1.7 Mealy Sequential Circuit Design

There are two basic types of sequential circuits: Mealy and Moore. In a Mealy circuit, the outputs depend on both the present state and the present inputs. In a Moore circuit, the outputs depend only on the present state. A general model of a Mealy sequential circuit consists of a combinational circuit, which generates the outputs and the next state, and a state register, which holds the present state (see Figure 1-17). The state register normally consists of D flip-flops. The normal sequence of events is (1) the X inputs change to a new value; (2) after a delay, the corresponding Z outputs and next state appear at the output of the combinational circuit; and (3) the next state is clocked into the state register and the state changes. The new state feeds back into the combinational circuit, and the process is repeated.

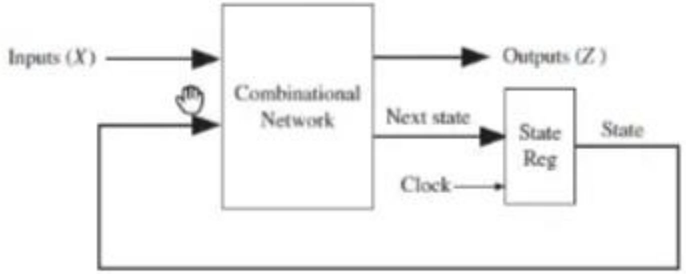


FIGURE 1-17: General Model of Mealy Sequential Machine

Mealy Machine Design Example 1: Sequence Detector

To illustrate the design of a clocked Mealy sequential circuit, let us design a sequence detector. The circuit has the form indicated in the block diagram in Figure 1-18.

tional circuit; and (3) the next state is clocked into the state register and the state changes. The new state feeds back into the combinational circuit, and the process is repeated.

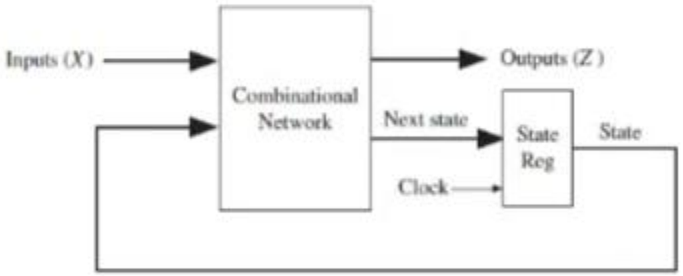


FIGURE 1-17: General Model of Mealy Sequential Machine

Mealy Machine Design Example 1: Sequence Detector

To illustrate the design of a clocked Mealy sequential circuit, let us design a sequence detector. The circuit has the form indicated in the block diagram in Figure 1-18.

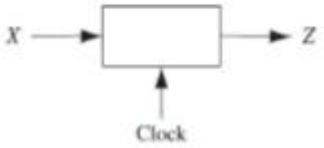


FIGURE 1-18: Block Diagram of a Sequence Detector

The circuit will examine a string of 0s and 1s applied to the X input and generate an output Z=1 only when the input sequence is 101. The input X can change only

$Z = 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0$

Let us construct a *state graph* for this sequence detector. We will start in a reset state designated S_0. If a 0 input is received, we can stay in state S_0 as the input sequence we are looking for does not start with 0. However, if a 1 is received, the circuit should go to a new state. Let us denote that state as S_1. When in S_1, if we receive a 0, the circuit must change to a new state (S_2) to indicate that the first two inputs of the desired sequence (10) have been received. If a 1 is received in state S_2, the desired input sequence is complete and the output should be a 1. The output will be produced as a Mealy output and will coincide with the last 1 in the detected sequence. Since we are designing a Mealy circuit, we are not going to go to a new state that indicates the sequence 101 has been received. When we receive a 1 in S_2, we cannot go to the start state since the circuit is not supposed to reset with every detected sequence. But the last 1 in a sequence can be the first 1 in another sequence; hence, we can go to state S_1. The partial state graph at this point is indicated in Figure 1-19.

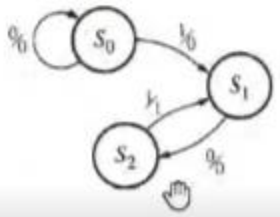


FIGURE 1-19: **Partial State Graph of the Sequence Detector**

When a 0 is received in state S_2, we have received two 0s in a row and must reset the circuit to state S_0. If a 1 is received when we are in S_1, we can stay in S_1 because the most recent 1 can be the first 1 of a new sequence to be detected. The final state graph is shown in Figure 1-20. State S_0 is the starting state, state S_1 indicates that a

S_2, we cannot go to the start state since the circuit is not supposed to reset with every detected sequence. But the last 1 in a sequence can be the first 1 in another sequence; hence, we can go to state S_1. The partial state graph at this point is indicated in Figure 1-19.

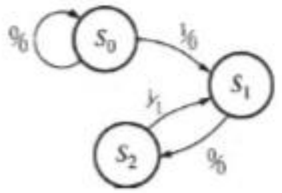


FIGURE 1-19: **Partial State Graph of the Sequence Detector**

When a 0 is received in state S_2, we have received two 0s in a row and must reset the circuit to state S_0. If a 1 is received when we are in S_1, we can stay in S_1 because the most recent 1 can be the first 1 of a new sequence to be detected. The final state graph is shown in Figure 1-20. State S_0 is the starting state, state S_1 indicates that a sequence ending in 1 has been received, and state S_2 indicates that a sequence ending in 10 has been received. Converting the state graph to a state table yields Table 1-3. In row S_2 of the Table, an output of 1 is indicated for input 1.

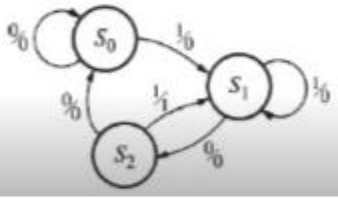


FIGURE 1-20: **Mealy State Graph for Sequence Detector**

TABLE 1-3: State Table for Sequence Detector

Present State	Next State		Present Output	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

Next, *state assignment* is performed, whereby specific flip-flop values are associated with specific states. There are two techniques to perform state assignment: (i) one-hot state assignment and (ii) encoded state assignment. In one-hot state assignment, one flip-flop is used for each state. Hence three flip-flops will be required if this circuit is to be implemented using the one-hot approach. In encoded state assignment, just enough flip-flops to have a unique combination for each state are sufficient. Since we have three states, we need at least two flip-flops to represent all states. We will use encoded state assignment in this design. Let us designate the two flip-flops as A and B. Let the flip-flop states $A = 0$ and $B = 0$ correspond to state S_0; $A = 0$ and $B = 1$ correspond to state S_1, and $A = 1$ and $B = 0$ correspond to state S_2. Now, the transition table of the circuit can be written as in Table 1-4.

TABLE 1-4: Transition Table for Sequence Detector

	A^+B^+		Z	
AB	$X = 0$	$X = 1$	$X = 0$	$X = 1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

as in Table 1-4.

TABLE 1-4: Transition Table for Sequence Detector

	A^+B^+		Z	
AB	$X = 0$	$X = 1$	$X = 0$	$X = 1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

From this table, we can plot the K-maps for the next states and the output Z. The next states are typically represented by A^+ and B^+. The 3 K-maps are shown in Figure 1-21.

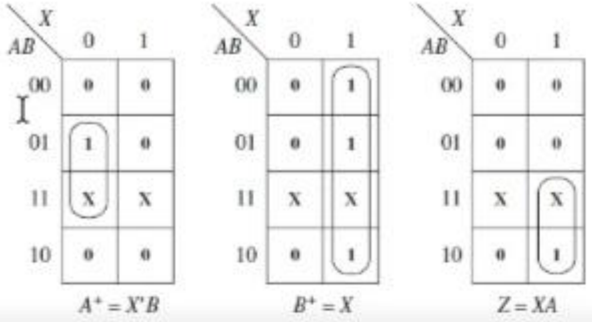
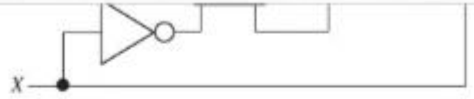


FIGURE 1-21: K-Maps for Next States and Output of Sequence Detector

DHURA R

The next step is deriving the flip-flop inputs to obtain the next states shown in Figure 1-21. If D flip-flops are used, one simply needs to give the expected next state of the flip-flop to the flip-flop input. So, for flip-flop A, $D_A = A^+$ and $D_B = B^+$. The resulting circuit is shown in Figure 1-22.



Mealy Machine Design Example 2: BCD to Excess-3 Code Converter

As an example of a more complex Mealy sequential circuit, we will design a serial code converter that converts an 8-4-2-1 binary-coded-decimal (BCD) digit to an excess-3-coded decimal digit. The input (X) will arrive serially with the least significant bit first. The outputs will be generated serially as well. Table 1-5 lists the desired inputs and outputs at times t_0, t_1, t_2, and t_3. After receiving four inputs, the circuit should reset to its initial state, ready to receive another BCD digit.

TABLE 1.5: Code Converter

X Input (BCD)				*Z* Output (excess -3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

The excess-3 code is formed by adding 0011 to the BCD digit. For example, however, the bits arrive sequentially, one bit at a time. Hence this code converter must be implemented sequentially.

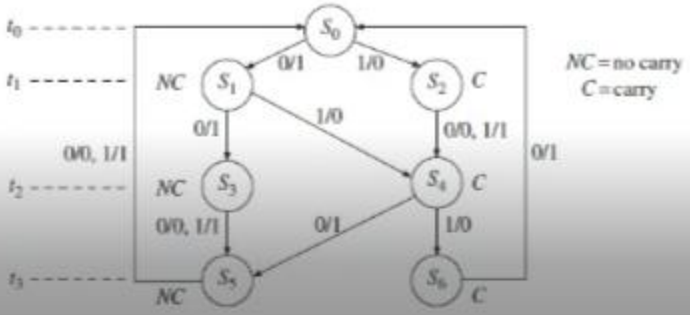
Let us now construct a state graph for the code converter (Figure 1-23(a)). Let us designate the start state as S_0. The first bit arrives, and one needs to add 1 to this bit, as it is the least significant bit (LSB) of 0011, the number to be added to the BCD digit to obtain the excess-3 code.

At t_0, we add 1 to the LSB, so if $X = 0$, $Z = 1$ (no carry), and if $X = 1$, $Z = 0$ (carry = 1). Let us use S_1 to indicate no carry after the first addition, and S_2 to indicate a carry of 1 after the addition to the LSB.

At t_1, we add 1 to the next bit, so if there is no carry from the first addition (state S_1), $X = 0$ gives $Z = 0 + 1 + 0 = 1$ and no carry (state S_3), and $X = 1$ gives $Z = 1 + 1 + 0 = 0$ and a carry (state S_4). If there is a carry from the first addition (state S_2), then $X = 0$ gives $Z = 0 + 1 + 1 = 0$ and a carry (S_4), and $X = 1$ gives $Z = 1 + 1 + 1 = 1$ and a carry (S_4).

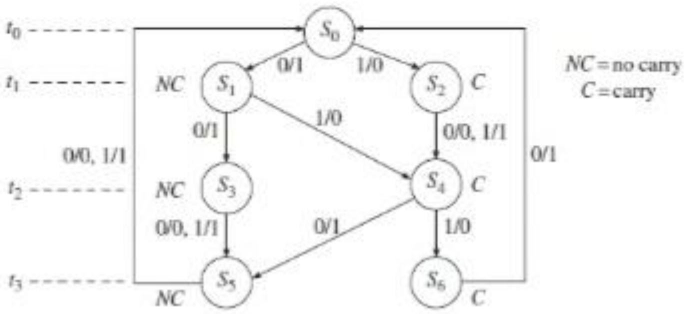
At t_2, 0 is added to X, and transitions to S_5 (no carry) and S_6 are determined in a similar manner. At t_3, 0 is again added to X, and the circuit resets to S_0.

FIGURE 1-23: State Graph and Table for Code Converter



similar manner. At t_3, 0 is again added to X, and the circuit resets to S_0.

FIGURE 1-23: State Graph and Table for Code Converter



(a) Mealy state graph

	NS		Z	
PS	X=0	X=1	X=0	X=1
S_0	S_1	S_2	1	0
S_1	S_3	S_4	1	0
S_2	S_4	S_4	0	1
S_3	S_5	S_5	0	1
S_4	S_5	S_6	1	0
S_5	S_0	S_0	0	1
S_6	S_0	–	1	–

(b) State table

Figure 1-23(b) gives the corresponding state table. At this point, we should verify that the table has a minimum number of states before proceeding (see Section 1-9). Then state assignment must be performed. Since this state table has seven states, three flip-flops will be required to realize the table in encoded state assignment. In the one-hot approach, one flip-flop is used for each state. Hence seven flip-flops will be required if this circuit is to be implemented using the one-a one-hot assignment may be preferred. In recent years, with the abundance of transistors on silicon chips, the emphasis on optimal state assignment has been reduced.

In order to reduce the amount of logic required, we will make a state assignment using the following guidelines (see *Fundamentals of Logic Design*, 7th ed., Cengage Learning, 2014 for details):

I. States that have the same next state (NS) for a given input should be given adjacent assignments (look at the columns of the state table).
II. States that are the next states of the same state should be given adjacent assignments (look at the rows).
III. States that have the same output for a given input should be given adjacent assignments.

Using these guidelines tends to clump 1s together on the Karnaugh maps for the next state and output functions. The guidelines indicate that the following states should be given adjacent assignments:

I.	(1, 2), (3, 4), (5, 6)	(in the $X = 1$ column, S_1 and S_2 both have NS S_4; in the $X = 0$ column, S_3 and S_4 have NS S_5, and S_5 and S_6 have NS S_0)
II.	(1, 2), (3, 4), (5, 6)	(S_1 and S_2 are NS of S_0; S_3 and S_4 are NS of S_1; and S_5 and S_6 are NS of S_4)
III.	(0, 1, 4, 6), (2, 3, 5)	

Figure 1-24(a) gives an assignment map, which satisfies the guidelines, and the corresponding transition table (Figure 1-24(b)). Since state 001 is not used, the next state and outputs for this state are don't cares. The next state and output equations are derived from this table in Figure 1-25. Figure 1-26 shows the realization of the code converter using NAND gates and D flip-flops.

Using these guidelines tends to clump 1s together on the Karnaugh maps for the next state and output functions. The guidelines indicate that the following states should be given adjacent assignments:

I.	(1, 2), (3, 4), (5, 6)	(in the $X = 1$ column, S_1 and S_2 both have NS S_4; in the $X = 0$ column, S_3 and S_4 have NS S_5, and S_5 and S_6 have NS S_0)
II.	(1, 2), (3, 4), (5, 6)	(S_1 and S_2 are NS of S_0; S_3 and S_4 are NS of S_1; and S_5 and S_6 are NS of S_4)
III.	(0, 1, 4, 6), (2, 3, 5)	

Figure 1-24(a) gives an assignment map, which satisfies the guidelines, and the corresponding transition table (Figure 1-24(b)). Since state 001 is not used, the next state and outputs for this state are don't cares. The next state and output equations are derived from this table in Figure 1-25. Figure 1-26 shows the realization of the code converter using NAND gates and D flip-flops.

FIGURE 1-24: State Assignment for BCD to Excess-3 Code Converter

$Q_2Q_3 \backslash Q_1$	0	1
00	S_0	S_1
01		S_2
11	S_5	S_3
10	S_6	S_4

(a) Assignment map

	$Q_1^+Q_2^+Q_3^+$		Z	
$Q_1Q_2Q_3$	$X=0$	$X=1$	$X=0$	$X=1$
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

(b) Transition table

assignments.

Using these guidelines tends to clump 1s together on the Karnaugh maps for the next state and output functions. The guidelines indicate that the following states should be given adjacent assignments:

I. (1, 2), (3, 4), (5, 6) (in the $X = 1$ column, S_1 and S_2 both have NS S_4; in the $X = 0$ column, S_3 and S_4 have NS S_5, and S_5 and S_6 have NS S_0)

II. (1, 2), (3, 4), (5, 6) (S_1 and S_2 are NS of S_0; S_3 and S_4 are NS of S_1; and S_5 and S_6 are NS of S_4)

III. (0, 1, 4, 6), (2, 3, 5)

Figure 1-24(a) gives an assignment map, which satisfies the guidelines, and the corresponding transition table (Figure 1-24(b)). Since state 001 is not used, the next state and outputs for this state are don't cares. The next state and output equations are derived from this table in Figure 1-25. Figure 1-26 shows the realization of the code converter using NAND gates and D flip-flops.

FIGURE 1-24: State Assignment for BCD to Excess-3 Code Converter

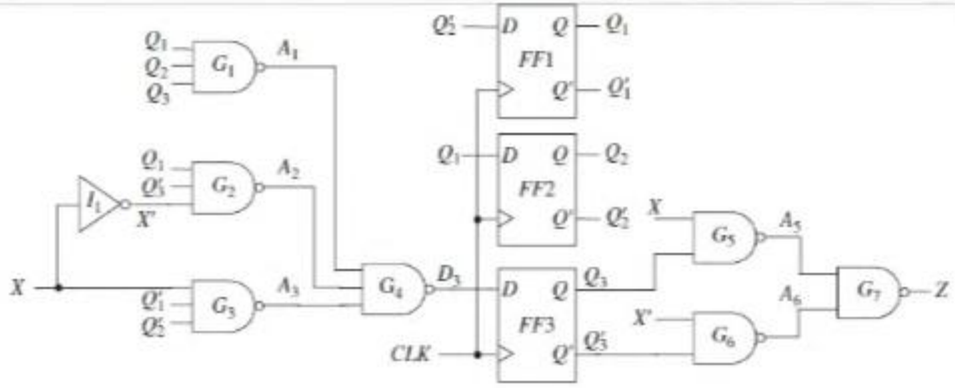
$Q_2Q_3 \backslash Q_1$	0	1
00	S_0	S_1
01		S_2
11	S_5	S_3
10	S_6	S_4

(a) Assignment map

	$Q_1^+Q_2^+Q_3^+$		Z	
$Q_1Q_2Q_3$	$X=0$	$X=1$	$X=0$	$X=1$
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

(b) Transition table

FIGURE 1-26: Realization of Code Converter



If J-K flip-flops are used instead of D flip-flops, the input equations for the J-K flip-flops can be derived from the next state maps. Given the present state flip-flop (Q) and the desired next state (Q^+), the J and K inputs can be determined from Table 1-6, also known as the excitation table. This table is derived from the truth table in Figure 1-12.

TABLE 1-6 Excitation Table for a J-K Flip-Flop

Q	Q^+	J	K	
0	0	0	X	(No change in Q; J must be 0, K may be 1 to reset Q to 0.)
0	1	1	X	(Change to $Q = 1$; J must be 1 to set or toggle.)
1	0	X	1	(Change to $Q = 0$; K must be 1 to reset or toggle.)
1	1	X	0	(No change in Q; K must be 0, J may be 1 to set Q to 1.)

Figure 1-27 shows the derivation of J-K flip-flops for the state table of