SCHOOL OF ELECTRONICS ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

VLSI LABORATORY REPORT

(EC-3095)

Submitted By

Name: Debagnik Kar

Roll No: 1804373

Section: ETC-06

Semester: 6th semester

Experiment: 3rd

Realization of flip flop, SR, JK, D and T flip flop. Conversion from one flip flop to another flip flop.

Experiment No - 03

Aim: Realization of flip flop, SR, JK, D, and T flip flop. Conversion from one flip flop to another flip flop (JK to D, JK to T, D to JK, SR to JK).

Software Requiered :-

EDA playground

Theory :-

Flip flop → A flip flop is a device which stores a single bit of data; one of its two states represents a "one" and the other represent a "zero". Such data storage can be used for storage of state and such a circuit is described as sequential logic in electronics.

SRFF → Set - Reset flip flop. It consist of two NOR gates and also two NAND gates. Also known as SR latch. Include two input Set (S) and Reset (R) and two output. Q & Q', clock applied.

Debagnik Kar 1804373

Truth Table of SR FF

Clock	S	R	Q	Qbar	
↑	0	0	Q	Q'	No change
↑	0	1	0	1	Reset
↑	1	0	1	0	Set
↑	1	1	?	?	Invalid
Any other case	x	x	x	x	

- JK FF → It is basically a gated SR flip flop with the addition of a clock i/p circuitry that prevent the invalid o/p condition that can occur when both i/p S & R are equal to logic 1.

Truth Table of JK FF

Clock	J	K	Q	Qbar	
↑	0	0			No change
↑	0	1	0	1	
↑	1	0	1	0	
↑	1	1			Toggle
Any other Case	x	x	x	x	

DFF → It is an edge triggered device which transfers data to Q on clock rising or falling edge. Data latches are level sensitive devices such as the data latch and the transparent latch.

Truth Table of DFF

Clock	D	Q	Qbar
↑	0	0	1
↑	1	1	0
Any other case	x	x	x

TFF → It is an edge triggered device i.e the low to high . or high to low transistions on a clock signal of narrow triggers that is provided as i/p will cause the change in o/p state of flip-flop.

Truth Table of TFF.

Clock	T	Q	Qbar
↑	0	No change	
↑	1	Toggle	
Any other case	x	x	x

Debagnik Kar 18043573

Codes:

Flip-Flop Models:

SR Flip-Flop:-

Design.sv:

```
// Code your design here
module srff(output Q,Qbar,input S,R,clock,reset);
  reg Q;
  assign Qbar=!Q;
  always@(posedge clock or posedge reset) begin
    if(reset)
      Q=0;
    else if({S,R} == 2'b00)
      Q=Q;
    else if({S,R} == 2'b01)
      Q=0;
    else if({S,R} == 2'b10)
      Q=1;
    else if({S,R} == 2'b11)
      Q=1'bz;
    else
      Q=1'bx;
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module test_srff;
  reg S,R,clock,reset;
  wire Q,Qbar;
  srff u1(Q,Qbar,S,R,clock,reset);
  always #5 clock=!clock;
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    clock = 0; S = 0; R = 0;
    reset = 1;
    #10 reset = 0;
    #20 S = 0; R = 1;
    #20 S = 1; R = 0;
    #20 S = 1; R = 1;
  end
endmodule
```

JK FlipFlop:-

Design.sv

```
// Code your design here
module jkff(output Q,Qbar,input J,K,clock,reset);
  reg Q, Qbar;
  always@(posedge clock or posedge reset) begin
    if(reset)
      Q = 0;
    else
      case({J,K})
        2'b00:begin Q = Q; Qbar = !Q;end
        2'b01:begin Q = 0; Qbar = !Q;end
        2'b10:begin Q = 1; Qbar = !Q;end
        2'b11:begin Q = !Q; Qbar = !Q;end
      endcase
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module test_jkff;
  reg J,K,clock,reset;
  wire Q,Qbar;
  jkff u1(Q,Qbar,J,K,clock,reset);
  always #5 clock=!clock;
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    clock = 0; J = 0; K = 0;
    reset = 1;
    #10 reset = 0;
    #20 J = 0; K = 1;
    #20 J = 1; K = 0;
    #20 J = 1; K = 1;
  end
endmodule
```

D FlipFlop:-

Design.sv

```
// Code your design here
module DFF(output Q,Qbar, input D,clk,rst);
  reg Q,Qbar;
  assign Qbar=!Q;
  always @(negedge clk)
  begin
    if(rst==1)
      Q <= 0;
    else
      Q <= D;
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module DFF_tb;
  reg D;
  reg clk;
  reg reset;
  wire Q,Qbar;
  DFF dff(Q,Qbar,D,clk,reset);
  initial begin
    clk=0;
    forever #5 clk = ~clk;
  end
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
    reset=1;
    D = 0;
    #50;
    reset=0;
    D = 1;
    #50;
    D = 0;
    #50;
    D = 1;
  end
endmodule
```

T FlipFlop:-

Design.sv

```
// Code your design here
module tff(output Q,Qbar,input T,clock,reset);
  reg Q;
  assign Qbar=!Q;
  always@(posedge clock or posedge reset) begin
    if(reset)
      Q <= 0;
    else if(T==1)
      Q = !Q;
    else if(T==0)
      Q = Q;
    else
      Q = 1'bx;
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module test_tff;
  reg T,clock,reset;
  wire Q,Qbar;
  dff u1(Q,Qbar,T,clock,reset);
  always #5 clock=!clock;
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    clock = 0; T=0;
    reset = 1;
    #10 reset = 0;
    #20 T = 1;
  end
endmodule
```

FlipFlop Conversions:-

D ff to JKff

Design.sv

```
// Code your design here
`include "DFF.sv"
module jkff(output Q,Qbar, input J,K,clk,rst);
  assign Q=!Qbar;
  wire w0,w1,w2,w3,w4;
  and g0(w0,J,Qbar);
  not g1(w1,K);
  and g2(w2,w1,Q);
  or g3(w3,w0,w2);
  DFF g4(Q,Qbar,w3,clk,rst);
endmodule
```

DFF.sv

```
// Code your design here
module DFF(output Q,Qbar, input D,clk,rst);
  reg Q,Qbar;
  assign Qbar=!Q;
  always @(posedge clk)
  begin
    if(rst==1'b1)
      Q = 1'b0;
    else
      Q = D;
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module DFF_tb;
  reg J,K;
  reg clk;
  reg reset;
  wire Q,Qbar;
  jkff ff0(Q,Qbar,J,K,clk,reset);
  initial begin
    clk=0;
    forever #5 clk = ~clk;
  end
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    reset=1;J = 0;K = 0;
    #25;reset=0;
    #25;J = 0;K = 1;
    #25;J = 1;K = 0;
    #25J = 1;K = 1;
  end
endmodule
```

SRff – JKff

Design.sv

```
// Code your design here
`include "SRFF.sv"
module jkff(output Q,Qbar,input J,K,clk,rst);
  wire w1,w2;
  and a1(w1,J,Qbar);
  and a2(w2,K,Q);
  srff L0(Q,Qbar,w1,w2,clk,rst);
endmodule
```

SRFF.sv

```
// Code your design here
module srff(output Q,Qbar,input S,R,clock,reset);
  reg Q;
  assign Qbar=!Q;
  always@(posedge clock or posedge reset) begin
    if(reset)
      Q=0;
    else if({S,R} == 2'b00)
      Q=Q;
    else if({S,R} == 2'b01)
      Q=0;
    else if({S,R} == 2'b10)
      Q=1;
    else if({S,R} == 2'b11)
      Q=1'bz;
    else
      Q=1'bx;
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module jkff_tb();
  reg j,k,clk,rst;
  wire q,qb;
  jkff U0(q,qb,j,k,clk,rst);
    always #5 clk=!clk;
  initial begin
      $dumpfile("dump.vcd");
      $dumpvars;
      clk=1;rst=1;#10;
      rst=0;j=0;k=0;#20;
      j=0;k=1;#20;
      j=1;k=0;#20;
      j=1;k=1;#20;
    end
endmodule
```

JKff – Tff

Design.sv

```
//T flipflops
module tff(output Q,Qbar, input T,clk,rst);
  jkff jk1(Q,Qbar,T,T,clk,rst);
endmodule
```

JKFF.sv

```
// Code your design here
module jkff(output Q,Qbar,input J,K,clock,reset);
  reg Q, Qbar;
  always@(posedge clock or posedge reset) begin
    if(reset)
      Q = 0;
    else
      case({J,K})
        2'b00:begin Q = Q; Qbar = !Q;end
        2'b01:begin Q = 0; Qbar = !Q;end
        2'b10:begin Q = 1; Qbar = !Q;end
        2'b11:begin Q = !Q; Qbar = !Q;end
      endcase
  end
endmodule
```

Testbench.sv

```
// Code your testbench here
// or browse Examples
module tb;
  reg T,D0,clk,rst;
  wire Q0,Q0bar,Q1,Q1bar;
  //dff df(Q0,Q0bar,D0,clk,rst);
  tff tf(Q1,Q1bar,T,clk,rst);
  always #5 clk =! clk;
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    clk = 0;
    rst = 1;
    D0=0;T=0;
    #10;
    rst = 0;
    D0=0;T=1;
    #45;
    D0=1;T=0;
  end
endmodule
```

JKff – Dff

Design.sv

```
// D flipflops
`include "JKFF.sv"
module dff(output Q,Qbar, input D,clk,rst);
  wire Dbar;
  not v0(Dbar,D);
  jkff jk0(Q, Qbar,D,Dbar,clk,rst);
endmodule
```

JKFF.sv

```
// Code your design here
module jkff(output Q,Qbar,input J,K,clock,reset);
  reg Q, Qbar;
  always@(posedge clock or posedge reset) begin
    if(reset)
      Q = 0;
    else
      case({J,K})
        2'b00:begin Q = Q; Qbar = !Q;end
        2'b01:begin Q = 0; Qbar = !Q;end
        2'b10:begin Q = 1; Qbar = !Q;end
        2'b11:begin Q = !Q; Qbar = !Q;end
      endcase
  end
endmodule
```

Tesbench.sv

```
// Code your testbench here
// or browse Examples
module tb;
  reg T,D0,clk,rst;
  wire Q0,Q0bar,Q1,Q1bar;
  dff df(Q0,Q0bar,D0,clk,rst);
  //tff tf(Q1,Q1bar,T,clk,rst);
  always #5 clk =! clk;
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    clk = 0;
    rst = 1;
    D0=0;T=0;
    #10;
    rst = 0;
    D0=0;T=1;
    #45;
    D0=1;T=0;
  end
endmodule
```

Observation:

Behaviourial models:

Fig 3.1: EPWave of SR FlipFlop

Fig 3.2: EPWave of JK FlipFlop

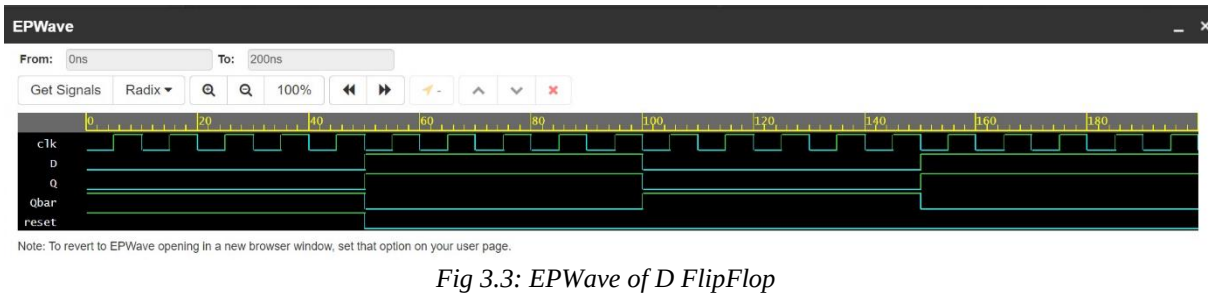

Fig 3.3: EPWave of D FlipFlop

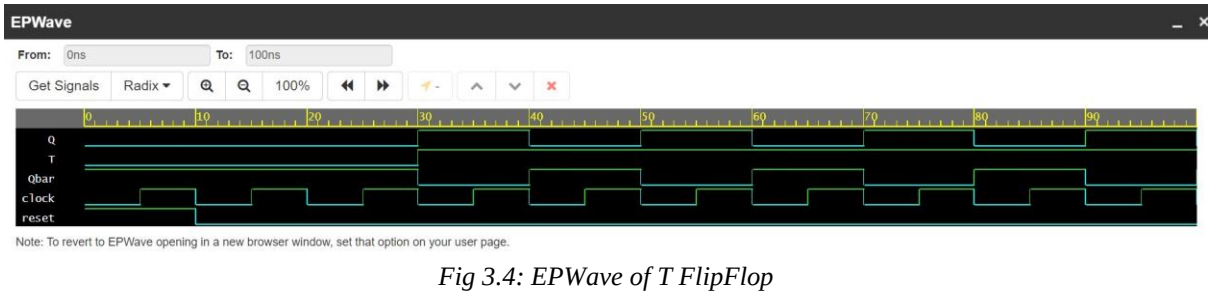

Fig 3.4: EPWave of T FlipFlop

Conversion of FlipFlops:

Fig 3.5: EPWave of JK FlipFlop from D flipflop

Fig 3.6: EPWave of converted JK FlipFlop from SR FlipFlop

Fig 3.7: EPWave of converted T flipflop from JK FlipFlop

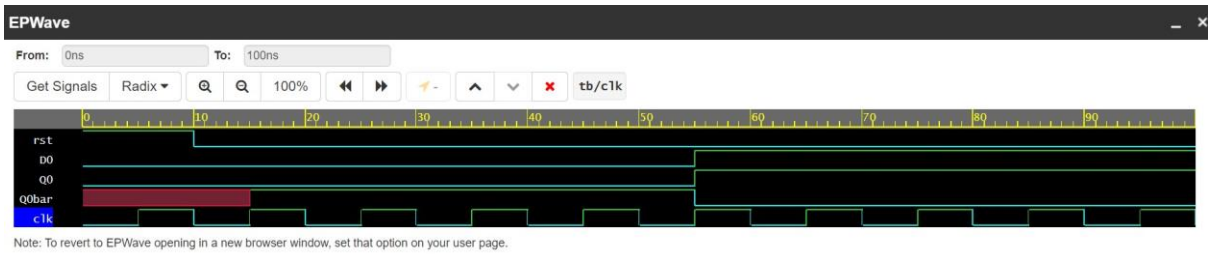

Fig 3.8: EPWave of converted D flipflop from JK flipflop

Conclusion - In this experiment we learned about different flip flops and we successfully designed SR, JK, D & T flip-flops by writing codes in EDA playground. We also got the output waveforms which was same as their truth table outputs. We also designed one flip flop from another flip flop (JK to D, JK to T, D to Jk & SR to Jk) and simulated their outputs.

Debagnik Kar 1809373