**Istanbul Technical University**



**Computer and Informatics Faculty**

**Computer Engineering Department**

**BLG222E COMPUTER ORGANIZATION**

**PROJECT 1: 16-BIT ALU DESIGN REPORT**

**OF GROUP 3**

Group Members:

150140012 Uğur Uysal

150140031 Pelin Hakverir

150140103 Hüseyin Aküzüm

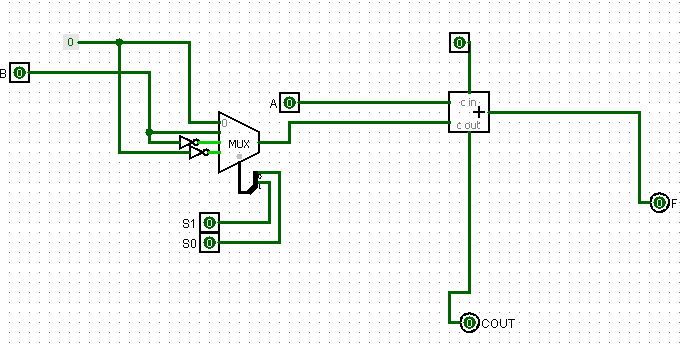
150140124 Şahin Olut

1. **Introduction**

In this project we designed a 16-Bit Arithmetic Logic Unit that performs arithmetical, logical and shifting operations according to a 4-bit selector, two 16-bit inputs and an 8-bit condition code register. We designed this unit by combining 1-bit arithmetic, logic and shift units with Logisim.

### Parts of The ALU

1. **1-Bit Arithmetic Unit**

****

Capabilities:

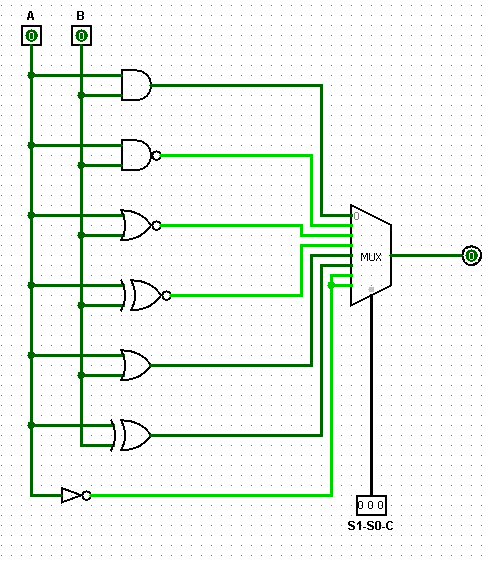
* Transfer A
* Increment A
* Addition
* Add with carry
* Subtract with borrow
* Subtraction
* Decrement A

This unit executes the required operations with a multiplexer and a full adder.

Selectors of the multiplexer are S1 and S0 selection bits. The operations are made according to S1, S0, C (carry flag) and A, B inputs’ bits. S3 and S2 bits are stable (S3=1, S2=0) during this operation.

The multiplexer output (S1-S0 truth table) and operation (unit output) summary is:

1. If the MUX output is 0 **Transfer A**, if C is 1 **Increment A**
2. If the MUX output is B **Addition**, if C is 1 **Addition with carry**
3. If the MUX output is B’ **Subtract with borrow**, if C is 1 **Subtraction**
4. If the MUX output is 1 **Decrement A**, if C is 1 **Transfer A**
5. **1-Bit Logic Unit**

****

Capabilities:

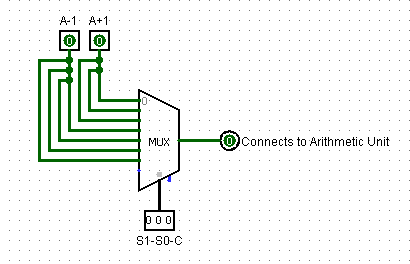
* AND
* NAND
* OR
* XOR
* NOR
* XNOR
* Complement A

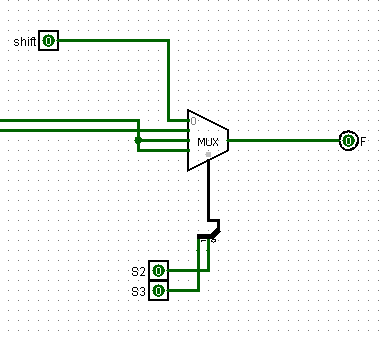
This unit performs the logical operations according to S1, S0 and C which are the selecter bits of the MUX.

Selectors of the multiplexer are S1, S0 selection bits and C(carry flag) bit. The operations are made between A and B according to S1, S0, C (carry flag). S3 and S2 bits are stable (S3=0, S2=1) during this operation.

The multiplexer output (S1-S0-C truth table) and operation (unit output) summary is:

1. If (S1=0, S0=0) “A **AND** B”, if C=1 “A **NAND** B”
2. If (S1=1, S0=0) “A **OR** B”, if C=1 “A **XOR** B”
3. If (S1=0, S0=1) “A **NOR** B”, if C=1 “A **XNOR** B”
4. If (S1=1, S0=1) Complement A, C is don’t care
5. **1-Bit Shifting Unit**

****

****

Capabilities:

* Logical shift right A into F
* Arithmetic shift right A into F
* Circular shift right A into F
* Logical shift left A into F
* Arithmetic shift left A into F
* Circular shift left A into F

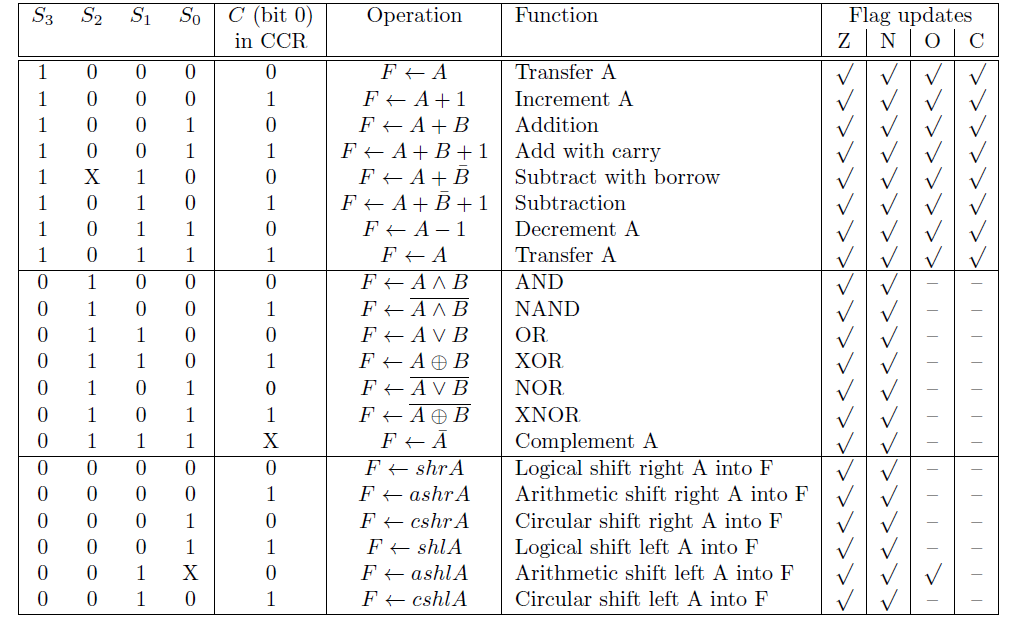
While we are shifting, we use the previous or next bit depending on the type and the direction of the required shifting. But the most significant bit is not shifted to least significant bit always; if the operation is logical shift left, it becomes 0 and if it is arithmetical shifting, it protects its sign and finally, if it is circular shift left, then most significant bit is shifted to least significant bit.

Selectors of the multiplexer are S1, S0 selection bits and C(carry flag) bit. The operations are made on A according to S1, S0, C (carry flag). S3 and S2 bits are stable (S3=0, S2=0) during this operation.

The multiplexer output (S1-S0 truth table) and operation (unit output) summary is:

1. If (S1=0, S0=0) **Logical shift right A into F**, if C=1 **Arithmetic shift right A into F**
2. If (S1=0, S0=1) **Circular shift right A into F**, if C=1 **Logical shift left A into F**
3. If (S1=1, S0=X(don’t care)) and C=0, **Arithmetic shift left A into F**
4. If (S1=1, S0=0) and C=1, **Circular shift left A into F**
5. **Conclusion**

The ALU that we designed has the same attributes with this truth table:



Our ALU’s final view is: