IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

OPTIMIZING FPGA LOGIC BLOCK ARCHITECTURE FOR ARITHMETIC OPERATORS

Dr. J. Rangarajan*1, Professor, A. komala*2 PG Student,
Dr.C.Selvi*3 Associate Professor, Anandakumar S*4, Assistant Professor,
Muthayammal Engineering College, Department of ECE

1.1 ABSTRACT

Hardened adder and carry logic is widely used in commercial FPGAs to improve the efficiency of arithmetic functions. There are many design choices and complexities associated with such hardening including: circuit design, FPGA architectural choices, and the CAD flow. However these choices have seen little study, and hence we explore a number of possibilities. We also highlight front-end elaboration optimizations that help ameliorate the restrictions placed on logic synthesis by hardened arithmetic. We show that hard adders and carry chains increase performance of simple adders by a factor of four or more, but on larger benchmark designs that contain arithmetic improve overall performance by 15% Approximate addition is a technique to trade off energy consumption and output quality in error-tolerant applications. a novel bit truncation approach is proposed to design dynamically energy-quality scalable adders with graceful degradation. As main idea, instead of being set to zero, the truncated input bits are set to the constant value that maximizes the output quality.

KEYWORDS: Field programmable gate arrays, Digital arithmetic, Logic design, Design automation

1.2 OBJECTIVE OF THE PROJECT

Approximation can increase performance or reduce power consumption with a simplified or inaccurate circuit in application contexts where strict requirements are relaxed. For applications related to human senses, approximate arithmetic can be used to generate sufficient results rather than absolutely accurate results. Approximate design exploits a tradeoff of accuracy in computation versus performance and power. However, required accuracy varies according to applications, and 100% accurate results are still required in some situations. In this paper, we propose an accuracy-configurable approximate (ACA) adder for which the accuracy of results is configurable during runtime. Because of its configurability, the ACA adder can adaptively operate in both approximate (inaccurate) mode and accurate mode. The proposed adder can achieve significant throughput improvement and total power reduction over conventional adder designs. It can be used in accuracy-configurable applications, and improves the achievable tradeoff between performance/power and quality. The ACA adder achieves approximately 30% power reduction versus the

conventional pipelined adder at the relaxed accuracy requirement.

1.3 INTRODUCTION

A key FPGA architecture question is which functions should be hardened and which should be left for implementation in the soft logic

[1]. Hardening a function makes an FPGA more efficient if the function occurs often in applications, and there is a large advantage when it is implemented in hard, rather than soft, logic. As adder-type arithmetic functions appear often and hardened adders are much faster than soft adders, commercial devices commonly have hardened adder and/or carry logic and routing [2]–[5].

There are many degrees of freedom in the electrical and architectural design of hard adder logic, and in the software used to map a complete application to such structures. While commercial devices use a wide variety of hardened adder circuits and architectures (indicating there is no general agreement on the best options), there has been little published work that explores the trade-offs of different hardening choices, or on the software flow used to map arithmetic to these structures. We study a number of these choices and determine their impact on the performance and area of both micro-benchmarks and complete designs. Some examples include: First, how should adders and LUTs interact? For instance, should there be fast (but less flexible) adder inputs, or are flexible (but slower) inputs coming from LUTs preferable? Second, what are the trade-offs in terms of performance and area between large, fast, multi-bit adders, and smaller, slower, but more flexible, single-bit adders? Third, should adjacent hard adder units use dedicated links for carry signals crossing soft logic block boundaries (which constrains the placement problem) or use the more flexible regular routing fabric? Fourth, how should hard adders be integrated with a fracturable LUT (a large LUT that can be split into two smaller LUTs)? Does this effect how many bits of arithmetic should be associated with
each LUT? These are important questions an architect must answer when embedding hard adders with soft logic, and we present quantitative measurements of the impact of each of these decisions.

Previous work in this area began in the early 1990's, when Hsieh et al. [6] described the Xilinx 4000 FPGA that had soft logic blocks that were capable of implementing two independent adder bits per block. They employed dedicated carry logic and routing from adjacent logic blocks for the carry signals. Woo [7] proposed adding additional flexibility to the fast carry links between logic blocks to enable flexible tree based mappings of addition /subtraction /comparison functions. Both Hseih and Woo targeted older FPGAs that had relatively fewer and smaller lookup tables in the logic block compared to the latest FPGAs.

CHAPTER-2

BASELINE ARCHITECTURES

2.1 Arithmetician is the generic term used to describe the operation of sending one or more analogue or digital signals over a common transmission line at different times or speeds and as such, the device we use to do just that is called a Arithmetic. In electronics, a Arithmetic (or mux; spelled sometimes as Arithmetic xor), also known as a data selector, is a device that selects between several analog or digital input signals and forwards the selected input to a single output line. The selection is directed a separate set of digital inputs known as select lines. A Arithmetic of 2^n inputs has n select lines, which are used to select which input line to send to the output.

A Arithmetic makes it possible for several input signals to share one device or resource, for example, one analog-to-digital converter for example, one analog-to-digit al converter or one communications transmission medium, instead of having one device per input signal.

Arithmetic's can also be used to implement Boolean functions of Arithmetic variables.

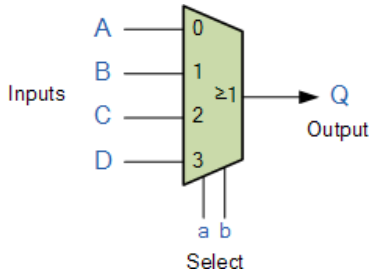

Fig: 2.1: Symbolic Diagram of Arithmetic

Conversely, a deArithmetic (or demux) is a device taking a single input and selecting signals of the output of the compatible mux, which is connected to the single input, and a shared selection line. A Arithmetic is often used with a complementary deArithmetic on the receiving end.

An electronic Arithmetic can be considered as a Arithmetic-input, single-output switch, and a deArithmetic as a single-input, Arithmetic-output switch. The schematic symbol for a Arithmetic is an isosceles trapezoid with the longer parallel side containing the input pins and the short parallel side containing the output pin. The schematic on the right shows a 2-to-1 Arithmetic on the left and an equivalent switch on the right. The sel wire connects the desired input to the output.

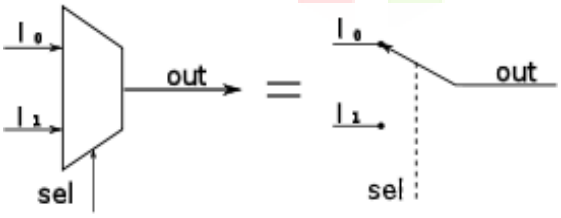

Fig: 2.2: Schematic Of A 2-To-1 Arithmetic. It Can Be Equated To A Controlled Switch

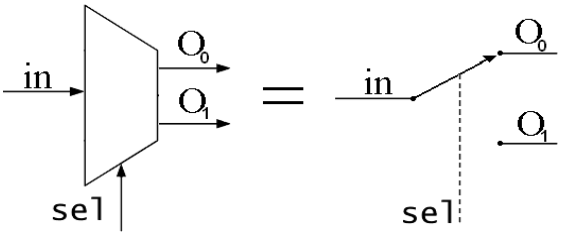

Fig: 2.3: Schematic Of A 1-To-2 DeArithmetic. Like An Arithmetic, It Can Be Equated To A Controlled Switch.

2.2 DIGITAL ARITHMETIC

In digital circuit design, the selector wires are of digital value. In the case of a 2-to-1 Arithmetic, a logic value of 0 would connect to the output while a logic value of 1 would connect to the output. In larger Arithmetic's, the number of selector pins is equal to where is the number of inputs.

For example, 9 to 16 inputs would require no fewer than 4 selector pins and 17 to 32 inputs would require no fewer than 5 selector pins. The binary value expressed on these selector pins determines the selected input pin.

2.3 2*1 ARITHMETIC

A 2-to-1Arithmetic has a boolean equation where and are the two inputs, is the selector input, and is the output:

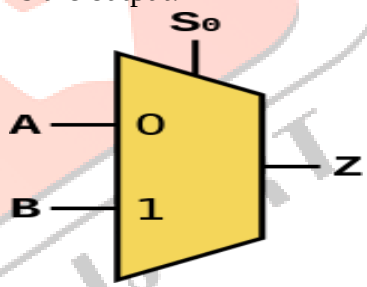

Fig: 2.4:2*1 Arithmetic Pin Diagram

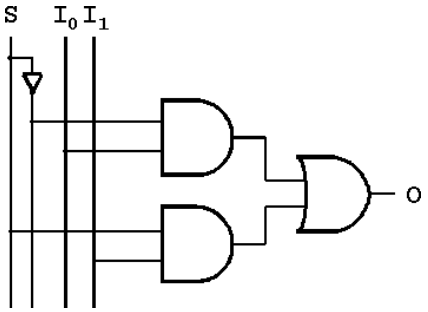

Fig: 2.5:2*1 Arithmetic Circuit Diagram

2.4 4×1 ARITHMETIC

2 x 1 mux need 4 AND and 1 OR and 2 NOT gates. Replace OR gate with Invert-input NAND , then above circuit will be replace with

2 NOT and 5 NAND , then 2 NANDS for 2 NOT's , then total 7 NAND gates required.

Output of 4*1mux is A'B' + A'B + AB' + AB. we can simplify this and we will get 1.
So i think we required 0 NAND gate.
A'B + AB' + AB A'(B+B) + A(B'+B) = A' + A =1

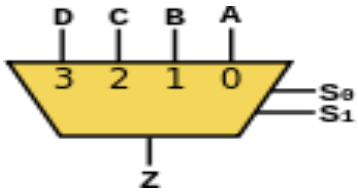

Fig: 2.6:4*1 Arithmetic Pin Diagram

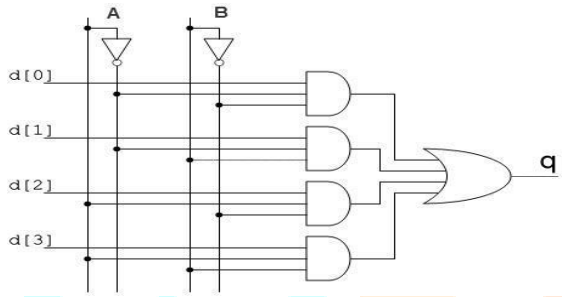

Fig: 2.7:4*1 Arithmetic Circuit Diagram

Select Data Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Table: 2.1: The Truth Table Of 4*1 Arithmetic

From the above truth table, we can write the output expressions as

If S1=0 and S0=0 then Y = D0 Therefore, Y = D0 (S1)¯(S0)

If S1= 0 and S0=1, the Y = D1 Therefore, Y = D1 (S1)¯S0

If S1=1 and S0=0, then Y = D2 Therefore, Y = D2 S1 (S0)

If S1=1 and S0=1 the Y = D3 Therefore, Y = D3 S1 S0

To get the total data output from the Arithmetic, all these product terms are to be summed and then the final Boolean expression of this Arithmetic is given as

$$Y = D0\,\overline{S1}\,\overline{S0} + D1\,\overline{S1}\,S0 + D2\,S1\,\overline{S0} + D3\,S1\,S0$$

2.5 8*1 ARITHMETIC

An 8-to-1 Arithmetic consists of eight data inputs D0 through D7, three input select lines S2 through S0 and a single output line Y. Depending on the select lines combinations, Arithmetic decodes the inputs.

The below figure shows the block diagram of an 8-to-1 Arithmetic with enable input that enable or disable the Arithmetic. Since the number data bits given to the MUX are eight then 3 bits (23=8) are needed to select one of the eight data bits

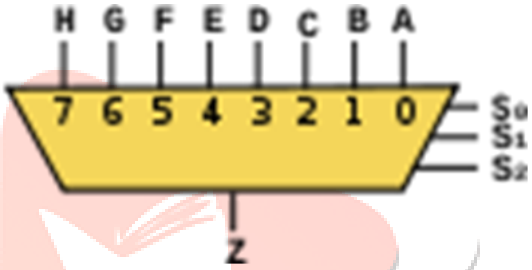

Fig: 2.8:8*1 Arithmetic

CHAPTER-3

THEORETICAL BACKGROUND

3.1 EXISTING METHODOLOGY

- Our proposed project focuses to reduce the amount of power that is exists in the existing system
- The number of transistors used in designing of the existing system can be reduced in the proposed work
- The proposed system is more efficient in terms of power consumption

3.2 AREA OF THE PROPOSED WORK AND APPLICATION

A Arithmetic is used to increase the efficiency of the communication system by allowing the transmission of data such as audio & video data from different channels via cables and single lines. Logic gates perform basic logical

functions and are the fundamental building blocks of digital integrated circuits. Most logic gates take an input of two binary values, and output a single value of a 1 or 0.

Full Adder is the adder which adds three inputs and produces two outputs. ...
A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another.

3.3 EXSITING SYSTEM

Full adder is the basic block of arithmetic circuit found in microcontroller and microprocessor inside arithmetic and logic unit (ALU). Improvement of the adder is thus essential for upgrade the performance of those circuits where adder is employed. Full adders, till now, have been designed using wide range of structures for improvement of various parameters like power consumption, speed performance and structure size.

Full adder is a combinational circuit that performs the addition operation of 3 input bits. It basically consists three inputs and two outputs. The input variables are expressed by A, B and Cin. The two output variables are expressed by sum (S) and

carry (Cout) [4]. Fig. 1 shows the essential block diagram of full adder cell. The Boolean expression for full adder operation is defined below:

$$SUM = ((A \oplus B) \oplus Cin)$$
$$Cout = AB + (A \oplus B)\, Cin$$

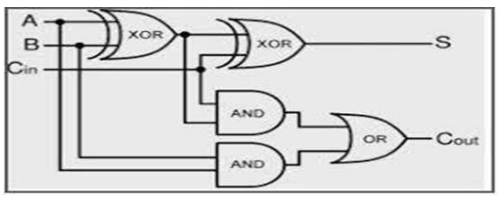

Fig: 3.1: Full Adder Circuit Diagram

CHAPTER 4

4.1 PROPOSED SYSTEM:

Adders are one of the key components in arithmetic circuits. Enhancing their performance can significantly improve the quality of arithmetic designs. This is the reason why the theoretical lower bounds on the delay and area of an adder have been analysed, and circuits with performance close to these bounds have been designed. In this paper, we present a novel adder design that is exponentially faster than traditional adders; however, it produces incorrect results, deterministically, for a very small fraction of input combinations. We have also constructed a reliable version of this adder that can detect and correct mistakes when they occur. This creates the possibility of a variable-latency adder that produces a correct result very fast with extremely high probability; however, in some rare cases when an error is detected, the correction term must be applied and the correct result is produced after some time. Since errors occur with extremely low probability, this new type of adder is significantly faster than state-of-the-art adders when the overall latency is averaged over many additions.

4.2 PROPOSED ARITHMETICFULL ADDER

There are various low power technique were present nowadays in which GDI is also one of them. Nowadays a GDI is the most commonly used technique for the reduction of power as it reduces the switching of output. Also a GDI technique is used to reduce the number of transistor for the reduction in area of chip. The most commonly cell in a GDI technique is an inverter cell which is formed by the combination of a single PMOS and NMOS. Here a single input is provided to the gate terminal of both and then output is taken out. As shows in the basic 4×1 Arithmetic formed by the help of GDI technique

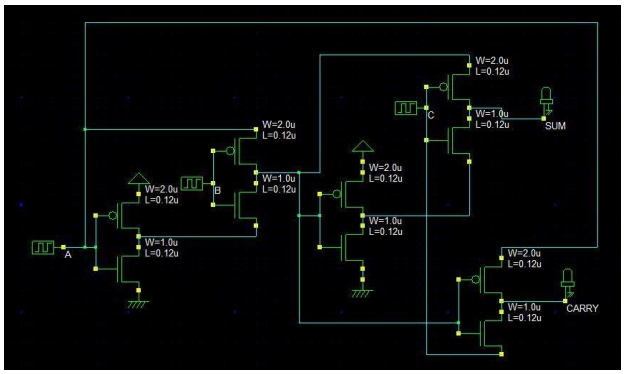

Fig: 4.3: Gdi Based ARITHMETIC Full adder

This technique allows reducing power consumption, delay and area of digital circuits, while maintaining low complexity of logic design. Performance comparison with traditional CMOS techniques is presented, with respect to the, delay and power dissipation, showing advantages of GDI as compared to other methods. A variety of logic gates have been implemented in 130 nm technology to compare the GDI technique with CMOS. This modified 4×1 Arithmetic using GDI technique is used for the 4×1 Arithmetic. Now, the modified 4×1 Arithmetic.

SOFTWARE REQUIREMENTS

Xilinx ISE-10.1 (Integrated Synthesis Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device.

CHAPTER 5

PROJECT DESIGN

5.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

The usage of optimization algorithms in order to ensure the correct symmetry of this type of circuits and, in parallel, step up the advantages in terms of gain, speed and occupied active-area is completely justified.

Furthermore, common performance trade-offs in the design of single-stage amplifiers involve: high gain as opposed to high bandwidth, high gain as opposed to high energy- efficiency, high gain as opposed to reduced noise impact, reduced current consumption as opposed to high gain-bandwidth product, reduced current consumption as opposed to reduced noise impact, and high gain as opposed to reduced active-area of complete circuitry.

5.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to besplaced for immediate need and also the hard copy output.

It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER-6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product.

It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

- **UNIT TESTING**
- **INTEGRATION TESTING**
- **FUNCTIONAL TESTING**
- **SYSTEM TESTING**

CHAPTER 7

CONCLUSION AND FUTERU SCOPE

The existing and proposed architecture are simulated by using H-spice simulator. Output of the existing USR and proposed USR are analysed from the results and it is concluded that the proposed architecture reduces the delay compared with the existing system. The existing and proposed design are simulated in synopsis H-spice tool.

The design has been evaluated in 130 nanometer technology. The spice code of the schematic design can be viewed in edit NL in H-spice tool. It is simulated. If there is any error, it shows job aborted which then can be even modified. The avanwaves in H-spice tool shows the simulated output waveforms

7.1 CONVENTIONAL FULL ADDER

Fig: 7.1: Graph of Conventional Full Adder

From here we got.

Avg power	peak power	temper	alter#
1.276e-04	1.949e-02	25.0000	1.0000

7.2 PROPOSED ARTHMETIC ADDER

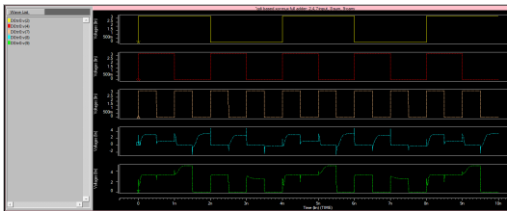

Fig:7.2: GDI Based ARTHMETIC Full Adder

From here we got,

avgpower	peakpower	temper	alter#
4.049e-03	2.277e-05	25.0000	1.0000

FIG: 7.3: PROPOSED ARTHMETIC

S.No.		Avg Power	Peak Power	Temper
1	CONVENTIONAL FULL ADDER	1.276e-04	1.949e-02	25.0000
2	GDI BASED ARITHMETICFULLADDER	2.277e-05	4.049e-03	25.0000
3	ARITHMETICFULLADDER	1.327e-04	2.705e-02	25.0000

Table: 7.1: Average power, Peak Power, Temper

7.3 CONCLUSION

We have proposed a new efficient Universal Shift Register using low power XOR-MUX Full adder based computational block for low power architecture and modified 4×1 Arithmetic, which gives better response in terms of speed in comparison to the conventional Universal Shift Register. We have achieved a design of a proposed Universal Shift Register which is faster and has reduced the delay of the conventional USR considerably. Since our objective was to reduce the delay of the USR and it has been reduced, hence we achieved our objective. So, we practically saw that the proposed USR with modified Arithmetic (using gdi technology) is much more efficient.

7.4 FUTURE SCOPE

Shift register can be used for the storage or the transfer of binary data. So, improving its performance can improve the overall efficiency of the system. Future work can be concentrated on area efficiency of the Logic gates and Arithmetic and can obtain better efficiency in terms of area. This proposed universal shift register can be implemented and can obtain better efficiency.

REFERENCES

1. Chippa.V.K, Chakradhar.S.T, Roy.K, and Raghunathan.A, “Analysis and characterization of inherent application resilience for approximate computing,” in Proc. 50th Annu. Design Autom. Conf. (DAC), 2013, pp. 1–9.

2. De Caro.D, Petra.N, StrolloA.G.M, Tessitore.F, and Napoli.E, “Fixed-width multipliers and multipliers-accumulators with min-max approximation error,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 60, no. 9, pp. 2375–2388, Sep. 2013.

3. Developer Reference for Intel Integrated Performance Primitives— FilterSobel. Accessed: Jan. 13, 2020. [Online]. Available: https://software.intel.com/en-us/ipp-dev-reference-filtersobel.

4. Esposito.D, De Caro.D, and Strollo.A.G.M, “Variable latency speculative parallel prefix adders for unsigned and signed operands,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 63, no. 8, pp. 1200–1209, Aug. 2016.

5. Esposito.D, De Caro.D, De Martino.M, and StrolloA.G.M, “Variable latency speculative Han–Carlson adders topologies,” in Proc. 11th PRIME Conf., Jun. 2015, pp. 45–48.

6. Gillani.G.A, Hanif.M.A, Verstoep.B, Gerez.S.H, Shafique.M, and Kokkeler.A.B.J, “MACISH: Designing approximate MAC accelerators with internal-self-healing,” IEEE Access, vol. 7, pp. 77142–77160, 2019.

7. Gupta.V, Mohapatra.D, Raghunathan.A, and Roy.K, “Low-power digital signal processing using approximate adders,” IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013