MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Computation Structures Group Memo 67

Forward Acting n x m Arbiter

by

Suhas S. Patil

April 1972
(Revised June 1972)

This research was done at Project MAC, M.I.T., and was supported in part by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr N00014-70-A-0362-0001.

Forward Acting n x m Arbiter

An n-resource m-user arbiter performs the coordination necessary for m-users to share resources from a pool consisting of n units of resource. The basic task of the n x m arbiter is to act on the request of a user and to decide which resource unit should be allocated to the user. When a resource unit is not available immediately the user must wait until one becomes free. The arbiter presented here is completely asynchronous and performs actions in parallel. The arbiter consists of asynchronous modules connected together to perform the needed coordination (synchronization). The arbiter is an example of a distributed asynchronous modular system.

This n x m arbiter is considerably simpler in structure, operation and detail than a previously reported n x m arbiter [1]. The scheme employed in the arbiter reported here is different from the earlier arbiter in fundamental ways. The major improvement in this arbiter is that unlike the previous arbiter there is no repetition of action involved. That is, the arbiter does not retract any steps -- all steps (without any exception) advance a request closer to being granted. Because this arbiter always steps forward, it is called a forward acting arbiter.

Both the scheme and the circuits for the forward acting arbiter are presented in this paper. The abstract specifications of the modules of the arbiter are expressed as P-nets. The circuits for the modules are also presented. These circuits use elementary arbiters in addition to AND, OR and NOT gates. An elementary arbiter is the basic circuit element which resolves conflicts between concurrent requests. In the circuits inside a module, the gates and the elementary arbiter are assumed to act within a known length of time. The operation of the n x m arbiter is insensitive to variation in propagation delays of signals between modules. The n x m arbiter, viewed as interconnection of modules, is a speed independent structure.

The work herein was supported in part by the Advanced Research Projects Agency, Department of Defense, under the Office of Naval Research Contract Nonr N00014-70-A-0362-0001.

Operation and Structure of n x m Arbiter

An n x m arbiter has n input links, one for each user (Figure 1). It also has n x m output links. The output link S_{ij} corresponds to allocation of resource i to user j. A user sends a ready signal to the n x m arbiter on link I_j to request allocation of a resource unit. When a resource unit is allocated to the user, the arbiter sends a ready signal on the appropriate output link. This ready signal then performs the action that is necessary to utilize the resource. When the use of the resource is completed, an acknowledge signal is sent in reply to the ready signal. Upon receiving the acknowledge signal, the arbiter knows that the resource is free and may allocate it to other waiting users. After the arbiter has recorded this information, an acknowledge signal is returned to the user to acknowledge completion of the task.

A 2 x 3 arbiter and its use in sharing 2 functional units by 3 users is shown in Figures 1 and 2. The operation of a forward acting n x m arbiter is explained with the aid of these figures. All links used in Figure 1 are two-wire links, one wire carries signal in the direction of the link and the other in the direction opposite that of the link. The n x m arbiter has an arbiter for each unit of resource and an arbiter for each user of the resources. The arbiters associated with resource units are called resource arbiters and those associated with users are called user arbiters. In Figure 1, A_1 and A_2 are resource arbiters and B_1, B_2 and B_3 are user arbiters. Basically, the resource arbiters prevent more than one user from getting access to a resource unit and the user arbiter is there to choose a particular resource unit to meet a user request if more than one resource unit is available for use.

The actions in the n x m arbiter take place in parallel. The signal representing a request from a user is broadcast to all resource arbiters by the W(wye) module. The resource arbiter associated with resources which are free allow these signals to pass through them. This action represents an offer of service by these resources to the user and as this offer is made, the resources are rendered engaged. It is the task of the resource arbiter to ensure that a resource unit is engaged to at most one user. The signals representing offers of service by the resources travel to the user arbiter associated with that user. The user arbiter then accepts one offer and turns down other offers by sending termination signals to all resource arbiters except the one whose offer was

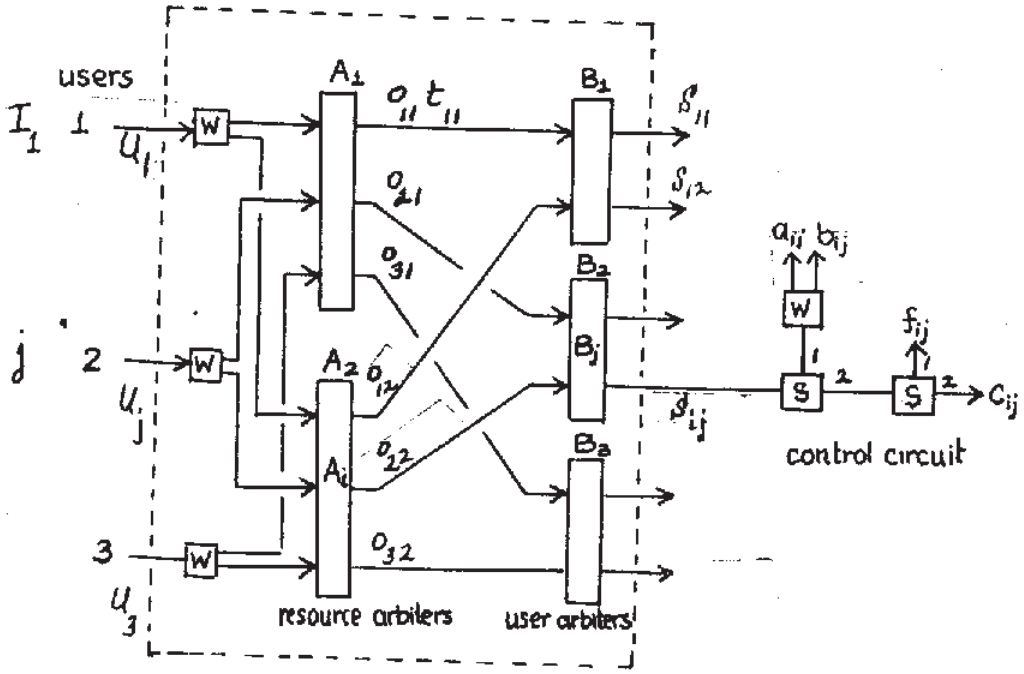


Figure 1 An nXm arbiter

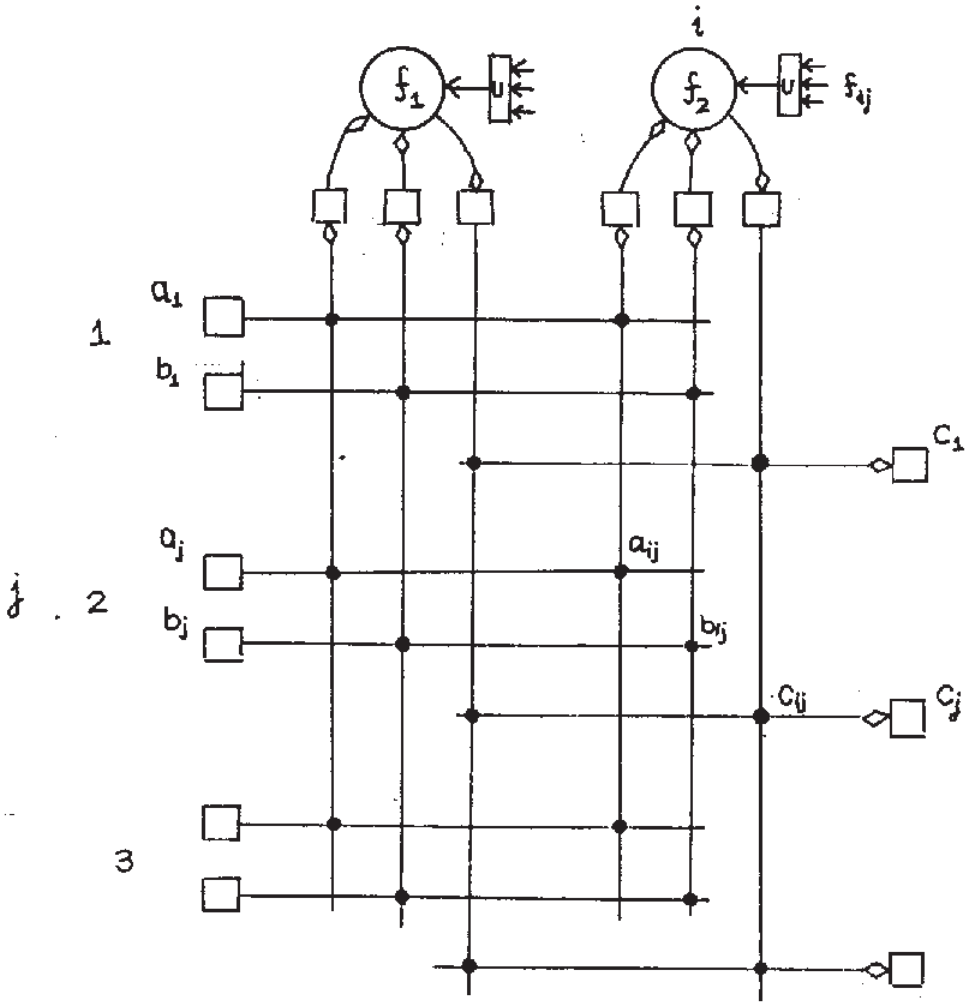


Figure 2 Data flow structure for using the functional units

accepted. If for some reason a resource arbiter has not yet made an offer to a user and the termination signal from that user is received, the resource arbiter promptly allows the request signal from the user to go through it which is like making an offer regardless of whether the resource was available. To the extent that the resource may not be available, this offer may be fake, but the resource arbiter can play this game because having received the termination signal from the user he knows full that the offer will not be accepted and therefore he has nothing to lose in making that offer. Thus, soon after the termination signals are sent, signals from all the remaining resource arbiters are received at the user arbiter. Upon receiving all of these signals, the user arbiters sends out the signal from the resource arbiter whose offer of service was accepted; the other signals are not sent out. They are terminated. Thus when user j gets service from resource i, the output link S_{ij} of the n x m arbiter gets a signal. This signal, called a service signal, initiates the action that is necessary to put the resource to the service of the user. In the specific use of n x m arbiter in Figures 1 and 2, the ready signal on link S_{ij} first moves input operands a_j, b_j to the operator i, applies the operator and then makes the result c_j available to the user. This task is performed with the control circuits shown in Figure 1 and the switching matrix shown in Figure 2. In the control circuit S is a sequence module and W is a wye module. The S module sequences action and the W module performs actions in parallel [2]. When the action is completed, a completion signal is returned on link S_{ij} to indicate that the use of the resource is completed. Upon receiving this signal the user arbiter sends a termination signal to the resource arbiter associated with the resource. The resource arbiter becomes free and an acknowledge signal is sent to the W module associated with the user. The W module then returns acknowledge signal to user to indicate the completion of the task.

Specification of the Resource and User Arbiters

Now we will provide a formal specification for the resource and user arbiters used in n x m arbiter. The signals involved in the operation of n x m arbiter are identified by names shown in Figure 3. This figure also shows the wires constituting the links and the way the modules are connected. The wires are named according to the function of the signals they carry. A ready signal from the user becomes a request to the resource arbiter, then it becomes an offer of resource and finally it becomes a signal indicating service by the resource. The end of service is indicated by a completion signal received at the user arbiter. This becomes a termination signal which in turn becomes an acknowledge signal. The operation of resource arbiters and user arbiters is expressed in terms of these signals in the diagrams shown in Figures 4 and 5. These diagrams are abstract nets called P-nets which are used to specify when different signals are to be sent out. P-nets represent a language for specification of interdependence among signals of an asynchronous system [3]. This language is derived from the Petri nets of Holt [4].

P-nets can be thought of as a game in which players advance tokens. There are some nodes in the net called places, drawn as circles, which may have tokens from the beginning and into which tokens can be placed in the course of the game. The other nodes of the net have names of signals associated with them. To get past these nodes it is necessary to bring one token from each incident arc. In addition if the signal written at the node is an input signal it is necessary to have received that input. In the act of going through a node, the signal written at the node is absorbed if it is an input node or if the signal written at the node is an output signal, the signal is sent out. A null signal may be associated with a node. A null signal is always available. In coming out of a node one token becomes available on each arc emergent from the node. A configuration of special interest is one in which arcs are drawn to several signals from a common place. In this case the player is allowed only one of the several alternative moves because any one move removes the token from the place and thereby rules out the other moves. If several moves are possible, this situation calls for resolution of conflict, i.e., it calls for arbitration. The other moves would not be ruled out by the move if the place has more than one token but this situation does not arise in our use of P-nets. If a node

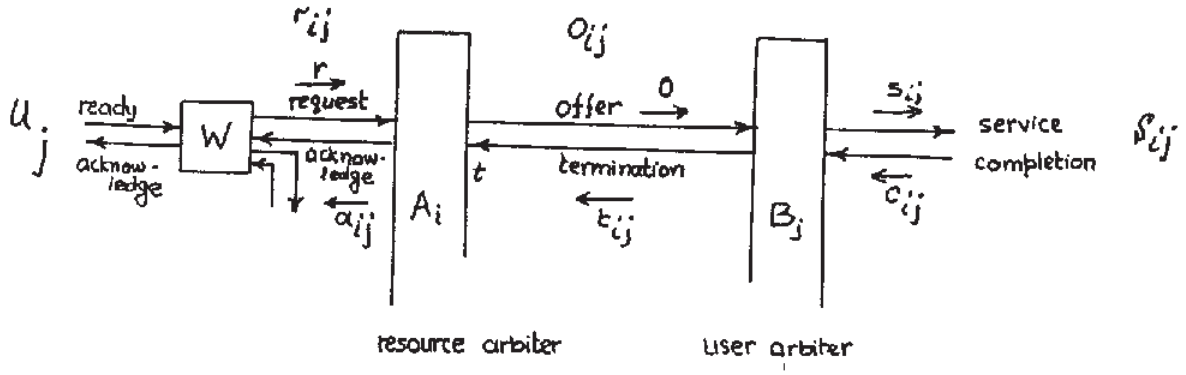


Figure 3 Names of the signals and wires

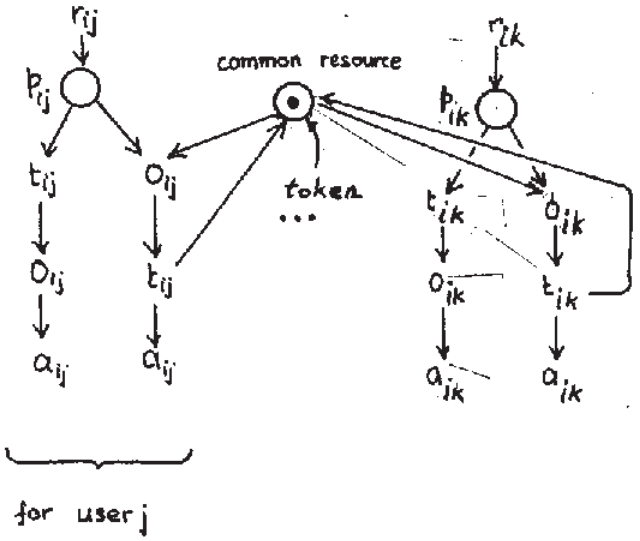


Figure 4 P-net for resource arbiter A_i

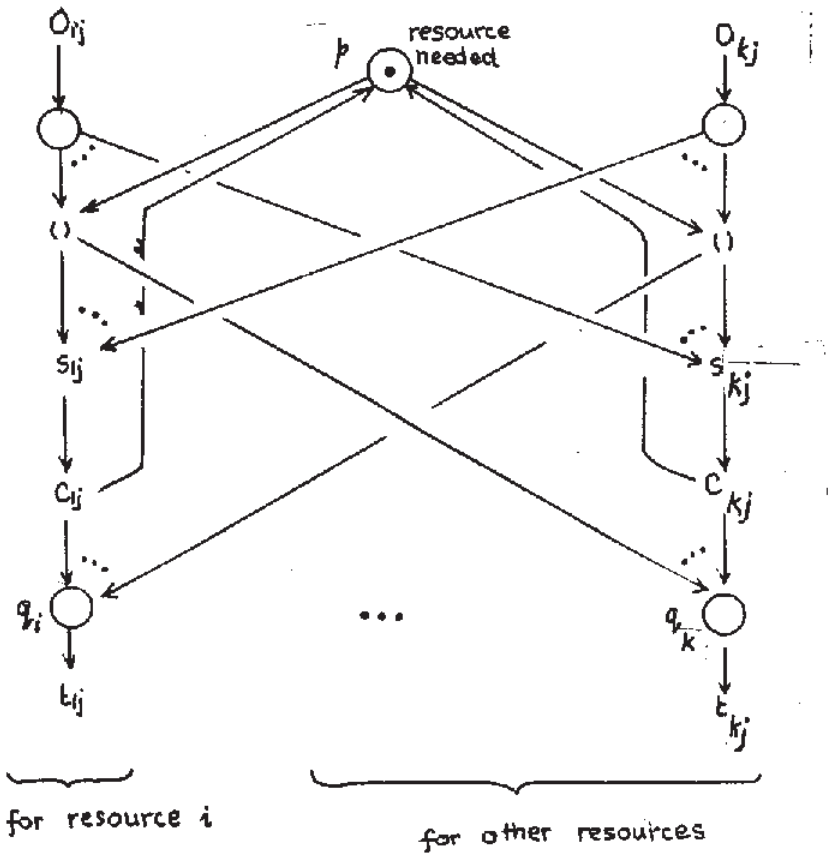


Figure 5 P-net for user arbiter B_j

does not have any input arcs the node is ready to be played as soon as the signal becomes available. A player is required to promptly proceed with any moves that are possible. That is, a player may rest only if no moves are possible. In such a case, he waits for signals which will make moves possible. The P-net for a resource arbiter is shown in Figure 4. As soon as request r_{ij} arrives a token is placed in place p_{ij}. Now if the resource is available, signal o_{ij} (offer of resource) is sent out. Otherwise no action takes place until t_{ij} is received. If the path on the right is allowed (i.e., offer of resource is made), the place representing the resource becomes empty. This means that the resource becomes engaged and is not available to the other users. The resource is freed when t_{ij} is received. The path on the left is followed when termination signal t_{ij} is received before the resource becomes available to this user. In this case signal o_{ij} is sent out after t_{ij} is received, and then acknowledge signal a_{ij} is sent out.

The P-net for the user arbiter is shown in Figure 5. Input signal o_{ij} represents the offer of resource i. The first such offer (one of such offers if several are received at about the same time), say o_{ij}, removes the token at p by firing of the null event associated with resource o_{ij}. At the same time tokens are sent down to input places of termination signals associated with other resources. Thus the termination signals are sent out. After the termination signals are sent out, we are assured that o_{jk} signals will be received from all other resources if they are not received already. When this happens, signal s_{ij} is sent out to indicate that resource i will service user j. When completion signal c_{ij} is received, tokens are placed in places p and q_i. Then the termination signal t_i is sent out. This completes the action of the user arbiter.

Circuits

The circuits for n x m arbiters presented here employ Muller's signalling convention in that all signals areprepresented by 0 to 1 transitions [5]. Therefore, after a circuit is used, it is necessary to reset all levels in the circuit to zero before using the circuits once again. This signalling convention is different from the signalling convention used by Clark [6] and Patil [2] in which all transitions whether from 0 to 1 or from 1 to 0 represent signals. Muller's signalling convention is used in n x m arbiter because it seems to be more suited to the problem and provides simpler circuits. It should be noted that a ready-acknowledge link operating in one convention can be easily transformed into a link operating in the other convention by use of the modules shown in Figure 9.

The circuit for a resource arbiter is shown in Figure 6. This circuit uses an m-input arbiter whose implementation in terms of elementary arbiters will be shown later. The m-input arbiter has m input-output pairs. Arbitration starts when one or more of input wires go high. One of the signals reaches the output wire and the other signals are blocked by the arbiter. If at any time the level of the input wire goes down to 0, the arbiter is freed if it was engaged to that input and the level of the output wire associated with this input is forced to come down to 0.

The remaining circuit associated with the link provides an alternate path for r_{ij} to get to o_{ij} when termination signal t_{ij} goes high. Essentially this circuit performs multiplexing of wire r_{ij}; when t_{ij} is 0 the logical connection goes through the arbiter and then t_{ij} is 1 the logical connection by-passes the arbiter.

The circuit for the user arbiter is slightly more complicated. It also has a multi-input arbiter in it. But in this case an engaged signal is brought out of the arbiter and also a block input is provided. The block input is turned on after the arbiter is engaged to some input, and the effect of the block signal is to keep the arbiter engaged to that link even when level in that link goes to 0. The block input ensures correct operation of the user arbiter in the resetting phase of operation. To explain the operation of the circuit we will trace the operation of the circuit from the initial condition. Initially levels of o_{ij}'s, c_{ij}'s, t_{ij}'s and s_{ij}'s are 0 and the t-bus and block bus are also at level 0. Now, suppose signal o_{ij} is received. This means that level

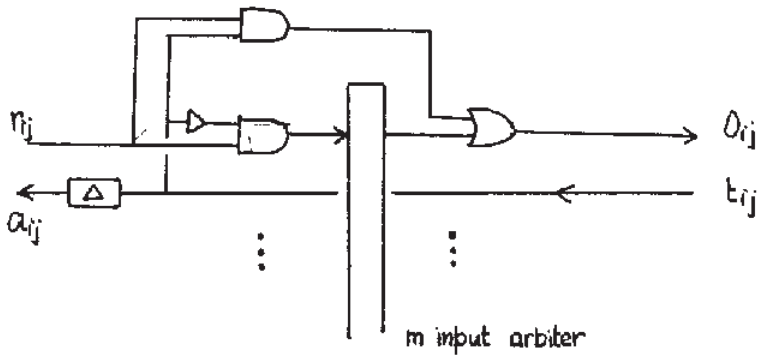


Figure 6. Resource arbiter

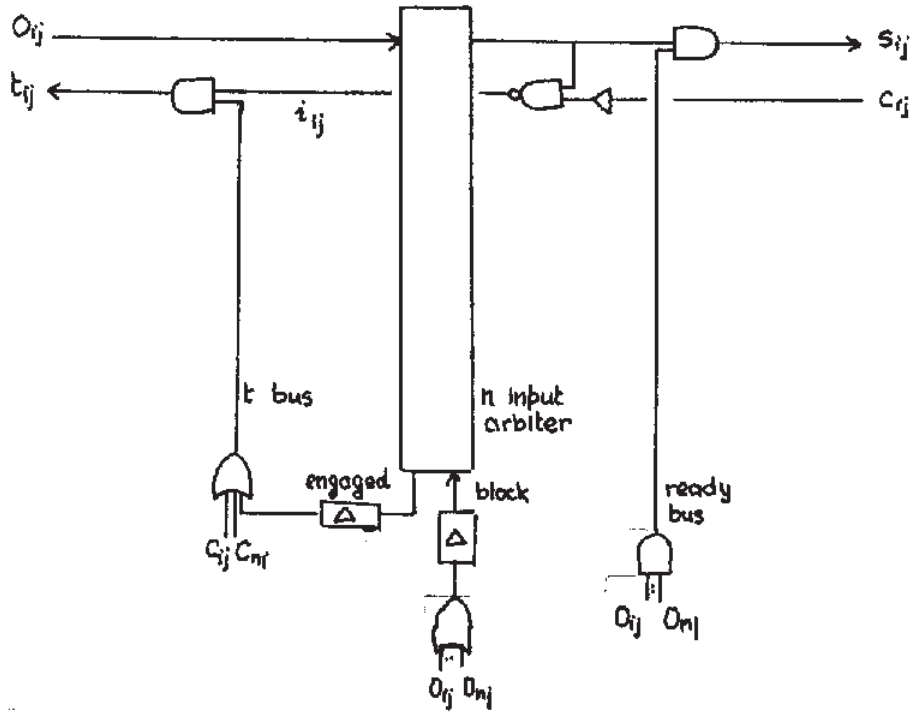


Figure 7 User arbiter

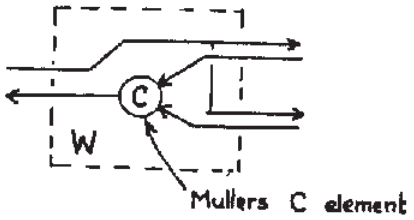
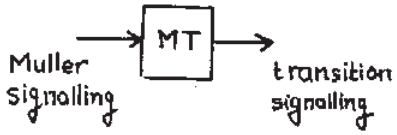
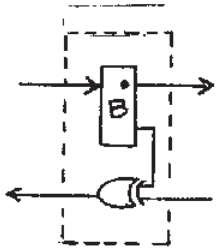
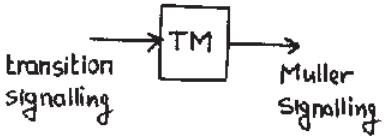
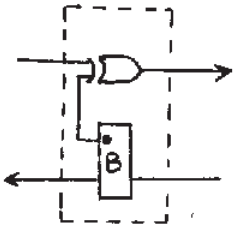


Figure 8 The wye module









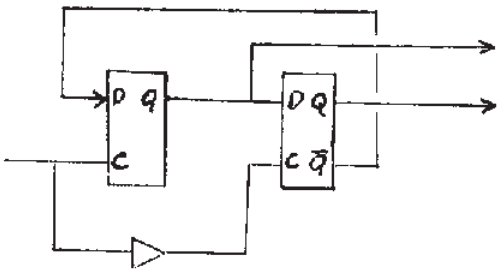


Figure 9 Transformation of signalling links

of o_{ij} goes high. The arbiter is engaged by o_{ij} and the corresponding output link of the multi-input arbiter goes high. This brings the level at point i_{ij} to 0 -- the value of 0 at i_{ij} is an inhibit signal to prevent the expected change in value of t-bus from reaching t_{ij}. Some time after the inhibit signal has taken effect, the engaged signal reaches the t-bus and the t-bus goes high. The effect of this is to send level 1 on all termination wires except t_{ij}. With this action we are assured that all o_{kj} wires will receive signals -- their level will go to 1. The block input to the multi-input arbiter may be ignored at this time because it plays an important role only in the resetting of levels. When all o_{ij}'s become 1, the ready bus becomes 1 and the gates in the path of s_{ij} are opened. The service wire s_{ij} associated with the o_{ij} wire now becomes 1. After service is performed, c_{ij} becomes 1. Level of 1 on c_{ij} travels to point i_{ij} and t_{ij} goes to 1. This completes the signalling performed by the user arbiter. Resetting of levels to initial condition so that the user arbiter may be used once again is explained below.

In the resetting phase when all o_{kj} become zero, the lock signal becomes 0, the arbiter is freed and the engaged signal becomes 0. At the same time s_{ij} becomes 0 and subsequently c_{ij} becomes 0. Finally t-bus becomes 0 and all t_{kj} wires reset to 0.

Multi-Input Arbiters from Elementary Arbiters

There are many ways in which multi-input arbiters can be implemented (Plummer [7]). Below we present a modular structure for obtaining multi-input arbiters from elementary arbiters (Figure 10). This structure, in its basic mode of operation, is similar to the one employed in construction of multi-input arbiters in the previously reported n x m arbiter. The T module and the elementary arbiter (EA) module are separately shown in Figures 11a and 11b. Under normal operation the T module behaves in a manner that makes wires 1 and 1', and wires 2 and 2' look as if they were connected together. The only difference is that the T module has memory in it; if wire 1 goes low after being high sufficiently long to permit wire 1' to become high, wire 1' remains high until wire 2' becomes high.

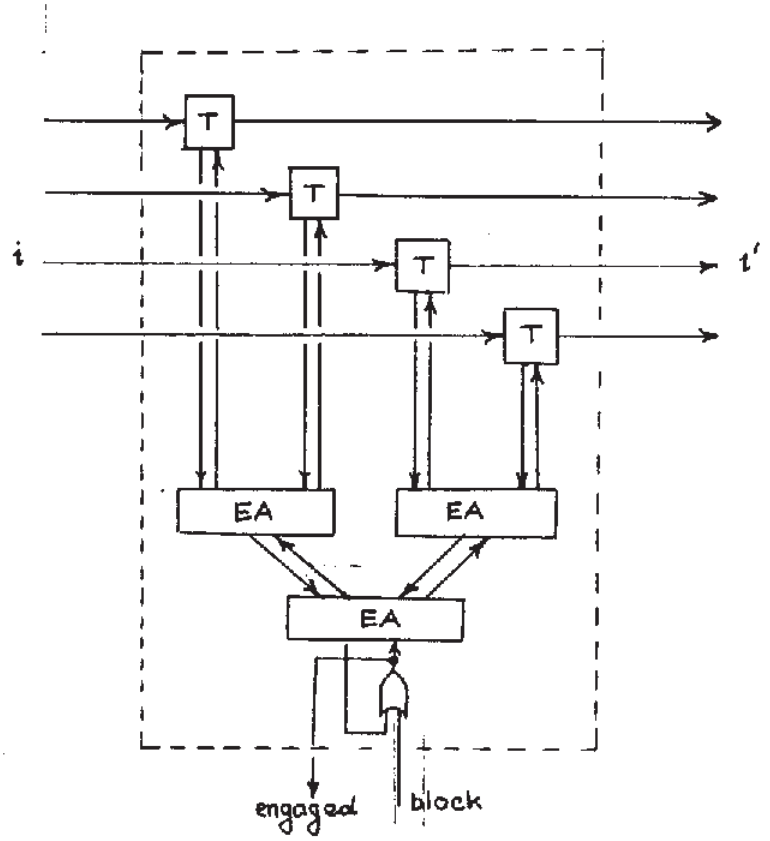


Figure 10 Multi input arbiter

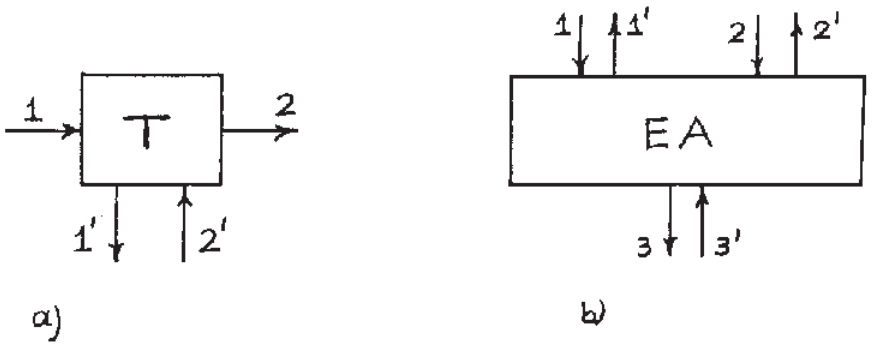


Figure 11 : T module and Elementary Arbiter

The operation of the elementary arbiter is explained next. Wires 1 and 2 may become high concurrently. When any one of these wires becomes high the elementary arbiter gets engaged to it; in the event the wires become high simultaneously, the elementary arbiter makes an arbitrary choice. Wire 3 then goes high. Subsequent to this wire 3' goes high and then either wire 1' or wire 2' goes high depending on whether the elementary arbiter is engaged to wire 1 or wire 2 (say wire 1' goes high).

In the resetting phase of operation, wire 1 goes low (remember that the elementary arbiter was engaged to wire 1 and wire 1' had become high). Then wire 3 goes low and later wire 3' goes low. Finally wire 1' goes low and the elementary arbiter is free to be engaged by the other input if it is waiting.

A perfect implementation of the elementary arbiter requires a solution to the synchronization problem. Practical circuits for it can be constructed by several methods including one presented by Plummer [7].

Conclusion

The n x m arbiter has come a long way since the first implementation suggested by author several years ago. The n x m arbiter is now simple enough and could be of practical use.

References

1. S. S. Patil, "N-server m-user arbiter," Project MAC Memo MAC-M-415, May 1969.

2. J. B. Dennis and S. S. Patil, Computation Structures. Notes for Subject 6.232, Department of Electrical Engineering, M.I.T., Cambridge, Mass., 1971.

3. S. S. Patil, Coordination of Asynchronous Events. Doctoral Thesis, M.I.T., June 1970 (Project MAC Technical Report MAC TR-72).

4. A. W. Holt and F. Commoner, Events and Conditions (in three parts). Applied Data Research, Wakefield, Mass., 1970. [Chapters I, II, IV and VI appear in the Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York, 1970, pp 3-52.]

5. D. E. Muller, Asynchronoug logics and application to information processing. Switching Theory in Space Technology, Stanford University Press, Stanford, California, 1963.

6. W. A. Clark, et. al., Macromodular computer systems. AFIPS Conference Proceedings, Vol. 30 (Spring 1967), pp 335-336.

7. W. W. Plummer, Asynchronous arbiters. IEEE Trans. on Computers, Vol. C-21, No. 1 (January 1972).