MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo 73

Circuit Implementation of Petri Nets

by

Suhas S. Patil

(This memo is identical to a paper printed and presented at an IEEE invited seminar at Honeywell, Boston, September 1971.)

December 1972

The work discussed in this article was done at Project MAC, MIT, and was supported in part by the National Science Foundation under research grant GJ-432, and in part by the Advanced Research Projects Agency, Department of Defense, under Naval Research Contract N00014-70-A-0362-0001.

Circuit Implementation of Petri Nets

Suhas S. Patil
Project MAC, M.I.T., Cambridge, Massachusetts

In earlier work the author presented modular structures for implementing Petri nets [1]. Those structures were completely speed independent in that they would perform correctly even if arbitrary delays were introduced in any wires. A price for complete speed independence is paid in the complexity of the modules in the structures. Where one can set an upper bound on transmission delays of signals in circuits, the complexity of circuits can be substantially reduced. The objective of this paper is to present such a simplified gate level implementation of Petri nets. The assumption of an upper bound on transmission delays is particularly appropriate when the interconnected parts are not separated by large distances. The relation between this work and the previous work is that the implementation of Petri nets presented here provides a systematic method for implementing the modules used in the earlier work. In that work the modules were specified abstractly and no circuits for them were given.

We will first consider a subclass of Petri nets known as conflict free Petri nets and then consider the general class of Petri nets. The reasons for doing this are several: The circuits for conflict free Petri nets have certain simplicity and elegance to them; these circuits are speed independent circuits of a kind while the circuits for the general Petri nets are not necessarily speed independent; and knowing the circuits for conflict free nets is very helpful in understanding the circuits for the general Petri nets.

Work reported herein was supported in part by Project MAC, an MIT research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-N00014-70-A-0362-0001.

Petri nets

It may be recalled that a Petri net [2] is a directed graph which can have two types of nodes, namely transitions and places, and the directed arcs can connect only transitions to places and places to transitions (Figure 1). The places can hold markers. A transition can fire only if it can get a marker from each of its input places. When each of the input places of a transition has a marker in it the transition is said to be in enabled state. An interesting case arises when two transitions which share an input place are enabled and the common input place has only one marker. This situation is known as a conflict; the two transitions are in conflict over the marker at the shared place because firing of any one of the transitions disables the other transition. Conflict free nets are nets in which the transitions may share input places but the initial marking distribution is such that in the operation of the net there is no possibility of a conflict arising. Another important property relating to nets is safety. A safe net is one in which no place will ever have more than one marker in it at the same time. A bounded net is one in which there is a fixed upper bound as to the number of markers that will be in a place. All bounded nets can be transformed into safe nets by means of simple transformations. The circuits we present are for safe nets because a marker will be represented by the signal on a wire, and the level on a wire can take only one of two values, 0 and 1.

Circuits for safe conflict free Petri nets

Conflict free Petri nets have the property that a transition which is once enabled is not disabled until it fires. This property, known as persistence, is what makes the simplified implementation of conflict free nets possible. This property is fully exploited in the circuits presented below.

In the circuits for the conflict free Petri nets, it is assumed that signal propagation delays on the wires connecting the output of one gate to other gates is the same; the delays may be arbitrarily long so long as they are all the same. Two delays can be considered to be the same if the difference between them is small compared to the time a gate takes to react to signals. The time that different gates take to produce output need not be the

same. In fact the delays may change from one operation to another without any harm.

The circuit for a conflict free Petri net is obtained by substituting circuit blocks for individual transitions and places as shown in Figure 2. The details of these circuit blocks are presented in Figure 3. The operation of various blocks used here are somewhat similar to similarly named asynchronous modules defined in the earlier work mentioned [1]. In describing these circuits we will present only the essential details and leave the detailed verification of their operation to the interested reader.

In the circuit block substituted for a place, the blocks IP' and EP' take care of multiple fan in and fan out respectively and the block P' serves the function of the place. Therefore when fan in and fan out are not involved blocks IP' and EP' need not be used. In a similar fashion, in the circuit blocks substituted for transitions, the blocks IT' and ET' take care of the multiple fan in and fan out of arcs and the block TR' performs the function of the transition.

In the operation of the circuits, as a basic rule signals are represented by changes in the levels of wires. It will be helpful to know what signals are represented by particular changes in various points in the circuit. On the wires coming out of the block for a transition, every change in the level represents a new marker. Thus on the input side of the block for a place each change in level again represents a new marker. On the output side of the block for a place, $0 \rightarrow 1$ change in level announces the availability of a marker at the place. Since we are considering the conflict free nets, only one of the transitions acknowledges this signal by a change on the acknowledge wire. Upon receiving the acknowledge signal, the place block brings the level of the outgoing wire from 1 back to 0. To the transition which had sent the acknowledge signal this signal represents a marker while to the other it represents a disable signal to cancel the earlier information about the presence of a marker at that place. When the transition controls some external event, a simple link consisting of a ready and an acknowledge wire goes out to the external world. A signal on the ready link means that the event should proceed to occur and should acknowledge its completion by returning a signal on the acknowledge wire.

The C gate in the IT' block is Miller's C circuit [3]. The C gate can be simply described as a consensus gate which holds on to the most recent consensus. Thus when all inputs become 1 the output becomes 1, and the output remains 1 until all inputs become 0. It is only the consensus of the input that affects the C gate. Thus inputs which change without bringing about a new consensus have no effect on the output of the C gate.

At the C gate in the IT' block, a consensus of 1 is formed when all input places signal the presence of markers. The C gate output then changes to 1 and this changes the state of the first flip flop in the B circuit in TR'. This change provides the acknowledge signal to inform the places that they should bring their outputs back to 0. To the transition which sent the acknowledge signal, the 1 to 0 change in the output of the place represents a marker, and in the case of the other transitions (the loosers) the 1 to 0 change resets the input such that no trace is left of the fact that a marker was available. When all input places of the transition under consideration perform the said change, the C gate acts again and sets its output to 0. The second flip flop in the B circuit in TR' now changes its state and a signal goes out to the outside world as a ready signal so that the associated event may proceed. When the event is completed and an acknowledge signal is returned, the ET' block fans out signals to all output places of the transition. The firing of a transition is thus completed. The treatment of initial markers in the net is quite simple. Each place with an initial marker is provided with an additional input into the IP' block, and to start the action in the net the initial markers are placed by sending signals through these inputs.

From practical considerations it will be interesting to note that the circuit block for a place really consists of just one exclusive OR gate, and that the ET' block is just an interconnection point. Moreover, if the transition does not control an event in the outside world, the ready-acknowledge pair of wires coming out of the TR' block can be shorted out. Note further that if none of the input places of a transition are shared as input places with any other transitions, then one can dispense with blocks P', EP' as well as TR' simplifying the circuit to just one exclusive OR gate at a place and one C gate at the transition. Continuing this further, in the case of Marked Graphs [2],

which are subclasses of Petri nets in which each place is an output place of exactly one transition and an input place of exactly one transition, the circuit reduces to just C gates. In addition either an exclusive OR gate or a NOT gate is required for each place with an initial marker.

Circuits for safe Petri nets

We explained earlier that simple circuits could be constructed for conflict free Petri nets because conflict free nets are persistent in that a transition once enabled is disabled only when it fires. In general Petri nets we do not have this property because of conflicts between transitions. Furthermore, when two transitions are in conflict the circuit must choose which transition is to fire. This need in the circuit is fulfilled by a structure called the conflict structure (Figure 4) which has arbiters to resolve conflicts [1, 4, 5]. The rest of the circuit is the same except the IT and TR blocks are modified to take the lack of persistence into account (Figure 5).

The actions in the straightforward firing of a transition are as follows. The output of the AND gate in the IT" block becomes 1 when all input places signal the presence of markers. A signal then goes to the conflict structure by way of the TR" block. The conflict structure blocks all conflicting transitions and returns an acknowledge signal to this transition. This acknowledge signal, which is represented by a 0 1 change, sets the flip flop in IT" block. A 0 → 1 change thus goes to the B circuit and the B circuit returns an acknowledge signal to instruct all input places to bring back their output levels from 1 to 0. As in the previous case this change in level represents a marker to this transition and a disable signal to the other transitions. When all input places of the transition complete this change, the output of the OR gate in the IT" block changes to 0. This change is then delayed by D on its way to reset the flip flop to allow sufficient time for the disabling of other transitions to be completed. When the flip flop is reset, the level on the request wire to the conflict structure drops to 0 to reset the arbiters, and a similar change in level goes to circuit B. The circuit B then changes the level on the outgoing wire to signal the firing of the transition.

The task of the conflict structure is to resolve conflicts between transitions by choosing one of the transitions and to block the other transitions until the chosen transition has disabled those transitions. The conflict structure does this by means of arbiters. There is one arbiter for each group of mutually conflicting transitions (Figure 4). The arbiter permits only one of the requests sent to it to pass through it. Thus there being at least one arbiter through which the requests from conflicting transitions pass through, only one of them succeeds in its attempt to fire. When a disable signal (bringing back the level of the request wire to 0) is received, the arbiter resets itself and sends out disable signal on the output side if the request had been sent out. Further details about the structure and function of the conflict structure may be found in reference [1], and details about arbiters can be found in references [4, 5].

Multi-input arbiters can be constructed out of elementary arbiters by use of sequence module and an appropriate fan in of elementary arbiters terminated by a sink module (Figure 6). The method of systematically constructing arbiters in this way was first introduced by the author in reference [4]. The arbiter presented here is much simpler in its internal construction primarily because this is not a completely speed independent circuit and thus one does not have to acknowledge all signals involved. The circuit for an elementary arbiter is presented by Plummer in reference [5]. The circuit is modified here to permit a disable signal from being sent regardless of whether the acknowledge signal is received. At the same time it has been possible to simplify the circuit somewhat.

Concluding remarks

It should be noted that the implementation presented here does not use any clock, and therefore it can be said to be a continuous time implementation of Petri nets. Discrete time implementation of Petri nets using a clock to gate actions in circuits is also possible but that kind of implementation has not been the subject of this paper. The implementation presented here is sufficiently simple to be of practical interest. Further simplification of circuits in the direction of requiring only OR gates instead of exclusive OR gates and requiring only C gate in certain transition blocks is possible in many special cases but a systematic method for identifying these situations is yet to be found.

In the circuits, the conflict structure makes decisions regarding which of several conflicting transitions should be allowed to proceed. If one wants to incorporate priority in the circuit, the conflict structure is the place to implement it.

The circuits presented here provide a convenient and methodical way for obtaining circuits for Petri nets, and since Petri nets are a powerful means of expressing the control structures (logic) of digital systems [6], it provides a convenient way for obtaining control circuits for computer systems.

Earlier work on the systematic design of asynchronous control circuits for computers has been done by Muller at the University of Illinois [3, 7] and Clark's group at Washington University, St. Louis [8]. Work on this line is also in progress at Carnegie Mellon University (Bell and Grason [9]).

1. Patil, Suhas S., Coordination of Asynchronous Events. MAC-TR-72, Project MAC, M.I.T., Cambridge, Massachusetts, June 1970.

2. Holt, A. W., and F. Commoner, Events and Conditions (in three parts). Applied Data Research, New York 1970. [A portion of these appears in Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York 1970, pp 3-52.]

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

4. Patil, Suhas S., n-server m-user Arbiter. Computation Structures Group Memo 42, Project MAC, M.I.T., Cambridge, Massachusetts, May 1969.

5. Plummer, William W., Asynchronous Arbiters. Computation Structures Group Memo 56, Project MAC, M.I.T., Cambridge, Massachusetts, February 1971.

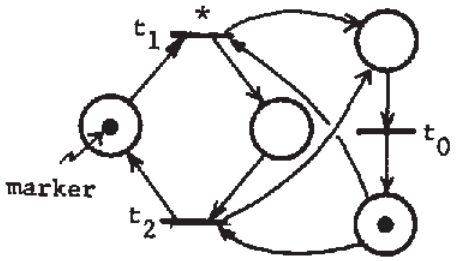
6. Dennis, Jack B., Modular, asynchronous control structures for a high performance processor. Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York 1970, pp 55-80.

7. Kimura, I. Some topics related to the logical circuits of the new Illinois computer. Denki Tsushin Gakkai Zasshi, Vol. 46, No. 11 (November 1963), pp 69-75. English translation: Electronics and Communications in Japan, Vol. 46, No. 11 (IEEE 1964), pp 92-99.

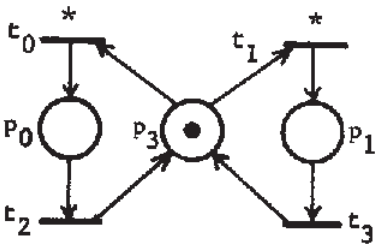
8. Clark, W. A., Macromodular computer systems. AFIPS Conference Proceedings, Vol. 30 (Spring 1967), pp 335-336.

9. Bell, C. G., and J. Grason, The register transfer module design concept. Computer Design, May 1971, pp 87-94.

* enabled transitions



a) A conflict free Petri net.



t_0 and t_1 are in conflict

b) A Petri net with a conflict.

Figure 1

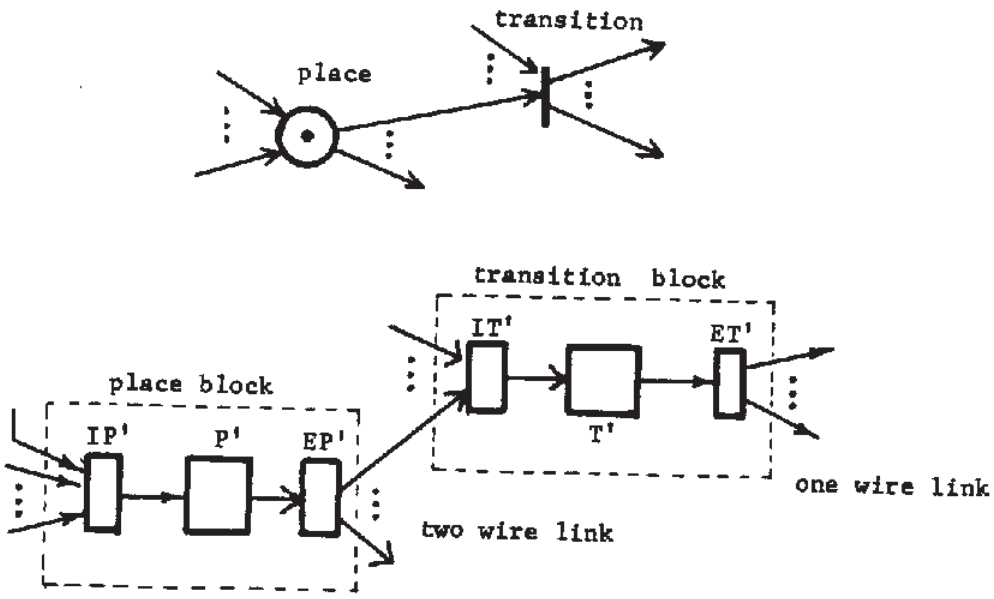
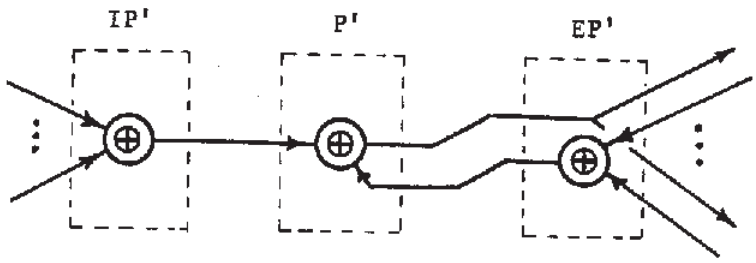
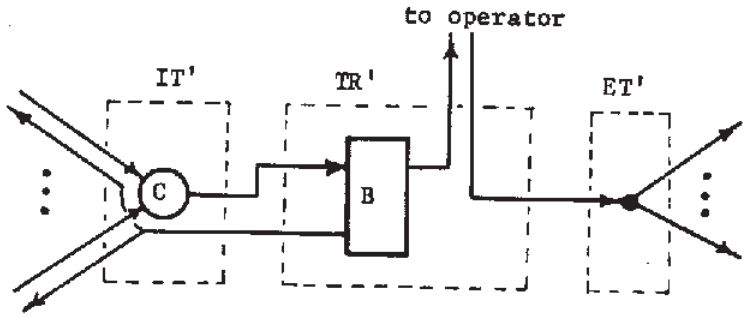


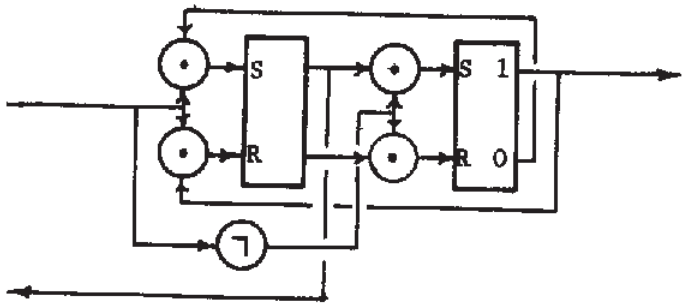
Figure 2. Circuit blocks for conflict free nets.



place block



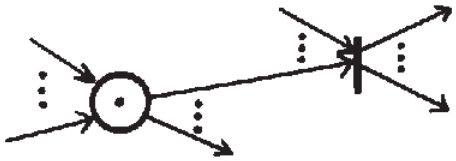
transition block



circuit B

Figure 3. Details of the circuit blocks.

a) Petri net



b)Corresponding circuit

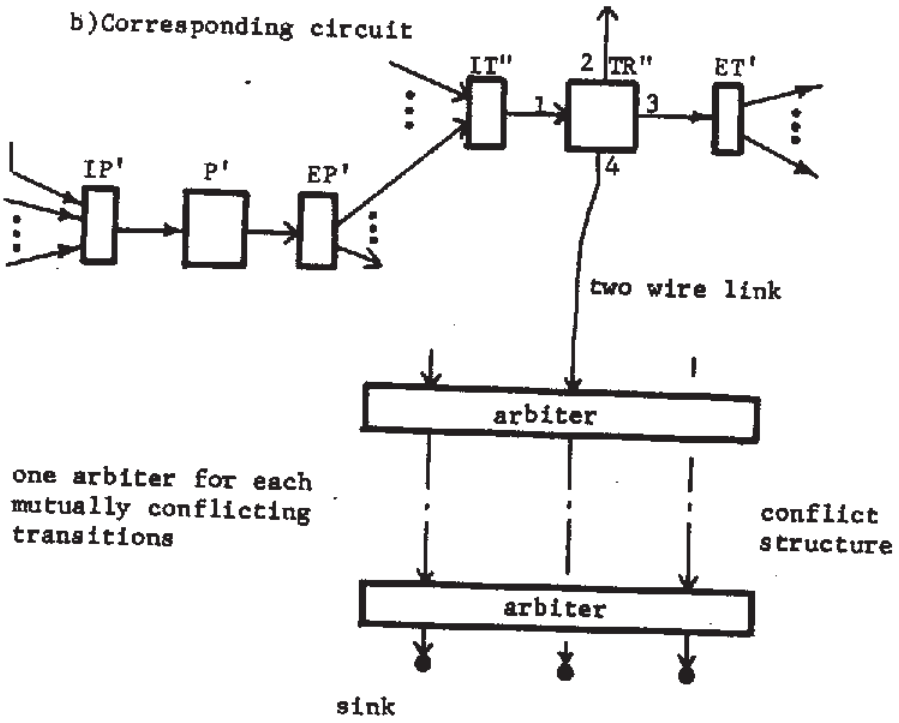


Figure 4 Circuit block for general Petri nets

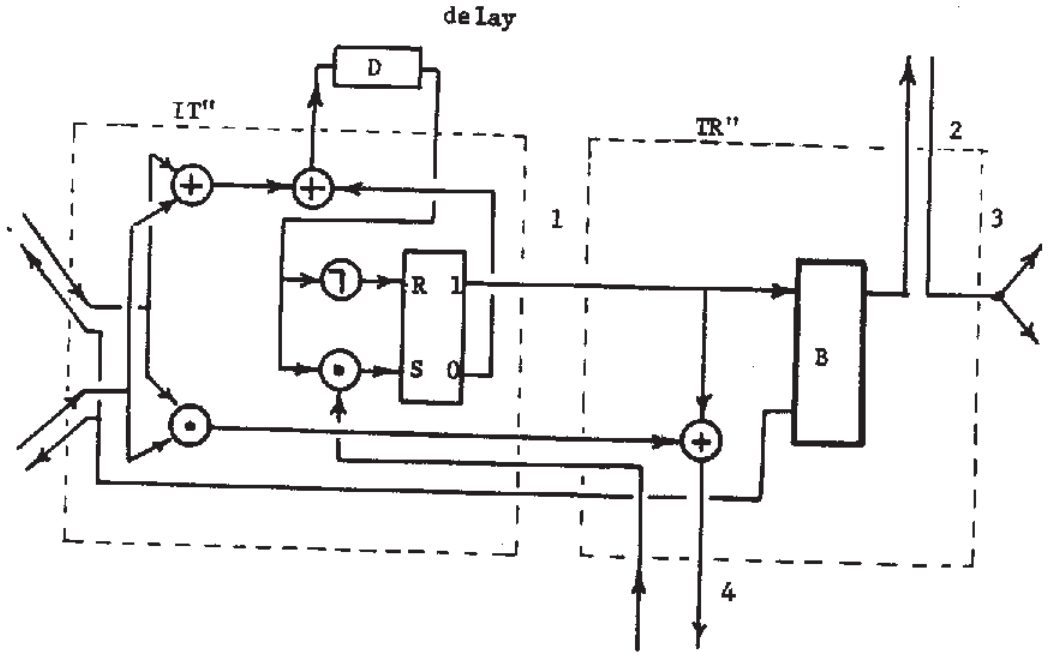


Figure 5. Transition block for general Petri nets.

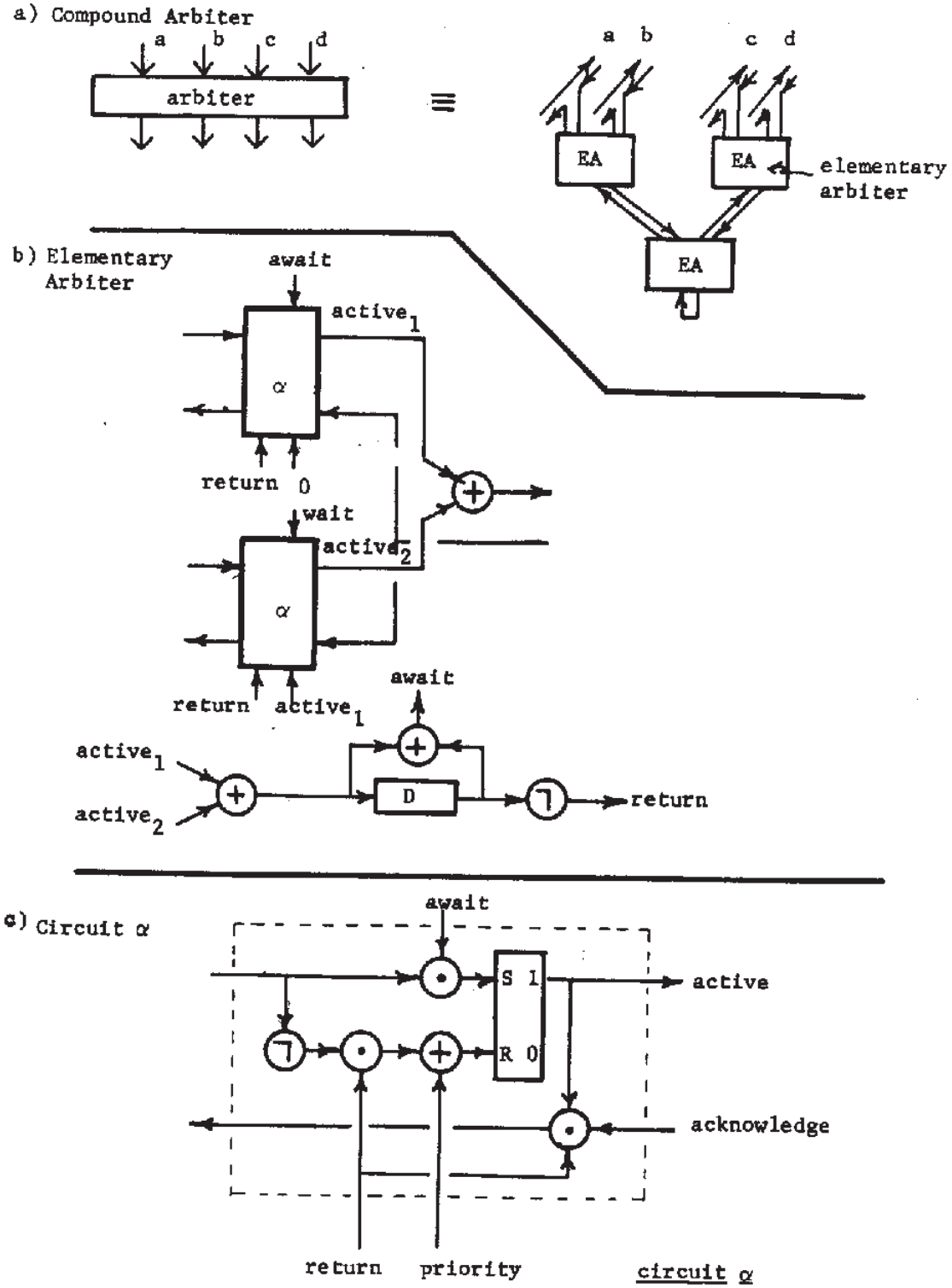


Figure 6. A circuit for the arbiter.