Divide by 32 asynchronous UP counter
EE210P - Digital System Design

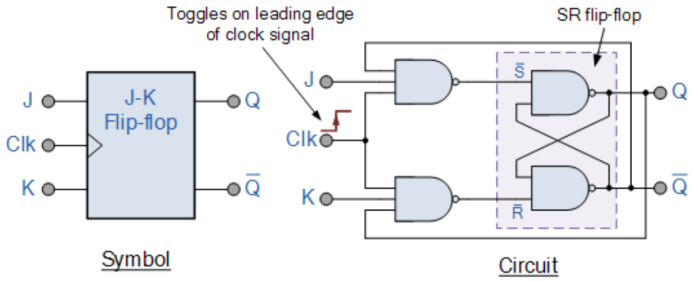
By Adarsh Santoria
B21176

***Abstract*—This report demonstrates the operation of Divide by 32 asynchronous UP counter using combinations of J K Flip Flops . Verilog language, which is a Hardware Descriptive Language (HDL), is used to describe the low level hardware, on Xilinx Vivado.**

I. Introduction

A. J K Flip Flop

This simple JK flip Flop is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The two inputs labelled "J" and "K" are not shortened abbreviated letters of other words, such as "S" for Set and "R" for Reset, but are themselves autonomous letters chosen by its inventor Jack Kilby to distinguish the flip-flop design from other types.

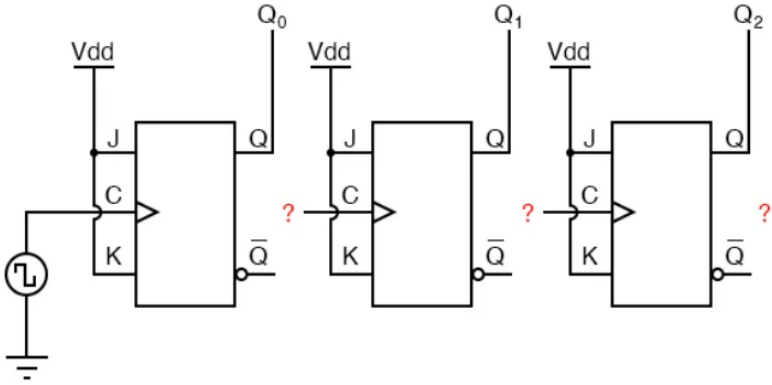


The truth table is displayed bellow -

	Clock	Input		Output		Description
	Clk	J	K	Q	$\overline{Q}$	
same as for the SR Latch	X	0	0	1	0	Memory no change
	X	0	0	0	1	
	↓	0	1	1	0	Reset Q » 0
	X	0	1	0	1	
	↓	1	0	0	1	Set Q » 1
	X	1	0	1	0	
toggle action	↓	1	1	0	1	Toggle
	↓	1	1	1	0	

B. Asynchronous UP counter

In asynchronous/ripple counter output of the first flip-flop is provided as the clock to the second flip-flop i.e flip-flop(FF) are not clocked simultaneously. Circuit is simpler, but speed is slow. When two FFs are connected in series and output of one FF is act as clock for 2nd FF. So the state of 2nd FF will change only when output and 1st FF is logic 1 and falling edge occur. The output frequency of Q1 is f/4(if f is clock frequency). It can generate 4 different unique states. This is known as divide by 4 circuits or mod 4 ripple counter. The circuit diagram for 3-bit is displayed here-



The general mechanism table for 3-bit up counter is displayed bellow -

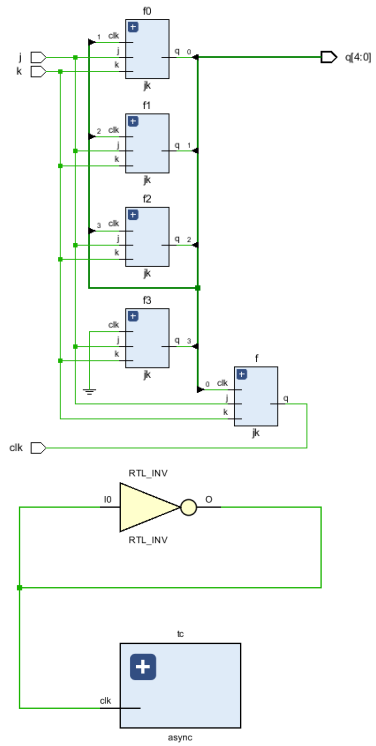
State	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

In code we deal with 5-bit (divide by 32) upcounter.

II. Synthesis and Simulations

References

[1] Schematic Diagram of UP Counter

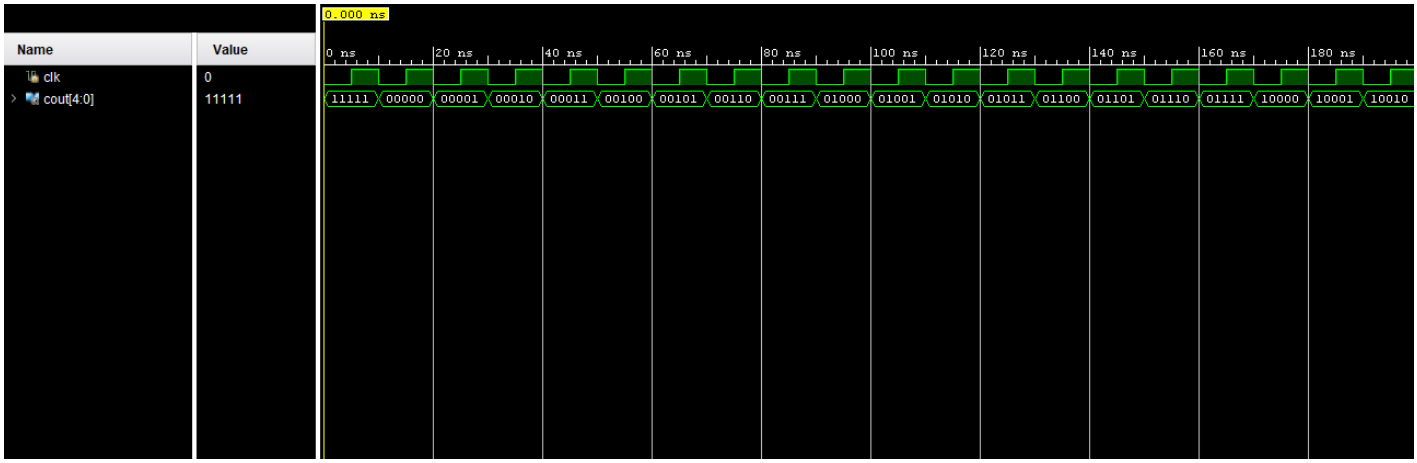


[2] Test Bench Waveform

```
1   `timescale 1ns / 1ps
2   module async(clk,cout);
3   output reg [4:0] cout;
4   input clk;
5   wire clk;
6
7   initial
8   cout=5'b0;
9   always @(negedge clk)
10  cout[0]<=~cout[0];
11  always @(negedge cout[0])
12  cout[1]<=~cout[1];
13  always @(negedge cout[1])
14  cout[2]<=~cout[2];
15  always @(negedge cout[2])
16  cout[3]<=~cout[3];
17  always @(negedge cout[3])
18  cout[4]<=~cout[4];
19  endmodule
20
```

[4] Test Bench Code

```
1   `timescale 1ns / 1ps
2   module test;
3   reg clk;
4   wire [4:0] cout;
5   async tc(clk,cout);
6   initial begin
7   clk=0;
8   #10;
9   end
10  always#5 clk=~clk;
11  endmodule
12
```



[3] Design Source Code

III. CONCLUSION

We build a UP counter which starts from 0 and ends at 31 counting up with the help of combinations of J K negative-edge Flip Flops.

REFERENCES

[1] Digital Design by Morris Mano