第九次實驗報告

題目:9-1~9-5

姓名:羅名志

學號:0813228

繳交日期:2022/5/13

一、9-1, 9-2

(1)實驗程式碼:

主程式碼:

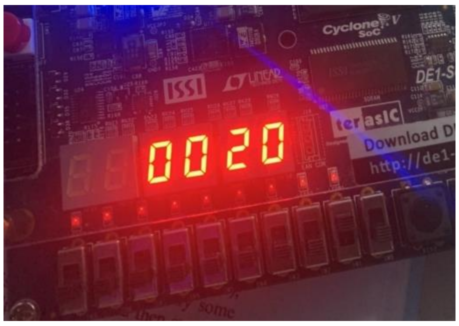
```
module labnine2(SW, KEY, LEDR, HEX3, HEX2, HEX1, HEX0);
	input [9:0]SW;
	input [0:0]KEY;
	output [0:0]LEDR;
	output [6:0]HEX3;
	output [6:0]HEX2;
	output [6:0]HEX1;
	output [6:0]HEX0;
	wire [3:0]out;
	wire [3:0]fake;


	test2(SW[8:4],~KEY[0], SW[3:0], SW[9], out);


	assign LEDR[0]=SW[9];
	assign fake[0]=SW[8];
	assign fake[3:1]=3'b000;
	display7seg seg1(SW[3:0],HEX1);
	display7seg seg2(out[3:0],HEX0);
	display7seg seg3(SW[7:4],HEX2);
	display7seg seg4(fake[3:0],HEX3);
endmodule

module display7seg(num,HEX);
	input [3:0]num;
	output reg[6:0]HEX;
	always@(num) begin
		case(num)
		0:HEX=7'b1000000;
		1:HEX=7'b1111001;
		2:HEX=7'b0100100;
		3:HEX=7'b0110000;
		4:HEX=7'b0011001;
```

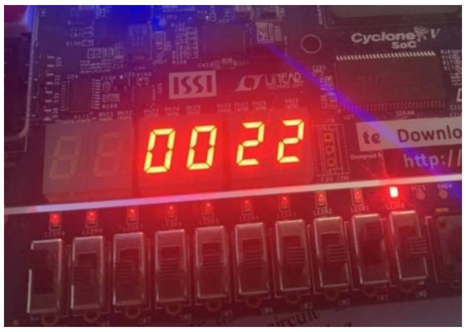
```
            5:HEX=7'b0010010;
            6:HEX=7'b0000010;
            7:HEX=7'b1111000;

            8:HEX=7'b0000000;
            9:HEX=7'b0010000;
            10:HEX=7'b0001000;
            11:HEX=7'b0000011;
            12:HEX=7'b1000110;
            13:HEX=7'b0100001;
            14:HEX=7'b0000110;
            15:HEX=7'b0001110;
            endcase
      end
endmodule
```

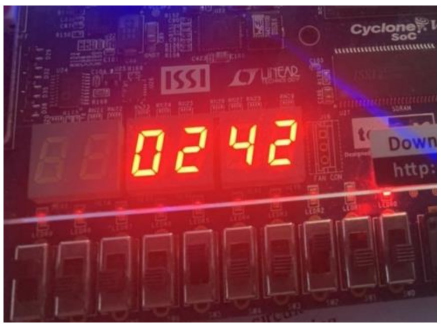
RAM 1-PORT:

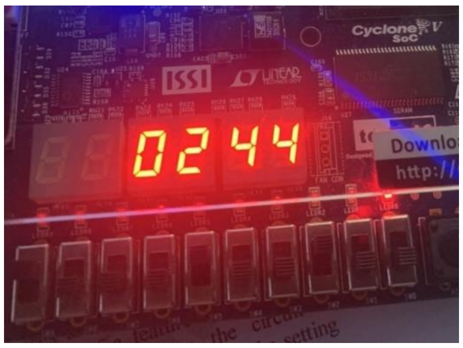
```
module test2 (
      address,
      clock,
      data,
      wren,
      q);

      input     [4:0]   address;
      input             clock;
      input     [3:0]   data;
      input             wren;
      output    [3:0]   q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
      tri1      clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

      wire [3:0] sub_wire0;
wire [3:0] q = sub_wire0[3:0];

altsyncram      altsyncram_component (
                .address_a (address),
                .clock0 (clock),
                .data_a (data),
                .wren_a (wren),
                .q_a (sub_wire0),
                .aclr0 (1'b0),
                .aclr1 (1'b0),
                .address_b (1'b1),
                .addressstall_a (1'b0),
                .addressstall_b (1'b0),
                .byteena_a (1'b1),
                .byteena_b (1'b1),
                .clock1 (1'b1),
                .clocken0 (1'b1),
                .clocken1 (1'b1),
                .clocken2 (1'b1),
                .clocken3 (1'b1),
                .data_b (1'b1),
                .eccstatus (),
                .q_b (),
                .rden_a (1'b1),
                .rden_b (1'b1),
                .wren_b (1'b0));
defparam
     altsyncram_component.clock_enable_input_a = "BYPASS",
     altsyncram_component.clock_enable_output_a = "BYPASS",
     altsyncram_component.intended_device_family = "Cyclone V",
     altsyncram_component.lpm_hint = "ENABLE_RUNTIME_MOD=NO",
     altsyncram_component.lpm_type = "altsyncram",
     altsyncram_component.numwords_a = 32,
     altsyncram_component.operation_mode = "SINGLE_PORT",
     altsyncram_component.outdata_aclr_a = "NONE",
     altsyncram_component.outdata_reg_a = "UNREGISTERED",
     altsyncram_component.power_up_uninitialized = "FALSE",
     altsyncram_component.read_during_write_mode_port_a =
"NEW_DATA_NO_NBE_READ",
            altsyncram_component.widthad_a = 5,
            altsyncram_component.width_a = 4,
            altsyncram_component.width_byteena_a = 1;


endmodule
```

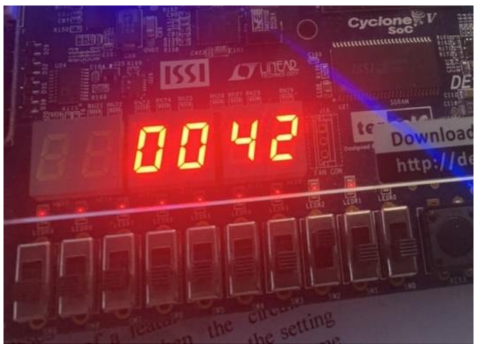
(2)實驗結果:

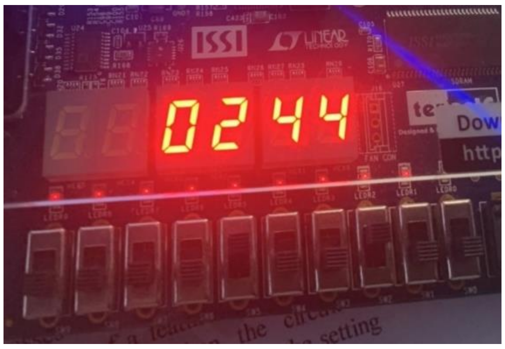






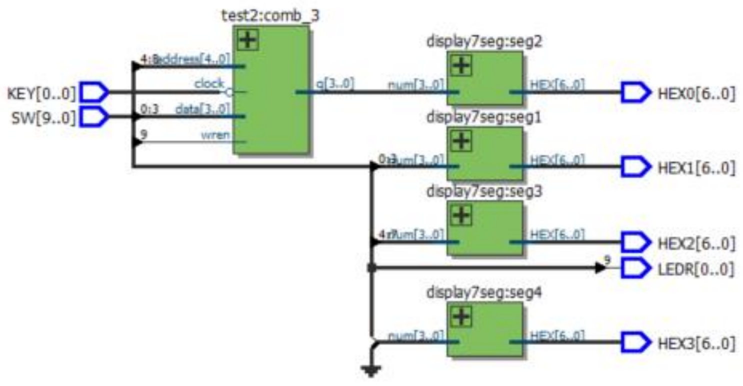






//1. SW[9]=0 / address=00 / data input=2 / data out=0
//2. SW[9]=1 / address=00 / data input=2 / data out=2(按下 KEY[0])
//3. SW[9]=1 / address=02 / data input=4 / data out=2
//4. SW[9]=1 / address=02 / data input=4 / data out=4(按下 KEY[0])
//5. SW[9]=0 / address=00 / data input=4 / data out=2(按下 KEY[0])
//6. SW[9]=0 / address=02 / data input=4 / data out=4(按下 KEY[0])

(3)RTL 布局



(4)問題與討論:
第一部分與第二部分只利用 RAM 1-port 寫入讀出即可，最後將結果顯示在七段顯示器上，算是這次 lab 最簡單的部分，但寫入讀出需分別在不同 clock 進行。

二、9-3

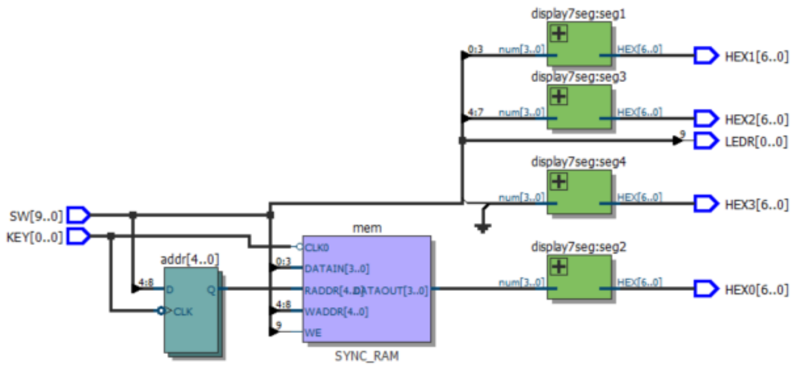
(1)實驗程式碼:

```
module labnine3(SW, KEY, LEDR, HEX3, HEX2, HEX1, HEX0);
	input [9:0]SW;
	input [0:0]KEY;
	output [0:0]LEDR;
	output [6:0]HEX3;
	output [6:0]HEX2;
	output [6:0]HEX1;
	output [6:0]HEX0;
	wire [3:0]out;
	wire [3:0]in;
	wire [3:0]fake;
	reg [4:0]addr;

	reg[3:0] mem[31:0];

	always@(negedge KEY[0]) begin
		addr[4:0]=SW[8:4];
		if(SW[9]==1) begin
			mem[addr]=SW[3:0];
		end
	end

	assign out=mem[addr];
	assign LEDR[0]=SW[9];
	assign fake[0]=SW[8];
	assign fake[3:1]=3'b000;
	display7seg seg1(SW[3:0],HEX1);
	display7seg seg2(out[3:0],HEX0);
	display7seg seg3(SW[7:4],HEX2);
	display7seg seg4(fake[3:0],HEX3);
endmodule

module display7seg(num,HEX);
	input [3:0]num;
	output reg[6:0]HEX;
	always@(num) begin
```

```
			case(num)
			0:HEX=7'b1000000;
			1:HEX=7'b1111001;
			2:HEX=7'b0100100;
			3:HEX=7'b0110000;
			4:HEX=7'b0011001;
			5:HEX=7'b0010010;
			6:HEX=7'b0000010;
			7:HEX=7'b1111000;

			8:HEX=7'b0000000;
			9:HEX=7'b0010000;
			10:HEX=7'b0001000;
			11:HEX=7'b0000011;
			12:HEX=7'b1000110;
			13:HEX=7'b0100001;
			14:HEX=7'b0000110;
			15:HEX=7'b0001110;
			endcase
		end
endmodule
```

(2)實驗結果:
同 part 2

(3)RTL 布局



(4)問題與討論:

用 always 達到跟 part2 一樣結果，只要按下 KEY[0]時判斷 SW[9]狀態，若為 1 則寫入 memory 並更新 address，若為 0 則單純更新 address，最後 assign 結果為 memory 中 address 的值並顯示出來，always 部分想了一段時間，但整體上沒有遇到太大的困難。

三、9-4

(1)實驗程式碼:

主程式碼:

```
module labnine4(CLOCK_50,SW,KEY,LEDR,HEX3,HEX2,HEX0);
        input CLOCK_50;
        input [0:0]KEY;
        input[9:0] SW;
        output [0:0]LEDR;
        output [6:0]HEX0;
        output [6:0]HEX2;
        output [6:0]HEX3;

        wire [3:0]out;
        reg [4:0]addr;
        reg [30:0]count;
        reg temp;
        wire [3:0]fake;
        reg [4:0]rec;

        test2(temp,SW[8:5],addr[4:0],SW[4:0],SW[9],out);

        always@(posedge CLOCK_50) begin

        if(~KEY[0]) begin
                count=0;
                temp=0;
                addr=0;
        end
        if(count==50000000) begin
                temp=1;
                count=0;
                rec[4:0]=addr[4:0];
                if(addr==5'b11111)begin
                        addr=0;
                end
                else begin
                        addr=addr+1;
```

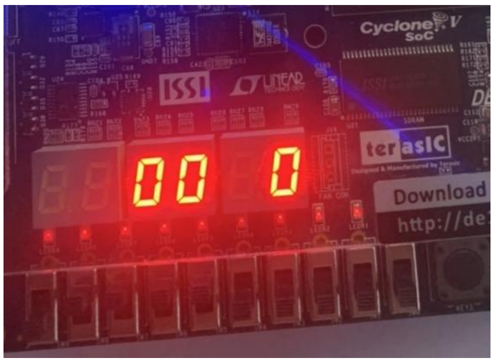
```
			end
		end
		else begin
			count=count+1;
			temp=0;
		end
		end

		assign LEDR[0]=SW[9];
		assign fake[0]=rec[4];
		assign fake[3:1]=3'b000;

		display7seg a(out[3:0],HEX0);
		display7seg b(rec[3:0],HEX2);
		display7seg c(fake[3:0],HEX3);

endmodule

module display7seg(num,HEX);
		input [3:0]num;
		output reg[6:0]HEX;
		always@(num) begin
			case(num)
			0:HEX=7'b1000000;
			1:HEX=7'b1111001;
			2:HEX=7'b0100100;
			3:HEX=7'b0110000;
			4:HEX=7'b0011001;
			5:HEX=7'b0010010;
			6:HEX=7'b0000010;
			7:HEX=7'b1111000;

			8:HEX=7'b0000000;
			9:HEX=7'b0010000;
			10:HEX=7'b0001000;
			11:HEX=7'b0000011;
			12:HEX=7'b1000110;
			13:HEX=7'b0100001;
```

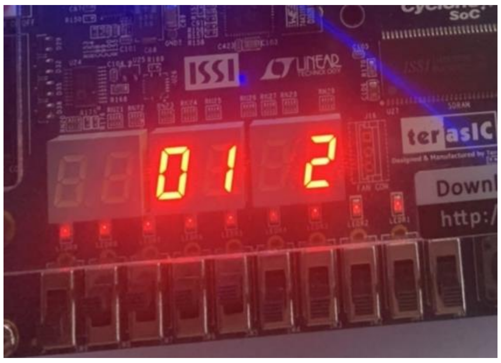
```
				14:HEX=7'b0000110;
				15:HEX=7'b0001110;
				endcase
		end
endmodule
```

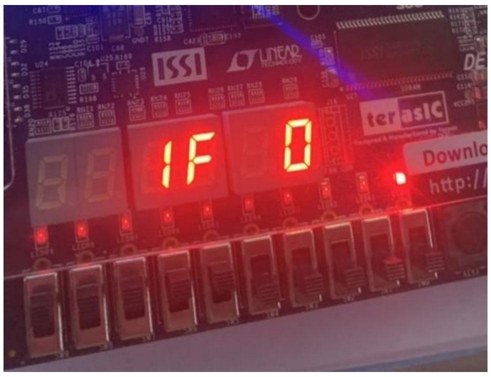
RAM 2-PORT:

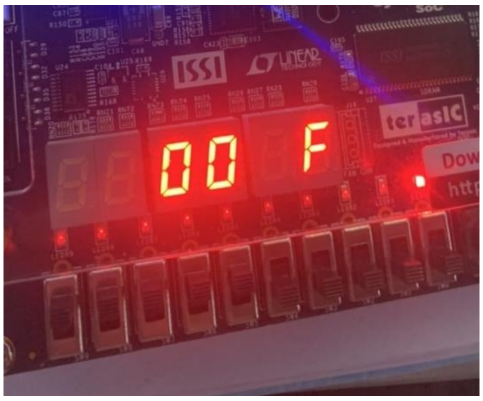
```
module test2 (
		clock,
		data,
		rdaddress,
		wraddress,
		wren,
		q);

		input			clock;
		input		[3:0]	data;
		input		[4:0]	rdaddress;
		input		[4:0]	wraddress;
		input			wren;
		output		[3:0]	q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
		tri1		clock;
		tri0		wren;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

		wire [3:0] sub_wire0;
		wire [3:0] q = sub_wire0[3:0];

		altsyncram	altsyncram_component (
					.address_a (wraddress),
					.clock0 (clock),
					.data_a (data),
					.wren_a (wren),
                        .address_b (rdaddress),
                        .q_b (sub_wire0),
                        .aclr0 (1'b0),
                        .aclr1 (1'b0),
                        .addressstall_a (1'b0),
                        .addressstall_b (1'b0),
                        .byteena_a (1'b1),
                        .byteena_b (1'b1),
                        .clock1 (1'b1),
                        .clocken0 (1'b1),
                        .clocken1 (1'b1),
                        .clocken2 (1'b1),
                        .clocken3 (1'b1),
                        .data_b ({4{1'b1}}),
                        .eccstatus (),
                        .q_a (),
                        .rden_a (1'b1),
                        .rden_b (1'b1),
                        .wren_b (1'b0));
    defparam
        altsyncram_component.address_aclr_b = "NONE",
        altsyncram_component.address_reg_b = "CLOCK0",
        altsyncram_component.clock_enable_input_a = "BYPASS",
        altsyncram_component.clock_enable_input_b = "BYPASS",
        altsyncram_component.clock_enable_output_b = "BYPASS",
        altsyncram_component.init_file = "hi.mif",
        altsyncram_component.intended_device_family = "Cyclone V",
        altsyncram_component.lpm_type = "altsyncram",
        altsyncram_component.numwords_a = 32,
        altsyncram_component.numwords_b = 32,
        altsyncram_component.operation_mode = "DUAL_PORT",
        altsyncram_component.outdata_aclr_b = "NONE",
        altsyncram_component.outdata_reg_b = "CLOCK0",
        altsyncram_component.power_up_uninitialized = "FALSE",
        altsyncram_component.read_during_write_mode_mixed_ports =
"DONT_CARE",
        altsyncram_component.widthad_a = 5,
        altsyncram_component.widthad_b = 5,
        altsyncram_component.width_a = 4,
        altsyncram_component.width_b = 4,
        altsyncram_component.width_byteena_a = 1;


endmodule
```

(2)實驗結果:



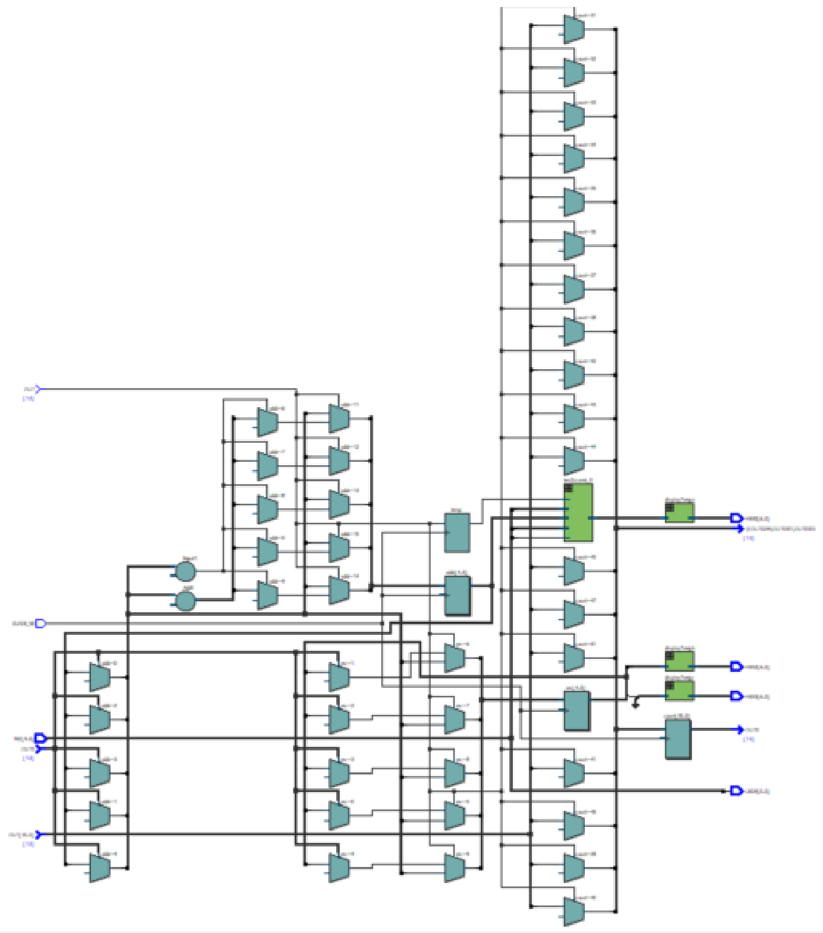






//address=0 input F

(3)RTL 布局



(4)問題與討論::

直接利用 RAM 2-port 操作 read/write 指令即可，但問題較多的是每一秒都要更新，因此需要在達到一秒時將 clock 值從 0 設成 1，並將 address +1，但顯示時需顯示尚未+1 的 address，因為送入 2-port module 的 address 是尚未加 1 的 address，因此須維持 address 跟 output 一致必須於一開始多等一秒(想不到更好的方法能解決一開始抓不到 address0 的值的問題)。

四、9-5

(1)實驗程式碼

主程式碼:

```
module labnine5(CLOCK_50,SW,KEY,LEDR,HEX3,HEX2,HEX0);
        input CLOCK_50;
        input [0:0]KEY;
        input[9:0] SW;
        output [0:0]LEDR;
        output [6:0]HEX0;
        output [6:0]HEX2;
        output [6:0]HEX3;

        wire [3:0]out;
        reg [4:0]addr,address;
        reg [30:0]count;
        reg temp;
        wire [3:0]fake;
        reg [4:0]rec;
        reg [3:0]change,ans;


        test3(address[4:0],temp,SW[8:5],change,out);

        always@(posedge CLOCK_50) begin

        if(~KEY[0]) begin
                count=0;
                temp=0;
                addr=0;
                address=0;
                temp=1;
        end


        if(count==50000000) begin

                count=0;
                ans=out;
                rec[4:0]=addr[4:0];
                if(addr==5'b11111)begin
                        addr=0;
                end
                else begin
                        addr=addr+1;
                end
        end

        //前  1 step  先更新
        else if(count==49999999) begin
                change=0;
                temp=0;
                address=addr;
                temp=1;
                count=count+1;
        end

        //input
        else if(count==25000000 & SW[9]==1)begin
                change=1;
                temp=1;
                count=count+1;
                address=SW[4:0];
        end
        else begin
                change=0;
                count=count+1;
                temp=0;
        end
        end

        assign LEDR[0]=SW[9];
        assign fake[0]=rec[4];
        assign fake[3:1]=3'b000;

        display7seg a(ans[3:0],HEX0);
```

```
        display7seg b(rec[3:0],HEX2);
        display7seg c(fake[3:0],HEX3);

endmodule

module display7seg(num,HEX);
        input [3:0]num;
        output reg[6:0]HEX;
        always@(num) begin
                case(num)
                0:HEX=7'b1000000;
                1:HEX=7'b1111001;
                2:HEX=7'b0100100;
                3:HEX=7'b0110000;
                4:HEX=7'b0011001;
                5:HEX=7'b0010010;
                6:HEX=7'b0000010;
                7:HEX=7'b1111000;

                8:HEX=7'b0000000;
                9:HEX=7'b0010000;
                10:HEX=7'b0001000;
                11:HEX=7'b0000011;
                12:HEX=7'b1000110;
                13:HEX=7'b0100001;
                14:HEX=7'b0000110;
                15:HEX=7'b0001110;
                endcase
        end
endmodule
```

RAM 1-PORT:

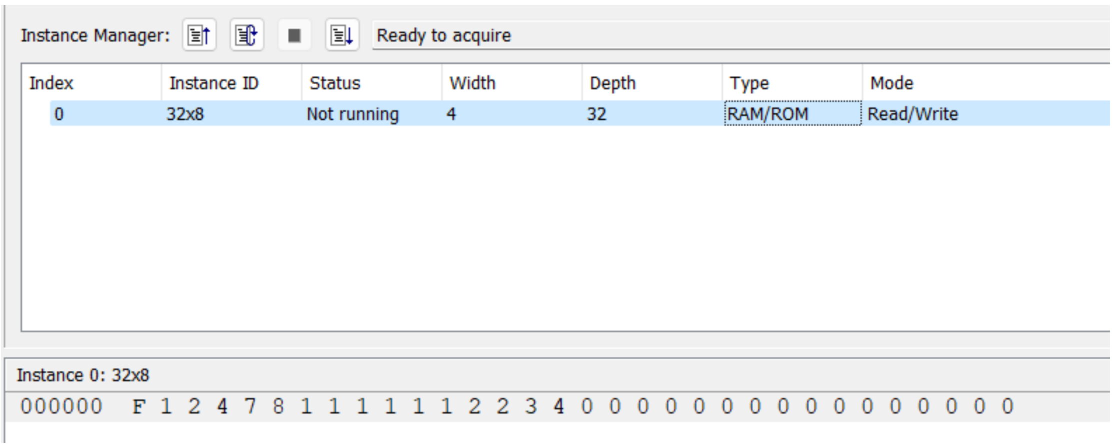
```
module test3 (
        address,
        clock,
        data,
        wren,
        q);
```

```
	input	[4:0]	address;
	input	  clock;
	input	[3:0]	data;
	input	  wren;
	output	[3:0]	q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [3:0] sub_wire0;
	wire [3:0] q = sub_wire0[3:0];

	altsyncram	altsyncram_component (
				.address_a (address),
				.clock0 (clock),
				.data_a (data),
				.wren_a (wren),
				.q_a (sub_wire0),
				.aclr0 (1'b0),
				.aclr1 (1'b0),
				.address_b (1'b1),
				.addressstall_a (1'b0),
				.addressstall_b (1'b0),
				.byteena_a (1'b1),
				.byteena_b (1'b1),
				.clock1 (1'b1),
				.clocken0 (1'b1),
				.clocken1 (1'b1),
				.clocken2 (1'b1),
				.clocken3 (1'b1),
				.data_b (1'b1),
				.eccstatus (),
				.q_b (),
                        .rden_a (1'b1),
                        .rden_b (1'b1),
                        .wren_b (1'b0));
        defparam
                altsyncram_component.clock_enable_input_a = "BYPASS",
                altsyncram_component.clock_enable_output_a = "BYPASS",
                altsyncram_component.init_file = "../labnine5/ram.mif",
                altsyncram_component.intended_device_family = "Cyclone V",
                altsyncram_component.lpm_hint =
"ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=32*8",
                altsyncram_component.lpm_type = "altsyncram",
                altsyncram_component.numwords_a = 32,
                altsyncram_component.operation_mode = "SINGLE_PORT",
                altsyncram_component.outdata_aclr_a = "NONE",
                altsyncram_component.outdata_reg_a = "UNREGISTERED",
                altsyncram_component.power_up_uninitialized = "FALSE",
                altsyncram_component.read_during_write_mode_port_a =
"NEW_DATA_NO_NBE_READ",
                altsyncram_component.widthad_a = 5,
                altsyncram_component.width_a = 4,
                altsyncram_component.width_byteena_a = 1;


endmodule
```

(2)實驗結果:

同 part4;

Read data from in-system memory:



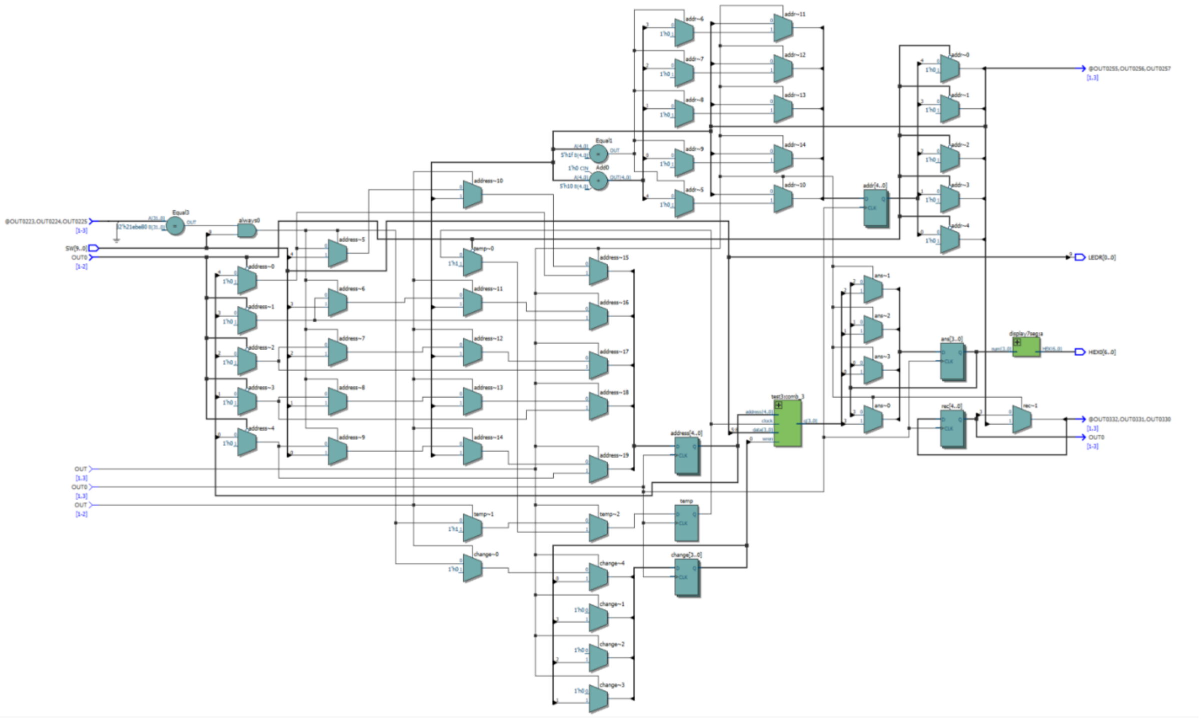
Instance Manager: Ready to acquire

Index	Instance ID	Status	Width	Depth	Type	Mode
0	32x8	Not running	4	32	RAM/ROM	Read/Write

Instance 0: 32x8

```
000000  F 1 2 4 7 8 1 1 1 1 1 1 2 2 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(3)RTL 布局



(4)問題與討論:

我的作法是在這一秒鐘的一半時判斷是否要寫入值，若要寫入即將 clock 從 0 設成 1 後使用 RAM 1-port 寫入，此時不將 output 值顯示出來，而在該瞬間過後則將寫入的訊號(SW[9])設為 0，最後在即將一秒時先更新 address 值避免 module 跑出的值為上一秒計算的值(上個 address 值)，最後利用 reg 把結果存起來並顯示。

與同學討論後發現我的作法有許多不好的地方，像是應該在一秒整時計算結果會更好等，因此這部分是我需要再努力改進的地方。