DESIGN OF SEQUENTIAL CIRCUITS USING HDL

1. Write the hdl code with behavior modeling for implementing jk and t flipflop.

JK flipflop :

module jk(q,qnot,clk,j,k,reset);

output q,qnot;

input j,k,clk,reset;

reg q;

always @(posedge clk or negedge reset)

case({j,k})

2'b00: if(~reset)

q=1'b0;

else

q=q;

2'b01: if(~reset)

q=1'b0;

else

q=1'b0;

2'b10: if(~reset)

q=1'b0;

else

q=1'b1;

2'b11: if(~reset)

q=1'b0;

else

q=~q;

endcase

assign qnot =~q;

endmodule

TESTBENCH :

module t\_jk;

reg j,k,clk,reset;

wire q,qnot;

jk f1(q,qnot,clk,j,k,reset);

always #5 clk=~clk;

initial

begin

$dumpfile("test1.vcd");

$dumpvars(1,f1.jk);

$monitor("%b%b=%b%b",j,k,q,qnot);

clk=1'b1;

j=1'b0;k=1'b0;#30

j=1'b0;k=1'b1;#30

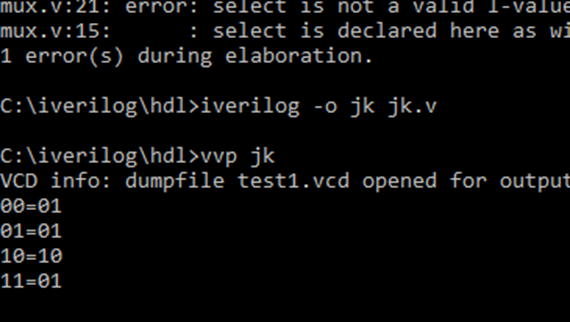
j=1'b1;k=1'b0;#30

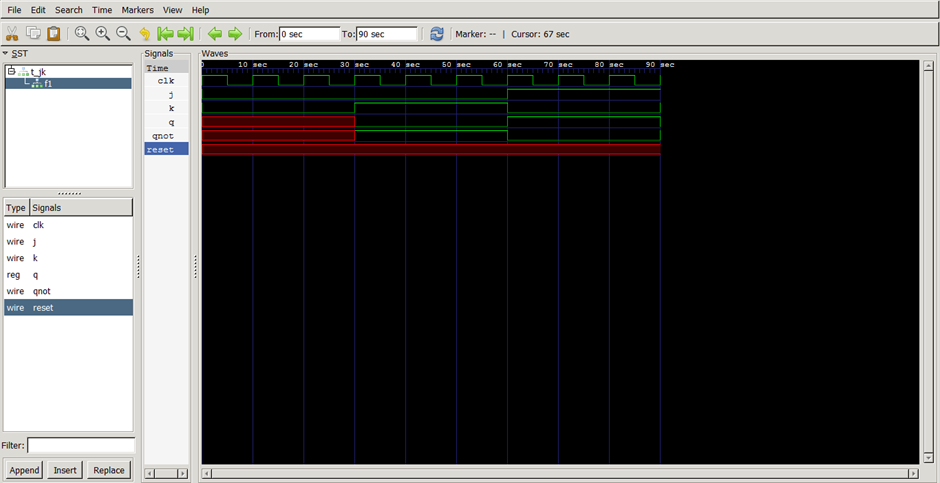
j=1'b1;k=1'b1;

$finish;

end

endmodule





T flipflop :

module tflip(q,qnot,clk,t,reset);

output q,qnot;

input t,clk,reset;

reg q;

always @(posedge clk or negedge reset)

case(t)

1'b0:if(~reset) q=1'b0;

else q=q;

1'b1:if(~reset) q=1'b0;

else q=~q;

endcase

assign qnot=~q;

endmodule

testbench :

module t\_tflip;

reg t,clk,reset;

wire q,qnot;

tflip f1(q,qnot,clk,t,reset);

always #5 clk=~clk;

initial

begin

$dumpfile("tflipflop.vcd");

$dumpvars(1,f1.tflip);

$monitor("%b=%b%b",t,q,qnot);

reset=1'b1;

clk=1'b1;

t=1'b0; #30

t=1'b1;#30

reset=1'b0;

reset=1'b1;

clk=1'b1;

t=1'b0; #30

t=1'b1;#30

$finish;

end

endmodule

module t\_tflip;

reg t,clk,reset;

wire q,qnot;

tflip f1(q,qnot,clk,t,reset);

always #5 clk=~clk;

initial

begin

$dumpfile("tflipflop.vcd");

$dumpvars(1,f1.tflip);

$monitor("%b=%b%b",t,q,qnot);

reset=1'b1;

clk=1'b1;

t=1'b0; #30

t=1'b1;#30

reset=1'b0;

reset=1'b1;

clk=1'b1;

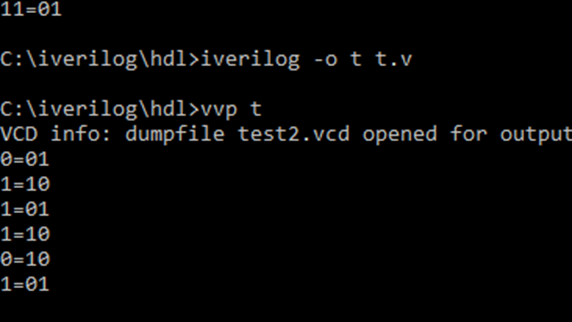
t=1'b0; #30

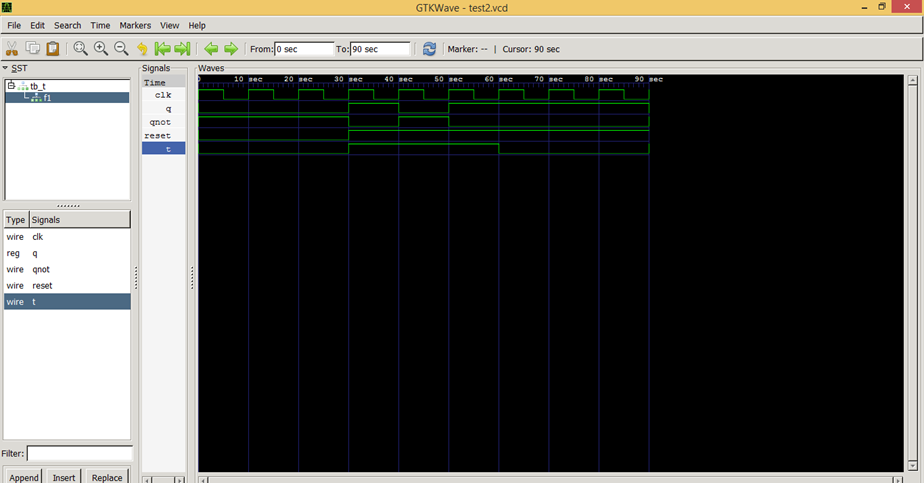
t=1'b1;#30

$finish;

end

endmodule





1. Implement Shift registers using HDL.

module shiftreg(a,lin,rin,clk,clr,i,select);

output [3:0]a;

input [3:0]i;

input lin,rin,clk,clr;

input [1:0]select;

reg[3:0]a;

always @(posedge clk or negedge clr)

if(~clr)

a=4'b0000;

else

case(select)

2'b00: a=a;

2'b01: a={rin,a[3:1]};

2'b10: a={a[2:0],lin};

2'b11: a=i;

endcase

endmodule

module t\_shiftreg;

reg lin,rin,clk,clr;

reg [3:0]i;

reg[1:0]select;

wire[3:0]a;

shiftreg t1(a,lin,rin,clk,clr,i,select);

always #5 clk=~clk;

initial

begin

$monitor("%b%b%b%b",a[3],a[2],a[1],a[0]);

clk=1'b1;

clr=1'b0;#10

clr=1'b1;

i=4'b0011;lin=0;rin=1;

select=2'b00; #30

repeat(4) select=2'b01;#30

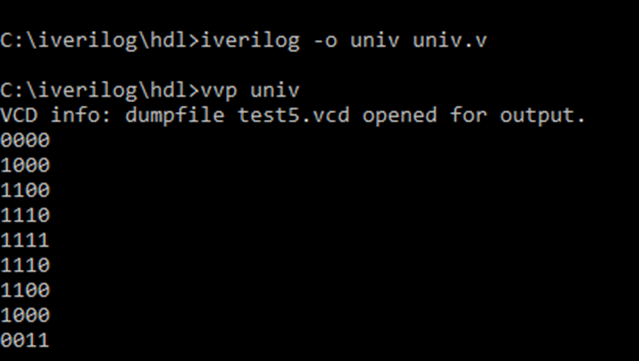
repeat(2) select=2'b10; #30

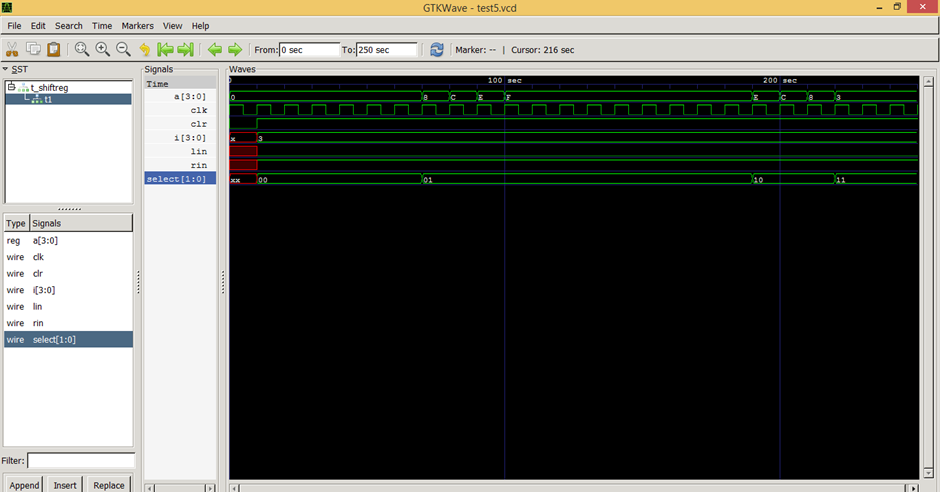
select=2'b11;

$finish;

end

endmodule





1. Write the HDL code for implementing a 3bit binary ripple counter.

module binrip(q,clk,reset);

output q;

input clk,reset;

reg q;

always @(negedge clk or posedge reset)

if(reset) q=1'b0;

else q=~q;

endmodule

module bin(a0,a1,a2,clk,reset);

output a0,a1,a2;

input clk,reset;

binrip f0(a0,clk,reset);

binrip f1(a1,clk,reset);

binrip f2(a3,clk,reset);

endmodule

module t\_binrip;

reg clk,reset;

wire a0,a1,a2;

bin b0(a0,a1,a2,clk,reset);

always #5 clk=~clk;

initial

begin

$dumpfile("ripple.vcd");

$dumpvars(1,b0.bin);

clk=1'b0;

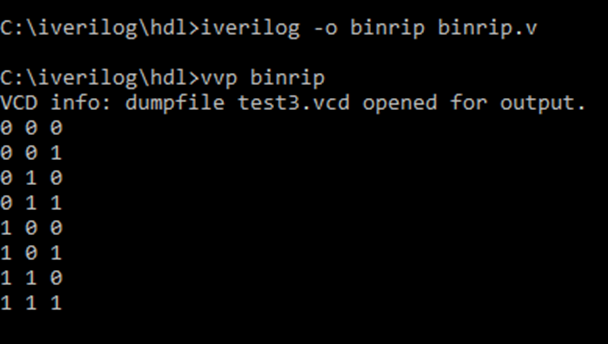
reset=1'b1;#20

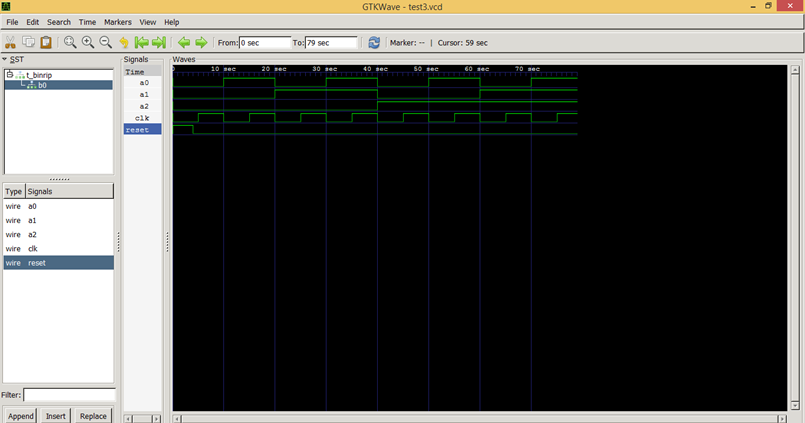
reset=1'b0;#20

$finish;

end

endmodule





1. Write the HDL code to implement a circuit for the following state diagram.

module jks(nxtstate,prtstate,clk,x);

output [1:0]nxtstate;

input clk,x;

input [1:0]prtstate;

reg [1:0]nxtstate;

always @(posedge clk)

case(prtstate)

2'b00: if(x) nxtstate=2'b00;

else nxtstate=2'b01;

2'b01: if(x) nxtstate=2'b10;

else nxtstate=2'b11;

2'b10: if(x) nxtstate=2'b10;

else nxtstate=2'b11;

2'b11: if(x) nxtstate=2'b11;

else nxtstate=2'b00;

endcase

endmodule

module t\_jks;

reg x,clk;

reg [1:0]prtstate;

wire [1:0]nxtstate;

jks f1(nxtstate,prtstate,clk,x);

always #5 clk=~clk;

initial

begin

$monitor("%b%b%b=%b%b",prtstate[1],prtstate[0],x,nxtstate[1],nxtstate[0]);

clk=1'b1;

prtstate=2'b00;x=1'b0; #20

prtstate=2'b00;x=1'b1; #20

prtstate=2'b01;x=1'b0; #20

prtstate=2'b01;x=1'b1; #20

prtstate=2'b10;x=1'b0; #20

prtstate=2'b10;x=1'b1; #20

prtstate=2'b11;x=1'b0; #20

prtstate=2'b11;x=1'b1;

$finish;

end

endmodule

