HIGH SCHOOL EDITION
Journal of Student Research
Volume 10 Issue 4 (2021)

Analysis of an 8-bit Arithmetic Logic Unit using Logisim Emulator

Alvin Tuo[1] and Yong Zhu[2#]

[1]Wyoming Seminary
[2#]Advisor, Wilkes University

ABSTRACT

Computers have become an increasingly integral part of our daily lives. This has been a culmination of nearly a century's work of mathematicians and engineers pushing the boundaries of what machines can do. Despite the ubiquity and prominence of computers, not many people understand the fundamental components. This is especially important now, as we are on the brink of a fundamental shift in the architecture of computers: from x86-based CPUs to low power, high efficiency RISC/ARM based processors. The technical study conducted below is an attempt to understand CPU functionality as well as search for areas of possible efficiency. This paper will dive in-depth on the fundamentals of computer components through use of early breadboard-type circuitry and the use of LS IC chips, as well as Logisim® simulator to analyze these components, in particular the CPU (Central Processing Unit) and the ALU (Arithmetic Logic Unit).

Background

A computer, in essence, is "an electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program." (Oxford Lang. Dictionary) Derived from the Latin word *putare*, or "to think", the first computers were basic adding machines constructed to add small numbers. French mathematician Blaise Pascal invented one of the first calculators in the 1640s to hasten laborious tax documentation work. These were basic calculators that relied on punch holes to store data and retrieve values. Although basic, they are the foundation of the computers we are more familiar with today.

The first mechanical computer was created by British mathematician Charles Babbage in the 1830s. As computing needs grew larger, codebreaking pioneer Alan Turing created the first modern electronic programmable computer, the ENIAC, to assist in the Allied war effort in breaking the German enigma code. Post-World War II came the advent of the transistor, arguably the single-most important invention in human history. Replacing vacuum tubes, the transistor was much more reliable and compact. Computers have advanced by leaps and bounds since 1945 - the phones in our pockets are more powerful than whole-room mainframes and the Apollo Guidance Computer which assisted the moon landing. Gordon Moore, founder of Intel, stipulated that every two years, the number of transistors in an IC will double.

As of 2021, Moore's Law has continually been applicable since the 1970s. As microprocessors shrink beneath the nanometer however, it will become harder and harder to pack more transistors into the CPU. This, in part, is driving research and development to increase efficiency amongst processors. The newest innovation in computers include system-on-chips (SoCs) packages and Reduced Instruction Set Computers (RISC). RISC is a move away from Intel-based x86 CISC-based processes to ARM-centered instruction sets. While the transition requires tremendous software changes, the benefits include less power consumption, higher performance, and simplified design. These recent advancements will further increase efficiency in CPUs across all devices, which is why it is so important to continue research in processor architecture.

Methods

In an analogue system, a consistent stream of data is present. While in some situations, this might be beneficial (i.e., representation of a sound wave on a graph or as voltage values are always approximations, resulting in a less-than-ideal listening situation), digital systems have taken over with the widespread proliferation of the computer. In digital systems, data is in two states: on or off, represented by 1s and 0s. These on/off signals are the only way for humans to communicate to a computer processor the functions we want it to perform. Logic gates, the fundamental building blocks of processors, can provide several functions required to form a unit capable of computing. These gates, such as AND/OR/XOR/etc. can be combined to form a system capable of arithmetic. In this paper, several of these gates will be linked to form an 8-bit ALU with basic AND/OR logic functions as well as representations of several situational instances. Originally, this computer was built on an actual physical breadboard, but physical constraints forced the ALU construction project to move onto Logisim, a simulator for logic gates. The following 8-bit ALU is formed from two basic parts: a full adder and a 1-bit ALU.

Arithmetic Logic Unit

The Arithmetic Logic Unit, or ALU, is a key component in the central processing unit. Found in every computer, the ALU is responsible for carrying out basic arithmetic and logical operations. These operations include:

- Addition and Subtraction
- Multiplication and Division*
- OR/AND gate functionality

*Multiplication and division functionality is not built-in on this 8-bit ALU model, as it requires the use of register shifts.

Data is submitted from the A register for inputs, and outputs are released to the B register. In a 1-bit ALU, there are 2 operands – A and B. The result, R, is a single integer. In more complex ALUs, there are status registers, which identify abnormal situations such as under/overflow, zero, negative values, and carryover values. The input of a complex ALU is sorted through an opcode, in which a bus carries the values needed to the ALU along with the desired operation. The clock of the computer, which ticks at a certain steady rate, runs the ALU one time per cycle.

Addition

The most basic function an ALU performs is addition. The following 8-bit ALU is constructed from two smaller ICs: A half-adder, and a 1-bit ALU. Using a combination of XOR and AND gates, a simple half-adder can be built. A full adder can be composed by two half-adders with an additional carry-in signal (Ci) and the output is represented in SUM and a carry-out signal (Co). The benefits of a full adder include a carry bit input, which allows for an extra bit in output as well as series functionality when stacked together. In the full adder, there are designated connectors for carry bits, Ci and Co. When using multiple adders, the lower bit's carry out is connected to the higher bit's carry in. In addition, the highest bit carry out denotes the occurrence of an overflow, which is beyond the data representation space. Without a status flag, an overflow will display a portion of the information available, resulting in values significantly different from the expected result.

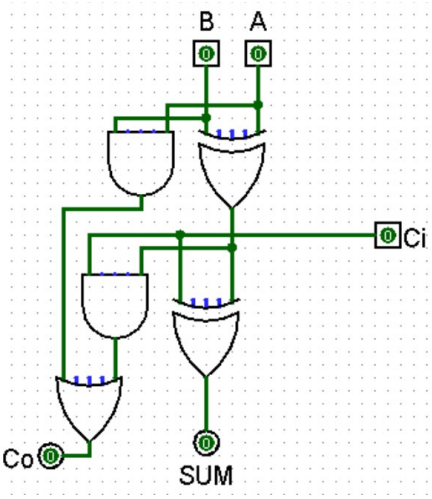

Figure 1. One-bit full adder created in Logisim.

The functions of the full adder shown in Figure 1 is represented in the following truth table.

Table 1. Truth Table for 1-bit full adder shown in Figure 1.

Inputs			Outputs	
A	**B**	**Ci**	**SUM**	**Co**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Note that in the case of all inputs being 1, the full adder is unable to display all of the data. Therefore, the data is only partially complete.

After a full adder is complete, it is possible to link up the adder with other XOR, AND, and OR gates to construct a 1-bit adder. This simple ALU has invert functionality (allowing for Subtraction in subsection 2) as well as logical functions such as OR/AND. With a 1-bit ALU finished, it is not difficult to assemble the 8-bit ALU; 8 1-bit ALUs are linked in series to achieve full 8-bit addition functionality. Of course, ALUs are much more complex than that. Implementation of status flags accounting for zero, negative, flow, and Co situations enhances the experience of diagnostics as well as provide an accurate representation of the current state of the ALU output.

Subtraction

Subtraction is another common arithmetic operation on an ALU. It is conducted in a similar method to addition. To subtract a number, that bit must be flipped. When adding a negative number, the operation is equivalent to subtraction. In this implementation (fig. 2), the input A and B can be inverted through a “Invert X” toggle. When this switch is toggled, it turns on the NOT gate and flips the value. In the 8-bit ALU, invert toggles are implemented system-wide. Inverting either A or B will cause the entire circuit to do so, and will only reset the following clock tick.

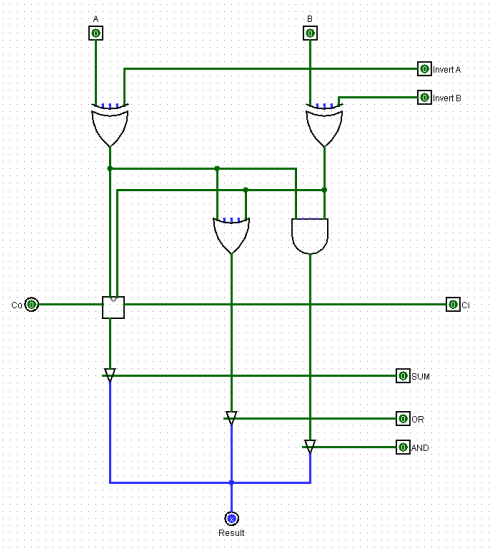

Figure 2. One-bit ALU with AND/OR functions created in Logisim.

Multiplication

Multiplication is a bit more complicated than simple addition. By design, ALUs themselves are unable to carry out multiplication. Rather, additional components and new hardware implementations are required. In order for an ALU to multiply, a shift is required in the register. A *shift register* is a register with controls for shifting its contents. Shift registers can be designed for either unidirectional or bidirectional shifts. They often have auxiliary operations for clearing the contents or for loading a new word in a single step. The diagram below shows the implementation of a shift register with operations for loading, left shifting, right shifting, and holding the current contents. Shift registers, with some modification, are used for generating error detection codes such as cyclic redundancy check (CRC) codes. These codes are widely used in networks and data storage. The multiplicand is shifted in one register, while the multiplier is stored in the second register. For each additional bit needed, a 1-bit shift must occur in the register. Multiplication is much more difficult to implement and may require the need for additional hardware in certain circumstances. In some cases, an ALU may be unable to multiply at all, for cost effectiveness as the addition of a multiplier may add too many logic gates.

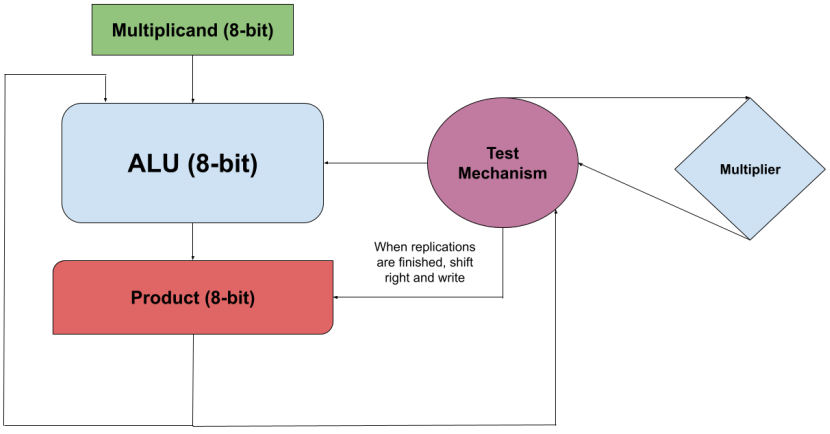

Figure 3. ALU Multiplication flow chart.

Division

Division is not a commonly used arithmetic operation inside of a CPU, so it is often not integrated into the core of the ALU. Division is done in a similar method to multiplication, using register storage for divisor/dividends and shifts when bits are removed. There are also several different methods, which may take shortcuts using algorithms. The main challenge with division is accounting for shift changes as well as what to do with the remainder (as well as how to signal the presence of a remainder and/or how to store that value).

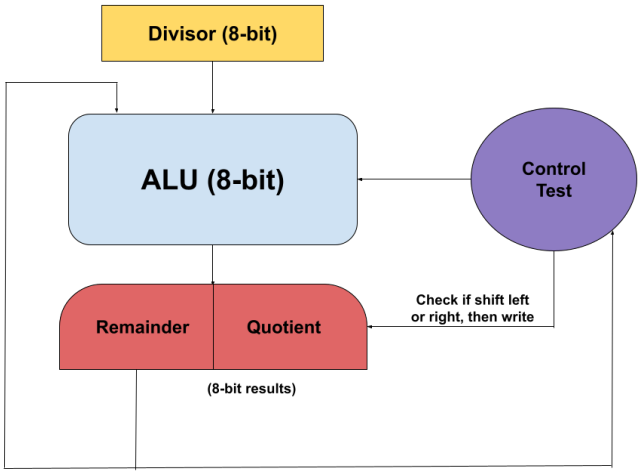

Figure 4. Logical outline for division mechanism.

Conclusion and Discussion

The primary focus of this technical report is to present findings on the operations of the inside of an ALU. The technical study conducted above is an attempt to understand CPU functionality as well as search for areas of possible efficiency. This paper dives in-depth on the fundamentals of computer components through use of early breadboard-type circuitry and the use of LS IC chips, as well as Logisim® simulator to analyze these components, in particular the CPU (Central Processing Unit) and the ALU (Arithmetic Logic Unit).

When Simulation mode is enabled, all processes of the ALU work as expected. Although the multiplication and division functions were not built into this ALU diagram below, it can be easily added using additional traces as well as the implementation of the A and B buses. Even though the ALU is a major piece in the full completed CPU (which itself, is a piece in a larger computer system), it is critical to processing arithmetic functions on binary data as well as implementing logic gates and filtering results based on flag truths.

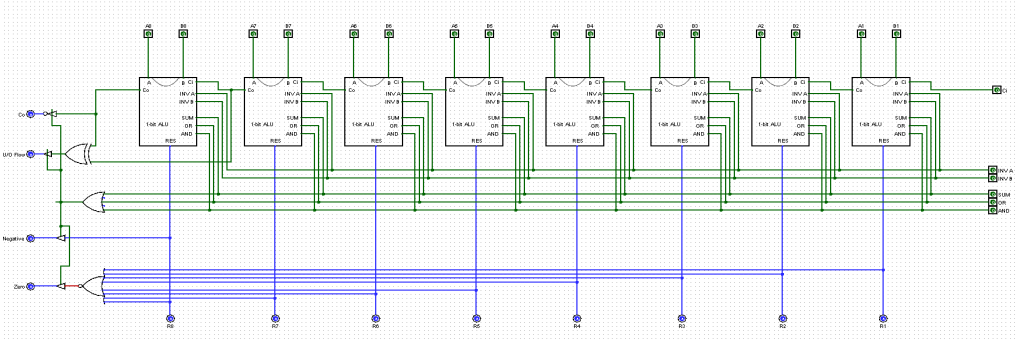

Figure 5. Completed 8-bit ALU diagram created in Logisim.

Limitations

There are indeed some limitations with this type of ALU design. First, there is no support for more than 8-bit operations. While modern computers are 32- and even 64-bit, this design is only able to process a small fraction of that amount. In addition, there are close to zero 8-bit CPUs still on the market today, with the notable exception of the 6502, which runs at 20Mhz (Modern CPUs are now able to hit 5Ghz, or 5000Mhz, a significant speed advantage per clock cycle). Lastly, with the advent of soldering and microfabrication facilities, the need for breadboarding a processor has been eliminated for all except educational purposes. These new machines can run traces at the same width of human hair with much more electrical potential.

Acknowledgements

I would like to thank Dr. Zhu (Wilkes University) for providing the physical resources for the project as well as assisting and guiding me through introduction to circuits. I would also like to thank Dr. Sha (Wilkes University) for also providing me assistance and reading resources throughout the summer months. I also thank Dr. Du (Wilkes University) for reading the initial concept as well as providing feedback for improvements. Lastly, I thank my parents and friends for their constant support through this project.

References

Patterson, D. A., Hennessy, J. L., & Alexander, P. (2015). *Computer organization and design: The hardware/ software interface* (3rd ed.). Morgan Kaufmann.

Rombauts, K. (2020, April 10). *Building an 8-bit computer in Logisim (Part 1 - Building Blocks)* [Lecture notes]. Medium. Retrieved August 17, 2021, from https://medium.com/@karlrombauts/building-an-8-bit-computer-in-logisim-part-1-building-blocks-a4f1e5ea0d03

Roth, C. H., Jr., & Kinney, L. L. (2010). *Fundamentals of logic design* (6th ed.). Cengage Learning.

Toomey, W., & Damian, M. (n.d.). *Logic Gates - Building an ALU* [Lecture notes]. Villanova University CSC. Retrieved August 17, 2021, from http://www.csc.villanova.edu/~mdamian/Past/csc2400fa13/assign/ALU.html