Arithmetic Logic Unit
Computer Architecture

José Douglas Gondim Soares, Rafael Brandão Cruz [1]

[1]Ciência da Computação – Universidade Federal do Ceará (UFC)
Rua Felipe Santiago 411, Cidade Universitária – 62.900-000 – Russas – CE – Brazil

douglasgondim15@gmail.com, rafaelbrandaocruz@hotmail.com

Abstract. *This paper describes the operation of an Arithmetic Logic Unit, as well as the steps to its design. It presents the tools and technologies used in the design and development of digital circuits*

1. Introduction

Today's computers are complex machines with such a great number of circuitries, that it's nearly impossible to understand them without the use of abstraction. As you head towards the bit level, you can see that computers are actually nothing more than a number of logic gates, organized in a smart manner. The purpose of this work is to understand how those logic gates work together, to realize simple arithmetic operations. For that, we have chosen to analyse a 2-bit processor that does only one operation (addition).

2. The functionality of an ALU

An ALU (Arithmetic Logic Unit) is the part of the computer that's responsible for making every arithmetic and logic operation. Simple architectures such as RISC (Reduced Instruction Set Computer) are computers that can't do a lot of mathematical operations in hardware. Those computers make important arithmetic operations, such as multiplication and division, using only a sequence of additions. Three times two, for example, is the same as the number two added to itself, three times.

The downside to RISC architectures is that they are usually slower.

There are also CISC (Complex Instruction Set Computer) architectures. That architecture, differently from RISC, is capable of making more complex operations on hardware. As a consequence, those are much faster, but are also harder to be implemented and more expensive, as well. Although, for simplicity purposes, we are only going to analyze a RISC processor. Our processor is only going to work with 2 bits, which means that it can only work with up to two raised to the power of two numbers, or in other words, four different numbers. So, our processor is going to be able to represent the interval of numbers between zero and three.

The addition of two bits is not complicated to understand. We have the following possible cases:

Table 1. Example

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10 (0 + carry bit)

3. Topology

The half adder circuit has two inputs and two outputs as shown in the diagram below. The inputs A and B represent two individual bits: the Sum output represents the sum of A + B and the Carry output is the carry bit from the addition.

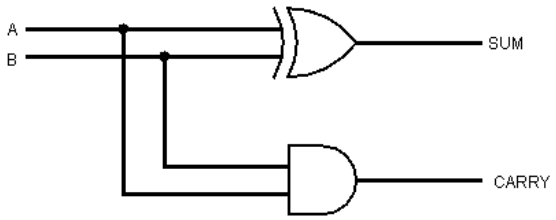

Figure 1. Half adder, which uses an AND gate and an exclusive-OR (XOR) gate

Table 2. Truth table for half adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

This circuit cannot take in a carry bit from a previous operation. The full adder circuit, on the other hand, has two bit inputs, a carry in, a sum out, and a carry out.

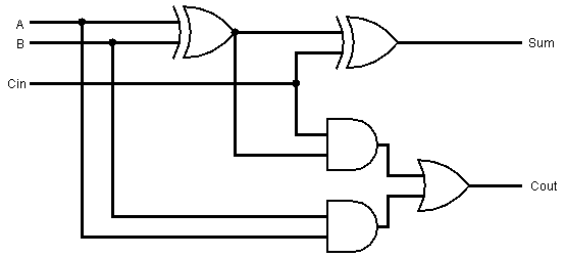

Figure 2. A full adder made by using two half adders and an OR gate

Table 3. Truth table for full adder

A	B	Cin	Sum	Cout
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

4. Conclusion

We have come to the conclusion that the ALU is the main component of any computer. Since we have an extremely high level of abstraction to deal with, it can get quite hard to understand what is going on at the lower levels. We hope that this work has made it clear how a simple ALU works. How even the most complex of instructions are translated to simple.

5. References

Mitchell, R., (2016) ”How to Build Your Own Discrete 4-Bit ALU”, https://www.allaboutcircuits.com/projects/how-to-build-your-own-discrete-4-bit-alu/, May.

Gonick, L. (1984) ”Introdução Ilustrada a Computação - com muito humor.”, otre, V.P. e Barbieri, E.L. Ed. Harper Row do Brasil Ltda. S. Paulo