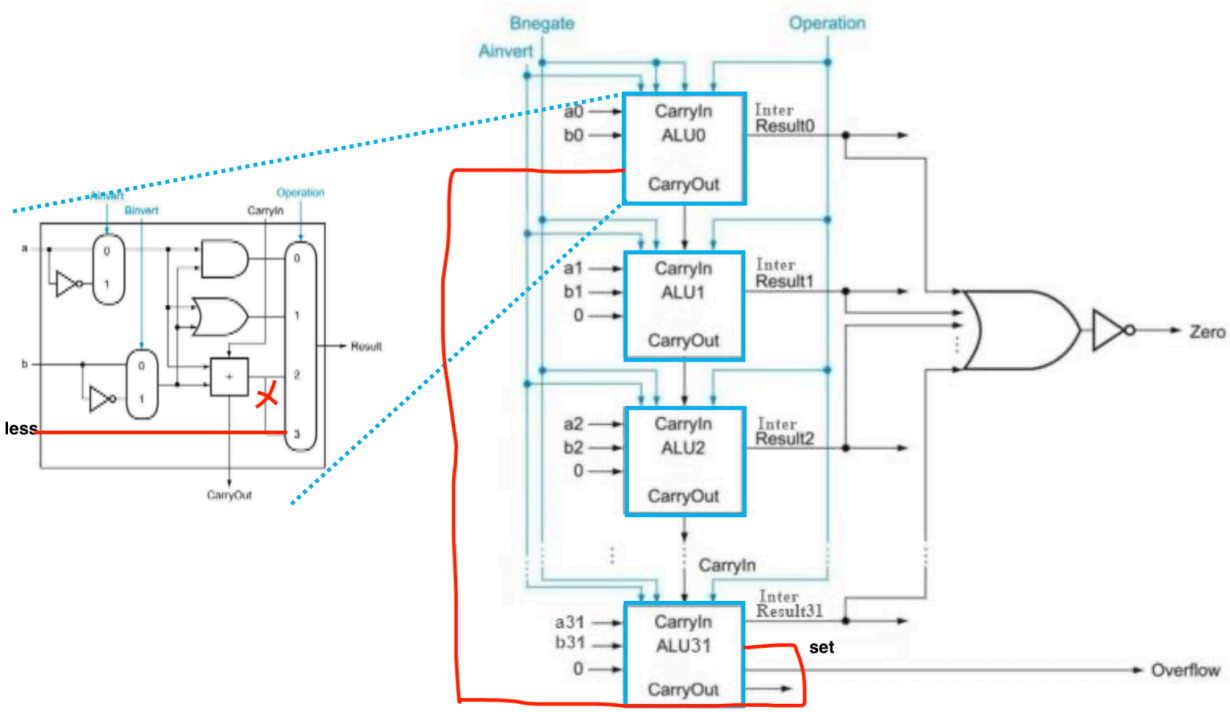
Computer Organization

Architecture diagram:



Detailed description of the implementation:

1.ALU_1bit:

```
/* Write your code HERE */
xor g1(temp1, src1, Ainvert);
xor g2(temp2, src2, Binvert);

always@(*)begin
   case(operation)
      2'b00:
         result = (temp1 & temp2);
      2'b01:
         result = (temp1 | temp2);
      2'b10:begin
         result = temp1^temp2^Cin;
         cout = ((temp1^temp2)&Cin)|(temp1&temp2);
      end
      2'b11:begin
         result = less;
         cout = ((temp1^temp2)&Cin)|(temp1&temp2);
      end
   endcase
end

endmodule
```

用xor, 先將ainvert, binvert做轉換

依據operation:

00做and, 01做or, 10做加法, 11就把less值給result

2.alu.v:

```
wire ainvert;
wire binvert;
wire [1:0] oper;
wire [31:0] R;
wire [31:0] c;
wire set;

assign set = (src1[31])^(~src2[31])^(c[30]);

assign ainvert = ALU_control[3];
assign binvert = ALU_control[2];
assign oper = ALU_control[1:0];

ALU_1bit alu0(
    .src1(src1[0]),
    .src2(src2[0]),
    .Ainvert(ainvert),
    .Binvert(binvert),
    .Cin(binvert),
    .operation(oper),
    .less(set),
    .result(R[0]),
    .cout(c[0])
);
```

set是用來判斷a是否小於b(利用加法邏輯式判斷)

把ALU_control拆成ainvert, binvert, 和operation

```
genvar i;

generate
   for(i=1; i<32; i=i+1)begin
      ALU_1bit
      alu(
         .src1(src1[i]),
         .src2(src2[i]),
         .Ainvert(ainvert),
         .Binvert(binvert),
         .Cin(c[i-1]),
         .operation(oper),
         .less(1'b0),
         .result(R[i]),
         .cout(c[i])
      );
   end
endgenerate


always@(*)begin
   if(rst_n == 1'b1)begin
      result= R;
      zero = ~(|result);
      cout = c[31];
   end
end

endmodule
```

分別把src1, src2, ainvert, binvert, operation輸入之前寫好的alu_1bit, 由於compiler的問題, 先將所有輸出存在其他地方,等下再處理

alu0和alu1~alu31的輸入不太一樣, ex:less, cin , 所以另外寫

下面的always部分, 會判斷rst_n是不是1, 如果是的話才執行

最後將暫存的值接回output

Implementation results:

```
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
*****************************************************
*        PATTERN RESULT TABLE           *
*****************************************************
*PATTERN *          Result          *ZCV *
*****************************************************
*    Congratulation! All data are correct!     *
*****************************************************
Correct Count: 9
Stopped at time : 205 ns : File "C:/Users/Bob Hsiao/Downloads/Computer Organization Lab2/testbench.v" Line 146
ISim>
```

Problems encountered and solutions:

1.第一個問題是不知道是否要宣告32個ALU_1bits, 因為嘗試過並不能直接用for迴圈,後來上網查才知道有generate用法

2.第二個問題是關於less, 一直在想如何能透過always的方法改變alu0的less, 後來決定不管always, 只要cout[30]一改動, alu0的less就會變, 這樣就解決問題了, code也比較簡單

Lesson learnt (if any):

verilog(之前幾乎不會)

Comment: