NMJ31804 – Principles of Computer Architecture

SEMESTER 2, 2021/22

LAB 4) Design of Memory Module RAM and ROM

Name: TAN YI JIE

Matric Number: 191020976

Program code: RK20 - Computer Engineering

Task 1

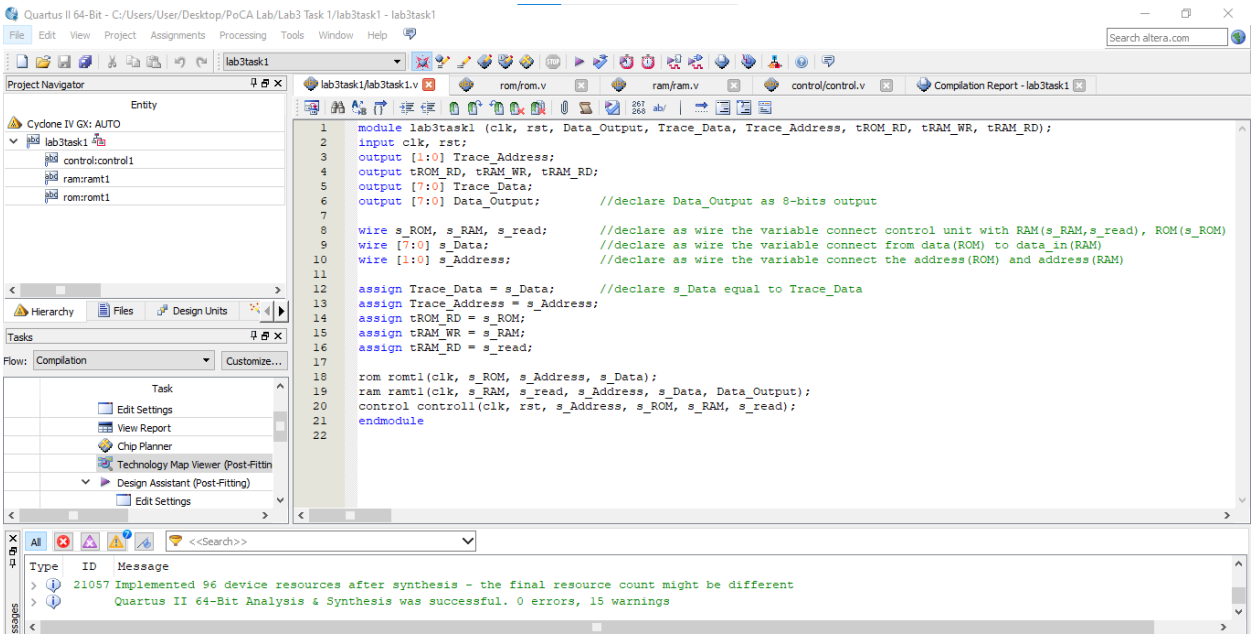

Figure 1: Verilog Code for lab3task1 Module

Figure 1 shows that the compilation of the lab3task1 module is successful. This module is the top level module which is the integration of all module.

Figure 2: Verilog Code for rom Module

Figure 2 shows the Verilog code for module of ROM.

```
module ram (clk, s_RAM, s_read, s_Address, s_Data, Data_Output);
input clk, s_RAM, s_read;
input [1:0] s_Address;
input [7:0] s_Data;
output [7:0] Data_Output;
reg [7:0] memory [3:0];
reg [7:0] Data_Output;

always @(posedge clk)
   begin
      if(s_RAM)
         memory[s_Address] <= s_Data;
      if(s_read)
         Data_Output <= memory[s_Address];
      else
         Data_Output <= 8'bz;
   end
endmodule
```

Figure 3: Verilog Code for ram Module

Figure 3 shows the Verilog code for module of RAM.

```
module control(clk, rst, s_Address, s_ROM, s_RAM, s_read);
input clk, rst;
output s_ROM, s_RAM, s_read;
output [1:0] s_Address;
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;
reg [2:0] present, next;
reg [1:0] s_Address;
reg [1:0] s_ROM, s_RAM, s_read;

always @(present)
   begin
      case(present)
      s1:   begin
            next <= s2;
            s_Address <= 2'b00;
            s_ROM <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end

      s2:   begin
            next <= s3;
            s_Address <= 2'b01;
            s_ROM <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end
```

```
      s3:   begin
            next <= s4;
            s_Address <= 2'b10;
            s_ROM <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end

      s4:   begin
            next <= s5;
            s_Address <= 2'b11;
            s_ROM <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end

      s5:   begin
            next <= s6;
            s_Address <= 2'b00;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end

      s6:   begin
            next <= s7;
            s_Address <= 2'b01;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end

      s7:   begin
            next <= s8;
            s_Address <= 2'b10;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end

      s8:   begin
            next <= s1;
            s_Address <= 2'b11;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end
      endcase
   end

   always @(posedge clk or negedge rst)
      begin
         if(rst == 0)
            present <= s1;
         else
            present <= next;
      end
endmodule
```

Figure 4: Verilog Code for control Module

Figure 4 shows the Verilog code for module of control unit which is the finite state machine.

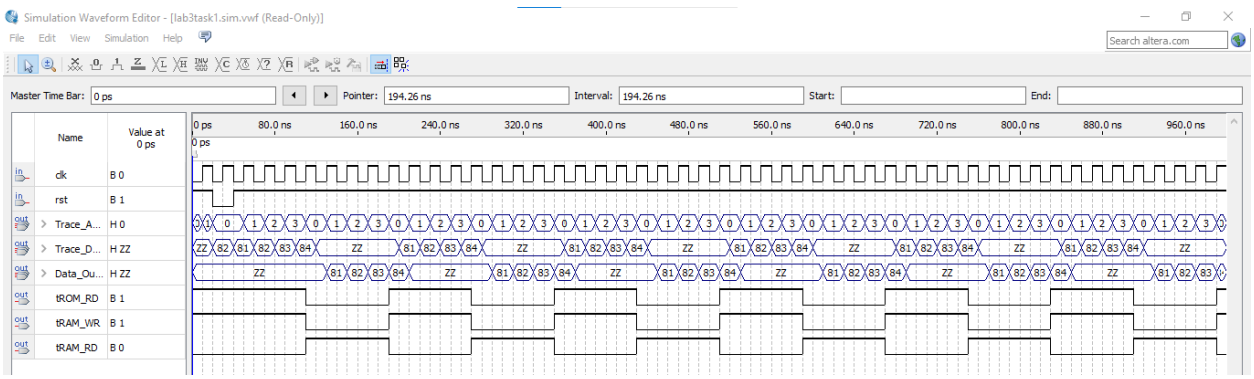

Figure 5: Simulation Result for lab3task1 Module

Figure 5 shows the simulation result for the lab3task1 module. All the output data for Trace_Address, Trace_Data, and Data_Output will have specific result in hexadecimal radix and the result will reflect it during each of the positive edge of the clock. During the second negative edge cycle of the clock, the system will read the value of Trace_Data from ROM and store in RAM, then it will read the RAM value after the sixth positive edge cycle of the clock such as 80, 81, 82, and 83. Otherwise, the output data for Trace_Data and Data_Output will show “ZZ”.

Task 2

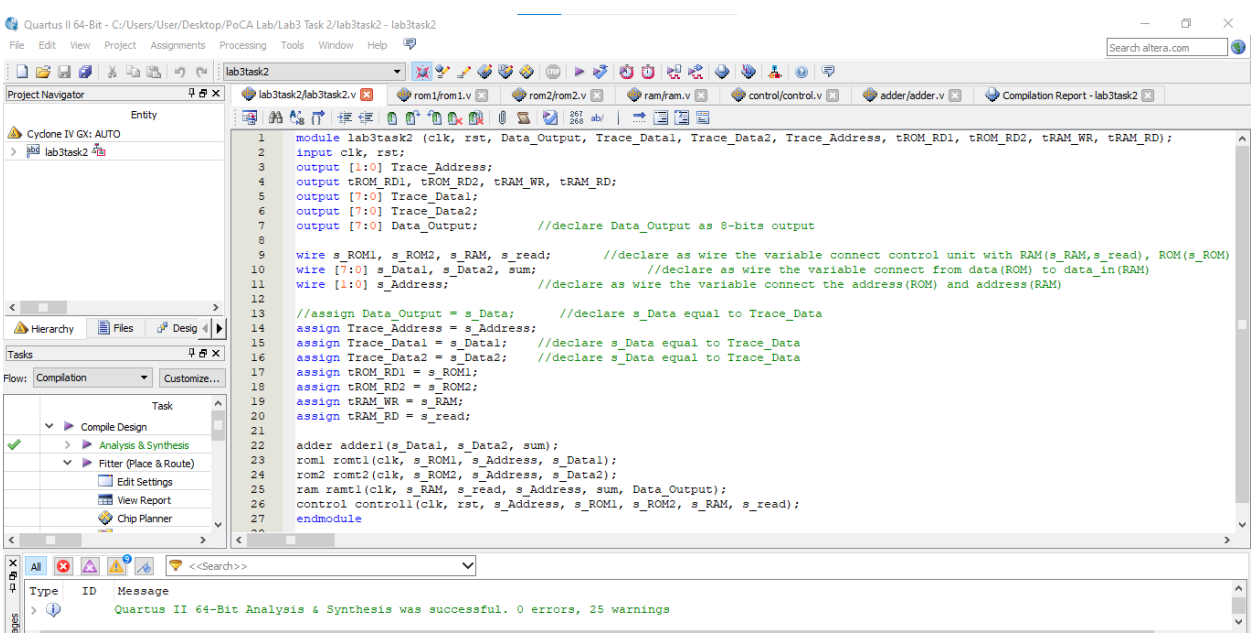

Figure 6: Verilog Code for lab3task2 Module

Figure 6 shows that the compilation of the lab3task2 module is successful. This module is the top level module which is the integration of all module.

```
module rom1(clk, s_ROM1, s_Address, s_Data1);
input clk, s_ROM1;
input [1:0] s_Address;
output [7:0] s_Data1;
reg [7:0] s_Data1;

always @(negedge clk)
   if (s_ROM1)
      begin
         case(s_Address)
         2'b00: s_Data1 <= 8'b01000000; //s_Data=40H
         2'b01: s_Data1 <= 8'b01000001; //s_Data=41H
         2'b10: s_Data1 <= 8'b01000010; //s_Data=42H
         2'b11: s_Data1 <= 8'b01000011; //s_Data=43H
         endcase
      end
   else
         s_Data1 <= 8'bz;
endmodule
```

Figure 7: Verilog Code for rom1 Module

Figure 7 shows the Verilog code for module of ROM1.

```
module rom2(clk, s_ROM2, s_Address, s_Data2);
input clk, s_ROM2;
input [1:0] s_Address;
output [7:0] s_Data2;
reg [7:0] s_Data2;

always @(negedge clk)
   if (s_ROM2)
      begin
         case(s_Address)
         2'b00: s_Data2 <= 8'b01100101; //s_Data=65H
         2'b01: s_Data2 <= 8'b01100110; //s_Data=66H
         2'b10: s_Data2 <= 8'b01100111; //s_Data=67H
         2'b11: s_Data2 <= 8'b01101000; //s_Data=68H
         endcase
      end
   else
         s_Data2 <= 8'bz;
endmodule
```

Figure 8: Verilog Code for rom2 Module

Figure 8 shows the Verilog code for module of ROM2.

```
module ram (clk, s_RAM, s_read, s_Address, s_Data, Data_Output);
input clk, s_RAM, s_read;
input [1:0] s_Address;
input [7:0] s_Data;
output [7:0] Data_Output;
reg [7:0] memory [3:0];
reg [7:0] Data_Output;

always @(posedge clk)
   begin
      if(s_RAM)
         memory[s_Address] <= s_Data;
      if(s_read)
         Data_Output <= memory[s_Address];
      else
         Data_Output <= 8'bz;
   end
endmodule
```

Figure 9: Verilog Code for ram Module

Figure 9 shows the Verilog code for module of RAM.

lab3task2/lab3task2.v | rom1/rom1.v | rom2/rom2.v | ram/ram.v | control/control.v | adder/adder.v | Compilation Report - lab3task2

```
module control(clk, rst, s_Address, s_ROM1, s_ROM2, s_RAM, s_read);
input clk, rst;
output s_ROM1, s_ROM2, s_RAM, s_read;
output [1:0] s_Address;
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;
reg [2:0] present, next;
reg [1:0] s_Address;
reg [1:0] s_ROM1, s_ROM2, s_RAM, s_read;

always @(present)
   begin
      case(present)
      s1:   begin
            next <= s2;
            s_Address <= 2'b00;
            s_ROM1 <= 1'b1;
            s_ROM2 <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end

      s2:   begin
            next <= s3;
            s_Address <= 2'b01;
            s_ROM1 <= 1'b1;
            s_ROM2 <= 1'b1;
            s_RAM <= 1'b1;

      s3:   begin
            next <= s4;
            s_Address <= 2'b10;
            s_ROM <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end

      s4:   begin
            next <= s5;
            s_Address <= 2'b11;
            s_ROM <= 1'b1;
            s_RAM <= 1'b1;
            s_read <= 1'b0;
            end

      s5:   begin
            next <= s6;
            s_Address <= 2'b00;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end

      s6:   begin
            next <= s7;

            s_Address <= 2'b01;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end

      s7:   begin
            next <= s8;
            s_Address <= 2'b10;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end

      s8:   begin
            next <= s1;
            s_Address <= 2'b11;
            s_ROM <= 1'b0;
            s_RAM <= 1'b0;
            s_read <= 1'b1;
            end
      endcase
   end

   always @(posedge clk or negedge rst)
      begin
         if(rst == 0)
```

```
            present <= s1;
        else
            present <= next;
    end
endmodule
```

Figure 10: Verilog Code for control Module

Figure 10 shows the Verilog code for module of control unit which is the finite state machine.

```
module adder(s_Data1, s_Data2, sum);

input [7:0] s_Data1;
input [7:0] s_Data2;
output [7:0] sum;

assign sum = s_Data1 + s_Data2;

endmodule
```

Figure 11: Verilog Code for adder Module

Figure 11 shows the Verilog code for module of adder.

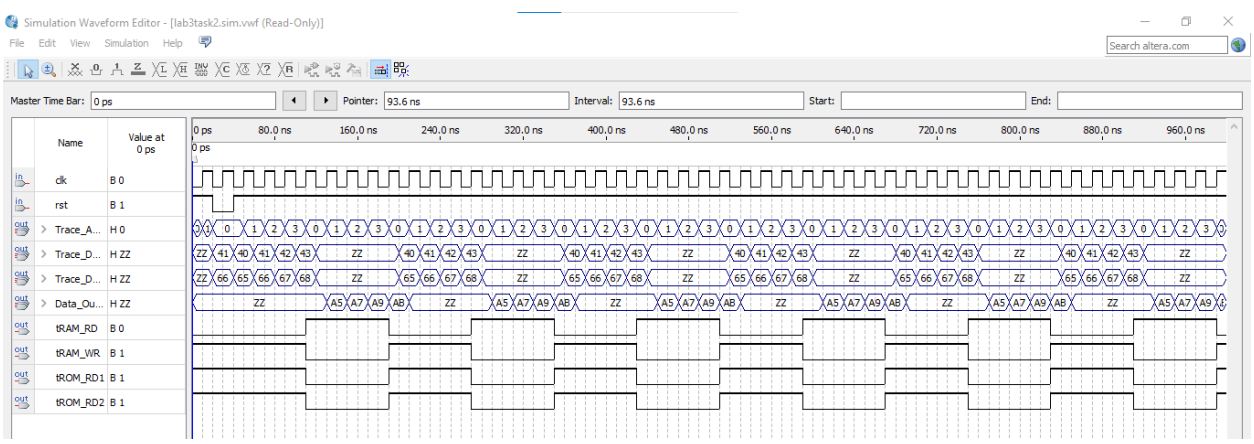

Figure 12: Simulation Result for lab3task2 Module

Figure 12 shows the simulation result for the lab3task2 module. All the output data for Trace_Address, Trace_Data1, Trace_Data2, and Data_Output will have specific result in hexadecimal radix and the result will reflect it during each of the negative edge of the clock. During the third negative edge cycle of the clock, the system will read the value of Trace_Data1 from ROM1 and Trace_Data2 from ROM2 then under the operation of adder and the result will store in RAM, then it will read the RAM value after the seventh positive edge cycle of the clock such as A5, A7, A9, and AB. Otherwise, the output data for Trace_Data1, Trace_Data2, and Data_Output will show “ZZ”.

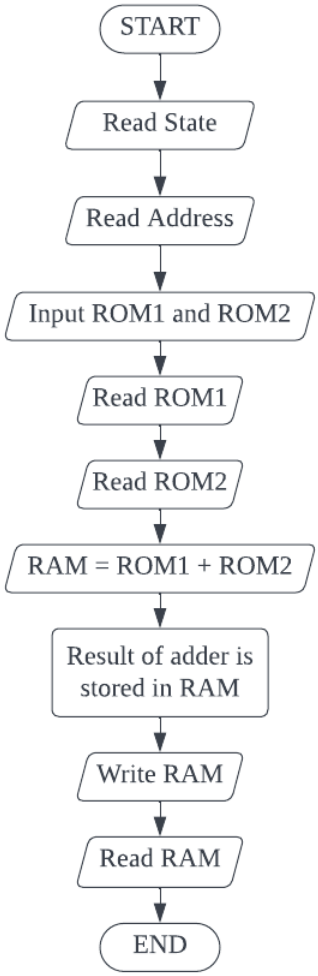

Figure 13: Flowchart of the system

Figure 13 shows the figure of the system for lab3task2 module.