第三次實驗報告

題目：

Adders, Subtractors, and Multipliers

姓名：黃子軒

學號：310581003

繳交日期：111 年 10 月 31 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，且試著熟悉對 clock 除頻以達到自記所需的 clock rate，更學習如何實現加法器、乘法器，並運用加法器、乘法器組合出日常生活中常見的計算機。

2. 實驗程式碼

```
module Lab3_part6(KEY, SW, LEDR, HEX0, HEX1, HEX2, HEX3);

input [3:0] KEY;

input [9:0] SW;

output [9:0] LEDR;

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

wire [7:0] out1, out2, A, B, C, D;

wire [15:0] m1, m2, SUM, display;

wire carry;


give_value gv(KEY[0], KEY[1], KEY[2], SW[9:0], A, B, C, D);


Multiplier_8bit M_1(A, B, m1);

Multiplier_8bit M_2(C, D, m2);

Full_adder_8bit FA_1(0,

                     m1[0], m1[1], m1[2], m1[3], m1[4], m1[5], m1[6], m1[7],

                      m2[0], m2[1], m2[2], m2[3], m2[4], m2[5], m2[6], m2[7],

                      SUM[0], SUM[1], SUM[2], SUM[3], SUM[4], SUM[5], SUM[6], SUM[7],
carry);

Full_adder_8bit FA_2(carry,

                      m1[8], m1[9], m1[10], m1[11], m1[12], m1[13], m1[14], m1[15],

                      m2[8], m2[9], m2[10], m2[11], m2[12], m2[13], m2[14], m2[15],

                      SUM[8], SUM[9], SUM[10], SUM[11], SUM[12], SUM[13], SUM[14],
SUM[15], LEDR[9]);
```

```
display d(KEY[3], SW[8], A, B, C, D, SUM, display);

seven_segment s1(display[15:12],HEX3);
seven_segment s2(display[11:8],HEX2);
seven_segment s3(display[7:4],HEX1);
seven_segment s4(display[3:0],HEX0);

endmodule


module Multiplier_8bit(a, b, p);
	input [7:0] a, b;
	output [15:0] p;
	wire A01, A02, A03, A04, A05, A06, A07, A08,
		A11, A12, A13, A14, A15, A16, A17, A18,
		A21, A22, A23, A24, A25, A26, A27, A28,
		A31, A32, A33, A34, A35, A36, A37, A38,
		A41, A42, A43, A44, A45, A46, A47, A48,
		A51, A52, A53, A54, A55, A56, A57, A58;


	assign p[0] = a[0]&b[0];
	Full_adder_8bit FA0(0,
				a[1]&b[0], a[2]&b[0], a[3]&b[0], a[4]&b[0], a[5]&b[0], a[6]&b[0],
a[7]&b[0], 0,
```

```
                a[0]&b[1], a[1]&b[1], a[2]&b[1], a[3]&b[1], a[4]&b[1], a[5]&b[1],
a[6]&b[1], a[7]&b[1],
                p[1], A01, A02, A03, A04, A05, A06, A07, A08);

    Full_adder_8bit FA1(0,
                A01, A02, A03, A04, A05, A06, A07, A08,
                a[0]&b[2], a[1]&b[2], a[2]&b[2], a[3]&b[2], a[4]&b[2], a[5]&b[2],
a[6]&b[2], a[7]&b[2],
                p[2], A11, A12, A13, A14, A15, A16, A17, A18);

    Full_adder_8bit FA2(0,
                A11, A12, A13, A14, A15, A16, A17, A18,
                a[0]&b[3], a[1]&b[3], a[2]&b[3], a[3]&b[3], a[4]&b[3], a[5]&b[3],
a[6]&b[3], a[7]&b[3],
                p[3], A21, A22, A23, A24, A25, A26, A27, A28);

    Full_adder_8bit FA3(0,
                A21, A22, A23, A24, A25, A26, A27, A28,
                a[0]&b[4], a[1]&b[4], a[2]&b[4], a[3]&b[4], a[4]&b[4], a[5]&b[4],
a[6]&b[4], a[7]&b[4],
                p[4], A31, A32, A33, A34, A35, A36, A37, A38);

    Full_adder_8bit FA4(0,
                A31, A32, A33, A34, A35, A36, A37, A38,
                a[0]&b[5], a[1]&b[5], a[2]&b[5], a[3]&b[5], a[4]&b[5], a[5]&b[5],
a[6]&b[5], a[7]&b[5],
                p[5], A41, A42, A43, A44, A45, A46, A47, A48);

    Full_adder_8bit FA5(0,
```

```
                                A41, A42, A43, A44, A45, A46, A47, A48,
                                a[0]&b[6], a[1]&b[6], a[2]&b[6], a[3]&b[6], a[4]&b[6], a[5]&b[6],
a[6]&b[6], a[7]&b[6],
                                p[6], A51, A52, A53, A54, A55, A56, A57, A58);

    Full_adder_8bit FA6(0,
                                A51, A52, A53, A54, A55, A56, A57, A58,
                                a[0]&b[7], a[1]&b[7], a[2]&b[7], a[3]&b[7], a[4]&b[7], a[5]&b[7],
a[6]&b[7], a[7]&b[7],
                                p[7], p[8], p[9], p[10], p[11], p[12], p[13], p[14], p[15]);
endmodule

module Full_adder_8bit(cin,
                                a0, a1, a2, a3, a4, a5, a6, a7,
                                b0, b1, b2, b3, b4, b5, b6, b7,
                                p0, p1, p2, p3, p4, p5, p6, p7, cout);

    input cin, a0, a1, a2, a3, a4, a5, a6, a7,
          b0, b1, b2, b3, b4, b5, b6, b7;
    output p0, p1, p2, p3, p4, p5, p6, p7, cout;
    wire c1, c2, c3, c4, c5, c6, c7, cin;

    Full_adder FA1(c1, p0, a0, b0, cin);
    Full_adder FA2(c2, p1, a1, b1, c1);
    Full_adder FA3(c3, p2, a2, b2, c2);
    Full_adder FA4(c4, p3, a3, b3, c3);
    Full_adder FA5(c5, p4, a4, b4, c4);
```

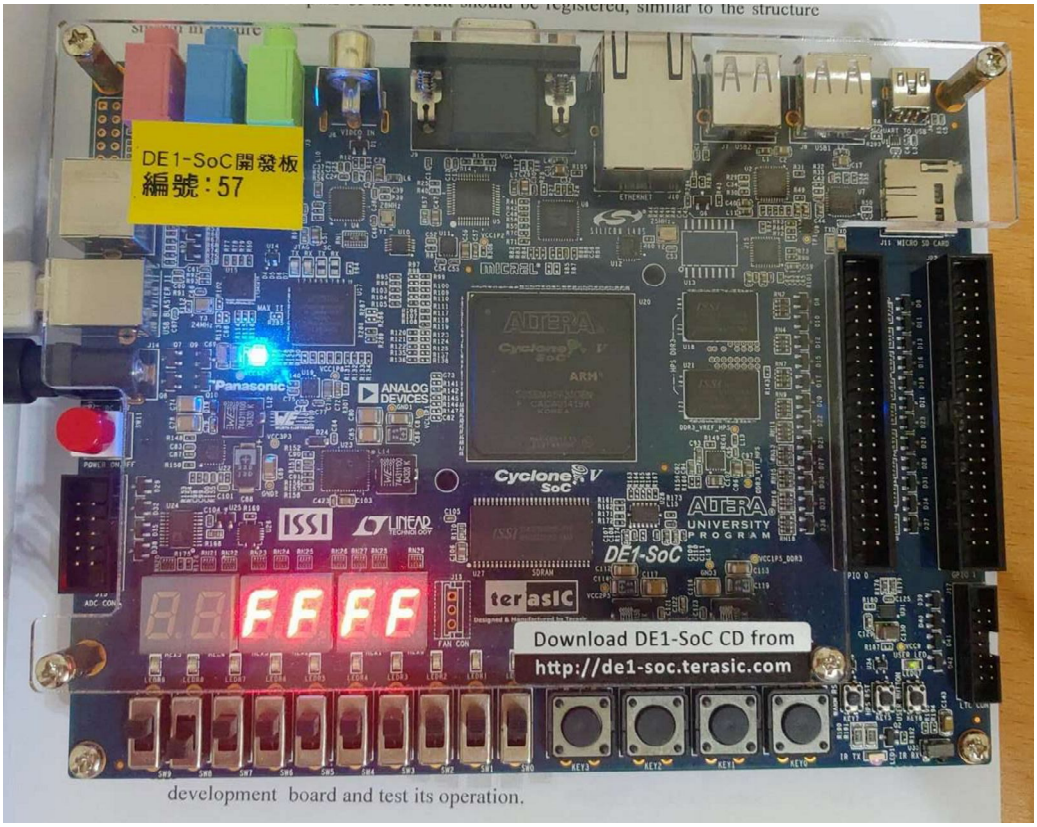
```
    Full_adder FA6(c6, p5, a5, b5, c5);
    Full_adder FA7(c7, p6, a6, b6, c6);
    Full_adder FA8(cout, p7, a7, b7, c7);
endmodule


module Full_adder(co, s, a, b, ci);
input a, b, ci;
output co, s;
wire d;
assign d=a^b;
assign s=ci^d;
assign co=(~d&b)|(d&ci);
endmodule

module give_value(KEY0, KEY1, KEY2, SW, A, B, C, D);
    input KEY0, KEY1, KEY2;
    input [9:0]SW;
    output reg [7:0]    A, B, C, D;

    always @(negedge KEY1 or negedge KEY0)
    begin
        if (KEY0==0)
            begin
            A<=0;
            B<=0;
```

```
        C<=0;
        D<=0;
        end
else if (SW[9]==1)
begin
        if (SW[8]==0)
                begin
                if (KEY2==1)
                        begin
                        A<=SW[7:0];
                        end
                else
                begin
                        B<=SW[7:0];
                end
        end
        else
                begin
                if (KEY2==1)
                        begin
                        C<=SW[7:0];
                        end
                else
                        begin
                        D<=SW[7:0];
                        end
```

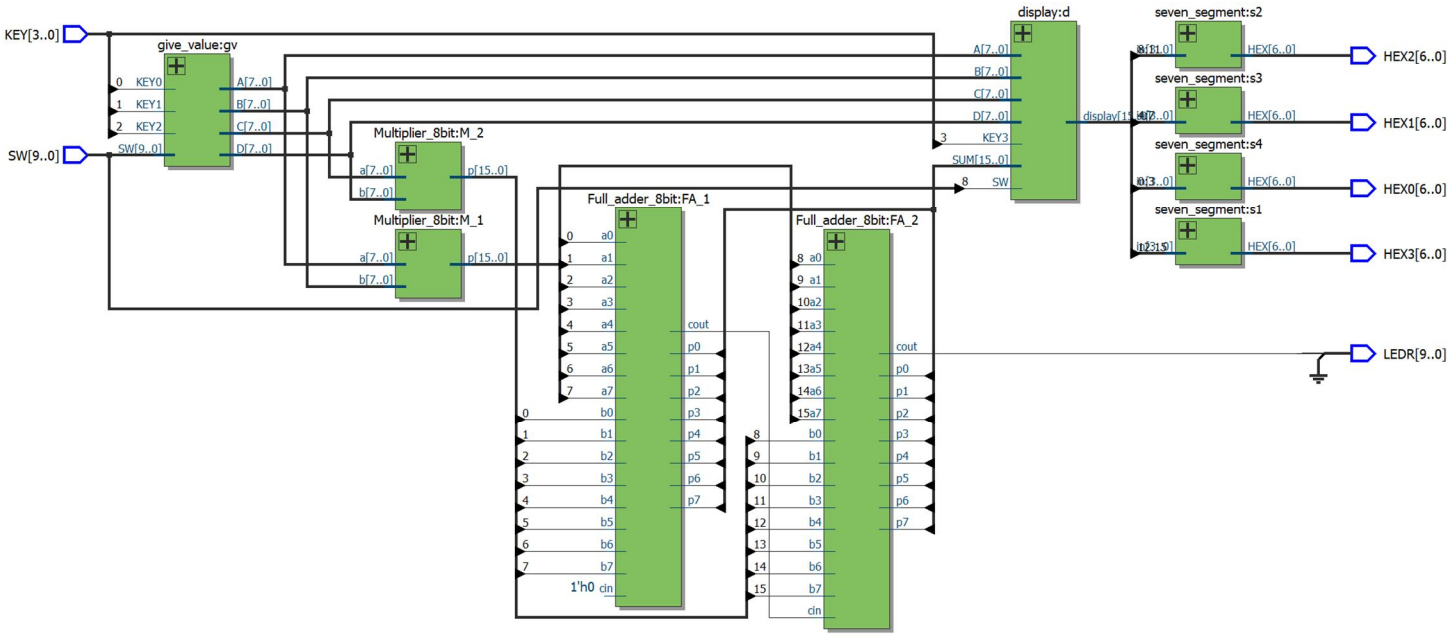
```
            end
        end
    end
endmodule

module display(KEY3, SW, A, B, C, D, SUM, display);
    input KEY3, SW;
    input [7:0] A, B, C, D;
    input [15:0] SUM;
    output reg [15:0] display;

    always @(KEY3 or SW)
    begin
    if (KEY3==0)
        display<=SUM;
    else
        begin
        if (SW==0)
            begin
            display[15:8]<=A;
            display[7:0]<=B;
            end
        else
            begin
            display[15:8]<=C;
            display[7:0]<=D;
```

```
                end
            end
        end
endmodule



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

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，讓我對加法器與乘法器的架構與應用更熟悉，受益良多！