SR latch(function

- Circuit Diagram and Truth Table, functions of SR Latch:

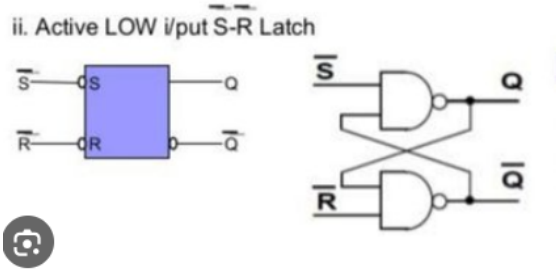


Table Truth table for SR Flip Flop

S	R	Q	Q+
1	1	0	0
1	1	1	1
0	1	X	0
1	0	X	1
0	0	X	1

1. **Characteristic Function (Next State Function):**
   - $Q_{\text{next}} = \overline{S} \cdot Q + S \cdot \overline{R} \cdot \overline{Q}$

   This equation expresses the next state of the latch (Q next) based on the current state (Q) and the inputs (S and R).
2. **Inverse Characteristic Function (Output Function):**
   - $Q' = \overline{Q}$

   This equation expresses the inverse state of Q (Q') in terms of the normal state (Q).

- Verilog code:

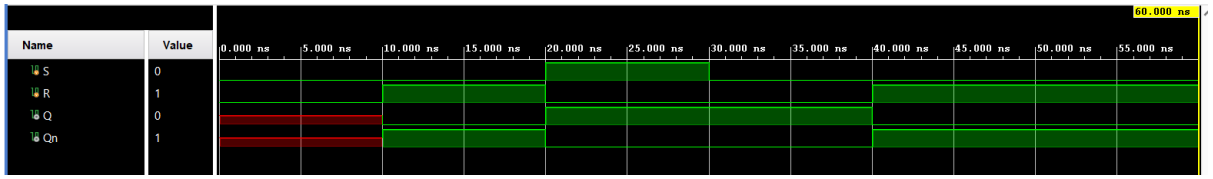
```
module SR_Latch(
input S,R,
output Q, Qn
    );

    NAND nand1(.I1(~S), .I2(Qn), .O(Q));
    NAND nand2(.I1(~R), .I2(Q), .O(Qn));

endmodule
```

- Testbench Code:

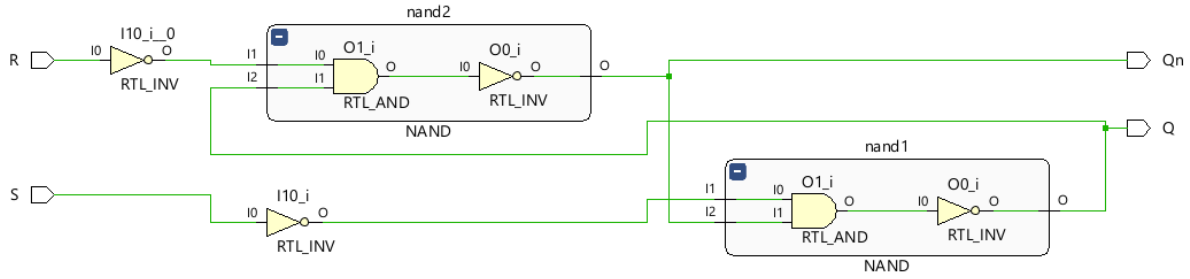
```
module SR_Latch_tb();
reg S,R;
wire Q,Qn;

SR_Latch srlatch1(S, R, Q,Qn);
initial
begin

S=0; R=0;
#10
S=0; R=1;
#10
```

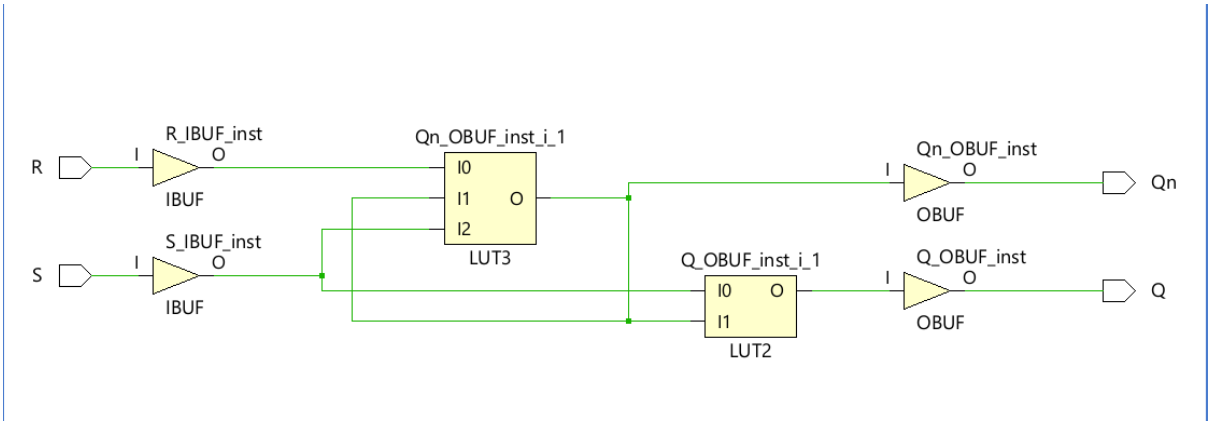
```
S=1; R=0;
#10
S=0; R=0;
#10
S=0; R=1;
#10
S=0; R=1;
#10
$finish();
end
endmodule
```

Simulation Wave:



- RTL and Technology Schematics:



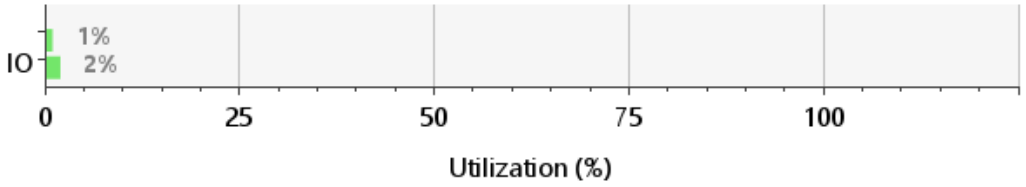


- Utilization and Timing Reports:

Maximum combinational delay is 10.159 and 2 LUTs used.

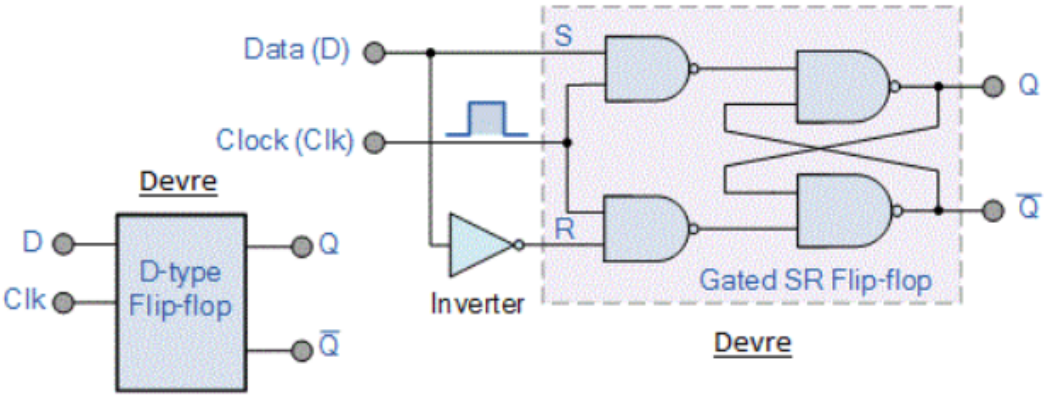
From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
R	Q	10.159	SLOW	3.465	FAST
R	Qn	9.620	SLOW	3.298	FAST
S	Q	9.739	SLOW	3.052	FAST
S	Qn	9.200	SLOW	3.115	FAST

Resource	Utilization	Available	Utilization %
LUT	2	63400	0.00
IO	4	210	1.90



D flipflop

- Circuit Diagram and Truth Table:



Input			Output	
D	reset	clock	Q	Q'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

- Verilog Code:

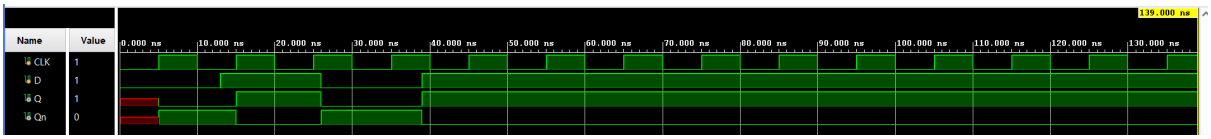
```
module D_FF (
    input CLK, D,
    output Q, Qn
    );

wire setnot, resetnot, dnot;

NOT not1 (D,dnot);
NAND nand1 (D, CLK, setnot);
NAND nand2 (dnot, CLK, resetnot);
NAND nand3 (setnot, Qn, Q);
NAND nand4 (resetnot, Q, Qn);
endmodule
```

- Testbench Code:

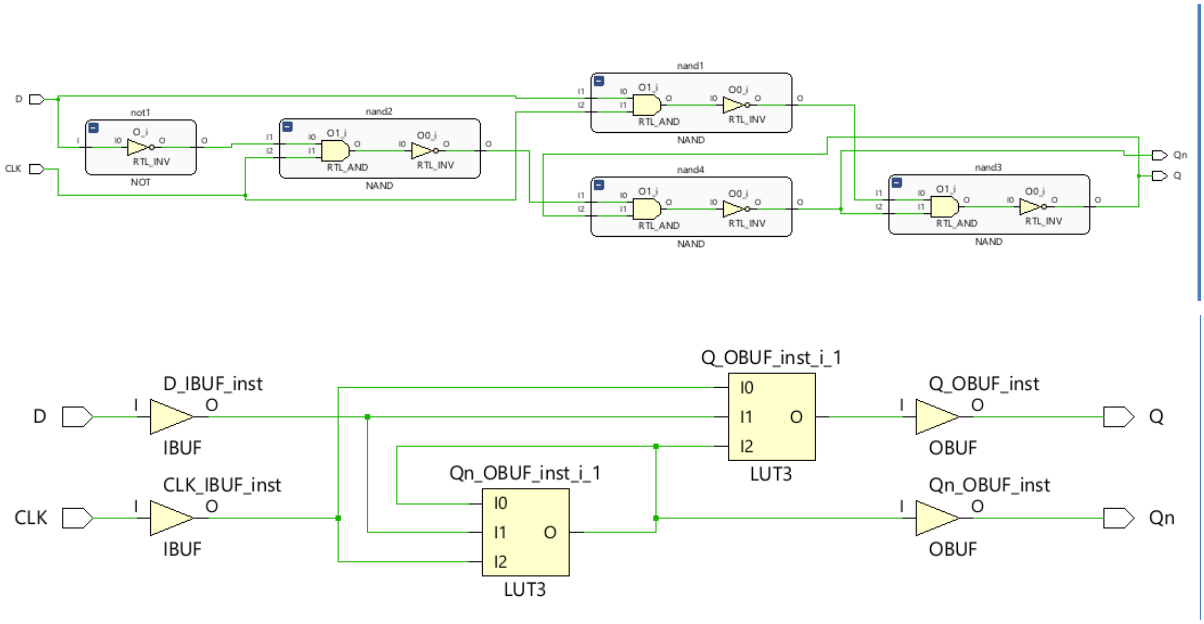
```
module D_FF_tb;
reg CLK, D;
wire Q, Qn;
D_FF myD_FF (
    .CLK(CLK),
    .D(D),
    .Q(Q),
    .Qn(Qn)
);
initial begin
    CLK = 0;
```

```
    D = 0;
    #10 D = 1;
    #10 D = 0;
    #10 D = 1;
    #100 $finish;
end

always begin
    #5 CLK = ~CLK;
end
endmodule
```

- Simulation Wave:



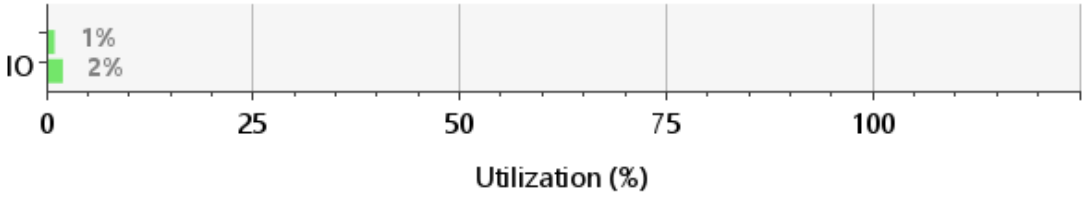
- RTL and Technology Schematics:



- Utilization and Timing Reports:

2 LUTs used and maximum combinational delay is 9.357ns. A period longer than this value is required. Hence, the clock's maximum frequency is limited to 0.107 GHz. This ensures that the design is capable of completing the longest path within a single period.

Resource	Utilization	Available	Utilization %
LUT	2	63400	0.00
IO	4	210	1.90



From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
CLK	Q	9.357	SLOW	2.647	FAST
CLK	Qn	8.418	SLOW	2.525	FAST
D	Q	8.846	SLOW	2.415	FAST
D	Qn	7.907	SLOW	2.282	FAST

D flipflop master slave

- Verilog Code:

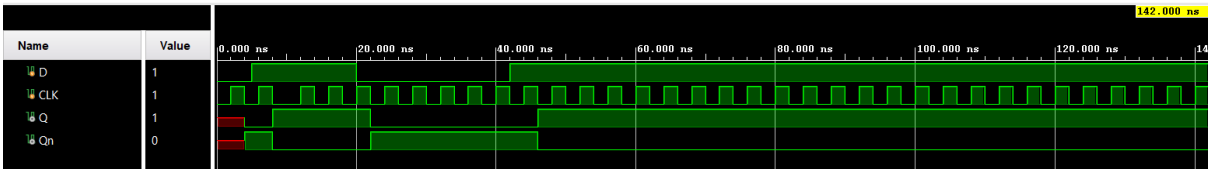
```
module Master_Slave_D_FF(
    input D,CLK,
    output Q, Qn
    );

    wire Qm, Qmnot;

    D_FF master(CLK, D, Qm, Qmnot);
    D_FF slave(!CLK, Qm, Q, Qn);

endmodule
```

- Testbench Code:

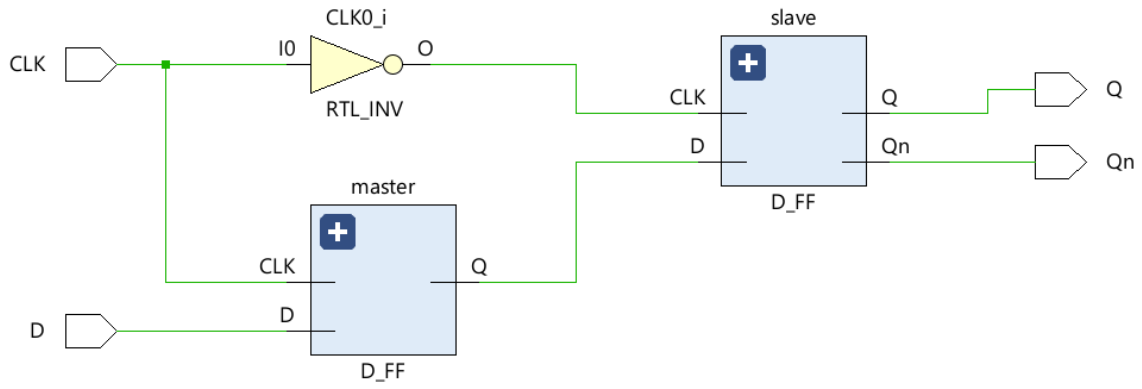
```
module Master_Slave_D_FF_tb;
reg D, CLK;
wire Q, Qn;
```

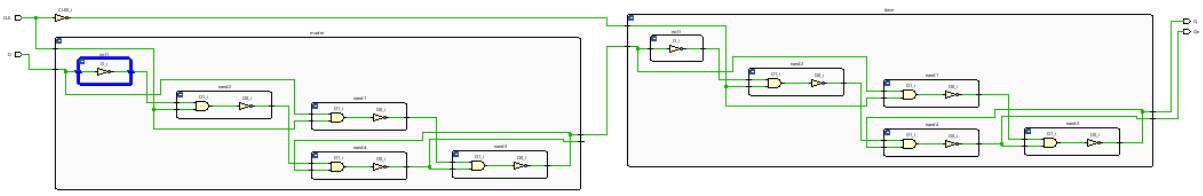
```
Master_Slave_D_FF myD_FF (
    .D(D),
    .CLK(CLK),
    .Q(Q),
    .Qn(Qn)
);

initial begin
    D = 0;
    CLK = 0;
    #5 D = 1;
    #5 CLK = 1;
    #5 CLK = 0;
    #5 D = 0;
    #22 D=1;
    #100 $finish;
end
always begin
    #2 CLK = ~CLK;
end
endmodule
```

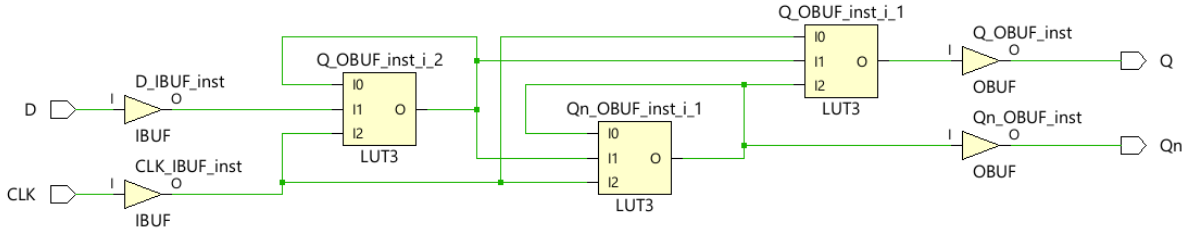
- Simulation Wave:



- RTL and Technology Schematics:





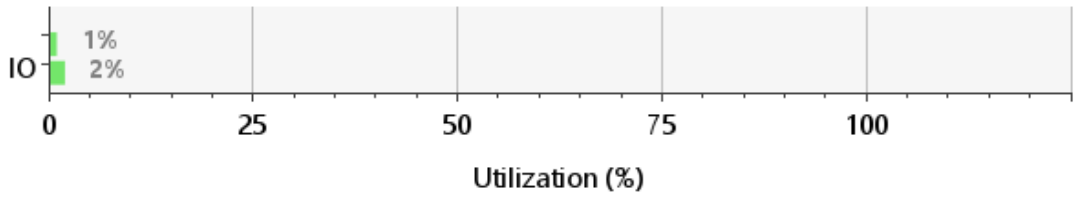


- Timing and Utilization Reports:

Maximum combinational delay is 9.474 and 3 LUTs are used.

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
CLK	Q	9.474	SLOW	2.728	FAST
CLK	Qn	8.840	SLOW	2.680	FAST
D	Q	8.856	SLOW	2.605	FAST
D	Qn	8.222	SLOW	2.415	FAST

Resource	Utilization	Available	Utilization %
LUT	3	63400	0.00
IO	4	210	1.90



- The Master-Slave D Flip-Flop is a digital circuit element composed of two stages. The first stage (master) performs the data sampling process, while the second stage (slave) transfers this data to the output. The master stage responds to the rising or falling edge of the clock signal, whereas the slave stage responds to the opposite

edge. This two-stage process mitigates potential issues observed in single-stage flip-flops. The clock signal ensures synchronized operation, preventing problems like metastability. In summary, the Master-Slave D Flip-Flop is utilized to store and transfer data in a controlled manner. In our Flip-Flop the falling edge of the clock effects the output.

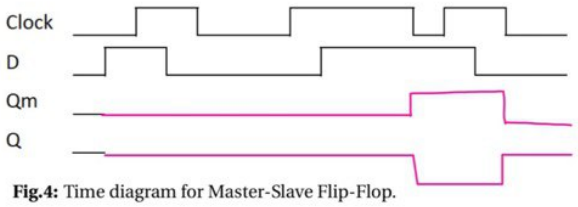


Fig.4: Time diagram for Master-Slave Flip-Flop.

D flipflop Behavioral

- Verilog Code:

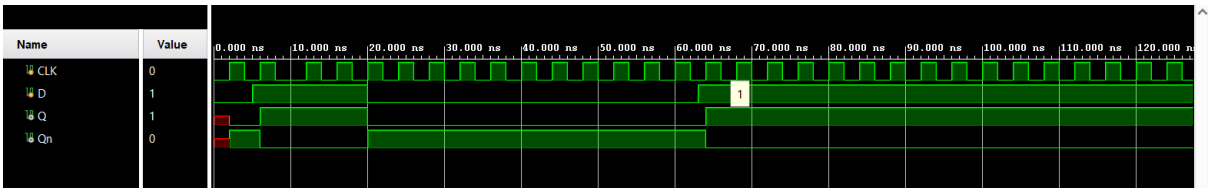
```
module D_FF_behav (
    input CLK, D,
    output Q, Qn
    );

wire setnot, resetnot, dnot;
reg ff;

    always@ (posedge CLK) begin
        ff <= D;
    end
    assign Q = ff;
    assign Qn = !ff;

endmodule
```

- Testbench Code:

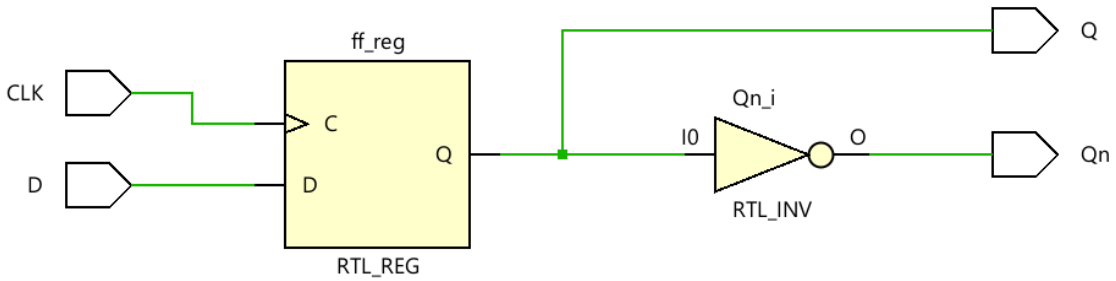
```
module D_FF_behav_tb;

reg CLK, D;
wire Q, Qn;

D_FF_behav myD_FF_behav (
    .CLK(CLK),
    .D(D),
    .Q(Q),
```

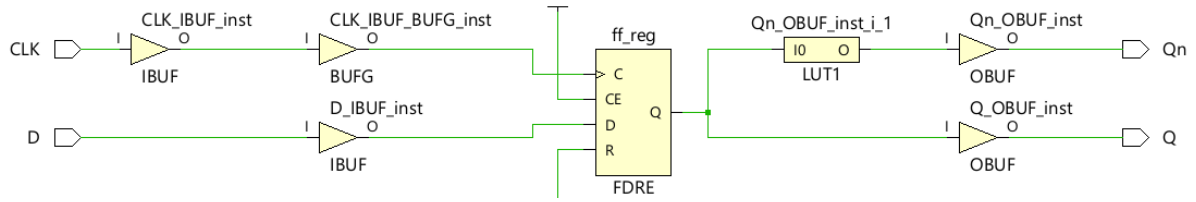
```
    .Qn(Qn)
);

initial begin
    CLK = 0;
    D = 0;

    #5 D = 1;
    #5 CLK = 1;
    #5 CLK = 0;
    #5 D = 0;
    #43 D=1;
    #100 $finish;
end
always begin
    #2 CLK = ~CLK;
end

endmodule
```

- Simulation Wave



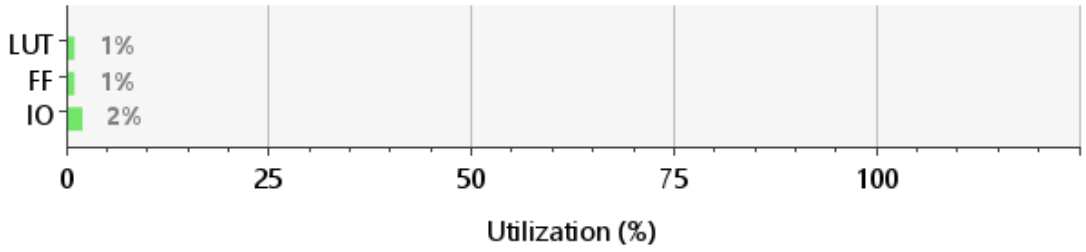
- RTL and Technology Schematics





- Utilization Report:

Resource	Utilization	Available	Utilization %
LUT	1	63400	0.00
FF	1	126800	0.00
IO	4	210	1.90



8bit Register

- Verilog Code:

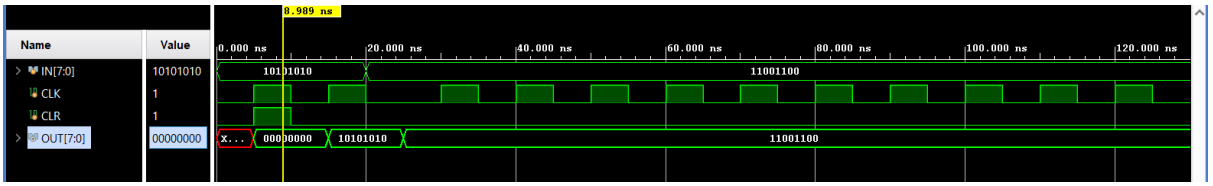
```
module bit8_reg(
    input [7:0] IN,
    input CLK,
    input CLR,
    output reg [7:0] OUT
    );

    always @(posedge CLK or posedge CLR) begin
        if (CLR==1'b1)
            OUT = 8'b0000_0000;
        else OUT=IN;
    end
endmodule
```

- Testbench Code:

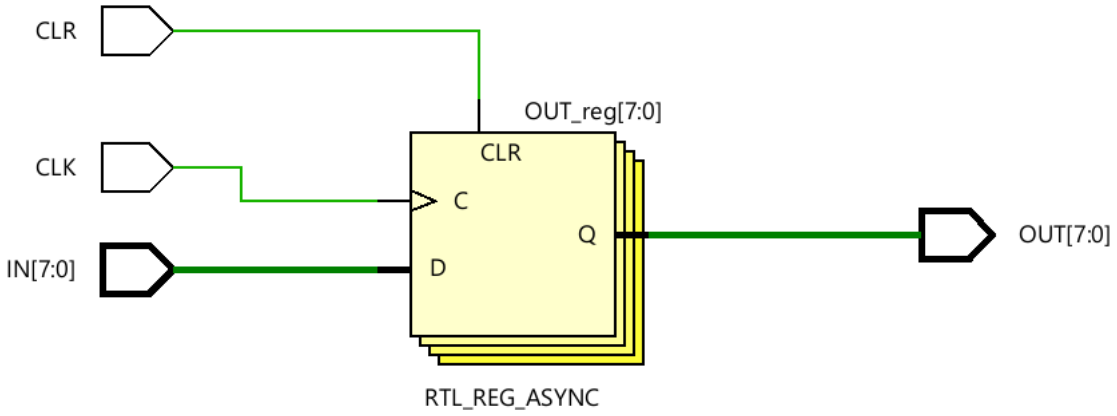
```
module tb_bit8_reg;
```

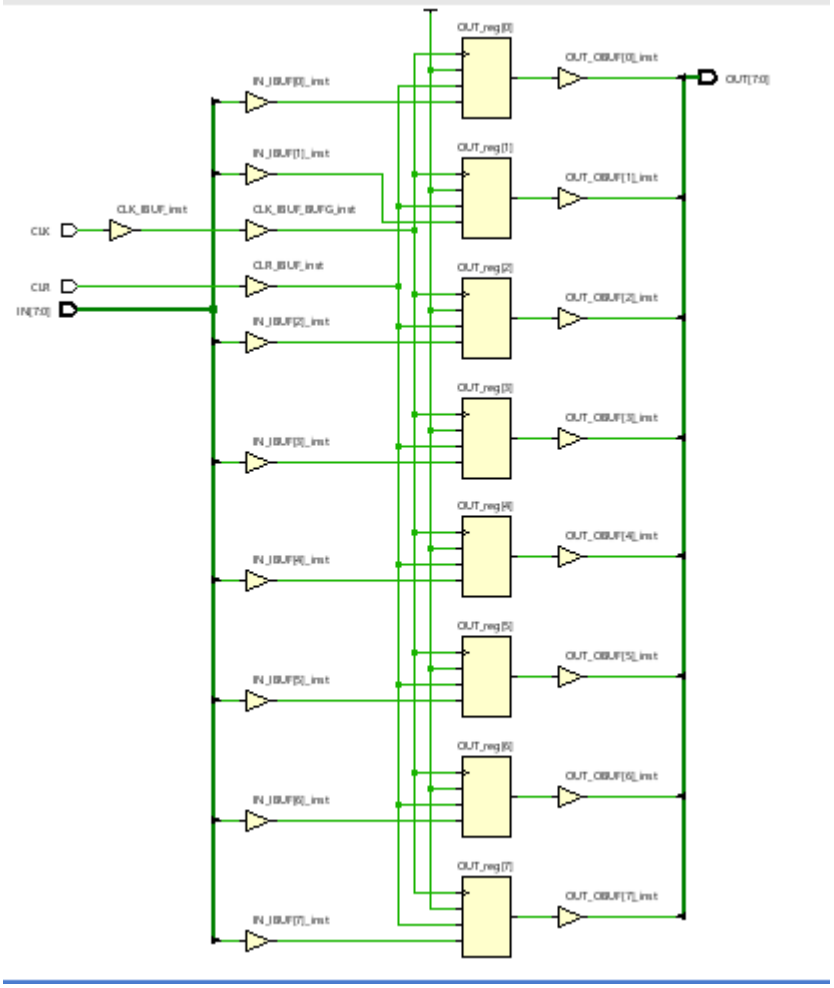
```
reg [7:0] IN;
reg CLK, CLR;
wire [7:0] OUT;
bit8_reg myBit8Reg (
    .IN(IN),
    .CLK(CLK),
    .CLR(CLR),
    .OUT(OUT)
);

initial begin
    IN = 8'b10101010;
    CLK = 0;
    CLR = 0;

    #5 CLR = 1;
    #5 CLR = 0;
    #10 IN = 8'b11001100;
    #5 CLK = 1;
    #5 CLK = 0;
    #100 $finish;
end

always begin
    #5 CLK = ~CLK;
end

endmodule
```

- Simulation Wave:arada bi tane daha clr



- RTL and Technology Schematics:

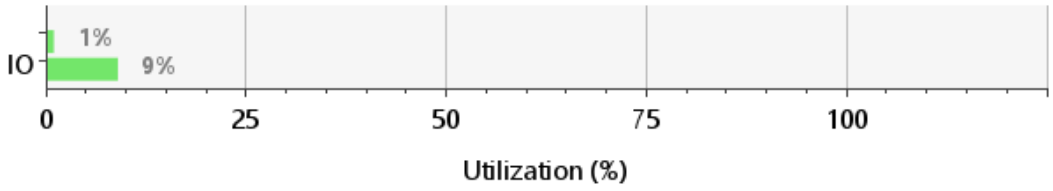




- Utilization Reports:

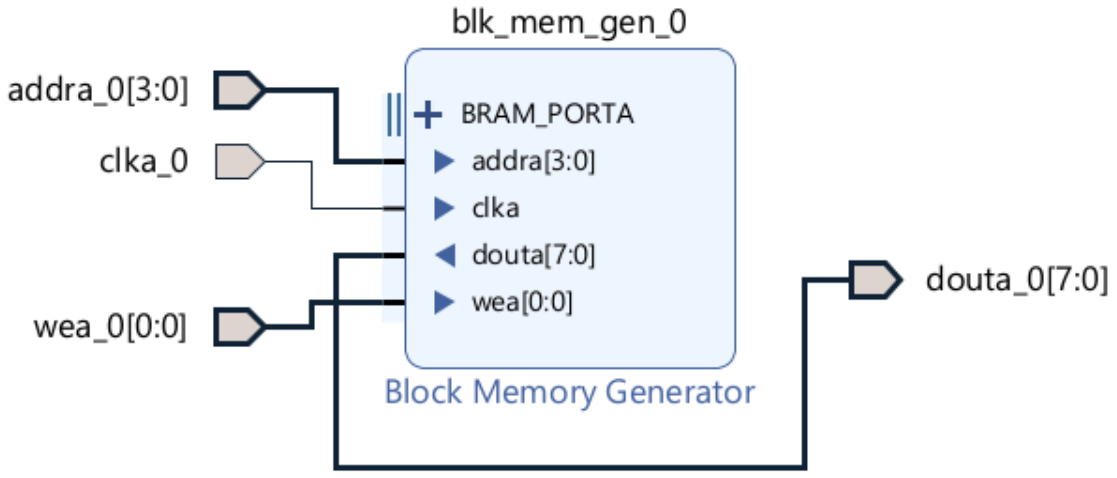
8 LUTs are used.

Resource	Utilization	Available	Utilization %
FF	8	126800	0.01
IO	18	210	8.57



We can define 4x8 bit of register array by: reg [7:0] myArray [0:3];

BRAM



- Verilog Code:

```
module Top_BRAM(
    input CLK, WE,
```

```
    input [3:0] addr,
    output [7:0] dout

    );
     wire [7:0] din;

    BRAM brami (.clka_0(CLK), .wea_0(WE), .addra_0(addr),
.douta_0(dout));
endmodule
```

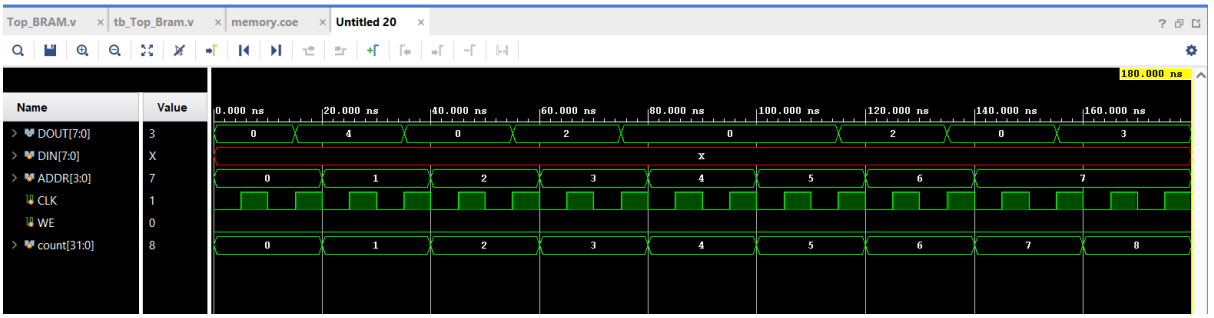
- Testbench Code:

```
module top_bram_TB( );
wire [7:0] DOUT;
reg [7:0] DIN;
reg [3:0] ADDR;
reg CLK;
reg WE=0;

integer count=0;
Top_BRAM uut(CLK, WE, ADDR, DOUT);
initial
begin
CLK=0;
forever #5 CLK=~CLK;
end
initial
begin
while (count<8)
begin
ADDR=count;
#20 count=count+1;
end
#20
$finish;
end
endmodule
```

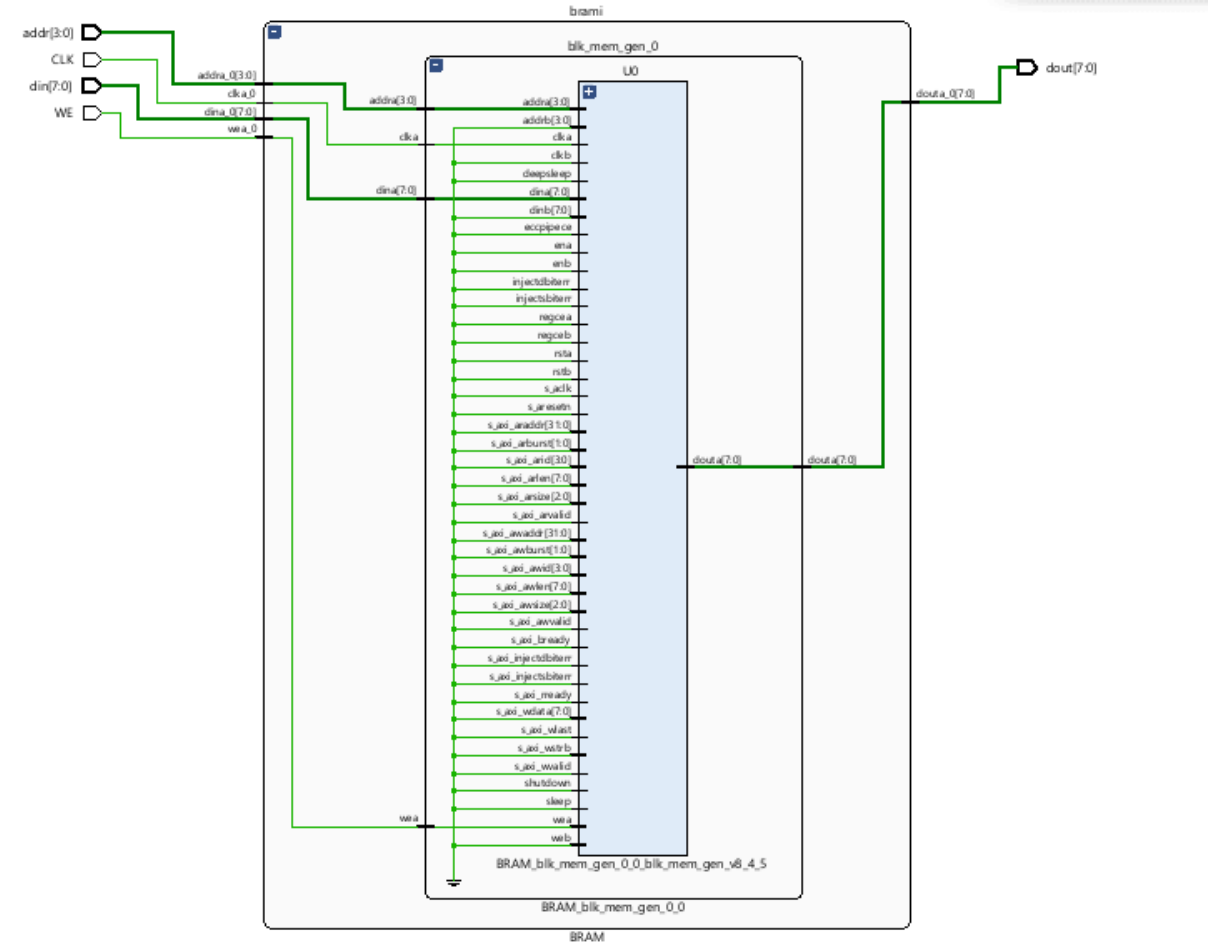
- My memory.coe file:

```
MEMORY_INITIALIZATION_RADIX=2;
MEMORY_INITIALIZATION_VECTOR=
00000100,
00000000,
00000010,
00000000,
00000000,
00000010,
00000000,
00000011,
00000100,
00000000,
00000010,
00000000,
00000000,
00000010,
00000000,
00000011;
```

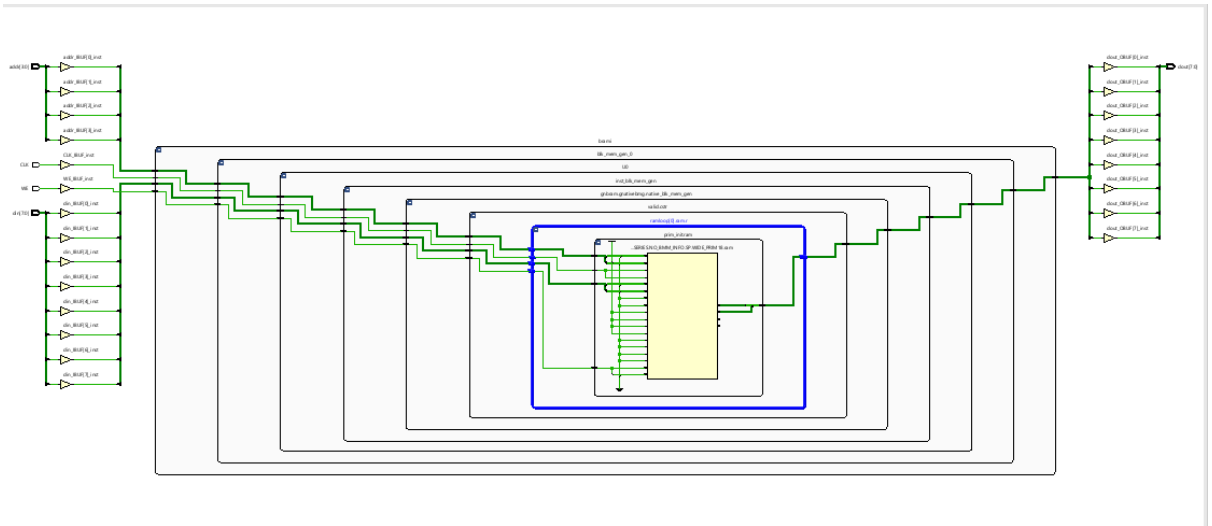
- Simulation results:



- RTL and Technology Schematics:

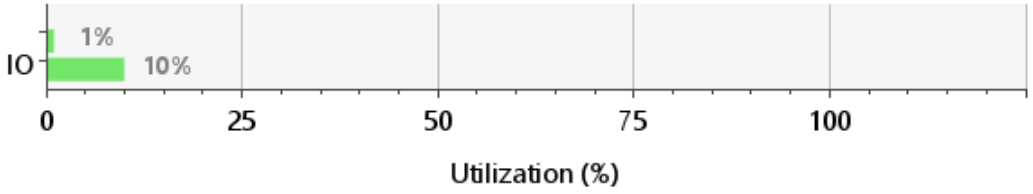

bram
addr[3:0]
CLK
din[7:0]
WE
addra_0[3:0]
clka_0
dina_0[7:0]
wea_0
blk_mem_gen_0
U0
addra[3:0]
clka
dina[7:0]
douta[7:0]
douta_0[7:0]
dout[7:0]
addrb[3:0]
clkb
deepsleep
dinb[7:0]
eccpipece
ena
enb
injectdbiterr
injectsbiterr
regcea
regceb
rsta
rstb
s_aclk
s_aresetn
s_axi_araddr[31:0]
s_axi_arburst[1:0]
s_axi_arid[3:0]
s_axi_arlen[7:0]
s_axi_arsize[2:0]
s_axi_arvalid
s_axi_awaddr[31:0]
s_axi_awburst[1:0]
s_axi_awid[3:0]
s_axi_awlen[7:0]
s_axi_awsize[2:0]
s_axi_awvalid
s_axi_bready
s_axi_injectdbiterr
s_axi_injectsbiterr
s_axi_rready
s_axi_wdata[7:0]
s_axi_wlast
s_axi_wstrb
s_axi_wvalid
shutdown
sleep
wea
web
BRAM_blk_mem_gen_0_0_blk_mem_gen_v8_4_5
BRAM_blk_mem_gen_0_0
BRAM

-



Utilization report:

Resource	Utilization	Available	Utilization %
BRAM	0.50	135	0.37
IO	22	210	10.48



- The `address` port specifies the address to access in the memory cell. The `clk` port represents the clock signal that synchronizes memory operations. The `we` port controls whether a write operation will occur. The `dout` port reads the data from the memory cell at the specified address. Such a structure is commonly used for applications like program memory or data memory, with memory access and control forming the core of the design.