University of Engineering and Technology,
UET Taxila.

Electrical Engineering Department

Subject: DLD_L (Project)

Submitted to: Syed Waqas Shah

Submitted by: Muhammad Saad, Ali Hassan, M. Husnain, Zunaira Javaid

Reg. N0: 20-EE-103, 20-EE-87, 20-EE-95, 20-EE-63

Section: C2

Date:20/01/2022

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Objective:

Our target is to design a binary counter with the provide sequence. The sequence we are provided is (0, 1, 3, 5, 6, 8, 10, 11, 14, 15).

Tool:

ModelSim 10.4

THEORY:

FLIP FLOP:

Flip-flop is a circuit that maintains a state until directed by input to change the state. A basic flip-flop can be constructed using four-NAND or four-NOR gates.

TYPES OF FLIP FLOP:

Following are the types of flip flop: 1) RS Flip Flop
2) JK Flip Flop
3) D Flip Flop
4) T Flip Flop Here we will discuss about JK flip flop

JK FLIP FLOP:

The J-K flip-flop is the most versatile of the basic flip-flops. It has the input-following character of the clocked D flip-flop but has two inputs, traditionally labeled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. **JK flip flop is also call as universal flip flop** A JK flip-flop is also called a universal flip-flop because it can be configured to work as an SR flip-flop, D flip-flop or T flip-flop.

Why we need JK flip flop?

The basic S-R NAND flip-flop circuit has many advantages and uses in sequential logic circuits but it suffers from two basic switching problems.

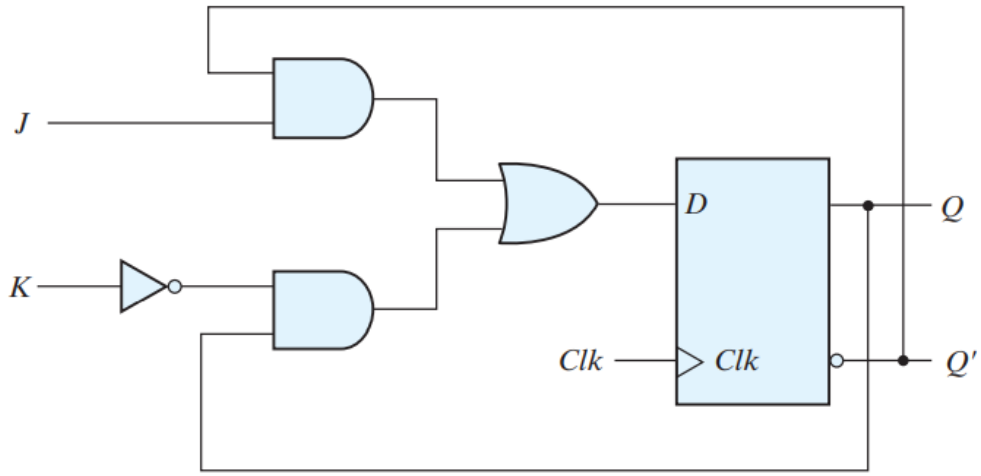
- • 1. the Set = 0 and Reset = 0 condition (S = R = 0) must always be avoided
- • 2. if Set or Reset change state while the enable (EN) input is high the correct latching action may not occur

Then to overcome these two fundamental design problems with the SR flip-flop design, the JK flip Flop was developed. 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 to distinguish the flip-flop design from other types. The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same "Set" and "Reset" inputs. The difference this time is that the "JK flip flop" has no invalid or forbidden input states of the SR Latch even when S and R are both at logic "1".

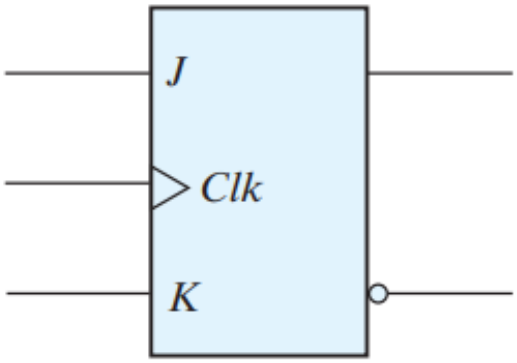
The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level "1". Due to this additional clocked input, a JK flip-flop has four possible input combinations, "logic 1", "logic 0", "no change" and "toggle". The symbol for a JK flip flop is similar to that of an SR bistable Latch as seen in the previous tutorial except for the addition of a clock input.

APPLICATION OF JK FLIP FLOP:

- • Registers. A single flip flop can store a 1 bit word. ...
- • Counters. Counter is a digital circuit used for a counting pulses or number of events and it is the widest application of flip-flops. …
- • Event Detectors. ...
- • Data Synchronizers. ...
- • Frequency Divider.



Circuit diagram of JK flip flop



Block Diagram of JK flip flop

TRUTH TABLE OF JK FLIP FLOP:

	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	

Code for Binary Counter with the provide sequence (0, 1, 3, 5, 6, 8, 10, 11, 14, 15):

```
module JK_flipflop (j,k,clock,reset,q, qbar);
input j,k,clock,reset;
output q, qbar;
```

```
reg q;
wire qbar;
assign qbar=~q;
always @ (posedge clock, negedge reset)
if (reset==0) q<=1'b0;
else
case ({j,k})
2'b00: q<=q;
2'b01: q<=1'b0;
2'b10: q<=1'b1;
2'b11: q<=~q;
endcase
endmodule
//main code using circuit diagram
module seq_count(Q3,Q2,Q1,Q0,clock,reset);
output Q3,Q2,Q1,Q0;
input clock,reset;
wire J3,K3,J2,K2,J1,K1,J0,K0,Q3,Q2,Q1,Q0;

//assign values to J3 and K3 according to kmap equations
assign J3=(Q1&~Q0);
assign K3=Q0&Q2;
JK_flipflop FFA (J3,K3,clock,reset,Q3);
```

```
//assign values to J2 and K2 according to kmap equations
assign J2=(~Q3&Q1)|(Q3&Q0);
assign K2=(~Q3&Q1)|(Q3&Q0);
JK_flipflop FFB (J2,K2,clock,reset,Q2);

//assign values to J1 and K1 according to kmap equations
assign J1=(Q0|Q3);
assign K1=(~Q3)|(Q2&Q0);
JK_flipflop FFC (J1,K1,clock,reset,Q1);

//assign values to J0 and K0 according to kmap equations
assign J0=(Q1&Q3)|(~Q3&~Q1);
assign K0=Q3|Q2;
JK_flipflop FF4 (J0,K0,clock,reset,Q0);
endmodule

//TESTBENCH
module zunaira;
reg clock,reset;
wire Q3,Q2,Q1,Q0;
seq_count ahu (Q3,Q2,Q1,Q0,clock,reset);
always
#5 clock=~clock;
```

```
initial
begin
reset=1'b0; clock=1'b0;
#2 reset=1'b1;
end

//Simulation Stop after 250nano seconds
initial #250 $finish;
endmodule
```