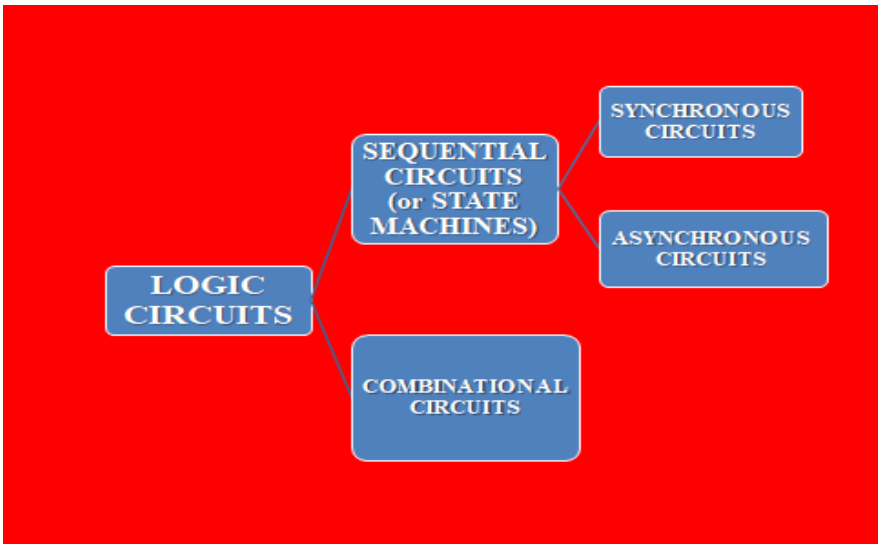
محاكيات الانظمة الالكترونية

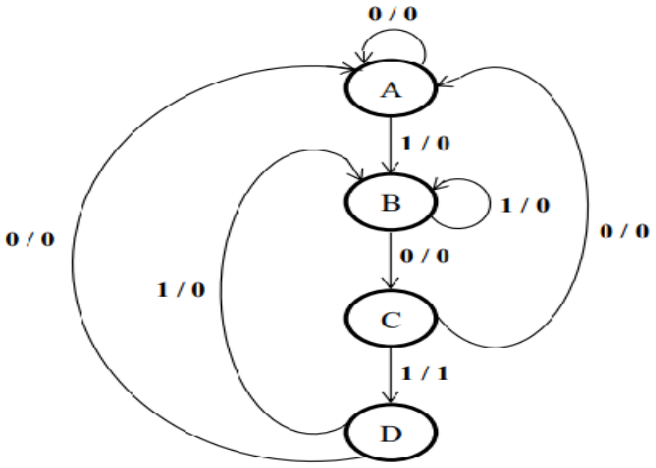
Logic circuits are of two types:

1. **Combinational circuits**
2. **Sequential circuits**

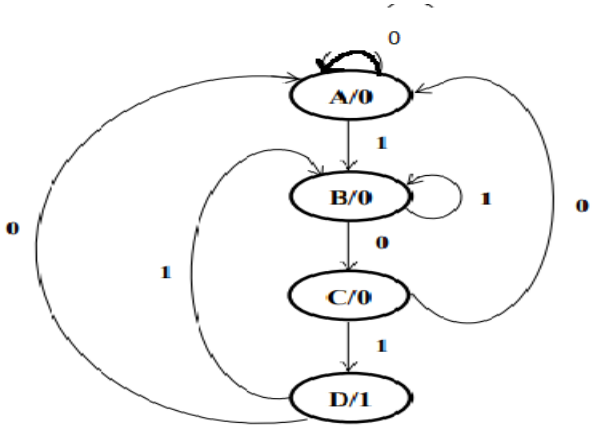
The combinational logic circuit is built from logical gates only. While the sequential logic circuit is mainly built from flip flops in addition to logical gates



The sequential logic circuit (or state machine) differs from the combinational circuit in that it has a memory. Its output(s) depend(s) on its present input(s) and the previous states of the circuit.
The state machines are either synchronous or asynchronous circuits. In the synchronous state machine, all the units (flip flops) of the circuit change their present states to the next states simultaneously (controlled by a master clock).
In the asynchronous state machine, there is no master clock controlling the transition of the circuit's units from the present states to the next states.
There are two models for designing state machines (the Mealy model The MOORE mode



State diagram
(Mealy model)



State diagram
(Moore model)
Directed lines: X
Circles: state-code/Z

Starting from an initial waiting state A, if a 0 is received the circuit stays in the same state. For a 1, however, there is a transition to state B, indicating the start of the required sequence. If, while in this state, a 0 is received, i.e. sequence (10), the circuit changes to state C. When in state C, if a 1 is received, completing the sequence (101), the circuit changes to state D, giving the required output. An input of 0 returns the circuit to state A to await the start of another sequence. When the circuit is in state D, a 1 returns it to state B, and a(0) returns it to state A.

Mealy State - Table

Present State	Next State		Output (Z)	
	Input (X)		Input (X)	
	0	1	0	1
A	A	B	0	0
B	C	B	0	0
C	A	D	0	1
D	A	B	0	0

Moore State - Table

Present State	Next State		Output (Z)
	Input (X)		
	0	1	
A	A	B	0
B	C	B	0
C	A	D	0
D	A	B	1

It is clear that the output column in the Moore state – table has a unique value for each state.

Analysis of Synchronous State Machines

The analysis process is the reverse of the design process. Here, we have a designed state machine (or sequential logic circuit) and it is required to obtain its state-diagram.

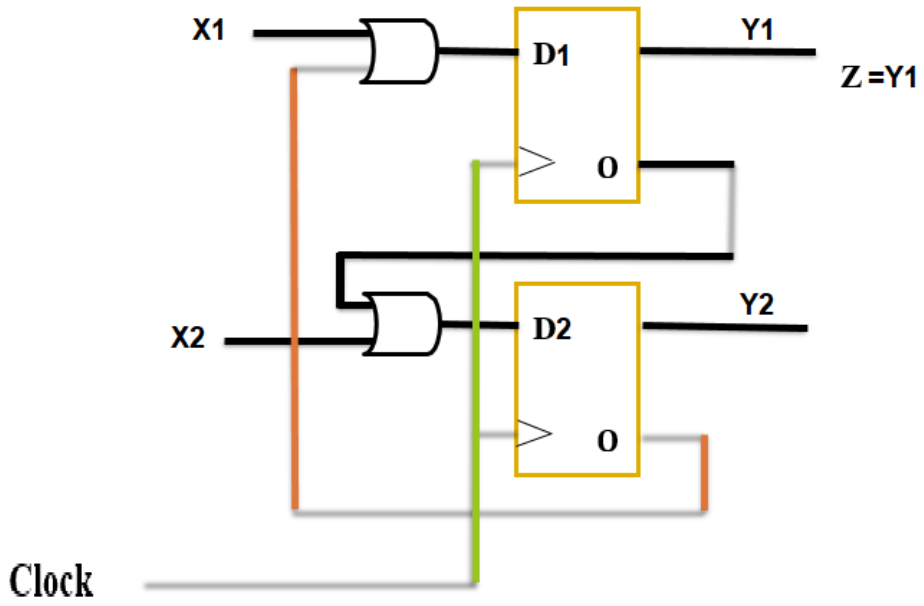
In the design, we start with a state-diagram and end with the circuit diagram. But in the analysis, we start with the circuit diagram and end with the state-diagram.

Therefore in the analysis, the steps of design are applied reversely:
From the circuit diagram,

1-the equations of the flip flops inputs as well as the output(s) equation(s) are derived
2-Drawing the K-map of each equation.
3-Construction of the binary-assignment table.
4-Obtaining the state-table.
5-Drawing the state-diagram.

Example 1 :

Let us analyze the following synchronous sequential circuit.



The following equations are derived from the circuit diagram:

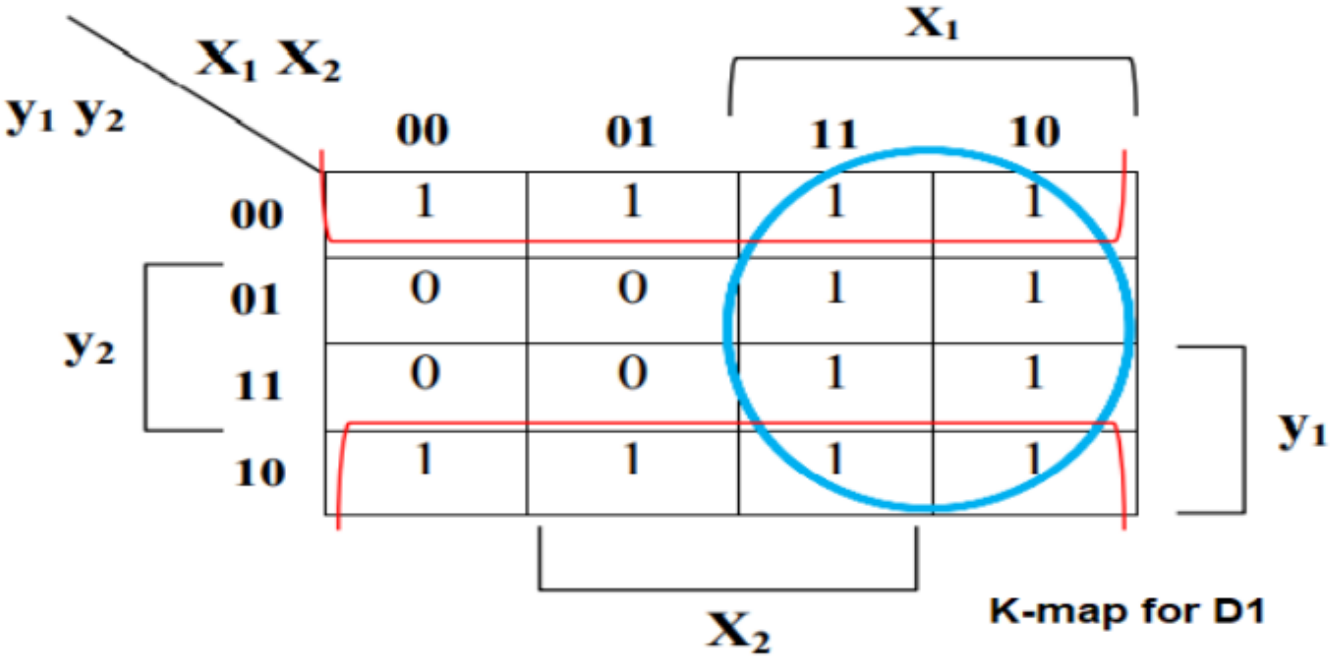
$$D_1 = X_1 + \overline{y_2}$$

$$D_2 = X_2 + \overline{y_1}$$

$$Z = y_1$$

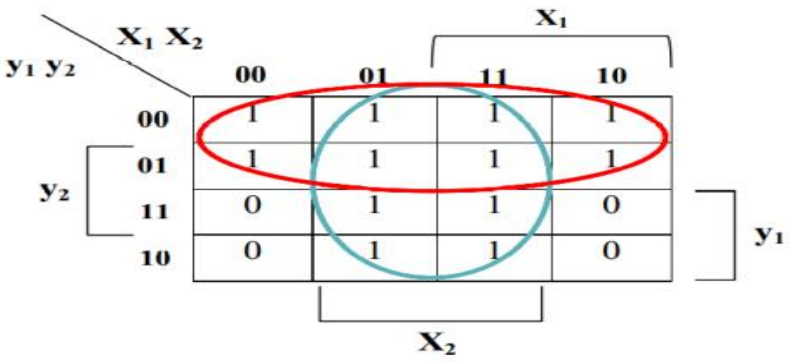
Since Z is independent of the input X, so the circuit was designed by the Mooremodel.

The K-map of the equation (D1 = X1 + y2 ')is obtained as follows:
The first term in the equation is X1, the cells in the K-map that gives the value ofthe group shown in blue color are filled by 1.
The Boolean expression of this group is X1.



$$D_1 = y_1^+ = X_1 + \overline{y_2}$$

Similarly, the K-map of the equation $D_2 = X_2 + \overline{y_1}$ is derived as below.



$$D_2 = y_2^+ = X_2 + \overline{y_1}$$

The binary-assignment table is obtained by collecting together the values of y_1^+ and y_2^+ shown in the above two K-maps.

Binary-assignment table

Present State $y_1\ y_2$	Next State Inputs ($X_1\ X_2$) 00 $y_1^+y_2^+$	01 $y_1^+y_2^+$	11 $y_1^+y_2^+$	10 $y_1^+y_2^+$	Output (Z)
0 0	1 1	1 1	1 1	1 1	
0 1	0 1	0 1	1 1	1 1	
1 1	0 0	0 1	1 1	1 0	
1 0	1 0	1 1	1 1	1 0	

Since Z = y1 then the last column of Z values is filled with the same values of y1 in the first column. The final binary assignment table will be:

Binary-assignment table

Present State $y_1\ y_2$	Next State Inputs ($X_1\ X_2$) 00 $y_1^+y_2^+$	01 $y_1^+y_2^+$	11 $y_1^+y_2^+$	10 $y_1^+y_2^+$	Output (Z)
0 0	1 1	1 1	1 1	1 1	0
0 1	0 1	0 1	1 1	1 1	0
1 1	0 0	0 1	1 1	1 0	1
1 0	1 0	1 1	1 1	1 0	1

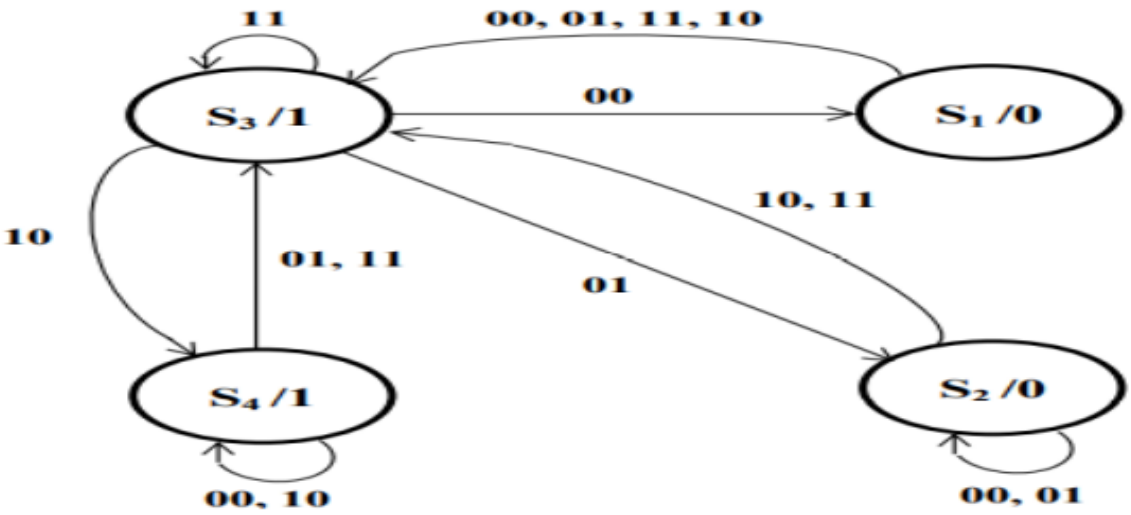
in the state-assignment table, each state is denoted by a binary code. Now, we have to represent the states by symbols instead of codes.

State code	State symbol
0 0	S_1
0 1	S_2
1 1	S_3
1 0	S_4

State-Table

Present State	Next State				Output (Z)
	Inputs (X_1 X_2)				
	00	01	11	10	
S_1	S_3	S_3	S_3	S_3	0
S_2	S_2	S_2	S_3	S_3	0
S_3	S_1	S_2	S_3	S_4	1
S_4	S_4	S_3	S_3	S_4	1

Finally, the state-diagram is constructed from the above state table.



State Diagram

Example 2 :

Give the state-diagram of a synchronous sequential logic circuit that has one input X and three outputs Z_1, Z_2, and Z_3. It consists of two D flip flops y_1 and y_2. The flip-flop input equations and the equations of the outputs are:

$$D_1 = \overline{y_1}\ y_2 X + y_1 \overline{X}$$

$$D_2 = \overline{y_1}\ \overline{y_2}\ X + y_2 \overline{X}$$

$$Z_1 = \overline{y_1}\ \overline{y_2}$$

$$Z_2 = y_1 \overline{y_2}\ X$$

$$Z_3 = y_1\ y_2$$

محاكيات الانظمة الالكترونية

SOLUTION

$y_1 y_2$ \ X	0	1
00	0	0
01	0	1
11	1	0
10	1	0

$$D_1 = y_1^+ = \overline{y_1}\, y_2 X + y_1 \overline{X}$$

$y_1 y_2$ \ X	0	1
00	0	1
01	1	0
11	1	0
10	0	0

$$D_2 = y_2^+ = \overline{y_1}\ \overline{y_2}\, X + y_2 \overline{X}$$

Binary-assignment table

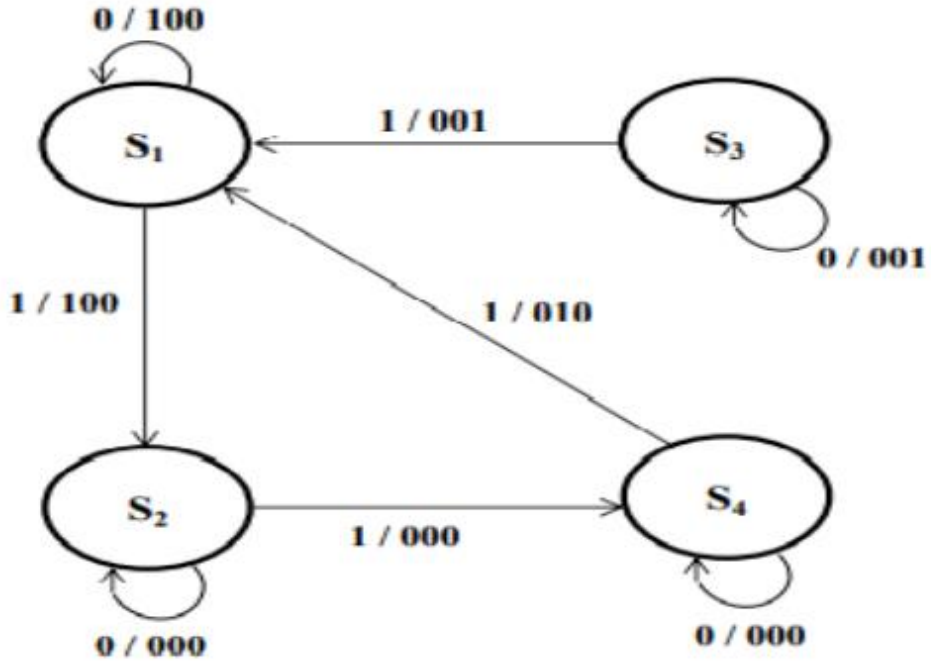
Present State $y_1 y_2$	Next State Input (X) 0 $y_1^+ y_2^+$	Next State Input (X) 1 $y_1^+ y_2^+$	Outputs Input (X) 0 $Z_1 Z_2 Z_3$	Outputs Input (X) 1 $Z_1 Z_2 Z_3$
0 0	0 0	0 1	1 0 0	1 0 0
0 1	0 1	1 0	0 0 0	0 0 0
1 1	1 1	0 0	0 0 1	0 0 1
1 0	1 0	0 0	0 0 0	0 1 0

محاكيات الانظمة الالكترونية

Let S1 = 00, S2 = 01, S3 = 11, S4 = 10. Therefore, the above table becomes

State table

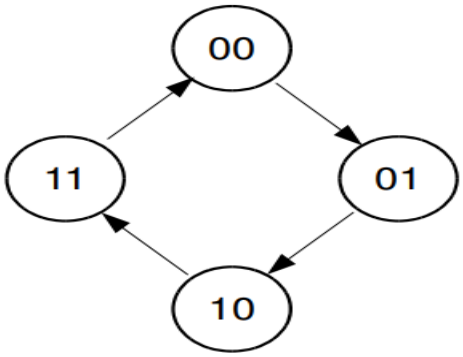
Present State	Next State Input (X) 0	Next State Input (X) 1	Outputs Input (X) 0 $Z_1Z_2Z_3$	Outputs Input (X) 1 $Z_1Z_2Z_3$
S_1	S_1	S_2	1 0 0	1 0 0
S_2	S_2	S_4	0 0 0	0 0 0
S_3	S_3	S_1	0 0 1	0 0 1
S_4	S_4	S_1	0 0 0	0 1 0



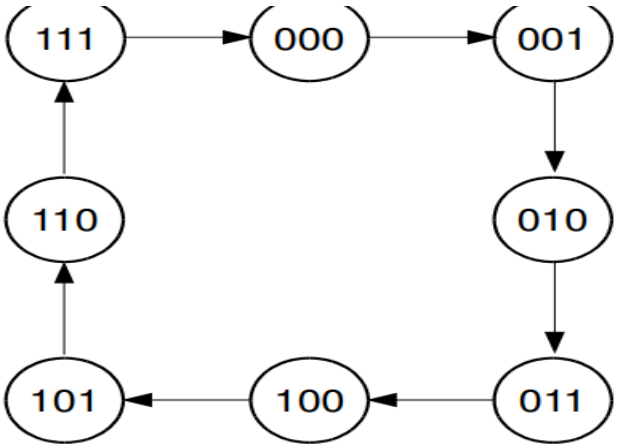
State Diagram

Counters

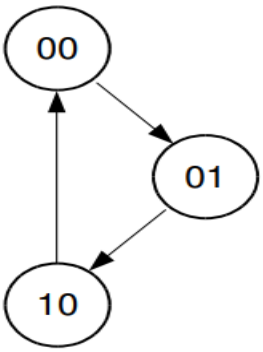
- A counter is a sequential circuit (aka. finite state machine) that cycles through a fixed sequence of states.
- The state of the counter is stored in Flip-Flops.
- An n-bit counter
  - has n Flip-Flop
  - can cycle through at most 2^n states.



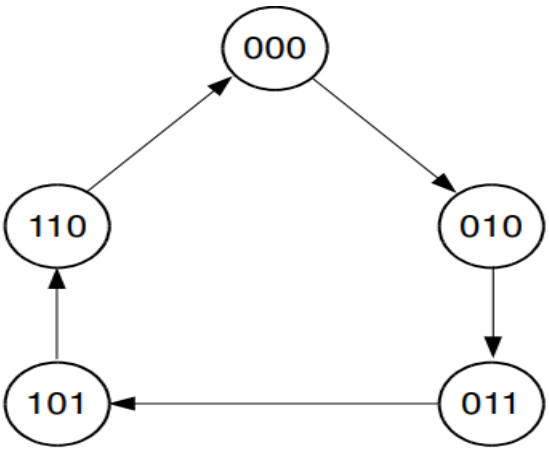
2-bit Counter



3-bit Counter



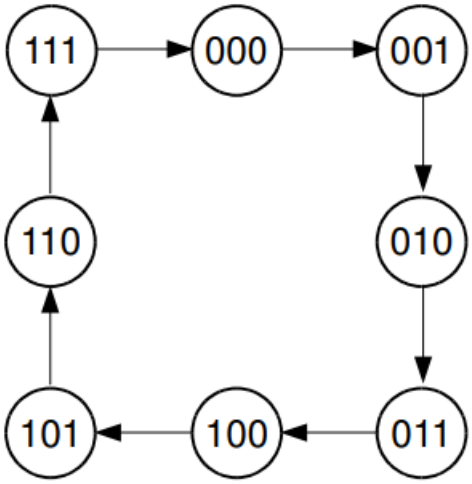
2-bit Counter
using only 3 states



3-bit Counter
using only 5 states

Binary Counters

- An n-bit binary counter is a counter that cycles through all 2 n states in ascending (or descending) order.



3-bit Binary Counter

Cycles through all 8 states in ascending order

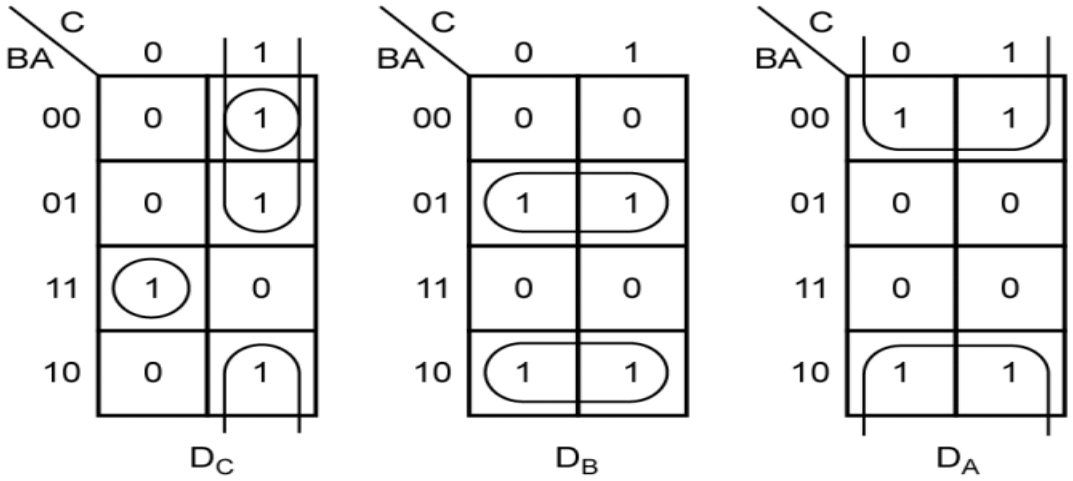
Binary Counters: Design

1. Draw a state graph that specifies the desired sequence Of the counter.
2. Construct a state table from the state graph. One Flip-Flop for each bit in the state.
3. Derive a K-map from the state table for each Flip-Flop. Input Select the type of Flip-Flop to be used.
4. Determine the input equation(s) for each Flip-Flop.

Example: State Table (using D FF) (0 to 7 to 0)

Present state			Next state			Flip-Flop inputs		
C	B	A	C^+	B^+	A^+	D_C	D_B	D_A
0	0	0	0	0	1			
0	0	1	0	1	0			
0	1	0	0	1	1			
0	1	1	1	0	0			
1	0	0	1	0	1			
1	0	1	1	1	0			
1	1	0	1	1	1			
1	1	1	0	0	0			DAWAH

Example: K-maps (for D FF inputs)



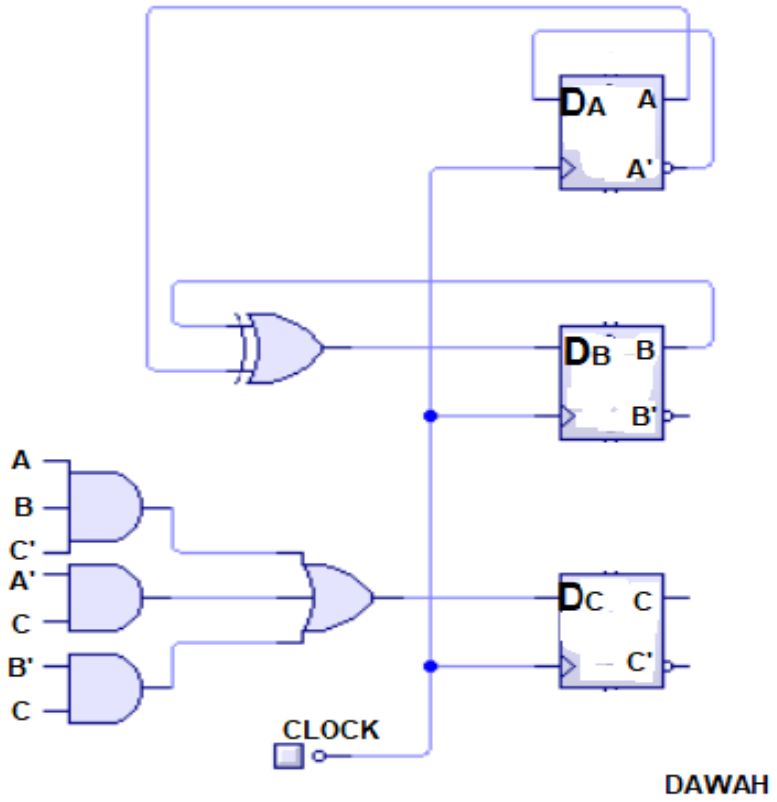
$D_A = A'$

$D_B = AB' + A'B = A \oplus B$

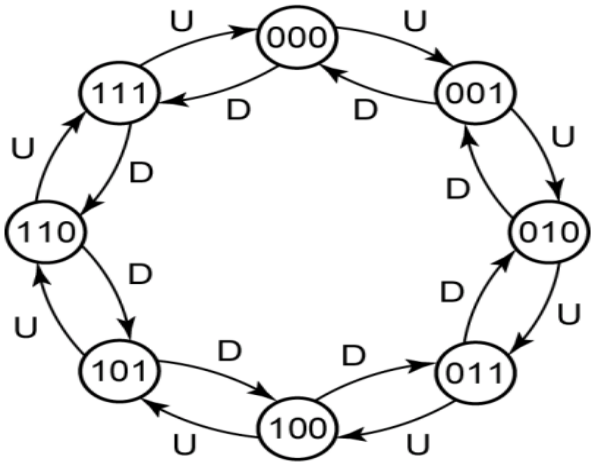
$D_C = B'C + A'C + ABC'$

Binary Counters: Design Example:

Circuit Diagram (using D FF)



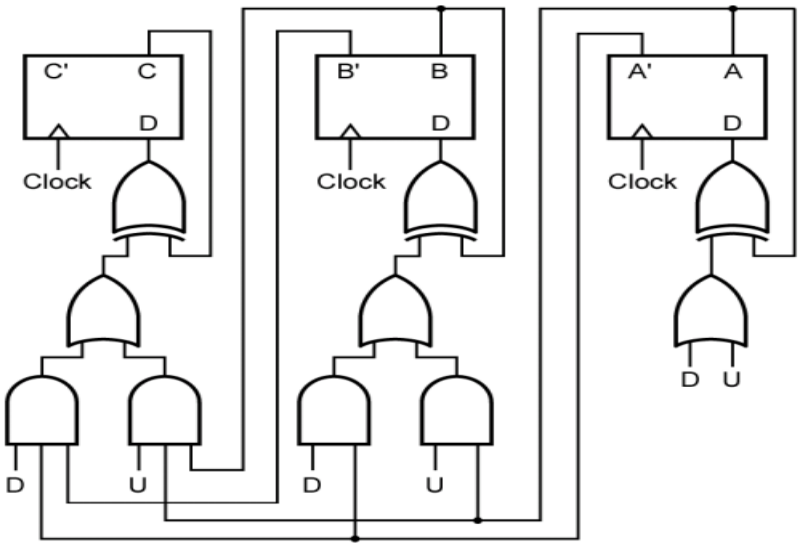
Binary Up-Down Counters.(U-UP),(D-DOWN)



CBA	$C^+B^+A^+$	
	U	D
000	001	111
001	010	000
010	011	001
011	100	010
100	101	011
101	110	100
110	111	101
111	000	110

What constraints must be placed on the U and D control signals?

Binary Up-Down Counters

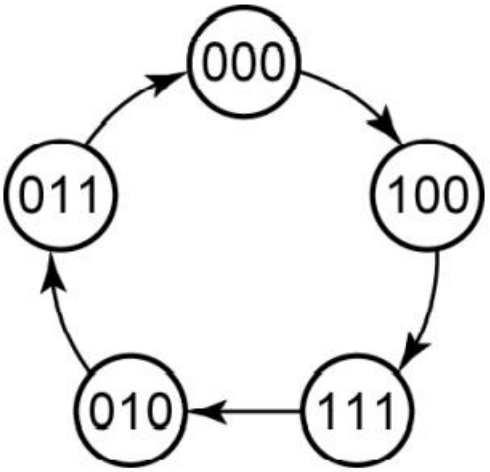


An **up/down counter** is one that is capable of progressing in either direction through a certain sequence. An up/down counter, sometimes called a bidirectional counter, can have
any specified sequence of states. A 3-bit binary counter that advances **upward** through its sequence
(0, 1, 2, 3, 4, 5, 6, 7)
and then can be reversed so that it goes through the
sequence in the opposite direction (7, 6, 5, 4, 3, 2, 1, 0) is an illustration of **up/down** sequential operation

UP UP
0, 1, 2, 3, 4, 5, 4, 3, 2, 3, 4, 5, 6, 7, 6, 5, etc.
DOWN DOWN

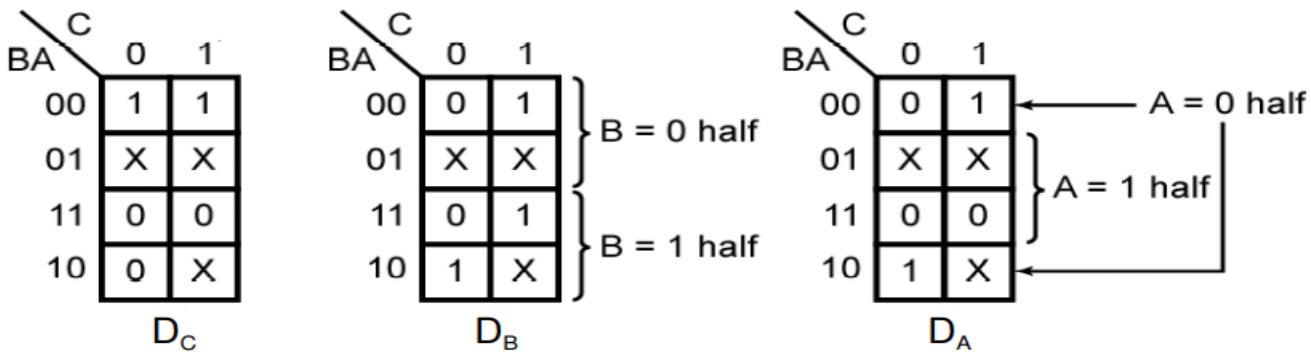
Example: Design the following counter using D- Flip-Flops.

محاكيات الانظمة الالكترونية

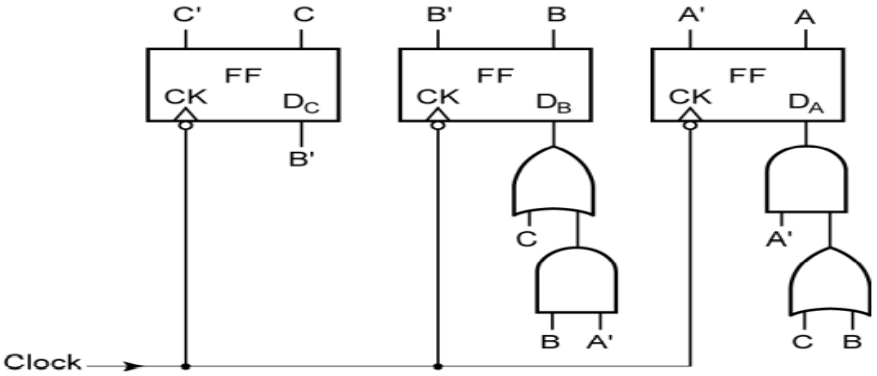


Present State			Next State			FF Inputs		
C	B	A	C^+	B^+	A^+	D_C	D_B	D_A
0	0	0	1	0	0			
0	0	1	x	x	x			
0	1	0	0	1	1			
0	1	1	0	0	0			
1	0	0	1	1	1			
1	0	1	x	x	x			
1	1	0	x	x	x			

Example: K-maps (for D FF inputs)



Circuit Diagram (using D FF)



General Models of Finite State Machines

A Moore state machine consists of combinational logic that determines the sequence and memory (flip-flops), as shown in Figure 9–1(a). A Mealy state machine is shown in part (b)

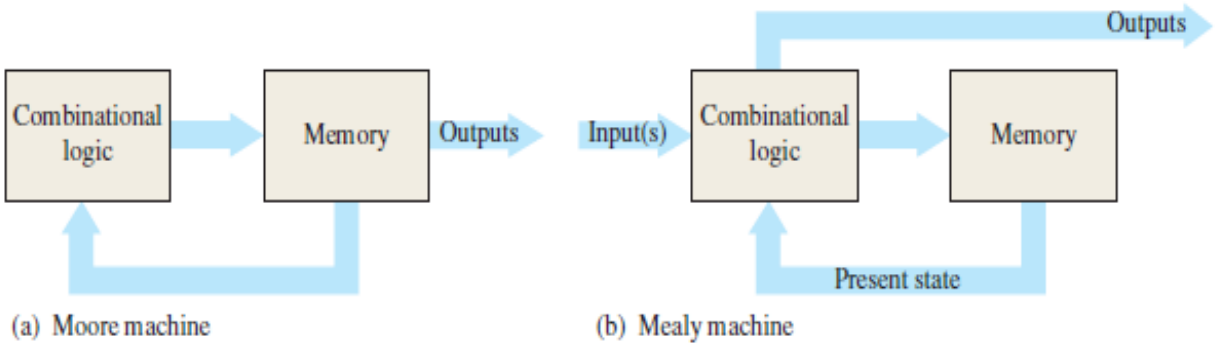


FIGURE 9–1 Two types of sequential logic.

FIGURE 9–1 Two types of sequential logic.

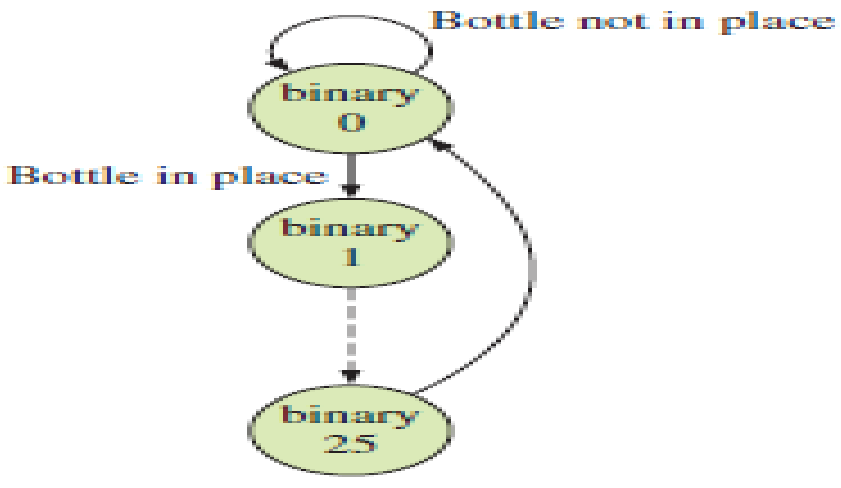
In the Moore machine, the combinational logic is a gate array with outputs that determine the next state of the flip-flops in the memory. There may or may not be inputs to the combinational logic. There may also be output combinational logic, such as a decoder.

If there is an input(s), it does not affect the outputs because they always correspond to and are dependent only on the present state of the memory. For the Mealy machine, the present state affects the outputs, just as in the Moore machine; but in addition, the inputs also affect the outputs.

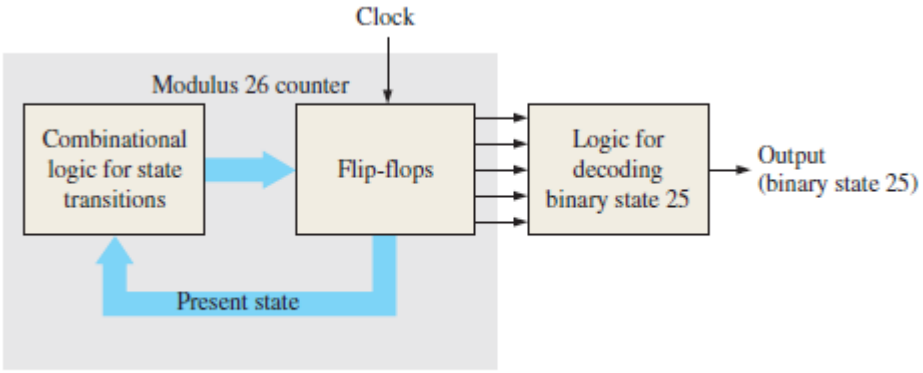
The outputs come directly from the combinational logic and not the memory.

Example of a Moore Machine

Figure 9–2(a) shows a Moore machine (modulus-26 binary counter with states (0 through25) that is used to control the number of tablets (25) that go into each bottle in an assembly
line. When the binary number in the memory (flip-flops) reaches binary twenty-five(11001), the counter recycles to 0 and the tablet flow and clock are cut off until the next bottle is in place.
The combinational logic for the state transitions sets the modulus of the counter so that it sequences from binary state 0 to binary state 25, where 0 is the reset or rest state
and the output combinational logic decodes binary state 25. There is no input in this case, other than the clock,
so the next state is determined only by the present state,
which makes this a Moore machine. One tablet is bottled for each clock pulse. Once a bottle is in place,
the first tablet is inserted at binary state 1, the second at binary state 2, and the twenty-fifth tablet when the binary state is 25.
Count 25 is decoded and used to stop the flow of tablets and the clock.
The counter stays in the 0 state until the next bottle is in position (indicated by a 1). Then the clock resumes, the count goes to 1, and the cycle repeats, as illustrated by the state diagram in Figure 9–2(b).



(b) State diagram

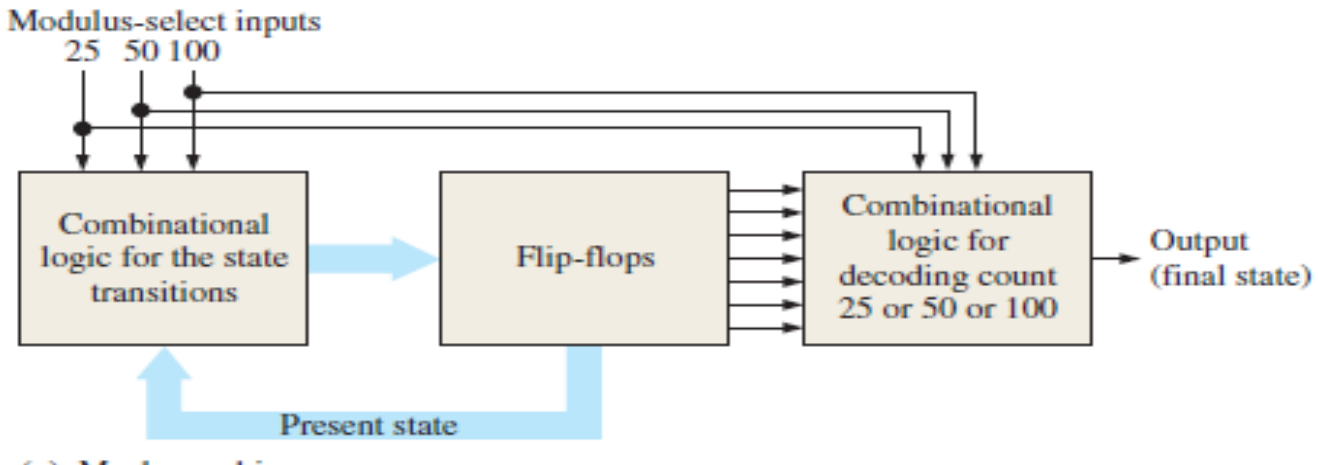


(a) Moore machine

FIGURE 9–2 A fixed-modulus binary counter as an example of a Moore state machine. The dashed line in the state **diagram means the states between binary 1 and 25 are not shown for simplicity**

Example of a Mealy Machine

Let's assume that the tablet-bottling system uses three different sizes of bottles: a 25-tablet bottle, a 50-tablet bottle, and a 100-tablet bottle. This operation requires a state machine with three different terminal counts: 25, 50, and 100.
One approach is illustrated in Figure 9–3(a).
The combinational logic sets the modulus of the counter depending on the modulus-select inputs.
The output of the counter depends on both the present state and the modulus-select inputs, making this a Mealy machine. The state diagram is shown in part (b).



(a) Mealy machine

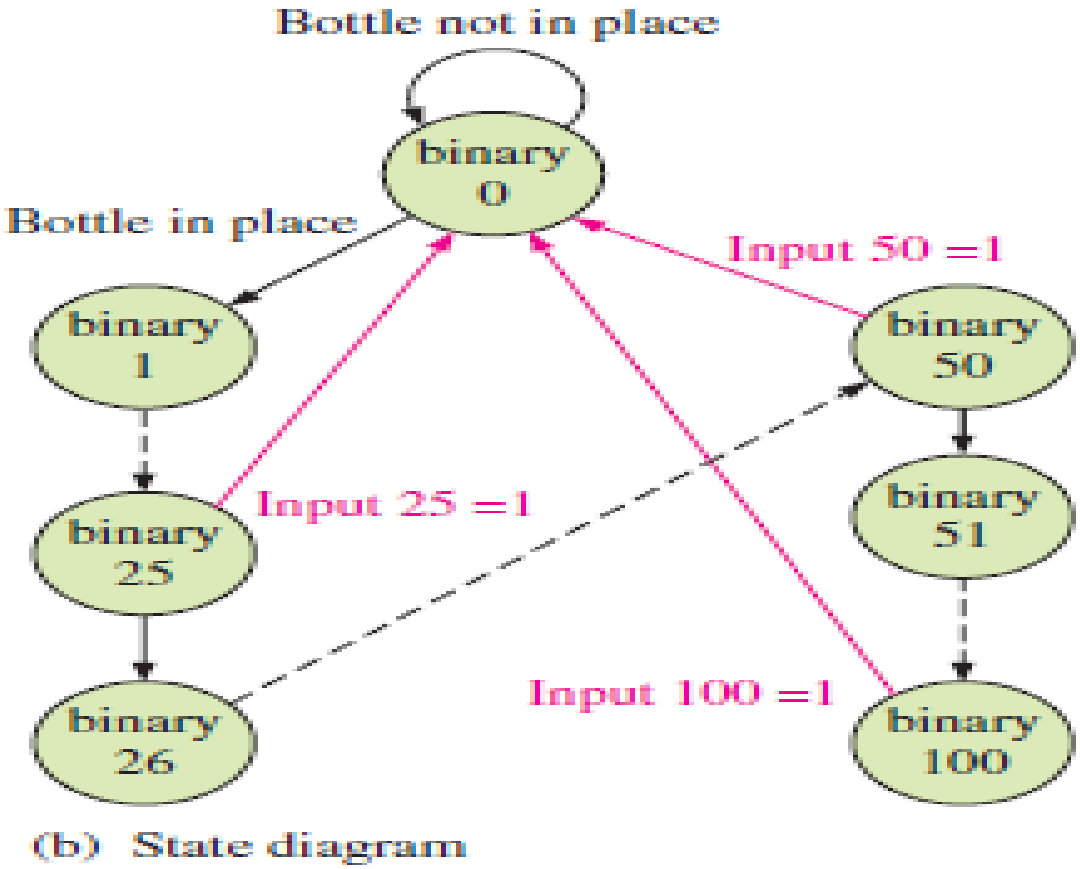


FIGURE 9–3 A variable-modulus binary counter as an example of a Mealy state machine. The red arrows in the state diagram represent the recycle paths that depend on the input
number. The black dashed lines mean the interim states are not shown for simplicity.

Asynchronous Counters

The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time. An asynchronous counter is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse
A counter is a register that goes through a predetermined

.sequence of states upon the application of clock pulses

Asynchronous counters

Synchronous counters

Asynchronous Counters (or Ripple counters)

the clock signal (CLK) is only used to clock the first FF

Each FF (except the first FF) is clocked by the preceding FF

Synchronous Counters

the clock signal (CLK) is applied to all FF, which means that all FF shares the same clock signal

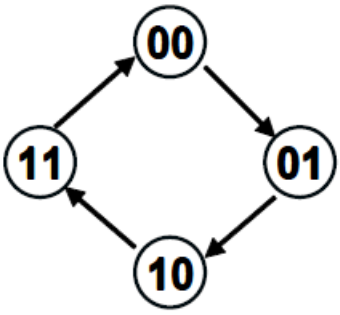
thus the output will change at the same time

Asynchronous (Ripple) UP Counters

The Asynchronous Counter that counts 4 number starts from 00 ,01,10,11 and back to 00 is called (MOD-4 Ripple Asynchronous Up-Counter)

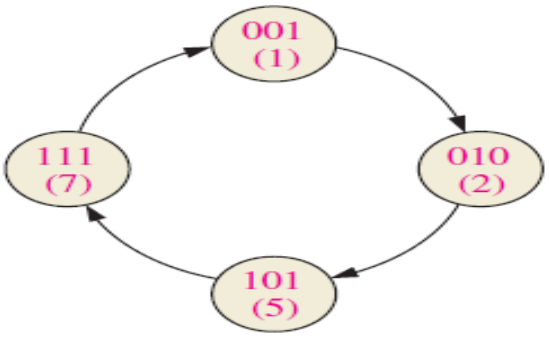
Next state table and state diagram

Present State	Next State
Q_1Q_0	Q_1Q_0
00	01
01	10
10	11
11	00



EXAMPLE

Design a counter with the irregular binary count sequence shown



in the state diagram of Figure Use D flip-flops.

Solution:

Step 1: The state diagram is as shown. Although there are only four states, a 3-bit counter binary states,
the invalid states (0, 3, 4, and 6) can be treated as “don’t cares” state, you must make sure that it goes back to a valid state.

Step 2: The next-state table is developed from the state diagram and is given in Table

Next-state table.

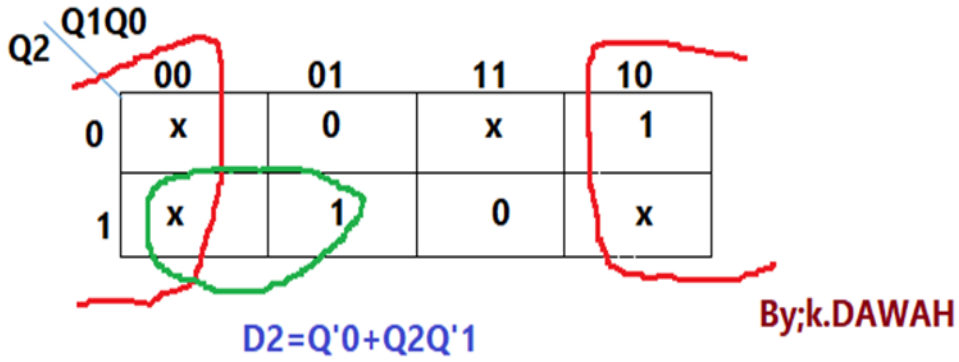
Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1

Step 3: The transition table for the D flip-flop is shown in Table

Transition table for a D flip-flop.

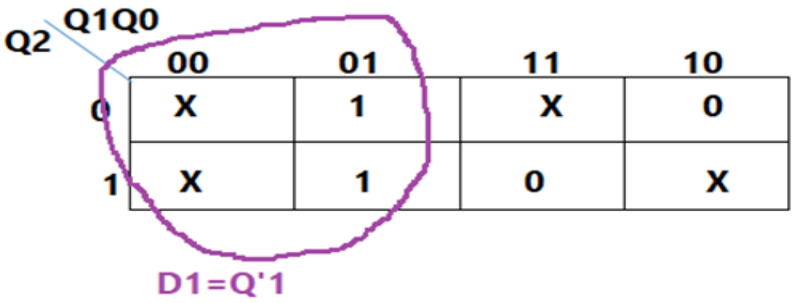
Output Transitions			Flip-Flop Input
Q_N		Q_{N+1}	D
0	⟶	0	0
0	⟶	1	1
1	⟶	0	0
1	⟶	1	1

Step 4: The *D* inputs are plotted on the present-state K-arnaugh maps Also “don’t cares” can be placed in the
cells corresponding to the invalid states of 000, 011, 100, and 110, as indicated by the red Xs.

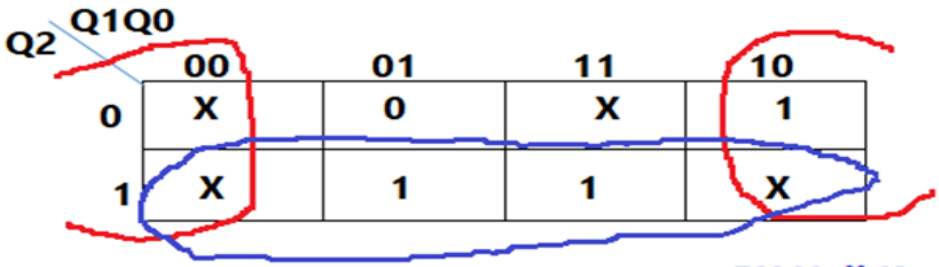


D2=Q'0+Q2Q'1

By;k.DAWAH



D1=Q'1



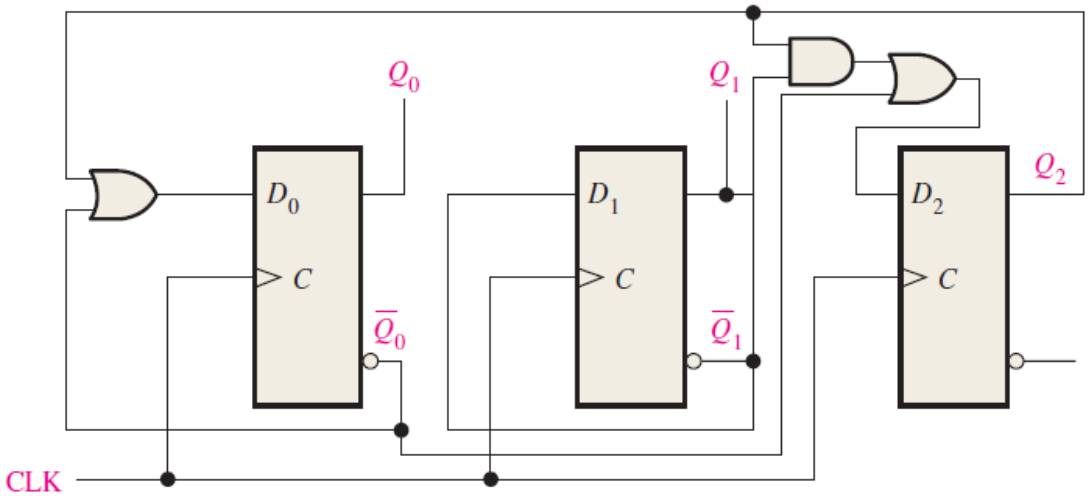
D0=Q'0 +Q2

BY Haji Kareem Dawah

Step 5: Group the 1s, taking advantage of as many of the “don’t care” states as possible for maximum simplification, as shown in The expression for each *D* input taken from the maps is as follows:

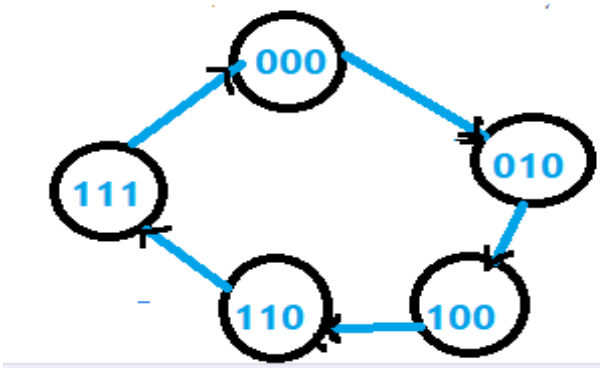
$$D_0 = \overline{Q}_0 + Q_2$$
$$D_1 = \overline{Q}_1$$
$$D_2 = \overline{Q}_0 + Q_2\overline{Q}_1$$

Step 6: The implementation of the counter is shown in Figure



EXAMPLE

Design a counter with the irregular binary count sequence shown in the state diagram of Figure Use D flip-flops.



محاكيات الانظمة الالكترونية

قسم هندسة تقنيات الحاسوب

Electronic Systems Simulators

Synchronous Asynchronus Sequential Networks

3E=2022-2023

MCA:DAWAH AUC-CET

كلية المعارف الجامعة

present state			next state		
Q2	Q1	Q0	Q2	Q1	Q0
0	0	0	0	1	0
0	1	0	1	0	0
1	0	0	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Q2 \ Q1Q0	00	01	11	10
0	0	x	x	1
1	1	x	0	1

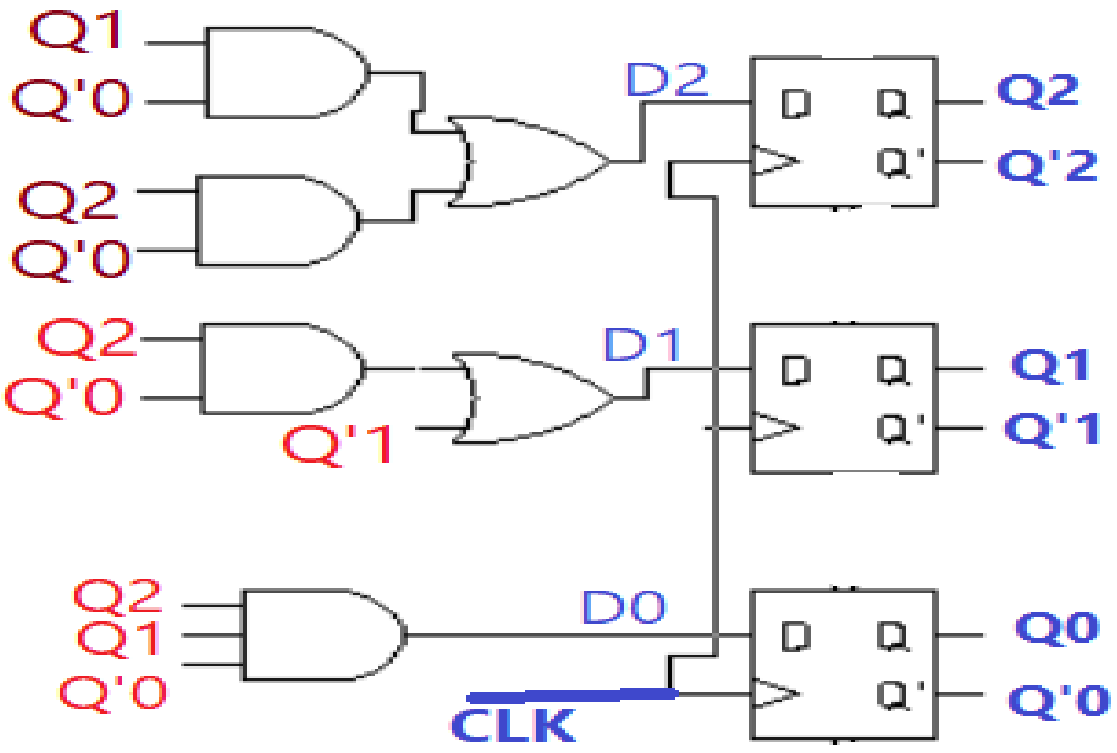
$D2 = Q1Q'0 + Q2\,Q'0$

Q2 \ Q1Q0	00	01	11	10
0	1	X	X	0
1	1	X	0	1

$D1 = Q1' + Q2Q'0$

Q2 \ Q1Q0	00	01	11	10
0	0	X	X	0
1	0	X	0	1

$D0=Q2Q1Q'0$



A Synchronous Counter Design Using D Flip-Flops

We will show how to design a synchronous counter which is capable of storing data and counting either up or down, based on input, using D flip-flops. Specifically, the counter will count up: 0, 1, 2, 3, 0, 1, 2, 3, … when the input x = 1, and count down when the input x = 0.

Suggested state definition tables, transition diagrams, transition tables, K-maps for the respective logic functions, and schematics of the implementation using flip-flops and logic gates for D flip-flop scenario will be given.

Brief Background

A flip-flop (also called a latch), is a circuit that has two stable states and is often used to store state information (e.g., on/off, 1/0, etc.). Indeed, it is a basic storage element used in sequential logic and a fundamental unit of digital electronic design for computer and communication systems, among others. the flip-flop circuit can change states when a signal is applied to one or more control inputs, conveniently resulting in one or two outputs. The flip-flop stores a single bit of data, and its two possible resulting states represent either a “one” or a “zero” condition. When used in finite-state machine design, the output and next state depend on both the current input as well as the current state, with the current state resulting from previous inputs. As a result, the flip-flop can be used to count pulses and synchronize variably-timed input signals with a basic reference signal .While the terms flip-flop and latch are sometimes used interchangeably, we generally refer to the unit as a flip-flop if it is clocked; if it is simple, we refer to it as a latch ,

The popular D (“data” or “delay”) flip-flop can really be thought of as a memory cell, a delay line, or a zero-order hold . Its true usefulness is its ability to capture the value of the D-input at a defined moment or portion of the clock cycle (such as the rising edge). This value, in turn, becomes the Q output , Inputs and resulting outputs can then be tracked and assessed by means of a truth table.

State	Definition	Binary values
S0	Reset/Initialize, no sequence	00
S1	Count 1	01
S2	Count 2	10
S3	Count 3	11

Figure 1: State Transition Diagram (D Flip-Flops)

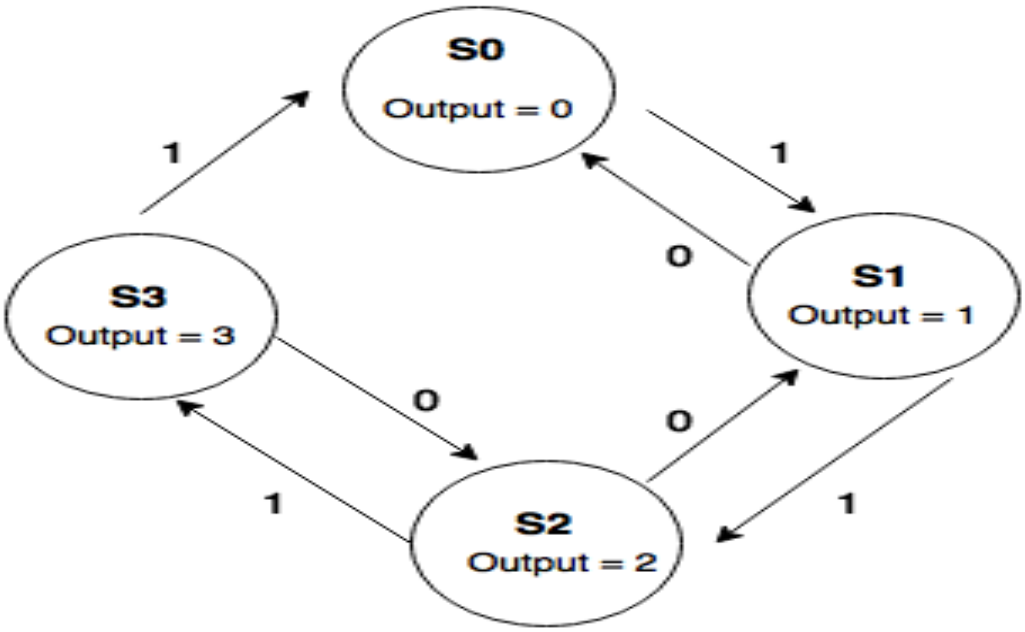


Table 2: State Transition Table (D Flip-Flops)

محاكيات الانظمة الالكترونية

Input	Present State			Next State			FF Inputs	
x	State	Q_1	Q_0	State	Q_1^+	Q_0^+	D_1	D_0
0	S_0	0	0	S_3	1	1	1	1
0	S_1	0	1	S_0	0	0	0	0
0	S_2	1	0	S_1	0	1	0	1
0	S_3	1	1	S_2	1	0	1	0
1	S_0	0	0	S_1	0	1	0	1
1	S_1	0	1	S_2	1	0	1	0
1	S_2	1	0	S_3	1	1	1	1
1	S_3	1	1	S_0	0	0	0	0

Synchronous Counter – State Transition Using D Flip-Flops

$x \setminus Q_1Q_0$	00	01	11	10
0	1	0	1	0
1	0	1	0	1

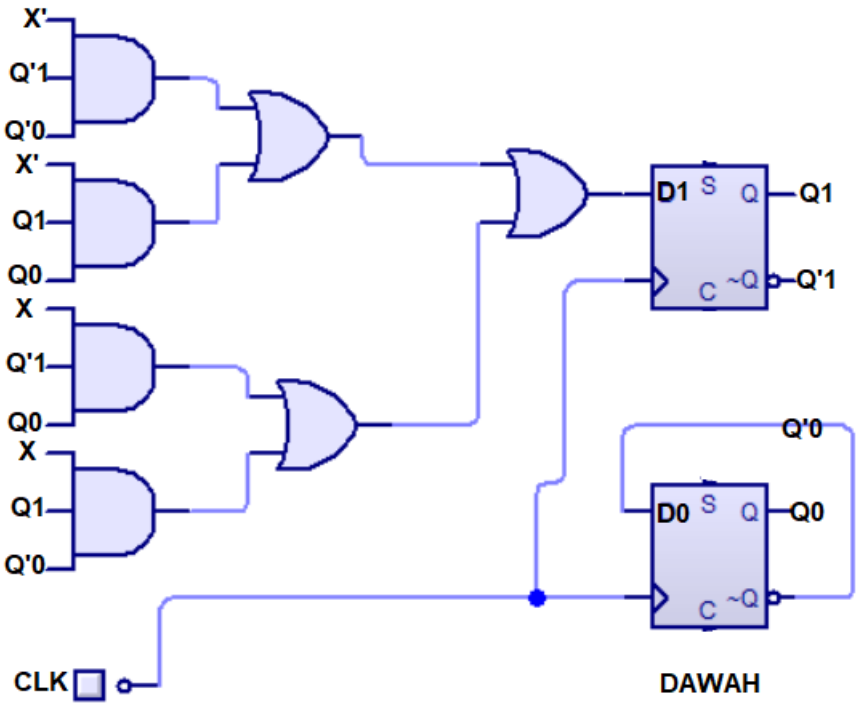
Synchronous Counter – (D_1) K-Map

$D_1 = x'Q_1'Q_0' + x'Q_1Q_0 + xQ_1'Q_0 + xQ_1Q_0'$

$x \setminus Q_1Q_0$	00	01	11	10
0	1	0	0	1
1	1	0	0	1

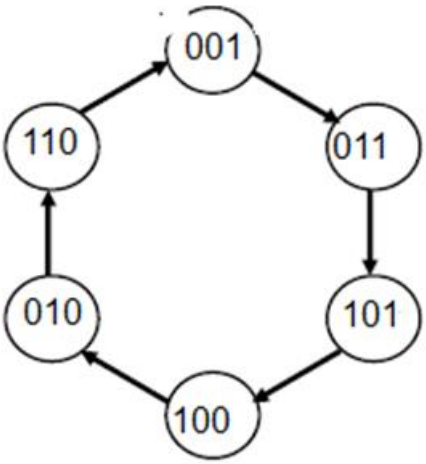
Synchronous Counter – (D_0) K-Map

$D_0 = \overline{Q}_0$



Example :Design a counter to count (1,4,5,3,2,6) using FSM?

Solution:



محاكيات الانظمة الالكترونية

قسم هندسة تقنيات الحاسوب

Electronic Systems Simulators

Synchronous Asynchronus Sequential Networks

3E=2022-2023

MCA:DAWAH AUC-CET

كلية المعارف الجامعة

Present state			Next state		
A	B	C	A'	B'	C'
0	0	0	x	x	x
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	x	x	x

by:k.DAWAH

A \ BC	00	01	11	10
0	X	0	1	1
1	0	1	X	0

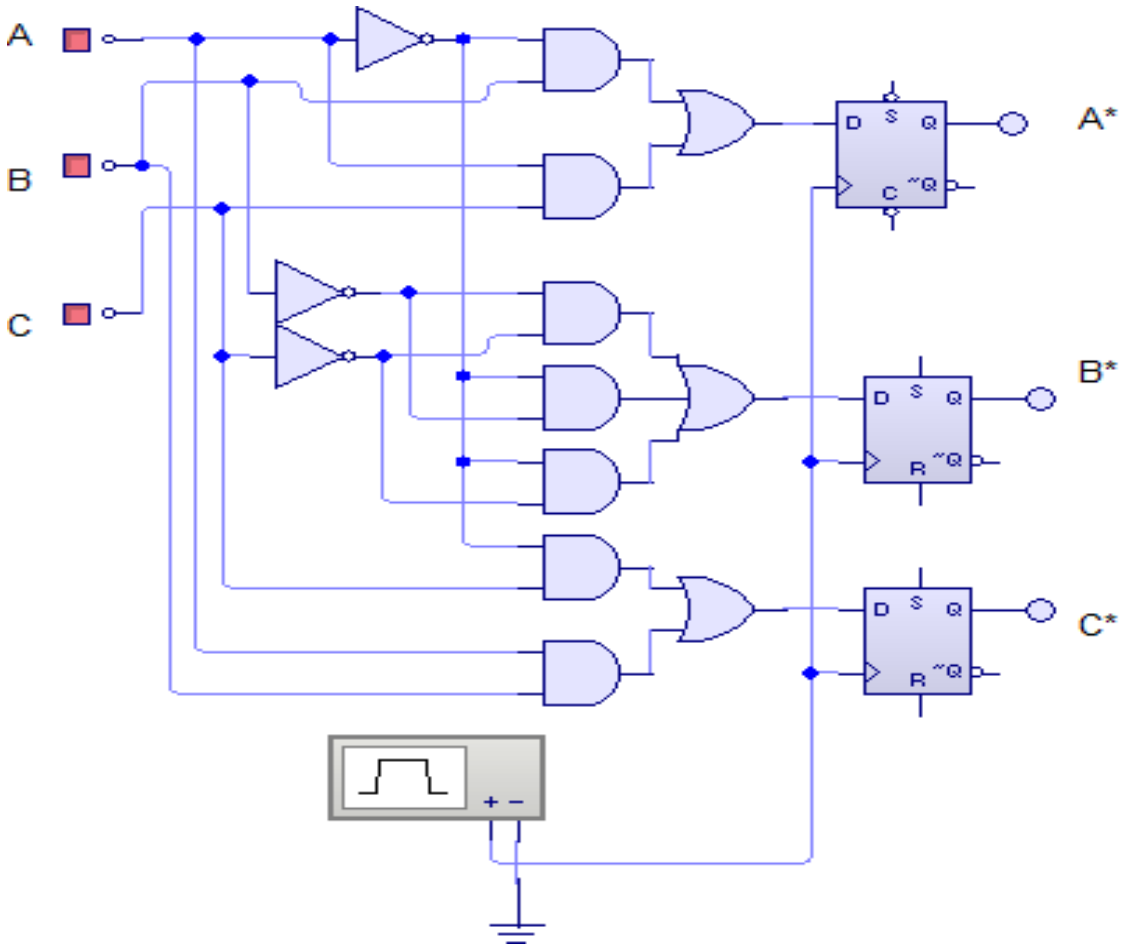
A \ BC	00	01	11	10
0	X	1	0	1
1	1	0	X	0

$A^* = \overline{A}\,B + AC$

$B^* = \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}$

A \ BC	00	01	11	10
0	X	1	1	0
1	1	0	X	1

$C^* = \overline{A}\,C + A\overline{C}$



A sequential circuit that goes through a prescribed sequence of states upon the application of input pulses is called a counter. The input pulses, called count pulses, may be clock pulses. In a counter, the sequence of states may follow a binary count or any other sequence of states. Counters are found in almost all equipment containing digital logic. They are used for counting the

number of occurrences of an even and are useful for generating timing sequences to control operations in a digital system.

A counter is a sequential circuit with 0 inputs and n outputs. Thus, the value after the clock transition depends only on old values of the outputs. For a counter, the values of the outputs are interpreted as a sequence of binary digits (see the section on binary arithmetic). We shall call the outputs o0, o1, ..., on-1. The value of the outputs for the counter after a clock transition is a binary number which is one plus the binary number of the outputs before the clock transition. We can explain this behavior more formally with a state table. As an example, let us take a counter with n = 4. The left side of the state table contains 4 columns, labeled o0, o1, o2, and o3. This means that the state table has 16 rows. Here it is in full:

o3	o2	o1	o0	o3′	o2′	o1′	o0′
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

As you can see, the right hand side of the table is always one plus the value of the left hand side of the table, except for the last line, where the value is 0 for all the outputs. We say that the counter wraps around.

Counters (with some variations) play an important role in computers. Some of them are visible to the programmer, such as the program counter (PC). Some of them are hidden to the programmer, and are used to hold values that are internal to the central processing unit, but nevertheless important. Important variations include:

- The ability to count up or down according to the value of an additional input
- The ability to count or not according the the value of an additional input
- The ability to clear the contents of the counter if some additional input is 1
- The ability to act as a register as well, so that a predetermined value is loaded when some additional input is 1 • The ability to count using a different representation of numbers from the normal (such as Gray-codes, 7- segment codes, etc)
- The ability to count with different increments that 1

Design of Counters

Example A counter is first described by a state diagram, which is shows the sequence of states through which the counter advances when it is clocked. Figure shows a state diagram of a 3-bit binary counter.

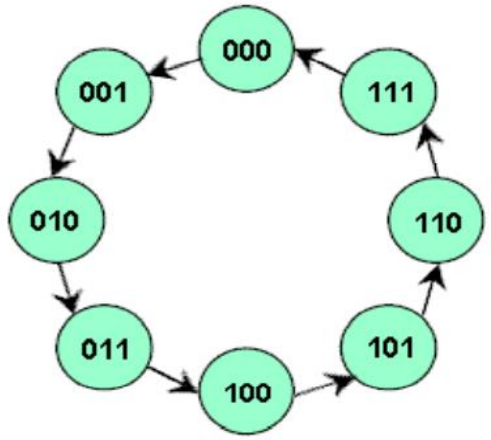
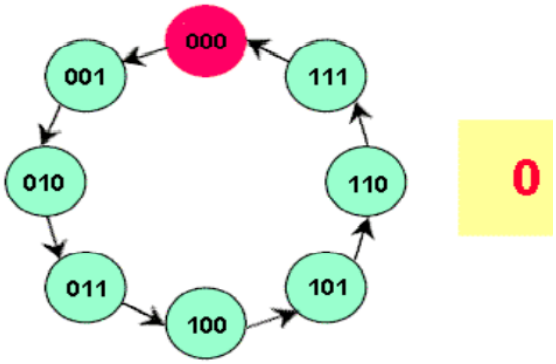


Figure . State diagram of a 3-bit binary counter.

The circuit has no inputs other than the clock pulse and no outputs other than its internal state (outputs are taken off each flip-flop in the counter). The next state of the counter depends entirely on its present state, and the state

transition occurs every time the clock pulse occurs. Figure shows the sequences of count after each clock pulse.



Once the sequential circuit is defined by the state diagram, the next step is to obtain the next-state table, which is derived from the state diagram in Figure and is shown in Table.

Table . State table

Present State	Next State
Q2 Q1 Q0	Q2 Q1 Q0
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0