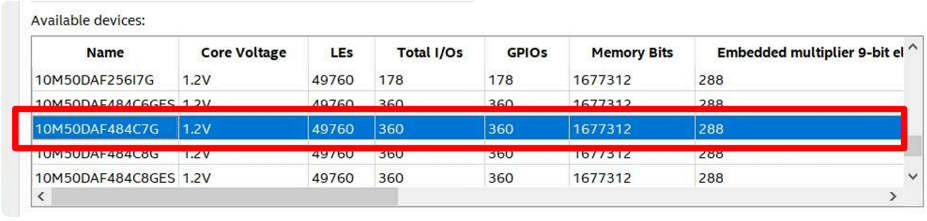
Week12 Moore Machine

Basic Setting

1. **Device Family:** MAX 10(DA/DF/DC/SA/SC)
2. **Available Devices:** 10M50DAF484C7G


Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit el
10M50DAF256I7G	1.2V	49760	178	178	1677312	288
10M50DAF484C6GES	1.2V	49760	360	360	1677312	288
10M50DAF484C7G	1.2V	49760	360	360	1677312	288
10M50DAF484C8G	1.2V	49760	360	360	1677312	288
10M50DAF484C8GES	1.2V	49760	360	360	1677312	288

3. **EDA Tool Settings, Simultation:** Modelsim-Altera, Verilog HDL
4. Click「Assignments > Pin Planner > Location」分配 Pin
5. 到裝置管理員，「其他裝置 > USB-Blaster」，右鍵「更新驅動程式」
6. 瀏覽電腦上的驅動程式
7. 路徑 `C:\intelFPGA_lite\16.1\quartus\drivers\usb-blaster` （16.1 或 18.1）
8. 到裝置管理員，「通用序列匯流排控制器 > Altera USB-Blaster」，有就代表成功
9. Click「Tools > Programmer」
10. Click「Hardware Setup > USB Blaster」，然後 Currently selected hardware 選「USB-Blaster [USB0]」
11. Click「Add file > ~.sof > Open」（注意是在 `outputfiles` 裡面）
12. Click「Start」顯示 Success 成功

Seven Display

- 0~15 七段顯示器對應
  - 0: 亮
  - 1: 不亮

```
4'd0: out = 7'b1000000;
4'd1: out = 7'b1111001;
4'd2: out = 7'b0100100;
4'd3: out = 7'b0110000;
4'd4: out = 7'b0011001;
4'd5: out = 7'b0010010;
4'd6: out = 7'b0000010;
4'd7: out = 7'b1111000;
4'd8: out = 7'b0000000;
4'd9: out = 7'b0010000;
4'd10: out = 7'b0001000;
4'd11: out = 7'b0000011;
4'd12: out = 7'b1000110;
4'd13: out = 7'b0100001;
4'd14: out = 7'b0000110;
4'd15: out = 7'b0001110;
```

Request

Lab I -- Moore machine (1/2)

- **Moore machine:輸出由當前的 state決定**
- **Mealy machine: 輸出由當前的 state和 input訊號決定**
- **完成一個 Moore machine**
  - **其 I/O 與 state 變化如右表**
  - **變動頻率為 1Hz**
  - **Reset為0時，State初始化為 S0 (非同步)**

目前狀態 (current-state)	下一個狀態 (next-state)		七段顯示器輸出 (output)
	In=0	In=1	
S0	S1	S2	0
S1	S2	S3	1
S2	S3	S4	2
S3	S4	S5	3
S4	S5	S6	4
S5	S6	S0	5
S6	S0	S1	6

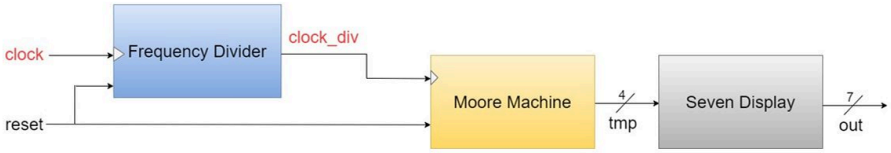
目前狀態 (current-state)	下一個狀態 (next-state)		七段顯示器輸出 (output)
	In=0	In=1	
S0	S1	S2	0
S1	S2	S3	1
S2	S3	S4	2
S3	S4	S5	3
S4	S5	S6	4
S5	S6	S0	5
S6	S0	S1	6

- **請將輸出的數 值顯示於七段顯示器**
- **系統架構圖請參考下方**
  - **Input: clock(MAX10_CLK1_50)、reset(SW0)、In(SW1)**
  - **Output: out(7 bits, HEX06~HEX00)**
- **請畫出Finite State Machine並說明其運作過程**

Logic Block

- Input: clock, reset, in (下面圖片少寫 in)
  - reset == 0 means 將 state 變為 s0
  - reset == 1 means state 繼續往下跑
  - in == 0, in == 1 不同 state

- Output: out



Verilog

fequency_driver.v

```
`define TimeExpire 32'd25000000
module frequency_driver(clk, rst, div_clk);
    input clk, rst;
    output div_clk;
    reg div_clk;
    reg [31:0] count;

    always@(posedge clk)
    begin
        if (!rst)
        begin
            count <= 32'd0;
            div_clk <= 1'b0;
        end
        else
        begin
            if (count == `TimeExpire)
            begin
                count <= 32'd0;
                div_clk <= ~div_clk;
            end
            else
            begin
                count <= count + 32'd1;
            end
        end
    end
endmodule
```

moore_machine.v

```
module moore_machine(clock_div, reset, in, tmp);
input clock_div, reset, in;
output reg [3:0] tmp;

parameter s0 = 4'b0000;
parameter s1 = 4'b0001;
parameter s2 = 4'b0010;
parameter s3 = 4'b0011;
parameter s4 = 4'b0100;
parameter s5 = 4'b0101;
parameter s6 = 4'b0110;

reg [3:0] current_state, next_state;

always @(posedge clock_div or negedge reset)
begin
        if(!reset)
        begin
                current_state <= s0;
        end
        else
        begin
                current_state <= next_state;
        end
end

always @(*)
begin
        case(current_state)
                s0: next_state <= (~in) ? s1 : s2;
                s1: next_state <= (~in) ? s2 : s3;
                s2: next_state <= (~in) ? s3 : s4;
                s3: next_state <= (~in) ? s4 : s5;
                s4: next_state <= (~in) ? s5 : s6;
                s5: next_state <= (~in) ? s6 : s0;
                s6: next_state <= (~in) ? s0 : s1;
                default: next_state <= s0;
        endcase
end

always @(*)
begin
        case(current_state)
                s0: tmp = s0;
                s1: tmp = s1;
                s2: tmp = s2;
                s3: tmp = s3;
                s4: tmp = s4;
                s5: tmp = s5;
                s6: tmp = s6;
                default: tmp = s0;
        endcase
end
endmodule


```

seven_display.v

```
module seven_display(in, out);

input [3:0] in;
output [6:0] out;
reg [6:0] out;

always @(in)
begin
    case(in)
        4'd0: out = 7'b1000000;
        4'd1: out = 7'b1111001;
        4'd2: out = 7'b0100100;
        4'd3: out = 7'b0110000;
        4'd4: out = 7'b0011001;
        4'd5: out = 7'b0010010;
        4'd6: out = 7'b0000010;
        4'd7: out = 7'b1111000;
        4'd8: out = 7'b0000000;
        4'd9: out = 7'b0010000;
        4'd10: out = 7'b0001000;
        4'd11: out = 7'b0000011;
        4'd12: out = 7'b1000110;
        4'd13: out = 7'b0100001;
        4'd14: out = 7'b0000110;
        default : out = 7'b0001110;
    endcase
end
endmodule
```

main.v

```
module main(clock, reset, in, out);
// module main(clock, reset, in, out, see); 看 Unsigned 用
input clock, reset, in;
output [6:0] out;
// output [3:0] see; 看 Unsigned 用

wire clock_div;
wire [3:0] tmp;

frequency_driver f1(
        .clk(clock),
        .rst(reset),
        .div_clk(clock_div));

moore_machine moore1(
        .clock_div(clock), // 非用 fpga 請直接填入 clock， 用 fpga 填 clock_div
        .reset(reset),
        .in(in),
        .tmp(tmp));

seven_display display1(
        .in(tmp),
        .out(out));

// assign see = tmp; // 看 Unsigned 用
endmodule
```