Memoirs Kyushu Inst. Tech.
(Eng.) No. 14, 1984, pp. 53-63

HARDWARE SIMULATION LANGUAGE HSL/I

by

Akikazu TAMAKI*, Hajime MIZUMACHI†
and Kiyoshi KATO**

(Received November 30, 1983)

SYNPOSIS

The authors describe of the CHDL called HSL/1 (Hardware Simulation Language/1) which is suitable and adaptable to over the logic design level. HSL/1 consists of two parts. The former is the system part by which a hardware system is described. The later is the command part which is the command for the simulation. This paper contains two examples. One is the description and simulation of the asynchronous circuit. It is shown that the circuit can correctly operate by having the proper delays. The other is the description of the simple model computer. The computer is designed by means of the horizontal microprogram. HSL/1 is also a good tool for the education of hardware systems in the logic design level.

1. INTRODUCTION

CHDL has developped in the aid of designing of large computer systems and LSI. In the organizing a computer system, there are several levels as followings [5],

1. PMS level
2. Program level
3. Logic design level
4. Circuit level

and each level may be divided into sublevels. Most of CHDLs are convenient and adaptable to few sublevels. The development of CHDLs requires many labors and long time. Therefore, the computer system, which is designed by using those CHDLs, is produced in large quantities.

We make the digital system which is used for the tool of the research and experiment in laboratories. It is assembled by the standard IC chips and printed circuit boads. The authors intend to develop the CHDL which is convenient and usuful to design of digital systems by assembling the standard IC chips in laboratories. The CHDL is required to be convenient to over the logic design level, because the logic design of laboratories is almost included in it. The logic design level is divided into two sublevels i.e. Register transfer sublevel and Switching circuit sublevel. It requires the good descriptivity and the module structure such as assembly of IC chips. The digital system which is described by the CHDL must be simulated so as to find logical errors, before it is practically assembled on the printed circuit boads. The register transfer level requires a synchronous simulation, and the switching circuit level requires an asynchronous simulation with propagation delays.

---

* Assistant, Dept. of Computer Sci.
** Professer
† Nippon Telegraph and Telephone Public Corpration

The authors has constructed the CHDL and its simulator which satisfy above needs and called HSL/1 (Hardware Simulation Language 1). A part of KARL [1] is converted with slightly modification for the hardware description part of HSL/1.

2. OVERVIEW OF HSL/1

HSL/1 is designed for the purpose of making, learning and simulation for correctness of digital systems which are small and manufuctured in the laboratory. Those systems are not integrated onto LSI chips, but made by assembling the standard IC chips, and produced in small quantity.

HSL/1 consists of the system part and the command part. The former represents the structure of a hardware system, i.e. the type of elements and the interconnection of them. It is exactly a computer description language, by which we can describe a digital system in the manner of building blocks such as we practically assemble the modules (IC chips, printed circuit boads and so on). The command part is a simulation command, and represents the circumstance of the hardware system. It indicates the initial state, the timing of input data, the period of simulation, the result of simulation to be displayed and so on.

Figure 1 shows the general flow of HSL/1. The hardware system, which is described by the system part, is simulated in the compiler driven method. To make compiling easy, the authors have constructed the virtual machine which is suitable to simulate a hardware system. The simulator of the virtual machine is programmed in PL/1. The compiler produces the intermediate module from a source program (a hardware system described in the system part). The unit pakage is a set of modules which is frequently used such as a subroutine package in the programming lenguage. We can regard the unit package as a set of IC chips. By connecting the intermediate modules and those of the unit package, the linkage editer produces the object module which is simulated by the virtual machine. In obedience of the command part, the virtual machine simulates and reports the results of simulation.

We can repeat the simulation of the same object module in the different condition, by rewriting the command part. That is corresponding to checking up the hardware system in various conditions, in the real hardware system.

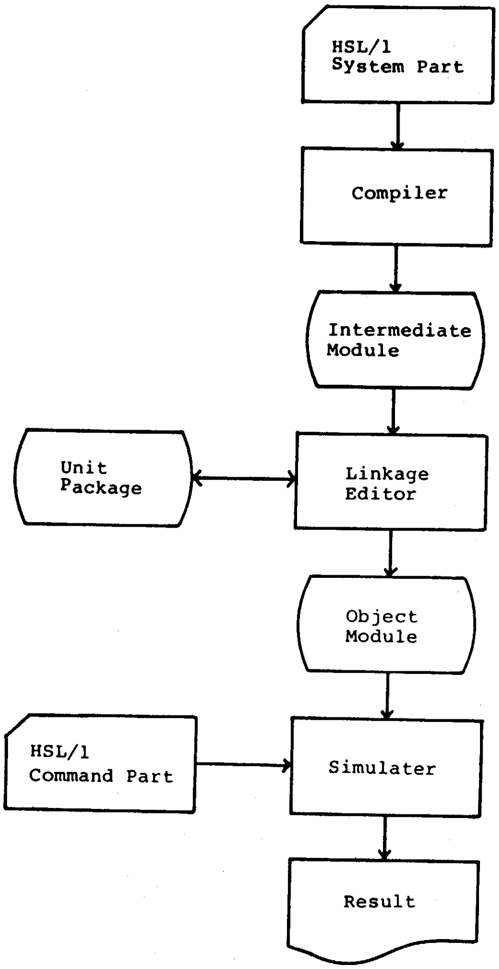


Fig. 1 The general flow of HSL/1.

3. SYSTEM PART AND COMMAND PART

SYSTEM PART

A hardware system described by the system part begins at UNIT and ends at TINU, and then it is compiled to an intermediate module. The system part consists of the declaration part and the statement part. The formar declares attribute of the elements used in the hardware system. Some attribute are followings.

REGISTER and TERMINAL attribute are able to have a high dimensional structure (inplemented up to five dimensionality). REGISTER has a memorial function but TERMINAL has not. SWITCH attribute means the data input and LIGHT attribute displays the output data. CLOCK attribute indicates the clock pulse with a period and a width or with only a period. We can declare both a single phase pulse and a multi phase pulse. For designing a computer system such as a micro computer system, MEMORY and a bus attribute are declared. MEMORY is declared the capacity of words and the length of word, and essentially equals to a two dimensional register array. A bus attribute declares bus line systems such as that of a computer system. There exist OUTBUS that is a one directional data path and BIBUS that is a bidirectional data path. OUTBUS is connected by means of tristate, logical and, logical or, and then BIBUS is tristate. CONSTANT attribute declares the constants which are logical one or logical zero. We can describe Roms in which programs and data are programmed in computer systems. CONNECT attribute indicates the connection of elements to those of the other modules.

The statement part describes the interconnection of elements, the functions of elements, the condition of data transfer and the propagation delays. The sort of function has the shift, the reflection and so on. In the data transfer condition, there are AT, ON, IF and CASE statement. AT statement describes that the data is transferred at the edge of the pulse, and we can describe a edge triggered flip-flop. ON statement is used for description of a master slave flip-flop. IF statement describes that the data is transferred if the control signal is logical one, and a D latch is described. CASE statement is the same as that of a programming language. We can also describe the propagation delay of rising up and rising down individually.

COMMAND PART

The command part gives the information to the simulator which simulates the object module made from intermediate modules and those of the unit package. The commands are followings.

INITIALIZE command is used for setting the initial state of the hardware system to be simulated. DELAYSET command is used for changing the magnitude of the propagation delay which has already described in the system part. CLOCKSET command is used for changing the period and width of the clock pulse which has already described in the system part. SWITCHIN command indicates the value of data and the time when the data is input. The data is not input to only SWITCH attribute, but also REGSTER and TERMINAL attribute. PRINTOUT command indicates the time, the condition and the element to be reported of the result. RUN command indicates the period of simulation by means of the time or the condition.

4. SIMULATOR

The hardware system is simulated by the virtual machine illustrated in Figure 2. The machine consists of a superviser and five processers, i.e. Executer, Clock Generator, I/O

Controller, Delay Controller and Initiator. They are programmed by PL/1. The elements of the hardware system, which is described in the system part, are represented in Data Memory. One word of Data Memory contains five items i.e. the value of element, the magnitude of up delay, that of down delay, two counters for up delay and down delay. The element can take six values, i.e. logical one, logical zero, up (transition of logical zero to one), down (transition of logical one to zero), tristate (high impedance) and warning (which means that the value is not determined). The interconnection and operation of elements, which are described by the statement part in the system part, are stored in Program Memory in the form of instruction code for Executer. The statement part is compiled and linked to the form of object module for Executer which contains the information of the system part. The information of CLOCK attribute is stored in Clock Table. The information of SWITCHIN and PRINTOUT command is stored in I/O Table. We can change the period and width of Clock Table by CLOCKSET command, and the magnitude of up delay and down delay in Data Memory by DELAYSET command. Initiator initializes the state of elements of the hardware system by setting the value of elements in Data Memory, in accordance of INITIALIZE command. Executer changes the value of elements in Data Memory by executing the program in Program Memory. Clock Generator generates clocks according to Clock Table and changes the value of clock terminal in Data Memory. I/O Controller inputs the data to the elements in Data Memory and prints out the value of elements, in accordance with I/O Table.

It can simulate in the synchronous and asynchronous methods. In the former, the magnitude of the delays described in the system part are neglected. In the later, that is regarded and the minimum sampling (simulation) time is 1 nano second for convenience' sake. Executer does not change the values directly, only sets the counter for up delay and down delay at changing zero to one and one to zero, respectively. The counters are decreased one by one each sampling time, so that Delay Controller sets the value of one when the counter for up delay is zero, or that it sets the value of zero when the counter for down delay is zero.

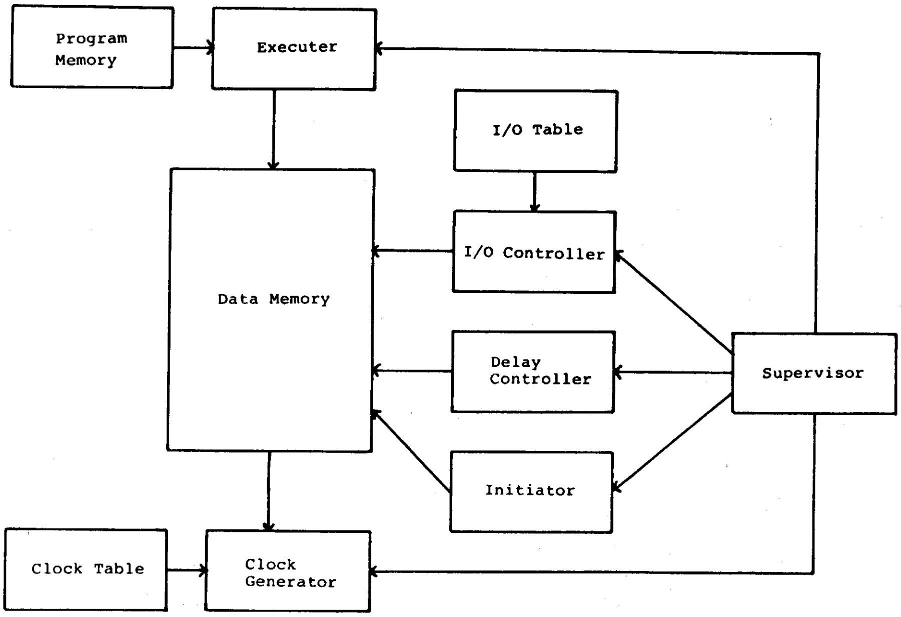


Fig. 2 The block diagram of the virtual machine.

Superviser controlls those processers so as to simulate the hardware system.

5. EXAMPLES

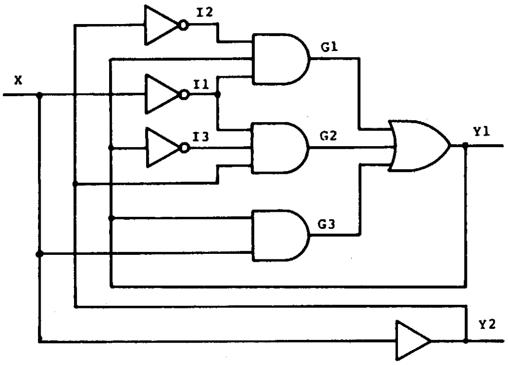
EXAMPLE OF ASYNCHRONOUS CIRCUTT

Consider the asynchronous circuit with the flow table, assignment and realization shown in Figure 3.(1) The assignment causes a race in the transition from the state 2 to the state 3 and the transition from the state 4 to the state 1. The final state reached is depend on the order of changes. We can described the realization by HSL/1 and simulate it by varying the magnitude of delays of gates. The description of the realization by the system part is also shown in Figure 3, named SAMPLE. I1, I2, I3, G1, G2, G3, Y1 and Y2 have TERMINAL attribute. X has CLOCK attribute whose period is 40 nano seconds and the width 20 nano seconds. Figure 4 shows the command part and the result of simulation in

State	Next state		Assignment	
	X=0	X=1	Y1	Y2
1	①	2	0	0
2	3	②	0	1
3	③	4	1	0
4	1	④	1	1

The circled states are stable, and the others unstable.

(a)



(b)

```
1        /***************************************************************
         *                                                              *
         *        EXAMPLE  OF  RACE  IN ASYNCHRONOUS  CIRCIUT           *
         *                                                              *
         ***************************************************************/
                  UNIT SAMPLE(MAIN).
2                 TERMINAL I1,I2,I3,G1,G2,G3,Y1,Y2.
3                 CLOCK X=20 BY 20 NS.
4                 I1 := NOT X.
5                 I2 := NOT Y2.
6                 I3 := NOT Y1.
7                 G1 := I1 AND I2 AND Y1.
8                 G2 := I1 AND I3 AND Y2.
9                 G3 := X AND Y1.
10                Y1 := G1 OR G2 OR G3.
11                Y2 := X.
12                TINU SAMPLE.
```

(c)

Fig. 3 An asynchronous circuit.

(a) flow table and assignment

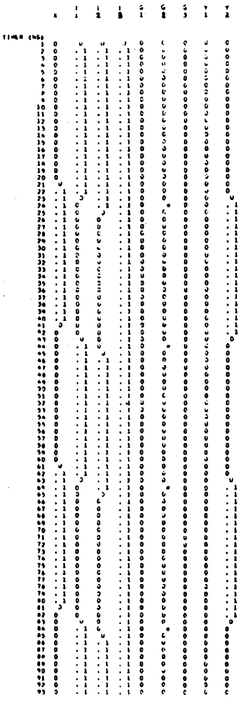
(b) realization

(c) description by system part of HSL/1

NOTE (1) It appears in p. 210 of [4].

```
INITIALIZE I1=1;
INITIALIZE I2=1;
INITIALIZE I3=1;
INITIALIZE G1=0;
INITIALIZE G2=0;
INITIALIZE G3=0;
INITIALIZE Y1=0;
INITIALIZE Y2=0;
PRINTOUT BY CYCLE(1) X;
PRINTOUT BY CYCLE(1) I1;
PRINTOUT BY CYCLE(1) I2;
PRINTOUT BY CYCLE(1) I3;
PRINTOUT BY CYCLE(1) G1;
PRINTOUT BY CYCLE(1) G2;
PRINTOUT BY CYCLE(1) G3;
PRINTOUT BY CYCLE(1) Y1;
PRINTOUT BY CYCLE(1) Y2;
RUN(200);
```

(a)

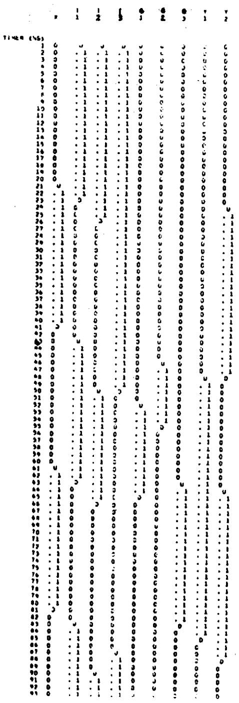


(b)

Fig. 4 **The simulation (all of the delays are equivalent).**
(a) command part of HSL/1
(b) timing chart

```
INITIALIZE Y2=0;
INITIALIZE Y1=0;
INITIALIZE G3=0;
INITIALIZE G2=0;
INITIALIZE G1=0;
INITIALIZE I3=1;
INITIALIZE I2=1;
INITIALIZE I1=1;
DELAYSET G2=(1,5);
DELAYSET Y2=(1,5);
PRINTOUT BY CYCLE(1) X;
PRINTOUT BY CYCLE(1) I1;
PRINTOUT BY CYCLE(1) I2;
PRINTOUT BY CYCLE(1) I3;
PRINTOUT BY CYCLE(1) G1;
PRINTOUT BY CYCLE(1) G2;
PRINTOUT BY CYCLE(1) G3;
PRINTOUT BY CYCLE(1) Y1;
PRINTOUT BY CYCLE(1) Y2;
RUN(200);
```

(a)



(b)

Fig. 5 **The simulation (the delays of G2 and Y2 are larger).**
(a) command part of HSL/1
(b) timing chart

which all of the delays are equivalent. When I1 is down and Y2 is up, the logical and of them is not determined. Therefore, G2 sometimes becomes warning, and then Y1 and Y2 repeat the state 1 and the state 2. Figure 5 shows the next simulation in which the down delays of G2 and Y2 are larger. The magnitude of delays of G2 and Y2 are rewriten by DELAYSET command. No warning appears in the timing chart, and the realization can correctly operate by having the proper delays.

EXAMPLE OF COMPUTER DESIGN

The authors have described the model computer called COMP-X which is used for the examination of information processing engnieer in Japan. Refer the specification to the appendix. The simple diagram shown in Figure 6 indicates the data path and the principal registers except the elements for the timing and state control. The description of the model computer is shown in Figure 7, named COMP-X. In this architecture, one execution of a instruction requires 8 states (from the state 0 to the state 7), and the preceding 4 states are a fetch cycle, and the secceeding 4 states an excute cycle. STATE (0: 2) is the 3 bits counter used for the 8 states. The data paths are controlled by the horizontal microprogram stored in Rom, named CODE, which is declared as CONSTANT attribute. It consists of 16 words corresponding to the instructions, and one word is 18 bits corresponding to the information of controlling the data paths. Table 1 shows the meanings of the each bit. In the state 3, an instruction is transferred from DBR to IR, and in the state 4, the instruction is decoded by the contents of CODE, and then the information is transferred to the control flip-flops (CR, CW, CH, CALU (2: 0), C11, C10, C9, C8, C7, C6, C5, C4, C3, C2,

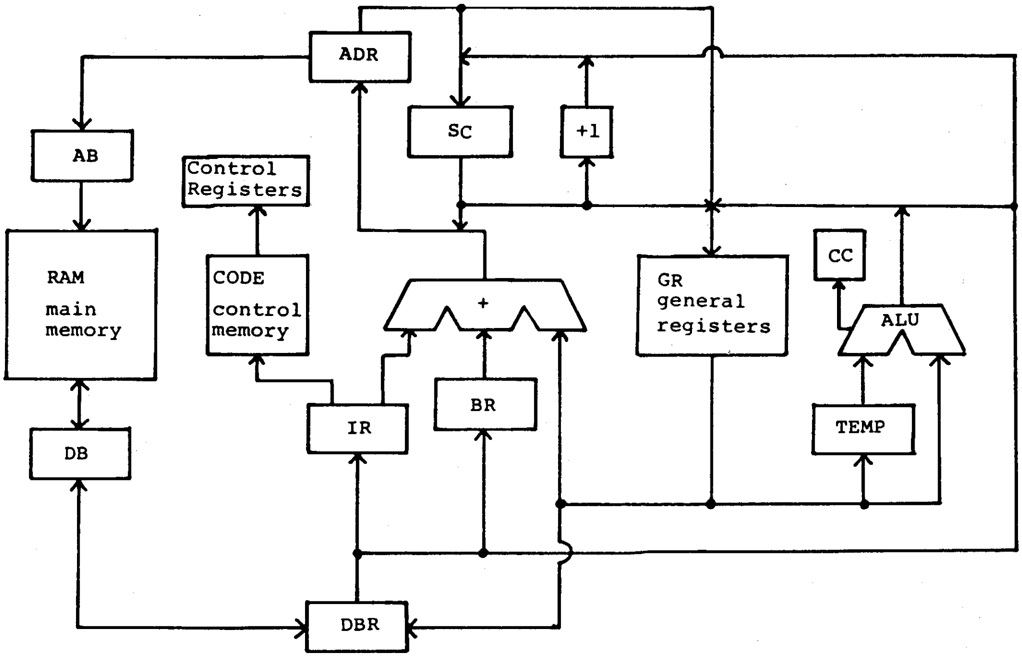


Fig. 6 The block diagram of COMP-X.

Table 1. The bitwise function of micro code

<table>
<tr><th>bit</th><th>function</th></tr>
<tr><td>0</td><td>DBR →GR</td></tr>
<tr><td>1</td><td>ADR →GR</td></tr>
<tr><td>2</td><td>BR: SC →GR</td></tr>
<tr><td>3</td><td>ALU →GR</td></tr>
<tr><td>4</td><td>GR →TEMP</td></tr>
<tr><td>5</td><td>ADR →SC (Jump on condition)</td></tr>
<tr><td>6</td><td>ADR →SC (Jump if GR is not zero)</td></tr>
<tr><td>7</td><td>ADR →SC (Halt and jump)</td></tr>
<tr><td>8</td><td>DBR →BR: SC</td></tr>
<tr><td>9</td><td>ADR →AB</td></tr>
<tr><td>10</td><td>DB →DBR</td></tr>
<tr><td>11</td><td>GR →DBR, GR→DB</td></tr>
<tr><td>12–14</td><td>ALU control
<table>
<tr><th colspan="3">bit</th><th rowspan="2">function</th></tr>
<tr><th>14</th><th>13</th><th>12</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>exor</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>and</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>subtract</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>add</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>shift</td></tr>
</table>
</td></tr>
<tr><td>15</td><td>halt</td></tr>
<tr><td>16</td><td>memory read</td></tr>
<tr><td>17</td><td>memory write</td></tr>
</table>

(note) Bit 0 is the rightmost bit.

```
NO.      SOURCE

         /********************************************************************/
         /*    COMP-X                                     CODE BY M0626      */
         /********************************************************************/

          UNIT COMP-X (MAIN).

              REGISTER IR(15:0).
              REGISTER BR(15:0).
              REGISTER GR(3:0;15:0).
              REGISTER SC(15:0).
              REGISTER CC.

              REGISTER TEMP(15:0).
              REGISTER ADR(15:0).
              REGISTER DBR(15:0).

              /* INTERNAL BUS BUFFER */
              REGISTER RD.
              REGISTER WR.
              REGISTER AB(15:0).
              REGISTER DB(15:0).

              /* STATUS REGISTER      */
              REGISTER HOLD.
              REGISTER STATE(2:0).

              /* MICRO CODE           */
              CONSTANT CODE(15:0;17:0)=001000000010000000B:       /* HJ   */
                                       000000000001000000B:       /* JNZ  */
                                       000000000000100000B:       /* JC   */
                                       100000011100000100B:       /* JSR  */
                                       000101000000011000B:       /* SFT  */
                                       0:                         /* -    */
                                       0:                         /* -    */
                                       0:                         /* -    */
                                       000000000000000010B:       /* LAI  */
                                       0:                         /* -    */
                                       100100011000011000B:       /* ADD  */
                                       100011011000011000B:       /* SUB  */
                                       100000011000000001B:       /* LD   */
                                       010001010000000000B:       /* ST   */
                                       100010011000011000B:       /* AND  */
                                       100001011000011000B.       /* EOR  */

              REGISTER CR.
              REGISTER CW.
              REGISTER CH.
              REGISTER CALU(2:0).
              REGISTER C11.
              REGISTER C10.
              REGISTER C9.
              REGISTER C8.
              REGISTER C7.
              REGISTER C6.
              REGISTER C5.
              REGISTER C4.
              REGISTER C3.
              REGISTER C2.
              REGISTER C1.
              REGISTER C0.

              /* CONTROL              */
              CLOCK CLK=1 BY 1 NS.
              TERMINAL NCLK:=NOT CLK.
              REGISTER JF.

              IF NOT HOLD THEN
                   CASE STATE(2:0) OF
                      ( 0 : ADR(15:0):=BR(15:8):SC(7:0).
                      )
                      ( 1 : RD:=@HIGH.
                            AB(15:0):=ADR(15:0).
                            AT NCLK DO SC(15:0):=INC SC(15:0). TA.
                      )
                      ( 2 : DBR(15:0):=DB(15:0).
                      )
                      ( 3 : RD:=@LOW.
                            IR(15:0):=DBR(15:0).
                      )
                      ( 4 : CR:CW:CH:CALU(2:0):C11:C10:C9:C8:C7:C6:C5:C4:
                            C3:C2:C1:C0:=CODE(IR(15:12);17:0).
                            ADR(15:8):=BR(15:8).
                            IF IR(9:8)=@LOW:@LOW
                                   THEN ADR(7:0):=IR(7:0).
                                   ELSE ADR(7:0):=IR(7:0)+GR(IR(9:8);7:0).
                            FI.
                      )
```

Fig. 7 The description of COMP-X by system part of HSL/1.

```
       ( 5 ; IF CR   THEN RD:=@HIGH. FI.
             IF CW   THEN WR:=@HIGH. FI.
             IF C11 THEN DB(15:0):=GR(IR(11:10);15:0). FI.
             IF C9   THEN AB(15:0):=ADR(15:0). FI.
             JF:=@LOW.
       )
       ( 6 ; IF C10 THEN DBR(15:0):=DB(15:0). FI.
             IF C6   THEN IF GR(IR(11:10);15:0)#@LOW(15:0)
                                THEN JF:=@HIGH.
                          FI.
             FI.
             IF C5   THEN JF:=(IR(11) AND (NOT CC))
                               OR
                               (IR(10) AND CC).
             FI.
             IF C4   THEN TEMP(15:0):=GR(IR(11:10);15:0). FI.
             IF C2   THEN GR(IR(11:10);15:0):=BR(15:8):SC(7:0). FI.
       )
       ( 7 ; IF CR   THEN RD:=@LOW. FI.
             IF CW   THEN WR:=@LOW. FI.
             IF CH   THEN HOLD:=@HIGH. FI.
             IF C8   THEN BR(15:8):=DBR(15:8).
                          SC(7:0):=DBR(7:0).
             FI.
             IF C7 OR JF THEN SC(7:0):=ADR(7:0). FI.
             IF C3   THEN GR(IR(11:10);15:0):=ALU(15:0).
                          CC:=ALUCC.
             FI.
             IF C1   THEN GR(IR(11:10);15:0):=@LOW(15:8):ADR(7:0).
             FI.
             IF C0   THEN GR(IR(11:10);15:0):=DBR(15:0). FI.
       )
     ESAC.
     AT CLK DO STATE(0:2):=INC STATE(0:2). TA.
FI.

/* ALU                      */
REGISTER ALU(15:0).
REGISTER ALUCC.

CASE CALU(2:0) OF
  ( 1 ; ALU(15:0):=TEMP(15:0) EXOR DBR(15:0).
  )
  ( 2 ; ALU(15:0):=TEMP(15:0) AND DBR(15:0).
  )
  ( 3 ; ALU(15:0):=TEMP(15:0) - DBR(15:0).
        ALUCC:=NOT ALU(15).
  )
  ( 4 ; ALU(15:0):=TEMP(15:0) + DBR(15:0).
        ALUCC:=NOT ALU(15).
  )
  ( 5 ; IF IR(8) THEN ALU(15:0):=SHR TEMP(15:0).
                 ELSE ALU(15:0):=DSHL TEMP(15:0).
        FI.
  )
ESAC.

/* RESET                    */
SWITCH RESET.

IF RESET THEN SC(15:0):=@LOW(15:0).
              BR(15:0):=@LOW(15:0).
              HOLD:=@LOW.
              STATE(2:0):=@LOW(2:0).
              RD:=@LOW.
              WR:=@LOW.
FI.

/* START                    */
SWITCH START.

IF START THEN HOLD:=@LOW.
              STATE(2:0):=@LOW(2:0).
FI.

/* MEMORY                   */
CONSTANT RAM(5:0:15:0)=0C404H:
                       085FFH:
                       01401H:
                       00000H:
                       00002H:
                       0.

IF RD THEN DB(15:0):=RAM(AB(7:0);15:0). FI.
IF WR THEN RAM(AB(7:0);15:0):=DB(15:0). FI.

TINU COMP-X.
```

Fig. 7 The description of COMP-X by system part of HSL/1. (continue)

C1, C0).

In the description, the shift instruction is allowed only one bit operation. The explanation of the other operation is omitted.

6. CONCLUSIONS AND ACKNOLEDGEMENT

The authors have explained the outline of HSL/1 system which is suitable to over the logic design level. A hardware system can be built up from modules, as the system part of HSL/1 has a module structure. We can describe a hardware system by HSL/1, in the imagination of assembling standard IC chips practically. As showing the example, HSL/1 simulates the asynchronous circuit with a race and hazard. The circuit can correctly operate or cannot, by changing the magnitude of the propagation delays. Then, HSL/1 is a good tool for learning the operation of an asynchronous circuit. COMP-X is described in the register transfer level, and the description is very simple. HSL/1 is useful for learnig a computer design. From above discussion, HSL/1 is also suitable to the education of hardware system in the logic design level.

The authors greatefully appreciate the active roles that Shin-ichi Kiyohara and Hiroyuki Fukuda played in the design and implementation.

REFERENCES

1) Hartenstein, R. W., Fundamentals of Structured Hardware Desing, North-Holland, 1977.
2) Breuer, M. A., Digital System Design Automation: Language, simulation & Database, Computer Science Press, 1975.
3) Breuer, M. A., Design Automation of Digital Systems, Prentice-Hall.
4) Friedman, A. D. and Menon, P. R., Theory & Design of Switching Circuits, Computer Science Press, 1975.
5) Mead, C. and Conway, L., Introduction to VLSI Systems, Addison-wesley, 1980.
Followings are described in Japanese.
6) Tamaki, A. et al., "Hardware Simulation System HSL/1 (I): Declaration of Hardware System", Bulletin of Kyushu Institute of Technology, No. 42, 1981.
7) Tamaki, A. et al., "Hardware Simulation System HSL/1 (II): Simulation by Virtual Machine", Bulletin of Kyushu Institute of Technology, No. 42, 1981.
8) Mizumachi, H., "Study of Computer Design Language", Master Thesis of Kyushu Institute of Technology, 1982.

APPENDIX

The Specification of COMP-X

Main memory:	2^{16} words, 16 bits/word.
Numerical data:	Fixed point with 16 bits, and a negative number is the form of 2's complement.

Registers

SC:	Sequence counter (program counter).
BR:	Base register with 16 bits, the lower 8 bits are always zero.
General registers:	16 bits, called GR0, GR1, GR2, GR3, which are also used for index registers except GR0. GR means general registers.
CC:	Condition code register, after the completion of Add of Subtract operation, set 1 if the result is negative, otherwise 0.

Instruction format

0 3	4 5	6 7	8 15
OP	GR	XR	AD

OP: Operation code.

GR: The number of GR or the condition of the Jump on Condition instruction.

XR: The number of index register. If XR=0, no address modification.

AD: Indicates the lower 8 bits of the operand address. The effective address (A_{ef}) is modified as following.

No index modification,

A_{ef} = upper 8 bits of BR + AD.

Index modification,

A_{ef} = (AD + lower 8 bits of GR indexed by XR field) mod 256 + upper 8 bits of BR.

Instructions

Binary	Instruction	Function
0000	Halt and jump	$A_{ef} \rightarrow$ SC, and halt. If the start button will be pushed, reruns at the address indicated by SC.
0001	Jump if GR is not zero	Jump to A_{ef}, if GR is not zero.
0010	Jump on condition	[GR]=00: No operation [GR]=01: If CC=1, then jump to A_{ef} [GR]=10: If CC=0, then jump to A_{ef} [GR]=11: Jump to A_{ef} (note) [GR] means the content of GR field.
0011	Jump to subroutine	SC+1 $\rightarrow$ GR, and [A_{ef}] $\rightarrow$ SC, BR, and then the lower 8 bits of BR are reset.
0100	Shift	The content of GR is shifted to the right (XR field=00) or left (XR field=01) for the number of [A_{ef}].
1000	Load address immediate	Lower 8 bits of $A_{ef} \rightarrow$ lower 8 bits of GR, and the upper 8 bits of the GR is reset.
1010	Add	GR+[A_{ef}] $\rightarrow$ GR, and CC is set if the result<0, reset otherwise.
1011	Subtract	GR−[A_{ef}] $\rightarrow$ GR, and CC is set if the result<0, reset othwewise.
1100	Load	[A_{ef}] $\rightarrow$ GR
1101	Store	GR $\rightarrow$ [A_{ef}]
1110	And	GR and [A_{ef}] $\rightarrow$ GR
1111	Exclusive or	GR exor [A_{ef}] $\rightarrow$ GR

(note) [A_{ef}] means the content of memory whose address is indicated by [A_{ef}]. GR used above are specified by GR field in the instruction.