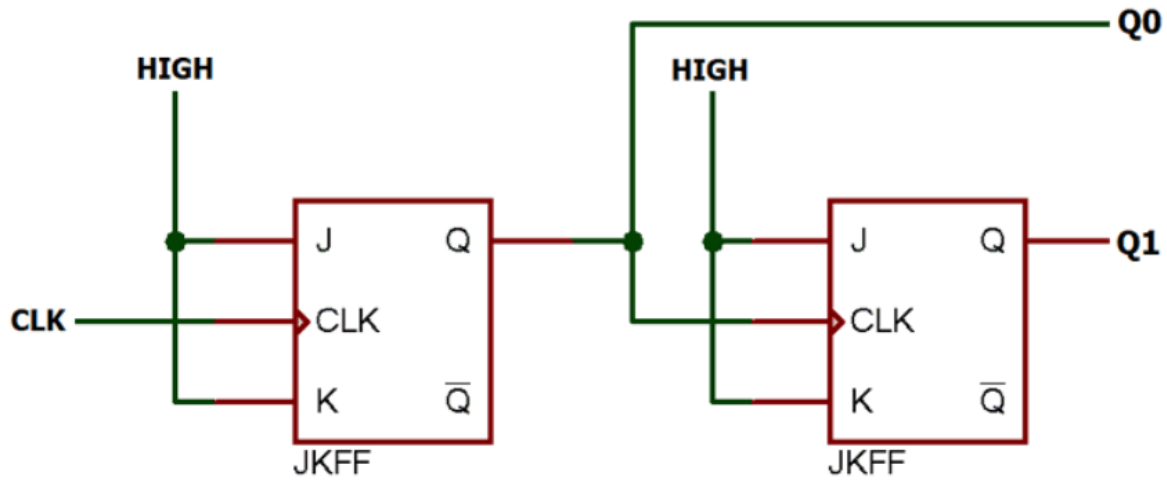
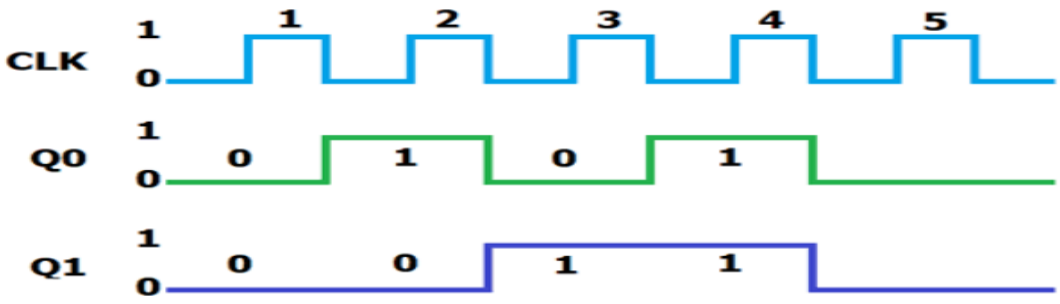
DAY 27

2 BIT ASYNCHRONOUS COUNTER:



A 2-bit ripple counter is shown in the above figure. The external clock is connected to the clock input of the first flip-flop. So, this is why the first flip-flop changes the state at the quick falling edge of the clock pulse. But the second flip-flop changes only when it is triggered by the Q output of the first flip-flop. Due to an essential propagation delay in the circuit through a flip-flop, the change in the input clock pulse and change of the Q output of the first flip-flop can never occur at the same time giving the exact result. Therefore, the triggering of flip-flops cannot be simultaneous



Explanation

The transitions of Q 0, Q 1, and clock pulse in the figure of the timing diagram above are simultaneous. This phenomenon occurred here although it is an asynchronous counter. There is a small delay between the clock and the first and second transitions in the above counter.
All the clear inputs are connected together so that a single pulse can clear all the flip-flops before the counting of bits. The clock pulse given into the first flip-flop is rippled through the other counters after the propagation delay. Hence, its name is a ripple counter.

The above two-bit ripple counter has four states. There is the placement of each counter to correspond to the account value. Like the 2-bit counter, a counter having n number of flip-flops can have 2 to the power n states. These numbers of states are asserted as mod numbers. Thus, for a 2-bit counter, there is a mod 4 counter by the mechanism of 2 to the power n. Therefore, a bit counter can also be called a mod 4 counter

RTL CODE:

```
module jkff(input j,k,clk, output reg q,output reg qbar);
  always@(negedge clk)
    begin
      case({j,k})

        2'b00: q <=q;
        2'b01: q <=0;
        2'b10: q <=1;
        2'b11: q <= ~q;
```

```
      endcase
      assign qbar= ~q;
    end
endmodule

module Asynchronous_counter(input [1:0]j,input[1:0]k,clk,output
q,qbar,[1:0]count);
  wire QA,QB;
  jkff a1(j[0],k[0],clk,QA,QB);
  jkff a2(j[1],k[1],QA,q,qbar);
  assign count={q,QA};
endmodule
```

TEST BENCH:

```
module testbench;
  reg [1:0] j;
  reg [1:0] k;
  reg clk;
  wire q, qbar;
  wire [1:0]count;
  Asynchronous_counter s1(j, k, clk, q, qbar, count);

  initial begin
    $dumpfile("dump.vcd");
```

```
  $dumpvars(1);
 end

 initial begin
  clk = 0;
  forever #10 clk = ~clk;
 end

 initial begin
  repeat (5) begin

   j = $random;
   k = $random;
   #5 ;
  end
 end

 initial
  #60 $finish();
Endmodule
```