**Lab 2. ALU**

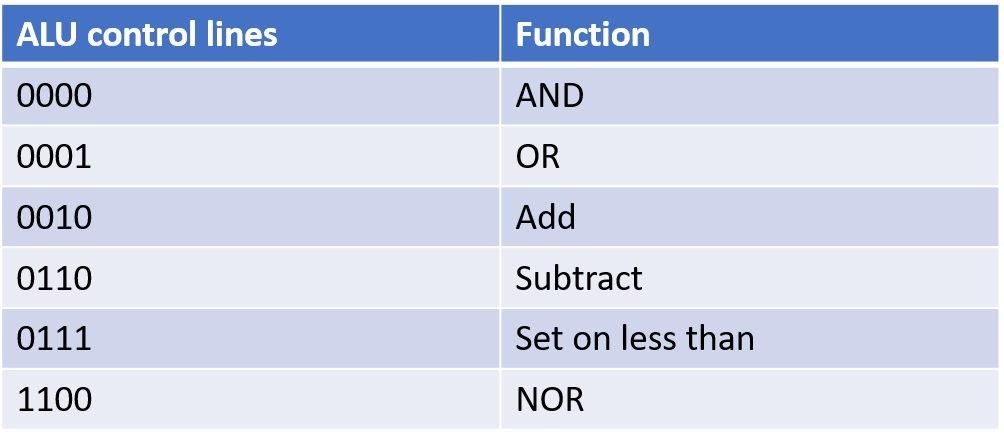
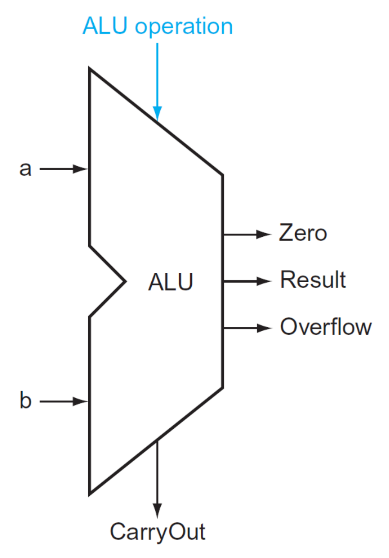
1. **Objectives:**
2. Learn how to design arithmetic logic unit (ALU).
3. **Resources:**
4. Download Logisim 2.7.1 from canvas
5. Download “Lab2 - ALU.circ” from Canvas. Inputs, outputs, and most of the necessary electronic components have been given in each circuit. **Do not make any changes on the given Inputs and outputs in the circuits, and Do not change the appearances of electronic components.**
6. **Requirements**
7. Write down your answers to the questions in the following “4. Tasks” section in your lab report. Do not forget to show your demo to TA if required.
8. This lab is separated into three parts and have three deadlines. Please submit the report for each part on time.

**<1> Part I (15 points): tasks 4.1.1 – 4.1.2, the deadline is 3/17.**

**<2> Part II (50 points): tasks 4.1.3 – 4.1.8, the deadline is 3/24.**

**<3> Part III (35 points): tasks 4.2.1 – 4.2.3, the deadline is 3/31.**

1. **Tasks**:
2. Design a 32-bit ALU that can realize the following functions:



(2) Use different methods to design an 8-bit Multiplication hardware.

**4.1 Design a 32-bit ALU**

To finish this task, please design the following circuit one by one.

**4.1.1 Design a 1-bit full adder (5 points)**

Explain your design method:

Design is directly from lecture slides

Circuit:

Diagram

Description automatically generated

**4.1.2 Design an 8-bit Ripple Carry Adder and Subtracter (10 points)**

**Requirement:** Only 1-bit full adders designed in 4.1.1, basic logic gates, and multiplexers (if needed) are allowed.

Explain your design method:

XOR each adder’s b input with sub. Send each adder’s carryout to the subsequent adder’s carryin. XOR the first adder’s carryin with a constant 0. XOR the last two carryouts together to get the overflow.

Circuit:

A picture containing text, clock

Description automatically generated

**4.1.3 Design a 4-bit Carry Lookahead Unit (10 points)**

Truth table:

Table

Description automatically generated with medium confidenceTable

Description automatically generatedTable

Description automatically generated with medium confidenceA picture containing nature

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedBackground pattern, table

Description automatically generatedTable

Description automatically generated

Generated circuit by Logisim:

Diagram, schematic

Description automatically generated

**4.1.4 Design a 4-bit Carry Lookahead Adder (10 points)**

**Requirement:** only1-bit full adders designed in 4.1.1, 4-bit Carry Lookahead Unit designed in 4.1.3, and basic logic gates are allowed.

Explain your design method:

Get generate and propagate bits from ANDing and ORing a and b. Have C0 be the carry-in for the first full adder and the others come from the Lookahead Unit. The carry-out of the MSB goes into C3.

Circuit:

Diagram

Description automatically generated

**4.1.5 Design a 16-bit Carry Lookahead Adder (10 points)**

**Requirement:** only4-bit Carry Lookahead Adders designed in 4.1.4, 4-bit Carry Lookahead Unit designed in 4.1.3, and basic logic gates are allowed.

Explain your design method:

Simpler version of the 4-bit Carry Lookahead Adder

Circuit:

Diagram

Description automatically generated

**4.1.6 Design a 32-bit Carry Lookahead Adder (5 points)**

**Requirement:** only16-bit Carry Lookahead Adders designed in 4.1.5 and basic logic gates are allowed.

Explain your design method:

Put two 16 bit Carry Lookahead Adders next to one another and send the carry-outs to C31 and C32

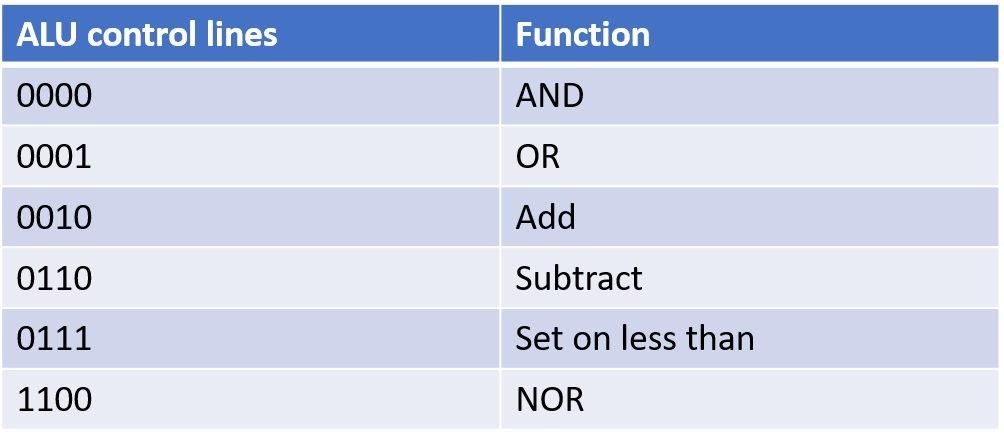
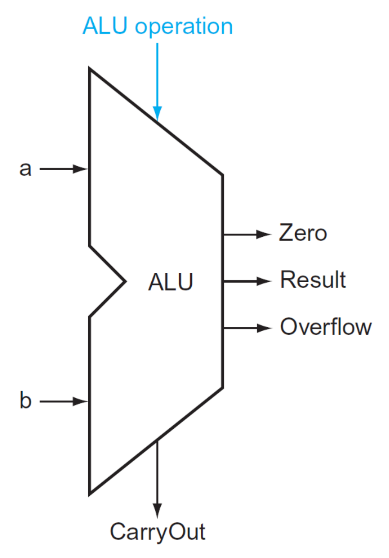
Circuit:

Diagram, schematic

Description automatically generated

**4.1.7 Design a 32-bit ALU (10 points)**

**Requirement:** only32-bit Carry Lookahead Adder designed in 4.1.6, multiplexers, extenders, splitters, and basic logic gates are allowed.



Explain your design method:

Compare a and b to find “Set on less than” input on 4x1 MUX. MUX each a and b with their respective inverses with the Ainvert and Binvert as the signal bit. The outputs go into both the 32-bit Adder and an AND and OR gate to the remaining 4x1 MUX inputs. The Overflow is found by XORing the two carry-outs from the Adder. The signal bits of the 4x1 MUX is driven from Operation and the output goes to both Result and a NOR gate to find Zero.

Circuit:

**Diagram

Description automatically generated**

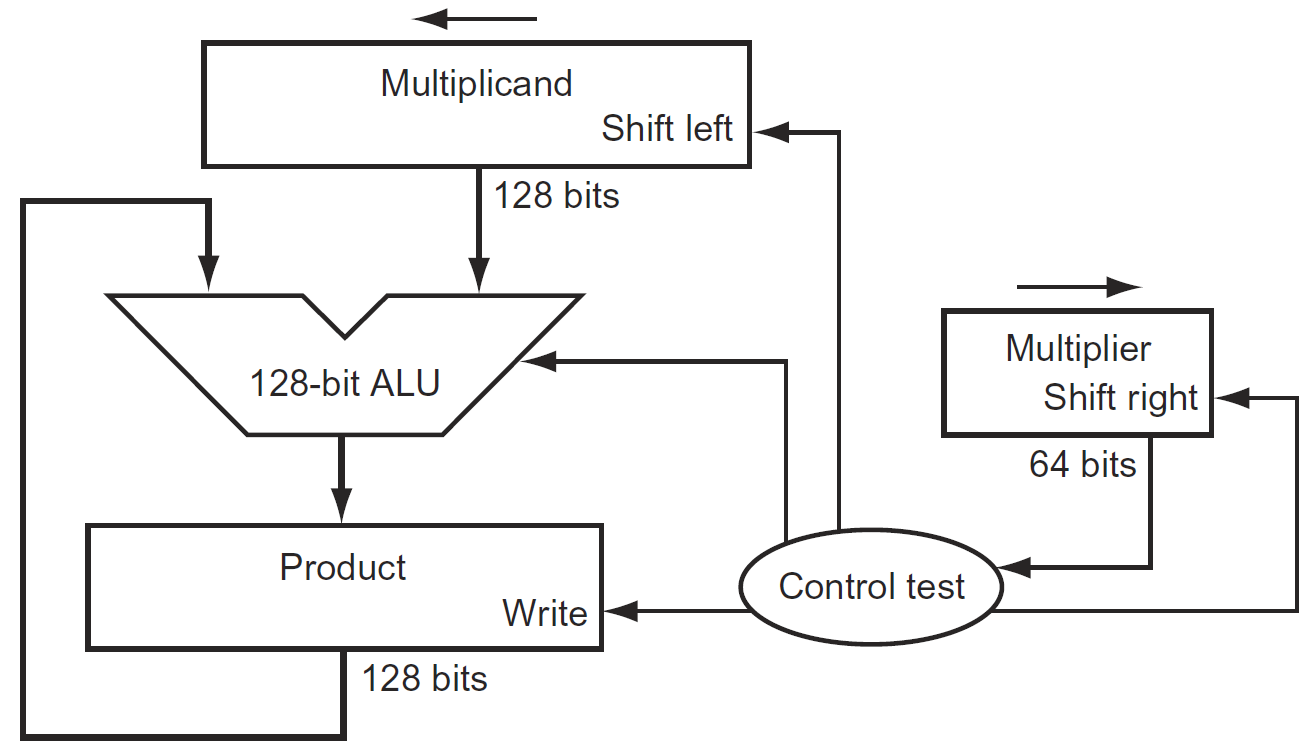
**4.1.8 Demo your design to TA (5 points)**

It works

* 1. **8-bit multiplication unit design**

Please design circuits to implement the following two different methods for multiplication, respectively.

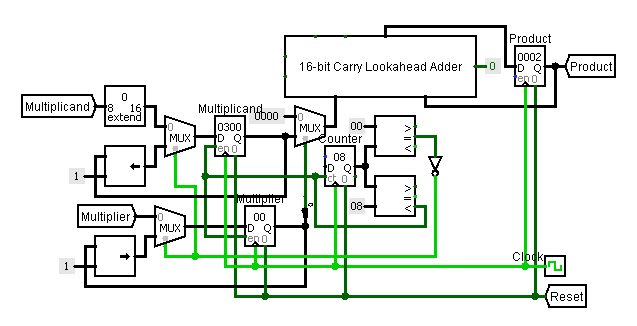
**4.2.1 Design 8-bit Multiplication unit (Version 1) (15 points)**

****

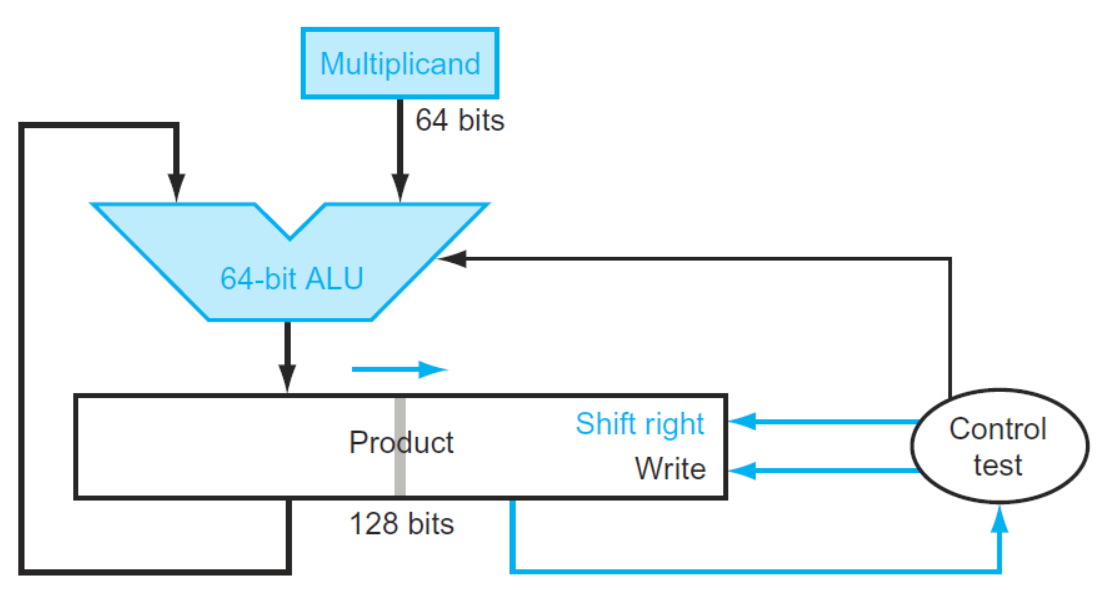
Explain your design method:

Load each number into a multiplexer with its shifter value. Have the clock select the two multiplexers. Load the output of each into a register which is enabled by the counter register in the counter circuit. All clock and reset inputs to the registers are from the clock and reset. The multiplicand and a constant zero are selected in another multiplexer by the Multiplier. The output goes into to adder along with the output of the product register. The product register is fed from the output of the adder.

Circuit:



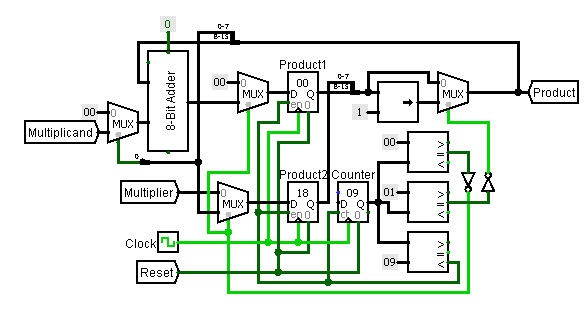
**4.2.2 Design 8-bit Multiplication unit (refined version) (15 points)**



Explain your design method:

Each number is loaded into a multiplexer. One is selected by the clock, the other by the least significant bit of the product. The output of the multiplicand’s multiplexer is an input to the adder along with half of the product. That output goes into another multiplexer selected from the counter circuit when the count is not zero. The output goes into the first product register, both registers are enabled by the counter circuit. The output of the multiplier’s multiplexer goes into the second product register. Both register outputs combine to go into both the right shift and the subsequent multiplexer, selected when the count does not equal one. The output is the product.

Circuit:



**4.2.3 Demo your design to TA (5 points)**

It also works