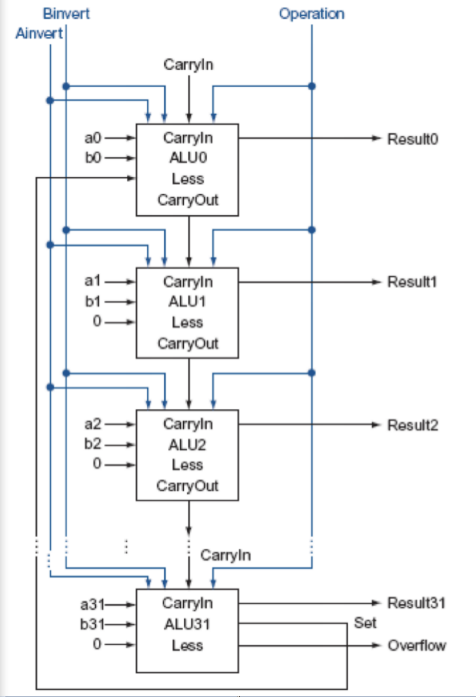
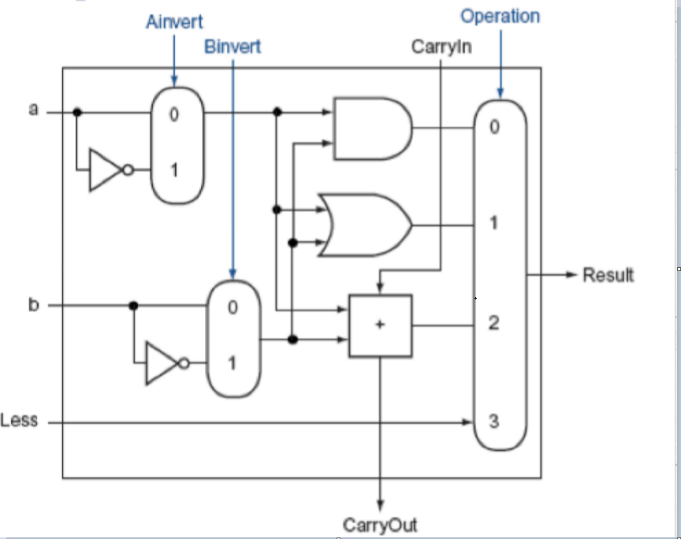
**Computer Organization**

**Architecture diagrams:**

**1-bit ALU: 32-bits ALU:**

****

****

**Hardware module analysis:**

Module of alu\_top:

* The module has 9 inputs, src1, src2, less, A\_invert, B\_invert, cin, operation, result,cout. Except operation, all of the input are 1 bit. Operation is 2 bit.
* The alu\_top module is used to deal with one of 32 bit’s operation.
* A\_invert and B\_invert is used to decide whether to choose the invert of the input.
* Cin is carry in used for adder.
* Operation is a 2 bit register, on different uses which will be assign different value

1. 00 for process of and (2) 01 for process of or (3) 10 for process of addition (4) 11 for process of less

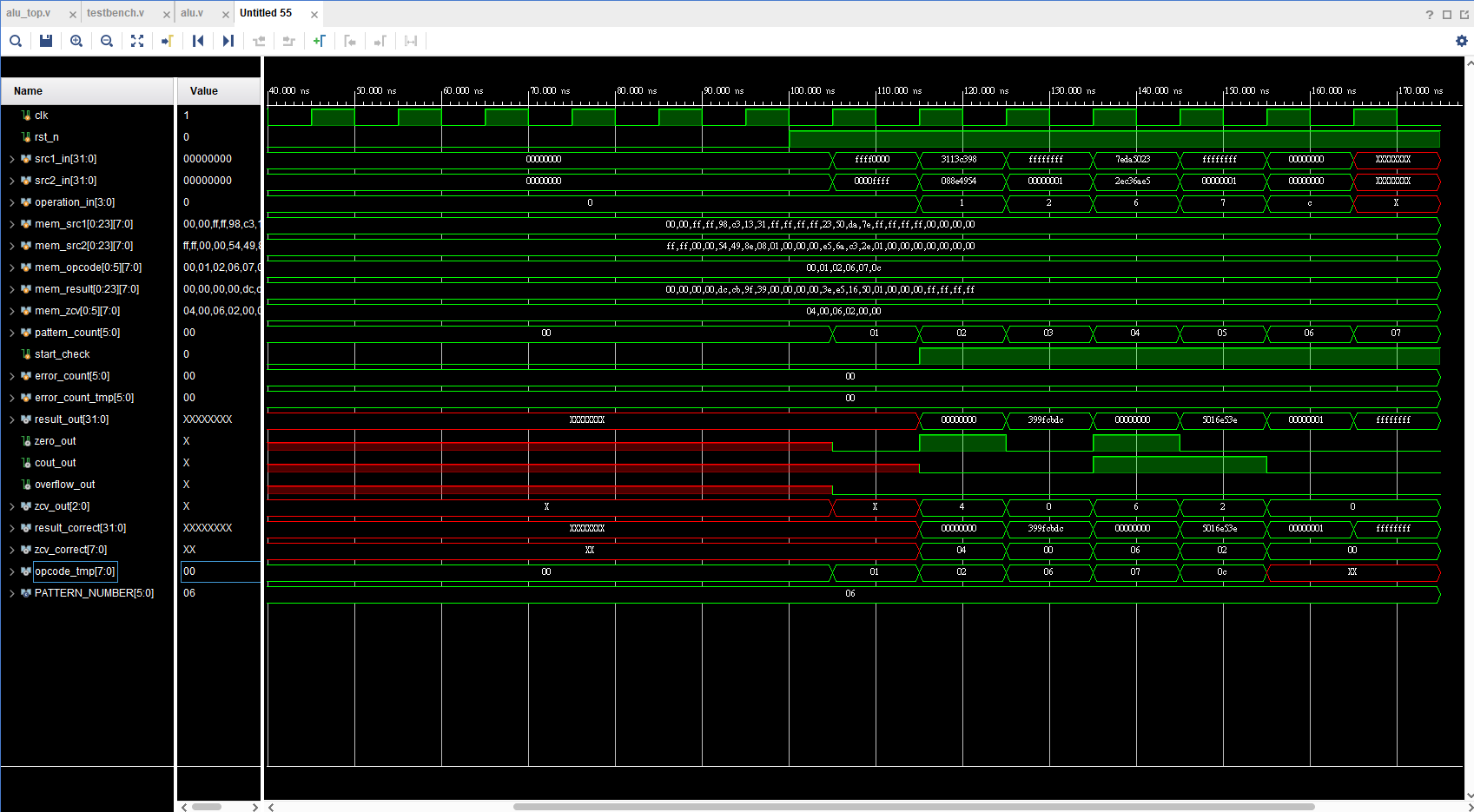
* The process of addition is implement by a full adder

Module of alu:

* The module has 9 inputs, clk, rst\_n, src1, src2, ALU\_control, result, zero, cout, overflow and their definition is same on the hw description
* The module is completed by a series of alu\_top module. This is complete by the generate block and how it connect is just like the architecture diagrams.
* The process of and/or is just and/or every bit of the inputs
* The process of addition is connected just like a ripple carry adder
* The process of subtraction is implemented by taking 2’s complement and add them
* The process of nor can transfer to and by de’morgen’s law taking invert of both of the inputs and and every bit of them
* The process of slt is implement by subtraction set every bit to 0 except LSB. LSB is determined by the signed bit of the result after the subtraction
* the output of zero is just check the result after the process
* the output of cout is the carry\_out if the last bit from its alu\_top module
* the output of overflow just check on 4 different occasions (1) positive + positive (2) negative + negative (3) positive – negative (4) negative – positive.

This is check after subtraction or addition.

**Experiment result:**

****

**Problems you met and solutions:**

1. the alu will take wrong alu\_control value when positive clock edge come, because the alu\_control changes the posedge clk coming.

<solution>

Add alu\_control after the always block. By doing so, when alu\_control

changes the always block will be active also, then it can catch all the correct

input information

1. the output of the cout will go to undetermined

<solution>

Alu\_top should process every bit of the cout not just after addition. If it doesn’t produce cout then we should set cout to 1’b0.

**Summary:**

The key of this lab, I think is design 1-bit alu and connect them to get 32-bit alu

The operation of 32-bit alu is just design like professor taught during classes.