数理解析研究所講究録
第482巻 1983年 73-97

Hardware Algorithms and Logic Design Automation

--- An Overview and Progress Report ---

Shuzo YAJIMA and Hiroto YASUURA

Faculty of Engineering
Kyoto University

1. Introduction

Advances in the fabrication technology of VLSI circuits will soon make it feasible to implement highly parallel computing systems consisting of hundreds or of thousands of computing elements. These highly parallel systems will operate cooperatively with software and achieve tremendous speed improvements of digital computing systems. Many researches have been carried out to establish effective design methods for either general or special purpose highly parallel systems[1].

Design of efficient algorithm for parallel computation is the key problem of design of highly parallel hardware systems as well as software. In this article, we will discuss the design problem of parallel algorithms, called hardware algorithms.

Various hardware algorithms have been proposed for several practically important problems such as sorting, arithmetic operations, matrix arithmetics and pattern matching[2]-[6]. To analyze these algorithms, several theoretical discussions

have been going on and a new complexity measure of parallel computation suitable for VLSI, called area, has been proposed [7][8]. VLSI and hardware algorithms make a new area of theoretical research on computational complexity. A theory of design and analysis of hardware algorithms for VLSI systems will be established in several years ahead.

Logic design automation systems will be indispensable tools for design of large highly parallel systems. A silicon compilation system will be developed which generates mask patterns for VLSI fabrication from a high level hardware algorithm description[9]. Design verification tools are also important components of design automation systems, since highly parallel computation is inherently too complicated for designers to think without any tools such as simulator and verifier.

In this article, we will briefly survey topics of researches on hardware algorithms and design automation systems. Section 2 and 3 are an overview and progress report of the research on hardware algorithms. Section 4 is the progress report of development of an interactive logic design and verification support system ISS.

In the next section, hardware algorithms of integer multiplication and sorting are presented as examples of hardware algorithms. A general consideration of hardware algorithms for VLSI is also discussed. In section 3, some results of theoretical researches on the complexity of hardware algorithms are presented. Design automation systems and computer aided design will be discussed in section 4. As an example of design verification tools, ISS developed by our group is presented.

2. Design of Hardware Algorithms

2.1 Integer Multiplication

Integer multiplication is widely used as a basic operation in general purpose computers, in process controllers and signal processors. Many high speed algorithms for integer multiplication in software and hardware have been proposed and used practically. In table 1, several hardware algorithms for integer multiplication are compared.

In applications not required so much high speed computation, add-and-shift multiplication is generally used. The speed of computation is improved, when a carry look-ahead adder is adopted[10]. Serial multiplication is implemented in signal processing in which operands are input serially[11]. These above algorithms are implemented by sequential circuits.

For high-speed multiplication by combinational circuits, array multiplication and matrix generation-reduction scheme are developed and implemented for practical use. Array multiplication is attractive for their compactness and regularity of its iterative array structure using one basic circuit type, but their speed of operation increases linearly with the operand length and thus slow for large words[10]. Matrix generation-reduction scheme is much faster for large operands since their speed of operation increases with the logarithm of the operand length [12]. The basic idea of this algorithm was proposed by Karatsuba and Ofman, and the most popular circuit based on it is known as Wallace's tree[13][14]. Several papers discussed about multiplier based on this algorithm with

algorithm	size	area	speed of computation
add-and-shift multiplication (ripple carry adder)	n	n	n^2
add-and-shift multiplication (carry look-ahead adder)	n	nlogn	nlogn
serial multiplication	n	n	n
array multiplication	n^2	n^2	n
matrix generation-reduction multiplication	n^2	n^2logn	logn
Brent-Kung's algorithm	-	nlogn	$\sqrt{n}$logn
Shonhage-Strassen's algorithm	n(log n(log log n))	-	logn

n: the length of operands

Table 1. Hardware algorithms for Integer Multiplication

n	8	16	32	64
matrix generation-reduction	22/672	24/2516	30/9064	34/35207
array	29/528	61/2336	125/9792	253/40064

(depth/size) 4-input NOR/OR gates

Table 2. Depth and Size of Multiplier

higher performance[10][12].

In table 2, an evaluation of the size (the number of gates) and the depth (the speed of computation) of circuits designed according to these two algorithms. Gates used in the design are 4-input NOR/OR gates. The circuit based on matrix generation-reduction algorithms is realized as a combination of Booth's algorithm, Wallece's tree and a carry look-ahead adder. The depth of the circuit is extremely smaller than the array multiplier for large n. This example shows that the design of a good hardware algorithm results in tremendous improvement on efficiency.

Although the size of circuits of these two algorithms are the same order, namely $O(n^2)$, the upper bounds of the area on VLSI have quite different order, $O(n^2)$ for array multiplication and $O(n^2 \log n)$ for matrix generation-reduction one. This difference is caused by the difference of complexity of interconnection in the circuits. In the next section, we will discuss a new circuit complexity analysis technique using the measure, called area.

Brent-Kung's algorithm in table 1 achieves the best upper bound of the area-time product[8]. Shonhage-Strassen's one is the best upper bound of the number of Boolean operations required to n-bit integer multiplication[15].

2.2 Sorting

Sorting is one of the most important operations in data processing. Many sequential and parallel sorting algorithms have been developed and practically used. In table 3, several sorting algorithms are compared.

Algorithm	the number of processing elements	the speed of computation
Bubble sort	O(1)	$O(n^2)$
Heap sort	O(1)	O(n log n)
Bitonic sort	O(n)	$O(\log^2 n)$
Rebound sort	O(n)	O(n)
Parallel enumeration sort	O(n)	O(n)
Parallel merge sort	O(log n)	O(n)
Parallel heap sort	O(log n)	O(n)
Sorting on mesh	O(n)	$O(n^{1/2})$
Parallel distributive sort	O(n)	O(log n)
Sorting by combinational circuit	$O(n^2)$	O(log n)

n: the number of sorted element

Table 3. Algorithms for Sorting

Bubble sort and Heap sort are softwawe algorithms and have time complexity $O(n^2)$ and $O(n \log n)$, respectively[15]. Heap sort is one of the fastest algorithms of software, because it is easy to show that the lower bound of time complexity of software algorithms is $\Theta(n \log n)$.

Many hardware algorithms for sorting have been proposed and some of them are implemented. Muller and Preparata showed that sorting of n elements can be performed by a combinational circuit with depth $O(\log n)$ and size $O(n^2)$[16]. Several hardware algorithms on multiprocessor systems have been proposed such as algorithms on mesh connected processors by Thompson-Kung[17] and by Nassimi-Sahni[18]. Parallel distributive sort by Maekawa[19] and Wilsow-Chow[20] are also this kind.

When we consider a sorting circuit which is attached to conventional computer systems, we will assume that data are transmitted one by one between the sorting circuit and memory devices. Rebound sort by Chen et. al.[21], Parallel enumeration sort by Yasuura and Takagi[22], Parallel merge sort by Todd[23], and Parallel heap sort by Tanaka[24] are all developed under the following assumptions; (1) a sorting circuit is separated from memory devices and (2) data transmission between the circuit and memory devices is serial. Under these assumptions, processing for sorting cannot be faster than data transmission. Since the time required for sorting in these algorithms is linearly proportional to the number of sorted elements, these algorithms achieve optimum order on time. In these algorithms, processes of sorting are efficiently overlapped with the input/output time.

Rebound sort and Parallel enumeration sort require O(n) processing elements, but each element has constant size and realized by a very simple circuit. Circuits for these algorithms have linear array structure and simple communication structure. Rebound sort is suitable for implementation by magnetic bubble circuits. Parallel enumeration sorting is implemented on the Bus Connected Cellular Array structure detail of which is discussed in the next subsection.

Parallel merge sort and Parallel heap sort require only O(log n) processing elements. However, each processing element should possess O(n) memory. Since memory can be integrated in higher density than logic circuits, several circuits have been implemented based on these algorithms using commercial LSI's.

2.3 Hardware Algorithms for VLSI

Kung and his group proposed the systolic algorithms which is suitable for VLSI implementation[4]-[6]. Systolic algorithms have the following properties[6].

(1) The algorithm can be implemented by only a few different types of simple cells.

(2) The algorithm's data and control flow is simple and regular, so that cells can be connected by a network with local and regular interconnections.

(3) The algorithm uses extensive pipelining and multiprocessing. Typically, several data streams move at constant velocity over fixed paths in the network, interacting at cells where they meet. In this way a large number of cells are active at one time so

that the computation speed can keep up with the data rate.

The first property reduses the cost of design and test, since a designer only designs and tests a few different, simple cells. Regular interconnection in the second property implies that the design can be made modular and extensible. This also means that the area for wiring on a chip will be reduced and propagation delay caused by these wires will decrease. By pipelining and multiprocessing, one can meet the performance requirement of a circuit. Pipelining makes it possible to overlap processing and input/output effectively.

Kung and his coauthors developed many systolic algorithms on one dimensional cellular arrays, two dimensional square meshes and hexagonal meshes.

We developed hardware algorithms which are realized on Bus Connected Cellular Array (BCA). Algorithms on BCA possess the following properties added to (1)-(3) of systolic algorithms [22][25].

(4) The algorithm uses grobal communications through buses. The communication control of the grobal communication is simple and distributed.

(5) Input and output scheme is simple and it is easy to attach to other circuits in a system. One can extend the circuit only connecting chips which include smaller circuits without changing input/output scheme.

(6) The restriction on performance of the algorithm shold be relaxed as much as possible. Processing time should depend on only the size of problem not on the size of the circuit.

The grobal communication using buses improves the speed of algorithm drastically and reduces the complexity of communication. Algorithms must be designed under realistic assumption of input/output protocols. Highly parallel input and output increases infeasiblly the complexity of communication of the outside of the algorithm, though the algorithm seems to achieve high performance. Hardware algorithms are inherently restricted their ability of processing by the size of circuits. However, algorithms should process problems smaller than their ability in time proportional to the size of problems. On BCA algorithms, these properties can be easily realized using grobal bus communication.

We proposed a sorting algorithm on BCA, called Parallel enumeration sort[22]. This algorithm can be introduced to conventional computer systems without changing their architecture. The processing time is linearly proportional to the number of data for sorting. The sorting circuit consists of a linear array of one type of simple cells each of which includes two registers, a comparator and a counter. These cells are connected by two buses. Since the circuit is extensible only by connecting the same circuit, we can implement a large circuit connecting chips including the circuits.

We have also developed BCA algorithms for pattern matching, matrix multiplication, very long integer multiplications and join opration in relational databases.

3. Analysis of Hardware Algorithms

3.1 Time Complexity

Combinational logic circuits are the most fundamental circuits in digital systems, which can realize the most highly parallel computation. Many researches have been carried out on depth and size of circuit required to realize Boolean functions [2][25][26]. Results of these researches can be applied to analysis of hardware algorithms.

The delay complexity of Boolean function is the smallest depth of circuits which realize f. Boolean functions which depend essentially on n variables have delay complexity $\Omega(\log n)$. Functions in several important classes such as linear functions, symmetric functions and threshold functions have delay complexity proportional to log n. It is also well known that there are functions required O(n) depth of circuits for their realization [26].

Integer addition and multiplication can be performed by circuits with depth O(log n), where n is the length of operands. For division and square rooting, algorithms with depth $O(\log^2 n)$ were developed. But it is not known whether these operations can be computed by circuits with depth O(log n) or not.

Unger proposed a method to construct a circuit with the logarithmic depth for a class of functions which can be computed by sequential circuits in linear time[27]. This class of functions includes many practical functions such as binary addition, comparison, counting and so on. We showed a method to reduce the depth of a circuit as a generalization of Unger's

method[28].

Moreover, we proved an interesting general result on relation between complexity of software algorithms and of hardware algorithms as follows.

Theorem[29] A function which can be computed by a T(n)-time bounded and S(n)-tape bounded deterministic Turing machine can be computed by a combinational circuit whose depth is proportional to S(n)log T(n).

In the proof of this theorem, we obtained a construction method of the combinational circuit. Thus we can construct a combinational circuit with depth $O(\log^{k+1}n)$ for a function which is computed in $\log^{k}n$ space by a polynomial time software algorithm. This result shows an upper bound of speed up ratio of hardware algorithm and software one.

3.2 Area Complexity

In VLSI circuits, it is well known that the area of a circuit depends on not only the number of logic elements included in the circuit but also the area for the wiring and input/output terminals[4]. Thompson[7] and Brent-Kung[8] proposed mathematical models of VLSI circuits and provided several techniques to evaluate the area of circuits theoretically. On these mathematical models of VLSI, many works have been carried out to estimate performance of VLSI circuits by a measure, area-time product[3]. Leiserson[30] and Valiant[31] discussed the area required for layouts of tree and planer graph circuits into the VLSI model.

A VLSI model is defined as follows:

(1) A circuit is embedded into a convex planer region R.

(2) Wires have minimal width λ (a positive constant).

(3) At most ν ($\nu \geq 2$, a positive constant) wires can overlap at any point of R.

(4) Logic elements each contain a $\lambda \times \lambda$ square and their shapes and area are given for each sort of elements.

(5) No logic element overlaps other logic elements and wires in R.

(6) Input and output of the circuit is performed through wires on the boundary of R.

The last condition of the definition of VLSI model was introduced by us and called the boundary conditions[32]. The boundary conditions is very realistic assumption of actual VLSI circuits. We can show that the area required for embedding of a binary tree circuit with n nodes is $\Theta(n \log n)$ under the boundary condition, though it is just $O(n)$ on the model except the boundary condition.

It is very important to estimate the area of a circuit at the stage of logic design before layout of the circuit. Several results have been obtained on evaluation methods of the area from feature of the circuit such as depth, width, size and structure which are easily measured from the logic design.

Trade-off between time and area is the most interesting subject of theoretical researches. Many results have been presented on area-time products of practical functions such as integer multiplication, integer addition, comparison, sorting,

FFT, matrix multiplication and so on[3]. On combinational circuits, trade-off between depth d and area A for an n-variable function is shown as follows;

$$A \log d = \Omega(n \log n)^{[32]}.$$

4. Logic Design Automation

4.1 Computer Aided Design and Design Automation

Computer Aided Design (CAD) or Design Automation (DA) technology is almost indispensable in large, complicated logic design. A goal of researches on CAD/DA systems is to develop a silicon compilation system[9]. In the silicon compilation system, a designer describes a design of a hardware algorithm in a high-level hardware algorithms description language which is free from many constraints caused by physical implementation. The designer also specifies several conditions on realization of the algorithm by silicon devices such as speed, area, technology used for implementation and so on. The silicon compilation system generates a logic design which realizes the algorithm and satisfies the specified conditions. Moreover, the system translates the design into several information to control fabrication processes of VLSI. One can make his own VLSI chips only writing a specification of hardware algorithm in a high-level language as well as software in high-level languages. There are still many problems that should be resolved for implementation of the silicon compilation system.

Many kinds of CAD techniques and systems for logic design have been developed and are in use: hardware description

language, logic diagram editor, logic function minimization programs, optimum circuit generation programs, logic simulator, logic design verifier, test pattern generator, etc. These systems are developed to relieve logic designers from troublesome, laborious work in logic design and verification. However, they are not utilized to a full extent because designers can not use them in an integrated way. Since the logic design stage is a repetition of design and verification, a logic design system should be an integrated one including a design language, editing system, design-aid programs, a design verification tool, etc. on a standardized data management. A CAD system for logic design usually needs some interaction by man for processing. An interactive sophisticated user interface is keenly needed for CAD systems.

We have developed Interactive Simulation System (ISS) --- an interactive logic design and verification support system for structured logic design [33]. One can carry out logic design in a structured way by utilizing functions of ISS all interactively. ISS has the following features.

(1) Interactive and Integrated Logic Design System

EDITOR, TRANSLATOR, LINKER, Interactive Simulator (IS) and other peripheral programs can be utilized easily by terminal commands (See Fig.1). A designer performs logic design and its verification alternately without bothering about management of many kinds of design data.

(2) Interactive Simulator(IS)

IS is used for design verification in ISS. A designer can

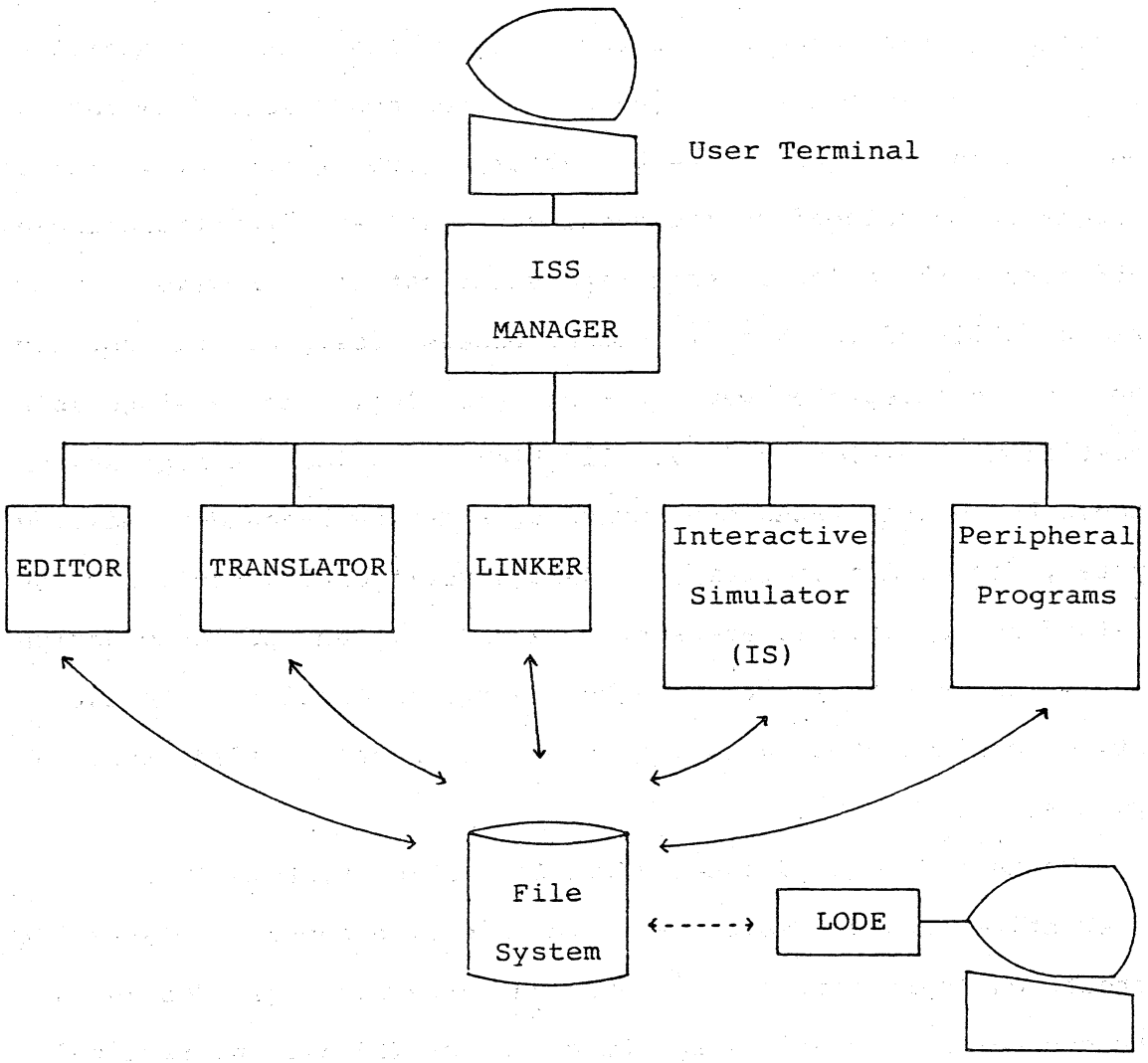

Fig. 1 Configuration of ISS

control the simulation steps interactively to find out design errors at an early stage of design cycle. IS can simulate a design described at multi-levels (gate, functional, and register transfer) in a structured logic design.

(3) Structured Hardware Design Language (SHDL)

SHDL is used to describe logic designs in ISS. SHDL can describe a design as a hierarchically constructed set of modules. Each module is described with its structure or behavior. There are three kinds of behavioral description which enable multi-level description of a design.

(4) Design Data File

The file system storing description data or simulation data is configurated "module oriented". ISS has the file system as a common interface among programs and is integrated around it.

4.2 Design Verification in ISS

A logic simulator is used for design verification in order to find out design errors based on simulation results which do not satisfy the design specification. Verification by simulation shows that "the design contains errors", not that "the design is correct". The usefulness of a simulator as a design verification tool depends on how early a user can find out design errors with it. The turn around time to feed back the verification results to the design step becomes shorter, the earlier an error can be found.

Interactive Simulator IS in ISS is an interactive simulator which has following functions.

(1) Interrupting function

A user can set breakpoints in simulation steps using subcommand AT, STEP, WHEN and ON. When the following conditions occur, the simulation process is suspended and falls into the suspended state.

AT condition occurs when the simulation time reaches the specified one. AT subcommand is used to set this condition.

STEP condition occurs when the simulation for the specified time interval ends. STEP subcommand is used to set this condition.

WHEN condition occurs when the simulated module behaves in the specified manner or falls into the specified state. WHEN subcommand is used to set this condition.

INPUT CONSTRAINTS ERROR condition occurs when the simulated module or submodules (blocks) receive input patterns which do not satisfy the input constraints of the module or submodules. ON subcommand is used to make this condition in effect. In SHDL the design specification of a module can be partially described as constraints on its input patterns. In general the module is designed to have the specified function only of the input patterns which satisfy the input constraints. A designer can use this useful information in design verification with a simulator. When an input pattern to a module violates input constraints, the design containing the module must have some errors and the module behaves incorrectly. The errors are likely to be in the design of modules which supply the input pattern or in the input constraints itself. A designer can tell of what kind the error is

from the type of the input constraints not satisfied. Moreover, a designer can easily prepare input patterns for a module according to input constraints description. Waste of computation time on unnecessary simulation for wrong input patterns. IS has a function to check input patterns during execution whether they satisfy input constraints of modules to which they drive. INPUT CONSTRAINTS ERROR condition occurs when these input constraints are violated.

While the simulation process stays in the suspended state, a user can examine the status of the module precisely and resume the simulation with modified input patterns specified adaptively.

(2) Display function of simulation results

A user can examine the values of signal lines and contents of memory elements at the specified simulation time in real time. LIST subcommand is used to display simulation results.

(3) Modification function of input patterns

A user can modify input patterns dynamically according to the simulation results. EDITWAVE subcommand is used.

(4) Simulation resuming function

A user can resume the simulation from the past as well as the present simulation time. When the input patterns are changed, he can go back to the time at which the changes are effetive and resume the simulation. This technique reduces much computation time. GO and RUN subcommand are available for this function.

Using these functions of IS, a designer can easily find design errors in the early stage of design cycle. It shortens the whole time spent for logic design. In Fig.2, an example of

```
00010 ISS: SIMULATE ONE_OF_FOUR_DECODER
00020 ENTER UNIT TIME(NS): 5
00030 SIMULATE: AT 51
00040 SIMULATE: GO
00050 <AT> TIME=< 51: 55>
00060 SIMULATE: LIST (SHR.QA SHR.QB SHR.QC SHR.QD COUNTER.JK1.Q COUNTER.JK2.Q)
00070 SHR.QA          < 46: 50>
00080 0
00090 SHR.QB          < 46: 50>
00100 0
00110 SHR.QC          < 46: 50>
00120 0
00130 SHR.QD          < 46: 50>
00140 0
00150 COUNTER.JK1.Q   < 46: 50>
00160 0
00170 COUNTER.JK2.Q   < 46: 50>
00180 0
00190 SIMULATE: WHEN LOAD (SHR.QA=C | SHR.QB=C | SHR.QC=C | SHR.QD=C)
00200 SIMULATE: GO
00210 <WHEN> LOAD TIME=< 71: 75>
00220 SIMULATE: LIST (SHR.QA SHR.QB SHR.QC SHR.QD)
00230 SHR.QA          < 66:70>
00240 1
00250 SHR.QB          < 66: 70>
00260 1
00270 SHR.QC          < 66: 70>
00280 0
00290 SHR.QD          < 66: 70>
00300 1
00310 SIMULATE: GO
00320 <WHEN> LOAD TIME=<131:135>
00330 SIMULATE: LIST (SHR.QA SHR.QB SHR.QC SHR.QD)
00340 SHR.QA          <126:130>
00350 X
00360 SHR.QB          <126:130>
00370 1
00380 SHR.QC          <126:130>
00390 1
00400 SHR.QD          <126:130>
00410 0
00420 SIMULATE: LIST L (1:130)
00430 L               <  1:130>
00440 XXXXX XXXXX XXXXX XXXXX XXXXX X
00450 SIMULATE: EDITWAVE
00460 ENTER LINENAME: L
00470 FROM: 86
00480 ENTER VALUE:
00490 0000000000000Q000000000000000000000000000000000000;
00500 SIMULATE: GO 86
00510 <WHEN> LOAD TIME=<131:135>
00520 SIMULATE: LIST (SHR.QA SHR.QB SHR.QC SHR.QD)
00530 SHR.QA          <126:130>
00540 0
00550 SHR.QB          <126:130>
00560 1
00570 SHR.QC          <126:130>
00580 1
00590 SHR.QD          <126:130>
00600 0
```

Fig.2 Interactive simulation with IS

interactive simulation is shown. ISS is implemented on FACOM M-200 in Data Processing Center of Kyoto University. Programs are mostly developed in PL/I and about 18 thousands steps in total. Users can use ISS from their TSS terminal interactively.

5. Conclusion

In this paper, we discussed several problems in the design of hardware algorithms and logic design automation. The theory of complexty of logic circuits and parallel computation will form the foundation of design of hardware algorithms which will become more important for larger VLSI systems. Especially, the relation between the complexities of software and hardware is very important for practical system design, because systems are combination of software and hardware.

Design automation is one of most highlighted fields in computer science. In logic design automation, we still have many hard problems to resolve for development of an efficient design system. Researches on high-level hardware design languages, automatic logic design from descriptions of these languages and design verification techniques for large systems have been increasing. Several techniques developed in the software engineering will be applied to these area.

References

[1] "Highly Parallel Computing" Edited by L.S.Hayens, IEEE Computer, vol.15, no.1, pp.7-96, Jan. 1982.

[2] S.Yajima, H.Yasuura and Y.kambayashi, "Design of Hardware Algorithms and Related Problems", IECE Technical Rep. AL81-86, Dec. 1981 (in Japanese).

[3] N.Tokura, "VLSI Algorithms and Area-Time Complexity", Joho-Shori vol.23, no.3, pp.176-186, March 1982 (in Japanese).

[4] C.A.Mead and L.A.Conway, "Introduction to VLSI Systems", Addison-Wesley, Reading, Mass., 1980.

[5] H.T.Kung, "The Structure of Parallel Algorithms", Advanced in Computers, vol.19, Academic Press, 1980.

[6] M.Foster and H.T.Kung, "The Design of Special-Purpose VLSI Chips", IEEE Computer, vol.13, no.1, Jan. 1980.

[7] C.D.Thompson, "Area-Time Complexity for VLSI", Proc. 11th Symposium on the Theory of Computing, pp.81-88, May 1979.

[8] R.P.Brent and H.T.Kung, "The Area-Time Complexity of Binary Multiplication", JACM, vol.28, no.3, pp.521-534, July 1981.

[9] J.P.Gray, "Introduction to Silicon Compilation", Proc. 16th DA Conference, pp.305-306, June 1979.

[10] K.Hwang, "Computer Arithmetic:Principle, Architecture and Design", John-Wiley & Sons, Reading, Mass., 1979.

[11] L.B.Jackson, S.F.Kaiser and H.S.McDonald, "An Approach to the Implementation of Digital Filters," IEEE Trans. Audio

Electro., AU-16, Sept. 1968.

[12] W.J.Stenzel, W.J.Kubitz and G.H.Garcia, "A Compact High-Speed Parallel Multiplication Scheme," IEEE Trans. on Comput., vol.C-26, no.10, pp.948-957, Oct. 1977.

[13] A.Karatsuba and Y.Ofman, "Multiplication of Multidigit Numbers with Computers", Dokl. Akad. Nauk. SSSR, no.145, Feb. 1962.

[14] C.S.Wallace, "A Suggestion for a Fast Multiplier", IEEE Trans. on Electro. Comput., vol EC-13, no.1, pp.14-17, Feb. 1964.

[15] A.V.Aho, J.E.Hopcroft and J.D.Ullman, "Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, Mass., 1974.

[16] D.E.Muller and F.P.Preparata, "Bounds to Complexities of Networks for Sorting and Switching", JACM, vol.22, no.2, pp.195-201, Apr. 1975.

[17] C.D.Thompson and H.T.Kung, "Sorting on a Mesh-Connected Parallel Computer", CACM, vol.20, no.4, Apr.1977.

[18] D.Nassimi and S.Sahni, "Bitonic Sort on a Mesh-Connected Parallel Computer", IEEE Trans. Comput., vol.C-28, no.1, Jan. 1979.

[19] M.Maekawa, "Parallel Sort and Join for High Speed Database Machine Operations", AFIPS Conf. Proc., vol.50, June 1981.

[20] L.E.Winslow and Y.C.Chow, "Parallel Sorting Machines :Their Speed and Efficieny", AFIPS Conf. Proc., vol.50, June 1981.

[21] T.C.Chen, V.Y.Lum and C.Tung, "The Rebound Sorter:An Efficient Sort Engine for Large Files", Proc. 4th VLDB, pp.312-318, Sept. 1978.

[22] H.Yasuura and N.Takagi, "A High-Speed Sorting Circuit Using Parallel Enumeration Sort", Trans. IECE, vol.J65-D, no.2, pp.179-186, Feb.1982 (in Japanese).

[23] S.Todd, "Algorithm and Hardware for a Merge Sort Using Multiple Processors", IBM Journal of R. & D., vol.22, no.5, Sept. 1978.

[24] Y.Tanaka, Y.Nozawa and A.Masuyama, "Pipeline Searching and Sorting Modules as Components of a Data Flow Database Computer", Proc. IFIP80, pp.427-432, Oct. 1980.

[25] H.Yasuura, "Hardware Algorithms for VLSI", Proc. Joint Conf. of 4 Institutes Related on Electric Engineering, 34-4, Oct. 1981 (in Japanese).

[26] J.E.Savage, "The Complexity of Computing", Wiley-Interscience, Reading, Mass., 1976.

[27] S.H.Unger, "Tree Realizations of Iterative Circuits", IEEE Trans. Comput., vol.c-26, no.4, pp.365-383, Apr. 1977.

[28] H.Yasuura, Y.Ooi and S.Yajima, "On Macroscopic Depth Reduction for Combinational Logic Circuits", IECE Technical Rep. EC81-1, Apr. 1981 (in Japanese).

[29] H.Yasuura, "Width and Depth of Combinational Logic Circuits", Information Processing Letters, vol.13, no.4, 5, End, pp.191-194, 1981.

[30] C.E.Leiserson, "Area-Efficiency Graph Layout (for VLSI)",

Proc. 21st FOCS, Oct. 1980.

[31] L.G.Valiant, "Universality Considerations in VLSI Circuits", IEEE Trans. on Comput., vol.C-30, no.2, pp.153-157, Feb.1981.

[32] H.Yasuura and S.Yajima, "On Area of Circuits on VLSI" (to appear).

[33] T.Sakai, Y.Tsuchida, H.Yasuura, Y.Ooi, Y.Ono, H.Kano, S.Kimura and S.Yajima, "An Interactive Simulation System for Structured Logic Design -- ISS", Proc. 19th DA Conf., June 1982.