EX. NO: 04	**FLIP-FLOPS**
DATE : 27/09/2024	

AIM: To implement flipflops in Verilog HDL and verify their functionalities through behavioral simulation.

SOFTWARE USED: Xilinx Vivado

HARDWARE USED: Basys3 FPGA Board

ADDERS:

VERILOG HDL CODE:

SR flipflop:

```
module sr_flipflop(s,r,clk,rst,q,qbar);
input s,r,clk,rst;
output reg q,qbar;
always @ (posedge clk)
   begin

   if (rst==0)
      begin
         case ({s,r})
         2'b00: q=q;
         2'b01: q=1'b0;
         2'b10: q=1'b1;
         2'b11: q=1'bx; //Invalid
         endcase
         qbar=~q;
      end
   else
      begin
         q=1'b0;qbar=~q;
      end
   end
endmodule
```

Test bench for SR flipflop:

```
module tb_sr_flipflop;
reg s,r,clk,rst;
wire q,qbar;
sr_flipflop SR(s,r,clk,rst,q,qbar);
always #50 clk = ~clk;
initial
          begin
          clk=1;
          rst=0;
           s=1'b0; r=1'b0;
          #100 s=1'b0; r=1'b1;
          #100 s=1'b1; r=1'b0;
          #100 s=1'b1; r=1'b1;
```

```
        #100 rst=1;
         s=1'b0; r=1'b0;
        #100 s=1'b0; r=1'b1;
        #100 s=1'b1; r=1'b0;
        #100 s=1'b1; r=1'b1;
        #100 $stop;
        end
initial begin
$monitor("At time=%t, clk=%b,rst=%b, s=%b,r=%b,q=%b,qbar=%b",$time,clk,rst,s,r,q,qbar);
        end
endmodule
```

JK flipflop:

```
module jk_flipflop(j,k,clk,rst,q,qbar);
input j,k,clk,rst;
output reg q,qbar;
always @ (posedge clk)
  begin
  if (rst==0)
    begin
      case ({j,k})
      2'b00: q=q;
      2'b01: q=1'b0;
      2'b10: q=1'b1;
      2'b11: q=~q;
      endcase
      qbar=~q;
    end
  else
    begin
      q=1'b0;qbar=~q;
    end
  end
endmodule
```

Test bench for JK flipflop:

```
module tb_jk_flipflop;
reg j,k,clk,rst;
wire q,qbar;
jk_flipflop jk(j,k,clk,rst,q,qbar);
always #50 clk = ~clk;
initial
        begin
        clk=1;
        rst=0;
         j=1'b0; k=1'b0;
        #100 j=1'b0; k=1'b1;
        #100 j=1'b1; k=1'b0;
        #100 j=1'b1; k=1'b1;
        #100 rst=1;
         j=1'b0; k=1'b0;
        #100 j=1'b0; k=1'b1;
        #100 j=1'b1; k=1'b0;
```

```
        #100 j=1'b1; k=1'b1;
        #100 $stop;
        end
initial begin
$monitor("At time=%t, clk=%b,rst=%b, j=%b,k=%b,q=%b,qbar=%b",$time,clk,rst,j,k,q,qbar);
        end
endmodule
```

D flipflop:

```
module d_flipflop(Q,D,clk,reset,Qbar);
input D,clk,reset;
output reg Q,Qbar;
always @(posedge clk or posedge reset)
begin
  if (reset == 1'b1 )
  begin
  Q = 1'b0;
  Qbar=~Q;
  end
  else
  begin
    Q = D;
    Qbar=~Q;
  end
end
endmodule
```

Test bench for D flipflop:

```
module tb_d_flipflop;
reg D,clk,reset;
wire Q,Qbar;
d_flipflop DF(Q,D,clk,reset,Qbar);
initial
  begin
  clk = 1'b1;
  forever #20 clk = ~clk ;
  end
initial
  begin
  reset = 1'b0;
    D = 1'b0;
  #40 D = 1'b1;
  #40 reset = 1'b1;
    D = 1'b0;
  #40 D = 1'b1;
  #40 $finish ;
  end
  initial begin
    $monitor("At time %0t: clk = %b, reset = %b, D = %b, Q = %b, Qbar = %b",
          $time, clk, reset, D, Q, Qbar);
  end
endmodule
```

T flipflop:

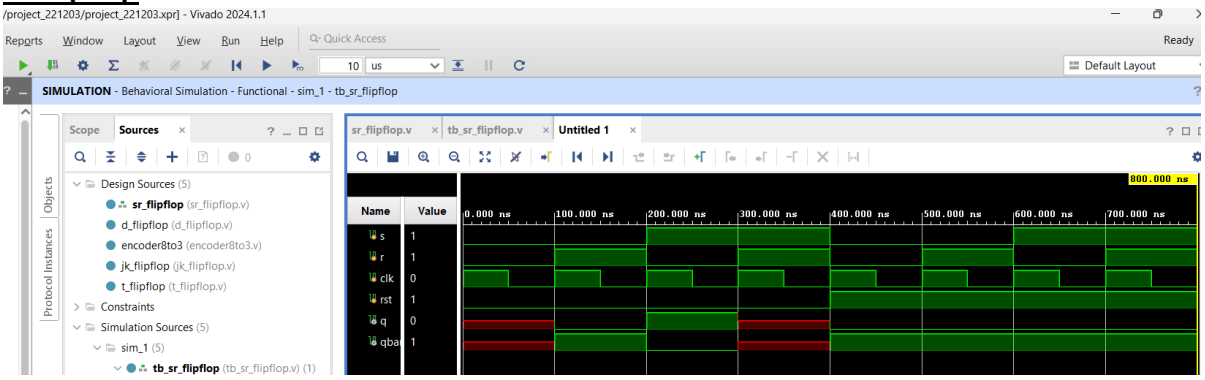
```
module t_flipflop (T,clk,reset,Q,Qbar);
   input  T,clk,reset;
   output reg Q,Qbar;
   always @(posedge clk or posedge reset)
   begin
      if (reset) begin
         Q = 1'b0;
         Qbar=~Q;end
      else
         if (T) begin
         Q = ~Q;
         Qbar=~Q; end
         else
         begin
         Q = Q;
         Qbar=~Q; end
   end
endmodule
```

Test bench for T flipflop:

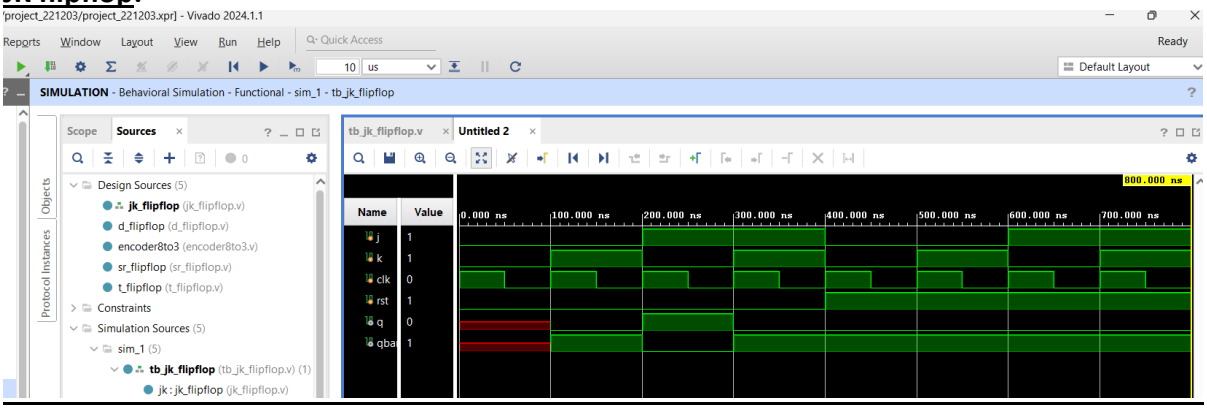
```
module tb_t_flipflop;
   reg T;
   reg clk;
   reg reset;
   wire Q,Qbar;
   t_flipflop TF(T,clk,reset,Q,Qbar);
   initial begin
      clk = 0;
      forever #5 clk = ~clk;
   end
   initial begin
      reset = 1; T = 0;
      #15;
      reset = 0;
      T = 1 ;
      #10 T = 0;
      #10 T = 1;
      #10 T = 0;
      #10 $finish;
   end
   initial begin
      $monitor("Time: %0t | T: %b | clk: %b | reset: %b | Q: %b", $time, T, clk, reset, Q);
   end
endmodule
```

SIMULATION WAVEFORM:

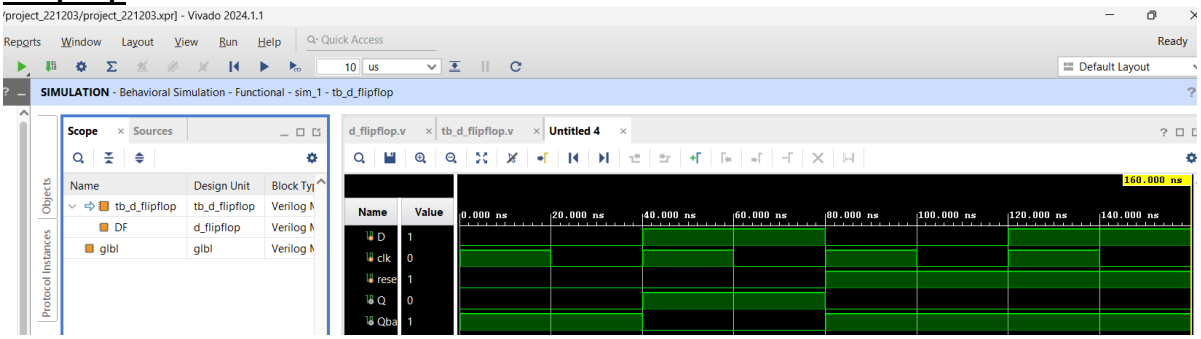
SR flipflop:



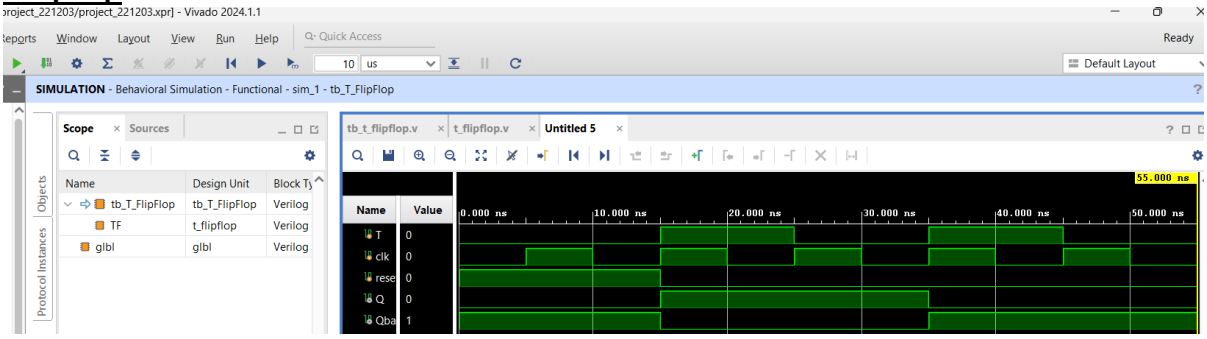
JK flipflop:



D flipflop:



T flipflop:



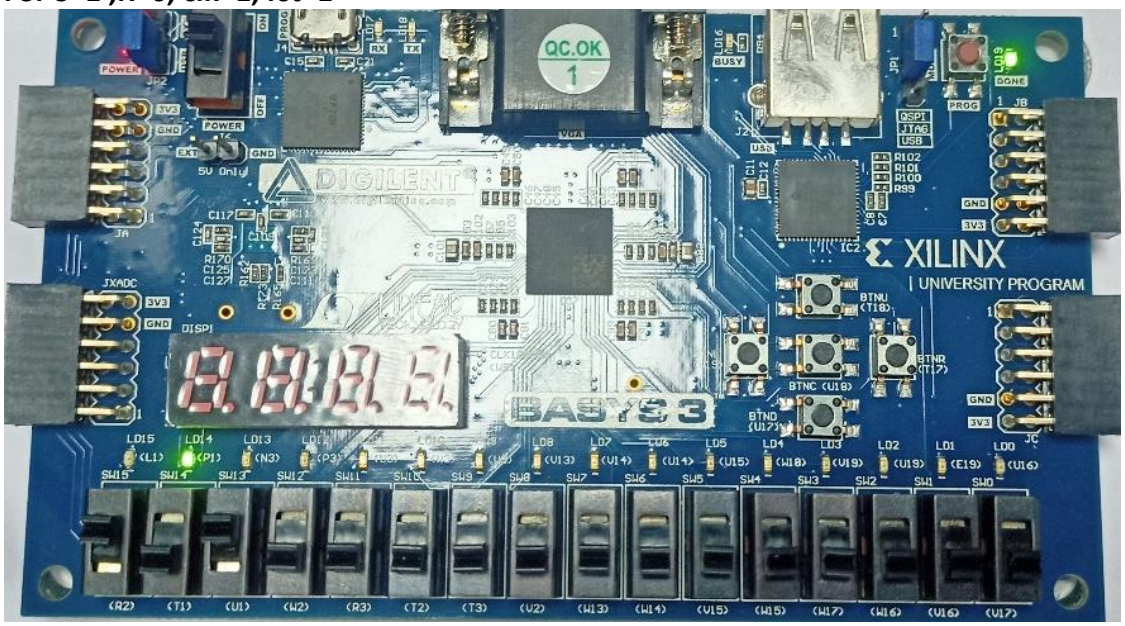
HADWARE OUTPUT:

SR flipflop:

S: U1 clk: W5 Q: L1

R: T1 rst: R2 Qbar: P1

For S=1 ,R=0, clk=1, rst=1

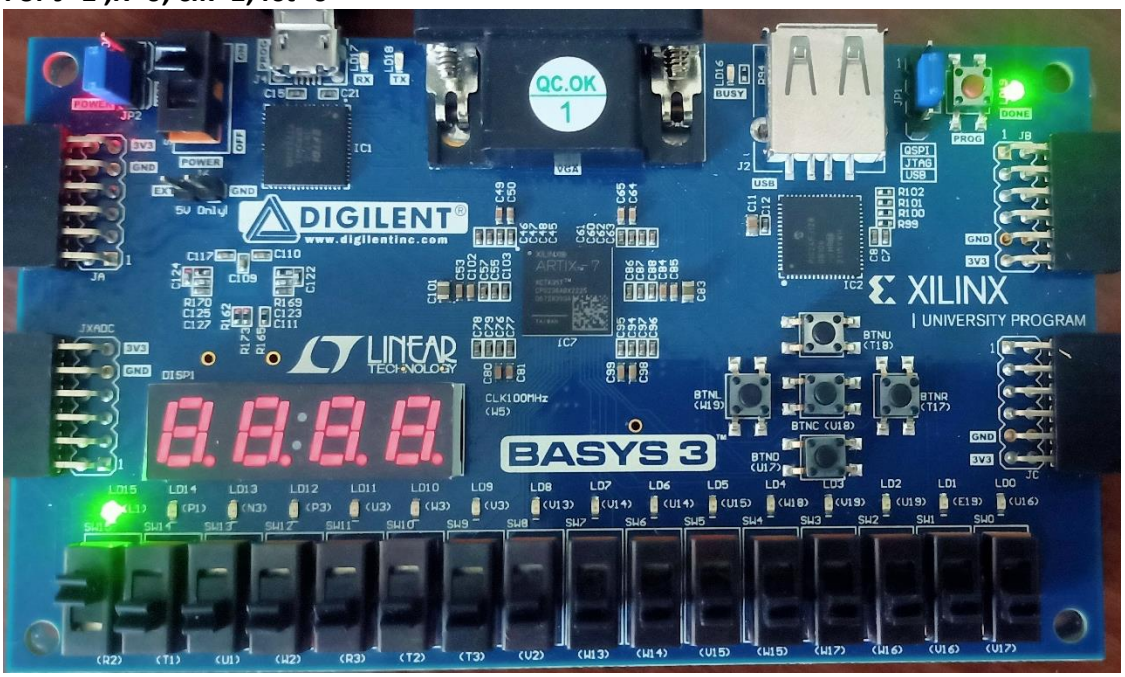


JK flipflop:

J: R2 clk: W5 Q: L1

K: T1 rst: U1 Qbar: P1

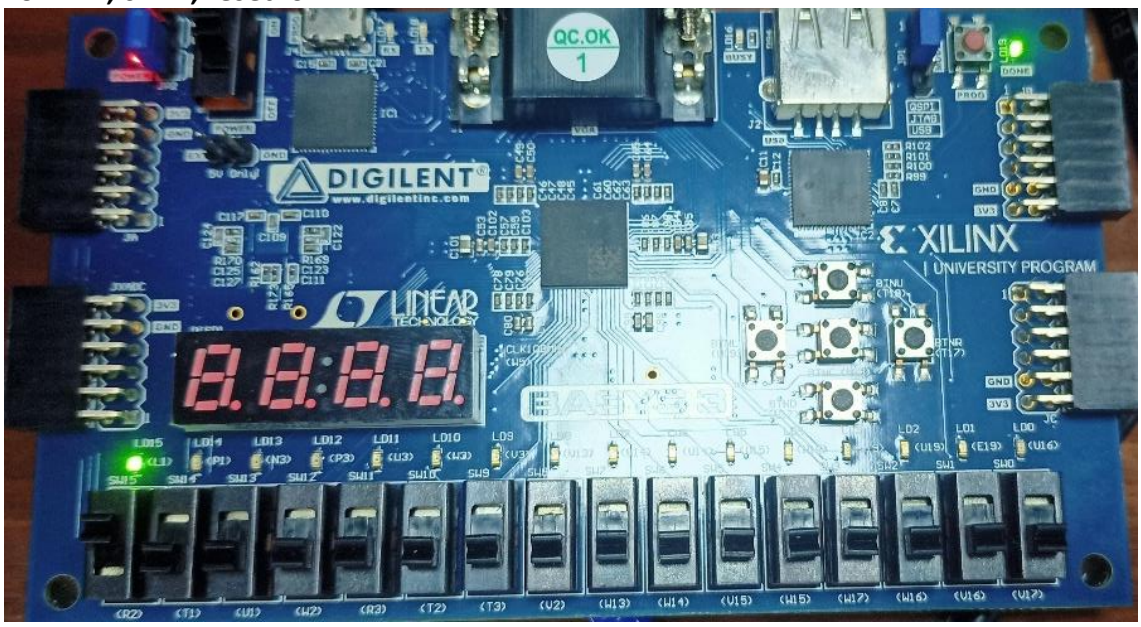
For J=1 ,K=0, clk=1, rst=0



D flipflop:

D: R2
clk: W5 Q: L1
reset: T1 Qbar: P1

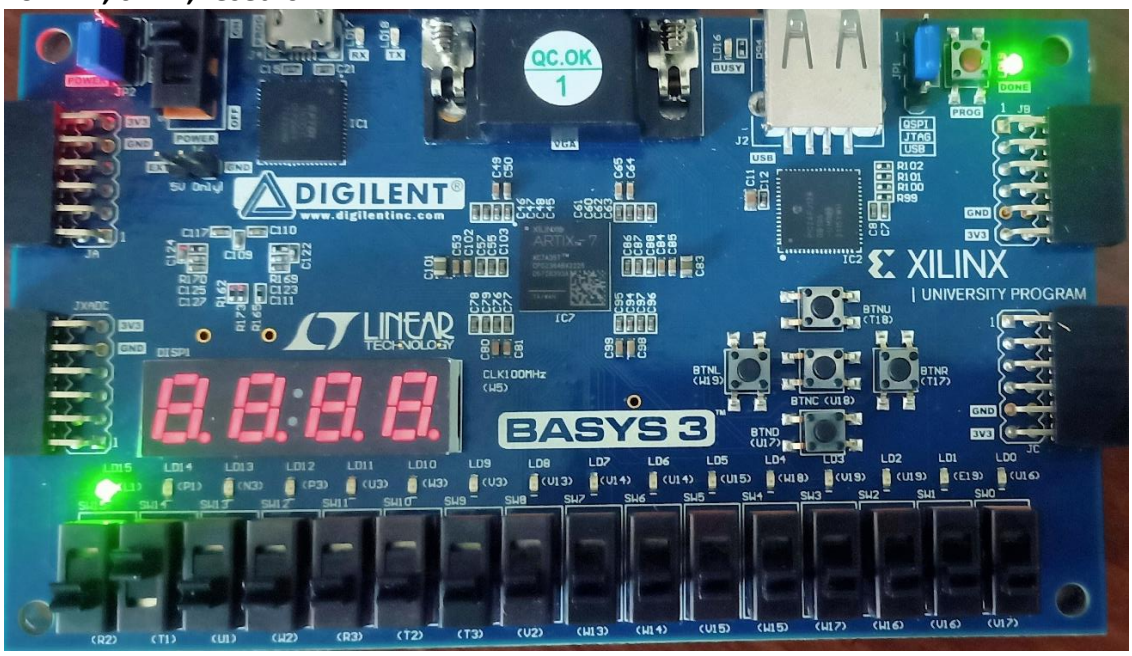
For D=1, clk=1, reset=0



T flipflop:

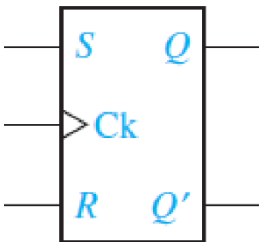
T: T1
clk: W5 Q: L1
reset: R2 Qbar: P1

For T=1, clk=1, reset=0

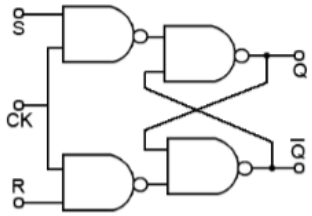


TRUTH TABLES AND CIRCUIT DIAGRAMS:

SR flipflop:



INPUTS			OUTPUT	STATE
CLK	S	R	Q	
X	0	0	No Change	Previous
↑	0	1	0	Reset
↑	1	0	1	Set
↑	1	1	-	Forbidden



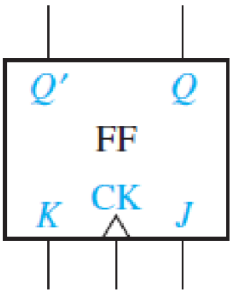
Qn+1 = S + QnR'

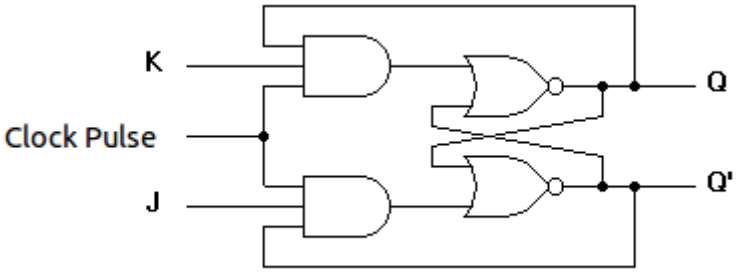
TRUTH TABLE

S	R	Q_N	Q_{N+1}
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	-

JK flipflop:

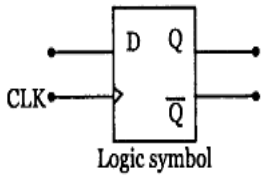
Trigger	Inputs		Output				Inference
			Present State		Next State		
CLK	J	K	Q	Q'	Q	Q'	
⊠	x	x	-		-		Latched
⊓	0	0	0	1	0	1	No Change
⊓			1	0	1	0	
⊓	0	1	0	1	0	1	Reset
⊓			1	0	0	1	
⊓	1	0	0	1	1	0	Set
⊓			1	0	1	0	
⊓	1	1	0	1	1	0	Toggles
⊓			1	0	0	1	





Qn+1 = JQn' + K'Qn

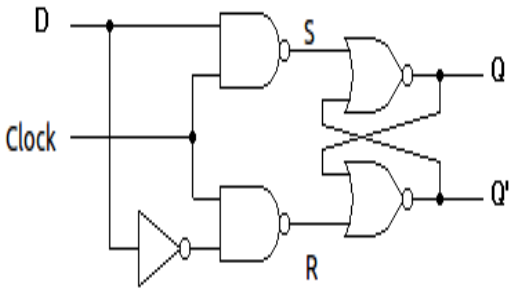
D flipflop:



Logic symbol

CLK	D	Qn	$\overline{Q}n$	Action
0	X	Q_{n-1}	$\overline{Q}_{n-1}$	HOLD
1	0	0	1	Reset
1	1	1	0	Set

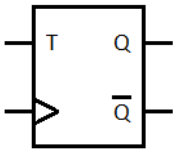
Input			Output	
D	reset	clock	Q	Q'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1



Qn+1 = D

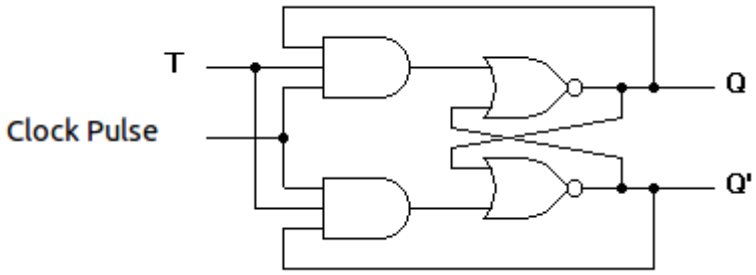
T flipflop:

Symbol



Inputs		Outputs	
CLK	T	Q_{n+1}	Action
0	X	Qn	No change
1	0	Qn	No change
1	1	$\overline{Q}n$	Toggle

	Previous		Next	
T	Q	Q'	Q	Q'
0	0	1	0	1
0	1	0	1	0
1	0	1	1	0
1	1	0	0	1



Qn+1 =T⊕Qn

RESULT:

Thus, flip-flops were successfully implemented in Verilog HDL and their functionalities were verified through behavioral simulation using Xilinx Vivado software on the Basys3 FPGA board.