

(21) Application No: 0413862.4

(22) Date of Filing: 21.06.2004

(71) Applicant(s):
ClearSpeed Technology PLC
(Incorporated in the United Kingdom)
3110 Great Western Court,
Hunts Ground Road, Stoke Gifford,
BRISTOL, BS34 8HP, United Kingdom

(72) Inventor(s):
Ken Cameron

(74) Agent and/or Address for Service:
Haseltine Lake & Co
Imperial House, 15-19 Kingsway,
LONDON, WC2B 6UD, United Kingdom

(51) INT CL⁷:
H04N 7/36

(52) UK CL (Edition X):
H4F FRP

(56) Documents Cited:
EP 0220110 A2 **US 5929939 A**
US 5448310 A

(58) Field of Search:
UK CL (Edition W) **H4F**
INT CL⁷ **H04N**
Other:

(54) Abstract Title: **Comparing data from sets at same positions**

(57) A function between a first set of data (a to h etc) and a succession of second sets of data (A to H etc) from an image frame is evaluated. Evaluating means (75, 76) for evaluating data items in the first set with data items in corresponding positions in a succession of second data sets; characterised by a plurality of evaluating means for simultaneously evaluating a data item in the first data set with a corresponding data item in each of n second data sets in the succession, and local memory means (71 to 73) for retaining at least one of the corresponding data items in a second data set, whereby the retained data item is reused in an evaluation of the next data item in the first data set with n data items in corresponding positions in at least the next second data set in the succession. The method avoids having to re-fetch the same data for successive operations. The results are accumulated (77, 78). The technique may perform a convolution between the data sets to emulate a filter or perform an effect (eg a blur or sharpen) or may be used as a video compression process to search for a block of image data in an image frame so as to encode a subsequent frame by reference to the closeness of the match and the offset of the matching block.

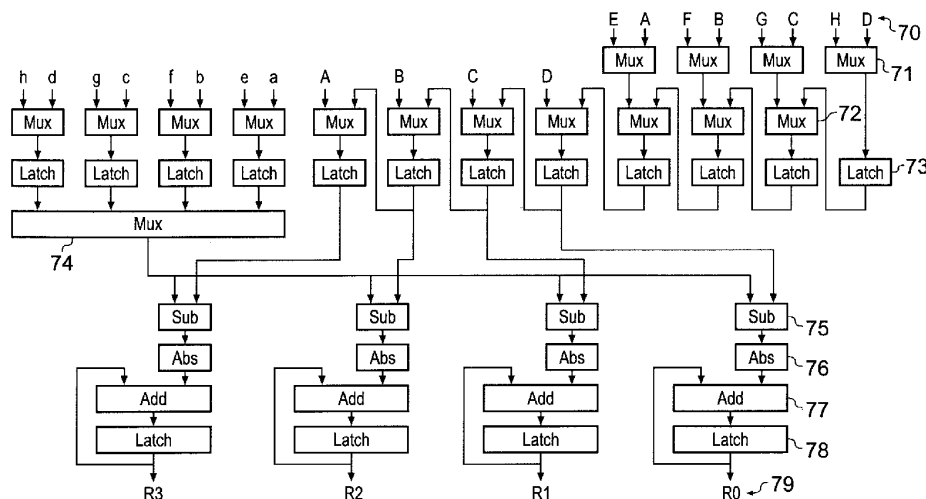


Fig. 8

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

The print reflects an assignment of the application under the provisions of Section 30 of the Patents Act 1977.

Original Printed on Recycled Paper

1/10

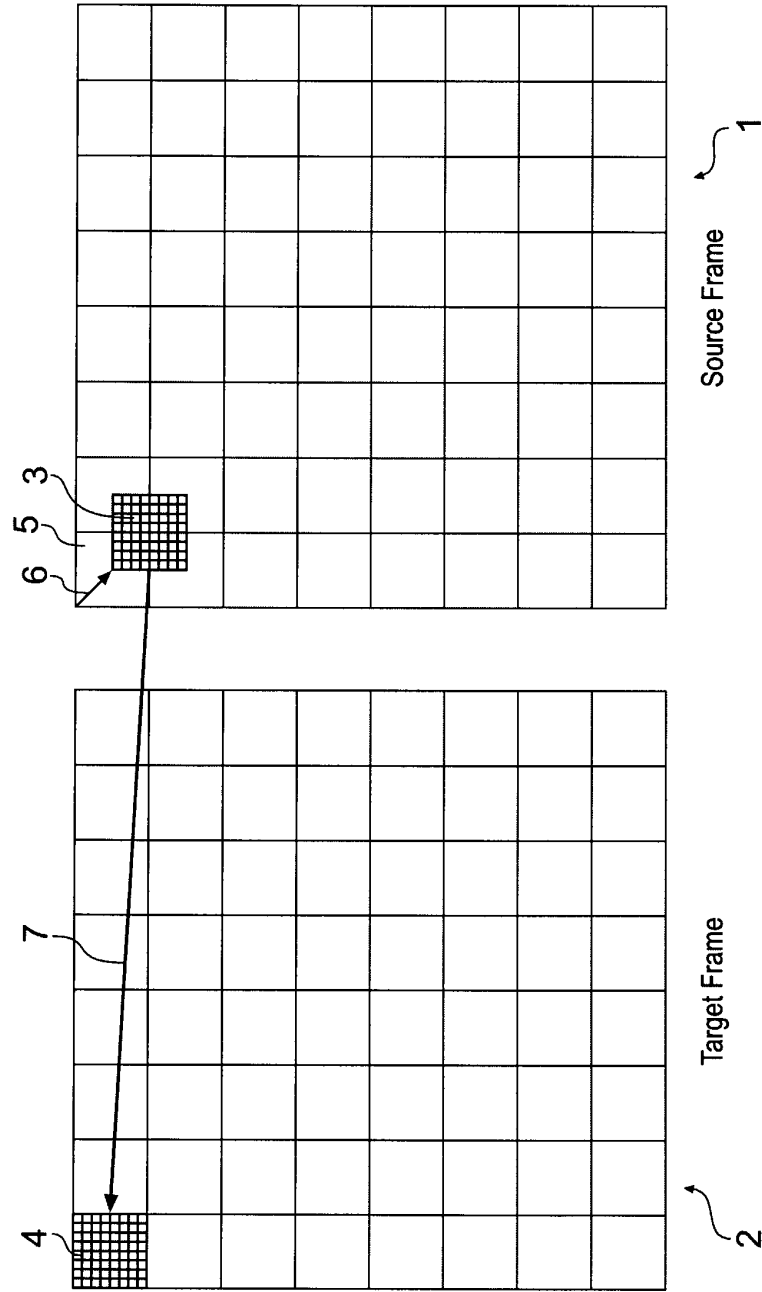


Fig. 1

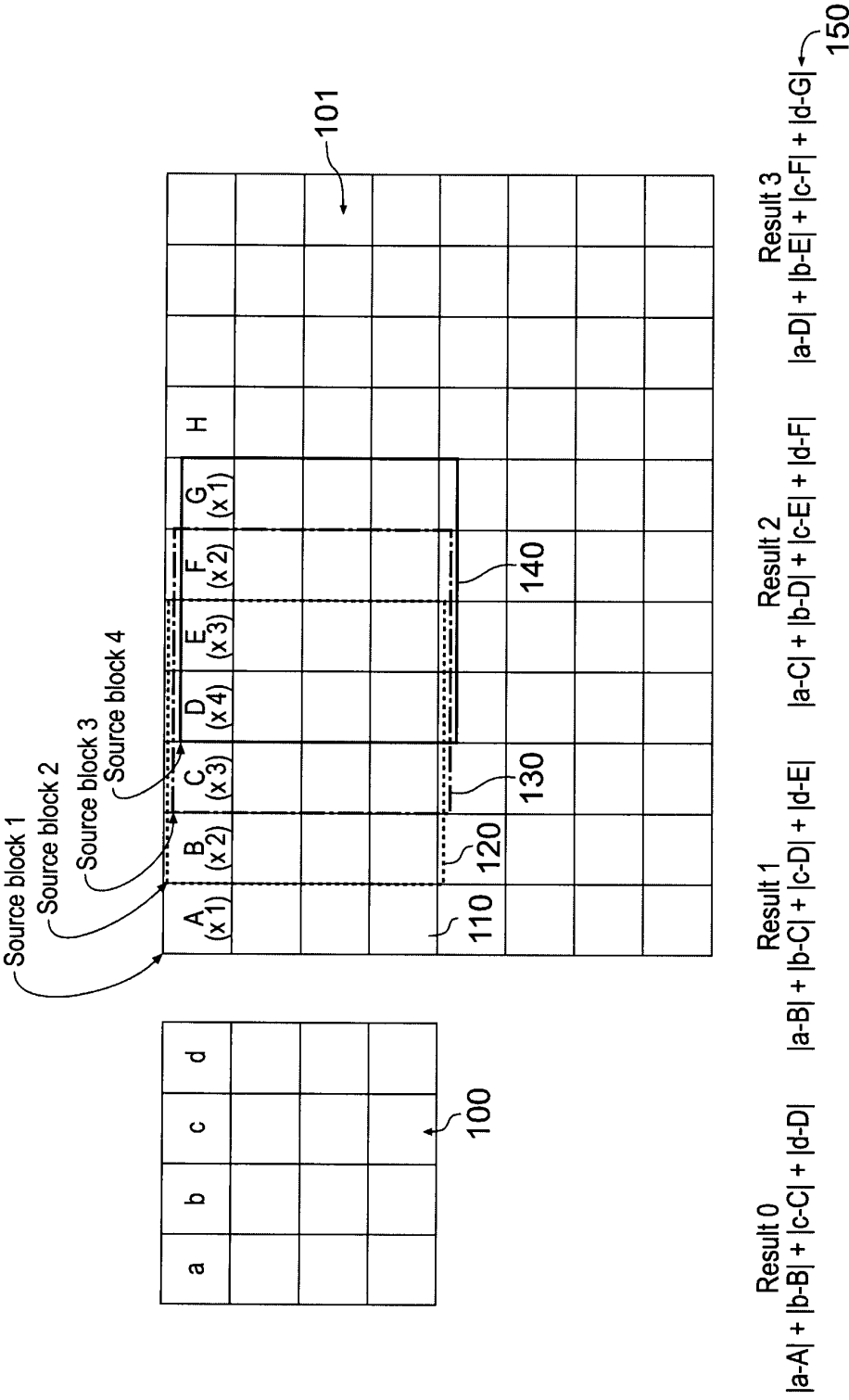


Fig. 2

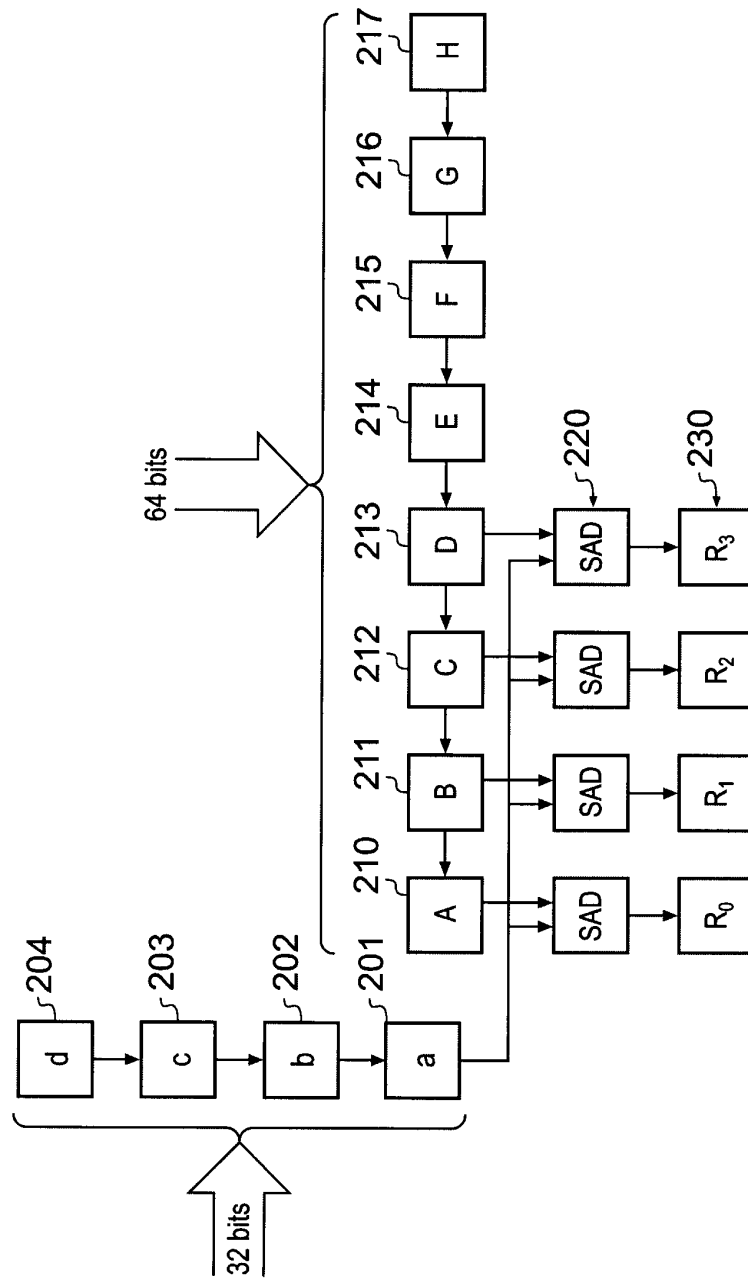


Fig. 3

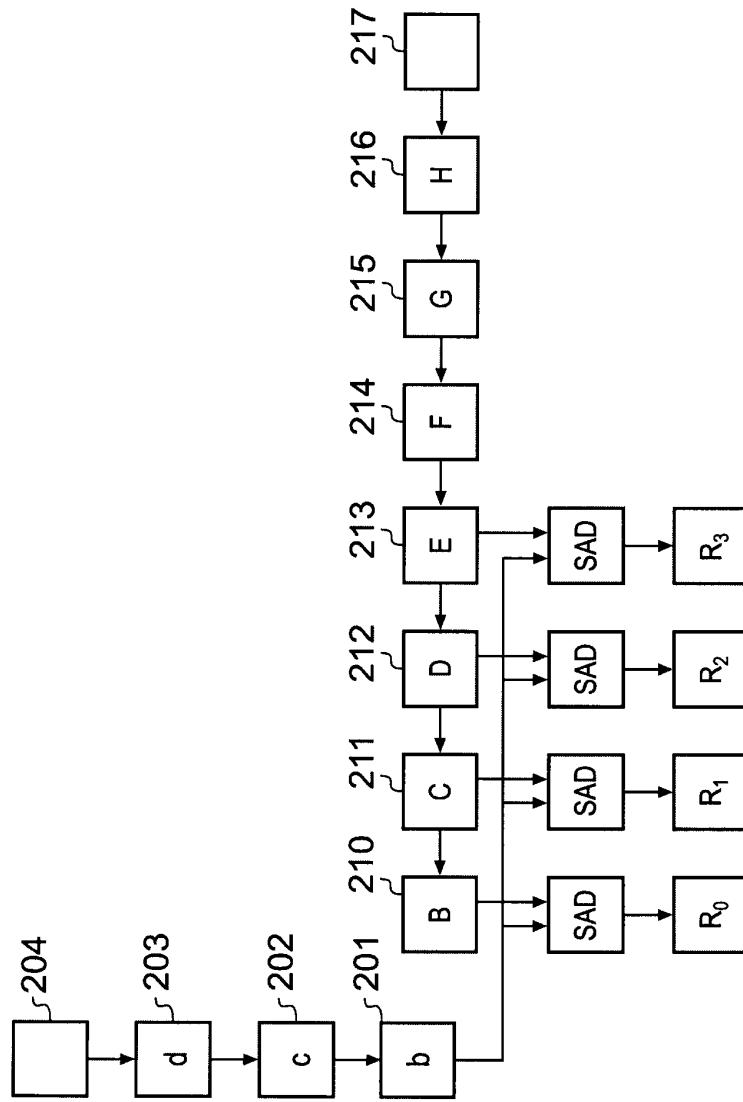


Fig. 4

5/10

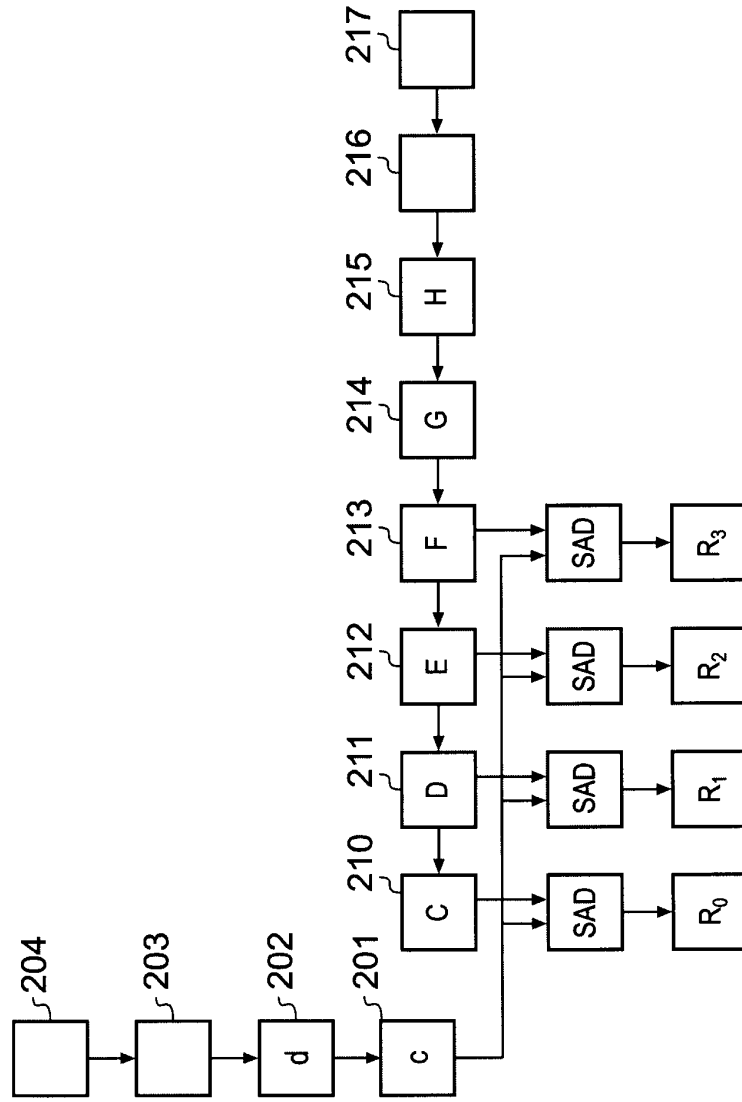


Fig. 5

6/10

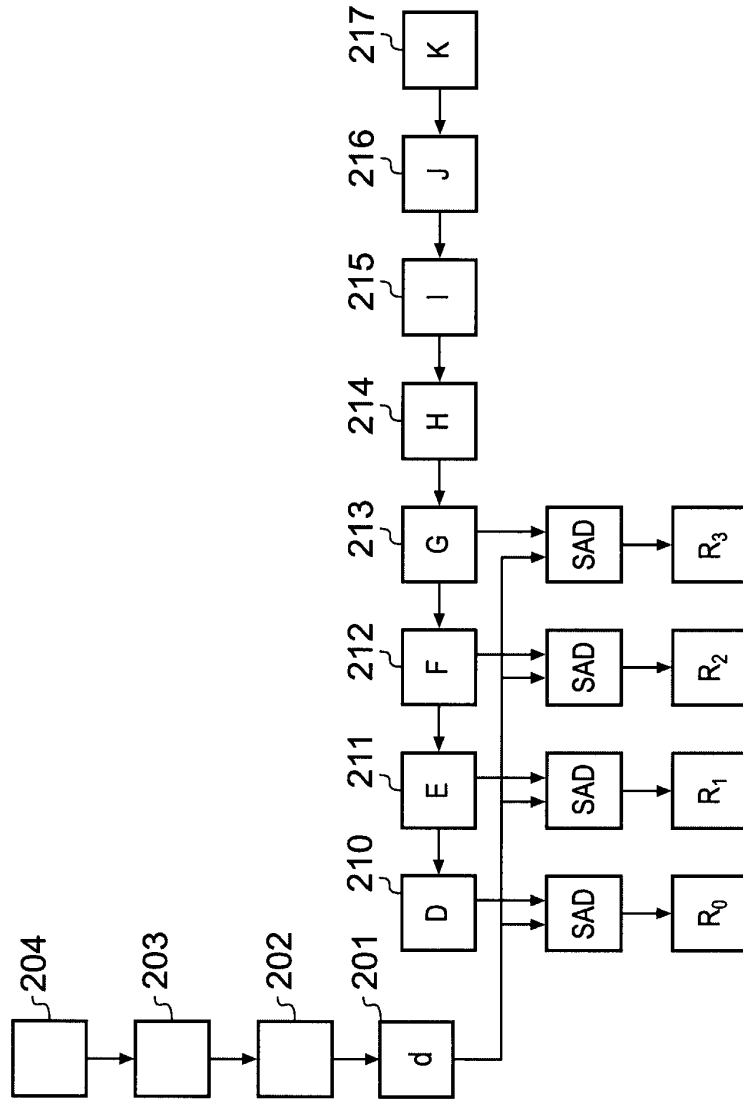


Fig. 6

7/10

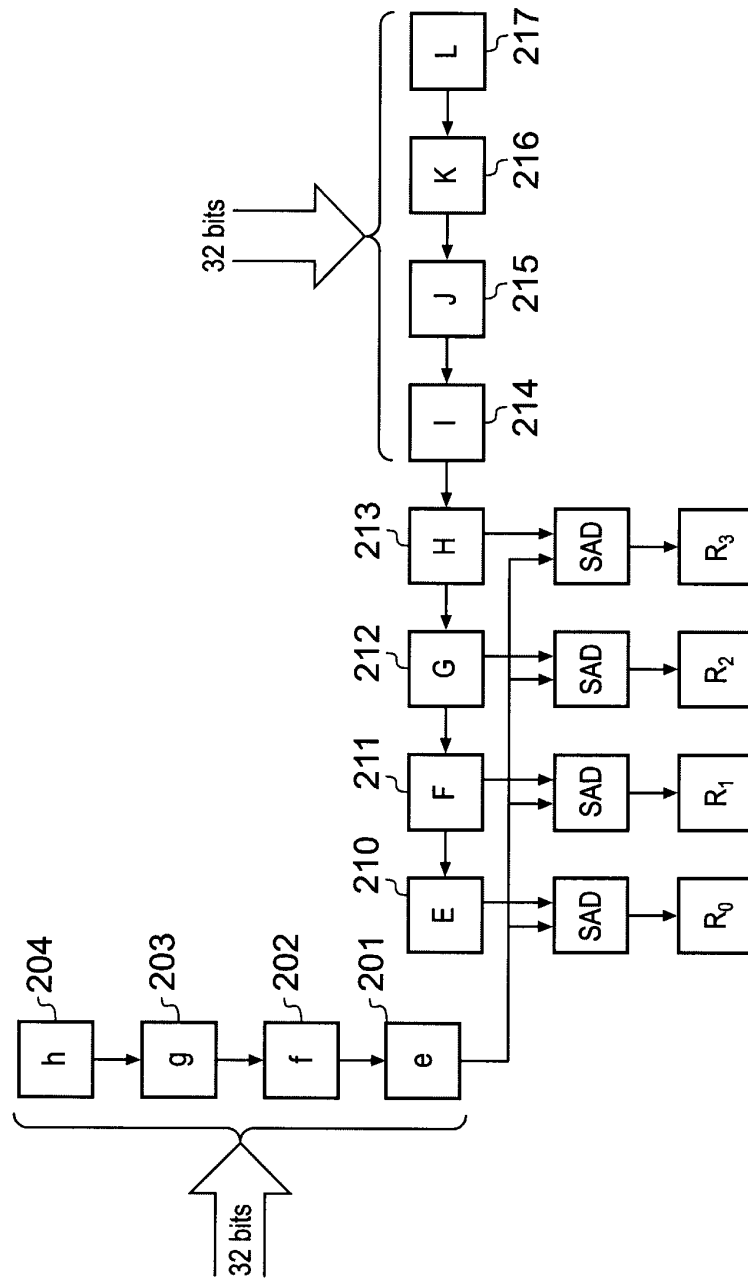


Fig. 7

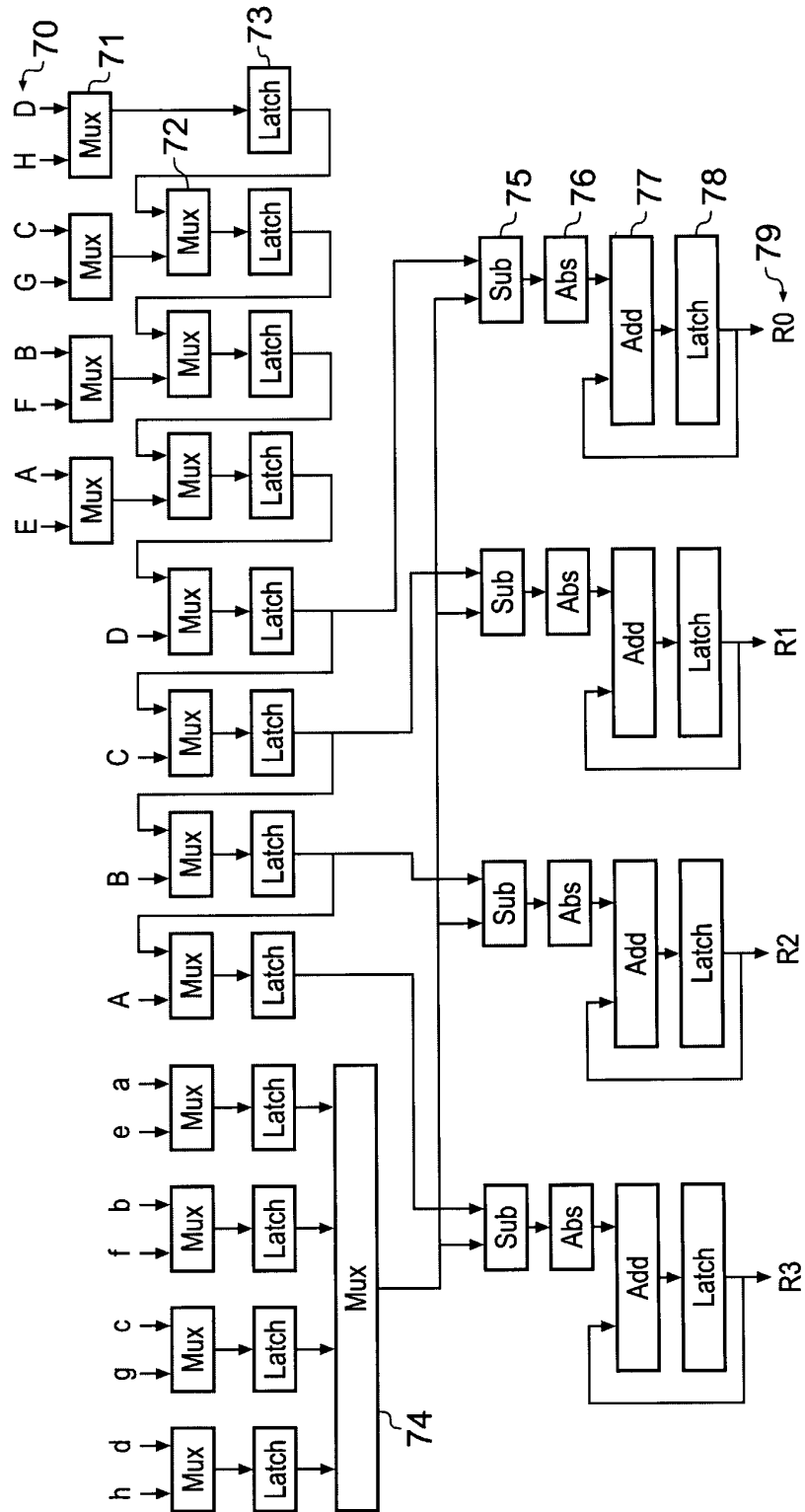


Fig. 8

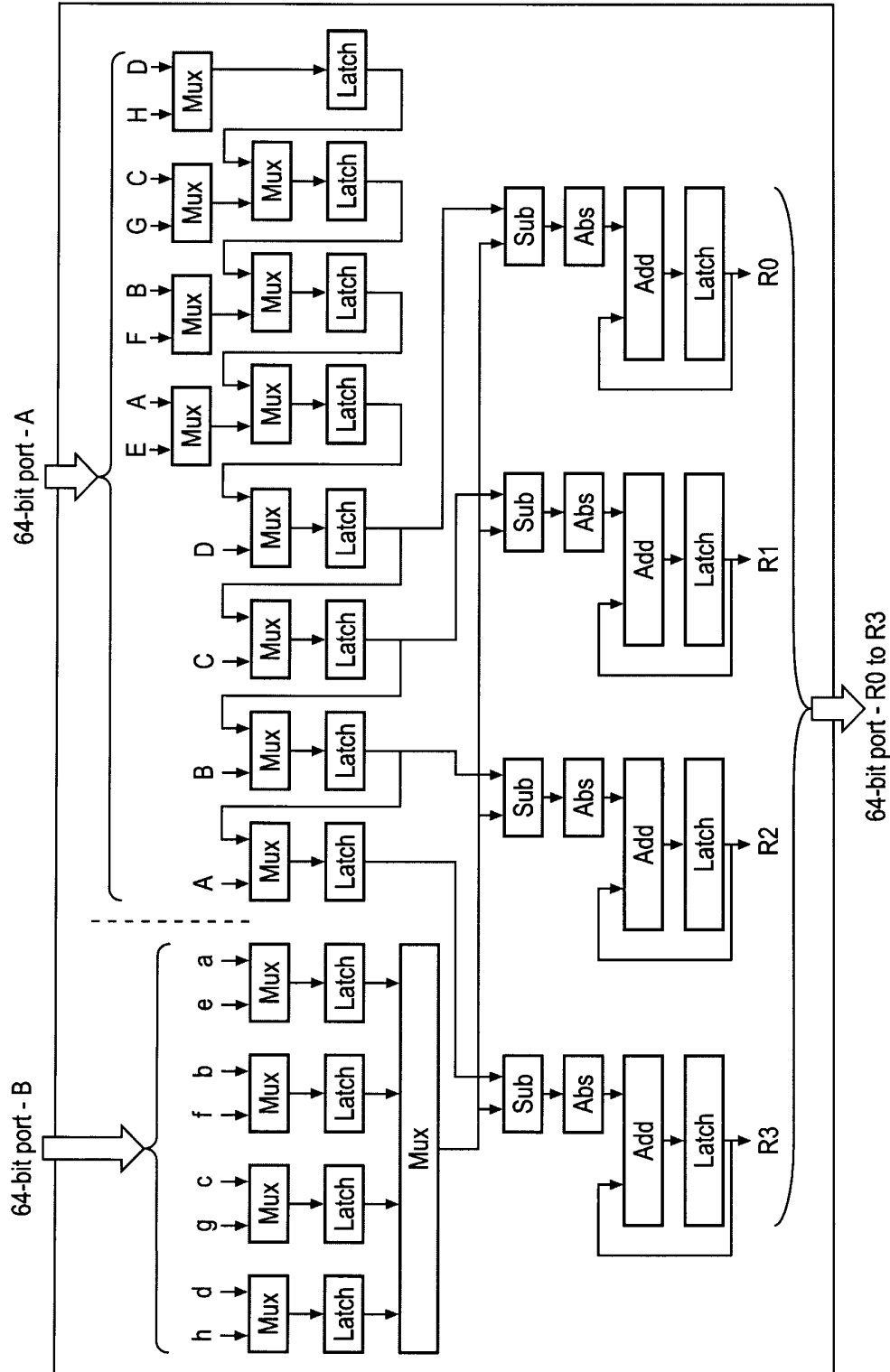


Fig. 9

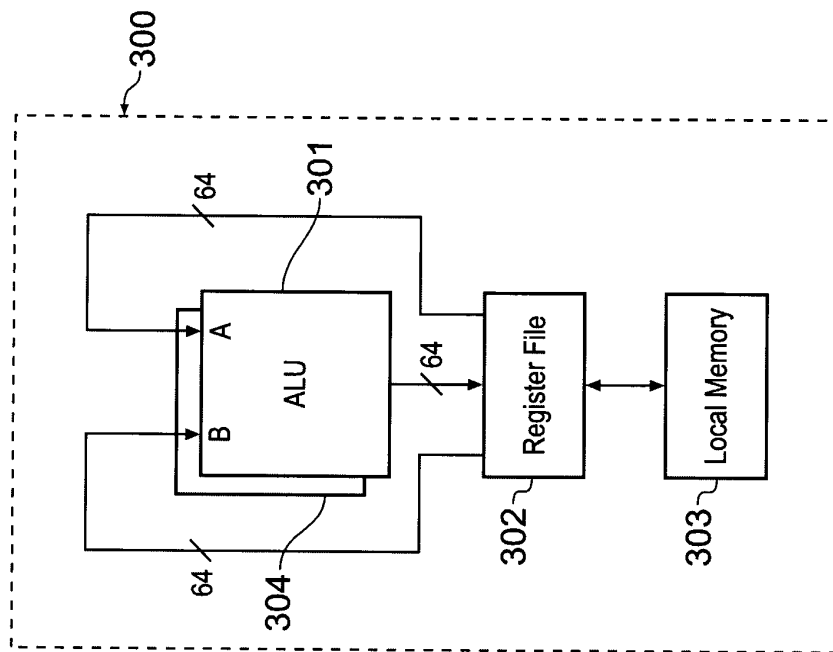


Fig. 10

DATA PROCESSING

Introduction

The present invention relates to a method, apparatus and system for processing image data, especially but not solely video data forming part of a video transmission or broadcast. Such processing may be to conserve bandwidth in a video transmission or to evaluate a function, such as a convolution.

In the discussion that follows, the invention is mostly described in the context of image (video) compression but the principles involved are equally applicable to more general data processing, where two sets of data items are processed to perform or evaluate a particular function. For the purposes of this specification, the major embodiment is a mechanism for searching for the appearance in an image frame of a block of data items (eg pixels) from another image frame as a precursor to real-time video compression by any one of a number of techniques known per se. Another embodiment utilises the same hardware to evaluate a function between a first set of data and a succession of sets of data from an image frame. In this embodiment, a convolution can be performed between the sets of data so as to emulate a filter or other image processing function such as a blur, sharpen, fade etc.

First, the current state of the art as regards video compression will be outlined.

The Current State of the Art in Video Compression

A known technique for compressing video data in order to conserve bandwidth is to compare successive frames of a video stream in order to identify similarities between them. Where similarities exist, the amount of data that needs to be transmitted can be reduced. In the following discussion, two successive frames in a video sequence will be considered but, in practice, it is common to apply the technique across multiple frames and to work backwards as well as forwards in the video stream.

Because each of the frames is usually similar, ie there is a high level of coherence, a compression technique can be used where only differences are sent.

In this discussion, all of the known techniques separate colour images into their components. Although this could theoretically apply to RGB components, the artefacts are unlikely to be acceptable. Therefore, it is preferable to divide a video signal into YUV components, using one luminance and two chrominance (chroma) components. The compression technique is then applied independently to each of these components or channels. When applied to YUV components, artefacts are less likely to be objectionable.

since they will typically be restricted to luminance artefacts or colour artefacts but not a colour channel artefact.

One known compression technique is JPEG. It has several variations. A still image can be divided into blocks and each block compressed using JPEG. Similarly, a sequence of still images can be compressed using JPEG for each frame, in which case the overall compression process is known as Motion JPEG (MJPEG). For real motion picture compression, the MPEG technique is used, in which compression is applied to every n th frame without reference to any of the other frames in the stream. With advances in technology, better versions of such independent compression become available

This technique gives images that can be decompressed completely independently at intervals. It is then necessary to generate all the intermediate frames, for example by reference to the independent frames. A simple way is to compare one frame with the next. There is less information because generally there will have been fewer changes between successive images. Utilisation of this reduced amount of information, eg by transmitting only the changes, will clearly achieve data compression. What is actually done instead of one-to-one comparison, for example by comparing top left pixel in one frame with top left pixel in the next and so on, is to take a region of the image that is to be compressed, for example a block of 8×8 or 16×16 or 8×16 or 16×8 pixels and, for each of these blocks, to search the previous image to find somewhere in that image that contains that region. If that region can actually be found, a greatly improved level of compression can be achieved since that whole block can be represented by a pair of co-ordinates defining its location or offset rather than anything about the block per se. That block can then be reproduced on decompression by copying it to the particular offset relative to the previous image. The process is repeated for the next block which, in all probability, is likely to be fairly close. The process effectively takes a window from the new frame and attempts to find it somewhere in the previous frame.

The reason why this works is because normally, in a series of video images, objects are moving relatively slowly. In all probability the whole image has simply shifted slightly, so the block may simply have moved a small number of pixels in one direction. In the case of a background it is unlikely to have changed from frame to frame. This will be true of all the pixels making up the background. On the other hand, in the case of a moving object, such as a vehicle, the block will be offset and there will not always be an exact match. If the match is so far out that no comparison can be found in

the original frame, for example when a scene changes completely or a new object appears in a scene, it is then possible to consider the block as an independent block to compress on its own without reference to any preceding block. As a half way house, a search can be made to find the best match and, if it is within an acceptable tolerance, that match can
5 be used as a basis for noting the differences and those differences will then use a much smaller amount of data. This is all part of the known MPEG technique.

One particular technique involves the use of a motion estimation algorithm to compare neighbouring “macroblocks” of 16x16 pixel elements using sum of absolute differences calculations. The motion estimation algorithm can be accelerated by creating
10 acceleration logic to calculate the sum of absolute differences across many macroblocks simultaneously. A 128-bit data path for a sum of absolute differences accelerator could include streams of 16 pixels from first and second frames entering a difference block to calculate the differences, a block for calculating the absolute value of the differences, and a summing block that adds the absolute values of these differences to the previously
15 calculated total of absolute values held in an accumulator.

However, performance is likely to be hindered by the limitations of the relatively narrow interfaces found on fixed processors. In the absence of additional logic to maximise data throughput across the processor interface, bus bandwidth may easily become the performance bottleneck for the entire operation. In that context, the main
20 challenge is the design of hardware to keep the accelerator working at proper capacity. Hardware accelerators with built-in DMA controllers and RAMs are often used to mitigate this problem.

Nevertheless, a mechanism is still needed to allocate a “score” to a match, representing the closeness or quality of match. That needs to be done for every block in
25 the new image. There are at least two recognised techniques for doing this: one involves the sum of absolute differences as above, the other uses the sum of the squares of the differences. The latter is generally accepted as giving better compression but requires a much larger amount of computation. The differences are those between corresponding pixels in the respective blocks of the previous and the current image

30 In a 16 x 16 block, with 256 pixels, it is necessary to subtract corresponding pixels one from the other, square the difference then keep a running total. This involves 256 subtractions, 256 squarings and a final addition. For a 16 x 16 block this means 511 additions and 256 multiplications per block. Where this is to be done in real time, it

clearly requires a lot of computing power so it is usual to use the less effective sum of absolute differences. The scoring function is not as accurate with this approach but it can be close enough. It has the added advantage of not requiring multipliers. Of course all this happens in real time because the technique is applied to real time video. Whereas
5 broadcast video compression requires vast racks of expensive equipment, lower resolution compression can be achieved with a reasonably fast PC fitted with an appropriate card.

The highest quality is achieved when each frame is dealt with independently without having to interpolate or invent any data in a frame. Quality can be selected at the cost of compression ratio. This technique does not take advantage of frame to frame
10 coherence so there is a fairly high bandwidth for the quality, for example about 1 Mb/sec for a straightforward Motion JPEG of ordinary TV quality. The same perceived quality can be achieved for 1/10th of the bandwidth when interpolation between frames is used. It is therefore desirable to use better compression methods.

In a known interpolation technique, some frames (I-frames) are compressed
15 individually, others (P-frames) can be interpolated between I-frames, and further frames (B-frames) can be interpolated between the P-frames and the I-frames but they do rely on having information from both forward and backward directions. A search of only the previous frame for regions that look similar would not be undertaken. Rather, the next frame would be searched as well. The larger the number of I-frames transmitted, the
20 higher the data rate. Conversely, the more P-frames and B-frames transmitted, the lower the data rate. The more 16 x 16 blocks for which a perfect (or a very good) match can be found, the less additional data will need to be sent to describe that 16 x 16 block.

For the sake of simplicity in explaining the operation of our technique and those represented in the prior art, we will assume a frame size of 64 x 64 pixels. This is of
25 course unrealistic in practice but makes the following discussion easier to envisage. The frame can be notionally divided into sixty four regions of 8 x 8 pixels, each requiring 64 bytes to represent it. When this block is overlaid on another 64 x 64 pixel image, there are 3,249 possible positions at which a match may be found. If a perfect match were actually found, only the co-ordinates of the block need be sent, plus an indication that the
30 block is identical. The further away from a perfect match, the greater the amount of digital data required to describe the difference between them. This will be discussed further with reference to Figure 1.

Currently, others skilled in the art tend to limit their search to a 64 x 64 (or thereabouts) sized block around the location where the previous block was situated. Motion vectors can then be used to advantage. For example, if there is a block that appears to be moving across a frame in a certain direction and if allowance is made for that movement rather than concentrating on the centre, there is more chance of finding that block in the next frame.

The larger the search area, the more chance there is of finding a better scoring match. The ultimate would be to search the entire frame. That implies three additions for every 16 x 16 block of one frame at every possible pixel offset of the other frame. Bearing in mind that in High Definition TV there are 2million pixels, the number of calculations needed per frame at 60Hz (allowing for searching forwards and backwards) becomes unmanageable – needing something like 80,000 adders running at 250MHz.

Nevertheless, some systems have been built that incorporate large numbers of adders. These typically use a mixture of FPGAs and ASICs. A single chip may carry out each of the 256 absolute difference calculations. A tree is used to summate all the calculations. An entire 16 x 16 block can be searched per cycle but this still requires tens or hundreds of chips to achieve full frame rate. Part of the problem is the amount of data needed to feed these devices. Each absolute difference typically requires 2 bytes of information. This approach implies billions of bytes per frame.

There is a trade off with this technique in that the deeper a search can be carried out, the more chance there is of getting a better match and therefore a lower bandwidth with either no loss of quality or potentially better quality. On the other hand, the larger the search area, the longer the compute time but there is only a finite amount of compute capability in a given machine. A balance therefore needs to be struck between acceptable rendition and compute speed and/or capability.

Prior attempts at dealing with this problem have simply involved a rather “brute force” solution of bolting on hardware as an accelerator to a processor and hoping for the best. More refined solutions involve making some of the operations, such as calculating the sum of absolute differences, available as a special SIMD instruction in otherwise “standard” processors. Some such special processors presented with 8- or 16-byte data can calculate the absolute differences and accumulate the calculations. At present, there are no processors that have the capability to perform the whole operation, leaving the complete functionality to be built up in software.

Our approach as far as video data processing is concerned is to partition the problem differently and solve some of the bandwidth issues, as will be described shortly.

The Current State of the Art in Function Evaluation

Functions between variables occur in many and varied applications. Often, it is
5 necessary to evaluate a function between two sets of data. These sets may involve blocks of data. If one of these sets of data represents an image, data processing may involve taking a succession of blocks of data from the image set and performing a particular process by convolution with another data set. This other data set may be a block from another image or image frame or may be a set of predetermined data designed to achieve
10 an intended effect when convolved with the blocks of data in the image set.

For example, the predetermined data set may be a pre-loaded set of values which, when convolved with the blocks of data from another data set, perform a filter function. Likewise, the pre-loaded data set may be convolved with a data set from an image frame to blur or sharpen the image. It is also possible that the first data set is derived from a first
15 image and the other data set is a succession of blocks from another image, the effect of convolution between the two sets emulating a fade or "special effects" function, marking a transition between successive images in a succession of images.

The current state of the art, like the video compression scenario discussed above, is heavy on compute hardware and/or slow on speed. On the other hand, the present
20 invention can outperform prior art processing several times over.

The present invention achieves significant advantages over the prior art by adopting a radical approach that permits multiple strands of data to be processed concurrently without the need to fetch each item of data as it is required. Rather, once fetched, each data item is held in local memory and re-used several times over. The
25 overall result is greatly increased processing speed.

The essential features of the invention in both of the aspects outlined above will become apparent from the following description.

Summary of the Invention

In a first aspect, the invention provides a method for evaluating a function between
30 a first set of data and a succession of second sets of data from an image frame, wherein each data item in the first data set is evaluated with data items in corresponding positions in the second sets of data, characterised in that a data item in the first block is evaluated simultaneously with a corresponding data item in each of n second data sets in said

succession and wherein at least one of said corresponding data items in a said second data set is/are locally stored and reused in an evaluation of the next data item in said first data set with data items in corresponding positions in at least the next second data set in the succession.

5 In a second aspect, the invention provides a method for searching in a second frame of image data for a first block of image data appearing in a first frame, wherein data items in the first block are compared with data items in corresponding positions in a succession of second blocks in the second frame, characterised in that a data item in the first block is compared simultaneously with a corresponding data item in each of n second
10 blocks in said succession and wherein at least one of said corresponding data items in a said second block is/are locally stored and reused in a comparison of the next data item in said first block with data items in corresponding positions in at least the next second block in the succession.

 In a third aspect, the invention provides apparatus for evaluating a function
15 between a first set of data and a succession of second sets of data from an image frame, comprising evaluating means for evaluating data items in the first set with data items in corresponding positions in a succession of second data sets; characterised by a plurality of evaluating means for simultaneously evaluating a data item in the first data set with a corresponding data item in each of n second data sets in said succession, and local
20 memory means for retaining at least one of said corresponding data items in a said second data set, whereby said retained data item is reused in an evaluation of the next data item in said first data set with n data items in corresponding positions in at least the next second data set in the succession.

 In a fourth aspect, the invention provides apparatus for searching in a second
25 frame of image data for a first block of image data appearing in a first frame, comprising comparing means for comparing data items in the first block with data items in corresponding positions in a succession of second blocks in the second frame, characterised by a plurality of comparison means for comparing n data items in the first block simultaneously with n data items in corresponding positions in n second blocks in
30 said succession, local memory means for retaining at least one of said n data items in a second block, whereby said retained data item can be reused in a comparison of the next data item in said first block with n data items in corresponding positions in at least the next second block in the succession.

In a fifth aspect, the invention provides a data processor for processing first and second sets of data, comprising an array of processing elements each in combination with an accelerator, the accelerator comprising apparatus as specified in the preceding paragraphs.

5 In a sixth aspect, the invention provides a method of compressing image data by encoding a first image frame by reference to a second image frame, the method comprising the steps of: searching in a succession of second blocks in said second image frame for a first block of image data appearing in the first image frame; and encoding the first block as a function of the closeness of the match between the first block and a
10 matching second block and the offset of the first block relative to said matching second block in said second frame; wherein said searching step comprises: comparing data items in the first block with data items in corresponding positions in a succession of second blocks in the second frame, characterised in that a data item in the first block is compared simultaneously with a corresponding data item in each of n second blocks in said
15 succession and wherein at least one of said corresponding data items in a said second block is/are locally stored and reused in a comparison of the next data item in said first block with data items in corresponding positions in at least the next second block in the succession.

 The invention also encompasses a computer program product comprising a
20 medium on which is stored a computer program adapted to perform the steps of the method as specified in the preceding paragraphs.

 The apparatus is preferably connected to a processing element to act as an accelerator. In the case of a processor having an array of processing elements, an accelerator may be connected to each processing element or the accelerators themselves
25 may be configured to act as the processing elements in an array. The array is preferably a Single Instruction Multiple Data (SIMD) array.

 The evaluation may consist of a convolution, in which case the method and apparatus may emulate a filter or may perform a particular function, such as a blur or sharpen, on image data. Where the first data set is a predetermined data set and the
30 succession of data sets are image data, the process may generate a third set of data, such as an image frame exhibiting a particular effect.

Brief Description of the Drawings

The invention will now be described with reference to the following drawings, in which:

Figure 1 is a schematic diagram indicating how a "target" image frame can be encoded by reference to "source" image frame using an 8 x 8 pixel window;

Figure 2 is a simplified illustration of the mechanism involved in the invention;

Figures 3 to 7 represent a sequence of operations carried out in performing a preferred embodiment of the invention in relation to image compression and as an aid to understanding how the invention operates;

Figure 8 is a schematic representation of the processing blocks used in a preferred implementation of the invention;

Figure 9 illustrates how a unit embodying the invention can be connected to a processor array; and

Figure 10 illustrates a basic processor element for the purpose of explanation.

Detailed Description of the Illustrated Embodiments

Video image processing

Consider first a processing element (PE), such as illustrated generically at 300 in Figure 10. The PE is preferably provided with a selected functionality represented by an arithmetic unit (ALU) 301. The PE also includes a local memory 303 and a local register file 302. The connection between the register file and the ALU includes three paths, two to the ALU, entering at ports A and B, and one return.

The PE can be used to implement a particular functionality, such as a floating point arithmetic. This function block, represented by 304 in Figure 10, is connected in parallel to the ALU. In the same way, a sum of absolute differences (SAD) unit could be connected in parallel with the ALU to perform a SAD function as a precursor to compression, as will be explained later.

As will become clearer, the present invention achieves a technique for re-using data by carrying out calculations in parallel paths and thereby using multiple adders to compute different values rather than using multiple adders combining to produce one result. Four such parallel paths are used in the preferred embodiment.

In the preferred embodiment of the PE, the paths to and from the register file are 64 bits wide. This provides sufficient bandwidth to the SAD unit, so that the datapaths do

not need to transfer data to it on every cycle. This allows the remaining register file bandwidth to be used by other functional units, such as the ALU.

Consider now Figure 1, which illustrates the objective of searching for a block of image data from one frame in a different frame, which may be a previous frame in a succession of frames as in the case of a stream of video image frames. The “target” frame 2 in Figure 1 is the frame of interest, ie the frame that is to be compressed. This frame is notionally divided into 64 blocks (say), each of 8x8 pixels, as previously mentioned. Compression of the target frame is by reference to a “source” frame 1. This may be the previous frame in a video stream or sequence, for example.

Each of the blocks in the target frame 2 is to be searched for in the source frame 1 by considering it to be a “window” that is moved over the target frame until a “match” is found. The moving and matching operation begins with the first block 4 or window from the top left corner in the target frame being located over the top left block 5 in the source frame 1. Eventually the window 4 visits each offset in the source frame, looking for a perfect match or a best match, if one exists. If a perfect match can be found the target block can simply be encoded by flagging it as identical to the matching source block 3 and providing data as to the offset of that block 3 relative to the starting position 5 of the block being searched for in the source frame. The offset is represented by arrow 6 in Figure 1. The arrow 7 merely indicates that the matching block 3 found in the source frame 1 is used to encode the target block 4 in the target frame 2.

It is unlikely that a perfect match will be found, so the technique looks for a closest match. Therefore, for each position of the window or block over the source frame a rating or “score” is allocated and the “best” overall score is taken to be the location of the block from the target frame in the source frame 1.

The process will now be explained in more detail with reference to Figure 2. This is an even more simplified illustration of the basic principles underlying the invention in order to enhance explanation without complicating the illustration. For the purposes of illustration, the technique is described with reference to a 4x4 block but the principle is applicable to an 8x8 or 16x16 block etc. It is important to note that the use of a 4x4 block for illustrative purposes has no bearing on the fact that in this example we use four parallel processing paths with four sets of adders processing four blocks at a time. In other words, there is no relationship between the size of window and the number of parallel processing paths.

The objective of the technique is to attempt to find a block 100 from the target frame (in this case a 4x4 block, purely for simplicity in explanation) in a source frame 101. The top row of pixels in the target block 100 is identified as a, b, c and d. Pixels in the top row of the source are identified as A, B, C, D, E, F, G, H etc. Where the target block is 4x4, the pixels in the second row could be labelled e, f, g, h and so on for the third and fourth rows. However, where a block larger than 4x4 is used, the top row of pixels in the source block will continue beyond d into e, f, g, h and so on.

Regarding the target block 100 as a sliding window, it is first positioned in the top left hand corner of the source frame in the position identified as 110. In this position, pixel a overlies pixel A, pixel b overlies pixel B, pixel c overlies pixel C, and pixel d overlies pixel D. In this position, the absolute differences between the corresponding pixels in the target block and each of the successive (in this case four) source blocks are calculated and accumulated in respective accumulators (indicated generally at 150 in Figure 2) for that position. Thus, in one clock cycle, the value $|a-A|$ is calculated for the position where pixel a overlies pixel A in the first of a succession of blocks in the source frame, labelled "source block 1" in Figure 2. In the same clock cycle, since the value B is already available (ie it has been fetched) the value $|a-B|$ can be calculated. Similarly, the values $|a-C|$ and $|a-D|$ can be calculated in the same clock cycle.

Inspection of these values shows that the top left pixel a in the target block has been processed with the top left pixel from each of the first four source blocks 110, 120, 130 and 140. The values thus calculated are put into accumulators not shown in Figure 2 but represented at 150 by Result 0 to Result 3). In the next clock cycle, the second pixel b in the target block can be processed with the second pixels in each of the four source blocks in the source frame. This time, the values of those calculations are $|b-B|$, $|b-C|$, $|b-D|$ and $|b-E|$, where E is the next pixel value from the source frame.

When the first four pixels in the target block have been processed with the corresponding pixels in each of the four source blocks 110 to 140, the partial sum $|a-A| + |b-B| + |c-C| + |d-D|$ will have been calculated and stored as "Result 0" in an accumulator. Similarly, the partial sum $|a-B| + |b-C| + |c-D| + |d-E|$ will have been calculated and accumulated as "Result 1" in Figure 2. The process similarly leads to the accumulation of the partial sums illustrated as "Result 2" and "Result 3" in Figure 2.

In other words, for each position of the window over the source block, in one clock cycle one of the pixels in the target block is compared simultaneously with the

corresponding pixel in a succession of source blocks and the results put into the respective accumulators. In the next clock cycle, the next pixel in the target block is compared with the next corresponding pixel in each of the succession of source blocks in parallel and the partial results accumulated once again. The sequence continues line by line until the whole of the target block has been processed. The target block is then moved four (in this case) pixels to the right, so that the top left pixel, a, is now positioned over the source frame block where the top left pixel value is E. The process continues as before, stepping along the lines of the source block, four pixels at a time (in this example), and then down to the next line, until the whole of the source frame has been searched.

As a variation of the above process, it is not essential that the source blocks are located at adjacent pixels. If they are instead offset by four pixels and only every fourth pixel is searched, the same level of reuse of data from the second blocks is attained but the data can be sub-sampled in order to obtain a quick and approximate match. This may then be followed by homing in for a more precise match position.

What is important to note about each comparison cycle (ie one pixel in the target frame being compared simultaneously with a corresponding pixel in each of the next set of source blocks in succession) is that the value A is used once, value B is used twice, value C is used three times and value D is used four times in this set of calculations. This reuse is represented by the numbers in parentheses under the relevant pixel in Figure 2, for example (x2) under pixel B. The opportunity is thus presented of using the same value more than once without having to fetch it more than once.

Considering the position where the window has been moved from the start position 110 to the end position 140 four pixels to the right, it can be seen that the end pixel values are used once, the next pixels in from the ends are used twice and the next pixels three times. If this is extrapolated to the situation where a whole row of the source frame is scanned, the remaining pixels, which comprise the majority of the pixels in the frame, are each used four times. There is thus a huge potential for re-using data that is already present, ie has already been fetched. It is therefore desirable for the pixel values that are reused to be held in local memory within the processing unit performing the technique until it is finished with, rather than having to fetch each value each time it is to be used

Figures 3 to 7 illustrate in more detail how this saving can be achieved. In Fig 3, the pixel values a-d of the target block are taken in sequence via registers 201-204 of a

shift register to respective inputs of sum of absolute differences (SAD) units 220.

Similarly, via registers 210-213, the values A-D representing the pixel values of the first source block are presented to the other inputs respectively of the SAD units 220. These units therefore calculate the sum of absolute differences between a and A, a and B, a and C, a and D and accumulate the results R0 to R3 in accumulators 230. For the moment, it is sufficient to note that the next set of values E-H of the next block in the source frame spaced four pixels along from the previous block, are introduced at 214-217 awaiting their turn in the next calculation.

Referring now to Figure 4, both sets of pixel values have been stepped to the next value in the sequence. So, value b has moved from 202 in Fig 3 to 201 in Fig 4 to replace value a, value c has moved from 203 to 202 to replace value b, and value d has moved from 204 to 203 to replace value c. Similarly for the pixel values A-D, value B has moved from 211 to 210 to replace value A, value C has moved from 212 to 211 to replace value B, and value D has moved from 213 to 212 to replace value C. New value E has entered at 213 in readiness for the next calculation and the remaining values F-H likewise have moved into 214-216.

Figure 5, 6 and 7 show the continuation of the above process, as the values a, b, c etc and A, B, C etc progress through the SAD units and the calculated values accumulate at R0 to R3. Figure 7 shows the position that has been reached when all four values a to d and A to D have passed through the process and have been replaced by the next set e to h and E to H. In this manner, a complete line of pixels can be scanned and the results of the SAD calculations stored in accumulators R0 to R3. The whole process is then repeated for the next row of pixels in both the target and source frames.

The accumulated values represent an indication or "score" of the closeness of the match between the target block and the source blocks. The algorithm/program operating the compression per se can use these scores to judge whether a perfect match has been found or, if not, where the closest match lies. That block can then be used to encode the target block by reference to the matching source block.

The pixels a, b, c, etc can be 32-bit inputs via an input port. The other input port for the A, B, C etc pixels can be 64-bit to capture two sets of 32-bit data. However, the ability exists to choose, as well as shifting across, to shift 32 out of 64 bits into the other four registers. This is shown in Figure 3, for the 64-bit width input and in Figure 7 for the 32-bit input. If there are more than four samples, eg in an 8x8 or 16x16 block version of

the invention, at every fourth step when the a, b, c, etc values have been shifted through 201-204, a whole new set of four values out of the block being searched for can be shifted into the register. In the 16x16 case, this has to be done four times, starting with a to d, then e to h and so on. The same thing will of course have to be done for the A to D etc
5 values, as shown at 214-217 in Figures 3 and 7, for example.

After all of the lines of the four source blocks have been searched or scanned in this way, the result registers R0-R3 will contain accumulated sums of absolute differences of the entire target block for each of the four possible offsets 110-140 shown in Figure 2, the accumulated results representing the scores for each of those offsets, as previously
10 mentioned. This process is repeated until the whole of the image frame has been scanned and "scored". The scores in the accumulators are utilised by the compression algorithm and then discarded once the entire block has been scanned and the block or window has been moved four pixels along the scan line. For a given 16x16 block at four offsets, the computations are effectively made in parallel using parallel processing paths, one for each
15 SAD unit.

It is useful to pause here and compare how the present invention differs from the current state of the art. It is worth remembering that the block being searched in the source frame is actually part of the target frame that is currently being compressed, eg for transmission by reference to the source frame. The current state of the art is not to search
20 the entire image for that block but instead to search only the region around where the block in question belongs in the frame, eg plus or minus 32 pixels. This is on the basis that, where there is a high degree of coherence between successive frames, the block is likely to be near its previous position. Potentially, better compression is possible by making the search area bigger and bigger, ultimately being the size of the frame itself, but
25 at the cost of time and hardware. However, the prior art has not yet managed to search the entire image in real time.

If a perfect match is found for the target block in the source frame, the target frame can simply copy the matching block from the source frame. If a perfect match cannot be found, one block is effectively subtracted from the other to create a "difference"
30 image that can then be used in the succeeding compression stage, bearing in mind that the differences are likely to be small. The compression itself, eg JPEG, can then be applied to the differences. Most of the difference values will (hopefully) be zero, so encoding needs only to be performed on the blocks that are different. There is a range of encoding

schemes that can be used to represent the differences. If there really is no match at all, eg because of a change of scene, there is very little option but to encode the whole of the new frame as an independent frame without reference to any preceding frame. There are a number of algorithms that can be used to determine how far a search should be carried out in any given frame. For example, a perfect match may lead to further searching being prematurely stopped. Other approaches may be used.

Whereas typical Motion JPEG may take 1Mb/s bandwidth, compression can achieve the same perceived quality in one tenth of that bandwidth if interpolation between frames is employed using better and better compression techniques.

10 The above description of the operation of the technique used shift registers as a means of explaining the process. However, in the preferred implementation shown in Figure 8, latches and multiplexers are used instead. In the left hand side of Figure 8, values a-h of the target window are presented to one of the inputs of a set of multiplexers (Mux) indicated generally at 72. The values can be momentarily held as one or the other by means of the latches 73 connected to the Mux outputs. Similarly, the first four source block pixel values A-D are available at a next group of four Mux circuits 72 and likewise the values A-H (shown at 70 in Figure 8) are available at further Mux circuits 71 feeding Mux circuits 72 and latch 73. The combination of Mux and latch circuits acts in the same way as the shift registers used in the functional description above.

20 Initially, the very first 64 bits of data brought into the processing unit, ie the values A-H at 70, are latched into the registers. They are then shifted down as the calculation progresses. After four such steps, it is necessary to bring in another four bytes at 70. However, there is a 64-bit path, so one or the other half of the input path has to be selected to bring in the next half. In the next step, the other half is brought in. Hence, after four cycles, new values of A-D would be brought in via the Mux array 71. This is probably better illustrated by considering the input values a, b, c etc and A, B, C etc as coming in via ports A and B as shown in Figure 9. The entire 64 bytes can be latched in one step. Thereafter, only four at a time will be latched in.

30 The requisite combination of source and target pixel values are thereby fed to the inputs of subtraction and absolute value circuits 75, 76 and thence to addition and latch circuits 78, 79 performing the necessary accumulation function. The results are accumulated in results registers R0 to R3.

If the input latches were considered as being equivalent to the shift registers described above (so that only one new byte is input at the end) only two new bytes would be needed every cycle instead of eight, so the bandwidth will have been reduced by a factor of 4. The PE array permits 32 bits per cycle to be brought from each PE memory
5 into the units described above but only 16 bits are actually needed because data has been re-used. Although four operations are ongoing at any one time, they are not independent.

The current state of the art tends to compute an entire 16 x 16 or 8 x 8 block in one go at one offset in its entirety before trying another offset, so there is no reuse of data, thereby leading to a bandwidth problem. The justification for this approach is that it is
10 straightforward to bring in two groups of 16 bytes into 16 byte line registers. However, this approach fails to recognise that, by using multiple adders to compute different values rather than using multiple adders combining to produce one result, data can thereby be re-used.

The above technique works equally for any width that is a multiple of four. It will
15 therefore work with 4 x 4; 4 x 8; 8 x 8; 8 x 16 and so on, up to 32 x 32, even though no known technique ventures beyond 16 x 16. The preferred choice of four is arrived at by considering that greater widths do not use any more values (the maximum remains at one byte per cycle) but at initialisation, the data has to be input in parallel. The only shift is along a scan line and when the scan moves down to the next line, the register has to be re-
20 set with the line data. Unless very wide searches are being conducted, eg by using many 16 x 16 or 32 x 32 blocks, the cost of "pre-loading" this data in the first place increases and thereby becomes a bigger portion of the overall performance. It is also beneficial to support down to 4 x 4 block sizes. Four gives the level of desired performance with the minimum and maximum sizes of interest but it could be any number. Taking it down to
25 one means that there is only one unit and data is being fed at one byte per cycle. Some bandwidth is saved using two.

The wider the window, the greater the bandwidth advantage but only when the search is conducted on something that is reasonably wide. There is a trade-off between enhanced performance and silicon area cost on a chip implementing the technique. The
30 best known units can at present only theoretically achieve at best half of the performance our system can achieve but in practice the known unit achieves only half of that, so our system exceeds the best of the prior art by a factor of 4.

The main advantage of the technique is scaled up performance without scaling up bandwidth. The unit in accordance with the invention can be “bolted” onto the side of a processor, a SIMD processor preferably, as an accelerator. Moreover, each of the PEs can be used within the software with a consequential improvement in flexibility regarding the choice of algorithm to direct the search rather than arriving at a best match by a series of approximations or interpolations. There is thus the dual advantage of hardware acceleration and programmability.

The present invention has a wide spectrum of application, from live broadcast TV encoding and streaming video data across the internet for example and especially commercial broadcasting where the reductions in bandwidth that the invention achieves is directly reflected in a corresponding reduction in transmission cost per channel but with no loss of perceived quality. It makes more commercial sense for a broadcaster to make a capital investment in a machine that does a better job of compression without sacrificing image quality than to pay increased costs per unit bandwidth on a continuing basis.

The unit described in the preferred embodiments of the invention exhibits advantages when it is attached to a processor in that it accelerates the functions performed by the processor. The unit could also be attached to each of the PEs of an array processor (such as a traditional multiprocessing environment or preferably a SIMD processor), where each PE has local memory, local ALU and local register file. Alternatively, the processors described in the present specification can themselves be arranged into an array with some local storage/memory for each one, plus any necessary control logic. This would then constitute a stand-alone device that performs only the search operation described without the need for a separate processor per se. If there are enough PEs, the searching operation can be performed in real time.

Considering a SIMD array of 64 or 96 PEs, the present invention enables 64 or 96 blocks to be searched in parallel whereby to gain an even more enhanced performance. Alternatively, the same block can be searched in different places in each PE or the search could be carried out in the same part of the frame (or in different parts of the frame) but with each PE looking for a different block. It may be possible to search for a target block by using only the corner pixels of the target block and of a succession of source blocks. Other variations are also possible. However, the technique of the present invention has advantages in each application.

Although the aspect of the present invention concerned with image frame processing has been described in terms of video compression, in which images are compared in order to reduce bandwidth, the ability to compare images can be used with advantage in other technological areas, such as straightforward image comparison. This
5 may be useful in say Optical Character Recognition (OCR) or for Identify Friend or Foe (IFF) systems, where rapid comparison of an image with a "target" is either advantageous or essential. It may then be more advantageous, instead of using sum of absolute differences, to conduct a higher quality sum of squares of differences.

Other data processing

10 So far, the invention has been described in terms of processing one set of data from a block in a first image frame with successive blocks of data from a second image frame. The apparatus described succeeds over the prior art by performing several "comparisons" concurrently and reusing data that has already been fetched without having to re-fetch the same set of data each time it is required for a next comparison. However,
15 the same apparatus can be used in a different way to perform other mathematical functions.

For example, if the target block was not a block from an image frame but was instead a block of predetermined data forming a set of values of interest, that data set could still be brought into the ports of the accelerator described. The data set could then
20 be used in the Mux and latch circuits of Figure 8 to evaluate a function relating that data set with a succession of data sets introduced into the other port. The function itself would be established through the selection of processing units connected to the outputs of the Mux and latch circuits in place of the SAD units 75, 76 in Figure 8. Thus, the two data sets could be multiplied together, with or without a scaling factor, to emulate a filter
25 function operable on the data set in the succession of second blocks. The individual data items would be stepped through the accelerator as before, using parallel processing paths and re-use of the data in the second blocks, also as before.

A useful function would be a convolution function. This would be achieved by replacing the add functionality by a multiplication functionality and could have a range of
30 applications such as filtering. The results of the evaluations of the data items in the first set with corresponding data items in the successive second sets of data may be used to create a third set of data composed of data items representing a convolution function of the data items in the first and second sets of data.

The above uses of the apparatus could have particular application to image data where it could perform a variety of picture effects. The functionality of the accelerator would be designed to provide specific functions, for example a “blur” or “sharpen” operation. Consider a 16 x 16 grid with values adding up to one, for example a central
5 value of 0.5 reducing rapidly to zeroes at the edges. If that value is multiplied by all the corresponding values of a data set relating to an image, a value between zero and one results. That constitutes a blur and the blur function can be chosen to be a simple linear or other function. That blur could then be stepped by the unit in accordance with the invention, four pixels at a time instead of pixel by pixel.

10 It is also conceivable that the processing can achieve a desired transition between one data set and another, such as a simple fade or other special transition between frames or images in a sequence.

Claims

- 1 A method for evaluating a function between a first set of data and a succession of
second sets of data from an image frame, wherein each data item in the first data set is
5 evaluated with data items in corresponding positions in the second sets of data,
characterised in that a data item in the first block is evaluated simultaneously with a *(set)*
corresponding data item in each of n second data sets in said succession and wherein at
least one of said corresponding data items in a said second data set is/are locally stored
and reused in an evaluation of the next data item in said first data set with data items in
10 corresponding positions in at least the next second data set in the succession.
2. A method for searching in a second frame of image data for a first block of image
data appearing in a first frame, wherein data items in the first block are compared with
data items in corresponding positions in a succession of second blocks in the second
15 frame, characterised in that a data item in the first block is compared simultaneously with
a corresponding data item in each of n second blocks in said succession and wherein at
least one of said corresponding data items in a said second block is/are locally stored and
reused in a comparison of the next data item in said first block with data items in
corresponding positions in at least the next second block in the succession.
20
- 3 A method as claimed in claim 1 or claim 2, wherein said data items in said first
block or in said first data set are adjacent and said data items in a said second block or in a
said second data set are adjacent.
- 25 4 A method as claimed in claim 1 or claim 2, further comprising making said
evaluations or said comparisons along a plurality of scan lines one after the other until the
data items in the first block or in the first data set have been compared with n successive
second blocks or data sets.
- 30 5 A method as claimed in claim 2, wherein said comparison is carried out for each
of a plurality of first blocks in said first frame with each of said succession of second
blocks in said second frame.

6. A method as claimed in any of claims 2 to 5, wherein the partial results of simultaneously comparing $\underline{n}$ said data items in said first block with $\underline{n}$ corresponding data items in each said second block in said succession are temporarily accumulated in a respective one of $\underline{n}$ accumulators.

5

7. A method as claimed in claim 6, wherein after said $\underline{n}$ data items in said first block have been compared with said $\underline{n}$ corresponding data items in $\underline{n}$ successive second blocks, the first block is then compared with a further $\underline{n}$ successive second blocks, each of which is spaced $\underline{n}$ data items from the previous second block in the succession.

10

8. A method as claimed in claim 6, further comprising assessing from the partial results in the accumulators, the closeness of the match between the first block and a matching second block in the second frame.

15 9. A method as claimed in claim 2, wherein said comparison step is based on a sum of absolute differences algorithm.

10. A method as claimed claim 2, wherein said comparison step is based on a sum of the squares of differences algorithm.

20

11. A method as claimed in claim 1, wherein said data items in said first set of data comprise a predetermined set of data items and wherein the results of said evaluations with corresponding data items in said successive second sets of data are used to create a third set of data composed of data items representing a convolution function of the data items in the first and second sets of data.

25

12. A method as claimed in claim 11, wherein said convolution operation performs a blur function.

30 13. A method as claimed in claim 11, wherein said convolution operation performs a sharpen function.

14 A method as claimed in claim 11, wherein said convolution operation performs a filter function

15 15. Apparatus for evaluating a function between a first set of data and a succession of second sets of data from an image frame, comprising evaluating means for evaluating data items in the first set with data items in corresponding positions in a succession of second data sets; characterised by a plurality of evaluating means for simultaneously evaluating a data item in the first data set with a corresponding data item in each of n second data sets in said succession, and local memory means for retaining at least one of said
10 corresponding data items in a said second data set, whereby said retained data item is reused in an evaluation of the next data item in said first data set with n data items in corresponding positions in at least the next second data set in the succession.

16. Apparatus for searching in a second frame of image data for a first block of
15 image data appearing in a first frame, comprising comparing means for comparing data items in the first block with data items in corresponding positions in a succession of second blocks in the second frame; characterised by a plurality of comparison means for comparing n data items in the first block simultaneously with n data items in corresponding positions in n second blocks in said succession, local memory means for
20 retaining at least one of said n data items in a second block, whereby said retained data item can be reused in a comparison of the next data item in said first block with n data items in corresponding positions in at least the next second block in the succession.

17. Apparatus as claimed in claim 15 or 16, further comprising a first input port for
25 receiving said data items in said first data set or block, a second input port for receiving said data items in said second data set or block, and an output port providing n results of simultaneously comparing or evaluating a data item in said first data set or block with n data items in corresponding positions in n second data sets or blocks in said succession.

30 18 Apparatus as claimed in claim 17, further comprising n accumulators for temporarily storing the n respective results.

19 Apparatus as claimed in claim 17, wherein said first input port comprises $\underline{n}$ multiplexers and latches in $\underline{n}$ parallel paths connected to a single multiplexer, whereby successive data items in said first data set or block can be sequenced through to one input of $\underline{n}$ said comparison means.

5

20. Apparatus as claimed in claim 19, wherein said second input port comprises a plurality of multiplexers and latches in parallel paths, whereby successive data items in said succession of second data sets or blocks can be sequenced through to the other input of said $\underline{n}$ comparison means.

10

21. Apparatus as claimed in claim 16, wherein said comparison means comprise means for performing the sum of absolute differences between the data items in said first and second blocks.

15

22. Apparatus as claimed in claim 16, wherein said comparison means comprise means for performing the sum of the squares of the differences between the data items in said first and second blocks.

20

23. Apparatus as claimed in claim 21 or claim 22, further comprising $\underline{n}$ accumulators, each connected to the output of a respective comparison means, whereby to accumulate the partial results of the comparisons between the data items in the first block and the data items in the succession of second blocks.

25

24. Apparatus as claimed in claim 24, wherein said accumulators each comprise an adder and a latch in series, with a feedback path from the output of the latch to the input of the adder.

25

A data processor for processing first and second sets of data, comprising a processing element in combination with an accelerator, the accelerator comprising

30

apparatus as claimed in any of claims 15 to 24.

26 A data processor for processing first and second sets of data, comprising an array of processing elements each in combination with an accelerator, the accelerator comprising apparatus as claimed in any of claims 15 to 24.

5 27. A data processor as claimed in claim 26, wherein said array of processor elements is a Single Instruction Multiple Data (SIMD) array.

28. A data processor for processing first and second sets of data, comprising an array of processing elements, each processing element comprising apparatus as claimed in any
10 of claims 15 to 24.

29. A method of compressing image data by encoding a first image frame by reference to a second image frame, the method comprising the steps of:

searching in a succession of second blocks in said second image frame for a first
15 block of image data appearing in the first image frame; and

encoding the first block as a function of the closeness of the match between the first block and a matching second block and the offset of the first block relative to said matching second block in said second frame;

wherein said searching step comprises:
20 comparing data items in the first block with data items in corresponding positions in a succession of second blocks in the second frame, characterised in that a data item in the first block is compared simultaneously with a corresponding data item in each of n second blocks in said succession and wherein at least one of said corresponding data items in a said second block is/are locally stored and reused in a comparison of the next data item in
25 said first block with data items in corresponding positions in at least the next second block in the succession.

30. A computer program product comprising a medium on which is stored a computer program adapted to perform the steps of the method of claim 1.

30

31 A computer program product comprising a medium on which is stored a computer program adapted to perform the steps of the method of claim 2.

32. A computer program product comprising a medium on which is stored a computer program adapted to provide a first set of data and a succession of second sets of data, to perform the steps of the method of claim 1 or claim 2 on said first and second sets of data, and to utilise the resulting data produced by the method.

5

33. A method of processing first and second sets of data, substantially as herein described with reference to the accompanying drawings.

34. A method of compressing a series of image frames, substantially as herein
10 described with reference to the accompanying drawings.

35. A data processor, substantially as herein described with reference to the accompanying drawings.



Application No: GB0413862.4

2⁴ Examiner: Iwan Thomas

Claims searched: 1, 2, 15, 16 and 29

Date of search: 29 October 2004

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
X	1, 2, 15, 16 and 29 at least	EP 0220110 A2 (FAIRCHILD) See pages 3-6 at least
X	1, 2, 15, 16 and 29 at least	US 5448310 A (KOPET) See cols. 3 & 19-21 and claim 6
A	-	US 5929939 A (OHTANI)

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC^W :

H4F

Worldwide search of patent documents classified in the following areas of the IPC⁰⁷

H04N

The following online and other databases have been used in the preparation of this search report

Online: WPI, EPODOC, PAJ