30 Days of RTL Coding

-By T Tushar Shenoy

Day 27

Problem Statement: Implementing a 16x4 ROM.

Theory:

ROM

Read-only memory (ROM) is a data storage device used in personal computers (PCs) and other electronic devices to store information securely. It houses programs or software instructions and includes the programming required to start a computer, which is required for boot-up. It also conducts significant input/output duties. The alteration of this sort of memory, known as "firmware," has been a source of design concern throughout the growth of the contemporary computer.

Because ROM is read-only memory, it can't be modified; it's also permanent and non-volatile, which means it keeps its data even if the power is turned off. Random-access memory (RAM), on the other hand, is volatile; when power is turned off, it is gone. The phrase "non-volatile memory," which is comparable, can be applied here. In its long-term state, ROM in computer is "stateful," whereas RAM is "stateless."

ROM in Computer has the Following Characteristics:

- Non-volatile memory is referred to as ROM.
- The information saved in ROM is irreversible.
- We can only read the information and applications stored on it.
- In binary format, information and applications are stored on ROM.
- It's utilized throughout the computer's boot-up procedure.

Types of ROM

They are four types of ROM in the computer:

- MROM (Masked read-only memory)
- PROM (Programmable read-only memory)
- EPROM (Erasable programmable read-only memory)
- EEPROM (Electrically erasable programmable read-only memory)

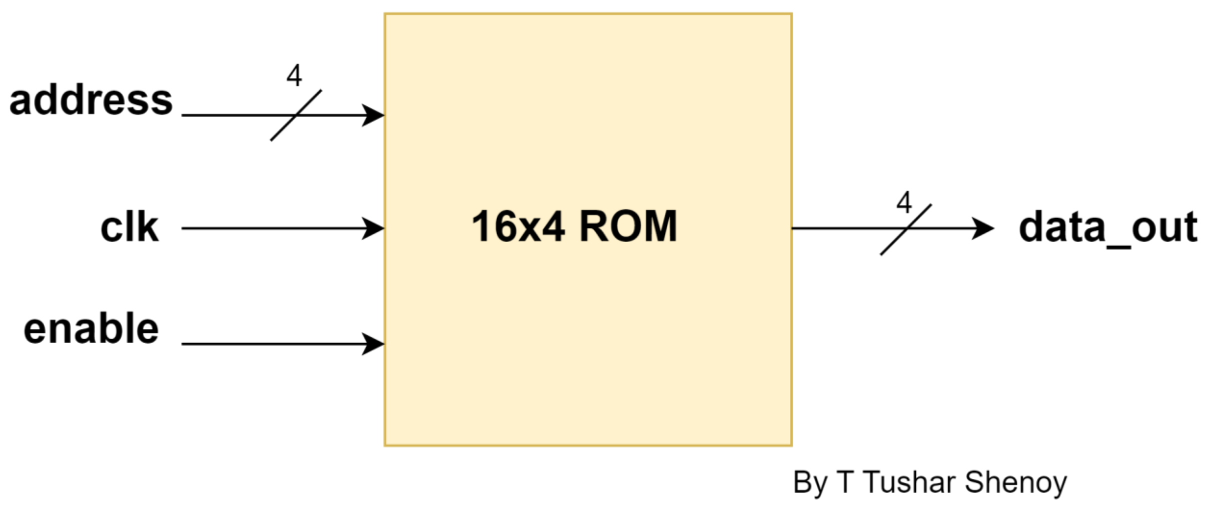


FIG: 16x4 ROM

Verilog Code:

```
//Verilog Code for 16x4 ROM
module ROM(clk,enable,address,data_out);

input clk,enable;
input [3:0]address;

output reg [3:0]data_out;

reg [3:0]mem[15:0];

always@(posedge clk)
begin
   if(enable)
      data_out<=mem[address];
   else
      data_out<=4'bxxxx;
end

initial begin
mem[0]=4'b1111;
mem[1]=4'b1110;
mem[2]=4'b1101;
mem[3]=4'b1100;
mem[4]=4'b1011;
mem[5]=4'b1010;
mem[6]=4'b1001;
mem[7]=4'b1000;
mem[8]=4'b0111;
```

```
mem[9]=4'b0110;
mem[10]=4'b0101;
mem[11]=4'b0100;
mem[12]=4'b0011;
mem[13]=4'b1010;
mem[14]=4'b0001;
mem[15]=4'b0000;
end

endmodule
```

Testbench Code:

```
//Testbench Code for 16x4 ROM
module ROM_tb();

reg clk,enable;
reg [3:0]address;

wire [3:0]data_out;

ROM dut(.clk(clk),.enable(enable),.address(address),.data_out(data_out));

initial begin
clk=1'b0;
enable=1'b0;
#10;
enable=1'b1;

address=4'b1010;
#10;
address=4'b1001;
#10;
address=4'b1000;
#10;
address=4'b0010;
#10;
address=4'b0001;
#10;
address=4'b1111;
#10;
```

```
address=4'b1110;
#10;
address=4'b1101;
#10;
address=4'b1100;

#10 $finish;
end

always #4 clk=~clk;
endmodule
```

Schematic:

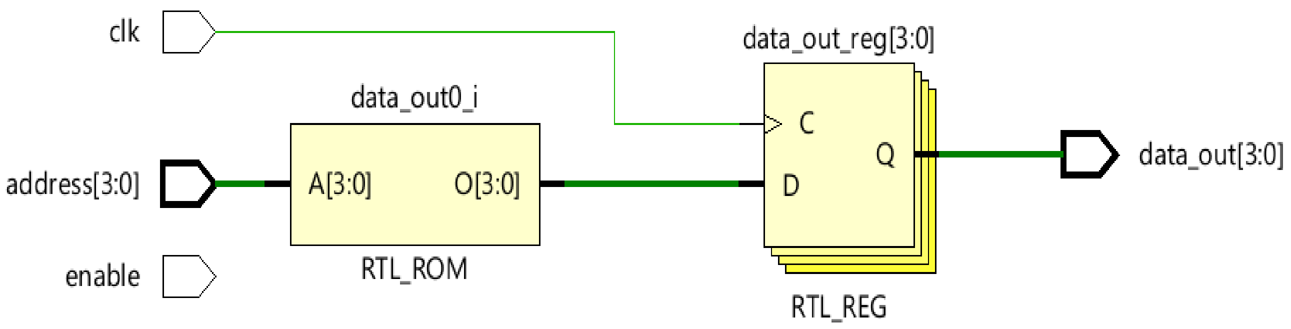
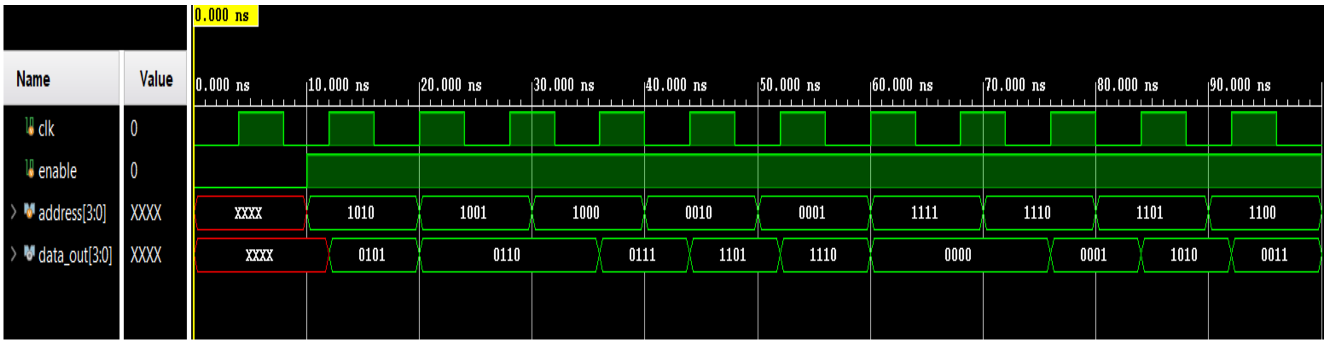


Fig: 16x4 ROM

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-27-16x4-ROM