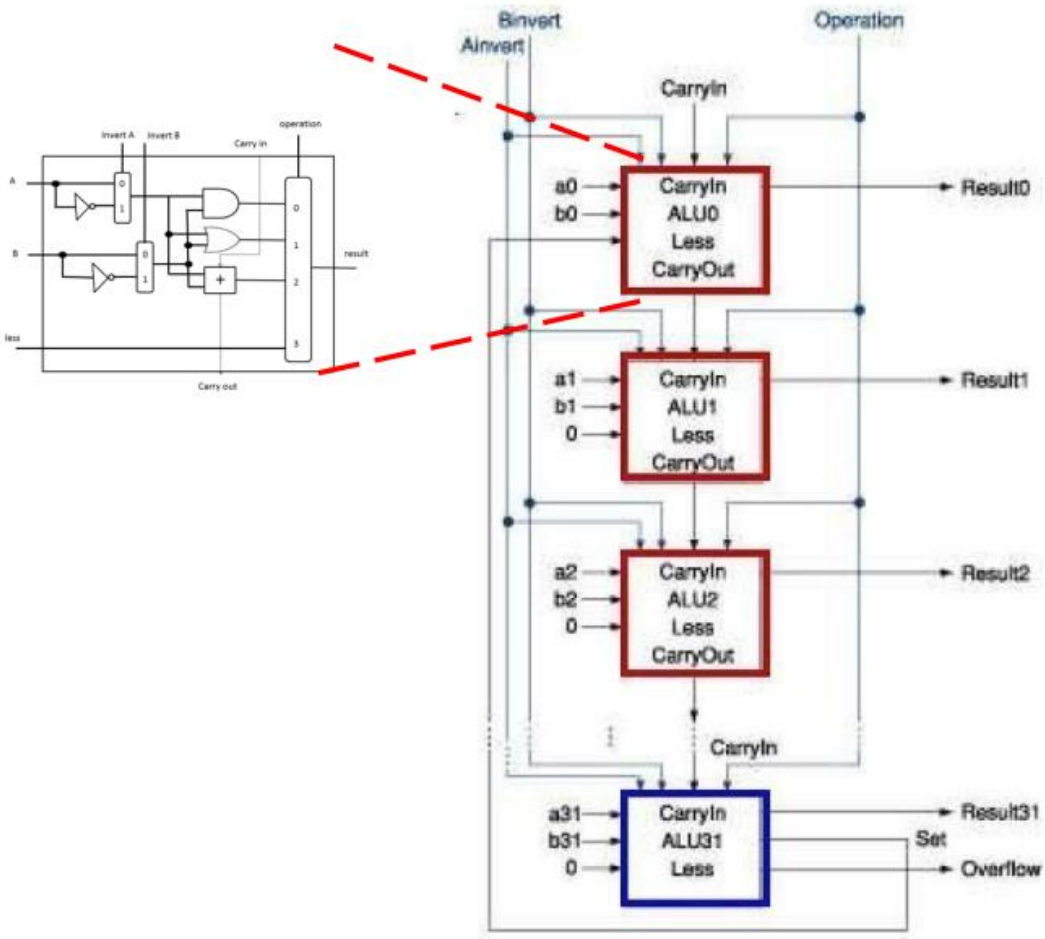
0510008 藍挺毓
0510026 陳司瑋

Computer Organization

Architecture diagrams:

1-bit ALU (Top) **32-bit ALU**



左邊的小方塊是 alu_top.v 的電路設計。+ 我們是用 full adder 實作。
另外我們多了 alu_bottom.v 是做藍色的方塊，另外處理了 overflow 跟 set 的部分。
右邊一整串是 alu.v 的設計方式，將 32 個 1-bit 的 alu 接在一起，做成 32-bit 的 ALU。另外 alu.v 還做了 zcv 的判斷。

Hardware module analysis:

說明請看 code 截圖上的註解

(1)alu_top.v

0510008 藍挺毓
0510026 陳司瑋

直接照 diagram 做 gate level 的設計

```
//same as diagram
xor a_invert(src1_temp,A_invert,src1);
xor b_invert(src2_temp,B_invert,src2);
and ANDgate(and_temp,src1_temp,src2_temp);
or  ORgate(or_temp,src1_temp,src2_temp);

//fulladder
xor AxorB(temp1,src1_temp,src2_temp);
xor xorcin(sum,temp1,cin);//cinxorAxorB
and cout_step1(temp2,temp1,cin);
or  carryout(cout,temp2,and_temp);//carry_out

//choose result according to operation
always@(*)begin
    case(operation)
    2'b00:result = and_temp;
    2'b01:result = or_temp;
    2'b10:result = sum;
    2'b11:result = less;
    endcase
end
```

(2)alu_bottom.v

```
//overflow
wire n_src1_temp, n_src2_temp;
wire n_sum;
wire v0,v1,v2;
not (n_src1_temp,src1_temp);
not (n_src2_temp,src2_temp);
not (n_sum,sum);
and (v0,n_src1_temp,n_src2_temp);//A' & B'
and (v1,v0,sum);//A' & B' & C
and (v2,and_temp,n_sum);//A & B & C'
or  (overflow,v2,v1);//case1 or case2

//set
reg s_temp;
assign set =s_temp;
always@(*)begin
    if(overflow)
        s_temp = (sum)?1'b0:1'b1;
    else s_temp = sum;
end
```

0510008 藍挺毓
0510026 陳司瑋

alu_bottom 比 alu_top 多了這個部分。
overflow 的部分是判斷是否有 overflow，
若{(src1)’ and(src2)’ and(sum 的結果)==1} or {(src1)and(src2)and(sum 的結果)’ ==1}則表示 overflow。因為 src 相同的話，overflow 一定要跟 src 一樣。
*這裡紙的 src 都是最後一位，因為是 bottom 的輸入。

set 是用來判斷 slt，拿最後一個 sum 的結果直接回給 less 的的一個 bit。
特別處理的是 overflow 的部分：
因為 slt 是用減法實作。正值減負值的 slt 結果應該要是 0，但如果 overflow 的話 msb 會是負的，也就是 1，所以要另外判斷是不為 overflow，並將 msb 取 invert。

(3)alu.v

```
assign zzeerroo=1'b0; // set a zero wire

//determine the first cin is 0 or 1
//we have to set it 1 if it is sub or slt operation
assign w_cin_lastbit = (ALU_control==4'b0110 || ALU_control==4'b0111) ? 1: 0 ;
```

//connect 32 1-bit ALU to become a 32-bit，最後一層用 alu_bottom，其他都用 alu_top

```
alu_top m0(.src1(src1[0]), .src2(src2[0]), .less(set), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(w_cin_lastbit), .operation(ALU_control[1:0]), .result(w_result[0]), .cout(carry_temp[0]) );
alu_top m1(.src1(src1[1]), .src2(src2[1]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[0]), .operation(ALU_control[1:0]), .result(w_result[1]), .cout(carry_temp[1]) );
alu_top m2(.src1(src1[2]), .src2(src2[2]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[1]), .operation(ALU_control[1:0]), .result(w_result[2]), .cout(carry_temp[2]) );
alu_top m3(.src1(src1[3]), .src2(src2[3]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[2]), .operation(ALU_control[1:0]), .result(w_result[3]), .cout(carry_temp[3]) );
alu_top m4(.src1(src1[4]), .src2(src2[4]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[3]), .operation(ALU_control[1:0]), .result(w_result[4]), .cout(carry_temp[4]) );
alu_top m5(.src1(src1[5]), .src2(src2[5]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[4]), .operation(ALU_control[1:0]), .result(w_result[5]), .cout(carry_temp[5]) );
alu_top m6(.src1(src1[6]), .src2(src2[6]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[5]), .operation(ALU_control[1:0]), .result(w_result[6]), .cout(carry_temp[6]) );
alu_top m7(.src1(src1[7]), .src2(src2[7]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[6]), .operation(ALU_control[1:0]), .result(w_result[7]), .cout(carry_temp[7]) );
alu_top m8(.src1(src1[8]), .src2(src2[8]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[7]), .operation(ALU_control[1:0]), .result(w_result[8]), .cout(carry_temp[8]) );
alu_top m9(.src1(src1[9]), .src2(src2[9]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[8]), .operation(ALU_control[1:0]), .result(w_result[9]), .cout(carry_temp[9]) );
alu_top m10(.src1(src1[10]), .src2(src2[10]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[9]), .operation(ALU_control[1:0]), .result(w_result[10]), .cout(carry_temp[10]) );
alu_top m11(.src1(src1[11]), .src2(src2[11]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[10]), .operation(ALU_control[1:0]), .result(w_result[11]), .cout(carry_temp[11]) );
alu_top m12(.src1(src1[12]), .src2(src2[12]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[11]), .operation(ALU_control[1:0]), .result(w_result[12]), .cout(carry_temp[12]) );
alu_top m13(.src1(src1[13]), .src2(src2[13]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[12]), .operation(ALU_control[1:0]), .result(w_result[13]), .cout(carry_temp[13]) );
alu_top m14(.src1(src1[14]), .src2(src2[14]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[13]), .operation(ALU_control[1:0]), .result(w_result[14]), .cout(carry_temp[14]) );
alu_top m15(.src1(src1[15]), .src2(src2[15]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[14]), .operation(ALU_control[1:0]), .result(w_result[15]), .cout(carry_temp[15]) );
alu_top m16(.src1(src1[16]), .src2(src2[16]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[15]), .operation(ALU_control[1:0]), .result(w_result[16]), .cout(carry_temp[16]) );
alu_top m17(.src1(src1[17]), .src2(src2[17]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[16]), .operation(ALU_control[1:0]), .result(w_result[17]), .cout(carry_temp[17]) );
alu_top m18(.src1(src1[18]), .src2(src2[18]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[17]), .operation(ALU_control[1:0]), .result(w_result[18]), .cout(carry_temp[18]) );
alu_top m19(.src1(src1[19]), .src2(src2[19]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[18]), .operation(ALU_control[1:0]), .result(w_result[19]), .cout(carry_temp[19]) );
alu_top m20(.src1(src1[20]), .src2(src2[20]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[19]), .operation(ALU_control[1:0]), .result(w_result[20]), .cout(carry_temp[20]) );
alu_top m21(.src1(src1[21]), .src2(src2[21]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[20]), .operation(ALU_control[1:0]), .result(w_result[21]), .cout(carry_temp[21]) );
alu_top m22(.src1(src1[22]), .src2(src2[22]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[21]), .operation(ALU_control[1:0]), .result(w_result[22]), .cout(carry_temp[22]) );
alu_top m23(.src1(src1[23]), .src2(src2[23]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[22]), .operation(ALU_control[1:0]), .result(w_result[23]), .cout(carry_temp[23]) );
alu_top m24(.src1(src1[24]), .src2(src2[24]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[23]), .operation(ALU_control[1:0]), .result(w_result[24]), .cout(carry_temp[24]) );
alu_top m25(.src1(src1[25]), .src2(src2[25]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[24]), .operation(ALU_control[1:0]), .result(w_result[25]), .cout(carry_temp[25]) );
alu_top m26(.src1(src1[26]), .src2(src2[26]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[25]), .operation(ALU_control[1:0]), .result(w_result[26]), .cout(carry_temp[26]) );
alu_top m27(.src1(src1[27]), .src2(src2[27]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[26]), .operation(ALU_control[1:0]), .result(w_result[27]), .cout(carry_temp[27]) );
alu_top m28(.src1(src1[28]), .src2(src2[28]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[27]), .operation(ALU_control[1:0]), .result(w_result[28]), .cout(carry_temp[28]) );
alu_top m29(.src1(src1[29]), .src2(src2[29]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[28]), .operation(ALU_control[1:0]), .result(w_result[29]), .cout(carry_temp[29]) );
alu_top m30(.src1(src1[30]), .src2(src2[30]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[29]), .operation(ALU_control[1:0]), .result(w_result[30]), .cout(carry_temp[30]) );

alu_bottom m31(.src1(src1[31]), .src2(src2[31]), .less(zzeerroo), .A_invert(ALU_control[3]), .B_invert(ALU_control[2]), .cin(carry_temp[30]), .operation(ALU_control[1:0]), .result(w_result[31]), .cout(w_cout), .overflow(w_overflow), .set(
```

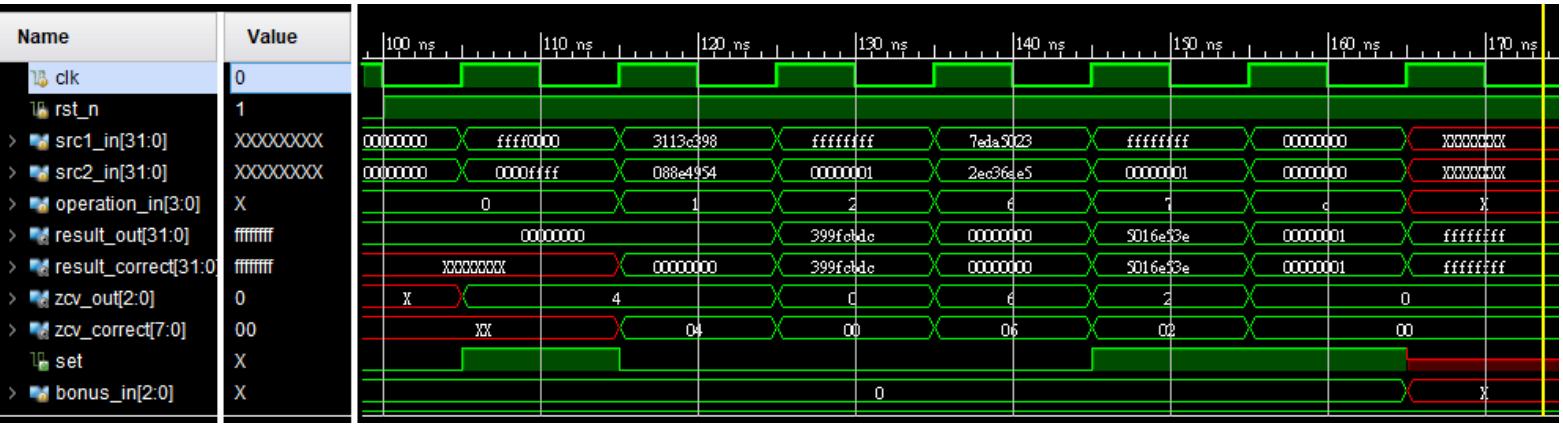
0510008 藍挺毓
0510026 陳司瑋

//set zcv

```
result = w_result;

//set cout if there is caryout in add or sub
if(ALU_control== 4'b0010||ALU_control==4'b0110) cout = w_cout;
else cout=1'b0;

//set zero=0 if the result is zero
if(result==32'b0) zero<=1;
else zero<=0;

overflow=w_overflow;
```

Experiment result:

```
# run 1000ns
*****************************************************
 Congratulation! All data are correct!
*****************************************************
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testbench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:14 . Memory (MB): peak = 796.375 ; gain = 18.102
```



波形圖示 out 跟 correct 的截圖

Problems you met and solutions:

1. 一開始我們寫的並不是 gate level，而其中有用到&和|，我們思考我們就是要寫 alu，不確定可不可以用&和|的邏輯位元，所以最後就改成 gate 寫 alu_top。也發現用 gate 寫 alu_top 只要照著 diagram 就幾乎完成了。
2. 自己的硬體設備上有些問題，所以用系計中的電腦寫，為了寫作業申請學生

0510008 藍挺毓
0510026 陳司瑋

證刷門禁卡的時候發現電資因為隸屬電機學院，所以不能使用系計中。會再詢問是否可以申請系計中門禁，或是把硬體設備處理好。

Summary:

因為修過 Dlab，所以這次的 lab 不難，不過因為電腦的關係用了很久。
建議的話，希望測資可以多一點，這樣比較好抓 bug。謝謝助教！