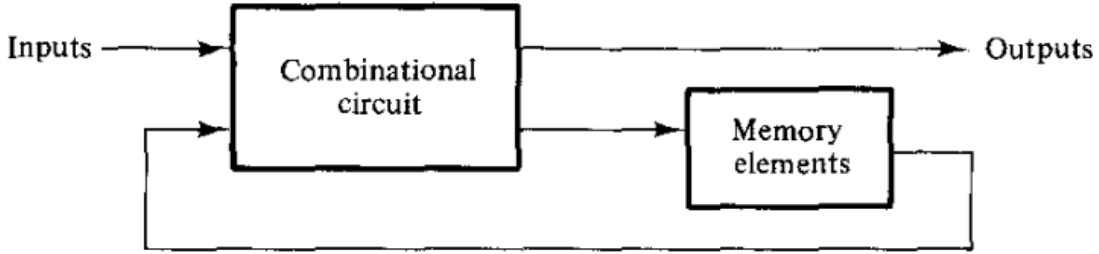
Module 5 - Design of Sequential Logic Circuits

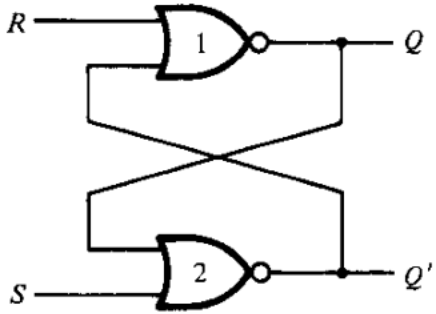
Dr. E.Papanasam
papanasam.e@vit.ac.in
AP Senior
School of Electronics
VIT Chennai

Sequential Circuit



SR Latch

- Asynchronous sequential circuits can be implemented by employing a basic flip-flop commonly referred to as an SR latch

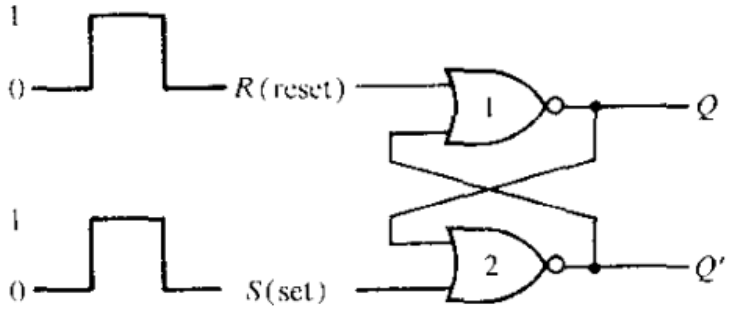


S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(After SR = 10)
0	1	0	1	
0	0	0	1	(After SR = 01)
1	1	0	0	

Flip-Flop

- A flip-flop circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit) until directed by an input signal to switch states
- The major differences among various types of flip-flops are in the number of inputs they possess and in the manner in which the inputs affect the binary state

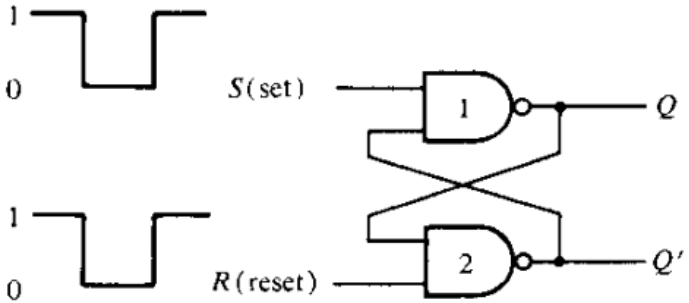
Basic Flip-Flop Circuit



(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after S = 1, R = 0)
0	1	0	1	
0	0	0	1	(after S = 0, R = 1)
1	1	0	0	

(b) Truth table

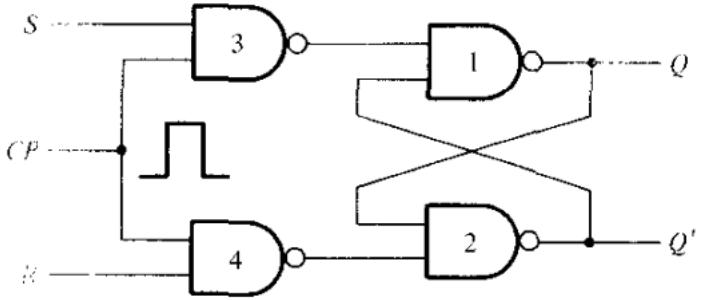


(a) Logic diagram

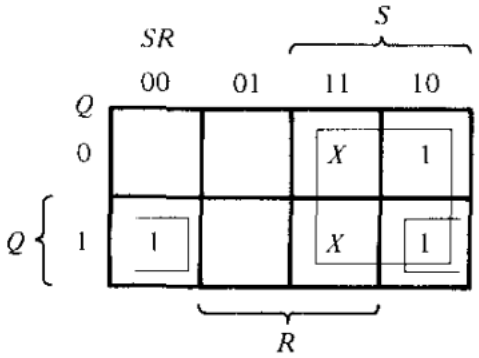
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after S = 1, R = 0)
0	1	1	0	
1	1	1	0	(after S = 0, R = 1)
0	0	1	1	

(b) Truth table

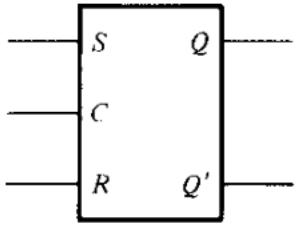
RS Flip-flop



Q	S	R	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

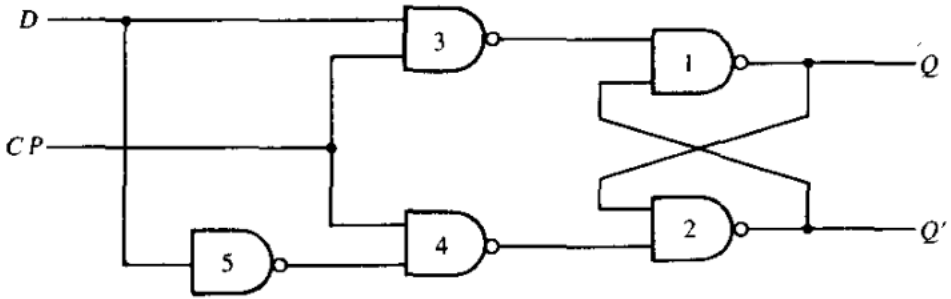


$$Q(t+1) = S + R'Q$$



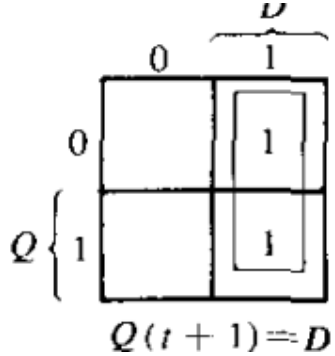
D Flip-Flop

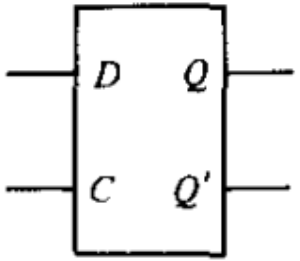
- Eliminate the undesirable condition of the indeterminate state in the RS flip-flop
- Ensure inputs S and R are never equal to I at the same time



D Flip-Flop

Q	D	$Q(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1

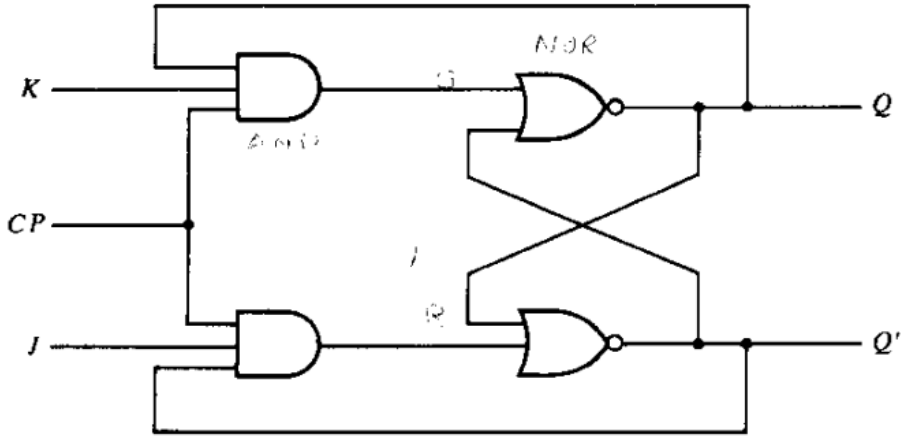




JK Flip-Flops

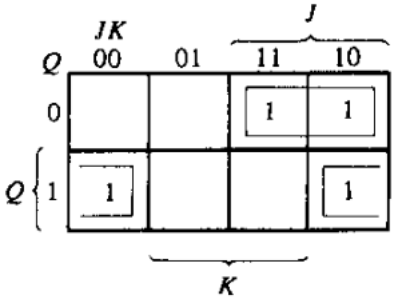
- Refinement of the RS flip-flop
- Indeterminate state of the RS type is defined in the JK type
- Inputs J and K behave like inputs S and R to set and clear the flip-flop, respectively
- The input marked J is for set and the input marked K is for reset
- When both inputs J and K are equal to 1, the flip-flop switches to its complement state

JK Flip-Flop



Q	J	K	Q (t + 1)
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

(b) Characteristic table

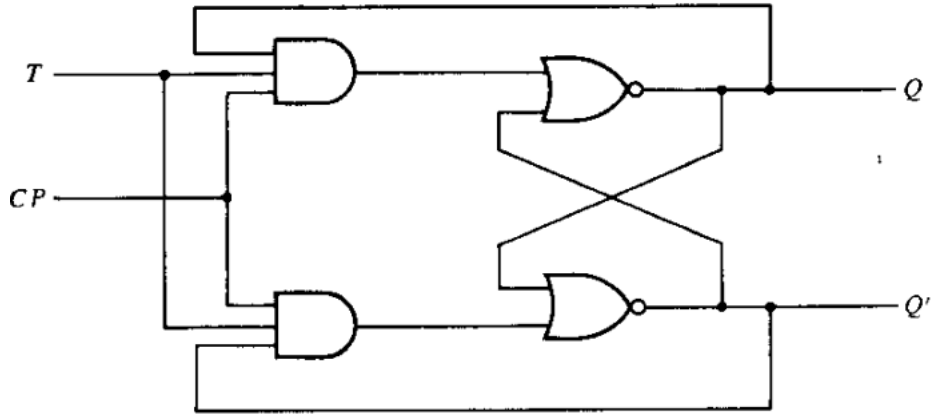


$Q(t+1) = JQ' + K'Q$

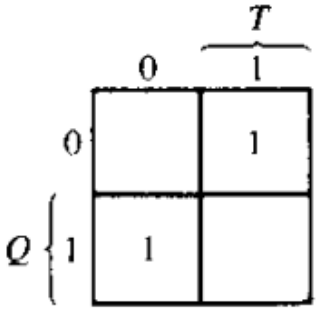
T Flip-Flops

- Single-input version of the JK flip-flop
- T flip-flop is obtained from the JK flip-flop when both inputs are tied together
- The designation T comes from the ability of the flip-flop to "toggle," or complement, its state.
- Regardless of the present state, the flip-flop complements its output when the clock pulse occurs while input T is I

T Flip-Flops



Q	T	Q(t + 1)
0	0	0
0	1	1
1	0	1
1	1	0

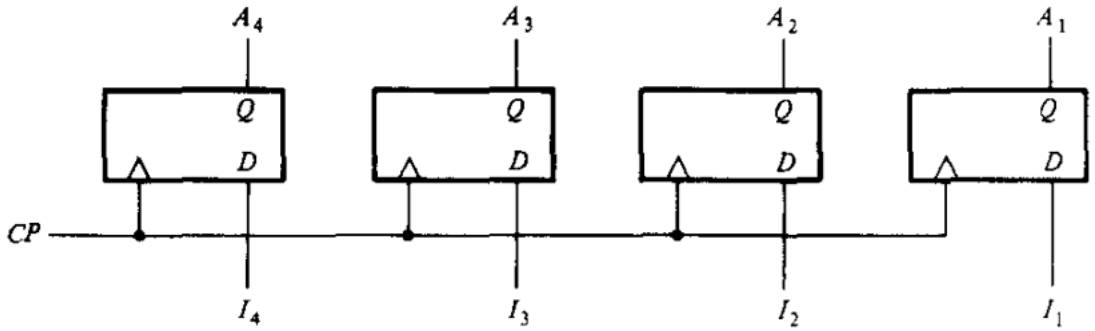


$$Q(t+1) = TQ' + T'Q$$

Registers

- A register is a group of binary cells suitable for holding binary information.
- A group of flip-flops constitutes a register, since each flip-flop is a binary cell capable of storing one bit of information
- An n-bit register has a group of n flip-flops and is capable of storing any binary information containing n bits
- In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks
- Register consists of a group of flip-flops and gates that affect their transition
- The flip-flops hold binary information and the gates control when and how new information is transferred into the register.

4-Bit Register



Shift registers

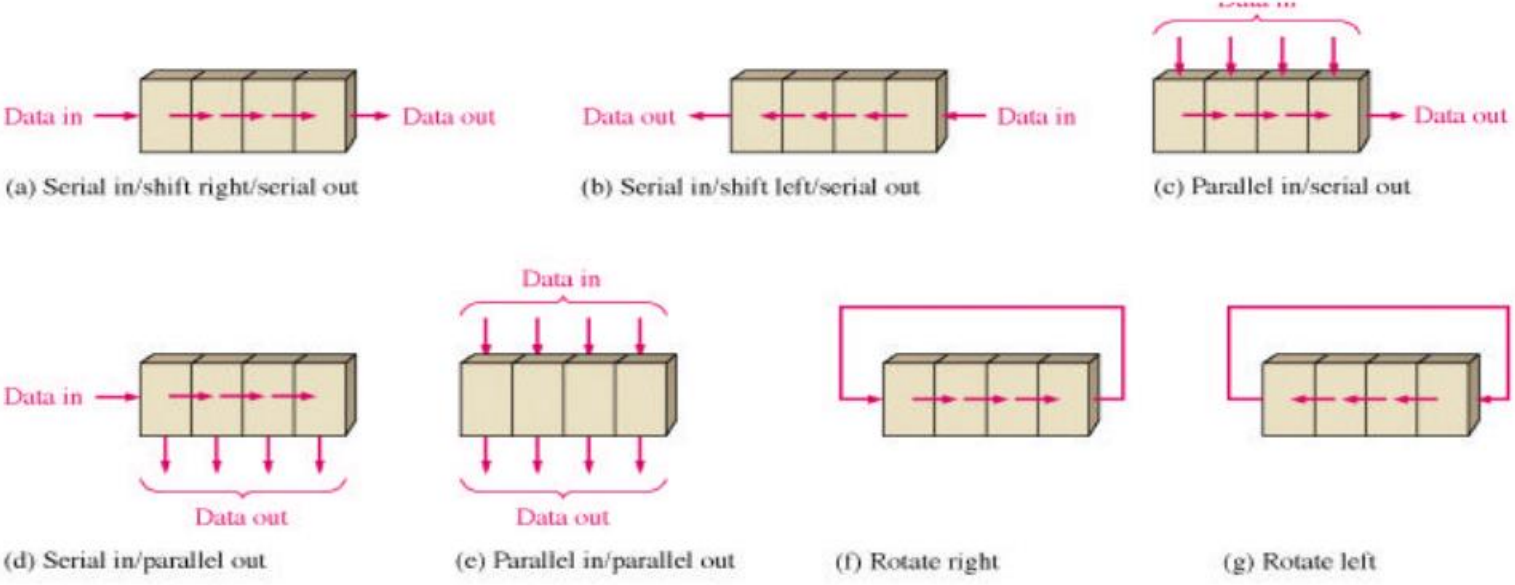
Register:

A set of n flip-flops
Each flip-flop stores one bit
Two basic functions: data storage and data movement

Shift Register:

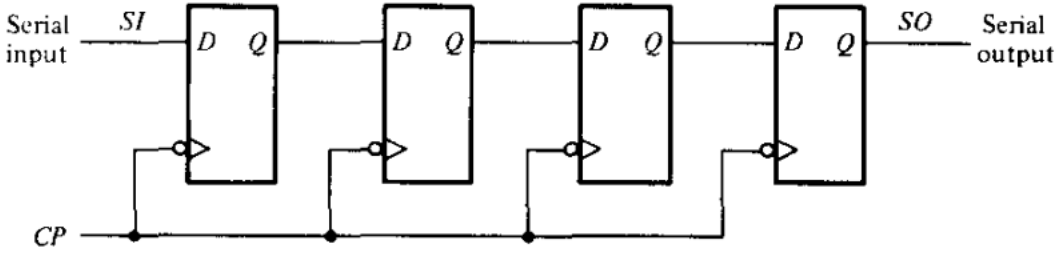
A register that allows each of the flip-flops to pass the stored information to its adjacent neighbour

Basic data movement in shift registers



Shift Registers

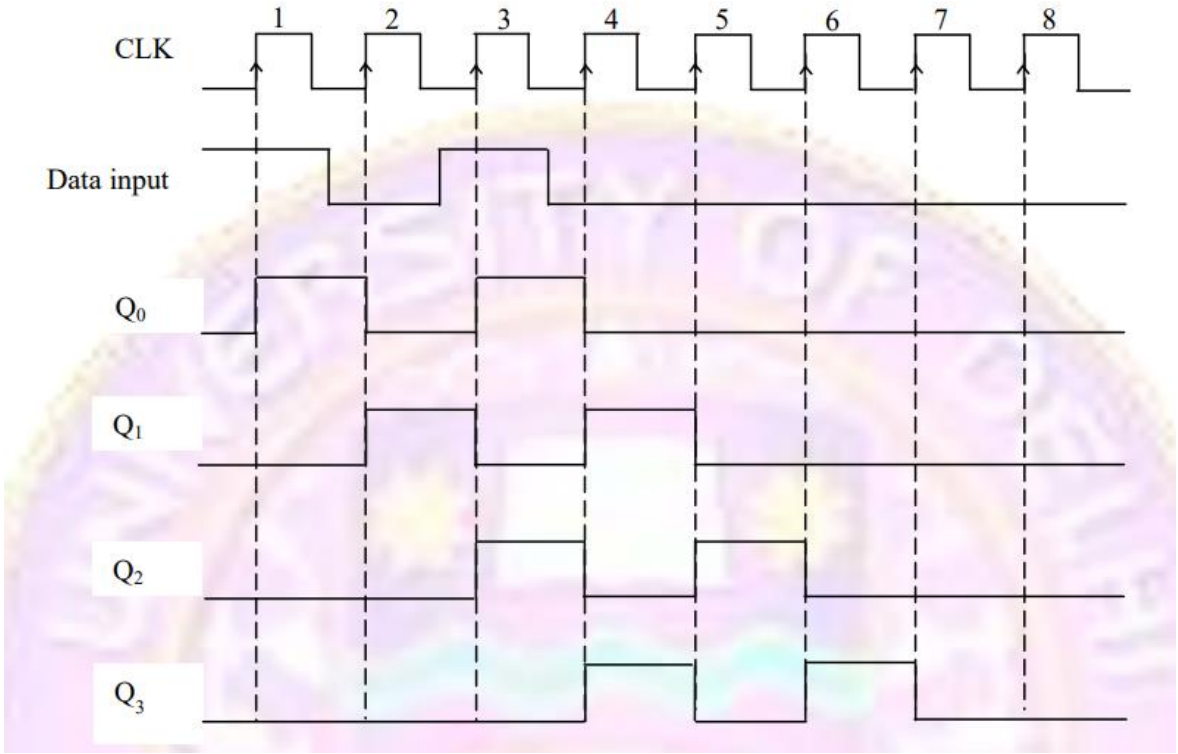
- A register capable of shifting its binary information either to the right or to the left is called a shift register
- The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop
- All flip-flops receive a common clock pulse that causes the shift from one stage to the next



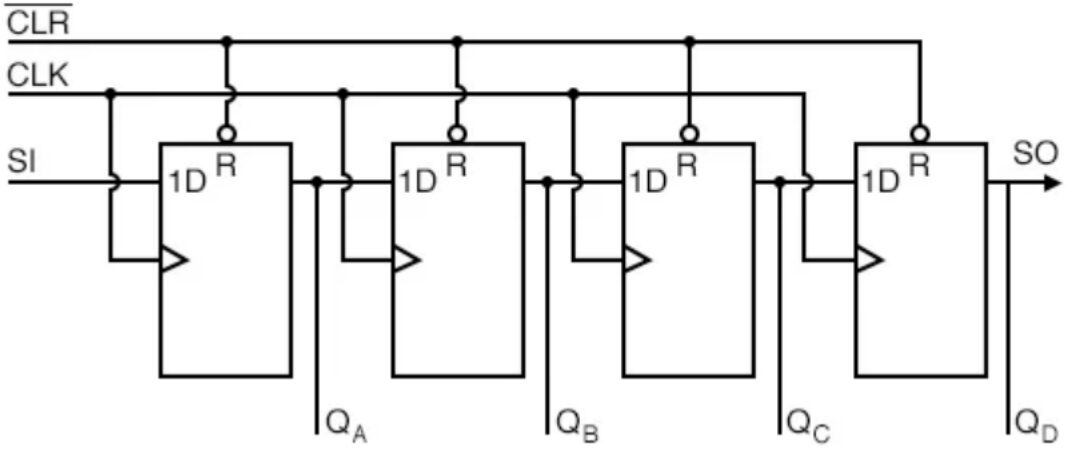
Shift Registers (SISO)

- The Q output of a given flip-flop is connected to the D input of the flip-flop at its right
- Each clock pulse shifts the contents of the register one bit position to the right
- The **serial input** determines what goes into the leftmost flip-flop during the shift
- The **serial output** is taken from the output of the rightmost flip-flop
- Applications: Ring counter

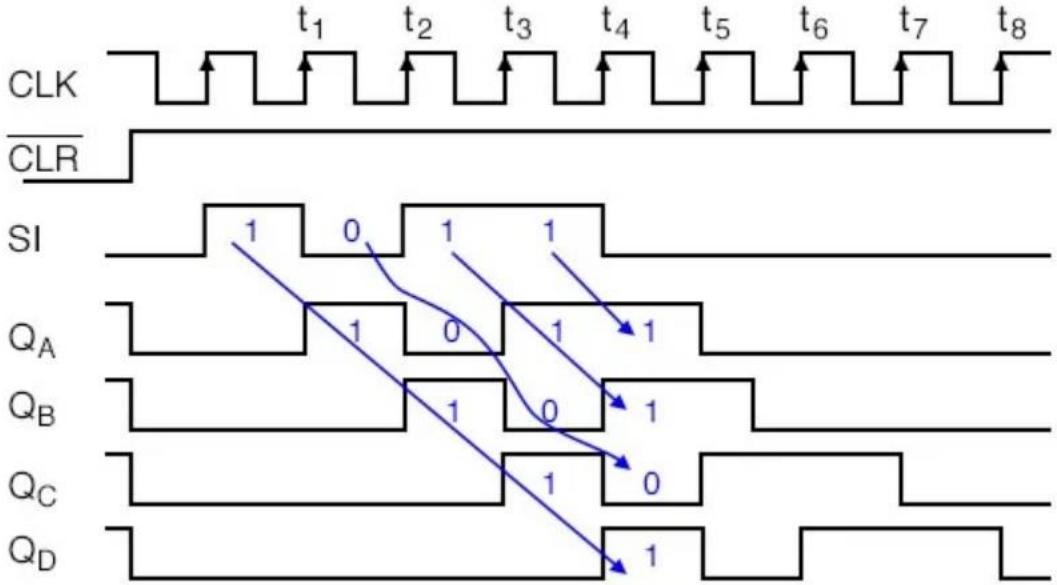
TIMING DIAGRAM OF 4-BIT SISO SHIFT REGISTER



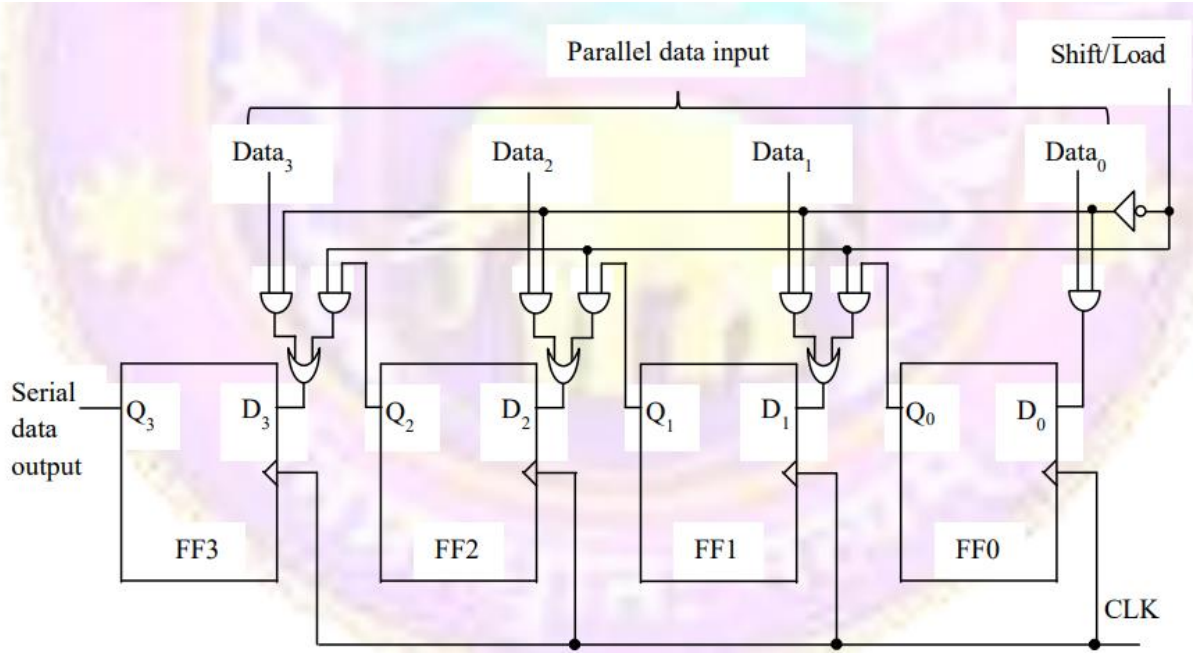
Serial In Parallel Out (SIPO)



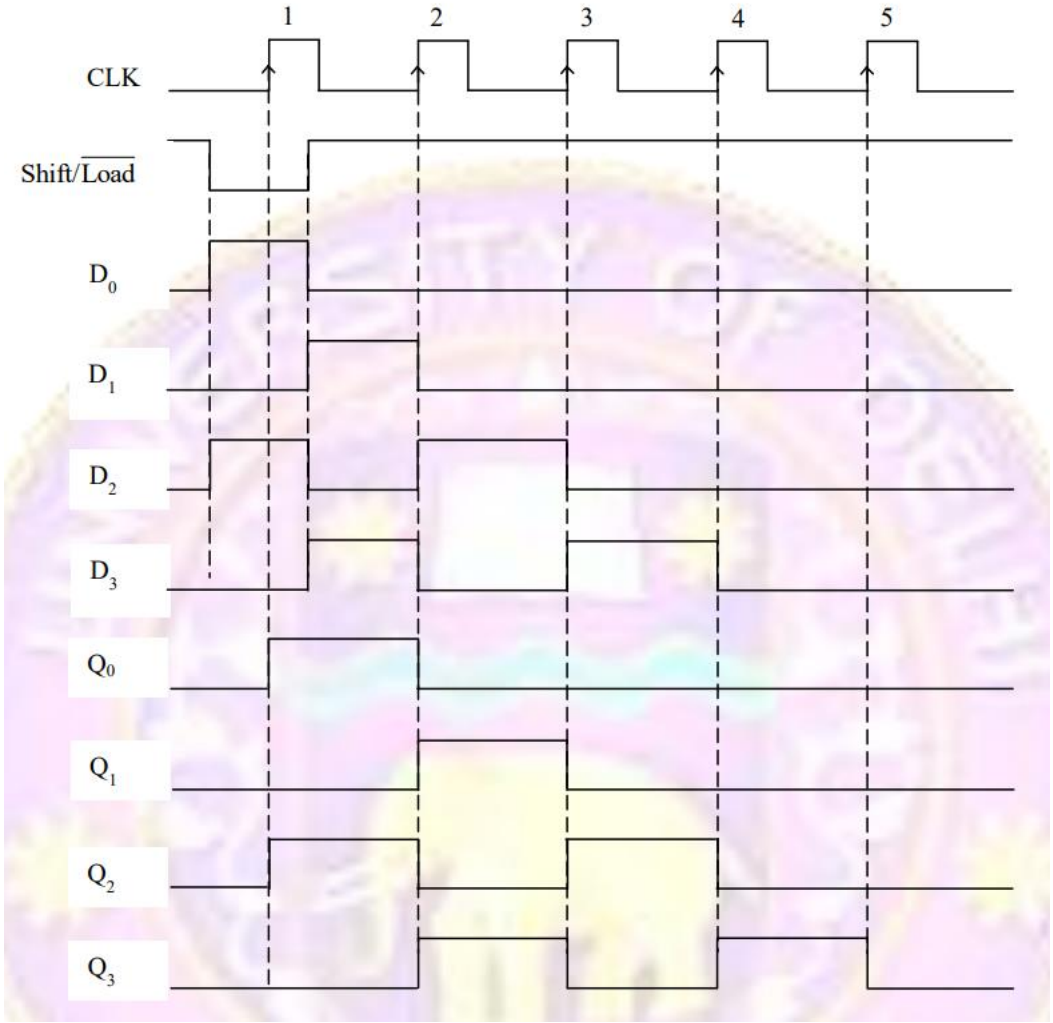
Serial In Parallel Out (SIPO)



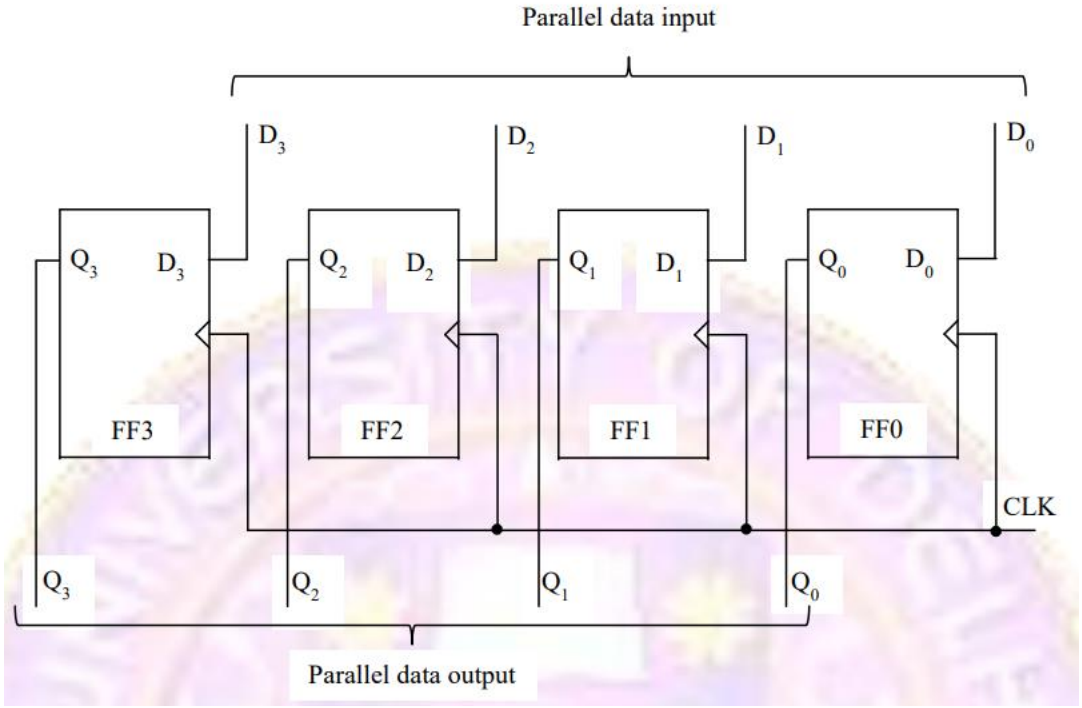
4-BIT PISO SHIFT REGISTER



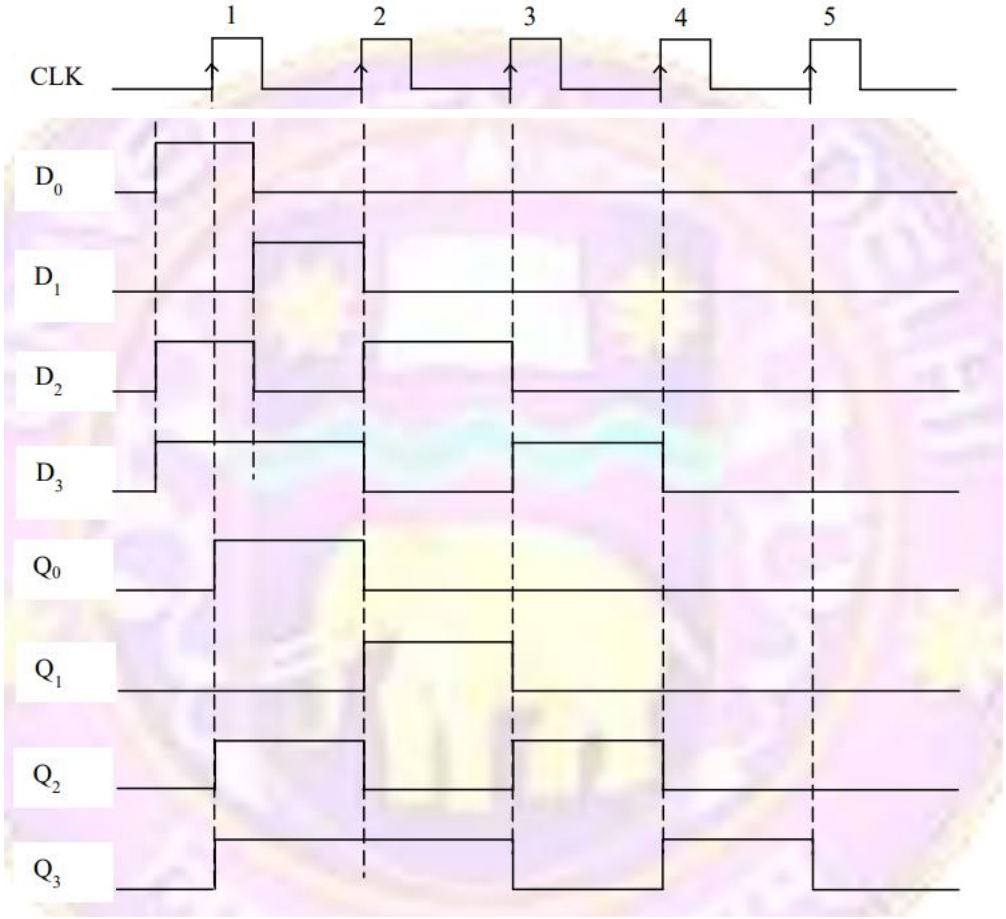
PISO



Parallel In Parallel Out (PIPO)



PIPO



Counters

- A sequential circuit that goes through a prescribed sequence of states upon the application of input pulses
- The input pulses, called count pulses, may be clock pulses or they may originate from an external source and may occur at prescribed intervals of time or at random
- 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.
- Used for counting the number of occurrences of an event
- Useful for generating timing sequences to control operations in a digital system

Binary Counter

- A counter that follows the binary sequence is called a binary counter
- An n-bit binary counter consists of n flip-flops and can count in binary from 0 to 2^{n-1}

Design of Synchronous Sequential Circuit

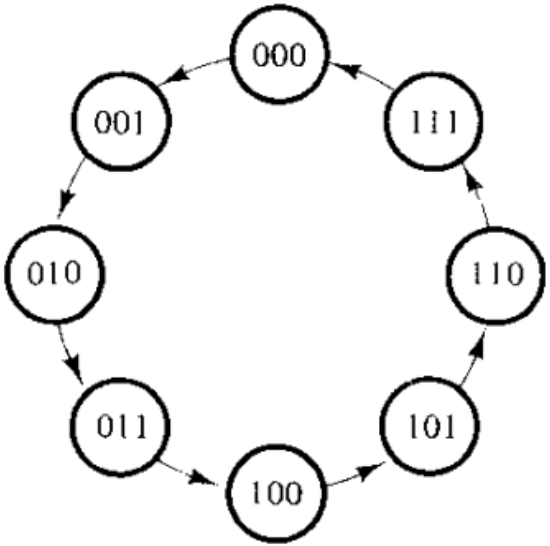
- The design of a clocked sequential circuit starts from
  - a set of specifications and culminates in a logic diagram
  - or
  - a list of Boolean functions from which the logic diagram can be obtained
- Sequential circuit requires a state table for its specification
- The first step in the design of sequential circuits is to obtain a state table or an equivalent representation state diagram

Design of Synchronous Sequential Circuit

- A synchronous sequential circuit is made up of flip-flops and combinational gates
- The design of the circuit consists of choosing the flip-flops and then finding a combinational gate structure that, together with the flip-flops, produces a circuit that fulfills the stated specifications
- The number of flip-flops is determined from the number of states needed in the circuit

Design of 3-bit Binary Up Counter

- State diagram and Excitation Table

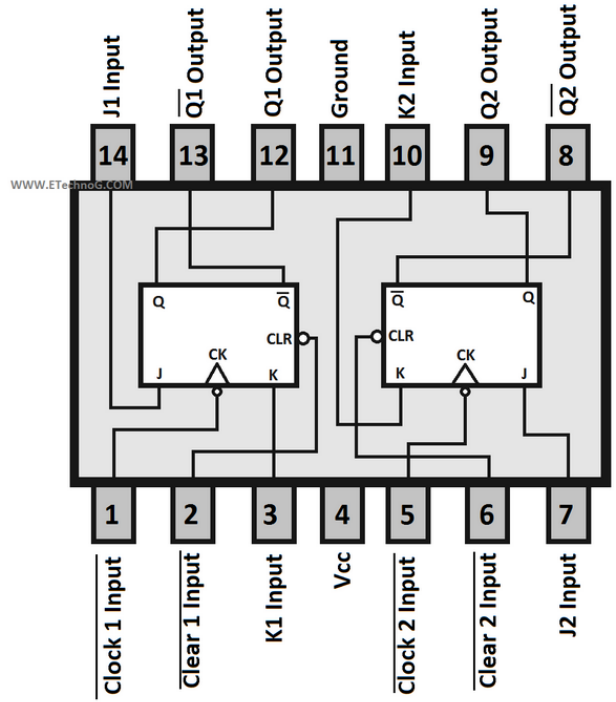


Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

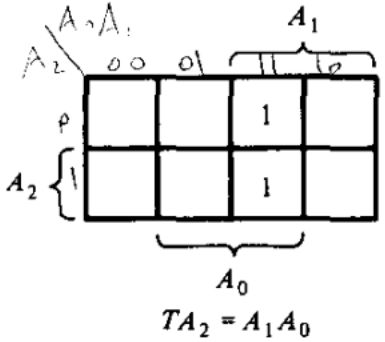
State Table

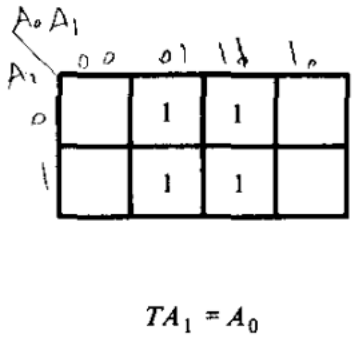
Design of 3-bit Binary Counter

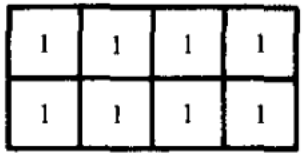
- K-map

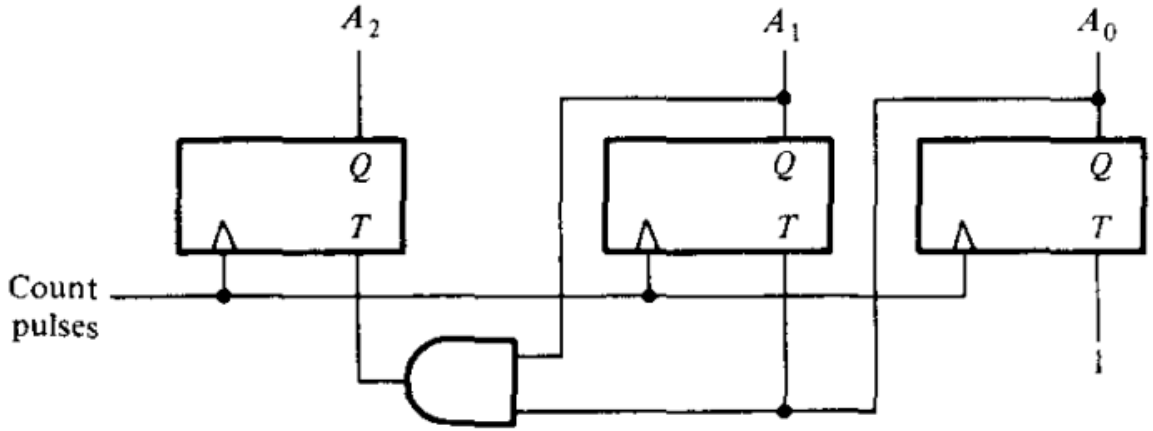


IC 7473 Pinout Diagram

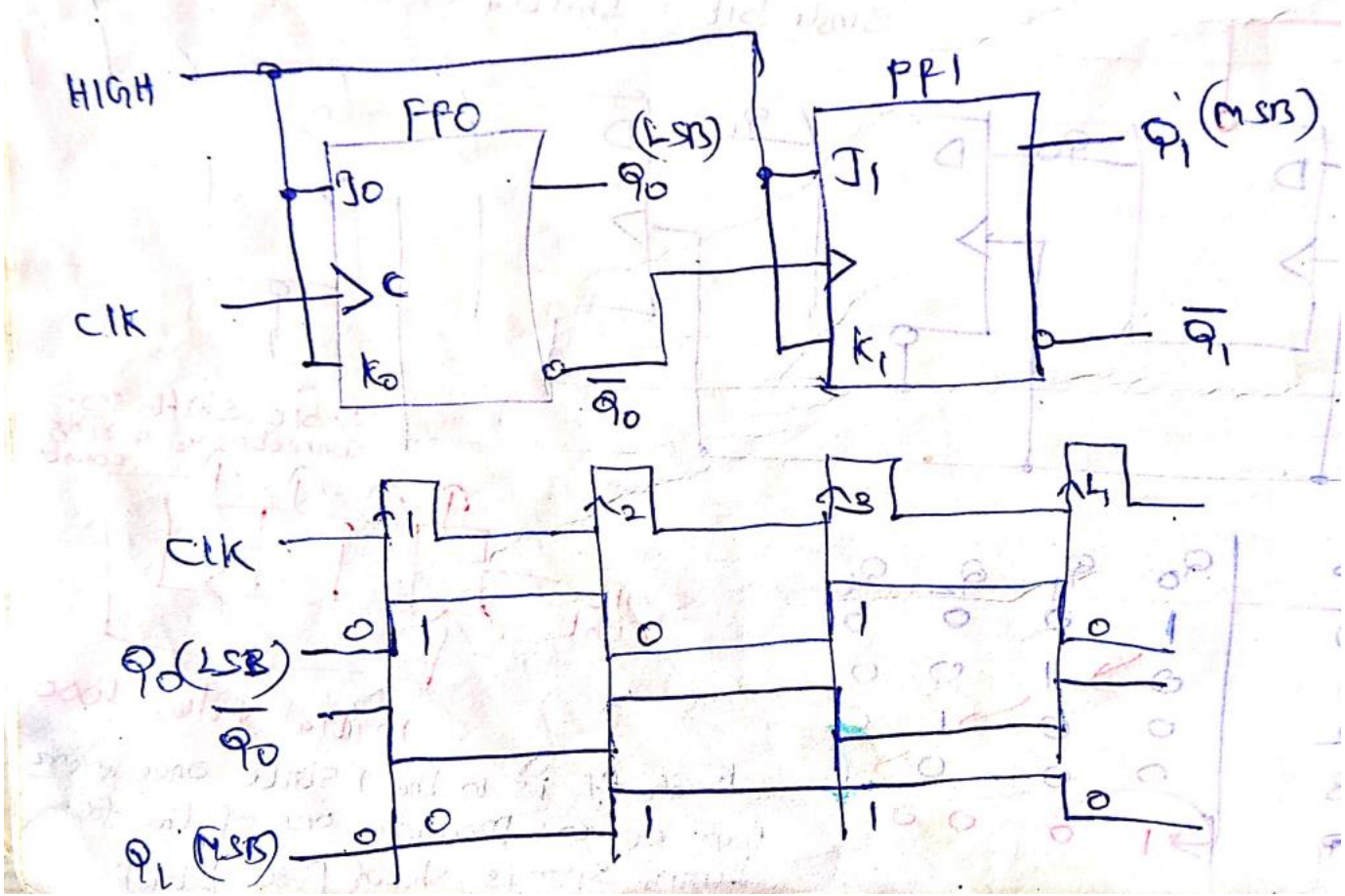


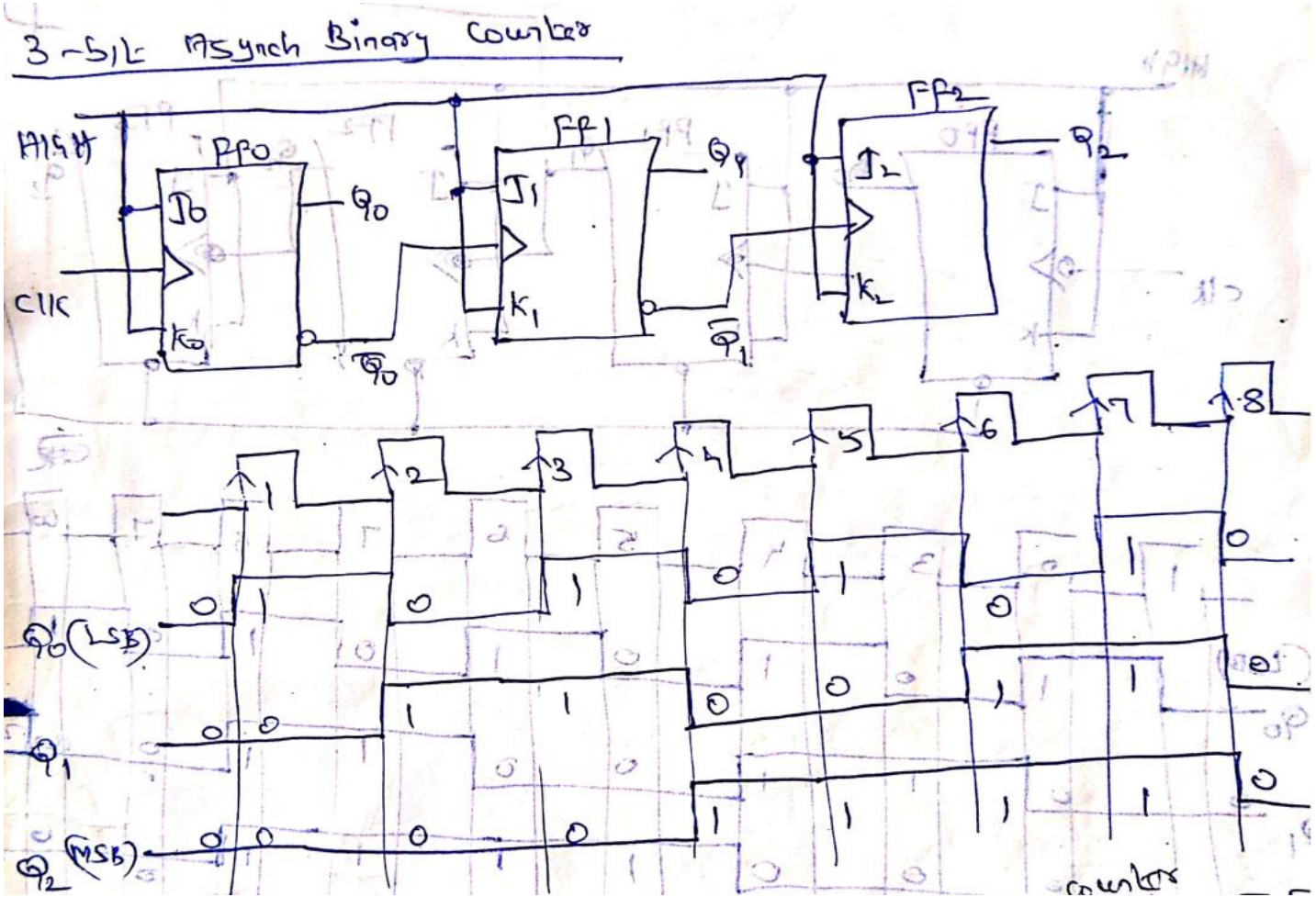






2-bit Asynch Binary counter

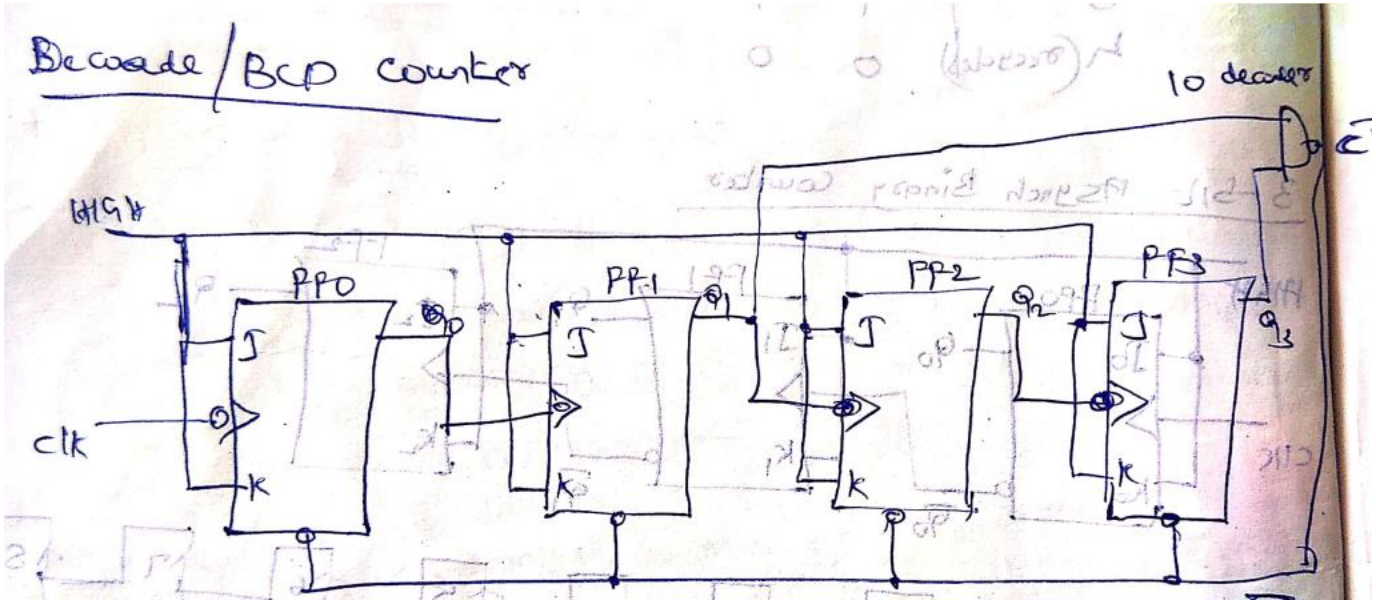


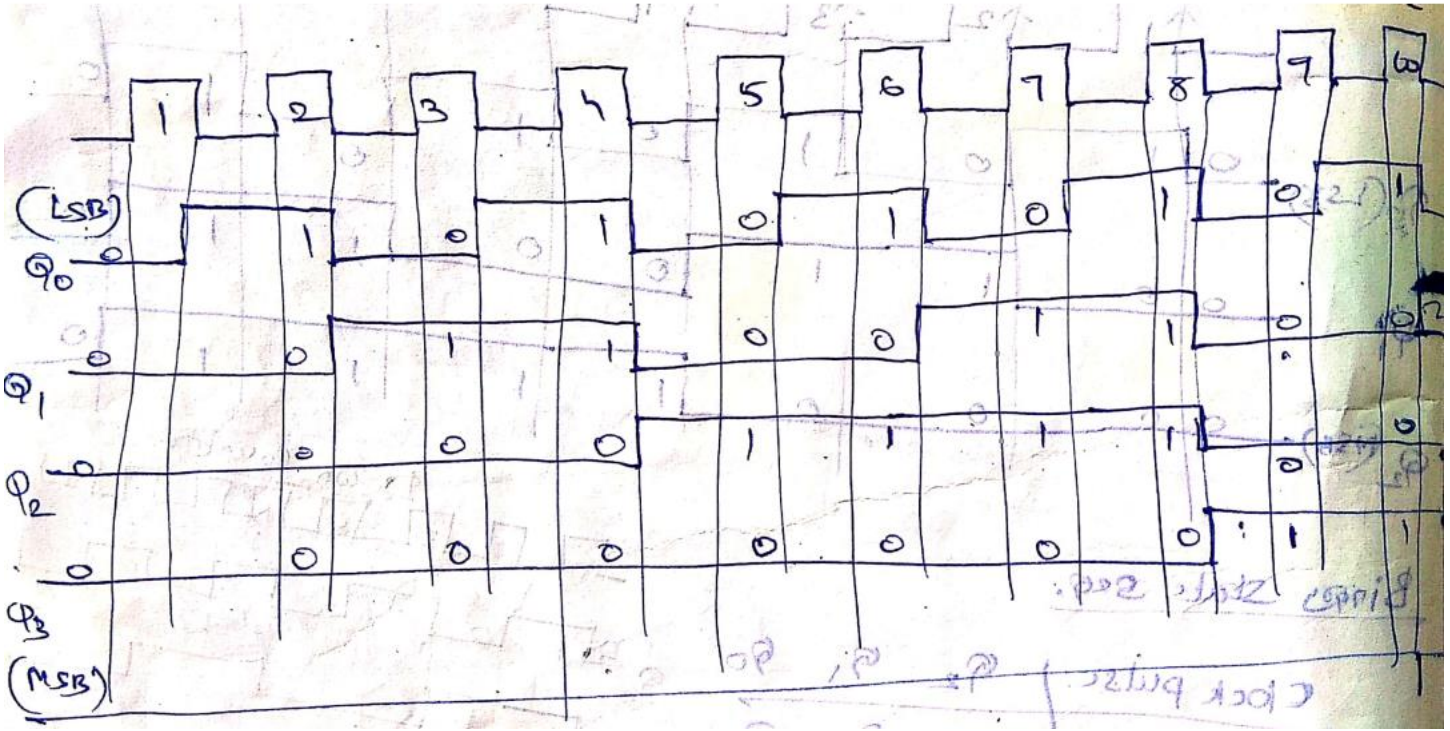

3-bit Asynch Binary Counter
HIGH
FF0
FF1
FF2
J0
K0
Q0
J1
K1
Q1
J2
K2
Q2
CLK
1
2
3
4
5
6
7
8
Q0 (LSB)
Q1
Q2 (MSB)
counter

Binary state seq.

Clock pulse	Q_2	Q_1	Q_0
Initial state	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (Recycled)	0	0	0

Decade Conter

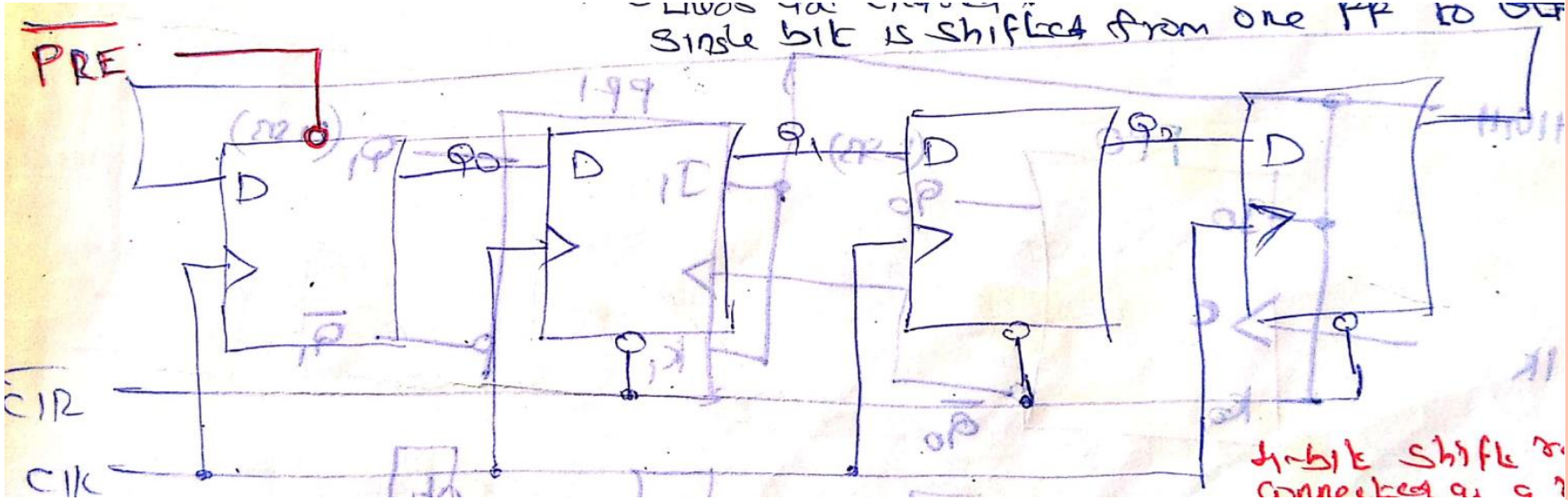



1
2
3
4
5
6
7
8
9
10
(LSB)
Q_0
Q_1
Q_2
Q_3
(MSB)

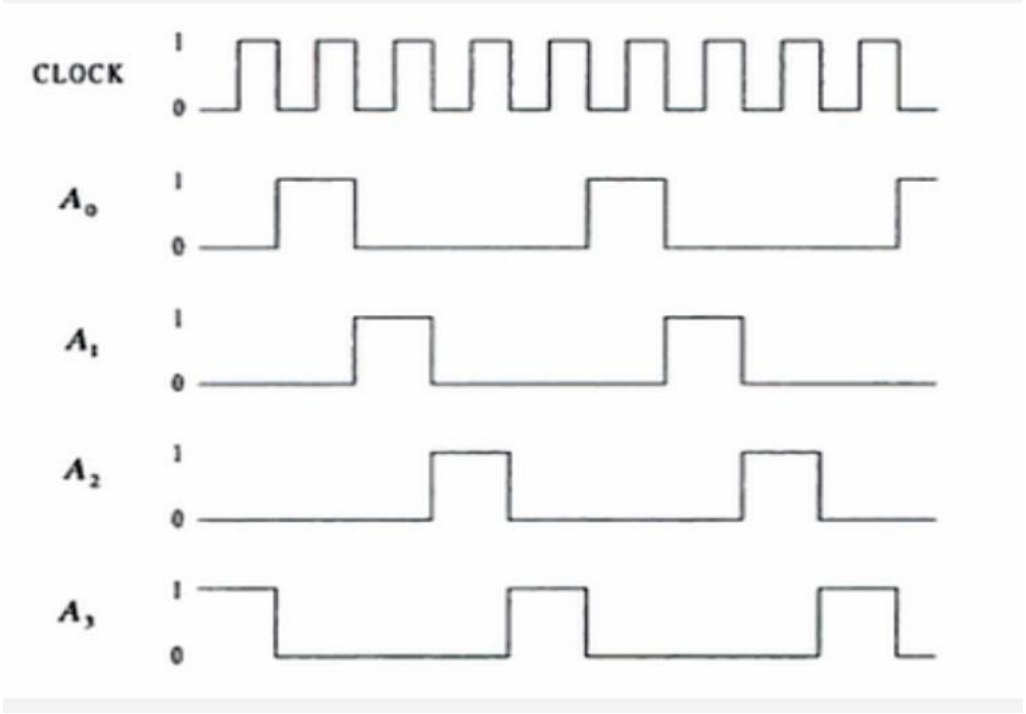
Ring Counter

- Developed by modifying a shift register
- Used in digital system to generate control pulses.
- A typical 4-bit ring counter is made of D-flip flops or JK-flip flop connected in cascade with the non-complemented output of the last stage connected as an input to the first stage.
- There are **no external inputs except the clock signal**
- Ring counter has **Mod = n 'n' is the number of bits**
- It means 4-bit ring counter has 4 states
- Divides the frequency of the clock signal by 'n'.
- n is the bit size of the ring counter.
- Can be used as a frequency divider

RING COUNTER

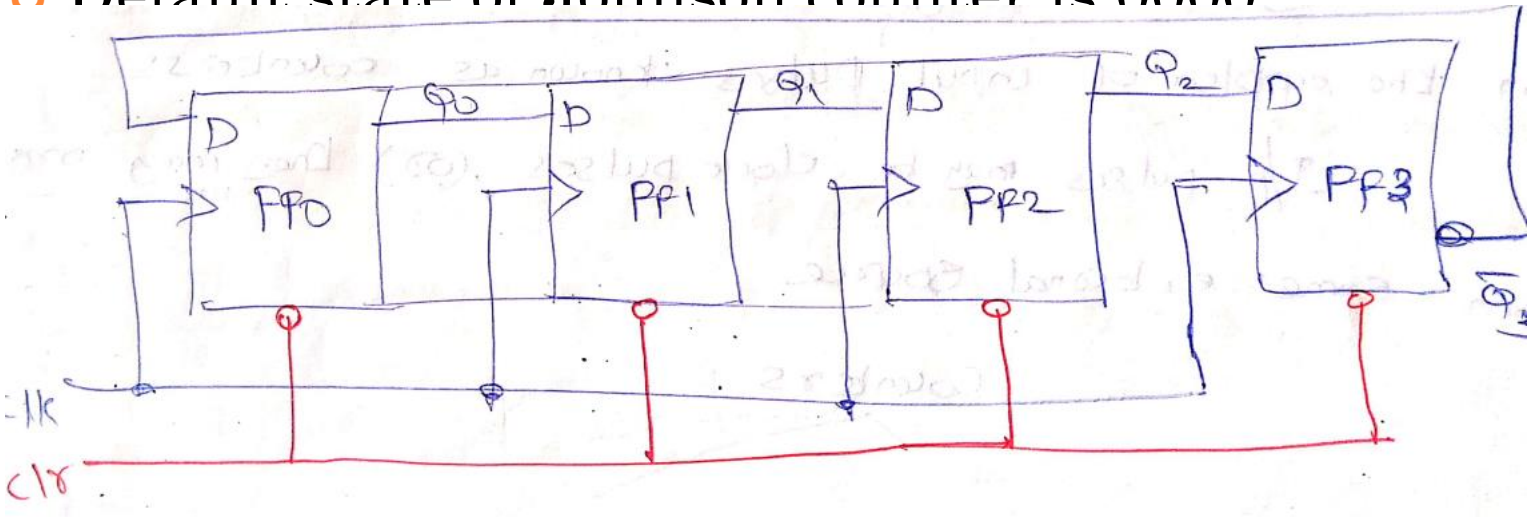


States	Q_A	Q_B	Q_C	Q_D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1


CLOCK
1
0
A_0
1
0
A_1
1
0
A_2
1
0
A_3
1
0

JOHNSON COUNTER

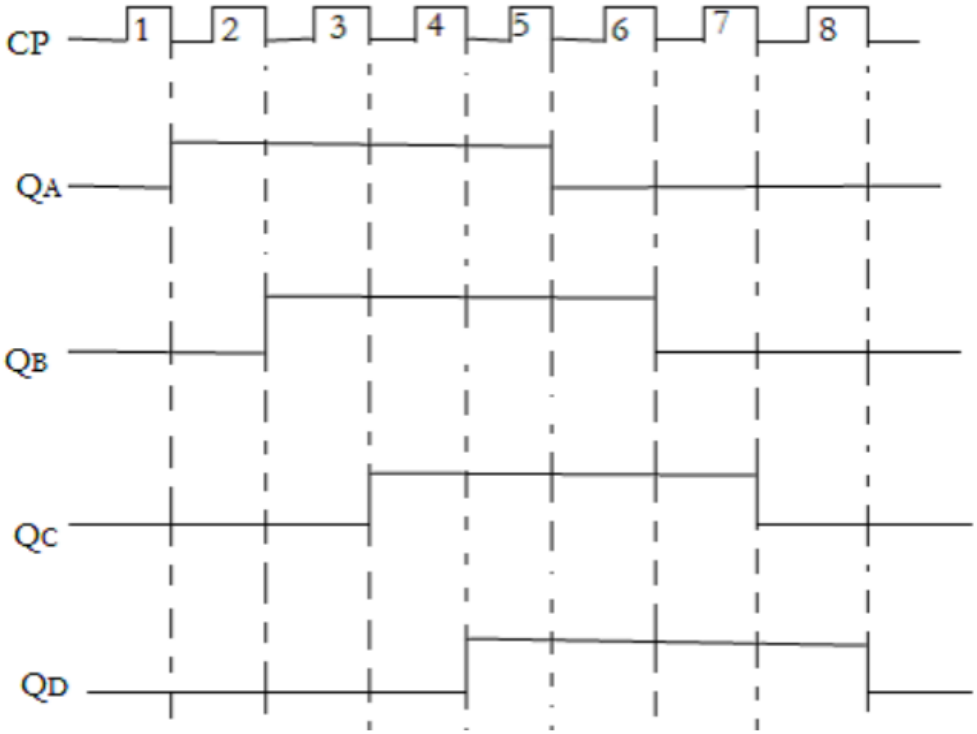
- Default state of Johnson counter is 0000



States	Q_A	Q_B	Q_C	Q_D
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1
8	0	0	0	1

JOHNSON COUNTER

- More outputs as compared to ring counter.
- It has same number of flip flop but it can count twice the number of states the ring counter can count.
- It count the data in a continuous loop
- It only needs half the number of flip-flops compared to the standard ring counter for the same MOD
- Divides a clock signal's frequency by '2n'.
- n is the bit size of the counter.
- Johnson counter uses less number of flip-flops compare to a typical ring counter.


CP
1
2
3
4
5
6
7
8
QA
QB
QC
QD

Verilog Modeling - Design Block

- **module** up_counter(**input** clk, reset, **output**[**3**:**0**] counter);
- **reg** [**3**:**0**] counter_up; // up counter
- **always** @(**posedge** clk **or posedge** reset)
- **begin if**(reset) counter_up <= **4'd0**;
- **else** counter_up <= counter_up + **4'd1**;
- **end**
- **assign** counter = counter_up;
- **endmodule**

TEST BENCH

- **module** upcounter_testbench();
- **reg** clk, reset;
- **wire** [**3**:**0**] counter;
- up_counter dut(clk, reset, counter);
- **Initial**
- **begin**
  - clk=**0**;
  - **forever #5** clk=~clk;
- **end**
- **initial**
- **begin**
  - reset=**1**;
  - **#20**; reset=**0**;
- **end**
- **endmodule**

- **module** down_counter(**input** clk, reset, **output** [**3**:**0**] counter);
- **reg** [**3**:**0**] counter_down; // down counter
- **always** @(**posedge** clk **or posedge** reset)
- **begin**
- **if**(reset) counter_down <= **4'hf**;
- **else** counter_down <= counter_down - **4'd1**;
- **end**
- **assign** counter = counter_down;
- **endmodule**

```
module downcounter_testbench();
reg clk, reset;
wire [3:0] counter;
down_counter dut(clk, reset, counter);
initial
begin
clk=0;
forever #5 clk=~clk;
end
initial
begin
reset=1; #20; reset=0;
end
endmodule
```

- **module** up_down_counter(**input** clk, reset,up_down, **output**[**3**:**0**] counter);
- **reg** [**3**:**0**] counter_up_down; // down counter **always** @(**posedge** clk **or posedge** reset)
- **begin if**(reset) counter_up_down <= **4'h0**;
- **else if**(~up_down)
- counter_up_down <= counter_up_down + **4'd1**;
- **else** counter_up_down <= counter_up_down - **4'd1**; **end**
- **assign** counter = counter_up_down;
- **endmodule**

- **module** updowncounter_testbench;
- **reg** clk, reset,up_down;
- **wire** [**3**:**0**] counter;
- up_down_counter dut(clk, reset,up_down, counter);
- **initial**
- **begin**
  - clk=**0**; **forever #5** clk=~clk;
- **end**
- **initial**
- **begin**
  - reset=**1**; up_down=**0**;
  - **#20**; reset=**0**; **#200**; up_down=**1**;
- **end**
- **endmodule**

4 BIT JOHNSON COUNTER:

```
module johnson_counter(Clock, Reset, Count_out);
input Clock,Reset;
output [3:0] Count_out;
reg [3:0] Count_temp;

always @(posedge(Clock) or Reset)
begin
    if(Reset == 1'b1)
begin
Count_temp = 4'b0000;
end
    else if(Clock == 1'b1)
          begin
Count_temp = {~Count_temp[0],Count_temp[3:1]};  end
          end
   //The Count value is assigned to final output port.
   assign Count_out = Count_temp;
endmodule
```

Ripple counter

```
module dff (   input d,
               input clk,
               input rstn,
               output reg q,
               output qn);
   always @ (posedge clk or negedge rstn)
      if (!rstn)
         q <= 0;
      else
         q <= d;

   assign qn = ~q;
endmodule
```

```
module ripple ( input clk,
                input rstn,
                output [3:0] out);
   wire  q0;
   wire  qn0;
   wire  q1;
   wire  qn1;
   wire  q2;
   wire  qn2;
   wire  q3;
   wire  qn3;
```

```
dff   dff0 ( .d (qn0),
             .clk (clk),
             .rstn (rstn),
             .q (q0),
             .qn (qn0));

dff   dff1 ( .d (qn1),
             .clk (q0),
             .rstn (rstn),
             .q (q1),
             .qn (qn1));
```

```
dff   dff2 ( .d (qn2),
             .clk (q1),
             .rstn (rstn),
             .q (q2),
             .qn (qn2));
```

```
dff   dff3 ( .d (qn3),
             .clk (q2),
             .rstn (rstn),
             .q (q3),
             .qn (qn3));
```

```
   assign out = {qn3, qn2, qn1, qn0};

endmodule
```

- https://www.fpga4student.com/2017/03/verilog-code-for-counter-with-testbench.html
- https://verilogcodes.blogspot.com/2015/10/verilog-code-for-4-bit-johnson-counter.html
- https://www.chipverify.com/verilog/verilog-ripple-counter-dff