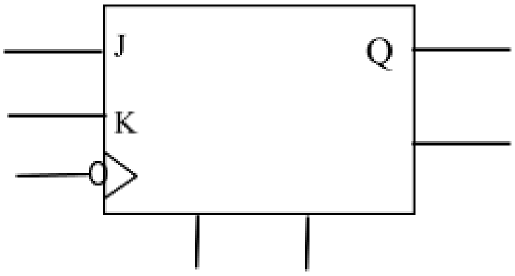
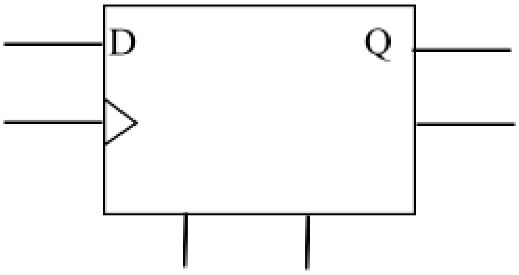
JK Flip-flop:



```
module JKFF(output Q,output Qb, input J,input K, input clk,input CLR,input PRE);

reg Q; // output Q and Qb defined as register
reg Qb;
  initial
    begin
      Q=0;
      Qb=1;
    end
  always @ (negedge clk or negedge CLR or negedge PRE)
begin
  if (CLR == 0)
    begin
    Q = 0;
    Qb = 1;
    end
 else
   if (PRE == 0)
    begin
    Q  = 1;
    Qb = 0;
    end
  else
    if ( J == 0 & K == 0 )
    begin
    Q = Q;
    Qb = ~Q;
    end
  else
    if ( J == 0 & K == 1 )
    begin
    Q = 0;
    Qb = 1;
    end
  else
    if ( J == 1 & K == 0 )
    begin
    Q = 1;
    Qb = 0;
    end
  else
    if ( J == 1 & K == 1 )
    begin
    Q = ~Q;
    Qb = ~Qb;
    end

end
endmodule
```



D Flip-flop

```
module DFF(output Q,output Qb, input D,input clk,input CLR,input PRE);
reg Q;       // output Q and Qb defined as register
reg Qb;
  initial
    begin
      Q=0;
      Qb=1;
    end
  always @(posedge clk or negedge CLR or negedge PRE)
begin
  if   (CLR == 0)
    begin
    Q = 0;
    Qb = 1;
    end
 else
   if   (PRE == 0)
    begin
    Q  = 1;
    Qb = 0;
    end
  else
    begin
    Q = D;
    Qb = ~D;
    end
end
endmodule
```

4 bit SISO:

```
module SISO (d_out,d_outb,d_in,clk,CLR,PRE);
        input d_in,clk,CLR,PRE;
        output d_out,d_outb;

        wire Q3,Q3b,Q2,Q2b,Q1,Q1b;

        D_FF  flip_flop1  (Q1,Q1b,d_in,clk,CLR,PRE);
        D_FF  flip_flop2  (Q2,Q2b,Q1,clk,CLR,PRE);
        D_FF  flip_flop3  (Q3,Q3b,Q2,clk,CLR,PRE);
        D_FF  flip_flop4  (d_out,d_outb,Q3,clk,CLR,PRE);

endmodule
```

2 bit asynchronous counter using JK Flip flops:

```
module two_bit_asynch_counter (Q1,Q0,Q1b,Q0b,clk,CLR,PRE);

         output Q1,Q0,Q1b,Q0b;
         input clk,CLR,PRE;

         JK_FF flip_flop1 (Q0,Q0b,1,1,clk,CLR,PRE);

         JK_FF flip_flop2 (Q1,Q1b,1,1,Q0,CLR,PRE);

endmodule
```