計算機組織 Midterm Project: ALU Design

112 學年度第 2 學期

老師：朱守禮 老師

學生：11127132 羅海綺、11127137 黃乙家、11127138 林雨臻

一、背景

為了使學生充分了解，該如何使用 Verilog HDL 與 Modelsim 模擬器，以計算機組織課程講義：

[1] Chapter 3:Arithmetic for Computers Part 1: ALU (Add/Subtraction)

[2] Chapter 3:Arithmetic For Computers Part 2: Multiplication and Division

[3] Chapter 3:Supplement: Verilog Concepts

為基礎，設計 ALU 與乘法器，以供 Final Project 之用。

二、方法

1.功能說明：本 Project 包含 AND, OR, SUB, SLT, SLL, MULTU…等 7 項功能。

2.設計要求：Datapath 與詳細架構圖(並以 PowerPoint 或 Word 設計繪製)。

(1) ALU:

(a) 設計重點說明：

- 包含 32-bits AND, OR, ADD, SUB, SLT 等功能。
- 使用 Gate-Level Modeling 與 Data Flow Modeling(Continuous Assignments)，從 Full Adder 開始，以 Ripple-Carry 的進位方式，連接 32 個 1-bit ALU Bit Slice。

- ◆ 本模組為組合邏輯(Combinational Logic)。

(b) Datapath 與詳細架構圖：請參考附錄(CO_ALU.pptx)。

(2) Multiplier:

(a) 設計重點說明：

- ◆ 為 32-bits 無號數乘法 Sequential Multiplier，須採用 Second Version Sequential Multiplier 來設計。
- ◆ 可使用 Always Block 或 Procedure Assignment 來設計，但【不接受迴圈形式的設計】。
- ◆ 本模組為循序邏輯(Sequential Logic)，因此須以 Clock 訊號同步。

(b) Datapath 與詳細架構圖：請參考附錄(CO.pptx)。

(3) Shifter:

(a) 設計重點說明：

- ◆ 32-bits Barrel Shifter，完成邏輯左移運算。
- ◆ 以 Data Flow Modeling(Continuous Assignments)完成，不能直接用 '>>'或'<<' operator，亦不可使用 Always Block 或 Procedure Assignment 來設計。
- ◆ 以 160 個 Mux 實現 Shifter。
- ◆ 本模組為組合邏輯(Combinational Logic)。

(b) Datapath 與詳細架構圖：請參考附錄。

(5) HiLo 暫存器:

(a) 設計重點說明：

- ◆ 乘法器計算完後，儲存計算結果之 64-bit 暫存器。
- ◆ 本模組為循序邏輯(Sequential Logic)，因此須以 Clock 訊號同步。

(b) Datapath 與詳細架構圖：請參考附錄(CO.pptx)。

(5) MUX:

(a) 設計重點說明：

- Data Flow Modeling 設計。
- 本模組為組合邏輯(Combinational Logic)。

(b) Datapath 與詳細架構圖：請參考附錄(CO.pptx)。

(6)ALU Control:

(a) 設計重點說明：

- Data Flow Modeling 設計。
- 本模組為組合邏輯(Combinational Logic)。

(b) Datapath 與詳細架構圖：請參考附錄(CO.pptx)。

- 以上皆不可包含延遲敘述。
- 不接受迴圈形式的設計。
- 不能使用 always@(*)敘述。

程式架構：

TotalALU:

ALUControl

ALU

ALUbit

FullAdder

MUX4_1

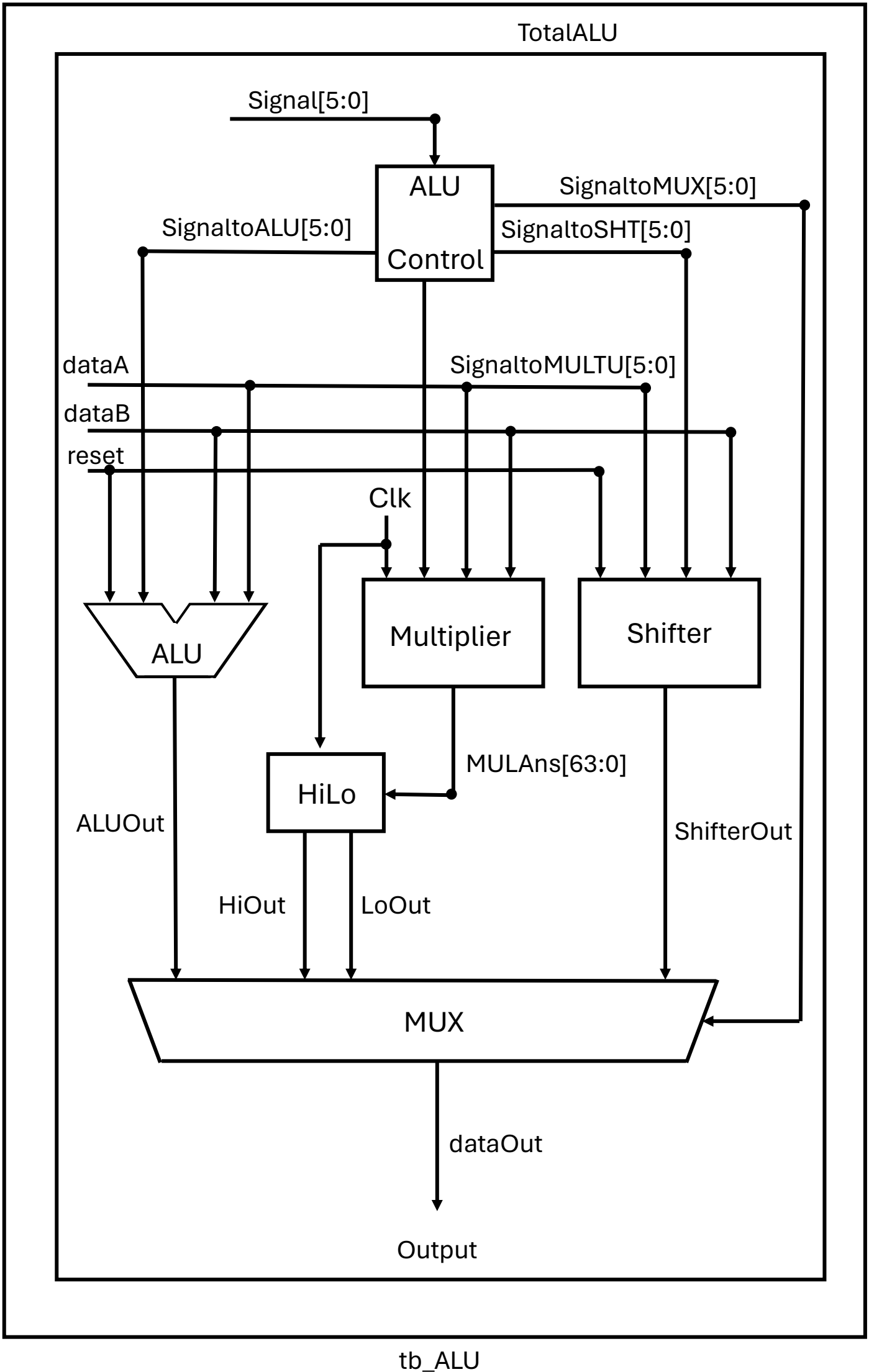
Multiplier

Shifter

MUX2_1

HiLo

MUX


TotalALU
Signal[5:0]
ALU
Control
SignaltoMUX[5:0]
SignaltoALU[5:0]
SignaltoSHT[5:0]
dataA
SignaltoMULTU[5:0]
dataB
reset
Clk
ALU
Multiplier
Shifter
HiLo
MULAns[63:0]
ALUOut
ShifterOut
HiOut
LoOut
MUX
dataOut
Output
tb_ALU

三、結果

(1) TotalALU:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module TotalALU 可連接的 ports
module TotalALU( clk, dataA, dataB, Signal, Output, reset );
// 定義哪些 ports 為 input, 哪些為 output
input reset ;
input clk ;
input [31:0] dataA ;
input [31:0] dataB ;
input [5:0] Signal ;
output [31:0] Output ;

wire [31:0] temp ;
```

```
// 定義各種參數常數(可提升可讀性)
parameter AND = 6'b100100;
parameter OR  = 6'b100101;
parameter ADD = 6'b100000;
parameter SUB = 6'b100010;
parameter SLT = 6'b101010;

parameter SLL = 6'b000010;

parameter MULTU= 6'b011001;
parameter MFHI= 6'b010000;
parameter MFLO= 6'b010010;
```

```
// 宣告 6 位元 wire, 給 ALUControl 以訊號控制每個 module
wire [5:0]  SignaltoALU ;
wire [5:0]  SignaltoSHT ;
wire [5:0]  SignaltoMULTU ;
wire [5:0]  SignaltoMUX ;
// 宣告 32 位元 wire, 為每個 module 的運算結果
wire [31:0] ALUOut, HiOut, LoOut, ShifterOut ;
wire [31:0] dataOut ;
// 32 位元乘法運算的結果為 64 位元
wire [63:0] MulAns ;
```

```
// 將 wire 接給每個 module
ALUControl ALUControl( .clk(clk), .Signal(Signal), .SignaltoALU(SignaltoALU),
                       .SignaltoSHT(SignaltoSHT), .SignaltoMULTU(SignaltoMULTU), .SignaltoMUX(SignaltoMUX) );
ALU ALU( .dataA(dataA), .dataB(dataB), .Signal(SignaltoALU), .dataOut(ALUOut), .reset(reset) );
Multiplier Multiplier( .clk(clk), .dataA(dataA), .dataB(dataB), .Signal(SignaltoMULTU), .dataOut(MulAns), .reset(reset) );
Shifter Shifter( .dataA(dataA), .dataB(dataB), .Signal(SignaltoSHT), .dataOut(ShifterOut), .reset(reset) );
HiLo HiLo( .clk(clk), .MulAns(MulAns), .HiOut(HiOut), .LoOut(LoOut), .reset(reset) );
MUX MUX( .ALUOut(ALUOut), .HiOut(HiOut), .LoOut(LoOut), .Shifter(ShifterOut), .Signal(SignaltoMUX), .dataOut(dataOut) );

// 把多工器的輸出設給最終輸出
assign Output = dataOut ;

endmodule
```

(2) ALUControl:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module ALUControl 可連接的 ports
module ALUControl( clk, Signal, SignaltoALU, SignaltoSHT, SignaltoMULTU, SignaltoMUX );
// 定義哪些 ports 為 input, 哪些為 output
input clk ;
input [5:0] Signal ;
output [5:0] SignaltoALU ;
output [5:0] SignaltoSHT ;
output [5:0] SignaltoMULTU ;
output [5:0] SignaltoMUX ;

// 宣告 6 位元與 7 位元的暫存器
reg [5:0] temp ;
reg [6:0] counter ;
```

```
// 定義參數常數(可提升可讀性)
// Signal (6-bits)
  parameter AND = 6'b100100; //   AND  : 36
  parameter OR  = 6'b100101; //   OR   : 37
  parameter ADD = 6'b100000; //   ADD  : 32
  parameter SUB = 6'b100010; //   SUB  : 34
  parameter SLT = 6'b101010; //   SLT  : 42
  parameter SLL = 6'b000000; //   SLL  : 00
  parameter MULTU= 6'b011001; //   MULTU: 25
```

```
  // 每當 Signal 有變化時, 驅動以下電路
  always@( Signal )
  begin
    if ( Signal == MULTU )
    begin
      // 若當前訊號為乘法運算, 初始化 counter 為 0
      counter = 0 ;
    end
  end
```

```
// 定義電路以 clk 正緣觸發
always@( posedge clk )
begin
    // 將 temp 設為 Signal 的值
    temp = Signal ;
    if ( Signal == MULTU )
    begin
        // 若當前訊號為乘法運算, 將 counter + 1
        counter = counter + 1 ;
        if ( counter == 32 )
        begin
            // 若 counter 為 32, 將 temp 設為 0b111111 以開啟 HiLo 的輸出
            temp = 6'b111111 ; // Open HiLo reg for Mul
            counter = 0 ; // 將 counter 重置為 0
        end
    end

end
```

```
56    // 將要給每個 module 的訊號設為 temp
57    assign SignaltoALU = temp ;
58    assign SignaltoSHT = temp ;
59    assign SignaltoMULTU = temp ;
60    assign SignaltoMUX = temp ;
61
62    endmodule
```

(3) ALU:

```
1     // 設定時間尺度
2     `timescale 1ns/1ns
3     // 定義 module ALU 可連接的 ports
4     module ALU(dataA, dataB, Signal, dataOut, reset);
5
6     // 定義哪些 ports 為 input, 哪些為 output
7     input [31:0] dataA;
8     input [31:0] dataB;
9     input [5:0] Signal;
10    output [31:0] dataOut;
11    input reset;
12
13    // 宣告兩 wire, cout 為 32 位元, set 為一位元
14    wire [31:0] cout;
15    wire set, bitInvert;
16
17    // Signal (6-bits)
18    // 定義各種參數常數(可提高可讀性)
19    parameter AND = 6'b100100; //  AND  : 36
20    parameter OR  = 6'b100101; //  OR   : 37
21    parameter ADD = 6'b100000; //  ADD  : 32
22    parameter SUB = 6'b100010; //  SUB  : 34
23    parameter SLT = 6'b101010; //  SLT  : 42
```

```
29    // 因為是 32-bit 運算, 將 32 個 ALU 單元串起來
30    ALUbit alu0( .a(dataA[0]), .b(dataB[0]), .bitInvert(bitInvert), .cin(cin), .less(set), .operation(Signal), .dataOut(dataOut[0]), .set(), .cout(cout[0]) );
31    ALUbit alu1( .a(dataA[1]), .b(dataB[1]), .bitInvert(bitInvert), .cin(cout[0]), .less(1'b0), .operation(Signal), .dataOut(dataOut[1]), .set(), .cout(cout[1]) );
32    ALUbit alu2( .a(dataA[2]), .b(dataB[2]), .bitInvert(bitInvert), .cin(cout[1]), .less(1'b0), .operation(Signal), .dataOut(dataOut[2]), .set(), .cout(cout[2]) );
33    ALUbit alu3( .a(dataA[3]), .b(dataB[3]), .bitInvert(bitInvert), .cin(cout[2]), .less(1'b0), .operation(Signal), .dataOut(dataOut[3]), .set(), .cout(cout[3]) );
34    ALUbit alu4( .a(dataA[4]), .b(dataB[4]), .bitInvert(bitInvert), .cin(cout[3]), .less(1'b0), .operation(Signal), .dataOut(dataOut[4]), .set(), .cout(cout[4]) );
35    ALUbit alu5( .a(dataA[5]), .b(dataB[5]), .bitInvert(bitInvert), .cin(cout[4]), .less(1'b0), .operation(Signal), .dataOut(dataOut[5]), .set(), .cout(cout[5]) );
36    ALUbit alu6( .a(dataA[6]), .b(dataB[6]), .bitInvert(bitInvert), .cin(cout[5]), .less(1'b0), .operation(Signal), .dataOut(dataOut[6]), .set(), .cout(cout[6]) );
37    ALUbit alu7( .a(dataA[7]), .b(dataB[7]), .bitInvert(bitInvert), .cin(cout[6]), .less(1'b0), .operation(Signal), .dataOut(dataOut[7]), .set(), .cout(cout[7]) );
38    ALUbit alu8( .a(dataA[8]), .b(dataB[8]), .bitInvert(bitInvert), .cin(cout[7]), .less(1'b0), .operation(Signal), .dataOut(dataOut[8]), .set(), .cout(cout[8]) );
39    ALUbit alu9( .a(dataA[9]), .b(dataB[9]), .bitInvert(bitInvert), .cin(cout[8]), .less(1'b0), .operation(Signal), .dataOut(dataOut[9]), .set(), .cout(cout[9]) );
40    ALUbit alu10( .a(dataA[10]), .b(dataB[10]), .bitInvert(bitInvert), .cin(cout[9]), .less(1'b0), .operation(Signal), .dataOut(dataOut[10]), .set(), .cout(cout[10]) );
41    ALUbit alu11( .a(dataA[11]), .b(dataB[11]), .bitInvert(bitInvert), .cin(cout[10]), .less(1'b0), .operation(Signal), .dataOut(dataOut[11]), .set(), .cout(cout[11]) );
42    ALUbit alu12( .a(dataA[12]), .b(dataB[12]), .bitInvert(bitInvert), .cin(cout[11]), .less(1'b0), .operation(Signal), .dataOut(dataOut[12]), .set(), .cout(cout[12]) );
43    ALUbit alu13( .a(dataA[13]), .b(dataB[13]), .bitInvert(bitInvert), .cin(cout[12]), .less(1'b0), .operation(Signal), .dataOut(dataOut[13]), .set(), .cout(cout[13]) );
44    ALUbit alu14( .a(dataA[14]), .b(dataB[14]), .bitInvert(bitInvert), .cin(cout[13]), .less(1'b0), .operation(Signal), .dataOut(dataOut[14]), .set(), .cout(cout[14]) );
45    ALUbit alu15( .a(dataA[15]), .b(dataB[15]), .bitInvert(bitInvert), .cin(cout[14]), .less(1'b0), .operation(Signal), .dataOut(dataOut[15]), .set(), .cout(cout[15]) );
46    ALUbit alu16( .a(dataA[16]), .b(dataB[16]), .bitInvert(bitInvert), .cin(cout[15]), .less(1'b0), .operation(Signal), .dataOut(dataOut[16]), .set(), .cout(cout[16]) );
47    ALUbit alu17( .a(dataA[17]), .b(dataB[17]), .bitInvert(bitInvert), .cin(cout[16]), .less(1'b0), .operation(Signal), .dataOut(dataOut[17]), .set(), .cout(cout[17]) );
48    ALUbit alu18( .a(dataA[18]), .b(dataB[18]), .bitInvert(bitInvert), .cin(cout[17]), .less(1'b0), .operation(Signal), .dataOut(dataOut[18]), .set(), .cout(cout[18]) );
49    ALUbit alu19( .a(dataA[19]), .b(dataB[19]), .bitInvert(bitInvert), .cin(cout[18]), .less(1'b0), .operation(Signal), .dataOut(dataOut[19]), .set(), .cout(cout[19]) );
50    ALUbit alu20( .a(dataA[20]), .b(dataB[20]), .bitInvert(bitInvert), .cin(cout[19]), .less(1'b0), .operation(Signal), .dataOut(dataOut[20]), .set(), .cout(cout[20]) );
51    ALUbit alu21( .a(dataA[21]), .b(dataB[21]), .bitInvert(bitInvert), .cin(cout[20]), .less(1'b0), .operation(Signal), .dataOut(dataOut[21]), .set(), .cout(cout[21]) );
52    ALUbit alu22( .a(dataA[22]), .b(dataB[22]), .bitInvert(bitInvert), .cin(cout[21]), .less(1'b0), .operation(Signal), .dataOut(dataOut[22]), .set(), .cout(cout[22]) );
53    ALUbit alu23( .a(dataA[23]), .b(dataB[23]), .bitInvert(bitInvert), .cin(cout[22]), .less(1'b0), .operation(Signal), .dataOut(dataOut[23]), .set(), .cout(cout[23]) );
54    ALUbit alu24( .a(dataA[24]), .b(dataB[24]), .bitInvert(bitInvert), .cin(cout[23]), .less(1'b0), .operation(Signal), .dataOut(dataOut[24]), .set(), .cout(cout[24]) );
55    ALUbit alu25( .a(dataA[25]), .b(dataB[25]), .bitInvert(bitInvert), .cin(cout[24]), .less(1'b0), .operation(Signal), .dataOut(dataOut[25]), .set(), .cout(cout[25]) );
56    ALUbit alu26( .a(dataA[26]), .b(dataB[26]), .bitInvert(bitInvert), .cin(cout[25]), .less(1'b0), .operation(Signal), .dataOut(dataOut[26]), .set(), .cout(cout[26]) );
57    ALUbit alu27( .a(dataA[27]), .b(dataB[27]), .bitInvert(bitInvert), .cin(cout[26]), .less(1'b0), .operation(Signal), .dataOut(dataOut[27]), .set(), .cout(cout[27]) );
58    ALUbit alu28( .a(dataA[28]), .b(dataB[28]), .bitInvert(bitInvert), .cin(cout[27]), .less(1'b0), .operation(Signal), .dataOut(dataOut[28]), .set(), .cout(cout[28]) );
59    ALUbit alu29( .a(dataA[29]), .b(dataB[29]), .bitInvert(bitInvert), .cin(cout[28]), .less(1'b0), .operation(Signal), .dataOut(dataOut[29]), .set(), .cout(cout[29]) );
60    ALUbit alu30( .a(dataA[30]), .b(dataB[30]), .bitInvert(bitInvert), .cin(cout[29]), .less(1'b0), .operation(Signal), .dataOut(dataOut[30]), .set(), .cout(cout[30]) );
61    ALUbit alu31( .a(dataA[31]), .b(dataB[31]), .bitInvert(bitInvert), .cin(cout[30]), .less(1'b0), .operation(Signal), .dataOut(dataOut[31]), .set(set), .cout(cout[31]) );
62
63    endmodule
```

(4) ALUbit:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module ALUbit 可連接的 ports
module ALUbit( a, b, bitInvert, cin, less, operation, dataOut, set, cout );
// 1 bit ALU
// 定義哪些 ports 為 input, 哪些為 output
input a, b, bitInvert, cin, less ;
input [5:0] operation ;
output dataOut, set, cout;

// 定義 wire 以連接每個 module
wire cin;
wire andOut, orOut, xorOut;

// 分別對 a, b 進行 AND, OR 運算
and(andOut, a, b);
or(orOut, a, b);
// 將 b 與 bitInvert 運算, 若 bitInvert 為 1, 則會將 b 進行 invert( 1 變 0, 0 變 1 )
xor(xorOut, bitInvert, b);
// 把 a 與處理過的 b 給加法器運算
FullAdder U_FA( .a(a), .b(xorOut), .cin(cin), .sum(set), .cout(cout) );
// 多工器以 operation 判斷要哪個運算的結果
MUX4_1 U_Mux4_1( .out(dataOut), .in0(andOut), .in1(orOut), .in2(set), .in3(less), .sel(operation) );

endmodule
```

(5) FullAdder:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module FullAdder 可連接的 ports
module FullAdder( a, b, cin, sum, cout );
// 定義哪些 ports 為 input, 哪些為 output
input a, b, cin;
output sum, cout;
// 根據加法器運算, sum 為 a XOR b XOR cin
assign sum = a ^ b ^ cin;
// cout 為 ( a AND b ) OR ( ( A XOR B ) AND cin )
assign cout = (a & b) | ((a ^ b) & cin);

endmodule
```

(6) MUX4_1:

```
// 定義 module MUX4_1 可連接的 ports
module MUX4_1(out, in3, in2, in1, in0, sel);
// 定義哪些 ports 為 input, 哪些為 output
output  out;
input   in3, in2, in1, in0;
input   [5:0]sel;

wire    out;

// Continuous assigment
// 根據 sel 選擇要輸出 in0 還是 in1 還是 in2 還是 in3
assign  out = (sel==6'b100100) ? in0 : (sel==6'b100101) ? in1 : (sel==6'b100000 || sel==6'b100010) ? in2 : in3;

endmodule
```

(7) Multiplier:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義哪些 ports 為 input, 哪些為 output
module Multiplier( clk, dataA, dataB, Signal, dataOut, reset ); // 2nd Multiplier

// 定義哪些 ports 為 input, 哪些為 output
input clk ;
input reset ;
input [31:0] dataA ; // Multiplicand
input [31:0] dataB ;
input [5:0] Signal ;
output reg [63:0] dataOut ;

// 宣告兩暫存器, product 為 64 位元, 乘數與被乘數 為 32 位元
reg [63:0] product;
reg [31:0] multiply, multiplicand ;
parameter MULTU = 6'b011001;
parameter OUT = 6'b111111;


 // 每當 dataA 有變化時, 將被乘數設為 dataA 的值
 always@(dataA)
begin
    multiplicand = dataA;
end


 // 每當 dataB 有變化時, 將乘數設為 dataB 的值
 always@(dataB)
begin
    multiply = dataB;
end

// 定義電路以 clk 或 reset 正緣觸發
always@( posedge clk or reset )
begin
    // 當 reset 為 1 時, 將 product 設為 0
    if ( reset )
    begin
        product <= 64'd0 ;
    end

    else
    begin
        case ( Signal )
        MULTU :
        begin
            // 若未到達 32 次, 判斷乘數第 0 位是否為 1, 如果是則將被乘數加到 product 左半部
            if (multiply[0] == 1'b1)
            begin
                product[63:32] = product[63:32] + multiplicand;
            end
            // 將乘積與乘數右移 1 位元
            product = product >> 1;
            multiply = multiply >> 1;
        end
        OUT :
        begin
            // 當 32 次做完後, 把結果給 dataOut
            dataOut = product ;
        end
        endcase
    end

end
endmodule
```

(8) Shifter:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module Shifter 可連接的 ports
module Shifter( dataA, dataB, Signal, dataOut, reset );
// 定義哪些 ports 為 input, 哪些為 output
input reset ;
input [31:0] dataA ;
input [31:0] dataB ;
input [5:0] Signal ;
output [31:0] dataOut ;

// orginal reg, reg only can used in always block and initial block, combinational block can't use reg
// 宣告 Barrel Shifter 五層的輸出為 wire
wire [31:0] temp, temp1, temp2, temp3, temp4 ;

// 定義參數常數
parameter SLL = 6'b000000;
```

```
MUX2_1 mux1_0( .in0(dataA[31]), .in1(dataA[30]), .sel(dataB[0]), .out(temp[31]) );
MUX2_1 mux1_1( .in0(dataA[30]), .in1(dataA[29]), .sel(dataB[0]), .out(temp[30]) );
MUX2_1 mux1_2( .in0(dataA[29]), .in1(dataA[28]), .sel(dataB[0]), .out(temp[29]) );
MUX2_1 mux1_3( .in0(dataA[28]), .in1(dataA[27]), .sel(dataB[0]), .out(temp[28]) );
MUX2_1 mux1_4( .in0(dataA[27]), .in1(dataA[26]), .sel(dataB[0]), .out(temp[27]) );
MUX2_1 mux1_5( .in0(dataA[26]), .in1(dataA[25]), .sel(dataB[0]), .out(temp[26]) );
MUX2_1 mux1_6( .in0(dataA[25]), .in1(dataA[24]), .sel(dataB[0]), .out(temp[25]) );
MUX2_1 mux1_7( .in0(dataA[24]), .in1(dataA[23]), .sel(dataB[0]), .out(temp[24]) );
MUX2_1 mux1_8( .in0(dataA[23]), .in1(dataA[22]), .sel(dataB[0]), .out(temp[23]) );
MUX2_1 mux1_9( .in0(dataA[22]), .in1(dataA[21]), .sel(dataB[0]), .out(temp[22]) );
MUX2_1 mux1_10( .in0(dataA[21]), .in1(dataA[20]), .sel(dataB[0]), .out(temp[21]) );
MUX2_1 mux1_11( .in0(dataA[20]), .in1(dataA[19]), .sel(dataB[0]), .out(temp[20]) );
MUX2_1 mux1_12( .in0(dataA[19]), .in1(dataA[18]), .sel(dataB[0]), .out(temp[19]) );
MUX2_1 mux1_13( .in0(dataA[18]), .in1(dataA[17]), .sel(dataB[0]), .out(temp[18]) );
MUX2_1 mux1_14( .in0(dataA[17]), .in1(dataA[16]), .sel(dataB[0]), .out(temp[17]) );
MUX2_1 mux1_15( .in0(dataA[16]), .in1(dataA[15]), .sel(dataB[0]), .out(temp[16]) );
MUX2_1 mux1_16( .in0(dataA[15]), .in1(dataA[14]), .sel(dataB[0]), .out(temp[15]) );
MUX2_1 mux1_17( .in0(dataA[14]), .in1(dataA[13]), .sel(dataB[0]), .out(temp[14]) );
MUX2_1 mux1_18( .in0(dataA[13]), .in1(dataA[12]), .sel(dataB[0]), .out(temp[13]) );
MUX2_1 mux1_19( .in0(dataA[12]), .in1(dataA[11]), .sel(dataB[0]), .out(temp[12]) );
MUX2_1 mux1_20( .in0(dataA[11]), .in1(dataA[10]), .sel(dataB[0]), .out(temp[11]) );
MUX2_1 mux1_21( .in0(dataA[10]), .in1(dataA[9]), .sel(dataB[0]), .out(temp[10]) );
MUX2_1 mux1_22( .in0(dataA[9]), .in1(dataA[8]), .sel(dataB[0]), .out(temp[9]) );
MUX2_1 mux1_23( .in0(dataA[8]), .in1(dataA[7]), .sel(dataB[0]), .out(temp[8]) );
MUX2_1 mux1_24( .in0(dataA[7]), .in1(dataA[6]), .sel(dataB[0]), .out(temp[7]) );
MUX2_1 mux1_25( .in0(dataA[6]), .in1(dataA[5]), .sel(dataB[0]), .out(temp[6]) );
MUX2_1 mux1_26( .in0(dataA[5]), .in1(dataA[4]), .sel(dataB[0]), .out(temp[5]) );
MUX2_1 mux1_27( .in0(dataA[4]), .in1(dataA[3]), .sel(dataB[0]), .out(temp[4]) );
MUX2_1 mux1_28( .in0(dataA[3]), .in1(dataA[2]), .sel(dataB[0]), .out(temp[3]) );
MUX2_1 mux1_29( .in0(dataA[2]), .in1(dataA[1]), .sel(dataB[0]), .out(temp[2]) );
MUX2_1 mux1_30( .in0(dataA[1]), .in1(dataA[0]), .sel(dataB[0]), .out(temp[1]) );
MUX2_1 mux1_31( .in0(dataA[0]), .in1(1'b0), .sel(dataB[0]), .out(temp[0]) );
```

```
52  // 第二層，2 位元左移
53  MUX2_1 mux2_0( .in0(temp[31]), .in1(temp[29]), .sel(dataB[1]), .out(temp1[31]) );
54  MUX2_1 mux2_1( .in0(temp[30]), .in1(temp[28]), .sel(dataB[1]), .out(temp1[30]) );
55  MUX2_1 mux2_2( .in0(temp[29]), .in1(temp[27]), .sel(dataB[1]), .out(temp1[29]) );
56  MUX2_1 mux2_3( .in0(temp[28]), .in1(temp[26]), .sel(dataB[1]), .out(temp1[28]) );
57  MUX2_1 mux2_4( .in0(temp[27]), .in1(temp[25]), .sel(dataB[1]), .out(temp1[27]) );
58  MUX2_1 mux2_5( .in0(temp[26]), .in1(temp[24]), .sel(dataB[1]), .out(temp1[26]) );
59  MUX2_1 mux2_6( .in0(temp[25]), .in1(temp[23]), .sel(dataB[1]), .out(temp1[25]) );
60  MUX2_1 mux2_7( .in0(temp[24]), .in1(temp[22]), .sel(dataB[1]), .out(temp1[24]) );
61  MUX2_1 mux2_8( .in0(temp[23]), .in1(temp[21]), .sel(dataB[1]), .out(temp1[23]) );
62  MUX2_1 mux2_9( .in0(temp[22]), .in1(temp[20]), .sel(dataB[1]), .out(temp1[22]) );
63  MUX2_1 mux2_10( .in0(temp[21]), .in1(temp[19]), .sel(dataB[1]), .out(temp1[21]) );
64  MUX2_1 mux2_11( .in0(temp[20]), .in1(temp[18]), .sel(dataB[1]), .out(temp1[20]) );
65  MUX2_1 mux2_12( .in0(temp[19]), .in1(temp[17]), .sel(dataB[1]), .out(temp1[19]) );
66  MUX2_1 mux2_13( .in0(temp[18]), .in1(temp[16]), .sel(dataB[1]), .out(temp1[18]) );
67  MUX2_1 mux2_14( .in0(temp[17]), .in1(temp[15]), .sel(dataB[1]), .out(temp1[17]) );
68  MUX2_1 mux2_15( .in0(temp[16]), .in1(temp[14]), .sel(dataB[1]), .out(temp1[16]) );
69  MUX2_1 mux2_16( .in0(temp[15]), .in1(temp[13]), .sel(dataB[1]), .out(temp1[15]) );
70  MUX2_1 mux2_17( .in0(temp[14]), .in1(temp[12]), .sel(dataB[1]), .out(temp1[14]) );
71  MUX2_1 mux2_18( .in0(temp[13]), .in1(temp[11]), .sel(dataB[1]), .out(temp1[13]) );
72  MUX2_1 mux2_19( .in0(temp[12]), .in1(temp[10]), .sel(dataB[1]), .out(temp1[12]) );
73  MUX2_1 mux2_20( .in0(temp[11]), .in1(temp[9]), .sel(dataB[1]), .out(temp1[11]) );
74  MUX2_1 mux2_21( .in0(temp[10]), .in1(temp[8]), .sel(dataB[1]), .out(temp1[10]) );
75  MUX2_1 mux2_22( .in0(temp[9]), .in1(temp[7]), .sel(dataB[1]), .out(temp1[9]) );
76  MUX2_1 mux2_23( .in0(temp[8]), .in1(temp[6]), .sel(dataB[1]), .out(temp1[8]) );
77  MUX2_1 mux2_24( .in0(temp[7]), .in1(temp[5]), .sel(dataB[1]), .out(temp1[7]) );
78  MUX2_1 mux2_25( .in0(temp[6]), .in1(temp[4]), .sel(dataB[1]), .out(temp1[6]) );
79  MUX2_1 mux2_26( .in0(temp[5]), .in1(temp[3]), .sel(dataB[1]), .out(temp1[5]) );
80  MUX2_1 mux2_27( .in0(temp[4]), .in1(temp[2]), .sel(dataB[1]), .out(temp1[4]) );
81  MUX2_1 mux2_28( .in0(temp[3]), .in1(temp[1]), .sel(dataB[1]), .out(temp1[3]) );
82  MUX2_1 mux2_29( .in0(temp[2]), .in1(temp[0]), .sel(dataB[1]), .out(temp1[2]) );
83  MUX2_1 mux2_30( .in0(temp[1]), .in1(1'b0), .sel(dataB[1]), .out(temp1[1]) );
84  MUX2_1 mux2_31( .in0(temp[0]), .in1(1'b0), .sel(dataB[1]), .out(temp1[0]) );
```

```
86  // 第三層，4 位元左移
87  MUX2_1 mux3_0( .in0(temp1[31]), .in1(temp1[27]), .sel(dataB[2]), .out(temp2[31]) );
88  MUX2_1 mux3_1( .in0(temp1[30]), .in1(temp1[26]), .sel(dataB[2]), .out(temp2[30]) );
89  MUX2_1 mux3_2( .in0(temp1[29]), .in1(temp1[25]), .sel(dataB[2]), .out(temp2[29]) );
90  MUX2_1 mux3_3( .in0(temp1[28]), .in1(temp1[24]), .sel(dataB[2]), .out(temp2[28]) );
91  MUX2_1 mux3_4( .in0(temp1[27]), .in1(temp1[23]), .sel(dataB[2]), .out(temp2[27]) );
92  MUX2_1 mux3_5( .in0(temp1[26]), .in1(temp1[22]), .sel(dataB[2]), .out(temp2[26]) );
93  MUX2_1 mux3_6( .in0(temp1[25]), .in1(temp1[21]), .sel(dataB[2]), .out(temp2[25]) );
94  MUX2_1 mux3_7( .in0(temp1[24]), .in1(temp1[20]), .sel(dataB[2]), .out(temp2[24]) );
95  MUX2_1 mux3_8( .in0(temp1[23]), .in1(temp1[19]), .sel(dataB[2]), .out(temp2[23]) );
96  MUX2_1 mux3_9( .in0(temp1[22]), .in1(temp1[18]), .sel(dataB[2]), .out(temp2[22]) );
97  MUX2_1 mux3_10( .in0(temp1[21]), .in1(temp1[17]), .sel(dataB[2]), .out(temp2[21]) );
98  MUX2_1 mux3_11( .in0(temp1[20]), .in1(temp1[16]), .sel(dataB[2]), .out(temp2[20]) );
99  MUX2_1 mux3_12( .in0(temp1[19]), .in1(temp1[15]), .sel(dataB[2]), .out(temp2[19]) );
100 MUX2_1 mux3_13( .in0(temp1[18]), .in1(temp1[14]), .sel(dataB[2]), .out(temp2[18]) );
101 MUX2_1 mux3_14( .in0(temp1[17]), .in1(temp1[13]), .sel(dataB[2]), .out(temp2[17]) );
102 MUX2_1 mux3_15( .in0(temp1[16]), .in1(temp1[12]), .sel(dataB[2]), .out(temp2[16]) );
103 MUX2_1 mux3_16( .in0(temp1[15]), .in1(temp1[11]), .sel(dataB[2]), .out(temp2[15]) );
104 MUX2_1 mux3_17( .in0(temp1[14]), .in1(temp1[10]), .sel(dataB[2]), .out(temp2[14]) );
105 MUX2_1 mux3_18( .in0(temp1[13]), .in1(temp1[9]), .sel(dataB[2]), .out(temp2[13]) );
106 MUX2_1 mux3_19( .in0(temp1[12]), .in1(temp1[8]), .sel(dataB[2]), .out(temp2[12]) );
107 MUX2_1 mux3_20( .in0(temp1[11]), .in1(temp1[7]), .sel(dataB[2]), .out(temp2[11]) );
108 MUX2_1 mux3_21( .in0(temp1[10]), .in1(temp1[6]), .sel(dataB[2]), .out(temp2[10]) );
109 MUX2_1 mux3_22( .in0(temp1[9]), .in1(temp1[5]), .sel(dataB[2]), .out(temp2[9]) );
110 MUX2_1 mux3_23( .in0(temp1[8]), .in1(temp1[4]), .sel(dataB[2]), .out(temp2[8]) );
111 MUX2_1 mux3_24( .in0(temp1[7]), .in1(temp1[3]), .sel(dataB[2]), .out(temp2[7]) );
112 MUX2_1 mux3_25( .in0(temp1[6]), .in1(temp1[2]), .sel(dataB[2]), .out(temp2[6]) );
113 MUX2_1 mux3_26( .in0(temp1[5]), .in1(temp1[1]), .sel(dataB[2]), .out(temp2[5]) );
114 MUX2_1 mux3_27( .in0(temp1[4]), .in1(temp1[0]), .sel(dataB[2]), .out(temp2[4]) );
115 MUX2_1 mux3_28( .in0(temp1[3]), .in1(1'b0), .sel(dataB[2]), .out(temp2[3]) );
116 MUX2_1 mux3_29( .in0(temp1[2]), .in1(1'b0), .sel(dataB[2]), .out(temp2[2]) );
117 MUX2_1 mux3_30( .in0(temp1[1]), .in1(1'b0), .sel(dataB[2]), .out(temp2[1]) );
118 MUX2_1 mux3_31( .in0(temp1[0]), .in1(1'b0), .sel(dataB[2]), .out(temp2[0]) );
```

```
120     // 第四層, 8 位元左移
121     MUX2_1 mux4_0( .in0(temp2[31]), .in1(temp2[23]), .sel(dataB[3]), .out(temp3[31]) );
122     MUX2_1 mux4_1( .in0(temp2[30]), .in1(temp2[22]), .sel(dataB[3]), .out(temp3[30]) );
123     MUX2_1 mux4_2( .in0(temp2[29]), .in1(temp2[21]), .sel(dataB[3]), .out(temp3[29]) );
124     MUX2_1 mux4_3( .in0(temp2[28]), .in1(temp2[20]), .sel(dataB[3]), .out(temp3[28]) );
125     MUX2_1 mux4_4( .in0(temp2[27]), .in1(temp2[19]), .sel(dataB[3]), .out(temp3[27]) );
126     MUX2_1 mux4_5( .in0(temp2[26]), .in1(temp2[18]), .sel(dataB[3]), .out(temp3[26]) );
127     MUX2_1 mux4_6( .in0(temp2[25]), .in1(temp2[17]), .sel(dataB[3]), .out(temp3[25]) );
128     MUX2_1 mux4_7( .in0(temp2[24]), .in1(temp2[16]), .sel(dataB[3]), .out(temp3[24]) );
129     MUX2_1 mux4_8( .in0(temp2[23]), .in1(temp2[15]), .sel(dataB[3]), .out(temp3[23]) );
130     MUX2_1 mux4_9( .in0(temp2[22]), .in1(temp2[14]), .sel(dataB[3]), .out(temp3[22]) );
131     MUX2_1 mux4_10( .in0(temp2[21]), .in1(temp2[13]), .sel(dataB[3]), .out(temp3[21]) );
132     MUX2_1 mux4_11( .in0(temp2[20]), .in1(temp2[12]), .sel(dataB[3]), .out(temp3[20]) );
133     MUX2_1 mux4_12( .in0(temp2[19]), .in1(temp2[11]), .sel(dataB[3]), .out(temp3[19]) );
134     MUX2_1 mux4_13( .in0(temp2[18]), .in1(temp2[10]), .sel(dataB[3]), .out(temp3[18]) );
135     MUX2_1 mux4_14( .in0(temp2[17]), .in1(temp2[9]), .sel(dataB[3]), .out(temp3[17]) );
136     MUX2_1 mux4_15( .in0(temp2[16]), .in1(temp2[8]), .sel(dataB[3]), .out(temp3[16]) );
137     MUX2_1 mux4_16( .in0(temp2[15]), .in1(temp2[7]), .sel(dataB[3]), .out(temp3[15]) );
138     MUX2_1 mux4_17( .in0(temp2[14]), .in1(temp2[6]), .sel(dataB[3]), .out(temp3[14]) );
139     MUX2_1 mux4_18( .in0(temp2[13]), .in1(temp2[5]), .sel(dataB[3]), .out(temp3[13]) );
140     MUX2_1 mux4_19( .in0(temp2[12]), .in1(temp2[4]), .sel(dataB[3]), .out(temp3[12]) );
141     MUX2_1 mux4_20( .in0(temp2[11]), .in1(temp2[3]), .sel(dataB[3]), .out(temp3[11]) );
142     MUX2_1 mux4_21( .in0(temp2[10]), .in1(temp2[2]), .sel(dataB[3]), .out(temp3[10]) );
143     MUX2_1 mux4_22( .in0(temp2[9]), .in1(temp2[1]), .sel(dataB[3]), .out(temp3[9]) );
144     MUX2_1 mux4_23( .in0(temp2[8]), .in1(temp2[0]), .sel(dataB[3]), .out(temp3[8]) );
145     MUX2_1 mux4_24( .in0(temp2[7]), .in1(1'b0), .sel(dataB[3]), .out(temp3[7]) );
146     MUX2_1 mux4_25( .in0(temp2[6]), .in1(1'b0), .sel(dataB[3]), .out(temp3[6]) );
147     MUX2_1 mux4_26( .in0(temp2[5]), .in1(1'b0), .sel(dataB[3]), .out(temp3[5]) );
148     MUX2_1 mux4_27( .in0(temp2[4]), .in1(1'b0), .sel(dataB[3]), .out(temp3[4]) );
149     MUX2_1 mux4_28( .in0(temp2[3]), .in1(1'b0), .sel(dataB[3]), .out(temp3[3]) );
150     MUX2_1 mux4_29( .in0(temp2[2]), .in1(1'b0), .sel(dataB[3]), .out(temp3[2]) );
151     MUX2_1 mux4_30( .in0(temp2[1]), .in1(1'b0), .sel(dataB[3]), .out(temp3[1]) );
152     MUX2_1 mux4_31( .in0(temp2[0]), .in1(1'b0), .sel(dataB[3]), .out(temp3[0]) );

154     // 第五層, 16 位元左移
155     MUX2_1 mux5_0( .in0(temp3[31]), .in1(temp3[15]), .sel(dataB[4]), .out(temp4[31]) );
156     MUX2_1 mux5_1( .in0(temp3[30]), .in1(temp3[14]), .sel(dataB[4]), .out(temp4[30]) );
157     MUX2_1 mux5_2( .in0(temp3[29]), .in1(temp3[13]), .sel(dataB[4]), .out(temp4[29]) );
158     MUX2_1 mux5_3( .in0(temp3[28]), .in1(temp3[12]), .sel(dataB[4]), .out(temp4[28]) );
159     MUX2_1 mux5_4( .in0(temp3[27]), .in1(temp3[11]), .sel(dataB[4]), .out(temp4[27]) );
160     MUX2_1 mux5_5( .in0(temp3[26]), .in1(temp3[10]), .sel(dataB[4]), .out(temp4[26]) );
161     MUX2_1 mux5_6( .in0(temp3[25]), .in1(temp3[9]), .sel(dataB[4]), .out(temp4[25]) );
162     MUX2_1 mux5_7( .in0(temp3[24]), .in1(temp3[8]), .sel(dataB[4]), .out(temp4[24]) );
163     MUX2_1 mux5_8( .in0(temp3[23]), .in1(temp3[7]), .sel(dataB[4]), .out(temp4[23]) );
164     MUX2_1 mux5_9( .in0(temp3[22]), .in1(temp3[6]), .sel(dataB[4]), .out(temp4[22]) );
165     MUX2_1 mux5_10( .in0(temp3[21]), .in1(temp3[5]), .sel(dataB[4]), .out(temp4[21]) );
166     MUX2_1 mux5_11( .in0(temp3[20]), .in1(temp3[4]), .sel(dataB[4]), .out(temp4[20]) );
167     MUX2_1 mux5_12( .in0(temp3[19]), .in1(temp3[3]), .sel(dataB[4]), .out(temp4[19]) );
168     MUX2_1 mux5_13( .in0(temp3[18]), .in1(temp3[2]), .sel(dataB[4]), .out(temp4[18]) );
169     MUX2_1 mux5_14( .in0(temp3[17]), .in1(temp3[1]), .sel(dataB[4]), .out(temp4[17]) );
170     MUX2_1 mux5_15( .in0(temp3[16]), .in1(temp3[0]), .sel(dataB[4]), .out(temp4[16]) );
171     MUX2_1 mux5_16( .in0(temp3[15]), .in1(1'b0), .sel(dataB[4]), .out(temp4[15]) );
172     MUX2_1 mux5_17( .in0(temp3[14]), .in1(1'b0), .sel(dataB[4]), .out(temp4[14]) );
173     MUX2_1 mux5_18( .in0(temp3[13]), .in1(1'b0), .sel(dataB[4]), .out(temp4[13]) );
174     MUX2_1 mux5_19( .in0(temp3[12]), .in1(1'b0), .sel(dataB[4]), .out(temp4[12]) );
175     MUX2_1 mux5_20( .in0(temp3[11]), .in1(1'b0), .sel(dataB[4]), .out(temp4[11]) );
176     MUX2_1 mux5_21( .in0(temp3[10]), .in1(1'b0), .sel(dataB[4]), .out(temp4[10]) );
177     MUX2_1 mux5_22( .in0(temp3[9]), .in1(1'b0), .sel(dataB[4]), .out(temp4[9]) );
178     MUX2_1 mux5_23( .in0(temp3[8]), .in1(1'b0), .sel(dataB[4]), .out(temp4[8]) );
179     MUX2_1 mux5_24( .in0(temp3[7]), .in1(1'b0), .sel(dataB[4]), .out(temp4[7]) );
180     MUX2_1 mux5_25( .in0(temp3[6]), .in1(1'b0), .sel(dataB[4]), .out(temp4[6]) );
181     MUX2_1 mux5_26( .in0(temp3[5]), .in1(1'b0), .sel(dataB[4]), .out(temp4[5]) );
182     MUX2_1 mux5_27( .in0(temp3[4]), .in1(1'b0), .sel(dataB[4]), .out(temp4[4]) );
183     MUX2_1 mux5_28( .in0(temp3[3]), .in1(1'b0), .sel(dataB[4]), .out(temp4[3]) );
184     MUX2_1 mux5_29( .in0(temp3[2]), .in1(1'b0), .sel(dataB[4]), .out(temp4[2]) );
185     MUX2_1 mux5_30( .in0(temp3[1]), .in1(1'b0), .sel(dataB[4]), .out(temp4[1]) );
186     MUX2_1 mux5_31( .in0(temp3[0]), .in1(1'b0), .sel(dataB[4]), .out(temp4[0]) );

188     // 若訊號為左移, 將 dataOut 設為第五層的輸出
189     assign dataOut = (Signal == SLL) ? temp4 : 32'b0 ;
190
191     endmodule
```

(9)MUX2_1:

```
// 定義 module MUX2_1 可連接的 ports
module MUX2_1(in0, in1, sel, out);
    // 定義哪些 ports 為 input, 哪些為 output
    input   in0, in1;
    input   sel;
    output  out;

    // 根據 sel 選擇要輸出 in1 還是 in0
    assign  out = sel ? in1 : in0;
endmodule
```

(10) HiLo:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module HiLo 可連接的 ports
module HiLo( clk, MulAns, HiOut, LoOut, reset );
// 定義哪些 ports 為 input, 哪些為 output
input clk ;
input reset ;
input [63:0] MulAns ;
output [31:0] HiOut ;
output [31:0] LoOut ;

// 定義一個 64 位元暫存器
reg [63:0] HiLo ;
```

```
// 定義電路以 clk 或 reset 正緣觸發
always@( posedge clk or reset )
begin
    // 若 reset 為 1, 則 HiLo 設為 0
    if ( reset )
    begin
        HiLo = 64'b0 ;
    end

    else
    begin
        // 否則設為乘法運算結果
        HiLo = MulAns ;
    end
end

// 因 32 位元乘法運算後結果為 64 位元, 將結果拆成兩半輸出
assign HiOut = HiLo[63:32] ;
assign LoOut = HiLo[31:0] ;

endmodule
```

(11) MUX:

```
// 設定時間尺度
`timescale 1ns/1ns
// 定義 module MUX 可連接的 ports
module MUX( ALUOut, HiOut, LoOut, Shifter, Signal, dataOut );
// 定義哪些 ports 為 input, 哪些為 output
input [31:0] ALUOut ;
input [31:0] HiOut ;
input [31:0] LoOut ;
input [31:0] Shifter ;
input [5:0] Signal ;
output [31:0] dataOut ;

reg [31:0] temp ;
```

```
// 定義各種參數常數(可提高可讀性)
parameter AND = 6'b100100;
parameter OR  = 6'b100101;
parameter ADD = 6'b100000;
parameter SUB = 6'b100010;
parameter SLT = 6'b101010;

parameter SLL = 6'b000000;

parameter MULTU= 6'b011001;
parameter MFHI= 6'b010000;
parameter MFLO= 6'b010010;
```

```
// 若訊號為 AND, OR, ADD, SUB, 或 SLT, 則將 dataOut 設為 ALU 的輸出
// 否則若為乘法運算取高位, 則將 dataOut 設為 Hi 的值
// 否則若為乘法運算取低位, 則將 dataOut 設為 Lo 的值
// 否則若為左移運算, 則將 dataOut 設為 Shifter 的結果
// 否則設為 0
assign dataOut = (Signal == AND || Signal == OR || Signal == ADD || Signal == SUB || Signal == SLT ) ?
ALUOut
:(Signal == MFHI) ?
HiOut
: (Signal == MFLO) ?
LoOut
: (Signal == SLL) ?
Shifter : 32'b0;

endmodule
```

(12)tbALU:

```
`timescale 1ns/ 1ns
module tb_ALU();
    reg clk, rst;
    reg[5:0] ctrl;
    reg[31:0] inputA, inputB, ans;
    wire[31:0] out;
    integer fp_r, fp_r_ans, eof;

    // 產生時脈, 週期：10ns
    initial begin
        clk = 1'b1;
        forever #5 clk = ~clk;
    end
```

```
15      initial begin
16          eof = 0;
17          rst = 1'b1;
18          #10;
19          rst = 1'b0;
20          /*
21              讀取輸入指令，檔名"input.txt"可自行修改
22              每一行為一筆輸入
23              格式為：控制訊號   InputA   InputB
24          */
25          fp_r = $fopen( "input.txt", "r" );
26          /*
27              讀取答案，檔名"ans.txt"可自行修改
28              每一行為一筆正確答案
29          */
30          fp_r_ans = $fopen( "ans.txt", "r" );
31          /*
32              自此開始模擬ALU並比對輸出結果
33              如結果正確，將輸出："Correct"
34              不正確將輸出執行結果與正確答案
35              以上輸出的第一個數字為cycle number
36          */
37          $display( "Start\n" );
38          eof = $fscanf(fp r ans, "%d", ans);
```

```
39          while( eof != -1 ) begin
40              $fscanf(fp_r, "%d%d%d", ctrl, inputA, inputB );
41              $write( "%d: Input: ", $time/10 );
42              if ( ctrl == 6'd36 ) $write( "AND(%d)", ctrl );
43              else if ( ctrl == 6'd37 ) $write( "OR(%d) ", ctrl );
44              else if ( ctrl == 6'd32 ) $write( "ADD(%d) ", ctrl );
45              else if ( ctrl == 6'd34 ) $write( "SUB(%d) ", ctrl );
46              else if ( ctrl == 6'd42 ) $write( "SLT(%d) ", ctrl );
47              else if ( ctrl == 6'd0 ) $write( "SLL(%d) ", ctrl );
48              else if ( ctrl == 6'd25 ) $write( "MULTU(%d) ", ctrl );
49              $display( "%d%d", inputA, inputB  );
```

```
50              if ( ctrl == 32'd25 ) begin
51                  #330;
52                  $display( "%d: Mul End\n", $time/10 );
53
54                  #10;
55                  #10;
56
57                  $display( "                    Move Hi" );
58                  ctrl = 6'd16;
59                  #10;
60                  if ( ans == out )
61                      $display( "%d: Correct: Your answer is:%d,\n                                Correct answer is:%d\n", $time/10, out, ans );
62                  else
63                      $display( "%d: Wrong Answer: Your answer is:%d,\n                                Correct answer is:%d\n", $time/10, out, ans );
64                  $display( "                    Move Lo" );
65                  ctrl = 6'd18;
66                  eof = $fscanf(fp_r_ans, "%d", ans);
67                  #10;
68                  if ( ans == out )
69                      $display( "%d: Correct: Your answer is:%d,\n                                Correct answer is:%d\n", $time/10, out, ans );
70                  else
71                      $display( "%d: Wrong Answer: Your answer is:%d,\n                                Correct answer is:%d\n", $time/10, out, ans );
72              end
73              else begin
74                  #10;
75                  if ( ans == out )
76                      $display( "%d: Correct: Your answer is:%d,\n                                Correct answer is:%d\n", $time/10, out, ans );
77                  else
78                      $display( "%d: Wrong Answer: Your answer is:%d,\n                                Correct answer is:%d\n", $time/10, out, ans );
79              end
80              eof = $fscanf(fp r ans, "%d", ans);
```

```
81          end
82          $fclose( fp_r );
83          $fclose( fp_r_ans );
84          $display( "Simulation End\n" );
85          $stop();
86      end
87
88      TotalALU alu( .clk(clk), .reset(rst), .dataA(inputA),
89                    .dataB(inputB), .Signal(ctrl), .Output(out) );
90
91  endmodule
```

四、討論

(1) Modelsim 驗證結果：

```
# Start
#
#                    1: Input: AND(36)        12        10
#                    2: Correct: Your answer is:         8,
#                                Correct answer is:         8
#
#                    2: Input: OR(37)         12        10
#                    3: Correct: Your answer is:        14,
#                                Correct answer is:        14
#
#                    3: Input: ADD(32)         12        10
#                    4: Correct: Your answer is:        22,
#                                Correct answer is:        22
#
#                    4: Input: SUB(34)         12        10
#                    5: Correct: Your answer is:         2,
#                                Correct answer is:         2
#
#                    5: Input: SLT(42)         12         2
#                    6: Correct: Your answer is:         0,
#                                Correct answer is:         0
#
#                    6: Input: SLL( 0)         12         2
#                    7: Correct: Your answer is:        48,
#                                Correct answer is:        48
#
#                     7: Input: MULTU(25)         10         3
#                    40: Mul End
#
#                     Move Hi
#                    43: Correct: Your answer is:         0,
#                                 Correct answer is:         0
#
#                     Move Lo
#                    44: Correct: Your answer is:        30,
#                                 Correct answer is:        30
#
# Simulation End
#
```

讀入 input.txt 後，根據每個 opcode 算出答案，

第一筆為 AND 運算，$12 \& 10 =$

$0b00000000000000000000000000001100 \& 0b00000000000000000000000000001010 =$

$0b00000000000000000000000000001000 = 8$

第二筆為 OR 運算，$12 \mid 10 =$

$0b00000000000000000000000000001100 \mid 0b00000000000000000000000000001010 =$

$0b00000000000000000000000000001110 = 14$

第三筆為 ADD 運算，$12 + 10 =$

$0b00000000000000000000000000001100 +$

$0b00000000000000000000000000001010 =$

$0b00000000000000000000000000010110 = 22$

第四筆為 SUB 運算，$12 - 10 = 12 + (-10) =$
$0b00000000000000000000000000001100 +$
$0b11111111111111111111111111110110 =$
$0b00000000000000000000000000000010 = 2$

第五筆為 SLT 運算，$12 < 2$，計算 $12 - 2$ 的 $sum =$
$0b00000000000000000000000000001100 +$
$0b11111111111111111111111111111110 =$
$0b00000000000000000000000000001010$

取 sum 的最高位 $= 0$，因此 $12 < 2 = 0$

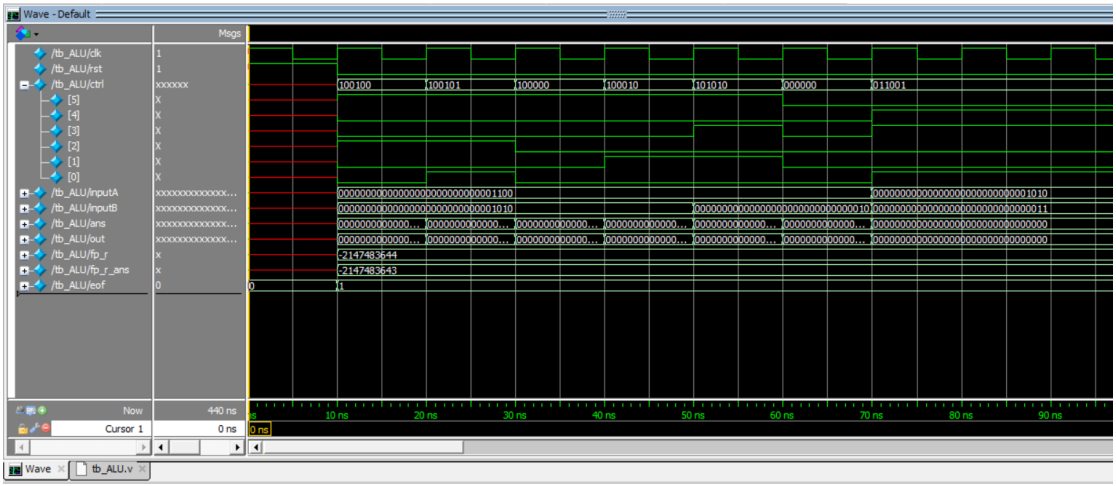
第六筆為 SLL 運算，$12 << 2 =$
$0b00000000000000000000000000001100 <<$
$0b00000000000000000000000000000010 =$
$0b00000000000000000000000000110000 = 48$

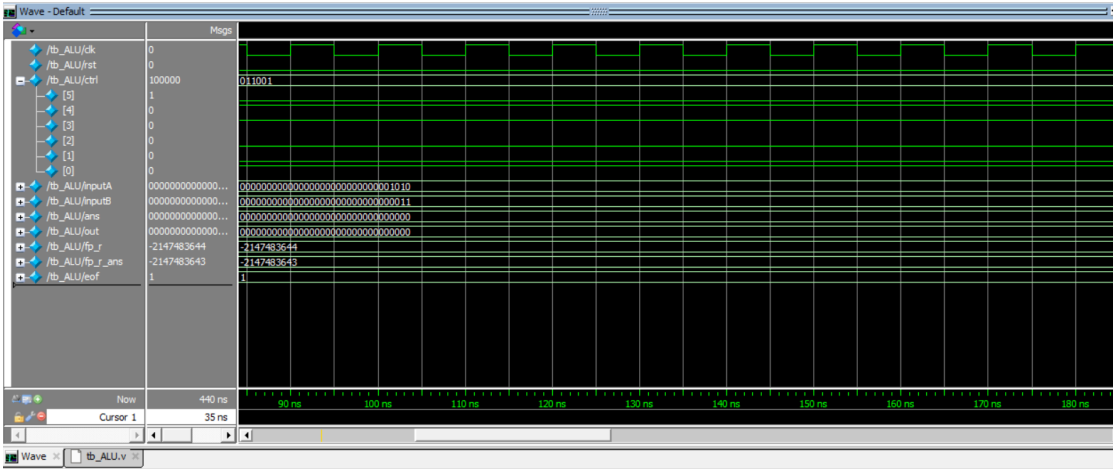
第七筆為 MUL 運算，使用第二代乘法器，$10 * 3 =$
$0b00000000000000000000000000001010 *$
$0b00000000000000000000000000000011$

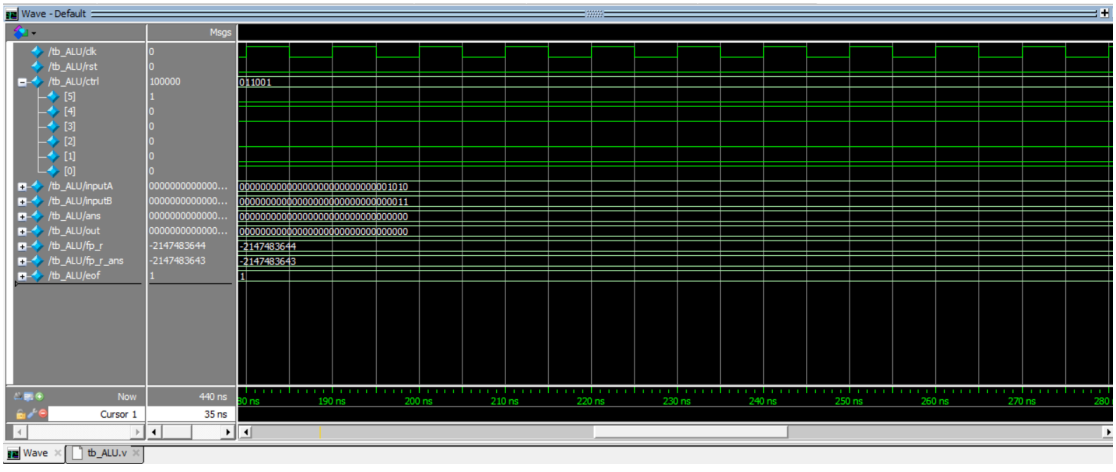
第一輪：確認乘數第 0 位的值，目前為 1，因此將乘數放進乘積的左半邊，並將乘數與 product 右移一位，$product =$
$0b00000000000000000000000000001100000000000000000000000000000000$

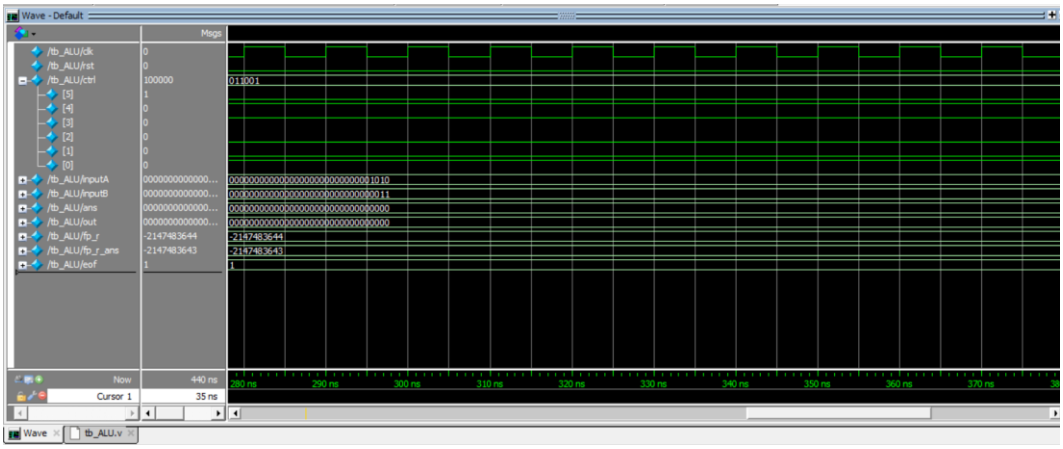
第二輪與第一輪相同，第三輪發現乘數第 0 位是 0，因此乘數與乘積直接右移即可，以此類推直到第 32 輪，最後 $product =$
$0b00000000000000000000000000000000000000000000000000000000011110$，Hi 暫存器為 $0b00000000000000000000000000000000$，Lo 暫存器為 $0b00000000000000000000000000011110$

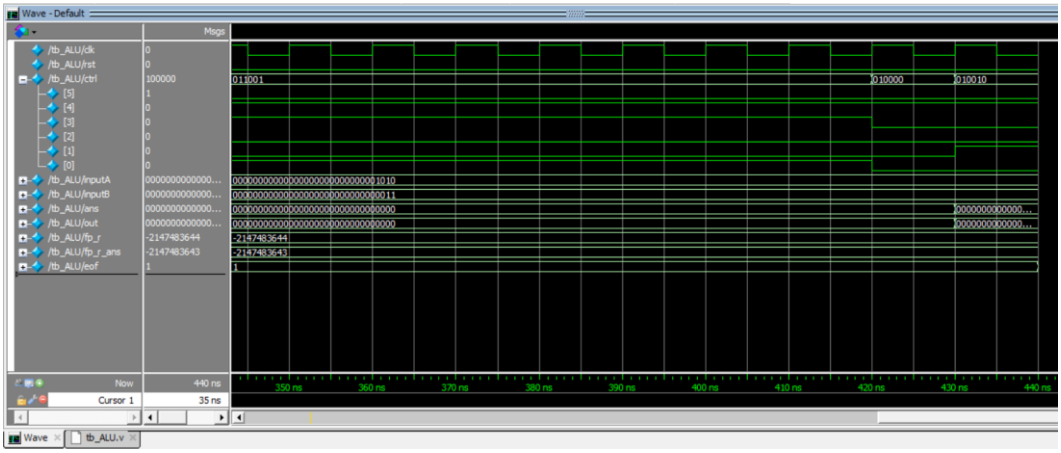
(2) Waveform 輸出圖形：





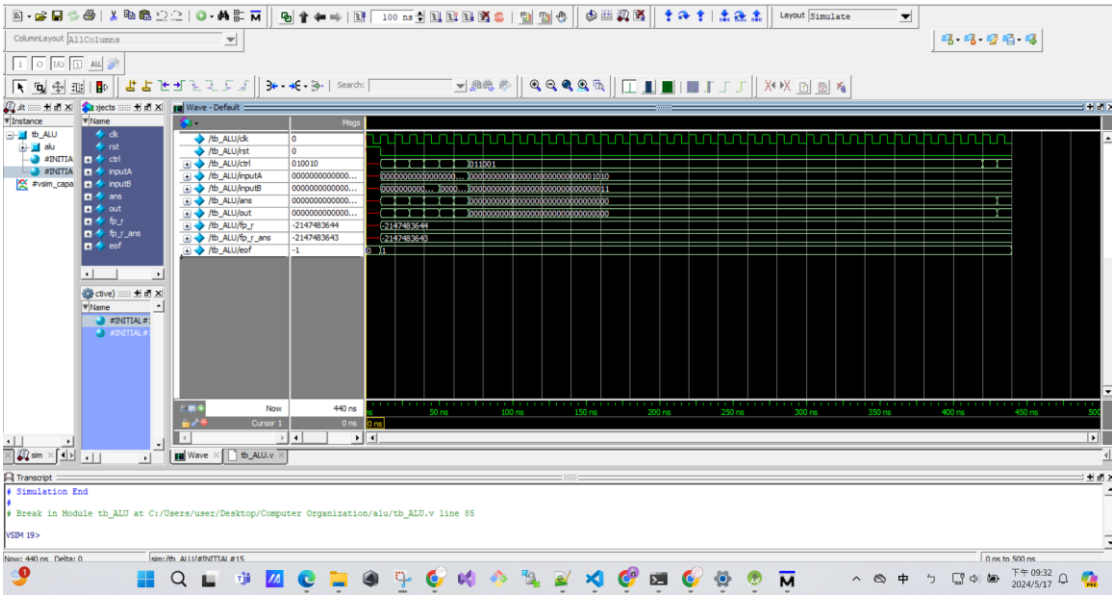



Wave - Default
/tb_ALU/clk
/tb_ALU/rst
/tb_ALU/ctrl
/tb_ALU/inputA
/tb_ALU/inputB
/tb_ALU/ans
/tb_ALU/out
/tb_ALU/fp_r
/tb_ALU/fp_r_ans
/tb_ALU/eof
011001
-2147483644
-2147483643
Now
Cursor 1
35 ns
Wave
tb_ALU.v


Wave - Default
/tb_ALU/clk
/tb_ALU/rst
/tb_ALU/ctrl
/tb_ALU/inputA
/tb_ALU/inputB
/tb_ALU/ans
/tb_ALU/out
/tb_ALU/fp_r
/tb_ALU/fp_r_ans
/tb_ALU/eof
011001
010000
010010
-2147483644
-2147483643
Now
Cursor 1
35 ns
Wave
tb_ALU.v

五、結論

完整波型圖：


Wave - Default
/tb_ALU/clk
/tb_ALU/rst
/tb_ALU/ctrl
/tb_ALU/inputA
/tb_ALU/inputB
/tb_ALU/ans
/tb_ALU/out
/tb_ALU/fp_r
/tb_ALU/fp_r_ans
/tb_ALU/eof
011001
-2147483644
-2147483643
Transcript
Simulation End
Break in Module tb_ALU at C:/Users/user/Desktop/Computer Organization/alu/tb_ALU.v line 85
VSIM 19>

各組員分工方式與負責項目：

(1) 分工方式：

組員討論寫法後，一人寫一支程式，其他人透過加註解及撰寫書面報告的方式理解程式，為求所有人都理解程式內容。

(2) 負責項目：

(a) 程式碼：黃乙家、林雨臻、羅海綺

(b) 書面報告：黃乙家、林雨臻、羅海綺

心得感想：

我們認為本次的計組期中專案太晚公告題目內容，因為儘管期中考週結束了，卻仍然有許多科目的 project 也才剛要開始，同時間我們組員中還有人必須準備多益檢定考試、程式考試，以及資結作業等等。

因為還不是非常熟悉 Verilog 的語法，因此在將程式轉換成電路的過程中，我們理解了一段時間。最後我們是由一個對於 Verilog 最熟悉的組員寫最複雜的 module，其他人在那個 module 完成時，負責加上註解與理解 module 內容，最後再整理到書面報告上，也因為這種分工模式，負責製作報告與加註解的組員們可以在一有疑惑時就立馬發問，直到理解。

此次專案中特別的一點是，除了完成程式需要達成的事情之外還加入了一些限制規範，希望我們能去思考如何善用語法，以另一種 model 撰寫而不影響到電路本身的運作，這使我們可以更加理解那些語法的實際作用。雖然專案題目說的這些需求有可能增加程式的複雜度，但這同時可以幫助我們以後在編寫硬體描述語言，想設計自己的電路的時候熟悉基本的概念。

對於 Debugging，我們認為應該要先讓大家學會如何以波形圖來抓出錯誤，可以的話再介紹 modelSim 更多使用方式，因為在業界上有許多 IC 設計公司還是以 ModelSim 作為 Verilog 模擬平台，這樣才能夠幫助大家儘早熟悉編寫 Verilog。

六、未來展望

(1)熟悉 Verilog 語法。

(2)期望期末專題時對於架構可以更加熟稔。