



NMJ31804 – Principles of Computer Architecture
SEMESTER 2, 2021/22

LAB 3) RAM and ROM

Name: TAN YI JIE

Matric Number: 191020976

Program code: RK20 - Computer Engineering

The screenshot displays the Quartus II 64-Bit IDE interface. The main window shows the Verilog code for a module named `lab3rom`. The code defines a module with inputs `clk`, `read`, and `addr`, and an output `data`. It includes a register `data` and a case statement for initializing the data based on the `addr` value. The Project Navigator on the left shows the project hierarchy, and the Tasks pane on the bottom left lists compilation tasks. The bottom status bar shows the compilation progress and messages, indicating that the synthesis was successful with 0 errors and 1 warning.

```

1 module lab3rom (clk, read, addr, data);
2   input clk, read;
3   input [1:0] addr;
4   output [7:0] data;
5   reg [7:0] data;
6
7   always @(posedge clk)
8     if (read)
9       begin
10         case (addr)
11           2'b00: data <= 8'b10000001;
12           2'b01: data <= 8'b10000010;
13           2'b10: data <= 8'b10000011;
14           2'b11: data <= 8'b10000100;
15         endcase
16       end
17   else
18     data <= 8'bz;
19 endmodule
20

```

Compilation Report - lab3rom

Task: Compilation

Task

- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
 - Edit Settings
 - View Report

Type ID Message

- > 16010 Generating hard_block partition "hard_block:auto_generated_inst"
- > 21057 Implemented 23 device resources after synthesis - the final resource count might be different
- > Quartus II 64-Bit Analysis & Synthesis was successful. 0 errors, 1 warning

Figure 1 shows that the compilation of the ROM module is successful.

Simulation Waveform Editor - [lab3rom.sim.vwf (Read-Only)]

File Edit View Simulation Help

Master Time Bar: 0 ps Pointer: 54.75 ns Interval: 54.75 ns Start: End:

Name	Value at 0 ps
clk	B 0
> addr	H 0
read	B 0
> data	H ZZ

Timing diagram showing signals: clk, addr, read, data. The data bus shows a sequence of values: ZZ, 83, 84, 81, 82, 83, 84, ZZ, 84, 81, 82, 83, 84, 81, 82, ZZ.

Figure 2 shows the simulation result for the ROM module, the output data will have specific result in hexadecimal radix when the input of read is HIGH and will reflect it during each of the positive edge of the clock. Otherwise, the output data will show “ZZ”.

2. Design of RAM

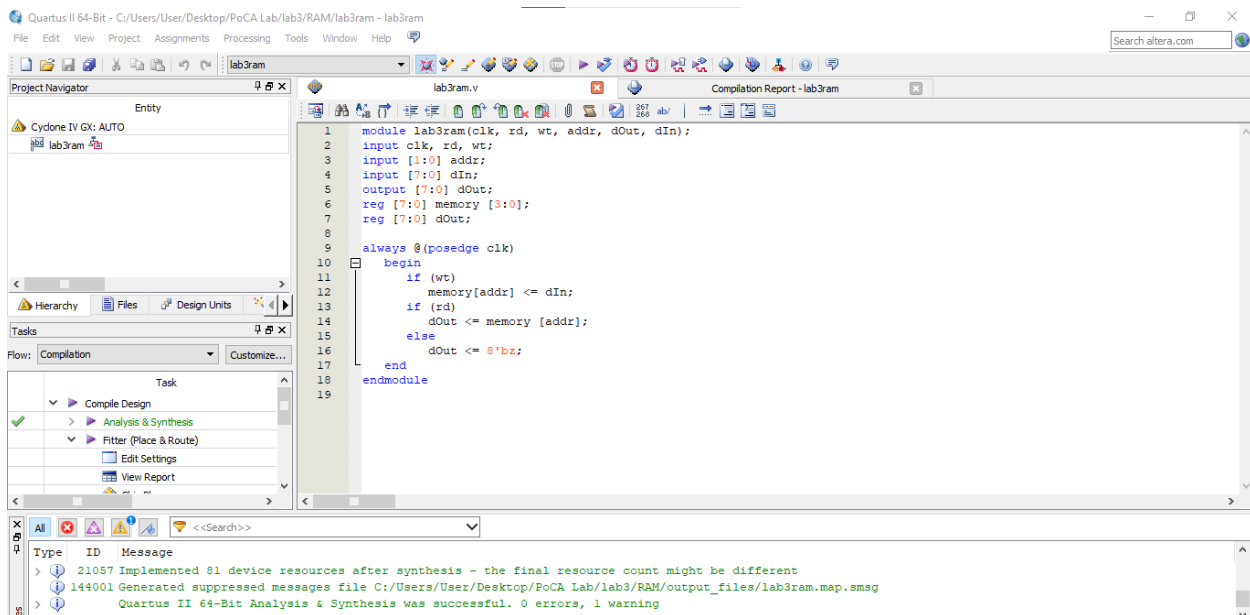


Figure 3: Verilog Code for RAM

Figure 3 shows that the compilation of the RAM module is successful.

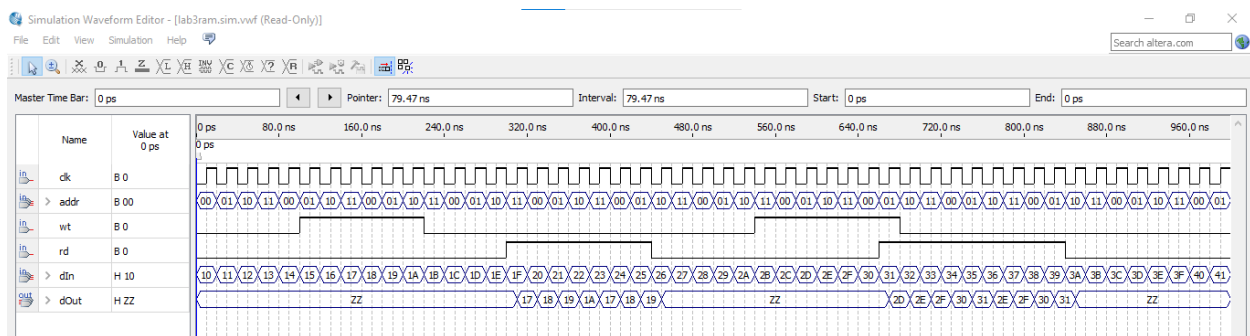


Figure 4: Simulation Result for RAM

Figure 4 shows the simulation result for the RAM module. For each cycle of positive edge of the clock (clk), the dOut will show the value that been write, wt and read, rd from dIn for the range of 2-bits only since the address input been assigned for 2-bits only. For example, the first dOut been write and read for the last 4 values which are 17, 18, 19, and 1A from the Din. Then the dOut will repeat the cycle of the 4 values until read (rd) been reached LOW. Otherwise, dOut will show "ZZ".

3. Design of control unit to transfer data from ROM to RAM

```
lab3fsm.v
Compilation Report - lab3fsm

1 module lab3fsm(clk, rst, addr, rom, ram, rd);
2   input clk, rst;
3   output rom, ram, rd;
4   output [1:0] addr;
5   parameter [2:0] s1 = 3'b000, s2 = 3'b001, s3 = 3'b010, s4 = 3'b011, s5 = 3'b100, s6 = 3'b101, s7 = 3'b110, s8 = 3'b111;
6   reg [2:0] pre, next;
7   reg [1:0] addr;
8   reg rom, ram, rd;
9
10  always @(pre)
11  begin
12    case (pre)
13    s1: begin
14      next <= s2;
15      addr <= 2'b00;
16      rom <= 1'b1;
17      ram <= 1'b1;
18      rd <= 1'b0;
19    end
20
21    s2: begin
22      next <= s3;
23      addr <= 2'b01;
24      rom <= 1'b1;
25      ram <= 1'b1;
26      rd <= 1'b0;
27    end
28
29    s3: begin
30      next <= s4;
31      addr <= 2'b10;
32      rom <= 1'b1;
33      ram <= 1'b1;
34      rd <= 1'b0;
35    end
36
37    s4: begin
38      next <= s5;
39      addr <= 2'b11;
40      rom <= 1'b1;
41      ram <= 1'b1;
42      rd <= 1'b0;
43    end
44
45    s5: begin
46      next <= s6;
47      addr <= 2'b00;
48      rom <= 1'b0;
49      ram <= 1'b0;
50      rd <= 1'b1;
51    end
52
53    s6: begin
54      next <= s7;
55
56      addr <= 2'b01;
57      rom <= 1'b0;
58      ram <= 1'b0;
59      rd <= 1'b1;
60    end
61
62    s7: begin
63      next <= s8;
64      addr <= 2'b01;
65      rom <= 1'b0;
66      ram <= 1'b0;
67      rd <= 1'b1;
68    end
69
70    s8: begin
71      next <= s1;
72      addr <= 2'b01;
73      rom <= 1'b0;
74      ram <= 1'b0;
75      rd <= 1'b1;
76    end
77  endcase
78  end
79
80  always @(posedge clk)
81  begin
82    if (rst == 0)
83      pre <= s1;
84  end
```

