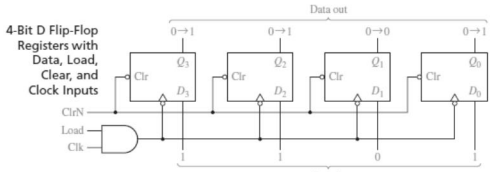
Module 5
Chapter 1: Registers and Counters.

5.1 REGISTERS AND REGISTER TRANSFERS:

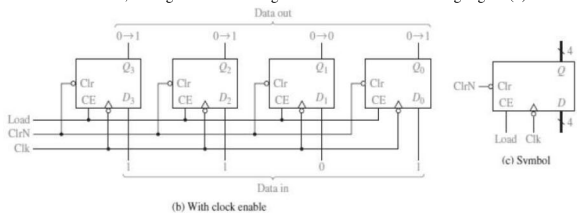
A *register* consists of a group of flip-flops with a common clock input. Registers are commonly used to store and shift binary data. *Counters* are another simple type of sequential circuits. A counter is usually constructed from two or more flip-flops which change states in a prescribed sequence when input pulses are received.



(a) Using gated clock

When Load = 0, the register is not clocked, and it holds its present value. Load = 1, the clock signal (Clk) is transmitted to the flip-flop clock inputs and the data applied to the D inputs will be loaded into the flip-flops on the falling edge of the clock. For example, if the Q outputs are 0000 (Q3 = Q2 = Q1 = Q0 = 0) and the data inputs are 1101 (D3 = 1, D2 = 1, D1 = 0 and D0 = 1), after the falling edge of clock, Q will change from 0000 to 1101 as indicated in the above Figure (The notation 0 → 1 at the flip-flop outputs indicates a change from 0 to 1). The flip-flops in the register have asynchronous clear inputs that are connected to a common clear signal, ClrN. The bubble at the clear inputs indicates that a logic 0 is required to clear the flip-flops. ClrN is normally 1, and if it is changed momentarily to 0, the Q outputs of all four flip-flops will become 0.

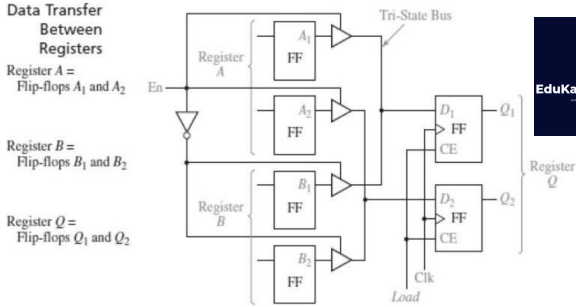
Gating the clock with another signal can cause timing problems. If flip-flops with clock enable are available, the register can be designed as shown in the following Figure (b).



(b) With clock enable

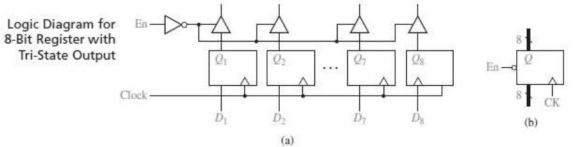
The load signal is connected to all four CE inputs. When Load = 0, the clock is disabled and the register holds its data. When Load = 1, the clock is enabled, and the data applied to the D inputs will be loaded into the flip-flops, following the falling edge of the clock. Figure (c) shows a symbol for the 4-bit register using bus notation for the D inputs and Q outputs. A group of wires that perform a common function is often referred to as a bus. A heavy line is used to represent a bus, and a slash with a number beside it indicates the number of bits in the bus. Transferring data between registers is a common operation in digital systems. The following Figure shows

how data can be transferred from the output of one of two registers into a third register using tri-state buffers.

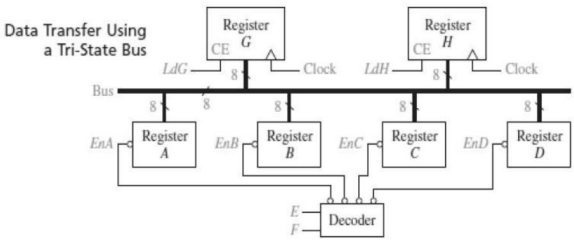


If En = 1 and Load = 1, the output of register A is enabled onto the tri-state bus and the data in register A will be stored in Q after the rising edge of the clock. If En = 0 and Load = 1, the output of register B will be enabled onto the tri-state bus and stored in Q after the rising edge of the clock.

The following Figure (a) shows an integrated circuit register that contains eight D flip-flops with tri-state buffers at the flip-flop outputs. These buffers are enabled when En = 0. A symbol for this 8-bit register is shown in Figure (b).



The following Figure (c) shows how data can be transferred from one of four 8-bit registers into one of two other registers. Registers A, B, C, and D are of the type shown in above Figure.



The outputs from these registers are all connected in parallel to a common tri-state bus. Registers G and H are 8-bit D type registers (PIPO). The flip-flop inputs of registers G and H are also connected to the bus. When *EnA = 0*, the tri-state outputs of register A are enabled onto the bus.

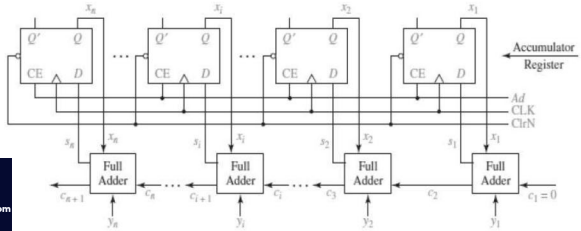
If *LdG = 1*, these signals on the bus are loaded into register G after the rising clock edge (or into register H if *LdH = 1)*. Similarly, the data in register B, C, or D is transferred to G (or H) when *EnB*, *EnC*, or *EnD* is 0, respectively and *LdG = 1* (or *LdH = 1*). If *LdG = LdH = 1*, both G and H will be loaded from the bus. The four enable signals may be generated by a decoder. The operation can be summarized as follows:

If *EF = 00*, A is stored in G (or H) If EF = 01, B is stored in G (or H).
If EF = 10, C is stored in G (or H) If EF = 11, D is stored in G (or H).

Note that 8 bits of data are transferred in parallel from register A, B, C, or D to register G or H. As an alternative to using a bus with tri-state logic, eight 4-to-1 multiplexers could be used, but this would lead to a more complex circuit.

Parallel Adder with Accumulator:

In computer circuits, it is frequently desirable to store one number in a register of flip-flops (called an accumulator) and add a second number to it, leaving the result stored in the accumulator. One way to build a parallel adder with an accumulator is to add a register to the adder as shown in the following Figure.

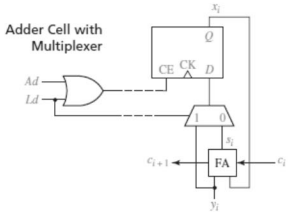


Suppose that the number X = xn . . . x2x1 is stored in the accumulator. Then, the number Y = yn . . . y2y1 is applied to the full adder inputs, and after the carry has propagated through the adders, the sum of X and Y appears at the adder outputs. An add signal (Ad) is used to load the adder outputs into the accumulator flip-flops on the rising clock edge. If si = 1, the next state of flip-flop xi will be 1. If si = 0, the next state of flip-flop xi will be 0.Thus, xi+ = si, and if Ad = 1, the number X in the accumulator is replaced with the sum of X and Y, following the rising edge of the clock.

Observe that the adder with accumulator is an iterative structure that consists of a number of identical cells. Each cell contains a full adder and an associated accumulator flip-flop. Cell i, which has inputs ci and yi and outputs ci = 1 and xi, is referred to as a typical cell.

Before addition can take place, the accumulator must be loaded with X. This can be accomplished in several ways. The easiest way is to first clear the accumulator using the asynchronous clear inputs on the flip-flops, and then put the X data on the Y inputs to the adder and add to the accumulator in the normal way. Alternatively, we could add multiplexers at the accumulator inputs so that we could select either the Y input data or the adder output to load into the accumulator. This would eliminate the extra step of clearing the accumulator but would add to the hardware complexity.

The following Figure shows a typical cell of the adder where the accumulator flip-flop can either be loaded directly from yi or from the sum output (si).When Ld = 1 the multiplexer selects yi, and yi is loaded into the accumulator flip-flop (xi) on the rising clock edge. When Ad = 1 and Ld = 0, the adder output (si) is loaded into xi. The Ad and Ld signals are ORed together to enable the clock when either addition or loading occurs. When Ad = Ld = 0, the clock is disabled and the accumulator outputs do not change.



5.2 SHIFT REGISTERS:

A shift register is a register in which binary data can be stored, and this data can be shifted to the left or right when a shift signal is applied. Bits shifted out one end of the register may be lost, or if the shift register is of cyclic type, bits shifted out one end are shifted back in the other end. There are two ways to shift data into a register – serial or parallel; and there are two ways to shift the data out of the register – serial or parallel. This leads to the construction of four basic types of registers, as shown in the following Figure. All of these configurations are commercially available as TTL MSI/LSI circuits.

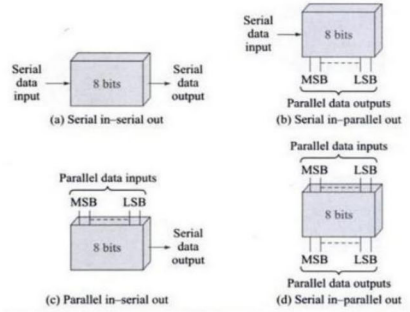
Examples:

Serial in – serial out (SISO): 54/74LS91, 8-bits

Serial in – parallel out (SIPO): 54/74164, 8-bits

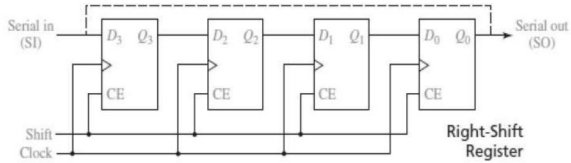
Parallel in – serial out (PISO): 54/74165, 8-bits

Parallel in – parallel out (PIPO): 54/74194, 4-bits & 54/74168, 8-bits.



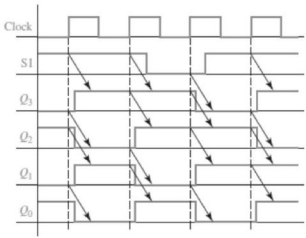
Types of Shift Registers.

The following Figure (a) illustrates a 4-bit right-shift register with serial input and output constructed from D flip-flops. When Shift = 1, the clock is enabled and shifting occurs on the rising clock edge. When Shift = 0, no shifting occurs and the data in the register is unchanged. The serial input (SI) is loaded into the first flip-flop (Q3) by the rising edge of the clock. At the same time, the output of the first flip-flop is loaded into the second flip-flop, the output of the second flip-flop is loaded into the third flip-flop, and the output of the third flip-flop is loaded into the last flip-flop. Because of the propagation delay of the flip-flops, the output value loaded into each flip-flop is the value before the rising clock edge.



(a) Flip-flop connections

Figure (b) illustrates the timing when the shift register initially contains 0101 and the serial input sequence is 1, 1, 0, 1.The sequence of shift register states is 0101, 1010, 1101, 0110, 1011.

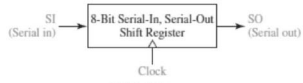


(b) Timing diagram

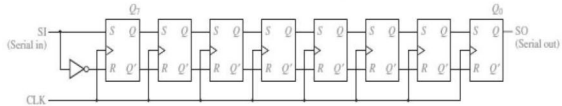
If we connect the serial output to the serial input, as shown by the dashed line, the resulting cyclic shift register performs an end-around shift. If the initial contents of the register is 0111, after one clock cycle the contents is 1011.After a second pulse, the state is 1101, then 1110, and the fourth pulse returns the register to the initial 0111 state.

Shift registers with 4, 8, or more flip-flops are available in integrated circuit form. The following Figure illustrates an 8-bit serial-in, serial-out shift register. Serial in means that data is shifted into the first flip-flop one bit at a time, and the flip-flops cannot be loaded in parallel. Serial out means that data can only be read out of the last flip-flop and the outputs from the other flip-flops are not connected to terminals of the integrated circuit.

8-Bit Serial-in, Serial-out Shift Register

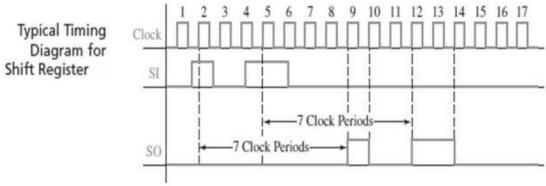


(a) Block diagram

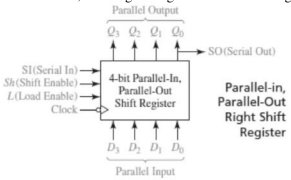


(b) Logic diagram

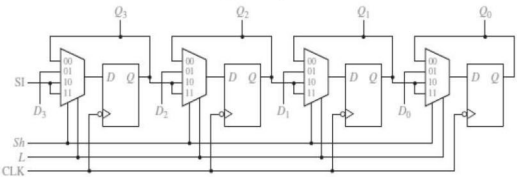
The inputs to the first flip-flop are S = SI and R = SI'. Thus, if SI = 1, a 1 is shifted into the register when it is clocked, and if SI = 0, a 0 is shifted in. The following Figure shows a typical timing diagram.



The following Figure (a) shows a 4-bit parallel-in, parallel-out shift register. Parallel-in implies that all four bits can be loaded at the same time, and parallel-out implies that all bits can be read out at the same time. The shift register has two control inputs, shift enable (*Sh*) and load enable (*L*). If *Sh = 1* (and *L = 1* or *L = 0*), clocking the register causes the serial input (*SI*) to be shifted into the first flip-flop, while the data in flip-flops Q3,Q2, and Q1 are shifted right. If *Sh = 0* and *L = 1*, clocking the shift register will cause the four data inputs (D3, D2, D1, D0) to be loaded in parallel into the flip-flops. If *Sh = L = 0*, clocking the register causes no change of state.



(a) Block diagram



(b) Implementation using flip-flops and MUXes

The following Table summarizes the operation of this shift register. All state changes occur immediately following the falling edge of the clock.

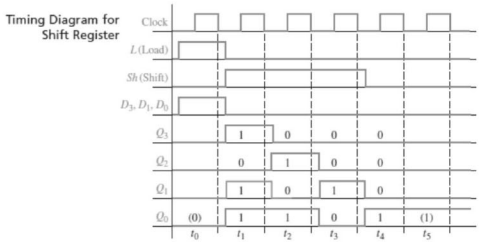
Shift Register Operation

Inputs		Next State				
Sh (Shift)	*L* (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	Action
0	0	Q_3	Q_2	Q_1	Q_0	No change
0	1	D_3	D_2	D_1	D_0	Load
1	X	SI	Q_3	Q_2	Q_1	Right shift

The shift register can be implemented using MUXes and D flip-flops, as shown in the above Figure (b). For the first flip-flop, when Sh = L = 0, the flip-flop Q3 output is selected by the MUX, so Q3+ = Q3 and no state change occurs. When Sh = 0 and L = 1, the data input D3 is selected and loaded into the flip-flop. When Sh = 1 and L = 0 or 1, SI is selected and loaded into the flip-flop. The second MUX selects Q2, D2, or Q3, etc. The next-state equations for the flip-flops are

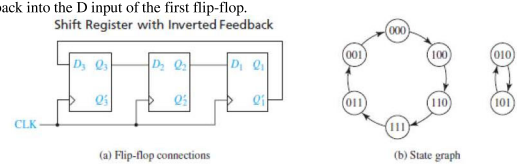
$$Q_3^+ = Sh' \cdot L' \cdot Q_3 + Sh' \cdot L \cdot D_3 + Sh \cdot SI$$
$$Q_2^+ = Sh' \cdot L' \cdot Q_2 + Sh' \cdot L \cdot D_2 + Sh \cdot Q_3$$
$$Q_1^+ = Sh' \cdot L' \cdot Q_1 + Sh' \cdot L \cdot D_1 + Sh \cdot Q_2$$
$$Q_0^+ = Sh' \cdot L' \cdot Q_0 + Sh' \cdot L \cdot D_0 + Sh \cdot Q_1$$

A typical application of this register is the conversion of parallel data to serial data. The output from the last flip-flop (Q0) serves as a serial output as well as one of the parallel outputs. The following Figure shows a typical timing diagram.



The first clock pulse loads data into the shift register in parallel. During the next four clock pulses, this data is available at the serial output. Assuming that the register is initially clear (Q3Q2Q1Q0 = 0000), that the serial input is SI = 0 throughout, and that the data inputs D3D2D1D0 are 1011 during the load time (t0), the resulting waveforms are as shown. Shifting occurs at the end of t1, t2, and t3, and the serial output can be read during these clock times. During t4, Sh = L = 0, so no state change occurs.

The following Figure (a) shows a 3-bit shift register with the Q1 output from the last flip-flop fed back into the D input of the first flip-flop.



(a) Flip-flop connections (b) State graph

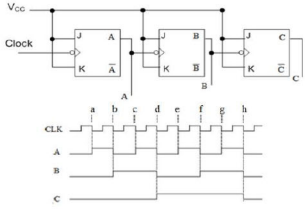
If the initial state of the register is 000, the initial value of D3 is 1, so after the first clock pulse, the register state is 100. Successive states are shown on the state graph of Figure (b). When the register is in state 001, D3 is 0, and the next register state is 000. Then, successive clock pulses take the register around the loop again. Note that states 010 and 101 are not in the main loop. If the register is in state 010, then a shift pulse takes it to 101 and vice versa; therefore, we have a secondary loop on the state graph. A circuit that goes through a fixed sequence of states is called a counter. A shift register with inverted feedback (Q1' connected to D3, as shown in above Figure) is often called a Johnson counter or Twisted ring counter. A shift register with non-inverted feedback (if Q1 connected to D3, in above Figure) is often called a Ring counter.

5.3 DESIGN OF BINARY COUNTERS:

A counter is a sequential circuit that goes through a prescribed sequence of states up on the application of input pulse. Counters are in two categories –

- **Ripple (Asynchronous) Counter** – consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the clock-pulse input of the next higher-order flip-flop. The flip-flop holding the LSB receives the clock-pulse.
- **Synchronous Counter** – the input pulses/ clock-pulses are applied to all clock-pulse inputs of all the flip-flops simultaneously.

Asynchronous Counters:



Negative Clock	C	B	A	State or Count
-	0	0	0	0
a	0	0	1	1
b	0	1	0	2
c	0	1	1	3
d	1	0	0	4
e	1	0	1	5
f	1	1	0	6
g	1	1	1	7
h	0	0	0	0 repeats

3-bit Binary Ripple Counter, Waveforms and Truth Table

The above Figure shows three negative-edge-triggered, JK flip-flops connected in cascade to form a 3-bit ripple counter. The system clock (a square wave), drives flip-flop A. The output of flip-flop A drives B, and the output of B drives flip-flop C. All the J and K inputs are tied to +VCC. Hence, flip-flops will toggle with a negative transition at its clock input.

Assume that, the flip-flops are all initially reset to 0 outputs. For every clock NT, flip-flop A will change state. Notice that, the waveform at the output of flip-flop A is one-half the clock frequency.

Since, A acts as clock for B, each time the waveform at A goes low, flip-flop B will toggle. Notice that, the waveform at the output of flip-flop B is one-half the frequency of A and one-fourth the clock frequency.

Since, B acts as clock for C, each time the waveform at B goes low, flip-flop C will toggle. The waveform at the output of flip-flop C is one-half the frequency of B and one-eighth the clock frequency.

Problem: What is the clock frequency of a 3-bit ripple counter, if the period of the MSB waveform is 24 μs?

Solution: Since there are eight clock cycles in one cycle of MSB, the period of the clock must be 24/8 = 3 μs. The clock frequency must be $1/(3*10^{-6}) = 333$ KHz.

NOTE:

1. A binary ripple counter in straight binary sequence will be as shown in the above table. A ripple counter having *n* flip-flops will have 2_n output conditions. For example, the three-flip-flop counter has $2_3 = 8$ output conditions (000 to 111).

2. A three-flip-flop counter is often referred to as modulus-8 (or Mod-8) counter, since it has eight states. The *modulus* of a counter is the total number of states through which the counter can progress.

NOTE:

No. of Flip-Flops	*1*	*2*	*3*	*4*	*5*	*n*
No. of States	*2*	*4*	*8*	*16*	*32*	2^n

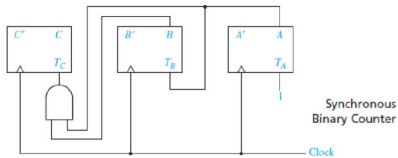
Problem: How many flip-flops are required to construct a mod-128 counter? A mod-32 counter? What is the largest decimal number that can be stored in a mod-64 counter?
Solution: A mod-128 counter must have seven flip-flops, since 27 = 128. Five flip-flops are needed to construct a mod-32 counter. The largest decimal number that can be stored in a mod-64 (six flip-flops) counter is 111111 = 63.

Synchronous Counters:

Synchronous means the operation of the flip-flops is synchronized by a common clock pulse so that when several flip-flops must change state, the state changes occur simultaneously.

Synchronous Binary Counters Using T Flip-Flops:

Consider the following Figure, a binary counter using three T flip-flops to count clock pulses.



All the flip-flops change state a short time following the rising edge of the input pulse. The state of the counter is determined by the states of the individual flip-flops; for example, if flip-flop C is in state 0, B in state 1, and A in state 1, the state of the counter is 011. Initially, assume that all flip-flops are set to the 0 state. When a clock pulse is received, the counter will change to state 001; when a second pulse is received, the state will change to 010, etc. The sequence of flip-flop states is CBA = 000, 001, 010, 011, 100, 101, 110, 111, 000, . . . Note that, when the counter reaches state 111, the next pulse resets it to the 000 state, and then the sequence repeats. First, design the counter by inspection of the counting sequence; then, use a systematic procedure which can be generalized to other types of counters. The problem is to determine the flip-flop inputs—TC,TB, and TA. From the preceding counting sequence, observe that A changes state every time a clock pulse is received. Because A changes state on every rising clock edge, TA must equal 1. Next, observe that B changes state only if A = 1.Therefore, A is connected to TB as shown, so that if A = 1, B will change state when a rising clock edge occurs. Similarly, C changes state when a rising clock edge occurs only if B and A are both 1. Therefore, an AND gate is connected to TC so that C will change state if B = 1 and A = 1 when a rising clock edge occurs. Now, verify that the circuit of above Figure counts properly by tracing signals through the circuit. Initially, CBA = 000, so only TA is 1 and the state will change to 001 when the first active clock edge arrives. Then, TB = TA = 1, and the state will change to 010 when the second active clock arrives. This process continues until finally when state 111 is reached, TC = TB = TA = 1, and all flip-flops return to the 0 state.
A state table (the following Table) shows the present state of flip-flops C, B, and A (before a clock pulse is received) and the corresponding next state (after the clock pulse is received). For example, if the flip-flops are in state CBA = 011 and a clock pulse is received, the next state will be C+ = B+ = A+ = 100. Although the clock is not explicit in the table, it is understood to be the input that causes the counter to go to the next state in sequence. A third column in the table is used to derive the inputs for TC,TB, and TA. Whenever the entries in the A and A+

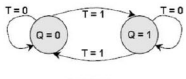
columns differ, flip-flop A must change state and TA must be 1. Similarly, if B and B+ differ, B must change state so TB must be 1. For example, if CBA = 011, C+B+A+ = 100, all three flip-flops must change state, so TCTBTA = 111.

State Table for Binary Counter

Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
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

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table



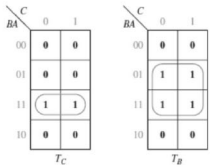
State Diagram

$Q \rightarrow Q_{n+1}$	T
0	0
0	1
1	0
1	1

Excitation Table

TC, TB, and TA are now derived from the table as functions of C, B, and A. By inspection, TA = 1. The following Figure shows the Karnaugh maps for TC and TB, from which; TC = BA and TB = A. These equations yield the same circuit derived previously.

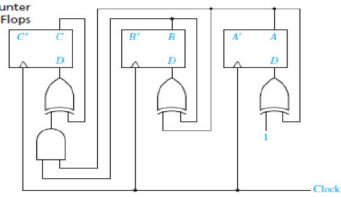
Karnaugh Maps for Binary Counter



Synchronous Binary Counters Using D Flip-Flops:

Next, redesign the binary counter to use D flip-flops instead of T flip flops. The easiest way to do this is to convert each D flip-flop to a T flip-flop by adding an XOR (exclusive-OR) gate, as shown in the following Figure (The rightmost XOR gate can be replaced with an inverter because A $\oplus$ 1 = A).

Binary Counter with D Flip-Flops

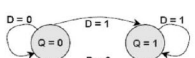


We can also derive the D flip-flop inputs for the binary counter starting with its state table (given below).

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	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

Characteristic Table



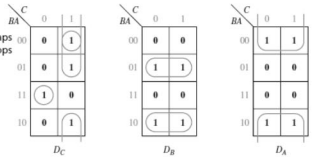
State Diagram

$Q \rightarrow Q_{n-1}$	D
0	0
0	1
1	0
1	1

Excitation Table

For a D flip-flop, Q+ = D. By inspection of the state table, the maps for DA, DB and DC are plotted as follows and D input equations are derived. This give the same logic circuit as was obtained by inspection.

Karnaugh Maps for D Flip-Flops

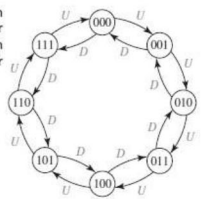


$D_A = A^+ = A'$
$D_B = B^+ = BA' + B'A = B \oplus A$
$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$

Binary Up-Down Counter:

The state graph and table for an up-down counter are shown in the following Figure.

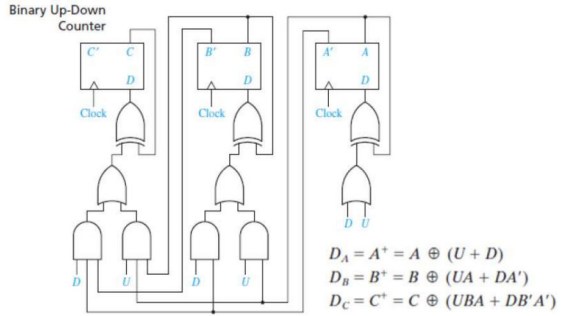
State Graph and Table for Up-Down Counter



	$C^+B^+A^+$	
CBA	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

When U = 1, the counter counts up in the sequence 000, 001, 010, 011, 100, 101, 110, 111, 000 . . . When D = 1, the counter counts down in the sequence 000, 111, 110, 101, 100, 011, 010, 001, 000 . . .When U = D = 0, the counter state does not change, and U = D = 1 is not allowed.

By inspection of the table in above Figure, we can verify that these are the correct equations for a down counter. For every row of the table, $A+ = A'$, so A changes state every clock cycle. For those rows where A = 0, $B+ = B'$. For those rows where B = 0 and A = 0, $C+ = C$. The up-down counter can be implemented using D flip-flops and gates, as shown in the following Figure. The corresponding logic equations are also given in the following Figure.



When U = 1 and D = 0, these equations reduce to equations for a binary up counter.

$D_A = A^+ = A'$

$D_B = B^+ = BA' + B'A = B \oplus A$

$D_C = C^+ = C'BA + CB' + CA' = C'BA + C\left(BA\right)' = C \oplus BA$

When U = 0 and D = 1, these equations reduce to –

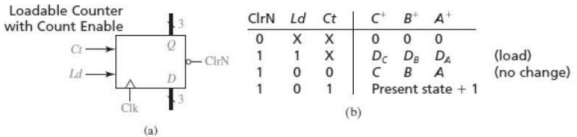
$D_A = A^+ = A \oplus 1 = A'$ $\quad (A\ Changes\ state\ every\ clock\ cycle)$

$D_B = B^+ = B \oplus A'$ $\quad (B\ Changes\ state\ when\ A = 0)$

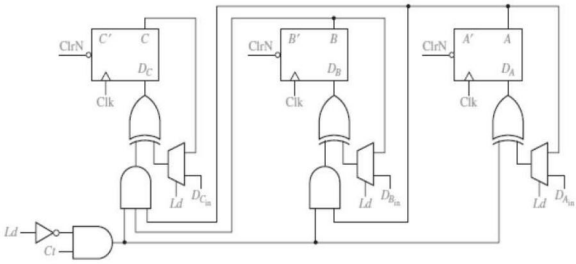
$D_C = C^+ = C \oplus B'A'$ $\quad (C\ Changes\ state\ when\ B = A' = 0)$

Loadable Counter:

The counter shown in the following Figure (a) has two control signals Ld (load) and Ct (count).



When $Ld = 1$ binary data is loaded into the counter on the rising clock edge, and when $Ct = 1$, the counter is incremented on the rising clock edge. When $Ld = Ct = 0$, the counter holds its present state. When $Ld = Ct = 1$, load overrides count, and data is loaded into the counter. The counter also has an asynchronous clear input that clears the counter when ClrN is 0. Figure (b) summarizes the counter operation. All state changes occur on the rising edge of the clock (except for the asynchronous clear). The following Figure shows how the loadable counter can be implemented using flip-flops, MUXes, and gates.



When Ld = 1, each MUX selects a Di input, and because the output of each AND gate is 0, the output of each XOR gate is Di, which gets stored in a flip-flop. When Ld = 0 and Ct = 1, each MUX selects one of the flip-flop outputs (C, B, or A). The circuit then becomes equivalent to a binary counter, and the counter is incremented on the rising clock edge. The next-state equations for the counter of above Figure are –

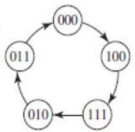
$$A^+ = D_A = \left(Ld' \cdot A + Ld \cdot D_{Ain}\right) \oplus Ld' \cdot Ct$$
$$B^+ = D_B = \left(Ld' \cdot B + Ld \cdot D_{Bin}\right) \oplus Ld' \cdot Ct \cdot A$$
$$C^+ = D_C = \left(Ld' \cdot C + Ld \cdot D_{Cin}\right) \oplus Ld' \cdot Ct \cdot B.A$$

When $Ld = 0$ and $Ct = 1$, these equations reduce to $A^+ = A'$, $B^+ = B \oplus A$, and $C^+ = C \oplus BA$, which are the equations previously derived for a 3-bit counter.

5.4 COUNTER FOR OTHER SEQUENCES:

In some applications, the sequence of states of a counter is not in straight binary order. The following Figure shows the state graph for such a counter. The arrows indicate the state sequence. If this counter is started in state 000, the first clock pulse will take it to state 100, the next pulse to 111, etc. The clock pulse is implicitly understood to be the input to the circuit and not shown on the graph. The corresponding state table for the counter is given the following Table. Note that the next state is unspecified for the present states 001, 101, and 110.

State Graph for Counter



State Table

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

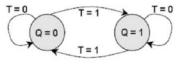
We will design the counter specified by the above Table using T flip-flops. We could derive TC, TB, and TA directly from this table, as in the preceding example. However, it is often more convenient to plot next-state maps showing C+, B+, and A+ as functions of C, B, and A, and then derive TC, TB, and TA from these maps.

Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0			
0	0	1	-	-	-			
0	1	0	0	1	1			
0	1	1	0	0	0			
1	0	0	1	1	1			
1	0	1	-	-	-			
1	1	0	-	-	-			
1	1	1	0	1	0			

From the first row of the table, the CBA = 000; and hence, the C+, B+, and A+ columns are filled in with 1, 0, and 0, respectively. From the second row, the CBA = 001; all three columns are filled in with don't-cares. From the third row, the CBA = 010; and the C+, B+, and A+ columns are filled with 0, 1, and 1, respectively. The next-state columns can be quickly completed by continuing in this manner. Next, we will derive the maps for the T inputs from the next-state maps.

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table



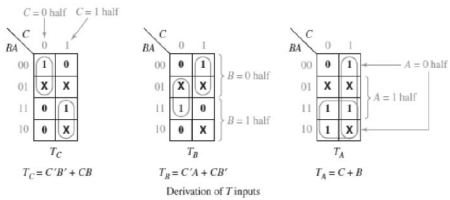
State Diagram

$Q \rightarrow Q_{n+1}$	T
0	0
0	1
1	0
1	1

Excitation Table

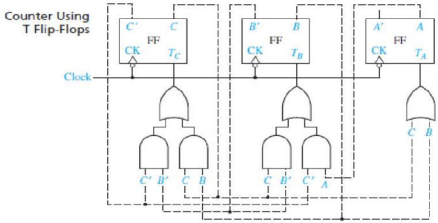
Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	x	x	x
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	-	-	-	x	x	x
1	1	0	-	-	-	x	x	x
1	1	1	0	1	0	1	0	1

Now, draw the K-maps for TC, TB, and TA separately, and derive the expressions

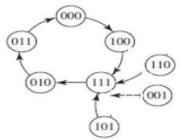


Derivation of *T* inputs

Finally, draw the counter circuit based on the expressions.



NOTE: In the above problem, for the design, initially, when the power is switched on, if circuit enters to an invalid state (state 1 or state 5 or state 6), then lock-in condition results. The designed counter will not give proper result. For this reason, all of the don't-care states in a counter should be checked to make sure that they eventually lead into the main counting sequence unless a power-up reset is provided. The solution is self-correcting counter.



Counter Design Using D Flip-Flops:

For a D flip-flop, Q+ = D, so the D input map is identical with the next-state map.

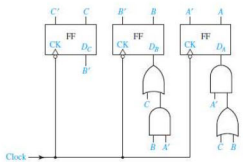
Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	D_C	D_B	D_A
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	x	x	x
0	1	0	0	1	1	0	1	1
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	-	-	-	x	x	x
1	1	0	-	-	-	x	x	x
1	1	1	0	1	0	0	1	0

Draw the K-map; and from the map, we get the following expressions:

$$D_C = C^+ = B' \qquad D_B = B^+ = C + BA'$$

$$D_A = A^+ = CA' + BA' = A'(C + B)$$

Corresponding counter circuit is –



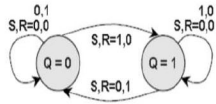
5.5 COUNTER DESIGN USING S-R AND J-K FLIP-FLOPS:

The procedures used to design a counter with S-R OR flip-flops are similar to the procedures discussed. However, instead of deriving an input equation for each D or T flip-flop, the S and R input equations must be derived. The following Table describes the behaviour of S-R flip-flops:

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

inputs not allowed

Characteristic Table



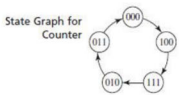
State Diagram

Q	Q^+	S	R
0	0	0	0
		0	1
0	1	1	0
1	0	0	1
1	1	0	0
		1	0

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

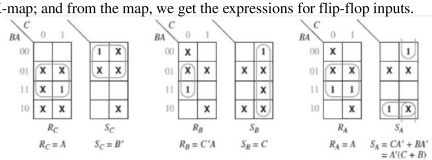
Excitation Table

Next, we will redesign the counter for following Figure using S-R flip-flops.

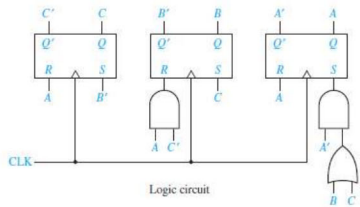


C	B	A	C^+	B^+	A^+	S_C	R_C	S_B	R_B	S_A	R_A
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	–	–	–	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	–	–	–	X	X	X	X	X	X
1	1	0	–	–	–	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

Draw the K-map; and from the map, we get the expressions for flip-flop inputs.



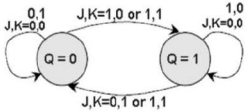
S-R flip-flop equations



The procedure used to design a counter with J-K flip-flops is very similar to that used for S-R flip-flops. The J-K flip-flop is similar to the S-R flip-flop except that J and K can be 1 simultaneously, in which case the flip-flop changes state.

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Characteristic Table

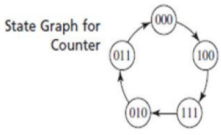


State Diagram

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

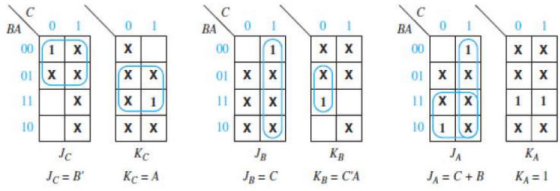
Excitation Table

Now, we will redesign the counter for following Figure using JK flip-flops.

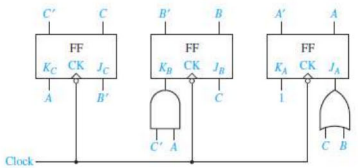


C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	–	–	–	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	–	–	–	X	X	X	X	X	X
1	1	0	–	–	–	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

Draw the K-map; and from the map, we get the expressions for flip-flop inputs; and finally draw the counter circuit using J-K flip-flops.

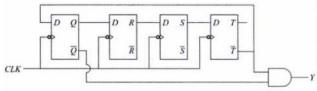


J-K flip-flop input equations



Logic circuit (omitting the feedback lines)

1] Construct a 4-bit Johnson counter using (a) D flip-flops (b) J-K flip-flops.



Clock	Serial in = $\overline{T}$	Q	R	S	T	Y = $\overline{Q}\overline{T}$
0	1	0	0	0	0	1
1	1	1	0	0	0	0
2	1	1	1	0	0	0
3	1	1	1	1	0	0
4	0	1	1	1	1	0
5	0	0	1	1	1	0
6	0	0	0	1	1	0
7	0	0	0	0	1	0
8	1	0	0	0	0	1
9	1	1	0	0	0	0 repeats