105-2 CA with Embedded DSD Homework 1 Report

B03901156 電機三 黃于瑄

1. 8-bit arithmetic logic unit(ALU)

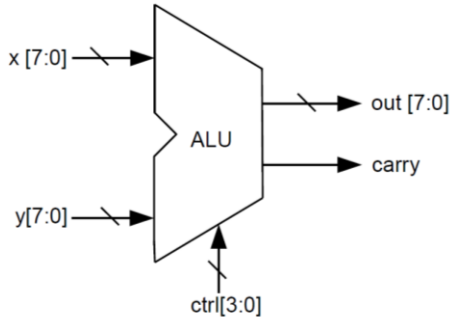


Figure 1

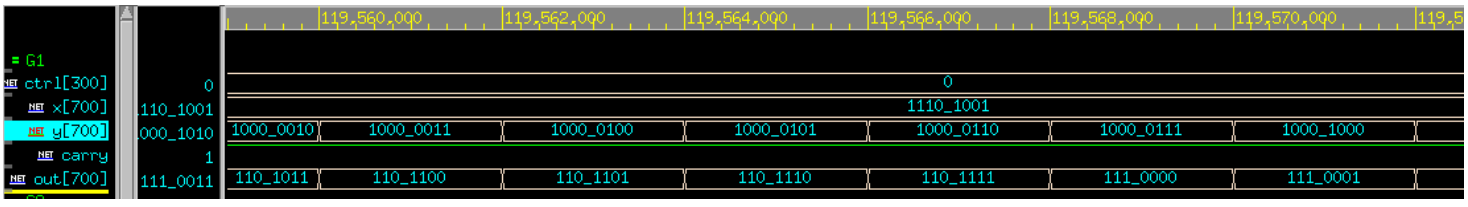
The implementation of alu_rtl_tb2.v and alu_tb2.v :

參考 Lab2 中 Lab2_test_alu.v 的寫法，對於每個operation皆考慮x及y所有可能的組合情形，並在console上印出test通過或失敗的訊息；以Addition的test pattern 為例:

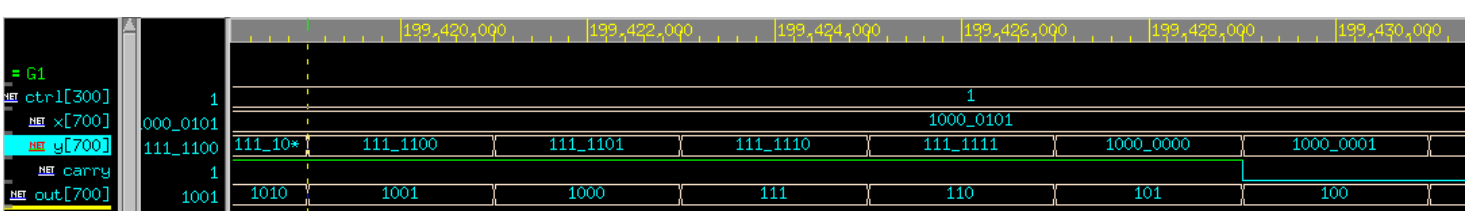
```
// test for Add
#(`CYCLE);
ctrl = 4'b0000;
for(i=0;i<256;i=i+1) //x's all situations
begin
  for(j=0;j<256;j=j+1) //y's all situations
  begin
  x=i[7:0];y=j[7:0];
  #20  true_out=x+y; tmp ={x[7],x}+{y[7],y}; true_carry=tmp[8];
     if(out !== true_out[7:0]) //check if "out" is the same as the ture output
     begin
       $display(" %b + %b should be %b. But your output is %b",x,y,true_out,out);
     end
     else $display( "ADD ----- PASS" );
     if(carry !== true_carry) //check if "carry" bit is the same as the ture carry
       $display(" %b + %b's carry should be %b. But your output is %b",x,y,true_carry,carry);
     end
     else $display( "ADD_carry ----- PASS" );
  end
end
```

而每個operation所得波形圖如下所示: (case ctrl)

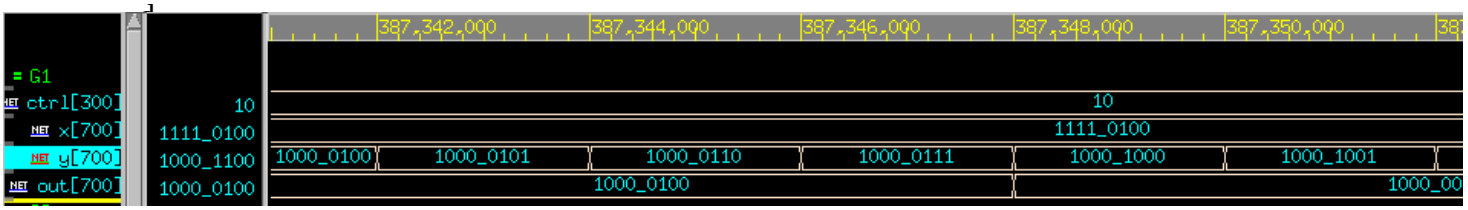
a. (ctrl == **4'b0000**) → out = x + y ;



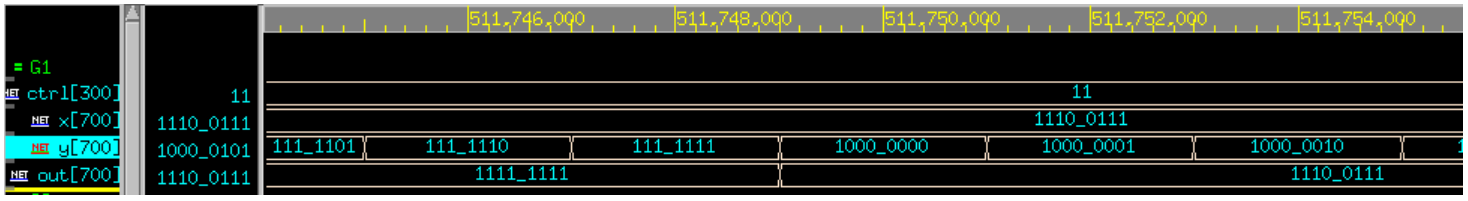
b. (ctrl == **4'b0001**) → out = x - y ;



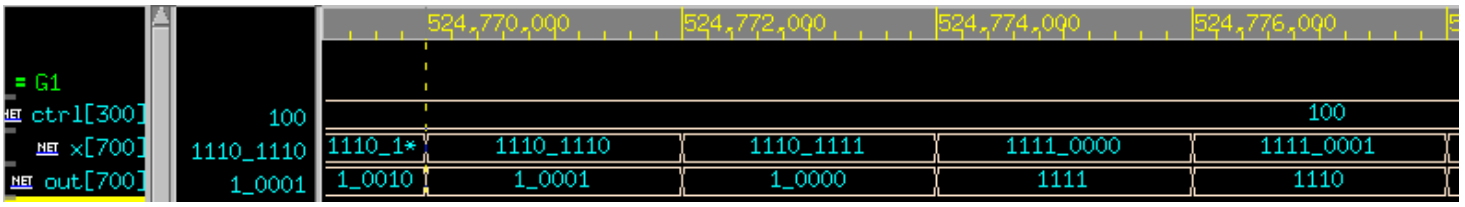
c. (ctrl == **4'b0010**) → out = x **&** y ;



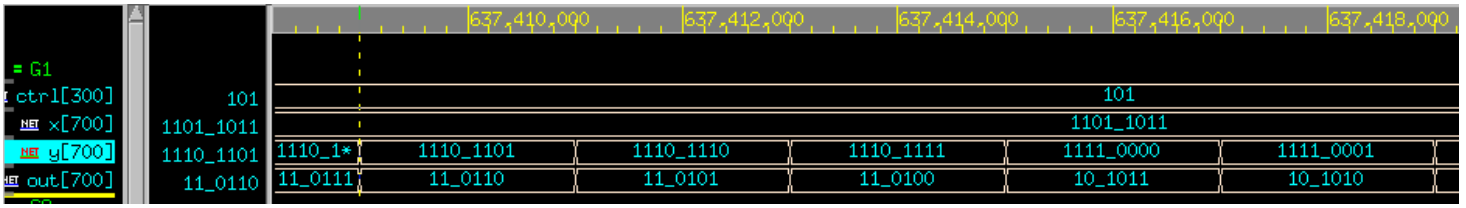
e. (ctrl == **4'b0011**) → out = x | y ;



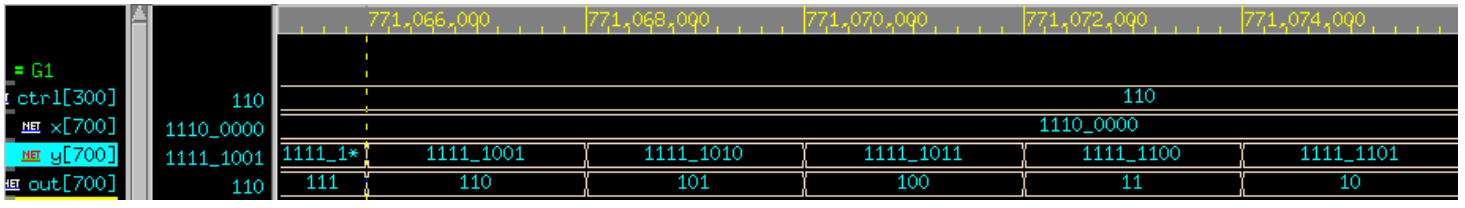
f. (ctrl == **4'b0100**) → out = ~ x ;



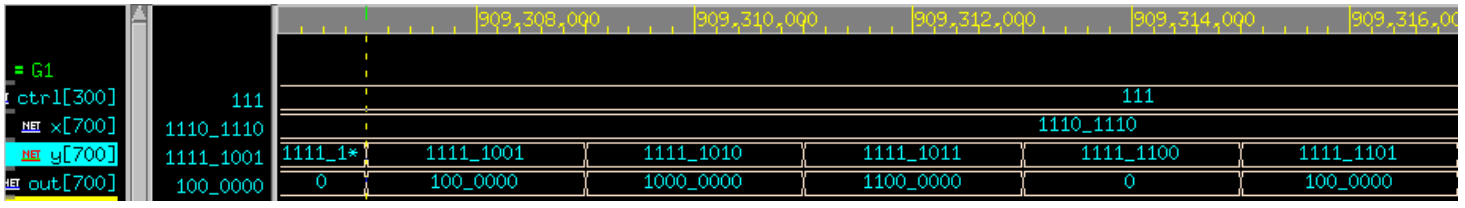
g. (ctrl == **4'b0101**) → out = x ^ y ;



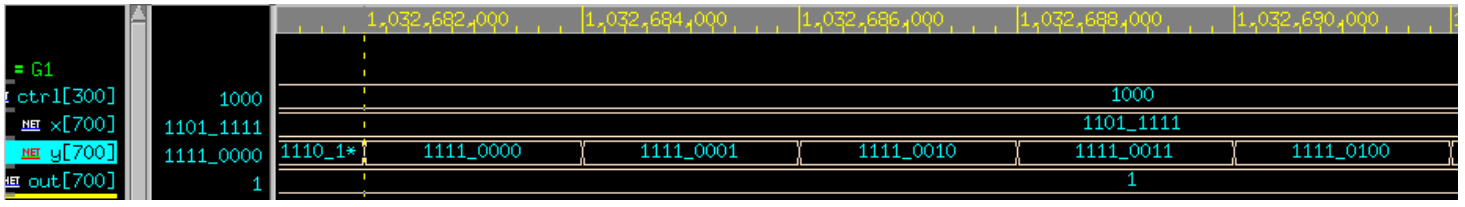
h. (ctrl == **4'b0110**) → out = ~x **&** ~y ;



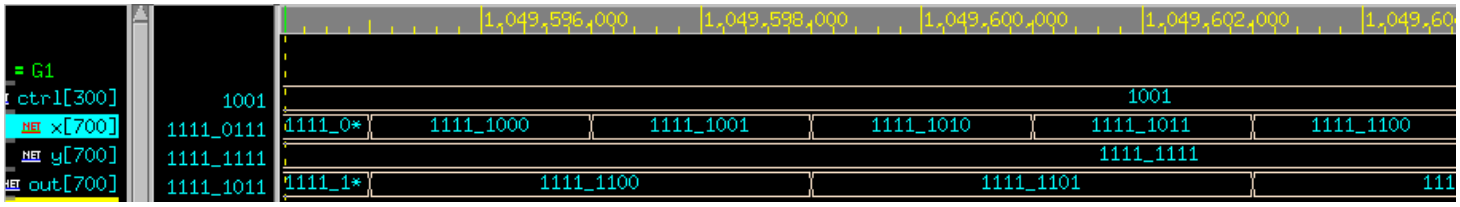
i. (ctrl == **4'b0111**) → out = **y** << **x[2:0]** ;



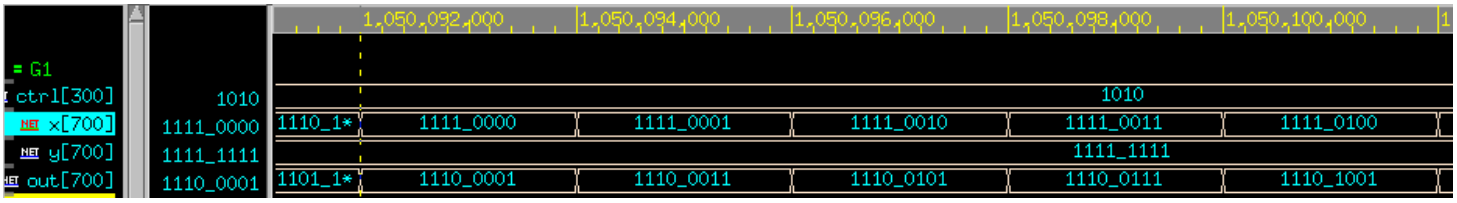
j. (ctrl == **4'b1000**) → out = **y** >> **x[2:0]** ;



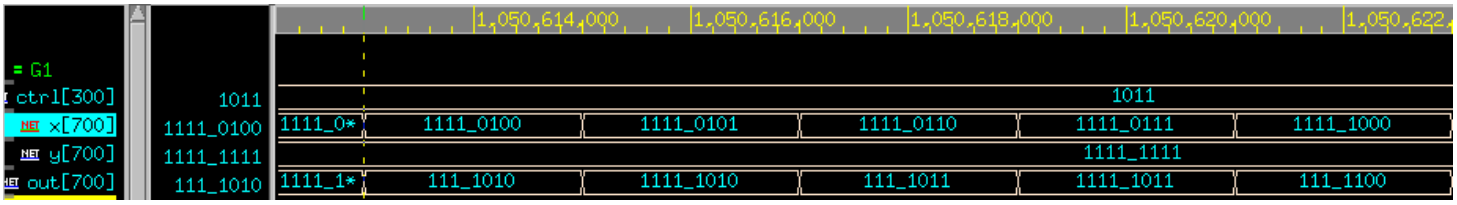
k. (ctrl == **4'b1001**) → out = **{x[7],x[7:1]}** ;



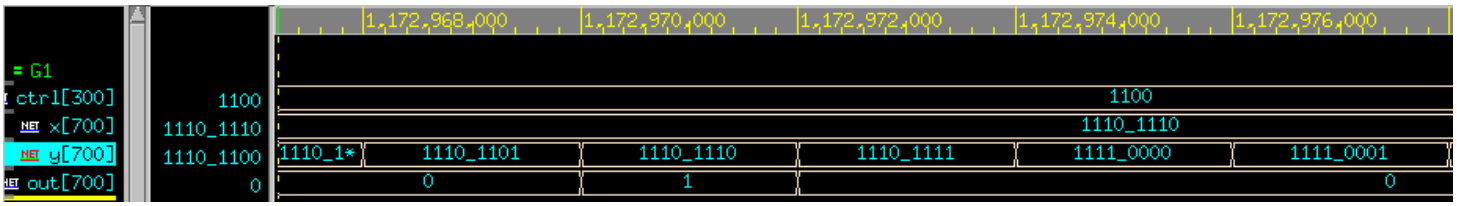
l. (ctrl == **4'b1010**) → out = **{x[6:0] , x[7]}** ;



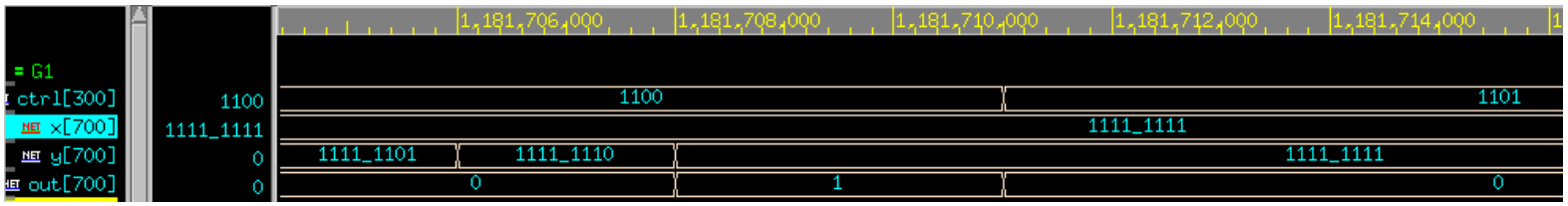
m. (ctrl == **4'b1011**) → out = **{x[0] , x[7:1]}** ;



n. (ctrl == **4'b1100**) → out = **(x==y) ? 1 : 0** ;



o. (ctrl == **4'b1101** or **4'b1110** or **4'b1111**) → out = **0** ;



2. 8x8 Register File

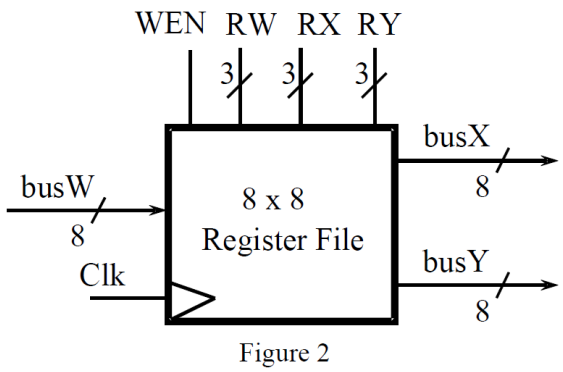


Figure 2

關於 $r0 = constant zero :

```
//==== sequential circuit ==== register file
    reg [7:0] registers[7:0];

    always@(posedge Clk) begin
        if(WEN)
          registers[RW] <= busW;
    end

    always@(RX or registers[RX])begin
        if(RX == 0) busX <= 0;
        else busX <= registers[RX];
    end

    always@(RY or registers[RY])begin
        if(RY == 0) busY <= 0;
        else busY <= registers[RY];
    end
```

3. Simple Calculator

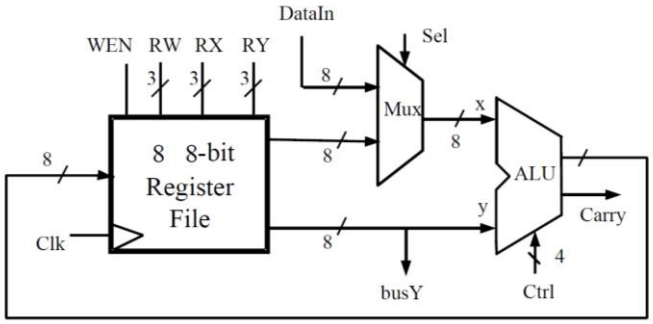


Figure 3

Design:

```
module simple_calculator(
    Clk,
    WEN,
    RW,
    RX,
    RY,
    DataIn,
    Sel,
    Ctrl,
    busY,
    Carry
);

    input        Clk;
    input        WEN;
    input  [2:0] RW, RX, RY;
    input  [7:0] DataIn;
    input        Sel;
    input  [3:0] Ctrl;
    output [7:0] busY;
    output       Carry;

//==== reg/wire definition ====
    reg signed [7:0] x;
    reg [7:0] busY, busX;
    reg Carry;
    reg signed [7:0] busW;
    reg [8:0] tmp;
    reg [7:0] neg;


// submodule instantiation ---multiplexer
    always@(DataIn or busX or Sel) begin
        case(Sel)
            1'b0: x = DataIn;
            1'b1: x = busX;
        endcase
    end
```

```

// combinational part --- ALU ---> Input: Ctrl, x, busY
    always@(Ctrl,x,busY)
    begin
      case(Ctrl)
        4'b0000: begin
            tmp ={x[7],x}+{busY[7],busY};
            Carry = tmp[8];
            busW = x+busY;
        end
        4'b0001: begin
            neg = ~busY + 00000001;
            tmp ={x[7],x}+{neg[7],neg};
            Carry = tmp[8];
            busW = x-busY;
        end
        4'b0010: busW = x & busY;
        4'b0011: busW = x | busY;
        4'b0100: busW = ~x;
        4'b0101: busW = x ^ busY;
        4'b0110: busW = ~x & ~busY;
        4'b0111: busW = busY << x[2:0];
        4'b1000: busW = busY >> x[2:0];
        4'b1001: busW = {x[7],x[7:1]};
        4'b1010: busW = {x[6:0] , x[7]};
        4'b1011: busW = {x[0] , x[7:1]};
        4'b1100: busW = (x==busY)?1:0;
        4'b1101: busW = 0;
        4'b1110: busW = 0;
        4'b1111: busW = 0;
      endcase
    end
```

```


//==== sequential circuit ==== register file
    reg [7:0] registers[7:0];

    always@(posedge Clk) begin
        if(WEN)
          registers[RW] <= busW;
    end

    always@(RX or registers[RX])begin
        if(RX == 0) busX <= 0;
        else busX <= registers[RX];
    end

    always@(RY or registers[RY])begin
        if(RY == 0) busY <= 0;
        else busY <= registers[RY];
    end

endmodule
```

波形圖: