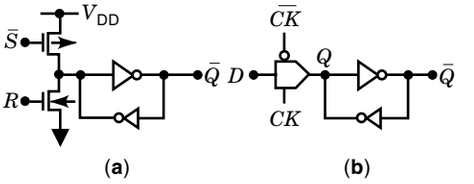

Figure 2. (a) Set-reset latch, (b) half (master) of D latch.

SEQUENTIAL CIRCUITS

DEFINITION AND STRUCTURE

Logic circuits are classified into combinational and sequential circuits. The classification has no ambiguity, because there are two qualitatively different digital circuit elements in their idealized model: logic gates, which respond instantaneously to the inputs, and memories, which retain the past input data. A combinational circuit is made of logic gates only. A sequential circuit is made of combinational circuits and memories, and some inputs to the combinational circuits are driven by the outside signal and the rest by the memory outputs. The combinational circuits drive the memories and store the logic operation results in them. The signal paths of a sequential circuit make a closed loop, as schematically shown in Fig. 1. As the digital signals circulate the loops, the processing is carried out in sequence. That is why the circuit is called sequential. Since some of the input signal to the combinational circuit comes from the memories that hold the data previously processed by the circuit, it is able to produce a logic answer not only from the present input data, but also using the past input data. If intelligence is defined as the capability of using experience for present decision making, a sequential circuit has elementary intelligence. The experience of a sequential circuit is the processed data of the past inputs. This is the most fundamental significance of a sequential circuit. A combinatorial circuit driven by uniformly delayed input signals is not a sequential circuit, even if it has memories. Since the combinatorial circuit is known (1,2), we consider the memories first. The sequential circuit memories have subtle details that characterize the circuit. In recent years, most of the logic circuits have been built as integrated circuits, and for a large-scale integration, CMOS is the preferred technology. The circuit examples are mostly CMOS circuits.

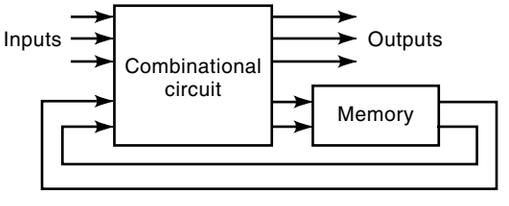

Figure 1. Structure of a sequential logic circuit.

MEMORIES

Memory devices used in a sequential logic circuit are of two different types. One type has positive feedback controlled by signals from the outside, and its memory retention time can be controlled by the control signals or by a clock. The clock is a systemwide memory control signal by which the processing proceeds in regular steps, and the memory is able to retain the data for any length of time. This type of memory is called a latch or a flip-flop. The other type of memory is either a cascaded chain of gates or some other continuous physical structure that is able to delay the input signal to the output. The output signal of the delay circuit at time t is the input signal at time $t - \tau$, where τ is the delay time.

The memory devices of the first type have popular names. The conventionally used devices are as follows: (1) set-reset latch, (2) D latch, (3) T flip-flop, and (4) J-K flip-flop. The T flip-flop changes the state every time the input makes a transition. The J-K flip-flop has several combinations of these features. All of the memory devices come in clocked and unclocked versions. The D- and the set-reset latches are preferentially used in integrated circuits, and the other types are referred to in McCluskey (3), Nagle (1), and Mano (2). In CMOS VLSI circuits, two types of latches—the set-reset latch and the D-latch—are most frequently used. Although there are several variations of latch circuits, a typical circuit used for ultra-high-speed CMOS circuits is shown in Figs. 2(a) and 2(b). In Fig. 2(a), if $\overline{S}$ is HIGH and R is LOW, the set-reset latch stays in the state it was previously forced into. Subject to this condition, if $\overline{S}$ is temporarily pulled down to LOW while R is kept LOW, the latch is set, or $\overline{Q}$ output becomes LOW if it has not been in the set state. Subject to the same condition, if R is temporarily pulled up to HIGH while $\overline{S}$ is kept HIGH, the latch is reset, or $\overline{Q}$ output becomes HIGH. If $\overline{S}$ is LOW and R is HIGH simultaneously, the latch's subsequent state is unpredictable. In a VLSI circuit such a conflict can be prevented by adding an extra logic circuit. J-K flip-flop, originally intended to avoid the control input conflict, is used infrequently. The circuit of Fig. 2(b) is the half of the D

latch. In a sequential logic circuit the second stage, in which the control clock CK and $\overline{CK}$ are exchanged, is directly cascaded to it, and the pair is called a master-slave D-latch pair. The circuit of Fig. 2(a) or 2(b) has a strong output driver (shown by the large inverter sign) that drives the output, and a weak driver (shown by the small inverter sign) that holds the input of the strong driver. The set/reset FET of Fig. 2(a) and the transmission gate of Fig. 2(b) must be able to overpower the output node held by the small inverter. This type circuit is faster than the conventional latch circuit using the tristable inverters.

The delay device used instead of the controllable memory device is generally called a delay line. In spite of the name, the traditional LC or the acoustic-delay lines are not used anymore. Rather, the natural delays of the cascaded logic gates, especially of simple inverters, are used. In this type of delay-memory circuit, certain requirements exist for the circuit's operational integrity. Figure 3, curve A, shows the input waveform to a delay line. If the delay line outputs a slowly rising (or falling) waveform like curve B, the delay line as a memory device is not acceptable. This is because the delay time of signal B referenced to signal A depends on where the logic threshold voltage, the boundary between the Boolean HIGH and LOW levels, is set. Depending on the logic gate driven by the delay line output, as well as the gate's operating condition (e.g., the timing of the other input signal), the delay time varies over a wide range. Since the delay line does not have a definite delay time, the circuit's operational integrity is compromised. A signal like curve C has a clearly defined delay time, and the delay line is acceptable as a memory. To create a waveform like C, the delay line circuit must have a crisply switching buffer at least at the output stage, or LC delay line structure. A passive RC low-pass filter is not acceptable, because the output waveform looks like curve B. A confusing issue in sequential circuit classification is that if a chain of logic gates makes a closed loop, the loop works as a memory device as well as a combinational logic circuit. It is impossible to separate the loop circuit into combinational and memory parts. From this viewpoint, any logic circuit that has a closed loop of gates is a sequential circuit, and this is its widest definition based on the circuit structure. The schematic of Fig. 1 cannot be used to characterize this type of circuit directly without further interpretation, which will be provided later.

CLASSIFICATION OF SEQUENTIAL CIRCUITS

In a sequential circuit, the combinational logic part has no feature to distinguish one circuit from the other. To classify a sequential circuit, we observe the structure of the loops. The key features by which to classify sequential circuits are (1) if a loop is a single loop or a multiple loop, and (2) how the loops are controlled. The most convenient means of classification is to single out the most convenient and most widely used synchronous sequential circuit by focusing attention on the function of the clock, and then to discuss the other, asynchronous sequential circuits, in regard to their special features.

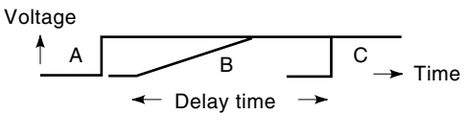

Figure 3. Delay line memory waveform requirement.

The memory device used for a conventional synchronous sequential circuit is a pair of D latches (Fig. 2) connected in the master-slave configuration. The rectangular box represents the latch. The symbols in the box, D is the input, C is the clock, and Q and $\overline{Q}$ are the normal and the inverted outputs [see Fig. 2(b)]. The latch pair replaces the memory block of Fig. 1. The combinational logic circuit is driven by outputs from the slave latches and by external signals. The external signals must be synchronized to the clock, to fit into the appropriate timing window. The master latch receives the signal processed by the combinational circuit. In the simplest implementation, all the master and all the slave latches are driven by a clock and its complementary, respectively. When the master latches are transparent (input open), the slave latches are their inputs closed, and vice versa. The clock transition must be fast to ensure that either the master latch is transparent and the slave latch is closed, or the master latch is closed and the slave latch is transparent. This is called an edge-triggered timing scheme.

At the beginning of a process cycle, the clock transition effects disconnection between the logic circuit and the master latches. The source data to the combinational logic circuit are secured first by the positive feedback retention mechanism of the master latches, to the designated Boolean levels. The Boolean levels are transferred to the slave latches that are transparent and to the combinational circuit; while that is going on, the combinational logic circuit has already started processing the data. Then the second clock transition effects disconnection between the master and the slave latches, and the slave latches apply positive feedback to retain the acquired data. The master latches are now open and are ready to accept the processed data as they arrive at the master latch's input terminals. During the data acquisition process by the master latch, the latch circuit's delay exercises intelligence to filter out the noise and the spurious pulses. At the end of the logic operation period set by the clock, the master latch is disconnected from the combinational logic circuit, and its output is connected to the slave latch. The master latches exercise intelligence to decide the Boolean level of the output of the combinational circuit by eliminating the spurious noise, and then by quantizing to generate the Boolean level. The processed data, or the answer of the cycle's logic operation, is now available to the slave latch. The synchronous sequential circuit is now ready to start a new clock cycle with new input data.

Synchronous sequential logic circuits have a number of advantages. Their step-by-step operation timed by the system-wide clock is easy to understand, design, and simulate, since the electrical phenomena in different cycles can be studied independently. This is perhaps the most important reason for their preferred use. Asynchronous sequential circuits are difficult to understand, design, and verify. Some of the problems of asynchronous sequential circuits, such as the race condition, do not exist in synchronous circuits, and in a synchronous system metastability failure does not occur if simple and uniform design discipline is followed. Furthermore, the data processing load can be optimally distributed over the clock cycles to attain the maximum throughput. Based on these advantages, synchronous sequential circuits are the predomi-

nant choice in implementing high-performance systems. The clock required to drive the system is simple. This scheme has a variation: The combinational logic circuits are inserted between the master and the slave latches, as well as between the slave and the master latches. This variation has more flexibility than the simplest scheme and is more tolerant to the rise/fall time requirement of the clock, but it requires a four-phase clock.

A sequential circuit that does not have the features of the previously defined synchronous sequential circuit belongs to a loosely defined group of asynchronous sequential circuits. Existence of the system-wide clock is crucial, but the choice of memory is also an issue: Specifically, a circuit using set-reset latches instead of D latches, or a circuit using the delay element as a memory device belong to asynchronous circuits. Practically, a large-scale synchronous sequential circuit may have an asynchronous sequential circuit as part of it, and clear distinction between them is impossible. Asynchronous sequential circuits are convenient for implementing simple function directly interfacing to the human hand.

REPRESENTATION OF SEQUENTIAL CIRCUIT OPERATION

Sequential circuits carry out the most complex signal processing operations in electronics. It is necessary to have a simple and intuitive way to represent their operation in graphical or tabular form. The circuit shown in Fig. 4 has two inputs, one output, and two D latches. The D latches are in the master-slave configuration, or the two cascaded stages of the simple latch circuit shown in Fig. 2(b) and driven by the complementary pair of clocks. The two latch outputs (y_1, y_2) can be any of the four combinations, (0,0) through (1,1). The input variables x and w, and the two outputs of the latches y_1 and y_2, determine the present state of the circuit at the clock phase in which the master and the slave latches are disconnected. The set of variables x, w, y_1, and y_2 (not the complemented form $\overline{y}_2$) define the total state of the sequential circuit.

The next states Y_1 and Y_2, and the output z of the circuit of Fig. 4, are written by Boolean algebra as follows:

$$Y_1 = y_2 + \overline{x} \qquad Y_2 = \overline{y}_1 + \overline{w} \qquad z = wy_1 + x\overline{y}_2$$

The output of latches 1 and 2 determines the present internal state of the circuit. The Boolean levels are written in a vector format (y_1, y_2), as (0,0) or (0,1), to designate the internal state. As the HIGH to LOW clock transition occurs, the connection to the master latches is closed, and the state of the master latches is transferred to the slave latches to become the next state of the circuit. The sequence of transitions caused by the clock transition and by the input signal describes the operation of the circuit. One convenient way to describe the circuit operation is to use the state transition diagram shown in Fig. 5. In this figure the four internal states of the circuit are shown within the circles. The arrowed curves connecting the circles are the state transitions caused by the clock and the input signals. The indications associated with the curves, **/*, show the input signal Boolean levels in the order of (w-x) in front of the slash, followed by the Boolean level of the output z.

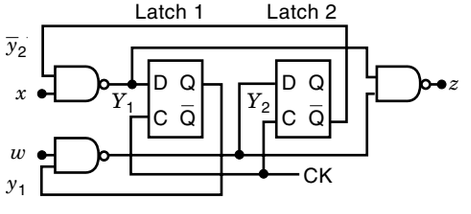

Figure 4. An example of a synchronous sequential circuit.

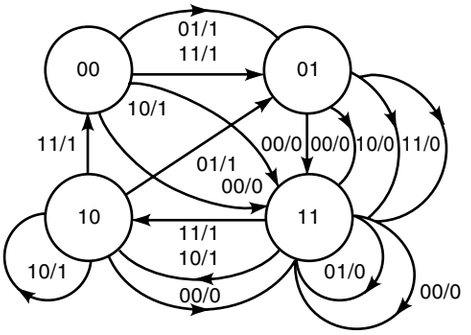

Figure 5. State transition diagram.

The state diagram of Fig. 5 is the *Mealy*-type state diagram (2), by which the input signals and the present internal state determine the output signals and the next internal state. In the Mealy diagram the clock of the memories affects the state transition and the output change, but, in addition, an asynchronous input change may affect the output. To make a completely synchronous representation of the system, the input signals must be assumed to have been delivered from the outside memories, which are driven by the same clock. There is a second, *Moore*-type, state diagram, in which only the present state and the inputs creating a transition to the next state are shown. The present outputs are determined by combinational circuits from the present internal state. In the Moore diagram the outputs are synchronized to the memory clock.

A second way of describing the circuit operation is to use the state transition table shown in Table 1. In this table, the columns are the four possible combinations of the input signals w and x, and the rows are the four possible present internal states of the circuit [as defined before, the vectorical combination of the latch outputs (y_1, y_2)]. The entries of the table are the next state of the circuit upon the clock transition followed by the output(s). Both the state diagram and the state table describe the circuit operation independent of the input signal sequence, and this compactness is the convenient feature of the two representations.

Table 1. State Transition Table

Present State	Input w and x			
$y_1\,y_2$	00	01	11	10
00	11/0	01/1	01/1	11/0
01	11/0	11/0	11/0	11/0
11	11/0	11/0	10/1	10/1
10	11/0	01/1	00/1	10/1
*	Next state Y_1, Y_2/output			

The state transition diagram and the state transition table give the peculiarities of the circuits. The circuit has two internal states, (00) and (01), which are *transitory*; if the clock edge arrives, the state changes to other state. The states (10) and (11) are not transitory; if in the state (10) and if the input is (10), the circuit returns to the same state after a clocktick. The state (11) has two possible inputs, (01) and (00), that bring the circuit back to the same state after a clocktick. If (01) and (00) input alternate and if the circuit is originally in the internal state (11), the circuit is stuck at state (11) forever. The circuit has no *stable* state. The circuit has instability: If the input is (11), the circuit goes through a sequence (00) → (01) → (11) → (10) and back to (00). The circuit works as a ringoscillator. As observed from this example, the input sequence independent representation provides many insights into sequential circuit operation.

MODEL OF ASYNCHRONOUS SEQUENTIAL CIRCUITS

Definition of asynchronous sequential circuits is not a simple matter, because the memory device, a latch, is not an atomlike building block of a logic circuit: Either it is a uniform delay line, or it consists of atomlike gates, and a memory can be built by connecting any number of logic gates in a closed loop. Such a generalized data storage device has a built-in data processing capability as well, and the inputs to the loop cannot be named by descriptive names, such as set, reset, and clock. This complication makes design, analysis, and representation of the operation of an asynchronous sequential circuit complex and its subclassification arbitrary. A simple memory device like a set-reset latch is used in asynchronous sequential circuits. Asynchronous sequential circuits that use unclocked set-reset latches, shown in Fig. 2(a), are often used to implement a simple state machine that accepts inputs directly from a human operator. The debounce circuit, which reshapes the pulse from toggling a mechanical switch, is perhaps the most popular use of a set-reset latch.

To design a sequential logic circuit that has specified functionality is the problem of synthesis. Circuit synthesis requires analysis methods, a few general methods to convert the functionality specifications to the best form for implementation, and a lot of insight and experience in system-level operation. We discuss the first two, the analysis methods and design specifications that are common to synchronous and asynchronous circuits, while using an asynchronous circuit as an example.

The circuit shown in Fig. 6 is an asynchronous sequential circuit having two loops made by conventional logic gates. The objective of the following analysis methods is to find out the peculiarities of operation of the circuit: If the peculiarities are found, we may say that we understand the circuit. The first

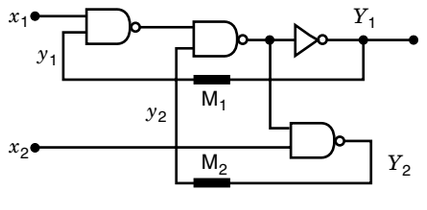

Figure 6. An example of an asynchronous sequential circuit.

Table 2. State Transition Table

*	Inputs x_1 and x_2			
$y_1\, y_2$	00	01	11	10
00	01	00	00	01
01	11	11	11	11
11	11	11	00	01
10	01	00	00	01
*	Y_1 and Y_2			

analysis step is to separate the logic operation and the delay of the gates. The logic gates in the figure are idealized logic gates that execute the logic operation instantly. The delay times of all the gates of the two loops are lumped into the fictitious delay elements M_1 and M_2 shown in the figure, which work as delay line memories. A digital signal propagates from the right to the left through the fictitious delay elements, and the input and the output have different node variable names. The outputs of the delay elements have node Boolean variables y_1 and y_2, and inputs Y_1 and Y_2, respectively. The input signals x_1 and x_2 and y_1 and y_2 define the total states of the asynchronous sequential circuit.

In a synchronous sequential circuit the one clock for all the memories has the function of pushing the state of the circuit forward. In an asynchronous sequential circuit any input signal may control the state of the memories, which are most often complex loops of gates. Then for the following analysis, as well as in the real operation of the circuit, only one input variable is allowed to change at a time. When the input variable changes, the circuit must have arrived at the steady state already, determined by the previous input signal change. An asynchronous sequential circuit operates subject to this stronger condition than a synchronous sequential circuit, and therefore the processing power and the operational flexibility are less than for a synchronous circuit.

Using the total state (x_1, x_2, y_1, y_2), the next state variables Y_1 and Y_2 are determined by the Boolean logic equations

$$Y_1 = \overline{x}_1 y_2 + \overline{y}_1 y_2 \qquad Y_2 = \overline{x}_1 y_2 + \overline{y}_1 y_2 + \overline{x}_2$$

and the state transition table is made, as shown in Table 2. In this example, Y_1 is the output.

The transition table shows various peculiarities of the circuit. If the input signal (x_1, x_2) is either (0,0) or (0,1) and if the initial internal state (y_1, y_2) is (1,1), the next internal state (Y_1, Y_2) is (1,1), or the internal state never changes. The circuit is in a steady state if $x_1 = 0$. Similarly, if the input signal is either (0,1) or (1,1) (or if $x_2 = 1$) and if the initial internal state is (0,0), the state never changes. In the two conditions the circuit is in a steady state that does not change with time. If the input signal is held at (0,0) and if the initial internal state is (0,0), the next internal state is (0,1), followed by (1,1), and the circuit arrives at the steady states for $x_1 = 0$ and $x_2 = 0$. If the internal state immediately after the input signal change is not one of the steady state, the circuit takes several steps to one of the steady states. This is not the only mode of operation of the circuit. If the input signal is held at (1,0) and if the initial internal state is (0,1), the next internal state is (1,1), and the next to the next internal state is (0,1), the same internal state as the initial internal state. The circuit oscillates between the two internal states. In this case

the circuit is equivalent to a three-stage cascaded enabled inverting gate, which works as a ringoscillator.

The state transition diagram or state transition table provides many details of asynchronous sequential circuit operation, other than those discussed before. For the purpose of circuit synthesis, it is convenient to represent the internal state not by a preassigned Boolean vector form like (0,0), but by a symbolic character A, B, ... Here we use identification like A: (0,0), B: (0,1), C: (1,1), and D: (1,0) in the state transition table, as in Fig. 7. This figure is to show various structures of a state transition table, and it is not related to the last example. Figure 7(a) shows that the next state of B for any value of input x is B. This means that state B is a state if, once entered, the circuit stays in forever. The state can be entered from another state, or it can be initially set. If the other entries *'s of Fig. 7(a) have no B state, the B state is an isolated internal state, and if the state transition table has an isolated internal state, the state diagram is separated to two independent parts. Figure 7(b) shows that as long as x is 1 the next state of B is C, and the next state of C is B. The states B and C alternate, and the circuit oscillates. This is an unstable circuit that is not practically usable as a logic circuit. In Fig. 7(c) the oscillation goes over the entire state as x makes a transition from 0 to 1 and stays at the value: The state sequence is A, D, B, C, and back to A.

The entries of the state transition table can be any of the four internal states, A, B, C, or D, in any order and in any number of duplications, although many of such randomly generated tables do not carry out useful functions. This great variability creates many other strange, often practically undesirable behaviors. In Fig. 7(d) we note the internal state A = (0,0), B = (0,1), C = (1,1), and D = (1,0). If the initial state is A, the two memory outputs that determine the internal state of the circuit both flip if x changes from 0 to 1. If the timing sequence is included in consideration, the change may take place in three different ways: (1) a simultaneous change, A → C, (2) A → B → C, and (3) A → D → C. In the state transition table of Fig. 7(d), all the three changes end up with the same final state C. The circuit executes the logic operation correctly. If the state transition diagram is as shown in Fig. 7(e), however, the final state depends on the timing sequence: (1) A simultaneous change, A → C, ends up with state C; (2) if A → B occurs first, the circuit settles at state B; (3) if A → D occurs first, the circuit settles at state D. Since the order of occurrence of the state change is not indicated in the logic diagram, the final state cannot be determined from the logic diagram alone. The problems discussed in relation to Figs. 7(d) and 7(e) are called the *race* condition, since the final destination depends on the order of the state change (5). Figure 7(d) is called a *noncritical* race, since the final state does not depend on the order. Noncritical race is usually harmless for logic circuit operation. Figure 7(e) is called a *critical* race, since the final state is determined by the order. A critical race condition can be avoided if there is a state sequence that leads to the correct final state, if the state change sequence occurs in an arbitrary order. In Fig. 7(f), if the state sequence occurs simultaneously, the circuit ends up with state C. The critical race condition is avoided by the intermediate state B, which has the right destination state C. If A → B occurs first, the sequence B → C follows, and the circuit ends up with a correct state. If A → D occurs first, however, the circuit ends up with state D, which is not the state the designer intends. From these observations, the critical race condition can be avoided in several ways: by providing an intermediate internal state like B, through which the circuit ends up with the correct final state, or by assigning proper binary state vectors to the symbolic internal states A, B, C, and D such that only one latch changes the state upon input transition. A number of possibilities of circuit operation exist, and the state table is a simple method by which to analyze complex cases.

OPERATION ON THE SET OF INTERNAL STATES

State is the central concept of sequential circuits. Operations on a set of states, such as simplification, choice of a better set of states for hardware minimization or for high reliability, or assigning binary vectors to a symbolic state, as we saw before are an important issue, both for circuit analysis and synthesis. If a sequential circuit is considered as a black box accessible only from the input and output terminals, the set of internal states within the black box creating the functionality may have many alternatives. If a circuit design already exists, the choice of internal states may not be the simplest or most de-

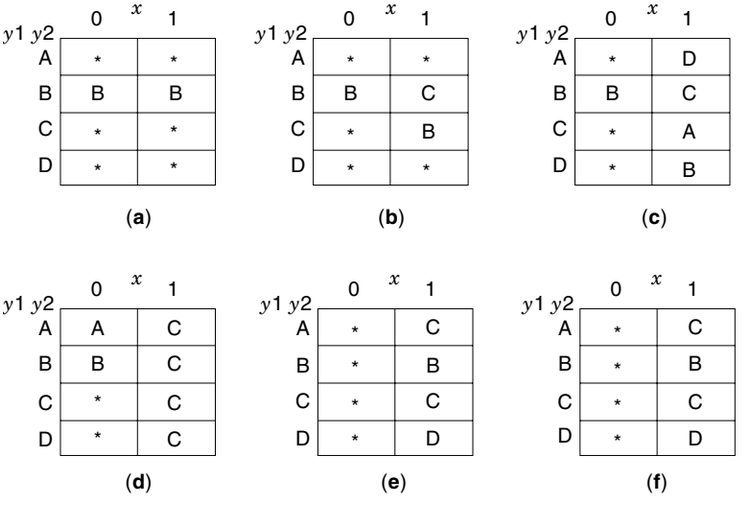

Figure 7. State transition diagram and asynchronous sequential circuit operation.

Table 3. Original State Transition Table

Initial State	Input x	
*	$x = 0$	$x = 1$
A	F/0	C/0
B	D/0	F/1
C	F/1	B/1
D	E/1	A/0
E	E/0	C/0
F	E/0	C/0
G	A/1	B/1
*	Final state/the output	

Table 4. Simplified State Transition Table

Initial State	Input x	
*	$x = 0$	$x = 1$
A	A/0	C/0
B	D/0	A/1
C	A/1	B/1
D	A/1	A/0
*	Final state/the output	

sirable. The first of such issues is simplification of the circuit by reduction of the number of internal states of a sequential circuit while maintaining the same input-output functionality. This is the first step to sequential circuit synthesis.

Suppose that the state transition table, Table 3, is given. State reduction can be carried out by several methods: (1) by inspection, (2) by partition, and (3) by using an implication table (1,2). By inspection, for input $x = 0$ and $x = 1$, states E and F go to the same state, E and C, respectively, and the outputs for the two cases are also the same at 0. Then the two states E and F are equivalent. Further reduction of the number of states, however, is not obvious by an inspection. As for the systematic methods useful in such cases, the partition method is referred to in the references, and a versatile and simple implication table method is briefly summarized here.

To display the equivalent/nonequivalent relationship of a pair of internal states, we make a diagram by stacking the boxes, as shown in Fig. 8. If two internal states are equivalent, the sequential circuit must generate the same output(s) for all the combinations of the input variable values. Any pair of internal states having at least one different output for a combination of inputs are not equivalent. The box at the column index B and the row index D, for instance, contains an indicator $\times$ since the two internal states are not equivalent by this criterion. Most of the boxes get $\times$ marks, and the internal state pairs are removed from further consideration. This table is called an implication table because of the internal states A through G, there can be internal states that are equivalent but are named by different symbolic indicators.

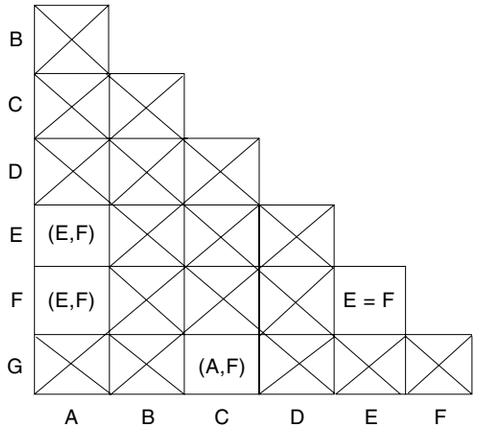

Figure 8. Implication table to find equivalent states.

Some of them are equivalent if the following situation occurs: If internal states A and E are to be equivalent, the internal states E and F must be equivalent for $x = 0$. As for $x = 1$, the equivalence condition is already satisfied by the same internal state C. The equivalency requirement of internal states A and F (namely, E = F) is written in the box at column A and row E, by a sign (E,F). Since internal E and F being equivalent *implies* internal states A and E being equivalent, the table is called an implication table. If the same procedure is repeated for the other empty boxes, that internal states A and E and internal states C and G are equivalent is implied by E and F and A and F being equivalent, respectively. The equivalence of states E and F is already established by inspection and is indicated by E = F.

Since states E and F are equivalent, the completed implication table shows that states A and E are equivalent, and states A and F are also equivalent. Then internal states C and G are also equivalent. The original seven states are grouped into equivalent states (A,E,F), B, (C,G), and D. The seven-state table is simplified to the four-state table shown in Table 4. Obviously, the circuit having four states is easier to implement than the circuit having seven states, yet the two implementations are functionally equivalent. Since m memories create a maximum of 2^m internal states, the number of the memories is reduced from 3 to 2 by the reduction process.

Synthesis of a sequential circuit having the specified functionality begins with conversion of the requirement in the common language into a state transition table or diagram. In this phase of the work the states carry symbolic names, such as A, B, C. The state transition table is then examined by the procedure described in this section if the equivalent states can be found, and if the total number of states can be reduced, to make the circuit simpler. In the next step the binary numbers are assigned to each state A, B, C, . . ., such that the circuit is race free. Then, using the simplified and the state-assigned transition table, the logic function tables for the combinational logic circuit, which creates the required output (z_1, z_2, . . .), and the next internal states (Y_1, Y_2, . . .) from the inputs (x_1, x_2, . . .) and the present state (y_1, y_2, . . .) are set up. The required combinational circuits are then synthesized using the standard technique (1,2) and they are integrated with the memory circuit. This process completes the synthesis procedure.

A NEW LOOK AT SEQUENTIAL LOGIC CIRCUITS

Synchronous sequential circuits have the merits of easily understood operation, simple and systematic design procedure, efficient use of combinational hardware, ability to carry out varieties of operation by adding small amounts of logic gates

and control signals (as in the processor datapath), and easy interfacing to the external circuits that have the same systemwide clock signal. These merits of synchronous sequential circuits are especially advantageous to the large-scale integrated circuit environment, and they have been the driving force of the rapid growth of microprocessor technology. Although these advantages will be exploited actively in the future, there are other advantages of synchronous sequential circuits that are clearly visible (e.g., ultra-high-speed electronics from the electronic circuit theorist's viewpoint).

The basic structure of the synchronous sequential circuit shown in Fig. 1 using the edge-triggered master-slave D-latch pair allows us, conceptually, to cut the circuit loop open between the master and slave latches. Then the slave latch provides the digital information to the combinational circuit, and the master latch receives the information processed by it. A digital signal is an object, very similar to an elementary particle in quantum mechanics, and the slave latch-combinational logic-master latch combination is equivalent to a measurement setup to determine the velocity of the pseudoparticle that carries the digital information. We note here that the quantum mechanical nature of the digital circuit originates from the impossibility of setting a clear threshold voltage that distinguishes the Boolean HIGH and LOW levels. Since the threshold voltage is uncertain, or hidden from the digital circuit theory, the digital circuit becomes a quantum mechanical object, very similar to what has been envisioned by Einstein, Podorsky, Rosen (4), and Shoji (6).

From this viewpoint a new idea of increasing the speed of digital signal processing to the limit emerges. By properly choosing the circuit between the latches, the quantum mechanical information-carrying particle can be transferred by the equivalent to the quantum mechanical tunnel effect, which takes, in its purely physical model, a short time. This new viewpoint suggests that the combinational circuit between the two latches need not be constructed from logic gates, but can and should be constructed from varieties of circuits including analog circuits. It is my belief that the advantage of a synchronous sequential circuit originates from its basic structure, the pair of information source and observation setup. The structure, once considered as the basic quantum mechanical arrangement, should be able to provide the fastest circuit we can build, yet it satisfies the basic requirement of digital circuits—the step function signal format—generated by clocked latches. From this viewpoint a sequential circuit will be a rich source of research in future electronic circuit theory.

The quantum mechanical nature of a digital signal shows up most clearly when, against the circuit designer's intention, a narrow isolated pulse is generated, or a simple step function waveform is converted into a multiple transition, a wavy step function wavefront during the combinational logic operation. The narrow isolated pulse is called a static hazard, and the multiple transition step function a dynamic hazard (3). A hazard is an essential mode of operation of a combinational logic circuit built from gates having nonzero delay, and this is clear from the following example. Suppose that a synchronous sequential circuit includes a NAND gate, whose two inputs A and B were the HIGH and the LOW logic levels, respectively. The output is at the default HIGH level. After the slave clocktick, the step function signal fronts arrive sometime later at A and B to make a HIGH to LOW transition at input A and a LOW to HIGH transition at input B. After the transitions are over, the output of the NAND gate is at the same HIGH logic level as before. The problem is that the timing relation of the wavefront is not always subjected to the control of the designer. If the LOW to HIGH transition of the B input occurs earlier than the opposite polarity A input transition, both inputs of the NAND gate become temporarily HIGH, and the output of the gate is a temporarily LOW logic level. A downgoing isolated pulse whose width is the time between the input B transition to the input A transition is generated. Hazard is an unexpected and unwelcome guest to a circuit designer, and it is significant in state-of-art high-speed CMOS logic circuits. One way to describe a high-frequenty logic circuit operation is as a continuous sequence of generation and erasure of hazard pulses. The hazard pulse is usually narrow, and it may or may not be wiped out as the signal further propagates the gate chain. Some hazard features survive and arrive at the destination latch. The latch is the final measure to screen all the extra features of the step function front, by cleaning up the deformed pulse from the combinational circuit at the clocktick. This function is an elementary intelligent function of a digital circuit, and thus the synchronous logic circuit does have the minimum required and fundamentally necessary features of intelligent signal processing. That is why the circuit is so widely used. In an asynchronous sequential circuit, this function is distributed over the circuit, and this confuses the issue.

BIBLIOGRAPHY

1. H. T. Nagle, Jr., B. D. Carroll, and J. D. Irwin, *An Introduction to Computer Logic,* Englewood Cliffs, NJ: Prentice-Hall, 1975.
2. M. M. Mano, *Digital Design,* 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.
3. E. J. McCluskey, *Logic Design Principles,* Englewood Cliffs, NJ: Prentice-Hall, 1986.
4. D. Bohm, *Quantum Theory,* Englewood Cliffs, NJ: Prentice-Hall, 1951.
5. M. Shoji, *Theory of CMOS Digital Circuits and Circuit Failures,* Princeton: Princeton Univ. Press, 1992.
6. M. Shoji, *Dynamics of Digital Excitation,* Norwell, MA: Kluwer, 1998.

MASAKAZU SHOJI
Bell Laboratories

SEQUENTIAL CIRCUITS AND FINITE STATE AUTOMATA, TESTING. See LOGIC TESTING.

SERVICES, HOME COMPUTING. See HOME COMPUTING SERVICES.