International Journal of Computational Research and Development (IJCRD)
Impact Factor: 4.775, ISSN (Online): 2456 - 3137
(www.dvpublication.com) Volume 1, Issue 2, 2016

LOW POWER AND LOW AREA MULTIPLICATION CIRCUITS THROUGH PARTIAL PRODUCT PERFORATION

E. Sindhu Dharani* & V. Sharmila Raj**
* PG Scholar, SSM College of Engineering, Komarapalayam, Tamilnadu
** Associate Professor, Department of ECE, SSM College of Engineering, Komarapalayam, Tamilnadu

Cite This Article: E. Sindhu Dharani & V. Sharmila Raj, "Low Power and Low Area Multiplication Circuits through Partial Product Perforation", International Journal of Computational Research and Development, Volume 1, Issue 2, Page Number 44-48, 2016.

Abstract:

Focus on hardware-level approximation by introducing the partial product perforation technique for designing approximate multiplication circuits. The partial product perforation method for creating approximate multipliers. It omit the generation of some partial products, thus reducing the number of partial products that have to be accumulated; we decrease the area, power. The major contributions of this work, the software-based perforation technique on the design of hardware circuits, obtaining the optimized design solutions regarding the power–area–error tradeoffs. Analyze in a mathematically rigorous manner the arithmetic accuracy of partial product perforation and prove that it delivers a bounded and predictable output error. Error analysis is not bound to specific multiplier architecture and can be applied with error guarantees to every multiplication circuit regardless of its architecture that compared with the respective exact design, the partial product perforation. Index Terms: Approximate Arithmetic Circuits, Approximate Computing, Approximate Multiplier, Error Analysis & Low Power.

1. Introduction:

Multipliers form an important hardware block in the DSP and Embedded applications. Multiplication speed determines processor speed. So high speed multipliers are needed in the processors for many applications. For increase the speed of multiplication different algorithms are used Multiplication is a most commonly used operation in many computing systems. A number (multiplicand) is added to itself a number of times as specified by another number (multiplier) to form a result (product). But the implementation of multiplier takes huge hardware resources and the circuit operates at low speed. Multiplication is one of the fundamental components in DSP and Embedded system. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the real time system. Multiplication consists of three major steps: 1) re-coding and generating partial products 2) reducing the partial products by partial product reduction schemes to two rows and 3) adding the remaining two rows of partial products by using a carry-propagate adder (e.g. Carry look ahead adder) to obtain the final product.

2. Overview of Multiplier:

Multiplication is a fundamental operation in most signal processing algorithms. Multipliers have large area, long latency and consume considerable power. Therefore low power multiplier design has an important part in low-power VLSI system design. Hence optimizing the speed and area of the multiplier is one of the major design issues. However, area and speed are usually conflicting constraints so that improvements in speed results in larger areas. Multiplication is a mathematical operation that include process of adding an integer to itself a specified number of times. Multipliers play an important role in today's DSP and various applications. Multiplier design. Should offer high speed, low power consumption. Multiplication involves mainly 3 steps

- ✓ Partial Product Generation
- ✓ Partial Product Reduction
- ✓ Final Addition

3. Techniques:

A. Partial Product Generators (PPG): An under-designed 16×16 multiplier using inaccurate 2 × 2 Partial Product Generators (PPG) while guaranteeing the minimum and maximum accuracy fixed at design time. Each PPG has fewer transistors compared with the accurate 2×2 one, reducing both dynamic and leakage energy at the cost of some accuracy loss novel iterative log approximate multiplier using Leading One Detectors (LODs) to support variable accuracy. A variety of computer arithmetic techniques can be used to implement a digital multiplier. Most techniques involve computing a set of partial products, and then summing the partial products together. This process is similar to the method taught to primary schoolchildren for conducting long multiplication on base-10 integers, but has been modified here for application to a base-2 (binary) numeral system.

B. Multiplication Algorithm:

- ✓ If the LSB of Multiplier is 1, then add the multiplicand into an accumulator.
- ✓ Shift the multiplier one bit to the right and multiplicand one bit to the left.
- ✓ Stop when all bits of the multiplier are zero.

For designing a multiplier circuit we should have circuitry to provide or do the following three things:

Step 1: It should be capable of identifying whether a bit 0 or 1 is.
Step 2: It should be capable of shifting left partial products. Array Multiplier is an efficient layout of a combinational multiplier. Array Multiplier gives more power consumption as well as optimum number of components required but delay for the multiplier is larger. Array Multiplier is less economical. It is a fast multiplier but hardware complexity is high. Wallace Multiplier is much faster than Array Multiplier. Minimizing the delay is important in reducing the overall multiplication time. Dadda Multiplier is the extended process of Wallace Multiplication .Minimum Number of reduction should be done in order to reduce the partial product. Dadda method requires same level as the Wallace Method. The 4-2 compressor is another widely used building block for high precision and high speed multipliers. The block diagram of a 4-2 compressor is shown in the figure, which has four inputs and three outputs.

4. Proposed System:

According to previous existing design, there is lots of issues in terms of power, area, and speed. In this research area, still there is lots of future work is require. This is to provide an estimate of the amount of energy and power consumed by the units to implement. The priorities of the future research objectives are, in order of importance, are:

- ✓ Robust and Safe Circuits.
- ✓ Design Time
- ✓ Area/Speed Balance

Approximate Compressor Design: The carry output in an exact compressor has the same value of the input cin in 24 out of 32 states. Therefore, an approximate design must consider this feature. In Design 1, the carry is simplified to cin by changing the value other 8 outputs. Since the Carry output has the higher weight of a binary bit, an erroneous value of this signal will produce a difference value of two in the output. For example, if the input pattern is "01001", the correct output is "010" that is equal to 2. By simplifying the carry output to cin, the approximate compressor will generate the "000" pattern at the output (i.e. a value of 0). This substantial difference may not be acceptable; however, it can be compensated or reduced by simplifying the cout and sum signals.

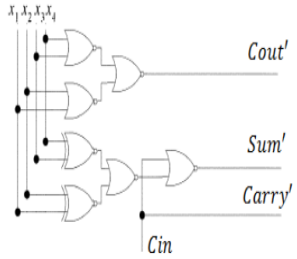

Figure: Approximate Compressor Design1

Approximate Compressor Design 2: A second design of an approximate compressor is proposed to further increase performance as well as reducing the error rate. Since the carry and cout outputs have the same weight, the proposed equations for the approximate carry and cout in the previous part can be interchanged. In this new design, carry uses the right hand side of the module and cout is always equal to cin; since cin is zero in the first stage, cout and cin will be zero in all stages. So, cin and coutcan be ignored in the hardware design.

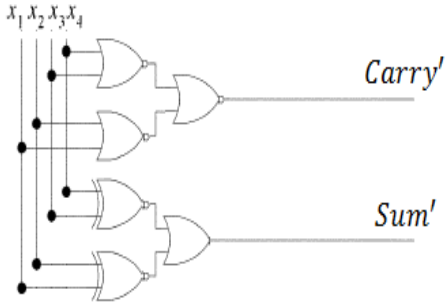

Figure: Approximate Compressor Design 2

5. Software Description:

Model Sim: Model sim is a hardware simulation and debug environment primarily targeted at smaller ASIC and FPGA design. Mode lsim combines simulation performance and capacity with the code coverage and debugging capabilities required to simulate multiple blocks and systems and attain ASIC gate-level sign-off. Comprehensive support of Verilog, System Verilog for Design, VHDL, and System C provide a solid foundation for single and multi-language design verification environments. Model sim easy to use and unified debug and simulation environment provide today's FPGA designers both the advanced capabilities that they are

growing to need and the environment that makes their work productive. Model sim is a verification and simulation tool for VHDL, Verilog, system Verilog, and mixed language designs.

Basic Simulation Flow: The following diagram shows the basic steps for simulating a design in Model Sim.

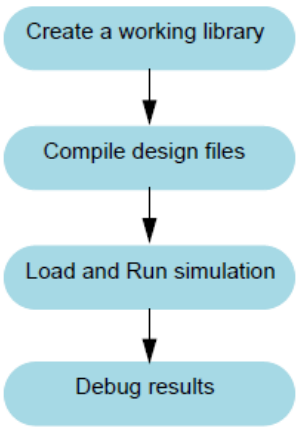

Figure: Basic Simulation Flow

Creating the Working Library. In Mode lsim, all designs are compiled into a library. You typically start a new simulation in Model sim by creating a working library called "work," which is the default library name used by the compiler as the default destination for compiled design units. Compiling your design after creating the working library, you compile your design units into it. The Model sim library format is compatible across all supported platforms. You can simulate your design on any platform without having to recompile your design.

Project Flow: A project is a collection mechanism for an HDL design under specification or test. Even though you don't have to use projects in Model sim, they may ease interaction with the tool and are useful for organizing files and specifying simulation. The flow is similar to the basic simulation flow. However, there are two important differences:

- ✓ You do not have to create a working library in the project flow; it is done for you automatically.
- ✓ Projects are persistent. In other words, they will open every time you invoke Model sim unless you specifically close them.

Multiple Library Flow: The diagram above shows the basic steps for simulating with multiple libraries. Model sim uses libraries in two ways:

As a local working library that contains the compiled version of your design;

As a resource library. The contents of your working library will change as you update your design and recompile.

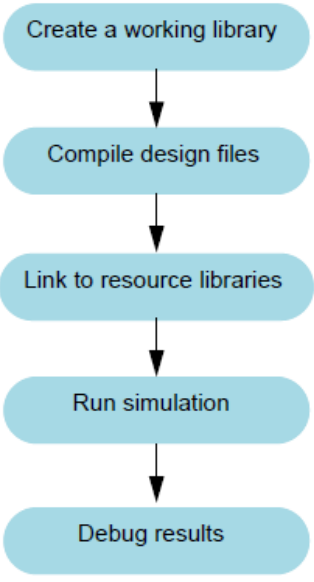

Figure: Multiple library flow

A resource library is typically static and serves as a parts source for your design. You can create your own resource libraries, or they may be supplied by another design team or a third party (e.g., a silicon vendor). You specify which resource libraries will be used when the design is compiled, and there are rules to specify in which order they are searched. A common example of using both a working library and a resource library is one where your gate-level design and test bench are compiled into the working library, and the design references

gate-level models in a separate resource library. You can also link to resource libraries from within a project. If you are using a project, you would replace the first step above with these two steps: create the project and add the test bench to the project.

6. Simulation Results:

Accurate Array

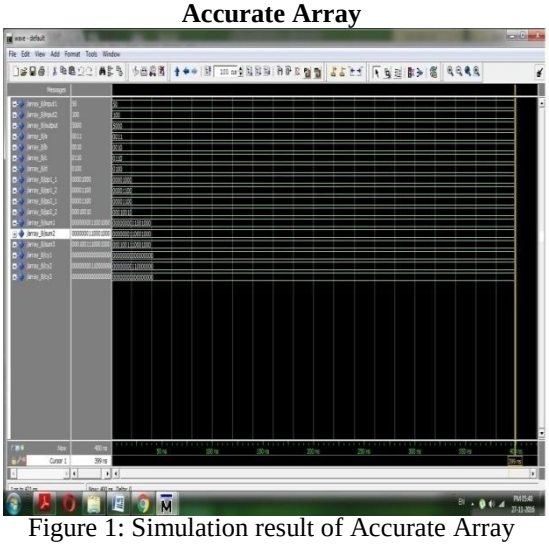

Figure 1: Simulation result of Accurate Array

Accuarte Wallace:

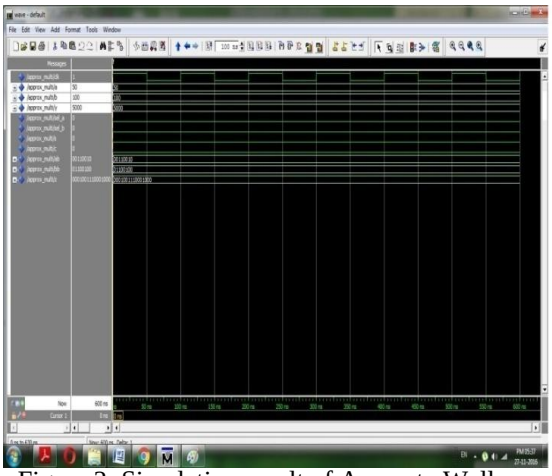

Figure 2: Simulation result of Accurate Wallace

Accurate Compressor 4:2

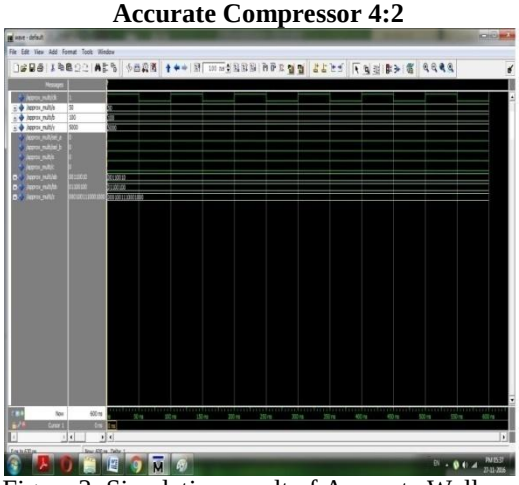

Figure 3: Simulation result of Accurate Wallace

Accurate Dadda 4:2

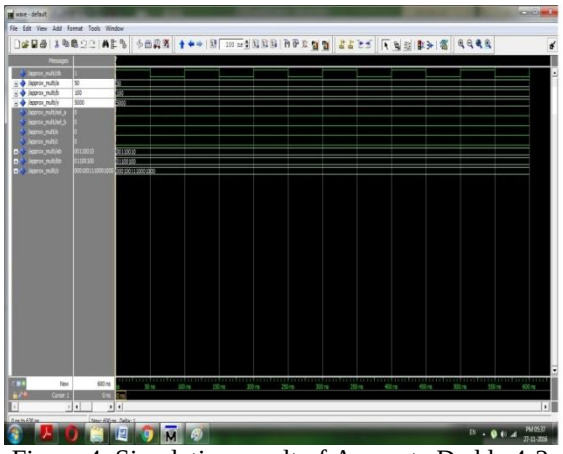

Figure 4: Simulation result of Accurate Dadda 4-2

7. Conclusion:

In this proposed work the partial product perforation technique for producing approximate hardware multipliers. The proposed technique omits a number of partial products enabling high area and power savings while retaining high accuracy. Through a rigorous error analysis, we analytically characterized the induced error metrics proving that the error is bounded and predictable and we proposed two error correction methods that trade a small increase in power for high error reduction. We explored product perforation on a large set of multiplier architectures, evaluating its impact on different architectures and error bounds. In comparison to the state-of the-art approximation techniques, we showed that the proposed approach achieves significant gains in power, area, and quality metrics of image processing and data analytics algorithms. Finally, we showed that our technique is scalable, offering better results as the multiplier's bit width increases.

8. References:

1. C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc.Conf. Design, Autom. Test Eur., Mar. 2014, Art. No. 95.
2. Narayanamurthy, H.. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large Scale Integr.ationnvol. 23, no. 6, pp. 1180–1184, Jun. 2015.
3. S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbias multiplier for approximate applications," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, Nov. 2015, pp. 418–425.
4. R. Venkatesan, A. Agarwal, K. Roy, and A. Raghu Nathan, "MACACO: Modeling and analysis of circuits for approximate computing" Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, Nov. 2011, pp. 667–673.

5. V. Gupta, D. Mohapatra, S. P. Park, A. Raghu Nathan, and K. Roy, "IMPACT: Imprecise adders for low—power in the approximate computing," in Proc. Int. Symp. Low Power Electron. Design, Aug. 2011, pp. 409–414.
6. C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Media Bench: A tool for evaluating and synthesizing multimedia and communications systems," in Proc. 13th Annu. IEEE/ACM Int. Symp. Micro architecture, Dec. 1997, pp. 330–335.
7. D. Zuras and W. H. McAllister, "Balanced delay trees and combinatorial division in VLSI," IEEE J. Solid-State Circuits, vol. 21, no. 5,pp. 814–819, Oct. 1986.