The University of Texas at Dallas

Dept of Electrical Engineering

EECT 6325: VLSI Design

Project 1

DESIGN AND ANALYSIS OF RANDOM ACCESS MEMORY

Done by:

Names
Sameer Nitin Chourikar
Vyasa Akilesh Sashank Akondi
Niket Kedarnath Ramani

Random Access Memory (RAM) :

Random Access Memory or the main memory is a hardware device that accepts information to be stored and recovered on a computer. However, unlike the Read Only Memory (ROM), RAM is a volatile memory and requires power to keep the data accessible. If the computer is turned off, all data contained in RAM is lost, while ROM (Read-Only Memory) is non-volatile and holds data permanently when the power is turned off. There are two main types of RAM: dynamic Random Access Memory (DRAM), and Static Random Access memory (SRAM). The RAM, in most personal computers, is Dynamic RAM. DRAM stores the binary information in the form of electric charges that is applied to capacitors and therefore the capacitors must be recharged continuously to retain their usage. The SRAM has lower access time, so it is faster compared to the DRAM. SRAM is costlier than DRAM. DRAM consumes less power. The SRAM internal circuitry is complex and hence less storage space is available when compared to the same size DRAM memory chip. DRAM has high packaging density than SRAM. There are mainly 5 types of DRAM: 1) Asynchronous DRAM (ADRAM) 2) Synchronous DRAM (SDRAM) 3) Double-Data-Rate SDRAM (DDR SDRAM) 4) Rambus DRAM (RDRAM) 5) Cache DRAM (CDRAM) .

General Description of Design :

Our design of the Random Access Memory consists of 3-bit address, 8-bit data, clock, read request and write request as the five inputs. The output is an 8-bit data. During the positive edge of the clock, it checks whether read or write request is enabled. If read request is enabled, then the data present in the requested address is transferred to the output. If write request is enabled, then whatever data is given at the input is transferred to the desired address location. The flip flop used in our design is D – flip flop.

Block Diagrams of Random Access Memory :

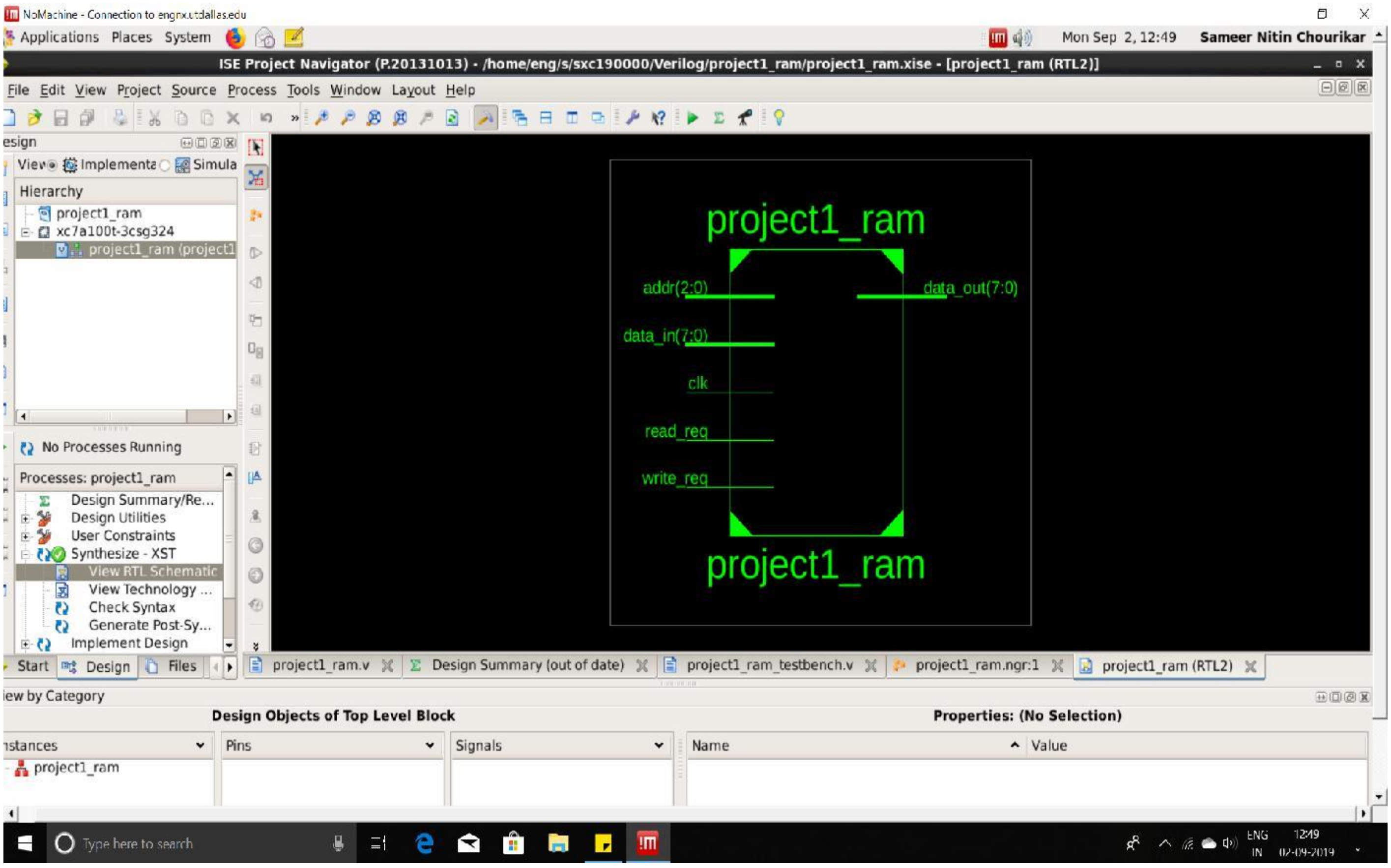

ISE Project Navigator (P.20131013) - /home/eng/s/sxc190000/Verilog/project1_ram/project1_ram.xise - [project1_ram (RTL2)]
project1_ram
addr(2:0)
data_in(7:0)
clk
read_req
write_req
data_out(7:0)
project1_ram

Figure 1

Figure 1 shows the block diagram of the Random Access Memory.

Inputs are as follows:

- ☐ addr[2:0] : 3 bit address to access the RAM
- ☐ data_in [7:0] : 8 bit data to be written in the memory
- ☐ clk : Clock signal
- ☐ read_req : Read request in the memory
- ☐ write_req : Write request in the memory

Outputs are as follows:

- ☐ data_out [7:0] : 8 bit data to be read from the memory

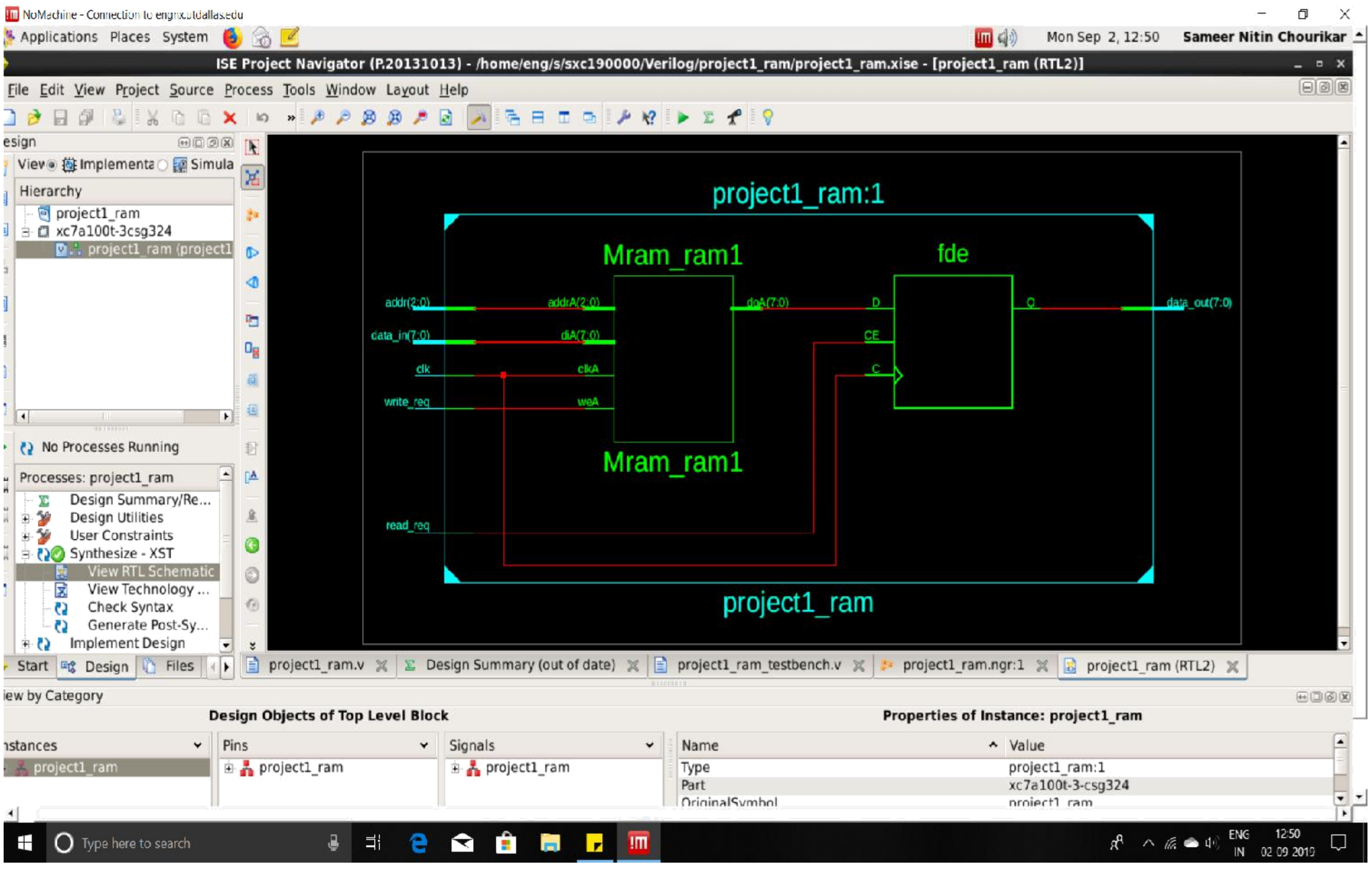


Figure 2

Figure 2 shows the internal block diagram of the Random Access Memory.

The first block in the figure is self explanatory.
fde in the second block is a **D- Flip Flop.** It is basically used for the **Read operation.**

The inputs for the D Flip Flop are as follows:

- ☐ Clk --> C : Clock Signal
- ☐ D : Data to be read
- ☐ read_req --> CE : Enable/Strobe input

The outputs for the D Flip Flop are as follows:

- ☐ O : Data which is read from the RAM

Verilog Code :

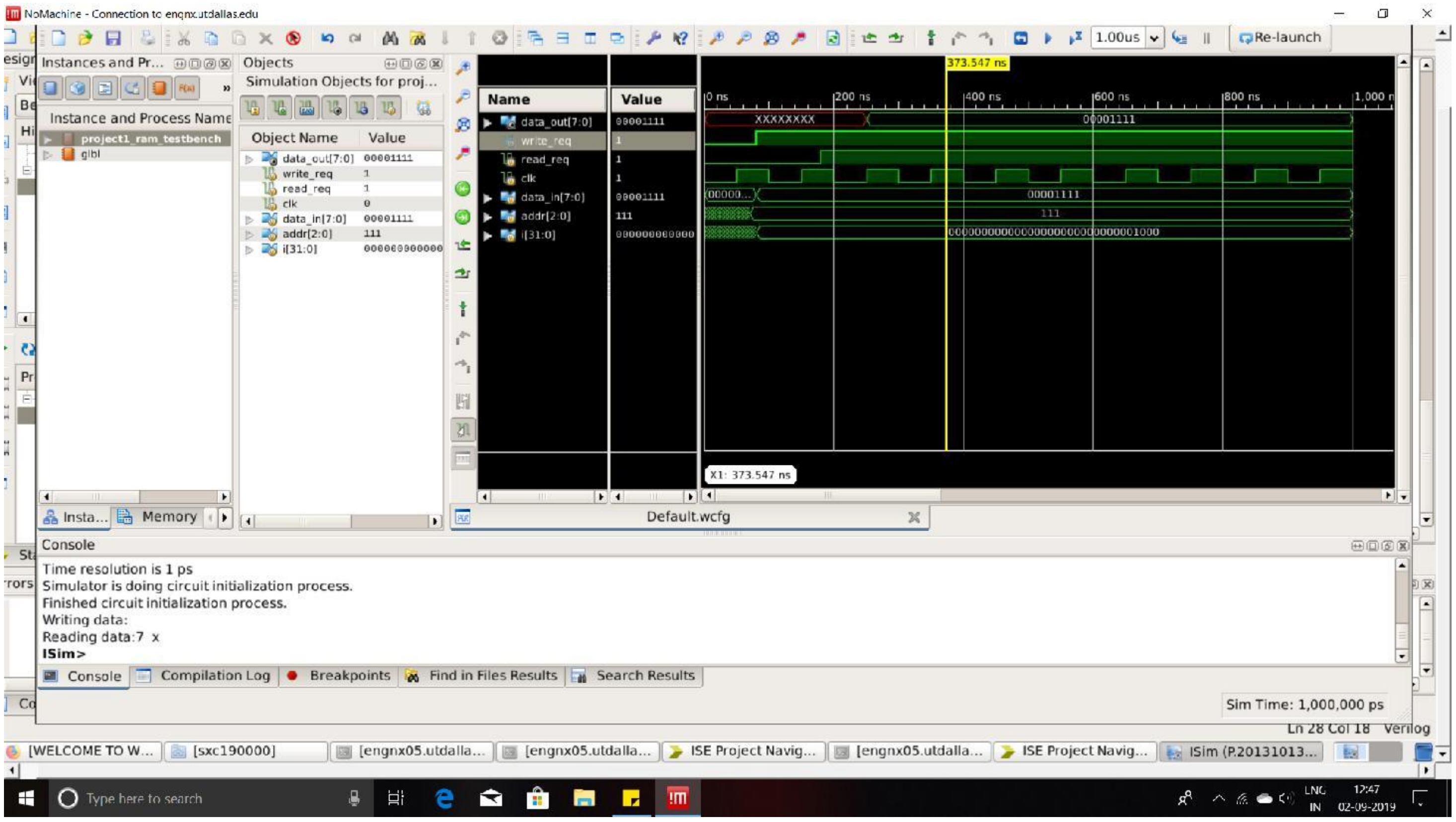
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:  18:38:22 08/31/2019
// Design Name:
// Module Name:  project1_ram
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module project1_ram(input write_req,input read_req, input clk, input [7:0] data_in, output reg [7:0]
data_out, input [2:0]addr);

reg [7:0] ram [0:7];

always @ (posedge clk)
begin
	if (write_req == 1)		//If data is available to write,
		begin
			ram [addr] <= data_in; //Data is written in the RAM at the current location of address end

end

always @ (posedge clk)
	begin
		if (read_req == 1)		//If data is available to read,
				begin
				 data_out <= ram [addr];	//Data is read from the RAM according to the location of the address
				 end
	end

endmodule
```

Testbench Code :

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:  17:20:19 09/01/2019
// Design Name:  project1_ram
// Module Name:  /home/eng/s/sxc190000/Verilog/project1_ram/project1_ram_testbench.v
// Project Name: project1_ram
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: project1_ram
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module project1_ram_testbench;

// Inputs
reg write_req;
reg read_req;
reg clk;
reg [7:0] data_in;
reg [2:0] addr;

// Outputs
wire [7:0] data_out;

// Instantiate the Unit Under Test
(UUT) project1_ram uut (
.write_req(write_req),
.read_req(read_req),
.clk(clk),
.data_in(data_in),
.data_out(data_out),
.addr(addr)
);
```

```
   integer i = 0;                //Initializing the counter


initial begin
// Initialize Inputs
write_req = 0;           //Initializing all the inputs to zero
read_req = 0;
clk = 0;
data_in = 0;
addr = 0;

// Wait 100 ns for global reset to finish
#100;                              //Delay

// Add stimulus here

end

initial begin
   forever begin
          #50 clk = ~clk;          //Clock generation
   end
end

  initial begin
    for (i = 0; i <=7 ; i = i +1 )
          begin
             addr = i;                         //Counter to go through all the 8 states of the
             #10;                              address //Delay
          end


$display ("Writing data:");
write_req = 1;                              //Write enable: ON
data_in = 8'b00001111;                      //Data to be written

#100;                                  //Delay
read_req = 1;                         //Read enable: ON
$display ("Reading data:",addr,data_out); //Reading the data currently at the address location in the RAM

    end


endmodule
```

Output Waveforms :



Name	Value
data_out[7:0]	00001111
write_req	1
read_req	1
clk	1
data_in[7:0]	00001111
addr[2:0]	111
i[31:0]	000000000000

```
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Writing data:
Reading data:7  x
ISim>
```

Explanation :

As seen from the output waveforms, when the signal write_req goes high, data is written in the memory. Similarly, when the signal read_req goes high, data is read from the memory. Here the address space in which the data has to be written and read is "111". In the test bench, the data to be written was "00001111". It was written at the desired address. The same data from the same address was read an displayed.