Analytical Comparison of Full-Adder Design Characteristics from the Years 2002 to 2016

Hunter Tanchin
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract—In an era of rapidly advancing technologies, it is often helpful to observe trends of the common characteristics of what engineers and scientists have discovered. Trends such as Moore’s Law help us to get an idea of where our technology is headed and gain a clearer vision for how we can push the boundaries of what we are able to accomplish. This paper analyzes a very small sample size of research publications from the last decade and a half on the topic of Arithmetic Logical Units. This paper will analyze characteristic trends, namely, power, space complexity, data paths, and execution time of the various full-adder designs. Included in the sample are CMOS, MTJ, and DWMN processing technologies. Despite their differences, these are still functionally the same and their trends are still apparent, applicable and useful in this comparison.

Keywords—Arithmetic Logical Unit, Floating Point Unit, John von Neuman stored program concept, MTJ, CMOS, ITRS, DWMN, gates, control path, data path, space optimization, execution time, energy consumption, power dissipation

I. INTRODUCTION

One of the most foundational concepts of digital computation, is the implementation the base two system for data transfer and storage. Information could be realized with a simple "charged" or "not charged" state of a device, such as a transistor. Today, by a combination of many transistors (a more recent invention) that are connected to one another in different arrangements, a various number of functions can be accomplished, resulting in computed, algebraic outputs (e.g., addition, division, comparisons (Boolean functions)). A device containing these transistors that outputs the results of computations, given one or more inputs, is called an ALU.

One specific ALU design that takes advantage of this concept is called the Floating Point Unit design. While summing integers such as 2 and 3 is useful for some computational applications, there are scenarios in which it is necessary to make a computation using fractional values that have much more precise values. For instance, with a Floating Point Unit device, the sum of decimal values 2.5 and 2.25 is 4.75. Adding these numbers using a standard integral ALU would return their integer sum: 2 + 2 = 4.

But how do the ALU and FPU actually come up with those values? What makes an adder behave a certain way while a multiplier gives a very different value when they are both composed of transistors? This question leads to the second important piece of an ALU.

Another foundational implementation for calculating values is the gate. There are multiple gate designs. Given two bit values (a 1 or 0) (more accurately, an applied voltage, such as 1.2, or the absence of a voltage: 0), these gates are designed to give one output bit (a 1 or 0) based on the desired relationship these values have to one another. By combining these gates in certain sets and patterns, algebraic computations are made.

So, how does the ALU know, for example, when to subtract and when to add? How do these voltages go through their respective paths, transistors and gates? This is where ‘instructions’ come in. While all values in an ALU are composed of ‘ones’ and/or ‘zeros’, some input bits can be interpreted as simply numbers while other input bits can be instructions. How the ones and zeros are directed through the ALU determines the behavior of the output.

From the entrance that receives input values to the exit that provides an output value, there are wires that are called ‘data bus paths’ and ‘control paths’. While these are physically the same metal conducting wires, their functions are different based on their purpose. When an instruction is passed to an ALU, this binary value is traveling along the data bus path (this is a bunch of wires, typically 32 or 64 in number). Instruction values control certain gates in the ALU by design to manipulate the computation of the numeric values. Control paths carry instruction bits through the ALU to store and read values in memory. The locations of these values are called registers.

Both of these designs are incredibly useful to nearly all digital processing applications. All computers run on basic designs surrounding these technological concepts. But greater achievements only come from improvements of these concepts. Much of the work on the improving computers over the last few decades has included maximizing the three basic resources needed to make computations with ALU devices: time, space, and energy. Currently, researchers are able to develop ALUs that have smaller components (optimizing available space) which allow electricity to travel shorter distances for computations (reducing execution time). However, by speeding up the movement of the electrons (increased energy consumption), more heat is lost within the components (increased power dissipation). As components get smaller, more can be fit onto a
chip. They evolution of reducing this individual transistor size is mapped out by the International Technology Roadmap for Semiconductors (ITRS). In the ITRS report, one can find the recorded and projected improvements to the size of semiconductor components. The physical restraints of semiconductors are projected to be limiting our ability to make further such size advances in the next couple of decades. This has made a need for new technologies!

This paper is intended to analyze some of the results of the previous decade and a half of work dedicated to improving the three characteristics of computational analysis in CMOS processing. This analysis will also look at and compare some of the future replacement technologies for CMOS. The next section outlines in more detail the eight design papers that were sampled in order to accomplish this.

II. LITERATURE REVIEW

Just after the turn of the century, the leading edge of semiconductor processing technology was starting to break the 130 nm mark. Just as processors utilizing this advancement were reaching the market, two papers were produced from research pertaining to advancing the performance and speed of CMOS ALU designs. This first, finalized in 2001 by a professor and graduate student from Louisiana State University [4], succeeded in modeling a faster 4-bit ALU that uses only 0.4 V while increasing the speed by 200-250%. The second paper, written in 2004, [6] chronicles an improved design of a 32-bit ALU (Altera’s NIOS 2.0 soft processor) by using “a barrel-shifter and custom instructions” to improve the runtime on Altera’s Apex 2KE FPGA architecture by 70% while only taking up half the size.

Four years after this paper was published, researchers at Tohoku University in Japan released a series of papers detailing a developing technology that was considered for replacing the widely-used, and mortal, CMOS technology: Magnetic Tunnel Junctions (MTJs). While the details of this technology are beyond the scope of this paper, the basic John von Neuman principle still holds and thus can be compared equivalently to CMOS technology ALU’s in terms of functional results. In the further development of this model, one of the papers that these researchers published explains how they developed a full-adder design with MTJ technology using Logic-in-Memory architecture [7]. The resulting design were compared with an 1800 nm CMOS ALU that had a similar space complexity and delay time. What this comparison showed was that this very immature MTJ technology used nearly 25% less power @ 500 MHz and eliminated the need for static power to maintain the bit voltage. This paper helped show that this technology is indeed a feasible replacement to CMOS designs.

In addition to technology advancements, it is important also to continue with design and modeling advancements that new technologies can be applied to. Such a project, released in 2011 in the VLSI journal by Iranian researchers [3], sought to improve the speed and transistor count of a full-adder (FA) by using a minority function. By comparing their design against six other top FA designs, the researchers found that their design produced results very similar to only by one other design, using slightly more power, but achieving faster delay times and lower PDP than any other design tested.

Another paper written in 2014 proposes two different designs that both take advantage of an ALU concept called “Reversible ALU” [5]. In this design, the ALU produces a functional output for each input (for example, given A=2 and B=3, the 2-bit ALU may produce C=A+B=5 and D=B-A=1). The two designs, both comparable to one another, outperformed the conventional ALU that they were tested against. Additionally, this paper showed both an implantation of these designs with quantum technology and also one with the still developing CMOS technology. A more recent paper (2015) completed here at UCF explored the results of using 15 nm CMOS technology in ALUs compared to 45 nm technology [1]. By using an IEEE 754 Single Precision Floating Point Unit implementation, these researchers found that using the smaller semiconductor significantly reduced the resources needed for computations. Their design used only two-thirds of the space while using a quarter of the energy to realize the same outputs.

Also last year, another Full-adder design was released [8] that used another CMOS complement/replacement technology: All-Spin Logic (ASL). Compared with a 40 nm CMOS FA, the ASL Full Adder took up an incredibly small area. At 3% of the space needed, the ASL FA used zero power to maintain a voltage. Extremely non-volatile! For these benefits, however, the cost is a 62% increase in delay time and a 1.84-fold increase in power consumption for processing @ 500 MHz.

The last ALU design paper [2] was just published by IEEE in August of 2016. As another alternative to CMOS, this domain-wall nanomagnet full-adder (DWMN-FA) outperforms the semiconductor technology by 52% and 41% in space complexity and device count, respectively. Similarly, it even beats out the MTJ technology by nearly equivalent amounts: 49% and 31%. High-speed and low-power modes were both tested and found to give similar results for the characteristic they are trying to improve as CMOS and MTJ devices, keeping in mind the smaller area and devices needed for such results.

III. DATA ANALYSIS

![Fig. 1](https://example.com/fig1.png)

Fig. 1. This is a graph displaying the number of operands and the datapath width used in ALU designs since 2002.
Fig. 2. This is a graph showing the node width and chip area designed in adder components over time since 2002.
* This is the average of the 38 nodes used in the ALU

Fig. 3. This is a graph displaying the amount power used per instruction by ALU designs since 2008.
** Average of Low Power and High Speed designs

Fig. 4. This is a graph displaying the time taken per execution by ALU designs since 2002

IV. CONCLUSION

Optimization of electronics is always a tradeoff. As seen in the graphs, the execution time of arithmetic instructions in a full-adder unit decreases over time. By having smaller packages (space complexity) in which data and control paths are shorter, or, better technology, we see faster times. At a cost, however, these devices sacrifice low energy consumption. As the technology advances, the amount of power drawn does also.

REFERENCES


<table>
<thead>
<tr>
<th>ALU or Floating Point Architecture Name</th>
<th>Device Technology</th>
<th>Datapath width (bits) or #bits in operands</th>
<th>Time for Operation or Design Type</th>
<th>ITRS Technology Node (nm) or Area (µm²) or Model of Chip used</th>
<th>Energy/Power Consumption (W or J) else indicate “low” or “high”</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS 15nm Floating-point Unit [1]</td>
<td>CMOS</td>
<td>32-bit Datapath</td>
<td>NO</td>
<td>NO</td>
<td>5 ns IEEE 754 SINGLE-PRECISION FP-UNIT 15nm</td>
</tr>
<tr>
<td>Full-Adder w/ modified function and bridge style [3]</td>
<td>CMOS</td>
<td>2-bit input (+ Cₘₙ) (Operands)</td>
<td>32.068 x 10⁻¹² s delay for high-speed, high-performance Full Adder</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Improved 32-bit FPGA ALU design [6]</td>
<td>FPGA</td>
<td>32-bit Datapath</td>
<td>5.0ns Apex 20KE LE</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Reversible ALU [5]</td>
<td>PFAG and HNG</td>
<td>Up to 16-bits (Operands)</td>
<td>Up to 8.26 ns Reversible ALU</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Spin-Logic Full-Adder Circuit [8]</td>
<td>ASL (Spintronic)</td>
<td>2-bit input (+ Cₘₙ) (Operands)</td>
<td>3.8 ns G-ASLG</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Majority Gate, Nanomagnetic Full-Adder [2]</td>
<td>MTJ</td>
<td>3-bit input (Operands)</td>
<td>219 ps delay for DWNM-FA</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Nonvolatile logic-in-memory Full-Adder design [7]</td>
<td>MTJ</td>
<td>2-bit input (+ Cₘₙ) (Operands)</td>
<td>10 ns/bit Proposed MTJ Full-Adder</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Low Voltage, Ripple-Carry Adder [4]</td>
<td>CMOS</td>
<td>4-bits (Operands)</td>
<td>~185 ns Ripple-Carry Adder</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>