30 Days of RTL Coding

-By T Tushar Shenoy

Day 26

Problem Statement: Implementing Dual Port Synchronous RAM 256x8

Theory:

RAM

Random Access Memory is the temporary memory used in a processor or the digital system which requires larger memory for storing temporary data. When designing any system on FPGA, sometimes we require a RAM block which is also called BRAM or block RAM. In this post, we'll how to describe a RAM in Verilog HDL. Most of the latest FPGAs have BRAM and if we synthesize this, it will be synthesized into a BRAM

A RAM has a data bus (sometimes called width of RAM) form which we can access content from the RAM or we can put a content on this, and if we give write command to the RAM, it'll write RAM with the content on the data bus. RAM has lots of addresses (sometimes called depth of RAM) and these addresses can be accessed by the address bus.

Other than data and address bus, RAM has one more input read/write. If it's state is changed, let's say if it is high, the content at data bus is written to the address provided to the RAM from address bus else the content of the address provided by address bus is reflected to the data bus.

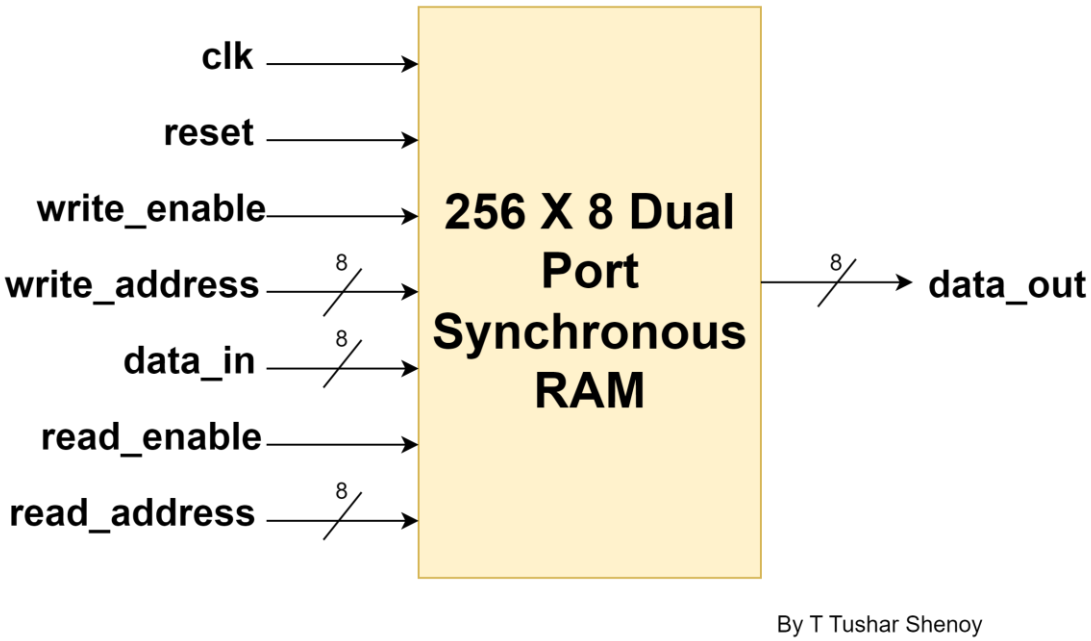

FIG: Dual Port Synchronous RAM

Verilog Code:

```
//Verilog code for Dual Port Synchronous RAM
module
RAM(clk,reset,write_enable,read_enable,write_address,read_address,data_in
,data_out);

parameter RAM_width=8;
parameter RAM_depth=256;
parameter address_size=8;

input clk,reset,write_enable,read_enable;
input [RAM_width-1:0]data_in;
input [address_size-1:0]write_address,read_address;

output reg [RAM_width-1:0]data_out;

reg [RAM_width-1:0]mem[RAM_depth-1:0];

integer i;

always@(posedge clk)
   begin
      if(reset)
        begin
         for(i=0;i<RAM_depth;i=i+1)
           mem[i]<=0;
        data_out<=0;
        end
      else
      begin
```

```
        if(write_enable)
         mem[write_address]<=data_in;
        else if(read_enable)
        data_out<=mem[read_address];
      end
    end
endmodule
```

Testbench Code:

```
//Testbench for Dual Port Synchronous RAM
module RAM_tb;

parameter RAM_width = 8;
parameter RAM_depth = 256;
parameter address_size = 8;

reg clk;
reg reset;
reg write_enable;
reg read_enable;
reg [RAM_width-1:0] data_in;
reg [address_size-1:0] write_address;
reg [address_size-1:0] read_address;

wire [RAM_width-1:0] data_out;

RAM
dut(.clk(clk),.reset(reset),.write_enable(write_enable),.read_enable(read_ena
ble),.write_address(write_address),.read_address(read_address),.data_in(data
_in),.data_out(data_out)
);

initial begin

    clk = 1'b0;
    reset = 1'b0;
    write_enable =1'b0;
    read_enable = 1'b0;
```

```
    data_in = 8'b0;
    write_address = 8'b0;
    read_address = 8'b0;


    reset = 1;
    #10;
    reset = 0;


    write_enable = 1;
    data_in = 8'b10101010;
    write_address = 8'b00100011;
    #10;
    write_enable = 0;


    read_enable = 1;
    read_address = 8'b00100011;
    #10;
    read_enable = 0;

    // Add more testcases Here
    $finish;
end
always  #5 clk = ~clk;


endmodule
```

Schematic:

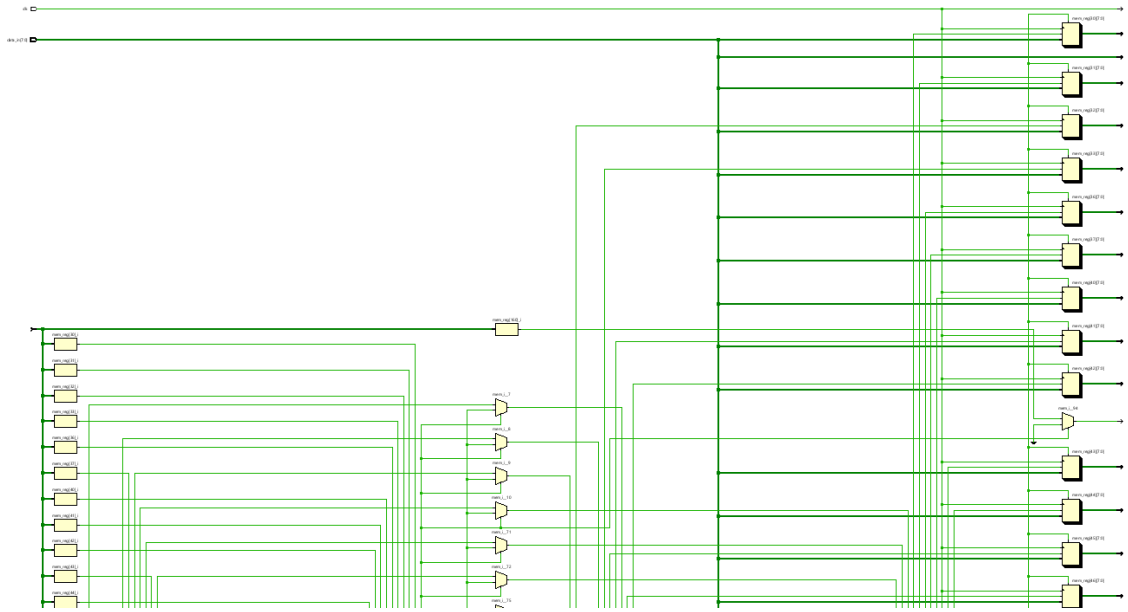

Fig: Dual Port Synchronous RAM (not complete schematic)

Simulation Output:

GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-26-Dual-Port-Synchronous-RAM