第六次實驗報告

題目：

Simple processor including memory

姓名：黃子軒

學號：310581003

繳交日期：**111** 年 **11** 月 **28** 日

1. 實驗目的

透過本次實驗，除了熟悉如何撰寫 verilog 並實現於 FPGA 板，並試著結合 memory 實現一簡易的 processor。

2. 實驗程式碼

(1) 主要程式

```
module Lab6(KEY,HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,LEDR);


 input [2:0]    KEY;

 output [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5;

 output [9:0] LEDR;


 wire Resetn, MClock, PClock;

 assign Resetn = KEY[0];

 assign MClock = KEY[1];

 assign PClock = KEY[2];


 reg [7:0] BusWires;

 assign LEDR[7:0] = BusWires;

 reg [7:0] R0, R1, R2;

 wire [7:0] DIN;


 parameter MV    = 2'b00;

 parameter MVI = 2'b01;

 parameter ADD = 2'b10;

 parameter SUB = 2'b11;


 reg [4:0] address;
```

```
wire [1:0] IR_current;
wire [3:0] Xreg_current, Yreg_current;
assign IR_current = DIN[7:6];
assign Xreg_current = DIN[5:3];
assign Yreg_current = DIN[2:0];

reg [1:0] IR_past;
reg [3:0] Xreg_past, Yreg_past;

always@(posedge PClock)
begin
	IR_past <= DIN[7:6];
	Xreg_past <= DIN[5:3];
	Yreg_past <= DIN[2:0];
end

// counter //
always@(posedge MClock, negedge Resetn)
begin
	if(!Resetn)
		address <= 0;
	else
		address <= address + 1;
end

romlpm rom(address, MClock, DIN);

seven_segment s1(DIN[7:4],HEX5);
```

```
seven_segment s2(DIN[3:0],HEX4);
seven_segment s3(R0[7:4],HEX3);
seven_segment s4(R0[3:0],HEX2);
seven_segment s5(R1[7:4],HEX1);
seven_segment s6(R1[3:0],HEX0);

// proc //
always@(posedge PClock, negedge Resetn)
begin
	if(!Resetn)
	begin
		R0 <= 0;
		R1 <= 0;
		BusWires <= 0;
	end
	else
	begin
		BusWires <= DIN;

		if(address != 0)
		begin
			case(IR_current)
				MV:
				begin
					if(IR_past!=MVI)
					begin
						if(Xreg_current == 0)
						begin
```

```
            if(Yreg_current == 1)
            begin
                R0 <= R1;
            end
            else
            begin
                R0 <= R2;
            end
        end
        else if(Xreg_current == 1)
        begin
            if(Yreg_current == 0)
                R1 <= R0;
            else
                R1 <= R2;
        end
        else
        begin
            if(Yreg_current == 0)
                R2 <= R0;
            else
                R2 <= R1;
        end
    end
    else if(IR_past==MVI)
    begin
        if(Xreg_past == 0)
            R0 <= DIN;
```

```
                    else if(Xreg_past == 1)
                        R1 <= DIN;
                    else
                        R2 <= DIN;
                end
            end

            MVI: ;

            ADD:R0 <= R0 + R1;

            SUB:
                R0 <= R0 - R1;

            default:
            begin
                R0 <= 0;
                R1 <= 0;
            end
        endcase
    end
    else begin
        R0 <= 0;
        R1 <= 0;
    end
  end
end
```

```
endmodule


// Seven-segment desplay //
module seven_segment(in,HEX);
  input [3:0] in;
  output wire [6:0] HEX;

  assign HEX[0] =
~in[3]&in[2]&~in[0]|~in[3]&~in[2]&~in[1]&in[0]|in[3]&in[2]&~in[1]&in[0]|in[3]&~in[2]&in[1]&in[0]
;
  assign HEX[1] =
~in[3]&in[2]&~in[1]&in[0]|in[3]&in[2]&in[1]|in[3]&~in[0]&in[2]|in[2]&in[1]&~in[0]|in[3]&in[1]&in[
0];
  assign HEX[2] = in[3]&in[2]&~in[0]|in[3]&in[2]&in[1]|~in[3]&~in[2]&in[1]&~in[0];
  assign HEX[3] =
in[3]&~in[2]&in[1]&~in[0]|~in[3]&in[2]&~in[1]&~in[0]|~in[2]&~in[1]&in[0]|in[2]&in[1]&in[0];
  assign HEX[4] = ~in[2]&~in[1]&in[0]|~in[3]&in[0]|~in[3]&in[2]&~in[1];
  assign HEX[5] =
~in[3]&~in[2]&in[0]|~in[3]&~in[2]&in[1]|~in[3]&in[1]&in[0]|in[3]&in[2]&~in[1]&in[0];
  assign HEX[6] = ~in[3]&in[2]&in[1]&in[0]|in[3]&in[2]&~in[1]&~in[0]|~in[3]&~in[2]&~in[1];
endmodule
```

(2) RAM

```
// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: romlpm.v
// Megafunction Name(s):
//                     altsyncram
```

```
//
// Simulation Library Files(s):
//                    altera_mf
// ============================================================
// ************************************************************
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 13.1.0 Build 162 10/23/2013 SJ Full Version
// ************************************************************
```



```
// synopsys translate_off
`timescale 1 ps / 1 ps
```

```
// synopsys translate_on
module romlpm (
  address,
  clock,
  q);

  input    [4:0]  address;
  input           clock;
  output   [7:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
  tri1     clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

  wire [7:0] sub_wire0;
  wire [7:0] q = sub_wire0[7:0];

  altsyncram  altsyncram_component (
              .address_a (address),
              .clock0 (clock),
              .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_a ({8{1'b1}}),
                .data_b (1'b1),
                .eccstatus (),
                .q_b (),
                .rden_a (1'b1),
                .rden_b (1'b1),
                .wren_a (1'b0),
                .wren_b (1'b0));
defparam
        altsyncram_component.address_aclr_a = "NONE",
        altsyncram_component.clock_enable_input_a = "BYPASS",
        altsyncram_component.clock_enable_output_a = "BYPASS",
        altsyncram_component.init_file = "lpmrom.mif",
        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 = "ROM",
        altsyncram_component.outdata_aclr_a = "NONE",
        altsyncram_component.outdata_reg_a = "UNREGISTERED",
```

```
        altsyncram_component.ram_block_type = "M10K",
        altsyncram_component.widthad_a = 5,
        altsyncram_component.width_a = 8,
        altsyncram_component.width_byteena_a = 1;


endmodule

// ============================================================
// CNX file retrieval info
// ============================================================
// Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC "0"
// Retrieval info: PRIVATE: AclrAddr NUMERIC "0"
// Retrieval info: PRIVATE: AclrByte NUMERIC "0"
// Retrieval info: PRIVATE: AclrOutput NUMERIC "0"
// Retrieval info: PRIVATE: BYTE_ENABLE NUMERIC "0"
// Retrieval info: PRIVATE: BYTE_SIZE NUMERIC "8"
// Retrieval info: PRIVATE: BlankMemory NUMERIC "0"
// Retrieval info: PRIVATE: CLOCK_ENABLE_INPUT_A NUMERIC "0"
// Retrieval info: PRIVATE: CLOCK_ENABLE_OUTPUT_A NUMERIC "0"
// Retrieval info: PRIVATE: Clken NUMERIC "0"
// Retrieval info: PRIVATE: IMPLEMENT_IN_LES NUMERIC "0"
// Retrieval info: PRIVATE: INIT_FILE_LAYOUT STRING "PORT_A"
// Retrieval info: PRIVATE: INIT_TO_SIM_X NUMERIC "0"
// Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone V"
// Retrieval info: PRIVATE: JTAG_ENABLED NUMERIC "0"
// Retrieval info: PRIVATE: JTAG_ID STRING "NONE"
// Retrieval info: PRIVATE: MAXIMUM_DEPTH NUMERIC "0"
```

// Retrieval info: PRIVATE: MIFfilename STRING "lpmrom.mif"

// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "32"

// Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "2"

// Retrieval info: PRIVATE: RegAddr NUMERIC "1"

// Retrieval info: PRIVATE: RegOutput NUMERIC "0"

// Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"

// Retrieval info: PRIVATE: SingleClock NUMERIC "1"

// Retrieval info: PRIVATE: UseDQRAM NUMERIC "0"

// Retrieval info: PRIVATE: WidthAddr NUMERIC "5"

// Retrieval info: PRIVATE: WidthData NUMERIC "8"

// Retrieval info: PRIVATE: rden NUMERIC "0"

// Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all

// Retrieval info: CONSTANT: ADDRESS_ACLR_A STRING "NONE"

// Retrieval info: CONSTANT: CLOCK_ENABLE_INPUT_A STRING "BYPASS"

// Retrieval info: CONSTANT: CLOCK_ENABLE_OUTPUT_A STRING "BYPASS"

// Retrieval info: CONSTANT: INIT_FILE STRING "lpmrom.mif"

// Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone V"

// Retrieval info: CONSTANT: LPM_HINT STRING "ENABLE_RUNTIME_MOD=NO"

// Retrieval info: CONSTANT: LPM_TYPE STRING "altsyncram"

// Retrieval info: CONSTANT: NUMWORDS_A NUMERIC "32"

// Retrieval info: CONSTANT: OPERATION_MODE STRING "ROM"

// Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"

// Retrieval info: CONSTANT: OUTDATA_REG_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: RAM_BLOCK_TYPE STRING "M10K"

// Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "5"

// Retrieval info: CONSTANT: WIDTH_A NUMERIC "8"

// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"

// Retrieval info: USED_PORT: address 0 0 5 0 INPUT NODEFVAL "address[4..0]"

// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"

// Retrieval info: USED_PORT: q 0 0 8 0 OUTPUT NODEFVAL "q[7..0]"

// Retrieval info: CONNECT: @address_a 0 0 5 0 address 0 0 5 0

// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

// Retrieval info: CONNECT: q 0 0 8 0 @q_a 0 0 8 0

// Retrieval info: GEN_FILE: TYPE_NORMAL romlpm.v TRUE

// Retrieval info: GEN_FILE: TYPE_NORMAL romlpm.inc FALSE

// Retrieval info: GEN_FILE: TYPE_NORMAL romlpm.cmp FALSE

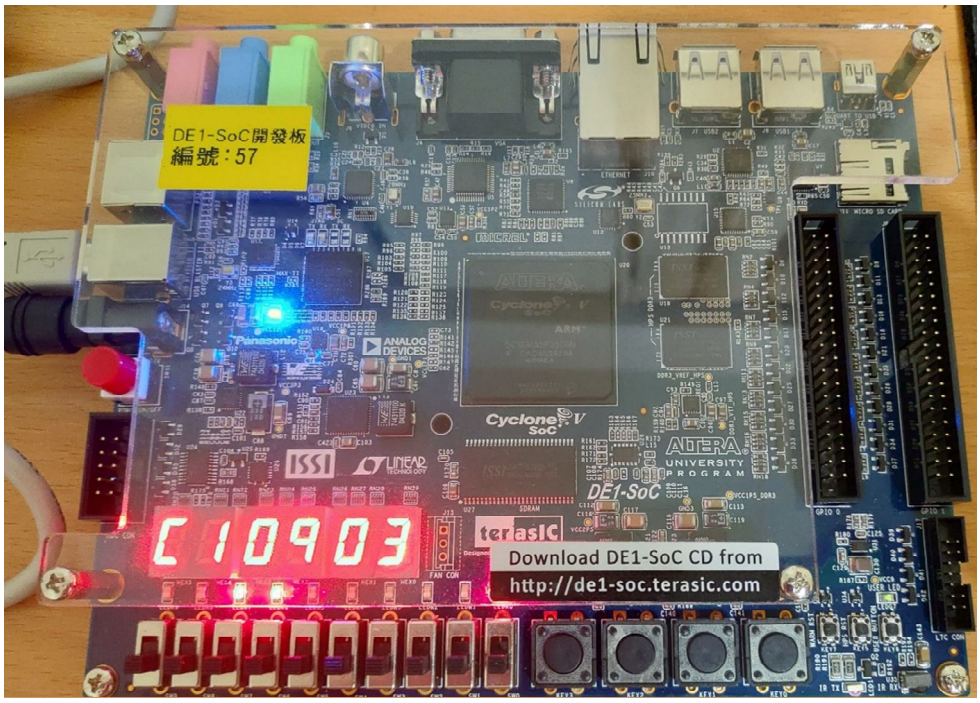
// Retrieval info: GEN_FILE: TYPE_NORMAL romlpm.bsf FALSE

// Retrieval info: GEN_FILE: TYPE_NORMAL romlpm_inst.v FALSE

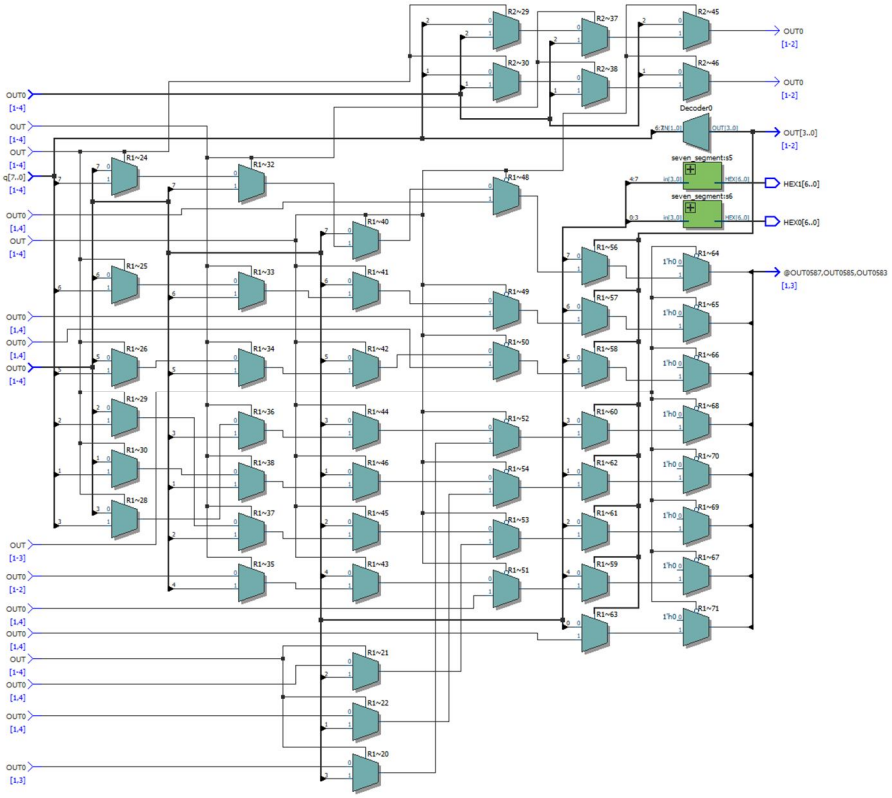
// Retrieval info: GEN_FILE: TYPE_NORMAL romlpm_bb.v TRUE

// Retrieval info: LIB_FILE: altera_mf

3. **實驗結果照片 (optional)**



4. **RTL 佈局(optional)**



5.問題與討論

經過本次實驗，不只結合上次實驗我提到的 memory module 的使用方法，更透過 verilog 實現出一個簡易的 processor，將過往於計算機結構所學到的電路架構實現，受益良多！