

STORED STATE ASYNCHRONOUS SEQUENTIAL CIRCUITS

ALAN B. HAYES

STORED STATE ASYNCHRONOUS SEQUENTIAL CIRCUITS

by

Alan B. Hayes

Abstract—A method is described for realizing asynchronous sequential circuits in a manner analogous to the state method for synchronous sequential circuits. The method simplifies the process of constructing asynchronous sequential circuits, allows utilization of existing MSI parts, and avoids the necessity for concern with races or hazards.

Index Terms—Asynchronous sequential circuits, Muller circuits, self-timed systems, stored state sequential circuits.

May 1980

STORED STATE ASYNCHRONOUS SEQUENTIAL CIRCUITS

ALAN B. HAYES

Abstract-A method is described for realizing asynchronous sequential circuits in a manner analogous to the stored state method for synchronous sequential circuits. The method simplifies the process of constructing asynchronous sequential circuits, allows utilization of existing MSI parts, and avoids the necessity for concern with races or hazards.

Index Terms-Asynchronous sequential circuits, Muller circuits, self-timed systems, stored state sequential circuits.

1. Introduction

Although the classical method [1, 2, 3] for realizing asynchronous sequential circuits is well developed, it requires the expenditure of a substantial amount of effort to find state assignments that produce hazard-free and race-free realizations. It also yields realizations that use large quantities of random gate logic, resulting in high part count with SSI circuits and little if any regularity. The circuit complexity of conventionally realized asynchronous state machines increases very rapidly as the number of states and inputs to the state machine goes up, making the direct realization of reliable machines with more than a relatively small number of states and inputs an intractable problem. While it is possible to decompose a single machine into a collection of smaller and more easily realizable machines [3, 4], this approach will typically increase the part count and may substantially slow the operation of the machine.

The introduction of Field Programmable ROM's simplified the process of realizing synchronous sequential circuits by allowing the development of stored state techniques [5, 6]. In this article a method is described for applying stored state techniques to asynchronous sequential circuits in an environment where all elements are Muller circuits or self-timed systems [7, 8]. In what follows, the terminology and conventions used for self-timed systems are compatible with those used in [8], to which the reader is referred for background and additional detail.

2. Signaling Protocol

Because an asynchronous system lacks a clock that can be used to delimit data transmissions, the signaling protocol used must somehow incorporate a means to indicate the limits of a transmission. To assure that the data transmitted from one element of the system to another is actually received, some form of closed loop or "handshaking" protocol is required.

A pulse, regardless of its duration, can be too fast for some element in a system, therefore transitions must be used in asynchronous signaling

conventions. The minimum number of transitions required is two; one to mark the initiation of the operation (generally called REQUEST), and another to mark the termination of the operation (generally called ACKNOWLEDGE). Protocols that use this minimum number of transitions per operation are called 2-cycle or nonreturn-to-zero (NRZ) signaling schemes.

An alternative to 2-cycle signaling is 4-cycle or return-to-zero (RZ) signaling, which uses two transitions each on the REQUEST and ACKNOWLEDGE lines per operation. 4-cycle signaling was first invented by Muller [7] and is used in many of his examples of speed independent circuits. Both 2-cycle and 4-cycle signaling can be realized in single-rail (1 wire) form, which is sensitive to the relative delays of the data paths and the associated REQUEST or ACKNOWLEDGE path; or in a delay-insensitive, double-rail (2 wire) form where the REQUEST or ACKNOWLEDGE signal is embedded in each bit of the associated data bus.

The 4-cycle, single rail signaling protocol shown in Figure 2-1 is used in the circuits described here. 2-cycle signaling could also be used, although the resulting circuits would typically be more complex. Seitz [8] has described conversion elements between single and double rail coding for applications where delay-insensitive communication between modules is required.

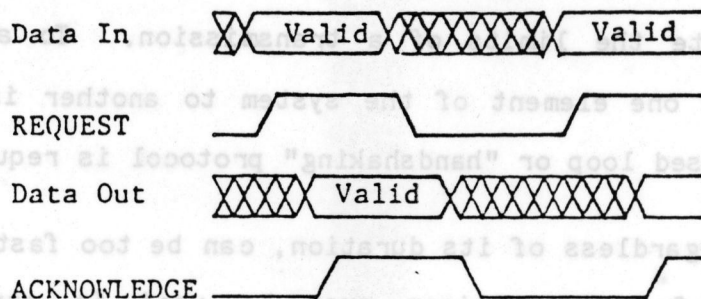
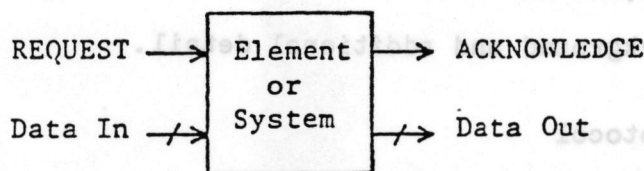


Figure 2-1: Four cycle signaling.

As shown in Figure 2-1, the link, which is defined as the REQUEST and ACKNOWLEDGE lines plus the input and output data busses, starts in the idle state with both REQUEST and ACKNOWLEDGE low. Input data to the element is required to be valid prior to the REQUEST going high and must remain valid until the ACKNOWLEDGE from the element goes high. Once the REQUEST has gone high, it must remain high until the ACKNOWLEDGE goes high. The REQUEST must be high and output data from the element must be valid prior to the ACKNOWLEDGE going high. The ACKNOWLEDGE must remain high and the output data must remain valid until the REQUEST goes low. The ACKNOWLEDGE must go low, returning the link to the idle state, before the REQUEST may again go high.

3. Construction Rules

The asynchronous sequential circuit (the machine) being described here will be an element of a larger system. It will be linked to other elements of the system by the REQUESTS, ACKNOWLEDGES, and data busses that are its inputs and outputs.

- In the initial state, all REQUESTS and ACKNOWLEDGES in the input and output set of the machine must be low.

The machine starts from a state in which all the communication links between itself and the rest of the system are idle.

- The input set must contain at least one REQUEST.

This rule guarantees that the machine can start.

- If the input set contains more than one REQUEST, then external arbitration must be used to preclude concurrent REQUESTS.
- The output set must contain the associated ACKNOWLEDGE to every REQUEST in the input set.
- The input set may contain any number of ACKNOWLEDGES.
- The output set must contain the associated REQUEST to every ACKNOWLEDGE in the input set.

The machine is only capable of single-input-change operation. These rules and the signalling protocol are intended to insure that the machine will not receive multiple-input-changes.

Since the machine has no control over the occurrence of the initial low-to-high transition on REQUESTS in its input set, external arbitration must be used to guarantee that only one of these links will be active at a time. The signalling protocol requires that a REQUEST that has gone high remain high until the associated ACKNOWLEDGE goes high. This and the rule that the associated ACKNOWLEDGE must be in the output set of the machine, allows the machine to preclude another transition on the REQUEST line until it is ready for that transition. Once the REQUEST goes low, the machine is similarly able to block the next transition on that or any other REQUEST line in the input set by holding the associated ACKNOWLEDGE high.

The rule that the associated REQUEST to every ACKNOWLEDGE in the input set of the machine must be in its output set, gives the machine the ability to control the number of REQUEST and ACKNOWLEDGE lines in its input set on which transitions may occur at any given time.

- On each state change, a transition will occur on one and only one of the REQUEST and ACKNOWLEDGE lines in the output set.
- There will be a one-to-one correspondence between transitions on the REQUEST and ACKNOWLEDGE lines in the input set and state change.

The limitation of output set REQUEST and ACKNOWLEDGE transitions to at most one per state change assures that the number of input set REQUEST and ACKNOWLEDGE lines on which a transition is possible will not exceed one. The requirement of at least one output set REQUEST and ACKNOWLEDGE transition per state change and the requirement for exactly one state change for each input REQUEST and ACKNOWLEDGE transition guarantees that the machine, once started, will continue to run, at least until returning to the initial state or some other state in which all of its links are idle. The last rule means that the machine operates in single-output-change mode.

Figure 3-1 shows a typical stored state table realization of a synchronous sequential circuit. The PROM contains the state table and generates from the current state and inputs, a next state and outputs which are the inputs to the register. State changes occur periodically and

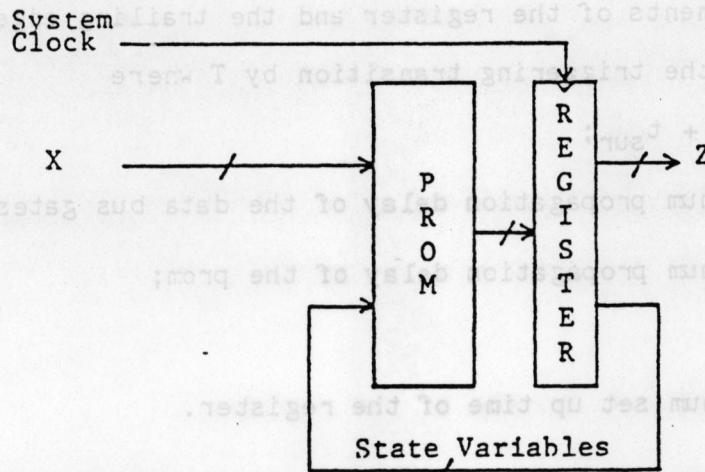


Figure 3-1: A typical stored state synchronous sequential circuit.

synchronously with the system clock.

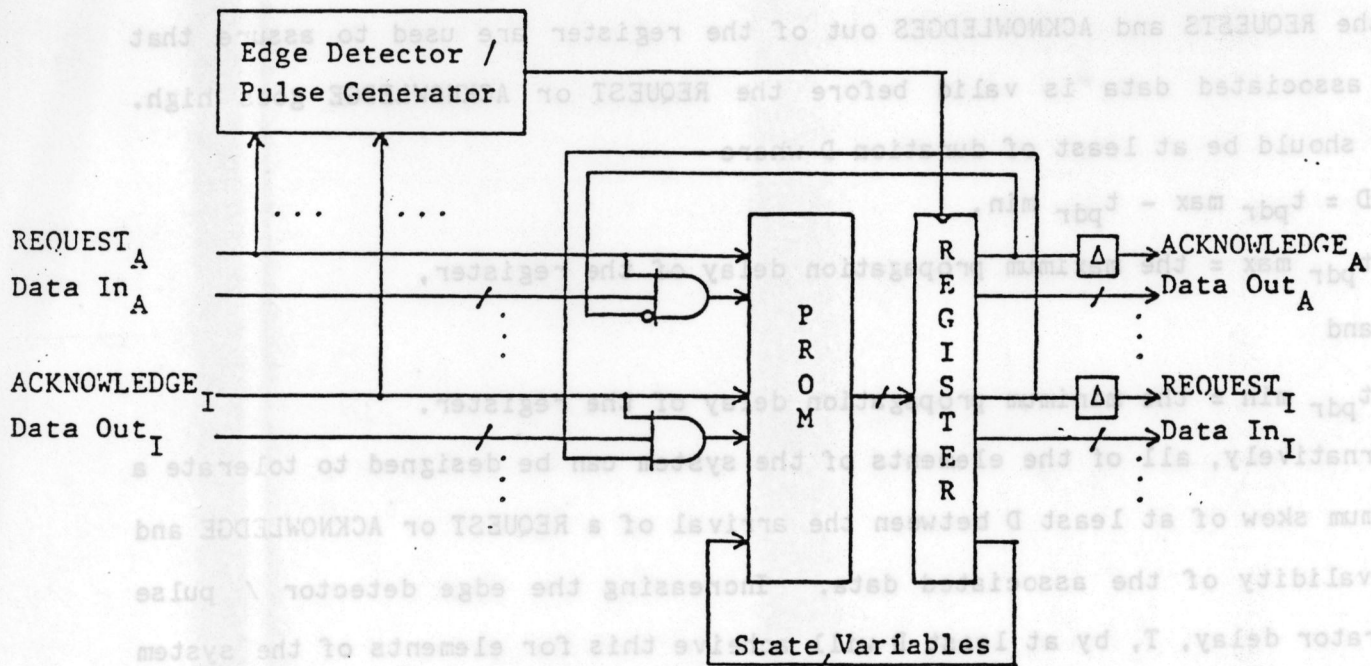


Figure 3-2: Asynchronous sequential circuit realization.

With the rules and signaling protocol outlined above, a race-free and hazard-free asynchronous sequential circuit may be realized as shown in figure 3-2. As in the above synchronous example, the PROM holds the state table of the machine and generates a next state and outputs which are the input to the register. State changes occur when the edge detector / pulse generator produces a pulse, which it does in response to each transition on one of the input REQUEST or ACKNOWLEDGE lines. The pulse must be of sufficient width to

satisfy the requirements of the register and the trailing edge of the pulse is to be delayed from the triggering transition by T where

$$T = t_{pdg} + t_{pdp} + t_{sur};$$

t_{pdg} = the maximum propagation delay of the data bus gates;

t_{pdp} = the maximum propagation delay of the prom;

and

t_{sur} = the minimum set up time of the register.

The gates on the data buses are required to assure that transitions on these lines outside the periods for which their stability is guaranteed can not affect state transitions which may be occurring concurrently. The delays on the REQUESTS and ACKNOWLEDGES out of the register are used to assure that the associated data is valid before the REQUEST or ACKNOWLEDGE goes high. They should be at least of duration D where

$$D = t_{pdr \max} - t_{pdr \min},$$

$t_{pdr \max}$ = the maximum propagation delay of the register,

and

$t_{pdr \min}$ = the minimum propagation delay of the register.

Alternatively, all of the elements of the system can be designed to tolerate a maximum skew of at least D between the arrival of a REQUEST or ACKNOWLEDGE and the validity of the associated data. Increasing the edge detector / pulse generator delay, T , by at least D will achieve this for elements of the system that are stored state asynchronous sequential circuits. A realization for the edge detector pulse generator is shown in figure 3-3.

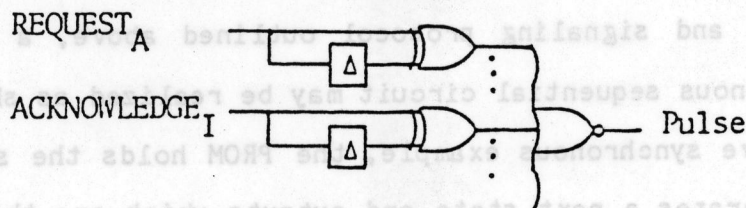


Figure 3-3: Edge Detector / Pulse Generator realization.

While this realization of the asynchronous sequential circuit is the most

straight forward, it has the disadvantage of using storage rather inefficiently. For an N state machine with I inputs and O outputs, a PROM is required of size

$$2^{(\lceil \log_2 N \rceil + I)} \text{ words}$$

by

$$\lceil \log_2 N \rceil + O \text{ bits}$$

where

$\lceil x \rceil$ = the least integer equal to or greater than x .

In most cases, the next state decision depends only on a small subset of the M inputs to the machine. In cases where P , the maximum size subset of inputs required to determine the next state, is significantly less than I , it is possible to reduce substantially the size of the PROM.

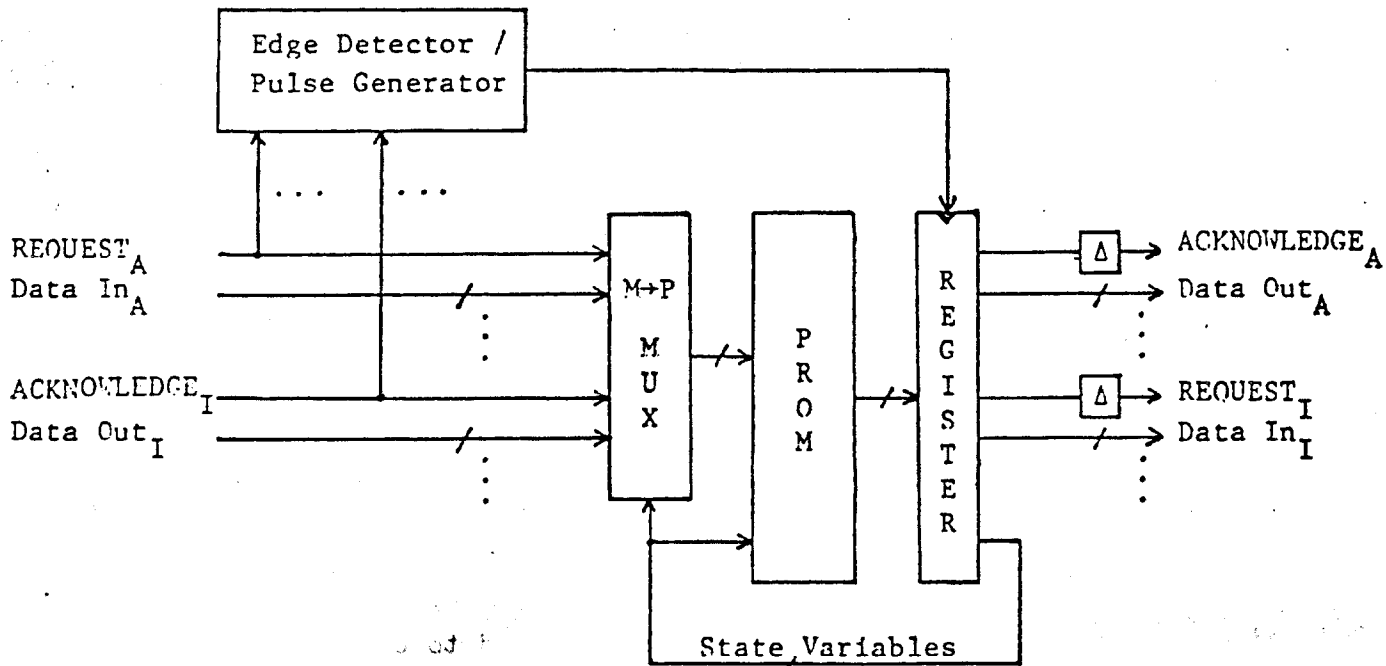


Figure 3-4: Stored state asynchronous sequential circuit with state-selected input set.

Figure 3-4 shows one way this reduction may be accomplished. Here, a subset of the input set is selected as a function of the current state of the machine. The resulting PROM size requirement is

$2^{(\lceil \log_2 N \rceil + P)}$ words

by

$\lceil \log_2 N \rceil + O$ bits.

This realization has the additional advantage that, in most cases, the gates on the data busses are no longer necessary as these lines will generally only be selected as PROM inputs during the periods for which their stability is guaranteed. Note that the delay T is now given by

$$T = \text{MAX}[t_{\text{pdm}}, t_{\text{selm}}] + t_{\text{pdp}} + t_{\text{sur}};$$

where

t_{pdm} = the propagation delay through the multiplexor;

and

t_{selm} = the select time of the multiplexor.

4. Conclusions

A method has been described for applying stored state techniques to the realization of asynchronous sequential circuits which operate in single-input-change and single-output-change mode. The signaling protocol used between the asynchronous sequential circuit and other elements of the system of which it is a part has been described, construction rules have been specified and block diagrams for two alternate realizations given.

Advantages of this method include:

- Ease of realization. Little effort is required beyond specifying the state table for the machine.
- Freedom from concern with races and hazards.
- Reduced part count and circuit complexity as compared to classically realized asynchronous sequential circuits.
- Fast debugging. This is partly due to the reduced part count and partly to the fact that a pulse is generated to cause each state transition. This makes the use of a clocked logic analyzer possible.
- Simple modification. In cases where the modifications do not require the addition of new inputs and where the change in the number of states does not exceed the PROM capacity, all that is

necessary to modify the state machine is to reprogram the PROM.

Disadvantages include:

- Slower circuits. The extra levels of logic needed to realize asynchronous sequential circuits by the stored state method means that they will be slower than their classical equivalents when both are realized in the same technology and at the same level of integration (VLSI NMOS for example).

This technique has been used over the past two years by a number of designers on the Burroughs sponsored Data Driven Computation Project at the University of Utah. DDM2, the current version of our data driven machine, incorporates several hundred stored logic asynchronous sequential circuits, the largest of which has 260 states, 81 inputs, and 61 outputs. The technique is also now being taught in the Computer Science Department's Switching Circuits Lab series as part of an effort to train students in asynchronous as well as synchronous digital design.

Affiliation of the Author - The author is with the Department of Computer Science, University of Utah, Salt Lake, Utah 84112. This work was supported by the Burroughs Corporation.

REFERENCES

- [1] D. A. Huffman, "The Synthesis of Sequential Switching Circuits," Journal of the Franklin Institute, Vol. 257, No. 3,4, March and April 1954, pp. 161-190,275-303, Reprinted in Moore; Sequential Machines: Selected Papers; Addison-Wesley; 1964.
- [2] D. A. Huffman, "Design of Hazard-free Switching Circuits," J. ACM, Vol. 4, No. 1, January 1957, pp. 47-63, .
- [3] S. H. Unger, Asynchronous Sequential Switching Circuits, Wiley-Interscience, New York, , 1969, .
- [4] C. L. Seitz, Graph Representations for Logical Machines, PhD dissertation, M. I. T., Jan 1971, .
- [5] C. R. Clare, Designing Logic Systems Using State Machines, McGraw-Hill, New York, , 1972, .
- [6] W. I. Fletcher, An Engineering Approach to Digital Design, Prentice-Hall, Englewood Cliffs, N. J., , 1980, .
- [7] R. E. Miller, Switching Theory, Wiley, New York, Vol. 2, 1965, Chapter 10 is a review of Muller's work on speed independent circuits. .
- [8] C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley, Reading, Mass, 1980, ch. 7, System Timing, Chapter by C. L. Seitz III.