MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo No. 54

Speed Independent Asynchronous Circuits

Jack B. Dennis and Suhas S. Patil

January 1971

A paper presented at the Fourth Hawaii International Conference on System Sciences, January 12-14, 1971. Published in the Proceedings of the Fourth Hawaii International Conference on System Sciences, (Western Periodicals Company, North Hollywood, California 1971), pp 55 - 58.

SPEED INDEPENDENT ASYNCHRONOUS CIRCUITS

Jack B. Dennis and Suhas S. Patil
Project MAC, MIT
Cambridge, Massachusetts

INTRODUCTION

The concept of speed independent switching circuits was first studied seriously by Professor David Muller of the University of Illinois more than a decade ago [7]. The subject concerns switching circuits whose correct operation is unaffected by variations in the speed of operation of the switching devices making up the circuit. Variations in switching speed may result from variations in conditions of manufacture, differences in lead lengths, and environmental factors such as temperature and power supply voltage. Speed independent design makes it possible to guarantee correct operation of a circuit at speeds limited only by the actual delays present in the physical devices of the circuit.

When the notion of speed independent switching circuits was introduced, logic designers felt with considerable justification, that the greater number of active elements required by speed independent designs rendered the concept impractical. Since then, many changes have taken place in technology including the advent of integrated circuits and speed independent design is now sufficiently attractive to merit further serious study.

The work reported here forms one aspect of current research on asynchronous systems in the Computation Structures Group at Project MAC. Our ideas derive from the publications of Muller and his colleagues [5, 6, 7, 8] and the recent work of Anatol Holt [4] on the semantics of concurrent systems. Other aspects of our research may be seen in reports by Patil [9] and Dennis [3]. Altman, Bruno and Denning [1, 2] have also reported on research that is directly related to ours.

CIRCUITS

The model we use for switching circuits is essentially the one used by Muller [7, 8]. A net is an interconnection of finitely many operators: Each operator models one gate or switching element, and has one or more input connections and one or more output connections. Each output connects to a unique node of the net. Each input connection to an operator comes from a distinct node of the network. The nodes of a network take on the values 0 or 1 during its operation. The value at a node is determined by the associated operator in response to signals (changes in value) at its input nodes.

The behavior of an operator is conveniently specified by means of a form of Karnaugh map called a transition diagram. Transition diagrams for operators that model several important types of switching elements are given in Figure 1. Each cell of a transition diagram represents a total state of the operator — that is, a combination of input values and value at the output node. For certain total states an operator is active, and the

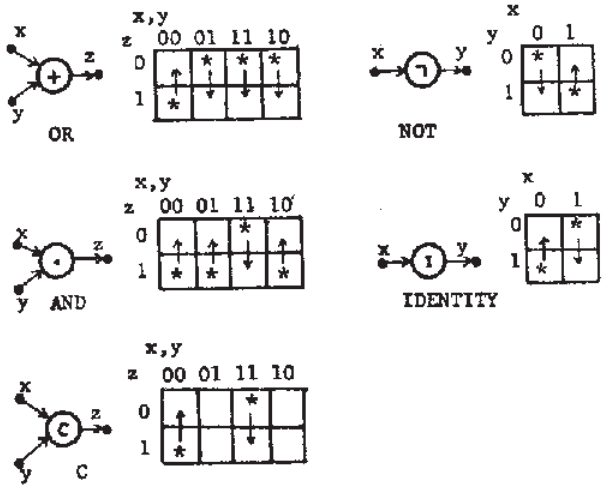


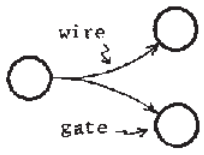
Figure 1

corresponding cells in the map are marked with asterisks. An operator that is active may fire — changing the value at its output node from 0 to 1 or from 1 to 0. The transition of an operator to a new stable state is indicated in the map. A state of a net is a set of binary values for the nodes of the net. In any given state of the net, certain operators are active. Any of these operators may fire yielding a new state.

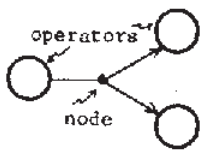
There are two alternate assumptions of interest concerning the insensitivity of switching circuits to delays. We may ask that a circuit operate correctly even if arbitrary finite delays are inserted in any of its wires. Circuits designed to behave correctly under this very strict requirement will be called type 1 switching circuits. Alternatively, we may assume that the delays on wires are negligible and require that a circuit behave correctly regardless of variations in switching speed of its gates. Circuits designed in accordance with this less restrictive requirement will be called type 2 circuits.

The manner in which type 1 and type 2 circuits are modelled by nets is shown in Figure 2. The relation between a type 2 circuit and the net that models it is direct (Figure 2b). Each wire that joins some gate output to the inputs of other gate gates is represented by a node in the net. In the case of a type 1 circuit, we introduce identity operators (Figure 2c) that model the unknown delays in transmission of signals on wires.

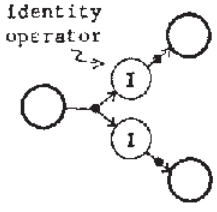
Ideally, one would like to realize switching circuits as type 1 interconnections of standard switching elements. However, we shall see that the circuits that may be synthesized by a type 1 interconnection of AND, OR, and NOT gates are so limited as to be uninteresting. Thus the usual set of primitive gates does not provide an adequate basis



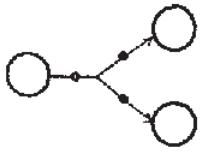
(a) interconnection of switching elements



(b) model for a type 2 connection



(c) model in for a type 1 connection



(d) simplified drawing for type 1 model

Figure 2

for the synthesis of useful type 1 circuits. We may ask whether there exists a small collection of building blocks that permits the synthesis of arbitrary type 1 switching circuits. Since such building blocks would have many instances in a large circuit, and since the electronic components within such units would be in a fixed local pattern it is reasonable to design the building blocks as type 2 interconnections of conventional gates. We have found that a large class of speed independent circuits may be synthesized according to this scheme, using a small set of primitive switching elements. Nevertheless, we do not as yet know a finite set of primitive elements that form a universal set for building asynchronous systems.

The AND, OR and NOT gates clearly do not constitute a satisfactory basis for type 1 circuits as a very important operator, the C-element of Muller (Figure 1), cannot be synthesized as a type 1 circuit of AND, OR and NOT elements. A circuit for the C-element must have some internal feedback and there may be arbitrary delays in the feedback path if the circuit is type 1. Thus if the input changes before the feedback has settled, the circuit will behave contrary to its specification. Nevertheless the C-element can be synthesized as a type 2 circuit using AND and OR gates [5].

PERSISTENT AND LIVE CIRCUITS

Now consider what happens if we continue to fire active operators in a net which models a circuit. The activity of the net either comes to a halt at a state where no operator is active, or the action of firing operators continues forever. In the latter case, as the circuits and therefore the nets are finite, some states must repeat. In this case, there must be a set of states such that it is possible to go from any one of them to any other by means of appropriate firing sequences. A maximal set of states such that the circuit can go from any one of them to any other will be called a state class. State classes are of interest to us because we view the activity of the circuits as an ongoing activity continuing from the distant past and going into the future ad infinitum. A net together with a state class will be called a system and represents the behavior of a circuit. A state class which is closed in that no firing sequence can take the net into a state not in the class will be called a terminal state class. A system in a terminal class continues to operate in that state class.

The notion of liveness of circuits and systems is closely related to the ideas expressed above. An operator is live if any firing sequence for the system can be extended to include a new firing of the operator. A system (i.e., a net operating in a state class) is live if every operator in the system is live. Liveness of a system means that at no point in its operation is any operator of the system rendered inoperative.

Next we introduce a property of systems called persistence. An operator in a system is persistent if

after becoming active, it cannot lose its active status as a result of any number of firings of other operators in the system. Thus a persistent operator ceases to be active only as a result of its own firing. Persistence of switching elements in circuits is defined similarly. A circuit is persistent if each one of its elements is persistent, and similarly a system is persistent if each one of its operators is persistent. It is important to note that for a type 1 circuit to be persistent, it is only necessary for the circuit operators to be persistent in the system representing the circuit; lack of persistence on the part of the interconnection operators does not matter (see Figure 3). The term "persistent" has been adopted from the work of Miller and Karp.

Persistent systems (nets) not involving generators (introduced later in this paper) correspond to Muller's semi-modular circuits. Persistence is a convenient way for achieving speed independence, but persistence is not a requirement for speed independence.

We will now focus our attention on the kind of live and persistent circuits that can be had as type 1 and type 2 circuits. We have obtained an important negative result which states that a type 1 circuit constructed out of AND, OR, NOT and IDENTITY elements is live and persistent if and only if the circuit has an odd number of NOT elements and every closed path (a circuit in the graph theoretic sense) in the circuit passes through every element of the circuit. A key fact in proving this is that, in a live and persistent type 1 circuit, a firing sequence which brings the circuit back to its initial state must fire every element of the circuit.

From the negative result stated above, the only live and persistent type 1 circuits which can be built out of AND, OR, NOT and IDENTITY elements are of the type shown in Figure 3. Therefore, we shall include the C-element with these elements as a basic switching element. With the C-element included, we can obtain type 1 circuits for an important class of systems called marked graphs (Holt [4]). We give a brief introduction to marked graphs below.

Marked graphs are directed graphs consisting of nodes joined by arcs. Some objects called markers are associated with arcs. When each of the arcs incident on a node has a marker, the node may fire, picking a marker from each of those arcs and placing a marker on each of the emergent arcs. The action proceeds as arcs lose and gain markers by the firing of nodes. Marked graphs in which no arc has more than one marker on it at any time are called safe marked graphs. Marked graphs constitute an important class of systems and are very useful in specifying certain kinds of control schemes for asynchronous computers.

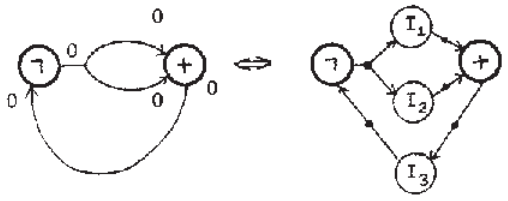
A type 1 circuit for a safe marked graph can be obtained by directly substituting C-elements for nodes, wires for arcs and NOT elements for markers. In this construction we permit C-elements with multiple inputs which are required for nodes with multiple incident arcs. Since the C-element can be constructed as a type 2 circuit using AND, OR and NOT elements, type 2 circuits for marked graphs can be constructed using AND, OR and NOT gates alone.

In terms of asynchronous systems, we are interested in broader classes of systems than those characterized by marked graphs. We are currently investigating appropriate building blocks for speed independent synthesis of several enclosing classes of systems.

INTERCONNECTION OF CIRCUITS

Circuits of the type we discussed so far are closed circuits in that they do not communicate with the outside. Yet for circuits to be useful for processing information, they must be able to exchange signals with the outside. In Muller's work communication with a circuit was arranged by removing either a NOT or an IDENTITY element from the circuit; the two wires so obtained formed a link of communication. These links operate with only one type of signal, and therefore cannot convey information bits (0 or 1). We will now introduce operators called generator and absorber to permit us to send information to and from circuits.

When a generator (Figure 4a) is in its initial condition in which all input and output wires have value 0, the operator changes one of the output wires to 1. Which of the output wires will become 1 is not known a priori. A generator thus represents a source of information. After having



A persistent type 1 circuit

operators I_1 and I_2 are not persistent

Figure 3

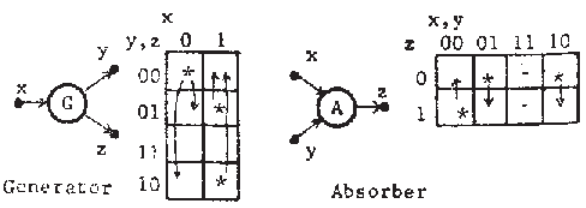


Figure 4

generated a bit in this way, the operator returns its output to 0 after the input has changed to 1.

The absorber (Figure 4b) changes its output to 1 when either of its inputs is changed to 1 (both inputs of this operator must not be 1 at the same time). When its input is restored to 0, the operator returns its output to 0. An absorber models an element which absorbs information. In a circuit an OR gate can serve as an absorber.

Generators and absorbers provide a way for connecting circuits just as NOT and IDENTITY elements do in Muller's work. To connect two circuits, a generator is removed from one circuit and an absorber is removed from the other and the links thus obtained are connected together. Just as the circuit obtained by interconnecting two live and persistent circuits by removing NOT and IDENTITY elements is live and persistent [6], the circuit obtained by interconnecting two live and persistent circuits by removing a generator from one and an absorber from the other is also live and persistent. Needless to say, the two circuits must be brought to appropriate states before being joined together. (Note that this interconnection property holds only if the two circuits are separate.)

The interconnection property stated above provides a way for constructing a live and persistent circuit from smaller live and persistent circuits. The asynchronous counter presented below illustrates these ideas.

THE COUNTER EXAMPLE

To conclude, we discuss an application of speed independent logic design which offers a number of advantages over conventional designs. In Figure 5 the counter C must record, at the highest possible rate, the number of signals received at the input connection 'count'. Moreover, it must be possible to sample the number of signals recorded at arbitrary times, placing this number in register R each time a signal arrives at the input connection 'sample'.

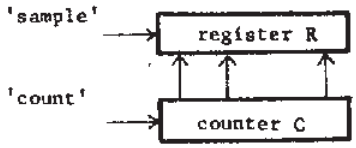


Figure 5

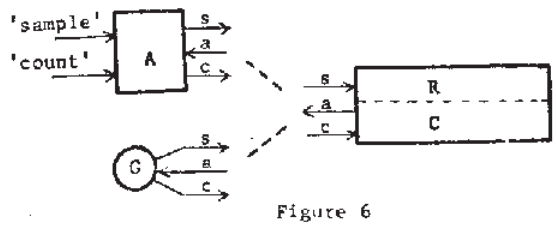


Figure 6

Using speed independent design an elegant implementation of this sytem is possible. It is capable of very fast counting, only the first stages need be constructed of fast devices, and it can accept 'sample' signals separated by only one 'count' signal. The system required is "nondeterminate" because the succession of numbers appearing in register R depends on the order in which signals arrive at the 'count' and 'sample' connections. As shown in Figure 6, we can divide the system into two parts joined by a three-wire link. The portion designated A permits a $0 \rightarrow 1$ signal to occur on just one of the wires c (count) or s (sample); an acknowledge signal on wire a and a reset cycle must follow before a subsequent $0 \rightarrow 1$ signal is sent. To the remainder of the system, A appears as a generator element. It performs the "synchronizing" function of deciding whether a 'count' signal or a 'sample' signal is to be acted on first when the two signals arrive nearly simultaneously.

The main portion of the counter is determinate—that is, the succession of numbers appearing in R is determined by the sequence of signals sent over the three-wire link (and the initial condition of the counter).

To further develop the design, we look for a solution in which the stages of C and R are repetitions of the same circuit as shown in Figure 7. For each pair of stages, the left stage acts as a generator and the right stage acts as an absorber. The two possible signal cycles will be called c-cycles and s-cycles. In each stage, one c-cycle occurs on the right hand link for each pair of c-cycles on the left hand link. Each stage C_i of the counter remembers one bit of the binary representation of the count, and must complement its state for each c-cycle on the left-hand link. An s-cycle on the left link causes a corresponding s-cycle on the right link, and causes the state of the counter stage to be placed in the associated register stage R_i. Cycles of action flow from left to right through the stages of the counter, each c- or s-cycle on a link being held up just until the

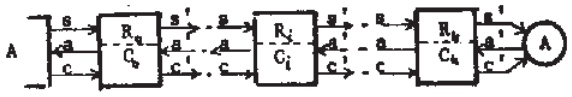


Figure 7

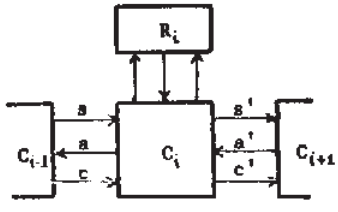


Figure 8

following stage acknowledges completion of its action for the preceding c- or s-cycle.

Each stage takes the form shown in Figure 8 in which communication between C_i and R_i is accomplished over a third three-wire link in the obvious manner. The action taken by C_i in response to an input c- or s-cycle depends on the bit value remembered by C_i:

Case	Input Cycle (left)	Remembered Bit		Register Cycle	Output Cycle (right)
		old	new		
1	C	0	1	none	none
2	C	1	0	none	C
3	S	0	0	0	S
4	S	1	1	1	S

Since the remembered bit must be examined in each case, the stored bit must traverse a cyclic path in the circuit for C_i. The circuit for C_i is shown in Figure 9. For simplicity, OR-elements with two to four input wires are usually indicated by vertical lines intersecting the input wires at points marked by an arrowhead and a plus sign. The circuit is a type 2 interconnection of OR, NOT, and C-elements, the type 2 connections occurring inside the dashed boxes. The initial condition of the circuit is with the output of each C-element at 0 except for the lower left C-element which has its output at 1 initially. This is indicated by the black dot. In this condition, the circuit is remembering 0 and is ready to respond to a c- or s-signal from the left link. The remembered bit is represented by the condition of the two wires labelled k and $\bar{k}$. The applicable case in the above table is recognized by the four similarly labelled C-elements in Figure 9. OR elements react to their outputs to initiate the actions specified in the table and send an acknowledge signal over the left link. The C-element in the right part of the figure acts just when all specified actions have been completed. The second group of four C-elements retains the value of the remembered bit. (Three C-elements are necessary in any information cycle of a circuit for operation without hang-up.)

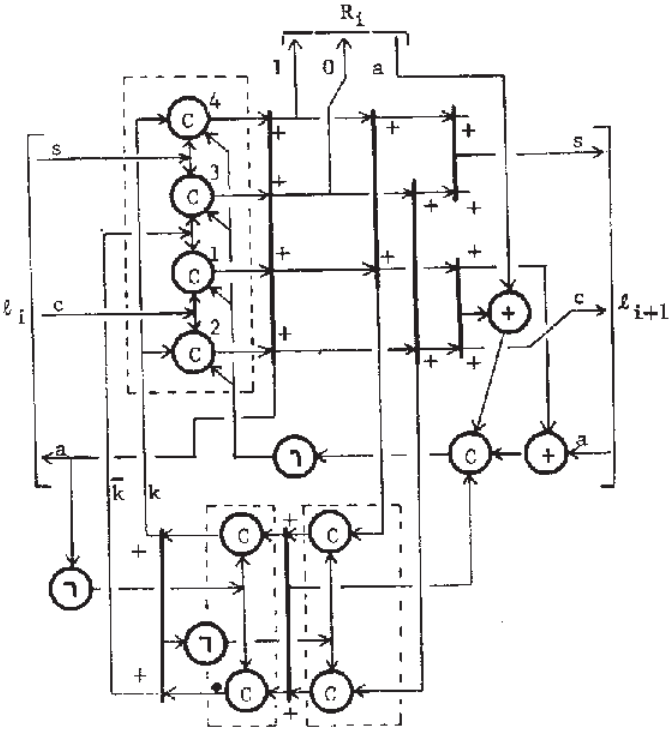


Figure 9

REFERENCES

1. Altman, S. M. and P. J. Denning. Decompositions of control networks. Conference Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, Association for Computing Machinery, New York 1970.

2. Bruno, J. and S. Altman. Asynchronous control networks. IEEE Conference Record of 1969 Tenth Annual Symposium on Switching and Automata Theory, October 1969, pp 61-73.

3. Dennis, J. B. Modular asynchronous control structures for a high performance computer. Conference Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, Association for Computing Machinery, New York, 1970.

4. Holt, A. W. and F. Commoner, Events and conditions. Conference Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, Association for Computing Machinery, New York 1970.

5. 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.

6. Miller, R. E. Switching Theory, Vol. II, John Wiley and Sons 1965, pp 220-226.

7. Muller, D. E. and W. S. Bartky. A theory of asynchronous circuits. Proceedings of an International Symposium on the Theory of Switching. The Annals of the Computation Laboratory of Harvard University, Vol. 29, Part I, Harvard University Press, Cambridge 1959, pp 204-243.

8. Muller, D. E. Asynchronous logics and application to information processing. Switching Theory in Space Technology, Stanford University Press 1963, pp 289-297.

9. Patil, S. S. Coordination of Asynchronous Events. Doctoral thesis, Department of Electrical Engineering, M.I.T., June 1970

This work was done at Project MAC, MIT, with support from the Advanced Research Projects Agency, DOD, under NRL Contract Nonr N00014-70-A-0362-0001.