Experiment-05 Implementation of flipflops using verilog

**AIM**:

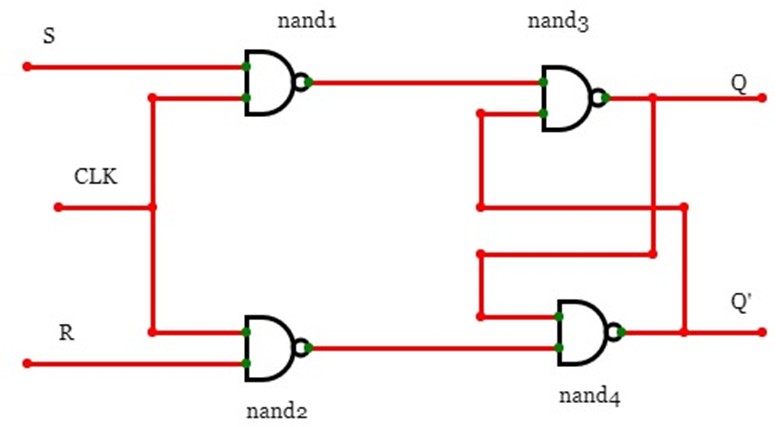
To implement all the flipflops using verilog and validating their functionality using their functional tables

**HARDWARE REQUIRED:** – PC, Cyclone II , USB flasher **SOFTWARE REQUIRED:** Quartus prime

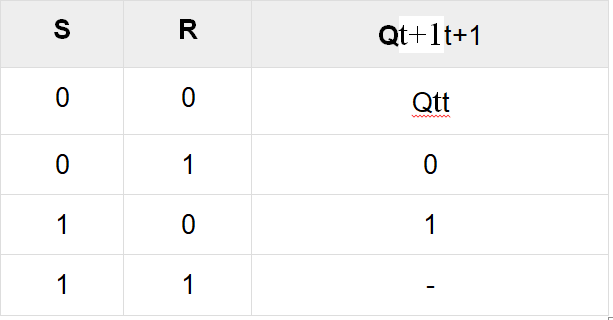
**THEORY:**

## SR Flip-Flop

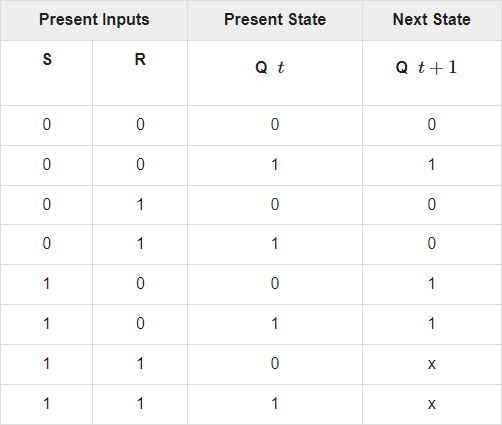
SR flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, SR latch operates with enable signal. The circuit diagram of SR flip-flop is shown in the following figure.

3

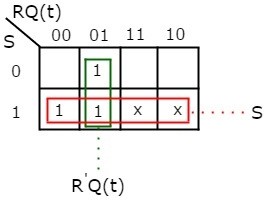
This circuit has two inputs S & R and two outputs Qtt & Qtt’. The operation of SR flipflop is similar to SR Latch. But, this flip-flop affects the outputs only when positive transition of the clock signal is applied instead of active enable. The following table shows the state table of **SR flip-flop.**



Here, Qtt & Qt+1t+1 are present state & next state respectively. So, SR flip-flop can be used for one of these three functions such as Hold, Reset & Set based on the input conditions, when positive transition of clock signal is applied. The following table shows the characteristic table of SR flip-flop. Present Inputs Present State Next State



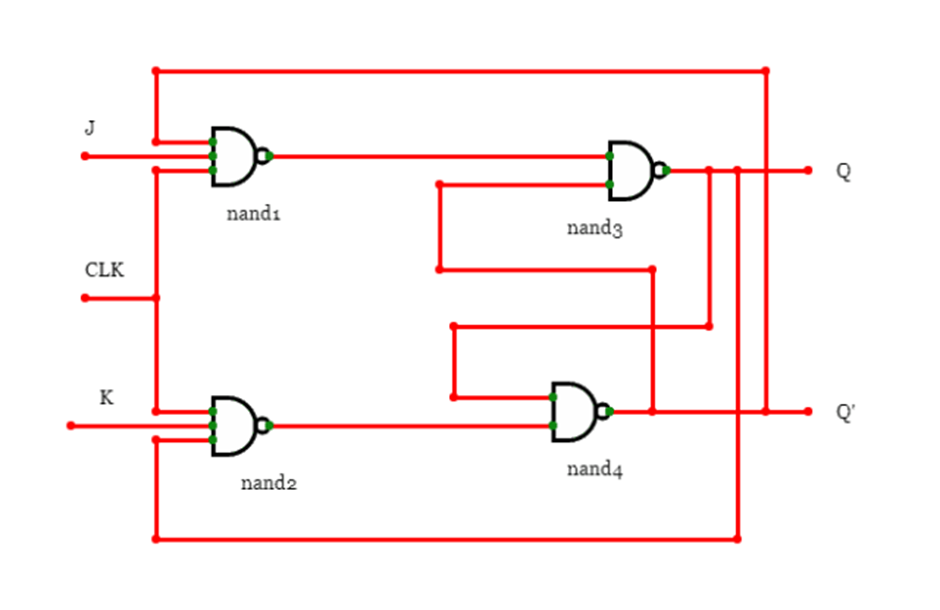
By using three variable K-Map, we can get the simplified expression for next state, Qt+1t+1. The three variable K- Map for next state, Qt+1t+1 is shown in the following figure.



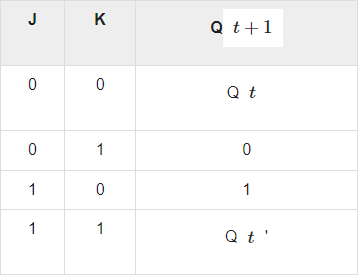
The maximum possible groupings of adjacent ones are already shown in the figure. Therefore, the simplified expression for next state Qt+1t+1 is Q(t+1)=S+R′Q(t)Q(t+1)=S+R′Q(t)

## JK Flip-Flop

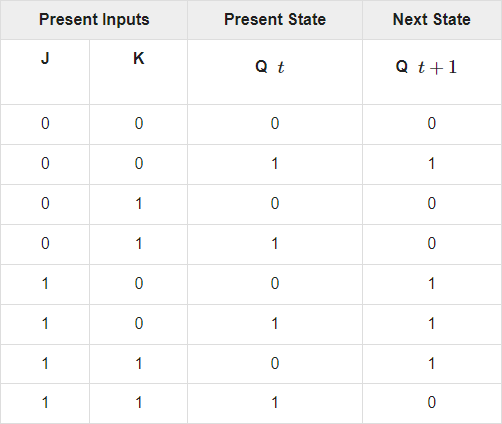
JK flip-flop is the modified version of SR flip-flop. It operates with only positive clock transitions or negative clock transitions. The circuit diagram of JK flip-flop is shown in the following figure.



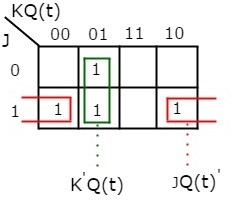
This circuit has two inputs J & K and two outputs Qtt & Qtt’. The operation of JK flip-flop is similar to SR flip-flop. Here, we considered the inputs of SR flip-flop as S = J Qtt’ and R = KQtt in order to utilize the modified SR flip-flop for 4 combinations of inputs. The following table shows the state table of JK flip-flop.



Here, Qtt & Qt+1t+1 are present state & next state respectively. So, JK flip-flop can be used for one of these four functions such as Hold, Reset, Set & Complement of present state based on the input conditions, when positive transition of clock signal is applied. The following table shows the characteristic table of JK flip-flop. Present Inputs Present State Next State



By using three variable K-Map, we can get the simplified expression for next state, Qt+1t+1. Three variable K-Map for next state, Qt+1t+1 is shown in the following figure.



The maximum possible groupings of adjacent ones are already shown in the figure. Therefore, the simplified expression for next state Qt+1t+1 is Q(t+1)=JQ(t)′+K′Q(t)Q(t+1)=JQ(t)′+K′Q(t)

# Procedure

1.Create a New Project:

Open Quartus and create a new project by selecting "File" > "New Project Wizard."

Follow the wizard's instructions to set up your project, including specifying the project name, location, and target device (FPGA).

2.Create a New Design File:

Once the project is created, right-click on the project name in the Project Navigator and select "Add New File."

Choose "Verilog HDL File" or "VHDL File," depending on your chosen hardware description language.

3.Write the Combinational Logic Code:

Open the newly created Verilog or VHDL file and write the code for your combinational logic.

4.Compile the Project:

To compile the project, click on "Processing" > "Start Compilation" in the menu.

Quartus will analyze your code, synthesize it into a netlist, and perform optimizations based on your target FPGA device.

5.Analyze and Fix Errors:

If there are any errors or warnings during the compilation process, Quartus will display them in the Messages window.

Review and fix any issues in your code if necessary. View the RTL diagram.

6.Verification:

Click on "File" > "New" > "Verification/Debugging Files" > "University Program VWF".

Once Waveform is created Right Click on the Input/Output Panel > " Insert Node or Bus" > Click on Node Finder > Click On "List" > Select All.

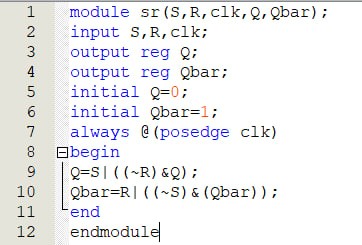
Give the Input Combinations according to the Truth Table and then simulate the

Output Waveform

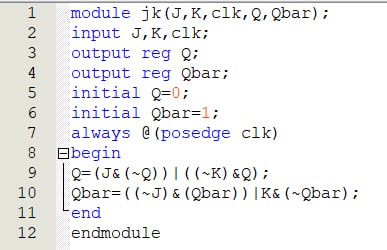
# PROGRAM:

Program for flipflops and verify its truth table in quartus using Verilog programming.

## SR-Flip Flop

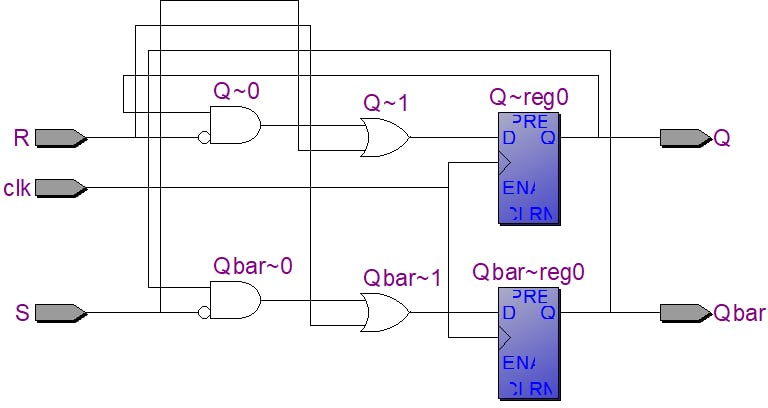


**JK Flip-Flop**

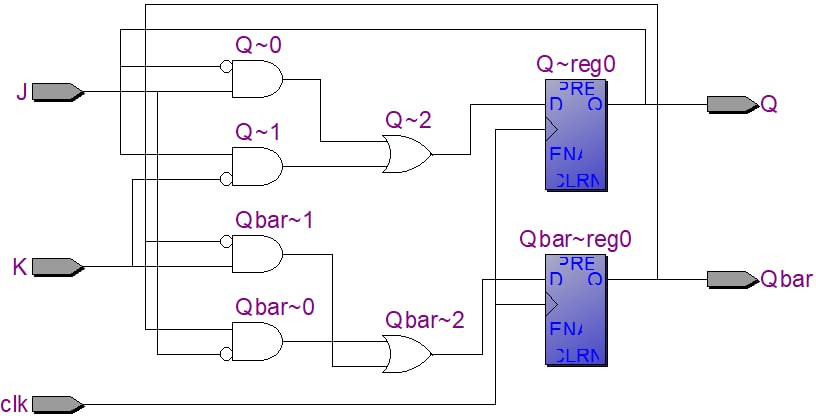


**RTL LOGIC FOR FLIPFLOPS:**

**SR Flip-Flop**

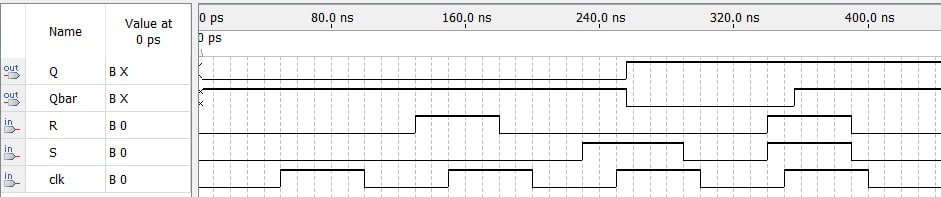


**JK Flip-Flop**

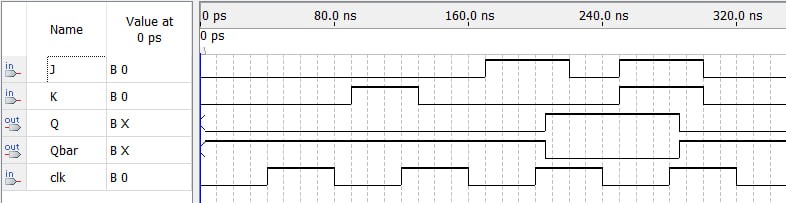


**TIMING DIGRAMS FOR FLIP FLOPS**

**SR Flip-Flop**



**JK Flip-Flop**



**RESULTS**

By this we have verified the truth table of JK and SR using verilog.