DIGITAL SYSTEM DESIGN APPLICATION – EHB 436E

Experiment V

Yiğit Bektaş GÜRSOY

040180063

Class Lecturer: Sıddıka Berna Örs Yalçın

Class Assistant:
Serdar Duran
Yasin Fırat Kula
Mehmet Onur Demirtürk

1. SR LATCH WITH NAND GATE

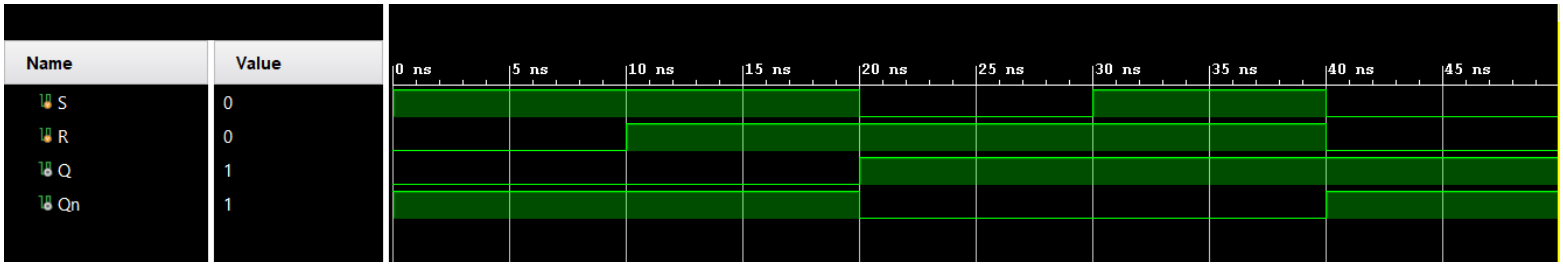
- Verilog Code

```
module SR_LATCH(
       input S,
       input R,
       output Q,
       output Qn
        );

NAND_gate nand1(
          .I1(S),
          .I2(Qn),
          .O(Q)
          );

NAND_gate nand2(
          .I1(R),
          .I2(Q),
          .O(Qn)
          );

endmodule
```

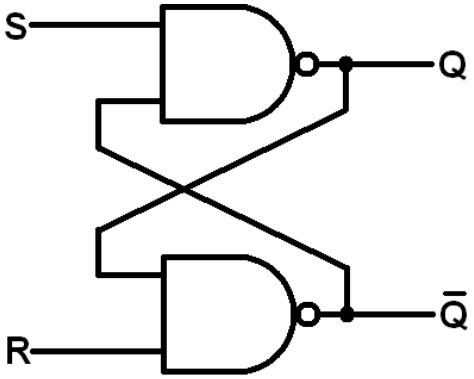
- Testbench Code

```
module SR_LATCH_tb();
    reg S;
    reg R;
    wire Q;
    wire Qn;

    SR_LATCH DUT(.S(S),
                 .R(R),
                 .Q(Q),
                 .Qn(Qn)
                 );
    initial
        begin
            S = 1'b0; R = 1'b0;
            #10 S = 1'b0; R = 1'b1;
            #10 S = 1'b1; R = 1'b0;
            #10 S = 1'b1; R = 1'b1;
            #10 $finish;
        end
endmodule
```

- Behavioral Simulation
  - In cases where S and R values are 1, the previous state is expected to be preserved. For S=1 and R=0, Q=0 and Qn=1 values are expected, for S=0 and R=1 values, Q=1 and Qn=1 values are expected. The simulation results below meet the expectations



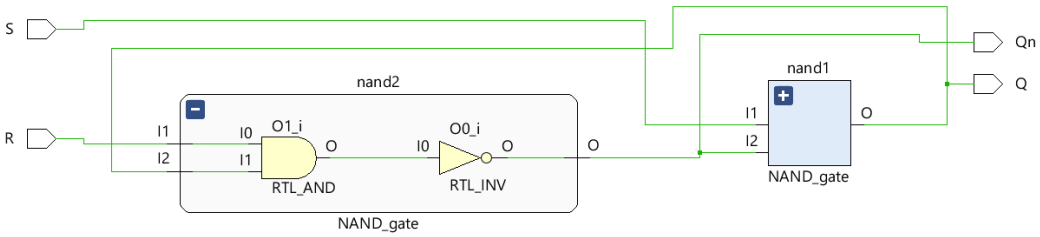
- Circuit Diagram



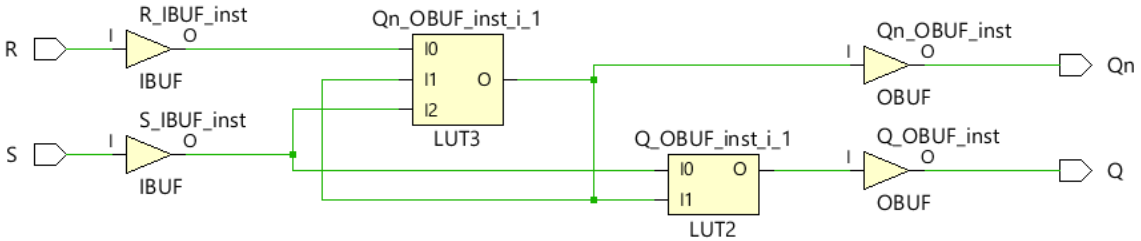
- Truth Table

S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

- Characteristic and inverse characteristic functions
  - $Q(t+1) = R' + S * Q(t)$
  - $Qn(t+1) = R * (S' + Qn'(t))$
- RTL Schematic



- Technology Schematic
  - There are 1 LUT2 and 1 LUT3 in technology schematic.



2. D FLIP-FLOP

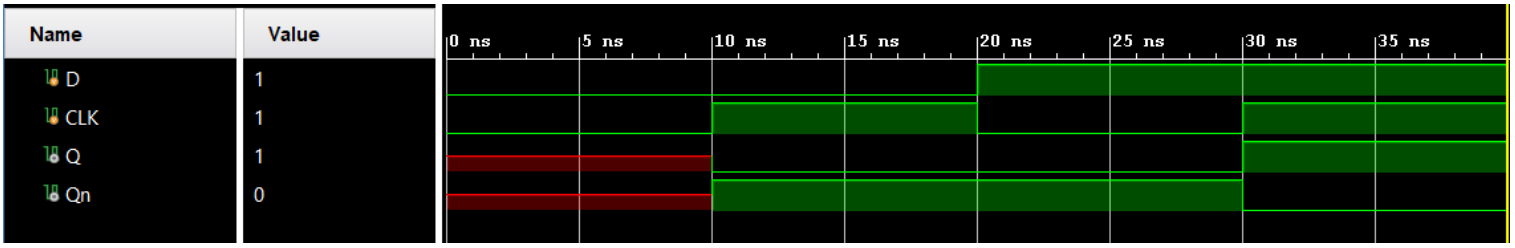
- Verilog Code

```
module D_FLIP_FLOP(
       input CLK,
       input D,

       output Q,
       output Qn
        );

wire [1:0]signal;

NAND_gate nand1(
          .I1(D),
          .I2(CLK),

          .O(signal[0])
          );

NAND_gate nand2(
          .I1(~D),
          .I2(CLK),

          .O(signal[1])
          );

NAND_gate nand3(
          .I1(signal[0]),
          .I2(Qn),

          .O(Q)
          );
NAND_gate nand4(
          .I1(signal[1]),
          .I2(Q),

          .O(Qn)
          );

endmodule
```

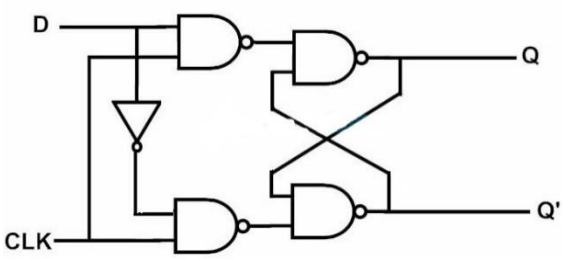
- Testbench Code

```
module D_FLIP_FLOP_tb();

    reg D;
    reg CLK;

    wire Q;
    wire Qn;

    D_FLIP_FLOP DUT(
                .D(D),
                .CLK(CLK),
                .Q(Q),
                .Qn(Qn)
                );
    initial
        begin
        D = 1'b0; CLK = 1'b0;
        #10 D = 1'b0; CLK = 1'b1;
        #10 D = 1'b1; CLK = 1'b0;
        #10 D = 1'b1; CLK = 1'b1;
        #10 $finish;
        end
 endmodule
```

- Behavioral Simulation
  - The simulation below is a simulation of D Flip-Flop. It transmits the value of the data to output Q whenever it is CLK=1. For example, in the case of D=1, let the CLK value be 1. As a result of this process, it is seen that the value of Q is 1. In transitions from 1 to 0 (negedge), the D value remains as it is.



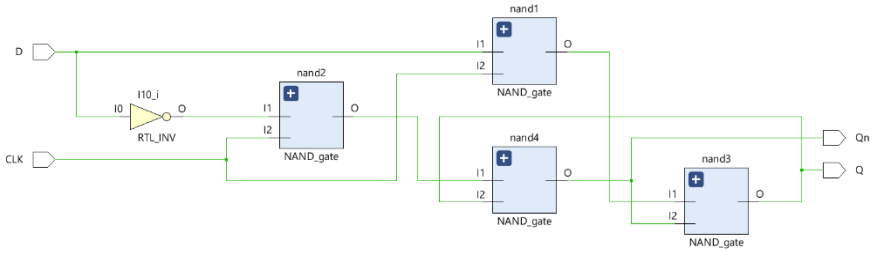
- D Flip-Flop Circuit



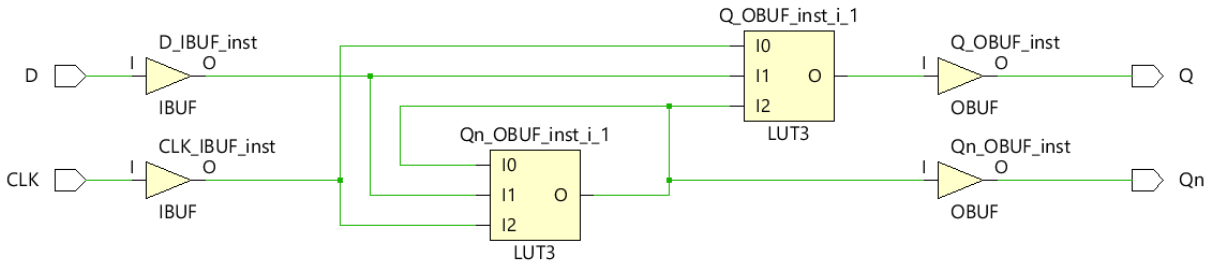
- Truth table

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

- RTL Schematic



- Technology Schematic
  - There are 2 LUT3 in technology schematic.



- Characteristic and inverse characteristic functions
  - $Q(t+1) = CLK * D + CLK' * Q(t)$
  - $Qn(t+1) = (CLK' + D') * (CLK + Qn'(t))$

- Combinational Delay
  - The maximum delay in my circuit is 7.200ns and max delay path is CLK to Q. Using this information, we can reach the frequency by making the maximum clock frequency 1/7.2ns. 1/7.2ns=138.888Mhz

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
CLK	Q	7.200	SLOW	2.186	FAST
D	Q	6.847	SLOW	2.065	FAST
CLK	Qn	6.624	SLOW	2.209	FAST
D	Qn	6.271	SLOW	2.101	FAST

- Constraint file
  - I fixed the error by adding the following command to the constraint file while creating the .bit file.

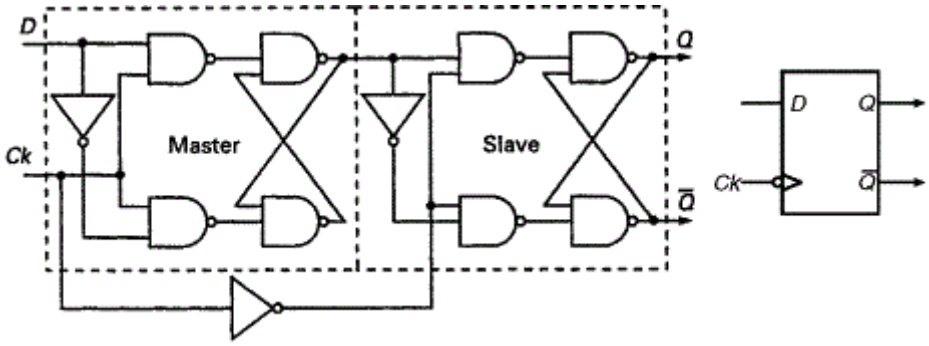
```
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets Qn_OBUF]
```

3. MASTER-SLAVE D FLIP-FLOP

- Verilog Code

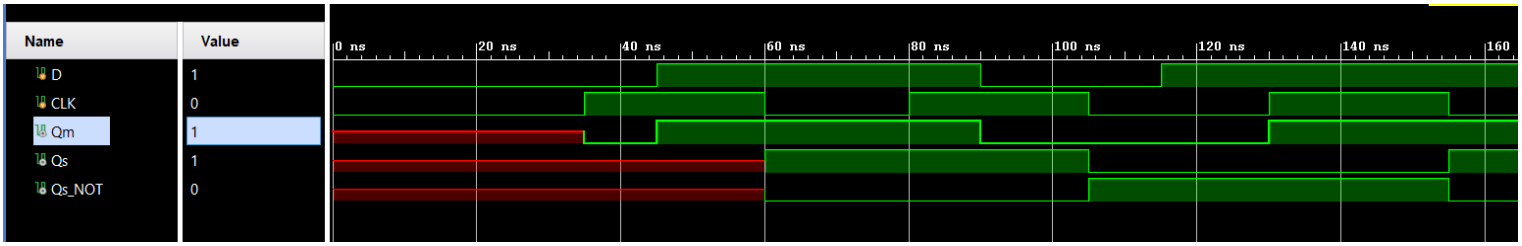
```
module MASTER_SLAVE_DFF(
        input D,
        input CLK,

        output Qs,
        output Qs_NOT
        );

wire Qm,Qm_not;

        D_FLIP_FLOP master(
                        .D(D),
                        .CLK(CLK),

                        .Q(Qm),
                        .Qn(Qm_not)
                        );

        D_FLIP_FLOP slave(
                        .D(Qm),
                        .CLK(~CLK),

                        .Q(Qs),
                        .Qn(Qs_NOT)
                        );

endmodule
```

- Circuit Diagram



- Behavioral Simulation and Explanation of Master-Slave D Flip-Flop

  - This structure contains two D flip-flops connected in series with each other. The first flip-flop is called the master, and the second flip-flop is called the slave flip-flop.
  - Regardless of the change in the CLK value for Qm, it takes whatever value is in the D input (Qm=D). Every time our CLK value for Qs goes from 1 to 0, when it receives a negedge signal, it moves the current D value to Qs. In addition, the CLK posedge signal is needed for the signal of Qm to activate. The negedge signal is needed for the Qs to be active.
  - Our simulation result below gave the expected outputs.



- Which edge of the clock signal effects the output?
  - ➢ The negative edge of the clock signal affects the output. It drives the D value to the output on each negative edge.

- Time diagram

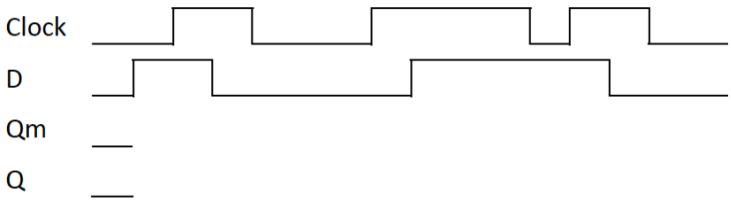
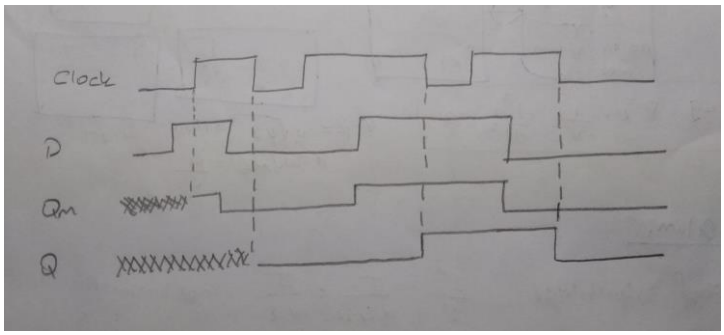


Fig.4: Time diagram for Master-Slave Flip-Flop.

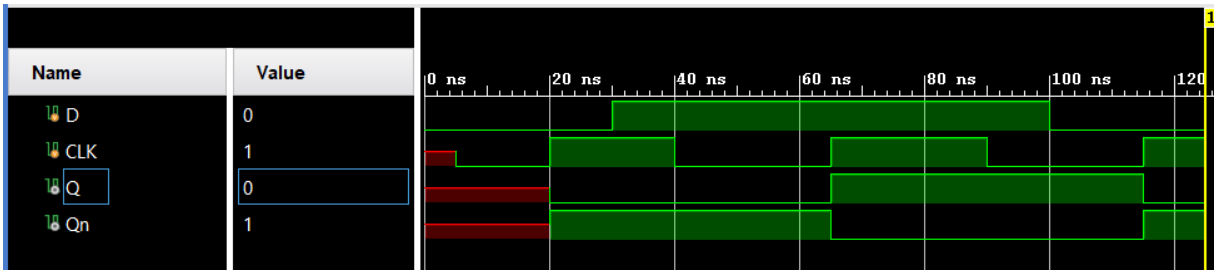


4. D FLIP-FLOP BEHAVIORAL DESIGN

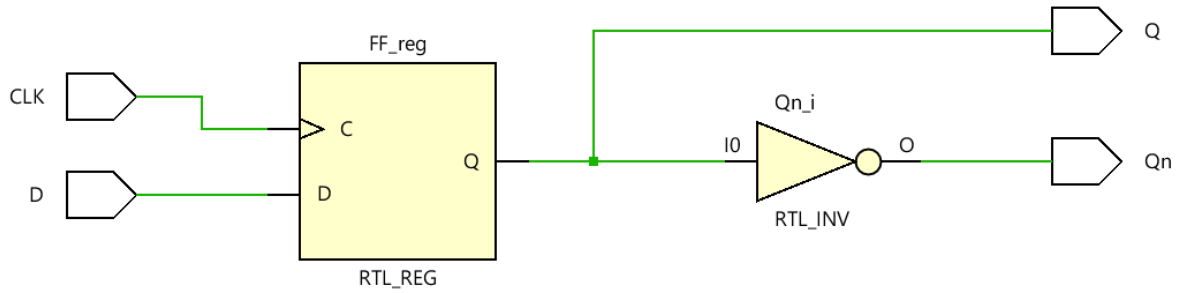
- Verilog Code

```
module BEHAVIORAL_DFF(
        input D,
        input CLK,

        output Q,
        output Qn
        );

    reg FF;
    always @ (posedge CLK)
    begin
        FF <= D;
    end

    assign Q = FF;
    assign Qn = ~FF;

endmodule
```

- Behavioral Simulation

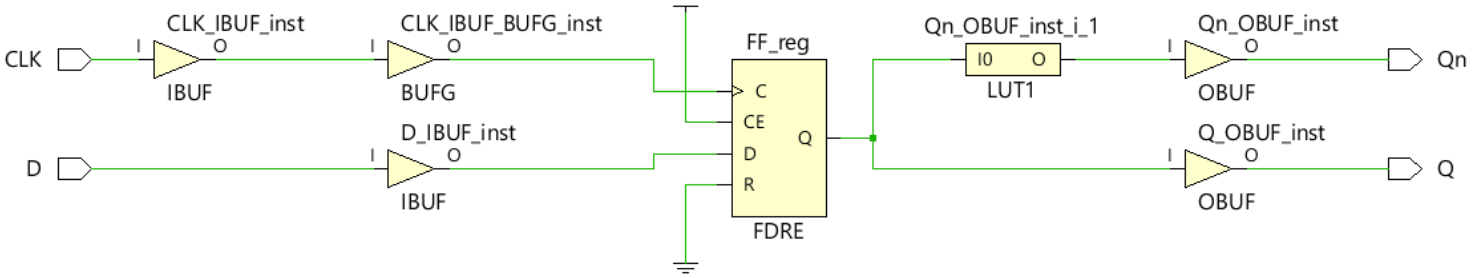


- RTL Schematic



- Technology Schematic
  - There are 1 LUT and 1 FDRE structure in technology schematic.

  - FDRE: This design element is a single D-type flip-flop with clock enable and synchronous reset.
  - When clock enable (CE) is High and synchronous reset (R) is not asserted, the data on the data input (D) of this design element is transferred to the corresponding data output (Q) during the clock (C) transition.
  - When R is active, it overrides all other inputs and resets the data output (Q)Low upon the next clock transition.
  - When CE is Low, clock transitions are ignored.
  - This flip-flop is asynchronously initialized when power is applied. When global set/reset (GSR) is active upon power-up or when GSR is asserted via the STARTUP block, the value of the INIT attribute is placed on the register's output.

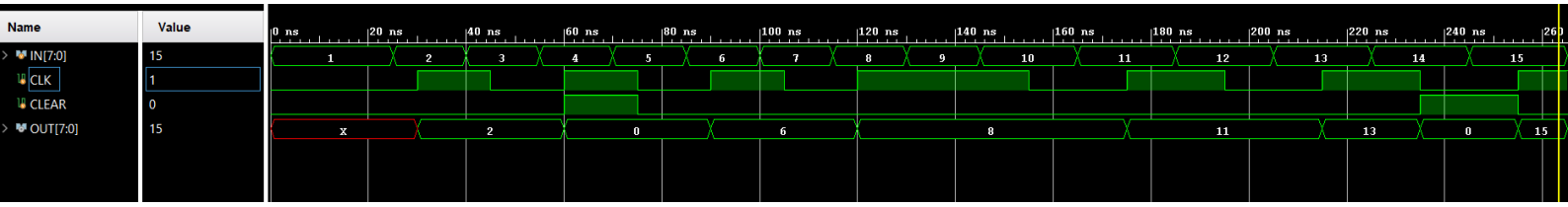


5. 8-BIT REGISTER

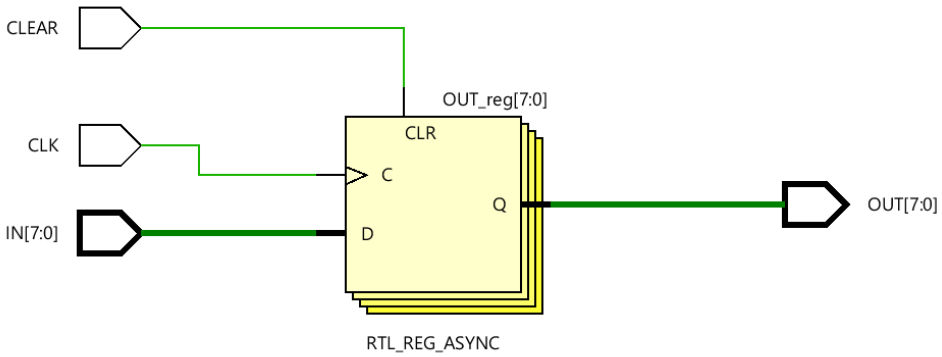
- Verilog Code

```
module REGISTER(
        input [7:0]IN,
        input CLK,
        input CLEAR,

        output reg [7:0]OUT
         );

    always @ (posedge CLK or posedge CLEAR)
    begin
        if (CLEAR)
            OUT <= 8'h00;
        else
            OUT <= IN;
    end
endmodule
```

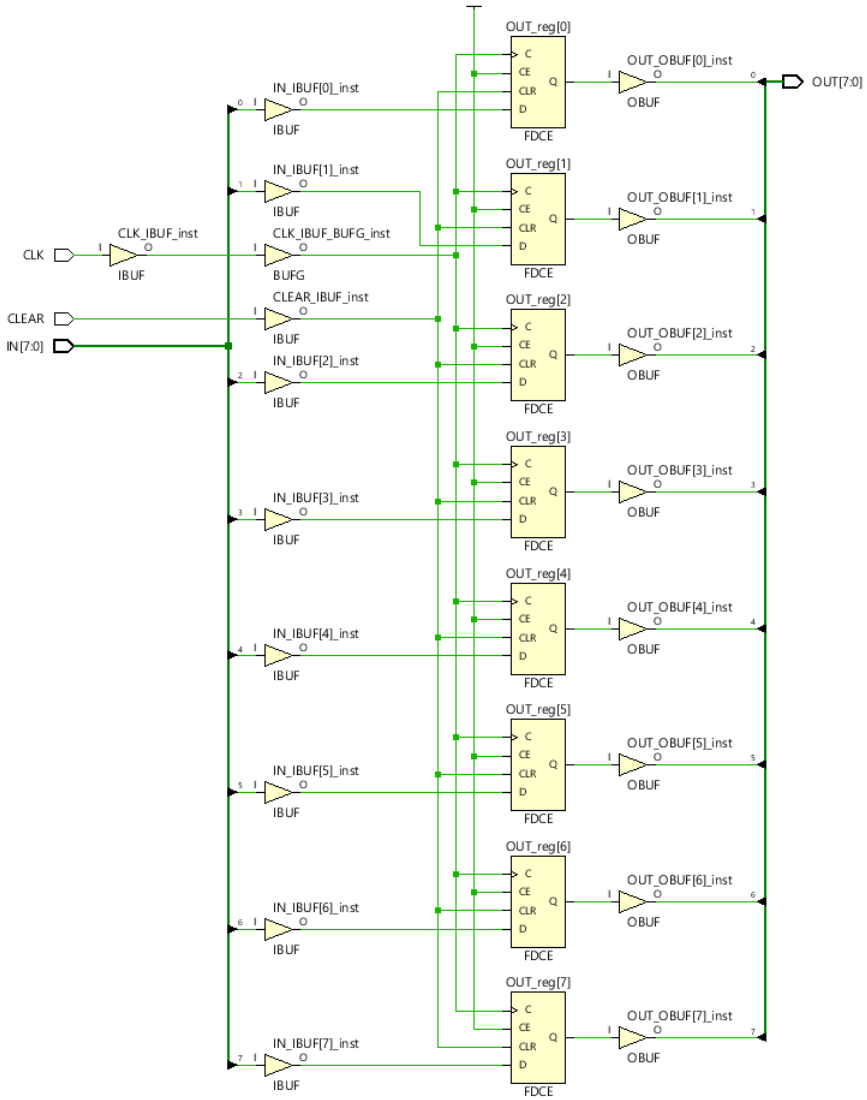
- Behavioral Simulation
  - In In this structure, CLEAR = 0 and every time our CLK value coincides with its positive edge, it takes the D value to the OUT value, while it maintains the OUT value in other cases. When our CLEAR value is 1, our OUT value is 0 and remains 0 independent of the CLK value as long as it is active. The simulation result below is an example of this explanation.



- RTL Schematic



- Technology Schematic



- There are 8 FDCE structure in my circuit.

- This design element is a single D-type flip-flop with clock enable and asynchronous clear.
- When clock enable (CE) is High and asynchronous clear (CLR) is not asserted, the data on the data input (D) of this design element is transferred to the corresponding data output (Q) during the clock (C) transition.
- When CLR is active, it overrides all other inputs and resets the data output (Q) Low.
- When CE is Low, clock transitions are ignored.
- This flip-flop is asynchronously initialized when power is applied. When global set/reset (GSR) is active upon power-up or when GSR is asserted via the STARTUP block, the value of the INIT attribute is placed on the register's output.

- If the REGISTER module is used as the submodule of our 4*8 register module, which we will define as REGISTER2, as in the code shown below, 4*8 registers can be obtained.

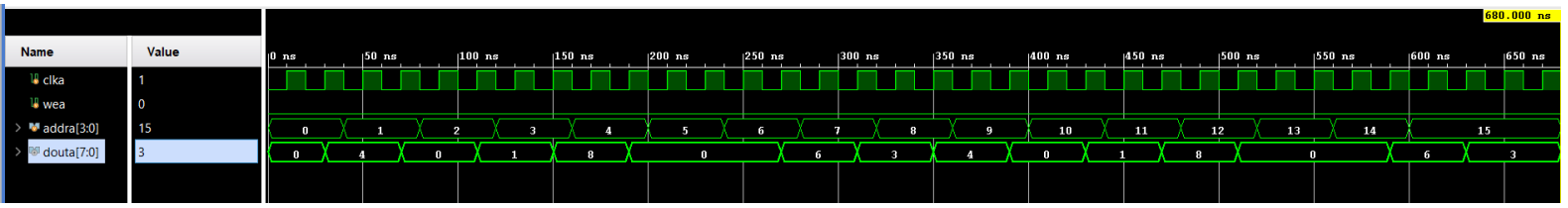
```
input [7:0] IN [3:0];
output [7:0] OUT [3:0];
```

6. BLOCK RAM

- Verilog Code

```
module RAM(
        input clka,
        input wea,
        input [3:0] addra,

        output [7:0]douta
        );
        wire [7:0]dina;
        BLOCK_RAM RAM(
                  .dina(dina),
                  .clka(clka),
                  .wea(wea),
                  .addra(addra),

                  .douta(douta)
                 );

endmodule
```

- Behavioral Simulation
  - I downloaded the memory.coe file in Ninova to the computer and opened it in vivado and wrote my own school number, 40180063, into the file. Then I set the clock frequency to 50Mhz in the test bench file as stated in the assignment. As seen in the output, data has started to be read in the 2.posedge signal. The simulation stated below met expectations.



- Functionalities of Block RAM ports
  - dina => Data input to be written into the memory through port A.
  - clka => Clock signal input.
  - wea => Enables Write operations through port A.
  - addra => Addresses the memory space for port A Read and Write operations.
  - douta => Data output from Read operations through port A.

- Structure of .coe file
  - The .coe file was used in the IP catalog for this assignment. The MEMORY_INITIALIZATION_RADIX variable is a variable that indicates in which base the number will be written, binary decimal hexadecimal, etc. After each value we write in the line, the values are separated by a comma, and when it comes to the end of the values, the values are terminated with a semicolon.

  - Memory.coe File with my student number which is 40180063

```
MEMORY_INITIALIZATION_RADIX=2;
MEMORY_INITIALIZATION_VECTOR=
00000100,
00000000,
00000001,
00001000,
00000000,
00000000,
00000110,
00000011,
00000100,
00000000,
00000001,
00001000,
00000000,
00000000,
00000110,
00000011;
```