FPGA lab5 0713365 洪承玄

題目：Substitute dual-port memory by single-port memory

1. 實驗程式碼

```
module LAB5 (CLOCK_50, SW, LEDR, HEX3, HEX2, HEX1, HEX0);
    input CLOCK_50;
    input [9:0] SW;
    output [0:0] LEDR;
    output [6:0] HEX3, HEX2, HEX1, HEX0;
    wire [7:0] out_data;
    reg [7:0] in_data = 0;
    reg [25:0] cnt = 0;
    reg [4:0] addr, addr_cnt;
    reg en;

    assign LEDR = SW[9];
    two_digit_seg data_display(out_data, HEX1, HEX0);
    two_digit_seg address_display(addr, HEX3, HEX2);

    memorytest RAM (
    .address(addr),
    .clock(CLOCK_50),
    .data(in_data),
    .wren(en),
    .q(out_data));
```

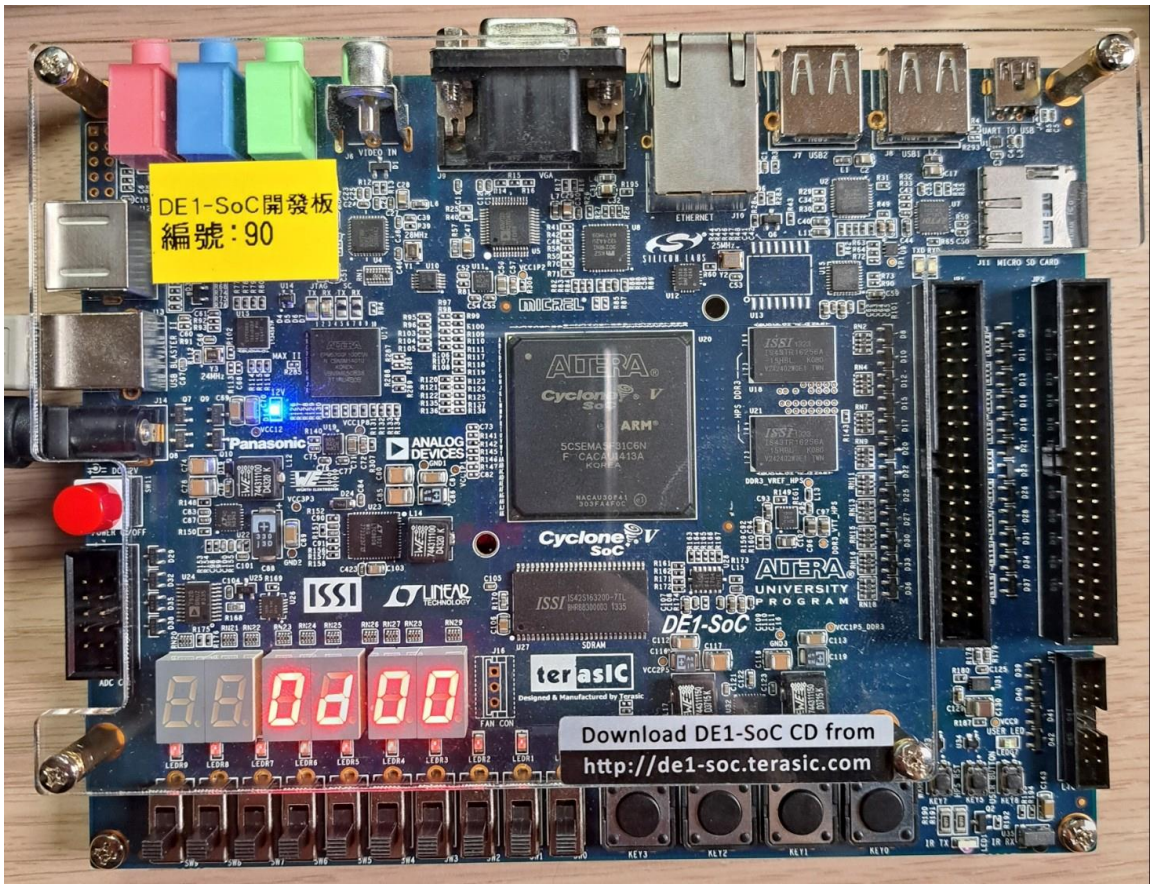
```
    always @(posedge CLOCK_50) begin
        //write mode
        if(cnt <= 10000) begin
            addr = SW[4:0];
            in_data = SW[7:0];
            cnt = cnt + 1;
            if(SW[9]) begin
                en = 1;
            end
            else begin
                en = 0;
            end
        end
        //read mode
        else if(cnt <= 50000000) begin
            en = 0;
            addr = addr_cnt;
            cnt = cnt + 1;
        end
        else begin
            en = 0;
            if(addr_cnt >= 31) begin
                addr_cnt = 0;
            end
            else begin
                addr_cnt = addr_cnt + 1;
            end
            cnt = 0;
        end
    end
endmodule
```

```
//bch to seg
module seg_decoder (bch, seg);
   input [3:0] bch;
   output reg [6:0] seg;

   always @(bch) begin
      case (bch)
         0 : seg = 7'b1000000;
         1 : seg = 7'b1111001;
         2 : seg = 7'b0100100;
         3 : seg = 7'b0110000;
         4 : seg = 7'b0011001;
         5 : seg = 7'b0010010;
         6 : seg = 7'b0000010;
         7 : seg = 7'b1111000;
         8 : seg = 7'b0000000;
         9 : seg = 7'b0010000;
         10 : seg = 7'b0001000;  //A
         11 : seg = 7'b0000011;  //b
         12 : seg = 7'b1000110;  //C
         13 : seg = 7'b0100001;  //d
         14 : seg = 7'b0000110;  //E
         15 : seg = 7'b0001110;  //F
         //lights out
         default : seg = 7'b1111111;
      endcase
   end
endmodule
```

```
//display two digit in seg
module two_digit_seg (num, seg1, seg0);
   input [7:0] num;
   output [6:0] seg1, seg0;

   seg_decoder de1(num[7:4], seg1);
   seg_decoder de0(num[3:0], seg0);
endmodule
```

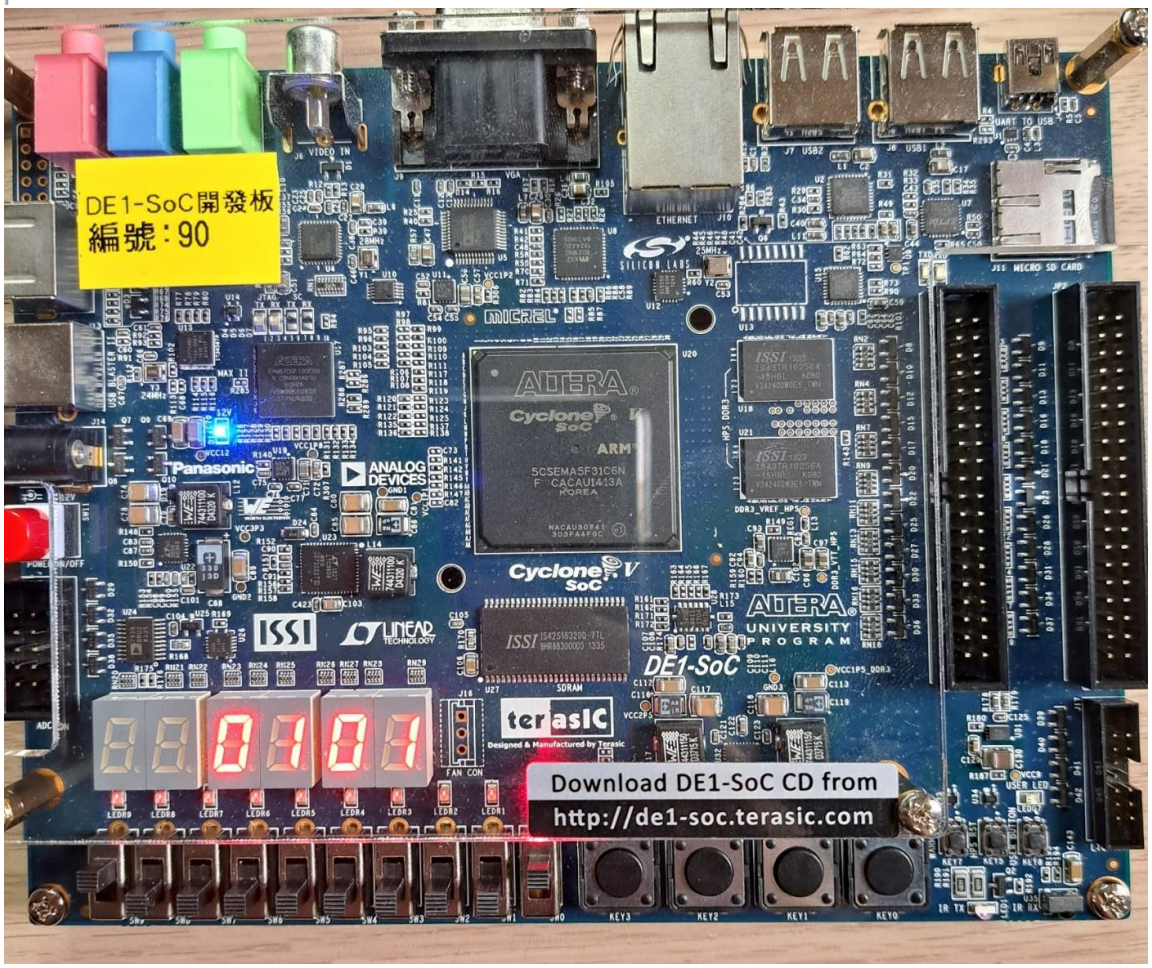
2. 實驗結果

```
Instance 0: 32x8
000000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020
```

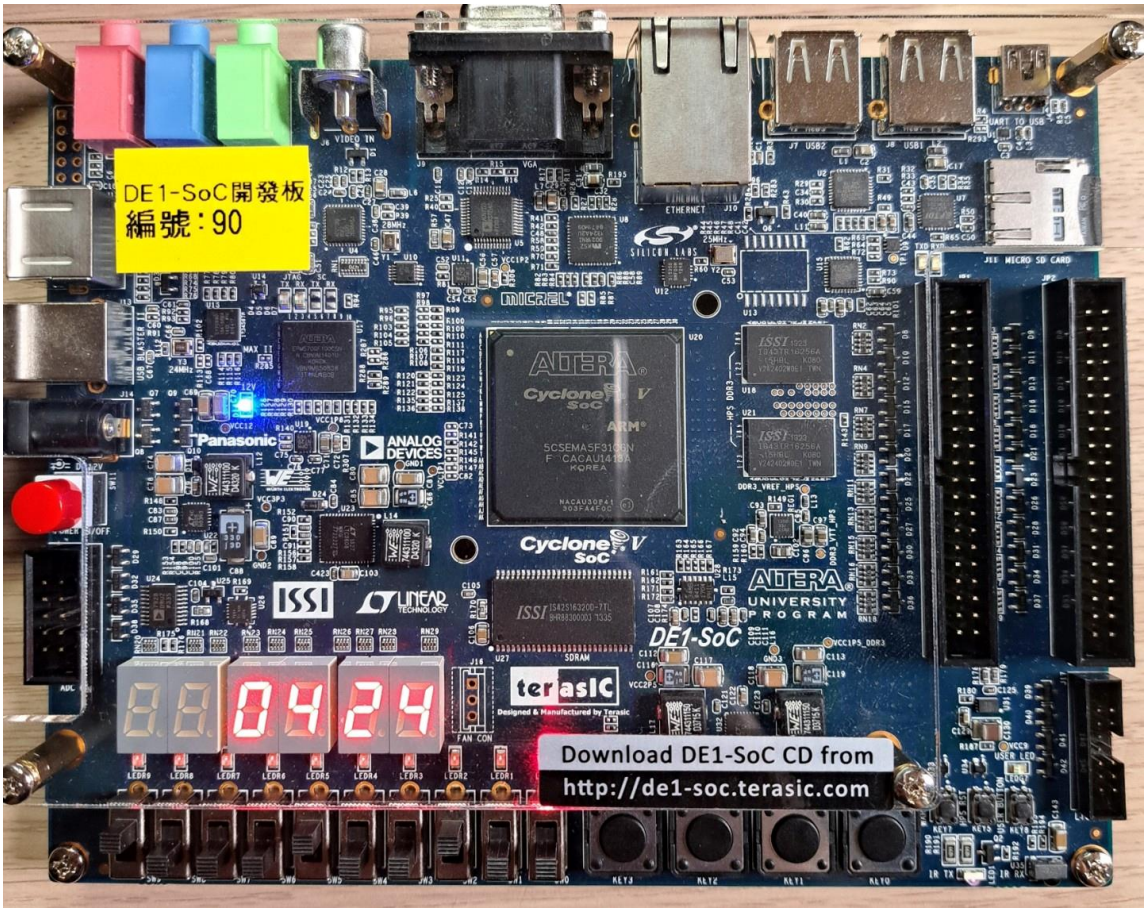


Instance 0: 32x8

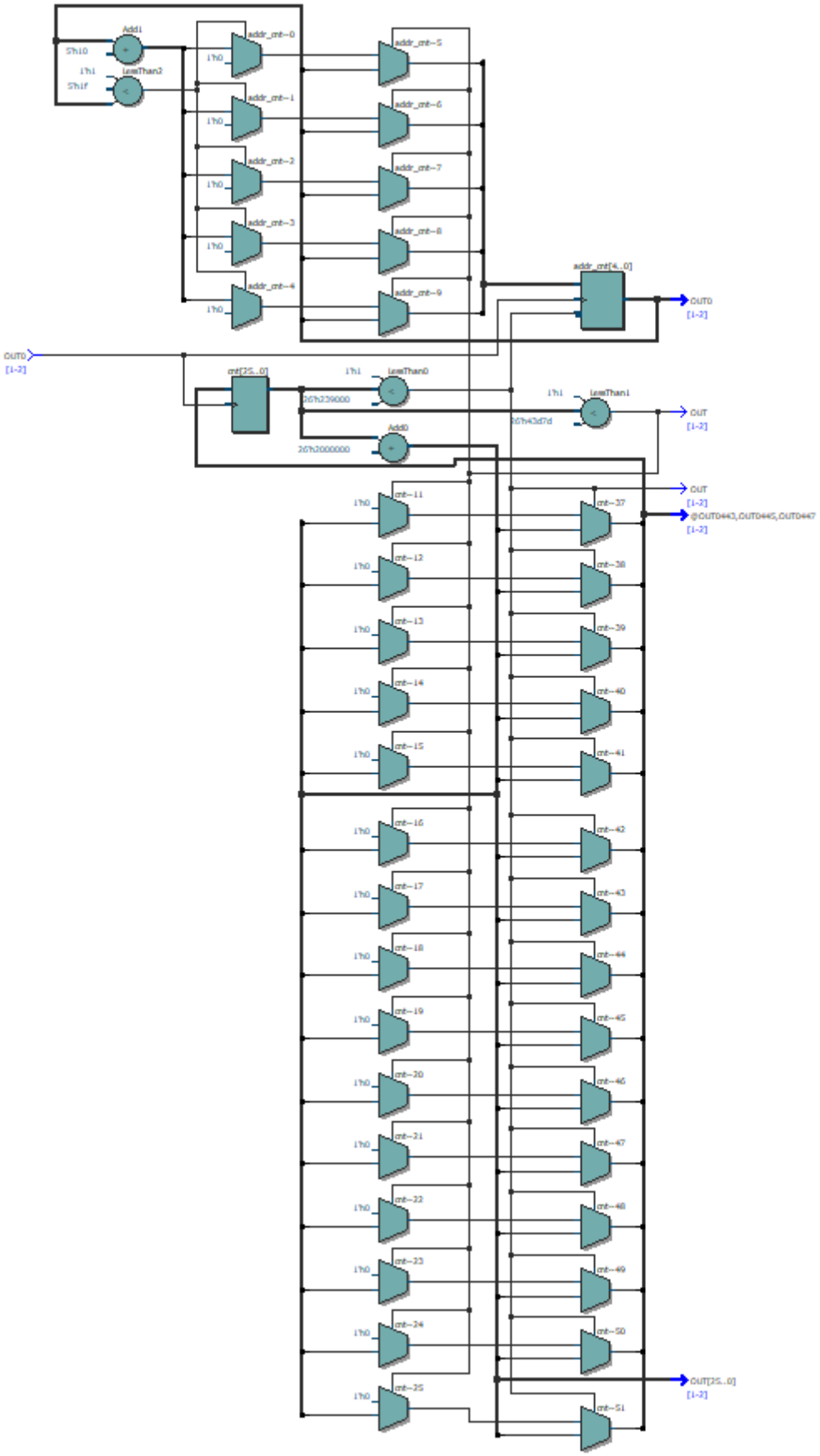
```
000000   00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020
```

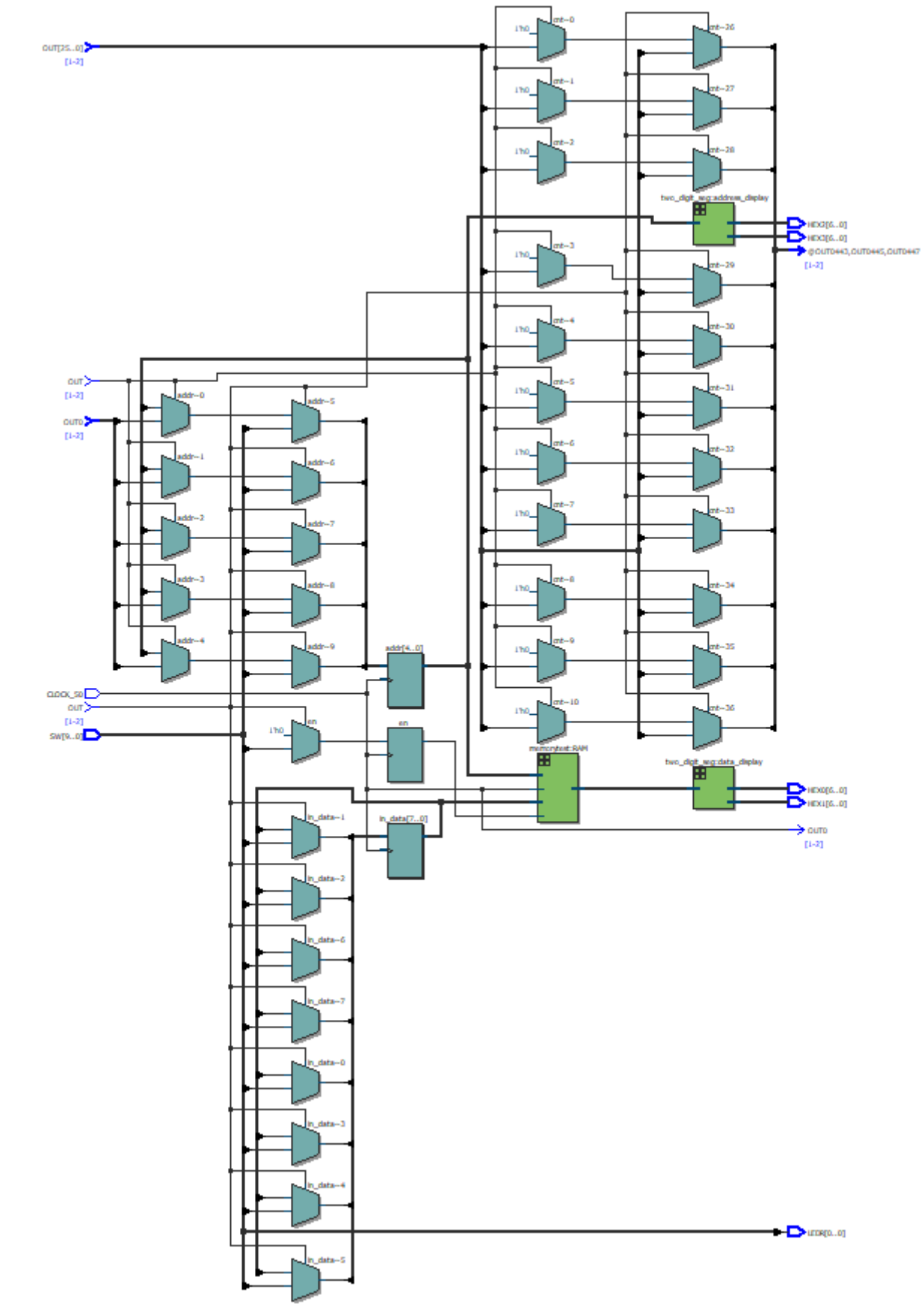


```
Instance 0: 32x8
000000   00 01 00 00 24 00 00 00 00 00 00 00 00 00 00 00
000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020
```



3. **RTL 佈局**


Add1
Shl0
1'h1
Shif
LessThan2
addr_cnt~0
addr_cnt~1
addr_cnt~2
addr_cnt~3
addr_cnt~4
addr_cnt~5
addr_cnt~6
addr_cnt~7
addr_cnt~8
addr_cnt~9
1'h0
addr_cnt[4..0]
OUT0
[1-2]
cnt[25..0]
LessThan0
26'h239000
LessThan1
26'h43d7d
Add0
26'h2000000
OUT
[1-2]
@OUT0443,OUT0445,OUT0447
[1-2]
cnt~11
cnt~12
cnt~13
cnt~14
cnt~15
cnt~16
cnt~17
cnt~18
cnt~19
cnt~20
cnt~21
cnt~22
cnt~23
cnt~24
cnt~25
cnt~37
cnt~38
cnt~39
cnt~40
cnt~41
cnt~42
cnt~43
cnt~44
cnt~45
cnt~46
cnt~47
cnt~48
cnt~49
cnt~50
cnt~51
OUT[25..0]
[1-2]



4. 問題與討論

- 建議不要在一個 clock edge 才寫入，因為 glitch，有可能會發生錯誤。改進方式是使用計數

器計算 clock 數量，並依數量大小區分兩種模式: 寫入和讀出，如此一來，看起來會是維持一秒的讀出，且寫入有很多 clock，穩定性提高。