Date: 19-10-2022

Assignment-9

Aim: To understand and implement the different types of Flipflops in digital Electronics.

Table of Contents

About Flip-flops:

- The flip-flops are basically the circuits that maintain a certain state unless and until directed by the input for changing that state.
- Basically it is a circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs.

There are basically 4 types of flip-flops:

1. SR Flip Flop
2. JK Flip Flop
3. D Flip Flop
4. T Flip Flop

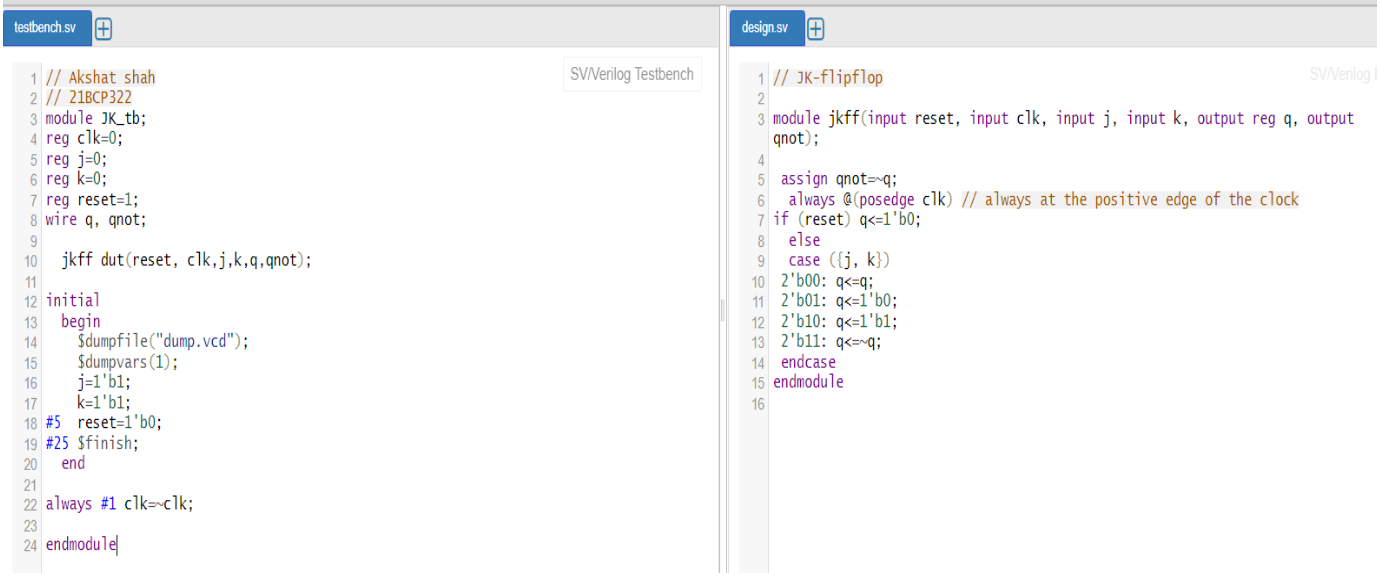
Applications of flip-flops:

- Frequency dividers
- Counters
- Storage registers
- Shift registers
- Data storage

List of Questions:

Q1. Write a Verilog code to implement J-K flip flop and validate the code via a suitable Test bench code.

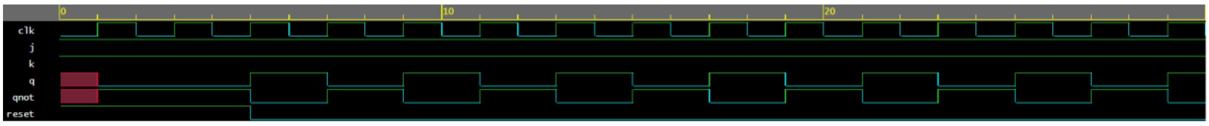
Verilog Code:



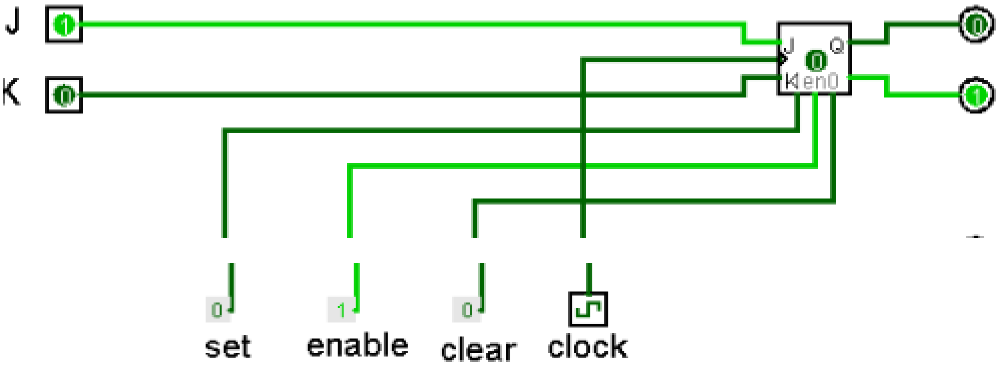
```verilog
// Akshat shah
// 21BCP322
module JK_tb;
reg clk=0;
reg j=0;
reg k=0;
reg reset=1;
wire q, qnot;

  jkff dut(reset, clk,j,k,q,qnot);

initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
    j=1'b1;
    k=1'b1;
#5  reset=1'b0;
#25 $finish;
  end

always #1 clk=~clk;

endmodule
```

```verilog
// JK-flipflop

module jkff(input reset, input clk, input j, input k, output reg q, output
qnot);

 assign qnot=~q;
  always @(posedge clk) // always at the positive edge of the clock
if (reset) q<=1'b0;
  else
  case ({j, k})
 2'b00: q<=q;
 2'b01: q<=1'b0;
 2'b10: q<=1'b1;
 2'b11: q<=~q;
 endcase
endmodule
```

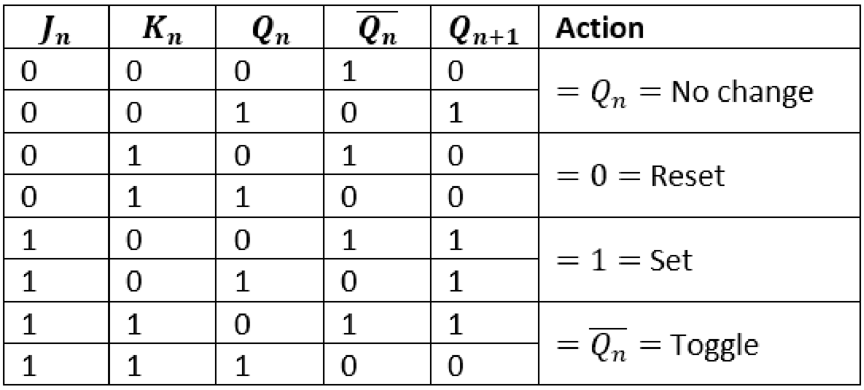
Waveform:



Logisim Circuit:

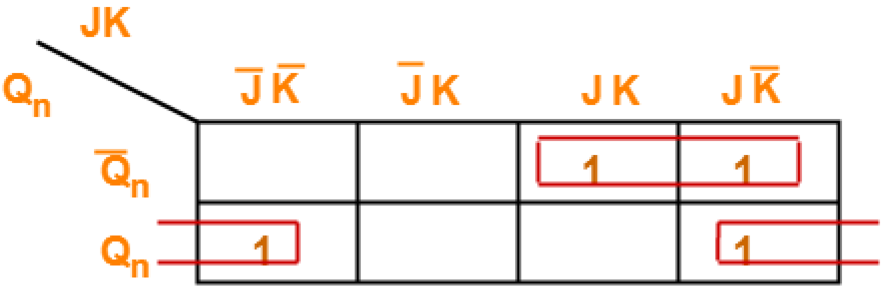


Truth table:



J_n	K_n	Q_n	$\overline{Q_n}$	Q_{n+1}	Action
0	0	0	1	0	$= Q_n =$ No change
0	0	1	0	1	
0	1	0	1	0	$= 0 =$ Reset
0	1	1	0	0	
1	0	0	1	1	$= 1 =$ Set
1	0	1	0	1	
1	1	0	1	1	$= \overline{Q_n} =$ Toggle
1	1	1	0	0	

From the truth table, the characteristic equation is represented as

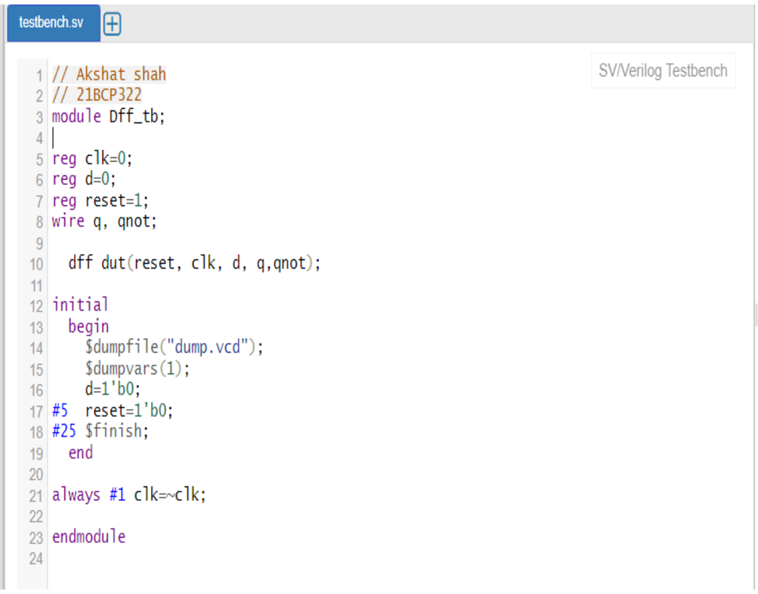


From the k-Map,

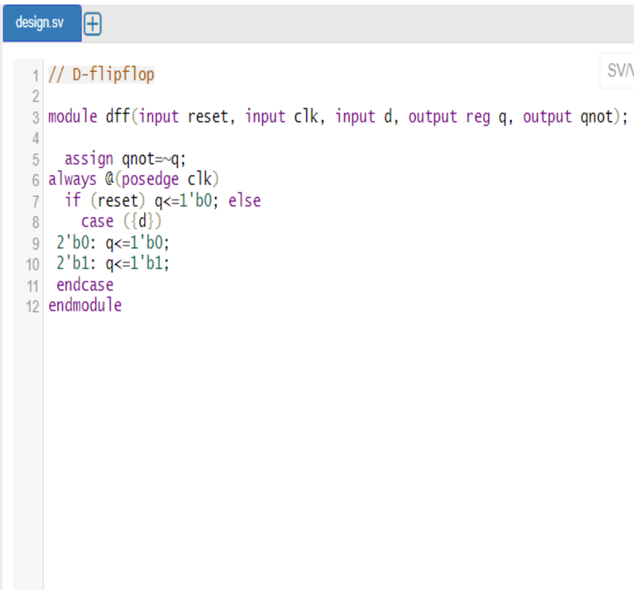
$$Q_{n+1} = Q'_nJ + Q_nK'$$

Q2. Write a Verilog code to implement D flip flop and validate the code via a suitable Test bench code.

Verilog Code:

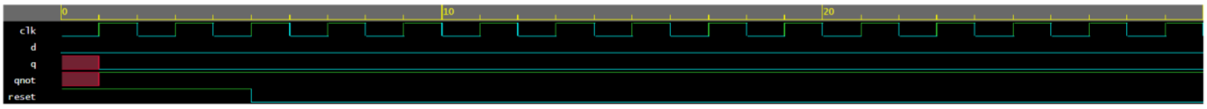


```verilog
// Akshat shah
// 21BCP322
module Dff_tb;

reg clk=0;
reg d=0;
reg reset=1;
wire q, qnot;

  dff dut(reset, clk, d, q,qnot);

initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
    d=1'b0;
#5  reset=1'b0;
#25 $finish;
  end

always #1 clk=~clk;

endmodule
```

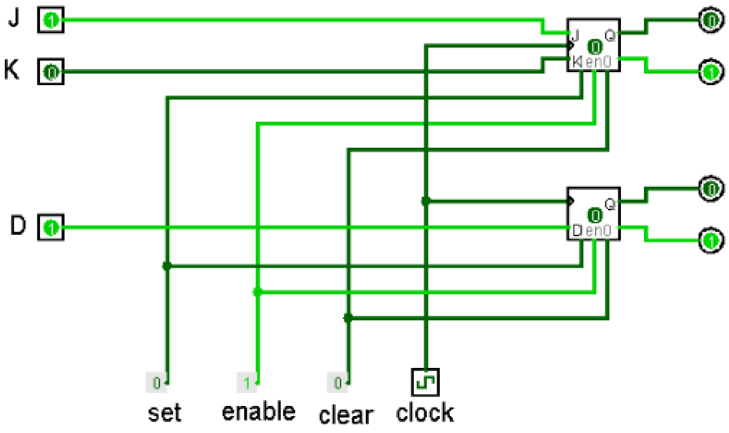


```verilog
// D-flipflop

module dff(input reset, input clk, input d, output reg q, output qnot);

  assign qnot=~q;
always @(posedge clk)
  if (reset) q<=1'b0; else
    case ({d})
 2'b0: q<=1'b0;
 2'b1: q<=1'b1;
 endcase
endmodule
```

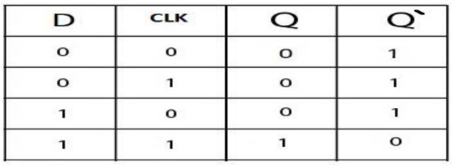
Waveform:



Circuit diagram:

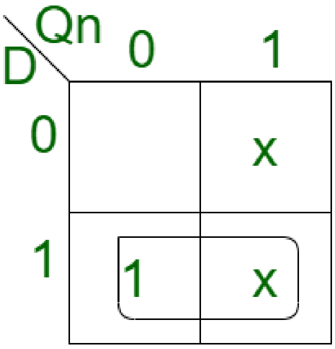


Truth table:



D	CLK	Q	Q`
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

From the truth table,



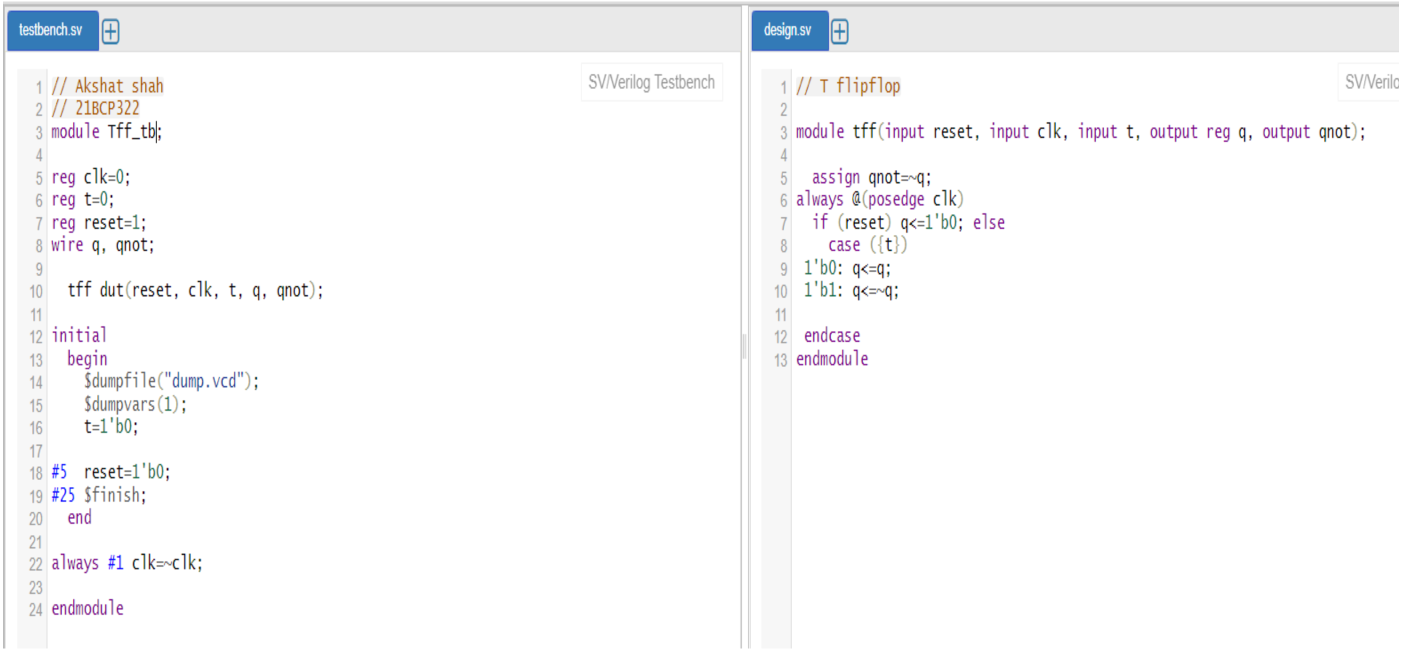
D \ Qn	0	1
0		x
1	1	x

We get

Qn+1 = D.

Q3. Write a Verilog code to implement T flip flop and validate the code via a suitable Test bench code.

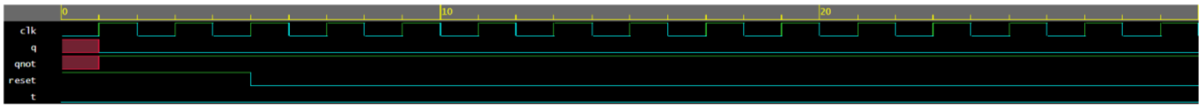
Verilog code:



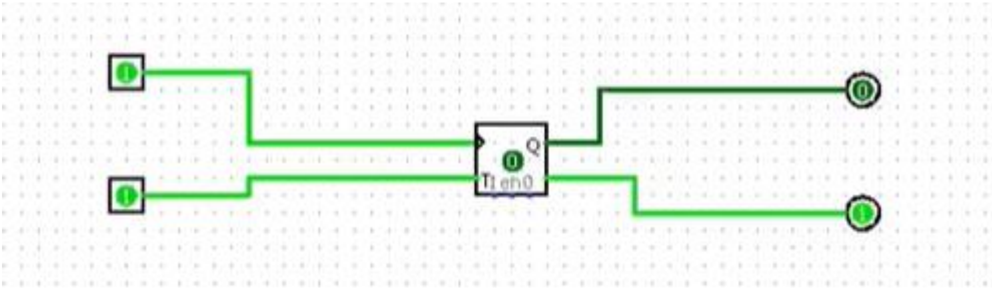
```verilog
// Akshat shah
// 21BCP322
module Tff_tb;

reg clk=0;
reg t=0;
reg reset=1;
wire q, qnot;

  tff dut(reset, clk, t, q, qnot);

initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
    t=1'b0;

#5  reset=1'b0;
#25 $finish;
  end

always #1 clk=~clk;

endmodule
```

```verilog
// T flipflop

module tff(input reset, input clk, input t, output reg q, output qnot);

  assign qnot=~q;
always @(posedge clk)
  if (reset) q<=1'b0; else
    case ({t})
 1'b0: q<=q;
 1'b1: q<=~q;

 endcase
endmodule
```

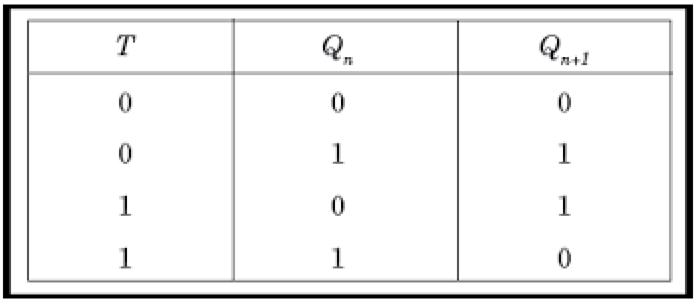
Waveform:



Circuit diagram:

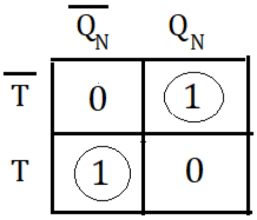


Truth table:



T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

From the truth table,



	$\overline{Q_N}$	Q_N
$\overline{T}$	0	1
T	1	0

We get

Q+1 = T.Q + T.Q