Information Processing in Japan Vol. 11, 1971

Description of Multi-Units in Design Language of Computer

HIROSHI HAGIWARA* AND YOSHISUKE KUROZUMI**

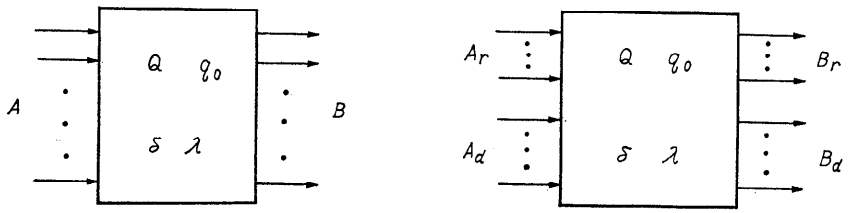
1. *Introduction*

The object of design language of computer is a description of computer system. It is necessary to describe multiprocess or on-line system. For this purpose, the communication between units must be described. We define a unit, explain the communication and designate several examples of programs.

2. *Definition of Unit*

A sequential circuit of multi-outputs Moore models is $(Q, q_0, A, B, \delta, \lambda)$, where Q is a finite set of states, q_0 in Q is the initial state, A is an input signal (vector), B is an output signal (vector), δ is a trandition function and λ is an output function.

We set limits to the above model and define a unit. An element of vector A is a binary vector of a bits. We divide this vector into two parts. One is the recieving part (r bits) and the other is the data part (d bits). If the recieving part is zero, an element of input signal A is A_0, no matter how the data part is. If the recieving part is not zero, an element of A is A_i. The same limitation is set to output signal B (Fig. 1).



(a) Sequential machine (b) Unit

Fig. 1.

For this limitation, we can distinguish between a terminal for timing clock and terminals for scnding and recieving data. Fig. 2 is a full adder unit. I is a recieving terminal, O is a sending one, and the other terminals arc data terminals.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 12, No. 6 (1971), pp. 320-325.

* Faculty of Engineering, Kyoto University.
** Faculty of Science, Kyoto Sangyo University.

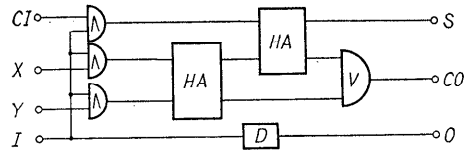


Fig. 2. Full-adder unit.

3. *Method of Communication Between Units*

3.1 *Simplex Communication*

When a direction of control signal is one way from one unit to the other unit, this communication is called simplex communication. Data are able to be transmitted in either direction. Fig. 3 shows communication from unit A to unit B.

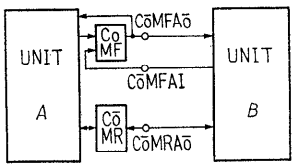


Fig. 3. Simplex communication.

A communication filp-flop (COMF) and a communication register COMR are used. These connections are as follows. The sending terminal of A is connected to the set terminal of COMF and the sending terminal of B is connected to the reset terminal of COMF. The output of COMF is connected to the recieving terminal of A and B. COMR is connected to the data terminals of A and B.

In this form there are two methods.

(1) Advance type

When A communicates with B, A confirms that COMF is off, inputs from COMR (or outputs to R) and sets COMF. After that, A can process its job, if A does not disturb COMR. On the other hand, as soon as B knows that COMF is on, B outputs to COMR (or inputs from COMR), resets COMF, and processes its job.

This communication is used in the control of the direct coupled I/O device (ex. typewriter).

(2) Synchronous type

It is assumed that data are transfered from A to B. When A communicates with B, A sets data into COMR, sets COMF and waits until COMF is reset. If COMF is reset, A can process its job. The action of B is the same as (1).

The difference between (1) and (2) is as follows. In (1), A can process its job as soon as A requests communication. But, in (2), A must wait until a request of A is recieved by B.

3.2 *Half-duplex Communication*

In half-duplex communication, a control is transmitted in either direction, but only in one direction at the same time. COMF and COMR are necessary. Two output terminals are connected to a set terminal and a reset terminal of COMF. The output of COMF is connected to the recieving terminal of A. B is connected in the same way as A. In this form, the synchronous type (2) is used. A problem occurs when A and B request communication at once. Both A and B set COMF and wait until COMF is reset. So, A and B wait forever. This is deadlock.

It is difficult to use the advance type in this communication. After A sets COMF, A processes its job, and when A requests the second communication, A can not distinguish that COMF (on) indicates either the first communication from A or the new communication from B.

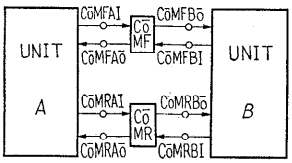


Fig. 4. Half-duplex communication.

3.3 *Full-duplex Communication*

In full-duplex communication system, data are transmitted in both directions at the same time. Fig. 5 shows the system has two simplex communications in both directions. The advance type control (1) can be used in this system. Even if communication requests happen at the same time, a unit which becomes interruptable previously recieves the requests. Deadlock does not occur.

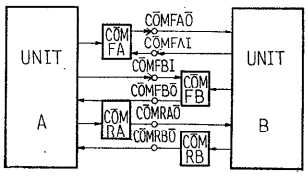


Fig. 5. Full-duplex communication.

If two units communicate in the synchronous type, deadlock occurs. If one unit uses the advance type and another uses the synchronous type, the unit which uses the advance type recieves communication previously.

4. *Description of Unit in T-language*

T-language which is proposed lately [1] is suitable to describe computers. The following programs describe simplex communication and half-duplex communication.

```
SIMPLEX: procedure /* SIMPEX COMMUNICATION */
         external START;
  UNITA: procedure /* UNITA-A PROCEDURE DECLARATION */
         register COMF, COMR(1 : 8), DATA(1 : 8);
         ... /* OTHER MODULE DECLARATION OF UNIT-A */
         external COMFAI, COMFAO, COMRAO(1 : 8), STAT;
         function COMFAO-COMF, reset COMF=COMFAI,
                  COMRAO=COMR;
 UNITAO: if STAT then wait;
         ... /* EXECUTION STEPS */
         if COMF then wait;
         COMR=DATA; COMF=1; /* TRAP STEP */
         ... /* EXECUTION STEPS */
         end /* UNIT-A END */
  UNITB: procedure /* UNIT-B PROCEDURE DECLARATION */
         register DATA(1 : 8);
         ... /* OTHER MODULE DECLARATION OF UNIT-B */
         external COMFAI, COMFAO, COMRAO(1 : 8);
 UNITBO: if COMFAO then wait; /* INTERRUPTION STEP */
         DATA=COMRAO; COMFAI=1;
         ... /* EXECUTION STEPS */ end /* UNIT-B END */
         function STAT=START; end /* PROGRAM END */
```

Fig. 6. Program of simplex communication.

```
HALF-DUPLEX: procedure /* HALF-DUPLEX COMMUNICATION */
             register COMF, COMR(1 : 8); external START;
      UNITA: procedure /* UNIT-A PROCEDURE DECLARATION */
             register DATA(1 : 8);
             ... /* OTHER MODULE DECLARATION OF UNIT-A */
             external STAT, COMFAI, COMFAO,
             COMRAI(1 : 8), COMRAO(1 : 8);
     UNITAO: if ~STAT then wait;
             ... /* EXECUTION STEPS */
             if ~COMFAO then wait; /* INTERRUPTION STEP */
             DATA=COMRAO; COMFAI=0:
             ... /* EXECUTION STEPS */
             COMRAI=DATA; COMFAI=1: /* TRAP STEP */
             if COMFAO then wait;
             ... /* EXECUTION STEPS */ end
      UNITB: procedure /* UNIT-B PROCEDURE DECLARATION */
             register DATA(1 : 8); external COMFBI, COMFBO,
             COMRBI(1 : 8), COMRBO(1 : 8);
     UNITBO: if ~COMBO then wait; /* INTERRUPTION STEP */
             DATA=COMFBO; COMFBI=0; ... /* EXECUTION STEPS */
             COMRBI=DATA; COMFBI=1: /* TRAP STEP */
             if COMFBO then wait; ... /* EXECUTION STEPS */
             end
             function STAT=START, COMF=COMFAI, COMF=COMFBI,
               COMFAO=COMF, COMFBO=COMF, COMR=COMRAI,
               COMR=COMRBI, COMRAO=COMR, COMRBO=COMR;
             end
```

Fig. 7. Program of Half-duplex communication.

5. *Conclusion*

We defined a unit and classified communication between these units. Our object is to study how this communication is described and converted.

Reference

[1] Hagiwara, H. & Y. Kurozumi: Design language of computer, *J. Information Processing of Japan*, 92–102 (Feb. 1971).