Fundamentals of Digital Logic and Microcomputer Design. M. Rafiquzzaman

5

SEQUENTIAL LOGIC DESIGN

This chapter describes analysis and design of synchronous sequential circuits. Topics include flip-flops, Mealy and Moore circuits, counters, and registers. An overview of RAMs, state machine design using ASM chart, and asynchronous sequential circuit is also included.

5.1 Basic Concepts

So far, we have considered the design of combinational circuits. The main characteristic of these circuits is that the outputs at a particular time t are determined by the inputs at the same time t. This means that combinational circuits require no memory. However, in practice, most digital systems contain combinational circuits along with memory. These circuits are called "sequential."

In sequential circuits, the present outputs depend on the present inputs and the previous states stored in the memory elements. These states must be fed back to the inputs in order to generate the present outputs. There are two types of sequential circuits: synchronous and asynchronous.

In a synchronous sequential circuit, a clock signal is used at discrete instants of time to ensure that all desired operations are initiated only by a train of synchronizing clock pulses. A timing device called the "clock generator" produces these clock pulses. The desired outputs of the memory elements are obtained upon application of the clock pulses and some other signal at their inputs. This type of sequential circuit is also called a "clocked sequential circuit." The memory elements used in clocked sequential circuits are called "flip-flops." The flip-flop stores only one bit. A clocked sequential circuit usually utilizes several flip-flops to store a number of bits as required. Synchronous sequential circuits are also called "state machines." In an asynchronous sequential circuit, completion of one operation starts the operation that is next in sequence. Synchronizing clock pulses are not required. Instead, time-delay devices are used in asynchronous sequential circuits as memory elements. Logic gates are typically used as time delay devices, because the propagation delay time associated with a logic gate is adequate to provide the required delay. A combinational circuit with feedback among logic gates can be considered as an asynchronous sequential circuit. One must be careful while designing asynchronous systems because feedback among logic gates may result in undesirable system operation. The logic designer is normally faced with many problems related to the instability of asynchronous system, so they are not commonly used. Most of the sequential circuits encountered in practice are synchronous because it is easy to design and analyze such circuits.

5.2 Flip-Flops

A flip-flop is a one-bit memory. As long as power is available, the flip-flop retains the bit. However, its output (stored bit) can be changed by the clock input. Flip-flops are designed using basic storage circuits called "latches." The most common latch is the SR (Set-Reset) latch. A flip-flop is a latch with a clock input. This convention will be used in this book.

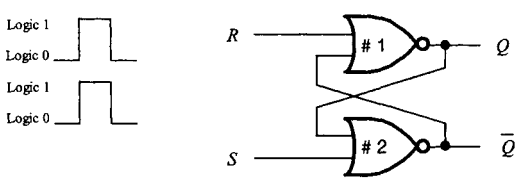
5.2.1 SR Latch

Figure 5.1 shows a basic latch circuit using NOR gates along with its truth table. The SR latch has two inputs, S (Set) and R (Reset), and two outputs Q (true output) and $\overline{Q}$ (complement of Q). To analyze the SR latch of Figure 5.1(a), note that a NOR gate generates an output 1 when all inputs are 0; on the other hand, the output of a NOR gate is 0 if any input is 1. Now assume that $S = 1$ and $R = 0$; the $\overline{Q}$ output of NOR gate #2 will be 0. This places 0 at both inputs of NOR gate #1. Therefore, output Q of NOR gate #1 will be 1. Thus, Q stays at 1. This means that one of the inputs to NOR gate #2 will be 1, producing 0 at the $\overline{Q}$ output regardless of the value of S. Thus, when the pulse at S becomes 0, the output $\overline{Q}$ will still be 0. This will apply 0 at the input of NOR #1. Thus, Q will continue to remain at 1. This means that when the set input $S = 1$ and the reset (clear) input $R = 0$, the SR latch stores a 1 ($Q = 1, \overline{Q} = 0$). This means that the SR latch is set to 1.

Consider $S = 0$, $R = 1$; the Q output of NOR gate #1 will be 0. This will apply 0 at both inputs of NOR gate #2. Thus, output $\overline{Q}$ will be 1. When the reset pulse input R returns to zero, the outputs continues to remain at $Q = 0$, and $\overline{Q} = 1$. This means that with set input $S = 0$ and reset input $R = 1$, the SR latch is cleared to 0 ($Q = 0, \overline{Q} = 1$).

Next, consider $Q = 1, \overline{Q} = 0$. With $S = 0$ and $R = 0$, the NOR gate #1 will have both inputs at 0. This will generate 1 at the Q output. The output $\overline{Q}$ of NOR gate #2 will be zero. Thus, the outputs Q and $\overline{Q}$ are unchanged when $S = 0$ and $R = 0$.

When $S = 1$ and $R = 1$, both Q and $\overline{Q}$ outputs are 0. This is an invalid condition because for the SR latch Q and $\overline{Q}$ must be complements of each other. Therefore, one must ensure that the condition $S = 1$ and $R = 1$ does not occur for the SR latch. This undesirable situation is indicated by a question mark (?) in the truth table. An SR latch can be built from NAND gates with active-low set and reset inputs. Figure 5.2 shows the NAND gate

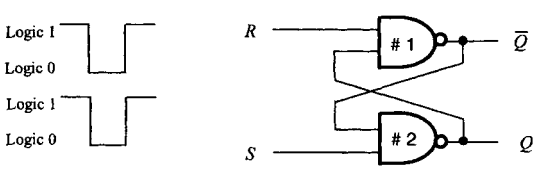


(a) NOR gate implementation

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	0	0?

(b) Truth table

FIGURE 5.1 SR Latch using NOR gates



(a) NOR gate implementation

S	R	Q	$\overline{Q}$
0	0	1	1?
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$

(b) Truth table

FIGURE 5.2 NAND implementation of an SR latch

implementation of an SR latch.

The SR latch with S and R inputs will store a 1 ($Q = 1$ and $\overline{Q} = 0$) when the S input is activated by a low input (logic 0) and $R = 1$. On the other hand, the latch will be cleared or reset to 0 ($Q = 0$, $\overline{Q} = 1$) when the R input is activated by a low input (logic 0) and $S = 1$.

Note that an active low signal can be defined as a signal that performs the desired function when it is low or 0. In Figure 5.2, the SR latch stores a 1 when $S = 0$ = active low and $R = 1$; on the other hand, the latch stores a 0 when $R = 0$ = active low and $S = 1$.

Note that the NAND gate produces a 0 if all inputs are 1; on the other hand, the NAND gate generates a 1 if at least one input is 0. Now, suppose that $S = 0$ and $R = 1$. This implies that the output of NAND gate #2 is 1. Thus, $Q = 1$. This will apply 1 to both inputs of NAND gate #1. Thus, $\overline{Q} = 0$. Therefore, a 1 is stored in the latch. Similarly, with inputs $S = 1$ and $R = 0$, it can be shown that $Q = 0$ and $\overline{Q} = 1$. The latch stores a 0.

With $S = 1$ and $R = 1$, both outputs of the latch will remain at the previous values. There will be no change in the latch outputs. Finally, $S = 0$ and $R = 0$ will produce a invalid condition ($Q = 1$ and $\overline{Q} = 1$). This is indicated by a question mark (?) in the truth table of Figure 5.2(b).

An SR latch can be used for designing a switch debouncing circuit. Mechanical switches are typically used in digital systems for inputting binary data manually. These mechanical ON-OFF switches (e.g., the keys in a computer keyboard) vibrate or bounce several times such that instead of changing state once when activated, a key opens and closes several times before settling at its final values. These bounces last for several milliseconds before settling down.

A debouncer circuit, shown in Figure 5.3, can be used with each key to get rid of the bounces. The circuit consists of an SR latch (using NOR gates) and a pair of resistors. In the figure, a single-pole double-throw switch is connected to an SR latch. The center contact (Z) is tied to +5 V and outputs logic 1. On the other hand, contacts X or Y provide logic 0 when not connected to contact Z. The values of the resistors are selected in such a way that X is HIGH when connected to Z or Y is HIGH when connected to Z.

When the switch is connected to X, a HIGH is applied at the R input, and $S = 0$, then $Q = 0$, $\overline{Q} = 1$. Now, suppose that the switch is moved from X to Y. The switch is disconnected from R and $R = 0$ because the ground at the R input pulls R to 0. The outputs Q and $\overline{Q}$ of the SR latch are unchanged because both R and S inputs are at 0 during the switch transition from X to Y. When the switch touches Y, the S input of the latch goes to HIGH and thus $Q = 1$ and $\overline{Q} = 0$. If the switch vibrates, temporarily breaking the connection, the S input of the SR latch becomes 0, leaving the latch outputs unchanged. If the switch bounces back connecting Z to Y, the S input becomes 1, the latch is set again, and the outputs of the SR latch do not change. Similarly, the switch transition from Y to X will get rid of switch bounces and will provide smooth transition.

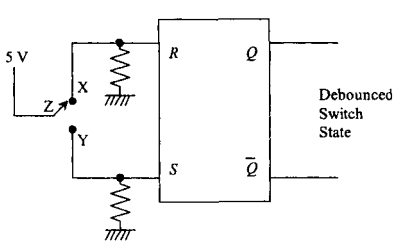
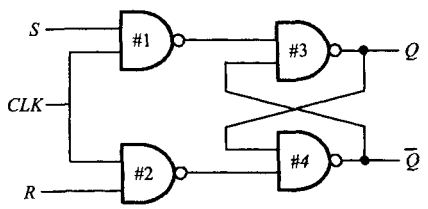


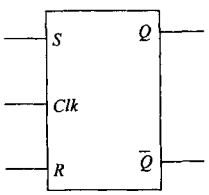
FIGURE 5.3 A debouncing circuit for a mechanical switch



(a) NAND gate implementation

S	R	Clk	Q^+	$\overline{Q^+}$
0	0	1	Q	$\overline{Q}$
0	1	1	0	1
1	0	1	1	0
1	1	1	?	?
X	X	0	Q	$\overline{Q}$

(b) Truth Table



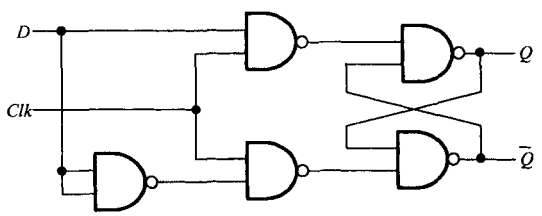
(c) Logic Symbol

FIGURE 5.4 RS Flip-Flop

5.2.2 RS Flip-Flop

An RS flip-flop is a clocked SR latch. This means that the RS flip-flop is same as the SR latch with a clock input. The SR flip-flop is an important circuit because all other flip-flops are built from it. Figure 5.4 shows an RS flip-flop.

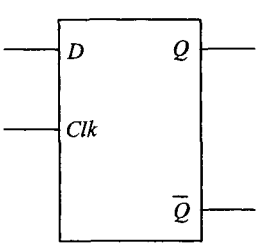
The RS flip-flop contains an SR latch with two more NAND gates. It has three inputs (S, CLK, R) and two outputs (Q and $\overline{Q}$). When $S = 0$ and $R = 0$ and CLK = 1, the outputs of both NAND gates #1 and #2 are 1. This means that the output of NAND gate #3 is 0 if $\overline{Q} = 1$ and is 1 if $\overline{Q} = 0$. This means that Q is unchanged as long as $S = 0$ and $R = 0$. On the other hand, the output of NAND gate #4 is 0 if $Q = 1$ and is 1 if $Q = 0$. Thus, $\overline{Q}$ is also unchanged. Suppose that $S = 1$, $R = 0$, and CLK = 1. This will produce 0 and 1



(a) NAND gate implementation

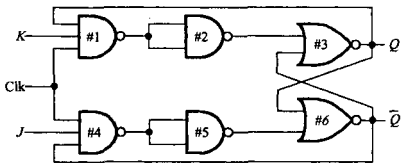
D	Clk	Q^+	$\overline{Q^+}$
0	1	0	1
1	1	1	0
X	0	Q	$\overline{Q}$

(b) Truth Table



(c) Logic Symbol

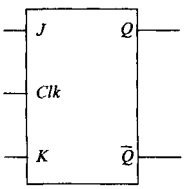
FIGURE 5.5 D Flip-Flop



(a) NAND gate implementation

J	K	Clk	$Q+$	$\overline{Q}+$
0	0	1	Q	$\overline{Q}$
1	0	1	1	0
0	1	1	0	1
1	1	1	$\overline{Q}$	Q
X	X	0	Q	$\overline{Q}$

(b) Truth Table



(c) Logic Symbol

FIGURE 5.6 JK Flip-Flop

at the outputs of NAND gates #1 and #2 respectively. This in turn will generate 1 and 0 at the outputs of NAND gates #3 and #4 respectively. Thus, the flip-flop is set to 1. When the clock is zero, the outputs of both NAND gates #1 and #2 are 1. This in turn will make the outputs of NAND gates #3 and #4 unchanged.

The other conditions in the function table can similarly be verified. Note that $S = 1$, $R = 1$, and CLK = 1 is combination of invalid inputs because this will make both outputs, Q and $\overline{Q}$ equal to 1. Also, Q and $\overline{Q}$ must be complements of each other in the RS flip-flop. Q^+ and $\overline{Q}^+$ are outputs of the flip-flop after the clock (CLK) is applied.

5.2.3 D Flip-Flop

Figure 5.5 shows the logic diagram, truth table and the logic symbol of a D flip-flop (Delay flip-flop). This type of flip-flop ensures that the invalid input combinations $S = 1$ and $R = 1$ for the RS flip-flop can never occur. The D flip-flop has two inputs (D and CLK) and two outputs (Q and $\overline{Q}$). The D input is same as the S input and the complement of D is applied to the R input. Thus, R and S can never be equal to 1 simultaneously.

The D flip-flop (called gated D flip-flop) transfers the D input to output Q when CLK = 1. Note that if CLK = 0, one of the inputs to each of the last two NAND gates will be 1; thus, outputs of the D flip-flop remain unchanged regardless of the values of the D input.

The D flip-flop is also called a "transparent latch." The term "transparent" is based on the fact that the output Q follows the D input when CLK = 1. Therefore, transfer of input to outputs is transparent, as if the flip-flop were not present.

5.2.4 JK Flip-Flop

The JK flip-flop is a modified version of the RS flip-flop such that the S and R inputs of the RS flip-flop correspond to the J and K inputs of the JK flip-flop. Furthermore, the invalid inputs $S = 1$ and $R = 1$ are allowed in the JK flip-flop. When J = 1, K = 1, and Clk = 1, the JK flip-flop complements its output. Otherwise, the meaning of the J and K inputs is the same as that of the S and R inputs respectively. Figure 5.6 shows a logic diagram of JK flip-flop along with its truth table. This is a NAND/NOR implementation, and is called a gated JK flip-flop. The circuit operation of Figure 5.6(a) is discussed in the following:

i) Suppose $Q = 1$, $\overline{Q} = 0$, and CLK = 1. With $J = 0$ and $K = 0$, the outputs of inverters

#2 and #5 are both 0. This means that the outputs of NOR gates #3 and #6 are 1 and 0 respectively. Therefore, the outputs of the flip-flop are unchanged

ii) Suppose $Q = 0$, $\overline{Q} = 1$, and CLK = 1. With $J = 1$ and $K = 0$, the outputs of inverters #2 and #5 are 0 and 1 respectively. This means that a 0 is produced at the output of NOR gate #6 ($\overline{Q} = 0$). Thus, apply a 0 at one of the inputs of NOR gate #3 generating a 1 at its output ($Q = 1$). The JK flip-flop is therefore set to 1 ($Q = 1$ and $\overline{Q} = 0$).

iii) Suppose $Q = 1$, $\overline{Q} = 0$ and CLK = 1. With $J = 0$ and $K = 1$, the outputs of the inverter #2 and #5 are 1 and 0 respectively. This means that the output of NOR gate #3 is 0. This will produce a 1 at the output of NOR gate #6. Thus, the flip-flop is cleared to zero ($Q = 0$ and $\overline{Q} = 1$).

iv) Suppose $Q = 1$, $\overline{Q} = 0$, and CLK = 1. With $J = 1$ and $K = 1$, the outputs of inverters #2 and #5 are 1 and 0 respectively. This will produce a 0 at the output of NOR gate #3 ($Q = 0$). This in turn will apply 0 at one of the inputs of NOR gate #6, making its output HIGH ($\overline{Q} = 1$). Thus, the output of the JK flip-flop is complemented. The other rows in the truth table of the JK flip-flop can similarly be verified.

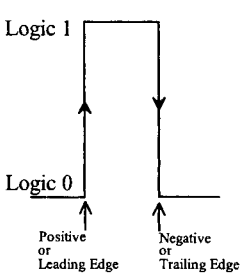
JK flip-flops are never built using the schematic of figure 5.6(a). This is because the schematic of Figure 5.6(a) will generate oscillations. For example, when J=1, K=1, and *Clk* =1, the outputs (Q and $\overline{Q}$) are complemented with the clock staying high after the first transition of the outputs. Since the outputs are fed back, the outputs will change continuously after being complemented once, causing oscillations. This undesirable behavior can be avoided using master-slave (edge-triggered) flip-flops discussed in the next section.

5.2.5 T Flip-Flop

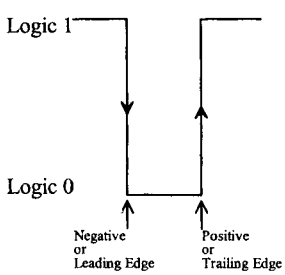
The T (Toggle) flip-flop complements its output when the clock input is applied with $T = 1$; the output remains unchanged when $T = 0$. The name "toggle" is based on the fact that the T flip-flop toggles or complements its output when the clock input is 1 with $T = 1$. T flip-flop is not available commercially. However, T flip-flop can be obtained from JK flip-flop in two ways. In the first approach, the J and K inputs of the JK flip-flop can be tied together to provide the T input; the output is complemented when $T = 1$ at the clock while the output remains unchanged when $T = 0$ at the clock. In the second approach, the J and K inputs can be tied to high; in this case, T is the clock input.

5.3 Master-Slave Flip-Flop

As mentioned before, sequential circuits contain combinational circuits with flip-flops in the feedback loop. These flip-flops generate outputs at the clock based on the inputs from



(a) Positive Pulse



(b) Negative Pulse

FIGURE 5.7 Clock Pulses

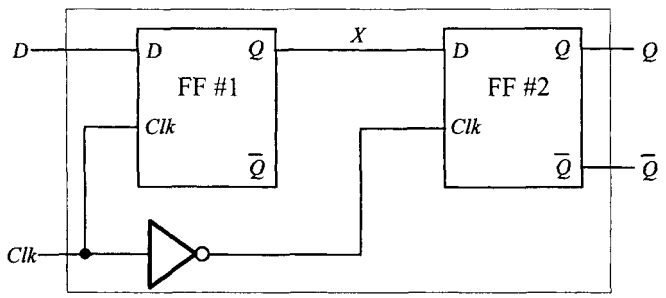


FIGURE 5.8 Typical Master-Slave D Flip-Flop

the combinational circuits. The feedback loop can create an undesirable situation if the outputs from the combinational circuits that are connected to the flip-flop inputs change values at the clock pulse simultaneously when flip-flops change outputs. This situation can be avoided if the flip-flop outputs do not change until the clock pulse goes back to 0. One way of accomplishing this is to ensure that the outputs of the flip-flops are affected by the pulse transition rather than pulse duration of the clock input.To understand this concept, consider the clock pulses shown in Figure 5.7. There are two types of clock pulses: positive and negative. A positive pulse includes two transitions: logic 0 to logic 1 and logic 1 to logic 0. A negative pulse also goes through two transitions: logic 1 to logic 0 and logic 0 to logic 1.

Assume that a positive pulse is used as the clock input of a D flip-flop. With the D input = 1, the output of the flip-flop will become 1 when the clock pulse reaches logic 1. Now, suppose that the D input changes to zero but the clock pulse is still 1. This means that the flip-flop will have a new output, 0. In this situation, the output of one flip-flop cannot be connected to the input of another when both flip-flops are enabled simultaneously by the same clock input. This problem can be avoided if the flip-flop is clocked by either the leading or the trailing edge rather than the signal level of the pulse. A master–slave flip-flop is used to accomplish this. Figure 5.8 shows a typical master-slave D flip-flop. A master-slave flip-flop contains two independent flip-flops. Flip-flop #1 (FF #1) works as a master flip-flop, whereas the flip-flop (FF #2) is a slave. An inverter is used to invert the clock input to the slave flip-flop.

Assume that the CLK is a positive pulse. Suppose that the D input of the master flip-flop (FF #1) is 1 and the CLK input = 1 (leading edge). The output of the inverter will apply a 0 at the CLK input of the slave flip-flop (FF #2). Thus, FF #2 is disabled. The master flip-flop will transfer a 1 to its Q output. Thus, X will be 1.

At the trailing edge of the CLK input, the CLK input of the master flip-flop is 0. Thus, FF #1 is disabled. The inverter will apply a 1 at the CLK input of the FF #2. Thus, 1 at the X input (D input of FF #2) will be transferred to the Q output of FF #2. When the CLK goes back to 0, the master flip-flop is separated. This avoids any change in the other inputs to affect the master flip-flop. The slave flip-flop will have the same output as the master.

5.4 Preset and Clear Inputs

Commercially available flip-flops include separate inputs for setting the flip-flop to 1 or clearing the flip-flop to 0. These inputs are called "preset" and "clear" inputs respectively.

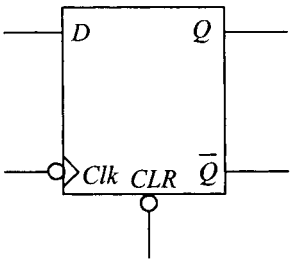
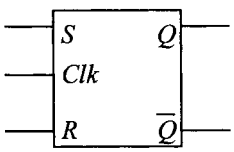


FIGURE 5.9 D Flip-Flop with Clear Input

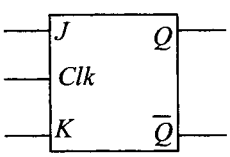


S	*R*	*Q*+	
0	0	*Q*	Unchanged
0	1	0	Reset
1	0	1	Set
1	1	?	Invalid

Q	*Q*+	*S*	*R*
0	0	0	*X*
0	1	1	0
1	0	0	1
1	1	*X*	0

(a) Symbolic Representation (b) Characterstic Table (c) Excitation Table

FIGURE 5.10 RS flip-flop

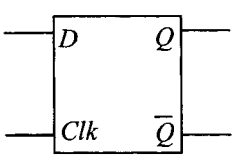


J	*K*	*Q*+	
0	0	*Q*	Unchanged
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}$	Complement

Q	*Q*+	*J*	*K*
0	0	0	*X*
0	1	1	*X*
1	0	*X*	1
1	1	*X*	0

(a) Symbolic Representation (b) Characterstic Table (c) Excitation Table

FIGURE 5.11 JK flip-flop

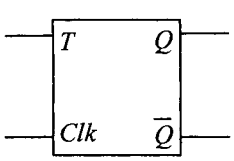


D	*Q*+	
0	0	Reset
1	1	Set

Q	*Q*+	*D*
0	0	0
0	1	1
1	0	0
1	1	1

(a) Symbolic Representation (b) Characterstic Table (c) Excitation Table

FIGURE 5.12 D flip-flop



T	*Q*+	
0	*Q*	Unchanged
1	$\overline{Q}$	Complement

Q	*Q*+	*T*
0	0	0
0	1	1
1	0	1
1	1	0

(a) Symbolic Representation (b) Characterstic Table (c) Excitation Table

FIGURE 5.13 T flip-flop

These inputs are useful for initializing the flip-flops without the clock pulse. When the power is turned ON, the output of the flip-flop is in undefined state. The preset and clear inputs can directly set or clear the flip-flop as desired prior to its clocked operation.

Figure 5.9 shows a D flip-flop with clear inputs. The triangular symbol indicates that the flip-flop is clocked at the positive edge of the clock pulse. In Figure 5.9, a circle (inverter) is used with the triangular symbol. This means that the flip-flop is enabled at the negative edge of the clock pulse. The circle at the clear input means that clear input must be 1 for normal operation. If the clear input is tied to ground (logic 0), the flip-flop is cleared to 0 ($Q = 0$, $\overline{Q} = 1$) irrespective of the clock pulse and the D input. The CLR input should be connected to 1 for normal operation. Some flip-flops may have a preset input that sets Q to 1 and $\overline{Q}$ to 0 when the preset input is tied to ground. The preset input is connected to 1 for normal operation.

5.5 Summary of Flip-Flops

Figures 5.10 through 5.13 summarize operations of all four flip-flops along with the symbolic representations, characteristic and excitation tables. In the figures, X represents don't care whereas $Q+$ indicates output Q after the clock pulse is applied.

The characteristic table of a flip-flop is similar to its truth table. It contains the input combinations along with the output after the clock pulse. The characteristic table is useful for analyzing a flip-flop.

The present state (present output), the next state (next output) after the clock pulse, and the required inputs for the transition are included in the excitation table. This is useful for designing a sequential circuit, in which one normally knows the transition from the present to the next state and wants to determine the required flip-flop inputs for the transition.

The D flip-flop is widely used in digital systems for transferring data. Several D flip-flops can be combined to form a register in the CPU of a computer. The 74HC374 is a 20-pin chip containing eight independent D flip-flops. It is designed using CMOS. The flip-flops are enabled at the leading edge of the clock. The 74LS374 is same as the 74HC374 except that it is designed using TTL.

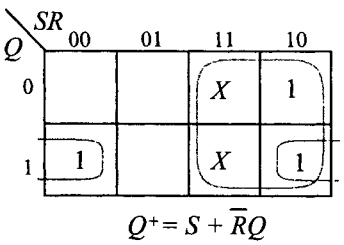
The JK flip-flop is a universal flip-flop and is typically used for general applications. Typical commercially available JK flip-flop includes the 74HC73 (or 74LS73A). The 74HC73 is a 14-pin chip. It contains two independent JK flip-flops in the same chip, designed using CMOS. Each flip-flop is enabled at the trailing edge of the clock pulse. Each flip-flop also contains a direct clear input. The 74HC73 is cleared to zero when the clear input is LOW. The 74LS73A is same as the 74HC73 except that it is designed using TTL. The T flip-flop is normally used for designing binary counters because binary counters require complementation.The T flip-flop is not commercially available. One way of obtaining a T Flip-flop is by connecting the J and K inputs of a JK flip-flop together (Section 5.2.5).

An example of a commercially available level-triggered flip-flop is the 74HC373 (or 74LS373). The 373 (20-pin chip) contains eight independent D latches with one enable input.

Sometimes the characteristic equation of a flip-flop is useful in analyzing the flip-flo p's operation. The characteristic equations for the flip-flops can be obtained from the truth tables. Figure 5.14 through 5.16 show how these equations are obtained using K-maps for RS, JK, T, and D flip-flops.

Q	S	R	$Q+$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Invalid
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Invalid

(a) Truth Table for RS-FF



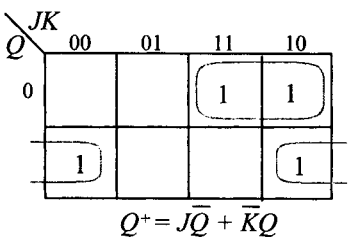
$Q^+ = S + \overline{R}Q$

(b) K-map for characteristic equation of RS-FF

FIGURE 5.14 Truth table and K-map for the characteristic equation of RS flip-flop

Q	J	K	$Q+$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(a) Truth Table for JK-FF

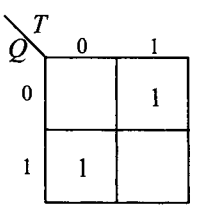


$Q^+ = J\overline{Q} + \overline{K}Q$

(b) K-map for characteristic equation of JK-FF

Q	T	$Q+$
0	0	0
0	1	1
1	0	1
1	1	0

(c) Truth Table for T-FF



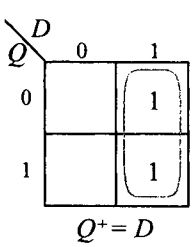
$Q^+ = T\overline{Q} + \overline{T}Q$

(d) K-map for characteristic equation of T-FF.

FIGURE 5.15 Truth table and K-map for the characteristic equation of JK and T flip-flops

Q	D	$Q+$
0	0	0
0	1	1
1	0	0
1	1	1

(a) Truth Table for D-FF



$Q^+ = D$

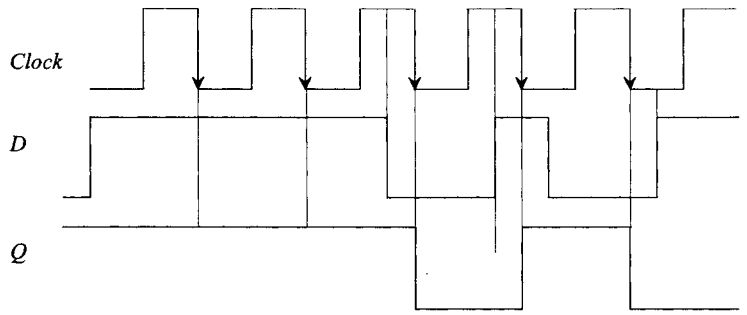
(b) K-map for characteristic equation of D-FF.

FIGURE 5.16 Truth table and K-map for the characteristic equation of D flip-flop

Example 5.1

Given the following clock and the D inputs for a negative-edge-triggered D flip-flop, draw the timing diagram for the Q output for the first five cycles shown. Assume Q is preset to 1 initially.

Solution:



5.6 Analysis of Synchronous Sequential Circuits

A synchronous sequential circuit can be analyzed by determining the relationships between inputs, outputs, and flip-flop states. A state table or a state diagram illustrates how the inputs and the states of the flip-flops affect the circuit outputs. Boolean expressions can be obtained for the inputs of the flip-flops in terms of present states of the flip-flops and the circuit inputs. As an example consider analyzing the synchronous sequential circuit of Figure 5.17.

The logic circuit contains two D flip-flops (outputs X, Y), one input A and one output B. The equations for the next states of the flip-flops can be written as

$$X^{+} = \overline{(X + Y) \cdot A}$$
$$Y^{+} = A + \bar{X}$$

Here $X+$ and $Y+$ represent the next states of the flip-flops after the clock pulse. The right side of each equation denotes the present states of the flip-flops (X, Y) and the input (A) that will produce the next state of each flip-flop. The Boolean expressions for the next state are obtained from the combinational circuit portion of the sequential circuit. The

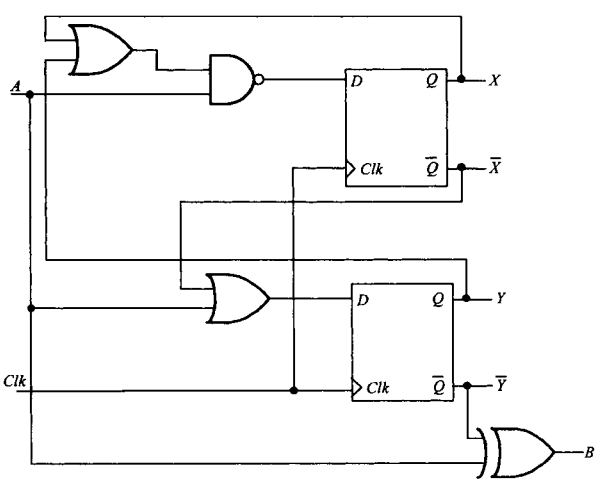


FIGURE 5.17 Analysis of a sequential circuit

TABLE 5.1 State Table for Figure 5.17

Present State		Input	Next State		Flip Flop Inputs		Output
X	Y	A	$X+$	$Y+$	D_X	D_Y	B
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	1
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
1	1	0	1	0	1	0	0
1	1	1	0	1	0	1	1

outputs of the combinational circuit are connected to the D inputs of the flip-flops. These D inputs provide the next states of the flip-flops after the clock pulse. The present state of the output B can be derived from the figure as follows:

$$B = A \oplus \overline{Y}$$

A state table listing the inputs, the outputs, and the states of the flip-flops along with the required flip-flop inputs can be obtained for Figure 5.17. Table 5.1 depicts a typical state table. The state table is formed by using the following equations (shown earlier):

$$X^+ = \overline{(X + Y) \cdot A}$$
$$Y^+ = A + \overline{X}$$

To derive the state table, all combinations of the present states of the flip-flops and input A are tabulated. There are eight combinations for three variables from 000 to 111. The values for the flip-flop inputs (next states of the flip-flops) are determined using the equations. For example, consider the top row with $X = 0$, $Y = 0$, and $A = 0$. Substituting in the equations for next states.

$$X^+ = \overline{(X + Y) \cdot A} = \overline{(0 + 0) \cdot 0} = 1$$
$$Y^+ = A + \overline{X} = 0 + \overline{0} = 1$$

Now, to find the flip-flop inputs, one should consider each flip-flop separately. Two D flip-flops are used. Note that for a D flip-flop, the input at D is same as the next state. The D input is transferred to the output Q at the clock pulse. Therefore, $X+ = D_x$ and $Y+ = D_y$.

The characteristic table of a D flip-flop, discussed before, is used to determine the flip-flop inputs that will change present states of the flip-flops to next state. The characteristic table of D flip-flop is provided here for reference:

D	Q^+
0	0
1	1

Therefore, for D flip-flops, the next states and the flip-flop inputs will be same in the state table. By inspecting the top row of the state table, it can be concluded that $D_x = 1$ and $D_y = 1$ because the next states $X+ = 1$ and $Y+ = 1$.

Finally, the output B can be obtained from the equation,

$$B = A \oplus \overline{Y}$$

TABLE 5.2 Another Form of the State Table

Present State		Next State				Flip Flop Inputs				Outputs	
		A=0		*A=1*		*A=0*		*A=1*		*A=0*	*A=1*
X	*Y*	*X+*	*Y+*	*X+*	*Y+*	D_X	D_Y	D_X	D_Y	*B*	*B*
0	0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	0	1	1	1	0	1	0	1
1	0	1	0	0	1	1	0	0	1	1	0
1	1	1	0	0	1	1	0	0	1	0	1

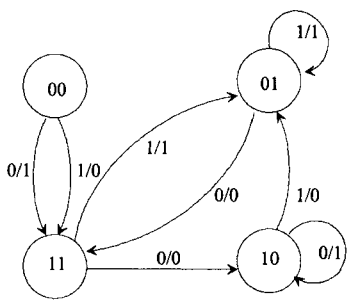


FIGURE 5.18 State diagram for Table 5.1

For example, consider the top row of the state table. $A = 0$ and $Y = 0$. Thus,

$$B = 0 \oplus \overline{0} = 0 \oplus 1 = 1$$

All other rows of the state table can similarly be verified. The state table of Table 5.1 can be shown in a slightly different manner. Table 5.2 depicts another form of the state table of Table 5.1.

A state table can be depicted in a graphical form. All information in the state table can be represented in the state diagram. A circle is used to represent a state in the state diagram. A straight line with an arrow indicator is used to show direction of transition from one state to another. Figure 5.18 shows the state diagram for Table 5.1.

Because there are two flip-flops (X, Y) in Figure 5.17, there are four states: 00, 01, 10 and 11. These are shown in the circle of the state diagram. Also, transition from one state to another is represented by a line with an arrow. Each line is assigned with *a/b* where *a* is input and *b* is output. From the example in Figure 5.18, with present state 10 and an input of 1, the output is 0 and the next state is 01. If the input (and/or output) is not defined in a problem, the input (and/or output) will be deleted in the state table and the state diagram.

The inputs of the flip-flops (D_x and D_y) in the state table are not necessary to derive the state diagram. In analyzing a synchronous sequential circuit, the logic diagram is given. The state equation, state table, and state diagram are obtained from the logic diagram. However, in order to design a sequential circuit, the designer has to derive the state table and the state diagram from the problem definition. The flip-flop inputs will be useful in the design. One must express the flip-flop inputs and outputs in terms of the present states of the flip-flops and the inputs. The minimum forms of these expressions can be obtained using a K-map. From these expressions, the logic diagram can be drawn.

5.7 Types of Synchronous Sequential Circuits

There are two types of Synchronous sequential circuits: the Mealy circuit and the Moore circuit. A synchronous sequential circuit typically contains inputs, outputs, and flip-flops. In the Mealy circuit, the outputs depend on both the inputs and the present states of the flip-flops. In the Moore circuit, on the other hand, the outputs are obtained from the flip-flops, and depend only on the present states of the flip-flops . Therefore, the only difference between the two types of circuits is in how the outputs are produced.

The state table of a Mealy circuit must contain an output column. The state table of a Moore circuit may contain an output column, which is dependent only on the present states of the flip-flops. A Moore machine normally requires more states to generate identical output sequence compared to a Mealy machine. This is because the transitions are associated with the outputs in a Mealy machine.

5.8 Minimization of States

A simplified form of a synchronous sequential circuit can be obtained by minimizing the number of states. This will reduce the number of flip-flops and simplify the complexity of the circuit implementations. However, logic designers rarely use the minimization procedures. Also, there are sometimes instances in which design of a synchronous sequential circuit is simplified if the number of states is increased. The techniques for reducing the number of states presented in this section are merely for illustrative purpose.

The number of states can be reduced by using the concept of equivalent states. Two states are equivalent if both states provide the same outputs for identical inputs. One of the states can be eliminated if two states are equivalent. Thus, the number of states can be reduced.

For example, consider the state diagram of Figure 5.19. Each state is represented by a circle with transition to the next state based on either an input of 0 or 1 generating an output.

Next, consider that a string of input data bits (d) in the sequence 0100111101 is applied at state V of the synchronous sequential circuit. For the given input sequence, the output and the state sequence can be obtained as follows:

State	*V*	*V*	*W*	*Y*	*Z*	*W*	*V*	*W*	*V*	*V*	*W*
Input	0	1	0	0	1	1	1	1	0	1	
Output	0	1	0	0	1	0	1	0	0	1	

With the sequential circuit in initial state V, a 0 input generates a 0 output and the

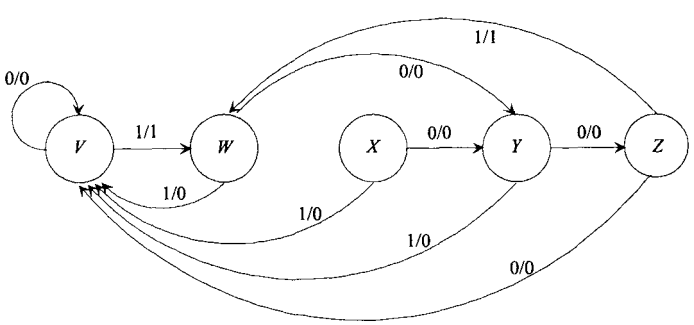


FIGURE 5.19 State diagram for minimization

TABLE 5.3 State table for minimization of states

Present State	Next State		Output	
	d=0	*d=1*	*d=0*	*d=1*
V	*V*	*W*	0	1
W	*Y*	*V*	0	0
X	*Y*	*V*	0	0
Y	*Z*	*V*	0	0
Z	*V*	*W*	0	1

TABLE 5.4 Replacing states by their equivalents

Present State	Next State		Output	
	d=0	*d=1*	*d=0*	*d=1*
V	*V*	*W*	0	1
W	*Y*	*V*	0	0
~~*X*~~	*Y*	*V*	0	0
Y	~~*Z*~~ *V*	*V*	0	0
~~*Z*~~	*V*	*W*	0	1

circuit stays in state *V*, whereas in state *V*, an input of 1 produces an output 1 and the circuit will move to the next state *W*. In state *W* and input = 0, the output is 0 and the next state is *Y*. The process thus continues.

The state table shown in Table 5.3 can be obtained from the state diagram in Figure 5.19. Next, the equivalent states will be determined to reduce the number of states. *V* and *Z* are equivalent because they have the same next states of *V* and *W* with identical inputs $d = 0$ and $d = 1$. Similarly, *W* and *X* are equivalent states. Table 5.4 shows the process of replacing of a state by its equivalent.

Because *V* and *Z* are equivalent, one of the states can be eliminated; *Z* is removed. Also, *W* and *X* are equivalent, so one of the states can be removed; *X* is thus eliminated in the state table. The row with present states *X* and *Z* is also eliminated. If they appear in the next state columns, they must be replaced by their equivalent states. In our case, the row for state *Y* contains *Z* in the next column. This is replaced by its equivalent state *V*. By inspecting the modified state table further, no more equivalent states are found. The state table after elimination of equivalent states is shown in Table 5.5.

Note that the original state diagram in Figure 5.19 requires five states. Figure 5.20 shows the reduced form of the state diagram with only three states. Three flip-flops are

TABLE 5.5 State table after the elimination of equivalent states

Present State	Next State		Output	
	d=0	*d=1*	*d=0*	*d=1*
V	*V*	*W*	0	1
W	*Y*	*V*	0	0
Y	*V*	*V*	0	0

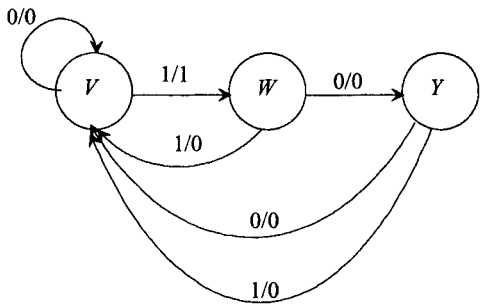


FIGURE 5.20 Reduced form of the state diagram

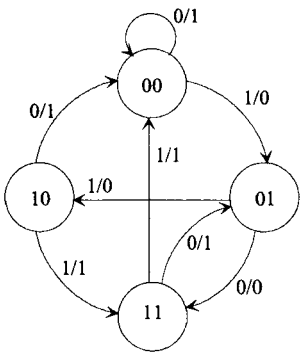


FIGURE 5.21 State diagram for Example 5.2

required to represent five states whereas two flip-flops will represent three states. Thus, one flip-flop is eliminated and the complexity of implementation may be reduced. Note that a synchronous sequential circuit can be minimized by determining the equivalent states, provided the designer is only concerned with the output sequences due to input sequences.

5.9 Design of Synchronous Sequential Circuits

The procedure for designing a synchronous sequential circuit is a three-step process as follows:

1. Derive the state table and state diagram from the problem definition. If the state diagram is given, determine the state table.
2. Obtain the minimum form of the Boolean equations for flip-flop inputs and outputs, if any, using K-maps.
3. Draw the logic diagram. Note that a combinational circuit is designed using a truth table whereas the synchronous sequential circuit design is based on the state table.

Example 5.2

Design a synchronous sequential circuit for the state diagram of Figure 5.21 using D flip-flops.

Solution

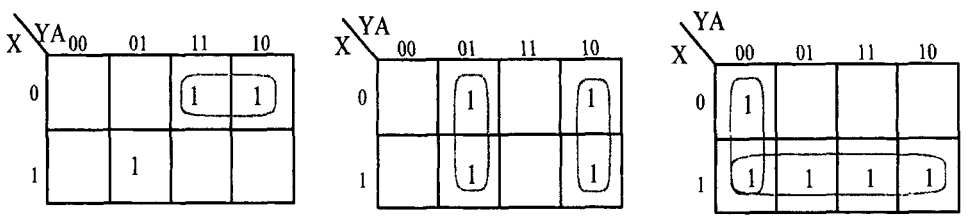
Step 1: Derive the state table. The state table is derived from the state diagram (Figure

TABLE 5.6 State Table for Example 5.2

Present State		Input	Next State		Flip Flop Inputs		Output
X	Y	A	$X+$	$Y+$	D_X	D_Y	Z
0	0	0	0	0	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	0	1
1	0	1	1	1	1	1	1
1	1	0	0	1	0	1	1
1	1	1	0	0	0	0	1

5.21) and the excitation table [Figure 5.12(c)] of the D flip-flop. Table 5.6 shows the state table.

The state table is obtained directly from the state diagram. In the state table, the next states are same as the flip-flop inputs because D flip-flops are used. This is evident from the excitation table of Figure 5.12(c).



(a) K-map for D_X
$D_X = X\overline{Y}A + \overline{X}Y$

(b) K-map for D_Y
$D_Y = \overline{Y}A + Y\overline{A} = Y \oplus A$

(c) K-map for Z
$Z = \overline{Y}\,\overline{A} + X$

FIGURE 5.22 K-maps for Example 5.2

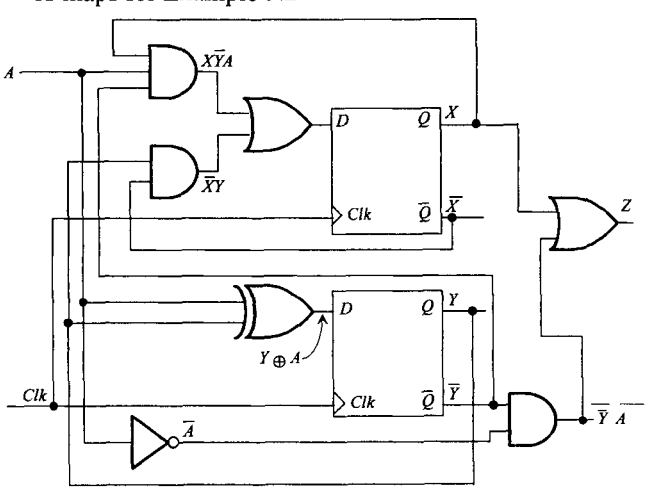


FIGURE 5.23 Logic diagram for Example 5.2

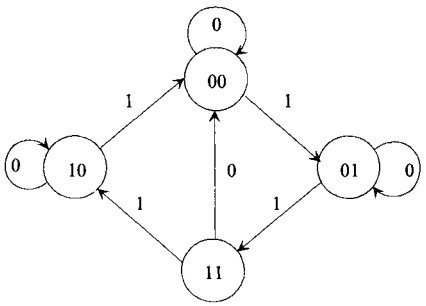


FIGURE 5.24 State diagram for Example 5.3

TABLE 5.7 State and Excitation Tables for Example 5.3

TABLE 5.7 (a) Excitation Table of JK flip-flop from Figure 5.11c

Q	$Q+$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

TABLE 5.7 (b) State Table for Example 5.2

Present State		Input	Next State		Flip Flop Inputs			
X	Y	A	$X+$	$Y+$	J_X	K_X	J_Y	K_Y
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	0	1	0	X	X	0
0	1	1	1	1	1	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	0	0	X	1	0	X
1	1	0	0	0	X	1	X	1
1	1	1	1	0	X	0	X	1

Step 2: Obtain the minimum forms of the equations for the flip-flop inputs and the output. Using K-maps and the output, the equations for flip-flop inputs are simplified as shown in Figure 5.22.

Step 3: Draw the logic diagram. The logic diagram is shown in Figure 5.23.

Example 5.3

Design a synchronous sequential circuit for the state diagram of Figure 5.24 using JK flip-flops.

Solution

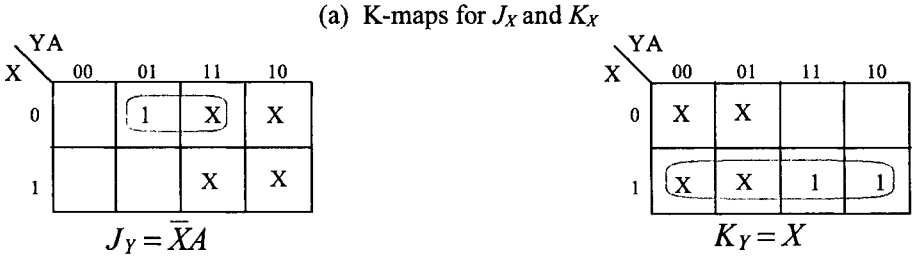
Step 1: Derive the state table. The state table can be directly obtained from the state diagram (Figure 5.24) and the excitation table [Figure 5.11(c)]. Table 5.7 shows the state table. For convenience, the excitation table of the JK flip-flop of Figure 5.11(c) is also included.

Let us explain how the state table is obtained. The input A is 0 or 1 at each state, so the left three columns show all eight combinations for X, Y, and A. The next state column is obtained from the state diagram. The flip-flop inputs are then obtained using the excitation table for the JK flip-flop. For example, consider the top row. From the state diagram, the present state (00) remains in the same state (00) when input $A = 0$ and the clock pulse is applied. The output of flip-flop X goes from 0 to 0 and the output of flip-flop Y goes from 0 to 0. From the excitation table of the JK flip-flop, $J_x = 0$, $K_x = X$, $J_x = 0$, and $K_x = X$. The other rows are obtained similarly.

Step 2: Obtain the minimum forms of the equations for the flip-flop inputs. Using K-maps, the equations for flip-flop inputs are simplified as shown in Figure 5.25.

Step 3: Draw the logic diagram as shown in Figure 5.26.

(a) K-maps for J_X and K_X



(b) K-maps for J_Y and K_Y

FIGURE 5.25 K-maps for Example 5.3

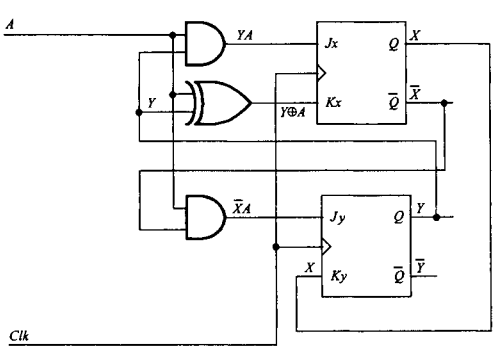


FIGURE 5.26 Logic Diagram for Example 5.3

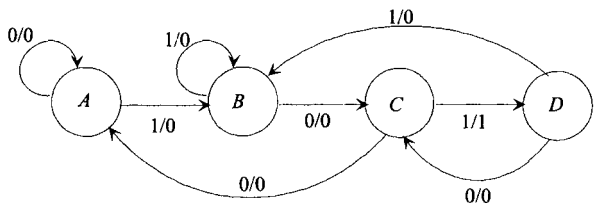


FIGURE 5.27 State Diagram for Example 5.4

Example 5.4

Design a synchronous sequential circuit with one input X and an output Z. The input X is a serial message and the system reads X one bit at a time. The output $Z = 1$ whenever the pattern 101 is encountered in the serial message. For example,

If input: 0 0 1 0 1 0 1 1 1 0 1 0 0 0 1 0 1

then output: 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1

Use T flip-flops.

Solution

Step 1: Derive the state diagram and the state table.

Figure 5.27 shows the state diagram. In this diagram each node represents a state. The labeled arcs (lines joining two nodes) represent state transitions. For example, when the system is in state C, if it receives an input 1, it produces an output 1 and makes a transition to the state D after the clock. Similarly, when the system is in state C and receives a 0 input, it generates a 0 output and moves to state A after the clock. This type of sequential circuit is called a *Mealy machine* because the output generated depends on both the input X and the present state of the system. It should be emphasized that each state in the state diagram actually performs a bookkeeping operation; these operations are summarized as follows

State	Interpretation
A	Looking for a new pattern
B	Received the first 1
C	Received a 1 followed by a 0
D	Recognized the pattern 101

The state diagram can be translated into a *state table,* as shown in Table 5.8. Each state can be represented by the binary assignment as follows:

Symbolic State	Binary State	
	y_1	y_0
A	0	0
B	0	1
C	1	1
D	1	0

TABLE 5.8 State Table for Example 5.4

Present State	Next State		Output Z	
	X=0	*X=1*	*X=0*	*X=1*
A	*A*	*B*	0	0
B	*C*	*B*	0	0
C	*A*	*D*	0	1
D	*C*	*B*	0	0

TABLE 5.9 Modified State Table for Example 5.4

Present State		Next State		Output Z	
y_1	y_0	$y_1 + y_0^+$ *X=0*	$y_1 + y_0^+$ *X=1*	Input *X=0*	Input *X=1*
0	0	0 0	0 1	0	0
0	1	1 1	0 1	0	0
1	1	0 0	1 0	0	1
1	0	1 1	0 1	0	0

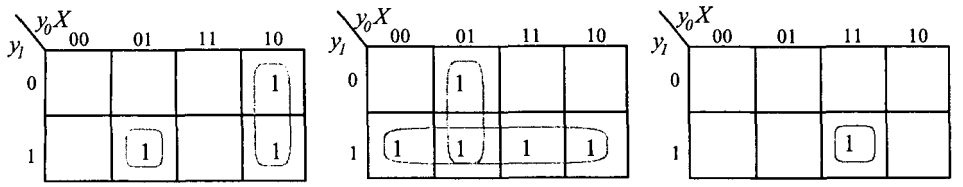
The state table in Table 5.8 can be modified to reflect this state assignment, as illustrated in Table 5.9. Note that the excitation table actually describes the required excitation for a particular state transition to occur. For example, with respect to a T flip-flop, for the transition 0 → 1 or 1 → 0, a 1 must be applied to the *T* input. Similarly, for transitions 0 → 0 or 1 → 1 (that is, no change of state), the *T* input must be made 0. Using this excitation table, the flip-flop input equations can be derived as illustrated in Table 5.9.

In this figure, the entries corresponding to the flip-flop inputs T_{y1} and T_{y0} are directly derived using the T flip-flop excitation table. For example, consider the present state y_1y_0 = 00. When the input X = 1, the next state is 01. This means that flip-flop y_1 should not change its states and flip-flop y_0 must change its state to 1. It follows that T_{y1} = 0 (because a 0 → 0 transition is required) and T_{y0} = 1 (because a 0 → 1 transition is required). The other entries for T_{y1} and T_{y0} may be obtained in a similar manner.

The state table of Table 5.9 is obtained using the excitation table for T flip-flop of Figure 5.13(c) redrawn as follows:

Present State		Input	Next State		Flip Flop Inputs		Ouput
*y*1	*y*0	*X*	y^+_1	y^+_0	T_{y1}	T_{y0}	*Z*
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	0	0	0
1	0	0	1	1	0	1	0
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	0
1	1	1	1	0	0	1	1

Step 2: Derive the minimum forms of the equations for the flip-flop inputs and the output. Using K-maps, the simplified equations for the flip-flops inputs and the output can be



(a) K-map for T_{y_1}
$T_{y_1} = y_0\overline{X} + y_1\overline{y_0}X$

(b) K-map for T_{y_0}
$T_{y_0} = y_1 + \overline{y_0}X$

(c) K-map for Z
$Z = y_1y_0X$

FIGURE 5.28 K-maps for Example 5.4

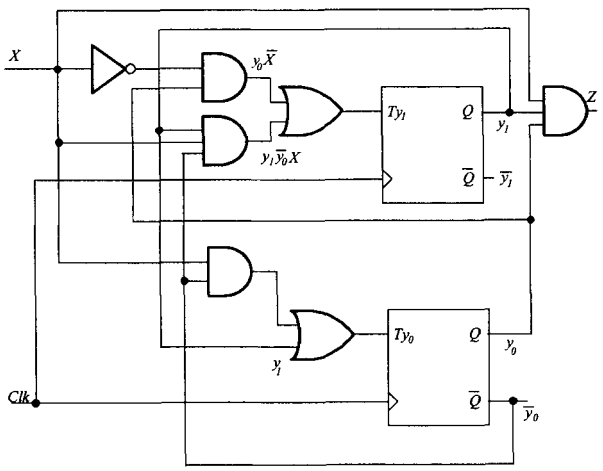


FIGURE 5.29 Logic Diagram for Example 5.4

obtained as shown in Figure 5.28.

Step 3: Draw the logic diagram as shown in Figure 5.29.

5.10 Design of Counters

A counter is a synchronous sequential circuit that moves through a predefined sequence of states upon application of clock pulses. A binary counter, which counts binary numbers in sequence at each clock pulse, is the simplest example of a counter. An n-bit binary counter contains n flip-flops and can count binary numbers from 0 to 2^{n-1}. Other binary counters may count in an arbitrary manner in a nonbinary sequence. The following examples will illustrate the straight binary sequence and nonbinary sequence counters.

Example 5.5

Design a two-bit counter to count in the sequence 00, 01, 10, 11, and repeat. Use T flip-flops.

Solution

Step 1: Derive the state diagram and the state table.

Figure 5.30 shows the state diagram. Note that state transition occurs at the clock pulse. No state transitions occurs if there is no clock pulse. Therefore, the clock pulse does not appear as an input. Table 5.10 shows the state table.

The excitation table of the T flip-flop is used for deriving the state table. For example, consider the top row. The state remains unchanged ($a_1 = 0$ and $a_{1+} = 0$) requiring

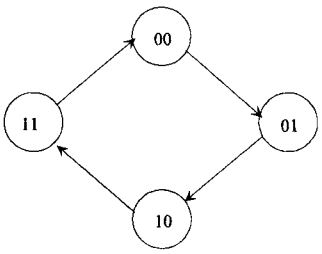


FIGURE 5.30 State Diagram for Example 5.5

TABLE 5.10 State table for Example 5.5

Present State		Next State		Flip Flop inputs	
a_1	a_0	a_1+	a_0+	T_{A1}	T_{A0}
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

a T input of 0 and thus $T_{A1} = 0$. a_0 is complemented from the present state to the next state, and thus $T_{A0} = 0$.

Step 2: Derive the minimum forms of the equations for the flip-flop inputs.

Using K-maps, the simplified equations for the flip-flop inputs can be obtained as shown in Figure 5.31.

Step 3: Draw the logic diagram as shown in Figure 5.32.

(a) K-map for T_{A_1}
$T_{A_1} = a_0$

(b) K-map for T_{A_0}
$T_{A_0} = 1$

FIGURE 5.31 K-maps for Example 5.5

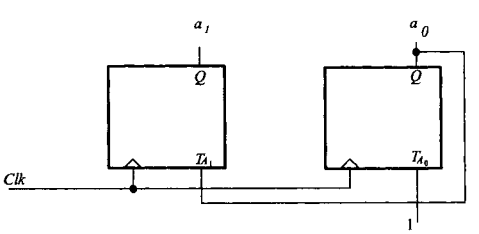


FIGURE 5.32 Logic Diagram for 2-bit Counter of Example 5.5

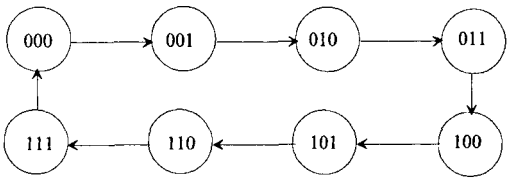


FIGURE 5.33 State Diagram for Example 5.6

TABLE 5.11 JK ff excitation table and State Table for Example 5.6

TABLE 5.11(a) Excitation Table of JK Flip-flop

Q	$Q+$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

TABLE 5.11(b) State Table for Example 5.6

Present State			Next State			Flip-Flop Inputs					
a_2	a_1	a_0	a_2+	a_1+	a_0+	Ja_2	Ka_2	Ja_1	Ka_1	Ja_0	Ka_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Example 5.6

Design a three-bit counter to count in the sequence 000 through 111, return to 000 after 111, and then repeat the count. Use JK flip-flops.

Solution

Step 1: Derive the state diagram and the state table.

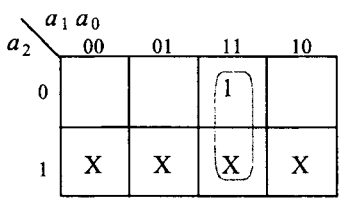
Figure 5.33 shows the state diagram. Table 5.11 shows the JK ff excitation table, and the state table. Consider the top row. The present state of a_2 changes from 0 to 0 at the clock, a_1 changes from 0 to 0, and a_0 changes from 0 to 1. From the JK flip-flop excitation table, for these transitions, $Ja_2 = 0$, $Ka_2 = X$, $Ja_1 = 0$, $Ka_1 = X$, and $Ja_0 = 1$, $Ka_0 = X$.

Step 2: Derive the minimum forms of the equations for the flip-flop inputs. Using K-maps, the simplified equations for the flip-flop inputs can be obtained as shown in Figure 5.34.

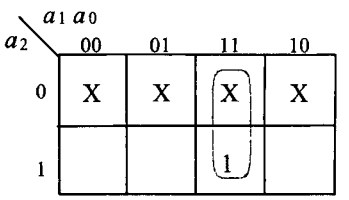
Step 3: Draw the logic diagram as shown in Figure 5.35.

Example 5.7

Design a 3-bit counter that will count in the sequence 000, 010, 011, 101, 110, 111, and repeat the sequence. The counter has two unused states. These are 001 and 100. Implement the counter as a self-correcting such that if the counter happens to be in one of the unused states (001 or 100) upon power-up or due to error, the next clock pulse puts it in one of

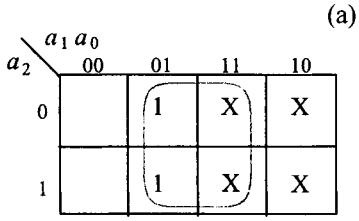


$Ja_2 = a_1a_0$

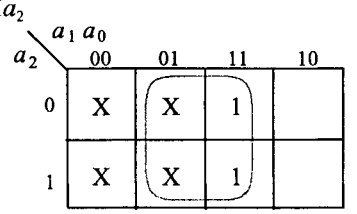


$Ka_2 = a_1a_0$

(a) K-Maps for Ja_2 and Ka_2

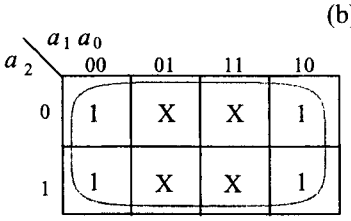


$Ja_1 = a_0$

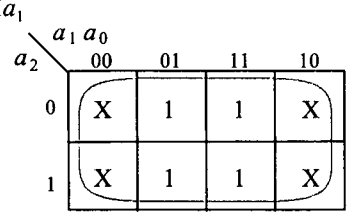


$Ka_1 = a_0$

(b) K-Maps for Ja_1 and Ka_1



$Ja_0 = 1$



$Ka_0 = 1$

(c) K-Maps for Ja_0 and Ka_0

FIGURE 5.34 K-Maps for Example 5.6

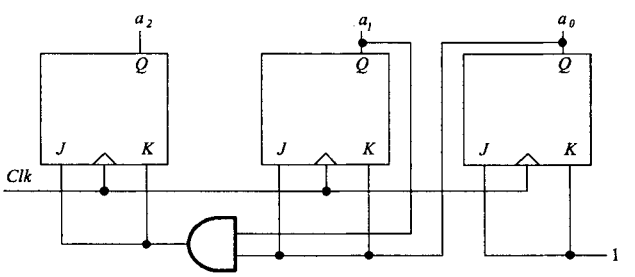


FIGURE 5.35 Logic Diagram for Example 5.6

the valid states and the counter provides the correct count. Use T Flip-flops. Note that the initial states of the flip-flops are unpredictable when power is turned ON. Therefore, all the unused (don't care) states of the counter should be checked to ensure that the counter eventually goes into the desirable counting sequence. This is called a self-correcting counter.

Solution

Step 1: Derive the state diagram and the state table. Figure 5.36 shows the state diagram. Note that in the state diagram it is shown that if the counter goes to an invalid state such as 001 upon power-up, the counter will then go to the valid state 011 and will count correctly. Similarly, for the invalid state 100, the counter will be in state 111

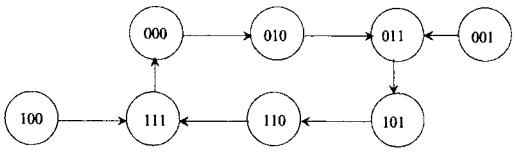


FIGURE 5.36 State Diagram for Example 5.7

TABLE 5.12 T-ff excitation table and State Table for Example 5.7

TABLE 5.12(a) Excitation Table for T Flip-Flop

Q	$Q+$	T
0	0	0
0	1	1
1	0	1
1	1	0

TABLE 5.12 (b) State Table for Example 5.7

Present State			Next State			Flip Flop Inputs		
a_2	a_1	a_0	a_2+	a_1+	a_0+	Ta_2	Ta_1	Ta_0
0	0	0	0	1	0	0	1	0
0	1	0	0	1	1	0	0	1
0	1	1	1	0	1	1	1	0
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

and the correct count will continue. This self-correcting feature will be verified from the counter's state table using T flip-flops as shown in Table 5.12.

Step 2: Derive the minimum forms of the equations for the flip-flop inputs.

Using K-maps, the simplified equations for the flip-flop inputs can be obtained, as shown in Figure 5.37. The unused states 001 and 100 are invalid and can never occur, so they are don't care conditions.

Now, let us verify the self-correcting feature of the counter. The flip-flop input equations are

$$Ta_2 = a_1 a_0$$

$$Ta_1 = \overline{a_1} + a_0$$

$$Ta_0 = a_2 + a_1\overline{a_0}$$

Suppose that the counter is in the invalid state 001 upon power-up or due to error, therefore, in this state, $a_2 = 0$, $a_1 = 0$, and $a_0 = 1$. Substituting these values in the flip-flop input equations, we get

$$Ta_2 = 0 \cdot 1 = 0$$

$$Ta_1 = \overline{0} + 1 = 1$$

$$Ta_0 = 0 + 0 \cdot \overline{1} = 0$$

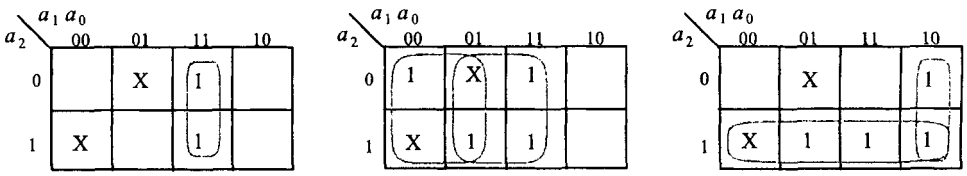


FIGURE 5.37 K-maps for example 5.7

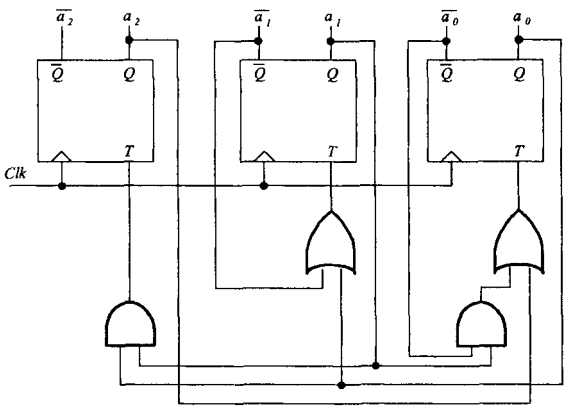


FIGURE 5.38 Logic Diagram for Example 5.7

Note that with $a_2a_1a_0 = 001$ and $Ta_2Ta_1Ta_0 = 010$, the state changes from 001 to 011. Therefore, the next state will be 011. The correct count will resume. Next, if the flip-flop goes to the invalid state 100 due to error or when power is turned ON. Substituting $a_2 = 1$, $a_1 = 0$, and $a_0 = 0$ gives

$$Ta_2 = 0 \bullet 0 = 0$$

$$Ta_1 = \overline{0} + 0 = 1$$

$$Ta_0 = 1 + 0 \bullet \overline{0} = 1$$

Note that with $a_2a_1a_0 = 100$ and $Ta_2Ta_1Ta_0 = 011$, the state changes from 100 to 111. Hence, the next state for the counter will be 111. The correct count will continue. Therefore, the counter is self-correcting.

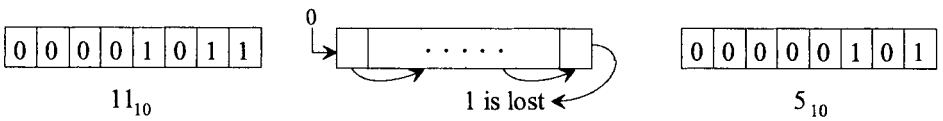
Step 3: Draw the logic diagram as shown in Figure 5.38.

5.11 Examples of Synchronous Sequential Circuits

Typical examples include registers, modulo-*n* counters and RAMs (Random Access Memories). They play an important role in the design of digital systems, especially computers. Veriolog and VHDL descriptions along with simulation results of typical synchronous. Sequential circuits are provided in Appendices I and J respectively.

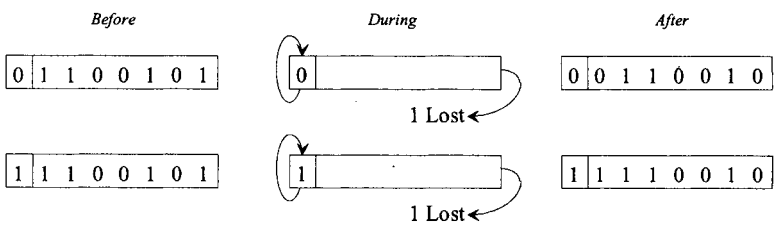
5.11.1 Registers

A register contains a number of flip-flops for storing binary information in a computer. The register is an important part of any CPU. A CPU with many registers reduces the number of accesses to the main memory, therefore simplifying the programming task and shortening execution time. A general-purpose register (GPR) is designed in this section. The primary task of the GPR is to store address or data for an indefinite amount of time, then to be able to retrieve the data when needed. A GPR is also capable of manipulating the stored data by shift left or right operations. Figure 5.39 contains a summary of typical shift operations. In logical shift operation, a bit that is shifted out will be lost, and the vacant position will be filled with a 0. For example, if we have the number $(11)_{10}$, after right shift, the following occurs:



It must be emphasized that a logical left or right shift of an unsigned number by n positions implies multiplication or division of the number by 2^n, respectively, provided that a 1 is not shifted out during the operation.

In the case of true arithmetic left or right shift operations, the sign bit of the number to be shifted must be retained. However, in computers, this is true for right shift and not for left shift operation. For example, if a register is shifted right arithmetically, the most significant bit (MSB) of the register is preserved, thus ensuring that the sign of the number will remain unchanged. This is illustrated next:



There is no difference between arithmetic and logical left shift operations. If

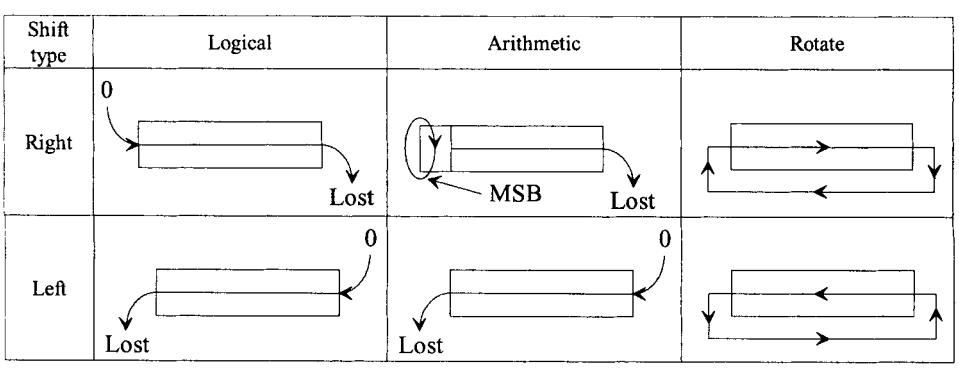
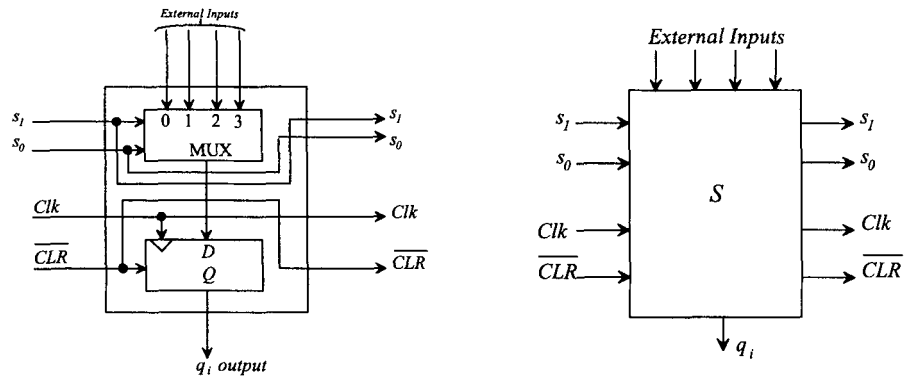


FIGURE 5.39 Summary of Typical Shift Operations



(a) Internal Organization of the Basic Cell *S*

(b) Block Diagram of the Basic Cell *S*

FIGURE 5.40 A Basic Cell for Designing a GPR

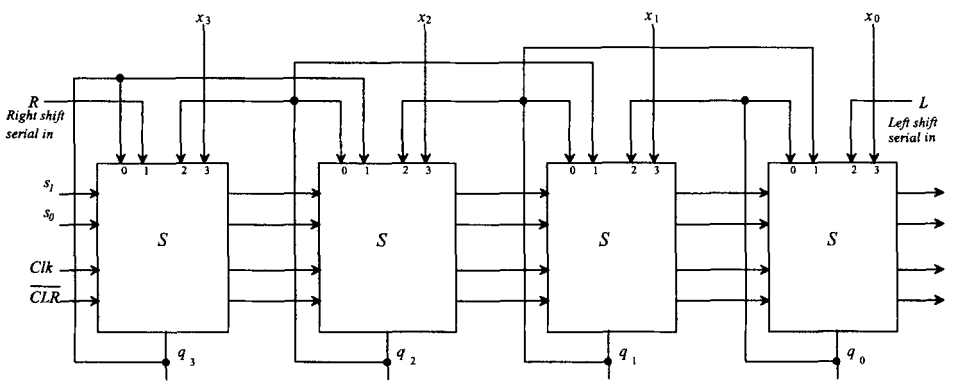


FIGURE 5.41 A 4-bit General Register

the most significant bit changes from 0 to 1, or vice versa, in an arithmetic left shift, the result is incorrect and the computer sets the overflow flag to 1. For example, if the original value of the register is $(3)_{10}$, the results of two successive arithmetic left shift operations are interpreted as follows:

Original	*After first shift*	*After second shift*
$0011_2 = (3)_{10}$	$0110_2 = (6)_{10}$	$1100_2 = (-4)$
	$3 \times 2 = 6$, correct	$6 \times 2 = 12$, not -4. incorrect

To design a GPR, first let us propose a basic cell *S*. The internal organization of the *S* cell is shown in Figure 5.40. A 4-input multiplexer selects one of the external inputs as the D flip-flop input, and the selected input appears as the flip-flop output *Q* after the clock pulse. The $\overline{CLR}$ input is an asynchronous clear input, and whenever this input is asserted (held low), the flip-flop is cleared to zero. Using the basic cell *S* as the building

TABLE 5.13 Truth Table for the General Register

Selection Input		Clock input	Clear Input	Operation
s_1	s_0	*Clk*	$\overline{CLR}$	
X	X	X	0	Clear
0	0	⌐↑	1	No Operation
0	1	⌐↑	1	Shift Right
1	0	⌐↑	1	Shift Left
1	1	⌐↑	1	Parallel Load

X means "don't care"

block, a 4-bit GPR can be designed. Its schematic representation is shown in Figure 5.41.

The truth table illustrating the operation of this register is shown in Table 5.13. This table shows that manipulation of the selection inputs S_1 and $S_0 = 11$, the external inputs x_3 through x_0 are selected as the D inputs for the flip-flop, the output q_i will follow the input x_i after the clock. By choosing the correct values for the serial shift inputs R and L, logical, arithmetic, or rotating shifts can be achieved.

This register can be loaded with any desired data in a serial fashion. For example, after four successive right shift operations, data $a_3\ a_2\ a_1\ a_0$ will be loaded into the register if the register is set in the right shift mode and the required data $a_3\ a_2\ a_1\ a_0$ is applied serially to input R.

5.11.2 Modulo-*n* Counters

The modulo-n counter counts in a sequence and then repeats the count. Modulo-n counters can be used to generate timing signals in a computer. The control unit inside the CPU of a computer translates instructions. The control unit utilizes timing signals that determines

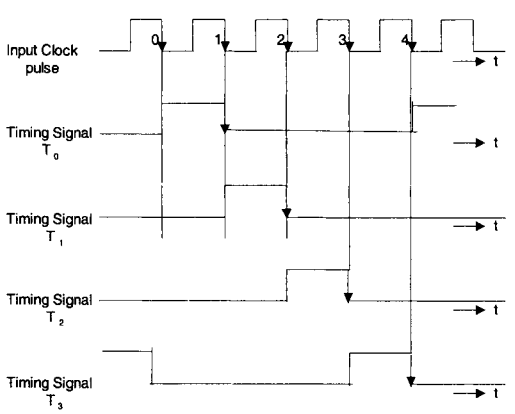


FIGURE 5.42 Timing Signals

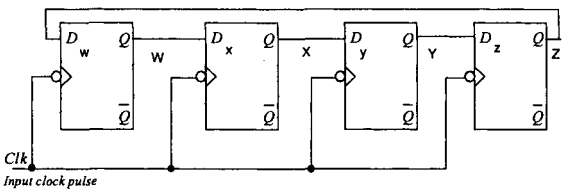


FIGURE 5.43 Four-bit Ring Counter

the time sequences in which the operations required by an instruction are executed. These timing signals shown in Figure 5.42 can be generated by a special modulo-*n* counter called the ring counter. For proper operation, a ring counter must be initialized with one flip-flop in the high state (Q=1) and all other flip-flops in the zero state (Q=0).

An *n*-bit ring counter transfers a single bit among the flip-flops to provide *n* unique states. Figure 5.43 shows a 4-bit ring counter. Note that the ring counter requires no decoding but contains n flip-flops for an *n*-bit ring counter. The circuit will count in the sequence 1000, 0100, 0010, 0001, and repeat. Although the circuit does not count in the usual binary counting sequence, it is still called a counter because each count corresponds to a unique set of flip-flop states. The state table for the 4-bit ring counter is provided below:

Present State				Next State				FF Inputs			
W	X	Y	Z	W+	X+	Y+	Z+	Dw	Dx	Dy	Dz
1	0	0	0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	0	1	0	0	0

From the above, using the present states along with the unused present states (not shown above) as don't cares, the following equations can be obtained using four K-maps (one for each FF input): `Dw=Z, Dx=W, Dy=X, Dz = Y.` This circuit is also known as a *circular shift register,* because the least significant bit shifted is not lost. This is the simplest shift-register counter. Thus, the schematic of Figure 5.43 can be obtained.

The main advantages of this circuit are design simplicity and the ability to generate timing signals without a decoder. Nevertheless, *n* flip-flops are required to generate *n* timing signals. This approach is not economically feasible for large values of *n*. To generate timing signals economically, a new approach is used. A modulo-2^n counter is first designed using *n* flip-flops. The *n* outputs from this counter are then connected to a *n*-to-2^n decoder as inputs to generate 2^n timing signals. The circuit depicted in Figure 5.44 shows how to generate four timing signals using a modulo-4 counter and a 2-to-4 decoder. In the preceding circuit, the Boolean equation for each timing signal can be derived as

$$T_0 = \overline{A}\,\overline{B}$$
$$T_1 = \overline{A}\,B$$
$$T_2 = A\,\overline{B}$$
$$T_3 = A\,B$$

These equations show that four 2-input AND gates are needed to derive the timing

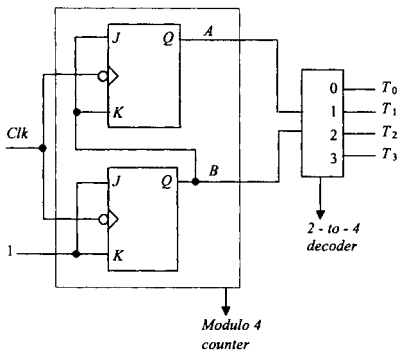


FIGURE 5.44 Modulo-4 Counter with a Decoder

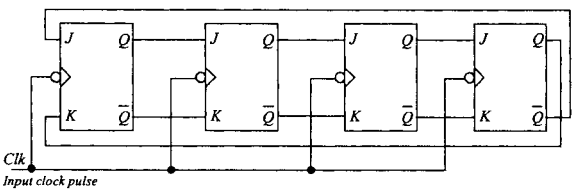


FIGURE 5.45 Four-bit Johnson Counter

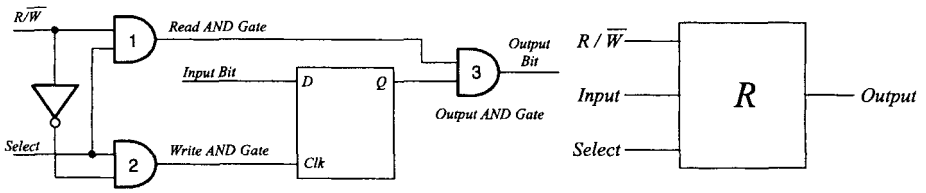
signals (assuming single-level decoding). The main advantage of this approach is that 2^n timing signals using only n flip-flops are generated. In this method, though, 2^n (n-input) AND gates are required to decode the n-bit output from the flip-flops into 2^n different timing signals. Yet the ring counter approach requires 2^n flip-flops to accomplish the same task.

Typical modulo-n counters provide trade-offs between the number of flip-flops and the amount of decoding logic needed. The binary counter uses the minimum number of flip-flops but requires a decoder. On the other hand, the ring counter uses the maximum number of flip-flops but requires no decoding logic. The Johnson counter (also called the Switch-tail counter or the Mobius counter) is very similar to a ring counter. Figure 5.45 shows a 4-bit Johnson counter using JK flip-flops. Note that the $\overline{Q}$ output of the right-hand flip-flop is connected to the J input of the leftmost flip-flop while the Q output of the rightmost flip-flop is connected to the K input of the leftmost flip-flop.

A Johnson counter requires the same hardware as a ring counter of the same size but can represent twice as many states. Assume that the flip-flops are initialized at 1000. The counter will count in the sequence 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000 and repeat.

5.11.3 Random-Access Memory (RAM)

As mentioned before, a RAM is read/write volatile memory. RAM can be classified into two types: static RAM (SRAM) and dynamic RAM (DRAM). A static RAM stores each bit in a flip-flop whereas the dynamic RAM stores each bit as charge in a capacitor. As long as power is available, the static RAM retains information. Because the capacitor can hold charge for a few milliseconds, the dynamic RAM must be refreshed every few milliseconds. This means that a circuit must rewrite that stored bit in a dynamic RAM every few milliseconds. Let us now discuss a typical SRAM implementation using D flip-flops. Figure 5.46 shows a typical RAM cell.



(a) A one-bit RAM (R)

(b) Block diagram of the one-bit RAM

FIGURE 5.46 A typical SRAM cell

In Figure 5.46(a), R/$\overline{W}$ = 1 means READ whereas R/$\overline{W}$ = 0 indicates a WRITE operation. Select = 1 indicates that the one-bit RAM is selected. In order to read the cell, R/$\overline{W}$ is 1 and select = 1. A 1 appears at the input of AND gate 3. This will transfer Q to the output. This is a READ operation. Note that the inverted R/$\overline{W}$ to the input of AND gate 2 is 0. This will apply a 0 at the input of the CLK input of the D flip-flop. The output of the D flip-flop is unchanged. In order to write into the one-bit RAM, R/$\overline{W}$ must be zero. This will apply a 1 at the input of AND gate 2. The output of AND gate 2 (CLK input) is 1. The D input is connected to the value of the bit (1 or 0) to be written into the one-bit RAM. With CLK = 1, the input bit is transferred at the output. The one-bit RAM is, therefore, written into with the input bit. Figure 5.47 shows a 4 x 2 RAM. It includes 8 RAM cells providing 2-bit output and 4 locations.

The RAM contains a 2 x 4 decoder and 8 RAM cells implemented with D flip-flops and gates. In contrast, a ROM consists of a decoder and OR gates. The four locations (00, 01, 10, 11) in the RAM are addressed by 2 bits (A_1, A_0). In order to read from location 00, the address A_1A_0 = 00 and R/$\overline{W}$ = 1. The decoder selects O_0 high. R/$\overline{W}$ = 1 will apply 0 at the clock inputs of the two RAM cells of the top row and will apply 1 at the inputs of the output AND gates, thus transferring the outputs of the two D flip-flops to the inputs of the two OR gates. The other inputs of the OR gate will be 0. Thus, the outputs of the two RAM cells of the top row will be transferred to DO_1 and DO_0, performing a READ operation. On the other hand, consider a WRITE operation: The 2-bit data to be written is presented

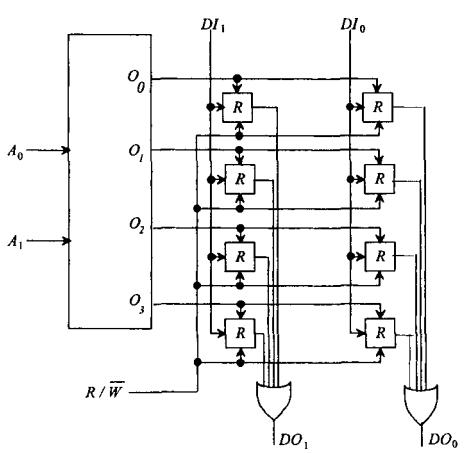


FIGURE 5.47 4 x 2 RAM

at DI_1 DI_0. Suppose $A_1A_0 = 00$. The top row is selected ($O_0 = 1$). Input bits at DI_1 and DI_0 will respectively be applied at the inputs of the D flip-flops of the top row. Because $R/\overline{W} = 0$, the clock inputs of both the D flip-flops of the top row are 1; thus, the *D* inputs are transferred to the outputs of the flip-flops. Therefore, data at DI_1 DI_0 will be written into the RAM.

5.12 Algorithmic State Machines (ASM) Chart

The performance of a synchronous sequential circuit (also referred to as a state machine) can be represented in a systematic way by using a flowchart called the Algorithmic State Machines (ASM) chart. This is an alternative approach to the state diagram. In the previous

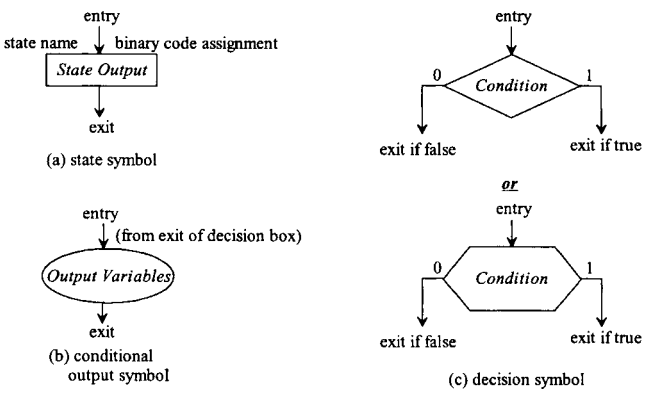


FIGURE 5.48 Symbols for an ASM Chart

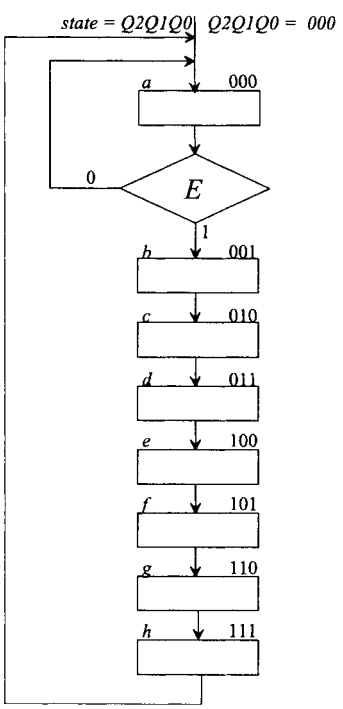


FIGURE 5.49 An ASM Chart for a 3-bit Counter with Enable Input

sections, it was shown how state diagrams could be used to design synchronous sequential circuit. An ASM chart can sometimes be used along with the state diagram for designing a synchronous sequential circuit. An ASM chart is similar to a flowchart for a computer program. The main difference is that the flowchart for a computer program is translated into software whereas an ASM chart is used to implement hardware. An ASM chart specifies the sequence of operations of the state machine along with the conditions required for their execution. Three symbols are utilized to develop the ASM chart: the state symbol, the decision symbol, and the conditional output symbol (see Figure 5.48).

The ASM chart utilizes one state symbol for each state. The state symbol includes the state name, binary code assignment, and outputs (if any) that are asserted during the specified state. The decision symbol indicates testing of an input and then going to an exit if the condition is true and to another exit if the condition is false. The entry of the conditional output symbol is connected to the exit of the decision symbol.

The ASM chart and the state diagram are very similar. Each state in a state diagram is basically similar to the state symbol. The decision symbol is similar to the binary information written on the lines connecting two states in a state diagram. Figure 5.49 shows an example of an ASM chart for a modulo-7 counter (counting the sequence 000, 001, ..., 111 and repeat) with an enable input. Q_2, Q_1, and Q_0 at the top of the ASM chart represent the three flip-flop states for the 3-bit counter.

Each state symbol is given a symbolic name at the upper left corner along with a binary code assignment of the state at the upper right corner. For example, the state '*a*' is assigned with a binary value of 000. The enable input E can only be checked at state *a*, and the counter can be stopped if $E = 0$; the counter continues if $E = 1$. This is illustrated by the decision symbol. Figure 5.50 shows the equivalent state diagram of the ASM chart for the 3-bit counter.

The ASM chart describes the sequence of events and the timing relationship between the states of a synchronous sequential circuit and the operations that occur for transition from one state to the next. An arbitrary ASM chart depicted in Figure 5.51 illustrates this. The chart contains three ASM blocks. Note that an ASM block must contain one state symbol and may include any number of decisions and conditional output symbols connected to the exit. The three ASM blocks are the ASM block for T_0 surrounded by the dashed lines and the simple ASM block defined by T_1 and T_2. Figure 5.52 shows the state diagram.

From the ASM chart of Figure 5.51, there are three states: T_0, T_1, and T_2. A ring counter can be used to generate these timing signals. During T_0, register X is cleared and flip-flop A is checked. If $A = 0$, the next state will be T_1. On the other hand, if $A = 1$, the circuit increments register X by 1 and then moves to the next state, T_2. Note that the following operations are performed by the circuit during state T_0:

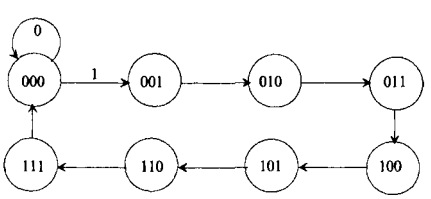


FIGURE 5.50 State Diagram for the 3-bit Counter

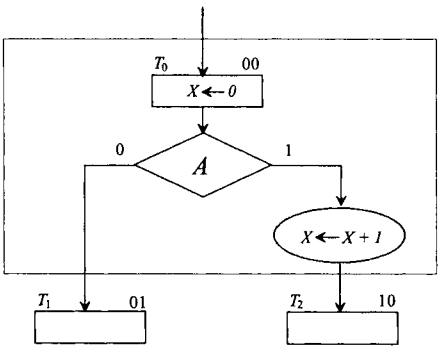


FIGURE 5.51 ASM Chart illustrating timing relationships between states

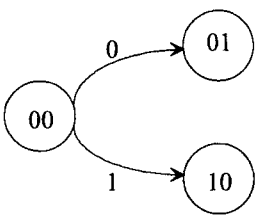


FIGURE 5.52 State Diagram for the ASM Chart of Figure 5.51

1. Clear register X.
2. Check flip-flop A for 1 or 0.
3. If $A = 1$, increment X by 1.

On the other hand, state machines do not perform any operations during T_1 and T_2. Note that in contrast, state diagrams do not provide any timing relationship between states. ASM charts are utilized in designing the controller of digital systems such as the control unit of a CPU. It is sometimes useful to convert an ASM chart to a state diagram and then utilize the procedures of synchronous sequential circuits to design the control logic.

State Machine Design using ASM chart

As mentioned before, an ASM chart is used to define digital hardware algorithms which can be utilized to design and implement state machines. This section describes a procedure for designing state machines using the ASM chart. This is a three step process as follows:

1. Draw the ASM chart from problem definition.
2. Derive the state transition table representing the sequence of operations to be performed.
3. Derive the logic equations and draw the hardware schematic. The hardware can be designed using either classical sequential design or PLAs as illustrated by the examples provided below.

In the following, a digital system is designed using an ASM chart that will operate as follows:

The system will contain a 2-bit binary counter. The binary counter will count in the sequence 00, 01, 10, and 11. The most significant bit of the binary count XY is X while Y is the least significant bit. The system starts with an initial count of 3. A start signal I (represented by a switch) initiates a sequence of operations. If I = 0, the system stays in the

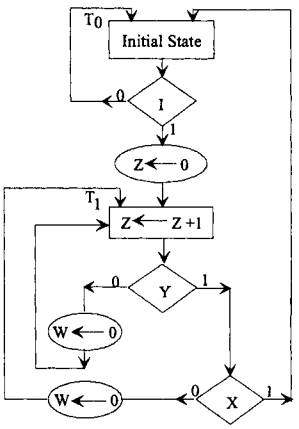


FIGURE 5.53 ASM Chart showing the sequence of operations for the binary counter

TABLE 5.14 State Transition Table

COUNTER		FLIP-FLOP W	CONDITIONS	STATE
X	Y	(Q)		
0	0	1	X = 0, Y = 0	T_0
0	1	0	X = 0, Y = 1	T_1
1	0	0	X = 1, Y = 0	T_1
1	1	1	X = 1, Y = 1	T_0

initial state T_0 with count of 3. On the other hand, I = 1 starts the sequence.

When I = 1, counter Z (represented by XY) is first cleared to zero. The system then moves to state T_1. In this state, counter Z is incremented by 1 at the leading edge of each clock pulse. When the counter reaches 3, the system goes back to the initial stateT_0, and the process continues depending on the status of the start switch I. The counter output will be displayed on a seven-segment display. An LED will be connected at the output of flip-flop W. The system will turn the LED ON for the count sequence 1, 2 by clearing flip-flop W to 0.

The flip-flop W will be preset to 1 in the initial state to turn the LED OFF. This can be accomplished by using input I as the PRESET input of flip-flop W. Use D flip-flops for the system.

***Step 1:* Draw the ASM chart.** Figure 5.53 shows the ASM chart. The symbol T_n is used without its binary value for the state boxes in all ASM charts in this section.

In the ASM chart of Figure 5.53, when the system is in initial state T_0, it waits for the start signal (I) to become HIGH. When I=1, Counter Z is cleared to zero and the system goes to stateT_1. The counter is incremented at the leading edge of each clock pulse. In state T_1 , one of the following possible operations occurs after the next clock pulse transition:

Either, if counter Z is 1 or 2, flip-flop W is cleared to zero and control stays in stateT_1 ;

or

If the Counter Z counts to 3, the system goes back to initial stateT_0.

The ASM chart consists of two states and two blocks. The block associated with T_0 includes one state box, one decision box, and one conditional box. The block in T_1 consists of one state box, two decision boxes and two conditional boxes.

Step 2: **Derive the state transition table representing the sequence of operations.**

One common clock pulse specifies the operations to be performed in every block of an ASM chart. Table 5.14 shows the State Transition Table.

The binary values of the counter along with the corresponding outputs of flip-flop W is shown in the transition table. In state T_0 , if I = 1, Counter Z is cleared to zero (XY = 00) and the system moves from state T_0 to T_1. In state T_1, Counter Z is first incremented to XY = 01 at the leading edge of the clock pulse; Counter Z then counts to XY = 10 at the leading edge of the following clock pulse. Finally, when XY = 11, the system moves to state T_0. The system stays in the initial state T_0 as long as I = 0; otherwise the process continues.

The operations that are performed in the digital hardware as specified by a block in the ASM chart occur during the same clock period and not in a sequence of operations following each other in time, as is usually interpreted in a conventional flowchart. For example, consider state T_1. The value of Y to be considered in the decision box is taken from the value of the counter in the present state T_1. This is because the decision boxes for Flip-flop W belong to the same block as state T_1. The digital hardware generates the signals for all operations specified in the present block before arrival of the next clock pulse.

Step 3: **Derive the logic equations and draw the hardware.**

The system can be divided into two sections. These are data processor and controller. The requirements for the design of the data processor are defined inside the state and conditional boxes. The logic for the controller, on the other hand, is determined from the decision boxes and the necessary state transitions.

The design of the data processor is typically implemented by using digital components such as registers, counters, multiplexers, and adders. The system can be designed using the theory of sequential logic already discussed. Figure 5.54 shows the hardware block diagram. The Controller is shown with the required inputs and outputs. The data processor includes a 2-bit counter, one flip-flop, and one AND gate. The counter is incremented by one at the positive edge of every clock pulse when control is in stateT_1. The counter is assumed to be in count 3 initially. It is cleared to zero only when control is in stateT_0 and

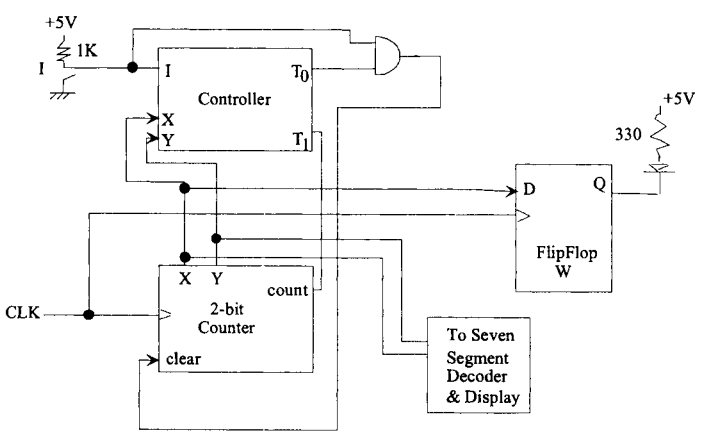


FIGURE 5.54 Hardware Schematic for the two-bit counter along with associated blocks

TABLE 5.15 State Table for the Controller

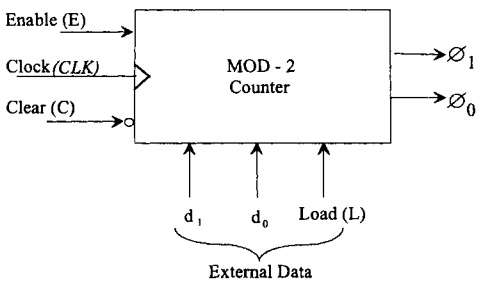
Present State (Controller)	Present States (counter)		Inputs (Controller)			Next States (counter)		Next Output States (controller)	
	X	Y	I	X	Y	X+	Y+	T_1	T_0
T_0	1	1	0	1	1	1	1	0	1
T_0	1	1	1	1	1	0	0	0	1
T_0	0	0	1	0	0	0	1	1	0
T_1	0	1	1	0	1	1	0	1	0
T_1	1	0	1	1	0	1	1	0	1

I=1. Therefore,T_0 and I are logically ANDed. The D-input of Flip-flop W is connected to output X of the counter to clear Flip-flop W during stateT_1. This is because if present count is 00 (X=0), the counter will be 01 after the next clock. On the other hand, if the present count is 01 (X=0), the count will be 10 after the next clock. Hence, X is connected to the D-input of Flip-flop W to turn the LED ON for count sequence 1, 2. A common clock is used for all flip-flops in the system including the flip-flops in the counter and Flip-flop W.

This example illustrates a technique of designing digital systems using the ASM chart. The two-bit counter can be designed using the concepts already described. In order to design the Controller, a state table for the controller must be derived. Table 5.15 shows the state table for the Controller. There is a row in the table for each possible transition between states. Initial stateT_0 stays inT_0 or goes from T_0 toT_1 depending on the status of the switch input (I). The same procedure for designing a sequential circuit described in Chapter 5 can be utilized. Since there are two controller outputs (T_1,T_0) and three inputs (I, X, Y), a three-variable K-map is required. The design of the final hardware schematic is left as an exercise to the reader.The system will contain D flip-flops with the same common clock and a combinational circuit. The design of the system using classical sequential design method may be cumbersome. Hence, other simplified methods using PLAs can be used as illustrated in the following.

A second example is provided below for designing a digital system using an ASM chart. The system has three inputs (*X, Y, Z*) and a 2-bit MOD-4 counter (W) to count from 0 to 3. The four counter states are T_0, T_1, T_2, and T_3. The operation of the system is initiated by the counter clear input, *C*. When $C = 0$, the system stays in initial state T_0. On the other hand, when $C = 1$, state transitions to be handled by the system are as follows:

INPUTS	STATE TRANSITIONS
$X = 0$	The system moves fromT_0 to T_1
$X = 1$	The system stays in T_0
$Y = 0$	The system moves back from T_1 to T_0
$Y = 1$	The system goes from T_1 to T_2
$Z = 0$	The system stays in T_2
$Z = 1$	The system moves fromT_2 toT_3 and then stays inT_3 indefinitely (for counter clear input C=1) until counter W is reset to zero (state T_0) by activating the counter clear input *C* to 0 to start a new sequence.



C	*L*	*E*	Clk	Action
0	X	X	X	clear($\varnothing 1 \varnothing 0$=00)
1	1	X	↑	load external data
1	0	1	↑	count up
1	0	0	↑	no operation

FIGURE 5.55 Block diagram and truth table of the 2-bit counter

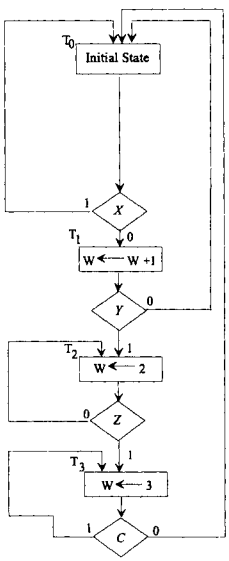


FIGURE 5.56 ASM Chart for the MOD-4 counter along with transitions

Use counter, decoder, and a PLA. Figure 5.55 shows the block diagram of the MOD-4 counter to be used in the design.

***Step 1:* Draw an ASM chart.**
The ASM chart is shown in Figure 5.56
***Step 2:* Derive the inputs, outputs, and a sequence of operations.**

The system will be designed using a PLA, a MOD-4 counter, and a 2 to 4 decoder. The MOD-4 counter is loaded or initialized with the external data if the counter control inputs *C* and *L* are both ones. The counter load control input *L* overrides the counter enable control input *E*.

The counter counts up automatically in response to the next clock pulse when the counter load control input $L = 0$ and the enable input *E* is tied to HIGH. Such normal activity is desirable for the situation (obtained from the ASM chart) when the counter goes through the sequenceT_0, T_1, T_2, T_3 for the specified inputs.

However, if the following situations occur, the counter needs to be loaded with data out of its normal sequence: If the counter is in initial state T_0 (Counter W=0 with C= 0) , it stays inT_0 for $X = 1$. This means that if the counter output is 00 and if $X = 1$, the

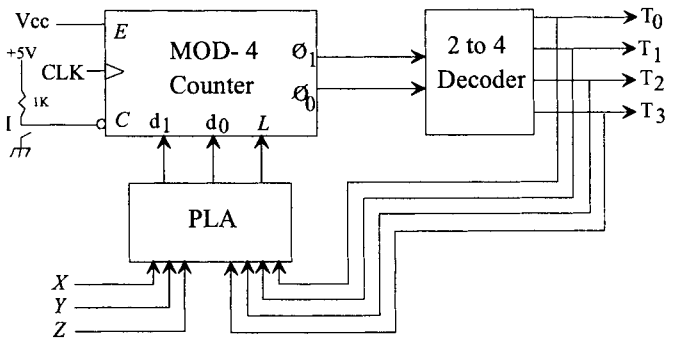
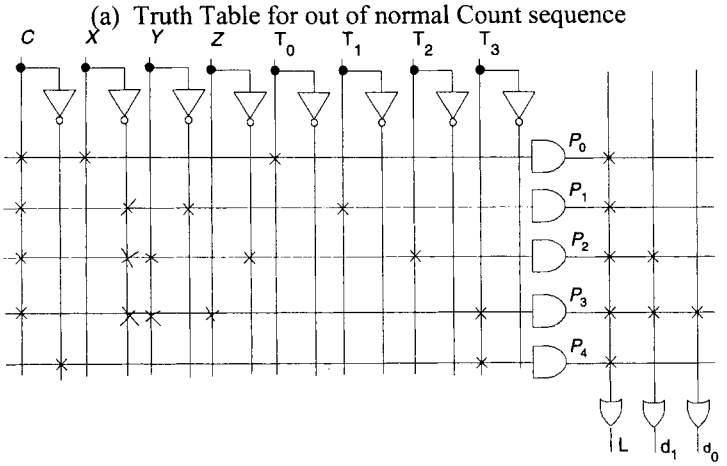


FIGURE 5.57 Hardware Schematic of the MOD-4 counter with PLA and decoder

Inputs								Outputs		
C	X	Y	Z	T_0	T_1	T_2	T_3	L	d_1	d_0
1	1	X	X	1	X	X	X	1	0	0
1	0	0	X	X	1	X	X	1	0	0
1	0	1	0	X	X	1	X	1	1	0
1	0	1	1	X	X	X	1	1	1	1
0	X	X	X	X	X	X	1	1	0	0
X = don't cares										

(a) Truth Table for out of normal Count sequence



(b) PLA implementation

FIGURE 5.58 PLA-based System

counter must be loaded with external data $d_1d_0 = 00$. Similarly, the other out of normal sequence count includes transitions ($C = 1$) fromT_1 toT_0 ($X = 0, Y = 0$), T_2 to T_2 ($X = 0$, $Y = 1$, $Z = 0$) with count 2, and T_3 toT_3 ($X = 0$, $Y = 1$, $Z = 1$); C is assumed to be HIGH during these transitions. Finally, if $C = 0$, transition from T_3 to T_0 occurs regardless of the values of X, Y, Z and the process continues. The appropriate external data must be loaded into the counter for out of normal count sequence by the PLA using the L input of the counter.

Step 3: **Derive the logic equations and draw a hardware schematic.**

Figure 5.57 depicts the logic diagram. Figure 5.58 shows the truth table and

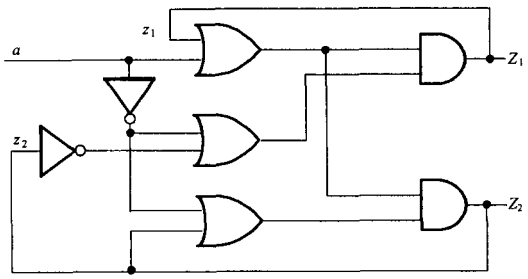


FIGURE 5.59 Asynchronous Sequential Circuit

hardware schematic for PLA-based implementation.

The equations for the product terms are: $P_0 = X\ T_0\ C$, $P_1 = \overline{X}\,\overline{Y}\,T_1C$, $P_2 = \overline{X}\,Y\,\overline{Z}T_2C$, $P_3 = \overline{X}Y\,ZT_3C$, $P_4 = T_3\,\overline{C}$, $L = P_0+P_1+P_2+P_3+P_4$, $d_1 = P_2+P_3$, $d_0 = P_3$

5.13 Asynchronous Sequential Circuits

Asynchronous sequential circuits do not require any synchronizing clocks. As mentioned before, a sequential circuit basically consists of a combinational circuit with memory. In synchronous sequential circuits, memory elements are clocked flip-flops. In contrast, memory in asynchronous sequential circuits includes either unclocked flip-flop or time-delay devices. The propagation delay time of a logic gate (finite time for a signal to propagate through a gate) provides its memory capability. Note that a sequential circuit contains inputs, outputs, and states. In synchronous sequential circuits, changes in states take place due to clock pulses. On the other hand, asynchronous sequential circuits typically contain a combinational circuit with feedback. The timing problems in the feedback may cause instability. Asynchronous sequential circuits are, therefore, more difficult to design than synchronous sequential circuits.

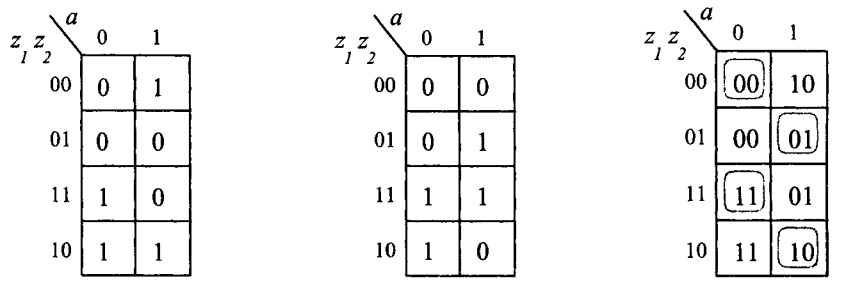
Asynchronous sequential circuits are used in applications in which the system must take appropriate actions to input changes rather than waiting for a clock to initiate actions. For proper operation of an asynchronous sequential circuit, the inputs must change one at a time when the circuit is in a stable condition (called the "fundamental mode of operation"). The inputs to the asynchronous sequential circuits are called "primary variables" whereas outputs are called "secondary variables."

Figure 5.59 shows an asynchronous sequential circuit. In the feedback loops, the uppercase letters are used to indicate next values of the secondary variables and the lowercase letters indicate present values of the secondary variables. For example, Z_1, and Z_2 are next values whereas z_1 and z_2 are present values. The output equations can be derived as follows:

$$Z_1 = (a + z_1)(\overline{a} + \overline{z_2})$$
$$Z_2 = (a + z_1)(\overline{a} + z_2)$$

The delays in the feedback loops can be obtained from the propagation delays between z_1 and Z_1 or z_2 and Z_2. Let us now plot the functions Z_1 and Z_2 in a map, and a transition table as shown in Figure 5.60.

The map for Z_1 in Figure 5.60(a) is obtained by substituting the values z_1, z_2, and a for each square into the equation for $Z1$. For example, consider $z_1z_2 = 11$ and $a = 0$.



z_1z_2 \ a	0	1
00	0	1
01	0	0
11	1	0
10	1	1

(a) Map for Z_1

z_1z_2 \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

(b) Map for Z_2

z_1z_2 \ a	0	1
00	(00)	10
01	00	(01)
11	(11)	01
10	11	(10)

(c) Transition Table

FIGURE 5.60 Map and Transition Table

$$\begin{aligned} Z_1 &= (a + z_1)(\bar{a} + \bar{z_2}) \\ &= (0 + 1)(\bar{0} + \bar{1}) \\ &= 1 \end{aligned}$$

$$\begin{aligned} Z_2 &= (a + z_1)(\bar{a} + z_2) \\ &= (0 + 1)(\bar{0} + 1) \\ &= 1 \end{aligned}$$

Similarly, values for all other sequences can be obtained similarly. The transition table of Figure 5.60(c) can be obtained by combining the binary values of two squares in the same position and placing them in the corresponding square in the transition table. Thus, the variable $Z = Z_1Z_2$ is placed in each square of the transition table. For example, from the first square of Figure 5.60(a) and (b), $Z = 00$. This is shown in the first square of Figure 5.60(c). The squares in the transition table in which $z_1z_2 = Z_1Z_2$ are circled to show that they are stable. The uncircled squares are unstable states.

Let us now analyze the behavior of the circuit due to change in the input variable. Suppose $a = 0$, $z_1z_2 = 00$, then the output is 00. Thus, 00 is circled and shown in the first square of Figure 5.60(c). Z is the next value of z_1z_2 and is a stable state. Next suppose that a goes from 0 to 1 and the value of Z changes from 00 to 01. Note that this causes an interim unstable situation because Z_1Z_2 is initially equal to z_1z_2. This is because as soon as the input changes from 0 to 1, this change in input travels through the circuit to change Z_1Z_2 from 00 to 01. The feedback loop in the circuit eventually makes z_1z_2 equal to Z_1Z_2; that is, $z_1z_2 = Z_1Z_2 = 01$. Because $z_1z_2 = Z_1Z_2$, the circuit attains a stable state. The state 01 is circled in the figure to indicate this. Similarly, it can be shown that as the input to an asynchronous sequential circuit changes, the circuit goes to a temporary unstable condition until it reaches a stable state when Z_1Z_2 = present state, z_1z_2. Therefore, as the input moves between 0 and 1, the circuit goes through the states 00, 01, 11, 10, and repeats the sequence depending on the input changes. A state table can be derived from the transition table. This is shown in Table 5.16, which is the state table for Figure 5.60(c).

TABLE 5.16 Transition Table

Present State		Next State			
		a=0		*a=1*	
0	0	0	0	1	0
0	1	0	0	0	1
1	0	1	1	1	0
1	1	1	1	0	1

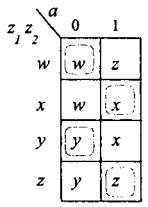


FIGURE 5.61 Flow Table

A flow table obtained from the transition table is normally used in designing an asynchronous sequential circuit. A flow table resembles a transition table except that the states are represented by letters instead of binary numbers. The transition table of Figure 5.60(c) can be translated into a flow table as shown in Figure 5.61. Note that the states are represented by binary numbers as follows: $w = 00$, $x = 01$, $y = 11$, $z = 10$. The flow table in Figure 5.61 is called a "primitive flow table" because it has only one stable state in each row.

An asynchronous sequential circuit can be designed using the primitive flow table from the problem definition. The flow table is then simplified by combining squares to a minimum number of states. The transition table is then obtained by assigning binary numbers to the states. Finally, a logic diagram is obtained from the transition table. The logic diagram includes a combinational circuit with feedback.

The design of an asynchronous sequential circuit is more difficult than the synchronous sequential circuit because of the timing problems associated with the feedback loop. This topic is beyond the scope of this book.

QUESTIONS AND PROBLEMS

5.1 What is the basic difference between a combinational circuit and a sequential circuit?

5.2 Identify the main characteristics of a synchronous sequential circuit and an asynchronous sequential circuit.

5.3 What is the basic difference between a latch and a flip-flop?

5.4 Draw the logic diagram of a D flip-flop using OR gates and inverters.

5.5 Assume that initially $x = 1$, $A = 0$, and $B = 1$ in figure P5.5. Determine the values of A and B after the positive edge of *Clk*.

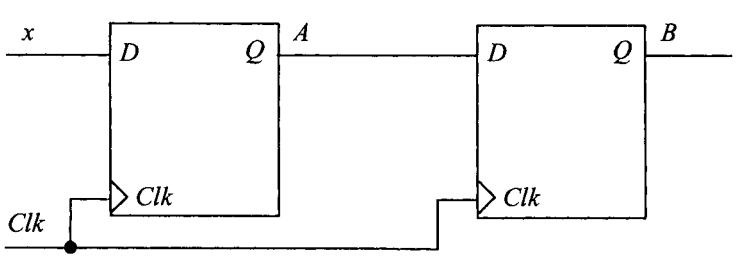


FIGURE P5.5

5.6 Draw the logic diagram of a JK flip-flop using AND gates and inverters.

5.7 Assume that initially $X = 1$, $A = 0$, and $B = 1$ in figure P5.7. Determine the values of A and B after one *Clk* pulse. Note that the flip-flops are triggered at the clock level.

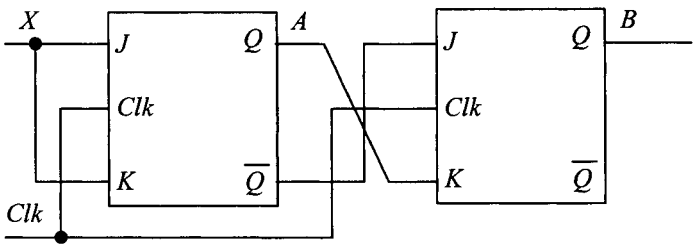


FIGURE P5.7

5.8 Given Figure P5.8, draw the timing diagram for Q and $\overline{Q}$ assuming a negative-edge triggered JK flip- flop. Assume Q is preset to 1 initially.

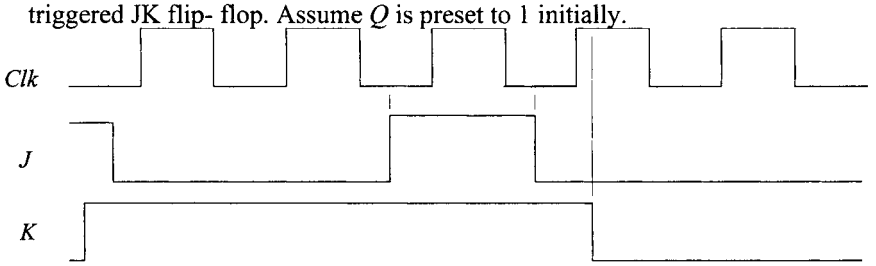


FIGURE P5.8

5.9 Given the timing diagram for a positive-edge triggered D flip-flop in Figure P5.9, draw the timing diagrams for Q and $\overline{Q}$. Assume Q is cleared to zero initially.

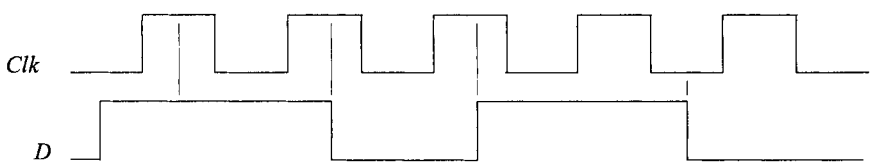


FIGURE P5.9

5.10 Given the timing diagram for a negative-edge triggered T flip-flop in Figure P5.10, draw the timing diagram for Q. Assume Q is preset to 1 initially.

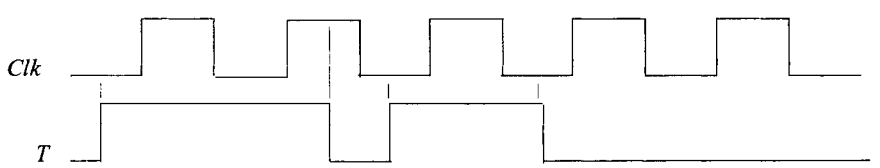


FIGURE P5.10

5.11 Why would you use an edge-triggered flip-flop rather than a level-triggered flip-flop?

5.12 What are the advantages of a master–slave flip-flop?

5.13 Draw the block diagram of a T flip-flop using (a) JK ff (b) D ff.

5.14 Draw a logic circuit of the switch debouncer circuit using NAND gates.

5.15 Analyze the clocked synchronous circuit shown in Figure P5.15. Express the next state in terms of the present state and inputs, derive the state table, and draw the state diagram.

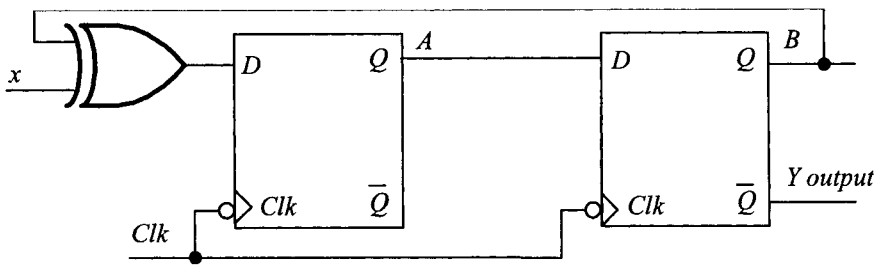


FIGURE P5.15

5.16 A synchronous sequential circuit with two D flip-flops *(a,b as outputs)*, one input *(x)*, and an output *(y)* is expressed by the following equations:

$$D_a = a\,\bar{b}\,x + \bar{a}\,b, \quad D_b = \bar{x}\,b + \bar{b}\,x$$
$$y = \bar{b}\,\bar{x} + a$$

(a) Derive the state table and state diagram for the circuit.
(b) Draw a logic diagram.

5.17 A synchronous sequential circuit is represented by the state diagram shown in Figure P5.17. Using JK flip-flops and undefined states as don't-cares:
(a) Derive the state table.
(b) Minimize the equation for flip-flop inputs using K-maps.
(c) Draw a logic diagram.

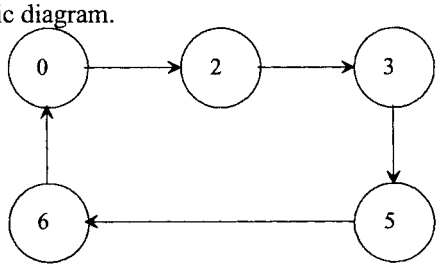


FIGURE P5.17

5.18 A sequential circuit contains two D flip-flops (A, B), one input (x), and one output (y), as shown in Figure P5.18.
Derive the state table and the state diagram of the sequential circuit.

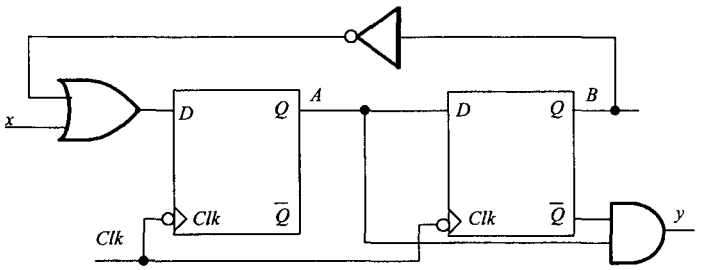


FIGURE P5.18

5.19 Design a synchronous sequential circuit using D flip-flops for the state diagram shown in Figure P5.19.

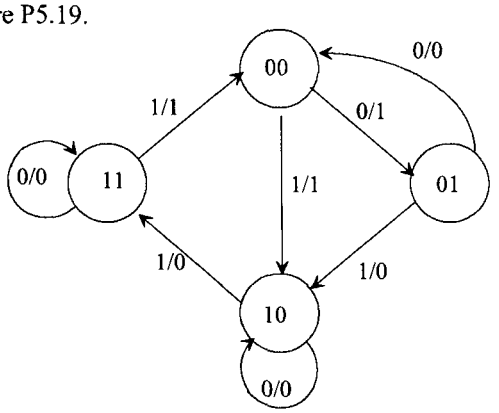


FIGURE P5.19

5.20 Design a 2-bit counter that will count in the following sequence: 00, 11, 10, 01, and repeat. Using T flip-flops:

(a) Draw a state diagram.
(b) Derive a state table.
(c) Implement the circuit.

5.21 Design a synchronous sequential circuit with one input x and one output y. The input x is a serial message, and the system reads x one bit at a time. The output y is 1 whenever the binary pattern 000 is encountered in the serial message. For example: If the input is 01000000, then the output will be 00001010. Use T flip-flops.

5.22 Analyze the circuit shown in Figure P5.22 and show that it is equivalent to a T flip-flop.

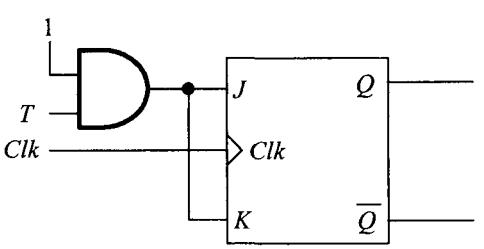


FIGURE P5.22

5.23 Design a BCD counter to count in the sequence 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, and repeat. Use T flip-flops.

5.24 Design the following nonbinary sequence counters using the type of flip-flop specified. Assume the unused states as don't cares. Is the counter self-correcting? Justify your answer.

(a) Counting sequence 0, 1, 3, 4, 5, 6, 7, and repeat. Use JK flip-flops.

(b) Counting sequence 0, 2, 3, 4, 6, 7, and repeat. Use D flip-flops.

(c) Counting sequence 0, 1, 2, 4, 5, 6, 7, and repeat. Use T flip-flops.

5.25 Design a 4-bit general-purpose register as follows:

S_1	S_0	Function
0	0	Load external data
0	1	Rotate left; ($A_0 \leftarrow A_3$, $A_i \leftarrow A_{i-1}$ for $i = 1,2,3$)
1	0	Rotate right; ($A_3 \leftarrow A_0$, $A_i \leftarrow A_{i+1}$ for $i = 0,1,2$)
1	1	Increment

Use Figure P5.25 as the building block:

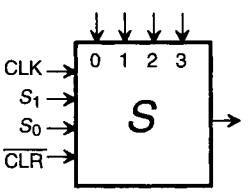


FIGURE P5.25

5.26 Design a logic diagram that will generate 19 timing signals. Use a ring counter with JK flip-flops.

5.27 Consider the 2-bit Johnson counter shown in Figure P5.27. Derive the state diagram. Assume the D flip-flops are initialized to $A = 0$ and $B = 0$.

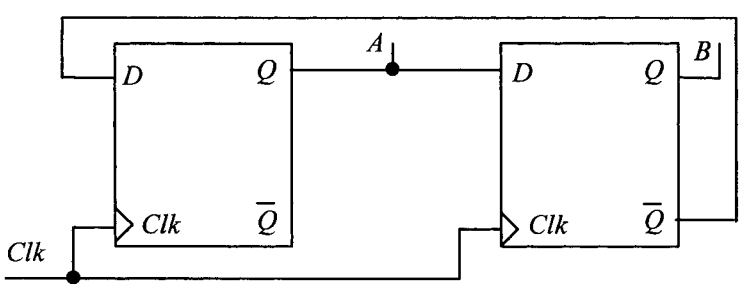


FIGURE P5.27

5.28 Assuming $AB = 10$, verify that the 2-bit counter shown in Figure P5.28 is a ring counter. Derive the state diagram.

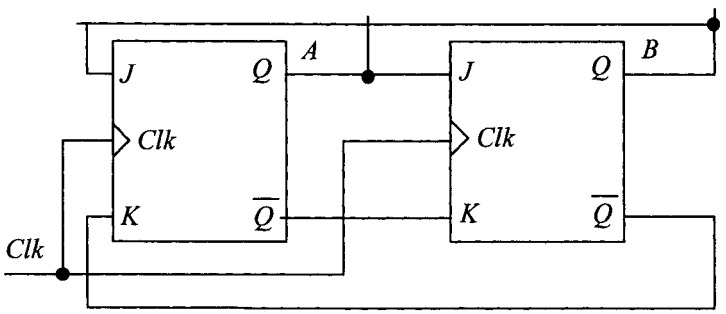


FIGURE P5.28

5.29 What is the basic difference between SRAM and DRAM?

5.30 Given a memory with a 24-bit address and 8-bit word size,
(a) How many bytes can be stored in this memory?
(b) If this memory were constructed from 1K × 1-bit RAM chips, how many memory chips would be required?

5.31 Draw an ASM chart for the following: Assume three states (*a*, *b*, *c*) in the system with one input *x* and two registers R_1 and R_2. The circuit is initially in state *a*. If $x = 0$, the control goes from state *a* to state *b* and, clears registers R_1 to 0 and sets R_2 to 1, and then moves to state *c*. On the other hand if $x = 1$, the control goes to state *c*. In state *c*, R_1 is subtracted from R_2 and the result is stored in R_1. The control then moves back to state *a* and the process continues.

5.32 Draw an ASM chart for each of the following sequence of operations:
(a) The ASM chart will define a conditional operation to perform the operation $R_2 \leftarrow R_2 - R_1$ during State T_0 and will transfer control to State T_1 if the control input c is 1; if c=0, the system will stay in T_0. Assume that R_1 and R_2 are 8-bit registers.
(b) The ASM chart in which the system is initially in State T_0 and then checks a control input c. If c=1, control will move from State T_0 to State T_1; if c=0, the system will increment an 8-bit register R by 1 and control will return to the initial state.

5.33 Draw an ASM chart for the following state diagram of Figure P5.33:

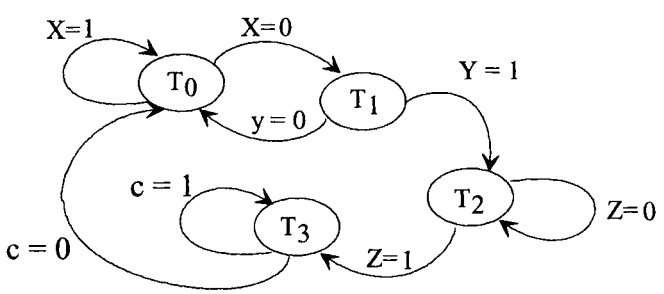


FIGURE P5.33

Assume that the system stays in initial state T_0 when control input c = 0 and input X = 1. The sequence of operations is started from T_0 when X = 0. When the system reaches state T_3, it stays in T_3 indefinitely as long as c = 1; the system returns to state T_0 when c = 0.

5.34 Derive the output equations for the asynchronous sequential circuit shown in Figure P5.34. Also, determine the state table and flow table.

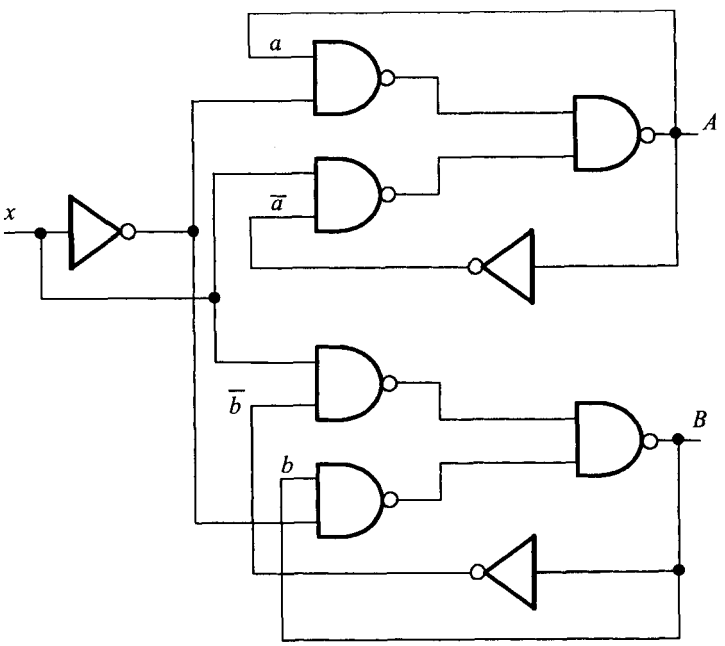


FIGURE P5.34