from, and which result memory the RRM command should release next time it is issued. Note: Software should also check that results are available in the indicated result memory by checking the value of the MA and MB bits.

Bit 2 MA 1=Results are available is result memory A

The MA bit is set to 1 when the compute engine completes writing all of the distances for a frame into result memory A. This value is set to zero when the Release Result Memory command is issued and CRM==0.

Bit 3 MB 1= Results are available is result memory B

The MB bit is set to 1 when the compute engine completes writing all of the distances for a frame into result memory B. This value is set to zero when the Release Result Memory command is issued and CRM==1.

Bit 4 Dfault 1=Accoustic data CRC fault. Indicates that the CRC value at the end of the acoustic data did not match the CRC value computed while the acoustic data was read in.

Bit 5 Future use. Value undefined.

Bit 6 Future use. Value undefined.

Bit 7 Future use. Value undefined.

Register 4 Command

Address = REGBASE+4*4 (and REGBASE+4*4+3 for big endian)

Write-only register.

Software should perform byte writes to this register with a single bit set to 1 and all other bits set to zero to issue a command to the accelerator.

Behaviour is not defined if more than one command is issued within a single write (i.e. if more than 1 bit is a 1).

Value on read is not defined.

Bit 0 LFV Load Feature Vector from the FV memory.

Tells the accelerator that a new feature vector has been placed into FV memory ready for use by the accelerator. The FV will be loaded into the compute engine when the compute engine completes its current task (or immediately if the compute engine is idle). Computation of distances for this FV will commence when the FV is loaded AND there is a released result memory. (At reset, both result memories are released). The results for the first, third, fifth etc. feature vectors loaded are written to Result Memory A. The results for the second, fourth, sixth etc. feature vectors loaded are written to Result Memory B.

Bit 1 RRM Release Result memory. Indicates that the results in the current result memory can now be overwritten with new results. The current result memory is indicated by the CRM bit in the status register.

Bit 2 CFVI Clear the FV memory empty interrupt

Bit 3 CRI Clear the Results Complete interrupt

Bit 4 Future use – write 0

Bit 5 Future use – write 0

Bit 6 Future use – write 0

Bit 7 Future use – write 0

The present invention may be included in a mobile phone, or in another consumer electronics device, a PC, Laptop or tablet PC, in-car electronics system, or plug-in card for an electronic device.

The methods and apparatus according to the invention are not limited to embedded mobile systems, but may be used with any type of computing device, including PCs, workstations, laptop computers, notebook computers, handheld computing devices, dictation machines, PDAs (personal digital assistants), personal organisers, mobile phones, games consoles, video players, etc. Embedded computers according to the present invention may be embedded in a wide range of apparatus and machinery, such as electronic entertainment products, domestic appliances, factory machinery, vehicles, etc.

Embodiments of the present invention may be implemented as an IP core in a System on Chip (SOC) or a discrete device. Multiple chips or bare die may be connected to provide a larger vocabulary system.

Although the apparatus may include all or some of an input for accepting audio input, a circuit for digitising the audio input, a circuit for dividing the audio input into a plurality of time-frames, and a circuit for analysing the spectral components of each time frame to characterise the audio in that frame, this is not essential. The apparatus may also include a circuit for calculating a score indicating the probability or likelihood that each audio frame corresponds to each of a plurality of different possible states, but again, this is not essential. Instead, the apparatus may include an input for receiving the pre-calculated scores.

In the described embodiments, n-gram language models are used. For example, the Bigram language model is based on outputting the probability that a given second word directly follows a given first word. However, it is possible to use alternative language models instead. It is possible to use a mixture of different models. For example, it is possible to run a word-dependent n-best decode applying a bigram language model, and the resulting word link record or word graph could then be rescored using a trigram language model.

The Viterbi beam search algorithm may be used in embodiments of the invention, to perform the search. A word level search may be performed. Dummy states may be used for the start and end of an utterance

The grammar level model may be used to add distance elements into the likelihood calculation. This is not essential, but tends to give much better results.

The HMM models described in the embodiments comprise self connections to each states and single connections to a neighbouring state. However, the present invention is not limited to such HMM models, and for example, HMM models with additional connections between states may also be used. Alternatively, embodiments of the invention may use alternative types of acoustic models, rather than HMM models.

The system may comprise a plurality of language models (e.g. English, French). A model of words may comprise actual words, phrases, silence, lipsmack sounds, and/or other sounds for recognition by the apparatus.

As previously discussed, the phone level has been found to be the best level for lexical tree searching, rather than the state level or the word level. However, this invention is not limited solely to phone level searching.

Pipelining may comprises processing data in different parts of a circuit at the same time, i.e. parallel processing.

In the following claims, the calculation means may comprise the distance calculation engine described above.

Further embodiments of the invention include the methods carried out in the accelerator, and the methods carried out in the speech recognition circuit.

While the invention has been described in terms of what are at present its preferred embodiments, it will be apparent to those skilled in the art that various changes can be

made to the preferred embodiments without departing from the scope of the invention, which is defined by the claims.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 March 2006 (23.03.2006)

PCT

(10) International Publication Number
WO 2006/030214 A2

(51) International Patent Classification:
G10L 15/28 (2006.01) *G10L 15/10* (2006.01)

(21) International Application Number:
PCT/GB2005/003554

(22) International Filing Date:
14 September 2005 (14.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0420464.0 14 September 2004 (14.09.2004) GB

(71) Applicant *(for all designated States except US)*: **ZENTIAN LIMITED** [GB/GB]; St. John's Innovation Centre, Cowley Road, Cambridge, Cambridgeshire CB4 0WS (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CATCHPOLE, Mark** [—/GB]; Ash House, Breckenwood Road, Fulbourn, Cambridge, Cambridgeshire CB1 5DQ (GB). **DOWSETT, Damian, Harris**; * (**). **REYNOLDS, Timothy**; * (**). **LARRY, Guy** [—/GB]; 1 Cutter Ferry Lane, Cambridge, Cambridgeshire CB4 1JR (GB).

(74) Agent: **COLLINS, John, David**; Marks & Clerk, 90 Long Acre, London WC2E 9RA (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A SPEECH RECOGNITION CIRCUIT AND METHOD

(57) Abstract: A speech recognition circuit comprising a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

WO 2006/030214 A2

A Speech Recognition Circuit and Method

The present invention relates to speech recognition circuits and methods. These circuits and methods have wide applicability, particularly for devices such as mobile electronic devices.

There is growing consumer demand for embedded speech recognition in mobile electronic devices, such as mobile phones, dictation machines, PDAs (personal digital assistants), mobile games consoles, etc. For example, email and text message dictation, note taking, form filling, and command and control applications are all potential applications of embedded speech recognition.

However, when a medium to large vocabulary is required, effective speech recognition for mobile electronic devices has many difficulties not associated with speech recognition systems in hardware systems such as personal computers or workstations. Firstly, the available power in mobile systems is often supplied by battery, and may be severely limited. Secondly, mobile electronic devices are frequently designed to be as small as practically possible. Thus, the memory and resources of such mobile embedded systems tends to be very limited, due to power and space restrictions. The cost of providing extra memory and resources in a mobile electronic device is typically much higher than that for a less portable device without this space restriction. Thirdly, the mobile hardware may be typically used in a noisier environment than that of a fixed computer, e.g. on public transport, near a busy road, etc. Thus, a more complex speech model and more intensive computation may be required to obtain adequate speech recognition results.

These restrictions have made it difficult to implement effective speech recognition in mobile devices, other than with very limited vocabularies.

Some prior art schemes have been proposed to increase the efficiency of speech recognition systems, in an attempt to make them more suitable for use in mobile technology.

In an article entitled "A low-power accelerator for the SPHINX 3 speech recognition system", in University of Utah, International conference on Compilers, Architectures and Synthesis for Embedded Systems, Nov 2003, Davis et al have proposed the idea of using a special purpose co-processor for up-front calculation of the computationally expensive Gaussian output probabilities of audio frames corresponding to particular states in the acoustic model.

In an article entitled "Hardware Speech Recognition in Low Cost, Low Power Devices", University of California, Berkeley, CS252 Class Project, Spring 2003, Sukun Kim et al describe using special purpose processing elements for each of the nodes in the network to be searched. This effectively implies having a single processing element for each phone in the network. An alternative suggested by Sukun Kim et al is to provide a processor for each state in the network.

In an article entitled "Dynamic Programming Search for Continuous Speech Recognition" in IEEE Signal Processing Magazine, Sept 1999, Ney et al discuss language model lookahead. Language model lookahead involves computation of a language model factor for each node (i.e. phone) in the lexical tree. This technique is also known as smearing. Each phone instance in the search network can be given a language model factor when it is used in the lexical tree search. Ney et al show that for an example bigram language model, the average number of states per 10 ms frame can be reduced from around 168,000 states with no language model lookahead to around 8,000 states when language model lookahead is used. They also show that bigram language model lookahead requires about a quarter of the states compared with unigram language model lookahead.

Although these prior art documents provide improvements to speech recognition in embedded mobile technology, further improvement is still needed to provide a larger vocabulary and better accuracy.

One aspect of the present invention provides a speech recognition circuit including a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers. The lexical tree includes a model of words. The speech recognition circuit also has a memory structure for receiving and storing state identifiers identified by a node identifier identifying nodes or groups of adjacent nodes, the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores. An accumulator is provided for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores. A selector circuit is used for selecting at least one node or group of nodes of the lexical tree according to said scores.

One suitable type of hardware for the memory structure includes a content addressable memory (CAM). A CAM is a memory unit which stores a series of data items using a series of addresses. However, the memory is accessed by specifying a data item, such that the CAM returns the corresponding address. This contrasts with a random access memory (RAM) in which the memory is accessed by specifying an address, such that the RAM returns the corresponding data item.

However, the memory structure is not limited to including a CAM. Other types of hardware are also possible, to provide this functionality. For example, a single chip which operates in the same way as a CAM and RAM may be used instead.

Embodiments of the present invention provide a solution to the problem of how to map a lexical tree search to a CAM system architecture. The realisation by the present inventors that certain speech recognition data structures can be mapped into the CAMs allows a lexical tree search to be performed using a CAM system architecture.

Further embodiments of the invention include a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

The node identifier may comprise a direct reference to the lexical tree. However, in some embodiments, the node identifier for at least some of the states includes a pointer to a node identifier for another state. For example, a state corresponding to the furthest part of the search path in the lexical tree may be referenced by a node identifier which directly links to a particular node or group of nodes in the lexical tree. In a lexical tree comprising phones, using a state model of triphones, the node identifier may indicate the position of a triphone in the lexical tree.

However, in this example, for states occuring further back in the search path, instead of supplying a node identifier linking directly to the lexical tree, instead a pointer to a node identifier of another state may be supplied. E.g. a triphone instance may have a pointer to another triphone instance, which has a pointer to another triphone instance, which has a pointer to a node or group of nodes in the lexical tree. Chains of reference may be set up in this way, where only the last state in the chain has a direct pointer to the lexical tree.

There may not be a one-to-one correspondence between the nodes of the lexical tree and the node identifiers. This will occur for a branched lexical tree, where the nodes represent monophones, but the acoustic model states represent triphones, i.e. groups of three adjacent monophones. Then, paths of three monophones will have unique identifiers to be stored in the memory structure, rather than single monophones having unique identifiers.

Phone instance numbers may be generated, and used to uniquely label each phone instance. They can be generated sequentially, using a counter. The phone instance numbers may be used as pointers between phone instances to assist in node identification. It is thus not essential to provide a direct node identifier for each phone instance to directly indicate a location in the lexical tree. The dynamic network of phone

instances provided in the memory structure may thus include both direct and relative references to the lexical tree.

The memory structure may be divided into one part which stores phone instance identifiers and direct references to the lexical tree, and a second part which stores phone instance identifiers and corresponding states. This can speed up the processing, by only storing the phone instances which are furthest on in the lexical tree in the first part of the memory structure.

The memory structure may also be divided into separately accessable units, to reduce the amount to data in each unit, thereby decreasing the chance of finding the same two states identifiers in different phone instances in any single memory unit, and increasing the chance of some state identifiers being completely absent from any single memory unit. This makes it easier to deal with the situation when the same two state identifiers are found, because a spare time slot is available for processing when a state identifier is not present.

A further aspect of the invention provides a distance calculation engine within a speech recognition system. The distance calculation engine may be included within an accelerator. The accelerator may include logic to interface with other parts of a speech recognition circuit, in addition to the distance engine, although this is not essential. For example, the accelerator may include one or more results memories for storing distances calculated by the distance calculation engine. The accelerator may also include at least one of a memory for storing one or more acoustic models, a decompressor for decompressing acoustic data that has been stored in a compressed format, a memory for storing feature vectors, a checksum or data signature calculation means, buffers for data storage, and data registers. The accelerator may be implemented in software or in hardware, or in a combination. It may be physically separate to the rest of the speech recognition circuit, although this is not essential.

The distance calculation engine may calculate one or more of a wide range of distance metrics and probability distributions. The distances may represent the likely correspondance of feature vectors to states in an acoustic model. In other words, the

distances can indicate the similarity of an audio data frame to each possible state in an acoustic model

There are a wide variety of probability distributions that can be used for the distance calculation stage of a speech recogniser, and a wide variety of distance metrics used. These are widely documented in the literature. A point is a simple example of a probability distribution.

A common choice is to use Gaussian Distributions and correspondingly the Mahalanobis Distance metric. The Gaussian probability distribution is then defined by a mean vector, which defines centre point in the N-dimensional space, and a Covariance matrix which defines the shape of the probability distribution. It is common to restrict the Covariance matrix to be a diagonal matrix (only N non-zero values along the diagonal of the NxN matrix) which significantly lowers the implementation cost by reducing the number of arithmetic operations.

In particular embodiments, the distance calculated is a Mahalanobis distance. Particular examples of this are described later in the specification.

In one embodiment, the distance engine autonomously computes all of the distances associated with a given feature vector. This may comprise computing distances for every state in the lexicon. The distance engine may operate in a pipelined manner with other stages of the recognition process. In this context a distance is an indication of the probability or likelihood that a feature vector corresponds to a particular state. An important class of distance computation in speech recognition is the calculation of output state probabilities in recognisers using Hidden Markov Models. Another possible use is in recognisers using Neural Networks.

The distance engine reads data from the acoustic models to use as parameters in the calculation. The acoustic models may be optionally stored in a compressed format. The distance engine may read and de-compress the acoustic models one (or more) times for each feature vector processed. Each reading of the acoustic models may require reading

the entire acoustic model, or various optimisations may be implemented to avoid reading parts of the acoustic model that are not required to be used for calculations with the current feature vector. The distance engine may use a de-compression method where the de-compression is sign or zero extension or may otherwise convert data of narrow or variable width to a wider data format. The distance engine may use a de-compression method where the de-compression is sign or zero extension or may otherwise convert data of narrow or variable width to IEEE standard single or double precision floating point format. The distance engine may use a decompression method where decompression is a codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data. The distance engine may use a decompression method where the decompression is decompression of a Huffman or Lempel-Ziv compressed stream. The distance engine may use a decompression method where decompression is decompression of run length encoded data. The distance engine may use a decompression method where decompression is decompression of difference encoded data. The distance engine may use a decompression method using any well known lossy or lossless compression scheme. The distance engine may use a decompression method using subspace distribution clustering. The distance engine may use a decompression method comprising any combination of the above described decompression types.

The distance engine may read the acoustic models from a dedicated on-chip memory. The distance engine may read the acoustic models from a dedicated off-chip memory. The distance engine may read the acoustic models from a shared on-chip memory. The distance engine may read the acoustic models from a shared off-chip memory. Any of these acoustic models may be compressed.

The distance engine may compute a CRC or checksum or similar signature as it reads in the acoustic model and compares this to a stored CRC, checksum, or signature, in order to check that the acoustic model has not been corrupted, and signals an error condition if such corruption is detected. The stored CRC, Checksum, or signature may had been pre-computed and stored in the model data, or it may be computed at the time the model data is loaded into the Acoustic Model Memory. It may be held in Acoustic Model Memory, or it may be loaded into a register or another memory from where it can be

accessed and compared when the CRC/checksum/signature is computed each time the Acoustic Model is loaded.

The distance engine may support the pass-through of data from the front-end to the search-stage. The data to be passed through will be supplied to the distance engine as an adjunct to the feature vector, and the distance engine will pass it to the search stage as an adjunct to the distance results. This provides a simple mechanism for passing frame-specific data that is not involved in the distance calculation through to the search stage, and keeping it associated with the correct frame, which may otherwise be complex in pipelined systems with multiple processors. The data passed through may be for any purpose. Examples might include silence detected, end-of-audio-stream detected, a frame number, information that some intervening frames have been dropped, or data from another input device such as a button or keyboard in a multi-modal interface.

The distance engine may be implemented in hardware, software, or a combination. Other stages may be implemented in hardware, software, or a combination. The distance engine may be implemented with any number representation format including fixed point or floating point arithmetic, or any mixture of number representation formats.

In particular, the other stages may be implemented on a CPU, or on a DSP and CPU. The "DSP" and "CPU" may each be implemented as software programmable devices.

The distance engine may implement one or more additional pipeline stages to overcome delays introduced by low bandwidth, high latency, or conflicted bus interfaces. The distance engine may also implement additional pipeline stages to maintain the same throughput while allowing more time for each distance calculation. Particular embodiments of the invention may include one or more of the above aspects.

A speech recognition apparatus according to the invention may be embedded in or included in a mobile electronic device such as a mobile telephone, PDA (personal digital assistant), etc.

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of the system architecture for a speech recognition apparatus according to an embodiment of the invention;

Figure 2 is a block diagram showing the main data structures used in the speech recognition apparatus of figure 1;

Figure 3 shows a block diagram of the Single Phone Instance Network Engine (PINE) of figure 1;

Figure 4 is a block diagram showing the Wave Front Manager architecture;

Figure 5 shows the use of link nodes to permit path merging;

Figure 6 shows a block diagram of a Path Merge Unit in the Phone Level;

Figure 7 shows the Phone Book architecture;

Figure 8 shows Lexical tree storage;

Figure 9 shows the data structure for the word link record;

Figure 10 shows a block diagram of Word Engine;

Figure 11 shows a Language Model interface architecture;

Figure 12 shows an example of a simple Loop Grammar;

Figure 13 shows a block diagram of grammar engine;

Figure 14 shows a single frame of the Audio Codec link protocol; and

Figure 15 illustrates the system timing which may be used in the embodiment of figure 1;

Figure 16 is a block diagram showing a speech recognition circuit according to an embodiment of the invention, and illustrating data flow between parts of the circuit;

Figure 17 is a block diagram showing a distance calculation engine according to an embodiment of the invention;

Figure 18 shows ideal data flow over several time steps within a speech recognition circuit having separate front end and search stages, in an embodiment of the invention;

Figure 19 shows an example of actual data flow over several time steps within a speech recognition circuit having separate front end and search stages, in a further embodiment of the invention;

Figure 20 shows ideal data flow over several time steps within a speech recognition circuit having a shared processor for front end and search stages, in an embodiment of the invention;

Figure 21 shows an example of actual data flow over several time steps within a speech recognition circuit having a shared processor for front end and search stages, in an embodiment of the invention;

Figure 22 shows ideal data flow in a speech accelerator pipeline with two data transfer stages, according to an embodiment of the invention; and

Figure 23 shows an example of actual data flow in a speech accelerator pipeline with two data transfer stages according to an embodiment of the invention.

Figure 1 shows a speech recognition system architecture for an embodiment of the invention. The speech recognition system is divided into several levels of operation: a

state level, a phone level, a word level, a grammar level, and a system level. Each level is shown in figure 1 as a grey shaded box, interconnected to other levels of the system.

The information passed between different levels or subcircuits of figure 1 is indicated on the figure 1 using the following identifiers.

Identifier	Definition
grammar_id	Identifies which grammar model is currently being used
grammar_node_id	Identifies a node in the grammar model
word_id	Uniquely identifies a word
wordlat_node_id	Identifies a node in the word lattice
LT_id	Lexical tree identifier. Identifies the lexical tree to be used
LT_branch_id	Identifies a unique branch in a lexical tree. A branch is defined as all of the monophones subsequent to and including the first monophone in a tree.
LT_node_id	Identifies a node in a lexical tree
monophone_id	Identifies a monophone
transvec_id	Identifies which transition vector to use
state_id	Identifies the state
phone_no	Unique phone instance identifier
prev_phone_no	Phone number of previous phone in network
triphone_id	Triphone identifier
LML_factor	Language model lookahead factor. The smeared language model score that must be added

The system level includes a general purpose processor and cache, a system controller, a DMA (direct memory access) controller, clocks, PLL (phased locked loop), GPIO (general purpose I/O interface), timer watch dog, memory interfaces to interface to main memory and non volatile storage, serial, host, debug and display interfaces. The system level also includes a model adaptation engine, a language model factor engine, a speech recognition system controller, a language model interface, and a recognition output interface.

The processor organises the transfer of the various models from the non volatile storage to the various model memories. It handles the language model factor requests. Language model factors are used to by the state level to apply the language model score. The language model (LM) interface provides the mechanism for obtaining LM factors as and when they are needed. The processor keeps a copy of the wave front in memory which is updated simultaneously with that in the wave front manager. Using its copy of the lexical tree, the processor can determine which language model scores may be needed in

the coming frames. Once calculated, the language model factors can be kept in memory for use in subsequent frames. The system level is also used for forming speech model adaptation, and acting as an overflow for the various buffers used by the architecture.

The system level includes the audio codec interface and the audio front end interface, connected to the processor via a APB (advanced peripheral bus) to AHB (advanced high performance bus) bridge.

The grammar level includes a grammar engine and grammar model(s). The grammar level sets out the rules of the particular recognition task at the highest level. During system initialisation a number of grammar models might be loaded, for example a command grammar and a dictation grammar. Command mode grammar is a restricted grammar which includes only a very limited lexical tree with command specific words. Some branches of this lexical tree may include multiple words together, where a command has such a structure, e.g. "open file menu". Typically, only a single instance of the lexical tree will be needed. Command mode is used because it can obtain a higher accuracy, due to the lexical tree constraint. In dictation mode grammar, a full lexical tree structure is used.

The grammar may be selected by a user during the recognition process. For example, the command grammar may be selected when the user presses a command button. The architecture supports fast swapping of grammar models. For example, a user may be in the middle of dictating a message when a "punctuation button" is pressed and the phrase "question mark" uttered. At this point, a punctuation grammar could be loaded, in order to increase recognition accuracy.

The word level comprises a word engine, which includes a word lattice and a word graph generator. The word level has two main functions, of keeping a record of the search path at the word level using a word lattice, and interfacing to the language model via a cache. The word lattice is a record of all of the words that have been searched up to the current frame. As new words are searched, they are appended into the lattice at the appropriate place. After the final frame in the current sentence, the n-best word lattice node identifiers from the n-best scoring tokens are passed back to the word

lattice. This enables the n-best sentences to be traced back through the lattice. The word lattice can also be post processed to produce a word graph which is essentially a compact version of the word lattice.

At the word level, a record is kept of all tokens that have passed through a word end. This results in a lattice of words. When the final feature vector has been processed, the best scoring token is identified and used to trace back the search path back to the start of the search.

The phone level includes a "phone book" for generating new triphone instances, a wave front manager for sending requests to the phone book to generate new phone instances, a phone request arbiter, and a path merge unit.

The phone book includes a lexical tree, which comprises a vocabulary of words, sounds and/or phrases organised according to the phones or monophones making up the sound. A lexical tree is commonly used in speech recognition rather than a linear lexicon, because it greatly reduces the amount of storage space required to store a word vocabulary, by avoiding duplication of common word beginnings. It is also easier to search when the word beginning is known.

Although it is possible to search a lexical tree based on monophones, it is well known that better results can be achieved in speech recognition by basing the lexical tree search on groups of more than one monophone, since the sound of one syllable commonly affects the pronunciation of the next syllable in the same word, or the first syllable in the next word. In this embodiment, groups of three monophones are used, and these are known as triphones. It is alternatively possible to use a different number of monophones for each group. The phone generator in the "phone book" uses data from the lexical tree to generate triphones.

In one embodiment of the invention, approximately 45 different cued mono-phones are considered. This gives of order 60,000 tri-phones. Each triphone in this embodiment is modelled by a Hidden Markov Model (HMM) with three states. There is no correspondence between individual states and individual phones of the triphone.

Alternatively, it is possible to model each triphone by a single state, or by any other number of states.

The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree. The search space can be envisaged as a wave moving through an acousto-linguistic search space. In order that the wave can be dynamically created, a record of the wave front phone instances must be maintained. This is done in a wave front phone CAM/RAM structure, within the wave front manager. When a token occurs in the final state of a phone instance in the PINE array at the front of the wave, a new phone instance must be generated at the phone level and passed to the PINE array.

The phone instance includes a phone_no, which is a phone instance identifier to identify the phone uniquely in the current search. The phone instance also includes a word lattice node identifier (labelled "word_lat_node_id") to identify the phone's position in the word lattice, state identifiers (labelled "state_ids") for the HMM, storage to hold the tokens as they move through the HMM and LM_factor which is used to by the PINE to apply the language model score. To handle the connectivity between phone instances, a phone instance points to a previous phone instance using the phone_no of the previous phone. This allows tokens to be passed between phones. A "token" is a hypothesised search path, represented by a non-zero score at a particular state, and tokens are propagated through the HMM with each time frame.

The phone instance network can be visualised as a wave with new phone instances being added to the front of the wave, and empty phone instances being removed from the back of the wave. In some circumstances, the back of the wave may be in the same position as the wave front. The back of the wave moves on through the lexical tree as tokens are either moved on or pruned.

Tokens, each representing a particular hypothesis, hop through the states in the HMMs, each hop consuming a feature vector. Each token explores all possible next states which typically involves placing a token in the next state and keeping one in the current state. When there is more than one token in a state, the least likely token can be removed

because both have the same word history. When a token reaches the last state of a phone instance at the from of the wave, the next phone instance(s) are requested from the phone level. The phone level uses the lexical tree to create the new phones. As well as looking up the state identification numbers for each HMM state, it also incorporates the correct language model factor.

The language model factor engine is used to smear the language model scores across the lexical tree. This operates in two ways depending on whether the factor is needed for a word start phone(s) or word internal phones. For word internal phones, LM factors can be computed as needed, provided that the LM scores are available in cache. This is because in general the storage requirements for the LM scores are less than the storage of the LM factors for all of the factors in the branch. However when a token is in a word end phone there may a significant number of word start phones e.g. up to $(\text{monophones})^2$ requiring a large amount of computation. To avoid this the LM engine will pre compute word start LM factors when it seems highly likely that a token will be passed to the word starts. These word start LM factors will be held in the LM factors cache. The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree.

The state level shown in figure 1 includes a phone instance network engine (PINE) array, a MHD (Mahalanobis distance) engine, an audio front end, a feature vector buffer for model adaptation, and compressed acoustic models. The state level is the lowest level, in which a network of phones is dynamically created.

The audio input for speech recognition is input to the front end unit of the state level. This audio input may be digital audio, or it may be analog audio which is converted to digital audio using an analog to digital converter. The audio input is divided into time frames, each time frame typically being of the order of 10ms.

For each audio input time frame, the audio signal is converted into a feature vector. This may be done by splitting the audio signal into spectral components. For example, in the present embodiment, the audio input signal is split into 12 frequency bands and one

energy band, and a feature vector is constructed using these 13 components, plus their first and second derivatives, a total of 39 components.

The feature vector is passed to the MHD (Mahalanobis distance) engine. This computes all possible state output likelihoods for the current frame as the state mean and covariance data is read from the acoustic model memory.

The likelihood of each state is determined by the distance between the feature vector and each state. To reduce the number of states, a technique known as "state tying" may be used. Multiple states are grouped together in 39D space, according to their proximity to one another. An average or central value from each group is used to represent the entire group. Thus, the effective number of states may be reduced from 40,000 to 8,000. The accuracy may be improved using a language model in addition to the word recognition lexical tree. A language model is based on the probability of sequential words. The above figure of 8,000 states gives best recognition results when used with a language model.

The MHD (Mahalanobis Distance) is a distance between two N-dimensional points, scaled by the statistical variation in each component. In embodiments of the invention, the Mahalanobis distance between the feature vector and each state is calculated, to determine similarity of the feature vector to each state.

The Feature Vector represents a point in an N-dimensional space. Different embodiments of the invention may use different values of N. Generally we see values of N in the range of 20 to 39, although values outside this range are not excluded.

The distance calculator stage of the recognition process computes a probability or likelihood that a feature vector corresponds to a particular state. Each state has an associated probability distribution in the N-dimensional space. The distance calculator computes the distance in the N-dimensional space from the Feature Vector to the probability distribution for each state. In some embodiments, the calculation is performed only when the search stage algorithms determine that the distance for that particular state is needed by the searchstage. In some embodiments, the distances for all

of the states are calculated and made available to the search stage, which allows pipelined operation. Embodiments that do the calculations on demand are preferable.

The Mahalanobis Distance (MHD) is extensively described in the literature. It may be calculated using the formula

MHD value = ln(P(x)) = -0.5[N*ln(2*pi)+ln|Cov|)] + (-0.5*[(x-u)'*inverse(Cov)*(x-u)])

where

N= number of dimensions in the space
x=feature vector, a point in the N dimensional space
u=mean vector
Cov=covariance matrix
|Cov|=determinant of covariance matrix
(a)' = transpose of matrix a

Different embodiments may use variations on this equation, for example the base for the logarithm is not always the natural base e that is used in the above example. The log of the probability is preferably calculated, rather than the probability, as it simplifies the implementation. Not all embodiments do this. In alternative embodiments, the likelihoods rather than probabilities are computed as it leads to simpler implementations of computations in the search stage.

The feature vector is used to calculate 8,000 MHD distances for each time frame, i.e. one distance for each of the 8,000 states. It is beneficial, although not essential, to perform the MHD calculations separately in a pre-processor, known as a MHD engine, because MHD calculation is a computationally intensive process.

Due to the 10ms frame length, a feature vector arrives at the MHD engine of the state level every 10ms. The MHD engine calculates the likelihood of each feature vector

corresponding to each of a number of states derived from an acoustic model. Each state is also a 39 dimensional vector, having the same spectral components as the feature vector. The MHD engine calculates the difference in 39-D space between the feature vector and each state to obtain the likelihood for that state. A parallel MHD or a single MHD arrangement may be used.

The feature vectors may be sent to a feature vector buffer for model adaptation, which determines whether the observed features fit well with the acoustic model, and may initiate an adaptation of the model if the fit is not sufficiently good.

Scores indicating the state likelihoods or probabilities are output from the MHD engine to the PINE array. It is preferable to use likelihoods rather than probabilities, because likelihoods can be added together to give a total likelihood. Probabilities must be multiplied, which can give rise to rounding errors. In this embodiment, eight separate pines are used, each of 4 kilobytes, thus storing 4000 phones per pine.

During recognition, the Phone Instance Network Engine (PINE) Array will contain a network of phone instances. Tokens representing search hypotheses in the form of a probability(likelihood) will exist in some of the HMM states of the phone instances. The tokens are propagated into all available states and updated with the corresponding state output probabilities. Maximum model, word end and beam pruning are applied to the tokens to ensure the search space is kept bounded. When a token is in the final state of a phone model and has avoided being pruned, the PINE makes a next phone request to the Phone level. The PINE array may receive a number of new phone models from the phone level which must be appended to the appropriate existing phone.

Figure 2 shows the data structures stored in the PINE array, and other main data structures used during the acoustic search. Each PINE includes a phone instance CAM and a phone instance RAM. The phone instance CAM stores the phone instance number (labelled "phone_no") which uniquely identifies each phone instance. The phone instance CAM also stores the state IDs of the three HMM states used to model the phone, and flags indicating the position of the phone. The "back" flag indicates if the phone is at the back of the wavefront. The "front" flag indicates if the phone is at the

front of the wavefront. The "empty" flag indicates if there are no tokens in any of the three states for that phone. The "end" flag indicates if the phone is at the end of a word.

The phone instance RAM stores the token scores for each of the three states corresponding to each phone. The st_dum field indicates a dummy state, which is used to represent a state prior to the first state of the first phone. The "tmp" field is used for passing a token from a previous phone in the lexical tree to the current phone. This will be explained in more detail later. The prev_phone_no field is for storing a pointer to the previous phone instance in the network. The wordlat_node_id identifies the position of the phone instance on the word lattice node, and the LM_factor stores the language model factor for that phone instance.

At the phone level a record of all phones on the wave front must be maintained. If a phone instance is on the wave front, the phone_no stored in the phone instance CAM will point to the details in the wave front phone CAM and RAM.

The wave front phone CAM and RAM keeps pointers into the grammar and lexical tree. When a token in a wave front phone reaches the last state, the PINE array makes a request to the phone level by passing back the phone_id. The phone_id is used as a key to determining the token's grammer_node_id and LT_node_id within the wave front CAM/RAM structure. If the token is at a word end the grammar model is checked. A grammar identifier points to the type of grammar.

At the word level, figure 2 shows an example word lattice structure, with silence having a word_id of 0, and a wordlat_node_id of 6. This is shown as connected to two word nodes – "which" has a word_id of 238 and a wordlat_node_id of 4, and "neither" has a word_id of 3556 and a wordlat_node_id of 5.

At the highest level, i.e. the grammar level, a simple grammar model is shown which will allow the phrase 'silence which house silence' or 'silence neither house silence' to be spoken. Whenever a word end is reached during the search, the grammar model must be consulted to determine the next possible word(s) or lexical tree(s). This allows the word lattice at the word level, consisting of a network of words, to be assembled.

Amazon / Zentian Limited
Exhibit 1002
Page 87

Whenever a word end is reached, the word is added to the lattice and the wordlat_node_id updated accordingly.

Figure 3 shows a single phone instance network engine (PINE). At system power up the state transition RAM will be loaded with the appropriate state transition vector but the CAM and RAM memories will be empty. When the system is initialised an initial set of phone model(s) will be instantiated with a token in the dummy state and a path score of zero. The phone models are received from the phone level.

When the audio data is processed, a stream of all possible state output probabilities will be received from the MHD engine. A counter that is synchronous to the stream of state output probabilities from the MHD produces a concurrent stream of state_ids. The job of the CAM is to perform a search to see if the state_id is instantiated anywhere in the CAM state_id fields. If it exists and the location doesn't have its empty flag set then this indicates that a token may need to be updated. The match signal from the CAM must be encoded into an address for the RAM. The corresponding RAM address can then be read. Any token that will end up in the identified state can then have it's path score updated and including any transition vector likelihood.

It is possible that a single state_id generates multiple CAM matches. To handle this situation the CAM and RAM are effectively decoupled by means of a match fifo and repeat count fifo. The state_id and corresponding state_output_probability are stored in the repeat count fifo. During a subsequent frame when there is no match, this state_id can be revisited, ignoring the matches already dealt with.

All searched phone instances corresponding to words in the current possible lexical tree are stored in the CAM-RAM. This includes phone instances at the beginning of the word, middle of the word and end of the word. These phone instances are connected by the "previous phone instance" field, which allows traceback through an entire word within the CAM-RAM.

A mechanism is thus required to pass tokens from one phone instance to the next. This mechanism is provided by the tmp field in the RAM.

The tmp field in the RAM holds tokens during token passing.

At the start of each time frame, the audio input is converted into a feature vector, the MHD values calculated, and the MHD values used to update the token scores. After update of all tokens, 2 passes are made through the CAM-RAM, firstly for pruning, and then for token passing. The pruning in this example removes tokens of below a threshold score, although other types of pruning are also possible. The pruning may involve removing a predetermined proportion of the tokens, chosen as those of lowest likelihood. A third possible method of pruning is removing all tokens with scores below a first predetermined value, but at word ends removing all tokens with scores below a second predetermined value, the second predetermined value being greater than the first predetermined value.

The token passing works as follows. Each phone instance points back to a previous phone, due to the prev_phone_no field stored in the RAM. During the token passing, the third token of a previous phone is grabbed and put into the tmp field of the current phone.

In other words, if a token is on the third state, and it is not pruned, it must enter the next state of the next phone. To do this, the tokens that lie in the third state are moved across to the tmp field.

For example, phone no. 345 needs to look at phone no. 320 to see if any tokens are in the third state. If yes, it takes these tokens and puts them in the tmp field.

One field in the CAM is labelled "front". This indicates whether the phone is on the wave front. If the phone is not on the wave front, then another phone will have picked up the token in the third state. However, if the phone is on the wave front, then no other phone will have picked up the token on the third state. Therefore, if there is a token in the third state, and the phone is on the wavefront, it is necessary to generate a next phone request.

The current phone number, word lattice identifier and word end flag are sent to the wave front manager, to request a next phone. If the word end flag indicates a word end, then the score is also passed to the wave front manager. The score can be passed in any case, but it is only required at a word end.

The phone level generates the next phone or phones. A new phone instance starts, and all of state1, state2 and state3 have scores pre-set to zero. The old score is provided in the tmp field.

The old phone is still within the wave, but it is unlikely to be at the back of the wave at this stage. When it gets to the back of the wave, it is deleted. Until it gets there, it remains in the CAM-RAM as before. The third token remains where it was, since tokens can either stay in the third state or move on.

Note that it is possible to end up with more that one token in each state. If this occurs, then the token with the poorer score is deleted. This does not affect the result of the lexical tree search, because both tokens on the same state have exactly the same word history.

Figure 4 shows a block diagram of the wave front manager. The wave front manager is responsible for maintaining a record of the set of phone models that exist at the front of the acoustic search space. This set of phone models is referred to as the "wave front." The key requirement for this block as to minimise the time between receiving a "next phone request" and generating the appropriate responses. The wave front manager receives the next phone requests from the PINE array and generates appropriate responses for the Phone Book, Word Engine, Path Merge Unit and Grammar engine The wave front manager also receives new wave front phone models from the Phone Book and updates the wave front.

The wave front manager receives the next phone request from the PINE array, via the phone request arbiter. The next phone request consists of a phone_no, word_end_flag and wordlat_node_id. If this phone is at the end of a word, the word_end_flag will be set.

The wave front manager uses the phone_no as input to the wave front phone CAM. As each phone_no is unique, it is guaranteed that only a single match will be generated. The wave front manager uses the CAM generated address to access the RAM. It then sends retrieved data from the RAM to the phone book, grammar engine, Path Merge Unit and word engine.

The wave front manager removes phone instance from CAM and RAM by writing 0's into CAM and RAM, and increments the empty count by one.

Sometime later, the wave front manager receives new phone instances from the phone book and place them in a fifo. The phone instances are taken from the fifo and written into the CAM and RAM at the next_empty_address. The empty_count is decremented by one. A match is run on the "occupy" flag, and the address is placed into next_empty_address register.

Both the phone level and the state level make use of a path merge unit to reduce the number of possible paths, by path merging. Figure 5 shows the use of link nodes to permit path merging. In the present embodiment, all word end phone instances are linked to word start triphones with a link node. During each frame the path merge unit looks for *word end* tokens that can be recombined. This is done by looking for word end phones that have the same word_id and word history. When they are found a link is added using the path merge unit in the PINE array.

Figure 6 shows a diagram of the architecture at the Phone level, showing the path merge unit. The path merge unit includes a path merge CAM and a path merge RAM. The path merge CAM stores word_id, word history for two words, and an "occupy" field. The path merge RAM stores a link_node_id. A CAM RAM occupancy manager keeps track of the empty addresses in the CAM and RAM. The path merge unit is only passed information from the wave front manager if the phone is a *word end phone*. For each frame, there is a check to see if there is an identical word with the same history, and if there is, the link_node_id is passed to the path merge unit. If there isn't an identical word with the same history, a new record is written into the Path merge CAM-RAM.

Figure 7 is a block diagram showing the Phone Book architecture. The Phone Book is tasked with generating the next phone models when given a pointer into the lexical tree. The key requirement for this block as to minimise the time between receiving a "lexical tree address" and generating the appropriate responses, in particular the next phone model instance(s).

The lexical tree is used to build the next triphones based on the current context.

Current triphone	Next triphone	comments
Left context = a Central context = b Right context = c	discarded Left context = b Central context = c **Right context = ?**	**next_leftmph_id stored in wave front manager** **current LT_node_id stored in wave front manager** **must be looked up. Large (< 45) for new word start**

The LT_node_id, next_left_mph_id and prev_phone_no are received at the phone generator from the wavefront manager. The LT_node_id is then used to address the lexical tree store and return the central monophone_id. One or more further accesses to memory are performed using the microcoded instructions, each memory access returning the right monophone_id.

A two stage lookup is accessed using the three monophone_ids (giving a triphone), the associated state_ids. The phone model is assembled, and returned to the PINE array. The wavefront manager is then updated by sending it relevant parts of the phone model.

The lexical tree store is a micro-coded ROM with both instruction and data fields. The lexical tree store holds a number of lexical trees. Each lexical tree may hold single words or multiple words. Each lexical tree comprises lexical tree elements, and each element has three fields. For now, the fields will be integers and will be converted to std_logic_vector later. The following is an example of the lexical tree store, with the LT data in word_id order.

```
ZERO   z iy r ow      19 8 12 11 0 0 0 0      sp 0
ONE    w ah n         18 2 10 0 0 0 0 0       sil 1
TWO    t uw           14 16 0 0 0 0 0 0       ah 2
THREE  th r iy        15 12 8 0 0 0 0 0       ay 3
```

```
FOUR   f ow r          6  11 12  0  0  0  0  0
FIVE   f ay v          6   3 17  0  0  0  0  0
SIX    s ih k s       13   7  9 13  0  0  0  0
SEVEN  s eh v ih n    13   4 17  7 10  0  0  0
EIGHT  ey t            5  14  0  0  0  0  0  0
NINE   n ay n         10   3 10  0  0  0  0  0
OH     ow             11   0  0  0  0  0  0  0
```

The following shows the LT data in MP (monophone) start order.

```
EIGHT  ey t            5  14  0  0  0  0  0  0
FIVE   f ay v          6   3 17  0  0  0  0  0
FOUR   f ow r          6  11 12  0  0  0  0  0
NINE   n ay n         10   3 10  0  0  0  0  0
OH     ow             11   0  0  0  0  0  0  0
SEVEN  s eh v ih n    13   4 17  7 10  0  0  0
SIX    s ih k s       13   7  9 13  0  0  0  0
TWO    t uw           14  16  0  0  0  0  0  0
THREE  th r iy        15  12  8  0  0  0  0  0
ONE    w ah n         18   2 10  0  0  0  0  0
ZERO   z iy r ow      19   8 12 11  0  0  0  0
```

```
eh 4
ey 5
f 6
ih 7
iy 8
k 9
n 10
ow 11
r 12
s 13
t 14
th 15
uw 16
v 17
w 18
z 19
```

The following is an example of the three fields of a lexical tree element. The first field is an instruction field, and the second and third fields are data fields.

Fields:	1st instr 2b	2nd data 6b	3rd data 6b	
	00	MP	MP	- raw data field.
	01	br	br	- branch address.
	10	br	br	- branch address.
	11	word_id		- double field.

Note that there might be no difference between "01" and "10"

To build the "ey" tree, we only have to consider "EIGHT", because this is the only word in the lexical tree which starts with "ey". EIGHT is represented as "ey t", with phone_nos 5 14 0 0 0 0 0 0. The first lexical tree element stores the raw data specifying the phone_nos, the second lexical tree element is an end field, and the subsequent lexical tree element is set to zero.

```
0  00  5 14   - raw data
1  11  8  0   - end field
2  00  0  0
```

To build the "f" tree, we have:

```
FIVE  f ay v      6  3 17  0  0  0  0  0
FOUR  f ow r      6  11 12  0  0  0  0  0

0  01  4  0  -- branch forward four places as well
1  00  6  3  -- as the implied other brach of next (one).
2  00  3 17
3  11  5  0  -- all addresses are relative offsets.
4  00  6 11
5  00 11 12
6  11  4  0
7  00  0  0
```

To build the "n" tree, we have:

```
NINE        n ay n        10  3 10  0  0  0  0  0

0  00 10  3
1  00  3 10
2  11  9  0
3  00  0  0
```

To build the "ow" tree, we have "ow" in both the first and second data fields, as no tree is going to be a single monophone long.

```
OH          ow            11  0  0  0  0  0  0  0

0  11 11  0
1  00  0  0
```

To build the "s" tree, we have:

```
SEVEN       s eh v ih n   13  4 17  7 10  0  0  0
SIX         s ih k s      13  7  9 13  0  0  0  0

0  01  6  0  -- branch forward six places.
1  00 13  4
2  00  4 17
3  00 17  7
4  00  7 10
5  11  7  0
6  00 13  7
7  00  7  9
8  00  9 13
9  11  6  0
10 00  0  0
```

To build the "t" tree, we have:

```
TWO   t u w       14 16 0 0 0 0 0 0

0 00 14 16
1 11 2 0
2 00 0 0
```

To build the "th" tree, we have:

```
THREE         th r i y     15 12 8 0 0 0 0 0

0 00 15 12
1 00 12 8
2 11 3 0
3 00 0 0
```

To build the "w" tree, we have:

```
ONE   w a h n      18 2 10 0 0 0 0 0

0 00 18 12
1 00 12 10
2 11 1 0
3 00 0 0
```

To build the "z" tree, we have:

```
ZERO  z i y r o w    19 8 12 11 0 0 0 0

0 00 19 8
1 00 8 12
2 00 12 11
3 11 0 0
4 00 0 0
```

Once all of the branches have been built, the next stage is to concatenate the tree. It appears that four elements make a page, and each element is two bytes.

```
                                                    page
ey:     00 5 14  11 8 0  00 0 0  00 0 0            5
f:      01 4 0  00 6 3  00 3 17  11 5 0            6
        00 6 11  00 11 12  11 4 0  00 0 0          7
n:      00 10 3  00 3 10  11 9 0  00 0 0           8
ow:     11 11 0  00 0 0  00 0 0  00 0 0            9
s:      01 6 0  00 13 4  00 4 17  00 17 7          10
        00 7 10  11 7 0  00 13 7  00 7 9           11
        00 9 13  11 6 0  00 0 0  00 0 0            12
t:      00 14 16  11 2 0  00 0 0  00 0 0           13
th:     00 15 12  00 12 8  11 3 0  00 0 0          14
w:      00 18 12  00 12 10  11 1 0  00 0 0         15
z:      00 19 8  00 8 12  00 12 11  11 0 0         16
```

Then, the start_lookup header is defined. Each row (page)is 8 bytes and hence the header takes 3 rows. The end_lookup header is two rows. Therefore, the lexical tree store proper starts at the 6th (#5) page.

```
sp sil ah ay eh ey  f ih          - MP
 0  1  2  3  4  5  6  7          - MP_id
 3  0  0  0  5  0  6  0          - data stored.

iy  k  n ow  r  s  t th
 8  9 10 11 12 13 14 15
 0  0  8  9  0 10 13 14

uw  v  w  z  .  .  .  .
16 17 18 19  .  .  .  .
 0  0 15 16  0  0  0  0
```

The end_lookup header is defined. 20 is out of bounds.

```
ey  f  n ow  s  t th  w
 5  6 10 11 13 14 15 18

z  .  .  .  .  .  .  .
19 20 20 20 20 20 20 20
```

To put the above information all together:

START LOOKUP	sp:ih	3 0 0 0	5 0 6 0 0
	iy:th	0 0 8 9	0 10 13 14 1
	uw: v	0 0 15 16	0 0 0 0 2
END LOOKUP	ey:w	5 6 10 11	13 14 15 18 3
	z	19 20 20 20	20 20 20 20 4
LEXICAL TREE STORE	ey:	00 5 14	11 8 0 00 0 0 00 0 0 5
	f:	01 4 0	00 6 3 00 3 17 11 5 0 6
		00 6 11	00 11 12 11 4 0 00 0 0 7
	n:	00 10 3	00 3 10 11 9 0 00 0 0 8
	ow:	11 11 0	00 0 0 00 0 0 00 0 0 9
	s:	01 6 0	00 13 4 00 4 17 00 17 7 10
		00 7 10	11 7 0 00 13 7 00 7 9 11
		00 9 13	11 6 0 00 0 0 00 0 0 12
	t:	00 14 16	11 2 0 00 0 0 00 0 0 13
	th:	00 15 12	00 12 8 11 3 0 00 0 0 14
	w:	00 18 12	00 12 10 11 1 0 00 0 0 15
	z:	00 19 8	00 8 12 00 12 11 11 0 0 16

```
 3 0   0 0   5 0   6 0
 0 0   8 9   0 10  13 14
 0 0   15 16  0 0   0 0
```

```
  5 6    10 11    13 14    15 18
 19 20    20 20    20 20    20 20
00 5 14  11 8 0  00 0 0  00 0 0
01 4 0  00 6 3  00 3 17  11 5 0
00 6 11  00 11 12  11 4 0  00 0 0
00 10 3  00 3 10  11 9 0  00 0 0
11 11 0  00 0 0  00 0 0  00 0 0
01 6 0  00 13 4  00 4 17  00 17 7
00 7 10  11 7 0  00 13 7  00 7 9
00 9 13  11 6 0  00 0 0  00 0 0
00 14 16  11 2 0  00 0 0  00 0 0
00 15 12  00 12 8  11 3 0  00 0 0
00 18 12  00 12 10  11 1 0  00 0 0
00 19 8  00 8 12  00 12 11  11 0 0
```

This information is converted to hexidecimal. The first five are byte fields.
The rest are:
2bit - uCode.
6bit - data.
2bit - null.
6bit - data

```
03 00    00 00    05 00    06 00
00 00    08 09    00 0A    0D 0E
00 00    0F 10    00 00    00 00
05 06    0A 0B    0D 0E    0F 12
13 14    14 14    14 14    14 14
05 0E    C8 00    00 00    00 00
84 00    06 03    03 11    C5 00
06 0B    0B 0C    C4 00    00 00
0A 03    03 0A    C9 09    00 00
CB 00    00 00    00 00    00 00
86 00    0D 04    04 11    11 07
07 0A    C7 00    0D 07    07 09
00 0D    C6 00    00 00    00 00
0E 10    C2 00    00 00    00 00
0F 0C    0C 08    C3 00    00 00
12 0C    0C 0A    C1 00    00 00
13 08    08 0C    0C 0B    C0 00
```

The above is the micro code for the digit lexical tree. It allows branching, and it holds the word_id. It knows which lexical trees it contains and where they are housed. It also stores the possible end monophone list.

Figure 8 shows a further example of the Lexical Tree RAM, giving examples of different types of branching in the lexical tree, and how these are represented in the RAM.

Figure 9 is a block diagram showing the data structure that is supported for the word link record, to enable word n-best decoding. The word engine maintains a record of the words that have been currently searched by the PINE array. It is used to determine the word history and allow trace back during or at the end of search. The word history is needed to determine the language model score and whether particular paths can be recombined or merged.

Figure 10 shows a block diagram of the word engine. This is designed to allow fast access to the word history (thus allowing rapid phone creation at the phone level), and to allow reasonably fast trace back. The Word Engine is primarily a memory store that fills incrementally. The decoupling of the wordlat_node_id and the physical address will allow the possibility of being able to keep the WLRs for a given word in contiguous addresses, thus avoiding the need for the pointer field.

The language model interface is designed to allow the Phone Book to quickly obtain the appropriate LM factor for the phone model that it is currently constructing. Based on the phones in the wave front, all possible next LM factors can be determined. This will be done in software together with the language model factor engine. The ARM 9 will maintain its own record of the wavefront using the same next phone requests received by the wave front manager and its own copy of the lexical tree.

Figure 11 shows the architecture of the LM interface. This allows the Phone book to quickly access the LM factor for the phone that is currently being generated. Prior to the phone generation phase LM_factors are computed for all possible phones that may be generated in the next frame.

The grammar engine is designed to give the Phone Book a quick response to requests for the next grammar node(s). In the case of a task grammar where the lexicon may be limited to a few words(e.g. yes, no), it may be desirable to use the same lexical tree. In this case it will be necessary for the grammar, to provide lexical tree pointers such that the phone generator can jump to the appropriate part of the lexical tree.

For a speech recognition system it is also necessary to specify the grammar. Again we will use HTK's Standard Lattice Format(SLF) to specify the grammar. In practice it will be convenient to use HTK's HParse format to specify the grammar using a high level grammar notation. This can then be converted into using the HParse tool. Consider having to specify a grammar for a digit recognition task. The HParse format for this is as follows:

$digit = one | two | three | four | five | six | seven | eight | nine | zero
$digit_sp = $digit sp
$noise = lipsmack | breath | sil
([sil] < $digit_sp | $noise > [sil])

where | denotes alternatives, < > denotes one or more repetitions, { } denotes zero or more repetitions, [] denotes that item is optional, and () contains the grammar.

Figure 12 shows the network that is defined by this grammar. The network starts and ends with a "sil" phone, representing silence. Between the start and end, a number of possible routes correspond to further silence, a breath, a lipsmack sound, or a detected word.

Figure 13 shows the grammar engine architecture. When a next phone request arrives, the next grammar node(s) must be retrieved from memory. In the case of a task grammar, the phone book should be given the lexical tree memory addresses to allow the phone to be determined.

Figure 14 shows a single frame of the Audio Codec (AC) link protocol used in some embodiments of the invention. The Advanced Audio CODEC Interface (AACI) is an ARM Primecell that provides an interface to off-chip AC97 audio codecs. AC97 is Intel's audio codec specification, initially created for the PC market. The AACI is an AMBA slave block that connects to the Advanced Peripheral bus (APB). The AACI connects to the off-chip codec using the AC-link protocol.

The AC-link protocol is a bi-directional, fixed clock rate, serial digital stream. AC-link handles multiple input and output PCM audio streams as well as register control accesses. It employs a time division multiplexed (TDM) scheme that divides each audio frame into 12 outgoing and 12 incoming data streams each with up to 20 bit sample resolution. The tag slots are used to indicate which of the subsequent slots contain valid data. Add and dat are the control register address and data slots for read and writes. The adc slots contain the outputs from the analog to digital converters. Assuming that the ADC sample rate is set to 16 kHz then the AC-link controller will receive 16 bits of data every third codec frame.

The AACI has 4 channels that can be operated simultaneously. They are accessed through DMA.

Figure 15 shows the system timing. Each frame is divided into three sections:

- Token update (MHD wash)
- First pass - inter phone token passing
- Second pass – garbage collection, max model pruning

The horizontal direction of the figure indicates elapsing time, and various subcircuits and processes are indicated along the vertical direction of the figure. The top part of the diagram shows frame count, and push to dictate button, and the lower part of the diagram shows various subcircuits and processes, within a single time frame, whilst the push to dictate button is engaged. These subcircuits and processes are language model interface, word engine, phone book, PINE array, MHD engine, mean & covar reg, and feature vector reg.

At the start of the time frame, the feature vector is generated, and the MHD values are calculated. The MHD values are then passed to the PINE array, which engages in beam pruning and token score updating. The phone book receives next phone requests and returns next phone responses. The word engine adds word link records, and the language model interface supplies language model factors.

In the second part of the frame, pruning thresholds are calculated for the PINE array, for the next frame, and the PINE array engages in token passing. The language model interface transfers language model factors for the next frame.

Whilst the "push to dictate" button remains activated, the above process repeats in a sequence of frames.

The embodiments of the invention which are now described relate to a distance engine for calculation of Mahalanobis Distances.

Figure 16 shows a block diagram of a speech recognition system according to an embodiment of the invention. The system includes a microphone 100, which outputs an audio signal to a combined AGC (automatic gain control) and ADC (analogue to digital converter) 101. In alternative embodiments, the AGC and ADC may be separate. In this embodiment, the incoming data comes from a microphone, and is then converted to digital format. However, it is also possible that the digital audio data may be received from a network, or may be read from storage means such as a memory, an optical disk, or a hard disk drive.

The digital audio data output from the ADC is input to an audio data buffer 102, and may be held here until required. The data is then output from the audio data buffer 102 to a front end circuit 103. The incoming audio data is divided into frames, as described above. In cases where the system is extremely busy, it is possible to drop some audio frames, to allow the system to catch up with any backlog.

In the audio front end circuit 103, the audio is split into spectral components to generate a feature vector, as described above. For example, the feature vector may have a number of frequency bands, and an energy band, plus their first and second derivatives. The feature vector is then output from the front end 103 to the distance calculation engine 104.

The distance calculation engine 104 converts feature vectors into Mahalanobis distances . The acoustic model may be chosen from a plurality of acoustic models, e.g. according

to the language used, and may be stored in a storage means 105 such as a memory, a disk drive, etc. The distances calculated by the distance calculation engine are then transferred to the search stage 106 of the speech processing circuit, which uses models such as one or more word models and/or language models to generate and output recognised text. Such word and language models may be stored in a separate storage means 107.

In one embodiment, the distance calculation engine uses Mahalanobis Distance (MHD) calculations to determine distance values for output to the search stage 106. In further embodiments, different types of distance calculations may be used, e.g as described above.

In some embodiments of the invention, the distance calculation engine is designed as a speech accelerator, to operate as a loosely bound co-processor for a CPU running speech recognition software. This has the advantage of reducing computational load on the CPU, and reducing memory bandwidth load for the CPU.

In some embodiments, the accelerator interface is memory-mapped. Other types of interface are also possible to support different system configurations, for example removable accelerator cards with SDIO (secure digital input/output), CardBus, or USB interfaces.

The speech recognition apparatus may also operate with a CPU and DSP (digital signal processor), where the DSP implements the "front-end" signal processing to produce a feature vector, and the CPU implements the search stage, reading the distance results from the distance calculation engine.

The system may be implemented in any combination of hardware or software.

In some embodiments of the invention the search stage 106 provides the distance calculation engine 104 with an active state list 1XX that indicates the subset of states for which the search stage requires distances to be calculated by the distance calculation

engine 104. This is an optimization that may reduce computation time and/or power consumption.

In preferred embodiments of the invention, the front end 103, distance calculation engine 104 and the search stage 106 operate in a pipelined manner. When operating in a pipelined manner it is unlikely that the search stage 106 will be able to provide the active state list 1XX early enough for the distance calculation engine 104 to implement the optimization of computing only the distances that will be required by the search stage 106. The distance calculation circuit 104 may calculate the MHD values for every state in the lexicon, per frame, whether it is subsequently required by the search stage or not. This allows the accelerator and software system to operate in a concurrent pipelined manner, which maximizes the system throughput.

The distance calculation engine 104 may have a plurality of memory areas for storing feature vectors, to allow a feature vector for one frame to be transferred to one memory area in the distance calculation engine 104, while another feature vector for another frame is being read from a second memory area, for calculation of a distance. The memory areas used for reading and writing may alternate with subsequent frames.

Figure 17 shows a block diagram of a distance calculation accelerator 200, used in an embodiment of the invention. A feature vector is received at input 212 of the accelerator 200. The feature vector is loaded into feature vector (FV) memory A 208 via a bus 211. In reality, there may be more than a single bus, but for simplicity, only one bus 211 is shown in the figure.

The registers 203 are discussed further at the end of this description.The registers 203 may include, for example, an accelerator status register, for indicating whether the acoustic model data is correctly stored in memory, as described in more detail below.

The FV registers 209 hold the feature vector whose distances are currently bring computed by the distance calculation engine 204. If the distance calculation engine 204 is implemented in software, the FV registers might be implemented as memory attached to the processor running the software. If the distance calculation engine 204 is

implemented in hardware the FV registers might be implemented as logic or as a memory block. The FV registers 209 are loaded from the FV memory 208 when the Distance Calculation Engine 204 has completed distance calculations for the current feature vector and a subsequent feature vector is available in the FV memory 208. In some embodiments there is an optional FIFO associated with FV registers 209 so that the FIFO is loaded from the FV memory 208 when there is a subsequent feature vector in the FV memory 208 and there is a space in the FIFO. In these embodiments the FV Registers 209 are loaded from the associated optional FIFO 209 when the Distance Calculation Engine 204 has completed calculations for the current feature vector and there is a subsequent feature vector in the associated optional FIFO 209. The presence of the FIFO 209 provides for a looser coupling between the Front End 103 and the accelerator 200 which improves performance in some system configurations.

Each feature vector is transferred to a distance calculation engine circuit 204, to obtain distances for each state of the acoustic model. The acoustic model is stored in an acoustic model memory 205, which is connected to the bus 211, for control of reading/writing to the memory. In some embodiments, this also allows acoustic models to be changed or updated via the bus. The acoustic model may be stored in compressed form in the acoustic model memory 205. A decompressor 206 is provided on the data output of the acoustic model memory. The decompressor 206 can decompress the results as they are needed, and thus allows a much smaller size of acoustic memory to be used, than with uncompressed data. From the decompressor 206, the decompressed acoustic model states are sent to the distance calculation engine 204.

The distance calculation engine 204 converts feature vectors and acoustic model states to distances. The accelerator 200 includes two separate results memories for storing these calculated distances, labelled as "result memory A" 201 and "result memory B" 202.

The control, sequencing, and address generation 210 is logic, state machines, and/or software programs that provide overall control of the other elements of the accelerator, taking inputs from the registers 203, and all the other elements and interfaces to determine when to start and stop the Distance Calculation Engine 204. The control 210

also produces addresses and control signals to control the reading and writing of the memories 201, 202, 205 and 208, to control the flow of data between the various elements of the accelerator 200, the operation of the Decompressor 206, CRC check 207 and Distance Calculation Engine 204, and to control the setting or clearing or status bits in the Registers 203 and to control signalling to the front end 103 and search stage 106. The control 210 is implemented using any of the widely known and used methods of controlling and sequencing in digital systems including logic gates, state machines, and embedded software systems. The control 210 might be a centralized control, or it might be distributed in some or all of the other elements of the accelerator 200, or some combination of these.

Each time a feature vector is loaded into the accelerator, the accelerator computes the distances for all states for that feature vector, and stores the results alternately in the A or B Results Memory. The results are written to one of the two memories 201, 202, while the other memory is available for reading. Thus, while the distance calculation engine 204 is storing the distances calculated for one audio frame, the other result memory may make available the distances calculated for an earlier audio frame, to the search stage of the speech recognition system. For non-real time processing, "earlier" audio frame may refer to an audio frame that has been processed earlier, rather than being limited to an audio frame that corresponds to speech at an earlier time.

Thus, the search stage software reads results alternatively from the A or B result memory, while the accelerator writes results into the other result memory, allowing concurrent pipelined operation of the accelerator and the CPU [and the DSP if present].

In some implementations the A and B results memories may be exposed to the software interface as separate memories appearing at different locations in the CPU and/or DSP memory map. Another implementation is to map each memory alternatively to the same range of addresses in the CPU and/or DSP memory map, so that the software always accesses results at the same location, and the memory containing the set of results the software currently needs to access is the one mapped there.

The accelerator may autonomously load and decompress an acoustic model, which is stored in a compressed format in the acoustic model memory 205. The Acoustic Model may be loaded into the Acoustic Model Memory by software running on the CPU prior to the first use of the Accelerator. The Acoustic Model Memory may be an on-chip memory, to maximize bandwidth and minimize power consumption of reading in the entire Acoustic Model every frame, which means 50 to 100 times per second, depending on the system design.

The Acoustic Model Memory may be a dedicated memory attached to, or incorporated within, the accelerator. Alternatively the Acoustic Model Memory may be a shared resource that is used by other system functions or applications when speech recognition is not running. To allow detection of situations where the acoustic model has been overwritten by another application, a pre-calculated CRC (cyclic redundancy check) signature may be stored with the acoustic model. Each time the acoustic model is read into the accelerator, a CRC may be computed and compared with the signature. If the comparison fails, an error bit is set in the accelerator status register.

Some embodiments of the invention relate to speech recognition systems comprising separate front end 103 and search stage 106 processors. These separate processors perform the front-end and search stages of the recognition process, with the accelerator 200 performing the distance calculations. Typically the front-end processing would be performed on a DSP and search stage would be performed on a CPU, although this description holds for any type of processors, even dedicated hardware implementations. As shown in figure 17, the front end 103 supplies the distance accelerator with feature vectors, and the search stage 106 uses distances calculated by the accelerator.

The accelerator is designed to operate in a concurrent pipelined manner with the other processors, allowing a high level of parallelism in the system.

Figure 18 shows how this pipelined operation works in an embodiment of the invention with ideal data flow, i.e. where each of the three recognition stages require exactly the same time to process each data frame, and the processors are never delayed, stalled, or diverted to other tasks.

The time axis is divided into equal "pipeline step times". If the step time is the same as a frame time then the system is processing audio in real time. If the step time is short than a frame time, the system is operating faster than real time, which may be valuable for batch processing of audio data, or catching up after some type of delay. If the step time is loner than a frame time, the system is operating slower than real time, which may be acceptable in some applications, for example where the person dictating pauses occasionally allowing the system to "catch up".

Once the pipeline has filled (which takes two step times), the DSP, the Accelerator, and the CPU are all operating in parallel. In this idealized example, the total time available for processing a frame is three step times – one step time in the DSP, one step time in the accelerator, and one step time in the search stage. There is a latency of three step times from the time the audio data enters the front end to the completion of the search stage for that frame, and the processing of one frame finishes each step time. If the system were not pipelined, then the processing of all stages for one frame would have to complete in one step time in order to achieve the same overall throughput of frames per second. Thus pipelining in this way can achieve three times the throughput with the same processing elements, compared to a non-pipelined system. A non-pipelined system would be one in which the processing of frame $n+1$ would not start until all stages of the processing of frame n was already complete.

If, for example, in a system running in real time with a frame time and step time of $1/100^{th}$ of a second, which is 10ms, each of the three processing elements has the entire 10ms available to complete the processing of one stage of the recognition task, giving 30ms of processing time overall for each frame. If the system were not pipelined, real time operation would require that all three processing elements would be complete in 10ms, allowing only 3.3ms for each of the three processing elements.

At the end of each pipeline step time, the DSP provides a feature vector to the Accelerator, and the accelerator provides a set of distance results to the CPU. When the CPU is performing the search stage for frame n, the Accelerator is computing the distances for frame $n+1$, and the DSP is performing the front-end stage for frame $n+2$.

During each pipeline step, the front end processing consumes one frame of digital audio data (although this is not indicated explicitly on the diagram).

Real world systems are more complex that the idealized system in many ways. The processing time for a frame is often highly variable and data dependent, especially in the search stage, and the processing time for each processing stage is likely to be quite different. The DSP and CPU are likely from time to time to be diverted to run other software tasks, either to service interrupts, or under control of an operating system scheduling regime. Communications between the processing elements may be stalled due to competing bus traffic, for example for DMA transfers.

Recognition software may implement "frame dropping" and other advanced techniques, and as a result not all frames are processed through all of the recognition stages. Processing elements cannot respond instantly to the availability of data from the preceding processing stage, resulting in delays.

Finite buffer space for information storage between stages means that an earlier stage may be stalled waiting for a space to store its output data when following processing stages are "falling behind".

The accelerator has been carefully designed to mitigate these problems. Figure 19 shows a pipelined system with real world complications added. The following text describes the interactions shown in the diagram. The time for the processing of various stages has been varied to show difference examples of buffering of data between stages and stalling of processing elements. The numbering corresponds to the numbered bubbles on the diagram. The text also makes reference to memories used to pass information into and out of the accelerator, which are shown in figure 17.

The front end has computed the feature vector for the first frame, FV1. It is loaded into the accelerator, causing the accelerator to commence computation of the distances for the frame.

The feature vector for the second frame, FV2, is passed to the accelerator. The accelerator is still busy computing the distances for the first frame, so this feature vector is held in the accelerators FV memory.

The front end has computed the feature vector for the third frame, FV3, but it cannot be delivered to the accelerator as the FV memory is still full, holding FV2. The time period labeled "stall delivering FV3" is the period between FV3 being computed and when it can first be delivered. Of course the DSP could be diverted to some other useful task, or it could buffer FV3 in its own memory and commence computation of FV4.

The arrow labeled "FV3 earliest" shows the earliest time that the feature vector FV3 can be delivered to the accelerator, which is immediately after the accelerator loads FV2 into its internal registers and frees the memory.

The arrow labeled "FV3 latest" shows the latest time that the feature vector FV3 can be delivered to the accelerator without causing the accelerator to stall for lack of a feature vector to work on. This example shows that there is a wide tolerance in the system to when FV3 can be delivered without affecting system performance – an entire accelerator frame processing time. The accelerator provides an interrupt and a status bit that can be polled to indicate the state of the FV memory. Regardless of which control method the DSP uses, the system tolerates long latency without reducing performance.

The accelerator signals to the search stage that the distances for frame 1 are available in result memory A. The search stage commences processing for frame 1, reading distances from result memory A.

The search stage has finished its processing for frame 1 before the accelerator has completed computing the distances for frame 2, so the search stage stalls waiting for the accelerator. Of course the CPU could be diverted to other useful tasks, including any search stage tasks that can be completed before distances are available. Again, an interrupt and status bits are provided by the accelerator to control this interaction between the accelerator and the CPU. .

The accelerator completes the distance calculation for frame 2, and signals to the CPU that it can commence reading the distances from result memory B.

The accelerator has completed the computation of distances for frame 3 before the search stage has finished processing frame 2. As soon as the search stage signals to the accelerator that it has completed frame 2 by releasing memory B, the accelerator signals that the distances for frame 3 are available, allowing the search stage to immediately move on to processing stage 3.

The accelerator is stalled waiting for feature vector FV4, due to the long processing time in the front end. In real systems this may be caused by a number of factors, including the DSP being diverted to another task for a period of time, or simply due to the DSP performance being insufficient to keep up with the accelerator's throughput.

The accelerator restarts as soon as FV4 is delivered by the front end.

The search stage is stalled waiting for distances for frame 4, a follow on consequence of the late delivery of FV4 to the accelerator.

FV5 is delivered to the accelerator while it is still computing the distances for the previous frame, allowing it to move onto processing frame 5 without stalling when it completes frame 4.

The accelerator signals that the distances for frame 4 are available and the search stage commences processing frame 4.

FV6 is delivered to the accelerator while it is still computing the distances for frame 5.

The availability of distances for frame 5 is signaled by the accelerator to the CPU.

Even though the accelerator has already received FV6, it is stalled waiting for a free result memory it can write the distances into.

The search stage releases result memory B at the end of its processing of frame 4. This allows the accelerator to start computing distances for frame 6.

The accelerator completes computing distances for frame 6 and signals to the CPU that the results are ready.

Further embodiments of the invention relate to systems with a Front End and Search Stage on a single processor. This single processor performs the front-end and search stages of the recognition process, with the accelerator performing the distance calculations. This configuration is likely to be the most common configuration where the accelerator is added to an embedded system such as a PDA or mobile phone that has a high-performance embedded application processor (but no DSP) available to run speech recognition software.

In this configuration the accelerator still operates in a concurrent pipelined manner with the front-end and search stages; it is just that the front-end and search stages are running on a single processor.

CPUs in embedded systems are generally relatively simple compared to desktop PCs and servers. Embedded CPUs in mobile phones, PDAs, in-car navigation and telematics systems, and other consumer electronics devices are typically "simple uni-processors" by which we mean not multi-processors and not supporting hardware multi-threading support – i.e. they can only execute a single thread of instructions (although they can be diverted to a different thread under the control of interrupts, exceptions, or an operating system). For this discussion, except where explicitly stated otherwise, we are talking about embedded processors that are "simple uni-processors". We are also using the term "CPU" and "processor" interchangeably.

Figure 20 shows how the system can be pipelined with a single embedded processor working with the accelerator, in an idealized system. In this idealized system, the time taken for the processor to perform the front-end and search stage processing is identical

to the time the accelerator takes to perform the distance calculations, and the processor is never delayed, stalled, or diverted to other tasks.

The time axis is divided into equal "pipeline step" times. If the step time is the same as a frame time then the system is processing audio in real time. If the step time is short than a frame time, the system is operating faster than real time, which may be valuable for batch processing of audio data, or catching up after some type of delay. If the step time is loner than a frame time, the system is operating slower than real time, which may be acceptable in some applications, for example where the person dictating pauses occasionally allowing the system to "catch up".

The first two pipeline step times are used to fill the pipeline. By the third pipeline step time, the pipeline is full, and all three stages of the recognition process are processed every pipeline step time, with each stage working on the processing of a different data frame.

During the first pipeline step time, the CPU completes the front-end processing for frame 1, and the feature vector passed to the accelerator so that distance calculations can start in the accelerator.

During the second pipeline step time, the front-end processing for frame 2 is completed by the CPU, in parallel with the accelerator computing the distances for frame 1.

During the third pipeline step time, the CPU runs the search stage for frame 1 reading distance results from the accelerator, and then runs the front-end processing for frame 3. The accelerator runs in parallel computing the distances for frame 2 using the feature vectors computed in the previous pipeline step.

Each subsequent pipeline step time is similar to the third pipeline step, with the frame numbers incremented. So for pipeline step $n = 4, 5, 6$, etc. the CPU runs the search stage for frame $n-2$ and then runs the front-end for frame n. In parallel the accelerator computes the distances for frame $n-1$.

Note that with this situation the production of feature vectors and the consumptions of results run in lock step - one feature vector is produced every time the search stage consumes one frame of distance results. This considerably reduces the need for elastic buffering between the distance accelerator and the software system.

Again, real world systems are more complex than the idealized system in many ways. For example, the processing time for a frame is often highly variable and data dependent, especially in the search stage, and the processing time for each processing stage is likely to be quite different. The CPU is likely from time to time to be diverted to run other software tasks, either to service interrupts, or under control of an operating system scheduling régime. Communications between the CPU and accelerator may be stalled due to competing bus traffic, for example for DMA transfers.

Recognition software may implement "frame dropping" and other advanced techniques, and as a result not all frames are processed through all of the recognition stages. Processing elements cannot respond instantly to the availability of data from the preceding processing stage, resulting in delays

Finite buffer space for information storage between stages means that an earlier stage may be stalled waiting for a space to store its output data when following processing stages are "falling behind".

The accelerator has been carefully designed to mitigate these problems. Figure 21 shows a pipelined system with real world complications added. The following text describes the interactions shown in the diagram. The time for the processing of various stages has been varied to show difference examples of buffering of data between stages and stalling of processing elements. The numbering corresponds to the numbered bubbles on the diagram. The text also makes reference to memories used to pass information into and out of the accelerator, which are shown in figure 21.

At the end of front-end processing for frame 1, the CPU passes the feature vectors for frame 1 to the accelerator, allowing distance calculations to start.

To fill the pipeline, the CPU immediately performs the front-end processing for frame 2 and passes the feature vectors to the accelerator. They are stored in the accelerator so that the accelerator can load them into its internal engine for processing as soon as the distance calculations for frame 1 are complete.

The CPU is now ready to start the search stage for frame 1, however the distance computation in the accelerator has not completed. The box labeled "Prep search stage 1" indicates a period where the CPU could do any preparatory search stage processing that does not require distances. Alternatively the CPU could be diverted to another task, or it can stall waiting for the distance results.

The accelerator signals that the distance results for frame 1 are now available. The search stage can commence on the CPU, reading distance results from the accelerator's Result Memory A. The accelerator automatically starts processing the feature vector for frame 2 that was delivered earlier.

The accelerator signals to the CPU that it has completed the distance calculations for frame 2. The CPU is still busy running the search stage for frame 1, and has still to run the front end for frame 3, so the completion signal is ignored until the CPU has need of it, when it starts running the search stage for frame 2. As the accelerator has not received the feature vector for frame 3, it has no task it can perform and so it stalls.

The CPU delivers the feature vector for frame 3, restarting the accelerator, which writes results into Result Memory A. Note that Result Memory A was previously "released" at the end of the processing of search stage 1, informing the accelerator that it can overwrite the old results held in that memory with new results. The CPU then runs the search stage for frame 2, reading distance results from the accelerator's Result Memory B.

The accelerator signals to the CPU that the distance results for frame 3 are complete in Result Memory A. Again, the CPU ignores this signal until it is ready to commence the search stage for frame 3, and the accelerator stalls waiting for the next feature vector.

The CPU delivers the feature vector for frame 4, restarting the accelerator. It them starts search stage 3 reading distances from the accelerator's result memory A.

Search stage 3 and front end 5 were completed quickly, so that the feature vector for frame 5 is delivered to the accelerator while it is still running. The CPU performs all the preparatory work it can for the search stage, and then stalls waiting for distances to be available.

The accelerator signals that the distances for frame 4 are available. The CPU starts search stage 4, reading the distances from the accelerator's result memory B. The accelerator automatically starts computing distances for frame 5 using the feature vector delivered earlier.

Note that with the pipelining scheme just presented which alternates front-end and search stage processing, the accelerator can never stall waiting for a result memory to be released because the feature vector the accelerator needs as an input is not produced until after the memory has been cleared. The CPU task schedule could be described with the shorthand F1 F2 S1 F3 S2 F4 S3 F5 S4.

It can seen readily from figure 21 and the accompanying description above, that with this CPU task schedule, that the operation of the CPU and accelerator are tightly coupled and whenever the processing time for a pipeline step on the CPU is unequal to the processing time on the accelerator, then it quickly results in one of them stalling.

In a system where either the CPU or the accelerator is always (or at least in most instances) slower than the other in completing its processing task, stalling is inevitable and throughput may not be improved by adding additional buffering (e.g. additional result memories or FIFO depth on the feature vector interface). [Unless there are other difficulties to mitigate such as very low-bandwidth buses linking the CPU to the accelerator, or competing bus traffic such as DMA transfers that can significantly reduce performance.

If the processing time for a pipeline step is highly variable on either the CPU or Accelerator, such that there are regular changes in which one of them takes the longer time, then significant enhancements in throughput may be obtained by adding additional buffering between them. This may have the disadvantage of adding additional latency through the system. This buffering may be implemented by adding additional depth to the Feature Vector FIFO, and/or adding additional Result Memories. Alternatively it may be by implementing elastic buffers in software.

If the accelerator has sufficient buffering on the feature vector interface, e.g. by including the optional FIFO shown in figure 21, then increasing the pipeline depth by computing extra front end frames up front will improve throughput at a cost of extra latency. This will result in a CPU task schedule of F1 F2 F3 S1 F4 S2 F5 S3 F6 S4 or F1 F2 F3 F4 S1 F5 S2 F6 S3 F7 S4, etc.

Of course it would also be possible to dynamically schedule whether the CPU should run the front-end or search stage code based on the availability, or not, of distance results and/or the space available to store more feature vectors and distance results. Given the fundamental relationship that one set of distance results is produced by the accelerator for every feature vector it consumes, and the restriction stated at the start of this section that we are discussing single-threaded CPUs, dynamic scheduling is likely to add little performance improvement while being complex to implement.

On a CPU supporting multi-threaded operation (which is a technique for improving performance in the presence of high memory system latency), or on a multi-processor, the highest system performance is likely to be achieved by running the front-end and search stages concurrently as separate threads, rather than alternating between the front-end and search stages. The system operation is then very similar to that described above, if each thread is thought of as a separate virtual processor. In some embodiments of the invention, elastic buffers may be used between stages, to accommodate varying time delays between the processing of frames at each stage. Elastic buffers in the interfaces between the three recognition stages may also significantly enhance performance of such a system.

Operation of the accelerator and the software system is not constrained to a rigid repetitive frame time. Instead the timing of the system is controlled dynamically by the availability of feature vectors into the accelerator, and the consumption of results out of the accelerator.

This allows the recognition process to fall behind real-time operation and to "catch up" again. It also allows the system to be robust in complex systems where interrupts, DMA bus traffic, memory caches, operating system task switching, and competing real-time software tasks all make guaranteeing rigid timing constraints impractical.

Elastic buffers at the major data transfer points between processing elements maximize system performance by allowing one element to continue useful processing whenever it can, regardless of whether other processing elements have been stalled or diverted to other tasks. Elastic buffers are useful for the following applications:

(1) For buffering raw audio data arriving in real time from the microphone(s) The data would typically be moved into the buffer by a DMA and/or high priority interrupt driven software process to ensure that hard real time constraints are met.

(2) For buffering feature vectors output from the front end. This is especially true if the front-end is implemented on a DSP or other processing element independent of the search stage.

(3) For buffering distance results from the accelerator. Again this is especially true if the front-end and search stages are implemented on independent processing elements. This buffering is also especially valuable if the interface between the search stage processor and the accelerator is very slow, has high latency, or is subject to competing bus traffic.

The Feature Vector and Results interfaces on the accelerator already provide a level of elasticity that should be sufficient in systems where the Front End and Search stages are running on the same CPU, and the accelerator has a high-bandwidth interface to the CPU with low level of competition from other bus traffic such as DMA.

Other embodiments may have deeper FIFOs. In one embodiment, the two result memories are an implementation of a FIFO where the implementation is exposed across the interface. It would be possible to have additional memories and use them in a round-robin fashion, or to use another implementation of a FIFO that hides the FIFO depth by only exposing one result memory at a time to the interface. An example of this would be to map each memory in turn to a memory range that is always used to read results, so that at any one time the "current" result memory is mapped to that location. This would hide the FIFO depth behind the interface so that the pipeline depth can be changed without changing the software (or hardware) that accesses the interface.

Slow or Conflicted Bus Interfaces may occur. In the system configurations described above, the search stage reads distances directly from the Accelerator's Result Memories. The result memories allow the search stage to randomly access the distance results, as they are needed. This will only give a high performance system if the interface or bus between the CPU and Accelerator provides the CPU with low latency, high bandwidth access to the Accelerator Result Memories.

In some systems it is not practical to provide such a low latency, high bandwidth interface between the CPU and the Accelerator. This may be for many reasons, including:

- The interface is a bus shared with other devices and/or processors that generate bus traffic that keeps the bus busy such that the CPU access to the accelerator is delayed waiting for the bus to be free. DMA transfers are one example of conflicting bus traffic.
- The interface is implemented as a serial bus, or narrow parallel bus, for cost or compatibility reasons, yielding low bandwidth and high latency
- The interface crosses an industry standard interface such as PCI, USB, SDIO, Cardbus, Firewire, PCIExpress etc. that introduces protocol overheads and/or latency as would be the case where the accelerator is in a plug-in peripheral to a system containing a CPU (or CPU and DSP).

In these situations it is possible to add another pipeline stage during which the result data is copied from the Accelerator into another memory (which we will call a "buffer memory") to which the CPU does have high bandwidth, low latency access. This copying can be performed by a DMA controller, or by software running as an interrupt handler, or a separate software thread on the CPU, or a software thread on another processor, or by other commonly known means. The device controlling the transfer may be on the CPU side of the interface or on the accelerator side of the interface, depending on the capabilities of the interface. This extra pipeline stage then allows an entire pipeline step time to copy the data from the accelerator to the other memory. The data copy can optionally be performed as a sequential burst, or a number of sequential bursts, to optimize transfers across the interface between the Accelerator and the buffer memory. The search stage, running on the CPU, can then randomly access the results in the buffer memory with low latency.

Similarly an extra pipeline stage can be added for transferring the Feature Vectors to the Accelerator, although given that the feature vectors are a much smaller object than the distance results, and that they are not accessed randomly across the interface but delivered once to the Accelerator for each frame, there are fewer situations where the performance benefit from this extra pipeline stage is as significant.

Figures 22 and 23 show idealized systems with the two extra pipeline stages for transfers added. The first shows a system with a CPU, DSP, and Accelerator with the two extra pipeline stages. The second shows a system with a CPU and DSP with the two pipeline stages added.

Other embodiments may add additional pipeline stages so that the distance computations are performed over two or more pipeline stages rather than being restricted to a single pipeline stage. This allows more time for the computation of the distance results for each frame, while maintaining the same throughput of distance results. This is similar in concept to the well-known pipelining techniques used to implement RISC microprocessors.

The Acoustic Models may occupy many megabytes of storage space. In embedded systems they will typically held in a non-volatile memory such as FLASH memory, or other types of non-volatile memory as is the appropriate currently economic technology. The Accelerator may read the acoustic model directly from the non-volatile memory, or more likely the acoustic model will be copied from the non-volatile memory into a higher bandwidth RAM memory of some type (e.g. SRAM or DRAM, SDRAM, etc.). The RAM memory may be part of the same integrated circuit as the Accelerator (i.e. "on chip"), or attached to the integrated circuit containing the Accelerator. ("off chip"). The having the RAM memory on-chip is particularly advantageous, as this configuration would give the lowest power consumption and highest bandwidth. An alternative solution would be for the acoustic model to be stored on a disk drive, or an another device accessible via a wired or wireless bus or network, and for the acoustic model to be loaded into the RAM memory from the disk drive, or over the wired or wireless bus or network.

On-chip RAM memory is however likely to come at a cost-premium to off-chip commodity memory devices. Many embedded systems face extreme cost pressures in the market place, especially high volume devices supplied by many manufacturers such as mobile phones. For a speech recognizer to be incorporated into such systems, the overall recognizer solution much be delivered at very low cost. There is therefore a significant economic benefit from reducing the cost of the RAM memory and the non-volatile memory. One way to reduce the cost of these memories is to compress the acoustic models and store the compressed model in the memory.

The Accelerator computes all of the distances associated with each Feature Vector that is delivered to it by the front end. This means that the accelerator must use the data from the entire Acoustic Model each time a new Feature Vector is presented to the Accelerator (with the exception of some specific optimizations that may determine that some data values need not be used in the distance computations). It is therefore possible to arrange for the accelerator to read through the acoustic data sequentially once for each Feature Vector, using each data value as it read from the Acoustic Model.

Reading through the entire Acoustic Model sequentially is ideally suited to placing a decompressor between the distance calculation engine and the RAM memory, because many compression schemes depend on the decompressor reading sequentially through the compressed data, and do not readily support random access into the compressed data.

Thus this configuration allows storing compressed data in the RAM memory (and the non-volatile memory), which can therefore be smaller and lower cost. There is no requirement for a memory that is large enough to hold the entire decompressed acoustic model, as the decompressed data is consumed as it is decompressed.

The decompression scheme could be any combination of:

- sign or zero extension or otherwise conversion of narrow or variable width data to a wider data format.
- sign or zero extension or otherwise conversion of narrow or variable width data to IEEE standard single or double precision floating point format
- codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data
- decompression of a Huffman or Lempel-Ziv compressed stream
- decompression of run length encoded data
- decompression of difference encoded data
- any well known compression scheme.

Additionally the model size can be reduced by using *subspace distribution clustering* and the decompressor and distance calculation engine designed to operate with such models.

The following description provides further information on the visible registers in some embodiments of the invention.

The register bank is designed to allow simple connection with an 8-bit data bus. Accesses to the register bank should be made with byte-read and byte-write accesses. To facilitate this, only the least significant 8 bits of the data bus are used.

The address decoder should respond so to byte accesses to byte 0 or byte 3 within the designated word. This way the same code can be used on an ARM processor regardless of whether the processor is configured for little endian or big endian operation.

REGBASE=Base address of register bank

Register 0 Product Version

8-bit Product version ID

This register value is changed when the functionality of the hardware charges in a way that changes the hardware/software interface.

Address = REGBASE (and REGBASE+3 for big endian)

Read only. Reads with value 0x01

Writes ignored

Register 1 Implementation Version

8-bit Implementation version ID

This register value is changed for each implementation to facilitate bug tracking and version control.

Address = REGBASE+4 (and REGBASE+4+3 for big endian)

Read only. Reads with value 0x01

Writes ignored

Register 2 Interrupt Enable

8-bit Interrupt Enable register

Address = REGBASE+4*2 (and REGBASE+4*2+3 for big endian)

Readable and writeable.

Value following reset is 0 for bits 0,1,2,3

Bit 0 I 1 = Enable FV memory empty interrupt on Int0 output.

Bit 1 J 1 = Enable Results complete interrupt on Int0 output.

Bit 2 K 1 = Enable FV memory empty interrupt on Int1 output.

Bit 3 L 1 = Enable Results complete interrupt on Int1 output.

Bit 4 Future use. Read value is not defined. Writes ignored.

Bit 5 Future use. Read value is not defined. Writes ignored.

Bit 6 Future use. Read value is not defined. Writes ignored.

Bit 7 Future use. Read value is not defined. Writes ignored.

Software should use a read-modify-write code sequence when setting this register so that future use bits are written with the value read from them.

Register 3 Status

Address = REGBASE+4*3 (and REGBASE+4*3+3 for big endian)

Read-only register. Writes are ignored.

Value following reset is 0 for bits 0,1,2,3

Bit 0 FVMB Feature Vector Memory is Busy

0 = OK to write new value to FV memory (FV memory is empty)

1 = Don't write new value to FV memory (FV memory is busy)

Bit 1 CRM Current Result Memory

0 = Result memory A

1 = Result memory B

This bit toggles every time the Release Result Memory (RRM) command is issued to the accelerator (register 3). It indicates which memory software should read results from, and which result memory the RRM command should release next time it is issued. Note: Software should also check that results are available in the indicated result memory by checking the value of the MA and MB bits.

Bit 2 MA 1=Results are available is result memory A

The MA bit is set to 1 when the compute engine completes writing all of the distances for a frame into result memory A. This value is set to zero when the Release Result Memory command is issued and CRM==0.

Bit 3 MB 1= Results are available is result memory B
The MB bit is set to 1 when the compute engine completes writing all of the distances for a frame into result memory B. This value is set to zero when the Release Result Memory command is issued and CRM==1.

Bit 4 Dfault 1=Accoustic data CRC fault. Indicates that the CRC value at the end of the acoustic data did not match the CRC value computed while the acoustic data was read in.

Bit 5 Future use. Value undefined.
Bit 6 Future use. Value undefined.
Bit 7 Future use. Value undefined.

Register 4 Command

Address = REGBASE+4*4 (and REGBASE+4*4+3 for big endian)
Write-only register.
Sofware should perform byte writes to this register with a single bit set to 1 and all other bits set to zero to issue a command to the accelerator.
Behaviour is not defined if more than one command is issued within a single write (i.e. if more than 1 bit is a 1).
Value on read is not defined.
Bit 0 LFV Load Feature Vector from the FV memory.
Tells the accelerator that a new feature vector has been placed into FV memory ready for use by the accelerator. The FV will be loaded into the compute engine when the compute engine completes its current task (or immediately if the compute engine is idle). Computation of distances for this FV will commence when the FV is loaded AND there is a released result memory. (At reset, both result memories are released). The results for the first, third, fifth etc. feature vectors loaded are written to Result Memory

A. The results for the second, fourth, sixth etc. feature vectors loaded are written to Result Memory B.

Bit 1 RRM Release Result memory. Indicates that the results in the current result memory can now be overwritten with new results. The current result memory is indicated by the CRM bit in the status register.
Bit 2 CFVI Clear the FV memory empty interrupt
Bit 3 CRI Clear the Results Complete interrupt
Bit 4 Future use – write 0
Bit 5 Future use – write 0
Bit 6 Future use – write 0
Bit 7 Future use – write 0

The present invention may be included in a mobile phone, or in another consumer electronics device, a PC, Laptop or tablet PC, in-car electronics system, or plug-in card for an electronic device.

The methods and apparatus according to the invention are not limited to embedded mobile systems, but may be used with any type of computing device, including PCs, workstations, laptop computers, notebook computers, handheld computing devices, dictation machines, PDAs (personal digital assistants), personal organisers, mobile phones, games consoles, video players, etc. Embedded computers according to the present invention may be embedded in a wide range of apparatus and machinery, such as electronic entertainment products, domestic appliances, factory machinery, vehicles, etc.

Embodiments of the present invention may be implemented as an IP core in a System on Chip (SOC) or a discrete device. Multiple chips or bare die may be connected to provide a larger vocabulary system.

Although the apparatus may include all or some of an input for accepting audio input, a circuit for digitising the audio input, a circuit for dividing the audio input into a plurality of time-frames, and a circuit for analysing the spectral components of each time frame

to characterise the audio in that frame, this is not essential. The apparatus may also include a circuit for calculating a score indicating the probability or likelihood that each audio frame corresponds to each of a plurality of different possible states, but again, this is not essential. Instead, the apparatus may include an input for receiving the pre-calculated scores.

In the described embodiments, n-gram language models are used. For example, the Bigram language model is based on outputting the probability that a given second word directly follows a given first word. However, it is possible to use alternative language models instead. It is possible to use a mixture of different models. For example, it is possible to run a word-dependent n-best decode applying a bigram language model, and the resulting word link record or word graph could then be rescored using a trigram language model.

The Viterbi beam search algorithm may be used in embodiments of the invention, to perform the search. A word level search may be performed. Dummy states may be used for the start and end of an utterance

The grammar level model may be used to add distance elements into the likelihood calculation. This is not essential, but tends to give much better results.

The HMM models described in the embodiments comprise self connections to each states and single connections to a neighbouring state. However, the present invention is not limited to such HMM models, and for example, HMM models with additional connections between states may also be used. Alternatively, embodiments of the invention may use alternative types of acoustic models, rather than HMM models.

The system may comprise a plurality of language models (e.g. English, French). A model of words may comprise actual words, phrases, silence, lipsmack sounds, and/or other sounds for recognition by the apparatus.

As previously discussed, the phone level has been found to be the best level for lexical tree searching, rather than the state level or the word level. However, this invention is not limited solely to phone level searching.

Pipelining may comprises processing data in different parts of a circuit at the same time, i.e. parallel processing.

In the following claims, the calculation means may comprise the distance calculation engine described above.

Further embodiments of the invention include the methods carried out in the accelerator, and the methods carried out in the speech recognition circuit.

While the invention has been described in terms of what are at present its preferred embodiments, it will be apparent to those skilled in the art that various changes can be made to the preferred embodiments without departing from the scope of the invention, which is defined by the claims.

CLAIMS:

1. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

a calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating circuit is implemented using a second processor, and wherein data is pipelined from the front end to the calculating circuit to the search stage.

2. A speech recognition circuit as claimed in claim 1, wherein the pipelining comprises alternating of front end and search stage processing on the first processor.

3. A speech recognition circuit as claimed in claim 1, comprising dynamic scheduling whether the first processor should run the front end or search stage code, based on availability or unavailablity of distance results and/or availability of space for storing more feature vectors and/or distance results.

4. A speech recognition circuit as claimed in claim 1, wherein the first processor supports multi-threaded operation, and runs the search stage and front ends as separate threads.

5. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

a buffer memory between the calculating circuit and the search stage, for receiving data passing from the calculating circuit to the search stage, wherein a processor in the search stage has higher bandwidth and/or lower latency access to the buffer compared to the bandwidth and/or latency of direct transfer between the calculating circuit and the search stage.

6. The speech recognition circuit of claim 5, wherein the data transfer from the calculating circuit to the buffer memory and/or from the buffer memory to the search stage is performed as one or more sequential bursts.

7. The speech recognition circuit of claim 5 or claim 6, wherein the data transfer to the buffer memory is performed in parallel with data transfer to the calculating circuit and/or in parallel with data transfer to the search stage.

8. The speech recognition circuit of any one of claims 5 to 7, further comprising a second buffer memory between the audio front end and the calculating circuit.

9. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

comprising an elastic buffer between at least one of the front end and calculating circuit, or the calculating circuit and search stage, and/or for buffering said audio signal.

10. An accelerator for calculating distances for a speech recognition circuit, the accelerator comprising:

calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

first and second storage circuit, each for storing calculated distances for at least one said audio time frame, and for making said stored distances available for use by another part of the speech recognition circuit;

control circuit for controlling read and write access to the first and second storage circuit, said control means being configured to allow writing to one said storage means while the other said storage means is available for reading, to allow first calculated distances for one audio time frame to be written to one said storage means while second calculated distances for an earlier audio time frame are made available for reading from the other said storage means.

11. A accelerator as claimed in claim 10, wherein the control means is configured to alternate which of the first and second storage means is available for reading, and which of the first and second storage means is available for writing, as each new feature vector is processed by the calculating circuit.

12. The accelerator of claim 10, wherein the distance is stored for each of a plurality of states.

13. The accelerator of any one of claims 10 to 12, wherein the first and second storage means are interfaced to a processor in the speech recognition circuit as different memory locations within the processor's memory map.

14. The accelerator of any one of claims 10 to 12, wherein the first and second storage means are interfaced to a processor in the speech recognition circuit as being mapped to a first fixed memory address when configured for reading, and/or to a second fixed memory address when configured for writing.

15. The accelerator of any one of claim 10 to 14, comprising releasing a result memory after the search stage has processed the contents, by informing the accelerator that it can overwrite the results in said memory.

16. The accelerator of any one of claims 10 to 15, comprising additional **results memories**.

17. The accelerator of any one of claims 10 to 14, comprising random access of the distance results in the **results memories** as they are needed by the search stage.

18. The accelerator of any one of claims 10 to 17, wherein the additional **results memories** are accessed in a round-robin fashion.

19. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

means for receiving or storing compressed data representing said acoustic model;

a decompressor for decompressing said compressed data for all states or selected states of the acoustic model, wherein the decompressed data is sent to the calculating means; and

output means for outputting calculated distances to another part of the speech recognition circuit.

20. The accelerator of claim 19, wherein the decompression scheme comprises one or more of the following:

sign or zero extension or otherwise conversion of narrow or variable width data to a wider data format;

sign or zero extension or otherwise conversion of narrow or variable width data to IEEE standard single or double precision floating point format;

codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data;

decompression of a Huffman or LPZ compressed stream;

decompression of run length encoded data;

decompression of difference encoded data; and

any well known compression scheme.

21. The accelerator of claim 19 or claim 20, wherein the decompressor is configured to operate with an acoustic model that uses subspace distribution clustering.

22. The accelerator of any one of claims 19 to 21, wherein acoustic states are decompressed one or more times for each feature vector processed by the accelerator.

23. A accelerator as claimed in claim 22, wherein the computed signature is a cyclic redundancy check (CRC) signature.

24. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

means for comparing a first version of a stored checksum of data representing said acoustic model and a second version of said stored checksum, wherein the second version is obtained from an updated measurement and calculation of the checksum; and

means for indicating an error status if the checksums do not match.

25. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

wherein said accelerator is configured to autonomously compute distances for every acoustic state defined by the acoustic model.

26. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; and

a memory for storing said acoustic model,

wherein the calculating means and the memory comprise part of the same physical circuit.

27. A accelerator according to claim 26, wherein said same physical circuit comprises a removable unit for removably connecting into the speech processing circuit.

28. A accelerator according to claim 26 or 27, wherein said memory is configured to be shared between the accelerator and another part of the speech recognition circuit.

29. A accelerator according to claim 28, further comprising means for generating a checksum or computed signature for the acoustic model data stored in the memory, and means for comparing checksums or computed signatures that have been calculated at different times, to indicate an error status if the checksums do not match.

30. A accelerator according to any one of claims 26 to 29, wherein the acoustic model memory is a RAM or flash memory.

31. The accelerator of any one of claims 1 to 30, wherein said distance comprises a Mahalanobis distance.

32. The accelerator of any one of the previous claims, being provided in the form of a separate physical circuit, device or package that is removably connectable to said speech recognition circuit.

33. A speech recognition circuit comprising the accelerator of any previous claim.

34. The speech recognition circuit of claim 33, comprising a shared processor for the front end and search stages.

35. The speech recognition circuit of claim 33, comprising a digital signal processor for the front end, and a processor for the search stage.

36. The speech recognition circuit of any one of the previous claims, comprising control means adapted to implement frame dropping, to discard one or more audio time frames.

37. The speech recognition circuit of any one of the previous claims, wherein the feature vector comprises a plurality of spectral components of an audio signal for a predetermined time frame.

38. The speech recognition circuit of any one of the previous claims, wherein the processor is configured to divert to another task if the data flow stalls.

39. The speech recognition circuit of any one of the previous claims, wherein the speech accelerator has an interrupt and status bit that can be polled to indicate the state of the feature vector memory

40. The speech recognition circuit of any one of the previous claims, wherein the accelerator signals to the search stage when distances for a new frame are available in a results memory

41. The speech recognition circuit of any one of the previous claims, wherein the search stage processor ignores the completion signal until the processor is ready to accept new distance data.

42. The speech recognition circuit of any one of the previous claims, comprising increasing the pipeline depth by computing extra front end frames in advance.

43. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end, said calculating means, and said search stage are connected to each other to enable pipelined data flow from one to another.

44. A speech recognition circuit as claimed in any one of the previous claims, wherein the audio front end is configured to input a digital audio signal.

45 A speech recognition method, comprising:

calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating means is implemented using a second processor, and wherein data is pipelined from the front end to the calculating means to the search stage.

46. A method as implemented by the apparatus of any previous claim.

47. A speech recognition circuit comprising:

a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words;

a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and

a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

48. The speech recognition circuit of claim 47, wherein for at least some of the states, the node identifier comprises a pointer to a node identifier for another state.

49. The speech recognition circuit of claim 47 or claim 48, wherein the lexical tree is arranged with each node corresponding to a phone, and each said state corresponds to a phone or group of adjacent phones in the lexical tree.

50. The speech recognition circuit of claim 49, wherein the memory structure comprises a first part for storing phone instance identifiers uniquely identifying phones or groups of phones and for storing said state identifiers, and the memory structure comprises a second part for storing said phone instance identifiers and node identifiers directly referencing the lexical tree.

51. The speech recognition circuit of claim 48 or 49, further comprising a counter for sequentially generating phone instance identifiers as unique identification numbers corresponding to phones or groups of phones in the lexical tree.

52. The speech recognition circuit of any previous claim, further comprising a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

53. The speech recognition circuit of any previous claim, wherein the memory structure is configured to hold a record for each of a plurality of nodes in the lexical tree, the record including said states, a previous node identifier for identifying a previous node in the lexical tree, and information indicating whether said each node is a previous node for another node stored in said memory structure,

the speech recognition circuit including a circuit to identify a first node which is not a previous node of another record, and to provide a new data record corresponding to a new node, the new node being the next node after the first node in the lexical tree, using a score associated with the first node.

54. The speech recognition circuit of claim 53, wherein said record comprises a field for storage of a score corresponding to a previous node.

55. The speech recognition circuit of claim 53 or claim 54, further comprising a circuit for overwriting an old record in the memory structure with said new record according to the value of said score for the old record and according to whether said old record is a previous node to said new node or to another node in said memory structure.

56. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises a plurality of memory elements, the memory structure being adapted for independent lookup of states in each of said memory elements.

57. The speech recognition circuit of claim 56, wherein one of said memory elements is configured to store data relating to a first lexical tree, and a second of said memory elements is configured to store data relating to a second lexical tree, the second lexical tree comprising a different model of words than the first lexical tree.

58. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises at least one content addressable memory and at least one

random access memory, the content addressable memory being configured to output addresses of the random access memory.

59. The speech recognition circuit of any one of the previous claims, further comprising a delay circuit for storing a state identifier having a plurality of occurrences in the memory structure, and storing the score update for said state identifier, the delay circuit being configured to provide said state identifier and score after a lookup for a second state identifier produces no match in the memory structure.

60. The speech recognition circuit of any one of the previous claims, comprising a circuit to set a flag indicating an unfavourable score for all records which are previous records and which have scores outside a predetermined range of values.

61. The speech recognition circuit of any one of the previous claims, wherein said memory structure comprises a first memory for receiving and storing said state identifiers, and a second memory for receiving and storing said node identifier, said first memory being configured for lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores, and the second memory being adapted to store a record identifier for a record in the first memory and a reference to the node of the lexical tree for the record identifier, said second memory storing a subset of said records which are not previous records.

62. A method of speech recognition, comprising

providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, the lexical tree comprising a model of words;

providing scores corresponding to said state identifiers;

receiving and storing state identifiers identified by node identifiers identifying a node or group of adjacent nodes in a memory structure;

looking up said memory structure to identify particular state identifiers;

reading of the scores corresponding to the state identifiers;

receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input;

modifying said scores by adding said score updates to said scores;

writing back of the scores to the memory structure after modification of the scores; and

selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

63. The method of claim 62, wherein for at least some of the states, the node identifier comprises a pointer to a node identifier for another state.

64. The method of claim 62 or claim 63, wherein the lexical tree is arranged with each node corresponding to a phone, and each said state corresponds to a phone or group of adjacent phones in the lexical tree.

65. The method of claim 64, wherein the memory structure comprises a first part for storing phone instance identifiers uniquely identifying phones or groups of phones and for storing said state identifiers, and the memory structure comprises a second part for storing said phone instance identifiers and node identifiers directly referencing the lexical tree.

66. The method of claim 64 or claim 65, further comprising a counter for sequentially generating phone instance identifiers as unique identification numbers corresponding to phones or groups of phones in the lexical tree.

67. The method of any one of claims 62 to 66, further comprising a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

68. The method of any one of claims 62 to 67, wherein the memory structure is configured to hold a record for each of a plurality of nodes in the lexical tree, the record including said states, a previous node identifier for identifying a previous node in the lexical tree, and information indicating whether said each node is a previous node for another node stored in said memory structure,

the speech recognition circuit including a circuit to identify a first node which is not a previous node of another record, and to provide a new data record corresponding to a new node, the new node being the next node after the first node in the lexical tree, using a score associated with the first node.

69. Themethod of claim 68, wherein said record comprises a field for storage of a score corresponding to a previous node.

70. The method of claim 68 or claim 69, further comprising a circuit for overwriting an old record in the memory structure with said new record according to the value of said score for the old record and according to whether said old record is a previous node to said new node or to another node in said memory structure.

71. The method of any one of claims 62 to 70, wherein the memory structure comprises a plurality of memory elements, the memory structure being adapted for independent lookup of states in each of said memory elements.

72. The method of claim 71, wherein one of said memory elements is configured to store data relating to a first lexical tree, and a second of said memory elements is configured to store data relating to a second lexical tree, the second lexical tree comprising a different model of words than the first lexical tree.

73. The method of any one of claims 62 to 72, wherein the memory structure comprises at least one content addressable memory and at least one random access memory, the content addressable memory being configured to output addresses of the random access memory.

74. The method of any one of claims 62 to 73, further comprising a delay circuit for storing a state identifier having a plurality of occurrences in the memory structure, and storing the score update for said state identifier, the delay circuit being configured to provide said state identifier and score after a lookup for a second state identifier produces no match in the memory structure.

75. The method of any one of claims 62 to 74, comprising a circuit to set a flag indicating an unfavourable score for all records which are previous records and which have scores outside a predetermined range of values.

76. The method of any one of claims 62 to 75, wherein said memory structure comprises a first memory for receiving and storing said state identifiers, and a second memory for receiving and storing said node identifier, said first memory being configured for lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores, and the second memory being adapted to store a record identifier for a record in the first memory and a reference to the node of the lexical tree for the record identifier, said second memory storing a subset of said records which are not previous records.

77. A mobile phone including the speech recognition circuit of any one of claims 1 to 76.

78. A portable electronic device including the speech recognition circuit of any one of claims 1 to 77.

79. A speech recognition circuit comprising:

lexical memory containing lexical data for word recognition, said lexical data comprising a lexical tree data structure comprising a model of words;

means for accessing a state model corresponding to each phone or each group of phones in the lexical tree;

a content addressable memory for storing content addressable data for each phone or group of phones, including states corresponding to said phone or group of phones, and for storing an address value for each said phone or group of phones;

a RAM configured to store accumulated scores for each said phone or group of phones, the accumulated scores being addressable by said address value for each said phone or group of phones;

means to obtain scores that each of a plurality of frames of an audio signals corresponds to each of a plurality of said states;

a counter to sequentially search for each said state in the content addressable memory, to obtain the corresponding address value if the state is found in the content addressable memory;

means to use said address value to access an accumulated likelihood and an accumulator to add said likelihood to the accumulated likelihood;

means to use the phones or groups of phones with the highest accumulated scores to obtain a plurality of next phones from the lexical tree which correspond to the next phone; and

output means for outputting a lexical tree path of highest likelihood.

80. A speech recognition apparatus comprising:

a lexical tree having a corresponding state model;

means for obtaining scores of an audio input corresponding to each of a plurality of states in said state model;

a content addressable memory for storing a marker indicating a part of the lexical tree, and one or more states associated with said part of the lexical tree;

a random access memory addressable by the CAM output, to output accumulated scores for states corresponding to said parts of the lexical tree;

adder means for adding likelihood to said accumulated likelihood, to be stored back in the RAM;

81. A speech recognition apparatus, comprising:

a CAM-RAM arrangement for storing records including pointers to a lexical tree, and accumulative scores for states within the lexical tree;

input means for obtaining scores that an audio frame corresponds to a particular state in the lexical tree;

an accumulator for calculating the updated scores and modifying the records in the CAM-RAM accordingly;

output means for outputting a path of highest likelihood in the lexical tree.

82. A speech recognition apparatus substantially as hereinbefore described with reference to any of figures 1 to 15 of the accompanying drawings.

83. A speech recognition method comprising:

storing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and scores corresponding to said state identifiers in a memory structure, the lexical tree comprising a model of words and the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

repeating the following sequence of steps for each of a plurality of incoming frames of an audio signal:

obtaining score updates corresponding to the likelihoods that said frame of the audio signal corresponds to each of a plurality of said states;

accessing said memory structure to obtain scores, updating the scores by adding score updates to the scores, and writing back the updated scores to the memory structure;

determining if scores for states furthest on in the lexical tree correspond to a significant likelihood, and if so, then accessing the lexical tree to determine the next set of possible states; and

storing the next set of possible states and said scores of significant likelihood in the memory structure;

84. A speech recognition circuit comprising:

a circuit for providing state identifiers which identify states corresponding to phones or groups of adjacent phones in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words;

a memory structure for receiving and storing state identifiers and phone instance identifiers uniquely identifying instances of phones or groups of phones in the lexical tree;

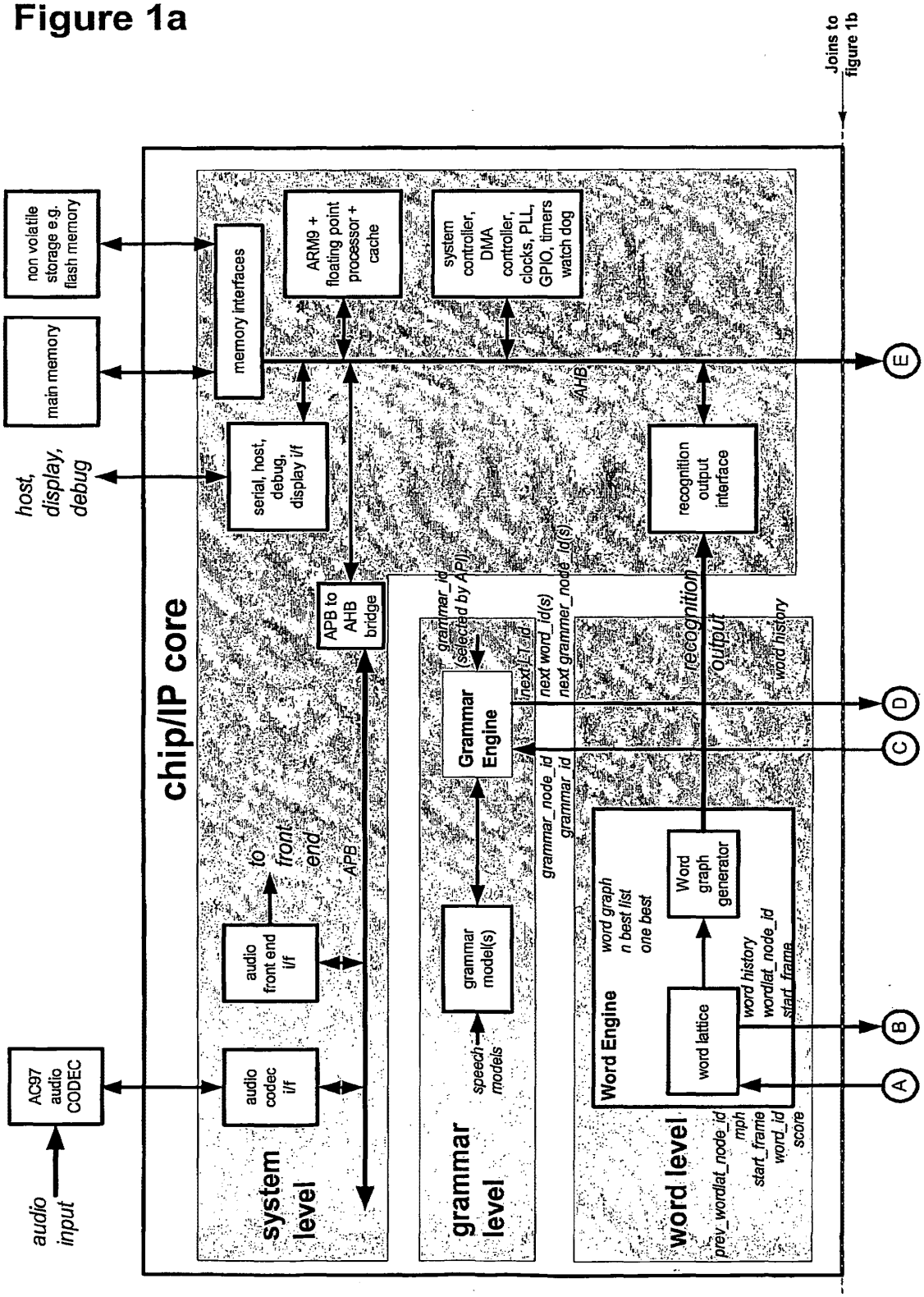
said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates

using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and

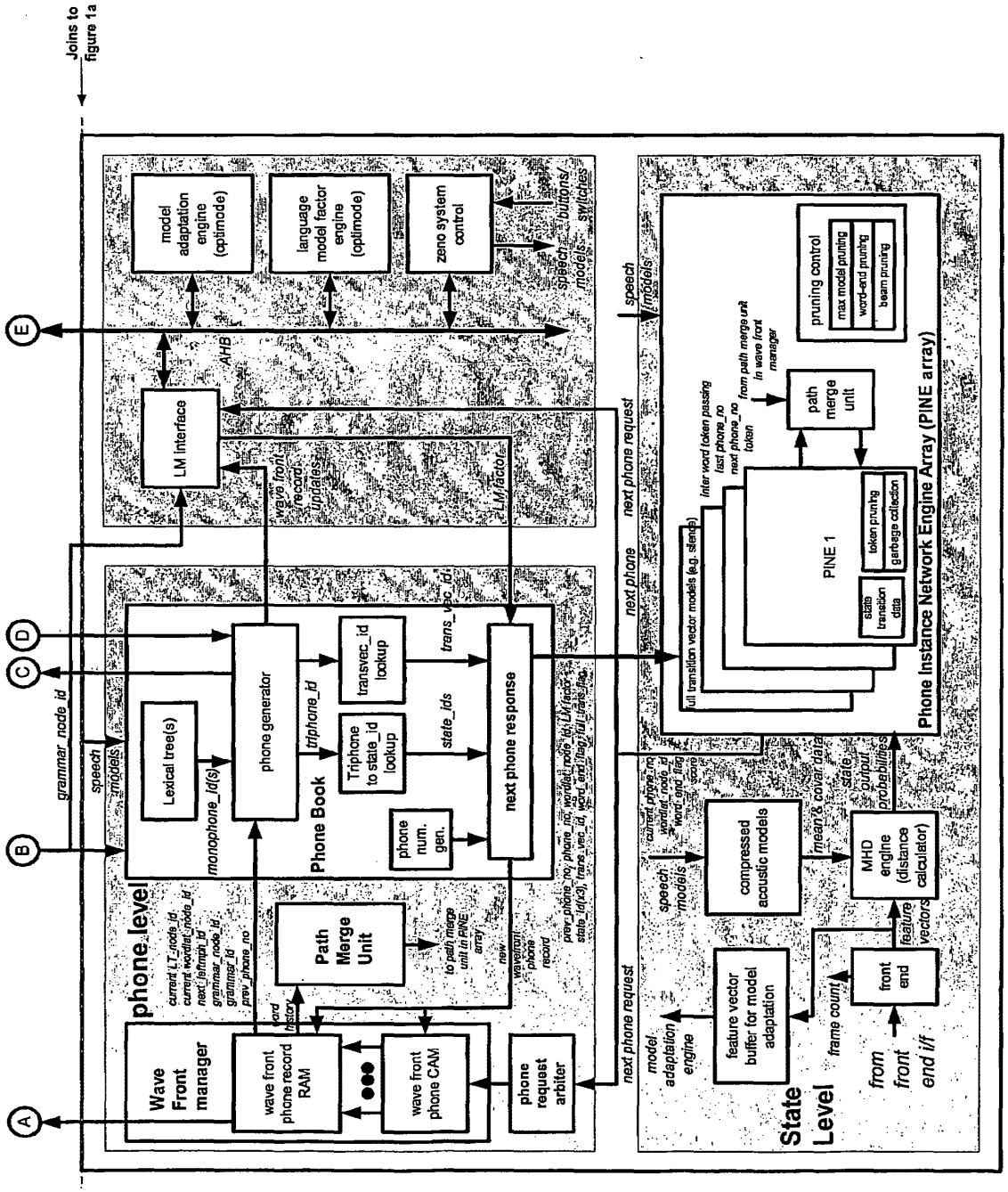
a selector circuit for selecting at least one phone instance identifier according to said scores.

Figure 1a



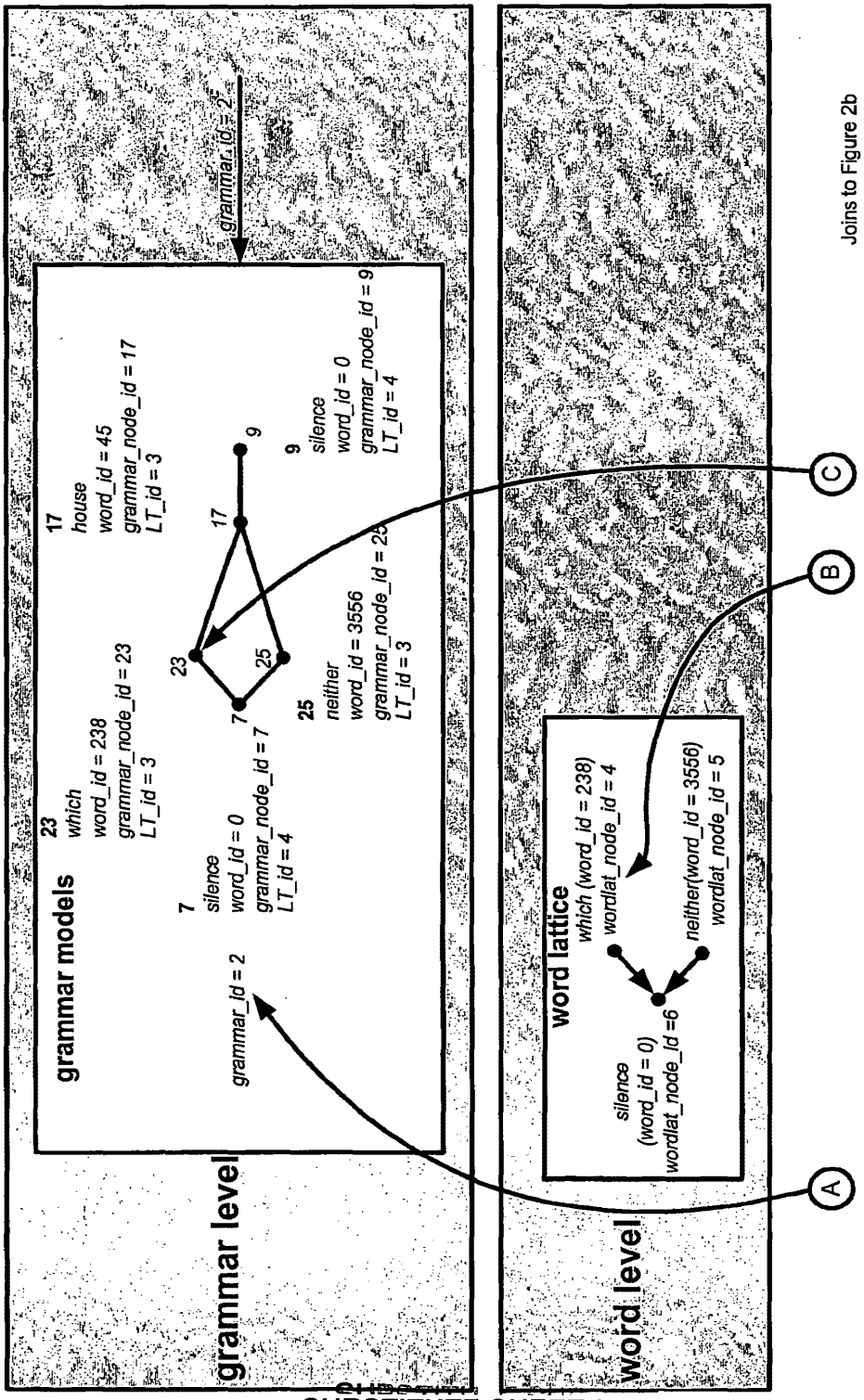
SUBSTITUTE SHEET (RULE 26)

Figure 1b



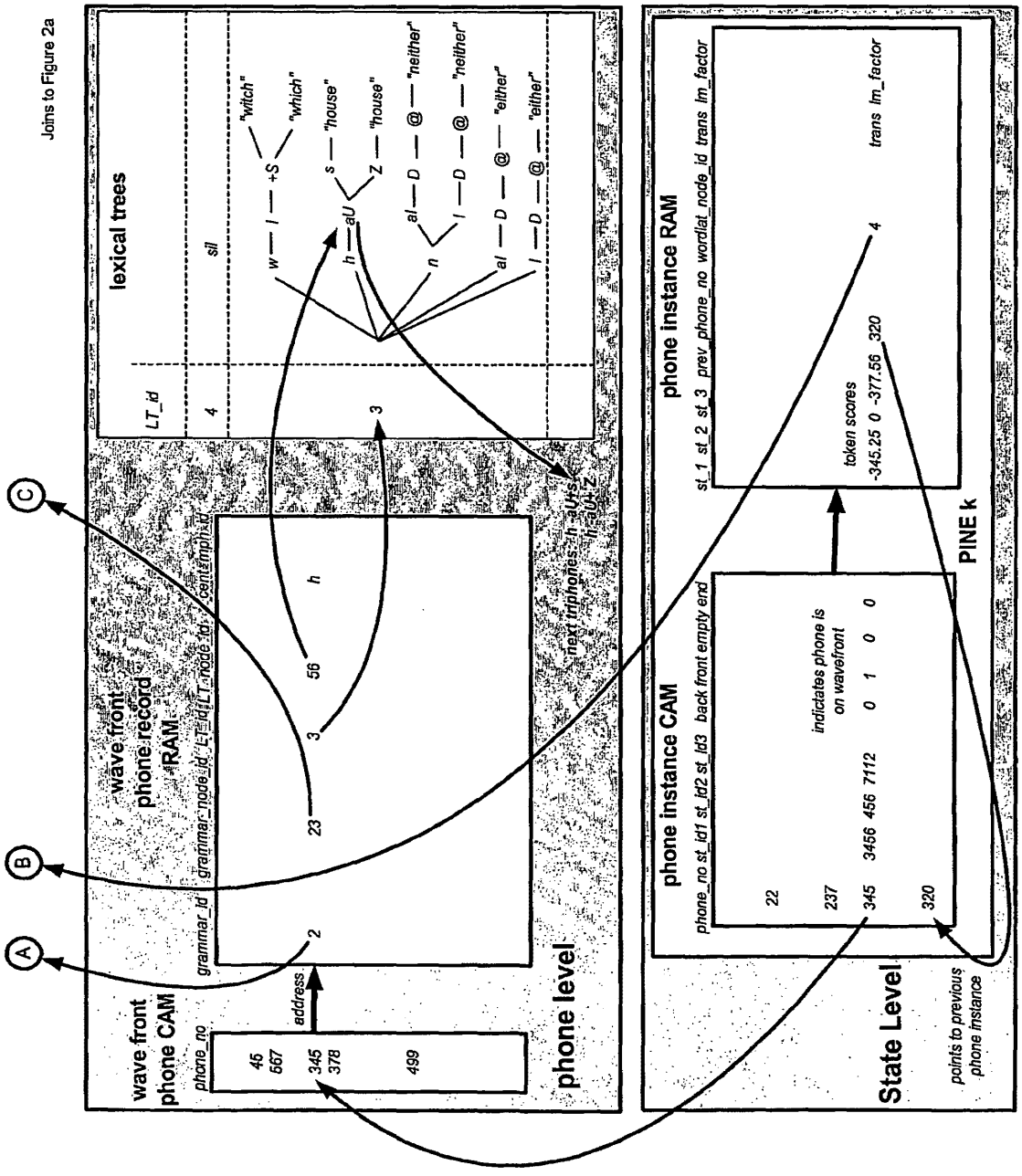
Amazon / Zentian Limited
Exhibit 1002
Page 146

Figure 2a



Amazon / Zentian Limited
Exhibit 1002
Page 147

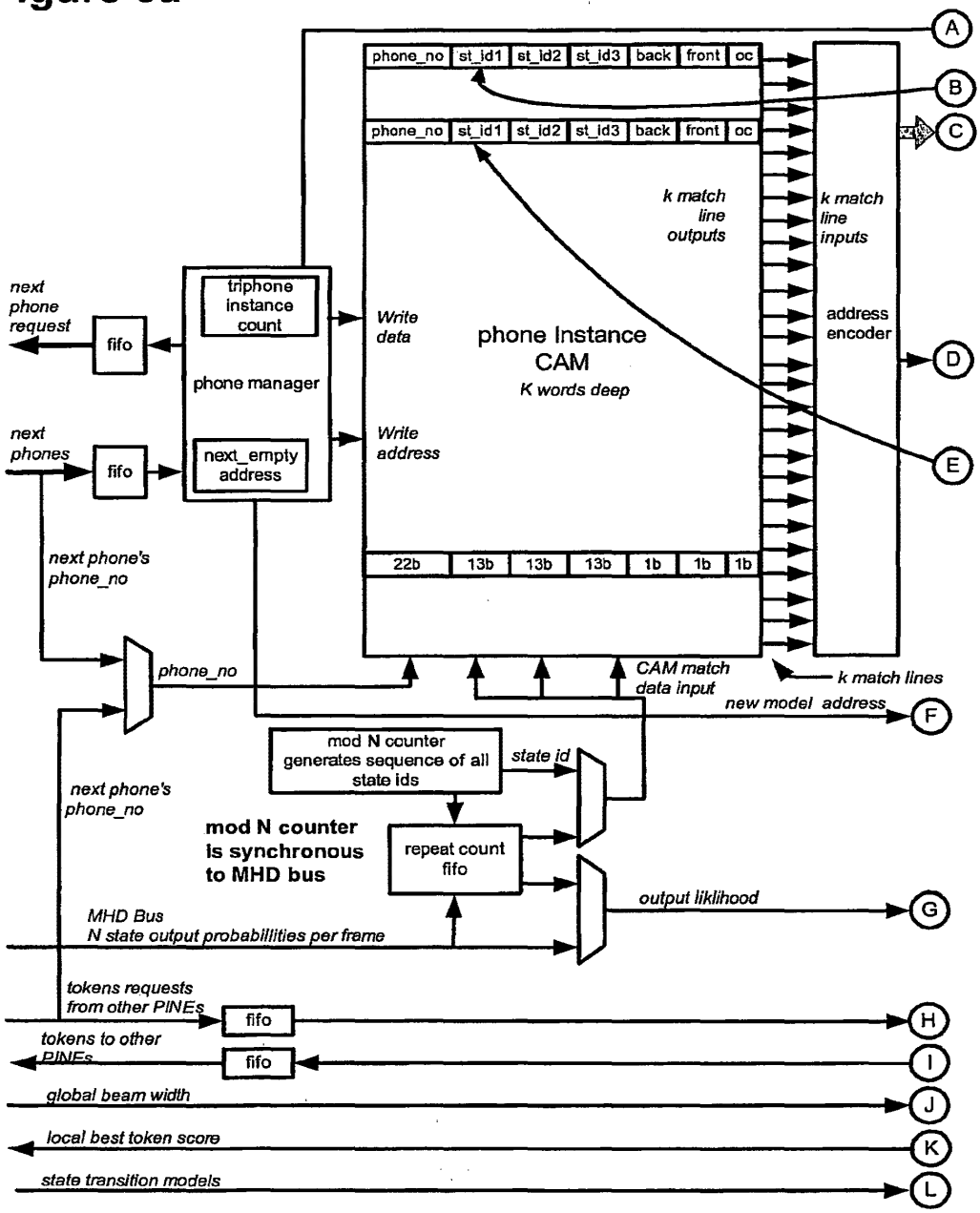
Figure 2b



SUBSTITUTE SHEET (RULE 26)

Amazon / Zentian Limited
Exhibit 1002
Page 148

Figure 3a



CAM abbreviations

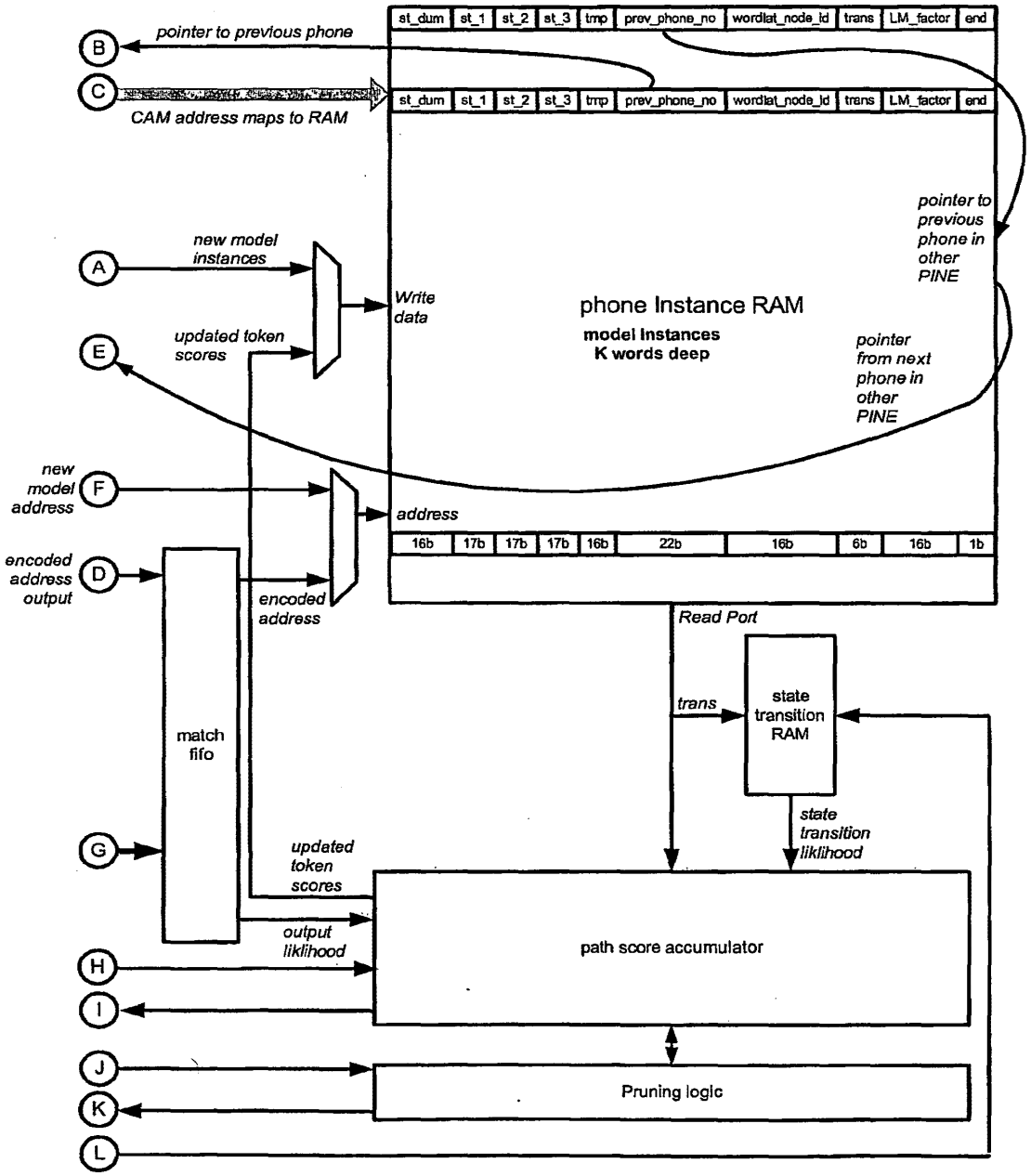
phone_no - unique phone instance indentifier
st_id1 - state_identifier1
back(wave_back_flag) - phone is at back of wave
front(wave_front_flag) - phone is at the front of wave
end(word_end_flag) - phone is the last in a word
oc - occupy - indicates that location has valid phone instance (it may not have any tokens)

RAM abbreviations

st_dum - dummy state to hold incoming token
st_1 - location for token in state 1, includes updated flag
st_2 - location for token in state 2, includes updated flag
st_3 - location for token in state 3, includes updated flag
tmp - location for temporary storage of token
prev_phone_no - pointer to previous phone instance
trans (trans_vec_id) - pointer to state transition vector in state transition RAM
LM_factor - language model factor
end - word end phone

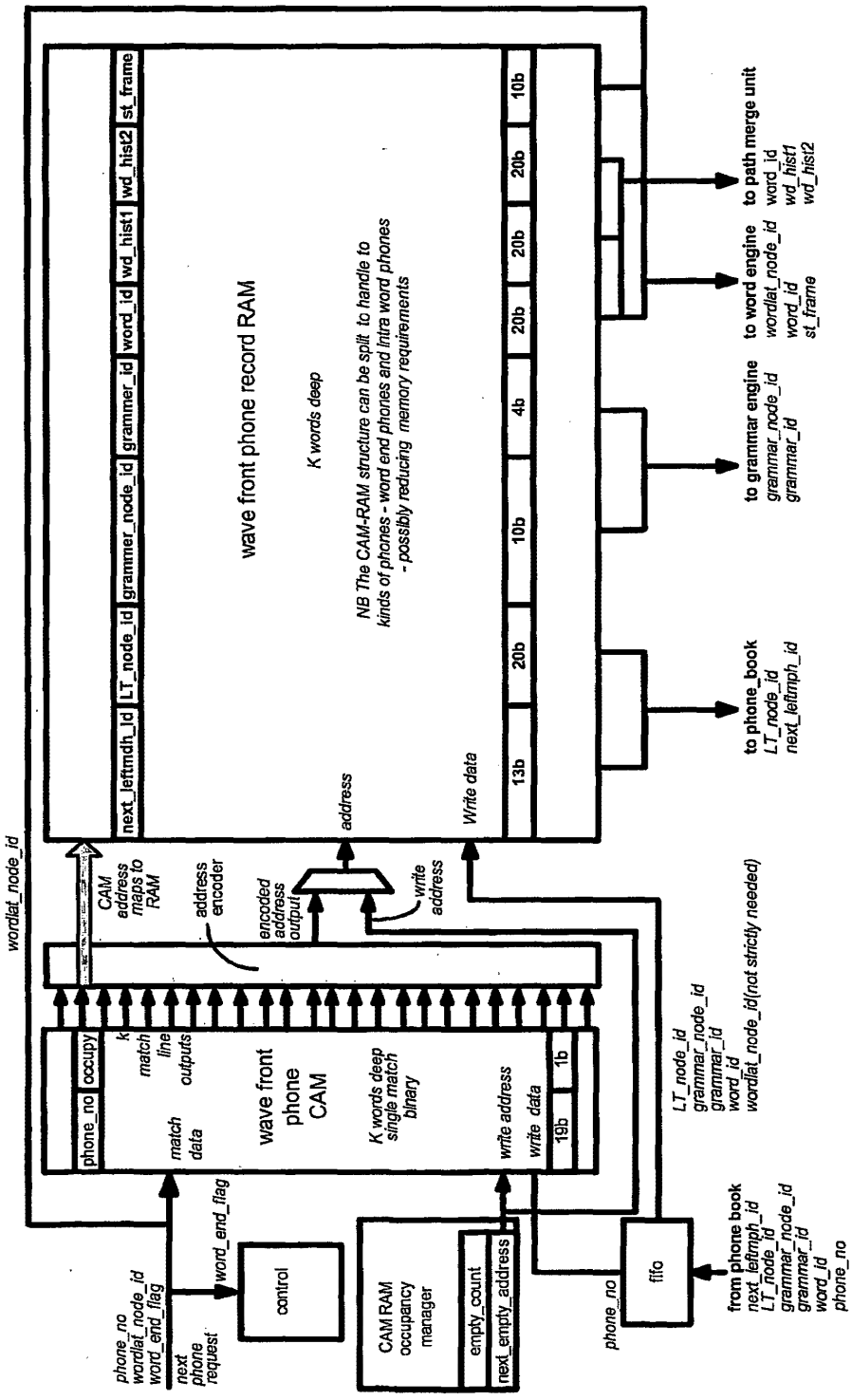
Amazon / Zentian Limited
Exhibit 1002
Page 149

Figure 3b



Amazon / Zentian Limited
Exhibit 1002
Page 150

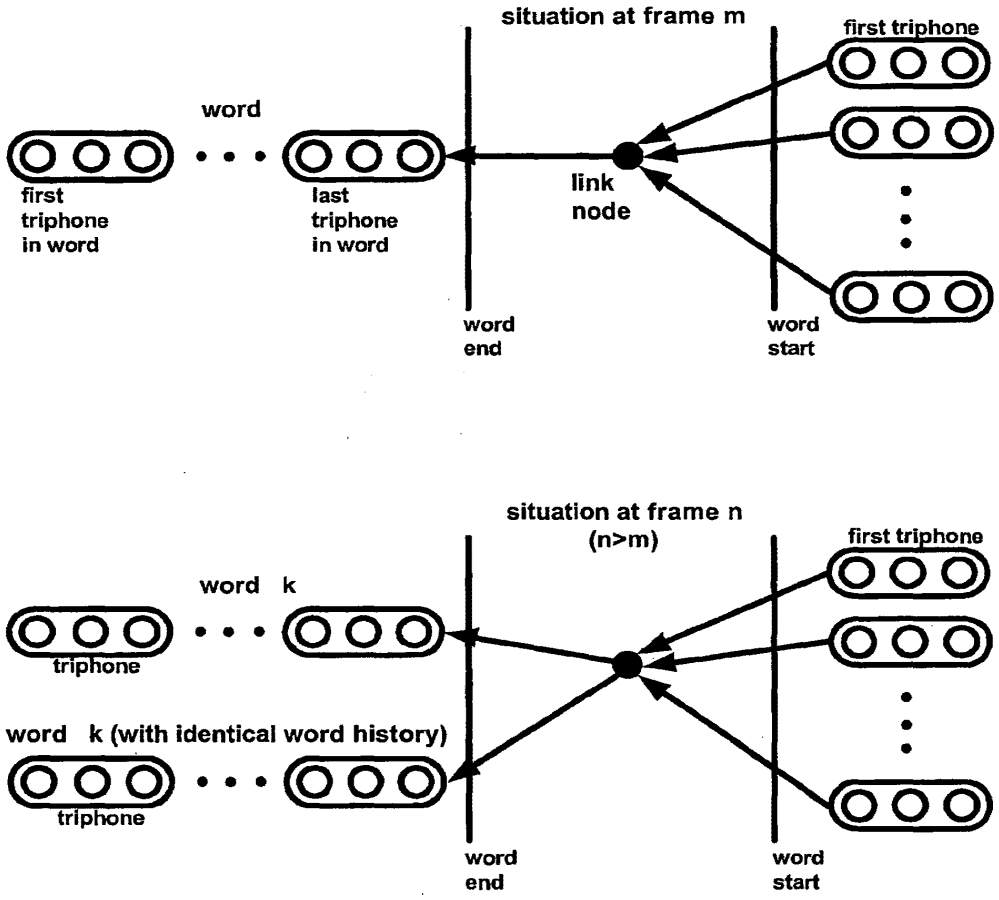
Figure 4



SUBSTITUTE SHEET (RULE 26)

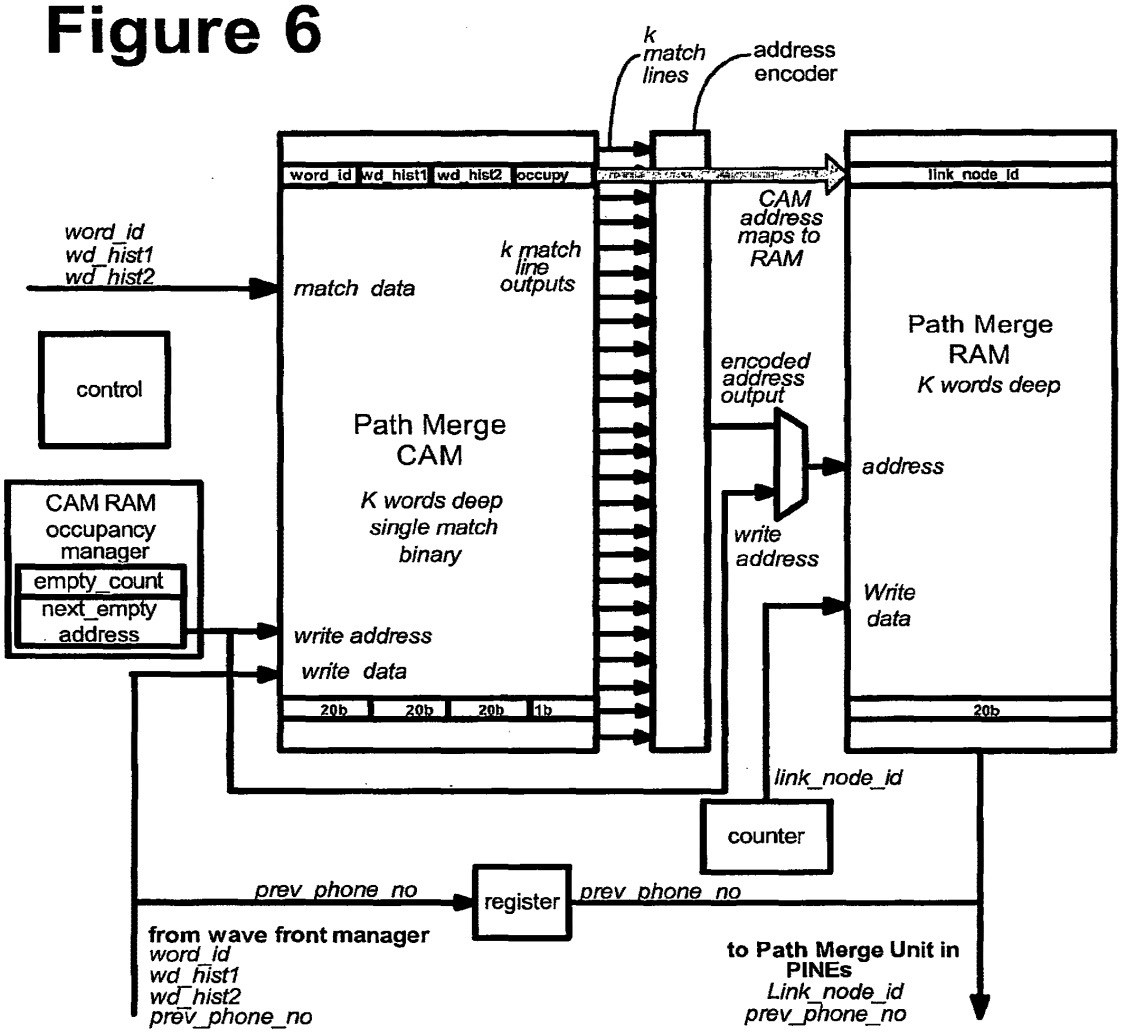
Amazon / Zentian Limited
Exhibit 1002
Page 151

Figure 5



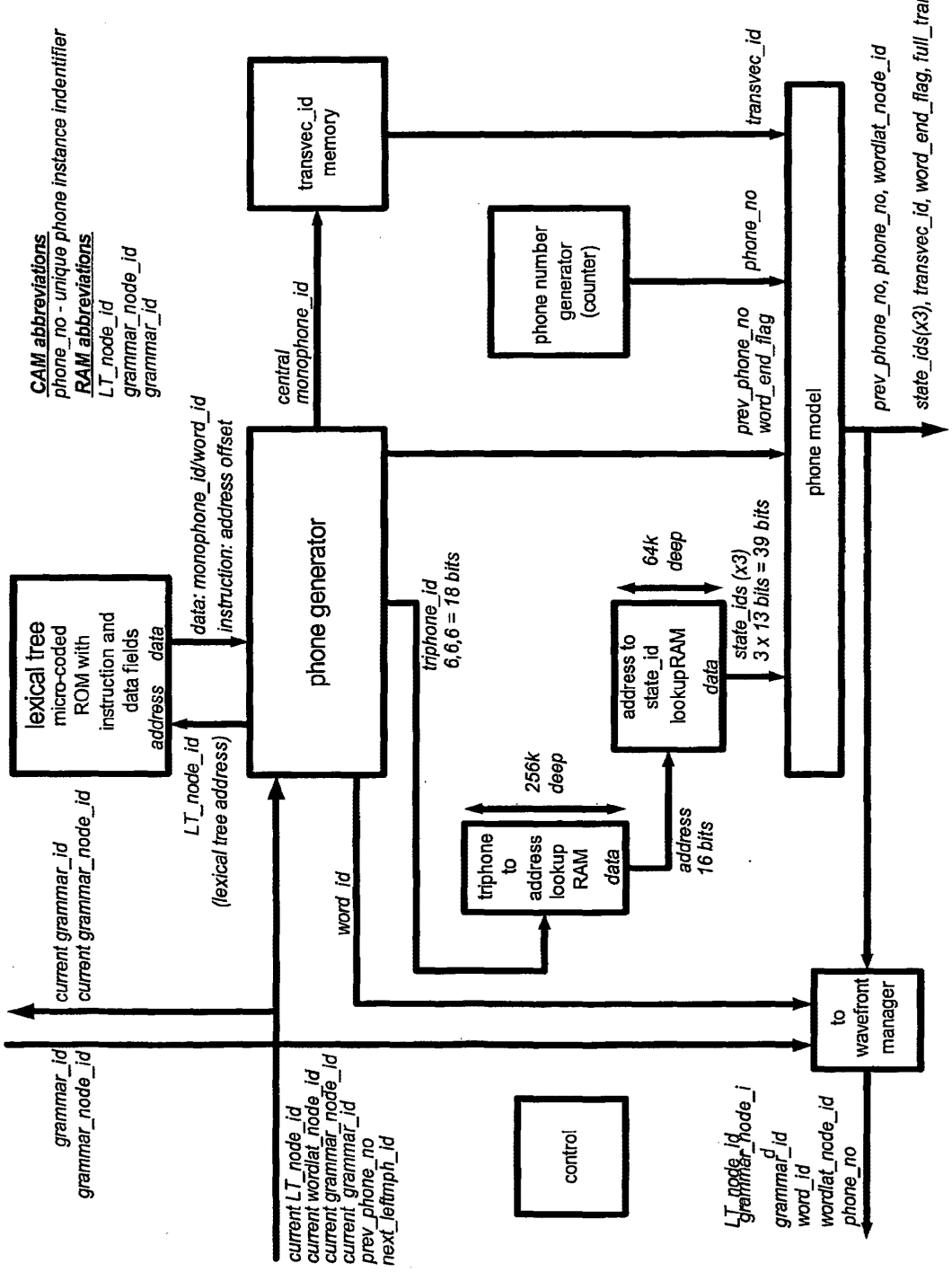
Amazon / Zentian Limited
Exhibit 1002
Page 152

Figure 6



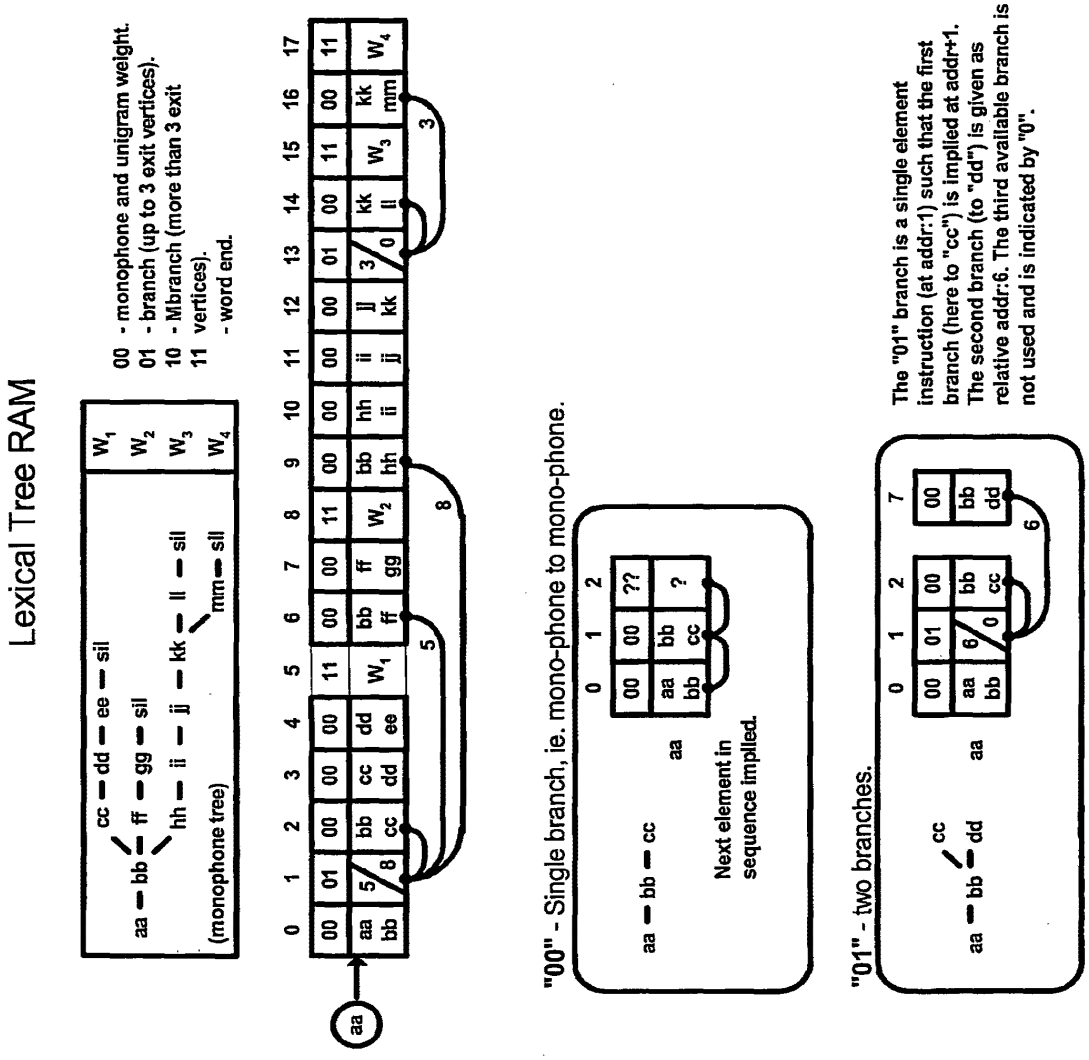
Amazon / Zentian Limited
Exhibit 1002
Page 153

Figure 7



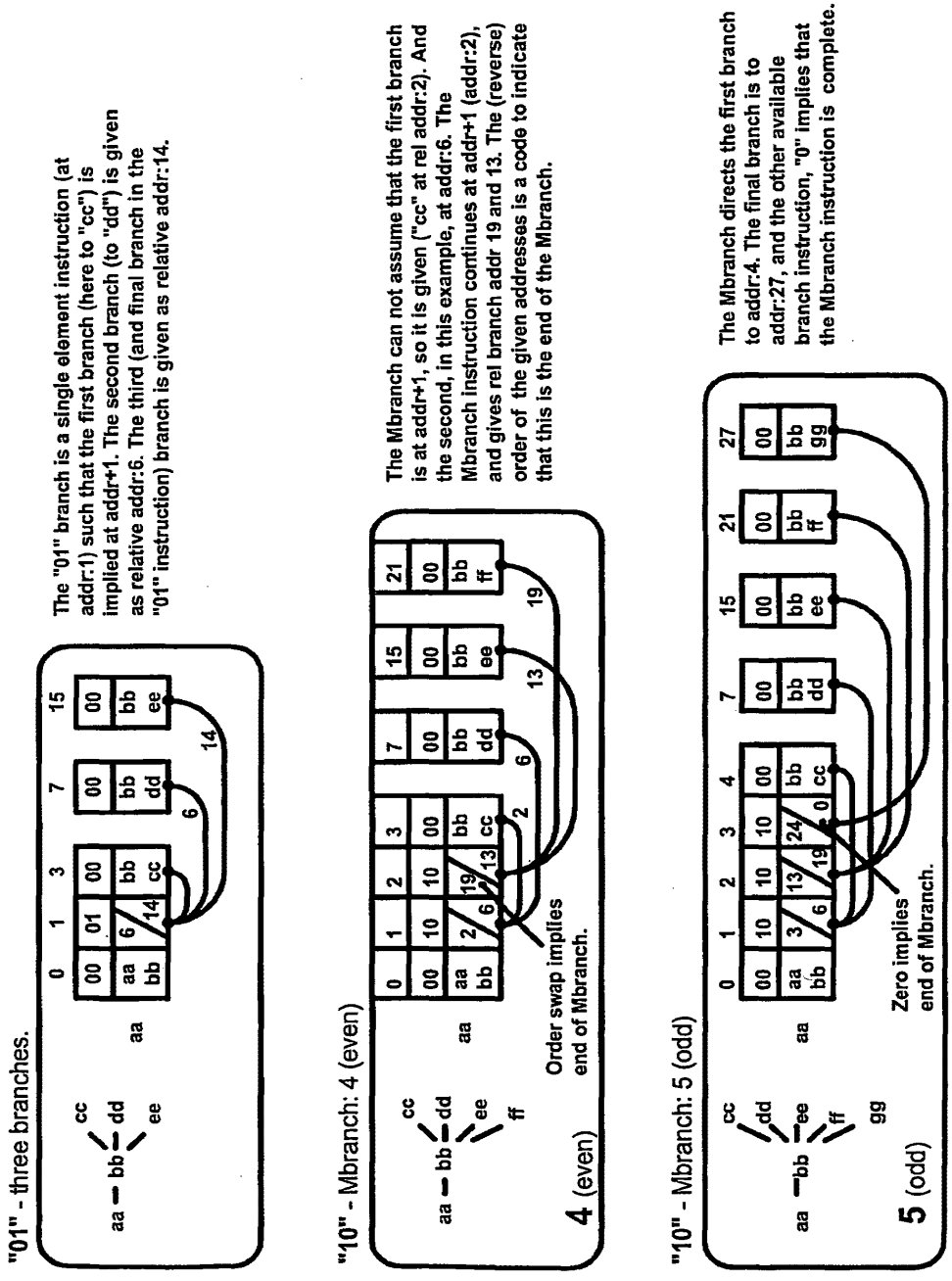
Amazon / Zentian Limited
Exhibit 1002
Page 154

Figure 8a



Amazon / Zentian Limited
Exhibit 1002
Page 155

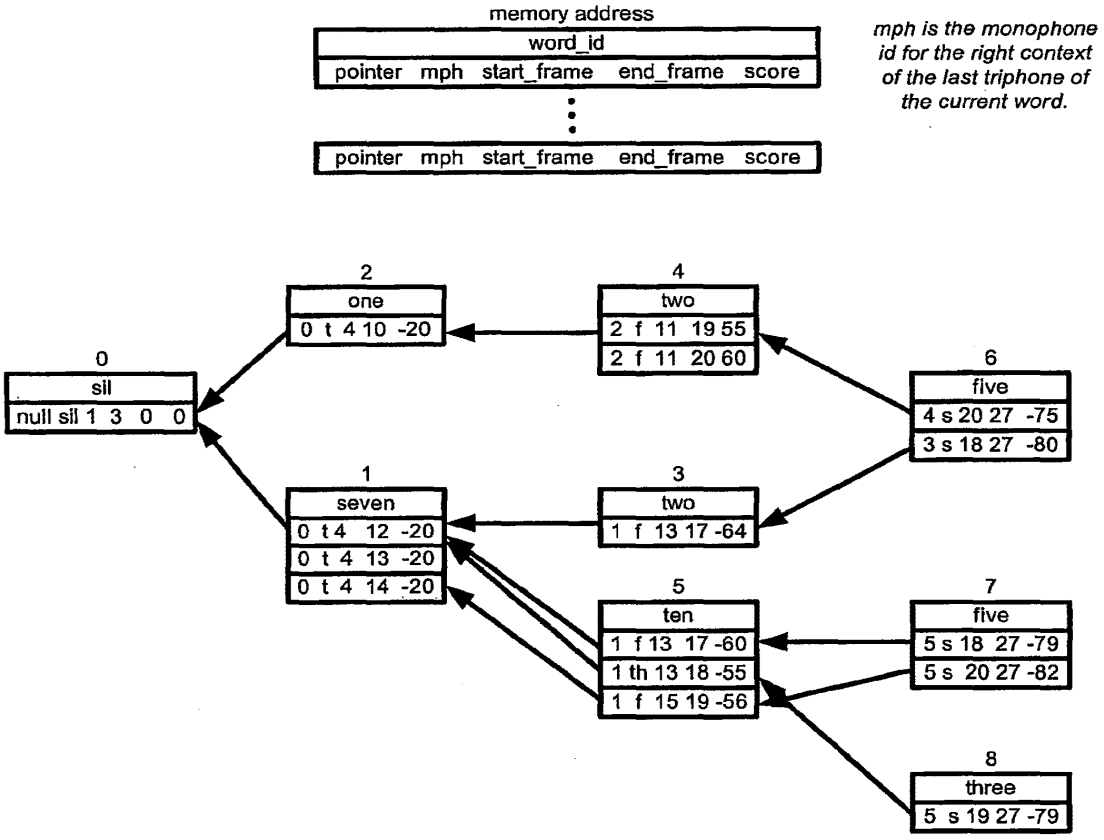
Figure 8b



Amazon / Zentian Limited
Exhibit 1002
Page 156

Figure 9

data structure for word link record to enable word n-best decoding



Notes

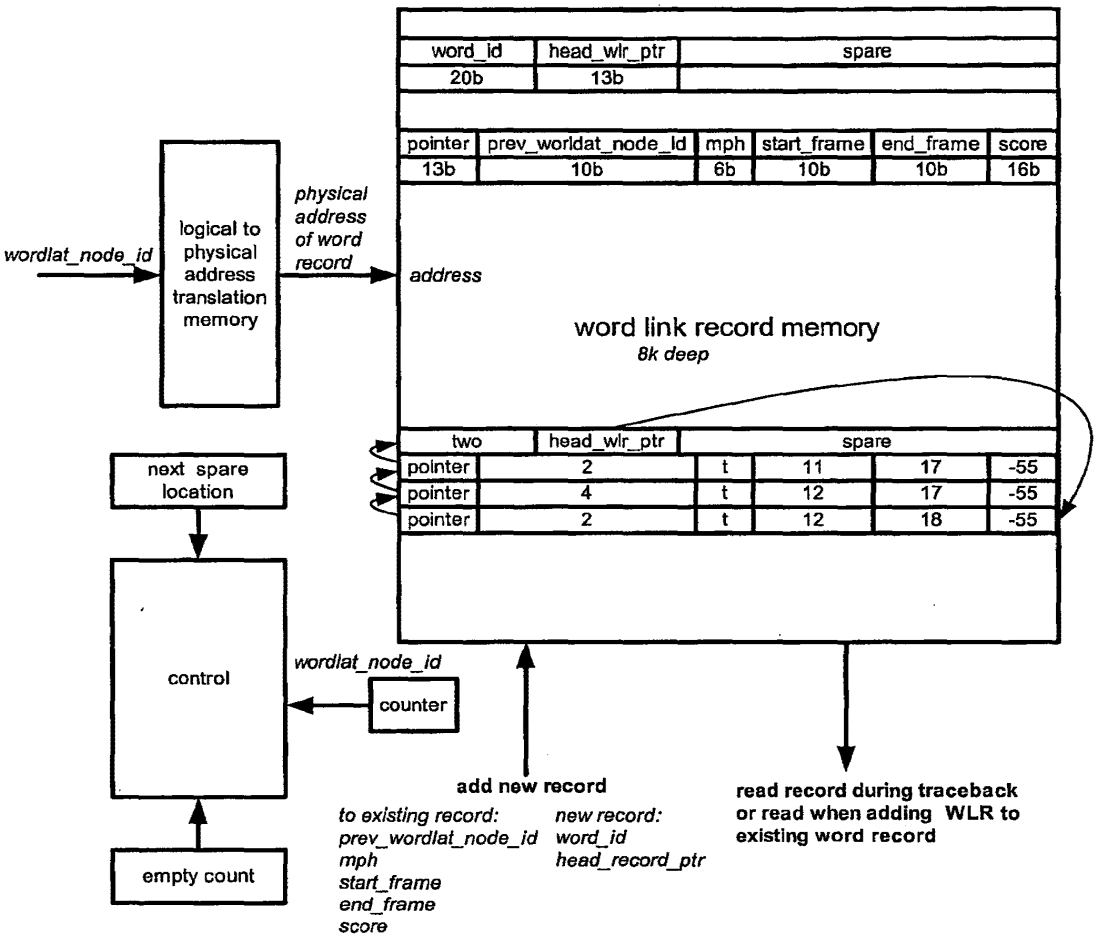
1. each record represents the passage of a single token through a word with a particular right monophone context.
2. each word's acoustic model with a particular right context monophone can only generate one token per frame. Thus frame number can be used to distinguish trace back paths with the same end_frame numbers but different right context monophones
3. the end_frame doesn't need to be passed to the PINEs because it is simply the current frame number
4. scores may be absolute or relative
5. the address is referred to as the wordlat_node_id elsewhere

Traceback

for the n-best tokens
1. using the associated wordlat_node_ids look up each word link record
2. look at associated records and use the end_frame number and monophone id to select associated pointer and score.
3. trace back recursively and select n-best paths

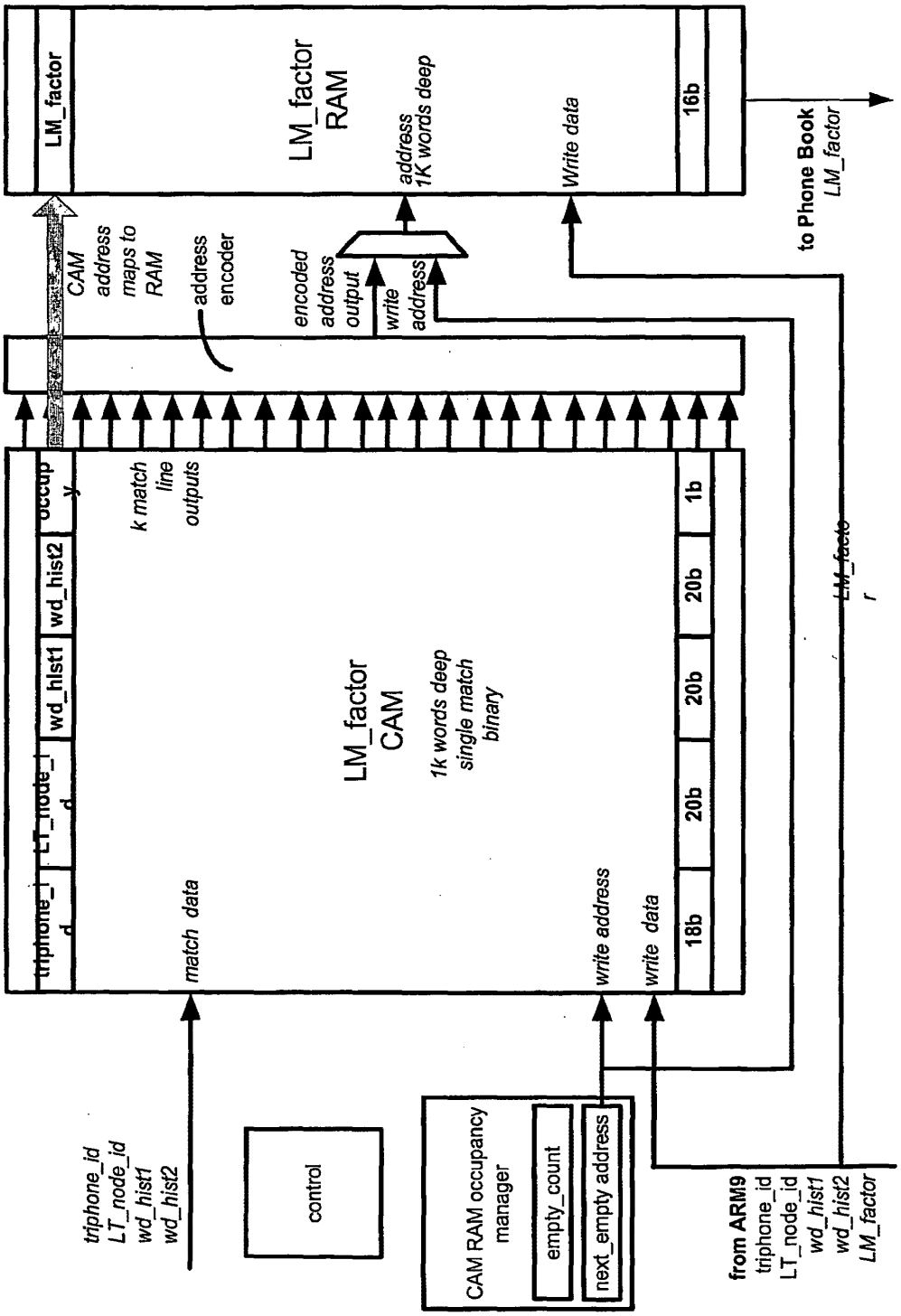
Amazon / Zentian Limited
Exhibit 1002
Page 157

Figure 10



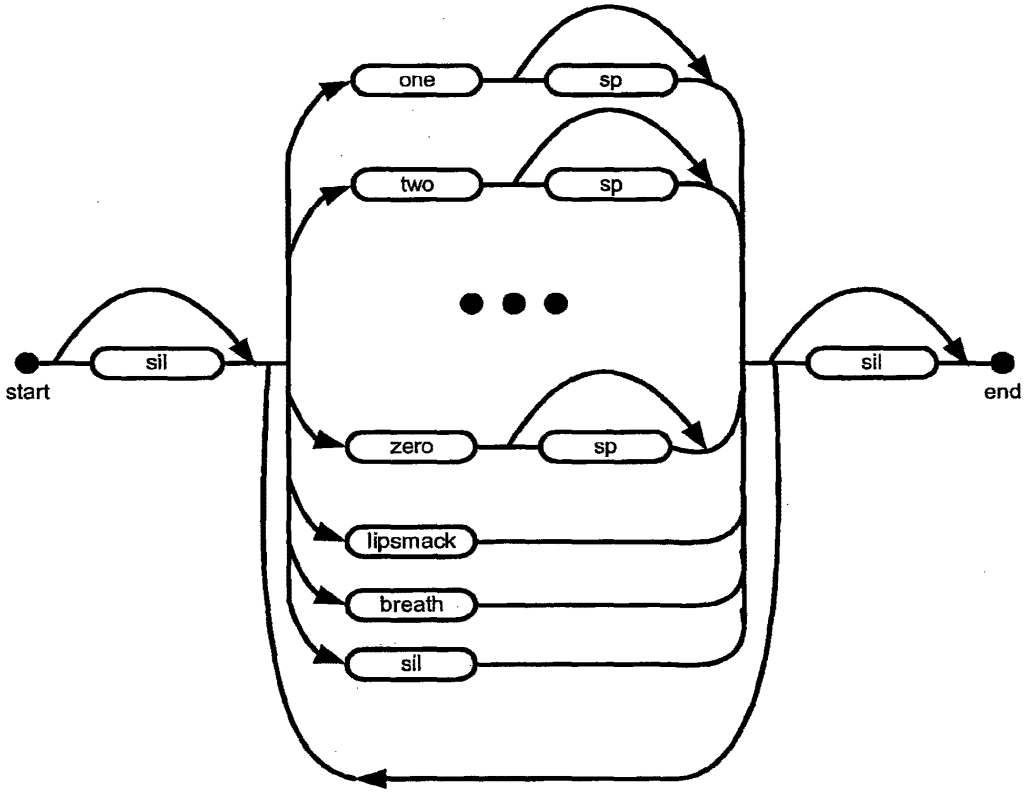
Amazon / Zentian Limited
Exhibit 1002
Page 158

Figure 11



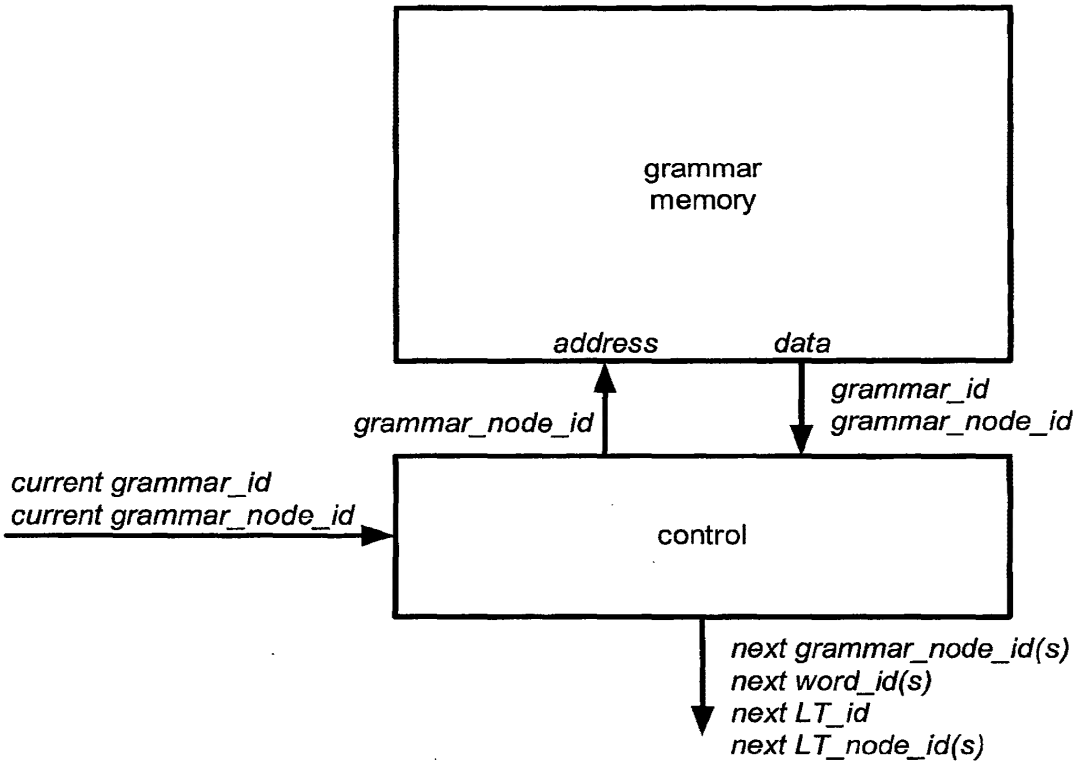
SUBSTITUTE SHEET (RULE 26)

Figure 12



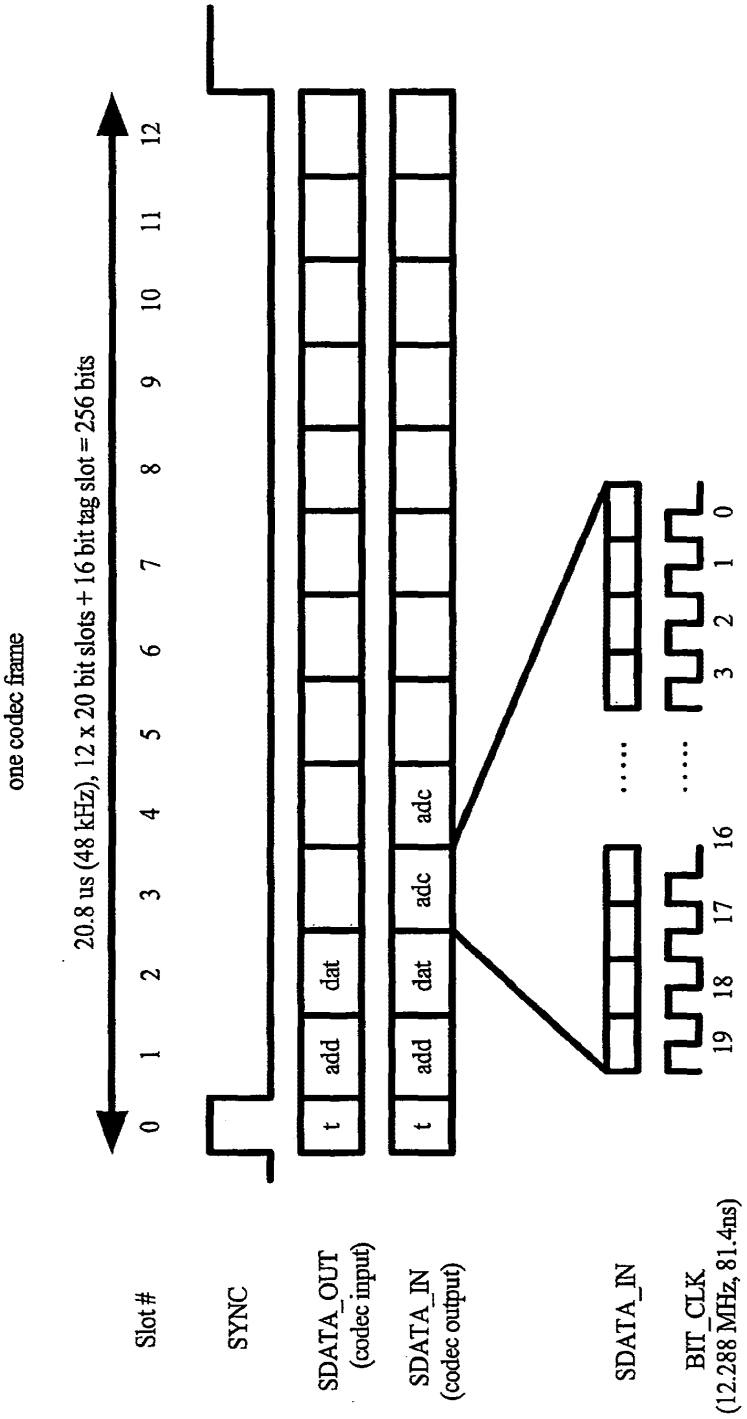
Amazon / Zentian Limited
Exhibit 1002
Page 160

Figure 13



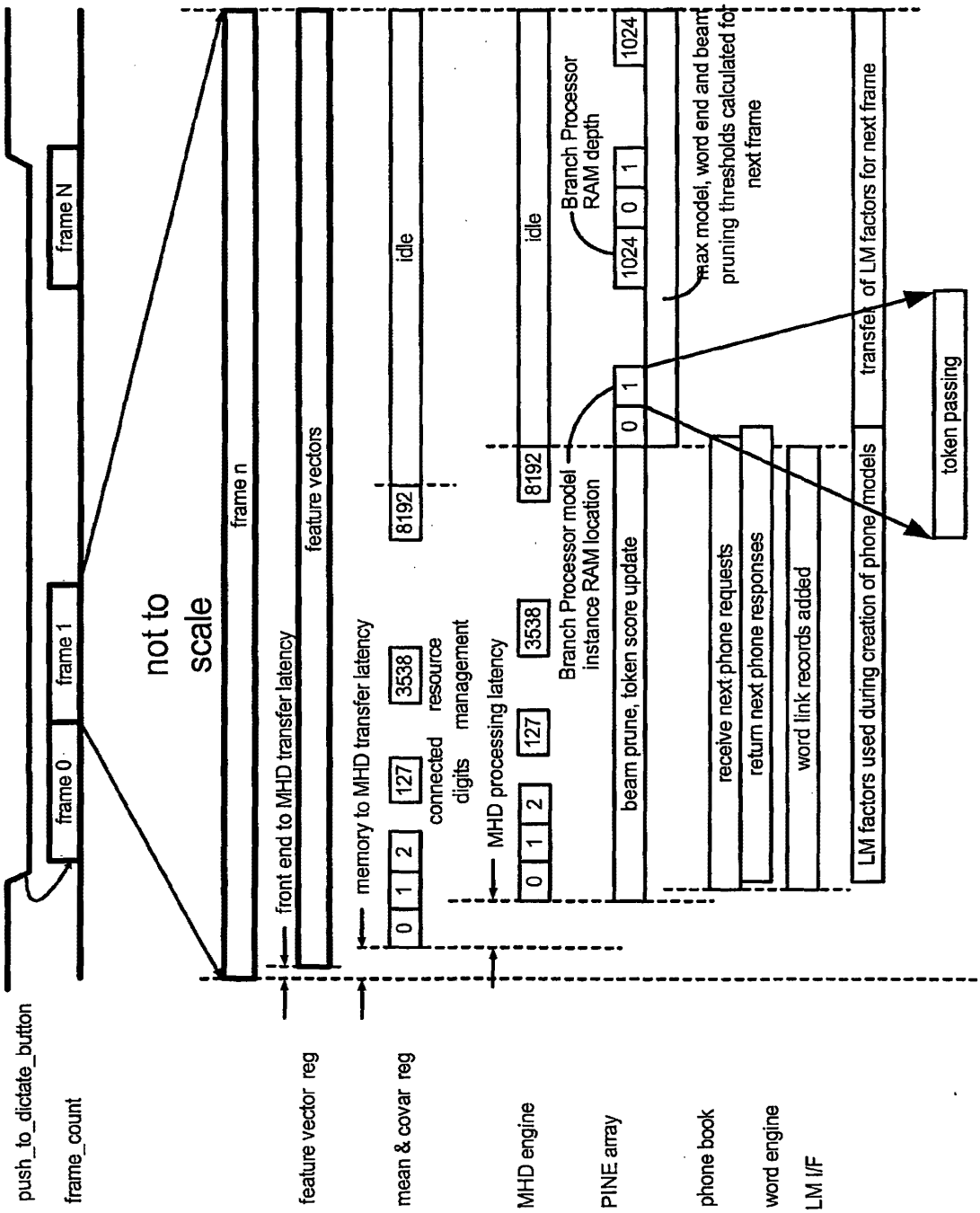
Amazon / Zentian Limited
Exhibit 1002
Page 161

Figure 14



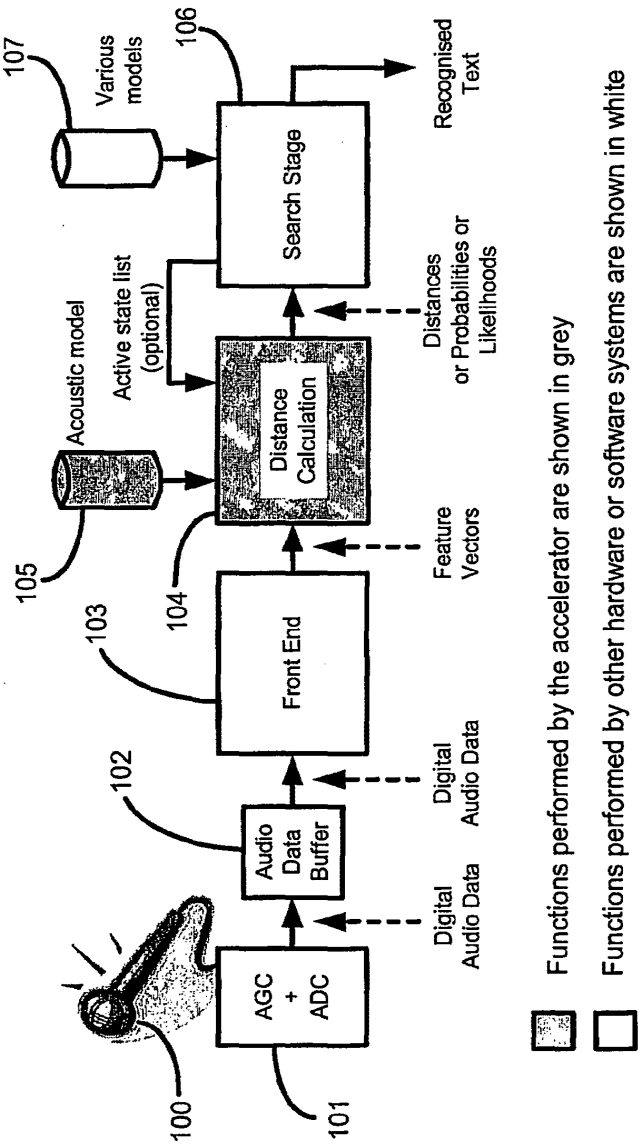
Amazon / Zentian Limited
Exhibit 1002
Page 162

Figure 15



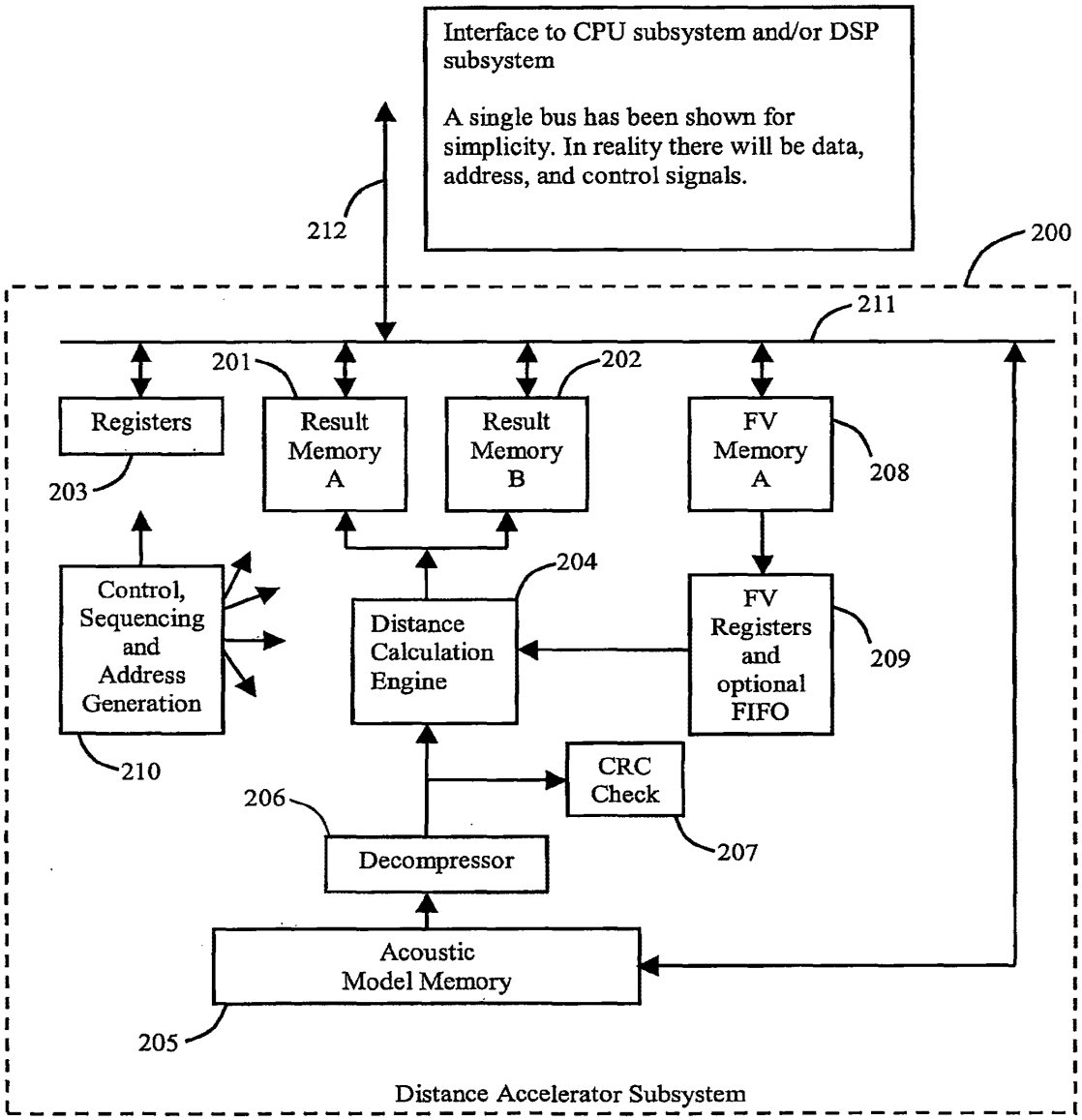
Amazon / Zentian Limited
Exhibit 1002
Page 163

Figure 16



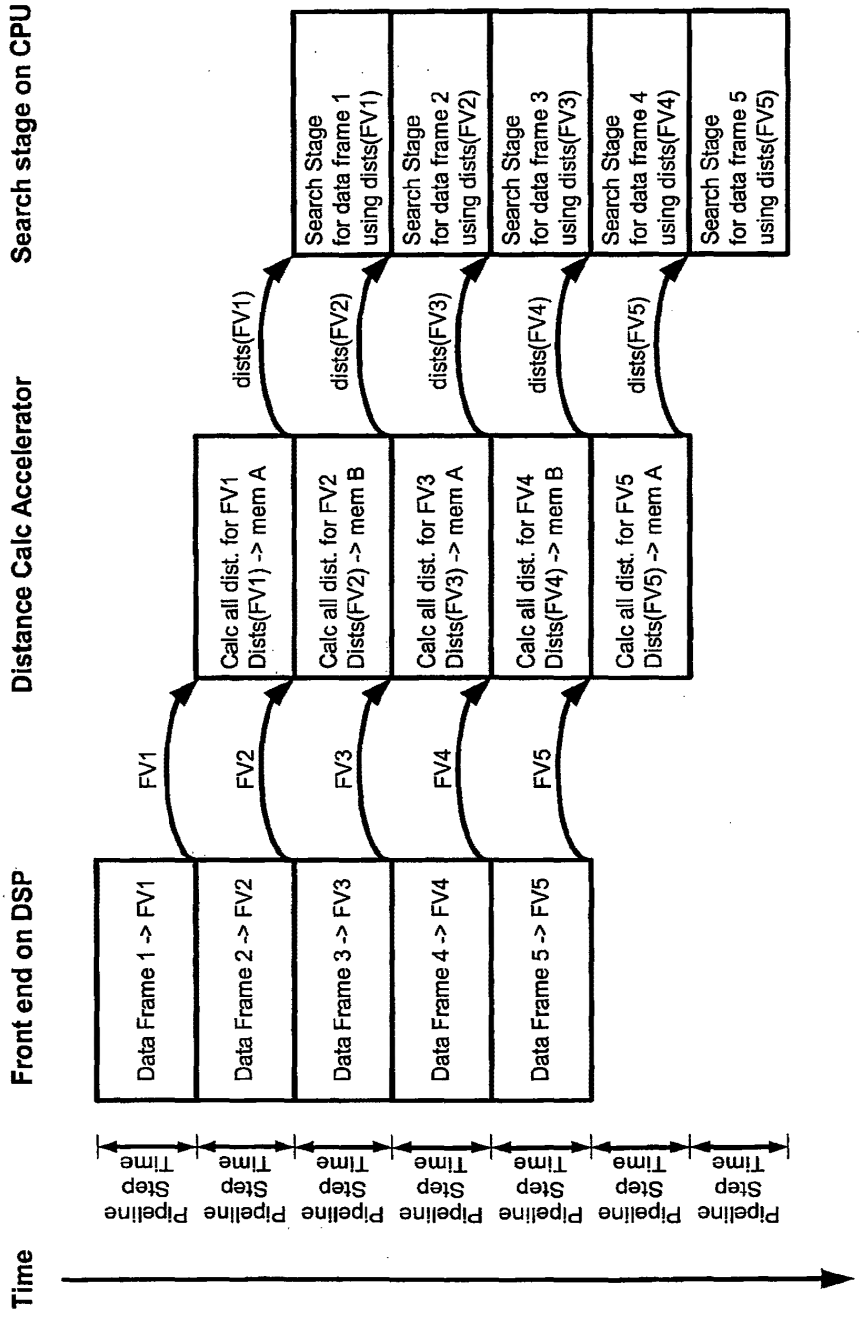
Amazon / Zentian Limited
Exhibit 1002
Page 164

FIGURE 17



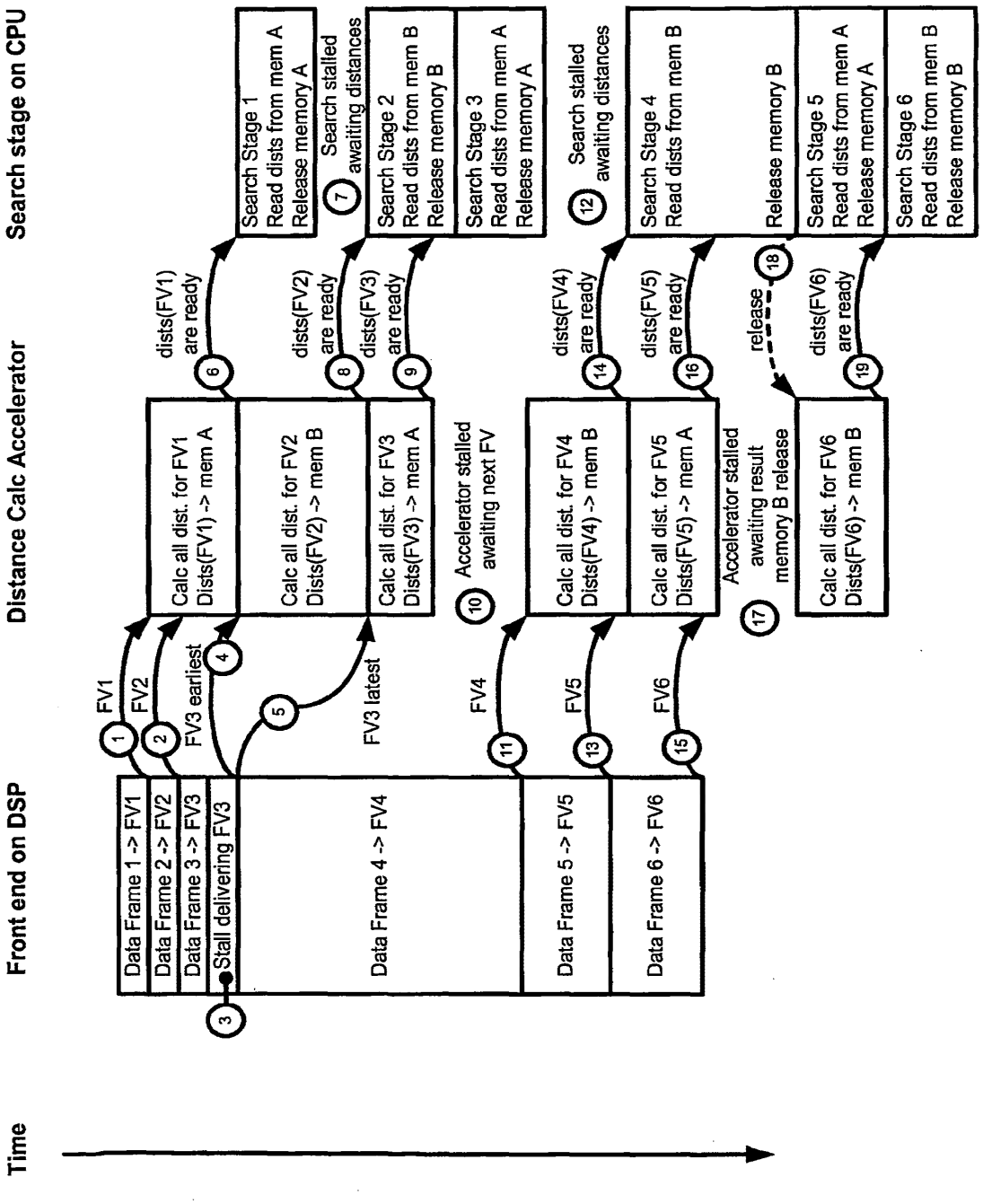
Amazon / Zentian Limited
Exhibit 1002
Page 165

Figure 18



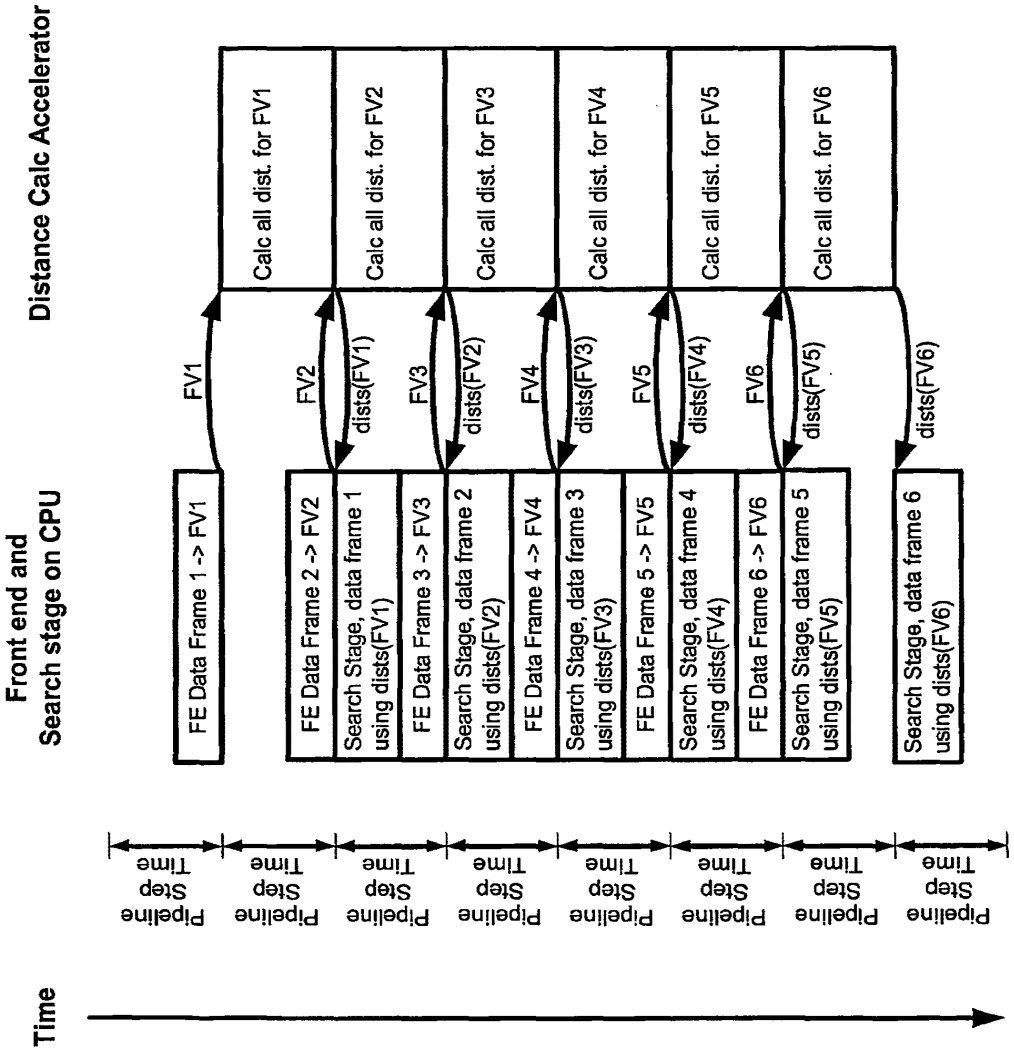
Amazon / Zentian Limited
Exhibit 1002
Page 166

Figure 19



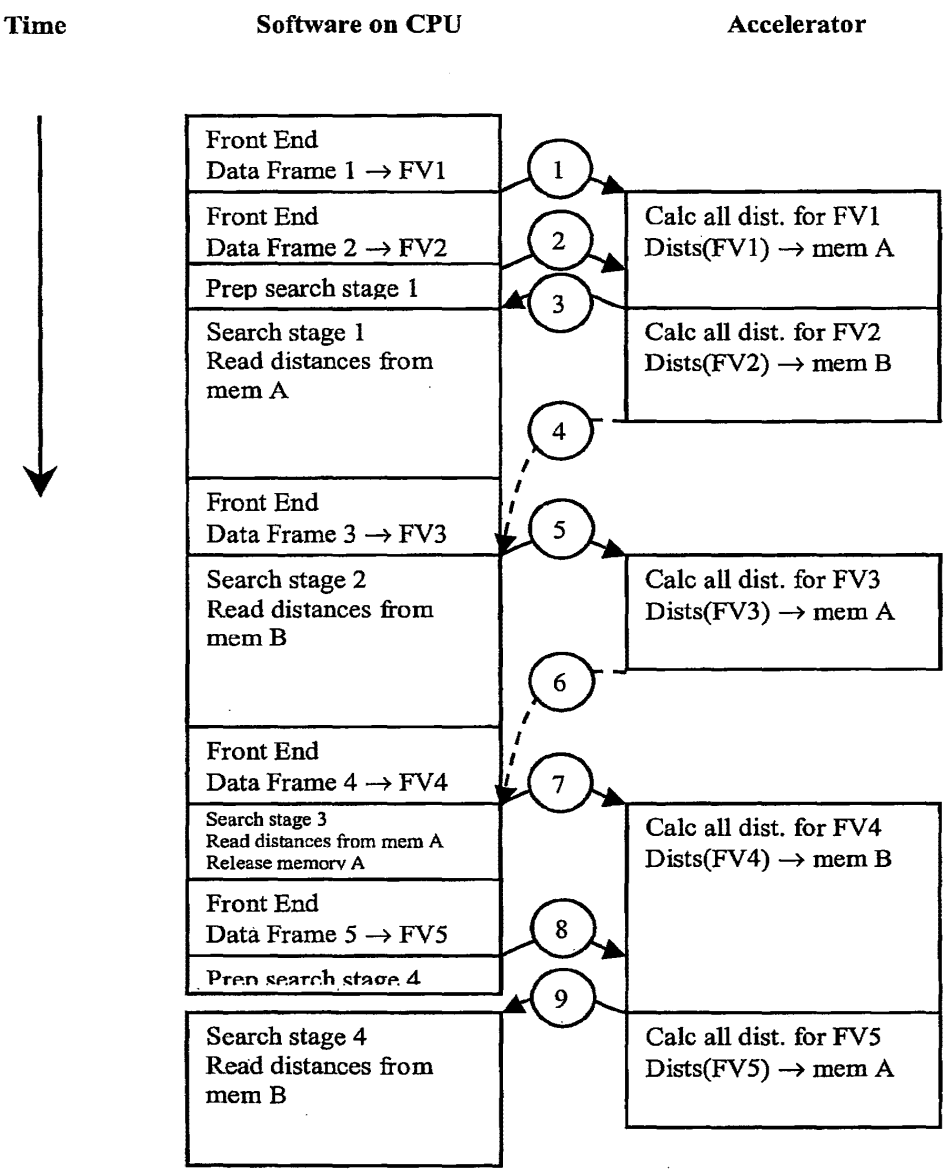
Amazon / Zentian Limited
Exhibit 1002
Page 167

Figure 20



Amazon / Zentian Limited
Exhibit 1002
Page 168

FIGURE 21



Amazon / Zentian Limited
Exhibit 1002
Page 169

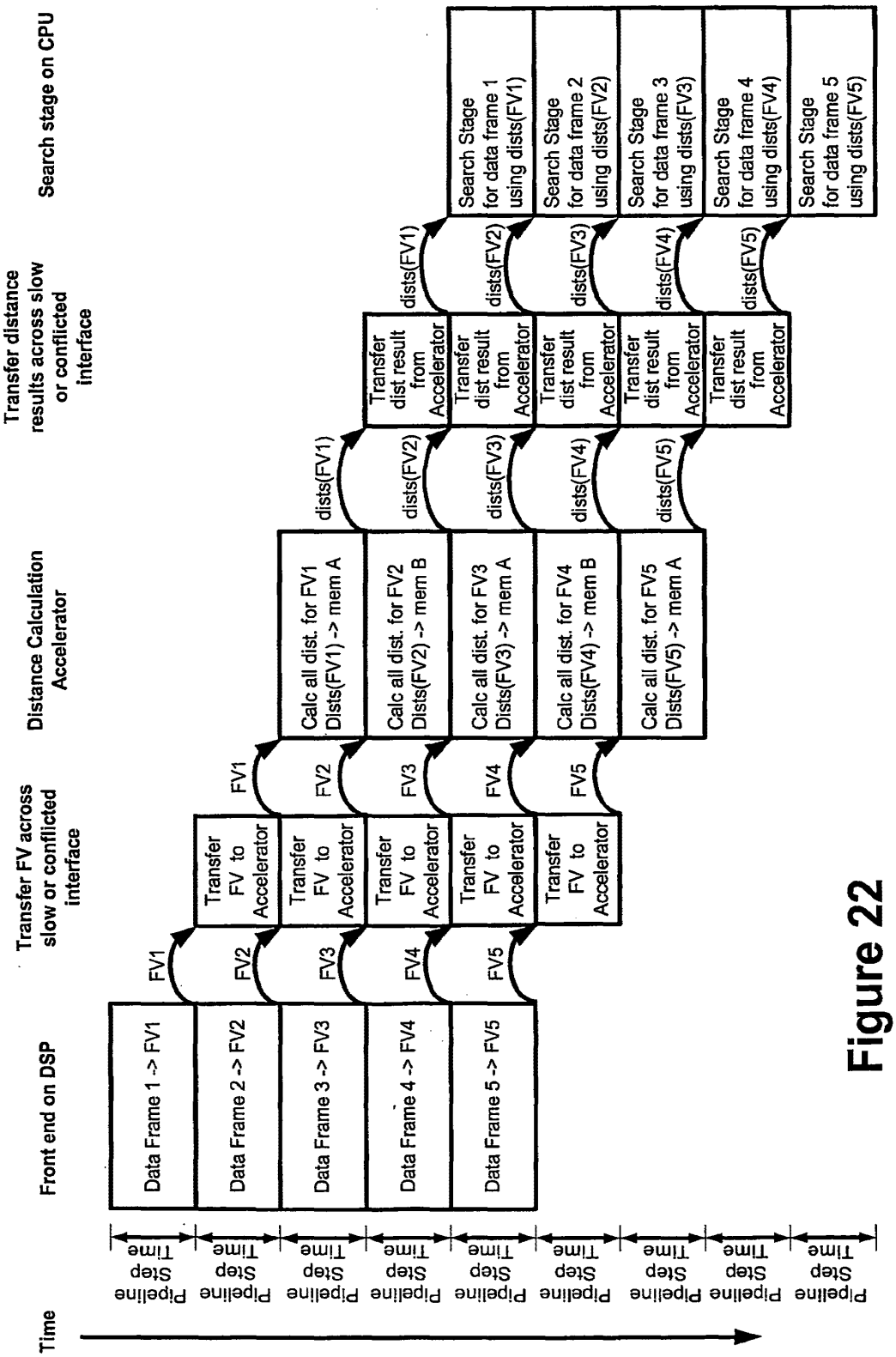
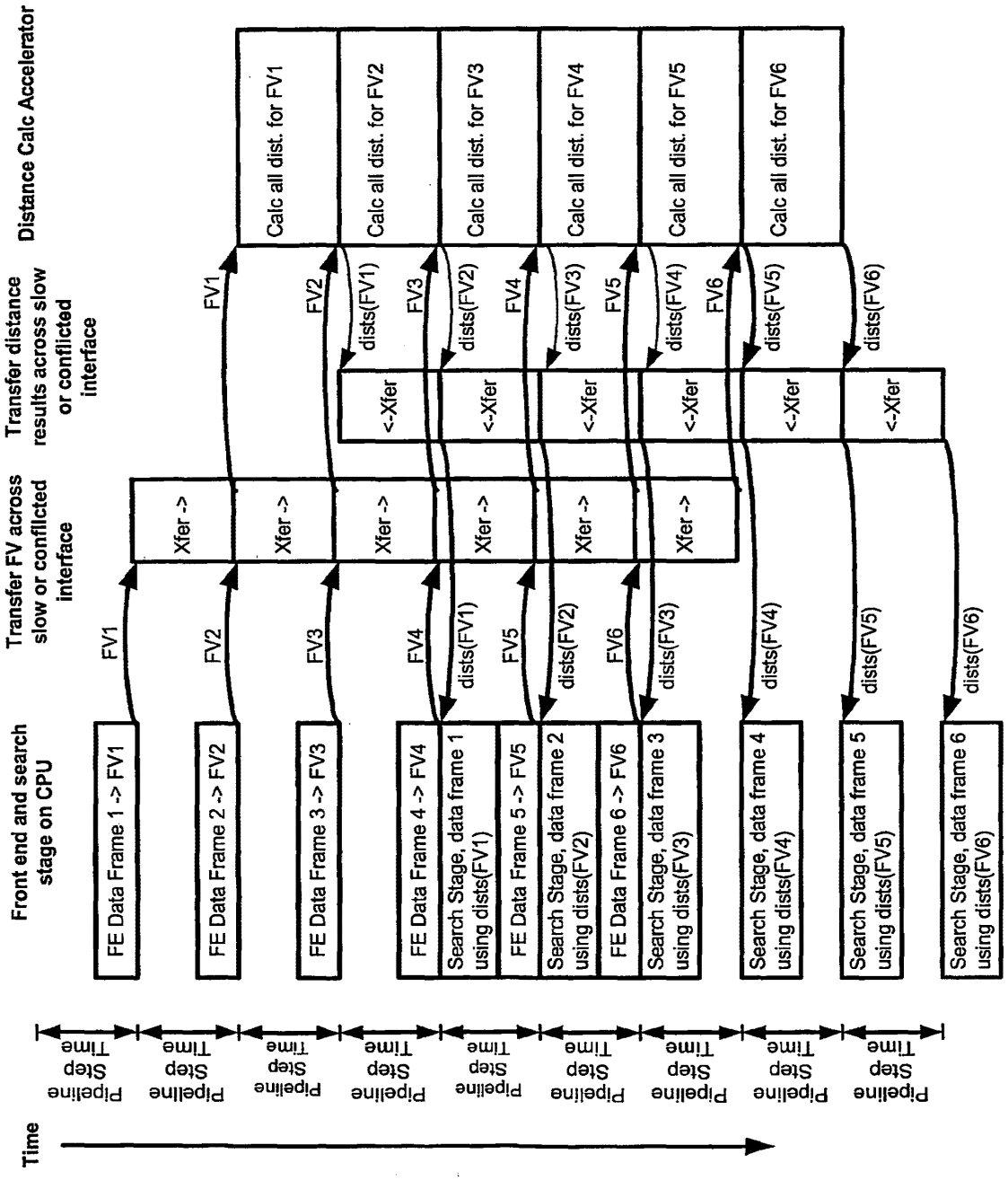


Figure 22

Amazon / Zentian Limited
Exhibit 1002
Page 170

Figure 23



SUBSTITUTE SHEET (RULE 26)

11/662704
IAP8 Rec'd PCT/PTO 14 MAR 2007

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled:

A SPEECH RECOGNITION CIRCUIT AND METHOD

the specification of which was filed on September 14, 2005 as International Application No. PCT/GB2005/003554.

In the event that the filing date and/or Application No. are not entered above at the time I execute this document, and if such information is deemed necessary, I hereby authorize and request my attorneys/agent(s) at **Dickstein Shapiro LLP**, 1825 Eye Street, NW, Washington, DC 20006-5403, to insert above the filing date and/or Application No. of said application.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by an amendment, if any, specifically referred to herein.

I acknowledge the duty to disclose all information known to me that is material to patentability in accordance with Title 37, Code of Federal Regulations, § 1.56.

FOREIGN PRIORITY CLAIM

I hereby claim foreign priority benefits under Title 35, United States Code § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

[ ] no such foreign applications have been filed

[x] such foreign application have been filed as follows:

Amazon / Zentian Limited
Exhibit 1002
Page 172

EARLIEST FOREIGN APPLICATION(S), IF ANY FILED WITHIN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

Application Number	Country	Date of Filing	Priority Claimed Under 35 USC 119
0420464.0	United Kingdom	September 14, 2004	x Yes No _

ALL FOREIGN APPLICATION(S), IF ANY FILED MORE THAN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

Application Number	Country	Date of Filing

CLAIM FOR BENEFIT OF EARLIER U.S. PROVISIONAL APPLICATIONS

I hereby claim priority benefits under Title 35, United States Code §119(e), of any United States provisional patent application(s) listed below:

[x] no such U.S. provisional applications have been filed.

[ ] such U.S. provisional application have been filed as follows:

Application Number	Date of Filing	Priority Claimed Under 35 USC 119

CLAIM FOR BENEFIT OF EARLIER U.S./PCT APPLICATION(S)

I hereby claim the benefit under Title 35, United States Code, §120 of the United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose all information that is material to patentability in accordance with Title 37, Code of Federal Regulations, §1.56 which became available to me between the filing date of the prior application and the national or PCT international filing date of this application:

Amazon / Zentian Limited
Exhibit 1002
Page 173

[x] no such U.S./PCT applications have been filed.

[ ] such U.S./PCT application have been filed as follows:

Application Number	Relationship	Parent Application	Date of Filing

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

I hereby appoint:

All practitioners at Customer Number 24998

all of **Dickstein Shapiro LLP**, 1825 Eye Street, NW, Washington, DC 20006-5403, jointly, and each of them severally, my attorneys at law/patent agent(s), with full power of substitution, delegation and revocation, to prosecute this application, to make alterations and amendments therein, to receive the patent, and to transact all business in the U. S. Patent and Trademark Office connected therewith.

The undersigned hereby authorizes the U.S. attorney or agent named herein to accept and follow instructions from Zentian Limited as to any action to be taken in the United States Patent and Trademark Office regarding this application without direct communication between the U.S. attorney or agent and the undersigned. In the event of a change in the persons from whom instructions may be taken, the U.S. attorney or agent named herein will be so notified by the undersigned.

Please mail all correspondence to Stephen A. Soffen, whose address is:

Dickstein Shapiro LLP
1825 Eye Street, NW
Washington, DC 20006-5403

Please direct telephone calls to: Stephen A. Soffen at (202) 420-4879.

Please direct facsimiles to: (202) 420-2201

Amazon / Zentian Limited
Exhibit 1002
Page 174

Full name of 1st inventor, if any	
Guy Larri	
Fourth inventor's signature	Date
	12 March 2007
Residence	
Cambridge, United Kingdom	
Citizenship United Kingdom	
Mailing Address 1 Cutter Ferry Lane Cambridge Cambridgeshire CB4 1JR UNITED KINGDOM	

Amazon / Zentian Limited
Exhibit 1002
Page 175

Full name of sole or 2nd inventor Mark Catchpole	
Sole or first inventor's signature	Date 14/3/07
Residence Cambridge, United Kingdom	
Citizenship United Kingdom	
Mailing Address Care of Zentian Ltd. St. John's Innovation Centre Cowley Road Cambridge CB4 0WS UNITED KINGDOM	

Amazon / Zentian Limited
Exhibit 1002
Page 176

Full name of 3rd inventor, if any Damian Kelly Harris-Dowsett	
Second inventor's signature	Date 12th March 07
Residence ENGLAND	
Citizenship BRITISH	
Mailing Address 22 Hollis Road Coventry, CV3 1AL United Kingdom.	

Amazon / Zentian Limited
Exhibit 1002
Page 177

Full name of 4th inventor, if any Timothy Brian Reynolds	
Third inventor's signature T. Reynolds	Date 13/2/07
Residence Cambridge, United Kingdom	
Citizenship United Kingdom	
Mailing Address 22 Camside, Cambridge CB4 1PQ	

Amazon / Zentian Limited
Exhibit 1002
Page 178

From the INTERNATIONAL BUREAU

PCT

NOTIFICATION CONCERNING
SUBMISSION OR TRANSMITTAL
OF PRIORITY DOCUMENT

(PCT Administrative Instructions, Section 411)

To:

COLLINS, John, David
Marks & Clerk
90 Long Acre
London WC2E 9RA
ROYAUME-UNI

Date of mailing *(day/month/year)* 02 December 2005 (02.12.2005)	
Applicant's or agent's file reference WPP90312	**IMPORTANT NOTIFICATION**
International application No. PCT/GB2005/003554	International filing date *(day/month/year)* 14 September 2005 (14.09.2005)
International publication date *(day/month/year)*	Priority date *(day/month/year)* 14 September 2004 (14.09.2004)
Applicant ZENTIAN LIMITED et al	

1. By means of this Form, which replaces any previously issued notification concerning submission or transmittal of priority documents, the applicant is hereby notified of the date of receipt by the International Bureau of the priority document(s) relating to all earlier application(s) whose priority is claimed. Unless otherwise indicated by the letters "NR", in the right-hand column or by an asterisk appearing next to a date of receipt, the priority document concerned was submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b).

2. *(If applicable)* The **letters "NR"** appearing in the right-hand column denote **a priority document which, on the date of mailing of this Form, had not yet been received by the International Bureau** under Rule 17.1(a) or (b). Where, under Rule 17.1(a), the priority document must be submitted by the applicant to the receiving Office or the International Bureau, but the applicant fails to submit the priority document within the applicable time limit under that Rule, **the attention of the applicant is directed** to Rule 17.1(c) which provides that no designated Office may disregard the priority claim concerned before giving the applicant an opportunity, upon entry into the national phase, to furnish the priority document within a time limit which is reasonable under the circumstances.

3. *(If applicable)* **An asterisk (*)** appearing next to a date of receipt, in the right-hand column, denotes **a priority document submitted or transmitted to the International Bureau but not in compliance with Rule 17.1(a) or (b)** (the priority document was received after the time limit prescribed in Rule 17.1(a) or the request to prepare and transmit the priority document was submitted to the receiving Office after the applicable time limit under Rule 17.1(b)). Even though the priority document was not furnished in compliance with Rule 17.1(a) or (b), the International Bureau will nevertheless transmit a copy of the document to the designated Offices, for their consideration. In case such a copy is not accepted by the designated Office as the priority document, Rule 17.1(c) provides that no designated Office may disregard the priority claim concerned before giving the applicant an opportunity, upon entry into the national phase, to furnish the priority document within a time limit which is reasonable under the circumstances.

Priority date	Priority application No.	Country or regional Office or PCT receiving Office	Date of receipt of priority document
14 September 2004 (14.09.2004)	0420464.0	GB	11 November 2005 (11.11.2005)

The International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland

Facsimile No. +41 22 338 82 70

Authorized officer
N. Lindner
Facsimile No. (41-22) 338.89.65
Telephone No. +41 22 338 9720

IAP8 Rec'd PCT/PTO 14 MAR 2007 11/662704

Docket No.: M0025.0369/P369
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Guy Larri et al.

Application No.: Not Yet Assigned | Confirmation No.: N/A

Filed: March 14, 2007 | Art Unit: N/A

For: A SPEECH RECOGNITION CIRCUIT AND METHOD | Examiner: Not Yet Assigned

FIRST PRELIMINARY AMENDMENT

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

INTRODUCTORY COMMENTS

Prior to examination on the merits, please amend the above-identified U.S. patent application as follows:

Amendments to the Specification begin on page 2 of this paper.

In the Drawings begin on page 3.

Amendments to the Claims are reflected in the listing of claims which begins on page 8 of this paper.

Remarks/Arguments begin on page 18 of this paper.

Amazon / Zentian Limited
Exhibit 1002
Page 180

AMENDMENTS TO THE SPECIFICATION

On page 8, between the fourth full paragraph and the last paragraph, add the following paragraphs:

A further aspect of the invention comprises a speech recognition circuit, comprising: an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame; calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words; a buffer memory between the calculating circuit and the search stage, for receiving data passing from the calculating circuit to the search stage, wherein a processor in the search stage has higher bandwidth and/or lower latency access to the buffer compared to the bandwidth and/or latency of direct transfer between the calculating circuit and the search stage. The data transfer from the calculating circuit to the buffer memory and/or from the buffer memory to the search stage may be performed as one or more sequential bursts. The data transfer to the buffer memory may be performed in parallel with data transfer to the calculating circuit and/or in parallel with data transfer to the search stage. A second buffer memory may be provided between the audio front end and the calculating circuit.

A further aspect of the invention comprises a speech recognition circuit, comprising: an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame; calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words; comprising an elastic buffer between at least one of the front end and calculating circuit, or the calculating circuit and search stage, and/or for buffering said audio signal.

A further aspect of the invention comprises an accelerator for a speech recognition circuit, the accelerator comprising: calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; means for comparing a first version of a stored checksum of data representing said acoustic model and a second version of said stored checksum, wherein the second version is obtained from an updated measurement and calculation of the checksum; and means for indicating an error status if the checksums do not match.

A further aspect of the invention comprises an accelerator for a speech recognition circuit, the accelerator comprising: calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; wherein said accelerator is configured to autonomously compute distances for every acoustic state defined by the acoustic model.

A further aspect of the invention comprises an accelerator for calculating distances for a speech recognition circuit, the accelerator comprising: calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; first and second storage circuit, which may be referred to as result memories, each for storing calculated distances for at least one said audio time frame, and for making said stored distances available for use by another part of the speech recognition circuit; control circuit for controlling read and write access to the first and second storage circuit, said control means being configured to allow writing to one said storage means while the other said storage means is available for reading, to allow first calculated distances for one audio time frame to be written to one said storage means while second calculated distances for an earlier audio time frame are made available for reading from the other said storage means.

Embodiments of the invention may comprise means for generating a checksum or computed signature for the acoustic model data stored in the memory, and means for comparing checksums or computed signatures that have been calculated at different times, to indicate an error status if the checksums do not match, one possible cause of such mismatch being that the acoustic model data has been overwritten by said other data and said error status being used to indicate that the acoustic model should be re-loaded into the said memory.

A further aspect of the invention comprises a speech recognition circuit comprising: lexical memory containing lexical data for word recognition, said lexical data comprising a lexical tree data structure comprising a model of words; means for accessing a state model corresponding to each phone or each group of phones in the lexical tree; a content addressable memory for storing content addressable data for each phone or group of phones, including states corresponding to said phone or group of phones, and for storing an address value for each said phone or group of phones; a RAM configured to store accumulated scores for each said phone or group of phones, the accumulated scores being addressable by said address value for each said phone or group of phones; means to obtain scores that each of a plurality of frames of an audio signals corresponds to each of a plurality of said states; a counter to sequentially search for each said state in the content addressable memory, to obtain the corresponding address value if the state is found in the content addressable memory; means to use said address value to access an accumulated likelihood and an accumulator to add said likelihood to the accumulated likelihood; means to use the phones or groups of phones with the highest accumulated scores to obtain a plurality of next phones from the lexical tree which correspond to the next phone; and output means for outputting a lexical tree path of highest likelihood.

A further aspect of the invention comprises speech recognition apparatus comprising: a lexical tree having a corresponding state model; means for obtaining scores of an audio input corresponding to each of a plurality of states in said state model; a content addressable memory for storing a marker indicating a part of the lexical tree, and one or more states associated with said part of the lexical tree; a random access memory addressable by the CAM output, to output accumulated scores for states corresponding to said parts of the lexical tree; adder means for adding likelihood to said accumulated likelihood, to be stored back in the RAM.

A further aspect of the invention comprises speech recognition apparatus, comprising: a CAM-RAM arrangement for storing records including pointers to a lexical tree, and accumulative scores for states within the lexical tree; input means for obtaining scores that an audio frame corresponds to a particular state in the lexical tree; an accumulator for calculating the updated scores and modifying the records in the CAM-RAM accordingly; output means for outputting a path of highest likelihood in the lexical tree.

A further aspect of the invention comprises a speech recognition method comprising: storing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and scores corresponding to said state identifiers in a memory structure, the lexical tree comprising a model of words and the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; repeating the following sequence of steps for each of a plurality of incoming frames of an audio signal; obtaining score updates corresponding to the likelihoods that said frame of the audio signal corresponds to each of a plurality of said states; accessing said memory structure to obtain scores, updating the scores by adding score updates to the scores, and writing back the updated scores to the memory structure; determining if scores for states furthest on in the lexical tree correspond to a significant likelihood, and if so, then accessing the lexical tree to determine the next set of possible states; and storing the next set of possible states and said scores of significant likelihood in the memory structure.

A further aspect of the invention comprises speech recognition circuit comprising: a circuit for providing state identifiers which identify states corresponding to phones or groups of adjacent phones in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers and phone instance identifiers uniquely identifying instances of phones or groups of phones in the lexical tree; said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving

scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one phone instance identifier according to said scores.

On page 16, on the paragraph abridging pages 16 and 17, please amend the text (specifically the term "search stage" o line 8) as follows:

The distance calculator stage of the recognition process computes a probability or likelihood that a feature vector corresponds to a particular state. Each state has an associated probability distribution in the N-dimensional space. The distance calculator computes the distance in the N-dimensional space from the Feature Vector to the probability distribution for each state. In some embodiments, the calculation is performed only when the search stage algorithms determine that the distance for that particular state is needed by the search_stage. In some embodiments, the distances for all of the states are calculated and made available to the search stage, which allows pipelined operation. Embodiments that do the calculations on demand are preferable.

AMENDMENTS TO THE DRAWINGS

Please substitute the attached replacement sheet of Figure 16 for the originally filed drawing of Figure 16.

Amazon / Zentian Limited
Exhibit 1002
Page 186

AMENDMENTS TO THE CLAIMS

1. (Original) A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

a calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating circuit is implemented using a second processor, and wherein data is pipelined from the front end to the calculating circuit to the search stage.

2. (Original) A speech recognition circuit as claimed in claim 1, wherein the pipelining comprises alternating of front end and search stage processing on the first processor.

3. (Original) A speech recognition circuit as claimed in claim 1, comprising dynamic scheduling whether the first processor should run the front end or search stage code, based on availability or unavailability of distance results and/or availability of space for storing more feature vectors and/or distance results.

4. (Original) A speech recognition circuit as claimed in claim 1, wherein the first processor supports multi-threaded operation, and runs the search stage and front ends as separate threads.

5. (Currently Amended) ~~A speech recognition circuit comprising:~~

~~an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;~~

~~calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and~~

~~a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;~~

~~a buffer memory between the calculating circuit to the search stage, wherein a processor in the search stage has higher bandwidth and/or lower latency access to the buffer compared to the bandwith and/or latency of direct transfer between the calculating circuit and the search stage~~

A speech recognition circuit as claimed in claim 1, wherein the said calculating circuit is configured to autonomously calculate distances for every acoustic state defined by the acoustic model.

Claim 6-9 (Canceled).

10. (Original) An accelerator for calculating distances for a speech recognition circuit, the accelerator comprising:

calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model,

Amazon / Zentian Limited
Exhibit 1002
Page 188

wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

first and second storage circuit, each for storing calculated distances for at least one said audio time frame, and for making said stored distances available for use by another part of the speech recognition circuit;

control circuit for controlling read and write access to the first and second storage circuit, said control means being configured to allow writing to one said storage means while the other said storage means is available for reading, to allow first calculated distances for one audio time frame to be written to one said storage means while second calculated distances for an earlier audio time frame are made available for reading from the other said storage means.

11. (Original) An accelerator as claimed in claim 10, wherein the control means is configured to alternate which of the first and second storage means is available for reading, and which of the first and second storage means is available for writing, as each new feature vector is processed by the calculating circuit.

12. (Original) The accelerator of claim 10, wherein the distance is stored for each of a plurality of states.

13. (Currently Amended) The accelerator of ~~any one of claims 10 to 12~~ claim 10, wherein the first and second storage means are interfaced to a processor in the speech recognition circuit as different memory locations within the processor's memory map.

14. (Currently Amended) The accelerator of ~~any one of claims 10 to 12~~ claim 10, wherein the first and second storage means are interfaced to a processor in the

Amazon / Zentian Limited
Exhibit 1002
Page 189

speech recognition circuit as being ~~each one~~ mapped to ~~the same fixed set of memory addresses when configured for reading, so that when the processor performs a read operation from a memory location within the said set of memory locations it will read data from whichever of the storage means is currently configured for reading~~ a first fixed memory address when configured for reading, and/or to a second fixed memory address when configured for writing.

15. (Currently Amended) The accelerator of ~~any one of claims 10 to 14~~ claim 10, comprising releasing a result memory after the search stage has processed the contents, by informing the accelerator that it can overwrite the results in said memory.

16. (Currently Amended) The accelerator of ~~any one of claims 10 to 15~~ claim 10, comprising additional result memories, where at any time at most one result memory is configured for writing, and at least one result memory is configured for reading.

17. (Currently Amended) The accelerator of ~~any one of claims 10 to 14~~ claim 10, comprising random access of the distance results in the result memories, as they are needed by the search stage.

18. (Currently Amended) The accelerator of ~~any one of claims 10 to 17~~ claim 10, wherein the result memories are accessed in a round-robin fashion.

19. (Original) A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model,

Amazon / Zentian Limited
Exhibit 1002
Page 190

wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

means for receiving or storing compressed data representing said acoustic model;

a decompressor for decompressing said compressed data for all states or selected states of the acoustic model, wherein the decompressed data is sent to the calculating means; and

output means for outputting calculated distances to another part of the speech recognition circuit.

20. (Currently Amended) The accelerator of claim 19 ~~or 20~~, wherein the decompression scheme comprises one or more of the following:

sign or zero extension or otherwise conversion of narrow or variable width data to a wider data format;

sign or zero extension or otherwise conversion of narrow or variable width data to IEEE standard single or double precision floating point format;

codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data;

decompression of a Huffman or Lempel-Ziv compressed stream;

decompression of run length encoded data;

decompression of difference encoded data; and

any well known compression scheme.

21. (Currently Amended) The accelerator of claim 19 ~~or claim 20~~, wherein the decompressor is configured to operate with an acoustic model that uses subspace distribution clustering.

22. The accelerator of ~~any one of claims 19 to 21~~ claim 19, wherein acoustic states are decompressed one or more times for each feature vector processed by the accelerator.

23. (Currently Amended) The accelerator of ~~claim 22, wherein the computed signature is a cyclic redundancy check (CRC) signature~~ claim 19, wherein the said calculating circuit is configured to autonomously calculate distances for every acoustic state defined by the acoustic model.

Claims 24 to 25 (Canceled).

26. (Currently Amended) An accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; and

a memory for storing said acoustic model,

wherein the calculating means and the memory ~~comprised part of the same physical circuit~~ are fabricated as circuits on a single integrated circuit .

27. (Canceled).

28. (Currently Amended) An accelerator according to claim ~~26 or~~ 27, wherein said memory is configured to be ~~shared between the accelerator and another part of the speech recognition circuit~~ used to hold other data not related to speech recognition during periods when speech recognition is not active.

29. (Original) An accelerator according to claim 22, further comprising means for generating a checksum or computed signature for the acoustic model data stored in the memory, and means for comparing checksums or computed signatures that have been calculated at different times, to indicate an error status if the checksums do not match.

30. (Currently Amended) An accelerator according to ~~any one of claims 26 to 29~~ claim 26, wherein the acoustic model memory is a RAM or flash memory.

31. (Canceled).

32. (Currently Amended) The accelerator of ~~any one of the preceding claims~~ claim 26, being provided in the form of a separate physical circuit, device or package that is removably connectable to said speech recognition circuit.

33. (Currently Amended) A speech recognition circuit comprising the accelerator of ~~any previous~~ claim 26.

34. (Currently Amended) The speech recognition circuit of claim 33, comprising a shared processor for the front end and search stages, where said shared processor does not perform a majority of the distance calculations.

35. (Currently Amended) The speech recognition circuit of claim 33, comprising a digital signal processor for the front end, and a general purpose microprocessor for the search stage.

36. (Currently Amended) The speech recognition circuit of ~~any one of the previous claims~~ claim 1, comprising control means adapted to implement frame dropping, to discard one or more audio time frames.

37. (Currently Amended) The speech recognition circuit of ~~any one of the previous claims~~ claim 1, wherein the feature vector comprises a plurality of spectral components of an audio signal for a predetermined time frame.

38. (Currently Amended) The speech recognition circuit of ~~any one of the previous claims~~ claim 1, wherein the processor is configured to divert to another task if the data flow stalls.

39. (Currently Amended) The speech recognition circuit of ~~any one of the previous claims~~ claim 1, wherein the speech accelerator has an interrupt ~~and status bit that can be polled to indicate the state of the feature vector memory~~ signal to inform the front end that the accelerator is ready to receive a next feature vector from the front end.

40. (Currently Amended) The speech recognition circuit of ~~any one of the previous claims~~ claim 1, wherein the accelerator signals to the search stage when distances for a new frame are available in a result memory.

41. (Canceled).

42. (Currently Amended) The speech recognition circuit of ~~any one of the previous claims~~ claim 1, comprising increasing the pipeline depth by computing extra front end frames in advance.

43. (Original) A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end, said calculating means, and said search stage are connected to each other to enable pipelined data flow from one to another.

44. (Currently Amended) A speech recognition circuit as ~~claimed in any one of the previous claims~~ claim 1, wherein the audio front end is configured to input a digital audio signal.

45. (Original) A speech recognition method, comprising:

calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating means is implemented using a second processor, and wherein data is pipelined from the front end to the calculating means to the search stage.

46. (Currently Amended) A method as implemented by the apparatus of ~~any previous~~ claim 1.

Claims 47-84. (Canceled).

85. (New) The accelerator of claim 10, wherein the said calculating circuit is configured to autonomously calculate distances for every acoustic state defined by the acoustic model

86. (New) A speech recognition circuit of claim 1, wherein said distance comprises a Mahalanobis distance.

REMARKS

By this Preliminary Amendment, the claims have been reduced in number and multiple dependencies have been eliminated.

The specification has been amended to correct minor typographical errors and to add text corresponding to the canceled claims. A substitute specification incorporating the changes is submitted herewith.

The drawings have been amended to add the reference numeral 108 and a line connecting it to the "active state list" on Fig. 16.

Prompt examination is earnestly solicited.

Dated: March 14, 2007

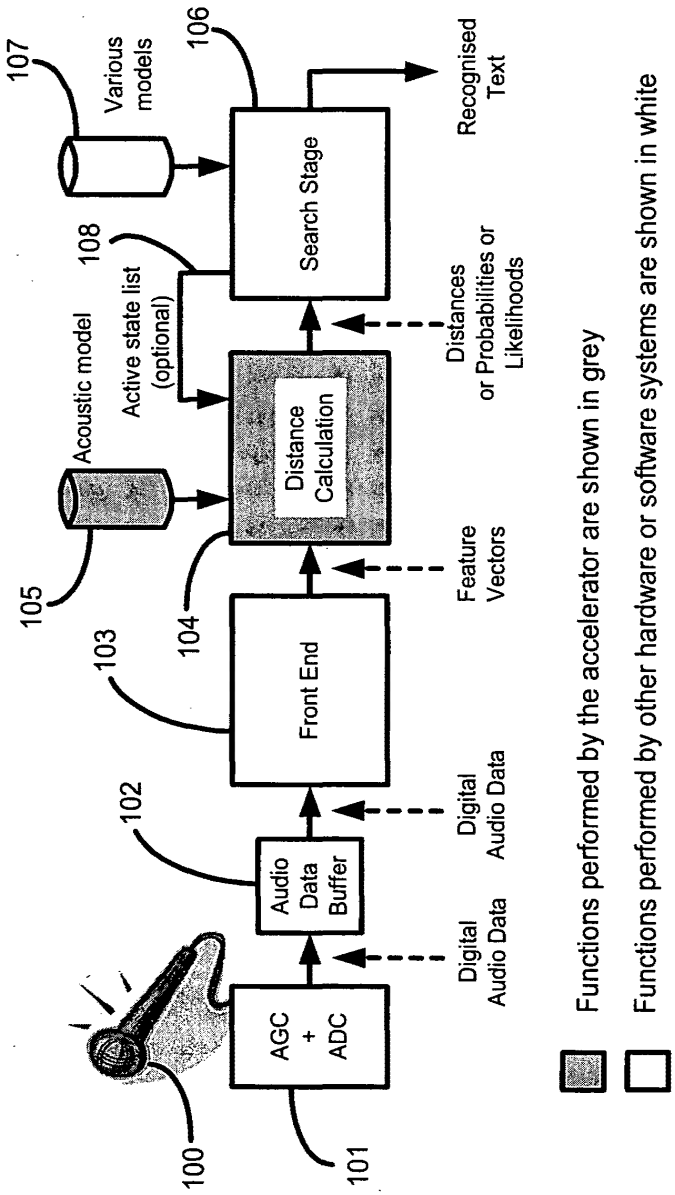
Respectfully submitted,

By _____

Stephen A. Soffen
Registration No.: 31,063
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
(202) 420-2200
Attorney for Applicant

Amazon / Zentian Limited
Exhibit 1002
Page 197

Figure 16


100
101
AGC + ADC
Digital Audio Data
102
Audio Data Buffer
Digital Audio Data
103
Front End
Feature Vectors
104
105
Acoustic model
Distance Calculation
108
Active state list (optional)
Distances or Probabilities or Likelihoods
107
Various models
106
Search Stage
Recognised Text
Functions performed by the accelerator are shown in grey
Functions performed by other hardware or software systems are shown in white

IAP8 Rec'd PCT/PTO 14 MAR 2007 11/662704

Docket No.: M0025.0369/P369
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Guy Larri et al.

Application No.: Not Yet Assigned | Confirmation No.: N/A

Filed: Concurrently Herewith | Art Unit: N/A

For: A SPEECH RECOGNITION CIRCUIT AND METHOD | Examiner: Not Yet Assigned

INFORMATION DISCLOSURE STATEMENT (IDS)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Pursuant to 37 CFR 1.56, 1.97 and 1.98, the attention of the Patent and Trademark Office is hereby directed to the references listed on the attached PTO/SB/08. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the references be made of record therein and appear among the "References Cited" on any patent to issue therefrom.

This Information Disclosure Statement accompanies the new patent application submitted herewith.

Copies of the references on the PTO/SB/08 are not provided. The non-patent literature documents should have been provided by the International Bureau.

In accordance with 37 CFR 1.97(g), the filing of this Information Disclosure Statement shall not be construed to mean that a search has been made or that no other material information as defined in 37 CFR 1.56(a) exists. In accordance with 37 CFR 1.97(h), the filing of this Information Disclosure Statement shall not be construed to be an admission that any patent, publication or other information referred to therein is "prior art" for this invention unless specifically designated as such.

It is submitted that the Information Disclosure Statement is in compliance with 37 CFR 1.98 and the Examiner is respectfully requested to consider the listed references.

The Director is hereby authorized to charge any deficiency in the fees filed, asserted to be filed or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our Deposit Account No. 04-1073, under Order No. M0025.0369/P369. A duplicate copy of this paper is enclosed.

Dated: March 14, 2007

Respectfully submitted,

By _____

Stephen A. Soffen
Registration No.: 31,063
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
(202) 420-2200
Attorney for Applicant

11/662704
IAP8 Rec'd PCT/PTO 14 MAR 2007

PTO/SB/08A/B (09-06)
Approved for use through 03/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	Not Yet Assigned
Filing Date	Concurrently Herewith
First Named Inventor	Guy Larri
Art Unit	N/A
Examiner Name	Not Yet Assigned
Attorney Docket Number	M0025.0369/P369

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number: Number-Kind Code[2] (*if known*)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	AA*	5,881,312	03-09-1999	Dulong	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3]-Number[4]-Kind Code[5] (*if known*)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. * CITE NO.: Those application(s) which are marked with an single asterisk (*) next to the Cite No. are not supplied (under 37 CFR 1.98(a)(2)(iii)) because that application was filed after June 30, 2003 or is available in the IFW. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
	CA	Glinski et al., " Spoken Language Recognition on a DSP Array Processor" IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 7, pp. 697-703, 1 July 1994.	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1]Applicant's unique citation designation number (optional). [2]Applicant is to place a check mark here if English language Translation is attached.

Examiner Signature		Date Considered	

DocCode - SCORE

SCORE Placeholder Sheet for IFW Content

Application Number: **11662704** Document Date: **03/14/2007**

The presence of this form in the IFW record indicates that the following document type was received in paper and is scanned and stored in the SCORE database.

- Drawings

Images of the original documents are scanned in gray scale or color and stored in SCORE. Bi-tonal images are also stored in IFW. Defects visible in both IFW and SCORE are indicative of defects in the original paper documents.

To access the documents in the SCORE database, refer to instructions developed by SIRA.

At the time of document entry (noted above):

- Examiners may access SCORE content via the eDAN interface.
- Other USPTO employees can bookmark the current SCORE URL (http://es/ScoreAccessWeb/).
- External customers may access SCORE content via the Public and Private PAIR interfaces.

Form Revision Date: December 8, 2006

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 March 2006 (23.03.2006)

PCT

(10) International Publication Number
WO 2006/030214 A2

(51) International Patent Classification:
G10L 15/28 (2006.01) *G10L 15/10* (2006.01)

(21) International Application Number:
PCT/GB2005/003554

(22) International Filing Date:
14 September 2005 (14.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0420464.0 14 September 2004 (14.09.2004) GB

(71) Applicant *(for all designated States except US)*: **ZENTIAN LIMITED** [GB/GB]; St. John's Innovation Centre, Cowley Road, Cambridge, Cambridgeshire CB4 0WS (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CATCHPOLE, Mark** [—/GB]; Ash House, Breckenwood Road, Fulbourn, Cambridge, Cambridgeshire CB1 5DQ (GB). **DOWSETT, Damian, Harris**; * (**). **REYNOLDS, Timothy**; * (**). **LARRY, Guy** [—/GB]; 1 Cutter Ferry Lane, Cambridge, Cambridgeshire CB4 1JR (GB).

(74) Agent: **COLLINS, John, David**; Marks & Clerk, 90 Long Acre, London WC2E 9RA (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A SPEECH RECOGNITION CIRCUIT AND METHOD

(57) Abstract: A speech recognition circuit comprising a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

WO 2006/030214 A2

A Speech Recognition Circuit and Method

The present invention relates to speech recognition circuits and methods. These circuits and methods have wide applicability, particularly for devices such as mobile electronic devices.

There is growing consumer demand for embedded speech recognition in mobile electronic devices, such as mobile phones, dictation machines, PDAs (personal digital assistants), mobile games consoles, etc. For example, email and text message dictation, note taking, form filling, and command and control applications are all potential applications of embedded speech recognition.

However, when a medium to large vocabulary is required, effective speech recognition for mobile electronic devices has many difficulties not associated with speech recognition systems in hardware systems such as personal computers or workstations. Firstly, the available power in mobile systems is often supplied by battery, and may be severely limited. Secondly, mobile electronic devices are frequently designed to be as small as practically possible. Thus, the memory and resources of such mobile embedded systems tends to be very limited, due to power and space restrictions. The cost of providing extra memory and resources in a mobile electronic device is typically much higher than that for a less portable device without this space restriction. Thirdly, the mobile hardware may be typically used in a noisier environment than that of a fixed computer, e.g. on public transport, near a busy road, etc. Thus, a more complex speech model and more intensive computation may be required to obtain adequate speech recognition results.

These restrictions have made it difficult to implement effective speech recognition in mobile devices, other than with very limited vocabularies.

Some prior art schemes have been proposed to increase the efficiency of speech recognition systems, in an attempt to make them more suitable for use in mobile technology.

In an article entitled "A low-power accelerator for the SPHINX 3 speech recognition system", in University of Utah, International conference on Compilers, Architectures and Synthesis for Embedded Systems, Nov 2003, Davis et al have proposed the idea of using a special purpose co-processor for up-front calculation of the computationally expensive Gaussian output probabilities of audio frames corresponding to particular states in the acoustic model.

In an article entitled "Hardware Speech Recognition in Low Cost, Low Power Devices", University of California, Berkeley, CS252 Class Project, Spring 2003, Sukun Kim et al describe using special purpose processing elements for each of the nodes in the network to be searched. This effectively implies having a single processing element for each phone in the network. An alternative suggested by Sukun Kim et al is to provide a processor for each state in the network.

In an article entitled "Dynamic Programming Search for Continuous Speech Recognition" in IEEE Signal Processing Magazine, Sept 1999, Ney et al discuss language model lookahead. Language model lookahead involves computation of a language model factor for each node (i.e. phone) in the lexical tree. This technique is also known as smearing. Each phone instance in the search network can be given a language model factor when it is used in the lexical tree search. Ney et al show that for an example bigram language model, the average number of states per 10 ms frame can be reduced from around 168,000 states with no language model lookahead to around 8,000 states when language model lookahead is used. They also show that bigram language model lookahead requires about a quarter of the states compared with unigram language model lookahead.

Although these prior art documents provide improvements to speech recognition in embedded mobile technology, further improvement is still needed to provide a larger vocabulary and better accuracy.

One aspect of the present invention provides a speech recognition circuit including a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers. The lexical tree includes a model of words. The speech recognition circuit also has a memory structure for receiving and storing state identifiers identified by a node identifier identifying nodes or groups of adjacent nodes, the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores. An accumulator is provided for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores. A selector circuit is used for selecting at least one node or group of nodes of the lexical tree according to said scores.

One suitable type of hardware for the memory structure includes a content addressable memory (CAM). A CAM is a memory unit which stores a series of data items using a series of addresses. However, the memory is accessed by specifying a data item, such that the CAM returns the corresponding address. This contrasts with a random access memory (RAM) in which the memory is accessed by specifying an address, such that the RAM returns the corresponding data item.

However, the memory structure is not limited to including a CAM. Other types of hardware are also possible, to provide this functionality. For example, a single chip which operates in the same way as a CAM and RAM may be used instead.

Embodiments of the present invention provide a solution to the problem of how to map a lexical tree search to a CAM system architecture. The realisation by the present inventors that certain speech recognition data structures can be mapped into the CAMs allows a lexical tree search to be performed using a CAM system architecture.

Further embodiments of the invention include a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

The node identifier may comprise a direct reference to the lexical tree. However, in some embodiments, the node identifier for at least some of the states includes a pointer to a node identifier for another state. For example, a state corresponding to the furthest part of the search path in the lexical tree may be referenced by a node identifier which directly links to a particular node or group of nodes in the lexical tree. In a lexical tree comprising phones, using a state model of triphones, the node identifier may indicate the position of a triphone in the lexical tree.

However, in this example, for states occuring further back in the search path, instead of supplying a node identifier linking directly to the lexical tree, instead a pointer to a node identifier of another state may be supplied. E.g. a triphone instance may have a pointer to another triphone instance, which has a pointer to another triphone instance, which has a pointer to a node or group of nodes in the lexical tree. Chains of reference may be set up in this way, where only the last state in the chain has a direct pointer to the lexical tree.

There may not be a one-to-one correspondence between the nodes of the lexical tree and the node identifiers. This will occur for a branched lexical tree, where the nodes represent monophones, but the acoustic model states represent triphones, i.e. groups of three adjacent monophones. Then, paths of three monophones will have unique identifiers to be stored in the memory structure, rather than single monophones having unique identifiers.

Phone instance numbers may be generated, and used to uniquely label each phone instance. They can be generated sequentially, using a counter. The phone instance numbers may be used as pointers between phone instances to assist in node identification. It is thus not essential to provide a direct node identifier for each phone instance to directly indicate a location in the lexical tree. The dynamic network of phone

instances provided in the memory structure may thus include both direct and relative references to the lexical tree.

The memory structure may be divided into one part which stores phone instance identifiers and direct references to the lexical tree, and a second part which stores phone instance identifiers and corresponding states. This can speed up the processing, by only storing the phone instances which are furthest on in the lexical tree in the first part of the memory structure.

The memory structure may also be divided into separately accessable units, to reduce the amount to data in each unit, thereby decreasing the chance of finding the same two states identifiers in different phone instances in any single memory unit, and increasing the chance of some state identifiers being completely absent from any single memory unit. This makes it easier to deal with the situation when the same two state identifiers are found, because a spare time slot is available for processing when a state identifier is not present.

A further aspect of the invention provides a distance calculation engine within a speech recognition system. The distance calculation engine may be included within an accelerator. The accelerator may include logic to interface with other parts of a speech recognition circuit, in addition to the distance engine, although this is not essential. For example, the accelerator may include one or more results memories for storing distances calculated by the distance calculation engine. The accelerator may also include at least one of a memory for storing one or more acoustic models, a decompressor for decompressing acoustic data that has been stored in a compressed format, a memory for storing feature vectors, a checksum or data signature calculation means, buffers for data storage, and data registers. The accelerator may be implemented in software or in hardware, or in a combination. It may be physically separate to the rest of the speech recognition circuit, although this is not essential.

The distance calculation engine may calculate one or more of a wide range of distance metrics and probability distributions. The distances may represent the likely correspondance of feature vectors to states in an acoustic model. In other words, the

distances can indicate the similarity of an audio data frame to each possible state in an acoustic model

There are a wide variety of probability distributions that can be used for the distance calculation stage of a speech recogniser, and a wide variety of distance metrics used. These are widely documented in the literature. A point is a simple example of a probability distribution.

A common choice is to use Gaussian Distributions and correspondingly the Mahalanobis Distance metric. The Gaussian probability distribution is then defined by a mean vector, which defines centre point in the N-dimensional space, and a Covariance matrix which defines the shape of the probability distribution. It is common to restrict the Covariance matrix to be a diagonal matrix (only N non-zero values along the diagonal of the NxN matrix) which significantly lowers the implementation cost by reducing the number of arithmetic operations.

In particular embodiments, the distance calculated is a Mahalanobis distance. Particular examples of this are described later in the specification.

In one embodiment, the distance engine autonomously computes all of the distances associated with a given feature vector. This may comprise computing distances for every state in the lexicon. The distance engine may operate in a pipelined manner with other stages of the recognition process. In this context a distance is an indication of the probability or likelihood that a feature vector corresponds to a particular state. An important class of distance computation in speech recognition is the calculation of output state probabilities in recognisers using Hidden Markov Models. Another possible use is in recognisers using Neural Networks.

The distance engine reads data from the acoustic models to use as parameters in the calculation. The acoustic models may be optionally stored in a compressed format. The distance engine may read and de-compress the acoustic models one (or more) times for each feature vector processed. Each reading of the acoustic models may require reading

the entire acoustic model, or various optimisations may be implemented to avoid reading parts of the acoustic model that are not required to be used for calculations with the current feature vector. The distance engine may use a de-compression method where the de-compression is sign or zero extension or may otherwise convert data of narrow or variable width to a wider data format. The distance engine may use a de-compression method where the de-compression is sign or zero extension or may otherwise convert data of narrow or variable width to IEEE standard single or double precision floating point format. The distance engine may use a decompression method where decompression is a codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data. The distance engine may use a decompression method where the decompression is decompression of a Huffman or Lempel-Ziv compressed stream. The distance engine may use a decompression method where decompression is decompression of run length encoded data. The distance engine may use a decompression method where decompression is decompression of difference encoded data. The distance engine may use a decompression method using any well known lossy or lossless compression scheme. The distance engine may use a decompression method using subspace distribution clustering. The distance engine may use a decompression method comprising any combination of the above described decompression types.

The distance engine may read the acoustic models from a dedicated on-chip memory. The distance engine may read the acoustic models from a dedicated off-chip memory. The distance engine may read the acoustic models from a shared on-chip memory. The distance engine may read the acoustic models from a shared off-chip memory. Any of these acoustic models may be compressed.

The distance engine may compute a CRC or checksum or similar signature as it reads in the acoustic model and compares this to a stored CRC, checksum, or signature, in order to check that the acoustic model has not been corrupted, and signals an error condition if such corruption is detected. The stored CRC, Checksum, or signature may had been pre-computed and stored in the model data, or it may be computed at the time the model data is loaded into the Acoustic Model Memory. It may be held in Acoustic Model Memory, or it may be loaded into a register or another memory from where it can be

accessed and compared when the CRC/checksum/signature is computed each time the Acoustic Model is loaded.

The distance engine may support the pass-through of data from the front-end to the search-stage. The data to be passed through will be supplied to the distance engine as an adjunct to the feature vector, and the distance engine will pass it to the search stage as an adjunct to the distance results. This provides a simple mechanism for passing frame-specific data that is not involved in the distance calculation through to the search stage, and keeping it associated with the correct frame, which may otherwise be complex in pipelined systems with multiple processors. The data passed through may be for any purpose. Examples might include silence detected, end-of-audio-stream detected, a frame number, information that some intervening frames have been dropped, or data from another input device such as a button or keyboard in a multi-modal interface.

The distance engine may be implemented in hardware, software, or a combination. Other stages may be implemented in hardware, software, or a combination. The distance engine may be implemented with any number representation format including fixed point or floating point arithmetic, or any mixture of number representation formats.

In particular, the other stages may be implemented on a CPU, or on a DSP and CPU. The "DSP" and "CPU" may each be implemented as software programmable devices.

The distance engine may implement one or more additional pipeline stages to overcome delays introduced by low bandwidth, high latency, or conflicted bus interfaces. The distance engine may also implement additional pipeline stages to maintain the same throughput while allowing more time for each distance calculation. Particular embodiments of the invention may include one or more of the above aspects.

A speech recognition apparatus according to the invention may be embedded in or included in a mobile electronic device such as a mobile telephone, PDA (personal digital assistant), etc.

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of the system architecture for a speech recognition apparatus according to an embodiment of the invention;

Figure 2 is a block diagram showing the main data structures used in the speech recognition apparatus of figure 1;

Figure 3 shows a block diagram of the Single Phone Instance Network Engine (PINE) of figure 1;

Figure 4 is a block diagram showing the Wave Front Manager architecture;

Figure 5 shows the use of link nodes to permit path merging;

Figure 6 shows a block diagram of a Path Merge Unit in the Phone Level;

Figure 7 shows the Phone Book architecture;

Figure 8 shows Lexical tree storage;

Figure 9 shows the data structure for the word link record;

Figure 10 shows a block diagram of Word Engine;

Figure 11 shows a Language Model interface architecture;

Figure 12 shows an example of a simple Loop Grammar;

Figure 13 shows a block diagram of grammar engine;

Figure 14 shows a single frame of the Audio Codec link protocol; and

Figure 15 illustrates the system timing which may be used in the embodiment of figure 1;

Figure 16 is a block diagram showing a speech recognition circuit according to an embodiment of the invention, and illustrating data flow between parts of the circuit;

Figure 17 is a block diagram showing a distance calculation engine according to an embodiment of the invention;

Figure 18 shows ideal data flow over several time steps within a speech recognition circuit having separate front end and search stages, in an embodiment of the invention;

Figure 19 shows an example of actual data flow over several time steps within a speech recognition circuit having separate front end and search stages, in a further embodiment of the invention;

Figure 20 shows ideal data flow over several time steps within a speech recognition circuit having a shared processor for front end and search stages, in an embodiment of the invention;

Figure 21 shows an example of actual data flow over several time steps within a speech recognition circuit having a shared processor for front end and search stages, in an embodiment of the invention;

Figure 22 shows ideal data flow in a speech accelerator pipeline with two data transfer stages, according to an embodiment of the invention; and

Figure 23 shows an example of actual data flow in a speech accelerator pipeline with two data transfer stages according to an embodiment of the invention.

Figure 1 shows a speech recognition system architecture for an embodiment of the invention. The speech recognition system is divided into several levels of operation: a

state level, a phone level, a word level, a grammar level, and a system level. Each level is shown in figure 1 as a grey shaded box, interconnected to other levels of the system.

The information passed between different levels or subcircuits of figure 1 is indicated on the figure 1 using the following identifiers.

Identifier	Definition
grammar_id	Identifies which grammar model is currently being used
grammar_node_id	Identifies a node in the grammar model
word_id	Uniquely identifies a word
wordlat_node_id	Identifies a node in the word lattice
LT_id	Lexical tree identifier. Identifies the lexical tree to be used
LT_branch_id	Identifies a unique branch in a lexical tree. A branch is defined as all of the monophones subsequent to and including the first monophone in a tree.
LT_node_id	Identifies a node in a lexical tree
monophone_id	Identifies a monophone
transvec_id	Identifies which transition vector to use
state_id	Identifies the state
phone_no	Unique phone instance identifier
prev_phone_no	Phone number of previous phone in network
triphone_id	Triphone identifier
LML_factor	Language model lookahead factor. The smeared language model score that must be added

The system level includes a general purpose processor and cache, a system controller, a DMA (direct memory access) controller, clocks, PLL (phased locked loop), GPIO (general purpose I/O interface), timer watch dog, memory interfaces to interface to main memory and non volatile storage, serial, host, debug and display interfaces. The system level also includes a model adaptation engine, a language model factor engine, a speech recognition system controller, a language model interface, and a recognition output interface.

The processor organises the transfer of the various models from the non volatile storage to the various model memories. It handles the language model factor requests. Language model factors are used to by the state level to apply the language model score. The language model (LM) interface provides the mechanism for obtaining LM factors as and when they are needed. The processor keeps a copy of the wave front in memory which is updated simultaneously with that in the wave front manager. Using its copy of the lexical tree, the processor can determine which language model scores may be needed in

the coming frames. Once calculated, the language model factors can be kept in memory for use in subsequent frames. The system level is also used for forming speech model adaptation, and acting as an overflow for the various buffers used by the architecture.

The system level includes the audio codec interface and the audio front end interface, connected to the processor via a APB (advanced peripheral bus) to AHB (advanced high performance bus) bridge.

The grammar level includes a grammar engine and grammar model(s). The grammar level sets out the rules of the particular recognition task at the highest level. During system initialisation a number of grammar models might be loaded, for example a command grammar and a dictation grammar. Command mode grammar is a restricted grammar which includes only a very limited lexical tree with command specific words. Some branches of this lexical tree may include multiple words together, where a command has such a structure, e.g. "open file menu". Typically, only a single instance of the lexical tree will be needed. Command mode is used because it can obtain a higher accuracy, due to the lexical tree constraint. In dictation mode grammar, a full lexical tree structure is used.

The grammar may be selected by a user during the recognition process. For example, the command grammar may be selected when the user presses a command button. The architecture supports fast swapping of grammar models. For example, a user may be in the middle of dictating a message when a "punctuation button" is pressed and the phrase "question mark" uttered. At this point, a punctuation grammar could be loaded, in order to increase recognition accuracy.

The word level comprises a word engine, which includes a word lattice and a word graph generator. The word level has two main functions, of keeping a record of the search path at the word level using a word lattice, and interfacing to the language model via a cache. The word lattice is a record of all of the words that have been searched up to the current frame. As new words are searched, they are appended into the lattice at the appropriate place. After the final frame in the current sentence, the n-best word lattice node identifiers from the n-best scoring tokens are passed back to the word

lattice. This enables the n-best sentences to be traced back through the lattice. The word lattice can also be post processed to produce a word graph which is essentially a compact version of the word lattice.

At the word level, a record is kept of all tokens that have passed through a word end. This results in a lattice of words. When the final feature vector has been processed, the best scoring token is identified and used to trace back the search path back to the start of the search.

The phone level includes a "phone book" for generating new triphone instances, a wave front manager for sending requests to the phone book to generate new phone instances, a phone request arbiter, and a path merge unit.

The phone book includes a lexical tree, which comprises a vocabulary of words, sounds and/or phrases organised according to the phones or monophones making up the sound. A lexical tree is commonly used in speech recognition rather than a linear lexicon, because it greatly reduces the amount of storage space required to store a word vocabulary, by avoiding duplication of common word beginnings. It is also easier to search when the word beginning is known.

Although it is possible to search a lexical tree based on monophones, it is well known that better results can be achieved in speech recognition by basing the lexical tree search on groups of more than one monophone, since the sound of one syllable commonly affects the pronunciation of the next syllable in the same word, or the first syllable in the next word. In this embodiment, groups of three monophones are used, and these are known as triphones. It is alternatively possible to use a different number of monophones for each group. The phone generator in the "phone book" uses data from the lexical tree to generate triphones.

In one embodiment of the invention, approximately 45 different cued mono-phones are considered. This gives of order 60,000 tri-phones. Each triphone in this embodiment is modelled by a Hidden Markov Model (HMM) with three states. There is no correspondence between individual states and individual phones of the triphone.

Alternatively, it is possible to model each triphone by a single state, or by any other number of states.

The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree. The search space can be envisaged as a wave moving through an acousto-linguistic search space. In order that the wave can be dynamically created, a record of the wave front phone instances must be maintained. This is done in a wave front phone CAM/RAM structure, within the wave front manager. When a token occurs in the final state of a phone instance in the PINE array at the front of the wave, a new phone instance must be generated at the phone level and passed to the PINE array.

The phone instance includes a phone_no, which is a phone instance identifier to identify the phone uniquely in the current search. The phone instance also includes a word lattice node identifier (labelled "word_lat_node_id") to identify the phone's position in the word lattice, state identifiers (labelled "state_ids") for the HMM, storage to hold the tokens as they move through the HMM and LM_factor which is used to by the PINE to apply the language model score. To handle the connectivity between phone instances, a phone instance points to a previous phone instance using the phone_no of the previous phone. This allows tokens to be passed between phones. A "token" is a hypothesised search path, represented by a non-zero score at a particular state, and tokens are propagated through the HMM with each time frame.

The phone instance network can be visualised as a wave with new phone instances being added to the front of the wave, and empty phone instances being removed from the back of the wave. In some circumstances, the back of the wave may be in the same position as the wave front. The back of the wave moves on through the lexical tree as tokens are either moved on or pruned.

Tokens, each representing a particular hypothesis, hop through the states in the HMMs, each hop consuming a feature vector. Each token explores all possible next states which typically involves placing a token in the next state and keeping one in the current state. When there is more than one token in a state, the least likely token can be removed

because both have the same word history. When a token reaches the last state of a phone instance at the from of the wave, the next phone instance(s) are requested from the phone level. The phone level uses the lexical tree to create the new phones. As well as looking up the state identification numbers for each HMM state, it also incorporates the correct language model factor.

The language model factor engine is used to smear the language model scores across the lexical tree. This operates in two ways depending on whether the factor is needed for a word start phone(s) or word internal phones. For word internal phones, LM factors can be computed as needed, provided that the LM scores are available in cache. This is because in general the storage requirements for the LM scores are less than the storage of the LM factors for all of the factors in the branch. However when a token is in a word end phone there may a significant number of word start phones e.g. up to $(\text{monophones})^2$ requiring a large amount of computation. To avoid this the LM engine will pre compute word start LM factors when it seems highly likely that a token will be passed to the word starts. These word start LM factors will be held in the LM factors cache. The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree.

The state level shown in figure 1 includes a phone instance network engine (PINE) array, a MHD (Mahalanobis distance) engine, an audio front end, a feature vector buffer for model adaptation, and compressed acoustic models. The state level is the lowest level, in which a network of phones is dynamically created.

The audio input for speech recognition is input to the front end unit of the state level. This audio input may be digital audio, or it may be analog audio which is converted to digital audio using an analog to digital converter. The audio input is divided into time frames, each time frame typically being of the order of 10ms.

For each audio input time frame, the audio signal is converted into a feature vector. This may be done by splitting the audio signal into spectral components. For example, in the present embodiment, the audio input signal is split into 12 frequency bands and one

energy band, and a feature vector is constructed using these 13 components, plus their first and second derivatives, a total of 39 components.

The feature vector is passed to the MHD (Mahalanobis distance) engine. This computes all possible state output likelihoods for the current frame as the state mean and covariance data is read from the acoustic model memory.

The likelihood of each state is determined by the distance between the feature vector and each state. To reduce the number of states, a technique known as "state tying" may be used. Multiple states are grouped together in 39D space, according to their proximity to one another. An average or central value from each group is used to represent the entire group. Thus, the effective number of states may be reduced from 40,000 to 8,000. The accuracy may be improved using a language model in addition to the word recognition lexical tree. A language model is based on the probability of sequential words. The above figure of 8,000 states gives best recognition results when used with a language model.

The MHD (Mahalanobis Distance) is a distance between two N-dimensional points, scaled by the statistical variation in each component. In embodiments of the invention, the Mahalanobis distance between the feature vector and each state is calculated, to determine similarity of the feature vector to each state.

The Feature Vector represents a point in an N-dimensional space. Different embodiments of the invention may use different values of N. Generally we see values of N in the range of 20 to 39, although values outside this range are not excluded.

The distance calculator stage of the recognition process computes a probability or likelihood that a feature vector corresponds to a particular state. Each state has an associated probability distribution in the N-dimensional space. The distance calculator computes the distance in the N-dimensional space from the Feature Vector to the probability distribution for each state. In some embodiments, the calculation is performed only when the search stage algorithms determine that the distance for that particular state is needed by the searchstage. In some embodiments, the distances for all

of the states are calculated and made available to the search stage, which allows pipelined operation. Embodiments that do the calculations on demand are preferable.

The Mahalanobis Distance (MHD) is extensively described in the literature. It may be calculated using the formula

MHD value = ln(P(x)) = -0.5[N*ln(2*pi)+ln|Cov|)] + (-0.5*[(x-u)'*inverse(Cov)*(x-u)])

where

N= number of dimensions in the space
x=feature vector, a point in the N dimensional space
u=mean vector
Cov=covariance matrix
|Cov|=determinant of covariance matrix
(a)' = transpose of matrix a

Different embodiments may use variations on this equation, for example the base for the logarithm is not always the natural base e that is used in the above example. The log of the probability is preferably calculated, rather than the probability, as it simplifies the implementation. Not all embodiments do this. In alternative embodiments, the likelihoods rather than probabilities are computed as it leads to simpler implementations of computations in the search stage.

The feature vector is used to calculate 8,000 MHD distances for each time frame, i.e. one distance for each of the 8,000 states. It is beneficial, although not essential, to perform the MHD calculations separately in a pre-processor, known as a MHD engine, because MHD calculation is a computationally intensive process.

Due to the 10ms frame length, a feature vector arrives at the MHD engine of the state level every 10ms. The MHD engine calculates the likelihood of each feature vector

corresponding to each of a number of states derived from an acoustic model. Each state is also a 39 dimensional vector, having the same spectral components as the feature vector. The MHD engine calculates the difference in 39-D space between the feature vector and each state to obtain the likelihood for that state. A parallel MHD or a single MHD arrangement may be used.

The feature vectors may be sent to a feature vector buffer for model adaptation, which determines whether the observed features fit well with the acoustic model, and may initiate an adaptation of the model if the fit is not sufficiently good.

Scores indicating the state likelihoods or probabilities are output from the MHD engine to the PINE array. It is preferable to use likelihoods rather than probabilities, because likelihoods can be added together to give a total likelihood. Probabilities must be multiplied, which can give rise to rounding errors. In this embodiment, eight separate pines are used, each of 4 kilobytes, thus storing 4000 phones per pine.

During recognition, the Phone Instance Network Engine (PINE) Array will contain a network of phone instances. Tokens representing search hypotheses in the form of a probability(likelihood) will exist in some of the HMM states of the phone instances. The tokens are propagated into all available states and updated with the corresponding state output probabilities. Maximum model, word end and beam pruning are applied to the tokens to ensure the search space is kept bounded. When a token is in the final state of a phone model and has avoided being pruned, the PINE makes a next phone request to the Phone level. The PINE array may receive a number of new phone models from the phone level which must be appended to the appropriate existing phone.

Figure 2 shows the data structures stored in the PINE array, and other main data structures used during the acoustic search. Each PINE includes a phone instance CAM and a phone instance RAM. The phone instance CAM stores the phone instance number (labelled "phone_no") which uniquely identifies each phone instance. The phone instance CAM also stores the state IDs of the three HMM states used to model the phone, and flags indicating the position of the phone. The "back" flag indicates if the phone is at the back of the wavefront. The "front" flag indicates if the phone is at the

front of the wavefront. The "empty" flag indicates if there are no tokens in any of the three states for that phone. The "end" flag indicates if the phone is at the end of a word.

The phone instance RAM stores the token scores for each of the three states corresponding to each phone. The st_dum field indicates a dummy state, which is used to represent a state prior to the first state of the first phone. The "tmp" field is used for passing a token from a previous phone in the lexical tree to the current phone. This will be explained in more detail later. The prev_phone_no field is for storing a pointer to the previous phone instance in the network. The wordlat_node_id identifies the position of the phone instance on the word lattice node, and the LM_factor stores the language model factor for that phone instance.

At the phone level a record of all phones on the wave front must be maintained. If a phone instance is on the wave front, the phone_no stored in the phone instance CAM will point to the details in the wave front phone CAM and RAM.

The wave front phone CAM and RAM keeps pointers into the grammar and lexical tree. When a token in a wave front phone reaches the last state, the PINE array makes a request to the phone level by passing back the phone_id. The phone_id is used as a key to determining the token's grammer_node_id and LT_node_id within the wave front CAM/RAM structure. If the token is at a word end the grammar model is checked. A grammar identifier points to the type of grammar.

At the word level, figure 2 shows an example word lattice structure, with silence having a word_id of 0, and a wordlat_node_id of 6. This is shown as connected to two word nodes – "which" has a word_id of 238 and a wordlat_node_id of 4, and "neither" has a word_id of 3556 and a wordlat_node_id of 5.

At the highest level, i.e. the grammar level, a simple grammar model is shown which will allow the phrase 'silence which house silence' or 'silence neither house silence' to be spoken. Whenever a word end is reached during the search, the grammar model must be consulted to determine the next possible word(s) or lexical tree(s). This allows the word lattice at the word level, consisting of a network of words, to be assembled.

Whenever a word end is reached, the word is added to the lattice and the wordlat_node_id updated accordingly.

Figure 3 shows a single phone instance network engine (PINE). At system power up the state transition RAM will be loaded with the appropriate state transition vector but the CAM and RAM memories will be empty. When the system is initialised an initial set of phone model(s) will be instantiated with a token in the dummy state and a path score of zero. The phone models are received from the phone level.

When the audio data is processed, a stream of all possible state output probabilities will be received from the MHD engine. A counter that is synchronous to the stream of state output probabilities from the MHD produces a concurrent stream of state_ids. The job of the CAM is to perform a search to see if the state_id is instantiated anywhere in the CAM state_id fields. If it exists and the location doesn't have its empty flag set then this indicates that a token may need to be updated. The match signal from the CAM must be encoded into an address for the RAM. The corresponding RAM address can then be read. Any token that will end up in the identified state can then have it's path score updated and including any transition vector likelihood.

It is possible that a single state_id generates multiple CAM matches. To handle this situation the CAM and RAM are effectively decoupled by means of a match fifo and repeat count fifo. The state_id and corresponding state_output_probability are stored in the repeat count fifo. During a subsequent frame when there is no match, this state_id can be revisited, ignoring the matches already dealt with.

All searched phone instances corresponding to words in the current possible lexical tree are stored in the CAM-RAM. This includes phone instances at the beginning of the word, middle of the word and end of the word. These phone instances are connected by the "previous phone instance" field, which allows traceback through an entire word within the CAM-RAM.

A mechanism is thus required to pass tokens from one phone instance to the next. This mechanism is provided by the tmp field in the RAM.

The tmp field in the RAM holds tokens during token passing.

At the start of each time frame, the audio input is converted into a feature vector, the MHD values calculated, and the MHD values used to update the token scores. After update of all tokens, 2 passes are made through the CAM-RAM, firstly for pruning, and then for token passing. The pruning in this example removes tokens of below a threshold score, although other types of pruning are also possible. The pruning may involve removing a predetermined proportion of the tokens, chosen as those of lowest likelihood. A third possible method of pruning is removing all tokens with scores below a first predetermined value, but at word ends removing all tokens with scores below a second predetermined value, the second predetermined value being greater than the first predetermined value.

The token passing works as follows. Each phone instance points back to a previous phone, due to the prev_phone_no field stored in the RAM. During the token passing, the third token of a previous phone is grabbed and put into the tmp field of the current phone.

In other words, if a token is on the third state, and it is not pruned, it must enter the next state of the next phone. To do this, the tokens that lie in the third state are moved across to the tmp field.

For example, phone no. 345 needs to look at phone no. 320 to see if any tokens are in the third state. If yes, it takes these tokens and puts them in the tmp field.

One field in the CAM is labelled "front". This indicates whether the phone is on the wave front. If the phone is not on the wave front, then another phone will have picked up the token in the third state. However, if the phone is on the wave front, then no other phone will have picked up the token on the third state. Therefore, if there is a token in the third state, and the phone is on the wavefront, it is necessary to generate a next phone request.

The current phone number, word lattice identifier and word end flag are sent to the wave front manager, to request a next phone. If the word end flag indicates a word end, then the score is also passed to the wave front manager. The score can be passed in any case, but it is only required at a word end.

The phone level generates the next phone or phones. A new phone instance starts, and all of state1, state2 and state3 have scores pre-set to zero. The old score is provided in the tmp field.

The old phone is still within the wave, but it is unlikely to be at the back of the wave at this stage. When it gets to the back of the wave, it is deleted. Until it gets there, it remains in the CAM-RAM as before. The third token remains where it was, since tokens can either stay in the third state or move on.

Note that it is possible to end up with more that one token in each state. If this occurs, then the token with the poorer score is deleted. This does not affect the result of the lexical tree search, because both tokens on the same state have exactly the same word history.

Figure 4 shows a block diagram of the wave front manager. The wave front manager is responsible for maintaining a record of the set of phone models that exist at the front of the acoustic search space. This set of phone models is referred to as the "wave front." The key requirement for this block as to minimise the time between receiving a "next phone request" and generating the appropriate responses. The wave front manager receives the next phone requests from the PINE array and generates appropriate responses for the Phone Book, Word Engine, Path Merge Unit and Grammar engine The wave front manager also receives new wave front phone models from the Phone Book and updates the wave front.

The wave front manager receives the next phone request from the PINE array, via the phone request arbiter. The next phone request consists of a phone_no, word_end_flag and wordlat_node_id. If this phone is at the end of a word, the word_end_flag will be set.

The wave front manager uses the phone_no as input to the wave front phone CAM. As each phone_no is unique, it is guaranteed that only a single match will be generated. The wave front manager uses the CAM generated address to access the RAM. It then sends retrieved data from the RAM to the phone book, grammar engine, Path Merge Unit and word engine.

The wave front manager removes phone instance from CAM and RAM by writing 0's into CAM and RAM, and increments the empty count by one.

Sometime later, the wave front manager receives new phone instances from the phone book and place them in a fifo. The phone instances are taken from the fifo and written into the CAM and RAM at the next_empty_address. The empty_count is decremented by one. A match is run on the "occupy" flag, and the address is placed into next_empty_address register.

Both the phone level and the state level make use of a path merge unit to reduce the number of possible paths, by path merging. Figure 5 shows the use of link nodes to permit path merging. In the present embodiment, all word end phone instances are linked to word start triphones with a link node. During each frame the path merge unit looks for *word end* tokens that can be recombined. This is done by looking for word end phones that have the same word_id and word history. When they are found a link is added using the path merge unit in the PINE array.

Figure 6 shows a diagram of the architecture at the Phone level, showing the path merge unit. The path merge unit includes a path merge CAM and a path merge RAM. The path merge CAM stores word_id, word history for two words, and an "occupy" field. The path merge RAM stores a link_node_id. A CAM RAM occupancy manager keeps track of the empty addresses in the CAM and RAM. The path merge unit is only passed information from the wave front manager if the phone is a *word end phone*. For each frame, there is a check to see if there is an identical word with the same history, and if there is, the link_node_id is passed to the path merge unit. If there isn't an identical word with the same history, a new record is written into the Path merge CAM-RAM.

Figure 7 is a block diagram showing the Phone Book architecture. The Phone Book is tasked with generating the next phone models when given a pointer into the lexical tree. The key requirement for this block as to minimise the time between receiving a "lexical tree address" and generating the appropriate responses, in particular the next phone model instance(s).

The lexical tree is used to build the next triphones based on the current context.

Current triphone	Next triphone	comments
Left context = a Central context = b Right context = c	discarded Left context = b Central context = c **Right context = ?**	**next_leftmph_id stored in wave front manager** **current LT_node_id stored in wave front manager** **must be looked up. Large (< 45) for new word start**

The LT_node_id, next_left_mph_id and prev_phone_no are received at the phone generator from the wavefront manager. The LT_node_id is then used to address the lexical tree store and return the central monophone_id. One or more further accesses to memory are performed using the microcoded instructions, each memory access returning the right monophone_id.

A two stage lookup is accessed using the three monophone_ids (giving a triphone), the associated state_ids. The phone model is assembled, and returned to the PINE array. The wavefront manager is then updated by sending it relevant parts of the phone model.

The lexical tree store is a micro-coded ROM with both instruction and data fields. The lexical tree store holds a number of lexical trees. Each lexical tree may hold single words or multiple words. Each lexical tree comprises lexical tree elements, and each element has three fields. For now, the fields will be integers and will be converted to std_logic_vector later. The following is an example of the lexical tree store, with the LT data in word_id order.

```
ZERO   z iy r ow      19 8 12 11 0 0 0 0      sp 0
ONE    w ah n         18 2 10 0 0 0 0 0 0     sil 1
TWO    t uw           14 16 0 0 0 0 0 0       ah 2
THREE  th r iy        15 12 8 0 0 0 0 0       ay 3
```

FOUR	f ow r	6 11 12 0 0 0 0 0	eh 4
FIVE	f ay v	6 3 17 0 0 0 0 0	ey 5
SIX	s ih k s	13 7 9 13 0 0 0 0	f 6
SEVEN	s eh v ih n	13 4 17 7 10 0 0 0	ih 7
EIGHT	ey t	5 14 0 0 0 0 0 0	iy 8
NINE	n ay n	10 3 10 0 0 0 0 0	k 9
OH	ow	11 0 0 0 0 0 0 0	n 10
			ow 11
The following shows the LT data in MP (monophone) start order.			r 12
			s 13
			t 14
EIGHT	ey t	5 14 0 0 0 0 0 0	th 15
FIVE	f ay v	6 3 17 0 0 0 0 0	uw 16
FOUR	f ow r	6 11 12 0 0 0 0 0	v 17
NINE	n ay n	10 3 10 0 0 0 0 0	w 18
OH	ow	11 0 0 0 0 0 0 0	z 19
SEVEN	s eh v ih n	13 4 17 7 10 0 0 0	
SIX	s ih k s	13 7 9 13 0 0 0 0	
TWO	t uw	14 16 0 0 0 0 0 0	
THREE	th r iy	15 12 8 0 0 0 0 0	
ONE	w ah n	18 2 10 0 0 0 0 0	
ZERO	z iy r ow	19 8 12 11 0 0 0 0	

The following is an example of the three fields of a lexical tree element. The first field is an instruction field, and the second and third fields are data fields.

Fields:	1st instr 2b	2nd data 6b	3rd data 6b	
	00	MP	MP	- raw data field.
	01	br	br	- branch address.
	10	br	br	- branch address.
	11	word_id		- double field.

Note that there might be no difference between "01" and "10"

To build the "ey" tree, we only have to consider "EIGHT", because this is the only word in the lexical tree which starts with "ey". EIGHT is represented as "ey t", with phone_nos 5 14 0 0 0 0 0 0. The first lexical tree element stores the raw data specifying the phone_nos, the second lexical tree element is an end field, and the subsequent lexical tree element is set to zero.

```
0 00 5 14   - raw data
1 11 8 0    - end field
2 00 0 0
```

To build the "f" tree, we have:

```
FIVE   f ay v      6  3 17  0  0  0  0  0
FOUR   f ow r      6  11 12  0  0  0  0  0

0  01  4  0  -- branch forward four places as well
1  00  6  3  -- as the implied other brach of next (one).
2  00  3 17
3  11  5  0  -- all addresses are relative offsets.
4  00  6 11
5  00 11 12
6  11  4  0
7  00  0  0
```

To build the "n" tree, we have:

```
NINE          n ay n          10  3 10  0  0  0  0  0

0  00 10  3
1  00  3 10
2  11  9  0
3  00  0  0
```

To build the "ow" tree, we have "ow" in both the first and second data fields, as no tree is going to be a single monophone long.

```
OH            ow              11  0  0  0  0  0  0  0

0  11 11  0
1  00  0  0
```

To build the "s" tree, we have:

```
SEVEN         s eh v ih n     13  4 17  7 10  0  0  0
SIX           s ih k s        13  7  9 13  0  0  0  0

0  01  6  0  -- branch forward six places.
1  00 13  4
2  00  4 17
3  00 17  7
4  00  7 10
5  11  7  0
6  00 13  7
7  00  7  9
8  00  9 13
9  11  6  0
10 00  0  0
```

To build the "t" tree, we have:

```
TWO   t uw       14 16 0 0 0 0 0 0

0 00 14 16
1 11 2 0
2 00 0 0
```

To build the "th" tree, we have:

```
THREE        th r iy     15 12 8 0 0 0 0 0

0 00 15 12
1 00 12 8
2 11 3 0
3 00 0 0
```

To build the "w" tree, we have:

```
ONE   w ah n      18 2 10 0 0 0 0 0

0 00 18 12
1 00 12 10
2 11 1 0
3 00 0 0
```

To build the "z" tree, we have:

```
ZERO  z iy r ow    19 8 12 11 0 0 0 0

0 00 19 8
1 00 8 12
2 00 12 11
3 11 0 0
4 00 0 0
```

Once all of the branches have been built, the next stage is to concatenate the tree. It appears that four elements make a page, and each element is two bytes.

```
                                                  page
ey:    00 5 14  11 8 0  00 0 0  00 0 0            5
f:     01 4 0  00 6 3  00 3 17  11 5 0            6
       00 6 11  00 11 12  11 4 0  00 0 0          7
n:     00 10 3  00 3 10  11 9 0  00 0 0           8
ow:    11 11 0  00 0 0  00 0 0  00 0 0            9
s:     01 6 0  00 13 4  00 4 17  00 17 7          10
       00 7 10  11 7 0  00 13 7  00 7 9           11
       00 9 13  11 6 0  00 0 0  00 0 0            12
t:     00 14 16  11 2 0  00 0 0  00 0 0           13
th:    00 15 12  00 12 8  11 3 0  00 0 0          14
w:     00 18 12  00 12 10  11 1 0  00 0 0         15
z:     00 19 8  00 8 12  00 12 11  11 0 0         16
```

Then, the start_lookup header is defined. Each row (page)is 8 bytes and hence the header takes 3 rows. The end_lookup header is two rows. Therefore, the lexical tree store proper starts at the 6th (#5) page.

```
sp sil ah ay eh ey  f ih          - MP
 0  1  2  3  4  5  6  7          - MP_id
 3  0  0  0  5  0  6  0          - data stored.

iy  k  n ow  r  s  t th
 8  9 10 11 12 13 14 15
 0  0  8  9  0 10 13 14

uw  v  w  z  .  .  .  .
16 17 18 19  .  .  .  .
 0  0 15 16  0  0  0  0
```

The end_lookup header is defined. 20 is out of bounds.

```
ey  f  n ow  s  t th  w
 5  6 10 11 13 14 15 18

z  .  .  .  .  .  .  .
19 20 20 20 20 20 20 20
```

To put the above information all together:

```
START       sp:ih     3 0    0 0     5 0    6 0    0
LOOKUP      iy:th     0 0    8 9     0 10   13 14    1
            uw: v      0 0   15 16    0 0    0 0    2

END         ey:w       5 6   10 11    13 14   15 18    3
LOOKUP      z        19 20   20 20    20 20   20 20    4

LEXICAL     ey:    00 5 14  11 8 0  00 0 0  00 0 0    5
TREE        f:     01 4 0  00 6 3  00 3 17  11 5 0    6
STORE              00 6 11  00 11 12  11 4 0  00 0 0    7
            n:     00 10 3  00 3 10  11 9 0  00 0 0    8
            ow:    11 11 0  00 0 0  00 0 0  00 0 0    9
            s:     01 6 0  00 13 4  00 4 17  00 17 7    10
                   00 7 10  11 7 0  00 13 7  00 7 9    11
                   00 9 13  11 6 0  00 0 0  00 0 0    12
            t:     00 14 16  11 2 0  00 0 0  00 0 0    13
            th:    00 15 12  00 12 8  11 3 0  00 0 0    14
            w:     00 18 12  00 12 10  11 1 0  00 0 0    15
            z:     00 19 8  00 8 12  00 12 11  11 0 0    16


 3 0   0 0   5 0   6 0
 0 0   8 9   0 10  13 14
 0 0  15 16   0 0   0 0
```

```
  5 6    10 11    13 14    15 18
 19 20    20 20    20 20    20 20
00 5 14  11 8 0  00 0 0  00 0 0
01 4 0  00 6 3  00 3 17  11 5 0
00 6 11  00 11 12  11 4 0  00 0 0
00 10 3  00 3 10  11 9 0  00 0 0
11 11 0  00 0 0  00 0 0  00 0 0
01 6 0  00 13 4  00 4 17  00 17 7
00 7 10  11 7 0  00 13 7  00 7 9
00 9 13  11 6 0  00 0 0  00 0 0
00 14 16  11 2 0  00 0 0  00 0 0
00 15 12  00 12 8  11 3 0  00 0 0
00 18 12  00 12 10  11 1 0  00 0 0
00 19 8  00 8 12  00 12 11  11 0 0
```

This information is converted to hexidecimal. The first five are byte fields.
The rest are:
2bit - uCode.
6bit - data.
2bit - null.
6bit - data

```
03 00    00 00    05 00    06 00
00 00    08 09    00 0A    0D 0E
00 00    0F 10    00 00    00 00
05 06    0A 0B    0D 0E    0F 12
13 14    14 14    14 14    14 14
05 0E    C8 00    00 00    00 00
84 00    06 03    03 11    C5 00
06 0B    0B 0C    C4 00    00 00
0A 03    03 0A    C9 09    00 00
CB 00    00 00    00 00    00 00
86 00    0D 04    04 11    11 07
07 0A    C7 00    0D 07    07 09
00 0D    C6 00    00 00    00 00
0E 10    C2 00    00 00    00 00
0F 0C    0C 08    C3 00    00 00
12 0C    0C 0A    C1 00    00 00
13 08    08 0C    0C 0B    C0 00
```

The above is the micro code for the digit lexical tree. It allows branching, and it holds the word_id. It knows which lexical trees it contains and where they are housed. It also stores the possible end monophone list.

Figure 8 shows a further example of the Lexical Tree RAM, giving examples of different types of branching in the lexical tree, and how these are represented in the RAM.

Figure 9 is a block diagram showing the data structure that is supported for the word link record, to enable word n-best decoding. The word engine maintains a record of the words that have been currently searched by the PINE array. It is used to determine the word history and allow trace back during or at the end of search. The word history is needed to determine the language model score and whether particular paths can be recombined or merged.

Figure 10 shows a block diagram of the word engine. This is designed to allow fast access to the word history (thus allowing rapid phone creation at the phone level), and to allow reasonably fast trace back. The Word Engine is primarily a memory store that fills incrementally. The decoupling of the wordlat_node_id and the physical address will allow the possibility of being able to keep the WLRs for a given word in contiguous addresses, thus avoiding the need for the pointer field.

The language model interface is designed to allow the Phone Book to quickly obtain the appropriate LM factor for the phone model that it is currently constructing. Based on the phones in the wave front, all possible next LM factors can be determined. This will be done in software together with the language model factor engine. The ARM 9 will maintain its own record of the wavefront using the same next phone requests received by the wave front manager and its own copy of the lexical tree.

Figure 11 shows the architecture of the LM interface. This allows the Phone book to quickly access the LM factor for the phone that is currently being generated. Prior to the phone generation phase LM_factors are computed for all possible phones that may be generated in the next frame.

The grammar engine is designed to give the Phone Book a quick response to requests for the next grammar node(s). In the case of a task grammar where the lexicon may be limited to a few words(e.g. yes, no), it may be desirable to use the same lexical tree. In this case it will be necessary for the grammar, to provide lexical tree pointers such that the phone generator can jump to the appropriate part of the lexical tree.

For a speech recognition system it is also necessary to specify the grammar. Again we will use HTK's Standard Lattice Format(SLF) to specify the grammar. In practice it will be convenient to use HTK's HParse format to specify the grammar using a high level grammar notation. This can then be converted into using the HParse tool. Consider having to specify a grammar for a digit recognition task. The HParse format for this is as follows:

$digit = one | two | three | four | five | six | seven | eight | nine | zero
$digit_sp = $digit sp
$noise = lipsmack | breath | sil
([sil] < $digit_sp | $noise > [sil])

where | denotes alternatives, < > denotes one or more repetitions, { } denotes zero or more repetitions, [] denotes that item is optional, and () contains the grammar.

Figure 12 shows the network that is defined by this grammar. The network starts and ends with a "sil" phone, representing silence. Between the start and end, a number of possible routes correspond to further silence, a breath, a lipsmack sound, or a detected word.

Figure 13 shows the grammar engine architecture. When a next phone request arrives, the next grammar node(s) must be retrieved from memory. In the case of a task grammar, the phone book should be given the lexical tree memory addresses to allow the phone to be determined.

Figure 14 shows a single frame of the Audio Codec (AC) link protocol used in some embodiments of the invention. The Advanced Audio CODEC Interface (AACI) is an ARM Primecell that provides an interface to off-chip AC97 audio codecs. AC97 is Intel's audio codec specification, initially created for the PC market. The AACI is an AMBA slave block that connects to the Advanced Peripheral bus (APB). The AACI connects to the off-chip codec using the AC-link protocol.

The AC-link protocol is a bi-directional, fixed clock rate, serial digital stream. AC-link handles multiple input and output PCM audio streams as well as register control accesses. It employs a time division multiplexed (TDM) scheme that divides each audio frame into 12 outgoing and 12 incoming data streams each with up to 20 bit sample resolution. The tag slots are used to indicate which of the subsequent slots contain valid data. Add and dat are the control register address and data slots for read and writes. The adc slots contain the outputs from the analog to digital converters. Assuming that the ADC sample rate is set to 16 kHz then the AC-link controller will receive 16 bits of data every third codec frame.

The AACI has 4 channels that can be operated simultaneously. They are accessed through DMA.

Figure 15 shows the system timing. Each frame is divided into three sections:

- Token update (MHD wash)
- First pass - inter phone token passing
- Second pass – garbage collection, max model pruning

The horizontal direction of the figure indicates elapsing time, and various subcircuits and processes are indicated along the vertical direction of the figure. The top part of the diagram shows frame count, and push to dictate button, and the lower part of the diagram shows various subcircuits and processes, within a single time frame, whilst the push to dictate button is engaged. These subcircuits and processes are language model interface, word engine, phone book, PINE array, MHD engine, mean & covar reg, and feature vector reg.

At the start of the time frame, the feature vector is generated, and the MHD values are calculated. The MHD values are then passed to the PINE array, which engages in beam pruning and token score updating. The phone book receives next phone requests and returns next phone responses. The word engine adds word link records, and the language model interface supplies language model factors.

In the second part of the frame, pruning thresholds are calculated for the PINE array, for the next frame, and the PINE array engages in token passing. The language model interface transfers language model factors for the next frame.

Whilst the "push to dictate" button remains activated, the above process repeats in a sequence of frames.

The embodiments of the invention which are now described relate to a distance engine for calculation of Mahalanobis Distances.

Figure 16 shows a block diagram of a speech recognition system according to an embodiment of the invention. The system includes a microphone 100, which outputs an audio signal to a combined AGC (automatic gain control) and ADC (analogue to digital converter) 101. In alternative embodiments, the AGC and ADC may be separate. In this embodiment, the incoming data comes from a microphone, and is then converted to digital format. However, it is also possible that the digital audio data may be received from a network, or may be read from storage means such as a memory, an optical disk, or a hard disk drive.

The digital audio data output from the ADC is input to an audio data buffer 102, and may be held here until required. The data is then output from the audio data buffer 102 to a front end circuit 103. The incoming audio data is divided into frames, as described above. In cases where the system is extremely busy, it is possible to drop some audio frames, to allow the system to catch up with any backlog.

In the audio front end circuit 103, the audio is split into spectral components to generate a feature vector, as described above. For example, the feature vector may have a number of frequency bands, and an energy band, plus their first and second derivatives. The feature vector is then output from the front end 103 to the distance calculation engine 104.

The distance calculation engine 104 converts feature vectors into Mahalanobis distances . The acoustic model may be chosen from a plurality of acoustic models, e.g. according

to the language used, and may be stored in a storage means 105 such as a memory, a disk drive, etc. The distances calculated by the distance calculation engine are then transferred to the search stage 106 of the speech processing circuit, which uses models such as one or more word models and/or language models to generate and output recognised text. Such word and language models may be stored in a separate storage means 107.

In one embodiment, the distance calculation engine uses Mahalanobis Distance (MHD) calculations to determine distance values for output to the search stage 106. In further embodiments, different types of distance calculations may be used, e.g as described above.

In some embodiments of the invention, the distance calculation engine is designed as a speech accelerator, to operate as a loosely bound co-processor for a CPU running speech recognition software. This has the advantage of reducing computational load on the CPU, and reducing memory bandwidth load for the CPU.

In some embodiments, the accelerator interface is memory-mapped. Other types of interface are also possible to support different system configurations, for example removable accelerator cards with SDIO (secure digital input/output), CardBus, or USB interfaces.

The speech recognition apparatus may also operate with a CPU and DSP (digital signal processor), where the DSP implements the "front-end" signal processing to produce a feature vector, and the CPU implements the search stage, reading the distance results from the distance calculation engine.

The system may be implemented in any combination of hardware or software.

In some embodiments of the invention the search stage 106 provides the distance calculation engine 104 with an active state list 1XX that indicates the subset of states for which the search stage requires distances to be calculated by the distance calculation

engine 104. This is an optimization that may reduce computation time and/or power consumption.

In preferred embodiments of the invention, the front end 103, distance calculation engine 104 and the search stage 106 operate in a pipelined manner. When operating in a pipelined manner it is unlikely that the search stage 106 will be able to provide the active state list 1XX early enough for the distance calculation engine 104 to implement the optimization of computing only the distances that will be required by the search stage 106. The distance calculation circuit 104 may calculate the MHD values for every state in the lexicon, per frame, whether it is subsequently required by the search stage or not. This allows the accelerator and software system to operate in a concurrent pipelined manner, which maximizes the system throughput.

The distance calculation engine 104 may have a plurality of memory areas for storing feature vectors, to allow a feature vector for one frame to be transferred to one memory area in the distance calculation engine 104, while another feature vector for another frame is being read from a second memory area, for calculation of a distance. The memory areas used for reading and writing may alternate with subsequent frames.

Figure 17 shows a block diagram of a distance calculation accelerator 200, used in an embodiment of the invention. A feature vector is received at input 212 of the accelerator 200. The feature vector is loaded into feature vector (FV) memory A 208 via a bus 211. In reality, there may be more than a single bus, but for simplicity, only one bus 211 is shown in the figure.

The registers 203 are discussed further at the end of this description.The registers 203 may include, for example, an accelerator status register, for indicating whether the acoustic model data is correctly stored in memory, as described in more detail below.

The FV registers 209 hold the feature vector whose distances are currently bring computed by the distance calculation engine 204. If the distance calculation engine 204 is implemented in software, the FV registers might be implemented as memory attached to the processor running the software. If the distance calculation engine 204 is

implemented in hardware the FV registers might be implemented as logic or as a memory block. The FV registers 209 are loaded from the FV memory 208 when the Distance Calculation Engine 204 has completed distance calculations for the current feature vector and a subsequent feature vector is available in the FV memory 208. In some embodiments there is an optional FIFO associated with FV registers 209 so that the FIFO is loaded from the FV memory 208 when there is a subsequent feature vector in the FV memory 208 and there is a space in the FIFO. In these embodiments the FV Registers 209 are loaded from the associated optional FIFO 209 when the Distance Calculation Engine 204 has completed calculations for the current feature vector and there is a subsequent feature vector in the associated optional FIFO 209. The presence of the FIFO 209 provides for a looser coupling between the Front End 103 and the accelerator 200 which improves performance in some system configurations.

Each feature vector is transferred to a distance calculation engine circuit 204, to obtain distances for each state of the acoustic model. The acoustic model is stored in an acoustic model memory 205, which is connected to the bus 211, for control of reading/writing to the memory. In some embodiments, this also allows acoustic models to be changed or updated via the bus. The acoustic model may be stored in compressed form in the acoustic model memory 205. A decompressor 206 is provided on the data output of the acoustic model memory. The decompressor 206 can decompress the results as they are needed, and thus allows a much smaller size of acoustic memory to be used, than with uncompressed data. From the decompressor 206, the decompressed acoustic model states are sent to the distance calculation engine 204.

The distance calculation engine 204 converts feature vectors and acoustic model states to distances. The accelerator 200 includes two separate results memories for storing these calculated distances, labelled as "result memory A" 201 and "result memory B" 202.

The control, sequencing, and address generation 210 is logic, state machines, and/or software programs that provide overall control of the other elements of the accelerator, taking inputs from the registers 203, and all the other elements and interfaces to determine when to start and stop the Distance Calculation Engine 204. The control 210

also produces addresses and control signals to control the reading and writing of the memories 201, 202, 205 and 208, to control the flow of data between the various elements of the accelerator 200, the operation of the Decompressor 206, CRC check 207 and Distance Calculation Engine 204, and to control the setting or clearing or status bits in the Registers 203 and to control signalling to the front end 103 and search stage 106. The control 210 is implemented using any of the widely known and used methods of controlling and sequencing in digital systems including logic gates, state machines, and embedded software systems. The control 210 might be a centralized control, or it might be distributed in some or all of the other elements of the accelerator 200, or some combination of these.

Each time a feature vector is loaded into the accelerator, the accelerator computes the distances for all states for that feature vector, and stores the results alternately in the A or B Results Memory. The results are written to one of the two memories 201, 202, while the other memory is available for reading. Thus, while the distance calculation engine 204 is storing the distances calculated for one audio frame, the other result memory may make available the distances calculated for an earlier audio frame, to the search stage of the speech recognition system. For non-real time processing, "earlier" audio frame may refer to an audio frame that has been processed earlier, rather than being limited to an audio frame that corresponds to speech at an earlier time.

Thus, the search stage software reads results alternatively from the A or B result memory, while the accelerator writes results into the other result memory, allowing concurrent pipelined operation of the accelerator and the CPU [and the DSP if present].

In some implementations the A and B results memories may be exposed to the software interface as separate memories appearing at different locations in the CPU and/or DSP memory map. Another implementation is to map each memory alternatively to the same range of addresses in the CPU and/or DSP memory map, so that the software always accesses results at the same location, and the memory containing the set of results the software currently needs to access is the one mapped there.

The accelerator may autonomously load and decompress an acoustic model, which is stored in a compressed format in the acoustic model memory 205. The Acoustic Model may be loaded into the Acoustic Model Memory by software running on the CPU prior to the first use of the Accelerator. The Acoustic Model Memory may be an on-chip memory, to maximize bandwidth and minimize power consumption of reading in the entire Acoustic Model every frame, which means 50 to 100 times per second, depending on the system design.

The Acoustic Model Memory may be a dedicated memory attached to, or incorporated within, the accelerator. Alternatively the Acoustic Model Memory may be a shared resource that is used by other system functions or applications when speech recognition is not running. To allow detection of situations where the acoustic model has been overwritten by another application, a pre-calculated CRC (cyclic redundancy check) signature may be stored with the acoustic model. Each time the acoustic model is read into the accelerator, a CRC may be computed and compared with the signature. If the comparison fails, an error bit is set in the accelerator status register.

Some embodiments of the invention relate to speech recognition systems comprising separate front end 103 and search stage 106 processors. These separate processors perform the front-end and search stages of the recognition process, with the accelerator 200 performing the distance calculations. Typically the front-end processing would be performed on a DSP and search stage would be performed on a CPU, although this description holds for any type of processors, even dedicated hardware implementations. As shown in figure 17, the front end 103 supplies the distance accelerator with feature vectors, and the search stage 106 uses distances calculated by the accelerator.

The accelerator is designed to operate in a concurrent pipelined manner with the other processors, allowing a high level of parallelism in the system.

Figure 18 shows how this pipelined operation works in an embodiment of the invention with ideal data flow, i.e. where each of the three recognition stages require exactly the same time to process each data frame, and the processors are never delayed, stalled, or diverted to other tasks.

The time axis is divided into equal "pipeline step times". If the step time is the same as a frame time then the system is processing audio in real time. If the step time is short than a frame time, the system is operating faster than real time, which may be valuable for batch processing of audio data, or catching up after some type of delay. If the step time is loner than a frame time, the system is operating slower than real time, which may be acceptable in some applications, for example where the person dictating pauses occasionally allowing the system to "catch up".

Once the pipeline has filled (which takes two step times), the DSP, the Accelerator, and the CPU are all operating in parallel. In this idealized example, the total time available for processing a frame is three step times – one step time in the DSP, one step time in the accelerator, and one step time in the search stage. There is a latency of three step times from the time the audio data enters the front end to the completion of the search stage for that frame, and the processing of one frame finishes each step time. If the system were not pipelined, then the processing of all stages for one frame would have to complete in one step time in order to achieve the same overall throughput of frames per second. Thus pipelining in this way can achieve three times the throughput with the same processing elements, compared to a non-pipelined system. A non-pipelined system would be one in which the processing of frame $n+1$ would not start until all stages of the processing of frame n was already complete.

If, for example, in a system running in real time with a frame time and step time of $1/100^{th}$ of a second, which is 10ms, each of the three processing elements has the entire 10ms available to complete the processing of one stage of the recognition task, giving 30ms of processing time overall for each frame. If the system were not pipelined, real time operation would require that all three processing elements would be complete in 10ms, allowing only 3.3ms for each of the three processing elements.

At the end of each pipeline step time, the DSP provides a feature vector to the Accelerator, and the accelerator provides a set of distance results to the CPU. When the CPU is performing the search stage for frame n, the Accelerator is computing the distances for frame $n+1$, and the DSP is performing the front-end stage for frame $n+2$.

During each pipeline step, the front end processing consumes one frame of digital audio data (although this is not indicated explicitly on the diagram).

Real world systems are more complex that the idealized system in many ways. The processing time for a frame is often highly variable and data dependent, especially in the search stage, and the processing time for each processing stage is likely to be quite different. The DSP and CPU are likely from time to time to be diverted to run other software tasks, either to service interrupts, or under control of an operating system scheduling regime. Communications between the processing elements may be stalled due to competing bus traffic, for example for DMA transfers.

Recognition software may implement "frame dropping" and other advanced techniques, and as a result not all frames are processed through all of the recognition stages. Processing elements cannot respond instantly to the availability of data from the preceding processing stage, resulting in delays.

Finite buffer space for information storage between stages means that an earlier stage may be stalled waiting for a space to store its output data when following processing stages are "falling behind".

The accelerator has been carefully designed to mitigate these problems. Figure 19 shows a pipelined system with real world complications added. The following text describes the interactions shown in the diagram. The time for the processing of various stages has been varied to show difference examples of buffering of data between stages and stalling of processing elements. The numbering corresponds to the numbered bubbles on the diagram. The text also makes reference to memories used to pass information into and out of the accelerator, which are shown in figure 17.

The front end has computed the feature vector for the first frame, FV1. It is loaded into the accelerator, causing the accelerator to commence computation of the distances for the frame.

The feature vector for the second frame, FV2, is passed to the accelerator. The accelerator is still busy computing the distances for the first frame, so this feature vector is held in the accelerators FV memory.

The front end has computed the feature vector for the third frame, FV3, but it cannot be delivered to the accelerator as the FV memory is still full, holding FV2. The time period labeled "stall delivering FV3" is the period between FV3 being computed and when it can first be delivered. Of course the DSP could be diverted to some other useful task, or it could buffer FV3 in its own memory and commence computation of FV4.

The arrow labeled "FV3 earliest" shows the earliest time that the feature vector FV3 can be delivered to the accelerator, which is immediately after the accelerator loads FV2 into its internal registers and frees the memory.

The arrow labeled "FV3 latest" shows the latest time that the feature vector FV3 can be delivered to the accelerator without causing the accelerator to stall for lack of a feature vector to work on. This example shows that there is a wide tolerance in the system to when FV3 can be delivered without affecting system performance – an entire accelerator frame processing time. The accelerator provides an interrupt and a status bit that can be polled to indicate the state of the FV memory. Regardless of which control method the DSP uses, the system tolerates long latency without reducing performance.

The accelerator signals to the search stage that the distances for frame 1 are available in result memory A. The search stage commences processing for frame 1, reading distances from result memory A.

The search stage has finished its processing for frame 1 before the accelerator has completed computing the distances for frame 2, so the search stage stalls waiting for the accelerator. Of course the CPU could be diverted to other useful tasks, including any search stage tasks that can be completed before distances are available. Again, an interrupt and status bits are provided by the accelerator to control this interaction between the accelerator and the CPU. .

The accelerator completes the distance calculation for frame 2, and signals to the CPU that it can commence reading the distances from result memory B.

The accelerator has completed the computation of distances for frame 3 before the search stage has finished processing frame 2. As soon as the search stage signals to the accelerator that it has completed frame 2 by releasing memory B, the accelerator signals that the distances for frame 3 are available, allowing the search stage to immediately move on to processing stage 3.

The accelerator is stalled waiting for feature vector FV4, due to the long processing time in the front end. In real systems this may be caused by a number of factors, including the DSP being diverted to another task for a period of time, or simply due to the DSP performance being insufficient to keep up with the accelerator's throughput.

The accelerator restarts as soon as FV4 is delivered by the front end.

The search stage is stalled waiting for distances for frame 4, a follow on consequence of the late delivery of FV4 to the accelerator.

FV5 is delivered to the accelerator while it is still computing the distances for the previous frame, allowing it to move onto processing frame 5 without stalling when it completes frame 4.

The accelerator signals that the distances for frame 4 are available and the search stage commences processing frame 4.

FV6 is delivered to the accelerator while it is still computing the distances for frame 5.

The availability of distances for frame 5 is signaled by the accelerator to the CPU.

Even though the accelerator has already received FV6, it is stalled waiting for a free result memory it can write the distances into.

The search stage releases result memory B at the end of its processing of frame 4. This allows the accelerator to start computing distances for frame 6.

The accelerator completes computing distances for frame 6 and signals to the CPU that the results are ready.

Further embodiments of the invention relate to systems with a Front End and Search Stage on a single processor. This single processor performs the front-end and search stages of the recognition process, with the accelerator performing the distance calculations. This configuration is likely to be the most common configuration where the accelerator is added to an embedded system such as a PDA or mobile phone that has a high-performance embedded application processor (but no DSP) available to run speech recognition software.

In this configuration the accelerator still operates in a concurrent pipelined manner with the front-end and search stages; it is just that the front-end and search stages are running on a single processor.

CPUs in embedded systems are generally relatively simple compared to desktop PCs and servers. Embedded CPUs in mobile phones, PDAs, in-car navigation and telematics systems, and other consumer electronics devices are typically "simple uni-processors" by which we mean not multi-processors and not supporting hardware multi-threading support – i.e. they can only execute a single thread of instructions (although they can be diverted to a different thread under the control of interrupts, exceptions, or an operating system). For this discussion, except where explicitly stated otherwise, we are talking about embedded processors that are "simple uni-processors". We are also using the term "CPU" and "processor" interchangeably.

Figure 20 shows how the system can be pipelined with a single embedded processor working with the accelerator, in an idealized system. In this idealized system, the time taken for the processor to perform the front-end and search stage processing is identical

to the time the accelerator takes to perform the distance calculations, and the processor is never delayed, stalled, or diverted to other tasks.

The time axis is divided into equal "pipeline step" times. If the step time is the same as a frame time then the system is processing audio in real time. If the step time is short than a frame time, the system is operating faster than real time, which may be valuable for batch processing of audio data, or catching up after some type of delay. If the step time is loner than a frame time, the system is operating slower than real time, which may be acceptable in some applications, for example where the person dictating pauses occasionally allowing the system to "catch up".

The first two pipeline step times are used to fill the pipeline. By the third pipeline step time, the pipeline is full, and all three stages of the recognition process are processed every pipeline step time, with each stage working on the processing of a different data frame.

During the first pipeline step time, the CPU completes the front-end processing for frame 1, and the feature vector passed to the accelerator so that distance calculations can start in the accelerator.

During the second pipeline step time, the front-end processing for frame 2 is completed by the CPU, in parallel with the accelerator computing the distances for frame 1.

During the third pipeline step time, the CPU runs the search stage for frame 1 reading distance results from the accelerator, and then runs the front-end processing for frame 3. The accelerator runs in parallel computing the distances for frame 2 using the feature vectors computed in the previous pipeline step.

Each subsequent pipeline step time is similar to the third pipeline step, with the frame numbers incremented. So for pipeline step $n = 4, 5, 6$, etc. the CPU runs the search stage for frame $n-2$ and then runs the front-end for frame n. In parallel the accelerator computes the distances for frame $n-1$.

Note that with this situation the production of feature vectors and the consumptions of results run in lock step - one feature vector is produced every time the search stage consumes one frame of distance results. This considerably reduces the need for elastic buffering between the distance accelerator and the software system.

Again, real world systems are more complex than the idealized system in many ways. For example, the processing time for a frame is often highly variable and data dependent, especially in the search stage, and the processing time for each processing stage is likely to be quite different. The CPU is likely from time to time to be diverted to run other software tasks, either to service interrupts, or under control of an operating system scheduling régime. Communications between the CPU and accelerator may be stalled due to competing bus traffic, for example for DMA transfers.

Recognition software may implement "frame dropping" and other advanced techniques, and as a result not all frames are processed through all of the recognition stages. Processing elements cannot respond instantly to the availability of data from the preceding processing stage, resulting in delays

Finite buffer space for information storage between stages means that an earlier stage may be stalled waiting for a space to store its output data when following processing stages are "falling behind".

The accelerator has been carefully designed to mitigate these problems. Figure 21 shows a pipelined system with real world complications added. The following text describes the interactions shown in the diagram. The time for the processing of various stages has been varied to show difference examples of buffering of data between stages and stalling of processing elements. The numbering corresponds to the numbered bubbles on the diagram. The text also makes reference to memories used to pass information into and out of the accelerator, which are shown in figure 21.

At the end of front-end processing for frame 1, the CPU passes the feature vectors for frame 1 to the accelerator, allowing distance calculations to start.

To fill the pipeline, the CPU immediately performs the front-end processing for frame 2 and passes the feature vectors to the accelerator. They are stored in the accelerator so that the accelerator can load them into its internal engine for processing as soon as the distance calculations for frame 1 are complete.

The CPU is now ready to start the search stage for frame 1, however the distance computation in the accelerator has not completed. The box labeled "Prep search stage 1" indicates a period where the CPU could do any preparatory search stage processing that does not require distances. Alternatively the CPU could be diverted to another task, or it can stall waiting for the distance results.

The accelerator signals that the distance results for frame 1 are now available. The search stage can commence on the CPU, reading distance results from the accelerator's Result Memory A. The accelerator automatically starts processing the feature vector for frame 2 that was delivered earlier.

The accelerator signals to the CPU that it has completed the distance calculations for frame 2. The CPU is still busy running the search stage for frame 1, and has still to run the front end for frame 3, so the completion signal is ignored until the CPU has need of it, when it starts running the search stage for frame 2. As the accelerator has not received the feature vector for frame 3, it has no task it can perform and so it stalls.

The CPU delivers the feature vector for frame 3, restarting the accelerator, which writes results into Result Memory A. Note that Result Memory A was previously "released" at the end of the processing of search stage 1, informing the accelerator that it can overwrite the old results held in that memory with new results. The CPU then runs the search stage for frame 2, reading distance results from the accelerator's Result Memory B.

The accelerator signals to the CPU that the distance results for frame 3 are complete in Result Memory A. Again, the CPU ignores this signal until it is ready to commence the search stage for frame 3, and the accelerator stalls waiting for the next feature vector.

The CPU delivers the feature vector for frame 4, restarting the accelerator. It them starts search stage 3 reading distances from the accelerator's result memory A.

Search stage 3 and front end 5 were completed quickly, so that the feature vector for frame 5 is delivered to the accelerator while it is still running. The CPU performs all the preparatory work it can for the search stage, and then stalls waiting for distances to be available.

The accelerator signals that the distances for frame 4 are available. The CPU starts search stage 4, reading the distances from the accelerator's result memory B. The accelerator automatically starts computing distances for frame 5 using the feature vector delivered earlier.

Note that with the pipelining scheme just presented which alternates front-end and search stage processing, the accelerator can never stall waiting for a result memory to be released because the feature vector the accelerator needs as an input is not produced until after the memory has been cleared. The CPU task schedule could be described with the shorthand F1 F2 S1 F3 S2 F4 S3 F5 S4.

It can seen readily from figure 21 and the accompanying description above, that with this CPU task schedule, that the operation of the CPU and accelerator are tightly coupled and whenever the processing time for a pipeline step on the CPU is unequal to the processing time on the accelerator, then it quickly results in one of them stalling.

In a system where either the CPU or the accelerator is always (or at least in most instances) slower than the other in completing its processing task, stalling is inevitable and throughput may not be improved by adding additional buffering (e.g. additional result memories or FIFO depth on the feature vector interface). [Unless there are other difficulties to mitigate such as very low-bandwidth buses linking the CPU to the accelerator, or competing bus traffic such as DMA transfers that can significantly reduce performance.

If the processing time for a pipeline step is highly variable on either the CPU or Accelerator, such that there are regular changes in which one of them takes the longer time, then significant enhancements in throughput may be obtained by adding additional buffering between them. This may have the disadvantage of adding additional latency through the system. This buffering may be implemented by adding additional depth to the Feature Vector FIFO, and/or adding additional Result Memories. Alternatively it may be by implementing elastic buffers in software.

If the accelerator has sufficient buffering on the feature vector interface, e.g. by including the optional FIFO shown in figure 21, then increasing the pipeline depth by computing extra front end frames up front will improve throughput at a cost of extra latency. This will result in a CPU task schedule of F1 F2 F3 S1 F4 S2 F5 S3 F6 S4 or F1 F2 F3 F4 S1 F5 S2 F6 S3 F7 S4, etc.

Of course it would also be possible to dynamically schedule whether the CPU should run the front-end or search stage code based on the availability, or not, of distance results and/or the space available to store more feature vectors and distance results. Given the fundamental relationship that one set of distance results is produced by the accelerator for every feature vector it consumes, and the restriction stated at the start of this section that we are discussing single-threaded CPUs, dynamic scheduling is likely to add little performance improvement while being complex to implement.

On a CPU supporting multi-threaded operation (which is a technique for improving performance in the presence of high memory system latency), or on a multi-processor, the highest system performance is likely to be achieved by running the front-end and search stages concurrently as separate threads, rather than alternating between the front-end and search stages. The system operation is then very similar to that described above, if each thread is thought of as a separate virtual processor. In some embodiments of the invention, elastic buffers may be used between stages, to accommodate varying time delays between the processing of frames at each stage. Elastic buffers in the interfaces between the three recognition stages may also significantly enhance performance of such a system.

Operation of the accelerator and the software system is not constrained to a rigid repetitive frame time. Instead the timing of the system is controlled dynamically by the availability of feature vectors into the accelerator, and the consumption of results out of the accelerator.

This allows the recognition process to fall behind real-time operation and to "catch up" again. It also allows the system to be robust in complex systems where interrupts, DMA bus traffic, memory caches, operating system task switching, and competing real-time software tasks all make guaranteeing rigid timing constraints impractical.

Elastic buffers at the major data transfer points between processing elements maximize system performance by allowing one element to continue useful processing whenever it can, regardless of whether other processing elements have been stalled or diverted to other tasks. Elastic buffers are useful for the following applications:

(1) For buffering raw audio data arriving in real time from the microphone(s) The data would typically be moved into the buffer by a DMA and/or high priority interrupt driven software process to ensure that hard real time constraints are met.

(2) For buffering feature vectors output from the front end. This is especially true if the front-end is implemented on a DSP or other processing element independent of the search stage.

(3) For buffering distance results from the accelerator. Again this is especially true if the front-end and search stages are implemented on independent processing elements. This buffering is also especially valuable if the interface between the search stage processor and the accelerator is very slow, has high latency, or is subject to competing bus traffic.

The Feature Vector and Results interfaces on the accelerator already provide a level of elasticity that should be sufficient in systems where the Front End and Search stages are running on the same CPU, and the accelerator has a high-bandwidth interface to the CPU with low level of competition from other bus traffic such as DMA.

Other embodiments may have deeper FIFOs. In one embodiment, the two result memories are an implementation of a FIFO where the implementation is exposed across the interface. It would be possible to have additional memories and use them in a round-robin fashion, or to use another implementation of a FIFO that hides the FIFO depth by only exposing one result memory at a time to the interface. An example of this would be to map each memory in turn to a memory range that is always used to read results, so that at any one time the "current" result memory is mapped to that location. This would hide the FIFO depth behind the interface so that the pipeline depth can be changed without changing the software (or hardware) that accesses the interface.

Slow or Conflicted Bus Interfaces may occur. In the system configurations described above, the search stage reads distances directly from the Accelerator's Result Memories. The result memories allow the search stage to randomly access the distance results, as they are needed. This will only give a high performance system if the interface or bus between the CPU and Accelerator provides the CPU with low latency, high bandwidth access to the Accelerator Result Memories.

In some systems it is not practical to provide such a low latency, high bandwidth interface between the CPU and the Accelerator. This may be for many reasons, including:

- The interface is a bus shared with other devices and/or processors that generate bus traffic that keeps the bus busy such that the CPU access to the accelerator is delayed waiting for the bus to be free. DMA transfers are one example of conflicting bus traffic.
- The interface is implemented as a serial bus, or narrow parallel bus, for cost or compatibility reasons, yielding low bandwidth and high latency
- The interface crosses an industry standard interface such as PCI, USB, SDIO, Cardbus, Firewire, PCIExpress etc. that introduces protocol overheads and/or latency as would be the case where the accelerator is in a plug-in peripheral to a system containing a CPU (or CPU and DSP).

In these situations it is possible to add another pipeline stage during which the result data is copied from the Accelerator into another memory (which we will call a "buffer memory") to which the CPU does have high bandwidth, low latency access. This copying can be performed by a DMA controller, or by software running as an interrupt handler, or a separate software thread on the CPU, or a software thread on another processor, or by other commonly known means. The device controlling the transfer may be on the CPU side of the interface or on the accelerator side of the interface, depending on the capabilities of the interface. This extra pipeline stage then allows an entire pipeline step time to copy the data from the accelerator to the other memory. The data copy can optionally be performed as a sequential burst, or a number of sequential bursts, to optimize transfers across the interface between the Accelerator and the buffer memory. The search stage, running on the CPU, can then randomly access the results in the buffer memory with low latency.

Similarly an extra pipeline stage can be added for transferring the Feature Vectors to the Accelerator, although given that the feature vectors are a much smaller object than the distance results, and that they are not accessed randomly across the interface but delivered once to the Accelerator for each frame, there are fewer situations where the performance benefit from this extra pipeline stage is as significant.

Figures 22 and 23 show idealized systems with the two extra pipeline stages for transfers added. The first shows a system with a CPU, DSP, and Accelerator with the two extra pipeline stages. The second shows a system with a CPU and DSP with the two pipeline stages added.

Other embodiments may add additional pipeline stages so that the distance computations are performed over two or more pipeline stages rather than being restricted to a single pipeline stage. This allows more time for the computation of the distance results for each frame, while maintaining the same throughput of distance results. This is similar in concept to the well-known pipelining techniques used to implement RISC microprocessors.

The Acoustic Models may occupy many megabytes of storage space. In embedded systems they will typically held in a non-volatile memory such as FLASH memory, or other types of non-volatile memory as is the appropriate currently economic technology. The Accelerator may read the acoustic model directly from the non-volatile memory, or more likely the acoustic model will be copied from the non-volatile memory into a higher bandwidth RAM memory of some type (e.g. SRAM or DRAM, SDRAM, etc.). The RAM memory may be part of the same integrated circuit as the Accelerator (i.e. "on chip"), or attached to the integrated circuit containing the Accelerator. ("off chip"). The having the RAM memory on-chip is particularly advantageous, as this configuration would give the lowest power consumption and highest bandwidth. An alternative solution would be for the acoustic model to be stored on a disk drive, or an another device accessible via a wired or wireless bus or network, and for the acoustic model to be loaded into the RAM memory from the disk drive, or over the wired or wireless bus or network.

On-chip RAM memory is however likely to come at a cost-premium to off-chip commodity memory devices. Many embedded systems face extreme cost pressures in the market place, especially high volume devices supplied by many manufacturers such as mobile phones. For a speech recognizer to be incorporated into such systems, the overall recognizer solution much be delivered at very low cost. There is therefore a significant economic benefit from reducing the cost of the RAM memory and the non-volatile memory. One way to reduce the cost of these memories is to compress the acoustic models and store the compressed model in the memory.

The Accelerator computes all of the distances associated with each Feature Vector that is delivered to it by the front end. This means that the accelerator must use the data from the entire Acoustic Model each time a new Feature Vector is presented to the Accelerator (with the exception of some specific optimizations that may determine that some data values need not be used in the distance computations). It is therefore possible to arrange for the accelerator to read through the acoustic data sequentially once for each Feature Vector, using each data value as it read from the Acoustic Model.

Reading through the entire Acoustic Model sequentially is ideally suited to placing a decompressor between the distance calculation engine and the RAM memory, because many compression schemes depend on the decompressor reading sequentially through the compressed data, and do not readily support random access into the compressed data.

Thus this configuration allows storing compressed data in the RAM memory (and the non-volatile memory), which can therefore be smaller and lower cost. There is no requirement for a memory that is large enough to hold the entire decompressed acoustic model, as the decompressed data is consumed as it is decompressed.

The decompression scheme could be any combination of:

- sign or zero extension or otherwise conversion of narrow or variable width data to a wider data format.
- sign or zero extension or otherwise conversion of narrow or variable width data to IEEE standard single or double precision floating point format
- codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data
- decompression of a Huffman or Lempel-Ziv compressed stream
- decompression of run length encoded data
- decompression of difference encoded data
- any well known compression scheme.

Additionally the model size can be reduced by using *subspace distribution clustering* and the decompressor and distance calculation engine designed to operate with such models.

The following description provides further information on the visible registers in some embodiments of the invention.

The register bank is designed to allow simple connection with an 8-bit data bus. Accesses to the register bank should be made with byte-read and byte-write accesses. To facilitate this, only the least significant 8 bits of the data bus are used.

The address decoder should respond so to byte accesses to byte 0 or byte 3 within the designated word. This way the same code can be used on an ARM processor regardless of whether the processor is configured for little endian or big endian operation.

REGBASE=Base address of register bank

Register 0 Product Version

8-bit Product version ID

This register value is changed when the functionality of the hardware charges in a way that changes the hardware/software interface.

Address = REGBASE (and REGBASE+3 for big endian)

Read only. Reads with value 0x01

Writes ignored

Register 1 Implementation Version

8-bit Implementation version ID

This register value is changed for each implementation to facilitate bug tracking and version control.

Address = REGBASE+4 (and REGBASE+4+3 for big endian)

Read only. Reads with value 0x01

Writes ignored

Register 2 Interrupt Enable

8-bit Interrupt Enable register

Address = REGBASE+4*2 (and REGBASE+4*2+3 for big endian)

Readable and writeable.

Value following reset is 0 for bits 0,1,2,3

Bit 0 I 1 = Enable FV memory empty interrupt on Int0 output.

Bit 1 J 1 = Enable Results complete interrupt on Int0 output.

Bit 2 K 1 = Enable FV memory empty interrupt on Int1 output.

Bit 3 L 1 = Enable Results complete interrupt on Int1 output.

Bit 4 Future use. Read value is not defined. Writes ignored.

Bit 5 Future use. Read value is not defined. Writes ignored.

Bit 6 Future use. Read value is not defined. Writes ignored.

Bit 7 Future use. Read value is not defined. Writes ignored.

Software should use a read-modify-write code sequence when setting this register so that future use bits are written with the value read from them.

Register 3 Status

Address = REGBASE+4*3 (and REGBASE+4*3+3 for big endian)

Read-only register. Writes are ignored.

Value following reset is 0 for bits 0,1,2,3

Bit 0 FVMB Feature Vector Memory is Busy

0 = OK to write new value to FV memory (FV memory is empty)

1 = Don't write new value to FV memory (FV memory is busy)

Bit 1 CRM Current Result Memory

0 = Result memory A

1 = Result memory B

This bit toggles every time the Release Result Memory (RRM) command is issued to the accelerator (register 3). It indicates which memory software should read results from, and which result memory the RRM command should release next time it is issued. Note: Software should also check that results are available in the indicated result memory by checking the value of the MA and MB bits.

Bit 2 MA 1=Results are available is result memory A

The MA bit is set to 1 when the compute engine completes writing all of the distances for a frame into result memory A. This value is set to zero when the Release Result Memory command is issued and CRM==0.

Bit 3 MB 1= Results are available is result memory B
The MB bit is set to 1 when the compute engine completes writing all of the distances for a frame into result memory B. This value is set to zero when the Release Result Memory command is issued and CRM==1.

Bit 4 Dfault 1=Accoustic data CRC fault. Indicates that the CRC value at the end of the acoustic data did not match the CRC value computed while the acoustic data was read in.

Bit 5 Future use. Value undefined.
Bit 6 Future use. Value undefined.
Bit 7 Future use. Value undefined.

Register 4 Command

Address = REGBASE+4*4 (and REGBASE+4*4+3 for big endian)
Write-only register.
Sofware should perform byte writes to this register with a single bit set to 1 and all other bits set to zero to issue a command to the accelerator.
Behaviour is not defined if more than one command is issued within a single write (i.e. if more than 1 bit is a 1).
Value on read is not defined.
Bit 0 LFV Load Feature Vector from the FV memory.
Tells the accelerator that a new feature vector has been placed into FV memory ready for use by the accelerator. The FV will be loaded into the compute engine when the compute engine completes its current task (or immediately if the compute engine is idle). Computation of distances for this FV will commence when the FV is loaded AND there is a released result memory. (At reset, both result memories are released). The results for the first, third, fifth etc. feature vectors loaded are written to Result Memory

A. The results for the second, fourth, sixth etc. feature vectors loaded are written to Result Memory B.

Bit 1 RRM Release Result memory. Indicates that the results in the current result memory can now be overwritten with new results. The current result memory is indicated by the CRM bit in the status register.
Bit 2 CFVI Clear the FV memory empty interrupt
Bit 3 CRI Clear the Results Complete interrupt
Bit 4 Future use – write 0
Bit 5 Future use – write 0
Bit 6 Future use – write 0
Bit 7 Future use – write 0

The present invention may be included in a mobile phone, or in another consumer electronics device, a PC, Laptop or tablet PC, in-car electronics system, or plug-in card for an electronic device.

The methods and apparatus according to the invention are not limited to embedded mobile systems, but may be used with any type of computing device, including PCs, workstations, laptop computers, notebook computers, handheld computing devices, dictation machines, PDAs (personal digital assistants), personal organisers, mobile phones, games consoles, video players, etc. Embedded computers according to the present invention may be embedded in a wide range of apparatus and machinery, such as electronic entertainment products, domestic appliances, factory machinery, vehicles, etc.

Embodiments of the present invention may be implemented as an IP core in a System on Chip (SOC) or a discrete device. Multiple chips or bare die may be connected to provide a larger vocabulary system.

Although the apparatus may include all or some of an input for accepting audio input, a circuit for digitising the audio input, a circuit for dividing the audio input into a plurality of time-frames, and a circuit for analysing the spectral components of each time frame.

to characterise the audio in that frame, this is not essential. The apparatus may also include a circuit for calculating a score indicating the probability or likelihood that each audio frame corresponds to each of a plurality of different possible states, but again, this is not essential. Instead, the apparatus may include an input for receiving the pre-calculated scores.

In the described embodiments, n-gram language models are used. For example, the Bigram language model is based on outputting the probability that a given second word directly follows a given first word. However, it is possible to use alternative language models instead. It is possible to use a mixture of different models. For example, it is possible to run a word-dependent n-best decode applying a bigram language model, and the resulting word link record or word graph could then be rescored using a trigram language model.

The Viterbi beam search algorithm may be used in embodiments of the invention, to perform the search. A word level search may be performed. Dummy states may be used for the start and end of an utterance

The grammar level model may be used to add distance elements into the likelihood calculation. This is not essential, but tends to give much better results.

The HMM models described in the embodiments comprise self connections to each states and single connections to a neighbouring state. However, the present invention is not limited to such HMM models, and for example, HMM models with additional connections between states may also be used. Alternatively, embodiments of the invention may use alternative types of acoustic models, rather than HMM models.

The system may comprise a plurality of language models (e.g. English, French). A model of words may comprise actual words, phrases, silence, lipsmack sounds, and/or other sounds for recognition by the apparatus.

As previously discussed, the phone level has been found to be the best level for lexical tree searching, rather than the state level or the word level. However, this invention is not limited solely to phone level searching.

Pipelining may comprises processing data in different parts of a circuit at the same time, i.e. parallel processing.

In the following claims, the calculation means may comprise the distance calculation engine described above.

Further embodiments of the invention include the methods carried out in the accelerator, and the methods carried out in the speech recognition circuit.

While the invention has been described in terms of what are at present its preferred embodiments, it will be apparent to those skilled in the art that various changes can be made to the preferred embodiments without departing from the scope of the invention, which is defined by the claims.

CLAIMS:

1. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

a calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating circuit is implemented using a second processor, and wherein data is pipelined from the front end to the calculating circuit to the search stage.

2. A speech recognition circuit as claimed in claim 1, wherein the pipelining comprises alternating of front end and search stage processing on the first processor.

3. A speech recognition circuit as claimed in claim 1, comprising dynamic scheduling whether the first processor should run the front end or search stage code, based on availability or unavailablity of distance results and/or availability of space for storing more feature vectors and/or distance results.

4. A speech recognition circuit as claimed in claim 1, wherein the first processor supports multi-threaded operation, and runs the search stage and front ends as separate threads.

5. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

a buffer memory between the calculating circuit and the search stage, for receiving data passing from the calculating circuit to the search stage, wherein a processor in the search stage has higher bandwidth and/or lower latency access to the buffer compared to the bandwidth and/or latency of direct transfer between the calculating circuit and the search stage.

6. The speech recognition circuit of claim 5, wherein the data transfer from the calculating circuit to the buffer memory and/or from the buffer memory to the search stage is performed as one or more sequential bursts.

7. The speech recognition circuit of claim 5 or claim 6, wherein the data transfer to the buffer memory is performed in parallel with data transfer to the calculating circuit and/or in parallel with data transfer to the search stage.

8. The speech recognition circuit of any one of claims 5 to 7, further comprising a second buffer memory between the audio front end and the calculating circuit.

9. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

comprising an elastic buffer between at least one of the front end and calculating circuit, or the calculating circuit and search stage, and/or for buffering said audio signal.

10. An accelerator for calculating distances for a speech recognition circuit, the accelerator comprising:

calculating circuit for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

first and second storage circuit, each for storing calculated distances for at least one said audio time frame, and for making said stored distances available for use by another part of the speech recognition circuit;

control circuit for controlling read and write access to the first and second storage circuit, said control means being configured to allow writing to one said storage means while the other said storage means is available for reading, to allow first calculated distances for one audio time frame to be written to one said storage means while second calculated distances for an earlier audio time frame are made available for reading from the other said storage means.

11. A accelerator as claimed in claim 10, wherein the control means is configured to alternate which of the first and second storage means is available for reading, and which of the first and second storage means is available for writing, as each new feature vector is processed by the calculating circuit.

12. The accelerator of claim 10, wherein the distance is stored for each of a plurality of states.

13. The accelerator of any one of claims 10 to 12, wherein the first and second storage means are interfaced to a processor in the speech recognition circuit as different memory locations within the processor's memory map.

14. The accelerator of any one of claims 10 to 12, wherein the first and second storage means are interfaced to a processor in the speech recognition circuit as being mapped to a first fixed memory address when configured for reading, and/or to a second fixed memory address when configured for writing.

15. The accelerator of any one of claim 10 to 14, comprising releasing a result memory after the search stage has processed the contents, by informing the accelerator that it can overwrite the results in said memory.

16. The accelerator of any one of claims 10 to 15, comprising additional **results memories**.

17. The accelerator of any one of claims 10 to 14, comprising random access of the distance results in the **results memories** as they are needed by the search stage.

18. The accelerator of any one of claims 10 to 17, wherein the additional **results memories** are accessed in a round-robin fashion.

19. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

means for receiving or storing compressed data representing said acoustic model;

a decompressor for decompressing said compressed data for all states or selected states of the acoustic model, wherein the decompressed data is sent to the calculating means; and

output means for outputting calculated distances to another part of the speech recognition circuit.

20. The accelerator of claim 19, wherein the decompression scheme comprises one or more of the following:

sign or zero extension or otherwise conversion of narrow or variable width data to a wider data format;

sign or zero extension or otherwise conversion of narrow or variable width data to IEEE standard single or double precision floating point format;

codebook decompression of a binary bitstream, where the codebook is stored as part of the acoustic model data;

decompression of a Huffman or LPZ compressed stream;

decompression of run length encoded data;

decompression of difference encoded data; and

any well known compression scheme.

21. The accelerator of claim 19 or claim 20, wherein the decompressor is configured to operate with an acoustic model that uses subspace distribution clustering.

22. The accelerator of any one of claims 19 to 21, wherein acoustic states are decompressed one or more times for each feature vector processed by the accelerator.

23. A accelerator as claimed in claim 22, wherein the computed signature is a cyclic redundancy check (CRC) signature.

24. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

means for comparing a first version of a stored checksum of data representing said acoustic model and a second version of said stored checksum, wherein the second version is obtained from an updated measurement and calculation of the checksum; and

means for indicating an error status if the checksums do not match.

25. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame;

wherein said accelerator is configured to autonomously compute distances for every acoustic state defined by the acoustic model.

26. A accelerator for a speech recognition circuit, the accelerator comprising:

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model, wherein the feature vector comprises a plurality of extracted and/or derived quantities from an audio signal during a defined audio time frame; and

a memory for storing said acoustic model,

wherein the calculating means and the memory comprise part of the same physical circuit.

27. A accelerator according to claim 26, wherein said same physical circuit comprises a removable unit for removably connecting into the speech processing circuit.

28. A accelerator according to claim 26 or 27, wherein said memory is configured to be shared between the accelerator and another part of the speech recognition circuit.

29. A accelerator according to claim 28, further comprising means for generating a checksum or computed signature for the acoustic model data stored in the memory, and means for comparing checksums or computed signatures that have been calculated at different times, to indicate an error status if the checksums do not match.

30. A accelerator according to any one of claims 26 to 29, wherein the acoustic model memory is a RAM or flash memory.

31. The accelerator of any one of claims 1 to 30, wherein said distance comprises a Mahalanobis distance.

32. The accelerator of any one of the previous claims, being provided in the form of a separate physical circuit, device or package that is removably connectable to said speech recognition circuit.

33. A speech recognition circuit comprising the accelerator of any previous claim.

34. The speech recognition circuit of claim 33, comprising a shared processor for the front end and search stages.

35. The speech recognition circuit of claim 33, comprising a digital signal processor for the front end, and a processor for the search stage.

36. The speech recognition circuit of any one of the previous claims, comprising control means adapted to implement frame dropping, to discard one or more audio time frames.

37. The speech recognition circuit of any one of the previous claims, wherein the feature vector comprises a plurality of spectral components of an audio signal for a predetermined time frame.

38. The speech recognition circuit of any one of the previous claims, wherein the processor is configured to divert to another task if the data flow stalls.

39. The speech recognition circuit of any one of the previous claims, wherein the speech accelerator has an interrupt and status bit that can be polled to indicate the state of the feature vector memory

40. The speech recognition circuit of any one of the previous claims, wherein the accelerator signals to the search stage when distances for a new frame are available in a results memory

41. The speech recognition circuit of any one of the previous claims, wherein the search stage processor ignores the completion signal until the processor is ready to accept new distance data.

42. The speech recognition circuit of any one of the previous claims, comprising increasing the pipeline depth by computing extra front end frames in advance.

43. A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end, said calculating means, and said search stage are connected to each other to enable pipelined data flow from one to another.

44. A speech recognition circuit as claimed in any one of the previous claims, wherein the audio front end is configured to input a digital audio signal.

45 A speech recognition method, comprising:

calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating means is implemented using a second processor, and wherein data is pipelined from the front end to the calculating means to the search stage.

46. A method as implemented by the apparatus of any previous claim.

47. A speech recognition circuit comprising:

a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words;

a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and

a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

48. The speech recognition circuit of claim 47, wherein for at least some of the states, the node identifier comprises a pointer to a node identifier for another state.

49. The speech recognition circuit of claim 47 or claim 48, wherein the lexical tree is arranged with each node corresponding to a phone, and each said state corresponds to a phone or group of adjacent phones in the lexical tree.

50. The speech recognition circuit of claim 49, wherein the memory structure comprises a first part for storing phone instance identifiers uniquely identifying phones or groups of phones and for storing said state identifiers, and the memory structure comprises a second part for storing said phone instance identifiers and node identifiers directly referencing the lexical tree.

51. The speech recognition circuit of claim 48 or 49, further comprising a counter for sequentially generating phone instance identifiers as unique identification numbers corresponding to phones or groups of phones in the lexical tree.

52. The speech recognition circuit of any previous claim, further comprising a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

53. The speech recognition circuit of any previous claim, wherein the memory structure is configured to hold a record for each of a plurality of nodes in the lexical tree, the record including said states, a previous node identifier for identifying a previous node in the lexical tree, and information indicating whether said each node is a previous node for another node stored in said memory structure,

the speech recognition circuit including a circuit to identify a first node which is not a previous node of another record, and to provide a new data record corresponding to a new node, the new node being the next node after the first node in the lexical tree, using a score associated with the first node.

54. The speech recognition circuit of claim 53, wherein said record comprises a field for storage of a score corresponding to a previous node.

55. The speech recognition circuit of claim 53 or claim 54, further comprising a circuit for overwriting an old record in the memory structure with said new record according to the value of said score for the old record and according to whether said old record is a previous node to said new node or to another node in said memory structure.

56. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises a plurality of memory elements, the memory structure being adapted for independent lookup of states in each of said memory elements.

57. The speech recognition circuit of claim 56, wherein one of said memory elements is configured to store data relating to a first lexical tree, and a second of said memory elements is configured to store data relating to a second lexical tree, the second lexical tree comprising a different model of words than the first lexical tree.

58. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises at least one content addressable memory and at least one

random access memory, the content addressable memory being configured to output addresses of the random access memory.

59. The speech recognition circuit of any one of the previous claims, further comprising a delay circuit for storing a state identifier having a plurality of occurrences in the memory structure, and storing the score update for said state identifier, the delay circuit being configured to provide said state identifier and score after a lookup for a second state identifier produces no match in the memory structure.

60. The speech recognition circuit of any one of the previous claims, comprising a circuit to set a flag indicating an unfavourable score for all records which are previous records and which have scores outside a predetermined range of values.

61. The speech recognition circuit of any one of the previous claims, wherein said memory structure comprises a first memory for receiving and storing said state identifiers, and a second memory for receiving and storing said node identifier, said first memory being configured for lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores, and the second memory being adapted to store a record identifier for a record in the first memory and a reference to the node of the lexical tree for the record identifier, said second memory storing a subset of said records which are not previous records.

62. A method of speech recognition, comprising

providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, the lexical tree comprising a model of words;

providing scores corresponding to said state identifiers;

receiving and storing state identifiers identified by node identifiers identifying a node or group of adjacent nodes in a memory structure;

looking up said memory structure to identify particular state identifiers;

reading of the scores corresponding to the state identifiers;

receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input;

modifying said scores by adding said score updates to said scores;

writing back of the scores to the memory structure after modification of the scores; and

selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

63. The method of claim 62, wherein for at least some of the states, the node identifier comprises a pointer to a node identifier for another state.

64. The method of claim 62 or claim 63, wherein the lexical tree is arranged with each node corresponding to a phone, and each said state corresponds to a phone or group of adjacent phones in the lexical tree.

65. The method of claim 64, wherein the memory structure comprises a first part for storing phone instance identifiers uniquely identifying phones or groups of phones and for storing said state identifiers, and the memory structure comprises a second part for storing said phone instance identifiers and node identifiers directly referencing the lexical tree.

66. The method of claim 64 or claim 65, further comprising a counter for sequentially generating phone instance identifiers as unique identification numbers corresponding to phones or groups of phones in the lexical tree.

67. The method of any one of claims 62 to 66, further comprising a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

68. The method of any one of claims 62 to 67, wherein the memory structure is configured to hold a record for each of a plurality of nodes in the lexical tree, the record including said states, a previous node identifier for identifying a previous node in the lexical tree, and information indicating whether said each node is a previous node for another node stored in said memory structure,

the speech recognition circuit including a circuit to identify a first node which is not a previous node of another record, and to provide a new data record corresponding to a new node, the new node being the next node after the first node in the lexical tree, using a score associated with the first node.

69. Themethod of claim 68, wherein said record comprises a field for storage of a score corresponding to a previous node.

70. The method of claim 68 or claim 69, further comprising a circuit for overwriting an old record in the memory structure with said new record according to the value of said score for the old record and according to whether said old record is a previous node to said new node or to another node in said memory structure.

71. The method of any one of claims 62 to 70, wherein the memory structure comprises a plurality of memory elements, the memory structure being adapted for independent lookup of states in each of said memory elements.

72. The method of claim 71, wherein one of said memory elements is configured to store data relating to a first lexical tree, and a second of said memory elements is configured to store data relating to a second lexical tree, the second lexical tree comprising a different model of words than the first lexical tree.

73. The method of any one of claims 62 to 72, wherein the memory structure comprises at least one content addressable memory and at least one random access memory, the content addressable memory being configured to output addresses of the random access memory.

74. The method of any one of claims 62 to 73, further comprising a delay circuit for storing a state identifier having a plurality of occurrences in the memory structure, and storing the score update for said state identifier, the delay circuit being configured to provide said state identifier and score after a lookup for a second state identifier produces no match in the memory structure.

75. The method of any one of claims 62 to 74, comprising a circuit to set a flag indicating an unfavourable score for all records which are previous records and which have scores outside a predetermined range of values.

76. The method of any one of claims 62 to 75, wherein said memory structure comprises a first memory for receiving and storing said state identifiers, and a second memory for receiving and storing said node identifier, said first memory being configured for lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores, and the second memory being adapted to store a record identifier for a record in the first memory and a reference to the node of the lexical tree for the record identifier, said second memory storing a subset of said records which are not previous records.

77. A mobile phone including the speech recognition circuit of any one of claims 1 to 76.

78. A portable electronic device including the speech recognition circuit of any one of claims 1 to 77.

79. A speech recognition circuit comprising:

lexical memory containing lexical data for word recognition, said lexical data comprising a lexical tree data structure comprising a model of words;

means for accessing a state model corresponding to each phone or each group of phones in the lexical tree;

a content addressable memory for storing content addressable data for each phone or group of phones, including states corresponding to said phone or group of phones, and for storing an address value for each said phone or group of phones;

a RAM configured to store accumulated scores for each said phone or group of phones, the accumulated scores being addressable by said address value for each said phone or group of phones;

means to obtain scores that each of a plurality of frames of an audio signals corresponds to each of a plurality of said states;

a counter to sequentially search for each said state in the content addressable memory, to obtain the corresponding address value if the state is found in the content addressable memory;

means to use said address value to access an accumulated likelihood and an accumulator to add said likelihood to the accumulated likelihood;

means to use the phones or groups of phones with the highest accumulated scores to obtain a plurality of next phones from the lexical tree which correspond to the next phone; and

output means for outputting a lexical tree path of highest likelihood.

80. A speech recognition apparatus comprising:

a lexical tree having a corresponding state model;

means for obtaining scores of an audio input corresponding to each of a plurality of states in said state model;

a content addressable memory for storing a marker indicating a part of the lexical tree, and one or more states associated with said part of the lexical tree;

a random access memory addressable by the CAM output, to output accumulated scores for states corresponding to said parts of the lexical tree;

adder means for adding likelihood to said accumulated likelihood, to be stored back in the RAM;

81. A speech recognition apparatus, comprising:

a CAM-RAM arrangement for storing records including pointers to a lexical tree, and accumulative scores for states within the lexical tree;

input means for obtaining scores that an audio frame corresponds to a particular state in the lexical tree;

an accumulator for calculating the updated scores and modifying the records in the CAM-RAM accordingly;

output means for outputting a path of highest likelihood in the lexical tree.

82. A speech recognition apparatus substantially as hereinbefore described with reference to any of figures 1 to 15 of the accompanying drawings.

83. A speech recognition method comprising:

storing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and scores corresponding to said state identifiers in a memory structure, the lexical tree comprising a model of words and the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

repeating the following sequence of steps for each of a plurality of incoming frames of an audio signal:

obtaining score updates corresponding to the likelihoods that said frame of the audio signal corresponds to each of a plurality of said states;

accessing said memory structure to obtain scores, updating the scores by adding score updates to the scores, and writing back the updated scores to the memory structure;

determining if scores for states furthest on in the lexical tree correspond to a significant likelihood, and if so, then accessing the lexical tree to determine the next set of possible states; and

storing the next set of possible states and said scores of significant likelihood in the memory structure;

84. A speech recognition circuit comprising:

a circuit for providing state identifiers which identify states corresponding to phones or groups of adjacent phones in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words;

a memory structure for receiving and storing state identifiers and phone instance identifiers uniquely identifying instances of phones or groups of phones in the lexical tree;

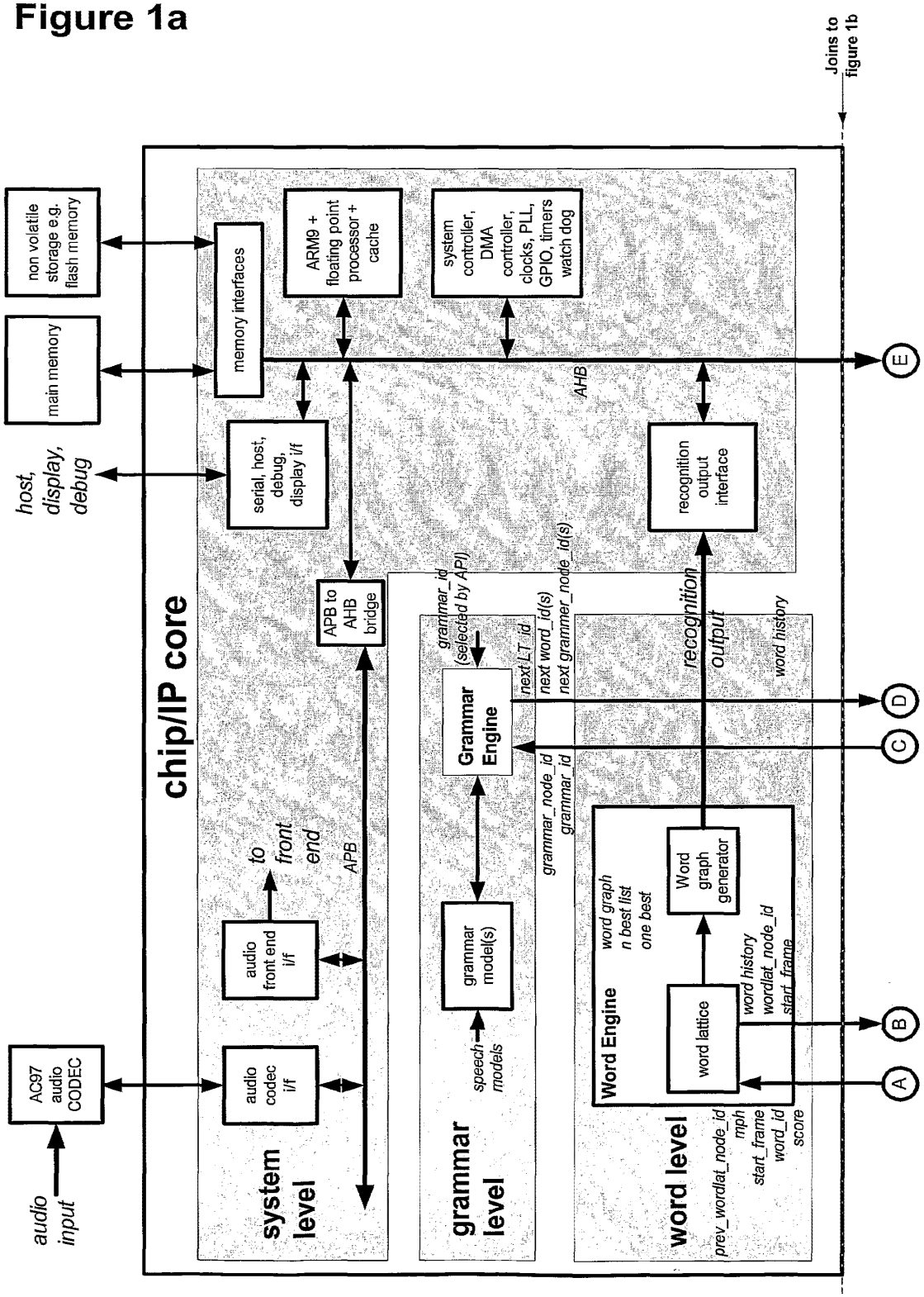
said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates

using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and

a selector circuit for selecting at least one phone instance identifier according to said scores.

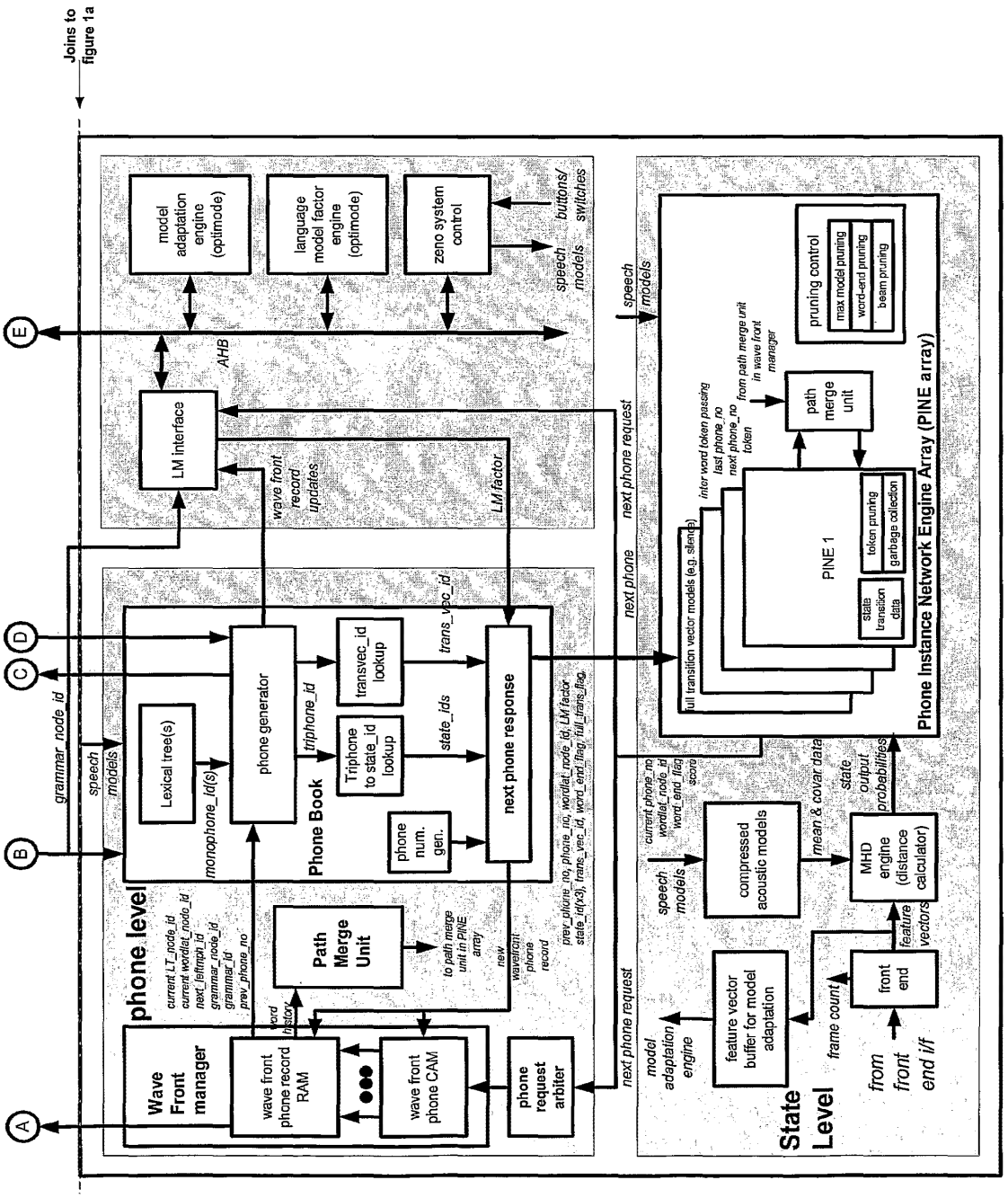
Figure 1a



SUBSTITUTE SHEET (RULE 26)

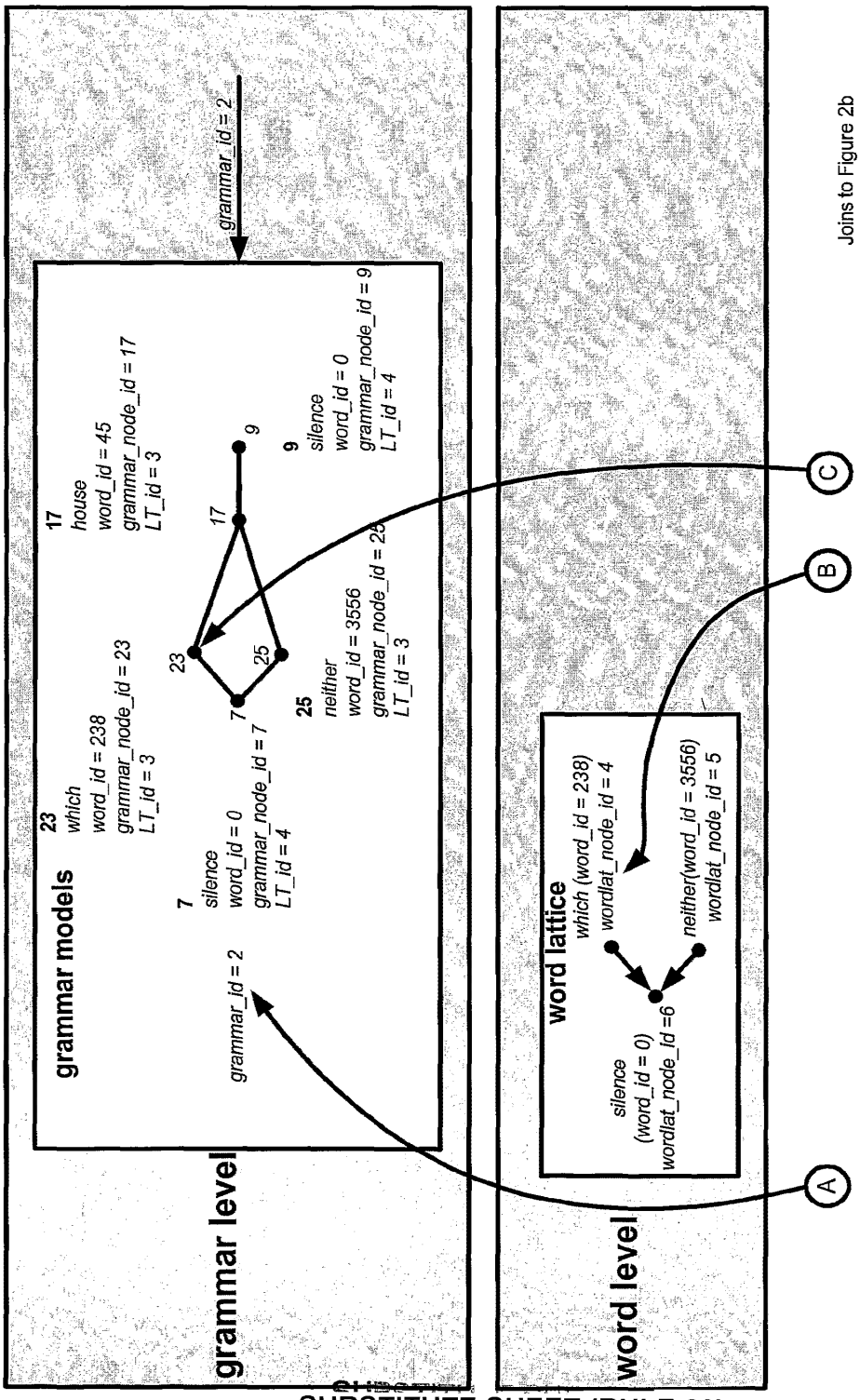
Amazon / Zentian Limited
Exhibit 1002
Page 280

Figure 1b



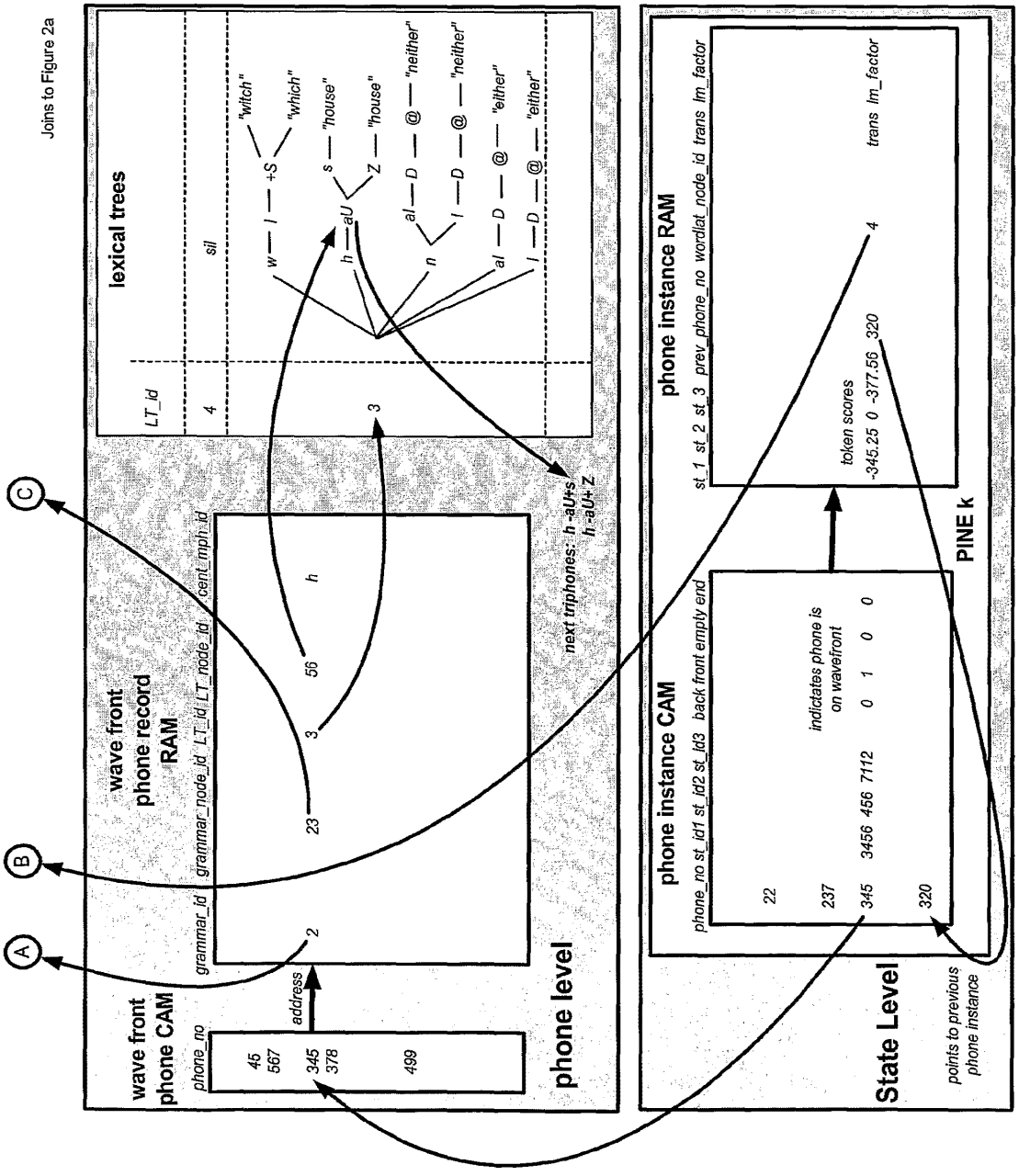
Amazon / Zentian Limited
Exhibit 1002
Page 281

Figure 2a



Amazon / Zentian Limited
Exhibit 1002
Page 282

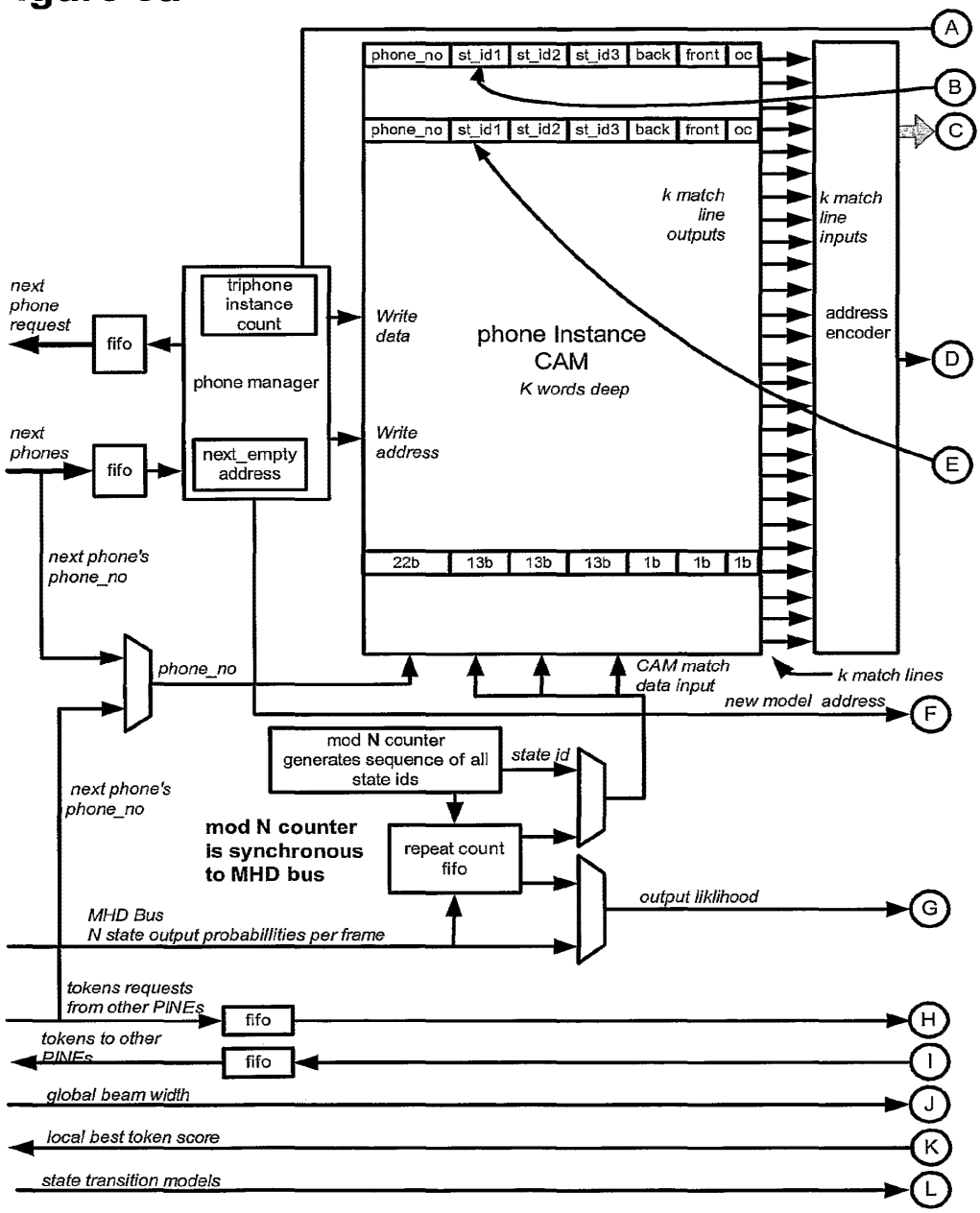
Figure 2b



SUBSTITUTE SHEET (RULE 26)

Amazon / Zentian Limited
Exhibit 1002
Page 283

Figure 3a



CAM abbreviations

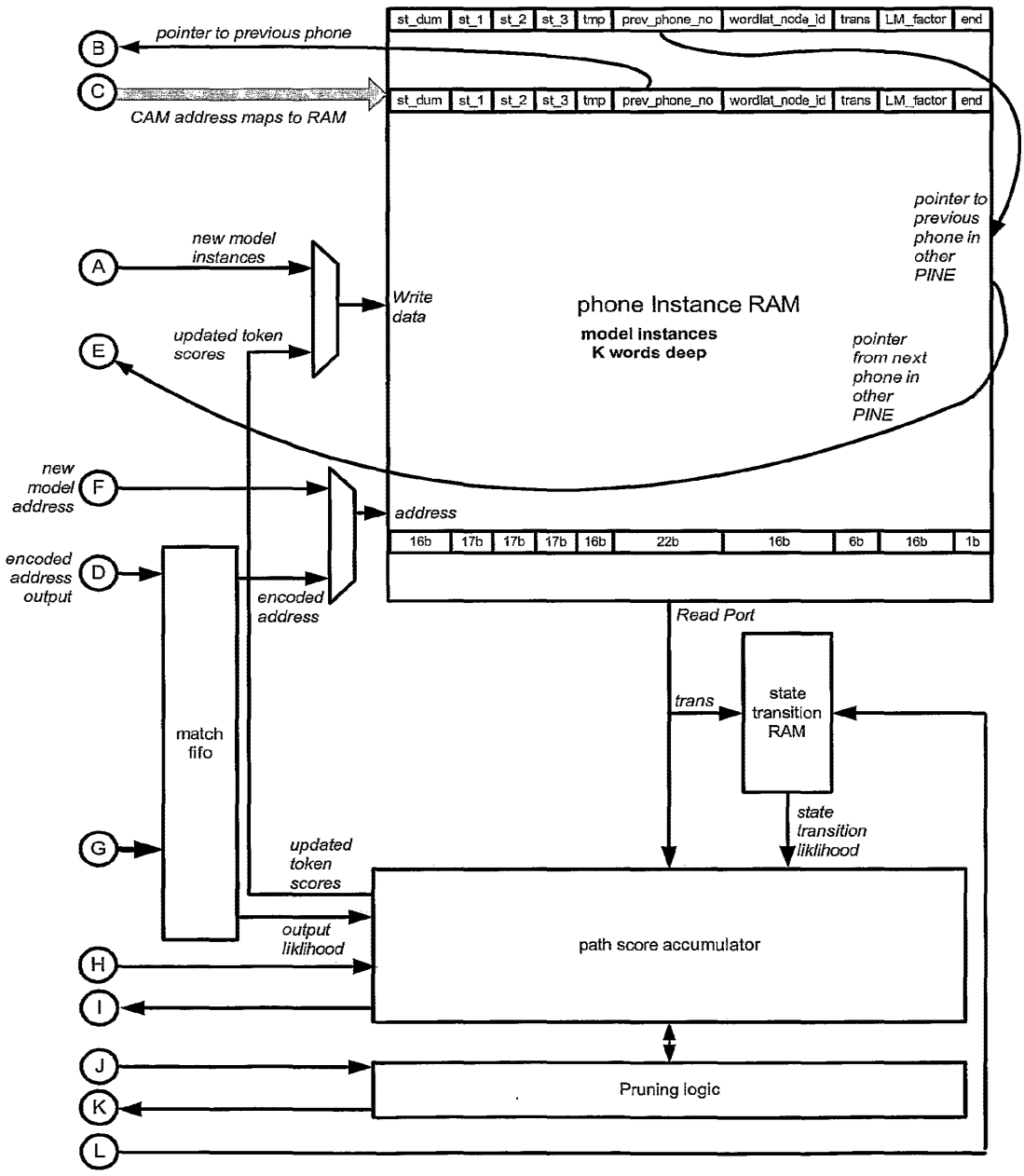
phone_no - unique phone instance indentifier
st_id1 - state_identifier1
back(wave_back_flag) - phone is at back of wave
front(wave_front_flag) - phone is at the front of wave
end(word_end_flag) - phone is the last in a word
oc - occupy - indicates that location has valid phone instance (it may not have any tokens)

RAM abbreviations

st_dum - dummy state to hold incoming token
st_1 - location for token in state 1, includes updated flag
st_2 - location for token in state 2, includes updated flag
st_3 - location for token in state 3, includes updated flag
tmp - location for temporary storage of token
prev_phone_no - pointer to previous phone instance
trans (trans_vec_id) - pointer to state transition vector in state transition RAM
LM_factor - language model factor
end - word end phone

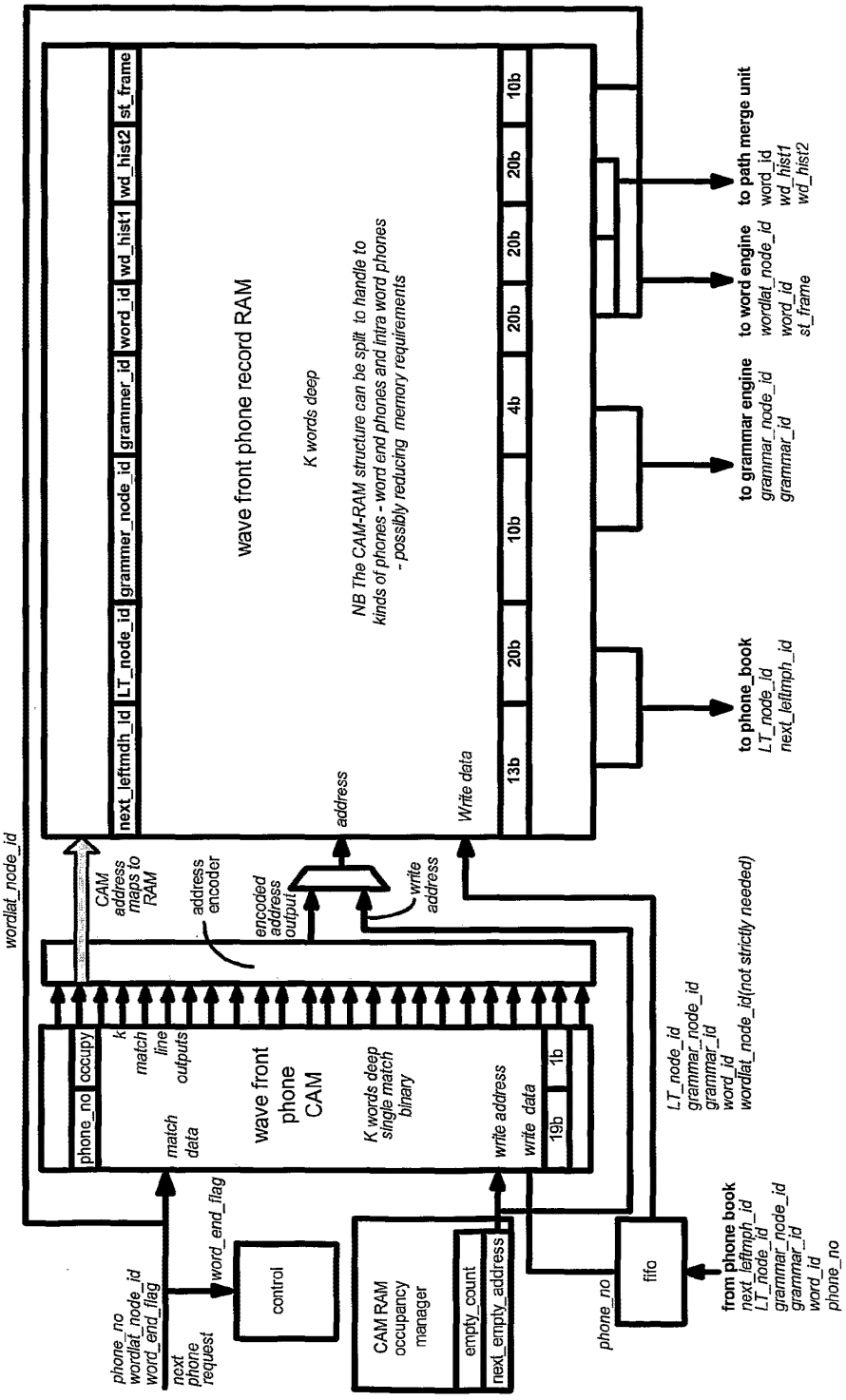
Amazon / Zentian Limited
Exhibit 1002
Page 284

Figure 3b



Amazon / Zentian Limited
Exhibit 1002
Page 285

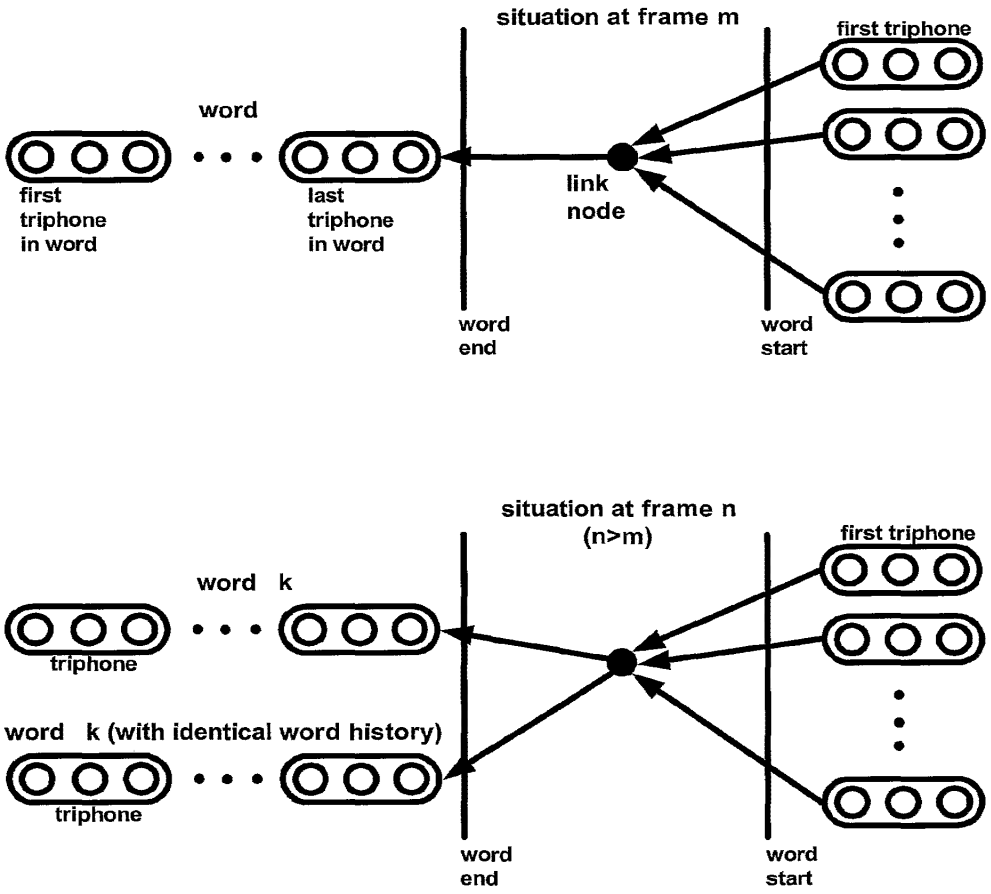
Figure 4



SUBSTITUTE SHEET (RULE 26)

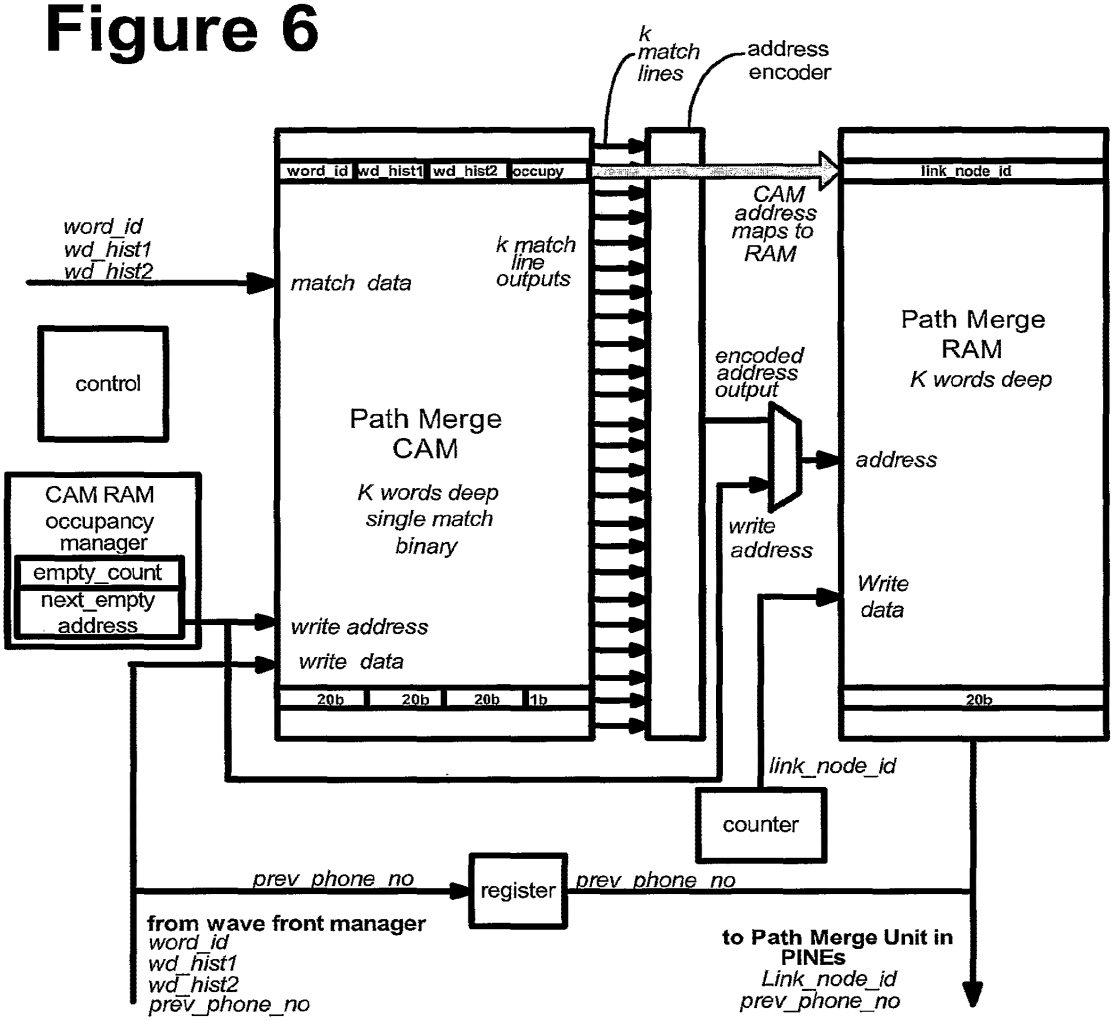
Amazon / Zentian Limited
Exhibit 1002
Page 286

Figure 5



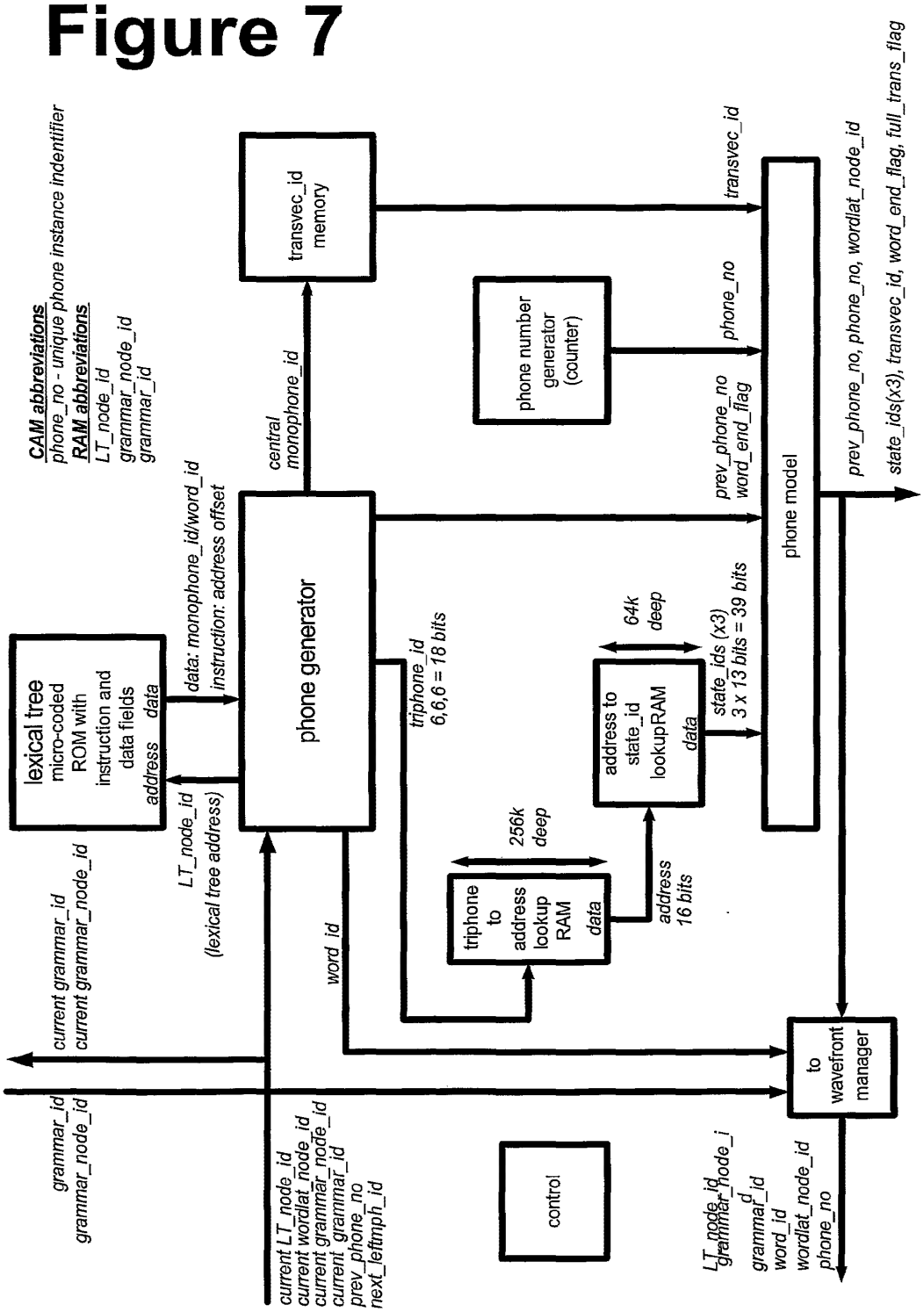
Amazon / Zentian Limited
Exhibit 1002
Page 287

Figure 6



Amazon / Zentian Limited
Exhibit 1002
Page 288

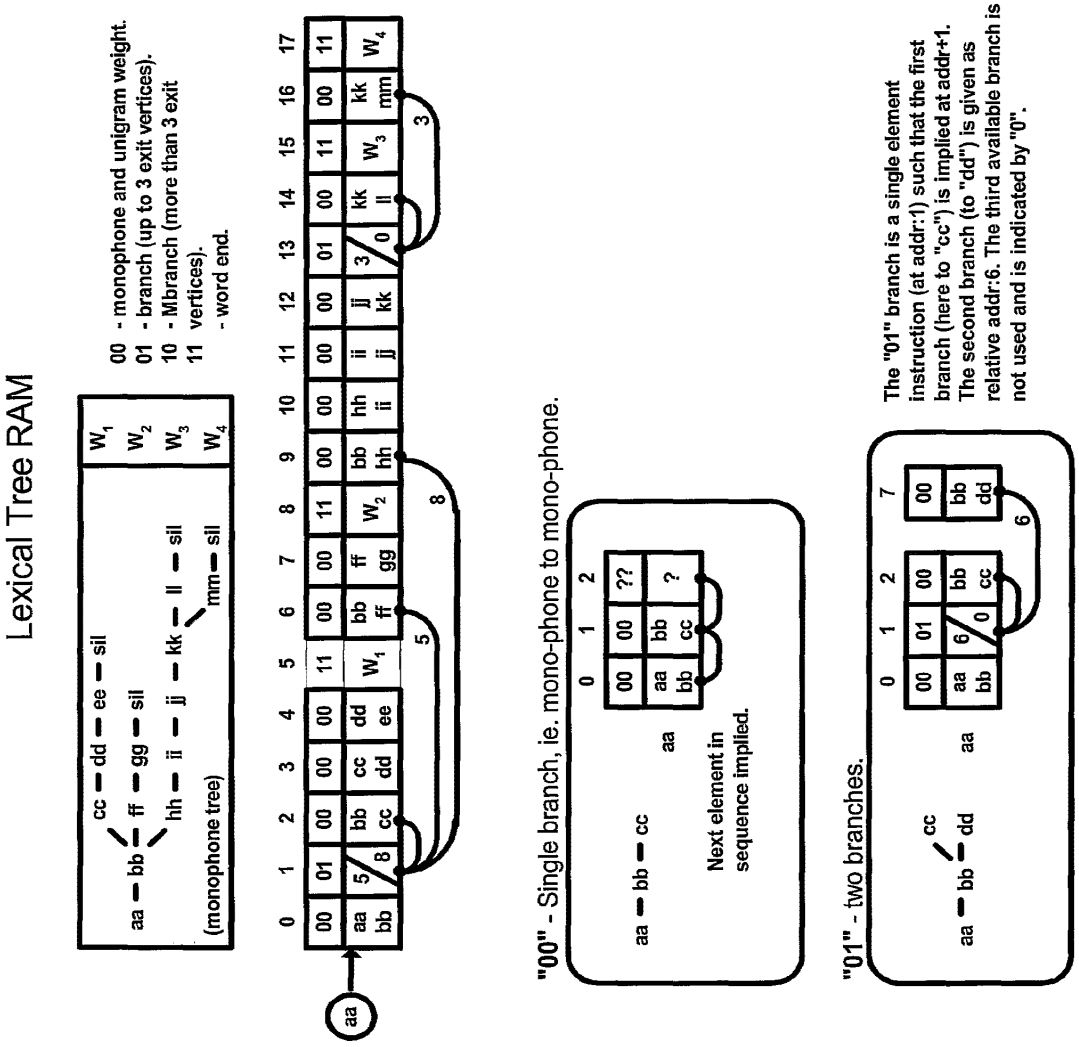
Figure 7



SUBSTITUTE SHEET (RULE 26)

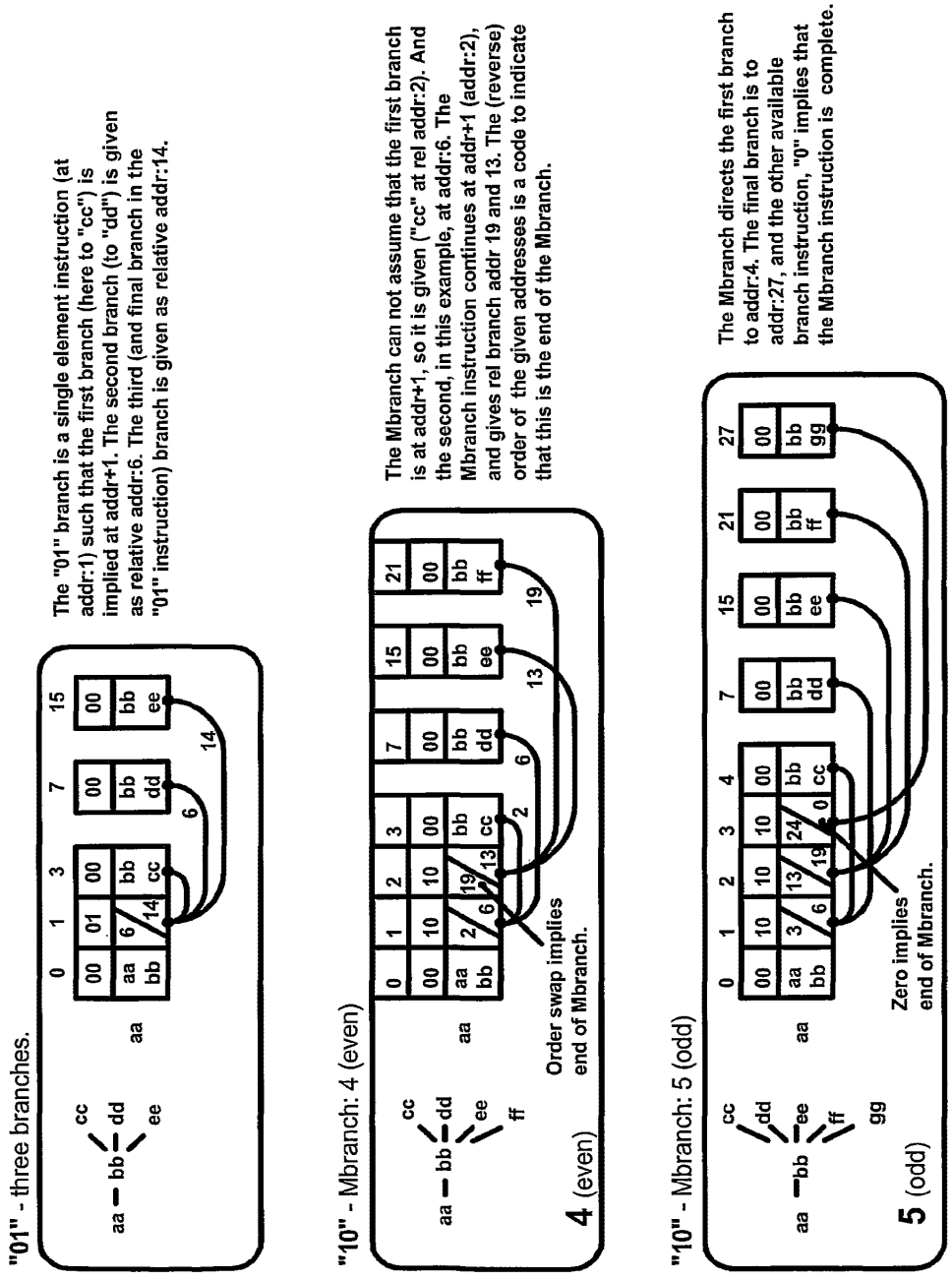
Amazon / Zentian Limited
Exhibit 1002
Page 289

Figure 8a



Amazon / Zentian Limited
Exhibit 1002
Page 290

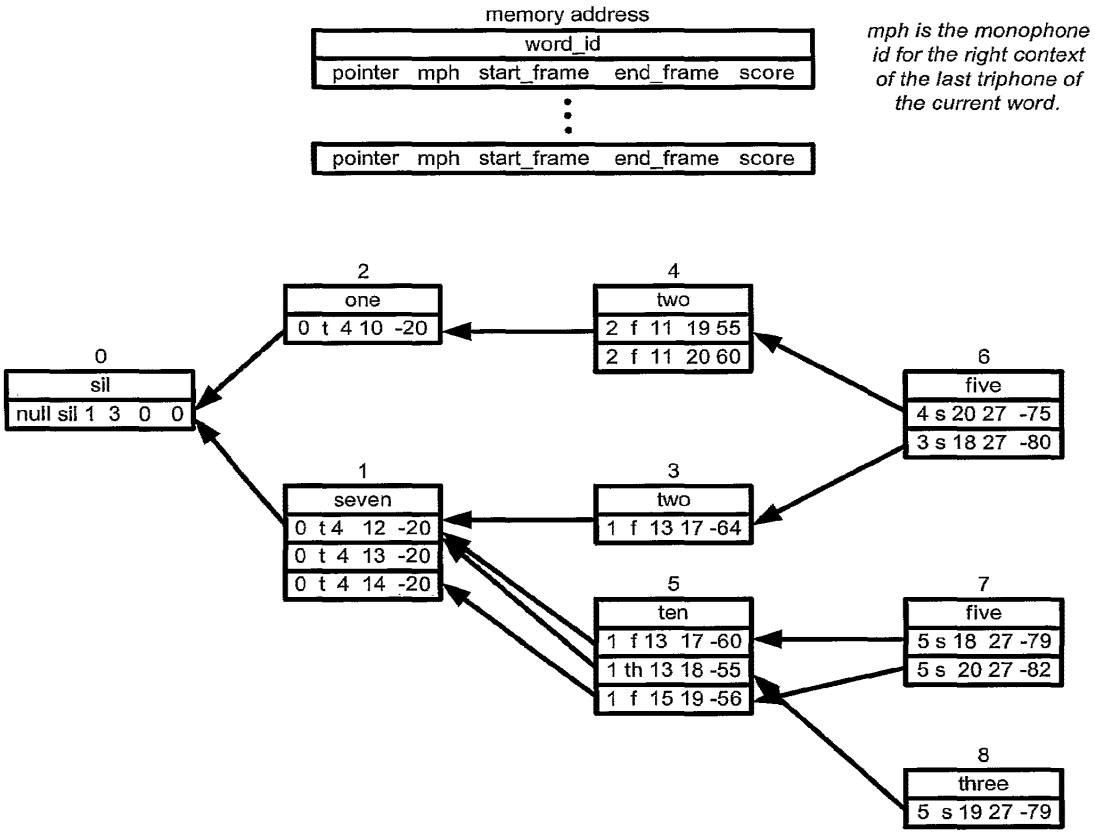
Figure 8b



Amazon / Zentian Limited
Exhibit 1002
Page 291

Figure 9

data structure for word link record to enable word n-best decoding



Notes

---

1. each record represents the passage of a single token through a word with a particular right monophone context.
2. each word's acoustic model with a particular right context monophone can only generate one token per frame. Thus frame number can be used to distinguish trace back paths with the same end_frame numbers but different right context monophones
3. the end_frame doesn't need to be passed to the PINEs because it is simply the current frame number
4. scores may be absolute or relative
5. the address is referred to as the wordlat_node_id elsewhere

Traceback

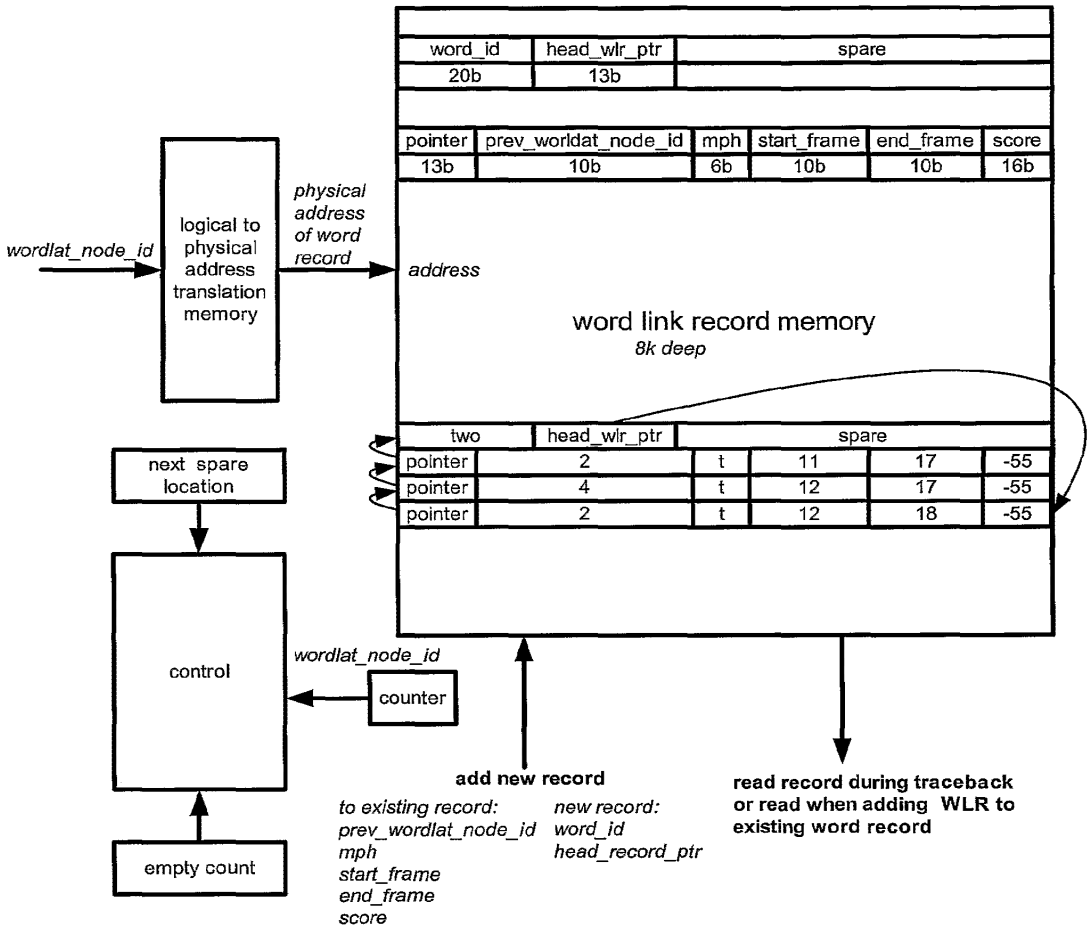
---

for the n-best tokens

1. using the associated wordlat_node_ids look up each word link record
2. look at associated records and use the end_frame number and monophone id to select associated pointer and score.
3. trace back recursively and select n-best paths

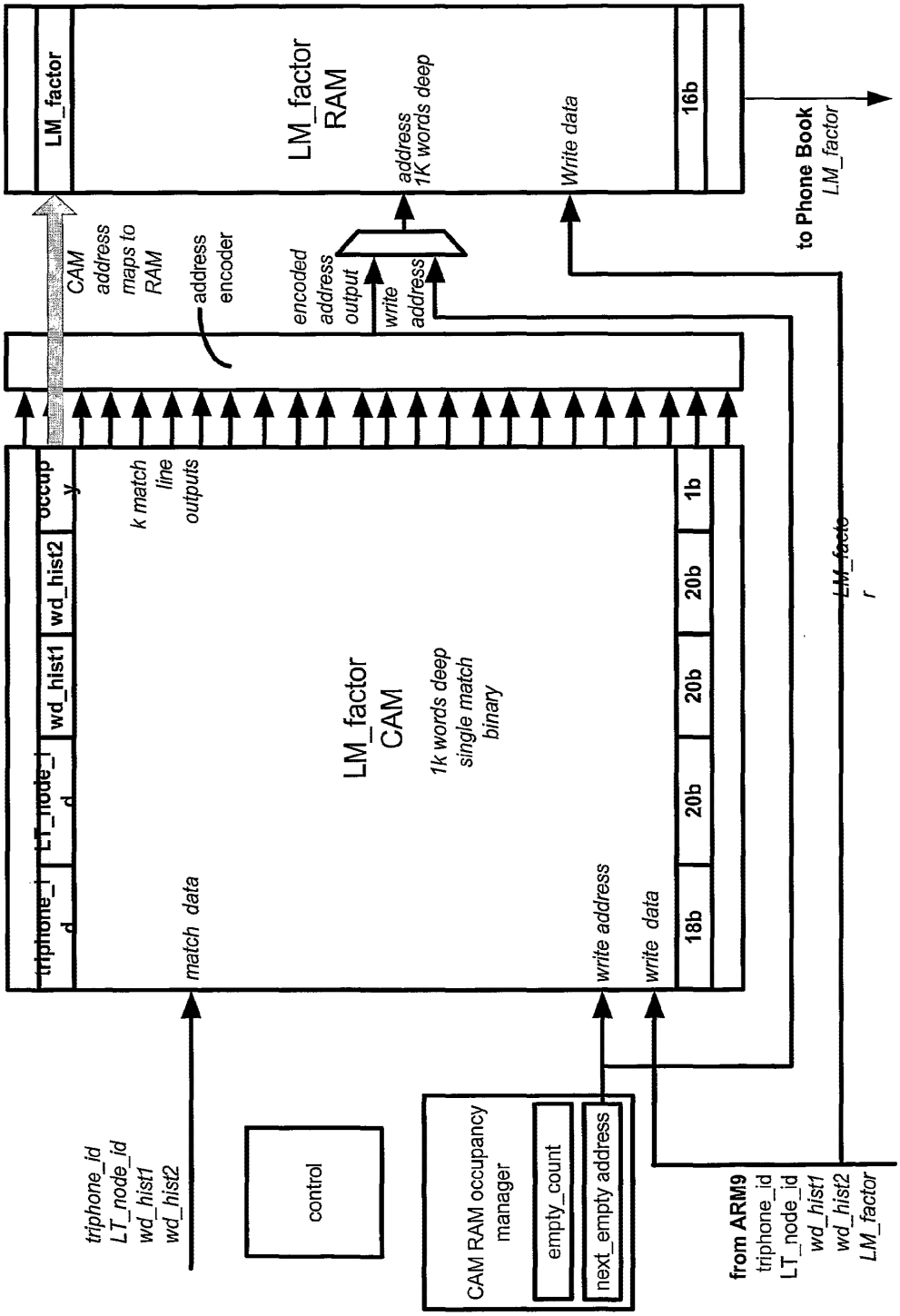
Amazon / Zentian Limited
Exhibit 1002
Page 292

Figure 10



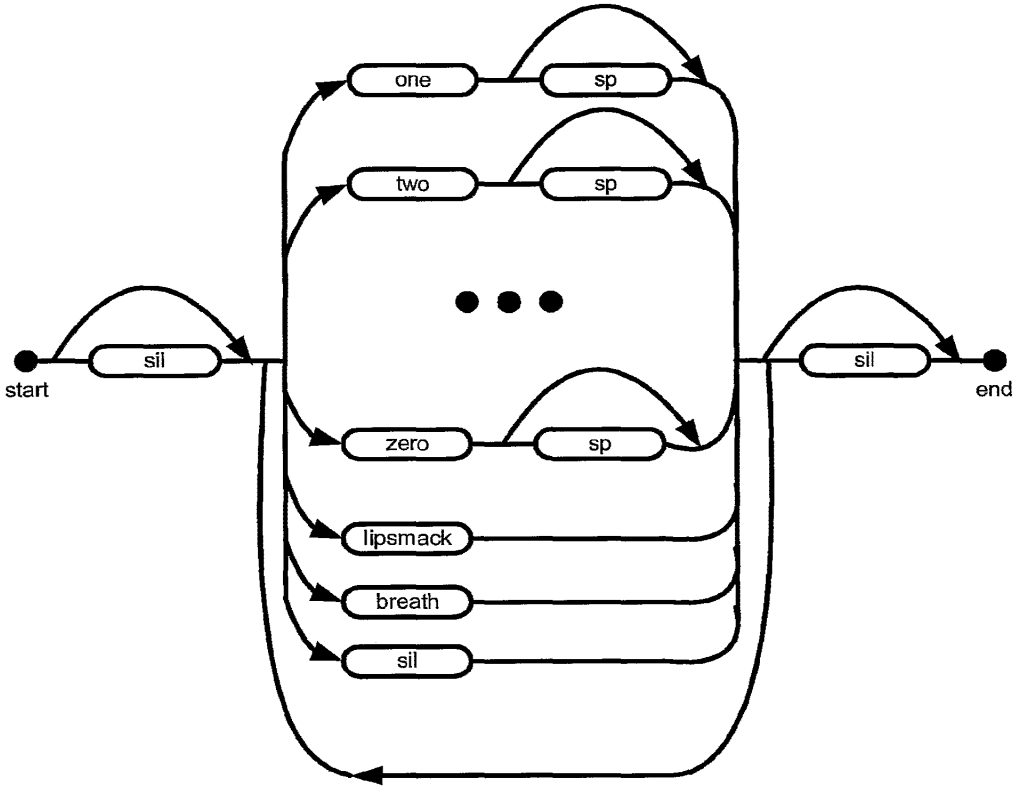
Amazon / Zentian Limited
Exhibit 1002
Page 293

Figure 11



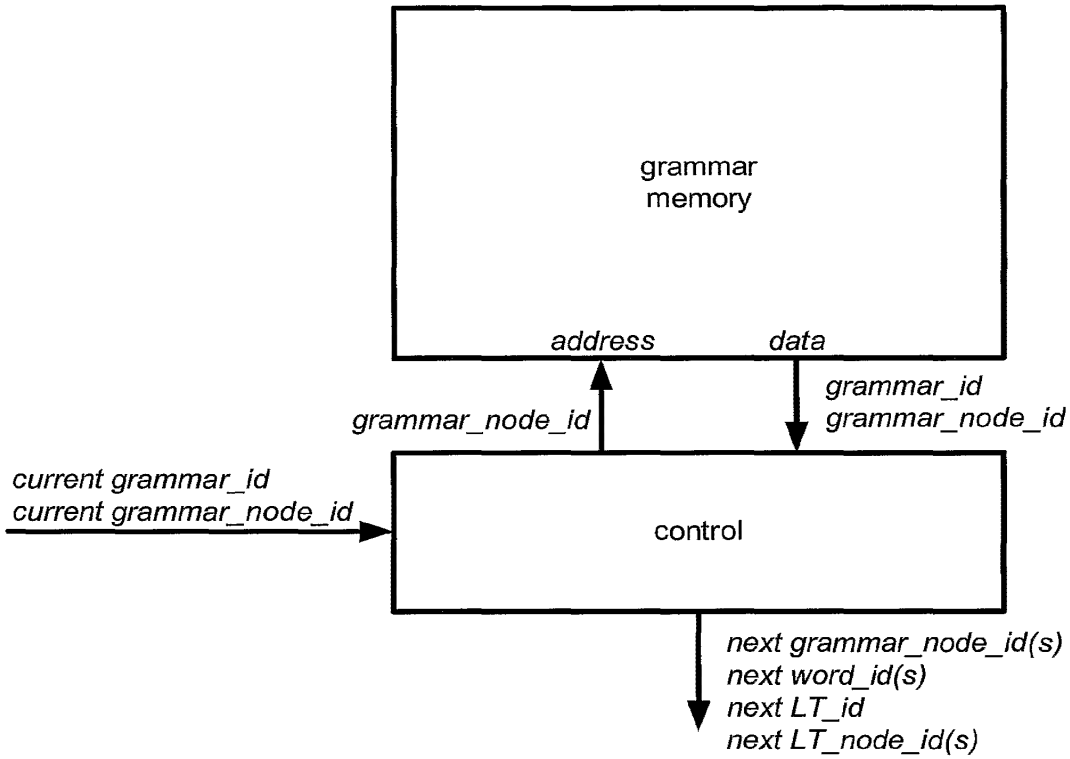
Amazon / Zentian Limited
Exhibit 1002
Page 294

Figure 12



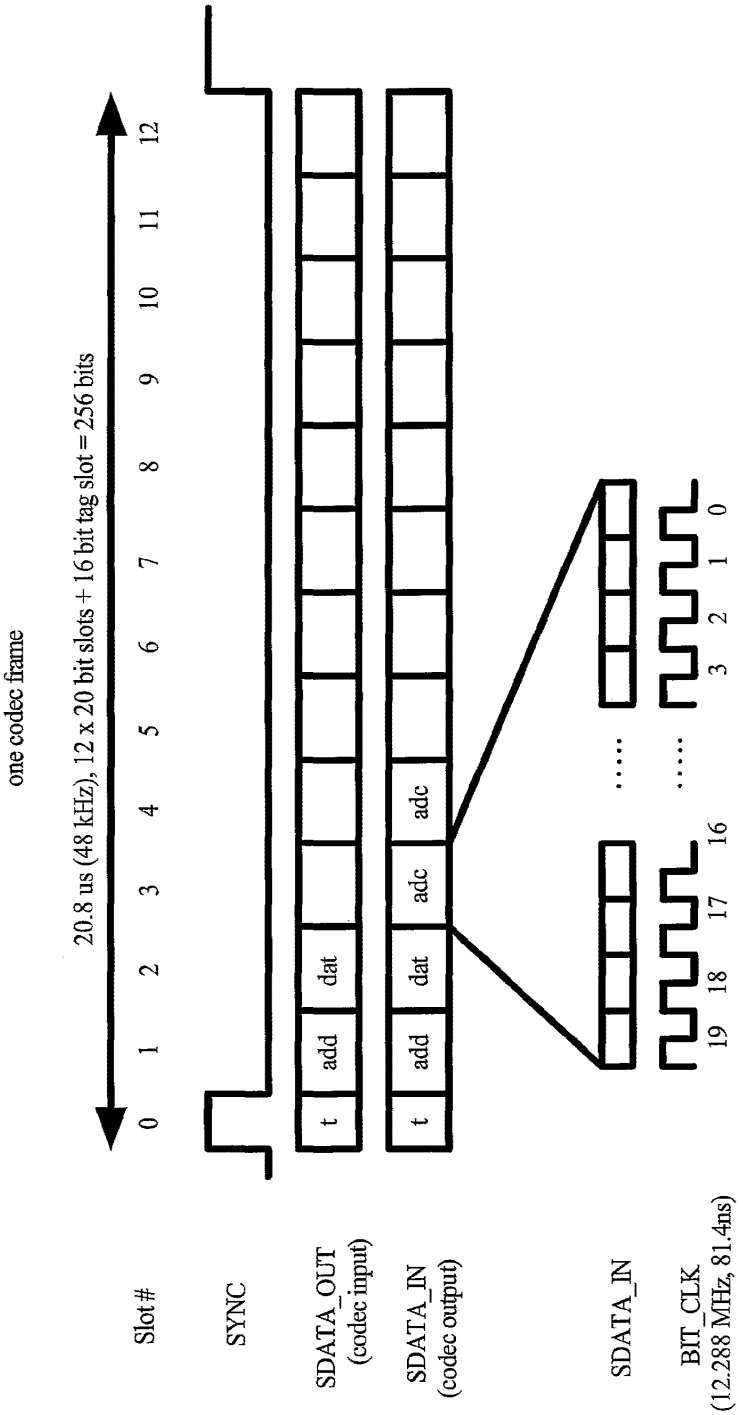
Amazon / Zentian Limited
Exhibit 1002
Page 295

Figure 13



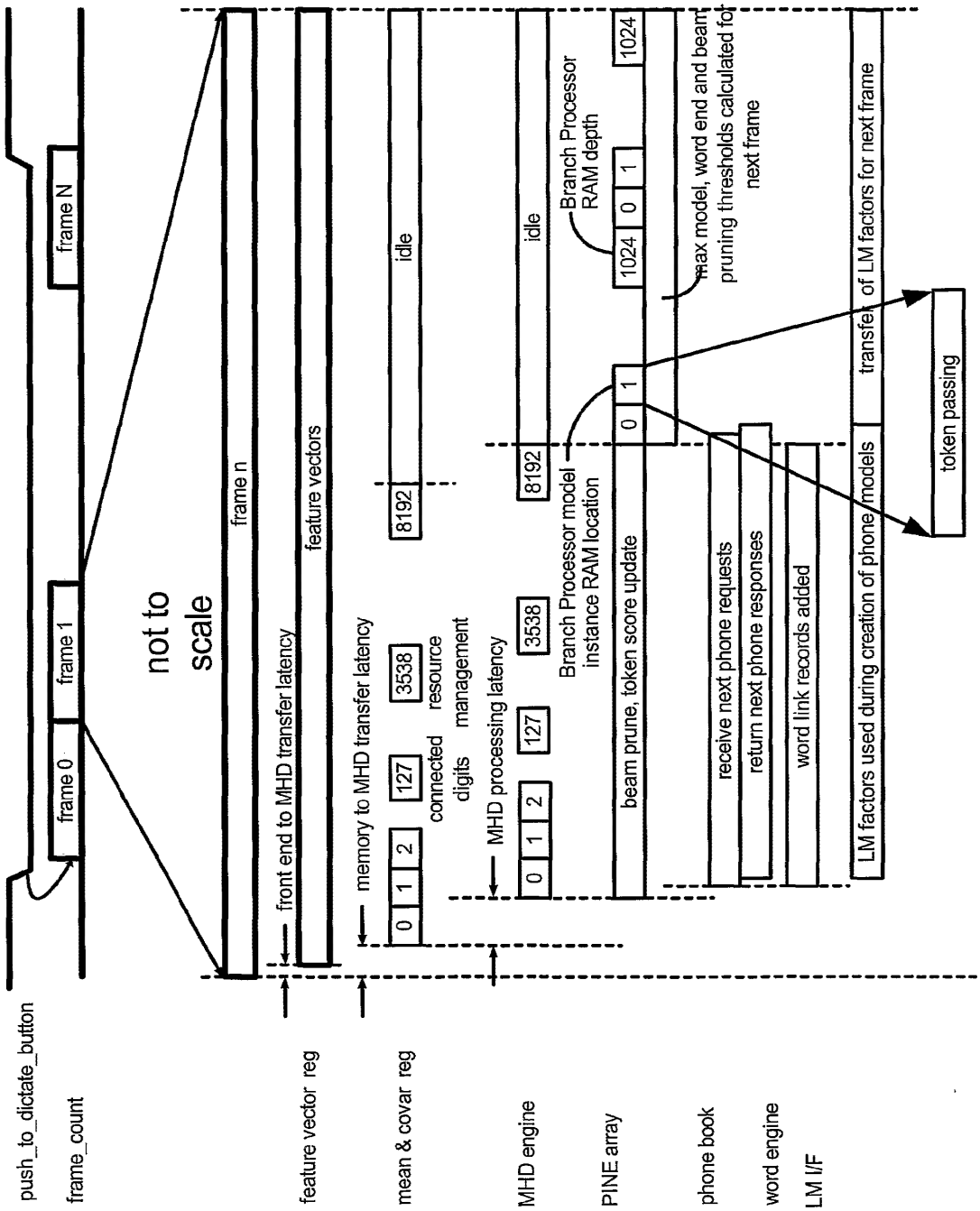
Amazon / Zentian Limited
Exhibit 1002
Page 296

Figure 14



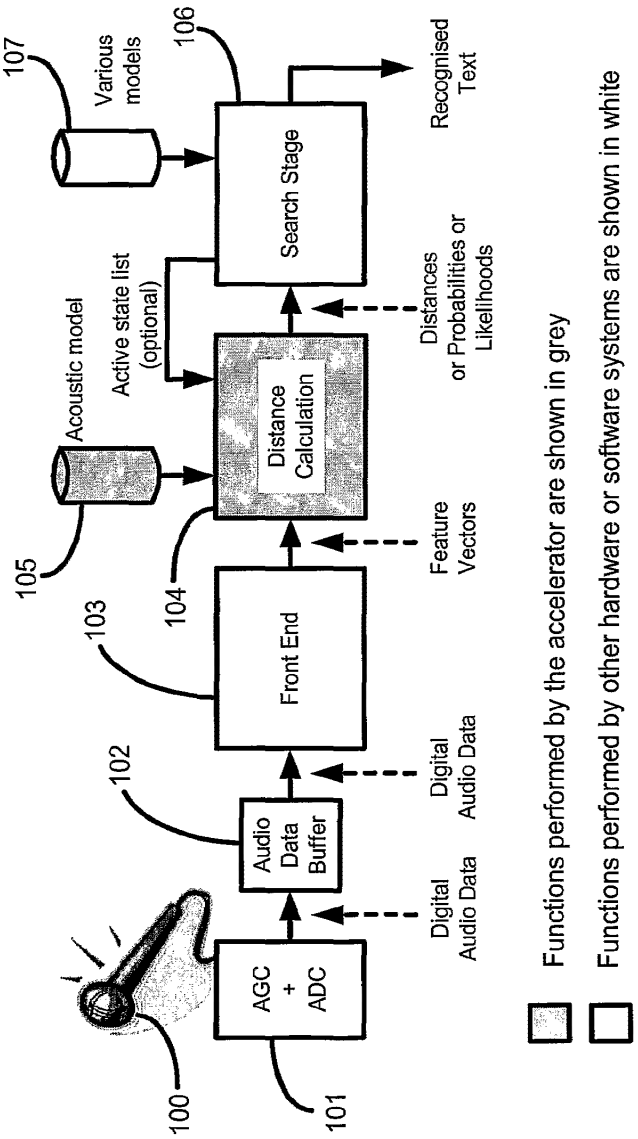
Amazon / Zentian Limited
Exhibit 1002
Page 297

Figure 15



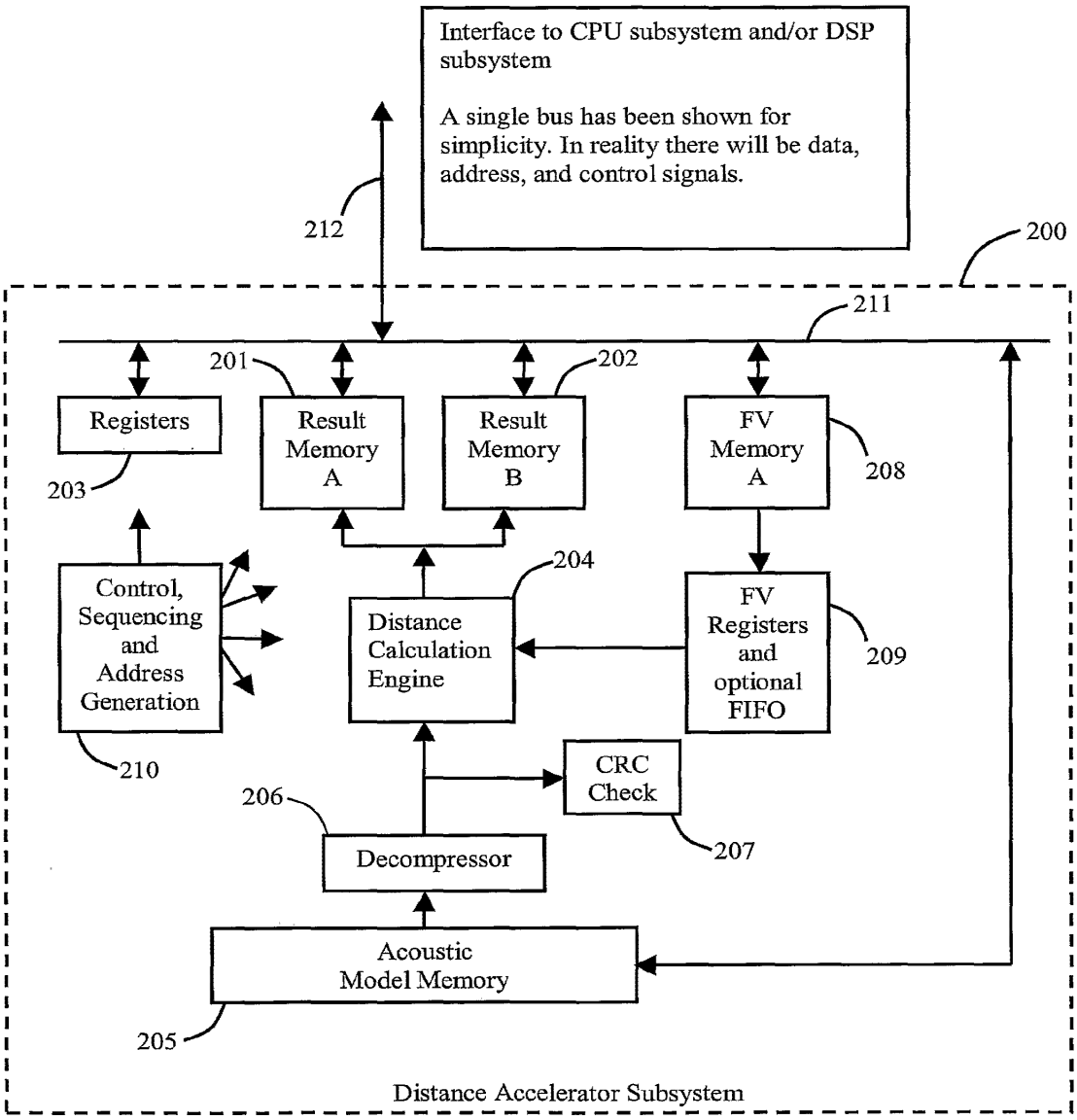
Amazon / Zentian Limited
Exhibit 1002
Page 298

Figure 16



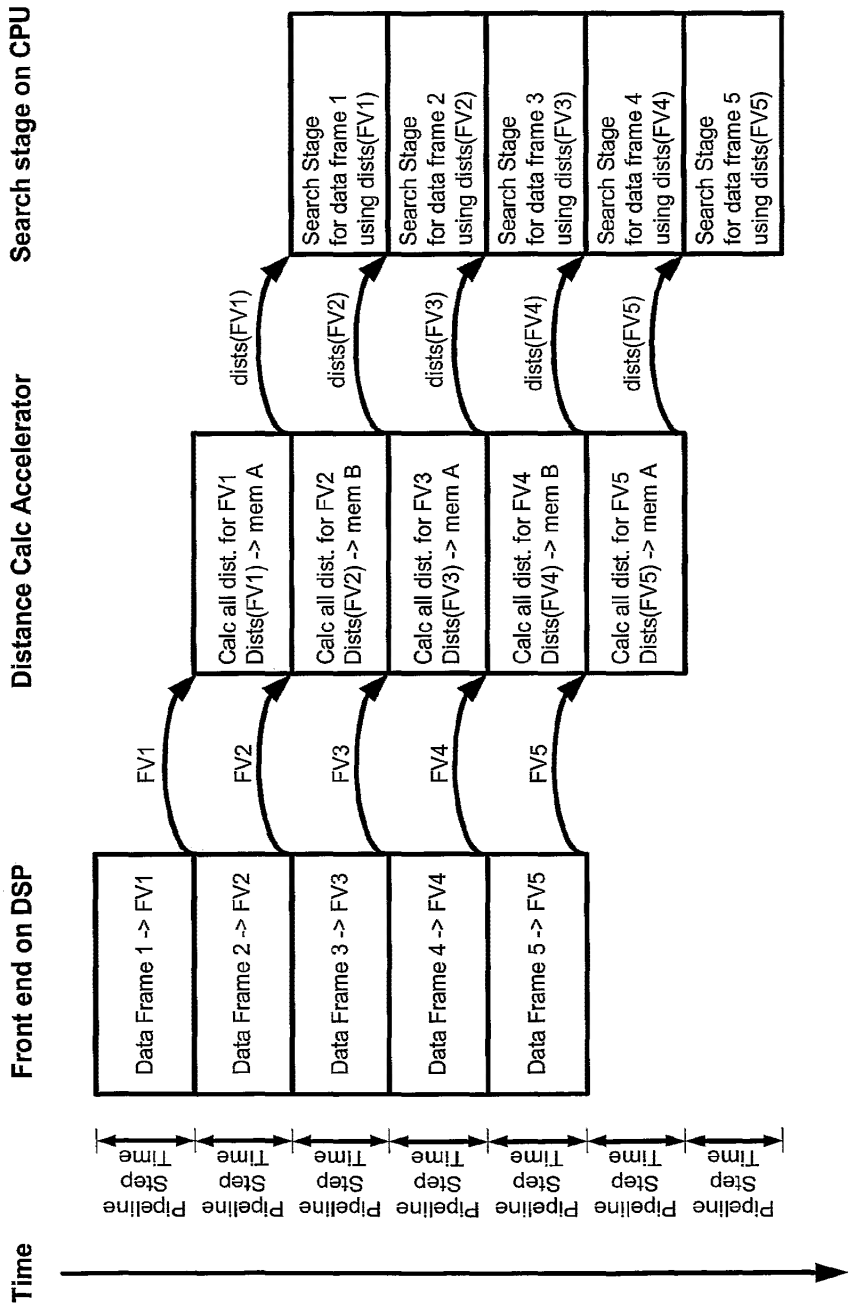
Amazon / Zentian Limited
Exhibit 1002
Page 299

FIGURE 17



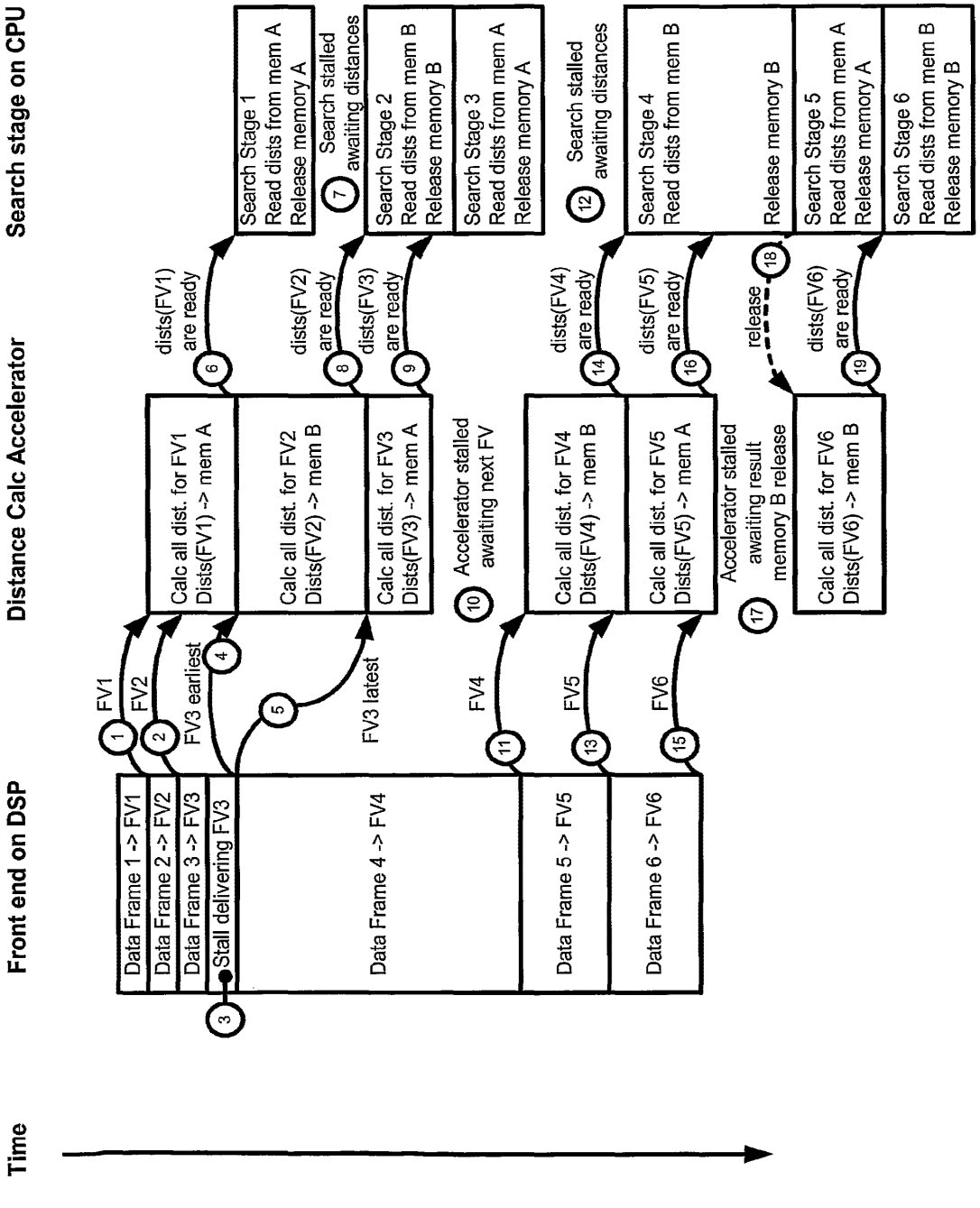
Amazon / Zentian Limited
Exhibit 1002
Page 300

Figure 18



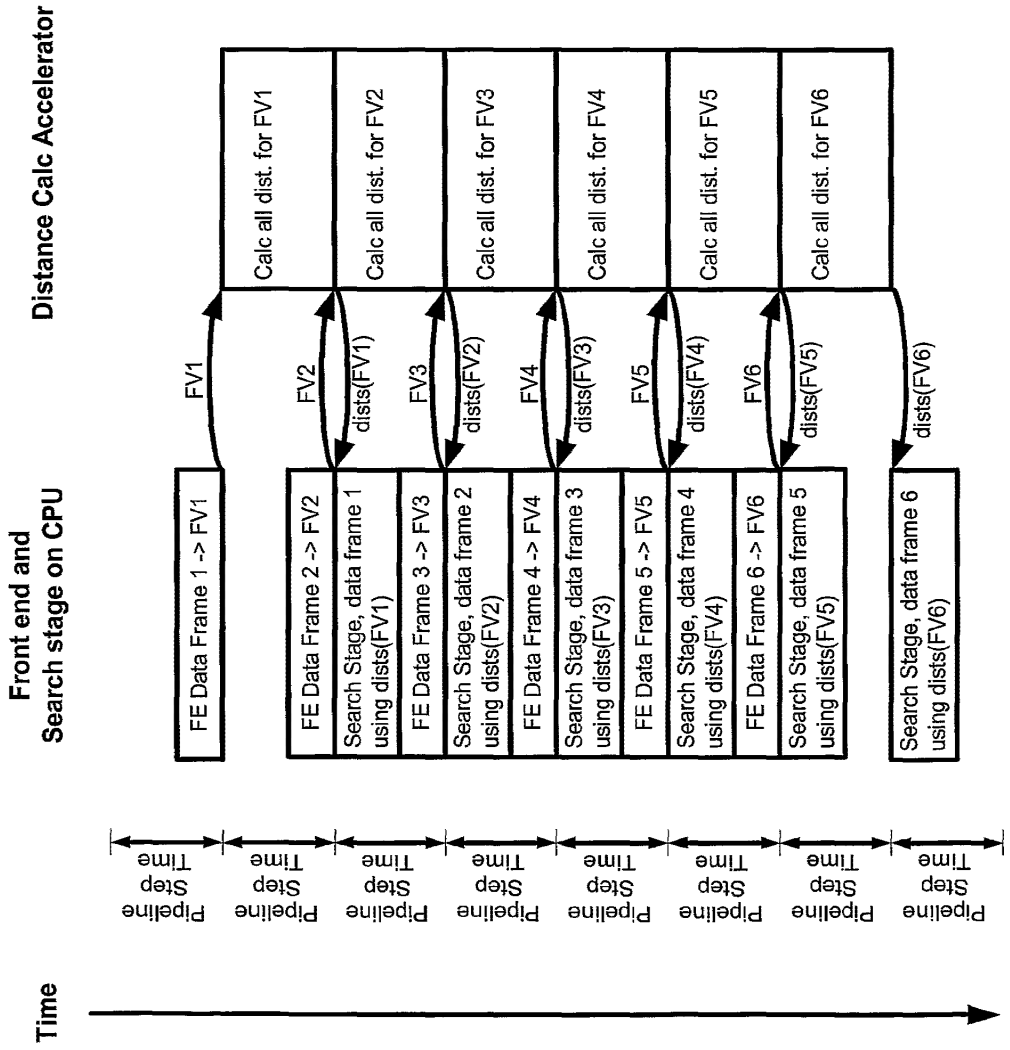
Amazon / Zentian Limited
Exhibit 1002
Page 301

Figure 19



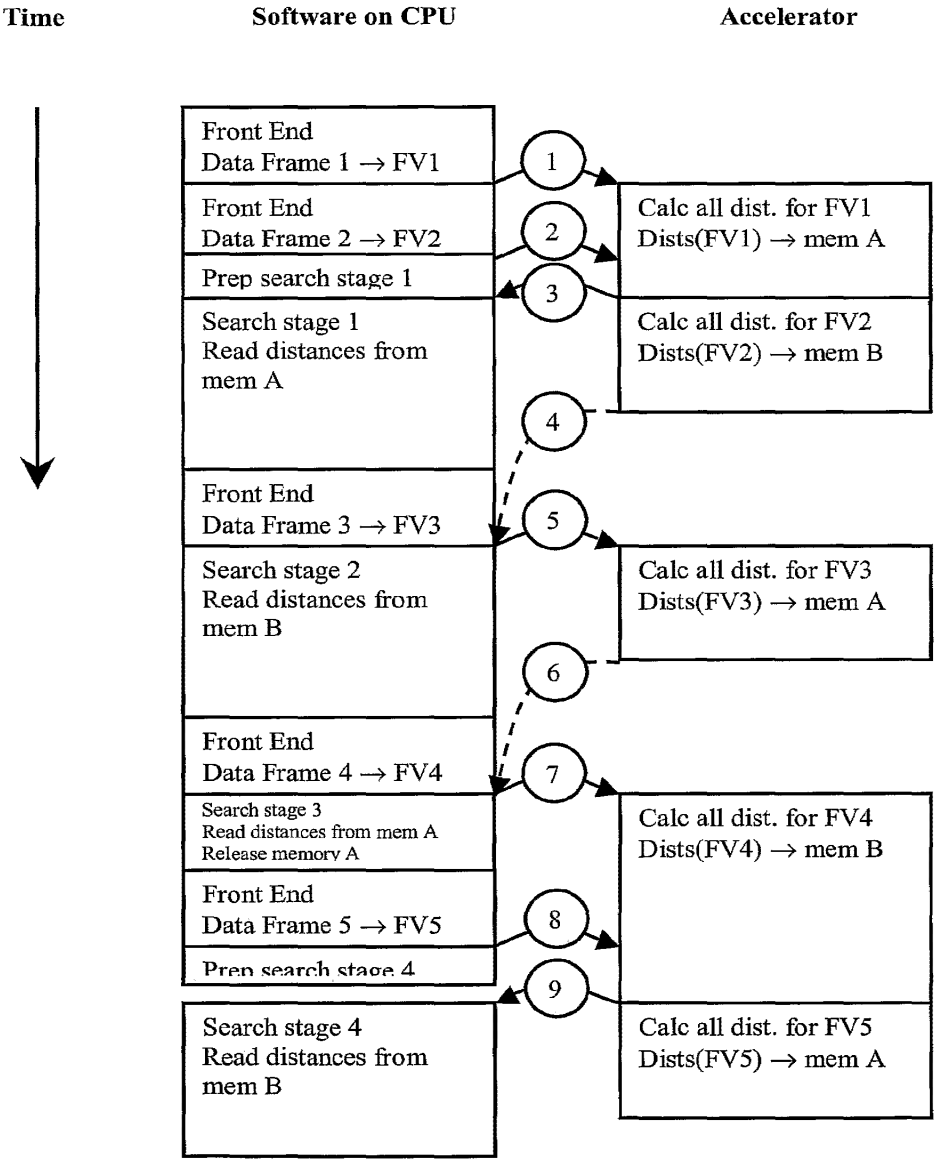
Amazon / Zentian Limited
Exhibit 1002
Page 302

Figure 20



Amazon / Zentian Limited
Exhibit 1002
Page 303

FIGURE 21



Amazon / Zentian Limited
Exhibit 1002
Page 304

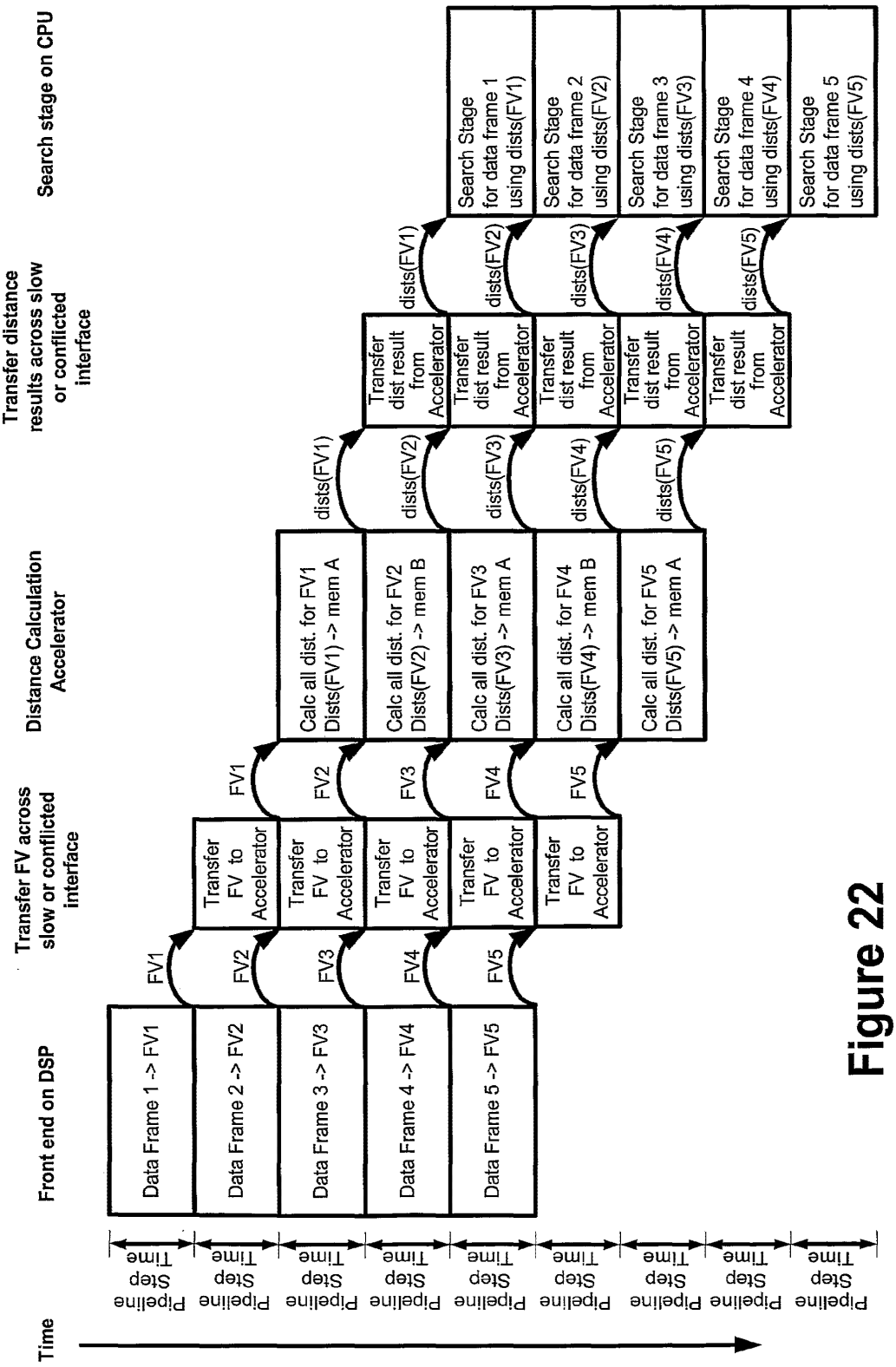
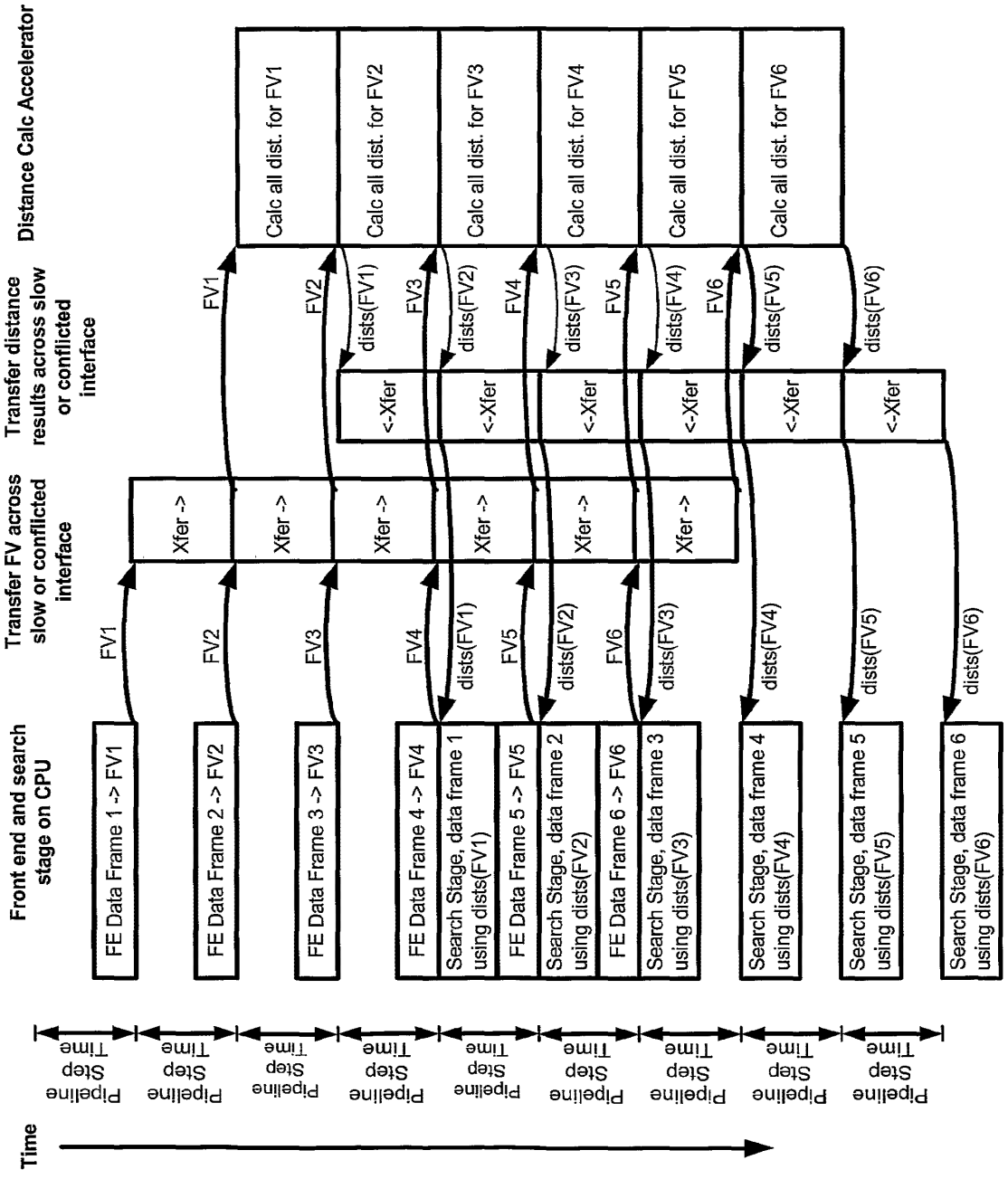


Figure 22

Amazon / Zentian Limited
Exhibit 1002
Page 305

Figure 23



SUBSTITUTE SHEET (RULE 26)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 March 2006 (23.03.2006)

PCT

(10) International Publication Number
WO 2006/030214 A3

(51) International Patent Classification:
G10L 15/28 (2006.01) *G10L 15/10* (2006.01)

(21) International Application Number:
PCT/GB2005/003554

(22) International Filing Date:
14 September 2005 (14.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0420464.0 14 September 2004 (14.09.2004) GB

(71) Applicant *(for all designated States except US)*: **ZENTIAN LIMITED** [GB/GB]; St. John's Innovation Centre, Cowley Road, Cambridge, Cambridgeshire CB4 0WS (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **CATCHPOLE, Mark** [—/GB]; Ash House, Breckenwood Road, Fulbourn, Cambridge, Cambridgeshire CB1 5DQ (GB). **DOWSETT, Damian, Harris**; * (**). **REYNOLDS, Timothy**; * (**). **LARRY, Guy** [—/GB]; 1 Cutter Ferry Lane, Cambridge, Cambridgeshire CB4 1JR (GB).

(74) Agent: **COLLINS, John, David**; Marks & Clerk, 90 Long Acre, London WC2E 9RA (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
24 August 2006

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2006/030214 A3

(54) Title: A SPEECH RECOGNITION CIRCUIT AND METHOD

(57) Abstract: A speech recognition circuit comprising a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/GB05/003554

International filing date: 14 September 2005 (14.09.2005)

Document type: Certified copy of priority document

Document details: Country/Office: GB
Number: 0420464.0
Filing date: 14 September 2004 (14.09.2004)

Date of receipt at the International Bureau: 11 November 2005 (11.11.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)



World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

GB2005/003554

PCT/GB 2005 / 0 0 3 5 5 4

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed William Morell

Dated 1 November 2005

Amazon / Zentian Limited
Exhibit 1002
Page 309

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

15SEP04 E925982-11 D03312
P01/7700 0.00-0420464.0 ACCOUNT CH

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

THE PATENT OFFICE
SW
14 SEP 2004
LONDON

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference GBP90312

2. Patent application number
(The Patent Office will fill in this part)

0420464.0

14 SEP 2004

3. Full name, address and postcode of the or of each applicant (underline all surnames)

Zentian Limited,
St. John's Innovation Centre
Cowley Road
Cambridge
Cambridgeshire CB4 0WS
United Kingdom

8318100 2

Patents ADP number (if you know it)

If the applicant is a corporate body, give the country/state of its incorporation

United Kingdom

4. Title of the invention A SPEECH RECOGNITION CIRCUIT AND METHOD

5. Name of your agent (if you have one)
"Address for service" in the United Kingdom to which all correspondence should be sent
(including the postcode)

Marks & Clerk
57 - 60 Lincoln's Inn Fields
London WC2A 3LS

Patents ADP number (if you know it)

18001

6. Priority: Complete this section if you are declaring priority from one or more earlier patent applications, filed in the last 12 months

Country	Priority application No (if you know it)	Date of filing (day / month / year)

7. Divisionals, etc: Complete this section only if this application is a divisional, application or resulted from an entitlement dispute

Number of earlier application	Date of filing (day / month / year)

8. Is a Patents Form 7/77 (Statement of inventorship and of right to grant of a patent) required in support of this request?
(Answer 'Yes' if:
a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Yes

Patents Form 1/77

Accompanying documents: A patent application must include a description of the invention. Not counting duplicates, please enter the number of pages of each item accompanying this form:

Continuation sheets of this form	0
Description	29
Claim(s)	9
Abstract	1
Drawing(s)	15

+ISSN

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent (Patents Form 7/77)	
Request for preliminary examination and search (Patents Form 9/77)	
Request for substantive examination (Patents Form 10/77)	
Any other documents (please specify)	

11. I/We request the grant of a patent on the basis of this application.

Signature(s) MHC Date: 14 September 2004

12. Name and daytime telephone number of person to contact in the United Kingdom

Patent Electronics Formalities
020 7400 3000

A Speech Recognition Circuit and Method

The present invention relates to speech recognition circuits and methods. These circuits and methods have wide applicability, particularly for devices such as mobile electronic devices.

There is growing consumer demand for embedded speech recognition in mobile electronic devices, such as mobile phones, dictation machines, PDAs (personal digital assistants), mobile games consoles, etc. For example, email and text message dictation, note taking, form filling, and command and control applications are all potential applications of embedded speech recognition.

However, when a medium to large vocabulary is required, effective speech recognition for mobile electronic devices has many difficulties not associated with speech recognition systems in hardware systems such as personal computers or workstations. Firstly, the available power in mobile systems is often supplied by battery, and may be severely limited. Secondly, mobile electronic devices are frequently designed to be as small as practically possible. Thus, the memory and resources of such mobile embedded systems tends to be very limited, due to power and space restrictions. The cost of providing extra memory and resources in a mobile electronic device is typically much higher than that for a less portable device without this space restriction. Thirdly, the mobile hardware may be typically used in a noisier environment than that of a fixed computer, e.g. on public transport, near a busy road, etc. Thus, a more complex speech model and more intensive computation may be required to obtain adequate speech recognition results.

These restrictions have made it difficult to implement effective speech recognition in mobile devices, other than with very limited vocabularies.

Some prior art schemes have been proposed to increase the efficiency of speech recognition systems, in an attempt to make them more suitable for use in mobile technology.

In an article entitled "A low-power accelerator for the SPHINX 3 speech recognition system", in University of Utah, International conference on Compilers, Architectures and Synthesis for Embedded Systems, Nov 2003, Davis et al have proposed the idea of using a special purpose co-processor for up-front calculation of the computationally expensive Gaussian output probabilities of audio frames corresponding to particular states in the acoustic model.

In an article entitled "Hardware Speech Recognition in Low Cost, Low Power Devices", University of California, Berkeley, CS252 Class Project, Spring 2003, Sukun Kim et al describe using special purpose processing elements for each of the nodes in the network to be searched. This effectively implies having a single processing element for each phone in the network. An alternative suggested by Sukun Kim et al is to provide a processor for each state in the network.

In an article entitled "Dynamic Programming Search for Continuous Speech Recognition" in IEEE Signal Processing Magazine, Sept 1999, Ney et al discuss language model lookahead. Language model lookahead involves computation of a language model factor for each node (i.e. phone) in the lexical tree. This technique is also known as smearing. Each phone instance in the search network can be given a language model factor when it is used in the lexical tree search. Ney et al show that for an example bigram language model, the average number of states per 10 ms frame can be reduced from around 168,000 states with no language model lookahead to around 8,000 states when language model lookahead is used. They also show that bigram language model lookahead requires about a quarter of the states compared with unigram language model lookahead.

Although these prior art documents provide improvements to speech recognition in embedded mobile technology, further improvement is still needed to provide a larger vocabulary and better accuracy.

One aspect of the present invention provides a speech recognition circuit including a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers. The lexical tree includes a model of words. The speech recognition circuit also has a memory structure for receiving and storing state identifiers identified by a node identifier identifying nodes or groups of adjacent nodes, the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores. An accumulator is provided for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores. A selector circuit is used for selecting at least one node or group of nodes of the lexical tree according to said scores.

One suitable type of hardware for the memory structure includes a content addressable memory (CAM). A CAM is a memory unit which stores a series of data items using a series of addresses. However, the memory is accessed by specifying a data item, such that the CAM returns the corresponding address. This contrasts with a random access memory (RAM) in which the memory is accessed by specifying an address, such that the RAM returns the corresponding data item.

However, the memory structure is not limited to including a CAM. Other types of hardware are also possible, to provide this functionality. For example, a single chip which operates in the same way as a CAM and RAM may be used instead.

Embodiments of the present invention provide a solution to the problem of how to map a lexical tree search to a CAM system architecture. The realisation by the present inventors that certain speech recognition data structures can be mapped into the CAMs allows a lexical tree search to be performed using a CAM system architecture.

Further embodiments of the invention include a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

The node identifier may comprise a direct reference to the lexical tree. However, in some embodiments, the node identifier for at least some of the states includes a pointer to a node identifier for another state. For example, a state corresponding to the furthest part of the search path in the lexical tree may be referenced by a node identifier which directly links to a particular node or group of nodes in the lexical tree. In a lexical tree comprising phones, using a state model of triphones, the node identifier may indicate the position of a triphone in the lexical tree.

However, in this example, for states occuring further back in the search path, instead of supplying a node identifier linking directly to the lexical tree, instead a pointer to a node identifier of another state may be supplied. E.g. a triphone instance may have a pointer to another triphone instance, which has a pointer to another triphone instance, which has a pointer to a node or group of nodes in the lexical tree. Chains of reference may be set up in this way, where only the last state in the chain has a direct pointer to the lexical tree.

There may not be a one-to-one correspondence between the nodes of the lexical tree and the node identifiers. This will occur for a branched lexical tree, where the nodes represent monophones, but the acoustic model states represent triphones, i.e. groups of three adjacent monophones. Then, paths of three monophones will have unique identifiers to be stored in the memory structure, rather than single monophones having unique identifiers.

Phone instance numbers may be generated, and used to uniquely label each phone instance. They can be generated sequentially, using a counter. The phone instance numbers may be used as pointers between phone instances to assist in node identification. It is thus not essential to provide a direct node identifier for each phone instance to directly indicate a location in the lexical tree. The dynamic network of phone

instances provided in the memory structure may thus include both direct and relative references to the lexical tree.

The memory structure may be divided into one part which stores phone instance identifiers and direct references to the lexical tree, and a second part which stores phone instance identifiers and corresponding states. This can speed up the processing, by only storing the phone instances which are furthest on in the lexical tree in the first part of the memory structure.

The memory structure may also be divided into separately accessable units, to reduce the amount to data in each unit, thereby decreasing the chance of finding the same two states identifiers in different phone instances in any single memory unit, and increasing the chance of some state identifiers being completely absent from any single memory unit. This makes it easier to deal with the situation when the same two state identifiers are found, because a spare time slot is available for processing when a state identifier is not present.

A speech recognition apparatus according to the invention may be embedded in or included in a mobile electronic device such as a mobile telephone.

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of the system architecture for a speech recognition apparatus according to an embodiment of the invention;

Figure 2 is a block diagram showing the main data structures used in the speech recognition apparatus of figure 1;

Figure 3 shows a block diagram of the Single Phone Instance Network Engine (PINE) of figure 1;

Figure 4 is a block diagram showing the Wave Front Manager architecture;

Figure 5 shows the use of link nodes to permit path merging;

Figure 6 shows a block diagram of a Path Merge Unit in the Phone Level;

Figure 7 shows the Phone Book architecture;

Figure 8 shows Lexical tree storage;

Figure 9 shows the data structure for the word link record;

Figure 10 shows a block diagram of Word Engine;

Figure 11 shows a Language Model interface architecture;

Figure 12 shows an example of a simple Loop Grammar;

Figure 13 shows a block diagram of grammar engine;

Figure 14 shows a single frame of the Audio Codec link protocol; and

Figure 15 illustrates the system timing which may be used in the embodiment of figure 1.

Figure 1 shows a speech recognition system architecture for an embodiment of the invention. The speech recognition system is divided into several levels of operation: a state level, a phone level, a word level, a grammar level, and a system level. Each level is shown in figure 1 as a grey shaded box, interconnected to other levels of the system.

The information passed between different levels or subcircuits of figure 1 is indicated on the figure 1 using the following identifiers.

Identifier	Definition

grammar_id	Identifies which grammar model is currently being used
grammar_node_id	Identifies a node in the grammar model
word_id	Uniquely identifies a word
wordlat_node_id	Identifies a node in the word lattice
LT_id	Lexical tree identifier. Identifies the lexical tree to be used
LT_branch_id	Identifies a unique branch in a lexical tree. A branch is defined as all of the monophones subsequent to and including the first monophone in a tree.
LT_node_id	Identifies a node in a lexical tree
monophone_id	Identifies a monophone
transvec_id	Identifies which transition vector to use
state_id	Identifies the state
phone_no	Unique phone instance identifier
prev_phone_no	Phone number of previous phone in network
triphone_id	Triphone identifier
LML_factor	Language model lookahead factor. The smeared language model score that must be added

The system level includes a general purpose processor and cache, a system controller, a DMA (direct memory access) controller, clocks, PLL (phased locked loop), GPIO (general purpose I/O interface), timer watch dog, memory interfaces to interface to main memory and non volatile storage, serial, host, debug and display interfaces. The system level also includes a model adaptation engine, a language model factor engine, a speech recognition system controller, a language model interface, and a recognition output interface.

The processor organises the transfer of the various models from the non volatile storage to the various model memories. It handles the language model factor requests. Language model factors are used to by the state level to apply the language model score. The language model (LM) interface provides the mechanism for obtaining LM factors as and when they are needed. The processor keeps a copy of the wave front in memory which is updated simultaneously with that in the wave front manager. Using its copy of the lexical tree, the processor can determine which language model scores may be needed in the coming frames. Once calculated, the language model factors can be kept in memory for use in subsequent frames. The system level is also used for forming speech model adaptation, and acting as an overflow for the various buffers used by the architecture.

The system level includes the audio codec interface and the audio front end interface, connected to the processor via a APB (advanced peripheral bus) to AHB (advanced high performance bus) bridge.

The grammar level includes a grammar engine and grammar model(s). The grammar level sets out the rules of the particular recognition task at the highest level. During system initialisation a number of grammar models might be loaded, for example a command grammar and a dictation grammar. Command mode grammar is a restricted grammar which includes only a very limited lexical tree with command specific words. Some branches of this lexical tree may include multiple words together, where a command has such a structure, e.g. "open file menu". Typically, only a single instance of the lexical tree will be needed. Command mode is used because it can obtain a higher accuracy, due to the lexical tree constraint. In dictation mode grammar, a full lexical tree structure is used.

The grammar may be selected by a user during the recognition process. For example, the command grammar may be selected when the user presses a command button. The architecture supports fast swapping of grammar models. For example, a user may be in the middle of dictating a message when a "punctuation button" is pressed and the phrase "question mark" uttered. At this point, a punctuation grammar could be loaded, in order to increase recognition accuracy.

The word level comprises a word engine, which includes a word lattice and a word graph generator. The word level has two main functions, of keeping a record of the search path at the word level using a word lattice, and interfacing to the language model via a cache. The word lattice is a record of all of the words that have been searched up to the current frame. As new words are searched, they are appended into the lattice at the appropriate place. After the final frame in the current sentence, the n-best word lattice node identifiers from the n-best scoring tokens are passed back to the word lattice. This enables the n-best sentences to be traced back through the lattice. The word lattice can also be post processed to produce a word graph which is essentially a compact version of the word lattice.

At the word level, a record is kept of all tokens that have passed through a word end. This results in a lattice of words. When the final feature vector has been processed, the

best scoring token is identified and used to trace back the search path back to the start of the search.

The phone level includes a "phone book" for generating new triphone instances, a wave front manager for sending requests to the phone book to generate new phone instances, a phone request arbiter, and a path merge unit.

The phone book includes a lexical tree, which comprises a vocabulary of words, sounds and/or phrases organised according to the phones or monophones making up the sound. A lexical tree is commonly used in speech recognition rather than a linear lexicon, because it greatly reduces the amount of storage space required to store a word vocabulary, by avoiding duplication of common word beginnings. It is also easier to search when the word beginning is known.

Although it is possible to search a lexical tree based on monophones, it is well known that better results can be achieved in speech recognition by basing the lexical tree search on groups of more than one monophone, since the sound of one syllable commonly affects the pronunciation of the next syllable in the same word, or the first syllable in the next word. In this embodiment, groups of three monophones are used, and these are known as triphones. It is alternatively possible to use a different number of monophones for each group. The phone generator in the "phone book" uses data from the lexical tree to generate triphones.

In one embodiment of the invention, approximately 45 different cued mono-phones are considered. This gives of order 60,000 tri-phones. Each triphone in this embodiment is modelled by a Hidden Markov Model (HMM) with three states. There is no correspondence between individual states and individual phones of the triphone. Alternatively, it is possible to model each triphone by a single state, or by any other number of states.

The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree. The search space can be envisaged as a wave moving through an acousto-

linguistic search space. In order that the wave can be dynamically created, a record of the wave front phone instances must be maintained. This is done in a wave front phone CAM/RAM structure, within the wave front manager. When a token occurs in the final state of a phone instance in the PINE array at the front of the wave, a new phone instance must be generated at the phone level and passed to the PINE array.

The phone instance includes a phone_no, which is a phone instance identifier to identify the phone uniquely in the current search. The phone instance also includes a word lattice node identifier (labelled "word_lat_node_id") to identify the phone's position in the word lattice, state identifiers (labelled "state_ids") for the HMM, storage to hold the tokens as they move through the HMM and LM_factor which is used to by the PINE to apply the language model score. To handle the connectivity between phone instances, a phone instance points to a previous phone instance using the phone_no of the previous phone. This allows tokens to be passed between phones. A "token" is a hypothesised search path, represented by a non-zero score at a particular state, and tokens are propagated through the HMM with each time frame.

The phone instance network can be visualised as a wave with new phone instances being added to the front of the wave, and empty phone instances being removed from the back of the wave. In some circumstances, the back of the wave may be in the same position as the wave front. The back of the wave moves on through the lexical tree as tokens are either moved on or pruned.

Tokens, each representing a particular hypothesis, hop through the states in the HMMs, each hop consuming a feature vector. Each token explores all possible next states which typically involves placing a token in the next state and keeping one in the current state. When there is more than one token in a state, the least likely token can be removed because both have the same word history. When a token reaches the last state of a phone instance at the from of the wave, the next phone instance(s) are requested from the phone level. The phone level uses the lexical tree to create the new phones. As well as looking up the state identification numbers for each HMM state, it also incorporates the correct language model factor.

The language model factor engine is used to smear the language model scores across the lexical tree. This operates in two ways depending on whether the factor is needed for a word start phone(s) or word internal phones. For word internal phones, LM factors can be computed as needed, provided that the LM scores are available in cache. This is because in general the storage requirements for the LM scores are less than the storage of the LM factors for all of the factors in the branch. However when a token is in a word end phone there may a significant number of word start phones e.g. up to $(\text{monophones})^2$ requiring a large amount of computation. To avoid this the LM engine will pre compute word start LM factors when it seems highly likely that a token will be passed to the word starts. These word start LM factors will be held in the LM factors cache. The phone level is responsible for generating the search space according to the word structure specified in the grammar model and the phone structure specified by the lexical tree.

The state level shown in figure 1 includes a phone instance network engine (PINE) array, a MHD (Mahalanobis distance) engine, an audio front end, a feature vector buffer for model adaptation, and compressed acoustic models. The state level is the lowest level, in which a network of phones is dynamically created.

The audio input for speech recognition is input to the front end unit of the state level. This audio input may be digital audio, or it may be analog audio which is converted to digital audio using an analog to digital converter. The audio input is divided into time frames, each time frame typically being of the order of 10ms.

For each audio input time frame, the audio signal is converted into a feature vector. This may be done by splitting the audio signal into spectral components. For example, in the present embodiment, the audio input signal is split into 12 frequency bands and one energy band, and a feature vector is constructed using these 13 components, plus their first and second derivatives, a total of 39 components.

The feature vector is passed to the MHD engine. This computes all possible state output lieklihoods for the current frame in a continuous stream as the state mean and covariance data is read from the acoustic model memory.

The likelihood of each state is determined by the distance between the feature vector and each state. To reduce the number of states, a technique known as "state tying" may be used. Multiple states are grouped together in 39D space, according to their proximity to one another. An average or central value from each group is used to represent the entire group. Thus, the effective number of states may be reduced from 40,000 to 8,000. The accuracy may be improved using a language model in addition to the word recognition lexical tree. A language model is based on the probability of sequential words. The above figure of 8,000 states gives best recognition results when used with a language model.

The MHD (Mahalanobis distance) is the distance between the feature vector and each state. The feature vector is used to calculate 8,000 MHD distances for each time frame, i.e. one distance for each of the 8,000 states. It is beneficial, although not essential, to perform the MHD calculations separately in a pre-processor, known as a MHD engine, because MHD calculation is a computationally intensive process.

Due to the 10ms frame length, a feature vector arrives at the MHD engine of the state level every 10ms. The MHD engine calculates the likelihood of each feature vector corresponding to each of a number of states derived from an acoustic model. Each state is also a 39 dimensional vector, having the same spectral components as the feature vector. The MHD engine calculates the difference in 39-D space between the feature vector and each state to obtain the likelihood for that state. A parallel MHD or a single MHD arrangement may be used.

The feature vectors may be sent to a feature vector buffer for model adaptation, which determines whether the observed features fit well with the acoustic model, and may initiate an adaptation of the model if the fit is not sufficiently good.

Scores indicating the state likelihoods or probabilities are output from the MHD engine to the PINE array. It is preferable to use likelihoods rather than probabilities, because likelihoods can be added together to give a total likelihood. Probabilities must be

multiplied, which can give rise to rounding errors. In this embodiment, eight separate pines are used, each of 4 kilobytes, thus storing 4000 phones per pine.

During recognition, the Phone Instance Network Engine (PINE) Array will contain a network of phone instances. Tokens representing search hypotheses in the form of a probability(likelihood) will exist in some of the HMM states of the phone instances. The tokens are propagated into all available states and updated with the corresponding state output probabilities. Maximum model, word end and beam pruning are applied to the tokens to ensure the search space is kept bounded. When a token is in the final state of a phone model and has avoided being pruned, the PINE makes a next phone request to the Phone level. The PINE array may receive a number of new phone models from the phone level which must be appended to the appropriate existing phone.

Figure 2 shows the data structures stored in the PINE array, and other main data structures used during the acoustic search. Each PINE includes a phone instance CAM and a phone instance RAM. The phone instance CAM stores the phone instance number (labelled "phone_no") which uniquely identifies each phone instance. The phone instance CAM also stores the state IDs of the three HMM states used to model the phone, and flags indicating the position of the phone. The "back" flag indicates if the phone is at the back of the wavefront. The "front" flag indicates if the phone is at the front of the wavefront. The "empty" flag indicates if there are no tokens in any of the three states for that phone. The "end" flag indicates if the phone is at the end of a word.

The phone instance RAM stores the token scores for each of the three states corresponding to each phone. The st_dum field indicates a dummy state, which is used to represent a state prior to the first state of the first phone. The "tmp" field is used for passing a token from a previous phone in the lexical tree to the current phone. This will be explained in more detail later. The prev_phone_no field is for storing a pointer to the previous phone instance in the network. The wordlat_node_id identifies the position of the phone instance on the word lattice node, and the LM_factor stores the language model factor for that phone instance.

At the phone level a record of all phones on the wave front must be maintained. If a phone instance is on the wave front, the phone_no stored in the phone instance CAM will point to the details in the wave front phone CAM and RAM.

The wave front phone CAM and RAM keeps pointers into the grammar and lexical tree. When a token in a wave front phone reaches the last state, the PINE array makes a request to the phone level by passing back the phone_id. The phone_id is used as a key to determining the token's grammer_node_id and LT_node_id within the wave front CAM/RAM structure. If the token is at a word end the grammar model is checked. A grammar identifier points to the type of grammar.

At the word level, figure 2 shows an example word lattice structure, with silence having a word_id of 0, and a wordlat_node_id of 6. This is shown as connected to two word nodes – "which" has a word_id of 238 and a wordlat_node_id of 4, and "neither" has a word_id of 3556 and a wordlat_node_id of 5.

At the highest level, i.e. the grammar level, a simple grammar model is shown which will allow the phrase 'silence which house silence' or 'silence neither house silence' to be spoken. Whenever a word end is reached during the search, the grammar model must be consulted to determine the next possible word(s) or lexical tree(s). This allows the word lattice at the word level, consisting of a network of words, to be assembled. Whenever a word end is reached, the word is added to the lattice and the wordlat_node_id updated accordingly.

Figure 3 shows a single phone instance network engine (PINE). At system power up the state transition RAM will be loaded with the appropriate state transition vector but the CAM and RAM memories will be empty. When the system is initialised an initial set of phone model(s) will be instantiated with a token in the dummy state and a path score of zero. The phone models are received from the phone level.

When the audio data is processed, a stream of all possible state output probabilities will be received from the MHD engine. A counter that is synchronous to the stream of state output probabilities from the MHD produces a concurrent stream of state_ids. The job

of the CAM is to perform a search to see if the state_id is instantiated anywhere in the CAM state_id fields. If it exists and the location doesn't have its empty flag set then this indicates that a token may need to be updated. The match signal from the CAM must be encoded into an address for the RAM. The corresponding RAM address can then be read. Any token that will end up in the identified state can then have it's path score updated and including any transition vector likelihood.

It is possible that a single state_id generates multiple CAM matches. To handle this situation the CAM and RAM are effectively decoupled by means of a match fifo and repeat count fifo. The state_id and corresponding state_output_probability are stored in the repeat count fifo. During a subsequent frame when there is no match, this state_id can be revisited, ignoring the matches already dealt with.

All searched phone instances corresponding to words in the current possible lexical tree are stored in the CAM-RAM. This includes phone instances at the beginning of the word, middle of the word and end of the word. These phone instances are connected by the "previous phone instance" field, which allows traceback through an entire word within the CAM-RAM.

A mechanism is thus required to pass tokens from one phone instance to the next. This mechanism is provided by the tmp field in the RAM.

The tmp field in the RAM holds tokens during token passing.

At the start of each time frame, the audio input is converted into a feature vector, the MHD values calculated, and the MHD values used to update the token scores. After update of all tokens, 2 passes are made through the CAM-RAM, firstly for pruning, and then for token passing. The pruning in this example removes tokens of below a threshold score, although other types of pruning are also possible. The pruning may involve removing a predetermined proportion of the tokens, chosen as those of lowest likelihood. A third possible method of pruning is removing all tokens with scores below a first predetermined value, but at word ends removing all tokens with scores below a

second predetermined value, the second predetermined value being greater than the first predetermined value.

The token passing works as follows. Each phone instance points back to a previous phone, due to the prev_phone_no field stored in the RAM. During the token passing, the third token of a previous phone is grabbed and put into the tmp field of the current phone.

In other words, if a token is on the third state, and it is not pruned, it must enter the next state of the next phone. To do this, the tokens that lie in the third state are moved across to the tmp field.

For example, phone no. 345 needs to look at phone no. 320 to see if any tokens are in the third state. If yes, it takes these tokens and puts them in the tmp field.

One field in the CAM is labelled "front". This indicates whether the phone is on the wave front. If the phone is not on the wave front, then another phone will have picked up the token in the third state. However, if the phone is on the wave front, then no other phone will have picked up the token on the third state. Therefore, if there is a token in the third state, and the phone is on the wavefront, it is necessary to generate a next phone request.

The current phone number, word lattice identifier and word end flag are sent to the wave front manager, to request a next phone. If the word end flag indicates a word end, then the score is also passed to the wave front manager. The score can be passed in any case, but it is only required at a word end.

The phone level generates the next phone or phones. A new phone instance starts, and all of state1, state2 and state3 have scores pre-set to zero. The old score is provided in the tmp field.

The old phone is still within the wave, but it is unlikely to be at the back of the wave at this stage. When it gets to the back of the wave, it is deleted. Until it gets there, it

remains in the CAM-RAM as before. The third token remains where it was, since tokens can either stay in the third state or move on.

Note that it is possible to end up with more that one-token in each state. If this occurs, then the token with the poorer score is deleted. This does not affect the result of the lexical tree search, because both tokens on the same state have exactly the same word history.

Figure 4 shows a block diagram of the wave front manager. The wave front manager is responsible for maintaining a record of the set of phone models that exist at the front of the acoustic search space. This set of phone models is referred to as the "wave front." The key requirement for this block as to minimise the time between receiving a "next phone request" and generating the appropriate responses. The wave front manager receives the next phone requests from the PINE array and generates appropriate responses for the Phone Book, Word Engine, Path Merge Unit and Grammar engine The wave front manager also receives new wave front phone models from the Phone Book and updates the wave front.

The wave front manager receives the next phone request from the PINE array, via the phone request arbiter. The next phone request consists of a phone_no, word_end_flag and wordlat_node_id. If this phone is at the end of a word, the word_end_flag will be set.

The wave front manager uses the phone_no as input to the wave front phone CAM. As each phone_no is unique, it is guaranteed that only a single match will be generated. The wave front manager uses the CAM generated address to access the RAM. It then sends retrieved data from the RAM to the phone book, grammar engine, Path Merge Unit and word engine.

The wave front manager removes phone instance from CAM and RAM by writing 0's into CAM and RAM, and increments the empty count by one.

Sometime later, the wave front manager receives new phone instances from the phone book and place them in a fifo. The phone instances are taken from the fifo and written into the CAM and RAM at the next_empty_address. The empty_count is decremented by one. A match is run on the "occupy" flag, and the address is placed into next_empty_address register.

Both the phone level and the state level make use of a path merge unit to reduce the number of possible paths, by path merging. Figure 5 shows the use of link nodes to permit path merging. In the present embodiment, all word end phone instances are linked to word start triphones with a link node. During each frame the path merge unit looks for *word end* tokens that can be recombined. This is done by looking for word end phones that have the same word_id and word history. When they are found a link is added using the path merge unit in the PINE array.

Figure 6 shows a diagram of the architecture at the Phone level, showing the path merge unit. The path merge unit includes a path merge CAM and a path merge RAM. The path merge CAM stores word_id, word history for two words, and an "occupy" field. The path merge RAM stores a link_node_id. A CAM RAM occupancy manager keeps track of the empty addresses in the CAM and RAM. The path merge unit is only passed information from the wave front manager if the phone is a *word end phone.* For each frame, there is a check to see if there is an identical word with the same history, and if there is, the link_node_id is passed to the path merge unit. If there isn't an identical word with the same history, a new record is written into the Path merge CAM-RAM.

Figure 7 is a block diagram showing the Phone Book architecture. The Phone Book is tasked with generating the next phone models when given a pointer into the lexical tree. The key requirement for this block as to minimise the time between receiving a "lexical tree address" and generating the appropriate responses, in particular the next phone model instance(s).

The lexical tree is used to build the next triphones based on the current context.

Current triphone	Next triphone	comments

Left context = a Central context = b Right context = c	discarded Left context = b Central context = c **Right context = ?**	**next_leftmph_id stored in wave front manager** **current LT_node_id stored in wave front manager** **must be looked up. Large (< 45) for new word start**

The LT_node_id, next_left_mph_id and prev_phone_no are received at the phone generator from the wavefront manager. The LT_node_id is then used to address the lexical tree store and return the central monophone_id. One or more further accesses to memory are performed using the microcoded instructions, each memory access returning the right monophone_id.

A two stage lookup is accessed using the three monophone_ids (giving a triphone), the associated state_ids. The phone model is assembled, and returned to the PINE array. The wavefront manager is then updated by sending it relevant parts of the phone model.

The lexical tree store is a micro-coded ROM with both instruction and data fields. The lexical tree store holds a number of lexical trees. Each lexical tree may hold single words or multiple words. Each lexical tree comprises lexical tree elements, and each element has three fields. For now, the fields will be integers and will be converted to std_logic_vector later. The following is an example of the lexical tree store, with the LT data in word_id order.

```
ZERO   z iy r ow        19  8 12 11  0  0  0  0
ONE    w ah n           18  2 10  0  0  0  0  0
TWO    t uw             14 16  0  0  0  0  0  0
THREE  th r iy          15 12  8  0  0  0  0  0
FOUR   f ow r            6 11 12  0  0  0  0  0
FIVE   f ay v            6  3 17  0  0  0  0  0
SIX    s ih k s         13  7  9 13  0  0  0  0
SEVEN  s eh v ih n      13  4 17  7 10  0  0  0
EIGHT  ey t              5 14  0  0  0  0  0  0
NINE   n ay n           10  3 10  0  0  0  0  0
OH     ow               11  0  0  0  0  0  0  0
```

```
sp 0
sil 1
ah 2
ay 3
eh 4
ey 5
f 6
ih 7
iy 8
k 9
n 10
ow 11
r 12
s 13
t 14
th 15
uw 16
v 17
w 18
z 19
```

The following shows the LT data in MP (monophone) start order.

```
EIGHT  ey t              5 14  0  0  0  0  0  0
FIVE   f ay v            6  3 17  0  0  0  0  0
FOUR   f ow r            6 11 12  0  0  0  0  0
NINE   n ay n           10  3 10  0  0  0  0  0
OH     ow               11  0  0  0  0  0  0  0
SEVEN  s eh v ih n      13  4 17  7 10  0  0  0
```

```
SIX     s ih k s          13  7  9 13  0  0  0  0
TWO     t uw              14 16  0  0  0  0  0  0
THREE th r iy             15 12  8  0  0  0  0  0
ONE     w ah n            18  2 10  0  0  0  0  0
ZERO    z iy r ow         19  8 12 11  0  0  0  0
```

The following is an example of the three fields of a lexical tree element. The first field is an instruction field, and the second and third fields are data fields.

Fields:	1st instr 2b	2nd data 6b	3rd data 6b	
	00	MP	MP	- raw data field.
	01	br	br	- branch address.
	10	br	br	- branch address.
	11	word_id		- double field.

Note that there might be no difference between "01" and "10"

To build the "ey" tree, we only have to consider "EIGHT", because this is the only word in the lexical tree which starts with "ey". EIGHT is represented as "ey t", with phone_nos 5 14 0 0 0 0 0 0. The first lexical tree element stores the raw data specifying the phone_nos, the second lexical tree element is an end field, and the subsequent lexical tree element is set to zero.

```
0  00  5 14   - raw data
1  11  8  0   - end field
2  00  0  0
```

To build the "f" tree, we have:

```
FIVE   f ay v      6  3 17  0  0  0  0  0
FOUR   f ow r      6 11 12  0  0  0  0  0

0  01  4  0  -- branch forward four places as well
1  00  6  3  -- as the implied other brach of next (one).
2  00  3 17
3  11  5  0  -- all addresses are relative offsets.
4  00  6 11
5  00 11 12
6  11  4  0
7  00  0  0
```

To build the "n" tree, we have:

```
NINE        n ay n        10 3 10 0 0 0 0 0

0  00 10  3
1  00  3 10
2  11  9  0
3  00  0  0
```

To build the "ow" tree, we have "ow" in both the first and second data fields, as no tree is going to be a single monophone long.

```
OH          ow            11 0 0 0 0 0 0 0

0  11 11  0
1  00  0  0
```

To build the "s" tree, we have:

```
SEVEN       s eh v ih n   13  4 17  7 10 0 0 0
SIX         s ih k s      13  7  9 13  0 0 0 0

0  01  6  0  -- branch forward six places.
1  00 13  4
2  00  4 17
3  00 17  7
4  00  7 10
5  11  7  0
6  00 13  7
7  00  7  9
8  00  9 13
9  11  6  0
10 00  0  0
```

To build the "t" tree, we have:

```
TWO   t uw      14 16 0 0 0 0 0 0

0  00 14 16
1  11  2  0
2  00  0  0
```

To build the "th" tree, we have:

```
THREE       th r iy     15 12 8 0 0 0 0 0

0  00 15 12
1  00 12  8
2  11  3  0
3  00  0  0
```

To build the "w" tree, we have:

```
ONE   w ah n      18 2 10 0 0 0 0 0

0  00 18 12
1  00 12 10
2  11  1  0
3  00  0  0
```

To build the "z" tree, we have:

```
ZERO  z iy r ow    19  8 12 11  0  0  0  0

0  00 19  8
1  00  8 12
2  00 12 11
3  11  0  0
4  00  0  0
```

Once all of the branches have been built, the next stage is to concatenate the tree. It appears that four elements make a page, and each element is two bytes.

```
                                                  page
ey:     00  5 14   11  8  0   00  0  0   00  0  0    5
 f:     01  4  0   00  6  3   00  3 17   11  5  0    6
        00  6 11   00 11 12   11  4  0   00  0  0    7
 n:     00 10  3   00  3 10   11  9  0   00  0  0    8
ow:     11 11  0   00  0  0   00  0  0   00  0  0    9
 s:     01  6  0   00 13  4   00  4 17   00 17  7   10
        00  7 10   11  7  0   00 13  7   00  7  9   11
        00  9 13   11  6  0   00  0  0   00  0  0   12
 t:     00 14 16   11  2  0   00  0  0   00  0  0   13
th:     00 15 12   00 12  8   11  3  0   00  0  0   14
 w:     00 18 12   00 12 10   11  1  0   00  0  0   15
 z:     00 19  8   00  8 12   00 12 11   11  0  0   16
```

Then, the start_lookup header is defined. Each row (page)is 8 bytes and hence the header takes 3 rows. The end_lookup header is two rows. Therefore, the lexical tree store proper starts at the 6th (#5) page.

```
sp sil  ah  ay  eh  ey   f  ih        - MP
 0   1   2   3   4   5   6   7        - MP_id
 3   0   0   0   5   0   6   0        - data stored.

iy   k   n  ow   r   s   t  th
 8   9  10  11  12  13  14  15
 0   0   8   9   0  10  13  14

uw   v   w   z   .   .   .   .
16  17  18  19   .   .   .   .
 0   0  15  16   0   0   0   0
```

The end_lookup header is defined. 20 is out of bounds.

```
ey  f  n ow  s  t th  w
 5  6 10 11 13 14 15 18

z  .  .  .  .  .  .  .
19 20 20 20 20 20 20 20
```

To put the above information all together:

START	sp:ih	3 0	0 0	5 0	6 0	0
LOOKUP	iy:th	0 0	8 9	0 10	13 14	1
	uw: v	0 0	15 16	0 0	0 0	2
END	ey:w	5 6	10 11	13 14	15 18	3
LOOKUP	z	19 20	20 20	20 20	20 20	4
LEXICAL	ey:	00 5 14	11 8 0	00 0 0	00 0 0	5
TREE	f:	01 4 0	00 6 3	00 3 17	11 5 0	6
STORE		00 6 11	00 11 12	11 4 0	00 0 0	7
	n:	00 10 3	00 3 10	11 9 0	00 0 0	8
	ow:	11 11 0	00 0 0	00 0 0	00 0 0	9
	s:	01 6 0	00 13 4	00 4 17	00 17 7	10
		00 7 10	11 7 0	00 13 7	00 7 9	11
		00 9 13	11 6 0	00 0 0	00 0 0	12
	t:	00 14 16	11 2 0	00 0 0	00 0 0	13
	th:	00 15 12	00 12 8	11 3 0	00 0 0	14
	w:	00 18 12	00 12 10	11 1 0	00 0 0	15
	z:	00 19 8	00 8 12	00 12 11	11 0 0	16

```
  3 0    0 0    5 0    6 0
  0 0    8 9    0 10   13 14
  0 0   15 16    0 0    0 0
  5 6   10 11   13 14   15 18
 19 20  20 20   20 20   20 20
00 5 14  11 8 0  00 0 0  00 0 0
01 4 0  00 6 3  00 3 17  11 5 0
00 6 11  00 11 12  11 4 0  00 0 0
00 10 3  00 3 10  11 9 0  00 0 0
11 11 0  00 0 0  00 0 0  00 0 0
01 6 0  00 13 4  00 4 17  00 17 7
00 7 10  11 7 0  00 13 7  00 7 9
00 9 13  11 6 0  00 0 0  00 0 0
00 14 16  11 2 0  00 0 0  00 0 0
00 15 12  00 12 8  11 3 0  00 0 0
00 18 12  00 12 10  11 1 0  00 0 0
00 19 8  00 8 12  00 12 11  11 0 0
```

This information is converted to hexidecimal. The first five are byte fields.
The rest are:
2bit - uCode.

6bit - data.
2bit - null.
6bit - data

03 00	00 00	05 00	06 00
00 00	08 09	00 0A	0D 0E
00 00	0F 10	00 00	00 00
05 06	0A 0B	0D 0E	0F 12
13 14	14 14	14 14	14 14
05 0E	C8 00	00 00	00 00
84 00	06 03	03 11	C5 00
06 0B	0B 0C	C4 00	00 00
0A 03	03 0A	C9 09	00 00
CB 00	00 00	00 00	00 00
86 00	0D 04	04 11	11 07
07 0A	C7 00	0D 07	07 09
00 0D	C6 00	00 00	00 00
0E 10	C2 00	00 00	00 00
0F 0C	0C 08	C3 00	00 00
12 0C	0C 0A	C1 00	00 00
13 08	08 0C	0C 0B	C0 00

The above is the micro code for the digit lexical tree. It allows branching, and it holds the word_id. It knows which lexical trees it contains and where they are housed. It also stores the possible end monophone list.

Figure 8 shows a further example of the Lexical Tree RAM, giving examples of different types of branching in the lexical tree, and how these are represented in the RAM.

Figure 9 is a block diagram showing the data structure that is supported for the word link record, to enable word n-best decoding. The word engine maintains a record of the words that have been currently searched by the PINE array. It is used to determine the word history and allow trace back during or at the end of search. The word history is needed to determine the language model score and whether particular paths can be recombined or merged.

Figure 10 shows a block diagram of the word engine. This is designed to allow fast access to the word history (thus allowing rapid phone creation at the phone level), and to allow reasonably fast trace back. The Word Engine is primarily a memory store that fills incrementally. The decoupling of the wordlat_node_id and the physical address

will allow the possibility of being able to keep the WLRs for a given word in contiguous addresses, thus avoiding the need for the pointer field.

The language model interface is designed to allow the Phone Book to quickly obtain the appropriate LM factor for the phone model that it is currently constructing. Based on the phones in the wave front, all possible next LM factors can be determined. This will be done in software together with the language model factor engine. The ARM 9 will maintain its own record of the wavefront using the same next phone requests received by the wave front manager and its own copy of the lexical tree.

Figure 11 shows the architecture of the LM interface. This allows the Phone book to quickly access the LM factor for the phone that is currently being generated. Prior to the phone generation phase LM_factors are computed for all possible phones that may be generated in the next frame.

The grammar engine is designed to give the Phone Book a quick response to requests for the next grammar node(s). In the case of a task grammar where the lexicon may be limited to a few words(e.g. yes, no), it may be desirable to use the same lexical tree. In this case it will be necessary for the grammar, to provide lexical tree pointers such that the phone generator can jump to the appropriate part of the lexical tree.

For a speech recognition system it is also necessary to specify the grammar. Again we will use HTK's Standard Lattice Format(SLF) to specify the grammar. In practice it will be convenient to use HTK's HParse format to specify the grammar using a high level grammar notation. This can then be converted into using the HParse tool. Consider having to specify a grammar for a digit recognition task. The HParse format for this is as follows:

```
$digit = one | two | three | four | five | six | seven | eight | nine | zero
$digit_sp = $digit sp
$noise = lipsmack | breath | sil
( [sil] < $digit_sp | $noise > [sil] )
```

where | denotes alternatives, < > denotes one or more repetitions, { } denotes zero or more repetitions, [] denotes that item is optional, and () contains the grammar.

Figure 12 shows the network that is defined by this grammar. The network starts and ends with a "sil" phone, representing silence. Between the start and end, a number of possible routes correspond to further silence, a breath, a lipsmack sound, or a detected word.

Figure 13 shows the grammar engine architecture. When a next phone request arrives, the next grammar node(s) must be retrieved from memory. In the case of a task grammar, the phone book should be given the lexical tree memory addresses to allow the phone to be determined.

Figure 14 shows a single frame of the Audio Codec link protocol. The Advanced Audio CODEC Interface (AACI) is an ARM Primecell that provides an interface to off-chip AC97 audio codecs. AC97 is Intel's audio codec specification, initially created for the PC market. The AACI is an AMBA slave block that connects to the Advanced Peripheral bus (APB). The AACI connects to the off-chip codec using the AC-link protocol.

The AC-link protocol is a bi-directional, fixed clock rate, serial digital stream. AC-link handles multiple input and output PCM audio streams as well as register control accesses. It employs a time division multiplexed (TDM) scheme that divides each audio frame into 12 outgoing and 12 incoming data streams each with up to 20 bit sample resolution. The tag slots are used to indicate which of the subsequent slots contain valid data. Add and dat are the control register address and data slots for read and writes. The adc slots contain the outputs from the analog to digital converters. Assuming that the ADC sample rate is set to 16 kHz then the AC-link controller will receive 16 bits of data every third codec frame.

The AACI has 4 channels that can be operated simultaneously. They are accessed through DMA.

Figure 15 shows the system timing. Each frame is divided into three sections:

- Token update (MHD wash)
- First pass - inter phone token passing
- Second pass – garbage collection, max model pruning

The horizontal direction of the figure indicates elapsing time, and various subcircuits and processes are indicated along the vertical direction of the figure. The top part of the diagram shows frame count, and push to dictate button, and the lower part of the diagram shows various subcircuits and processes, within a single time frame, whilst the push to dictate button is engaged. These subcircuits and processes are language model interface, word engine, phone book, PINE array, MHD engine, mean & covar reg, and feature vector reg.

At the start of the time frame, the feature vector is generated, and the MHD values are calculated. The MHD values are then passed to the PINE array, which engages in beam pruning and token score updating. The phone book receives next phone requests and returns next phone responses. The word engine adds word link records, and the language model interface supplies language model factors.

In the second part of the frame, pruning thresholds are calculated for the PINE array, for the next frame, and the PINE array engages in token passing. The language model interface transfers language model factors for the next frame.

Whilst the "push to dictate" button remains activated, the above process repeats in a sequence of frames.

The methods and apparatus according to the invention are not limited to embedded mobile systems, but may be used with any type of computing device, including PCs, workstations, laptop computers, notebook computers, handheld computing devices, dictation machines, PDAs (personal digital assistants), personal organisers, mobile phones, games consoles, video players, etc. Embedded computers according to the present invention may be embedded in a wide range of apparatus and machinery, such

as electronic entertainment products, domestic appliances, factory machinery, vehicles, etc.

Embodiments of the present invention may be implemented as an IP core in a System on Chip (SOC) or a discrete device. Multiple chips or bare die may be connected to provide a larger vocabulary system.

Although the apparatus may include all or some of an input for accepting audio input, a circuit for digitising the audio input, a circuit for dividing the audio input into a plurality of time-frames, and a circuit for analysing the spectral components of each time frame to characterise the audio in that frame, this is not essential. The apparatus may also include a circuit for calculating a score indicating the probability or likelihood that each audio frame corresponds to each of a plurality of different possible states, but again, this is not essential. Instead, the apparatus may include an input for receiving the pre-calculated scores.

In the described embodiments, n-gram language models are used. For example, the Bigram language model is based on outputting the probability that a given second word directly follows a given first word. However, it is possible to use alternative language models instead. It is possible to use a mixture of different models. For example, it is possible to run a word-dependent n-best decode applying a bigram language model, and the resulting word link record or word graph could then be rescored using a trigram language model.

The Viterbi beam search algorithm may be used in embodiments of the invention, to perform the search. A word level search may be performed. Dummy states may be used for the start and end of an utterance

The grammar level model may be used to add distance elements into the likelihood calculation. This is not essential, but tends to give much better results.

The HMM models described in the embodiments comprise self connections to each states and single connections to a neighbouring state. However, the present invention is

not limited to such HMM models, and for example, HMM models with additional connections between states may also be used. Alternatively, embodiments of the invention may use alternative types of acoustic models, rather than HMM models.

The system may comprise a plurality of language models (e.g. English, French). A model of words may comprise actual words, phrases, silence, lipsmack sounds, and/or other sounds for recognition by the apparatus.

As previously discussed, the phone level has been found to be the best level for lexical tree searching, rather than the state level or the word level. However, this invention is not limited solely to phone level searching.

While the invention has been described in terms of what are at present its preferred embodiments, it will be apparent to those skilled in the art that various changes can be made to the preferred embodiments without departing from the scope of the invention, which is defined by the claims.

CLAIMS:

1. A speech recognition circuit comprising:

a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words;

a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and

a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

2. The speech recognition circuit of claim 1, wherein for at least some of the states, the node identifier comprises a pointer to a node identifier for another state.

3. The speech recognition circuit of claim 1 or claim 2, wherein the lexical tree is arranged with each node corresponding to a phone, and each said state corresponds to a phone or group of adjacent phones in the lexical tree.

4. The speech recognition circuit of claim 3, wherein the memory structure comprises a first part for storing phone instance identifiers uniquely identifying phones or groups of phones and for storing said state identifiers, and the memory structure comprises a second part for storing said phone instance identifiers and node identifiers directly referencing the lexical tree.

5. The speech recognition circuit of claim 3 or 4, further comprising a counter for sequentially generating phone instance identifiers as unique identification numbers corresponding to phones or groups of phones in the lexical tree.

6. The speech recognition circuit of any previous claim, further comprising a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

7. The speech recognition circuit of any previous claim, wherein the memory structure is configured to hold a record for each of a plurality of nodes in the lexical tree, the record including said states, a previous node identifier for identifying a previous node in the lexical tree, and information indicating whether said each node is a previous node for another node stored in said memory structure,

the speech recognition circuit including a circuit to identify a first node which is not a previous node of another record, and to provide a new data record corresponding to a new node, the new node being the next node after the first node in the lexical tree, using a score associated with the first node.

8. The speech recognition circuit of claim 7, wherein said record comprises a field for storage of a score corresponding to a previous node.

9. The speech recognition circuit of claim 7 or claim 8, further comprising a circuit for overwriting an old record in the memory structure with said new record according to the value of said score for the old record and according to whether said old record is a previous node to said new node or to another node in said memory structure.

10. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises a plurality of memory elements, the memory structure being adapted for independent lookup of states in each of said memory elements.

11. The speech recognition circuit of claim 10, wherein one of said memory elements is configured to store data relating to a first lexical tree, and a second of said

memory elements is configured to store data relating to a second lexical tree, the second lexical tree comprising a different model of words than the first lexical tree.

12. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises at least one content addressable memory and at least one random access memory, the content addressable memory being configured to output addresses of the random access memory.

13. The speech recognition circuit of any one of the previous claims, further comprising a delay circuit for storing a state identifier having a plurality of occurrences in the memory structure, and storing the score update for said state identifier, the delay circuit being configured to provide said state identifier and score after a lookup for a second state identifier produces no match in the memory structure.

14. The speech recognition circuit of any one of the previous claims, comprising a circuit to set a flag indicating an unfavourable score for all records which are previous records and which have scores outside a predetermined range of values.

15. The speech recognition circuit of any one of the previous claims, wherein said memory structure comprises a first memory for receiving and storing said state identifiers, and a second memory for receiving and storing said node identifier, said first memory being configured for lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores, and the second memory being adapted to store a record identifier for a record in the first memory and a reference to the node of the lexical tree for the record identifier, said second memory storing a subset of said records which are not previous records.

16. A method of speech recognition, comprising

providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, the lexical tree comprising a model of words;

providing scores corresponding to said state identifiers;

receiving and storing state identifiers identified by node identifiers identifying a node or group of adjacent nodes in a memory structure;

looking up said memory structure to identify particular state identifiers;

reading of the scores corresponding to the state identifiers;

receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input;

modifying said scores by adding said score updates to said scores;

writing back of the scores to the memory structure after modification of the scores; and

selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

17. The speech recognition circuit of claim 1, wherein for at least some of the states, the node identifier comprises a pointer to a node identifier for another state.

18. The speech recognition circuit of claim 1 or claim 2, wherein the lexical tree is arranged with each node corresponding to a phone, and each said state corresponds to a phone or group of adjacent phones in the lexical tree.

19. The speech recognition circuit of claim 3, wherein the memory structure comprises a first part for storing phone instance identifiers uniquely identifying phones or groups of phones and for storing said state identifiers, and the memory structure comprises a second part for storing said phone instance identifiers and node identifiers directly referencing the lexical tree.

20. The speech recognition circuit of claim 3 or 4, further comprising a counter for sequentially generating phone instance identifiers as unique identification numbers corresponding to phones or groups of phones in the lexical tree.

21. The speech recognition circuit of any previous claim, further comprising a counter for sequentially generating state identifiers, and using said generated state identifiers to sequentially lookup said states in the memory structure.

22. The speech recognition circuit of any previous claim, wherein the memory structure is configured to hold a record for each of a plurality of nodes in the lexical tree, the record including said states, a previous node identifier for identifying a previous node in the lexical tree, and information indicating whether said each node is a previous node for another node stored in said memory structure,

the speech recognition circuit including a circuit to identify a first node which is not a previous node of another record, and to provide a new data record corresponding to a new node, the new node being the next node after the first node in the lexical tree, using a score associated with the first node.

23. The speech recognition circuit of claim 7, wherein said record comprises a field for storage of a score corresponding to a previous node.

24. The speech recognition circuit of claim 7 or claim 8, further comprising a circuit for overwriting an old record in the memory structure with said new record according to the value of said score for the old record and according to whether said old record is a previous node to said new node or to another node in said memory structure.

25. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises a plurality of memory elements, the memory structure being adapted for independent lookup of states in each of said memory elements.

26. The speech recognition circuit of claim 10, wherein one of said memory elements is configured to store data relating to a first lexical tree, and a second of said memory elements is configured to store data relating to a second lexical tree, the second lexical tree comprising a different model of words than the first lexical tree.

27. The speech recognition circuit of any one of the previous claims, wherein the memory structure comprises at least one content addressable memory and at least one random access memory, the content addressable memory being configured to output addresses of the random access memory.

28. The speech recognition circuit of any one of the previous claims, further comprising a delay circuit for storing a state identifier having a plurality of occurrences in the memory structure, and storing the score update for said state identifier, the delay circuit being configured to provide said state identifier and score after a lookup for a second state identifier produces no match in the memory structure.

29. The speech recognition circuit of any one of the previous claims, comprising a circuit to set a flag indicating an unfavourable score for all records which are previous records and which have scores outside a predetermined range of values.

30. The speech recognition circuit of any one of the previous claims, wherein said memory structure comprises a first memory for receiving and storing said state identifiers, and a second memory for receiving and storing said node identifier, said first memory being configured for lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores, and the second memory being adapted to store a record identifier for a record in the first memory and a reference to the node of the lexical tree for the record identifier, said second memory storing a subset of said records which are not previous records.

31. A mobile phone including the speech recognition circuit of any one of claims 1 to 15.

32. A portable electronic device including the speech recognition circuit of any one of claims 1 to 15.

33. A speech recognition circuit comprising:

lexical memory containing lexical data for word recognition, said lexical data comprising a lexical tree data structure comprising a model of words;

means for accessing a state model corresponding to each phone or each group of phones in the lexical tree;

a content addressable memory for storing content addressable data for each phone or group of phones, including states corresponding to said phone or group of phones, and for storing an address value for each said phone or group of phones;

a RAM configured to store accumulated scores for each said phone or group of phones, the accumulated scores being addressable by said address value for each said phone or group of phones;

means to obtain scores that each of a plurality of frames of an audio signals corresponds to each of a plurality of said states;

a counter to sequentially search for each said state in the content addressable memory, to obtain the corresponding address value if the state is found in the content addressable memory;

means to use said address value to access an accumulated likelihood and an accumulator to add said likelihood to the accumulated likelihood;

means to use the phones or groups of phones with the highest accumulated scores to obtain a plurality of next phones from the lexical tree which correspond to the next phone; and

output means for outputting a lexical tree path of highest likelihood.

34. A speech recognition apparatus comprising:

a lexical tree having a corresponding state model;

means for obtaining scores of an audio input corresponding to each of a plurality of states in said state model;

a content addressable memory for storing a marker indicating a part of the lexical tree, and one or more states associated with said part of the lexical tree;

a random access memory addressable by the CAM output, to output accumulated scores for states corresponding to said parts of the lexical tree;

adder means for adding likelihood to said accumulated likelihood, to be stored back in the RAM;

35. A speech recognition apparatus, comprising:

a CAM-RAM arrangement for storing records including pointers to a lexical tree, and accumulative scores for states within the lexical tree;

input means for obtaining scores that an audio frame corresponds to a particular state in the lexical tree;

an accumulator for calculating the updated scores and modifying the records in the CAM-RAM accordingly;

output means for outputting a path of highest likelihood in the lexical tree.

36. A speech recognition apparatus substantially as hereinbefore described with reference to any of figures 1 to 15 of the accompanying drawings.

37. A speech recognition method comprising:

storing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and scores corresponding to said state identifiers in a memory structure, the lexical tree comprising a model of words and the memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

repeating the following sequence of steps for each of a plurality of incoming frames of an audio signal:

obtaining score updates corresponding to the likelihoods that said frame of the audio signal corresponds to each of a plurality of said states;

accessing said memory structure to obtain scores, updating the scores by adding score updates to the scores, and writing back the updated scores to the memory structure;

determining if scores for states furthest on in the lexical tree correspond to a significant likelihood, and if so, then accessing the lexical tree to determine the next set of possible states; and

storing the next set of possible states and said scores of significant likelihood in the memory structure;

38. A speech recognition circuit comprising:

a circuit for providing state identifiers which identify states corresponding to phones or groups of adjacent phones in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words;

a memory structure for receiving and storing state identifiers and phone instance identifiers uniquely identifying instances of phones or groups of phones in the lexical tree;

said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores;

an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and

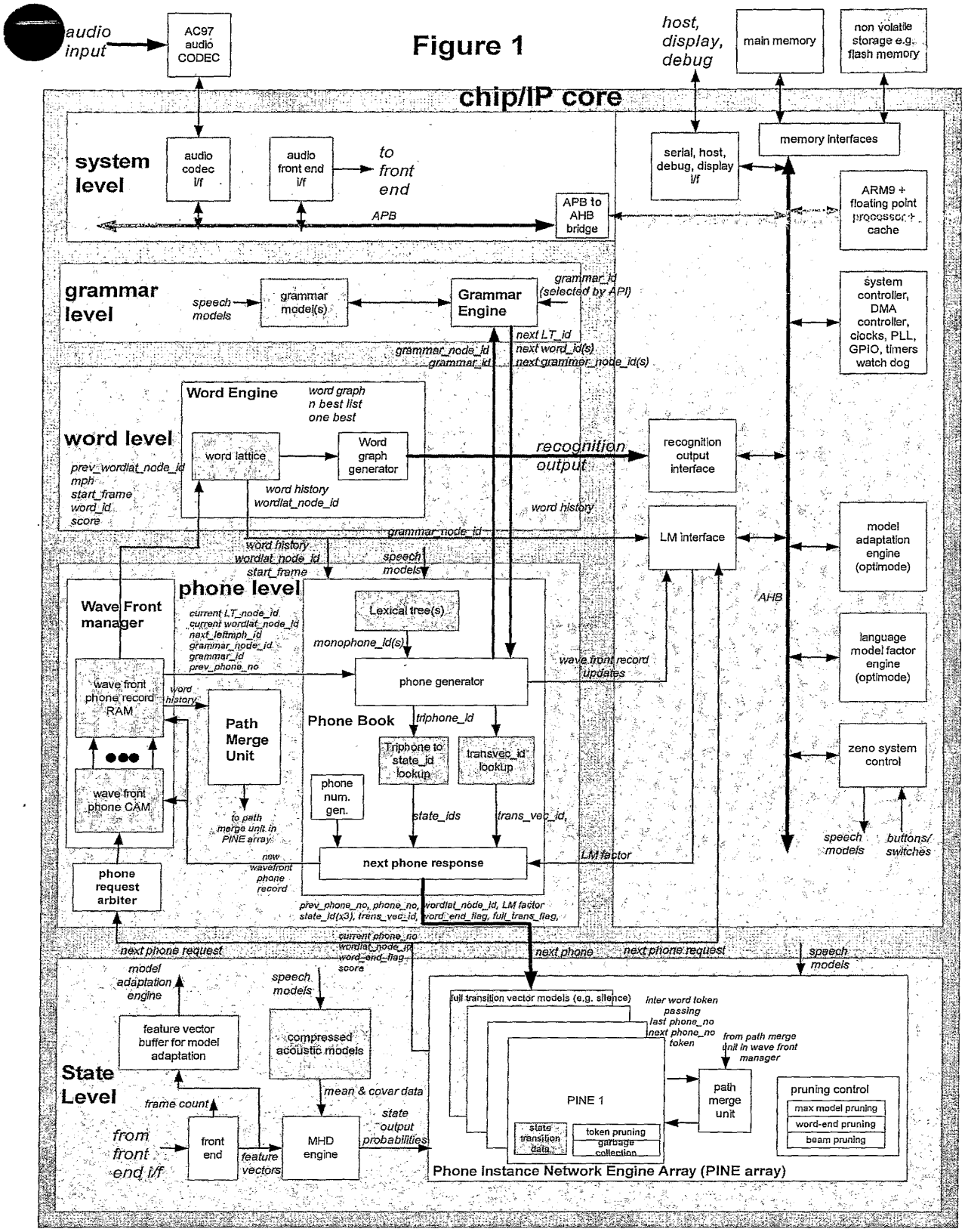
a selector circuit for selecting at least one phone instance identifier according to said scores.

ABSTRACT:

A Speech Recognition Circuit and Method

A speech recognition circuit comprising a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

Figure 1


audio input
AC97 audio CODEC
host, display, debug
main memory
non volatile storage e.g. flash memory
chip/IP core
system level
audio codec I/f
audio front end I/f
to front end
APB
APB to AHB bridge
serial, host, debug, display I/f
memory interfaces
ARM9 + floating point processor + cache
system controller, DMA controller, clocks, PLL, GPIO, timers watch dog
grammar level
speech models
grammar model(s)
Grammar Engine
grammar_id (selected by API)
next LT_id
next word_id(s)
next grammar_node_id(s)
grammar_node_id
grammar_id
word level
Word Engine
word graph
n best list
one best
word lattice
Word graph generator
recognition output
recognition output interface
prev_wordlat_node_id
mph
start_frame
word_id
score
word history
wordlat_node_id
word history
grammar_node_id
LM interface
model adaptation engine (optimode)
word history
wordlat_node_id
start_frame
speech models
phone level
Wave Front manager
current LT_node_id
current wordlat_node_id
next_leftmph_id
grammar_node_id
grammar_id
prev_phone_no
Lexical tree(s)
monophone_id(s)
wave front record updates
AHB
language model factor engine (optimode)
wave front phone record RAM
word history
phone generator
Path Merge Unit
Phone Book
triphone_id
Triphone to state_id lookup
transvec_id lookup
zeno system control
wave front phone CAM
phone num. gen.
to path merge unit in PINE array
state_ids
trans_vec_id
speech models
buttons/ switches
new wavefront phone record
next phone response
LM factor
phone request arbiter
prev_phone_no, phone_no, wordlat_node_id, LM factor
state_id(x3), trans_vec_id, word_end_flag, full_trans_flag,
current phone_no
wordlat_node_id
word_end_flag
score
next phone request
next phone
next phone request
speech models
model adaptation engine
speech models
full transition vector models (e.g. silence)
inter word token passing
last phone_no
next phone_no token
from path merge unit in wave front manager
feature vector buffer for model adaptation
compressed acoustic models
State Level
mean & covar data
PINE 1
path merge unit
pruning control
max model pruning
word-end pruning
beam pruning
frame count
state output probabilities
from front end i/f
front end
feature vectors
MHD engine
state transition data
token pruning
garbage collection
Phone Instance Network Engine Array (PINE array)

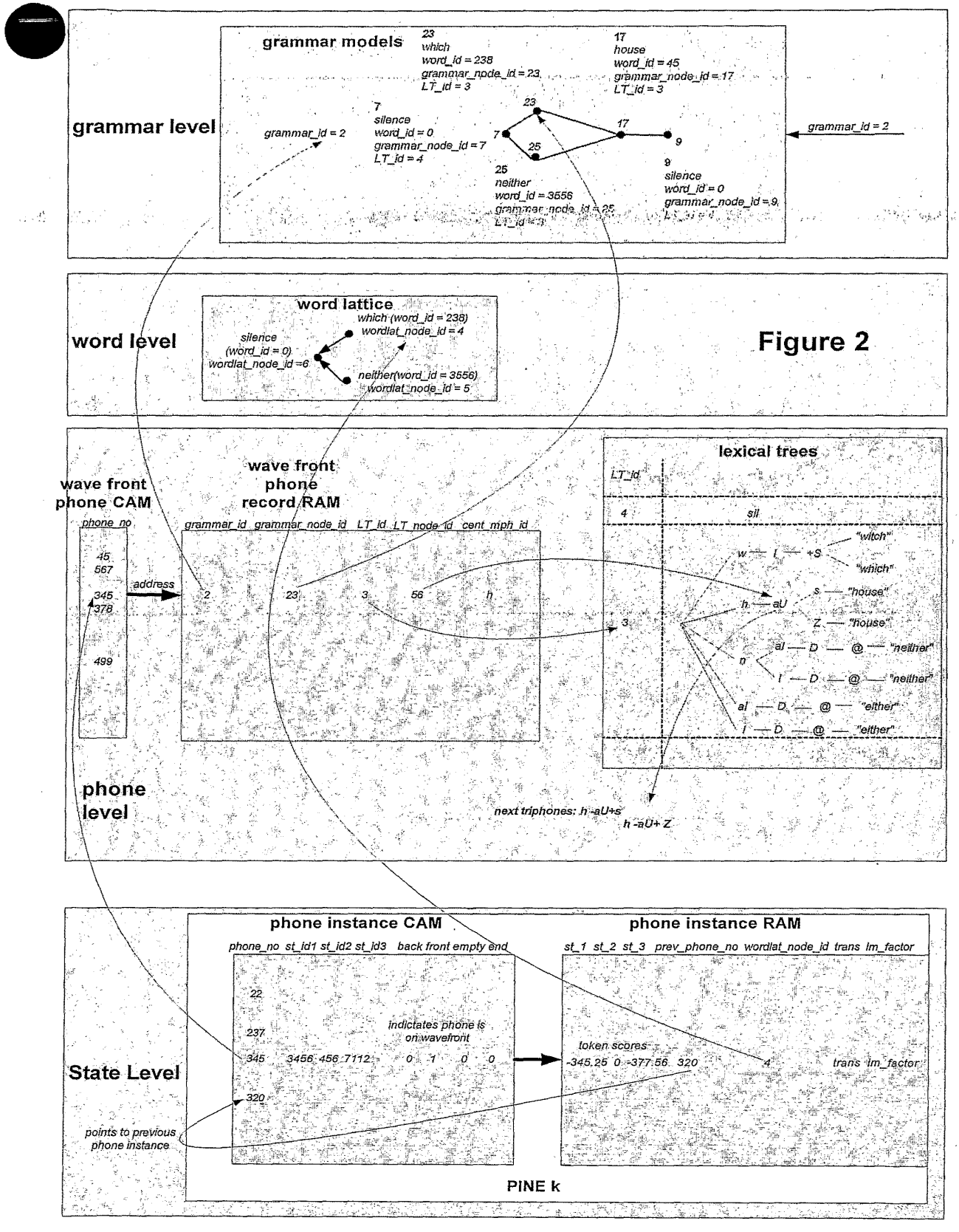
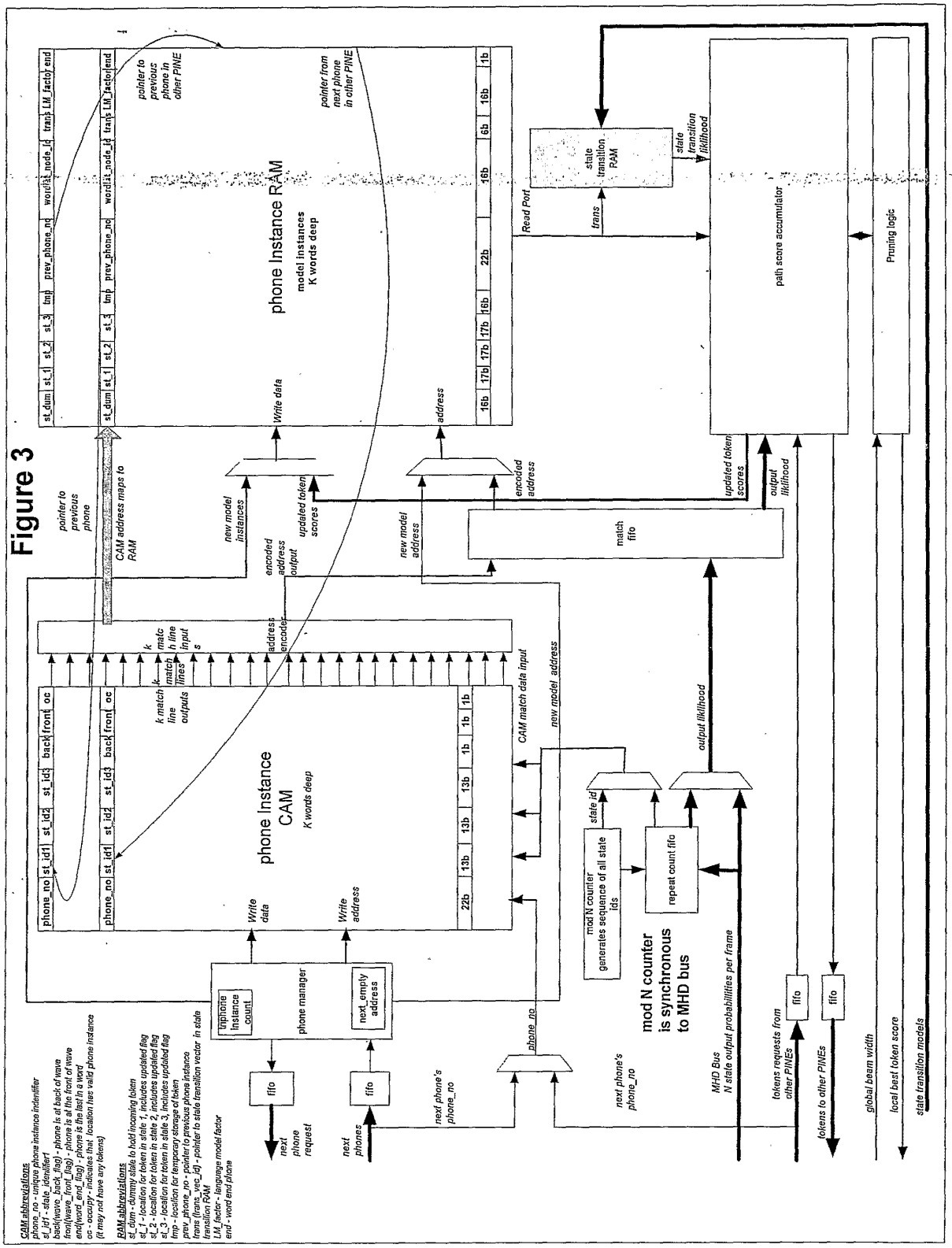
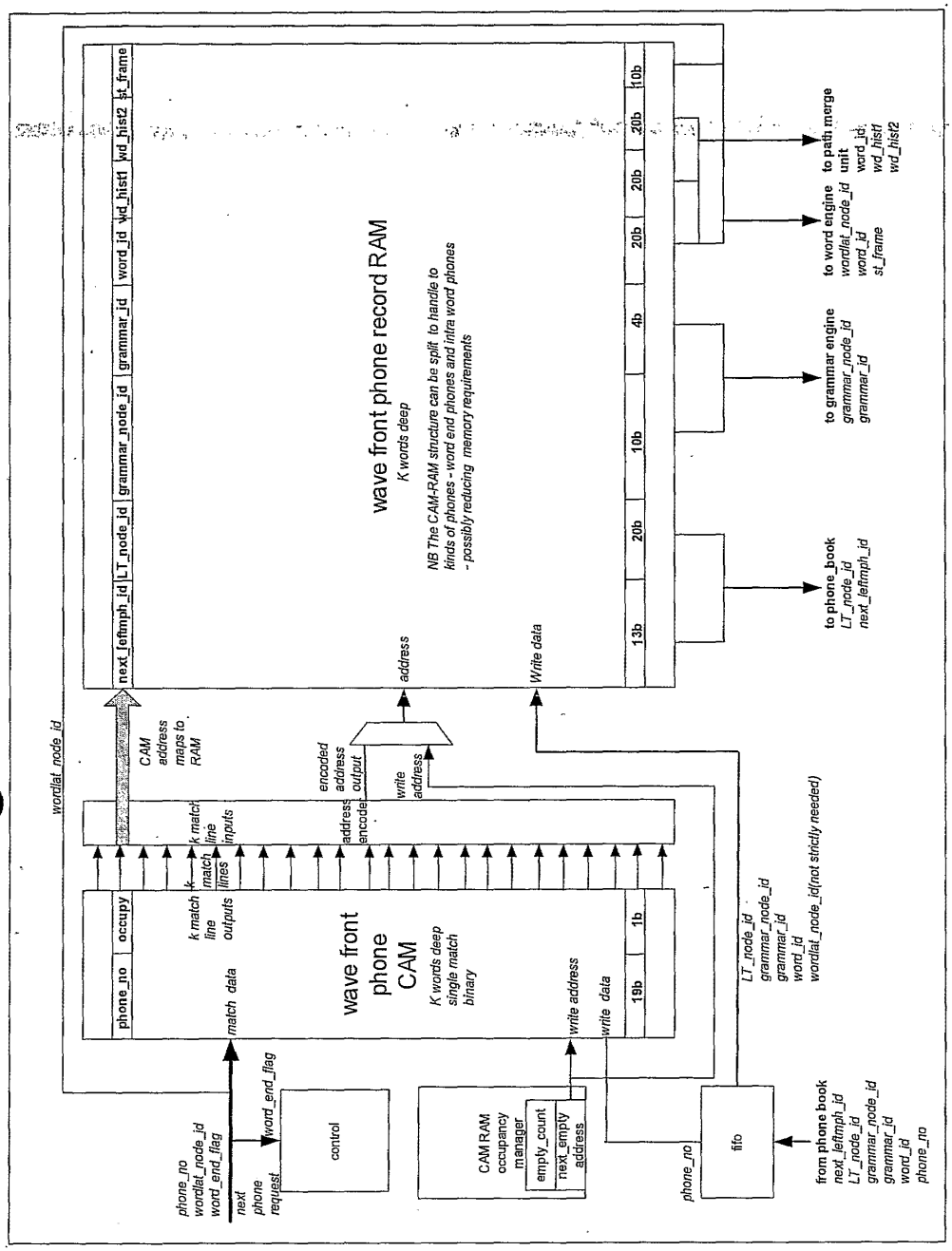

grammar level
grammar models
23
which
word_id = 238
grammar_node_id = 23
LT_id = 3
17
house
word_id = 45
grammar_node_id = 17
LT_id = 3
7
silence
word_id = 0
grammar_node_id = 7
LT_id = 4
grammar_id = 2
grammar_id = 2
25
neither
word_id = 3656
grammar_node_id = 25
LT_id = 3
9
silence
word_id = 0
grammar_node_id = 9
word level
word lattice
which (word_id = 238)
wordlat_node_id = 4
silence
(word_id = 0)
wordlat_node_id =6
neither(word_id = 3656)
wordlat_node_id = 5
wave front
phone CAM
phone_no
45
567
345
378
499
address
wave front
phone
record RAM
grammar_id grammar_node_id LT_id LT_node_id cent mph_id
2
23
3
56
h
lexical trees
LT_id
4
sil
3
"witch"
"which"
"house"
"house"
"neither"
"neither"
"either"
"either"
phone
level
next triphones: h -aU+s
h -aU+ Z
phone instance CAM
phone_no st_id1 st_id2 st_id3 back front empty end
22
237
345 3456 456 7112 0 1 0 0
320
indicates phone is
on wavefront
points to previous
phone instance
phone instance RAM
st_1 st_2 st_3 prev_phone_no wordlat_node_id trans lm_factor
token scores
-345.25 0 -377.56 320
4
trans lm_factor
State Level
PINE k

Figure 2



Amazon / Zentian Limited
Exhibit 1002
Page 357

Figure 4


wordlat_node_id
CAM address maps to RAM
next_leftmph_id
LT_node_id
grammar_node_id
grammar_id
word_id
wd_hist1
wd_hist2
st_frame
wave front phone record RAM
K words deep
NB The CAM-RAM structure can be split to handle to kinds of phones - word end phones and intra word phones - possibly reducing memory requirements
address
Write data
13b
20b
10b
4b
20b
20b
20b
10b
to phone_book
LT_node_id
next_leftmph_id
to grammar engine
grammar_node_id
grammar_id
to word engine
wordlat_node_id
word_id
st_frame
to path merge unit
word_id
wd_hist1
wd_hist2
encoded address output
address encoder
write address
k match line inputs
k match line lines
k match line outputs
phone_no
occupy
match data
wave front phone CAM
K words deep single match binary
write address
write data
19b
1b
LT_node_id
grammar_node_id
grammar_id
word_id
wordlat_node_id(not strictly needed)
phone_no wordlat_node_id word_end_flag
next phone request
word_end_flag
control
CAM RAM occupancy manager
empty_count
next_empty address
phone_no
fifo
from phone book
next_leftmph_id
LT_node_id
grammar_node_id
grammar_id
word_id
phone_no

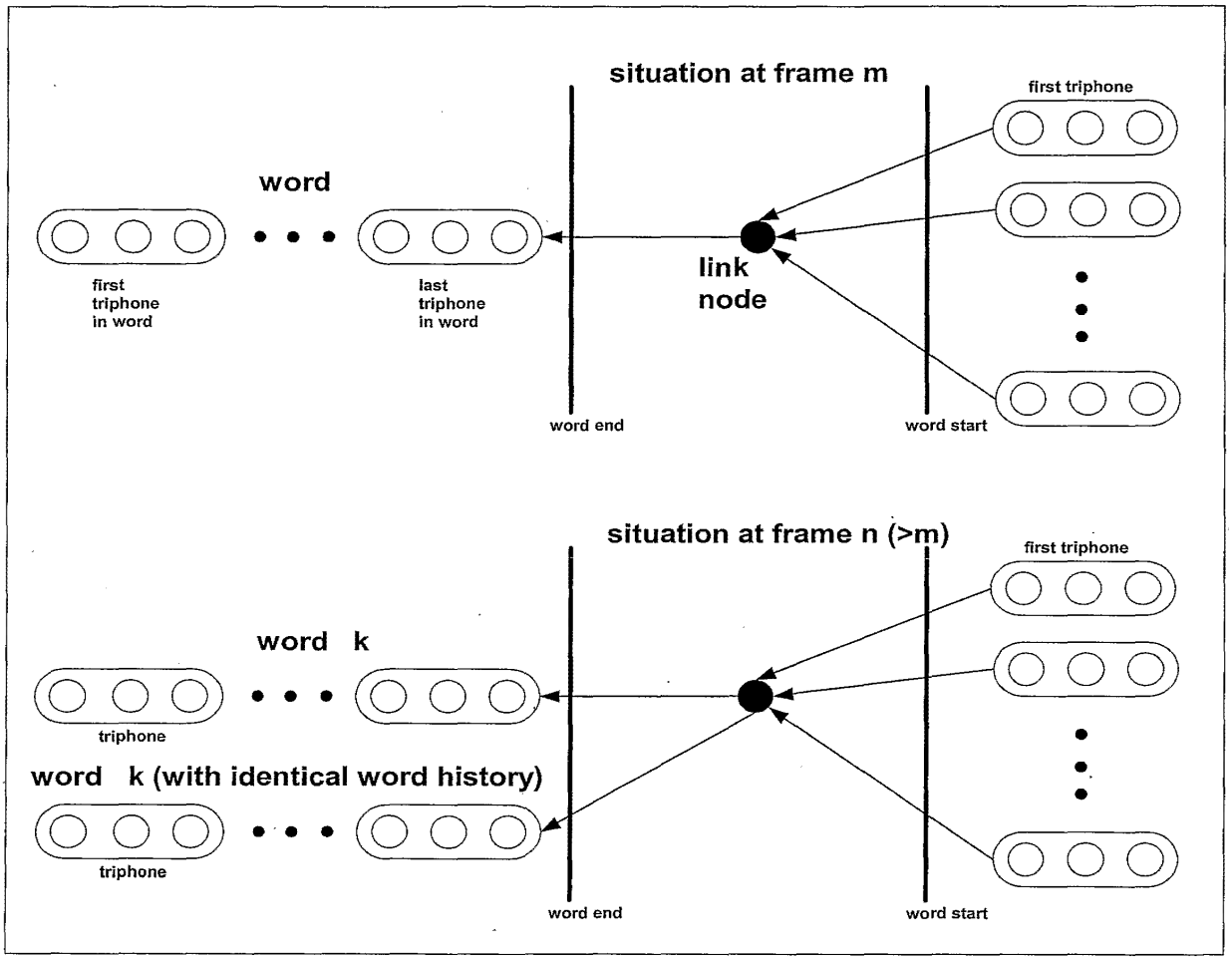

situation at frame m
first triphone
word
first triphone in word
last triphone in word
link node
word end
word start
situation at frame n (>m)
first triphone
word k
triphone
word k (with identical word history)
triphone
word end
word start

Figure 5

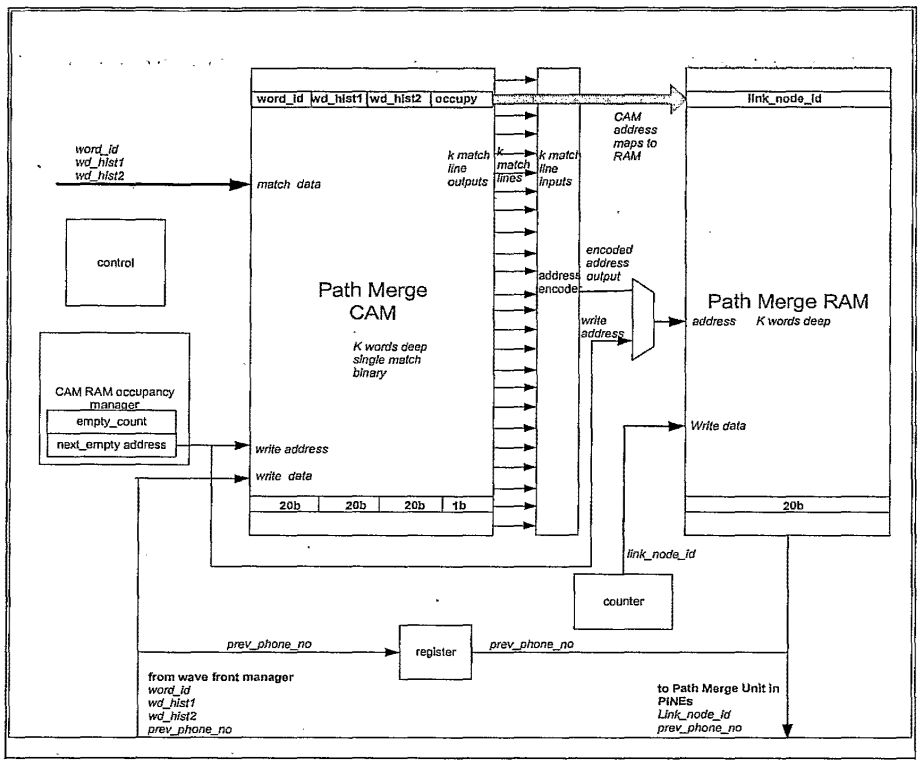
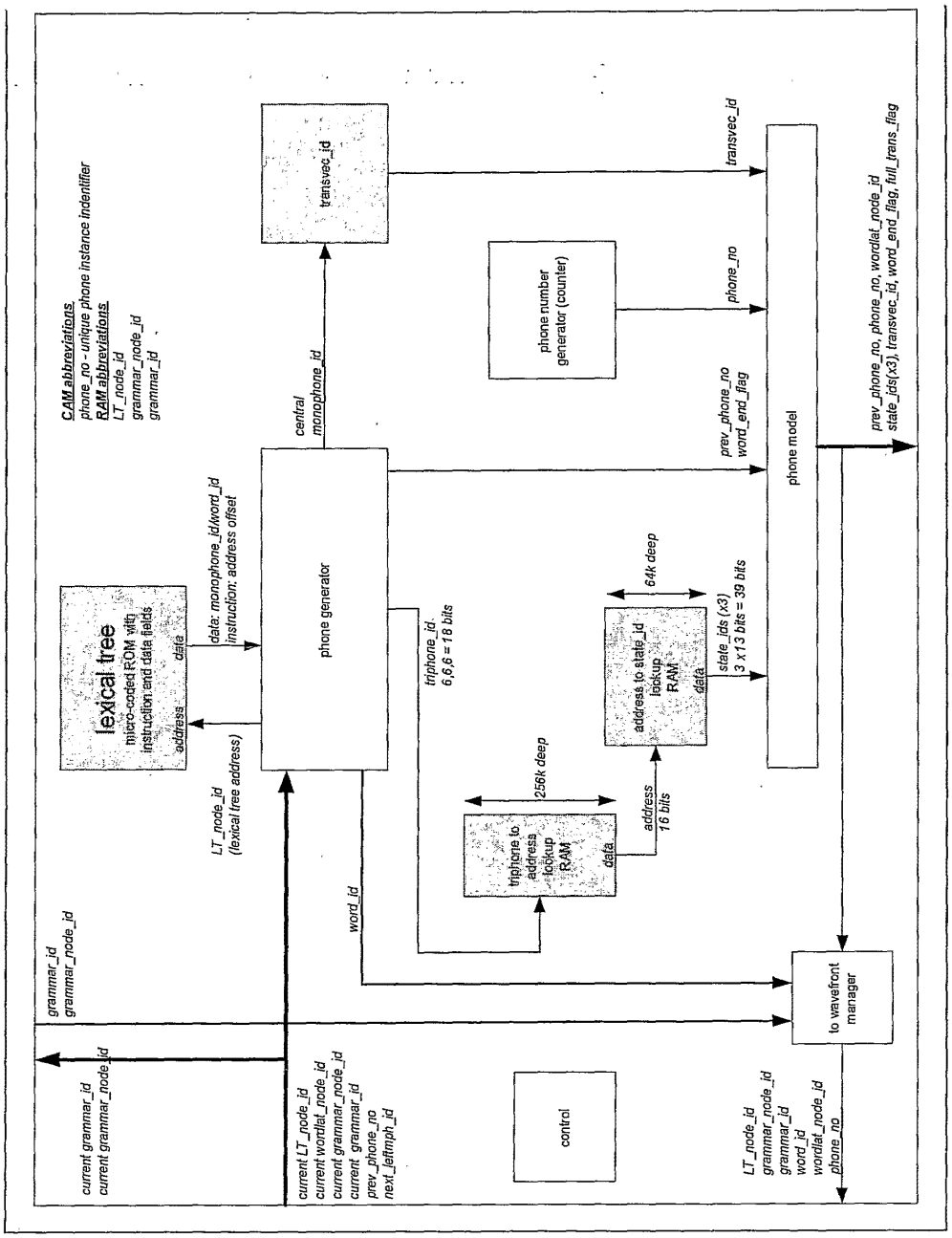
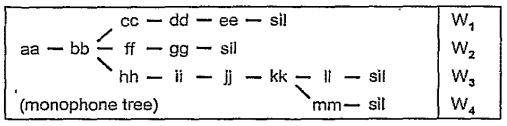

word_id
wd_hist1
wd_hist2
occupy
link_node_id
CAM address maps to RAM
word_id
wd_hist1
wd_hist2
match data
k match line outputs
k match lines
k match line inputs
control
Path Merge CAM
K words deep
single match
binary
encoded address output
address encode
write address
Path Merge RAM
address
K words deep
CAM RAM occupancy manager
empty_count
next_empty address
write address
write data
Write data
20b
20b
20b
1b
20b
link_node_id
counter
prev_phone_no
register
prev_phone_no
from wave front manager
word_id
wd_hist1
wd_hist2
prev_phone_no
to Path Merge Unit in PINEs
Link_node_id
prev_phone_no

Figure 6

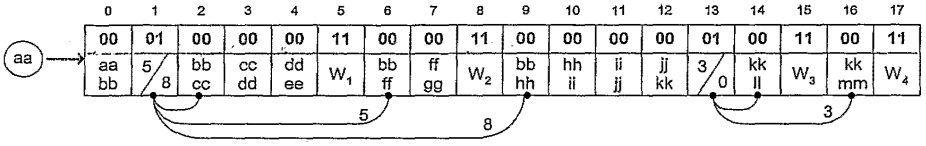
Figure 7


CAM abbreviations
phone_no - unique phone instance indentifier
RAM abbreviations
LT_node_id
grammar_node_id
grammar_id
lexical tree
micro-coded ROM with instruction and data fields
address
data
data: monophone_id/word_id
instruction: address offset
LT_node_id
(lexical tree address)
phone generator
central monophone_id
transvec_id
phone number generator (counter)
phone_no
prev_phone_no
word_end_flag
phone model
transvec_id
prev_phone_no, phone_no, wordlat_node_id
state_ids(x3), transvec_id, word_end_flag, full_trans_flag
triphone_id:
6,6,6 = 18 bits
256k deep
triphone to address lookup RAM
data
address
16 bits
64k deep
address to state_id lookup RAM
data
state_ids (x3)
3 x 13 bits = 39 bits
word_id
grammar_id
grammar_node_id
current grammar_id
current grammar_node_id
current LT_node_id
current wordlat_node_id
current grammar_node_id
current grammar_id
prev_phone_no
next_leftmph_id
control
to wavefront manager
LT_node_id
grammar_node_id
grammar_id
word_id
wordlat_node_id
phone_no

Lexical Tree RAM



00 - monophone and unigram weight.
01 - branch (up to 3 exit vertices).
10 - Mbranch (more than 3 exit vertices).
11 - word end.



"00" - Single branch, ie. mono-phone to mono-phone.

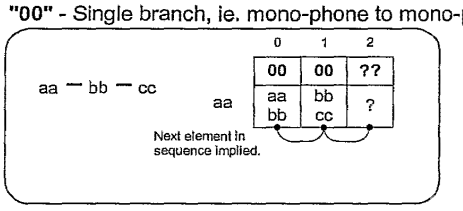
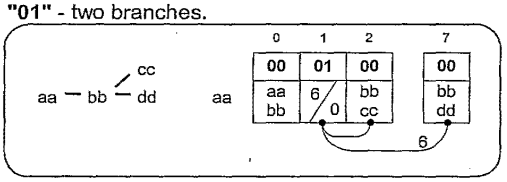


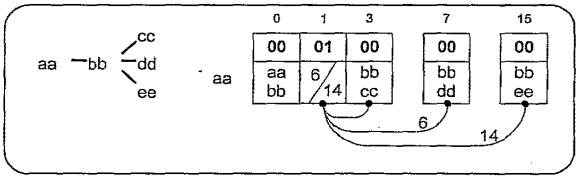
Figure 8

"01" - two branches.



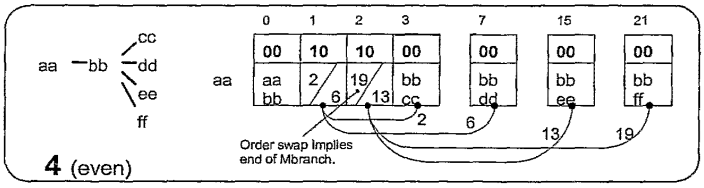
The "01" branch is a single element instruction (at addr:1) such that the first branch (here to "cc") is implied at addr+1. The second branch (to "dd") is given as relative addr:6. The third available branch is not used and is indicated by "0".

"01" - three branches.



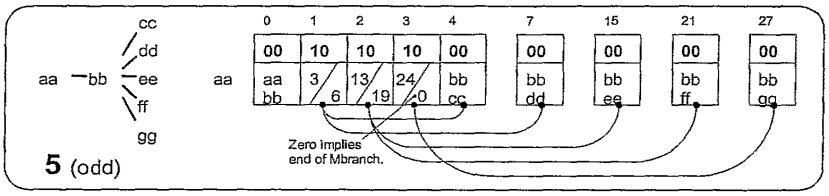
The "01" branch is a single element instruction (at addr:1) such that the first branch (here to "cc") is implied at addr+1. The second branch (to "dd") is given as relative addr:6. The third (and final branch in the "01" instruction) branch is given as relative addr:14.

"10" - Mbranch: 4 (even)



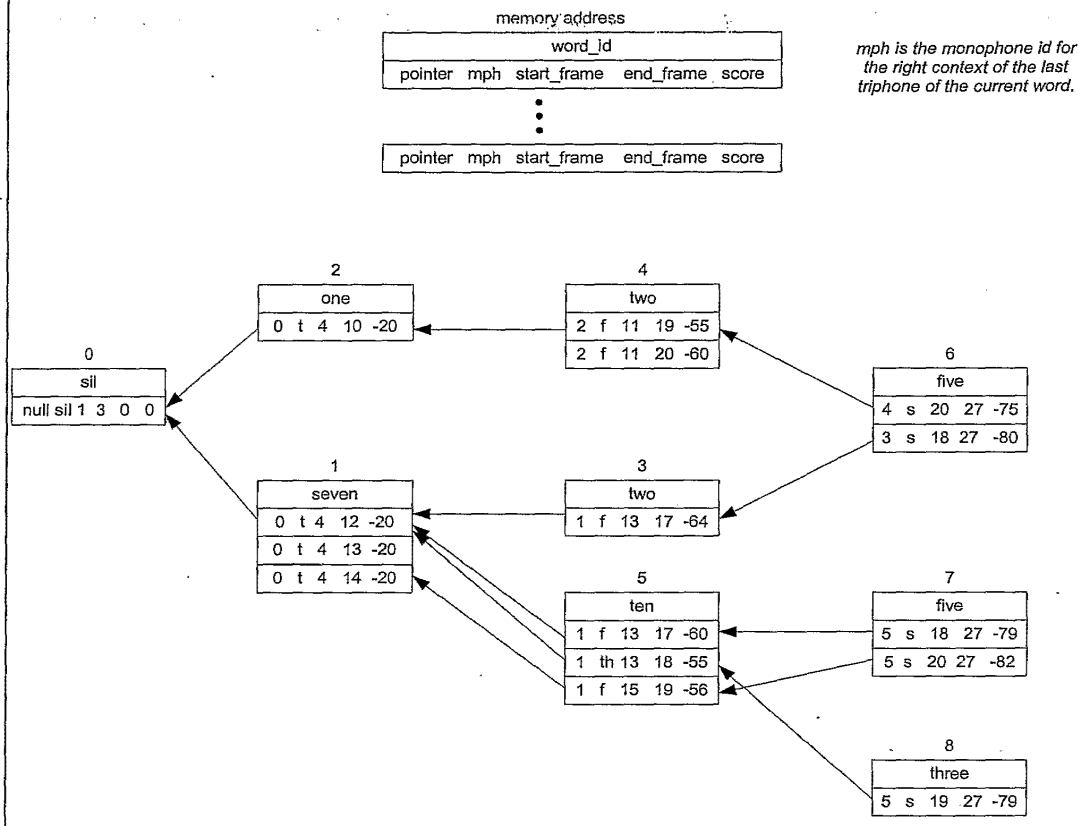
The Mbranch can not assume that the first branch is at addr+1, so it is given ("cc" at rel addr:2). And the second, in this example, at addr:6. The Mbranch instruction continues at addr+1 (addr:2), and gives rel branch addr 19 and 13. The (reverse) order of the given addresses is a code to indicate that this is the end of the Mbranch.

"10" - Mbranch: 5 (odd)



The Mbranch directs the first branch to addr:4. The final branch is to addr:27, and the other available branch instruction, "0" implies that the Mbranch instruction is complete.

data structure for word link record to enable word n-best decoding



Notes

---

1. each record represents the passage of a single token through a word with a particular right monophone context.
2. each word's acoustic model with a particular right context monophone can only generate one token per frame. Thus frame number can be used to distinguish trace back paths with the same end_frame numbers but different right context monophones
3. the end_frame doesn't need to be passed to the PINEs because it is simply the current frame number
4. scores may be absolute or relative
5. the address is referred to as the wordlat_node_id elsewhere

Traceback

---

for the n-best tokens

1. using the associated wordlat_node_ids look up each word link record
2. look at associated records and use the end_frame number and monophone id to select associated pointer and score.
3. trace back recursively and select n-best paths

Figure 9

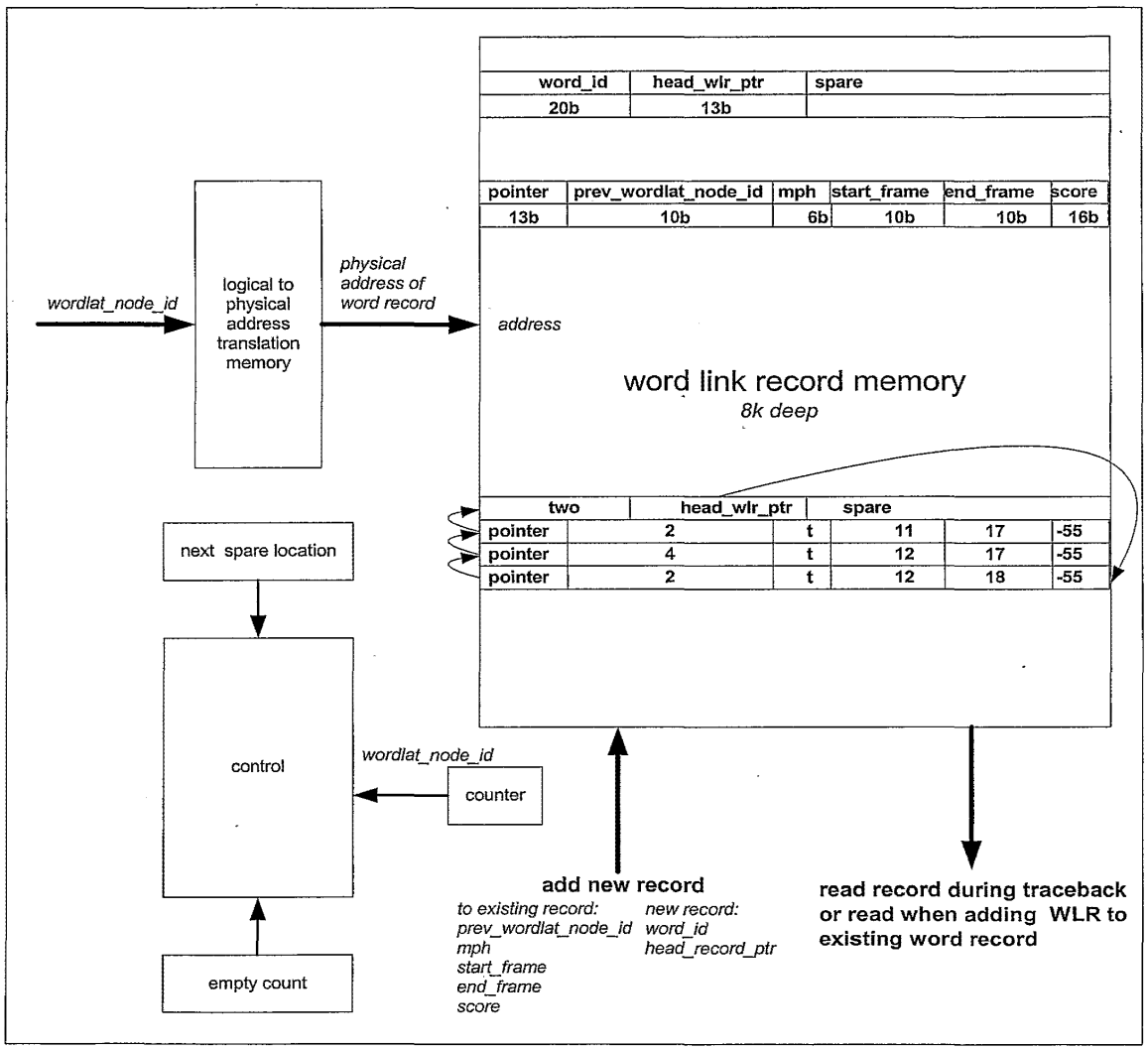

word_id
head_wlr_ptr
spare
20b
13b
pointer
prev_wordlat_node_id
mph
start_frame
end_frame
score
13b
10b
6b
10b
10b
16b
wordlat_node_id
logical to physical address translation memory
physical address of word record
address
word link record memory
8k deep
two
head_wlr_ptr
spare
pointer 2 t 11 17 -55
pointer 4 t 12 17 -55
pointer 2 t 12 18 -55
next spare location
control
wordlat_node_id
counter
empty count
add new record
to existing record:
prev_wordlat_node_id
mph
start_frame
end_frame
score
new record:
word_id
head_record_ptr
read record during traceback or read when adding WLR to existing word record

Figure 10

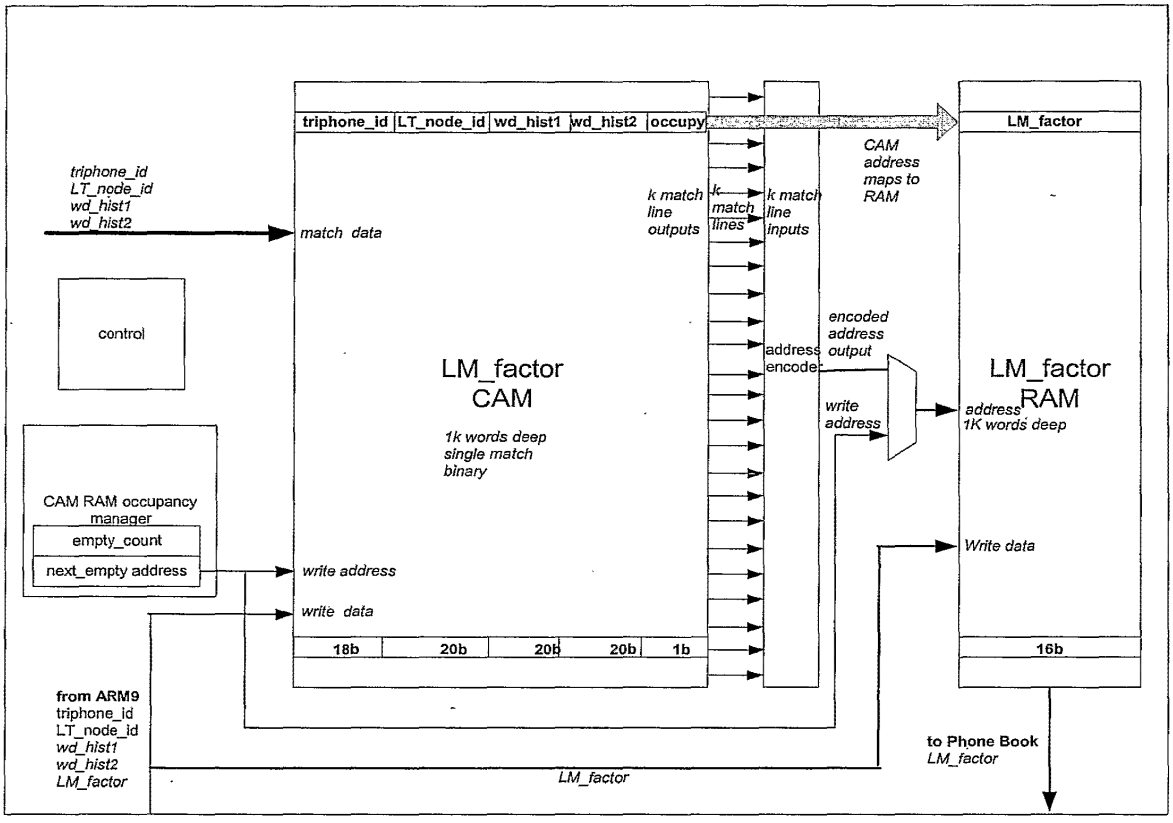

triphone_id
LT_node_id
wd_hist1
wd_hist2
occupy
LM_factor
CAM address maps to RAM
triphone_id
LT_node_id
wd_hist1
wd_hist2
match data
k match line outputs
k match lines
k match line inputs
control
encoded address output
address encoder
LM_factor CAM
1k words deep
single match
binary
write address
LM_factor RAM
1K words deep
CAM RAM occupancy manager
empty_count
next_empty address
write address
write data
Write data
18b
20b
20b
20b
1b
16b
from ARM9
triphone_id
LT_node_id
wd_hist1
wd_hist2
LM_factor
LM_factor
to Phone Book
LM_factor

Figure 11

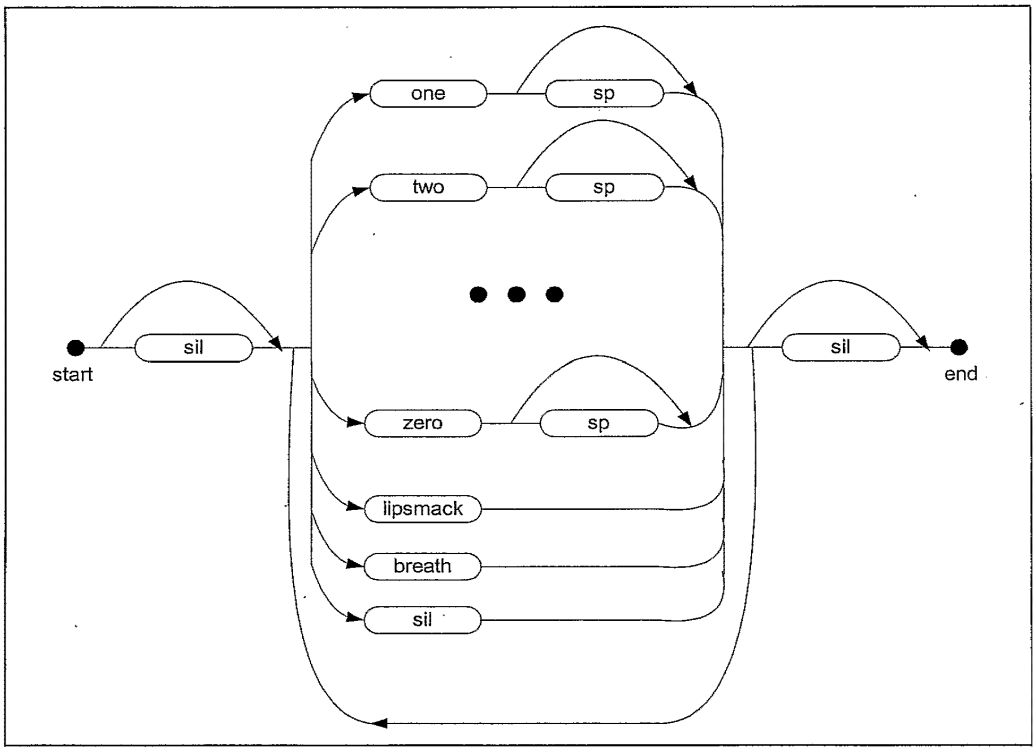


Figure 12

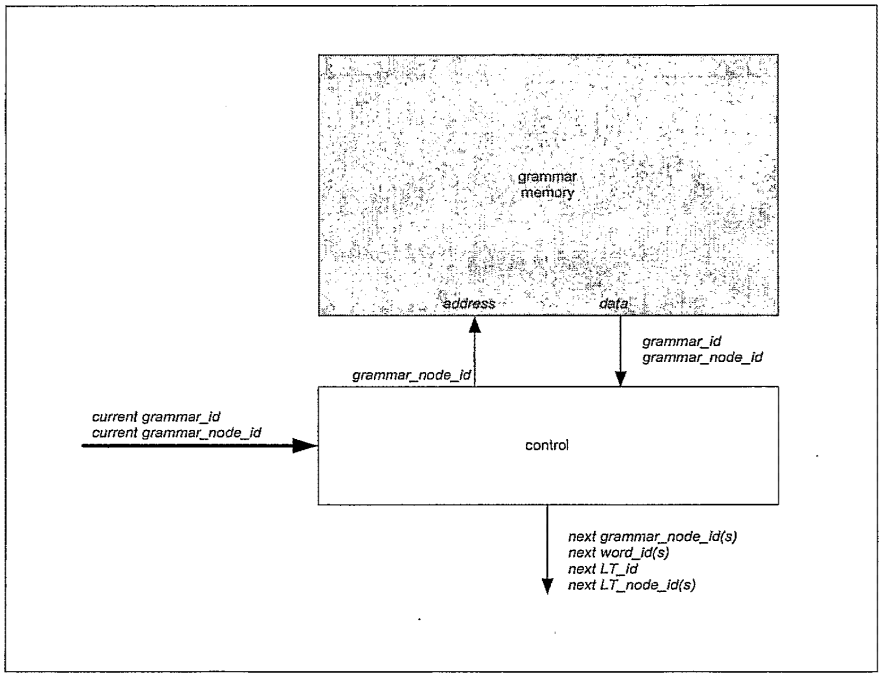

grammar
memory
address
data
grammar_id
grammar_node_id
grammar_node_id
current grammar_id
current grammar_node_id
control
next grammar_node_id(s)
next word_id(s)
next LT_id
next LT_node_id(s)

Figure 13

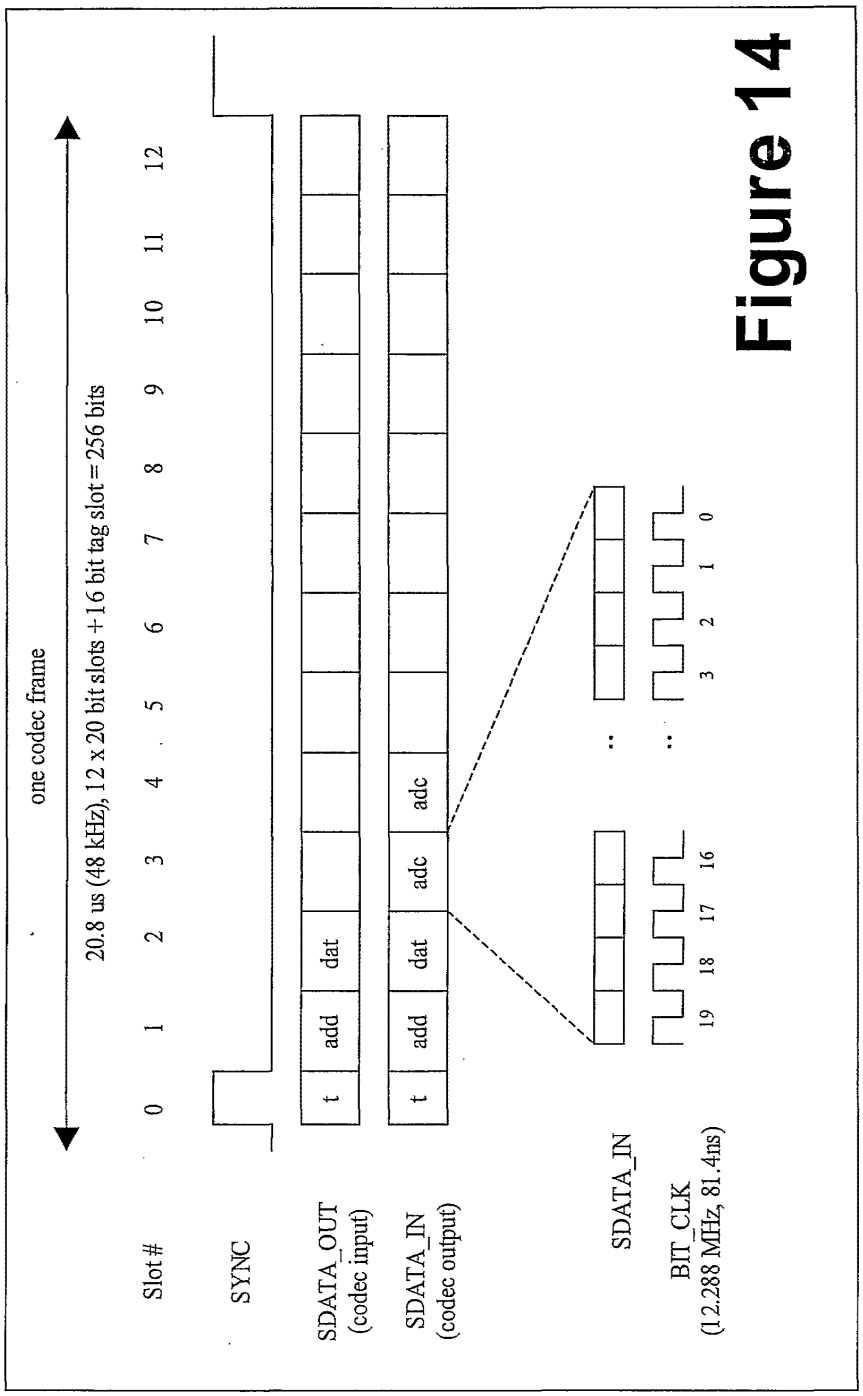

one codec frame
20.8 us (48 kHz), 12 x 20 bit slots + 16 bit tag slot = 256 bits
Slot #
0 1 2 3 4 5 6 7 8 9 10 11 12
SYNC
SDATA_OUT (codec input)
t add dat
SDATA_IN (codec output)
t add dat adc adc
SDATA_IN
BIT_CLK (12.288 MHz, 81.4ns)
19 18 17 16 3 2 1 0

Figure 14

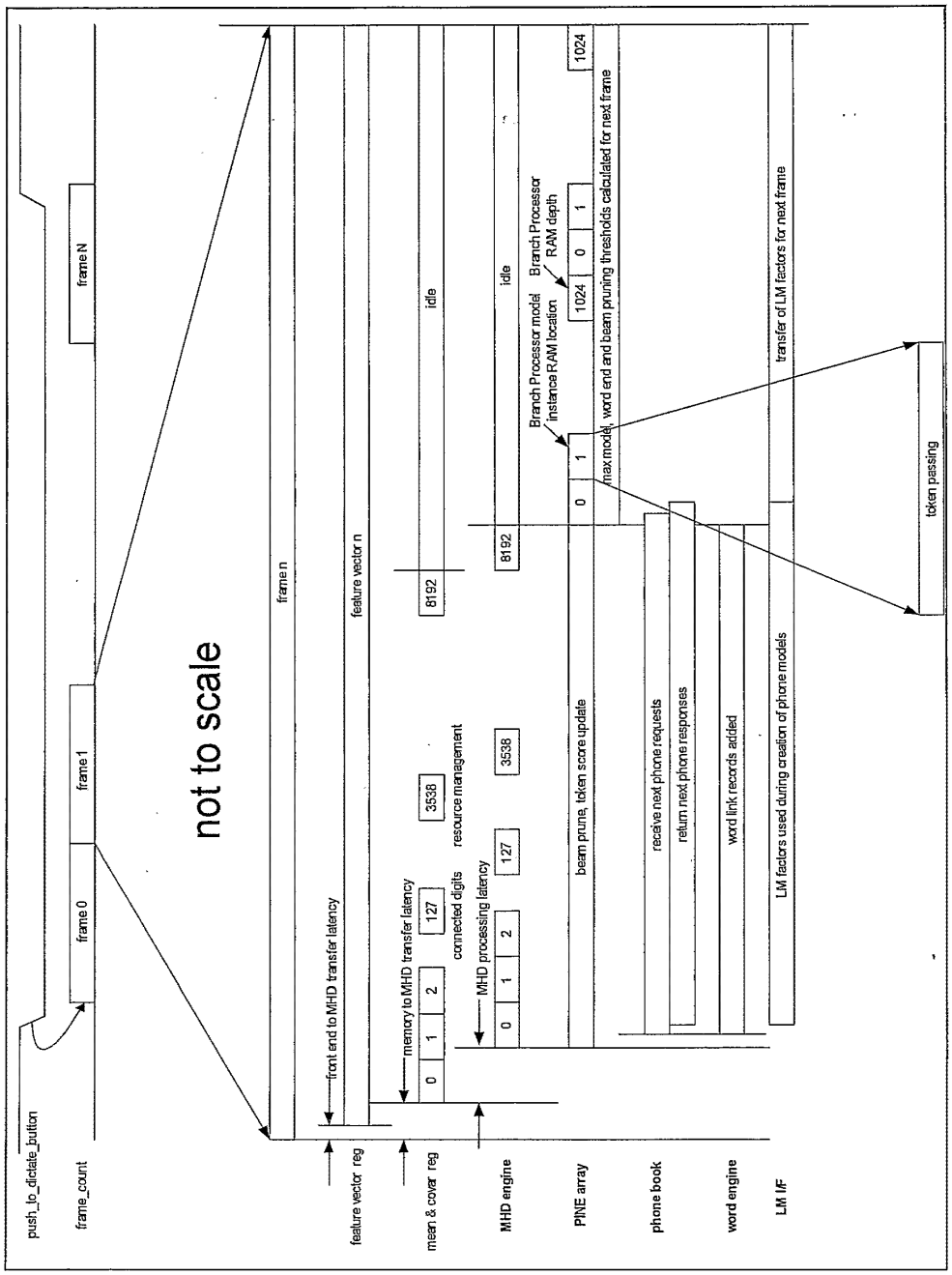

not to scale
push_to_dictate_button
frame_count
frame 0
frame 1
frame N
frame n
feature vector reg
front end to MHD transfer latency
feature vector n
mean & cover reg
memory to MHD transfer latency
0
1
2
127
3538
resource management
connected digits
MHD engine
MHD processing latency
0
1
2
127
3538
8192
idle
PINE array
beam prune, token score update
0
1
8192
idle
Branch Processor model instance RAM location
Branch Processor RAM depth
1024
0
1
1024
max model, word end and beam pruning thresholds calculated for next frame
phone book
receive next phone requests
return next phone responses
word engine
word link records added
LM I/F
LM factors used during creation of phone models
transfer of LM factors for next frame
token passing

Figure 15

84/27

U.S. Appl. No. 11/662704 International Appl No. GB05/003554

Application filed by: ☐ 20 months ☑ 30 months

INTERNATIONAL APPLICATION PAPERS IN THE APPLICATION FILE:

- ☑ International application (RECORD COPY)
- ☐ Article 19 amendments
- ☐ PCT/IB/331
- ☐ PCT/IPEA/409 IPER (PCT/IPEA/416 on front)
- ☐ Annexes to 409
- ☑ Priority document(s) No. 1
- ☐ INTERNATIONAL APPLICATION ON DOUBLE SIDED PAPER (COPIES MADE)
- ☐ Request form PCT/RO/101
- ☐ PCT/IB/302
- ☑ PCT/ISA/210-Search Report
- ☐ Search Report references
- ☐ Other _____

RECEIPTS FROM THE APPLICANT: (other than checked above)

- ☑ Basic National Fee (paid or authorized to charge)
- Translation of international application as filed:
  - ☑ Description
  - ☑ Claims
  - ☑ Words in the drawing figure(s)
  - ☐ Article 19 amendments
  - ☐ Annexes to 409
- ☑ Oath / Declaration
- ☐ DNA diskette
- ☑ Preliminary amendment(s) filed _____
- ☑ Information Disclosure Statement
- ☑ Assignment document
- ☐ Power of attorney/Change of address
- ☐ Substitute specification
- ☐ Verified small status claim
- ☐ Other _____

Notes: Use IA from IB

35 U.S.C. 371 - Receipt of Request (PTO-1390)	14 Mar '07
Date acceptable oath / declaration received	"
Date complete 35 U.S.C 371 requirements met	"
102(e) Date	"
Date of completion of DO/EO 906 - Notification of Missing 102(e) Requirements	
Date of completion of DO/EO 907 - Notification of Acceptance for 102(e) date	
Date of completion of DO/EO 911 - Application accepted under 35 U.S.C. 1.11	
Date of completion of DO/EO 905 - Notification of Missing Requirements	
Date of completion of DO/EO 916 - Notification of Defective Response	
Date of completion of DO/EO 903 - Notification of Acceptance	26 Jun '08
Date of completion of DO/EO 909 - Notification of Abandonment	

WIPO Publication
Publication No.
WO06/030214 A2

Publication Date
23 Mar '06

Publication Language
English

Not Published
☐ U.S. only
Designated
☑ EP request

Screening done by: SA

S = 59
C = 17
A = 61
D = 27
T = 104

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 March 2006 (23.03.2006)

PCT

(10) International Publication Number
WO 2006/030214 A2

(51) International Patent Classification:
G10L 15/28 (2006.01) *G10L 15/10* (2006.01)

(21) International Application Number:
PCT/GB2005/003554

(22) International Filing Date:
14 September 2005 (14.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0420464.0 14 September 2004 (14.09.2004) GB

(71) Applicant *(for all designated States except US)*: **ZENTIAN LIMITED** [GB/GB]; St. John's Innovation Centre, Cowley Road, Cambridge, Cambridgeshire CB4 0WS (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CATCHPOLE, Mark** [—/GB]; Ash House, Breckenwood Road, Fulbourn, Cambridge, Cambridgeshire CB1 5DQ (GB). **DOWSETT, Damian, Harris**; * (**). **REYNOLDS, Timothy**; * (**). **LARRY, Guy** [—/GB]; 1 Cutter Ferry Lane, Cambridge, Cambridgeshire CB4 1JR (GB).

(74) Agent: **COLLINS, John, David**; Marks & Clerk, 90 Long Acre, London WC2E 9RA (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A SPEECH RECOGNITION CIRCUIT AND METHOD

(57) Abstract: A speech recognition circuit comprising a circuit for providing state identifiers which identify states corresponding to nodes or groups of adjacent nodes in a lexical tree, and for providing scores corresponding to said state identifiers, the lexical tree comprising a model of words; a memory structure for receiving and storing state identifiers identified by a node identifier identifying a node or group of adjacent nodes, said memory structure being adapted to allow lookup to identify particular state identifiers, reading of the scores corresponding to the state identifiers, and writing back of the scores to the memory structure after modification of the scores; an accumulator for receiving score updates corresponding to particular state identifiers from a score update generating circuit which generates the score updates using audio input, for receiving scores from the memory structure, and for modifying said scores by adding said score updates to said scores; and a selector circuit for selecting at least one node or group of adjacent nodes of the lexical tree according to said scores.

WO 2006/030214 A2

PATENT APPLICATION FEE DETERMINATION RECORD

Effective December 8, 2004

Application or Docket Number: 11/662704

CLAIMS AS FILED - PART I

SMALL ENTITY TYPE ☐ OR OTHER THAN SMALL ENTITY

	(Column 1)	(Column 2)	SMALL ENTITY RATE	SMALL ENTITY FEE		OTHER THAN SMALL ENTITY RATE	OTHER THAN SMALL ENTITY FEE
U.S. NATIONAL STAGE FEES							
BASIC FEE			BASIC FEE	150	OR	BASIC FEE	
EXAMINATION FEE			EXAM. FEE	100		EXAM. FEE	
SEARCH FEE			SEARCH FEE	200		SEARCH FEE	
FEE FOR EXTRA SPEC. PGS.	minus 100 =	/ 50 = 1	X $ 125 =	125		X $ 250 =	
TOTAL CHARGEABLE CLAIMS	39 minus 20 =	* 19	X $ 25 =	475	OR	X $ 50 =	
INDEPENDENT CLAIMS	6 minus 3 =	* 3	X $ 100 =	300	OR	X $ 200 =	
MULTIPLE DEPENDENT CLAIM PRESENT		☐	+ $ 180 =		OR	+ $ 360 =	
* If the difference in column 1 is less than zero, enter "0" in column 2			TOTAL	1350	OR	TOTAL	

CLAIMS AS AMENDED - PART II

SMALL ENTITY OR OTHER THAN SMALL ENTITY

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE	SMALL ENTITY ADDITIONAL FEE		OTHER THAN SMALL ENTITY RATE	OTHER THAN SMALL ENTITY ADDITIONAL FEE
AMENDMENT A	Total	*	Minus	**	=	X $ 25 =		OR	X $ 50 =	
	Independent	*	Minus	***	=	X $ 100 =		OR	X $ 200 =	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM ☐					+ $ 180 =		OR	+ $ 360 =	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE	SMALL ENTITY ADDITIONAL FEE		OTHER THAN SMALL ENTITY RATE	OTHER THAN SMALL ENTITY ADDITIONAL FEE
AMENDMENT B	Total	*	Minus	**	=	X $ 25 =		OR	X $ 50 =	
	Independent	*	Minus	***	=	X $ 100 =		OR	X $ 200 =	
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM ☐					+ $ 180 =		OR	+ $ 360 =	
						TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than '20', enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than '3', enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

MULTIPLE DEPENDENT CLAIM FEE CALCULATION SHEET

(FOR USE WITH FORM PTO-875)

SERIAL NO. 11/662704

FILING DATE

APPLICANT(S)

CLAIMS

	AS FILED		AFTER 1st AMENDMENT		AFTER 2nd AMENDMENT	
	IND.	DEP.	IND.	DEP.	IND.	DEP.
1	1		1			2
2		1		1		
3		1		1		
4		1		1		
5	1		1	1		
6		1				
7		2				
8		(1)				
9	1					
10	1		1			
11		1		1		
12		1		1		
13		3		1		
14		3		1		
15		(1)		1		
16		(1)		1		
17		(1)		1		
18		(1)		1		
19	1		1			
20		1		1		
21		2		1		
22		(1)		1		
23		(1)		1		
24	1					
25	1					
26	1		1			
27		1	C	C		
28		2		(1)		
29		2		1		
30		(1)		1		
31		(1)	C	C		
32		(1)		1		
33		(1)		1		
34		(1)		1		
35		(1)		1		
36		(1)		1		
37		(1)		1		
38		(1)		1		
39		(1)		1		
40		(1)		1		
41		(1)	C	C		
42		(1)		1		
43	1		1			
44		(1)		1		
45	1		1			
46		(1)		1		
47	1					
48		1				
49		2				
50		2				
TOTAL IND.		⬇		⬇		⬇
TOTAL DEP.		⬅		⬅		⬅
TOTAL CLAIMS						

	AS FILED		AFTER 1st AMENDMENT		AFTER 2nd AMENDMENT	
	IND.	DEP.	IND.	DEP.	IND.	DEP.
51		(1)				
52		(1)				
53		(1)				
54		(1)				
55		(1)				
56		(1)				
57		(1)				
58		(1)				
59		(1)				
60		(1)				
61		(1)				
62	1					
63		1				
64		2				
65		2				
66		(1)				
67		(1)				
68		(1)				
69		(1)				
70		(1)				
71		(1)				
72		(1)				
73		(1)				
74		(1)				
75		(1)				
76		(1)				
77		(1)				
78		(1)				
79	1					
80	1					
81	1					
82		(1)				
83	1					
84	1					
85				1		
86				1		
87						
88						
89						
90						
91						
92						
93						
94						
95						
96						
97						
98						
99						
100						
TOTAL IND.	17	⬇	6	⬇		⬇
TOTAL DEP.	79	⬅	33	⬅		⬅
TOTAL CLAIMS	96		39			

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/662,704	03/14/2007	2626	1350	M0025.0369/P369	39	6

CONFIRMATION NO. 2632

24998
DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

FILING RECEIPT



Date Mailed: 07/10/2008

Receipt is acknowledged of this non-provisional patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Applicant(s)

Guy Larri, Cambridge, UNITED KINGDOM;
Mark Catchpole, Cambridge, UNITED KINGDOM;
Damian Kelly Harris-Dowsett, Coventry, UNITED KINGDOM;
Timothy Brian Reynolds, Cambridge, UNITED KINGDOM;

Assignment For Published Patent Application

ZENTIAN LIMITED, Cambridge, UNITED KINGDOM

Power of Attorney: The patent practitioners associated with Customer Number 24998

Domestic Priority data as claimed by applicant

This application is a 371 of PCT/GB05/03554 09/14/2005

Foreign Applications

UNITED KINGDOM 0420464.0 09/14/2004

If Required, Foreign Filing License Granted: 06/26/2008

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 11/662,704**

Projected Publication Date: 10/16/2008

Non-Publication Request: No

Early Publication Request: No
**** SMALL ENTITY ****

Amazon / Zentian Limited
Exhibit 1002
Page 387

Title

Speech Recognition Circuit and Method

Preliminary Class

704

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as

set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

U.S. APPLICATION NUMBER NO.	FIRST NAMED APPLICANT	ATTY. DOCKET NO.
11/662,704	Guy Larri	M0025.0369/P369

24998
DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

INTERNATIONAL APPLICATION NO.	
PCT/GB05/03554	
I.A. FILING DATE	PRIORITY DATE
09/14/2005	09/14/2004

CONFIRMATION NO. 2632
371 ACCEPTANCE LETTER

OC000000030706569

Date Mailed: 07/10/2008

NOTICE OF ACCEPTANCE OF APPLICATION UNDER 35 U.S.C 371 AND 37 CFR 1.495

The applicant is hereby advised that the United States Patent and Trademark Office in its capacity as a Designated / Elected Office (37 CFR 1.495), has determined that the above identified international application has met the requirements of 35 U.S.C. 371, and is ACCEPTED for national patentability examination in the United States Patent and Trademark Office.

The United States Application Number assigned to the application is shown above and the relevant dates are:

03/14/2007	03/14/2007
DATE OF RECEIPT OF 35 U.S.C. 371(c)(1), (c)(2) and (c)(4) REQUIREMENTS	DATE OF COMPLETION OF ALL 35 U.S.C. 371 REQUIREMENTS

A Filing Receipt (PTO-103X) will be issued for the present application in due course. **THE DATE APPEARING ON THE FILING RECEIPT AS THE " FILING DATE" IS THE DATE ON WHICH THE LAST OF THE 35 U.S.C. 371 (c)(1), (c)(2) and (c)(4) REQUIREMENTS HAS BEEN RECEIVED IN THE OFFICE. THIS DATE IS SHOWN ABOVE.** *The filing date of the above identified application is the international filing date of the international application (Article 11(3) and 35 U.S.C. 363).* Once the Filing Receipt has been received, send all correspondence to the Group Art Unit designated thereon.

The following items have been received:

- Indication of Small Entity Status
- Copy of the International Application filed on 03/14/2007
- Copy of the International Search Report filed on 03/14/2007
- Preliminary Amendments filed on 03/14/2007
- Information Disclosure Statements filed on 03/14/2007
- Oath or Declaration filed on 03/14/2007
- Request for Immediate Examination filed on 03/14/2007
- U.S. Basic National Fees filed on 03/14/2007
- Assignment filed on 03/14/2007
- Priority Documents filed on 03/14/2007
- Power of Attorney filed on 03/14/2007
- Specification filed on 03/14/2007
- Claims filed on 03/14/2007
- Abstracts filed on 03/14/2007
- Drawings filed on 03/14/2007

FORM PCT/DO/EO/903 (371 Acceptance Notice)

Applicant is reminded that any communications to the United States Patent and Trademark Office must be mailed to the address given in the heading and include the U.S. application no. shown above (37 CFR 1.5)

SHAKEEL AHMED

Telephone: (703) 308-9140 EXT 208

FORM PCT/DO/EO/903 (371 Acceptance Notice)

Amazon / Zentian Limited
Exhibit 1002
Page 391

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369

CONFIRMATION NO. 2632

24998
DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

PUBLICATION NOTICE

OC000000032625326

Title: Speech Recognition Circuit and Method

Publication No.US-2008-0255839-A1
Publication Date:10/16/2008

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Office of Data Managment, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Amazon / Zentian Limited
Exhibit 1002
Page 392

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

24998 7590 05/06/2010

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

EXAMINER	
MCFADDEN, SUSAN IRIS	
ART UNIT	PAPER NUMBER
2626	

MAIL DATE	DELIVERY MODE
05/06/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/662,704	LARRI ET AL.	
	Examiner	Art Unit	
	Susan McFadden	2626	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 1 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 March 2007*.

2a)☐ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,10-23,26,28-30,32-40,42-46,85 and 86* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☒ Claim(s) *1-5,10-23,26,28-30,32-40,42-46,85 and 86* are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 March 2007* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Election/Restrictions

1. Restriction is required under 35 U.S.C. 121 and 372.

This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1.

In accordance with 37 CFR 1.499, applicant is required, in reply to this action, to elect a single invention to which the claims must be restricted.

Group 1 , claim(s) 1-5,36-40,42-46, and 86, drawn to speech recognition.

Group 2, claim(s) 10-18 and 85, drawn to calculating distances.

Group 3, claim(s) 19-23,26,28-30, and 32-35, drawn to bandwidth compression.

2. The inventions listed as Groups 1-3 do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons: the speech recognition feature of Group 1 is not necessary for Group 3.

3. A telephone call was made to Stephen Soffen on April 26, 2010 to request an oral election to the above restriction requirement, but did not result in an election being made.

Applicant is advised that the reply to this requirement to be complete must include (i) an election of a species or invention to be examined even though the requirement may be traversed (37 CFR 1.143) and (ii) identification of the claims encompassing the elected invention.

The election of an invention or species may be made with or without traverse. To preserve a right to petition, the election must be made with traverse. If the reply does not distinctly and specifically point out supposed errors in the restriction requirement, the election shall be treated as an election without traverse.

4. Applicant is reminded that upon the cancellation of claims to a non-elected invention, the inventorship must be amended in compliance with 37 CFR 1.48(b) if one or more of the currently named inventors is no longer an inventor of at least one claim remaining in the application. Any amendment of inventorship must be accompanied by a request under 37 CFR 1.48(b) and by the fee required under 37 CFR 1.17(i).

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Susan McFadden whose telephone number is 571-272-7621. The examiner can normally be reached on Monday-Friday, 9:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Richemond Dorvil can be reached on 571-272-7602. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Susan McFadden/
Primary Examiner, Art Unit 2626
May 4, 2010

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 2632

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/662,704	03/14/2007 RULE	704	2626	M0025.0369/P369

APPLICANTS

Guy Larri, Cambridge, UNITED KINGDOM;
Mark Catchpole, Cambridge, UNITED KINGDOM;
Damian Kelly Harris-Dowsett, Coventry, UNITED KINGDOM;
Timothy Brian Reynolds, Cambridge, UNITED KINGDOM;

**** CONTINUING DATA ***************************

This application is a 371 of PCT/GB05/03554 09/14/2005

**** FOREIGN APPLICATIONS ***************************

UNITED KINGDOM 0420464.0 09/14/2004

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED **** ** SMALL ENTITY **

06/26/2008

Foreign Priority claimed	☑ Yes ☐ No	☐ Met after Allowance	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
35 USC 119(a-d) conditions met	☑ Yes ☐ No		UNITED KINGDOM	27	39	6
Verified and Acknowledged	/SUSAN IRIS MCFADDEN/ Examiner's Signature	Initials				

ADDRESS

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403
UNITED STATES

TITLE

Speech Recognition Circuit and Method

FILING FEE RECEIVED	FEES: Authority has been given in Paper	
1350	No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010								
	1	÷								
	2	÷								
	3	÷								
	4	÷								
	5	✓								
	6	-								
	7	-								
	8	-								
	9	-								
	10	÷								
	11	÷								
	12	÷								
	13	÷								
	14	÷								
	15	÷								
	16	÷								
	17	÷								
	18	÷								
	19	÷								
	20	÷								
	21	÷								
	22	÷								
	23	÷								
	24	-								
	25	-								
	26	÷								
	27	-								
	28	÷								
	29	÷								
	30	÷								
	31	-								
	32	÷								
	33	÷								
	34	÷								
	35	÷								
	36	÷								

Amazon / Zentian Limited
Exhibit 1002
Page 399

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010								
	37	÷								
	38	÷								
	39	÷								
	40	÷								
	41	-								
	42	÷								
	43	÷								
	44	÷								
	45	÷								
	46	÷								
	47	-								
	48	-								
	49	-								
	50	-								
	51	-								
	52	-								
	53	-								
	54	-								
	55	-								
	56	-								
	57	-								
	58	-								
	59	-								
	60	-								
	61	-								
	62	-								
	63	-								
	64	-								
	65	-								
	66	-								
	67	-								
	68	-								
	69	-								
	70	-								
	71	-								
	72	-								

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010								
	73	-								
	74	-								
	75	-								
	76	-								
	77	-								
	78	-								
	79	-								
	80	-								
	81	-								
	82	-								
	83	-								
	84	-								
	85	÷								
	86	÷								

Amazon / Zentian Limited
Exhibit 1002
Page 401

Docket No.: M0025.0369/P369
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Guy Larri et al.

Application No.: 11/662,704 | Confirmation No.: 2632

Filed: March 14, 2007 | Art Unit: 2626

For: A SPEECH RECOGNITION CIRCUIT AND METHOD | Examiner: S. I. McFadden

RESPONSE TO RESTRICTION REQUIREMENT

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Restriction Requirement set forth in the Office Action mailed May 6, 2010, Applicant hereby elects Group I, claims 1-5, 36-40, 42-46, and 86, for continued examination without traverse.

Dated: July 6, 2010

Respectfully submitted,

By _____
Stephen A. Soffen
Registration No.: 31,063
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
(202) 420-2200
Attorney for Applicant

DSMDB-2793222v1

Electronic Patent Application Fee Transmittal

Application Number:	11662704
Filing Date:	14-Mar-2007
Title of Invention:	Speech Recognition Circuit and Method
First Named Inventor/Applicant Name:	Guy Larri
Filer:	Stephen A. Soffen/Teresa Joel
Attorney Docket Number:	M0025.0369/P369

Filed as Small Entity

U.S. National Stage under 35 USC 371 Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 1 month with $0 paid	2251	1	65	65

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
		Total in USD ($)		**65**

Electronic Acknowledgement Receipt

EFS ID:	7961347
Application Number:	11662704
International Application Number:	
Confirmation Number:	2632
Title of Invention:	Speech Recognition Circuit and Method
First Named Inventor/Applicant Name:	Guy Larri
Customer Number:	24998
Filer:	Stephen A. Soffen/Teresa Joel
Filer Authorized By:	Stephen A. Soffen
Attorney Docket Number:	M0025.0369/P369
Receipt Date:	06-JUL-2010
Filing Date:	14-MAR-2007
Time Stamp:	18:11:00
Application Type:	U.S. National Stage under 35 USC 371

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$65
RAM confirmation Number	8097
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1		369.PDF	76495 93549e484b321ee01cbb300cd0382c555331e450	yes	2

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Extension of Time	1	1
Response to Election / Restriction Filed	2	2

Warnings:

Information:

2	Fee Worksheet (PTO-875)	fee-info.pdf	30086 212852a5706f60d4ce6911c328475e4fcde93516	no	2

Warnings:

Information:

Total Files Size (in bytes):	106581

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/22 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless if displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) FY 2009 *(Fees pursuant to the Consolidated Appropriations Act, 2005 (H. R. 4818).)*	Docket Number (Optional) M0025.0369/P369
Application Number 11/662,704-Conf. #2632	Filed March 14, 2007
For A SPEECH RECOGNITION CIRCUIT AND METHOD	
Art Unit 2626	Examiner S. I. McFadden

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Small Entity Fee	
[X]	One month (37 CFR 1.17(a)(1))	$130	$65	$ 65.00
[ ]	Two months (37 CFR 1.17(a)(2))	$490	$245	$
[ ]	Three months (37 CFR 1.17(a)(3))	$1110	$555	$
[ ]	Four months (37 CFR 1.17(a)(4))	$1730	$865	$
[ ]	Five months (37 CFR 1.17(a)(5))	$2350	$1175	$

[X] Applicant claims small entity status. See 37 CFR 1.27.

[ ] A check in the amount of the fee is enclosed.

[X] Payment by credit card. Form PTO-2038 is attached.

[ ] The Director has already been authorized to charge fees in this application to a Deposit Account.

[X] The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 04-1073 .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

[ ] applicant/inventor.

[ ] assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

[X] attorney or agent of record. Registration Number 31,063

[ ] attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 _____

Signature — July 6, 2010 (Date)

Stephen A. Soffen (Typed or printed name) — (202) 420-4879 (Telephone Number)

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

[ ] Total of 1 forms are submitted.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

24998 7590 08/17/2010

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

EXAMINER	
MCFADDEN, SUSAN IRIS	
ART UNIT	PAPER NUMBER
2626	

MAIL DATE	DELIVERY MODE
08/17/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/662,704	LARRI ET AL.	
	Examiner	Art Unit	
	Susan McFadden	2626	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 July 2010*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,36-40,42-46 and 86* is/are pending in the application.

4a) Of the above claim(s) *10-23,26,28-30,32-35 and 47-85* is/are withdrawn from consideration.

5)☒ Claim(s) *1-5,36,42-44, and 86* is/are allowed.

6)☒ Claim(s) *45 and 46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 March 2007* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Specification

1. The abstract of the disclosure is objected to because it should be on a separate page. Correction is required. See MPEP § 608.01(b).

2. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Rejections - 35 USC § 101

1. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

2. Claims 45 and 46 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. While the claim(s) recite a series of steps or acts to be performed, a statutory "process" under 35 USC 101 must (1) be tied to another statutory category (such as a manufacture or a machine), or (2) transform underlying subject matter (such as an article or material) to a different state or thing. The instant claim(s) neither transform underlying subject matter nor positively recite structure associated with another statutory category, and therefore do not define a statutory process. In regard to claims 45 and 46, a computer or other structure is not recited.

Allowable Subject Matter

3. Claims 1-5,36,42-44 and 86 are allowed.

4. The following is a statement of reasons for the indication of allowable subject matter: The prior art, Wells et al. (7,328,153) shows a speech recognition system with an audio front end that calculate feature vectors, a calculating circuit for calculating

distances. In regard to claim 1, the prior art of record does not show or suggest that the calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model, or a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words, wherein said audio front end and said search stage are implemented using a first processor, and said calculating circuit is implemented using a second processor, and wherein data is pipelined from the front end to the calculating circuit to the search stage.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Susan McFadden whose telephone number is 571-272-7621. The examiner can normally be reached on Monday-Friday, 9:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Richemond Dorvil can be reached on 571-272-7602. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Susan McFadden/
Primary Examiner, Art Unit 2626
August 12, 2010

Notice of References Cited	Application/Control No. 11/662,704	Applicant(s)/Patent Under Reexamination LARRI ET AL.	
	Examiner Susan McFadden	Art Unit 2626	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-7,328,153	02-2008	Wells et al.	704/231
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Amazon / Zentian Limited
Exhibit 1002
Page 413

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	Art Unit
	Susan McFadden	2626

SEARCHED			
Class	Subclass	Date	Examiner

SEARCH NOTES		
Search Notes	Date	Examiner
Searched WEST	8-11-10	SM

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	Art Unit
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010	08/11/2010							
	1	÷	=							
	2	÷	=							
	3	÷	=							
	4	÷	=							
	5	÷	=							
	6	-	-							
	7	-	-							
	8	-	-							
	9	-	-							
	10	÷	N							
	11	÷	N							
	12	÷	N							
	13	÷	N							
	14	÷	N							
	15	÷	N							
	16	÷	N							
	17	÷	N							
	18	÷	N							
	19	÷	N							
	20	÷	N							
	21	÷	N							
	22	÷	N							
	23	÷	N							
	24	-	-							
	25	-	-							
	26	÷	N							
	27	-	-							
	28	÷	N							
	29	÷	N							
	30	÷	N							
	31	-	-							
	32	÷	N							
	33	÷	N							
	34	÷	N							
	35	÷	N							
	36	÷	=							

Amazon / Zentian Limited
Exhibit 1002
Page 415

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010	08/11/2010							
	37	÷	=							
	38	÷	=							
	39	÷	=							
	40	÷	=							
	41	-								
	42	÷	=							
	43	÷	=							
	44	÷	=							
	45	÷	✓							
	46	÷	✓							
	47	-								
	48	-								
	49	-								
	50	-								
	51	-								
	52	-								
	53	-								
	54	-								
	55	-								
	56	-								
	57	-								
	58	-								
	59	-								
	60	-								
	61	-								
	62	-								
	63	-								
	64	-								
	65	-								
	66	-								
	67	-								
	68	-								
	69	-								
	70	-								
	71	-								
	72	-								

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010	08/11/2010							
	73	-								
	74	-								
	75	-								
	76	-								
	77	-								
	78	-								
	79	-								
	80	-								
	81	-								
	82	-								
	83	-								
	84	-								
	85	÷	N							
	86	÷	=							

IAP8 Rec'd PCT/PTO 14 MAR 2007

PTO/SB/08A/B (09-06)
Approved for use through 03/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	Not Yet Assigned
Filing Date	Concurrently Herewith
First Named Inventor	Guy Larri
Art Unit	N/A
Examiner Name	Not Yet Assigned
Attorney Docket Number	M0025.0369/P369

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (*if known*)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	AA*	5,881,312	03-09-1999	Dulong	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (*if known*)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. * CITE NO.: Those application(s) which are marked with an single asterisk (*) next to the Cite No. are not supplied (under 37 CFR 1.98(a)(2)(iii)) because that application was filed after June 30, 2003 or is available in the IFW. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
	CA	Glinski et al., " Spoken Language Recognition on a DSP Array Processor" IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 7, pp. 697-703, 1 July 1994.	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1]Applicant's unique citation designation number (optional). [2]Applicant is to place a check mark here if English language Translation is attached.

Examiner Signature	/Susan Mcfadden/	Date Considered	08/09/2010

DSMDB-2229147v01

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /S.M./

WEST Search History for Application 11662704

Creation Date: 2010081117:13

Interference Searches

Query	DB	Op.	Plur.	Thes.	Date
speech adj recognition and audio near front adj end and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	PGPB	OR	YES		08-11-2010
speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	PGPB	OR	YES		08-11-2010
speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010

Prior Art Searches

Query	DB	Op.	Plur.	Thes.	Date
natural adj language and hierarchical and semantic	USPT	OR	YES		04-26-2010
larri.in.	USPT	OR	YES		04-26-2010
larri.in. and speech adj recognition	USPT	OR	YES		04-26-2010
larri.in. and accelerator	USPT	OR	YES		04-26-2010
larri.in. and accelerator	PGPB	OR	YES		04-26-2010
larri.in. and accelerator and decompressor	PGPB	OR	YES		04-26-2010
accelerator and decompressor	USPT	OR	YES		04-26-2010
(accelerator and decompressor).clm.	USPT	OR	YES		04-26-2010
(accelerator and speech).clm.	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic).clm.	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and speech).clm.	USPT	OR	YES		04-26-2010
accelerator and distances and acoustic and speech adj recognition	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and speech adj recognition	USPT	OR	YES		04-26-2010
	USPT	OR	YES		04-26-2010

Amazon / Zentian Limited
Exhibit 1002
Page 419

accelerator and calculating near distances and acoustic and audio near signal					
(accelerator and distances and acoustic and speech adj recognition) and (accelerator).clm.	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and audio	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and (audio or speech)	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and (audio or speech) and recognition	USPT	OR	YES		04-26-2010
accelerator and distances and acoustic and (audio or speech)	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and (audio or speech)).clm.	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and (audio or speech)).clm.	PGPB	OR	YES		04-26-2010
larri.in.	USPT	OR	YES		08-11-2010
larri.in. and audio and speech adj recognition	USPT	OR	YES		08-11-2010
larri.in. and audio and acoustic	USPT	OR	YES		08-11-2010
larri.in. and speech	USPT	OR	YES		08-11-2010
larri.in. and speech	PGPB	OR	YES		08-11-2010
larri.in. and speech adj recognition	PGPB	OR	YES		08-11-2010
audio and front adj end and feature adj vector and distance and similarity and acoustic	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words and lexical	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words and lexical near tree	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words and search	USPT	OR	YES		08-11-2010

Amazon / Zentian Limited
Exhibit 1002
Page 420

(audio or speech adj recognition) and front adj end and feature adj vector and distance and similarity and acoustic and words and search	USPT	OR	YES		08-11-2010
(audio or speech adj recognition) and front adj end and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
speech adj recognition and audio near front adj end and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
(speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor)	USPT	OR	YES		08-11-2010

Amazon / Zentian Limited
Exhibit 1002
Page 421

Docket No.: M0025.0369/P369
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Guy Larri et al.

Application No.: 11/662,704 | Confirmation No.: 2632

Filed: March 14, 2007 | Art Unit: 2626

For: A SPEECH RECOGNITION CIRCUIT AND METHOD | Examiner: S. I. McFadden

AMENDMENT

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

INTRODUCTORY COMMENTS

In response to the Office Action dated August 17, 2010, please amend the above-identified U.S. patent application as follows:

Amendments to the Claims begin on page 2 of this paper.

Remarks begin on page 6 of this paper.

Amazon / Zentian Limited
Exhibit 1002
Page 422

AMENDMENTS TO THE CLAIMS

1. (Original) A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

a calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating circuit is implemented using a second processor, and wherein data is pipelined from the front end to the calculating circuit to the search stage.

2. (Original) A speech recognition circuit as claimed in claim 1, wherein the pipelining comprises alternating of front end and search stage processing on the first processor.

3. (Original) A speech recognition circuit as claimed in claim 1, comprising dynamic scheduling whether the first processor should run the front end or search stage code, based on availability or unavailablity of distance results and/or availability of space for storing more feature vectors and/or distance results.

4. (Original) A speech recognition circuit as claimed in claim 1, wherein the first processor supports multi-threaded operation, and runs the search stage and front ends as separate threads.

5. (Previously presented) A speech recognition circuit as claimed in claim 1, wherein the said calculating circuit is configured to autonomously calculate distances for every acoustic state defined by the acoustic model.

Amazon / Zentian Limited
Exhibit 1002
Page 423

Claims 6 – 35. (Canceled).

36. (Previously presented) The speech recognition circuit of claim 1, comprising control means adapted to implement frame dropping, to discard one or more audio time frames.

37. (Previously presented) The speech recognition circuit of claim 1, wherein the feature vector comprises a plurality of spectral components of an audio signal for a predetermined time frame.

38. (Previously presented) The speech recognition circuit of claim 1, wherein the processor is configured to divert to another task if the data flow stalls.

39. (Previously presented) The speech recognition circuit of claim 1, wherein the speech accelerator has an interrupt signal to inform the front end that the accelerator is ready to receive a next feature vector from the front end.

40. (Previously presented) The speech recognition circuit of claim 1, wherein the accelerator signals to the search stage when the distances for a new frame are available in a result memory.

41. (Canceled).

42. (Previously presented) The speech recognition circuit of claim 1, comprising increasing the pipeline depth by computing extra front frames in advance.

43. (Original) A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

Amazon / Zentian Limited
Exhibit 1002
Page 424

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end, said calculating means, and said search stage are connected to each other to enable pipelined data flow.

44. (Previously presented) The speech recognition circuit of claim 1, wherein the audio front end is configured to input a digital audio signal.

45. (Currently amended) A speech recognition method, comprising:

calculating a feature vector from an audio signal using an audio front end, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model using a calculating circuit; and

using ~~said calculated distances~~ a search stage to identify words within a lexical tree using said calculated distances, the lexical tree comprising a model of words;

~~wherein said audio front end and said search stage are implemented using a first processor, and said calculating means is implemented using a second processor, and~~ wherein data is pipelined from the front end, to the calculating circuit, and to the search stage.

Claims 46-85. (Canceled).

86. (Previously presented) A speech recognition circuit of claim 1, wherein said distance comprises a Mahalanobis distance.

87. (New) A storage medium storing processor implementable code for controlling at least one processor to implement a speech recognition method, the code comprising:

Amazon / Zentian Limited
Exhibit 1002
Page 425

code to calculate a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

code to calculate a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

code to identify words within a lexical tree using said calculated distances, the lexical tree comprising a model of words,

wherein data is pipelined by the code from the feature calculation, to the distance calculation, and to the word identification.

Amazon / Zentian Limited
Exhibit 1002
Page 426

REMARKS

Claim 45 has been amended. Claims 10-23, 26, 28-30, 32-35, 46, and 85 have been canceled without prejudice. Claim 87 has been added. Claims 1-5, 36-40, 42-46, 86, and 87 are currently pending. Applicant thanks the Examiner for the allowance of claims 1-5, 36, 42-44, and 86.

Applicant notes that claims 37-40 have not been treated in this Office Action. Applicant assumes that these claims are allowable because they depend from allowed claim 1 and requests that these claim be allowed.

Claim 45 stands rejected under 35 U.S.C. § 101 for being directed toward non-statutory subject matter.[1] This rejection is respectfully traversed. Claim 45 has been amended to address this rejection. In particular, claim 45 has been amended to recite a "speech recognition method, comprising: calculating a feature vector from an audio signal using an audio front end ... calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model using a calculating circuit; and using a search stage to identify words within a lexical tree using said calculated distances." (Emphasis added). Accordingly, Applicant requests that the rejection be withdrawn and the claim be allowed.

Applicant also submits that new claim 87 is allowable over the art of record and requests that it be allowed.

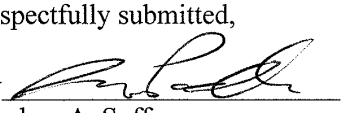
[1] Applicant notes that claim 46 was also rejected under 35 U.S.C. § 101. This claim has been canceled without prejudice and thus is not discussed.

Amazon / Zentian Limited
Exhibit 1002
Page 427

In view of the above, Applicant submits that the present application is in condition for allowance.

Dated: January 18, 2011

Respectfully submitted,

By 

Stephen A. Soffen
Registration No.: 31,063
Brian Parke
Registration No.: 59,226
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
(202) 420-2200
Attorneys for Applicant

Electronic Patent Application Fee Transmittal

Application Number:	11662704
Filing Date:	14-Mar-2007
Title of Invention:	Speech Recognition Circuit and Method
First Named Inventor/Applicant Name:	Guy Larri
Filer:	Stephen A. Soffen/Kanessa Howard
Attorney Docket Number:	M0025.0369/P369

Filed as Small Entity

U.S. National Stage under 35 USC 371 Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 2 months with $0 paid	2252	1	245	245

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
		Total in USD ($)		**245**

Electronic Acknowledgement Receipt	
EFS ID:	9250543
Application Number:	11662704
International Application Number:	
Confirmation Number:	2632
Title of Invention:	Speech Recognition Circuit and Method
First Named Inventor/Applicant Name:	Guy Larri
Customer Number:	24998
Filer:	Stephen A. Soffen/Kanessa Howard
Filer Authorized By:	Stephen A. Soffen
Attorney Docket Number:	M0025.0369/P369
Receipt Date:	18-JAN-2011
Filing Date:	14-MAR-2007
Time Stamp:	14:59:47
Application Type:	U.S. National Stage under 35 USC 371

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$245
RAM confirmation Number	1557
Deposit Account	041073
Authorized User	SOFFEN,STEPHEN A.
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. 1.492 (National application filing, search, and examination fees) Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)	

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees) Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees) Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Extension of Time	EOT.PDF	280939 b2904693b263a6ab2b1fd5728b4d7f6a0c934900	no	1
Warnings:					
Information:					
2		amendment.PDF	280928 a0e68b83ab3865a37433663e2d4aff56a7be39ba	yes	7

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Amendment/Req. Reconsideration-After Non-Final Reject	1	1
	Claims	2	5
	Applicant Arguments/Remarks Made in an Amendment	6	7

Warnings:					
Information:					
3	Fee Worksheet (PTO-875)	fee-info.pdf	30128 266a6de87b154416816d138dd720829b58bea9f8	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	591995		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/22 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless if displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) FY 2009 *(Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).)*	Docket Number (Optional) M0025.0369/P369
Application Number 11/662,704-Conf. #2632	Filed March 14, 2007
For A SPEECH RECOGNITION CIRCUIT AND METHOD	
Art Unit 2626	Examiner S. I. McFadden

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Small Entity Fee	
[ ]	One month (37 CFR 1.17(a)(1))	$130	$65	$
[x]	Two months (37 CFR 1.17(a)(2))	$490	$245	$ 245.00
[ ]	Three months (37 CFR 1.17(a)(3))	$1110	$555	$
[ ]	Four months (37 CFR 1.17(a)(4))	$1730	$865	$
[ ]	Five months (37 CFR 1.17(a)(5))	$2350	$1175	$

[x] Applicant claims small entity status. See 37 CFR 1.27.

[ ] A check in the amount of the fee is enclosed.

[x] Payment by credit card. Form PTO-2038 is attached.

[ ] The Director has already been authorized to charge fees in this application to a Deposit Account.

[x] The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 04-1073 .

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

[ ] applicant/inventor.

[ ] assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

[x] attorney or agent of record. Registration Number 59,226

[ ] attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 _____

Signature	January 18, 2011 Date
Brian Parke Typed or printed name	(202) 420-3581 Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

[ ] Total of 1 forms are submitted.

Amazon / Zentian Limited
Exhibit 1002
Page 434

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 11/662,704	Filing Date 03/14/2007	☐ To be Mailed

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☒		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

		(Column 1)		(Column 2)	(Column 3)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
	01/18/2011	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
AMENDMENT	Total (37 CFR 1.16(i))	* 16	Minus	** 38	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	* 4	Minus	***6	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

		(Column 1)		(Column 2)	(Column 3)					
		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
AMENDMENT	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/PEGGY YARBOROUGH/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Docket No.: M0025.0369/P369
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Guy Larri et al.

Application No.: 11/662,704 | Confirmation No.: 2632

Filed: March 14, 2007 | Art Unit: 2626

For: A SPEECH RECOGNITION CIRCUIT AND METHOD | Examiner: S. I. McFadden

SUPPLEMENTAL AMENDMENT

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

INTRODUCTORY COMMENTS

Supplemental to the Amendment filed on January 18, 2011, please amend the above-identified U.S. patent application as follows:

Amendments to the Claims begin on page 2 of this paper.

Remarks begin on page 6 of this paper.

Amazon / Zentian Limited
Exhibit 1002
Page 436

AMENDMENTS TO THE CLAIMS

1. (Original) A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

a calculating circuit for calculating distances indicating the similarity between a feature vector and a plurality of predetermined acoustic states of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end and said search stage are implemented using a first processor, and said calculating circuit is implemented using a second processor, and wherein data is pipelined from the front end to the calculating circuit to the search stage.

2. (Original) A speech recognition circuit as claimed in claim 1, wherein the pipelining comprises alternating of front end and search stage processing on the first processor.

3. (Original) A speech recognition circuit as claimed in claim 1, comprising dynamic scheduling whether the first processor should run the front end or search stage code, based on availability or unavailablity of distance results and/or availability of space for storing more feature vectors and/or distance results.

4. (Original) A speech recognition circuit as claimed in claim 1, wherein the first processor supports multi-threaded operation, and runs the search stage and front ends as separate threads.

5. (Previously presented) A speech recognition circuit as claimed in claim 1, wherein the said calculating circuit is configured to autonomously calculate distances for every acoustic state defined by the acoustic model.

Claims 6-35. (Canceled).

36. (Previously presented) The speech recognition circuit of claim 1, comprising control means adapted to implement frame dropping, to discard one or more audio time frames.

37. (Previously presented) The speech recognition circuit of claim 1, wherein the feature vector comprises a plurality of spectral components of an audio signal for a predetermined time frame.

38. (Previously presented) The speech recognition circuit of claim 1, wherein the processor is configured to divert to another task if the data flow stalls.

39. (Previously presented) The speech recognition circuit of claim 1, wherein the speech accelerator has an interrupt signal to inform the front end that the accelerator is ready to receive a next feature vector from the front end.

40. (Previously presented) The speech recognition circuit of claim 1, wherein the accelerator signals to the search stage when the distances for a new frame are available in a result memory.

41. (Canceled).

42. (Previously presented) The speech recognition circuit of claim 1, comprising increasing the pipeline depth by computing extra front frames in advance.

43. (Original) A speech recognition circuit, comprising:

an audio front end for calculating a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating means for calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

a search stage for using said calculated distances to identify words within a lexical tree, the lexical tree comprising a model of words;

wherein said audio front end, said calculating means, and said search stage are connected to each other to enable pipelined data flow.

44. (Previously presented) The speech recognition circuit of claim 1, wherein the audio front end is configured to input a digital audio signal.

45. (Previously presented) A speech recognition method, comprising:

calculating a feature vector from an audio signal using an audio front end, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

calculating a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model using a calculating circuit; and

using a search stage to identify words within a lexical tree using said calculated distances, the lexical tree comprising a model of words;

wherein data is pipelined from the front end, to the calculating circuit, and to the search stage.

Claims 46-85. (Canceled).

86. (Previously presented) A speech recognition circuit of claim 1, wherein said distance comprises a Mahalanobis distance.

87. (Currently amended) A non-transitory storage medium storing processor implementable code for controlling at least one processor to implement a speech recognition method, the code comprising:

code for controlling the processor to calculate a feature vector from an audio signal, wherein the feature vector comprises a plurality of extracted and/or derived quantities from said audio signal during a defined audio time frame;

code for controlling the processor to calculate a distance indicating the similarity between a feature vector and a predetermined acoustic state of an acoustic model; and

code for controlling the processor to identify words within a lexical tree using said calculated distances, the lexical tree comprising a model of words,

wherein data is pipelined by the processor pursuant to the code from the feature calculation, to the distance calculation, and to the word identification.

Amazon / Zentian Limited
Exhibit 1002
Page 440

REMARKS

Claim 87 has been amended so that it is clearly directed to statutory subject matter under 35 U.S.C. § 101

In view of the above, Applicant submits that the present application is in condition for allowance.

Dated: January 27, 2011

Respectfully submitted,

By _____

Stephen A. Soffen
Registration No.: 31,063
Brian Parke
Registration No.: 59,226
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
(202) 420-2200
Attorneys for Applicant

Electronic Acknowledgement Receipt	
EFS ID:	9318710
Application Number:	11662704
International Application Number:	
Confirmation Number:	2632
Title of Invention:	Speech Recognition Circuit and Method
First Named Inventor/Applicant Name:	Guy Larri
Customer Number:	24998
Filer:	Stephen A. Soffen/Kanessa Howard
Filer Authorized By:	Stephen A. Soffen
Attorney Docket Number:	M0025.0369/P369
Receipt Date:	27-JAN-2011
Filing Date:	14-MAR-2007
Time Stamp:	13:03:37
Application Type:	U.S. National Stage under 35 USC 371

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		supp_amendment.pdf	173120 3b379f7f6d780a2d7c1bd0808edce8264f662940	yes	6

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Supplemental Response or Supplemental Amendment	1	1
	Claims	2	5
	Applicant Arguments/Remarks Made in an Amendment	6	6
Warnings:			
Information:			
	Total Files Size (in bytes):	173120	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/662,704	03/14/2007	

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☒		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT	01/27/2011	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 16	Minus	** 39	= 0	X $26 =	0	OR	X $ =	
	Independent (37 CFR 1.16(h))	* 4	Minus	***6	= 0	X $110 =	0	OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/FRANCES Y. FIELDS/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

24998 7590 02/07/2011

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

EXAMINER	
MCFADDEN, SUSAN IRIS	
ART UNIT	PAPER NUMBER
2626	

DATE MAILED: 02/07/2011

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

TITLE OF INVENTION: SPEECH RECOGNITION CIRCUIT AND METHOD

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$755	$300	$0	$1055	05/09/2011

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PTOL-85 (Rev. 08/07) Approved for use through 08/31/2010.

Amazon / Zentian Limited
Exhibit 1002
Page 445

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

24998 7590 02/07/2011

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

TITLE OF INVENTION: SPEECH RECOGNITION CIRCUIT AND METHOD

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$755	$300	$0	$1055	05/09/2011

EXAMINER	ART UNIT	CLASS-SUBCLASS
MCFADDEN, SUSAN IRIS	2626	704-238000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

❑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

❑ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Amazon / Zentian Limited
Exhibit 1002
Page 446

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

24998 7590 02/07/2011

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

EXAMINER	
MCFADDEN, SUSAN IRIS	
ART UNIT	PAPER NUMBER
2626	

DATE MAILED: 02/07/2011

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 651 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 651 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

PTOL-85 (Rev. 08/07) Approved for use through 08/31/2010.

Notice of Allowability	Application No. 11/662,704	Applicant(s) LARRI ET AL.	
	Examiner Susan McFadden	Art Unit 2626	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *Amendment filed 1-27-11*.

2. ☒ The allowed claim(s) is/are *1-5,36-40,42-45,86 and 87*.

3. ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☒ All b) ☐ Some* c) ☐ None of the:
      1. ☒ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☐ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

/Susan McFadden/
Primary Examiner, Art Unit 2626

Amazon / Zentian Limited
Exhibit 1002
Page 448

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010	08/11/2010	01/31/2011						
1	1	÷	=	=						
2	2	÷	=	=						
3	3	÷	=	=						
4	4	÷	=	=						
5	5	÷	=	=						
	6	-	-	-						
	7	-	-	-						
	8	-	-	-						
	9	-	-	-						
	10	÷	N	-						
	11	÷	N	-						
	12	÷	N	-						
	13	÷	N	-						
	14	÷	N	-						
	15	÷	N	-						
	16	÷	N	-						
	17	÷	N	-						
	18	÷	N	-						
	19	÷	N	-						
	20	÷	N	-						
	21	÷	N	-						
	22	÷	N	-						
	23	÷	N	-						
	24	-	-	-						
	25	-	-	-						
	26	÷	N	-						
	27	-	-	-						
	28	÷	N	-						
	29	÷	N	-						
	30	÷	N	-						
	31	-	-	-						
	32	÷	N	-						
	33	÷	N	-						
	34	÷	N	-						
	35	÷	N	-						
6	36	÷	=	=						

Amazon / Zentian Limited
Exhibit 1002
Page 449

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010	08/11/2010	01/31/2011						
7	37	÷	=	=						
8	38	÷	=	=						
9	39	÷	=	=						
10	40	÷	=	=						
	41	-	-	-						
11	42	÷	=	=						
14	43	÷	=	=						
12	44	÷	=	=						
15	45	÷	✓	=						
	46	÷	✓	-						
	47	-		-						
	48	-		-						
	49	-		-						
	50	-		-						
	51	-		-						
	52	-		-						
	53	-		-						
	54	-		-						
	55	-		-						
	56	-		-						
	57	-		-						
	58	-		-						
	59	-		-						
	60	-		-						
	61	-		-						
	62	-		-						
	63	-		-						
	64	-		-						
	65	-		-						
	66	-		-						
	67	-		-						
	68	-		-						
	69	-		-						
	70	-		-						
	71	-		-						
	72	-		-						

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	05/04/2010	08/11/2010	01/31/2011						
	73	-		-						
	74	-		-						
	75	-		-						
	76	-		-						
	77	-		-						
	78	-		-						
	79	-		-						
	80	-		-						
	81	-		-						
	82	-		-						
	83	-		-						
	84	-		-						
	85	÷	N	-						
13	86	÷	=	=						
16	87			=						

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

ORIGINAL		INTERNATIONAL CLASSIFICATION	
CLASS	SUBCLASS	CLAIMED	NON-CLAIMED
704	238	G 1 0 L / ()	

CROSS REFERENCE(S)

CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1														
2	2														
3	3														
4	4														
5	5														
6	36														
7	37														
8	38														
9	39														
10	40														
11	42														
14	43														
12	44														
15	45														
13	86														
16	87														

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	16	
/Susan McFadden/ Primary Examiner.Art Unit 2626	2-1-11	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	16

Amazon / Zentian Limited
Exhibit 1002
Page 452

WEST Search History for Application 11662704

Creation Date: 2011012415:32

Interference Searches

Query	DB	Op.	Plur.	Thes.	Date
speech adj recognition and audio near front adj end and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	PGPB	OR	YES		08-11-2010
speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	PGPB	OR	YES		08-11-2010
speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
speech adj recognition and audio near front near end and distance and search near stage and tree	PGPB	OR	YES		01-24-2011

Prior Art Searches

Query	DB	Op.	Plur.	Thes.	Date
natural adj language and hierarchical and semantic	USPT	OR	YES		04-26-2010
larri.in.	USPT	OR	YES		04-26-2010
larri.in. and speech adj recognition	USPT	OR	YES		04-26-2010
larri.in. and accelerator	USPT	OR	YES		04-26-2010
larri.in. and accelerator	PGPB	OR	YES		04-26-2010
larri.in. and accelerator and decompressor	PGPB	OR	YES		04-26-2010
accelerator and decompressor	USPT	OR	YES		04-26-2010
(accelerator and decompressor).clm.	USPT	OR	YES		04-26-2010
(accelerator and speech).clm.	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic).clm.	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and speech).clm.	USPT	OR	YES		04-26-2010
accelerator and distances and acoustic and speech adj recognition	USPT	OR	YES		04-26-2010
	USPT	OR	YES		04-26-2010

Amazon / Zentian Limited
Exhibit 1002
Page 453

accelerator and calculating near distances and acoustic and speech adj recognition					
accelerator and calculating near distances and acoustic and audio near signal	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and speech adj recognition) and (accelerator).clm.	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and audio	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and (audio or speech)	USPT	OR	YES		04-26-2010
accelerator and calculating near distances and acoustic and (audio or speech) and recognition	USPT	OR	YES		04-26-2010
accelerator and distances and acoustic and (audio or speech)	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and (audio or speech)).clm.	USPT	OR	YES		04-26-2010
(accelerator and distances and acoustic and (audio or speech)).clm.	PGPB	OR	YES		04-26-2010
larri.in.	USPT	OR	YES		08-11-2010
larri.in. and audio and speech adj recognition	USPT	OR	YES		08-11-2010
larri.in. and audio and acoustic	USPT	OR	YES		08-11-2010
larri.in. and speech	USPT	OR	YES		08-11-2010
larri.in. and speech	PGPB	OR	YES		08-11-2010
larri.in. and speech adj recognition	PGPB	OR	YES		08-11-2010
audio and front adj end and feature adj vector and distance and similarity and acoustic	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words and lexical	USPT	OR	YES		08-11-2010
(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words and lexical near tree	USPT	OR	YES		08-11-2010
	USPT	OR	YES		08-11-2010

Amazon / Zentian Limited
Exhibit 1002
Page 454

(audio or speech) and front adj end and feature adj vector and distance and similarity and acoustic and words and search					
(audio or speech adj recognition) and front adj end and feature adj vector and distance and similarity and acoustic and words and search	USPT	OR	YES		08-11-2010
(audio or speech adj recognition) and front adj end and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
speech adj recognition and audio near front adj end and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor	USPT	OR	YES		08-11-2010
(speech adj recognition and audio and feature adj vector and distance and similarity and acoustic and words and search and (first and second) adj processor)	USPT	OR	YES		08-11-2010
speech adj recognition and audio near front near end and distance and search near stage	USPT	OR	YES		01-24-2011
speech adj recognition and audio near front near end and distance and search near stage and tree	USPT	OR	YES		01-24-2011
speech adj recognition and audio near front near end and distance and search near stage and tree	PGPB	OR	YES		01-24-2011
speech adj recognition and audio near front near end and distance and search near stage and tree	PGPB	OR	YES		01-24-2011
speech adj recognition and audio near front near end and distance and search near stage and tree	PGPB	OR	YES		01-24-2011
speech adj recognition and audio near front near end and distance and search near stage and tree and (hardware and software)	PGPB	OR	YES		01-24-2011
speech adj recognition and audio near front near end and distance and search near stage and tree and (hardware and software) and medium	PGPB	OR	YES		01-24-2011
larri.in.	USPT	OR	YES		01-24-2011
larri.in. and speech adj recognition	USPT	OR	YES		01-24-2011
larri.in. and speech adj recognition	USPT	OR	YES		01-24-2011
larri.in. and speech adj recognition	PGPB	OR	YES		01-24-2011

Amazon / Zentian Limited
Exhibit 1002
Page 455

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11662704	LARRI ET AL.
	Examiner	**Art Unit**
	Susan McFadden	2626

ORIGINAL		INTERNATIONAL CLASSIFICATION										
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED					
704	238	G	1	0	L	15 / 00 (2006.01.01)						

CROSS REFERENCE(S)						
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1														
2	2														
3	3														
4	4														
5	5														
6	36														
7	37														
8	38														
9	39														
10	40														
11	42														
14	43														
12	44														
15	45														
13	86														
16	87														

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	16	
/Susan McFadden/ Primary Examiner.Art Unit 2626	2-1-11	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	16

Amazon / Zentian Limited
Exhibit 1002
Page 456

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

7590 05/23/2011

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

EXAMINER	
MCFADDEN, SUSAN IRIS	
ART UNIT	PAPER NUMBER
2626	
MAIL DATE	DELIVERY MODE
05/23/2011	PAPER

Notice of Abandonment

This application is abandoned in view of:

1. ☐ The applicant's failure to timely file a proper reply to the Office letter mailed on _____.
   (a) ☐ A reply was received on _____ (with a Certificate of Mailing or Transmission date _____), which is after the expiration of the period for reply (including a total extension of ____ month(s)) which expired on _____.
   (b) ☐ No reply has been received.
2. ☒ Applicant's failure to timely pay the required issue fee and publication fee, if applicable, within the statutory period of three months from the mailing date of the Notice of Allowance (PTOL-85).
   (a) ☐ The issue fee and publication fee, if applicable, was received on _____ (with a Certificate of Mailing or Transmission date _____), which is after the expiration of the statutory period for payment of the issue fee (and publication fee) set in the Notice of Allowance (PTOL-85).
   (b) ☐ The submitted fee of $_____ is insufficient. A balance of $_____ is due.
   The issue fee required by 37 CFR 1.18 is $_____.
   The publication fee, if required by 37 CFR 1.18(d), is $_____.
   (c) ☒ The issue fee and publication fee, if applicable, has not been received.
3. ☐ Applicant's failure to timely file corrected drawings as required by, and within the three-month period set in, the Notice of Allowability (PTO-37).
   (a) ☐ Proposed corrected drawings were received on _____ (with a Certificate of Mailing or Transmission dated _____), which is after the expiration of the period for reply.
   (b) ☐ No corrected drawing have been received.
4. ☐ The letter of express abandonment which is signed by the attorney or agent of record, the assignee of the entire interest, or all of the applicants.
5. ☐ The letter of express abandonment which is signed by an attorney or agent (acting in a representative capacity under 37 CFR 1.34(a)) upon the filling of a continuing application.
6. ☐ Drawings received on _____ were disapproved by examiner. See examiner's response dated _____.
7. ☐ Corrected drawings were received on _____ , which is after the expiration of the one-month period for reply set in examiner's response dated _____.
8. ☐ No corrected drawings have been received in reply to one-month period set in examiner's response dated _____.
9. ☐ The reason(s) below:

Petitions to revive under 37 CFR 1.137(a) or (b), or request to withdraw the holding of abandonment under 37 CFR 1.181, should be promptly filed to minimize any negative effects on patent term.

(571)-272-4200 or 1(888)-786-0101
Patent Publication Branch
Office of Data Management

FORM PTO-ABN0 (Rev. 06/09)

PTO/SB/64 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR REVIVAL OF AN APPLICATION FOR PATENT ABANDONED UNINTENTIONALLY UNDER 37 CFR 1.137(b)	Docket Number (Optional) M0025.0369/P369

First named inventor: Guy Larri

Application No: 11/662,704-Conf. #2632 Art Unit: 2626

Filed: March 14, 2007 Examiner: S. I. McFadden

Title: A SPEECH RECOGNITION CIRCUIT AND METHOD

Attention: Office of Petitions
Mail Stop Petition
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
FAX (571) 273-8300

NOTE: If information or assistance is needed in completing this form, please contact Petitions Information at (571) 272-3282.

The above-identified application became abandoned for failure to file a timely and proper reply to a notice or action by the United States Patent and Trademark Office. The date of abandonment is the day after the expiration date of the period set for reply in the office notice or action plus an extensions of time actually obtained.

APPLICANT HEREBY PETITIONS FOR REVIVAL OF THIS APPLICATION

NOTE: A grantable petition requires the following items:
(1) Petition fee;
(2) Reply and/or issue fee;
(3) Terminal disclaimer with disclaimer fee – required for all utility and plant applications filed before June 8, 1995; and for all design applications; and
(4) Statement that the entire delay was unintentional.

1. Petition fee

[x] Small entity – fee $ 810.00 (37 CFR 1.17(m)). Applicant claims small entity status. See 37 CFR 1.27.

[ ] Other than small entity – fee $ _____ (37 CFR 1.17(m))

2. Reply and/or fee

A. The reply and/or fee to the above-noted Office action in the form of _____ (identify type of reply):

[ ] has been filed previously on _____ .

[ ] is enclosed herewith.

B. The issue fee and publication fee (if applicable) of $ 1,070.00 .

[ ] has been paid previously on _____ .

[x] is enclosed herewith.

3. Terminal disclaimer with disclaimer fee

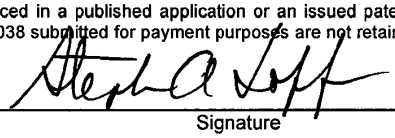
[X] Since this utility/plant application was filed on or after June 8, 1995, no terminal disclaimer is required.

[ ] A terminal disclaimer (and disclaimer fee (37 CFR 1.20(d)) of $ _____ for a small entity or $ _____ for other than a small entity) disclaiming the required period of time is enclosed herewith (see PTO/SB/63).

4. STATEMENT: The entire delay in filing the required reply from the due date for the required reply until the filing of a grantable petition under 37 CFR 1.137(b) was unintentional. [NOTE: The United States Patent and Trademark Office may require additional information if there is a question as to whether either the abandonment or the delay in filing a petition under 37 CFR 1.137(b) was unintentional (MPEP 711.03(c), subsections (III)(C) and (D)).]

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.



Signature

May 25, 2011
Date

Stephen A. Soffen
Typed or printed name

31,063
Registration Number, if applicable

DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
Address

(202) 420-4879
Telephone Number

Enclosures:
[X] Fee Payment
[ ] Reply
[ ] Terminal Disclaimer Form
[ ] Additional sheets containing statements establishing unintentional delay
[ ] Other: _____

PART B -FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571) 273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

TITLE OF INVENTION: A SPEECH RECOGNITION CIRCUIT AND METHOD

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
Non-Provisional	yes	$755.00	$300.00	$1,070.00	05/09/2011

EXAMINER	ART UNIT	CLASS-SUBCLASS
S. I. McFadden	2626	

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- [ ] Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- [ ] "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 Dickstein Shapiro LLP
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Zentian Limited

(B) RESIDENCE: (CITY and STATE OR COUNTRY): Cambridge, United Kingdom

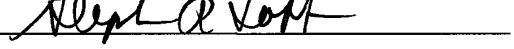
Please check the appropriate assignee category or categories (will not be printed on the patent): [ ] Individual [X] Corporation or other private group entity [ ] Government

4a. The following fee(s) are enclosed:
- [X] Issue Fee
- [X] Publication Fee (No small entity discount permitted)
- [X] Advance Order -# of Copies 5

4b. Payment of Fee(s):
- [ ] A check in the amount of the fee(s) is enclosed.
- [X] Payment by credit card. Form PTO-2038 is attached.
- [ ] The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 04-1073

5. **Change in Entity Status** (from status indicated above)
- [ ] a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- [ ] b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature  Date May 25, 2011

Typed or printed name Stephen A. Soffen Registration No. 31,063

DSMDB-2928679v1PTOL-85 (Rev. 08/08) Approved for use through 08/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Electronic Patent Application Fee Transmittal

Application Number:	11662704
Filing Date:	14-Mar-2007
Title of Invention:	SPEECH RECOGNITION CIRCUIT AND METHOD
First Named Inventor/Applicant Name:	Guy Larri
Filer:	Stephen A. Soffen/Kanessa Howard
Attorney Docket Number:	M0025.0369/P369
Filed as Small Entity	

U.S. National Stage under 35 USC 371 Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Petition-revive unintent. abandoned appl	2453	1	810	810
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Utility Appl issue fee	2501	1	755	755
Publ. Fee- early, voluntary, or normal	1504	1	300	300
Extension-of-Time:				
Miscellaneous:				
Printed copy of patent - no color	8001	5	3	15
		Total in USD ($)		**1880**

Electronic Acknowledgement Receipt	
EFS ID:	10168624
Application Number:	11662704
International Application Number:	
Confirmation Number:	2632
Title of Invention:	SPEECH RECOGNITION CIRCUIT AND METHOD
First Named Inventor/Applicant Name:	Guy Larri
Customer Number:	24998
Filer:	Stephen A. Soffen/Kanessa Howard
Filer Authorized By:	Stephen A. Soffen
Attorney Docket Number:	M0025.0369/P369
Receipt Date:	25-MAY-2011
Filing Date:	14-MAR-2007
Time Stamp:	16:06:21
Application Type:	U.S. National Stage under 35 USC 371

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		Petition_to_Revive_Filing.pdf	163364 d2512e7e486bae2888c33caee6252eb7dc45f841	yes	3

	Multipart Description/PDF files in .zip description				
	Document Description		**Start**	**End**	
	Petition for review by the Office of Petitions.		1	2	
	Issue Fee Payment (PTO-85B)		3	3	
Warnings:					
Information:					
2	Fee Worksheet (PTO-875)	fee-info.pdf	35190 f302cb142f548412638ffdee5c32ad55918adfb8	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	198554		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PART B -FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571) 273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403



Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

TITLE OF INVENTION: A SPEECH RECOGNITION CIRCUIT AND METHOD

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
Non-Provisional	yes	$755.00	$300.00	$1,070.00	05/09/2011

EXAMINER	ART UNIT	CLASS-SUBCLASS
S. I. McFadden	2626	

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- [ ] Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- [ ] "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 Dickstein Shapiro LLP
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Zentian Limited
(B) RESIDENCE: (CITY and STATE OR COUNTRY): Cambridge, United Kingdom

Please check the appropriate assignee category or categories (will not be printed on the patent): [ ] Individual [X] Corporation or other private group entity [ ] Government

4a. The following fee(s) are enclosed:
- [X] Issue Fee
- [X] Publication Fee (No small entity discount permitted)
- [X] Advance Order -# of Copies 5

4b. Payment of Fee(s):
- [ ] A check in the amount of the fee(s) is enclosed.
- [X] Payment by credit card. Form PTO-2038 is attached.
- [ ] The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 04-1073

5. **Change in Entity Status** (from status indicated above)
- [ ] a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- [ ] b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature Stephen A. Soffen Date May 25, 2011
Typed or printed name Stephen A. Soffen Registration No. 31,063

DSMDB-2928679v1PTOL-85 (Rev. 08/08) Approved for use through 08/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

The PTO did not receive the following listed item(s) credit card

05/26/2011 MBELETE2 00000041 041073 11662704
01 FC:2501 755.00 DA
02 FC:1504 300.00 DA
03 FC:8001 15.00 DA

PTO/SB/64 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR REVIVAL OF AN APPLICATION FOR PATENT ABANDONED UNINTENTIONALLY UNDER 37 CFR 1.137(b)	Docket Number (Optional) M0025.0369/P369

First named inventor: Guy Larri

Application No: 11/662,704-Conf. #2632 Art Unit: 2626

Filed: March 14, 2007 Examiner: S. I. McFadden

Title: A SPEECH RECOGNITION CIRCUIT AND METHOD

Attention: Office of Petitions
Mail Stop Petition
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
FAX (571) 273-8300

NOTE: If information or assistance is needed in completing this form, please contact Petitions Information at (571) 272-3282.

The above-identified application became abandoned for failure to file a timely and proper reply to a notice or action by the United States Patent and Trademark Office. The date of abandonment is the day after the expiration date of the period set for reply in the office notice or action plus an extensions of time actually obtained.

APPLICANT HEREBY PETITIONS FOR REVIVAL OF THIS APPLICATION

NOTE: A grantable petition requires the following items:

(1) Petition fee;
(2) Reply and/or issue fee;
(3) Terminal disclaimer with disclaimer fee – required for all utility and plant applications filed before June 8, 1995; and for all design applications; and
(4) Statement that the entire delay was unintentional.

1. Petition fee

[x] Small entity – fee $ 810.00 (37 CFR 1.17(m)). Applicant claims small entity status. See 37 CFR 1.27.

[ ] Other than small entity – fee $ _____ (37 CFR 1.17(m))

2. Reply and/or fee

A. The reply and/or fee to the above-noted Office action in the form of _____ (identify type of reply):

[ ] has been filed previously on _____ .

[ ] is enclosed herewith.

B. The issue fee and publication fee (if applicable) of $ 1,070.00 .

[ ] has been paid previously on _____ .

[x] is enclosed herewith.

3. Terminal disclaimer with disclaimer fee

[X] Since this utility/plant application was filed on or after June 8, 1995, no terminal disclaimer is required.

[ ] A terminal disclaimer (and disclaimer fee (37 CFR 1.20(d)) of $ _____ for a small entity or $ _____ for other than a small entity) disclaiming the required period of time is enclosed herewith (see PTO/SB/63).

4. STATEMENT: The entire delay in filing the required reply from the due date for the required reply until the filing of a grantable petition under 37 CFR 1.137(b) was unintentional. [NOTE: The United States Patent and Trademark Office may require additional information if there is a question as to whether either the abandonment or the delay in filing a petition under 37 CFR 1.137(b) was unintentional (MPEP 711.03(c), subsections (III)(C) and (D)).]

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

Signature	May 26, 2011 Date
Stephen A. Soffen Typed or printed name	31,063 Registration Number, if applicable
DICKSTEIN SHAPIRO LLP 1825 Eye Street, NW Washington, DC 20006-5403 Address	(202) 420-4879 Telephone Number

Enclosures:

[X] Fee Payment

[ ] Reply

[ ] Terminal Disclaimer Form

[ ] Additional sheets containing statements establishing unintentional delay

[ ] Other: _____

PART B -FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571) 273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

TITLE OF INVENTION: A SPEECH RECOGNITION CIRCUIT AND METHOD

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
Non-Provisional	yes	$755.00	$300.00	$1,070.00	05/09/2011

EXAMINER	ART UNIT	CLASS-SUBCLASS
S. I. McFadden	2626	

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- [ ] Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- [ ] "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 Dickstein Shapiro LLP
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Zentian Limited
(B) RESIDENCE: (CITY and STATE OR COUNTRY): Cambridge, United Kingdom

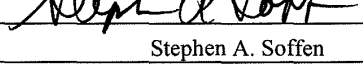
Please check the appropriate assignee category or categories (will not be printed on the patent): [ ] Individual [X] Corporation or other private group entity [ ] Government

4a. The following fee(s) are enclosed:
- [X] Issue Fee
- [X] Publication Fee (No small entity discount permitted)
- [X] Advance Order -# of Copies 5

4b. Payment of Fee(s):
- [ ] A check in the amount of the fee(s) is enclosed.
- [X] Payment by credit card. Form PTO-2038 is attached.
- [X] The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 04-1073

5. **Change in Entity Status** (from status indicated above)
- [ ] a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- [ ] b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature  Date May 25, 2011

Typed or printed name Stephen A. Soffen Registration No. 31,063

DSMDB-2928679v1PTOL-85 (Rev. 08/08) Approved for use through 08/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Electronic Patent Application Fee Transmittal

Application Number:	11662704
Filing Date:	14-Mar-2007
Title of Invention:	SPEECH RECOGNITION CIRCUIT AND METHOD
First Named Inventor/Applicant Name:	Guy Larri
Filer:	Stephen A. Soffen/Deborah Proctor
Attorney Docket Number:	M0025.0369/P369

Filed as Small Entity

U.S. National Stage under 35 USC 371 Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Petition-revive unintent. abandoned appl	2453	1	810	810
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Utility Appl issue fee	2501	1	755	755
Publ. Fee- early, voluntary, or normal	1504	1	300	300
Extension-of-Time:				
Miscellaneous:				
Printed copy of patent - no color	8001	5	3	15
		Total in USD ($)		**1880**

Electronic Acknowledgement Receipt

EFS ID:	10172796
Application Number:	11662704
International Application Number:	
Confirmation Number:	2632
Title of Invention:	SPEECH RECOGNITION CIRCUIT AND METHOD
First Named Inventor/Applicant Name:	Guy Larri
Customer Number:	24998
Filer:	Stephen A. Soffen/Deborah Proctor
Filer Authorized By:	Stephen A. Soffen
Attorney Docket Number:	M0025.0369/P369
Receipt Date:	26-MAY-2011
Filing Date:	14-MAR-2007
Time Stamp:	11:38:14
Application Type:	U.S. National Stage under 35 USC 371

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$1880
RAM confirmation Number	11223
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1		M00250369.PDF	204809 c2953500a532671c1e88f436ca14e4600a889217	yes	3

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Petition for review by the Office of Petitions.	1	2
Issue Fee Payment (PTO-85B)	3	3

Warnings:

Information:

2	Fee Worksheet (PTO-875)	fee-info.pdf	35042 ca7971d3d63bfc973e5524d7bc5ef7f56e18aa94	no	2

Warnings:

Information:

Total Files Size (in bytes):	239851

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DICKSTEIN SHAPIRO, LLP
1825 EYE STREET, NW
Washington, DC 20006-5403

MAILED

JUN 06 2011

OFFICE OF PETITIONS

In re Application of :
Guy Larri, et. al. :
Application No. 11/662,704 : DECISION ON PETITION
Filed: March 14, 2007 :
Attorney Docket No. M0025.0369/P369 :

This is a decision on the petition under the unintentional provisions of 37 CFR 1.137(b), filed May 26, 2011, to revive the above-identified application.

The application became abandoned for failure to timely file a reply to the Notice of Allowance and Fee(s) Due mailed February 7, 2011. A Notice of Abandonment was mailed on May 23, 2011.

Since the petition satisfies the requirements of 37 CFR 1.137(b) in that petitioner has supplied (1) the reply in the form of $755 for payment of the issue fee and $300 for payment of the publication fee; (2) the petition fee of $810; and (3) a proper statement of unintentional delay, the petition is **GRANTED**.

Office records reveal that petitioner submitted $755 for payment of the issue fee, $300 for payment of the publication fee and $15 for payment of five advance order soft copies on May 25, 2011 and May 26, 2011. However, since the fees were submitted twice, the duplicate payment processed on May 26, 2011, in the amount of $1,070, will be refunded back to the credit card.

This application file is being referred to the Office of Data Management for further processing into a patent.

Telephone inquiries concerning this decision should be directed to the undersigned at (571) 272-8226.

Andrea Smith
Petitions Examiner
Office of Petitions

UNITED STATES PATENT & TRADEMARK OFFICE
Washington, D.C. 20231

REQUEST FOR PATENT FEE REFUND

1 Date of Request:	06/02/11	2 Serial/Patent #	11/662,704

	3 Please refund the following fee(s):	4 PAPER NUMBER	5 DATE FILED	6 AMOUNT
	Filing			$
	Amendment			$
	Extension of Time			$
	Notice of Appeal/Appeal			$
	Petition			$
X	Issue (ISSUE FEE $755 & PUB FEE $300)	WFEE	05/25/11	$ 1,055.00
	Cert of Correction/Terminal Disc.			$
	Maintenance			$
	Assignment			$
X	Other (5 ADVANCED ORDER SOFT COPIES)	WFEE	05/25/11	$ 15.00

7 TOTAL AMOUNT OF REFUND: $ 1,070.00

8 TO BE REFUNDED BY: CREDIT CARD

	10 REASON:
	Overpayment
X	Duplicate Payment
	No Fee Due (Explanation):

	Treasury Check
	Credit Deposit A/C #:

9 [ ][ ]--[ ][ ][ ][ ]

11 REFUND REQUESTED BY:

TYPED/PRINTED NAME: Andrea M. Smith TITLE: Petitions Examiner

SIGNATURE: /Andrea M. Smith/ PHONE: 2-3226

OFFICE: Office of Petitions

****

THIS SPACE RESERVED FOR FINANCE USE ONLY:

APPROVED: [signature] DATE: 06/07/11

Instructions for completion of this form appear on the back. After completion, attach white and yellow copies to the official file and mail or hand-carry to:

Office of Finance
Refund Branch
Crystal Park One, Room 802B

FORM PTO 1577
(01/90)

PART B -FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571) 273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403

OPAP MAY 25 2011 PATENT & TRADEMARK OFFICE

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	03/14/2007	Guy Larri	M0025.0369/P369	2632

TITLE OF INVENTION: A SPEECH RECOGNITION CIRCUIT AND METHOD

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
Non-Provisional	yes	$755.00	$300.00	$1,070.00	05/09/2011

EXAMINER	ART UNIT	CLASS-SUBCLASS
S. I. McFadden	2626	

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list
(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Dickstein Shapiro LLP
2
3

Credit Card Refund Total: $1070.00

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE	(B) RESIDENCE: (CITY and STATE OR COUNTRY)
Zentian Limited	Cambridge, United Kingdom

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are enclosed:
- ☒ Issue Fee
- ☒ Publication Fee (No small entity discount permitted)
- ☒ Advance Order -# of Copies 5

4b. Payment of Fee(s):
- ☐ A check in the amount of the fee(s) is enclosed.
- ☒ Payment by credit card. Form PTO-2038 is attached.
- ☐ The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 04-1073

5. Change in Entity Status (from status indicated above)
- ☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- ☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature Stephen A. Soffen Date May 25, 2011
Typed or printed name Stephen A. Soffen Registration No. 31,063

DSMDB-2928679v1PTOL-85 (Rev. 08/08) Approved for use through 08/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

The PTO did not receive the following listed item(s) Credit Card

05/26/2011 MBELETE2 00000041 041073
01 FC:2501 755.00 DA
02 FC:1504 300.00 DA
03 FC:8001 15.00 DA

Amazon / Zentian Limited
Exhibit 1002
Page 475

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/662,704	07/12/2011	7979277	M0025.0369/P369	2632

24998 7590 06/22/2011

DICKSTEIN SHAPIRO LLP
1825 EYE STREET NW
Washington, DC 20006-5403

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 1043 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Guy Larri, Cambridge, UNITED KINGDOM;
Mark Catchpole, Cambridge, UNITED KINGDOM;
Damian Kelly Harris-Dowsett, Coventry, UNITED KINGDOM;
Timothy Brian Reynolds, Cambridge, UNITED KINGDOM;

AO 120 (Rev. 08/10)

TO:	REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK
Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court for the Western District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
6:22-cv-00123	2/2/2022	for the Western District of Texas

PLAINTIFF	DEFENDANT
Zentian Ltd.	Amazon.com, Inc.

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1	7,979,277	7/12/2011	Zentian Ltd.
2	7,587,319	9/8/2009	Zentian Ltd.
3	10,971,140	4/6/2021	Zentian Ltd.
4	10,062,377	8/28/2018	Zentian Ltd.
5	10,839,789	11/17/2020	Zentian Ltd.

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading		
	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1			
2			
3			
4			
5			

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

AO 120 (Rev. 08/10)

TO:	REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK
Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court for the Western District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
6:22-cv-00122	2/2/2022	for the Western District of Texas

PLAINTIFF	DEFENDANT
Zentian Ltd.	Apple, Inc.

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1	7,979,277	7/12/2011	Zentian Ltd.
2	7,587,319	9/8/2009	Zentian Ltd.
3	10,971,140	4/6/2021	Zentian Ltd.
4	10,062,377	8/28/2018	Zentian Ltd.
5	10,839,789	11/17/2020	Zentian Ltd.

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1		
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy