Advances Power Efficiency of an Arithmetic Logic Unit through Size and Design

Ayesha A. Arif
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract— In this paper various Arithmetic Logic Unit (ALU) designs and their metrics are discussed. The metrics found were used to investigate any trends in size and power efficiency within each design. Chip size, number of bits (operands), and the power efficiency were the main metrics measured for the researched ALU designs. Many of the designs cited span over 14 years of research and provide significantly different approaches. The ALU design using Probabilistic Domain Transformation is one of the most abstract ALU designs discussed. Another design that uses a unique approach is the QCA full adder. Both designs offer a different perspective on ways to improve ALU design.

Keywords— Arithmetic Logic Unit, operand, probabilistic, clock rate, floating point unit, ITRS, chip size, power efficiency

I. INTRODUCTION

Each Arithmetic Logic Unit is unique to a particular design. Various designs and implementation techniques exist to improve power efficiency and size. A few ALU designs are the Ripple Carry Adder (RCA), the Carry Look Ahead (CLA), and the Carry Select Adder (CSA). These designs provide a sufficient amount of information to materialize the effects of size and design on power efficiency.

The RCA is the least gate dependent design. It allows efficiency in size, while having a large gate delay. The delay in a computation causes more power consumption.

The tradeoff between power and size isn’t always justifiable. The CLA is a design that has a faster computation time, but also requires more gates. This design takes up more space, which allows all of the carriers to be used at the same time.

Each design’s disadvantages are either reduced or manipulated to gain an effective outcome. The CSA is designed to allow calculations to occur simultaneously. The calculations are subject to be redundant causing a lot of wasted energy. Out of the three, this design is, by far, the fastest.

Every ALU design has the ability to add or subtract. With these two operations, multiplication and division can also be calculated. Computing the operands is fundamental to ALU design. Depending on what the operands are used for, more power can be consumed. For example, add and subtract are very fast computations. On the other hand, multiplication and division are more abstract and require a lot more power to work.

The ALU is a set of Boolean gates designed to allow addition and subtraction to occur within a CPU. The ALU is a piece of hardware that can compute an input and produce and arithmetic output. The calculations are completed when a value is received from a specific address (opcode) through the data bus. Only one operation can be completed at a time.

Here is a way to realize what is generally occurring in an ALU. If two values are being added together, the first value is retrieved from memory by locating a specific (unique) address. The value is brought back to the CPU using a data bus. While that value is being held the other value is being retrieved using the same method. The values retrieved are stored in different registers before computation occurs. The result obtained through the computation will be stored in a new register.

A data bus allows communication between components within a computer. Any information that needs to be used/computed and or located needs to go through a data bus. In the example of addition above, a data bus is used to retrieve the desired register file. The ITRIS technology node is a set of instructions/guidelines to follow when it comes to semiconductor devices. Its goal is to keep up with the latest technological needs and find ways to overcome the obstacles such as chip size using a cost effective method. Execution time is the time needed to compute the desired outcome. Execution time relies heavily on the design of the ALU. If there are more gate delays, it takes more time to compute the desired outcome. Redundancies within the design can also lead to a longer execution time. With any given ALU design, power dissipation is inevitable. Whenever a choice is made within the Boolean gate power is being dissipated. The objective of each designer is to find a way to reduce the power dissipation with the ALU design, while keeping the other metrics in mind. Each time a design grows in size so does the energy consumption. Naturally more energy is needed to implement more calculations. Manipulating the design to perform the same operation without taking up as much space can allow the energy consumption to reduce immensely.

In Section 2 the Literature of ten ALU designs will be discussed from year 2015 to 2001. In Section 3 metrics will be analyzed through graphs. Finally, section 4 will summarize the analysis.
II. LITERATURE REVIEW

In 2015 many designs were introduced and tested. The floating point unit was tested in a 15 nm ITRS node compared to a 45nm ITRS node in [1]. In [2] a Quantum-dot Cellular Automata (QCA) was used to replace the commonly used binary adders in digital system designs. [6] Presented information based on a RISC processor that only contained an ALU, Universal Shift Register and a Barrel shifter. In one the size was reduced by 30 nm and the power consumption by 1.414 mW. The QCA adder used an 18 nm^2 cell area to produce low power consumption compared to a binary adder. The 16-bit RISC processor used Xilinx Kintex XC7K16078-3bg676 28 nm Technology to create a total power consumption on .220 Watts. The results for each design had resulted in a more efficient size and power design. There was as an interesting study using Probabilistic Domain Transformation used in 2014. In the design values were converted into random samples through Digital Signal Processing techniques. The random samples were then used to perform calculations using addition instead of multiplication. Once the values are calculated they are converted back and the value is then presented. This technique allows more calculations to be performed with a higher power efficiency, 166.63 nJ, compared to using a 4.09µs energy efficient multiplier.[3]

In 2013 a clock gating technique was studied for a 64-bit green ALU. Through using this technique, clock power was reduced 67.74% and 65.84% and leakage was reduced 93.82% and 93.71% for 1 GHz and 10 GHz frequencies[7]. The overall power was also reduced significantly through this design. For 1GHz, 10 GHz, 100 GHz, and 1 THz, the dynamic power resulted in 35.10%, 36.03%, 39.21% and 39.53%. Like [1], [10] also investigates the Floating-Point ALU. This design was implemented in 2009 using IEEE-754 standard for the floating point number, the multiplication and division techniques are executed. A RISC processor is used in the execution of the design. The desired outcomes of the 4-level pipelining design are obtain with the use of very little power.

A 32-bit (ALU) is used to support a design-for-test (DFT) in [9]. The DFT detects the faults in the clock frequency. Using this method in 2005, energy was reduced from 32% to 22% for the 180nm CMOS and 12% to 21% for the 65nm CMOS. [9] If this design is implemented low-power operations would occur and this would give the device a longer life span. In 2003 a 2-D pipelining design was introduced to improve throughput for adders and multipliers. [4] This resulted in increasing the throughput drastically to reduce the delay. Along with the throughput, power deficiency was also reduced. Overall, this design allows calculations to be completed a lot faster than the original design, while using less power because the number of delays are reduced. [4] Throughput is also the main topic discussed in [5]. In this paper a self-timed circuit is created where no delays will occur. It was concluded that the Baugh-Wooley design was the best design to gain a higher throughput. The design used the least amount of gates, therefore making the area smaller. In a study done in 2001, it was discovered that the a redesigned ALU using the novel deep-stack quaternary tree architecture offered a 19% performance improvement.[8] This along with the margining configurations proved that the speed offers a trend of decreasing with scaling in SOI technology. [8]

III. DATA ANALYSIS

The graph below describes the amount of bits used in each design over the years. There is very little consistency seen within Figure 1 due to the variety of designs that were implemented.

![Graph describing the trend of bits over 14 years](image1)

![Graph describing the trend of chip size over 14 years](image2)
The graph in Figure 3 describes the trend of frequency over the past 14 years. Frequency played a large part in power reduction in some of the designs discussed.

In Figure 4 the trend of power over the past 14 years is shown. With the data gathered, it can be seen that with the latest designs there is very little power loss.

IV. CONCLUSION

Upon reading and analyzing the referenced papers, there seems to be many ways to manipulate an ALU to achieve unique outcomes. Trends, such as Moore’s law, appear to show promise in design. With the chips size becoming smaller, new innovations occur to still gain efficiency within design in attempt to maximize computation, decrease power consumption, and still minimize size. Through cited research, the basic ALU design was always designed for a specific method. Most designs were comparisons and most resulted in improving the ALU. Signals are also a powerful tool that can be used with ALU design to make the process of computing more efficient.

REFERENCES

<table>
<thead>
<tr>
<th>ALU or Floating Point Architecture Name</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 Model of Chip used</th>
<th>Energy/Power Consumption (W or J) else indicate “low” or “high”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy and Area Analysis of a Floating-Point Unit [1]</td>
<td>32 bits (Operands)</td>
<td>N/A</td>
<td>N/A</td>
<td>IEEE-754 Single Precision</td>
</tr>
<tr>
<td>Ultra-area-efficient fault-tolerant QCA full adder [2]</td>
<td>1 bit (Operands)</td>
<td>Ultra-area-efficient fault-tolerant QCA full adder</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>High Throughput Power-aware FIR Filter Design based on Fine-grain Pipeline Multipliers and Adders [4]</td>
<td>16 bit (Operands)</td>
<td>FIR</td>
<td>FIR</td>
<td>N/A</td>
</tr>
<tr>
<td>NULL convention multiply and accumulate unit with conditional rounding, scaling, and saturation [5]</td>
<td>32 bit (Operands)</td>
<td>N/A</td>
<td>Modified Baugh-Wooley MAC and Booth2 algorithms</td>
<td>N/A</td>
</tr>
<tr>
<td>Design &amp; analysis of 16 bit RISC processor using low power pipelining [6]</td>
<td>16 bit (Operands)</td>
<td>RISC</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>64 bit green ALU design using clock gating technique on ultra scale FPGA [7]</td>
<td>64 bit (Operands)</td>
<td>Clock gating</td>
<td>Clock gating</td>
<td>N/A</td>
</tr>
<tr>
<td>Sub-500-ps 64-b ALUs in 0.18- μm SOI/Bulk CMOS: Design and Scaling Trends [8]</td>
<td>64 bits (Operands)</td>
<td>Han-Carlson</td>
<td>Han-Carlson</td>
<td>N/A</td>
</tr>
<tr>
<td>Design of a 1.7-GHz low-power delay-fault-testable 32-b ALU in 180-nm CMOS technology [9]</td>
<td>32 bits (Operands)</td>
<td>DFT</td>
<td>DFT</td>
<td>N/A</td>
</tr>
<tr>
<td>32 bit Multiplication and Division ALU Design Based on RISC Structure [10]</td>
<td>32 bits (Operands)</td>
<td>N/A</td>
<td>RISC</td>
<td>IEEE-754 standard</td>
</tr>
</tbody>
</table>