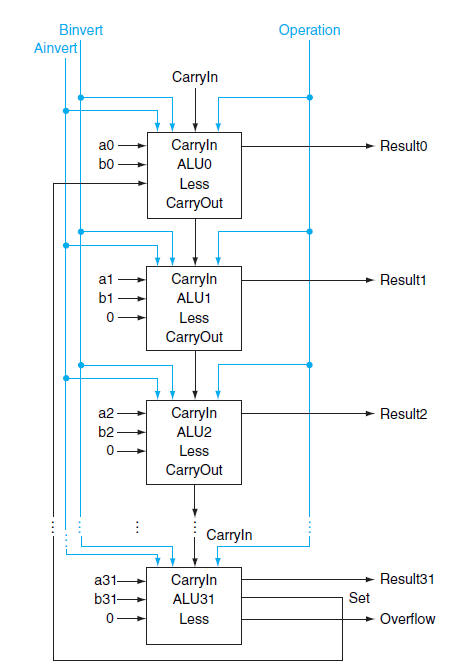
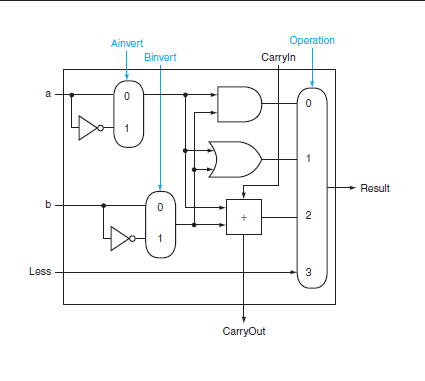
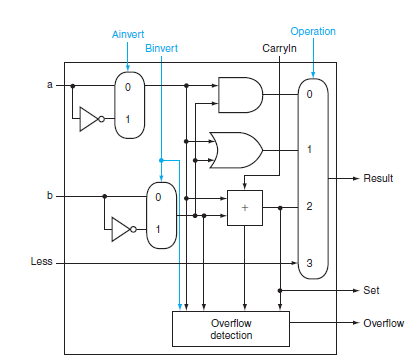
**Computer Organization**

**Architecture diagrams:**

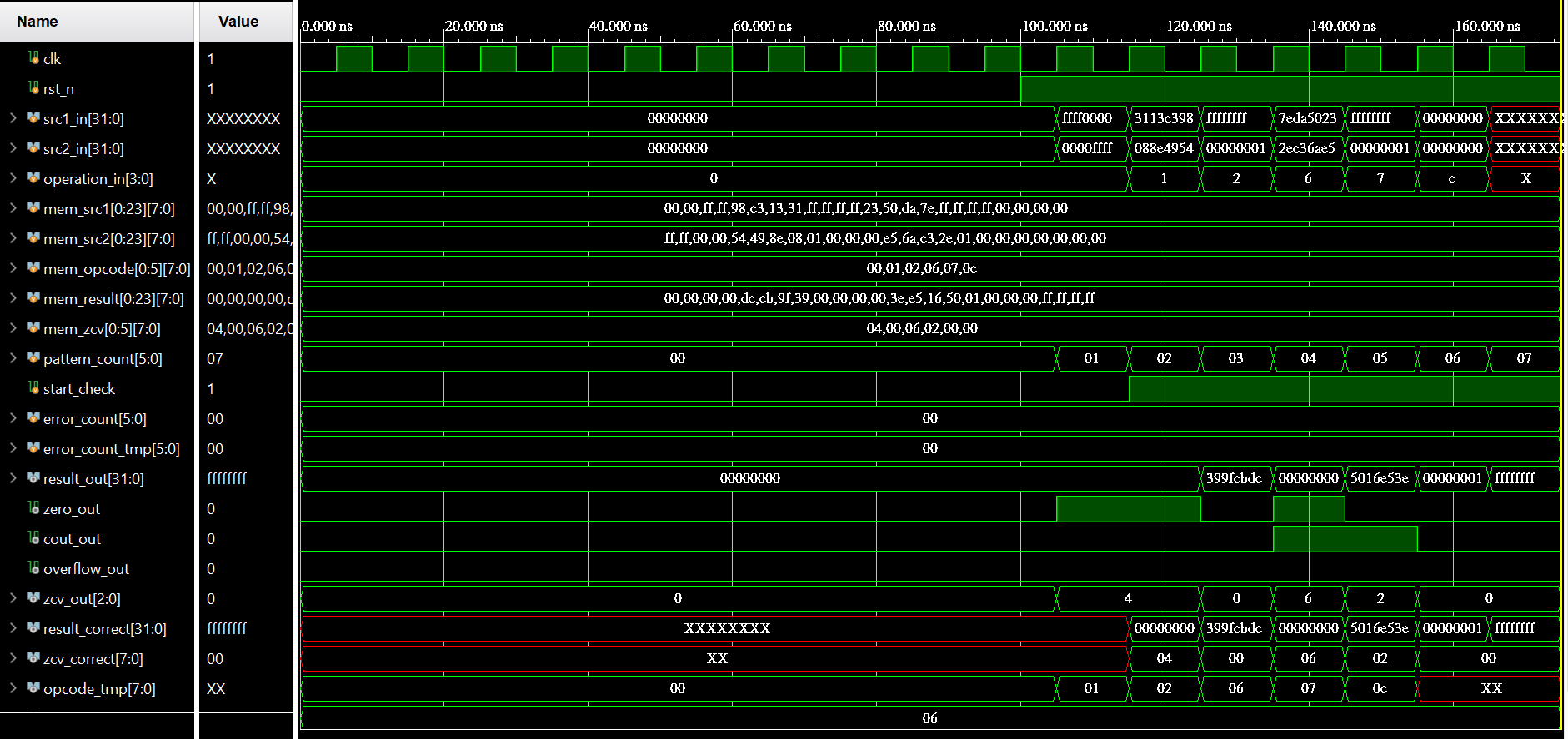
The architecture diagram is based on the provided diagram for the CO LAB1:

1. 32-bit ALU: 
2. 1-bit ALU-top：
3. 1-bit ALU-bottom：

**Hardware module analysis:**

the implementation is using two modules:

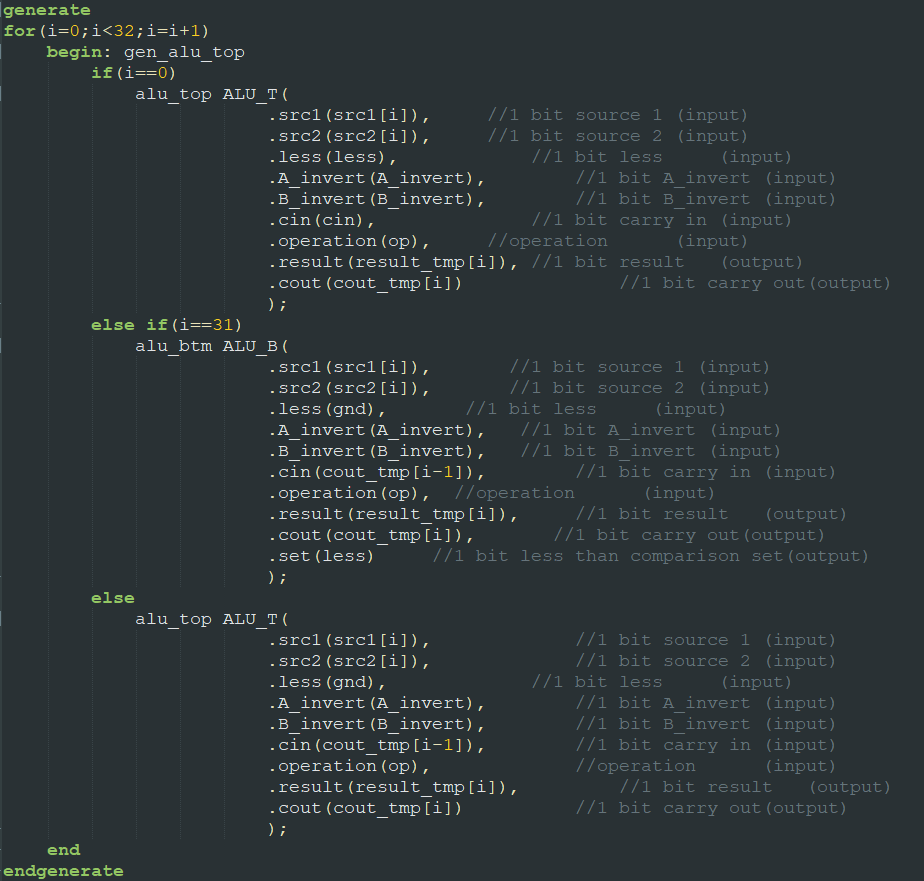
* 1-bit ALU\_top, which give the result based on the operation and sources given: which using a multiplexor to output the result.
* 1-bit ALU\_bottom, which is slightly modified from ALU\_top, by adding the ***set*** output for **SLT operation.**
* 4 basic operation of the 1-bit ALU:
  + AND
  + OR
  + addition
  + less
* 6 operations of the overall ALU:
  + AND
  + OR
  + addition: is done by the full adder mechanism
  + 
  + subtraction: make src2’s 2-complement
    - 
  + set less than: subtracted src2 from src1, if the result of the most significant bit is 1 (which is a negative sign), then output 0000 0001; else output 0000 0000.
  + NOR: using DeMorgan’s Theorem ( AND operation between inverted src1 and src2)
    - 

**Experiment result:**

**Problems you met and solutions:**

1. Using 1-bit ALU, the code will be very trivial as the module is used repeatly( for 32 times).

**solution:**

* using *generate-for loop:*
* 

2. The addition and subtraction’s operations have a very strange output that the half of the result is correct.

case:

src1 = FFFF FFFF

src2 = 0000 0001

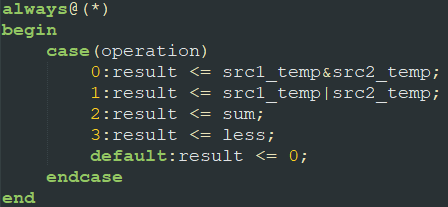
op: addition

result = FFFF 0000 ( correct:0000 0000)

cout = 0

overflow = 0

**solution:**

* this problem occurred is due to the incorrectness of the statement in the always block sensitivity list.
* ****
* before: only consider the input data(*src1 & src2)* and operation(*op*) in sensitivity list but not the *sum*, but there is time needed for data to transfer from the less significant bit ALU to the most significant bit ALU, thus the sum may vary after the input
  + 
* after: consider every varying data in module
  + 

**Summary:**

This 32-bit ALU is implemented using 32 1-bit ALU and some conditional statement to perform AND, OR ,addition, subtraction, set less than and nor operation.