

(19) Japan Patent Office (JP)
**(12) Japanese Unexamined Patent
Application Publication (A)**

(11) Japanese Unexamined Patent
Application Publication Number
H1-309185

⁵¹ Int. Cl. ⁴	Identification codes	JPO file number	(43) Publication date
G 06 F 15/60	370	A-8125-5B	December 13, 1989
		Request for examination	Not yet requested
			Number of claims 30
			(Total of 23 pages)

⁵⁴ Title of the invention Computer-aided design system for ASIC

(21) Japanese Patent Application H1-7550

(22) Date of application January 12, 1989

Priority claim (32) January 13, 1988 (33) United States (US) (31) 143,821

⁷² Inventor	Hideaki Kobayashi	c/o International Chip Corporation, Suite 890, 1401 Main Street, Columbia, South Carolina 29201 United States
⁷² Inventor	Akihiro Shindo	c/o Ricoh Co., Ltd., 1-3-6 Nakamagome, Ohta-ku, Tokyo
⁷¹ Applicant	International Chip Corporation	Suite 890, 1401 Main Street, Columbia, South Carolina 29201 United States
⁷¹ Applicant	Ricoh Co., Ltd.	1-3-6 Nakamagome, Ohta-ku, Tokyo
⁷⁴ Agent	Patent attorney Tamotsu Aoyama	(and 1 other)

Specification

1. TITLE OF THE INVENTION

Computer-aided design system for ASIC

2. SCOPE OF PATENT CLAIMS

(1) A computer-aided design system for designing an application-specific integrated circuit directly from the functional specifications of an integrated circuit that is characterized by the fact that it is equipped with

an input means that can be operated by a user to define the specifications of a function for an integrated circuit,

and a conversion means that may be operated using a computer to convert the above-noted functional specifications into the definition of the structural level of an integrated circuit.

(2) The system described in Claim (1) wherein the functional specifications have a series of operations and conditions, and the definition of the structure level has blocks and an interconnection between the blocks.

(3) The system described in Claim (1) wherein the input means is equipped with a flowchart editor to create a flowchart having elements representing a series of operations and conditions that describe the specifications of a function of the integrated circuit.

(4) The system described in Claim (1) wherein the means that may be operated using a computer includes an element library that

defines an integrated circuit hardware element for realizing the operation and conditions, and an element selection means to select an appropriate hardware element corresponding to a block for a structural definition from the element library.

(5) A computer-aided design system for designing an application-specific integrated circuit directly from the functional specifications of an integrated circuit that is characterized by the fact that it is equipped with

a macro library to define the operations and conditions, an input means that can be operated by a user to define the specifications of a function for an integrated circuit wherein these functional specifications have a series of operations and conditions, and the input means further includes a means that allows the user to specify a macro that has been selected from the macro library for the operations and conditions contained in each limited series of operations and conditions,

an element library to define an integrated circuit hardware element for realizing the operation and conditions, and an element selection means for selecting the hardware elements from the element library to execute the operations or conditions determined by the macro from the element library for a macro identified by the above-noted specifications.

(6) The system described in Claim (5) wherein the element selection means is equipped with a knowledge base that contains the rules for selecting hardware elements from the element library, and an interface engine means to select an appropriate hardware element from the element library in accordance with the rules in the above-noted knowledge base.

(7) The system described in Claim (5) wherein the input means is equipped with a flowchart editor that contains a flowchart having elements that represent a series of operations and conditions.

(8) The system described in Claim (7) that is equipped with a flowchart simulator that simulates the functions defined within the flowchart in order to enable the user to verify the operations of the integrated circuit.

(9) The system described in Claim (5) wherein the input means is equipped with a means to receive the input by a user of a list that defines a series of operations and conditions.

(10) The system described in Claim (5) wherein the element selection means includes a data passage generating means to generate a data passage for the hardware element that has been selected by this element selection means in cooperation with the element selection means.

(15) A computer-aided design system for designing an application-specific integrated circuit directly from the functional specifications of an integrated circuit that is characterized by the fact that it is equipped with

a macro library to define the operations and conditions,

a flowchart editor to create a flowchart having elements that represent a series of operations and conditions wherein the flowchart editor includes a macro specification means that enables a user to specify a macro that has been selected from the macro library for each operation and condition represented in the flowchart,

an element library to define an integrated circuit hardware element to realize the operation and conditions,

an element selection means to select the hardware elements to execute the operations or conditions determined by the macros from the element library,

and a data passage generating means to generate a data passage for the hardware element selected by the element selection means in cooperation with the element selection means.

(11) The system described in Claim (10) wherein the data passage generating means includes a knowledge base that contains the rules for selecting data passages between hardware elements or between a hardware element and an interface engine, and the data passage between the hardware elements that were selected by the element selection means is selected in accordance with the rules in the above-noted knowledge base and the notes of the specific macro.

(12) The system described in Claim (10) that includes a control generator to generate a controller and a control passage for a hardware element that has been selected by the cell selection means.

(13) The system described in Claim (5) that is equipped with a net list generation means to cooperate with the element selection means, and that will output a net list to determine the hardware elements required to complete the functional conditions of the integrated circuit.

(14) The system described in Claim (13) that is equipped with a mask data generating means to generate the mask data required for the manufacture of an integrated circuit having specific functional conditions from a net list.

(16) The system described in Claim (15) wherein the element selection means is equipped with a knowledge base that contains the rules for selecting hardware elements from the element library and an interface engine,

and will select the hardware elements from the element library according to the rules in the knowledge base,

and the data passage generating means includes a knowledge base that contains the rules for selecting data passages between hardware elements or between a hardware element and an interface engine, and the data passage between the hardware elements that were selected by the element selection means is selected in accordance with the rules in the above-noted knowledge base and the notes of the specific macro.

(17) The system described in Claim (15) that includes a control generator to generate a controller and a control passage for a hardware element that has been selected by the cell selection means.

(18) A computer-aided design system for designing an application-specific integrated circuit directly from the functional specifications of an integrated circuit that is characterized by the

fact that it is equipped with

a flowchart editing means that includes a means that can not only enable a user to create a flowchart that has a square box form that represents the operations and a rhombus that represents the conditions, as well as a line having an arrow that indicates the transition between the operations and conditions, but that will also provide the specific operations or conditions to be executed on each box form or rhombus,

an element library to define an integrated circuit hardware element to realize the operation and conditions,

a knowledge base that contains the rules for selecting hardware elements from the element library and the rules for generating data and control passage for use with the hardware elements, and an expert system to convert the flowchart defined by the flowchart editing means in cooperation with the above-noted knowledge base into a net list to determine the required hardware elements and data and control passages that are required for the integrated circuit equipped with the specific functional requirements.

(19) The system described in Claim (18) that is equipped with a means to generate the mask data required to manufacture an integrated circuit equipped with the special functional requirements from a net list.

(20) A computer-aided design method for designing an application-specific integrated circuit to execute a desired function, wherein this method involves

(23) The method described in Claim (22) that wherein the process to generate the data passage includes the application of the data passage rules stored in the knowledge base to the selected element, and the generation of the data passage.

(24) The method described in Claim (23) that includes a process to generate a control passage for use with the hardware element of the integrated circuit that was selected.

(25) The method described in Claim (20) that includes a process to generate a net list to determine the hardware elements required to execute the desired functions of the integrated circuit and connection conditions between each element in relation to the integrated circuit hardware elements that were selected.

(26) The method described in Claim (25) that includes the generation of mask data required to manufacture an integrated circuit having the desired function from a net list.

(27) A knowledge-based design method for designing an application-specific integrated circuit to execute a desired function, wherein this method is characterized by the fact that it involves

the storage of the definitions of operations and conditions,

the storage of data describing the hardware elements of available integrated circuits in order to execute the operations and conditions that have been stored,

the description of the operations and conditions required for the application-specific integrated circuit,

the specification of the above-noted stored definitions that correspond to the required operations and conditions to be executed in relation to the operations and conditions that have been described,

and the selection of an integrated circuit hardware element from the stored data for a specific definition in order to execute the desired function of an application-specific integrated circuit.

(21) The design method described in Claim (20) wherein the process to select the hardware element will add the element selection rules stored in the knowledge base for the specific definitions of the operation and conditions to be executed.

(22) The method described in Claim (21) that includes a process to generate a data passage for use with the hardware element of the integrated circuit that was selected.

the storage of macros that define the required operations and conditions in a macro library,

the storage of the available hardware elements of an integrated circuit in an element library in order to execute the operations and conditions,

the storage of the rules for use with the hardware elements selected from the element library in order to execute the operations and conditions as defined by the stored macros,

the description of a proposed series of operations and conditions for performing the functions to be executed by the integrated circuit for the desired application-specific integrated circuit,

the specification of a series of macros selected from the macro library corresponding to the operations and conditions for each described operation and condition,

and the application of the rules in the knowledge base to specific macros in order to select the hardware elements required to execute the desired functions of the application-specific integrated circuit from the element library.

(28) The method described in Claim (27) wherein the process to describe the operations and conditions includes the generation of a flowchart that contains elements representing the operations and conditions.

(29) The method described in Claim (27) that includes the storage of the rules for generating the data passage between hardware elements in a knowledge base, and the application of the rules in the knowledge base to a means to specify the rules in the knowledge base in order to generate the data passage for the hardware elements that were selected.

(30) The method described in Claim (29) that includes a process to generate a controller and a control passage for the hardware element that was selected.

3. DETAILED DESCRIPTION OF THE INVENTION

FIELD OF INDUSTRIAL APPLICATION

The present invention relates to the design of integrated circuits, and more specifically, to a method and apparatus for computer-aided design to design integrated circuits.

PRIOR ART

The design specifications created by VLSI design engineers may be presented in the form of schematics, circuit parameters, or special hardware description language (HDLs). From the design specifications at the structural level, the description of the hardware parts and interconnections will be converted to a description at the specific chip placement level that describes the actual microstructure features of the integrated circuit chip. This description of the specific chip placement level will provide the mask data required to produce the chip concerned.

The development of very large scale integration (VLSI) technology has enabled the formation of highly complex circuit systems in a single chip. With this complexity and the need to design VLSI chips at an early stage, it is necessary to use computer-aided design, or in other words, CAD technology, in order to deal with the increase in volume and special applications. CAD technology has been successfully used to design and inspect integrated circuits at both the structural and physical placement levels. For example, CAD systems may be used to help convert

An application specific integrated circuit (ASIC) is an integrated circuit chip that is designed to execute specific functions, unlike general purpose integrated circuit chips such as microprocessors and memory chips. Sophisticated design engineers with special knowledge of VLSI circuit design may be tasked with the design of ASICs. During the design process, the VLSI design engineer will take into account the specific plan or purpose to be realized and the work (tasks) to be performed by the integrated circuit, and will create design specifications at the structural level in order to determine the required interconnections between each component as well as the various hardware components required to perform the desired functions or actions. System controls must also be designed in order to perform synchronized operations between these parts. This work requires general and extensive knowledge of the various hardware components, the connections between the required elements, signal level compatibility, timing compatibility, physical placement and the like, as well as the need to achieve desired objectives or plans. During each design process, designers must perform long, monotonous analyses.

VLSI structure-level descriptions into microstructured mask data at the physical placement (layout) level required for actual production. While existing CAD systems help to facilitate the VLSI design process, the current practice requires advanced VLSI design engineers to create the required structural level hardware descriptions.

Only a few technicians have the advanced technology skills to create hardware descriptions for this type of integrated circuit. Even when using the available VLSICAD tools, this type of design can take a long time and can be error-prone due to the human labors involved. Better and cheaper methods for custom integrated circuits are required.

MEANS FOR SOLVING THE PROBLEM AND OPERATION

The CAD system and method of the present invention makes it possible for a user to use an easy-to-understand functional level representation to generate the detailed information that is required to directly create an ASIC that can determine the functional requirements of the intended integrated circuit and that can execute

the specific functions from the functional requirements described above. In this way, the present invention enables design and manufacture even if the user is not ASIC designer or technician and lacks the knowledge of special experts capable of advanced VLSI design technology.

The functional specifications of the desired ASIC described above may be determined in a suitable manner, such as in the form of a list, or preferably as a flowchart. Flowcharts may be easily understood by software and hardware engineers with various skill levels and degrees of training, and are extremely effective in describing the order of logical operations. The systems and methods of the present invention may be translated from flowcharts (or other functional specifications) into the definitions at the structural levels that can be used to directly manufacture ASIC. These structure level definitions also include a list of the integrated circuit hardware cells required to complete the specifications of a specific function. These cells may be selected from element libraries that store hardware elements that have been pre-designed in order to achieve various functions and technical specifications or details. Further, the present invention may also generate a control path for the system controller and selected integrated circuit hardware elements.

KBSC uses a knowledge-based expert system with a knowledge database extracted from expert ASIC designers with advanced expert technology in VLSI design, and from the flowchart, will generate a net list describing the interconnection conditions of the selected hardware elements and each element.

EXAMPLES OF EMBODIMENT

Figures 1a, 1b, and 1c show three different levels representing an integrated circuit design. Figure 1a shows a functional or operational representation in a flowchart format. The flowchart is a graphical representation of an algorithm and consists of two types of blocks or states: operations and conditions (or decisions). In the flowchart, the operations are represented by rectangular blocks, while the conditions are represented by diamonds. The transition between each operation and each condition is indicated by a line with an arrow. Figure 1b shows the structural or logical representation level of an integrated circuit. In this representation, blocks are used to indicate the hardware elements of integrated

The list of hardware elements and their interconnection requirements may be represented in the form of net lists. The well-known manual operations or existing VLSI CAD placement system may be used to generate geometrical graphical information from net lists, such as the detailed chip-level mask data required to manufacture chip-format ASICs.

In a preferred example of embodiment of the system and method of the present invention that is described in detail below, this is referred to as a knowledge-based silicon compiler (KBSC). KBSC is an ASIC design technology that is based on artificial intelligence and expert system technology. The user interface of the KBSC is a flow chart editor that allows designers to represent the VLSI system in a flowchart format.

circuits that execute various functions, and the lines connecting the blocks indicate the flow of data between each block and the path of the control signal. A block may represent a hardware element such as, for example, an adding machine, a comparator register, or a system controller, etc. Figure 1c is a representation of the specific layout level of an integrated circuit design, and it provides the detailed mask data required for the actual manufacture of the apparatus and connection conductors that constitute the integrated circuit.

As noted earlier, integrated circuit design at the structural level is desirable for design engineers who are highly specialized in VLSI design. However, the KBSC system of the present invention provides expertise in VLSI design, and by applying this expertise as a result of the present invention, it will be possible to design an integrated circuit at the functional level. By allowing designers to work on flowcharts instead of graphical representation of logic circuits, the design work for custom integrated circuits will become

easier, faster, and cheaper, while increasing reliability at the same time. Using a simple flowchart of the level of operation, the designers will handle the algorithms, and it is sufficient for designers to know only the logical steps required to complete the task, rather than the specific means to complete the task. Designing with flowcharts can reduce tasks related to testing. This is because flowchart work is closer to the algorithm from the standpoint of the designers. On the other hand, the existing VLSI design tools require designers to represent algorithms in complex schematics at the structural level, necessitating testing tasks. Because schematics enable a mixture of consideration of the hardware and function, it is difficult for schematics to overcome the computational functions that must be introduced into the intended design. The use of flowcharts to design custom design integrated circuits allows a large number of system designers to be involved in the VLSI technology, which was previously known by only a small number of designers, in order to obtain the hardware descriptions.

The flow of the overall system used in the present invention is shown in Figure 2. The user enters the function (operation) specifications of the circuit into KBSC (Knowledge-based silicon compiler) 10 in the form of flowchart 11.

The system includes flowchart editor 21 to create and edit a flowchart, and flowchart simulator 22 to simulate and inspect the flowchart. The operation to be executed by each rectangle in the flowchart may be selected from macro library 23. Program 30, which may be referred to as a path synthesizer and cell selector (PSCS), includes data and control signal synchronizer module 31, which is a knowledge-based system for data and control signal synchronization. PSCS further includes element selector 32 to select the elements required in the system design. Element selector 32 will select the appropriate elements required for each operation and condition shown in the flowchart, making this selection from element library 34 of the hardware elements that were designed previously. Control signal generator 33 will generate the custom-design system control signals in order to control the operations of the other hardware elements. Knowledge base 35 includes knowledge for use by ASIC design experts as required for the data path synchronizer and element selection. As a result of the functional flowchart input in this configuration, the PSCS will generate a system control signal, select all of the other hardware elements, generate the data and control passages, and will further

KBSC 10 generates net list 15 from flowchart 11. Net list 15 includes a custom-generated system controller, the other hardware elements required to enable the required operations, and the information on the interconnects that connect said hardware elements and system controller. This net list can be used as the input into an existing VLSI layout and routing tool 16 in order to generate mask data 18 for use in the geometric layout.

OVERVIEW OF THE SYSTEM

Figure 3 shows a basic element or module having a KBSC system. In this example of embodiment, these elements or modules are in the form of software programs, and it would be easy for those skilled in the art to realize these elements in other suitable forms of hardware.

Referring to Figure 3, KBSC system 10 includes program 20 that may be called EDSIM.

generate a net list describing all of the design information.

The KBS system uses a hierarchical element selection ASIC design approach as shown in Figure 4. Rather than generating each required hardware element from scratch, or in other words, from a group of various elements that perform the desired functions, the KBS system is equipped with element library 34 of the hardware elements that have various operational capabilities in various forms or given configurations that were previously designed or tested and validated. Macro library 23 includes macros that define the various operations embodied in the flowchart. There are also several hardware elements within element library 34 with other geometrical forms and characteristics to execute the functions specific to each macro function in macro library 23. By using a rule-based expert system with knowledge base 35 drawn from expert ASIC designers, the KBSC system will select the optimal element from element library 34 in order to execute the desired function.

Referring to Figure 3, the elements selected by element selector 32, the control information generated by control signal generator 33, and the data and the control passage generated by data/control

passage synchronizer 31 will all be used by PSCS program 30 to generate net list 15. The net list is a list that shows each block in a circuit, and the interconnections between the input and output terminals of each block. This net list provides all of the information needed to build an integrated circuit. The commercially-available element arrangement and element connection computer-assisted design system will receive the above-noted net list data as input, and will perform the arrangement (layout) of each element in the chip, or will determine the necessary connections, and will generate the mask data directly used by the chip factory (foundry) in the manufacture of integrated circuits.

The main memory stores the next data temporarily, executable code, design data (flowcharts, logic, etc.), databases (element library), and knowledge base. The CPU generates flowcharts and performs automatic synchronization of simulations and designs.

EXAMPLE OF THE FLOWCHART

The example flowchart shown in Figure 5 may be considered as a portion of a large system such as that shown in Figure 5 to make a graphical representation of a flowchart for a single net list. In this illustrated flowchart, two Variables VAL1 and VAL2 are compared, and if they are equal, they may be aggregated together. In this example, the first operation (Action 1) includes the operation to transfer Variable VAL1 to Register A. The second operation includes the movement of Variable VAL2 to Register B. Condition 1 includes a Compare operation to compare the values of Registers A and B. The third operation (Action 3) adds (Add) the values of Registers A and B, and stores the results in Register C (Store).

When manufacturing integrated circuits for performing the functions shown in Figure 5, KBSC displays a flowchart description of the operations of the system for the interconnection conditions between hardware elements. These hardware elements may be controlled by a system controller that generates all of the control signals. There are two types of variables included in the system controller, as shown below.

(1) Input variables: Input variables may be made by hardware

CONDITIONS REQUIRED BY THE SYSTEM

The KBSC system may be operated by a suitably programmed general purpose digital computer. For example, in an example of a KBSC system, the system may be operated by a workstation such as a Sun3 and VAX Station-II/GPX running UNIX Operating System and X Window Manager. This workstation requires at least 8 megabytes of main memory and 20 megabytes of hard disk space. The monitor used is an 8-bit flat-plane color television. The software uses the C programming language and the INGRES relational database.

The human interface mainly uses pop-up menus, buttons, and command languages for special purposes. The permanent data of the integrated circuit design may be stored in the form of a database to facilitate extraction and updates.

elements or may be inputted from outside the controller. These correspond to the conditions in the flowchart.

(2) Output variables: The output variables may be made by the system controller and correspond to the operations in the flowchart.

Figure 6 shows the results illustrating the flowchart of Figure 5 for a hardware element. The flowchart operations and conditions may be used for element selection, and for data and control path synchronization. The VAL1 and VAL2 registers and the data passageways derived from each register are generated prior to Operation 1 in the present example of embodiment. Operation 1 forces the generation of the data of Register A. Similarly, Operation 2 forces the ranking of the data of Register B. The comparator will be positioned as a result of the compare operation in Condition 1. This compare operation will force the generation of the input variables for the system controller according to 1) the selection of the elements of the comparator, 2) the definition of the inputs of the comparator elements for Registers A and B, 3) the definition of the passages connecting the comparator to Registers A and B, and 4) determining whether these are equal, larger, or smaller. Similarly, the add operation in Operation 3 selects the addition element, creates the parameter of the adding machine for the register, and generates the data passage. A block list may be generated for the flowchart after this process. The block list includes information about the system controller and various blocks required to execute the required operations, and each block

is connected by a data passage and controlled by the system controller via the control passage. These blocks may be prepared in order to select elements from the element library to make an element list.

INTER-OPERATIONS FLOWCHART EDITOR AND SIMULATOR

Creating and verifying flowcharts is the first stage of the VLSI design methodology. The conversion of one algorithm to one equivalent flowchart may be performed by flowchart editor 21 (Figure 3). Verification of the edited flowchart may be performed by flowchart simulator 22. The flowchart editor and simulator may be integrated within the same work environment in order to perform the flowchart editing operations by the designer of the intimate interface.

EDSIM is a program that includes flowchart editor 21 and flowchart simulator 22. This EDSIM provides functions such as loading or saving flowcharts. The EDSIM generates an intermediate file for each flowchart called a state list. This intermediate file may be used by PSCS program 30 to generate a net list. FLOWCHART EDITOR

Flowchart editor 21 is a software module that may be used to display, create, and edit flowcharts. This module may be controlled via the flowchart editing window shown in Figure 7. The flowchart editor, along with the editing function, checks for design errors.

The flowchart editor operations are described below. The primary editing function includes the generation, editing, and deletion of states, conditions, and transitions. The generation operation allows the designer to add new states and conditions or transitions to the flowchart. Editing allows the designer to change the position, condition, or transition of a state in a flowchart, and deletion allows the designer to remove a state, condition, or transition from an existing flowchart. The state including the operation is indicated by a square box, while the condition is indicated by a diamond, and the transition is indicated by a line with arrows showing the direction of the transition.

Editing operations allow the designer to grant motion to each box. These operations include macro names and notes (Argument). Examples of notes include the setting and clearing of external signals. A list of the basic macros available in macro library 23 is shown in Table 1.

Table 1

Macro(マクロ)	Description(記述)
ADD (A, B, C)	$c = A + B$
SUB (A, B, C)	$c = A - B$
MULT (A, B, C)	$c = A * B$
DIV (A, B, C)	$c = A \text{ div } B$
DECR (A)	$A = A - 1$
INCR (A)	$A = A + 1$
CLR (A)	$A = 0$
REG (A, B)	$B = A$
CMP (A, B)	Compare A to B and set EQ, LT, GT signals (AとBを比較し、EQ, LT, GT信号を設定)
CMPQ (A)	Compare A to Q and set EQ, LT, GT signals (AとQを比較し、EQ, LT, GT信号を設定)
NEGATE (A)	$A = \text{NOT } A$
MOD (A, B, C)	$c = A \text{ modulus } B$
POW (A, B, C)	$c = A ^ B$
DC2 (A, S1, S2, S3, S4)	Decode A into S1, S2, S3, S4 (AをS1, S2, S3, S4にデコード)
EC2 (S1, S2, S3, S4, A)	Encode S1, S2, S3, S4 into A (S1, S2, S3, S4をAにデコード)
MOVE (A, B)	$B = A$
CALL sub-flowchart (A, B, ...)	Call a sub-flowchart. Pass A, B... (サブフローチャート呼び出す。A, B...を通過)
START (A, B, ...)	Beginning state of a sub-flowchart (サブフローチャートの状態の始まり)
STOP (A, B, ...)	Ending state of a sub-flowchart (サブフローチャートの状態の終わり)

Set state - Set the next state to be simulated

Set the display of the details or summary - Display the details or summary information during a simulation

Set break - Set the break point

Clear break - Clear all of the break points

View break - View the current break point

Step - Proceed by one transition

Execute - Execute a flowchart

Stop - Stop the flowchart execution

History-on or History-off - Turn the recording of history ON or OFF

Cancel - Cancel an operation that is being executed

Help - Display of the help screen

Close - Close the simulator window

The simulation results may be displayed in the simulator window.

Alternatively, the editor window may track the simulated portion as the flowchart is simulated. Flowchart tracking facilitates the editing of flowcharts when errors are found.

ELEMENT SELECTION

Element selector 32 is a knowledge-based system for selecting an optimal set of elements from element library 34 in order to realize a VLSI system. This selection may be made based on the description of the functions within the flowchart, and may be

The flowchart editor graphically displays the flowchart according to the simulation of the flowchart by the flowchart simulator. This graphic representation includes boxes, diamonds, and lines as shown in Figure 7. Everything is drawn on the display screen and displayed in the same shape as in a conventional flowchart. By displaying the flowchart on the display screen during the simulation, designers can simultaneously design and validate the flowchart.

FLOWCHART SIMULATOR

Flowchart simulator 22 is a software module that may be used to simulate a flowchart. This module may be controlled via the simulator window shown in Figure 8. The flowchart simulator simulates the transitions between flowchart states or conditions. The flowchart simulator operations are shown below.

Edit data - Change the value within a register or memory

realized by the macros assigned to each operation displayed in the flowchart. The elements selected to create the VLSI system depend on various conditions such as the function of the element, the type of manufacturing technology, the power limitations that may be used, and the delay time, etc. The element selector uses a knowledge base extracted from a VLSI design specialist (expert) in order to select the elements.

It is necessary to have conformity between the functions in the flowchart and the elements from the element library in order to design the VLSI system based on the description of the flowchart during utilization by the user. In order to ensure this conformity, artificial intelligence techniques must be used because the element selection step is complex and is based on a large number of design parameters and constraints.

The technical ideas used in element selection are similar to those used in compilation in software. In software compilation, many subroutines may be combined from the library. In the design of a VLSI system, functional macros may be represented in the library element.

Figure 4 shows the technical concept of the hierarchical element selection. The element selection step is comprised of two steps.

- (1) Selection of functional macros
- (2) Selection of geometrical elements

A group of basic macros is shown in Table 1. One macro corresponds to one operation within the flowchart. For example, it is possible to consider the operation to add A and B, and to store those results C. This function may be represented by the addition macro, ADD (X, Y, Z).

The flowchart editor and flowchart simulator may be used to draw boxes and diamonds and lines in the shape of the flowchart, and may be further used to assign the macros selected from the macro library 23 to each operation shown within the flowchart, and to verify functionality within the flowchart. The flowchart may be converted to an intermediate shape (state list) for input into the element selector.

The element selector uses a rule-based expert system to select the suitable elements for executing each operation. If the element library has a plurality of elements of different shapes to execute the operations specified by the macros, the elements will be selected based on conditions such as the element function, the process technology that may be used, the delay time, or power limit values, etc.

The knowledge base of element selector 32 includes information or rules related to each of the following parameters.

- (1) Selection of macros
- (2) Consolidation of two macros
- (3) Connection of a macro and an element

(4) Consolidation of two elements

(5) Diagnosis of errors

As a rule, the above-noted information may be stored in knowledge base 35.

CREATION OF THE ELEMENT LIST

Figure 9 shows the stage of the element list creation.

The first stage of element list creation is conversion to a structure that may be utilized by the element selector described in the flowchart. This transformation is called a state list. The block list may be created by the interface engine from the state list. This block list contains a list of the functional blocks that may be used in an integrated circuit. At this stage, the following rules for forms may be utilized.

- Preparation of notes for data passages
- Creation of operations for macros
- Connection of each block

The rules prepare for the optimization of elements and diagnosis of errors at this level.

The element selector will create a block for the element selected from element library 34. The element selector also selects the optimal element for a single block. This operation also includes the

formulation of rules for the selection of suitable elements from the element library. Each element stores information in four formats:

- (1) Functional level information: A description of elements at the register transfer level
- (2) Logical level information: A description of flip-flop and gates
- (3) Circuit level information: A description at the transistor level
- (4) Layout level information: Details of the geometric mask level

The attributes of the element are as follows:

- Element name
- Description
- Function
- Width
- Height
- State
- Technology
- Minimum delay time
- Typical delay time
- Maximum delay time
- Electric power
- File
- Designer

- Date
- Comments
- Test technician

In the element selection step, the information described above can be used. Some of parameters that can be used to generate the macros for elements are shown below.

- (1) Macro name
- (2) Functions to be executed
- (3) Complexity of the chip
- (4) Manufacturing technology
- (5) Possible delay time
- (6) Power consumption
- (7) Bit size of the macro data passage

- (1) Read the flowchart intermediate file and form the state list
- (2) Current context = START
- (3) Start the interface engine and load the current context rules
- (4) Execute one of the following according to the current context:
 - (a) Amend the state list to ensure the accurate implementation
 - (b) Create a block list, macro list, and data passages
 - (c) Optimize the block list and data passages, and perform an error check
 - (d) Convert the block to elements
 - (e) Optimize the element list and perform an error check
 - (f) Create the net list
 - (g) Optimize the net list and perform an error check, then go to 7 after completion (go to 7)
- (5) Load new context rules if the current context has changed
- (6) Go to 4 (go to 4)
- (7) Output net list file and stf file, then stop

The operations described in Step 4 will be described in the next section. The rule language and PSCS display are described.

The rule language of PSCS is declarative and is designed to facilitate the editing of rules. In order to help professionals understand the structure of the knowledge base, the language

GENERATION OF THE NET LIST

The net list may be generated after the element has been selected by PSCS. The PSCS may also use the macro definitions to connect the terminals of the element to other elements. The PSCS uses a state-to-state transition from an intermediate list of display formats of the flowchart, such as a state list, to generate the net list. The PSCS includes the following knowledge in order to generate the net list:

- 1) Data passage sync
- 2) Data passage optimization
- 3) Definition of macros
- 4) Element library
- 5) Error detection and correction

As a rule, the above-noted information may be stored in knowledge base 35. The knowledge technician may assist the ASIC design expert in creating the above-noted rules. Macro library 23 and element library 34 may be stored in a KBSC database.

Several operations may be performed by the PSCS. The highest level of the PSCS operations is described below.

provides a means to display knowledge. This operation allows the format of the data structure to be stated on a rule basis, allowing experts to view the data structure and understand the various structures utilized by the system. For example, an expert can analyze the structure of the wiring and determine the components. Experts can incorporate those components into the rules. When a new object is determined, experts can declare a new structure and modify it to link an existing structure. In this way, the growth of the data structure can be better visualized by experts. Further, this allows designers to subsequently modify or add rules.

The following aspects are included in the rule language:

- i) Display of the knowledge of the shape of the record structure
- ii) Display of the state in the preamble of the rules
- iii) Facilitation of the creation and destruction of structures in rules operations
- iv) Description of the duties in the operation of the rules
- v) Facilitation of input and output in rules operations
- vi) Facilitation of stimulation of C function from rules operations

The formats for rules that may be used are shown below.

Rule	No.	Context	context	This is context that says
IF	IF	IF		that this rule is active.
THEN	THEN	THEN		
WHERE	WHERE	Rule No.		

INTERFACE METHOD

The interface method is based on a fast pattern matching algorithm. This rule may be stored within the network, making it possible to avoid the requirement for repeated statements via the rules. As a result, this can increase the speed of execution. The conflict resolution method is based on the following:

- (1) The rules containing data at the closest time point may be selected.
- (2) The rules with the most complex conditions may be selected.
- (3) The rule that was declared first may be selected.

RULE EDITOR

The PSCS provides a rule editor that performs interactions to enable experts to update the set of rules. This rule may be stored in the database such that the editing ability of the database package can be used to edit the rule.

In order to execute this operation, the expert must be familiar with various knowledge structures and interface processes. If this is not possible, it will be necessary to obtain the help of a knowledge engineer.

The PSCS provides a menu that permits the configuration of various options. A mechanism is provided for setting various debug flags, and there is a mechanism for display options, as well as for overall control of the PSCS.

There is also a mechanism to save and display a block list that has been created by a user. The shape of the block list created by the user can be saved to a file and later printed using a plotter. It can be reset in order to restart the PSCS display process.

EXAMPLES OF THE PSCS RULES

Rule 1

IF If a block does not exist
 THEN The system controller shall be generated

Rule 2

IF If there is a state with a macro, AND this macro has
 not been created in a block
 THEN Find the corresponding macro in the library and
 generate a block for this macro.

Rule 3

IF If there is a transition between two states, AND if
 there is a macro with the same note
 THEN Create a connection from the register corresponding to
 the first macro to another register corresponding to the
 second macro.

Rule 4

IF If only one register is connected from another register
 THEN Combine these registers into one register

Rule 5

IF If there are two comparators AND the input data width
 is the same size AND the input is the same AND the
 output of the comparator is used to perform the same
 operation
 THEN Combine these comparators into one comparator

Rule 6

IF If there is data without a register
 THEN Create one register for this data

Rule 7

IF If all of the blocks are already mutually connected
 AND a single block has several terminals that have not

been connected

THEN Exclude these blocks and their terminals or output an error message

Rule 8

IF If memory is used, but no blocks have been created for that memory

THEN Create a memory block along with read and write terminals for data, addresses, data and control

Rule 9

IF If a register has one connection to the counter,

THEN Combine the register and counter to remove the register and its terminals

Rule 10

IF If there is a connection from several different blocks to one terminal of one block,

THEN Insert a multiplexer, remove the connections to the terminals, connect those terminals to the input of the multiplexer, and connect the output of the multiplexer to the one block noted above

The key points of the other rules include the following:

- Remove elements that can be replaced by using the output of other elements
- Simplify multiple trees
- Utilize fan-outs from elements

[placeholder]

EXAMPLE OF THE DESIGN OF A CONTROLLER FOR USE WITH A SOFT DRINK VENDING MACHINE

The following example shows how the above-noted aspects of the present invention may be used below in the design of an application-specific integrated circuit (ASIC). In this example, the ASIC was used to design a vending machine controller. The vending machine controller will receive a signal each time a coin is placed in the coin receiving unit. When the total value of the coin received is correct, the amount of the coin is recorded, and the controller will output a signal to dispense a single beverage. If a coin that was received has a higher value than the price of the beverage, the controller will dispense change to achieve the correct amount.

In this example of a vending machine controller, a pattern may

be created after using text to teach the digital system controller. See Fletcher, William L., An Engineering Approach to Digital Design, Prentice-Hall, Inc., pp. 491-505. By referring to this text to more fully describe the requirements of the vending machine controller, this example clearly shows how complicated the design method was prior to the emergence of the present invention in order to design the controller hardware.

Figure 10 shows a flowchart of the controller of the vending machine. This flowchart may be entered into the KBSC system by the user via the flowchart editor. Looking over the flowchart, the controller will receive a signal that a coin is present when a coin is present in the coin receiving unit. State 0 and Cond 0 define the weight states that indicate that the unit is waiting for a coin to be loaded. CP indicates that there are coins present, while !CP indicates that there are no coins. State 1 and Cond 1 determine when a coin has been cleared from the coin receiving unit. In State 20, after the coin has been received, the macro instruction ADD 3.1 (1c, cv, sum) will provide the system with instructions for 1c (last coin) and cv (coin value), storing the total. Working with State 21, the macro instruction will move the value of the register sum to register cv. In State 22, macro CMP.1 will return signals EQ, GT and LT by comparing the register cv value to PR (price of soft

drink). The condition Cond 2 inspects the results of the comparison operation CMP.1. If the result is "not big" (! GT. CMP.1), then a test will be performed to see if condition Cond 3 is equal to the result (EQ. CMP.1). If the result is "not equal" (! EQ. If CMP.1), the control will return to State 0 and wait for other coins to be loaded. If Cond 3 is EQ, State 4 will generate a control signal to dispense the beverage, and the macro instruction CLR.1 (cv) will reset cv to 0, waiting for another customer.

If the total amount of coins that were loaded exceeds the beverage price, State 30 will output an operation to "return coin". Further, macro DECR.1 (cv) will reduce the value of cv by the total amount of the coins that were returned. In State 31, cv and PR will be compared again. If cv is still larger than PR, the control will go to State 30 and return other coins. The condition Cond 5 will perform testing to determine if the CMP.2 results are equal, and will either dispense the beverage (droppop) or will return to state 0, and waiting for other coins to be loaded. The macros shown in Figure 10 that cooperate with the states correspond to those defined in Table 1 above, and will determine the specific operations to be executed in each state.

Appendix A shows an intermediate file or state list that may be created in the flowchart shown in Figure 10. This state list may be created as an output from EDSIM program 20, and may be used as input into PSCS program 30 (Figure 3).

Figure 11 shows each macro that may be used in the flowchart of Figure 10 and the corresponding hardware blocks. The comparison macro CMP (A,B) will generate a register to store Value A, a register to store Value B, and a block for the comparator. Further, it will generate a control passage for the system controller for use with the EQ, LT, and GT signals that may be generated as a result of the comparison. Another macro ADD (A, B, C) will generate registers for each Input value A and B, along with register and adding machine (adder) blocks for use with Output value C. The macro DECR(A) will generate a counter block. PSCS program 30 will generate the macros used in the flowchart of Figure 10 for the corresponding hardware elements, and will generate the hardware blocks shown in Figure 12. The generation of the blocks shown in the figure by PSCS program 30 is based on Rule 1 and Rule 2 of the example rules that have already been described.

Figure 13 shows the connection of the block of Figure 12 along with the data passage and the control passage. Rule 3 was used by the data/control synthesizer program 31 in the generation of the data and control passages.

Figure 14 shows the results of the optimization of the circuit through the application of Rule 4 to eliminate redundant registers. As a result of the application of this rule, Registers R2, R3, R7, R8, and R9 in Figure 13 were removed. Figure 15 shows a block diagram of the results of further optimization with the omission of

redundant comparators. This optimization was completed in the PSCS program as a result of the application of Rule 5.

Because the system controller block was determined, the other required hardware blocks and data of the integrated circuit, as well as the control passage and PSCS program 30 will generate net list 15 to determine these hardware portions and the required connections. The mask data for the manufacture of integrated circuits from this net list may be directly created using the available VLSI CAD tools.

```

APPENDIX A
name rpop;
data path @ic<0:5>, cv<0:5>, sum<0:5>, @pr<0:5>;
{
state4 : state0;
state3 0 : state3 1;
state2 1 : state2 2;
state2 0 : state2 1;

state0 : !cp state0;
state0 : cp state1;
state1 : cp state1;
state1 : !cp state2 0;
state2 2 : GT.CMP.1 state3 0;
state2 2 : !GT.CMP.1 * EQ.CMP.1 state4;
state2 2 : !GT.CMP.1 * !EQ.CMP.1 state0;
state3 1 : GT.CMP.2 state3 0;
state3 1 : !GT.CMP.2 * EQ.CMP.2 state4;
state3 1 : !GT.CMP.2 * !EQ.CMP.2 state0;
state3 0 : returncois;
state3 0 : DECR.1(cv);
state4 : dropop;
state4 : CLR.1(cv);
state3 1 : CMP.2(cv,pr);
state2 2 : CMP.1(cv,pr);
state2 1 : MOVE.1(sum,cv);
state2 0 : ADD3.1(ic,cv,sum);
}

```

4. BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1a shows a representation of the design at the functional level for a portion of a desired circuit shown in flowchart format.

Figure 1b shows a representation of the design at the structural level for an integrated circuit.

Figure 1c shows a representation of the design at the physical layout-level circuit that may be used in the manufacture of an integrated circuit chip.

Figure 2 is a block diagram showing how the integrated circuit will produce the mask data from the flowchart description as a result of the KBSC system of the present invention.

Figure 3 is a diagram showing the details of the first portion of the KBSC system.

Figure 4 is a diagram that shows how the ASIC design system of the present invention extracts an integrated circuit hardware element from the element library.

Figure 5 is a diagram that shows an example flowchart to determine the sequence of the functional operations that may be executed by an integrated circuit.

Figure 6 is a diagram illustrating the structure showing the hardware blocks and interconnection conditions for the integrated circuit defined in Figure 5.

Figure 7 is a diagram that shows the flowchart editor window.

Figure 8 shows the flowchart simulator window.

Figure 9 shows the steps included in the element list generation.

Figure 10 is a flowchart that shows an example of a vending machine system.

Figure 11 shows the hardware portions corresponding to each of the three macros used in the flowchart in Figure 10.

Figure 12 is an initial block diagram that shows the hardware portion of the integrated circuit determined in the flowchart in Figure 10.

Figure 13 is a block diagram corresponding to Figure 12 showing interconnections between each block.

Figure 14 is a block diagram corresponding to Figure 13 after optimization of the register.

Figure 15 is a block diagram corresponding to Figure 14 after further optimization.

10 ...KBSC, 11 ...Flowchart,
15 ...Net list, 16 ...CAD tool,
18 ...Mask data, 23 ...Macro library,
34 ...Element library, 35 ...Knowledge base

Patent	applicant
	Internatio
	nal Chip Corporation (and 1
	other)
Agent	Patent attorney Tamotsu Aoyama
	(and 1
	other)

Clean copy of the drawings (no changes to the content)

Figure 1a

Functional Level:
機能レベル

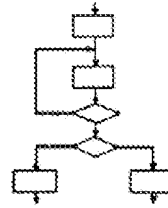


Figure 1b

Structural Level:
構造レベル

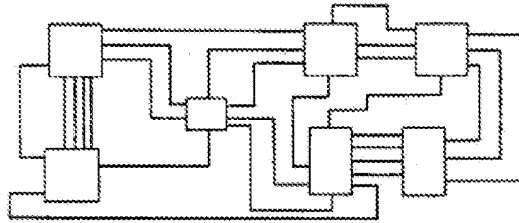


Figure 1c

Physical Layout Level:
物理的レイアウトレベル

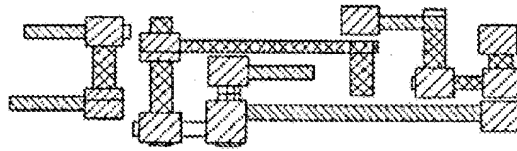


Figure 2

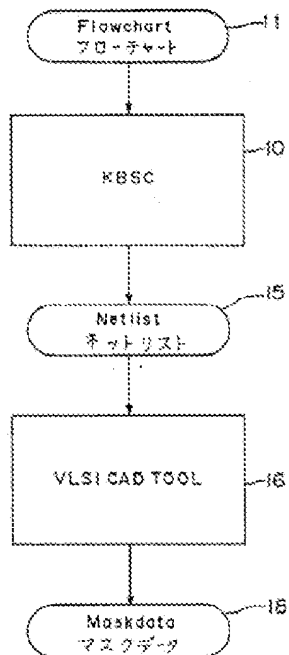


Figure 9

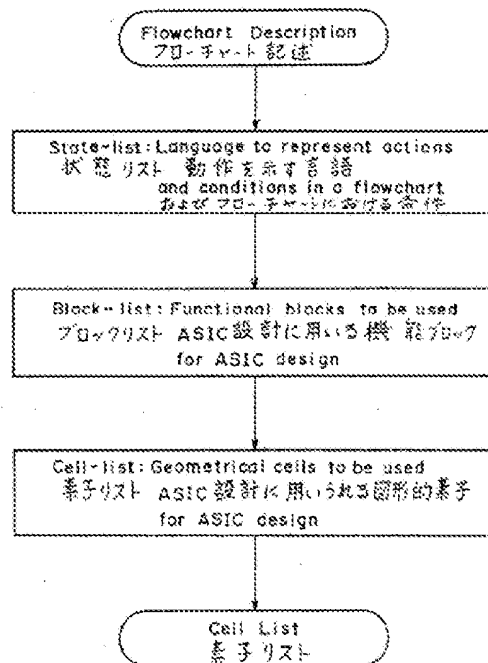


Figure 3

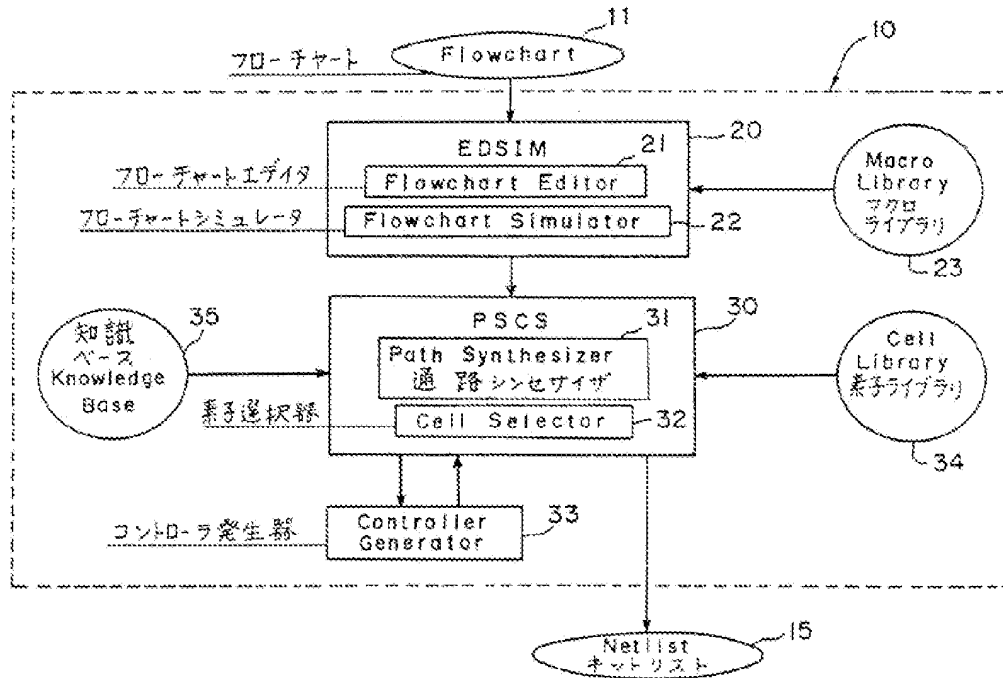


Figure 4

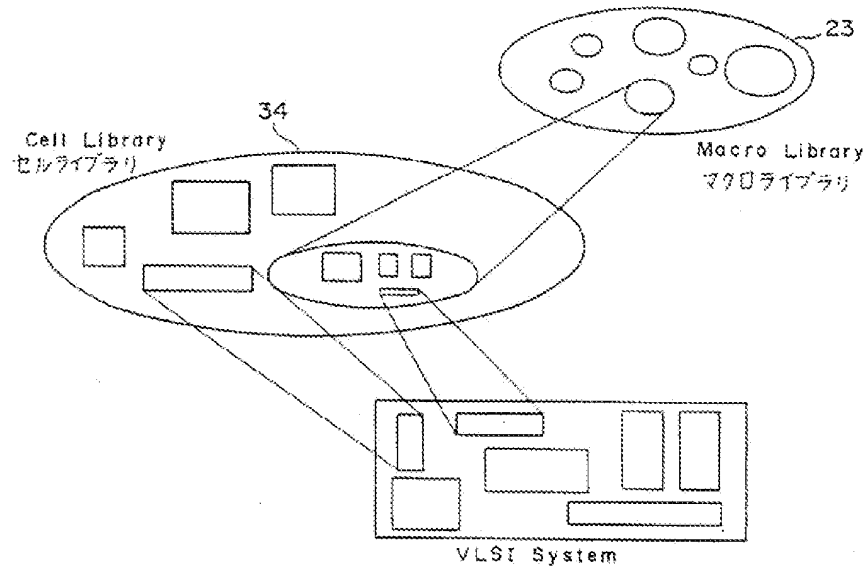


Figure 5

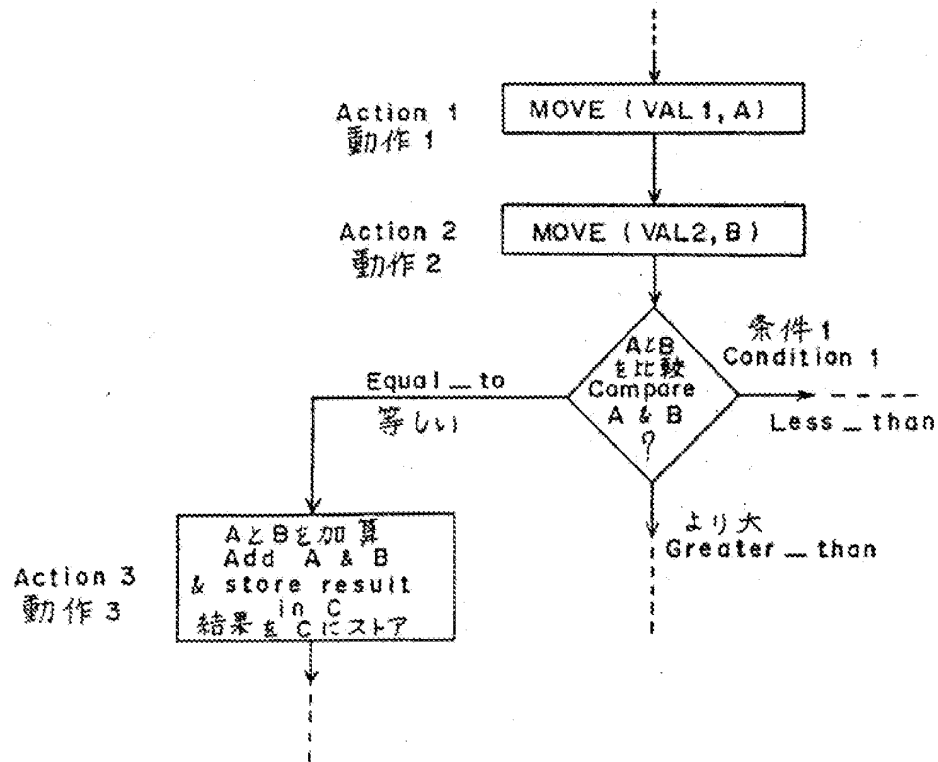


Figure 6

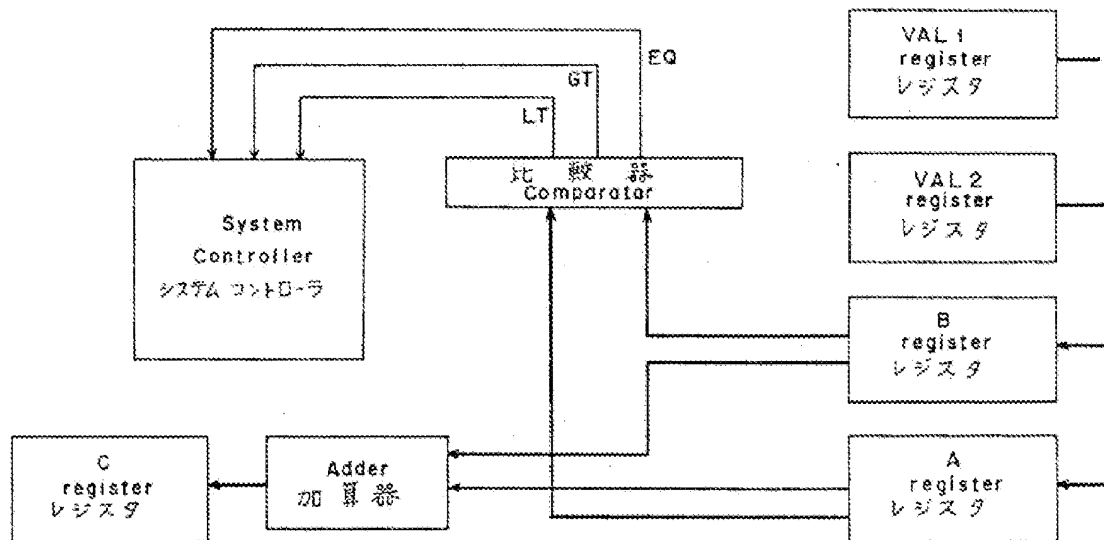


Figure 7

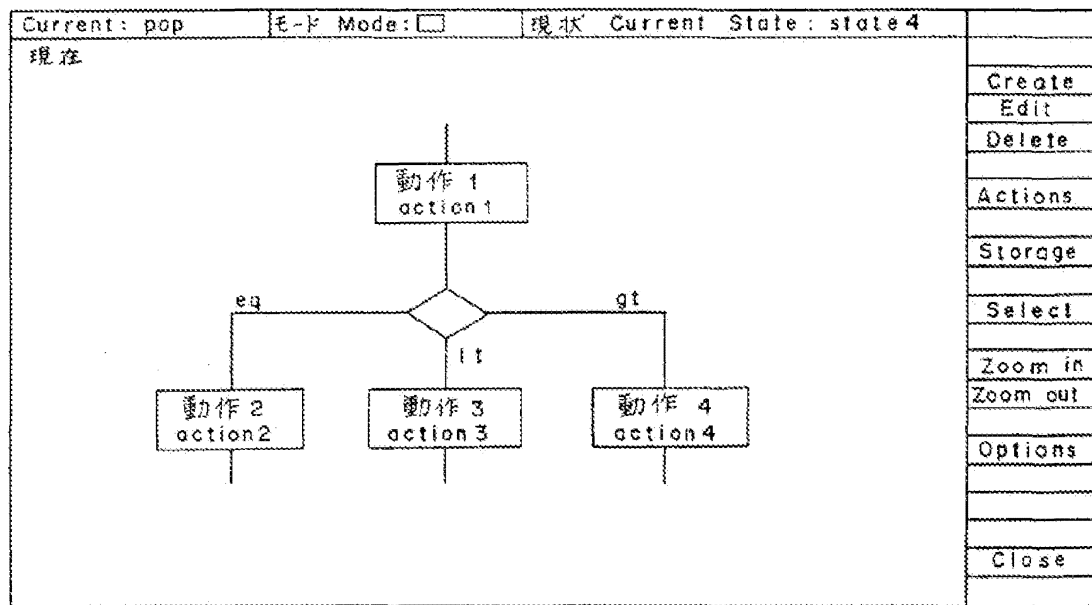


Figure 8

Edit Data	Set Breaks	Step	History ON	Cancel
Show Data	Clear Breaks	Execute	Detail	Help
Set State	Show Breaks	Stop		Close

* * * Ready * * *

Figure 10

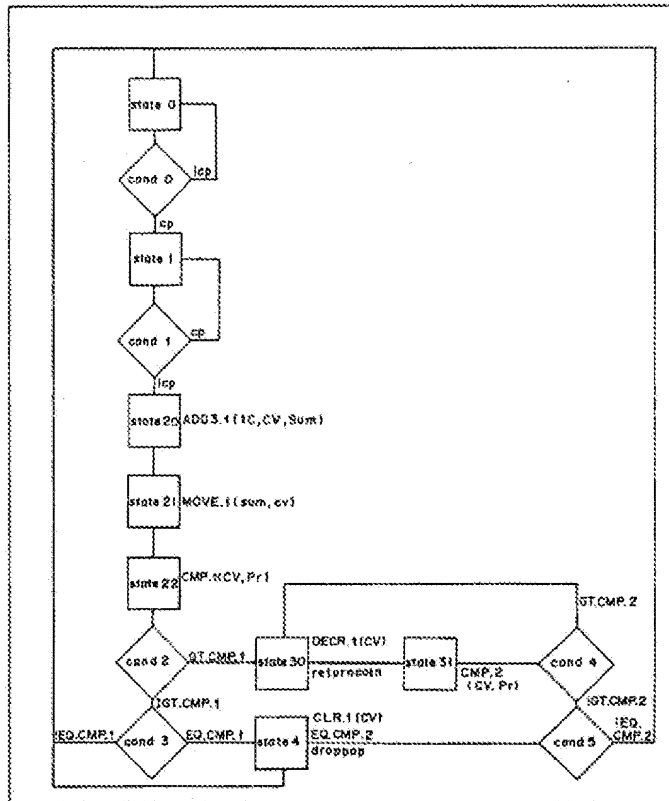


Figure 11

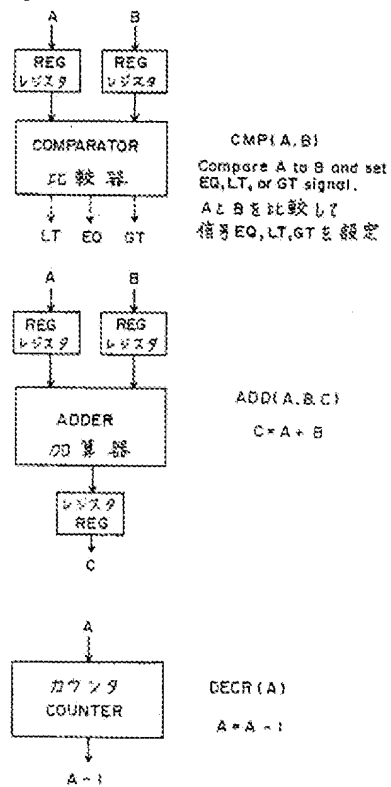


Figure 12

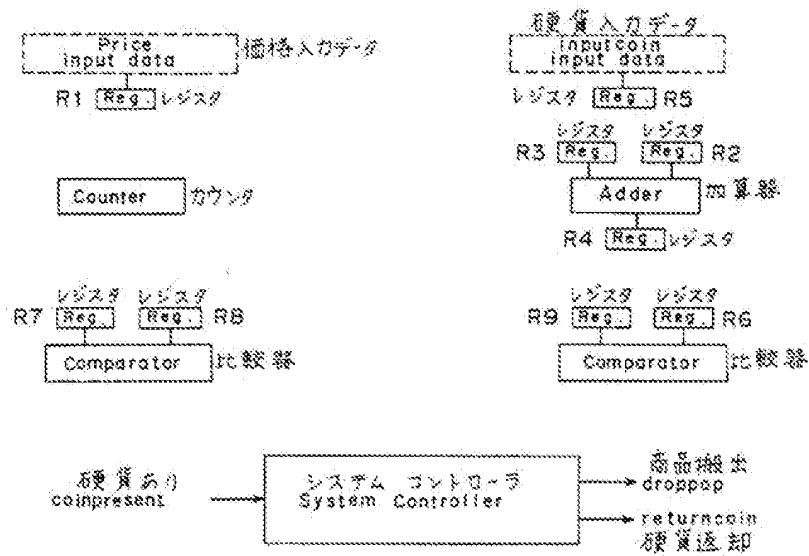


Figure 13

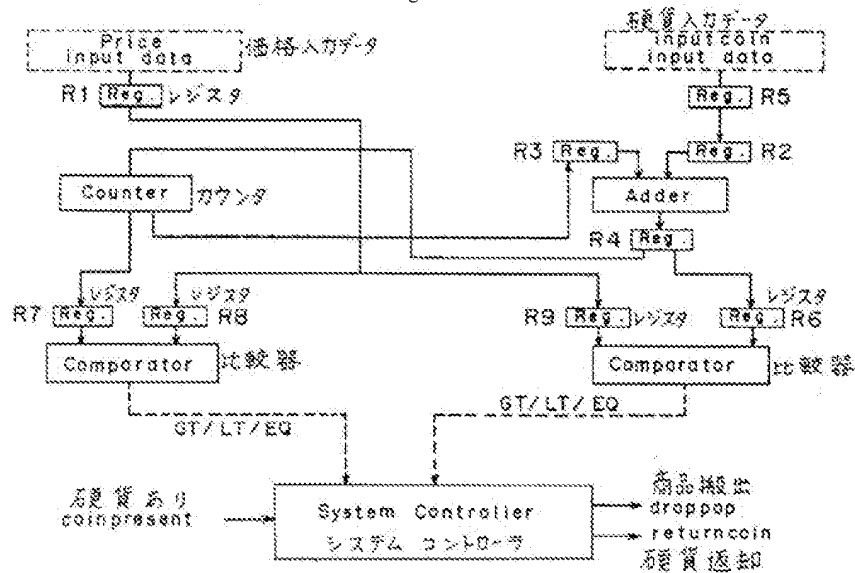


Figure 14

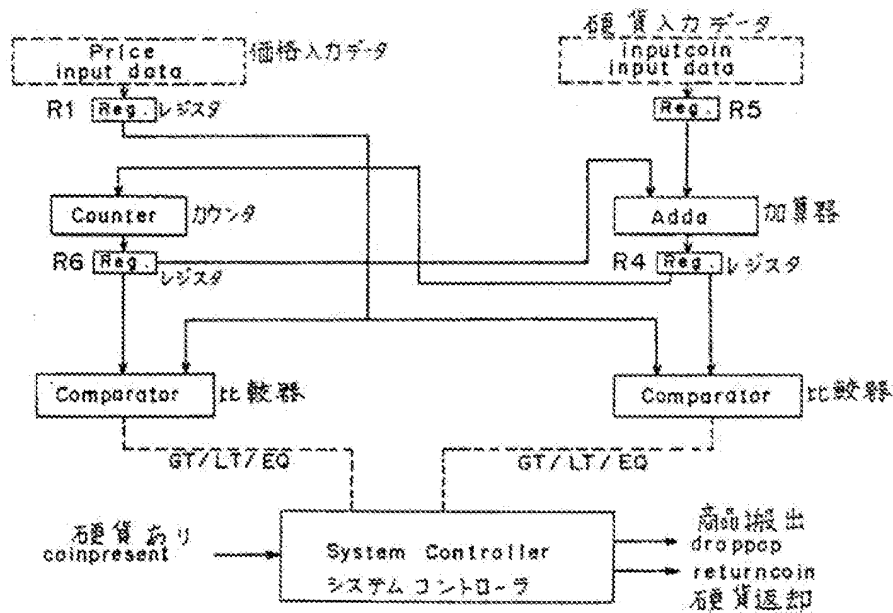
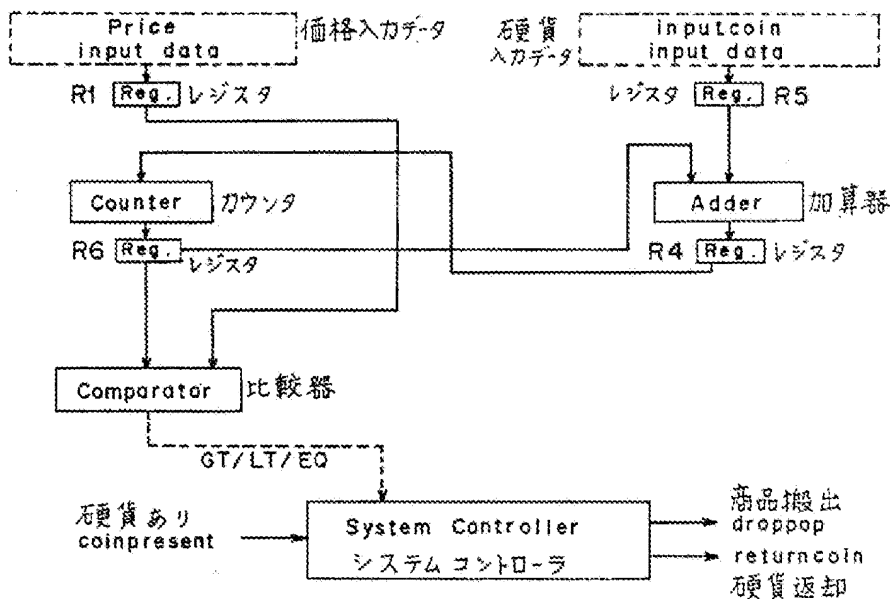


Figure 15



Amendment (Formality)

To: Commissioner of the Patent Office May 18, 1989

1. Identification of the trial case

[stamp:] [illegible]

1989 Japanese Patent Application No. 007550

2. Title of the Invention

Computer-aided design system for ASIC

3. Party making the amendment

Relationship to the trial case Patent applicant

Name International Chip Corporation

(and 1 other)

4. Agent

Address : 2-1-61 Shiromi, Chuo-ku, Osaka-shi, Osaka 540

Twin 21 MID Tower Telephone (06) 949-1261

Name Patent attorney (6214) Tamotsu Aoyama

[stamp:] Seal of Patent attorney Tamotsu Aoyama

5. Date of the amendment order

April 25, 1989 (Mailing date)

6. Target of the amendment: Drawings

7. Details of the amendment: As

shown in the clean copy
and attachment of the
drawings initially attached
to the application (no
change to the content)

[seal:] Japan Patent Office 05/19/1989 [illegible]