Experiment -9

Title: Implementation of a SR Flip-Flop.

Objective: To implement a SR Flip-Flop in VHDL using Behavioral modelling.

Theory:

Flip-Flop:

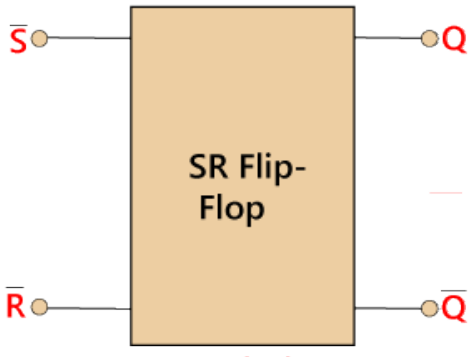
A flip-flop is a device which stores a single *bit* (binary digit) of data; one of its two states represents a "one" and the other represents a "zero". Such data storage can be used for storage of state, and such a circuit is described as sequential logic in electronics. When used in a finite-state machine, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronising variably-timed input signals to some reference timing signal.

SR Flip-Flop:

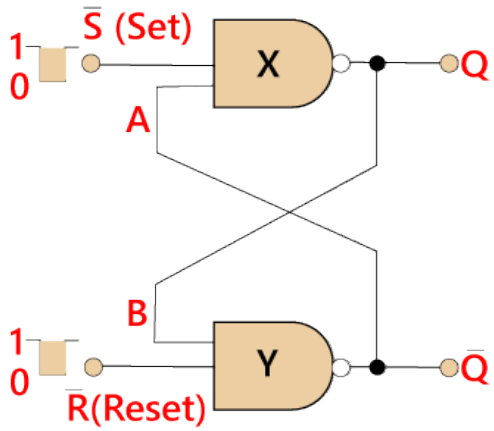
The SR flip flop is a 1-bit memory bistable device having two inputs, i.e., SET and RESET. The SET input 'S' sets the device or produces the output 1, and the RESET input 'R' reset the device or produces the output 0. The SET and RESET inputs are labelled as **S** and **R**, respectively.

The SR flip flop stands for "Set-Reset" flip flop. The reset input is used to get back the flip flop to its original state from the current state with an output 'Q'. This output depends on the set and reset conditions, which is either at the logic level "0" or "1".

Block Diagram:



Circuit Diagram:



Truth Table:

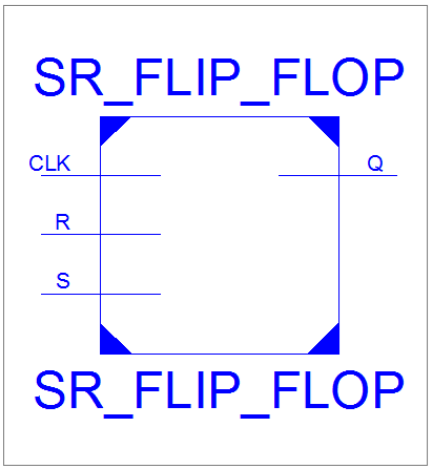
S	R	Q	Q’
0	0	0	1
0	1	0	1
1	0	1	0
1	1	∞	∞

Code:

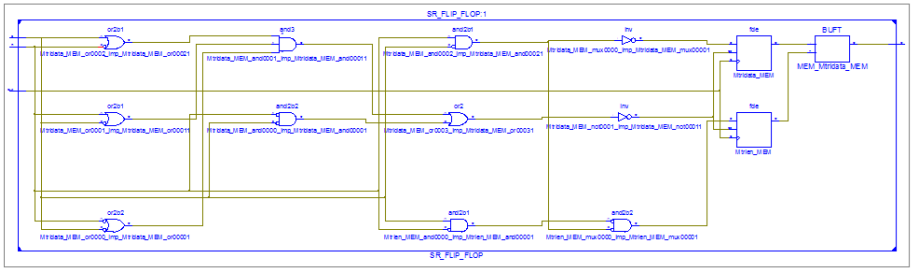
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity SR_FLIP_FLOP is
    Port ( S : in  STD_LOGIC;
           R : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           Q : out  STD_LOGIC);
end SR_FLIP_FLOP;
```

```
architecture Behavioral of SR_FLIP_FLOP is

begin
process(CLK, S, R) is
        variable MEM: STD_LOGIC;
        begin
        if(CLK = '1' and CLK'EVENT) then
                if(S = '0' AND R = '0') Then
                        MEM := MEM;
                elsif(S = '0' AND R = '1') Then
                        MEM := '0';
                elsif(S = '1' AND R = '0') Then
                        MEM := '1';
                elsif(S = '1' AND R = '1') Then
                        MEM := 'Z';
                end if;
        end if;
        Q <= MEM;
        end process;

end Behavioral;
```

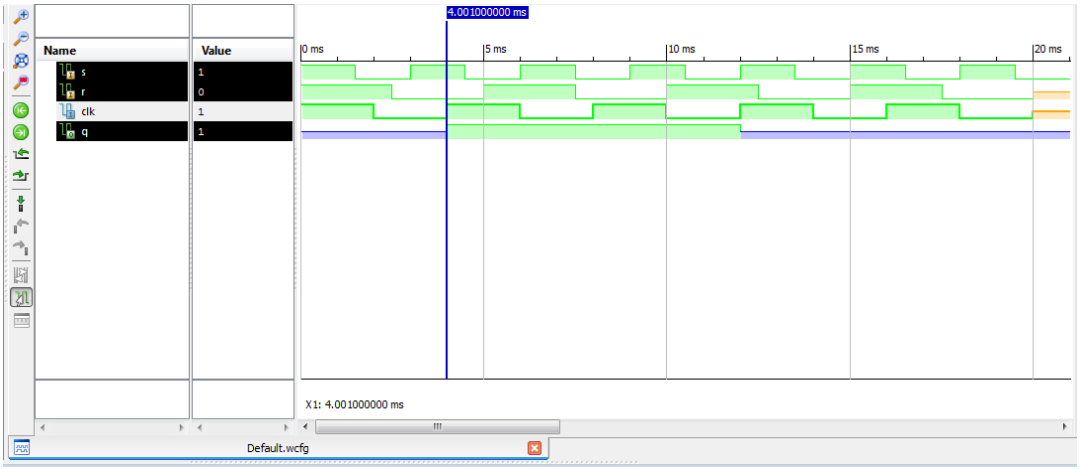
Entity Diagram:



RTL Schematic:



Output:



Result:

We have implemented a SR Flip-Flop in VHDL using Behavioral modelling, and the Entity Diagram, the RTL schematic and the outputs are shown above.

Conclusion:

In this experiment, we learnt about behavioral modelling and clock events in VHDL and used it to implement a SR Flip-Flop

Experiment-10

Title: Implementation of a D Flip-Flop.

Objective: To implement a D Flip-Flop in VHDL using Behavioral modelling.

Theory:

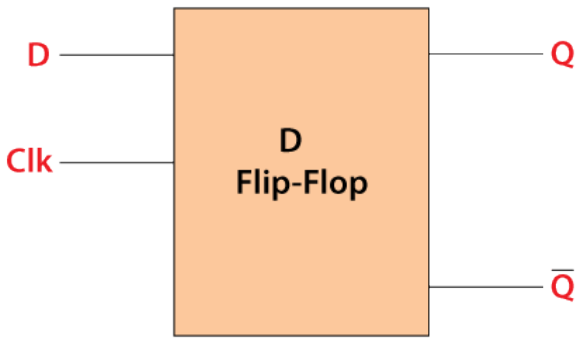
Flip-Flop:

A flip-flop is a device which stores a single bit (binary digit) of data; one of its two states represents a "one" and the other represents a "zero". Such data storage can be used for storage of state, and such a circuit is described as sequential logic in electronics. When used in a finite-state machine, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronising variably-timed input signals to some reference timing signal.

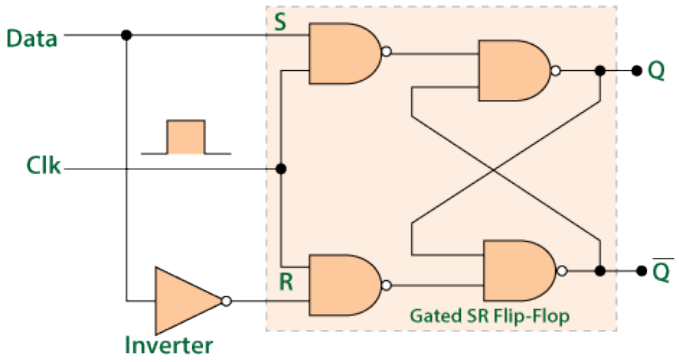
D Flip-Flop:

The D flip flop is **the most important flip flop from other clocked types**. It ensures that at the same time, both the inputs, i.e., S and R, are never equal to 1. The Delay flip-flop is designed using a gated SR flip-flop with an inverter connected between the inputs allowing for a single input D(Data).

Block Diagram:



Circuit Diagram:



Truth Table:

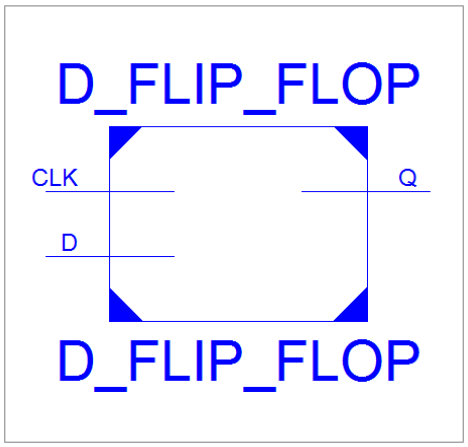
Clock	D	Q	Q'
↓ » 0	0	0	1
↑ » 1	0	0	1
↓ » 0	1	0	1
↑ » 1	1	1	0

Code:

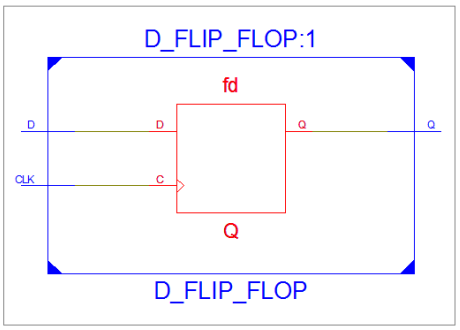
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity D_FLIP_FLOP is

    Port ( D, CLK : in  STD_LOGIC;

           Q : out  STD_LOGIC);
```

```
end D_FLIP_FLOP;

architecture Behavioral of D_FLIP_FLOP is

begin
Process(CLK)
begin
if(CLK = '1' and CLK'EVENT)
then
Q <= D;
end if;
end process;
end Behavioral;
```

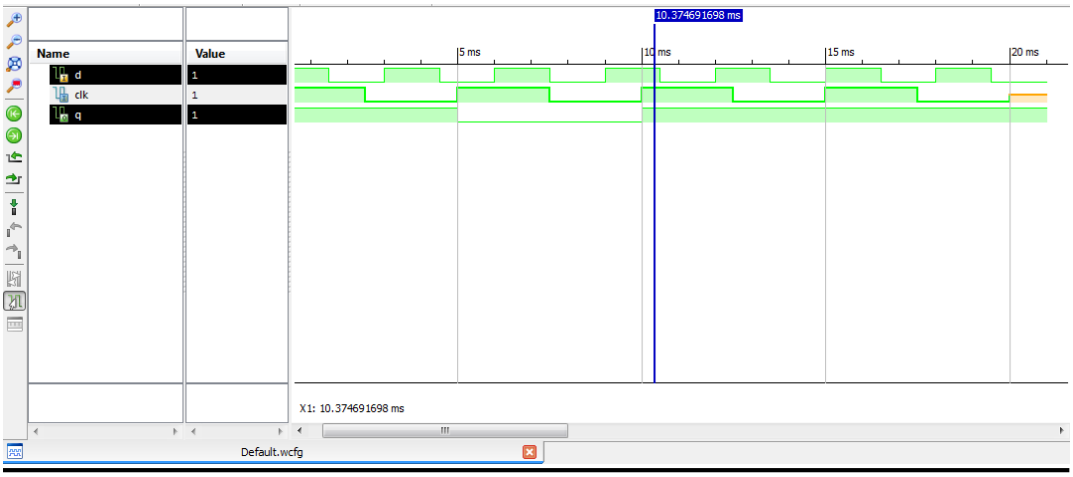
Entity Diagram:



RTL Schematic:



Output:



Result:

We have implemented a D Flip-Flop in VHDL using Behavioral modelling, and the Entity Diagram, the RTL schematic and the outputs are shown above.

Conclusion:

In this experiment, we learnt about behavioral modelling and clock events in VHDL and used it to implement a D Flip-Flop.

Experiment -11

Title: Implementation of a JK Flip-Flop.

Objective: To implement a JK Flip-Flop in VHDL using Behavioral modelling.

Theory:

Flip-Flop:

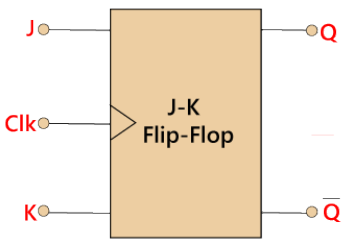
A flip-flop is a device which stores a single bit (binary digit) of data; one of its two states represents a "one" and the other represents a "zero". Such data storage can be used for storage of state, and such a circuit is described as sequential logic in electronics. When used in a finite-state machine, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronising variably-timed input signals to some reference timing signal.

JK Flip-Flop:

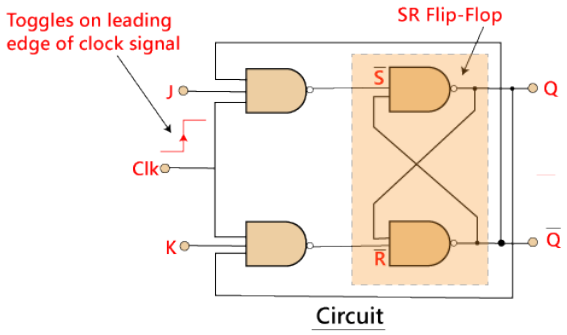
The JK flip flop is one of the most used flip flops in digital circuits. The JK flip flop is a universal flip flop having two inputs 'J' and 'K'. In SR flip flop, the 'S' and 'R' are the shortened abbreviated letters for Set and Reset, but J and K are not. The J and K are themselves autonomous letters which are chosen to distinguish the flip flop design from other types.

The JK flip flop works in the same way as the SR flip flop works. The JK flip flop has 'J' and 'K' flip flop instead of 'S' and 'R'. The only difference between JK flip flop and SR flip flop is that when both inputs of SR flip flop is set to 1, the circuit produces the invalid states as outputs, but in case of JK flip flop, there are no invalid states even if both 'J' and 'K' flip flops are set to 1.

Block Diagram:



Circuit Diagram:



Truth Table:

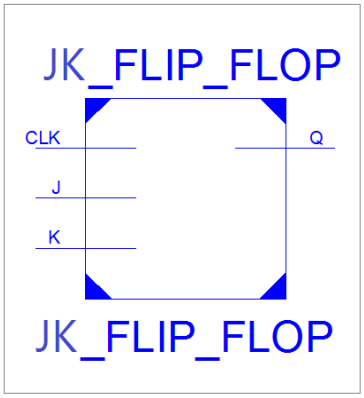
J	K	Q	Q'
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	0

Code:

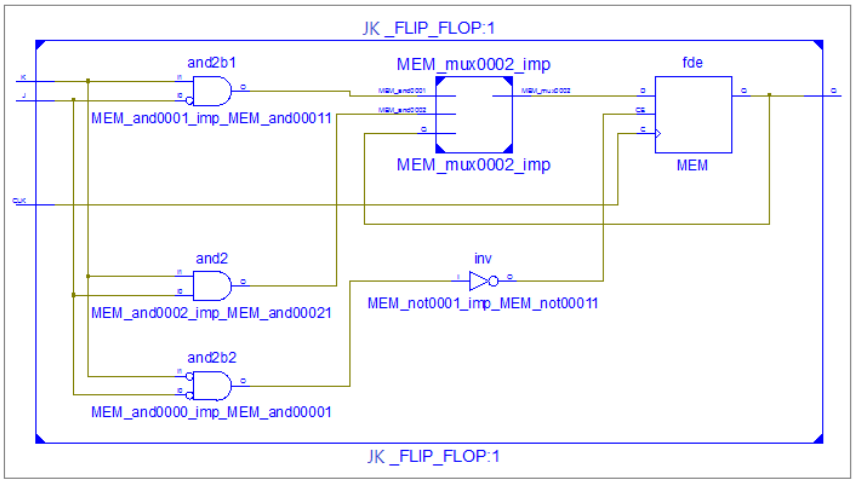
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity JK_FLIP_FLOP is
```

```
    Port ( J : in  STD_LOGIC;
           K : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           Q : out  STD_LOGIC);
end JK_FLIP_FLOP;

architecture Behavioral of JK_FLIP_FLOP is

begin
process(CLK, J, K) is
           variable MEM: STD_LOGIC;
           begin
           if(CLK = '1' and CLK'EVENT) then
                    if(J = '0' AND K = '0') Then
                             MEM := MEM;
                    elsif(J = '0' AND K = '1') Then
                             MEM := '0';
                    elsif(J = '1' AND K = '1') Then
                             MEM := NOT MEM;
                    else
                             MEM := '1';
                    end if;
           end if;
           Q <= MEM;
           end process;

end Behavioral;
```

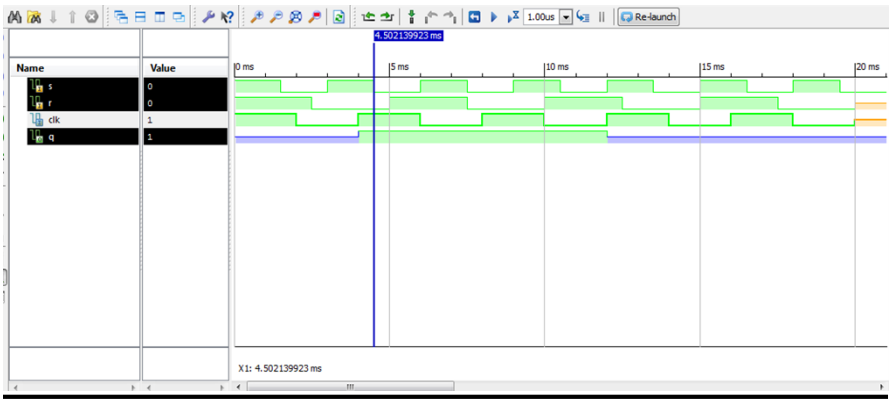
Entity Diagram:



RTL Schematic:



Output:



Result:

We have implemented a JK Flip-Flop in VHDL using Behavioral modelling, and the Entity Diagram, the RTL schematic and the outputs are shown above.

Conclusion:

In this experiment, we learnt about behavioral modelling and clock events in VHDL and used it to implement a JK Flip-Flop.