**PROJECT REPORT**

**TITLE: 2-BIT COMPUTER**

**INDEX**

**INTRODUCTION**

**COMPONENTS REQUIRED**

**AND GATE**

**OR GATE**

**NOT GATE**

**DCD HEX DISPLAY**

**TRUTH TABLES**

**2-bit Adder**

**2-bit Subtractor**

**2-bit Multiplier**

**2-bit Comparator**

**2-bit Divider**

**SOFTWARE IMPLEMENTATION**

**CONCLUSION**

**REFERENCES**

**INTRODUCTION**

A **computer** is an electronic device that manipulates information, or data. It has the ability to **store**, **retrieve**, and **process** data. It is a device that can be [instructed](https://en.wikipedia.org/wiki/Computer_programming) to carry out an arbitrary set of [arithmetic](https://en.wikipedia.org/wiki/Arithmetic) or [logical](https://en.wikipedia.org/wiki/Boolean_algebra) operations automatically. The ability of computers to follow a sequence of operations, called a [*program*](https://en.wikipedia.org/wiki/Computer_program), make computers very applicable to a wide range of tasks.

Since ancient times, simple manual devices like the [**abacus**](https://en.wikipedia.org/wiki/Abacus), aided people in doing calculations. Early in the [Industrial Revolution](https://en.wikipedia.org/wiki/Industrial_Revolution), some mechanical devices were built to automate long tedious tasks, such as **guiding patterns for**[**looms**](https://en.wikipedia.org/wiki/Loom). More sophisticated electrical machines did specialized [analog](https://en.wikipedia.org/wiki/Analogue_electronics" \o "Analogue electronics) calculations in the early 20th century. The first [digital](https://en.wikipedia.org/wiki/Digital_data) electronic calculating machines were developed during [World War II](https://en.wikipedia.org/wiki/World_War_II). The speed, power, and versatility of computers have increased continuously and dramatically since then.

Conventionally, a modern computer consists of at least one [processing element](https://en.wikipedia.org/wiki/Processing_element), typically a [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit) (CPU), and some form of [memory](https://en.wikipedia.org/wiki/Memory_(computers)). The processing element carries out arithmetic and logical operations, and a sequencing and control unit can change the order of operations in response to stored [information](https://en.wikipedia.org/wiki/Data).

**CENTRAL PROCESSING UNIT (CPU)**

The control unit, ALU, and registers are collectively known as a [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit) (CPU). Early CPUs were composed of many separate components but since the mid-1970s CPUs have typically been constructed on a single [integrated circuit](https://en.wikipedia.org/wiki/Integrated_circuit) called a [*microprocessor*](https://en.wikipedia.org/wiki/Microprocessor).

**ARITHMETIC LOGIC UNIT (ALU)**

The ALU is capable of performing two classes of operations: arithmetic and logic. The set of arithmetic operations that a particular ALU supports may be limited to addition and subtraction, or might include multiplication, division, [trigonometry](https://en.wikipedia.org/wiki/Trigonometry) functions such as sine, cosine, etc., and [square roots](https://en.wikipedia.org/wiki/Square_root). Some can only operate on whole numbers ([integers](https://en.wikipedia.org/wiki/Integer)) whilst others use [floating point](https://en.wikipedia.org/wiki/Floating_point) to represent [real numbers](https://en.wikipedia.org/wiki/Real_number), albeit with limited precision. However, any computer that is capable of performing just the simplest operations can be programmed to break down the more complex operations into simple steps that it can perform. Therefore, any computer can be programmed to perform any arithmetic operation—although it will take more time to do so if its ALU does not directly support the operation. An ALU may also compare numbers and return [Boolean truth values](https://en.wikipedia.org/wiki/Truth_value) (true or false) depending on whether one is equal to, greater than or less than the other ("is 64 greater than 65?"). Logic operations involve [Boolean logic](https://en.wikipedia.org/wiki/Boolean_logic): [AND](https://en.wikipedia.org/wiki/Logical_conjunction), [OR](https://en.wikipedia.org/wiki/Logical_disjunction), [XOR](https://en.wikipedia.org/wiki/Exclusive_or), and [NOT](https://en.wikipedia.org/wiki/Negation). These can be useful for creating complicated [conditional statements](https://en.wikipedia.org/wiki/Conditional_(programming)) and processing [Boolean logic](https://en.wikipedia.org/wiki/Boolean_logic).

*In our mini-project, we have attempted to design and realise a 2-bit computer whose ALU performs specific calculations and operations, namely:*

1. *Addition*
2. *Subtraction*
3. *Multiplication*
4. *Division*
5. *Comparison*

In a nutshell, firstly, we thought of a particular logic for each operation and then by using our knowledge of the Karnaugh Maps, we realised all the combinational circuits) using the basic gates (AND, OR, NOT).

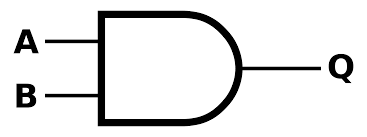
**COMPONENTS REQUIRED**

1. IC 7408

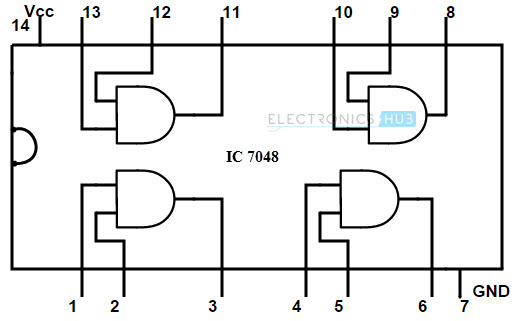
IC 7408 is a Quad 2-input AND Gate IC.

The output of an AND gate is true only when both the inputs are true. Else, the output is false in any other case.

The logical symbol for the AND gate is:

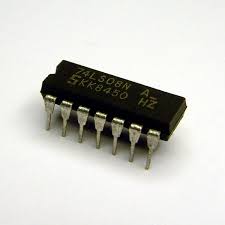


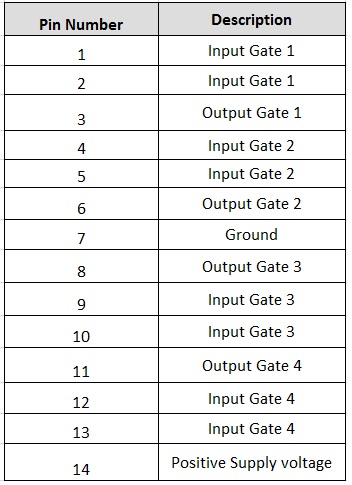
The internal diagram of IC 7408 is shown below:

[](http://www.electronicshub.org/wp-content/uploads/2015/06/7048.jpg)

7408 is a TTL series AND gate. It has 4 AND gates in it. Each pin if IC 7408 and its purpose are explained below.

**IC 7408 TRUTH TABLE OF AND**

**Pin description**  
[](http://www.electronicshub.org/wp-content/uploads/2015/06/pin-descriptioj.jpg)

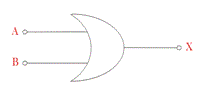
Pin 14 is supplied with the maximum input of 5.2 volts D.C. If the supply voltage increases 5.2 Volts, then the IC may damage due to high supply.

1. IC 7432

IC 7432 is a Quad 2-Input OR gate IC,

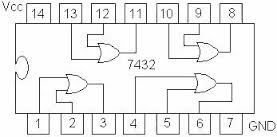
**OR gate** performs the LOGICAL OR which means outputs is logical 1 if at least one of the inputs is 1. Only when all the inputs are in low state or logical 0, the output is low or 0 and in all other input conditions, the output will be high or logical 1.

The logical symbol of the OR gate is:



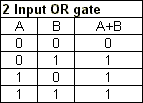
In this IC there are fourteen pins.

The internal gate diagram of IC 7432 is as shown below:



The 7th pin is to be grounded while the 14th pin has to be provided with a positive power supply.

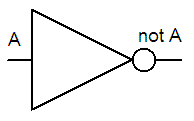
**IC 7432 TRUTH TABLE FOR OR GATE**

****  

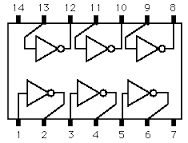
1. IC 7404

The 7404 contains six independent invertors or NOT gates. An invertor takes an input signal and “reverses” it. A low input signal becomes high and vice versa.

The Logical Symbol for NOT gate is:

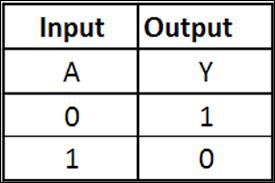


The internal diagram of IC7404 is as follows:



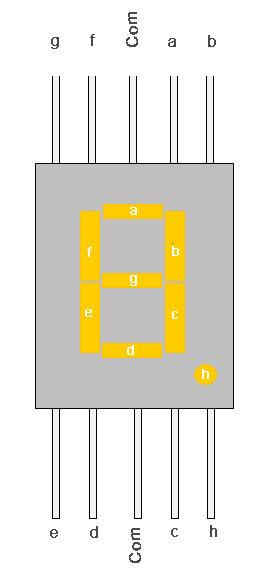
The 7th pin is to be grounded while the 14th pin has to be provided with a positive power supply.

**IC 7404 TRUTH TABLE FOR NOT**

1. **DCD Hex Display**

This device is a 7-segment digital hex display.



Using a seven-segment display, shows the hexadecimal digit corresponding to the four-bit input. If any of the inputs are not 0/1 (either floating or error), then the display shows a dash ('-'). A separate one-bit input controls the display of the decimal point.

**Truth Tables for the desired operations:**

1. ADDITION (2-bit adder)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **C2** | **C1** | **C0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |

1 1 1 1 1 1 0

Here, A and B are the first 2-bit number, and C and D are the second 2-bit input. AB and CD are added to obtain the output in the form of C2,C1,C0.

Using K-maps, the expressions for C2, C1 and C0 are:

C0 = BCD + ABD + AC

C1 = ~AB~CD + ABCD + ~A~BC + ~AC~D + ~A~B~C + A~C~D

C2 = ~BD + B~D

The above expressions are realised in the form of a combinational circuit.

1. **SUBTRACTION (2-bit subtractor)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **S** | **C1** | **C0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Here, the 2-bit inputs are AB and CD. S denotes the sign. (-ve or +ve) and C1, C0 are the 2-bit output of the subtraction.

Using K-maps,

C1 = ~A~BD + A~C~D + ~ACD + AB~C

C0 = B~D + ~BD

1. **MULTIPLICATION (2-bit multiplier)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A1** | **A0** | **B1** | **B0** | **C3** | **C2** | **C1** | **C0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

So, similarly,

Obtaining the expressions using K-maps:

C3 = A1A0B1B0

C2 = A1~A0B1 + A1B1~B0

C1 = ~A1A0B1 + A0B1~B0 + A1~A0B0 + A1~B1BO

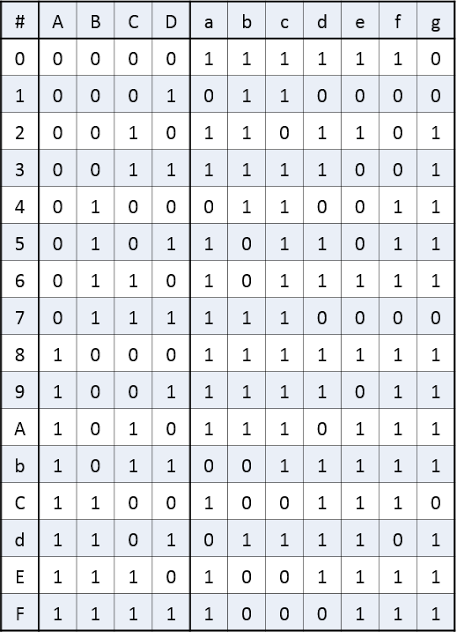
C0 = A0B0

1. **COMPARISION (2-bit comparator)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **C3** | **C2** | **C1** | **C0** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Here, “A” will be displayed on the DCD whenever AB > CD. And, “B” when CD>AB. Else, “E” will be displayed whenever both the inputs are equal.

TRUTH TABLE FOR THE DCD



Here, ABCD is the 4-bit representation of the numbers from 0 to 15.

Whereas, abcdefg are the LED outputs.

1. DIVISION (2-bit divider)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | Q3 | Q2 | Q1 | Q0 | R1 | R0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

Note: 0/0 is an indeterminate form. So, “E” will be displayed on the DCD. “E” stands for ERROR.

Here, AB and CD are the 2-bit inputs.

Q3Q2Q1Q0  is the four-bit quotient and R1R0 is the remainder.

Using K-maps, the expressions are:

R2 = A~BCD

R1 = ~ABC + BC~D

Q3 = ~C~D

Q2 = ~C~D

Q1 = ~C~D + A~C

Q0 = B~CD + AC~D + ABC

**SOFTWARE IMPLEMENTATION**

We implemented our logics on the Multisim 14.0.0.

Firstly, each operation was performed successfully on different sheets, using the truth tables mentioned above.

The inputs to the gates were given from the Word Generator while the outputs were obtained on the DCD hex display.

Then came the difficult task of combining all the operations and creating the final ALU having all the operations at one place.

So, we put each operation into a specific box, assuming it to be a black box where only the inputs and outputs are shown by following the procedure as given:

* Go to “place” situated on the toolbar in the Multisim desktop.
* Select “New Hierarchical Block” from the dropdown menu. A dialog box appears.
* Input your block “name”, number of “input pins”, and number of “output pins” into the desired labels. Click “Ok”.
* A hierarchical block would be created. Double click on the block and select “open sub sheet”.
* Copy the sheet whose circuit is required to be put into the black box, give the respective inputs and outputs to the terminals and you are done.

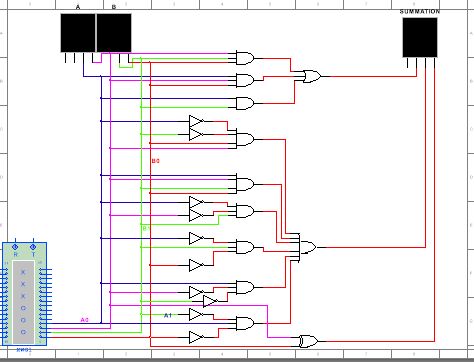
Place > New Hierarchical Block > Open Sub Sheet

Now, each black box (containing a particular operation) is given same inputs from the word generator ( 2 – bit numbers: A,B as inputs) and separate outputs connected to the DCD hex display.

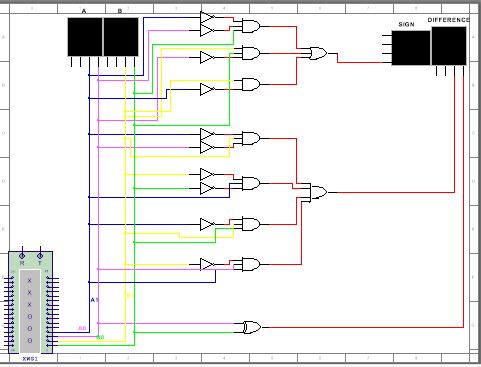
Hence, the desired ALU of our 2-bit Computer was obtained.

Following are the combinational circuits of our five operations implemented on the Multisim. (Version 14)

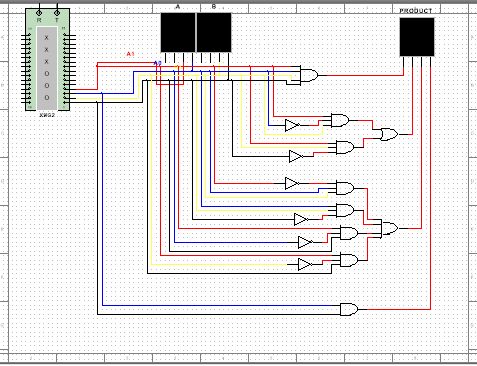
1. **ADDITION**



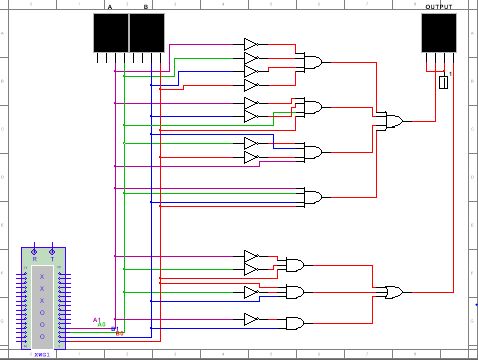
1. **SUBTRACTION**



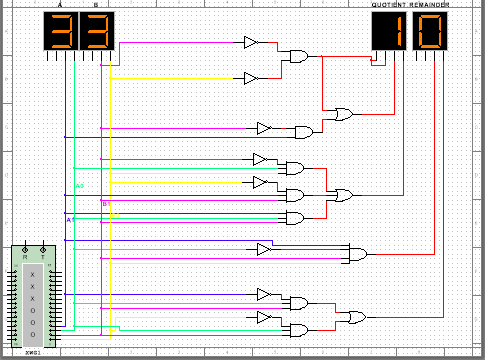
1. **MULTIPLICATION**



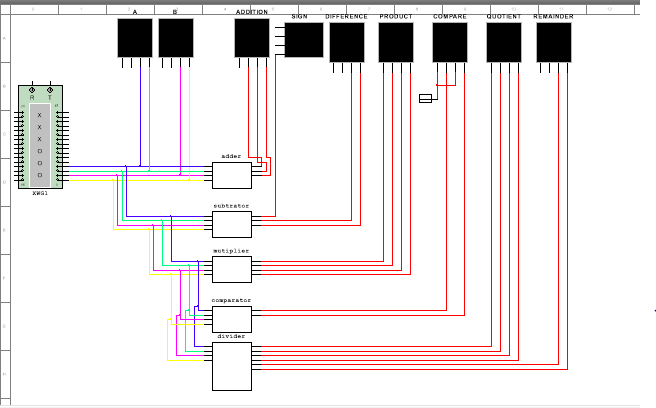
1. **COMPARISION**



1. **DIVISION**



**The FINAL Design: Containing all the five operations.**



**CONCLUSION**

Our ALU has five operations. These five operations are carried out for 2-bit inputs.

This mini-project helped us to learn many new aspects and internal concepts of combinational circuits.

We were able to gain more information about the various features of *Multisim.*

To conclude, we were successful in designing a logic that could help us in performing calculations, similar to what any other computational machine would do!

Future Plans: We are planning to implement the same approach to 4-bit inputs, both signed and unsigned, involving floating point values as well.

**REFERENCES**

1. Digital Design

Author: M. Morris Mano & Michael D. Ciletti

1. [www.cburch.com](http://www.cburch.com)
2. [www.ni.com](http://www.ni.com)
3. [www.electronics-tutorials.ws](http://www.electronics-tutorials.ws)