INSTITUTE OF ENGINEERING AND TECHNOLOGY, INDORE

ELECTRONICS & INSTRUMENTATION ENGINEERING

EIR7E1 : Circuit Design Using HDL

Design Negative Edge DFF

SUBMITTED TO

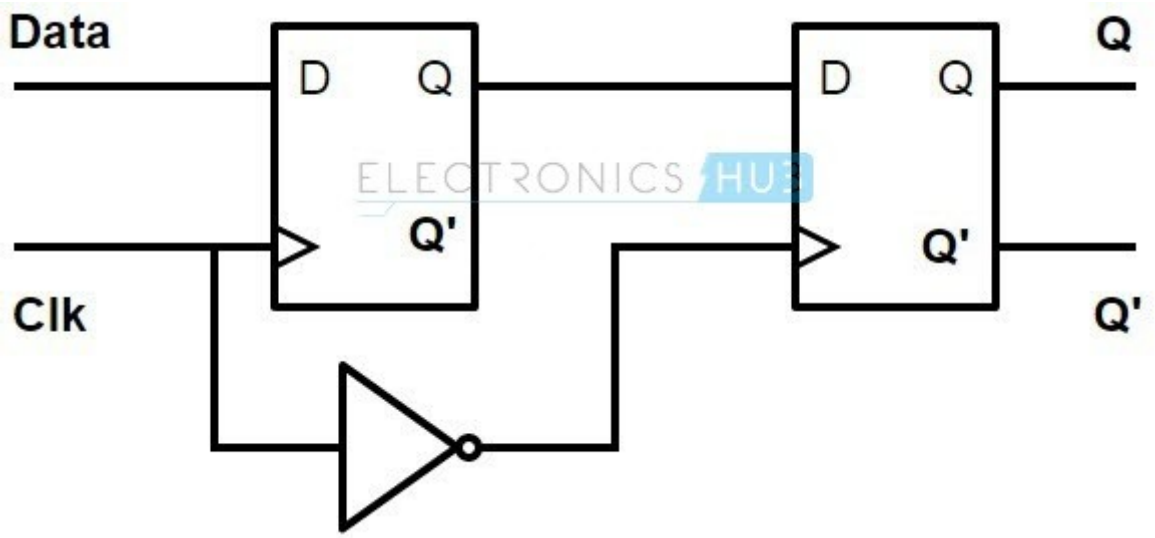
Dr. Vaibhav Neema

SUBMITTED BY

Sanidhya Joshi
17E7040

Negative Edge D Flip Flop:

Master slave D flip flop can be designed by the series connection of two gated D latches and connecting an inverted enable input either to of the two latches. Only the change in Master latch will bring change in Slave latch. So these are called Master Slave flip flops. The total circuit of master slave flip flop is triggered either on the rising edge of the clock signal or on falling edge of clock signal depending on the design.
The symbolic representation of a master slave D flip flop that responds to the clock at its falling edge as shown below.



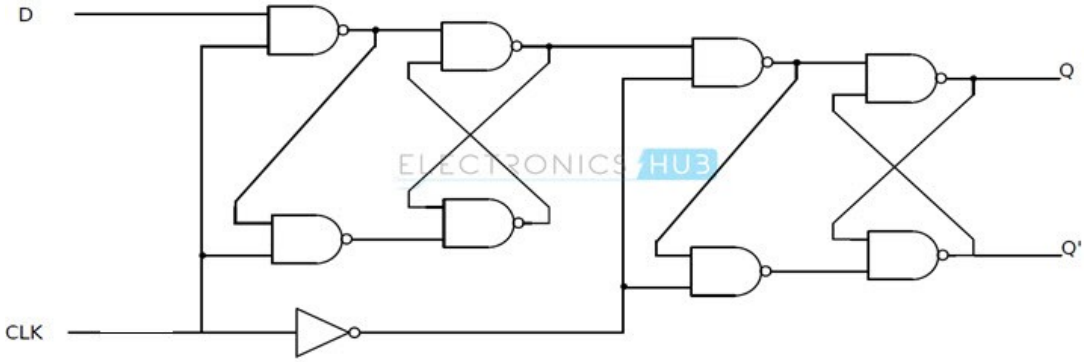
Operation

The operation of positive edge triggered Master Slave D flip flop is explained below.

•If clock is low, the enable signal to master flip flop is high. When clock signal changes from low to high, the master flip flop stores the data from the D input. Simultaneously at the second flip flop , the enable signal goes low to high along with clock signal because of the double inversion. The data locked by the master flip flop during the rising edge are passed to the slave flip flop.

•When clock signal goes high to low, the slave flipflop will receive the master flip flop output as its input and changes its state. Master flipflop will accept latest values from the inputs on next rising edge.

A simple modification will turn the above device in to negative edge triggering device. A negative edge triggered master slave D flip flop is formed by eliminating first inverter along the clock signal path.



DESIGN CODE:

```
Library ieee;
use ieee.std_logic_1164.all;

Entity ip2_nand is
Port ( a,b:in std_logic;
       op:out std_logic);
End entity;

Architecture da of ip2_nand is
begin
      Op<=a nand b;
End architecture;

Library ieee;
use ieee.std_logic_1164.all;

entity d_latch is
Port(clk,d:in std_logic;
         q,qbar:out std_logic);
End entity;

Architecture rch of d_latch is
signal p1,p2,p3,q1,q2:std_logic;

component ip2_nand is
Port ( a,b:in std_logic;
       op:out std_logic);
End component;
Begin
U1: ip2_nand port map (clk,d,p1);
U2: ip2_nand port map (d,d,p3);
U3: ip2_nand port map (clk,p3,p2);
U4: ip2_nand port map (p1,q2,q1);
U5: ip2_nand port map (q1,p2,q2);
q<=q1;
qbar<=q2;
end rch;
```

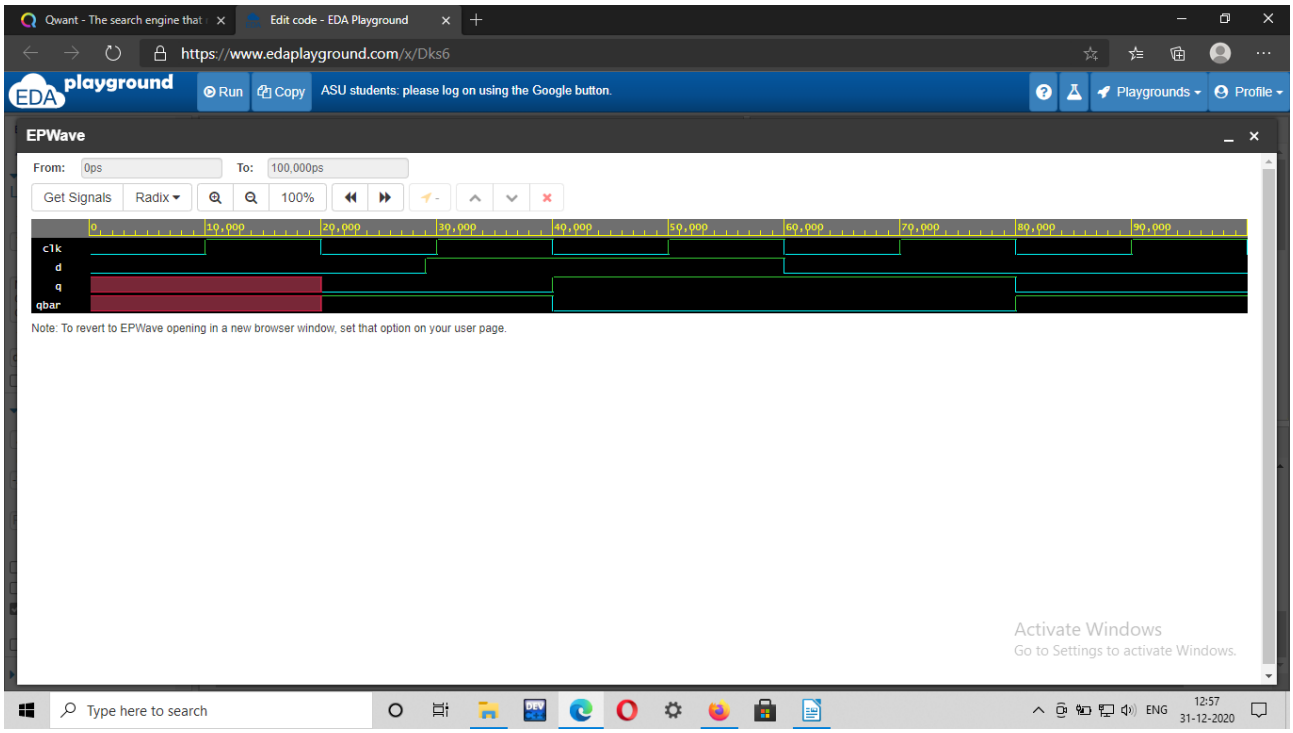
```
Library ieee;
use ieee.std_logic_1164.all;

entity my_dff is
Port (clk,d:in std_logic;
    q,qbar:out std_logic);
End entity;

Architecture rch of my_dff is
Signal q1,q2,q3:std_logic;

component ip2_nand is
Port ( a,b:in std_logic;
       op:out std_logic);
End component;

component d_latch is
Port(clk,d:in std_logic;
        q,qbar:out std_logic);
End component;

Begin
master   : d_latch port map (clk,d,q1,q2);
nand4not : ip2_nand port map (clk,clk,q3);
slave    : d_latch port map (q3,q1,q,qbar);
End rch;
```

TESTBENCH CODE:

```
library ieee;
use ieee.std_logic_1164.all;

entity dff_tb is
end entity;

architecture tb of dff_tb is

component my_dff is
Port (clk,d : in std_logic;
    q, qbar : out std_logic);
end component ;

signal d, clk,q, qbar : std_logic;

begin
dut: my_dff port map (clk,d,q,qbar);
Clock : process
begin
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end process;

stim : process
begin

d <= '0';
wait for 29 ns;
d <= '1';
wait for 31 ns;
d <= '0';
wait for 42 ns;
d <= '1';
wait for 23 ns;
```

```
end process;
end tb;
```

WAVEFORM:



You can visit this website to view my code and also run the simulation:

[Edit code - EDA Playground]