Memoirs of the School of Engineering, Okayama University, Vol. 16-2, March 1982

Modifications of an Asynchronous Ring Arbiter

Takuji OKAMOTO[*]

(Received January 27, 1982)

Synopsis

Some modifications of an asynchronous ring arbiter are proposed. This arbiter is composed of a chain of cells. Each cell has one to one correspondence to a device. In the chain, there exists only one privilege to arbitrate conflicts of requests from many devices. A class of modifications is high speed arbiters, obtained by increasing the number of connecting wires between two adjacent cells. As the results, the time required for the privilege to pass through a cell decreases by about one-half compared with the original arbiter. Another class of modifications is arbiters with priority rules. They are obtained by adding a few hardware to the original arbiter. The priority order of request acknowledgements in all the cells is specified. Using above modifications, conflicts of requests in many digital systems may be feasibly arbitrated.

1. Introduction

There are many systems, in which more than two devices operated under independent clock pulses share a common resource such as a memory module, a common bus, etc. In such systems, conflicts on privilege acquisitions to possess a common resource may occur frequently.

---

* Department of Electronics

The asynchronous arbiter is a circuit to arbitrate such conflicts, so that more than two devices may never possess a common resource simultaneously and if there exist more than one requests, at least one of them will be acknowledged in a finite time.

Many asynchronous arbiters have been proposed since Plummer's proposal. They are divisible into the First-arrival-first-service type in which requests are acknowledged by order of arrival [1]-[5] and the Daisy chain type in which requests are acknowledged in order of device arrangements [5]-[12].

The ring arbiter [7] described in this paper belongs to the latter type. It has been applied to bus control modules in a mini-computer complex system, a μ programed computer system HOP, etc [13]-[15].

This arbiter is composed of a chain of cells having the same structure. Each of them is assigned to each device. There exists only one privilege for the exclusive common resource possession in this chain. It is going round all the cells. The request issued from any device is received as an input signal in the corresponding cell and acknowledged at the time when the privilege reaches the corresponding cell. Then the acknowledgement is sent to the corresponding device as an output signal and after the device releases the request, the privilege goes to the next cell. In succeeding descriptions, this arbiter is called as the basic one and the cell used in this arbiter is called as the basic one.

The basic arbiter allows each basic cell to share the common resource on an equitable basis. Besides, it has an excellent extendability in addition to its simple configuration. In above applications, such features are effectively utilized. However, this arbiter can not be applied to systems, in which the fast response time or the preferential service to the specified device is required.

This paper proposes modifications of an asynchronous ring arbiter, applicable to such systems. In Chapter 2, the operation of the basic arbiter and the action of the basic cell are described in detail. In Chapter 3, at first, two high speed arbiters obtained by increasing the number of connecting wires between two adjacent cells are shown, and the next, two arbiters with a priority rule are shown.

2. Basic Arbiter

2.1 Structure of Basic Arbiter

The basic arbiter dealed with in this paper is composed of the chain of basic cells as shown in Fig.1. A basic cell C_i ($1 \leq i \leq n$, $n \geq 3$) is prepared for the device i. There is only one level signal P, as a privilege, going round all the chain along directions of arrows. It allows each device to possess the common resource exclusively. At the time when P reaches C_i, if the request is not issued, it is directly transmitted to the next cell C_{i+1}, and if the request is issued, it is transmitted to C_{i+1} after this request is satisfied. The transmission of P is performed in hand shake style (see Fig.2).

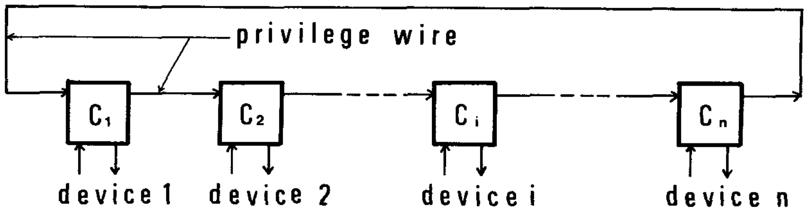

Fig.1 Structure of Ring Arbiter

This arbiter has features as follows.

(1) Each device can possess the common resource on an equitable basis.

(2) The arbiter has an excellent extendability.

(3) The structure of the basic cell is very simple.

(4) Few additional circuits are required to initialize this arbiter.

(5) This arbiter has a large noise immunity.

The block diagram of C_i is shown in Fig.2. C_i is connected with the device i by a request wire and an acknowledgement wire. Variables r_i and A_i are assigned to signals on the request wire and the acknowledgement wire, respectively. The value of r_i is held at logic level 1 when and only when the request is issued from the corresponding device i and A_i is held at 1 when and only when it is acknowledged.

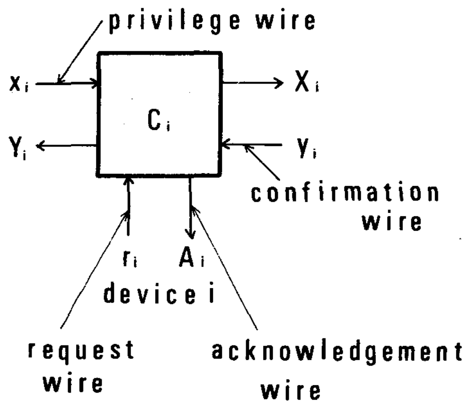

Fig.2 Block Diagram of a Cell

As mentioned above, adjacent two basic cells (C_i & C_{i-1} or C_i & C_{i+1}) are connected by a privilege wire and a confirmation wire.

Variables x_i and X_i are assigned to an input signal given from C_{i-1} through the privilege wire and an output signal given to C_{i+1} through the privilege wire. Variables y_i and Y_i are also assigned to an input signal given from C_{i+1} through the confirmation wire and an output signal given to C_{i-1} through the confirmation wire, respectively. x_i(X_i) is held at logic level 1 only in the interval when C_i is receiving (sending) P from C_{i-1}(to C_{i+1}) and y_i(Y_i) is held at logic level 1 only in the interval when C_i is receiving (sending) the confirmation signal to confirm the receipt of the privilege from C_{i+1} (to C_{i-1}).

This arbiter operates without any malfunction under assumptions that the device i satisfies the following conditions.

(a1) r_i can change from logic level 0 to 1 (hereafter referred to as $r_i(0\to1)$) when and only when A_i=0, and r_i can change from logic level 1 to 0 (hereafter referred to as $r_i(1\to0)$) when and only when A_i=1.

(a2) If the device i once possesses the common resource, its possession time is finite and longer than $T_1(>0)$ without fail. On the contrary, it takes more than $T_2(>0)$ for the device to issue the request again after the termination of a resource possession.

(a3) The device i can possess the common resource exclusively when r_iA_i=1.

Both assumptions (a1) and (a2) are related to the occurrence of request and its relinquishment in the device i, and the assumption (a3) is related to the common resource possession of the device i. These assumptions are satisfied enough in usual environments.

2.2 Action of Basic Cell

The circuit of the basic cell is shown in Fig.3. Variables (x,y, r) and (X,Y,A) are called as input variables and output variables respectively. In succeeding descriptions of this chapter, suffixes showing cell numbers are omitted, provided there is no misunderstanding.

RS1 is an RS flip-flop comparing the arrival time of P with the occurrence time of request, and RS2 is an RS flip-flop remembering P. Two output values Q_1, Q_2 in RS1 and one output value Q_3 in RS2 are selected as internal state variables.

The state of a cell is defined as one dimensional array (x y r Q_1 Q_2 Q_3 X Y A) of input variables, internal state variables and output variables. State transitions between stable states in fundamental mode operation are shown in Fig.4. Labels attached to arrows show

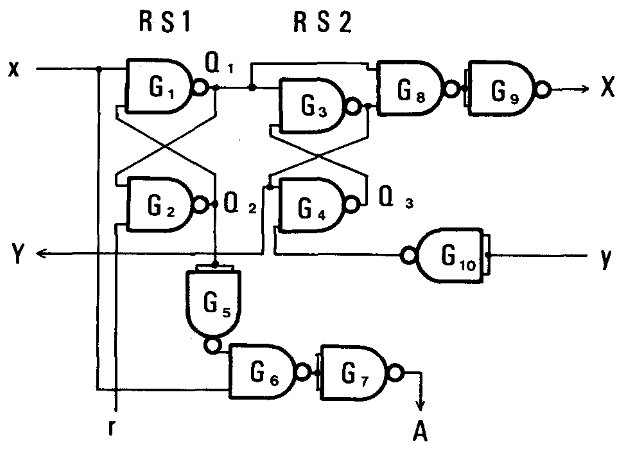

Fig.3 Configuration of a Basic Cell

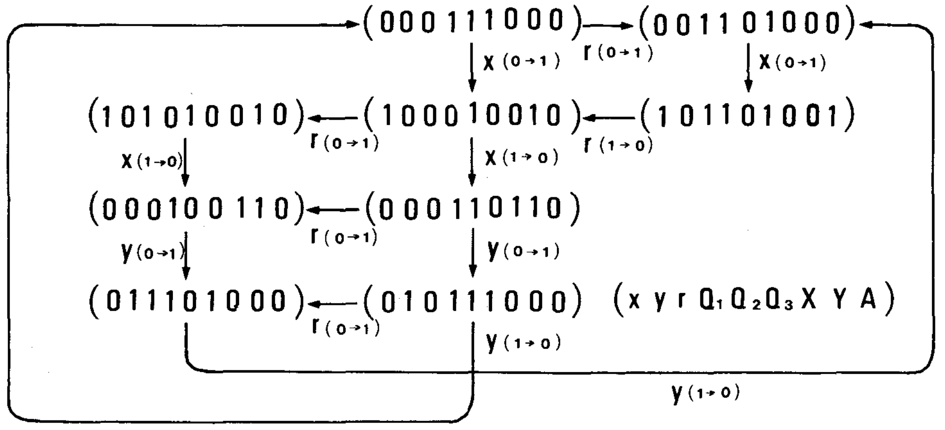

Fig.4 State Diagram of a Basic Cell

input changes causing corresponding transitions.

If r=0 before the arrival of P, the cell is in the state (000 111 000) and if r=1 before the arrival of P, it is in the state (001 101 000). Suppose that P arrives when the cell is in the state (000 111 000). x(0→1) arises. Subsequently, Q_1(1→0), Y(0→1) and Q_3(1→0) arise in order, and finally, the state reaches (100 010 010), sending Y=1 to the preceding cell.

When the cell is in the state (001 101 000), if P arrives, A(0→1) arises and the corresponding device starts to possess the common resource. If it finishes the common resource possession, r(1→0) arises and then Q_2(0→1) arises. Subsequently, Q_1(1→0), Y(0→1) and Q_3(1→0) arise in order and A(1→0) arises independently in parallel with these changes. And finally, the state becomes (100 010 010).

The next action of the cell is as follows, regardless of whether the request has been acknowledged or not.

If r=0 after Y(0→1), variable changes x(1→0), Q_1(0→1) and X(0→1) arise in order and then the state will be (000 110 110), and if r=1 after them, it will be (001 100 110) after Q_2(1→0). Consequently, the cell starts to send P to the next one. When the next cell receives P, y(0→1) arises at first and thereafter Q_3(0→1), Y(1→0) and X(1→0) arise in order. Then, the cell reaches state (010 111 000) or (011 101 000) according to whether r=0 or 1 and finally it returns to the initial state by y(1→0).

In above descriptions, the behavior of a cell is analyzed in fundamental mode. The next, we will analyze it in nonfundamental mode. In this mode, more than two inputs may changes in a very small interval of time. There are only four possible combinations of such changes in this cell, which are {x(1→0), r(0→1)}, {y(0→1), r(0→1)}, {y(1→0), r(0→1)} and {x(0→1), r(0→1)}. Because, xy=0. If any one of these combinations happens, a race will occur. It is a noncritical race in each of preceding three combinations. However, it is a critical race in the last combination.

If both x(0→1) and r(0→1) occur in a very small interval of time, RS1 may fall into metastable state. Therefore, the basic cell is protected from the possibility of malfunctions based on the occurrence of this state by raising threshold voltages in gates G_1 and G_2. The internal configuration of these gates is shown in Fig.5. Diodes D_1 and D_2 are inserted to raise the threshold voltage and diodes D_3, D_4,

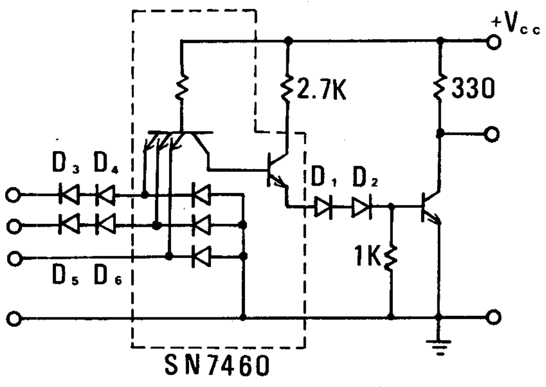

Fig.5 Internal Configuration of High Threshold Gate

D_5 and D_6 are added to match input levels in these gates with output ones from preceding stages. Consequently, even if the metastable state occurs, the cell will operate without any difference from the fundamental mode operation except that it reaches either of two states ((101 010 010), (101 101 001)) delayed by the interval at the metastable state. It is also apparent that there is no hazard in each race.

2.3 Initialization

It is necessary for the proper operation of the basic arbiter to generate only one privilege in the chain at initial condition. In this arbiter, the initialization can be performed by supplying a pulse to only one cell. Suppose it to be C_i without loss of generality. Additional circuits to C_i for the initialization are shown in Fig.6. At first, when the value of R turns to logic level 1 from 0, the states of C_i and C_{i-1} turn to (000 110 010) and (010 111 000) after a certain interval, respectively. The remaining cells stay in (000 111 000). Where it is assumed that no request is occurred in the interval of initialization. Thereafter, as soon as the value of R returns to logic level 0, a level signal $Q_3=0$ in C_i starts to go round all the cells as a privilege P ($X_i(0 \rightarrow 1)$).

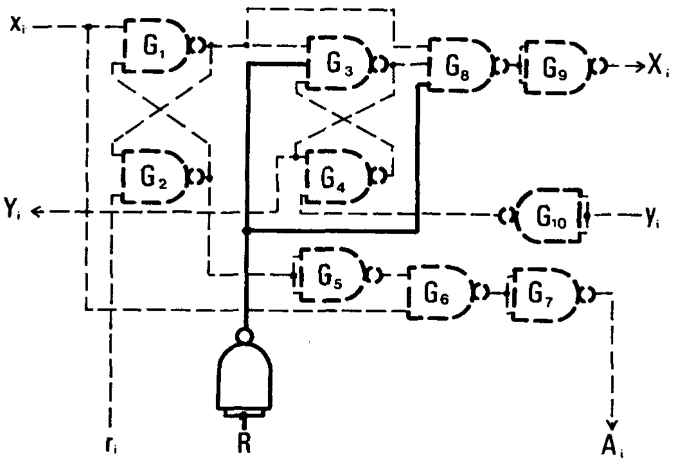

Fig.6 Initialization of Basic Arbiter

3. Modifications of Basic Arbiter

3.1 Speed-up of Basic Cell

Assuming that the propagation delay time in any gate is equal to d [nsec]. It takes 10d [nsec] for the privilege to pass through a basic cell (from $x_i(0\to1)$ to $X_i(0\to1)$), unless any request exists ($r_i=0$). Consequently, it takes 10nd [nsec] for the privilege to make a round all the cells.

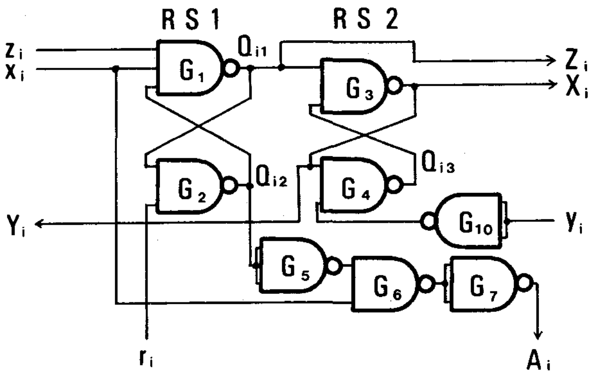

Fig.7 Configuration of a Cell for 3 Wire Type Arbiter

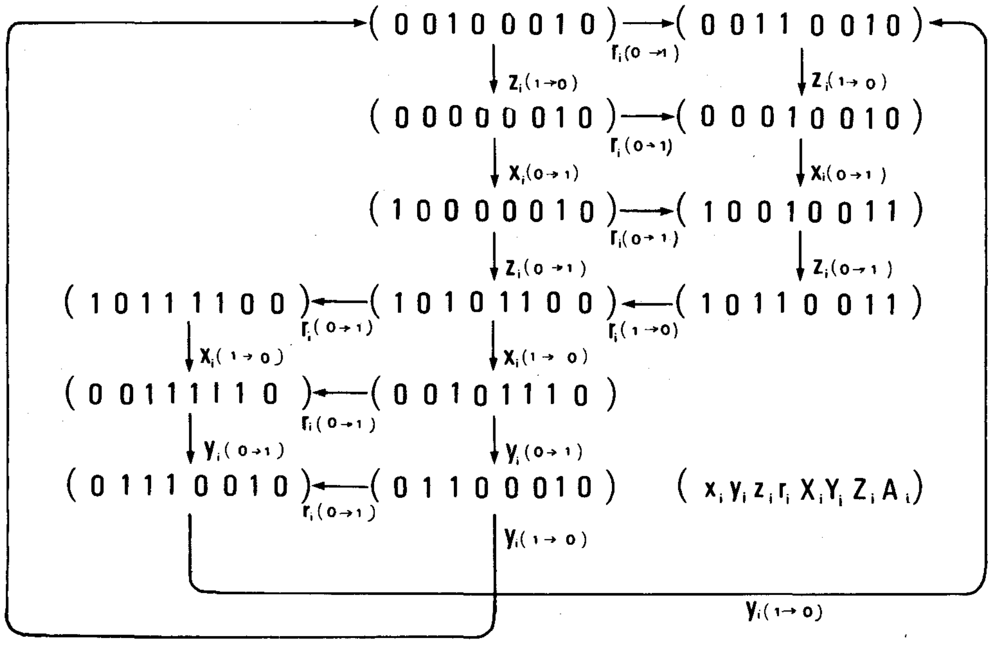

Fig.8 State Diagram of a Cell for 3 Wire Type Arbiter

Let us consider modifications of the basic cell for speed-up. Fig.7 shows a cell obtained by addiong one more wire to the basic cell to interconnect two adjacent cells. Hereafter the arbiter composed of such cells is referred to as 3 wire type arbiter. In this cell, two input signals of G_8 in the basic cell are sent directly to the next cell. As the result, a 3 input NAND gate is used as G_1.

Let us define the state of the cell C_i in Fig.7 as $(x_i y_i z_i r_i\ X_i Y_i Z_i A_i)$. The state diagram in Fig.8 shows transitions between stable states, where the cell C_i operates in fundamental mode. It is apparent from Fig.7 that it takes 6nd [nsec] for P to make a round all the cells.

The next, Fig.9 shows a cell obtained by adding one more wire to the cell (shown in Fig.7) to interconnect with two adjacent ones. Hereafter the arbiter composed of such cells is referred to as 4 wire type arbiter. In this cell C_i, the release of the privilege in the preceding cell C_{i-1} is informed not by the signal $Q_{i-1,1}$ (the output of G_1 in C_{i-1}), but by the signal Z_{i-2} (the output of G_4 in C_{i-2}). As the result, the input signal z_i in C_i is equal to the output signal $Z_{i-2}(=W_{i-1})$.

Let us define the state of the cell shown in Fig.9 as $(x_i y_i z_i w_i\ r_i\ X_i Y_i Z_i W_i A_i)$. The state diagram is shown in Fig.10. Only stable states are shown. Calculating as in 3 wire type, it takes 4nd [nsec] for P to make a round of all the cells.

In either 3 wire type arbiter or 4 wire type arbiter, the initialization can be performed in the similar way as the basic one and the possibility of malfunctions also disappears. Besides, changes of

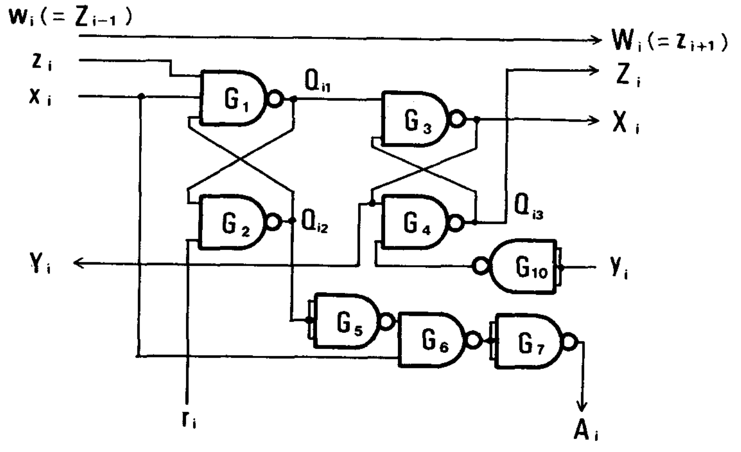

Fig.9 Configuration of a Cell for 4 Wire Type Arbiter

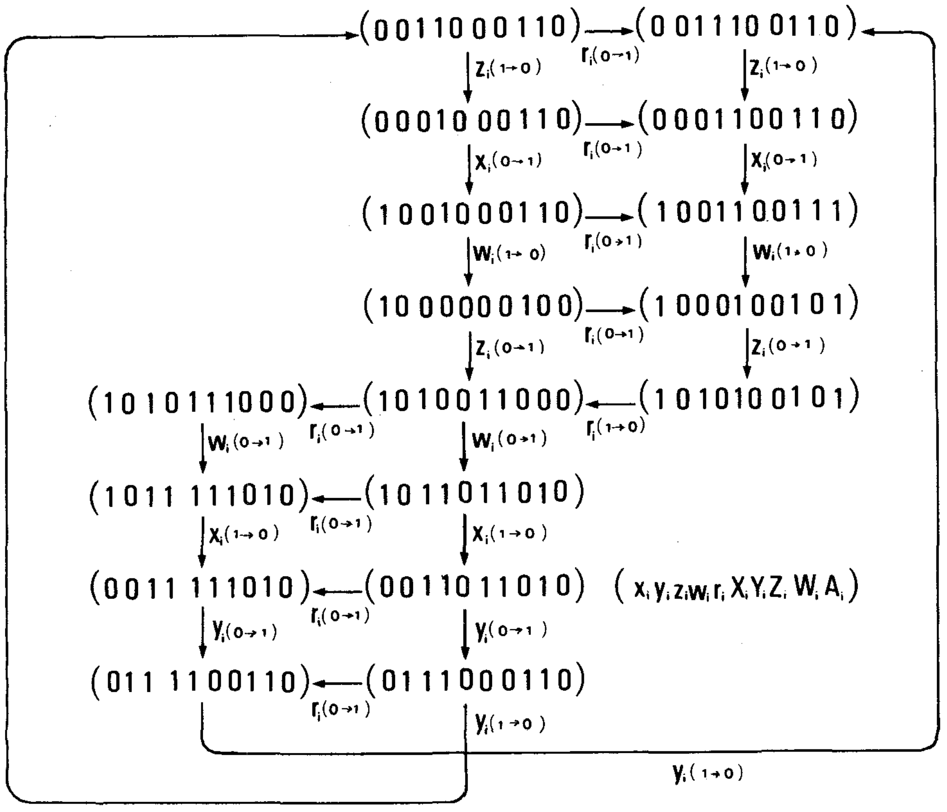

Fig.10 State Diagram of a Cell for 4 Wire Type Arbiter

internal state variables accompanied with the occurrence of the request and the arrival of P ($z_i(0\to 1)$) are just the same as in the basic cell.

3.2 Supplement of Priority

One method to add a priority to the basic arbiter is one that, if there is no request in the chain, P is fixed to the specified cell C_1 and if any request is issued in any cell except C_1, it starts to go round. Fig.11 shows an arbiter with a priority rule obtained by modifying the basic one, based on such a policy.

r_0 is a variable, the value of which is logic level 1 when and only when any request except r_1 exists. The privilege P is fixed at $C_1(X_n(=x_1)=1)$ as long as $r_0=0$. When any request is issued except r_1, $r_0(0\to 1)$ will arise. Then, P starts to make a round all the cells. If the device 1 is possessing the common resource, P starts to make a round at the end of its possession. When P reaches C_1 again,

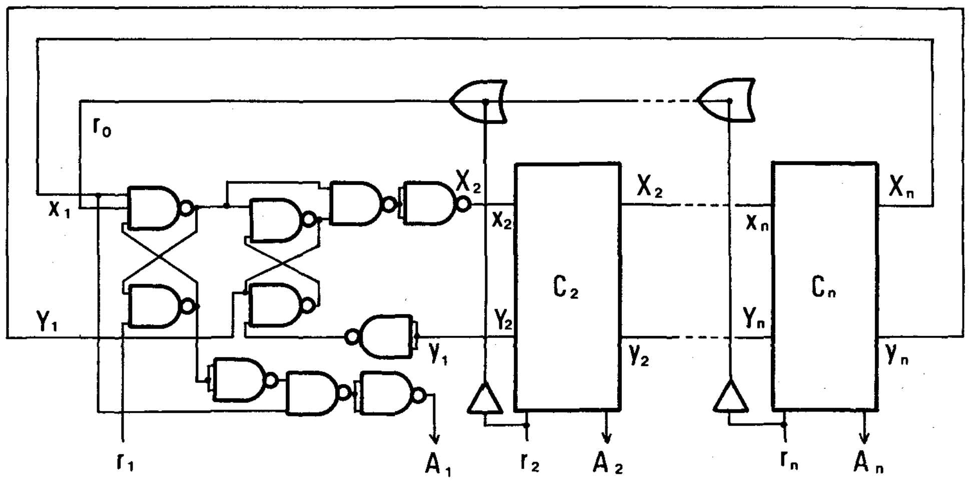

Fig.11 An Arbiter with a Priority Rule (1)

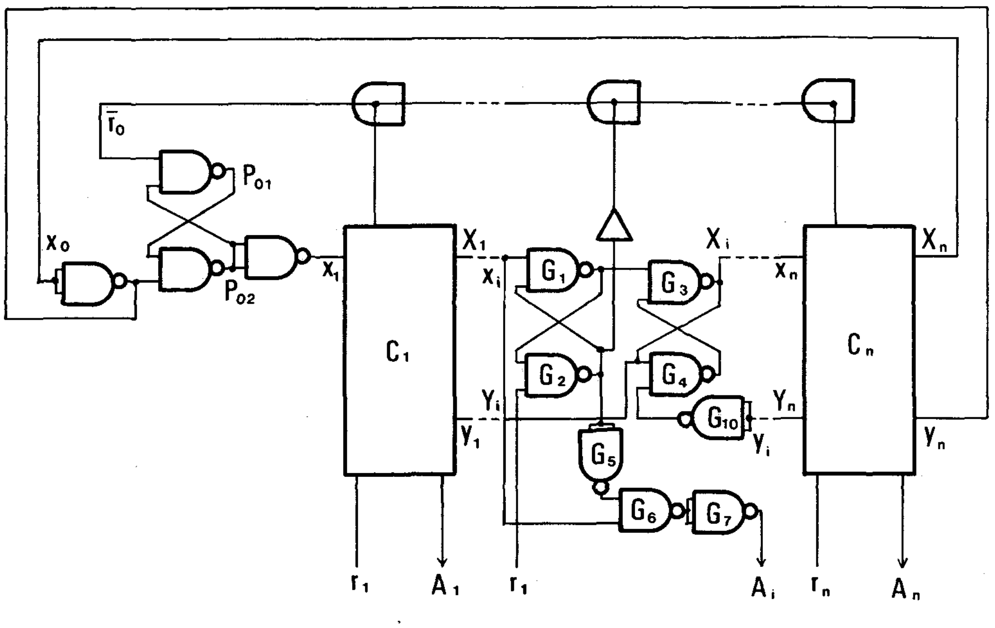

Fig.12 An Arbiter with a Priority Rule (2)

the state of the arbiter is returned to the original state. Then, if there exists any request except $r_1(r_0=1)$, P will make a round again all the cells.

This arbiter is suited to systems, in which only one specified device issues requests frequently.

Another modification with a priority rule is shown in Fig.12.

Cells used in this arbiter are obtained by removing G_8 and G_9 from the cell in Fig.3. When there is no request in the chain ($\bar{r}_0$=1), P is staying in the control circuit (x_0=0, P_{01}=0, P_{02}=1). If any request is issued from any device, $\bar{r}_0(1 \to 0)$ and $x_1(0 \to 1)$ arise in order and P starts to make a round of the chain.

If $\bar{r}_0(1 \to 0)$ arises, only the leading edge of P is preferentially transmitted to succeeding cells in order. This edge arrives again at the control circuit after a round of the chain. Then the transmission of the trailing edge starts to go round cells. And finally, it returns to the control circuit. Then, P stays in the control circuit until a new request is issued.

This arbiter is suited to systems, in which a priority rule among all the devices is required.

4. Conclusions

In this paper, some modifications of an asynchronous ring arbiter composed of a chain of cells are proposed. They are divided into two classes. One is a class with high speed ability, which is obtained by allowing the number of connecting wires between adjacent cells to increase in the original arbiter. The other is a class with priority rules, which is obtained by modifying only one of cells in the original arbiter or by adding a few circuit in a part of it. Besides, many other modifications with priority rules may be also realizable according to requirements of systems.

Consequently, this ring arbiter may be applicable to a wide range of computer systems.

References

[1] W.W.Plummer : "Asynchronous arbiters", IEEE Trans. Comput., Vol.C-21, No.1, pp.37-42 (Jan. 1972).

[2] T.NANYA and N.KOIKE : "A construction of asynchronous arbiter", Trans. IECE Japan, Vol.J 57-D, No.4, pp.242-244 (April 1974).

[3] R.C.Pearce, A.Field and W.D.Little : "Asynchronous arbiter module", IEEE Trans. Comput., Vol.C-24, No.9, pp.931-932 (Sep. 1975).

[4] S.HORIGUCHI : "A new type of multi-input asynchronous arbiter",

Trans. IECE Japan, Vol. J61-D, No.9, pp.597-604 (Sep. 1978).

[5] H.YASUURA and S.YAJIMA : "Design of asynchronous arbiters from the standpoint of asynchronous sequential circuit theory", Trans. IECE Japan, Vol. J61-D, No.12, pp.917-924 (Dec. 1978).

[6] P.Corsini : "n-user asynchronous arbiter", Electron. Lett., 11, 1, pp.1-2, (Jan. 1975).

[7] T.OKAMOTO, M.SAKAI, T.HONDO and T.HOSOMI : "A realization of asynchronous arbiters", Trans. IECE Japan, Vol. J59-D, No.8, pp.582-583 (Aug. 1976).

[8] H.MASUYAMA and N.YOSHIDA : "Ring arbiter composed of asynchronous control cells", Trans. IECE Japan, Vol. J63-D, No.2, pp.197-199 (Feb. 1980).

[9] H.MASUYAMA and N.YOSHIDA : "A simple ring arbiter using NOR-latch cell", Trans. IECE Japan, Vol. J63-D, No.6, pp.550-551 (June 1980).

[10] H.YASUURA, K.IWAMA and S.YAJIMA : "A ring arbiter using oscillation of asynchronous logic circuits", Trans. IECE Japan, Vol. J64-D, No.1, pp.80-81 (Jan. 1981).

[11] T.OKAMOTO and M.FUJIWARA : "Design of an asynchronous ring arbiter", Trans. IECE Japan, Vol. J64-D, No.9, pp.846-853 (Sep. 1981).

[12] T.OKAMOTO and M.FUJIWARA : "An asynchronous ring arbiter with built-in testability", Trans. IECE Japan, Vol. J65-D, No.1, to be published.

[13] T.MATSUURA, T.SAKEMOTO, S.YANO, M.FUJII, N.TOKURA and T.OKAMOTO : "A minicomputer complex and its operating system", Trans. of Information Processing, Vol.18, No.9, pp.913-920 (Sep. 1977).

[14] T. OKAMOTO and T.KOBAYASHI : "A consideration on two subsystems coupled closely through a telephone line", Trans. IECE Japan, Vol. J63-D, No.5, pp.425-432 (May 1980).

[15] K.INOUE, T.TANIZAWA, K.TANIGUCHI and T.OKAMOTO : "Implementation of functional programming language FPL", Trans. IECE Japan, to be published.