2:1

Module 2\_1 mux(I0,I1,Y,S);

Input I1,I0,S;

Output Y;

Reg Y;

Always@(s,I0,I1)

Begin

If(s==0)

Begin

Y=I0;

end

Else

Begin

Y=I1;

End

End

Endmodule

1:2 demux

Module demux(I,s0,y0,y1);

Input I;

Input s0;

Output y0;

Output y1;

Reg y0,y1;

Always @(I or 3)

Begin

If(s==0)

Y0=I;

Else

Begin

Y1=I;

End

End

End module

Module sr\_ff(q,qb,clk,a,r)

Input a,r,clk;

Output q,qb;

Reg q,qb;

Always @(posedgeclk)

Begin

Case(s,r)

2'b00:q<=q;

2'b01:q<=1'b0;

2'b10:q<=1'b1;

2'b11:q<=1'bz;

End case

Assign qb=~q;

End

Endmodule

Jk flip flop

Module jkff(j,k,q,qb,clk);

Input I,k,clk;

Output q,qb;

Reg q,qb;

Always(posedge clk)

Begin

Case({J,K})

2'b00:q<=q;

2'b01:q<=1'b0;

2'b10:q<=1'b1;

2'b11:q<=1'qb;

Endcase

End

Assign qb=~q;

End module

Dflipflop

Module D\_ff(q,qb,d,clk)

Input clk,d;

Output q,qb;

Reg q,qb;

Always @(posedge clock)

Begin

Case{D}

1'b0:q<=1'b0;

1'b1:q<=1'b1;

End case

End

Assign qb<=~q;

Endmodule

Bcd adder

Module bcd\_adder(a,b,carry\_in,sum,carry)

Input [3:0] a,b;

Input carry\_in;

Output[3:0]sum;

Reg[4:0]sum\_temp;

Reg[3:0]sum;

Reg carry;

Always @(a,b,carry\_in)

Begin

Sum\_temp=a+b+carry\_in;

If(sum\_temp>a)

Begin

Sum\_temp=sum\_temp+6;

Carry=1;

Sum=sum+temp[3:0];

End

Else

Begin

Carry=0;

Sum=sum\_trmp[3:0];

End

End endmodule