Low Powered Self-Testable ALU

M. Nagharjun¹, V. Ravi¹
¹School of Electronics Engineering(SENSE), Vellore Institute of Technology, Chennai

Email: nagharjun.m2018@vitstudent.ac.in

Abstract-The Arithmetic and Logic Unit(ALU) is a Significant Segment of a CPU as it does all the arithmetic and logical operations and computations required by all the processes running inside the CPU. Being such an important part inside the processor, it requires more power and thus it would be efficient and feasible if it can run on low power and also can test for any faults in the circuit itself. This paper completely focuses on building a Low Powered ALU with the implementation of Built-In Self-Test(BIST) mechanism for efficient arithmetic and logical operations. In the design, power-efficient circuits such as Wallace Tree Multiplier and Carry Look-Ahead Adder is being used. The created design is synthesized and simulated on the Xilinx Vivado software tool and implemented on Digilent Xilinx Basys 3 Artix-7 (XC7A35T-1CPG236C) Field Programmable Gate Array (FPGA). The overall power consumption of the design is 75mW and the ALU has self-testability function.

Keywords: Arithmetic and Logic Unit, Xilinx, Wallace Tree Multiplier, Carry Look-Ahead Adder, Built-In Self-Test, Field Programmable Gate Array.

1. Introduction

As day by day, more and more transistors are embedded in the Integrated Circuits, and power dissipation plays an important role in any type of circuit produced. Looking into the Arithmetic and Logical Unit which is a fundamental part of the Central Processing Unit, needs to be optimized to dissipate lower power. The Field Programmable Gate Arrays(FPGA’s) are reprogrammable, flexible and versatile hardware device which can be programmed to any digital circuit using Hardware Description Languages like Verilog or VHDL and has many applications. Due to these advantages, the Arithmetic and Logic Unit has been programmed and implemented through FPGA. The main function of the ALU is to perform bit-wise Arithmetic and Logical operations assigned to the ALU by the processor. In the Fetch – Decode – Execute process of the CPU, the ALU performs the Execute process i.e., the required computations for many different processes running in the device.

2. Problem Statement

As ALU is one of the main part of the processes done by the CPU, it faces many issues. The main concern is Power Dissipation and Consumption. It hinders the efficiency of the circuit. Whenever a circuit is needed to be efficient, it requires higher performance which automatically leads to higher power being consumed and dissipated because of the higher number of many different complex
operations to be done in a short amount of time. The amount of power dissipating is directly proportional to the amount of heat released. Every similar transistor has a same Junction Temperature value where, if the temperature due to the heat released by the transistors crosses the Junction Temperature, the transistor may get damaged and can cause circuit failure. Due to this, the ability of the CPU to reach higher clock rates has been restricted. The other major problem is Area Consumption of the circuit. As the modern circuits needs new and upgraded functionalities, the density of transistors in the circuit increases. So efficient use of area is required. Another problem is Faults present in the circuit which leads to the production of wrong outputs and computations. This may also be a case due to higher power being dissipated and due to transistor aging which disrupts the logic of the circuit.

3. Literature Survey

Many different types of methods have been proposed to address this problem statement and provided, have been efficient. Some of the effective methods have been discussed.

Jamuna S et al., [1] proposed the BIST logic design for ALU on FPGA. This mainly focused of the detection of faults in the circuit by using BIST architecture with 94.2% code coverage. T Esther Rani et al., [2] proposed a method which focusses on reducing the power of the ALU by 70% compared to traditional design by using low power 1-bit full adders with 10 transistors. Typically, a 1-bit full adder requires a total of 16 transistors which has been reduced here to 10 to get an effective and efficient result. AnkitMitra et al., [3] proposed the Implementation and Design of Low Power 16-bit ALU with Clock Gating Technique where, whenever the circuit is not in use, the clock signals are frozen and power is saved. N Sureka et al., [4] proposed An Efficient High Speed Wallace Tree Multiplier. The focuses on building a low powered and efficient Wallace tree multiplier by using Carry Save Adder to reduce the latency which is 44.4% faster than the traditional Wallace tree multiplier.

4. Methodology

As discussed in the previous topics, the ALU being such a major part inside the processor consumes and dissipates high power. Power Consumption and Dissipation can be reduced if we reduce the delay in the circuit as it performs operations in a lesser amount of time which decreases the overall power consumption. One of the delays that can be reduced is the propagation delay. The circuits for these algorithms are complex but it is power efficient. Now as the ALU is power efficient, another concern is the generation of faulty outputs. This is most commonly due to Stuck at Faults where a part of the circuit is stuck at either 0 or 1. Taking this problem into consideration, a Built-In Self-Test mechanism is being introduced where the circuit tests itself for any faults.

![Figure 1: Total system Architecture](image-url)
The final architecture of the system is shown in the Figure 1. The TEST CONTROLLER is the main block which holds all the sub blocks together. The MULTIPLEXER TO SELECT BETWEEN NORMAL OR TEST block is used to select between a Normal ALU and Testing ALU. In the case of Normal ALU, the input is directly fed into the system. In the case of Testing ALU, the input is fed by the LFST TEST PATTERN GENERATOR block which consists of one or two LFSR’s depending on the type of operation. Next comes the ALU block where the main arithmetic or logical operation takes place. In the case of a Normal ALU, the output is shown without undergoing any further actions. In the case of Testable ALU, the output is fed to the OUTPUT RESPONSE block which is the MISR and it is compared against the GOLDEN SIGNATURE stored inside the READ ONLY MEMORY block which contains the correct outputs, in the COMPARATOR block. The output of the comparator block checks whether the output is FAULTY OR NOT FAULTY.

4.1. Arithmetic and Logic Unit
The ALU deals with both Arithmetic and Logical operations. The Arithmetic part performs Addition, Subtraction, Multiplication, Division, Increment, and Decrement. The Logical part performs Logic Gate related operations and Shifting of bits. Each and every Arithmetic and Logical operation has a fixed opcode and an operation can be called upon by inputting the opcode for the required operation. The CPU fetches the instruction from memory and supplies it to the Instruction Register which decodes the instruction into opcodes and operands. According to the opcodes, the operation is performed using operands.

Table 1 : Opcodes for ALU Operation

| OPCODES | ALU OPERATION          |
|---------|-----------------------|
| 0000    | A + B                 |
| 0001    | A - B                 |
| 0010    | A * B                 |
| 0011    | A / B                 |
| 0100    | A + 1                 |
| 0101    | A - 1                 |
| 0110    | A AND B               |
| 0111    | A OR B                |
| 1000    | A XOR B               |
| 1001    | NOT A                 |
| 1010    | A NAND B              |
| 1011    | A NOR B               |
| 1100    | SHIFT RIGHT           |
| 1101    | SHIFT LEFT            |
| 1110    | GREATER COMPARE       |
| 1111    | EQUAL COMPARE         |
Table 1 shows the various Arithmetic and Logical operations performed by the ALU and its control bit for calling the operations. For example, if 1011 is the OPCODE, then the function of the ALU is to perform the NOR operation between the given two operands and if 0000 is selected, the operation of the ALU is to perform addition(+) operation between the given two inputs.

4.1.1. Carry Look-Ahead Adder.
The carry look-ahead adder is also called a fast adder. Unlike the conventional ripple carry adder, the carry look-ahead adder reduces the propagation delay which is caused due to the waiting time for the ith full adder to get the i-1th full adder carry, by using a complex architecture. The carry look-ahead adder has 2 signals, Carry Propagate (P) and Carry Generate (G). Each and every bit of the sum and carry has been calculated in parallel using these given equations whereas in the conventional adder, each block has to wait until the carry of the previous block to arrive. Compared to other conventional and frequently used adders, the Carry Look Ahead Adder consumes both less power and low area[9].

\begin{align*}
\text{Carry Propagate} & : P_i = A_i \oplus B_i \\
\text{Carry Generate} & : G_i = A_i \land B_i \\
\text{Sum} & : S_i = P_i \oplus C_{i-1} \\
\text{Carry} & : C_{i+1} = G_i \lor (P_i \land C_i)
\end{align*}

The Sum and Carry are calculated using the above equations.

![Figure 2: Carry Look-ahead Adder Architecture](image)

Figure 2 shows the circuit of carry look-ahead adder. This architecture consists of two 4-bit Carry Look-Ahead Adders, one 2-bit Carry Look-Ahead Adder and blocks to calculate Carry Generate(G_i) and Carry Propagate(P_i) using the equations given above.
4.1.2. Wallace Tree Multiplier

It is a hardware implementation of an effective and efficient digital circuit which is used for multiplication. This circuit also reduces the propagation delay to a great extent. As each and every bit of both of the operands are multiplied, partial products are being generated. The output is produced by reducing the partial products using half and full adders in parallel until we get a 2-layered partial product, and finally either using half adders or carry look-ahead adder to produce the final output. This architecture is used due to its ability to perform operations in low power and also has faster-switching speed. Compared with the conventional array multiplier, the order of the delay comes down from \( O(n) \) to \( O(\log n) \) in the Wallace tree multiplier. Comparing with widely used other multipliers like Booth Multiplier, Array Multiplier, the Wallace Tree Multiplier consumes less power and low area[10].

![Figure 3: Steps for Wallace Tree Multiplier](image)

The figure 3 shows the operation of an 8-bit Wallace tree multiplier. Each of the Squares represent the bits. The Black squares represent the bits that are static and does not need to undergo any operation. The Grey squares represent the bits which are going to undergo either Full Adder or Half Adder operation. At Step 1, AND operation is performed between each and every bits of the two 8-bit numbers. At Step 2, Step3, Step 4 and Step 5, Full Adder operations and Half Adder operations are performed simultaneously and sequentially. The final 16-bit output is being generated.

4.2. Built In Self Test

BIST is a self-testing mechanism for digital IC’s where random test vectors generated by Linear Feedback Shift Register(LFSR) are applied to the circuit under test under the control of the BIST controller. The output responses are compacted by the signature analyzer which is the MISR in the OUTPUT RESPONSE to produce a Signature Response Value which is further compared with the Golden Signature Value which is stored inside the ROM or some memory. If the Signature Response Value and Golden Signature are the same, then the output is not faulty. If the Signature Response Value and Golden Signature are different, the output is faulty. In this way, we can detect any faults present internally in the circuit.
4.2.1. **LFSR**
This hardware circuit is based on a shift register with feedback. This circuit generates pseudo-random patterns for testing the circuit. To produce 8-bit patterns, we use 8 D-Flipflops. The points where the feedback are taken are called taps and it affects the generation of pseudo random patterns.

The above diagram depicts the architecture of the Many to One Linear Feedback Shift Register used. The taps are made at 1\textsuperscript{st}, 2\textsuperscript{nd}, 3\textsuperscript{rd} and 7\textsuperscript{th} points to produce a 8-bit pseudo random number.

4.2.2. **Output Response or MISR**
We cannot compare all of the outputs produced by the ALU, so we need to compact the outputs. Multiple Input Signature Register(MISR) is used to compact the outputs produced by the ALU. MISR is a hybrid of LFSR which is used for the compaction of the output responses of the ALU to produce a the Syndrome also called as the Signature Response which is compared with the Golden Signature.

In figure 5, the LFSR produces many pseudo random patterns and supplies to the ALU. The ALU produces many outputs which are supplied as inputs to the OUTPUT RESPONSE which compacts the multiple inputs and produces a single Signature Response value.
5. Implementation and Result

The whole implementation of the design is done in Basys 3 Artix 7 FPGA board. The complete RTL schematic and architecture of the design is given.

Figure 6: RTL Design

Figure 6 shows the total view of the RTL architecture of the design.

Figure 7: RTL Design Simplified

The RTL design in Figure 7 has 2 LFSR’s, 2 MISR’s, 2 Multiplexers, 2 registers x and y to store the values of A and B if the ALU functions normally or the test patterns in the case of testing, 1 ALU block which has Wallace Tree Multiplier and Carry Look Ahead Adder, memory element with the signature response value and a comparator to check whether the circuit is faulty. When the NORMAL_TESTABLE_SEL line has logic 1, then the testing part will be done and when NORMAL_TESTABLE_SEL line has logic 0, then the normal ALU will function.

5.1. Power

The overall power consumed by the design is 75mW. It consists of 72mW in Static Power and 3mW in Dynamic Power.

| TOTAL ON-CHIP POWER |  |
|---------------------|--|
| Static Power (96%)  | 0.072W |
| Dynamic Power (4%)  | 0.001W |

Table 2: Overall Power Consumption
5.2. Area
The total area occupied by the design is shown below.

![Overall Area Consumption](image)

| LUT’s | FLIP FLOPS |
|-------|------------|
| 286   | 120        |

Table 3: Overall Area Consumption

6. Simulation and Output
The entire simulation of the proposed design is done in Vivado. There are in total 16 different arithmetic and logical operations that can be simulated and here 5 of the important and frequently used simulations are done. Along with this, the Self-Testable ability of the ALU is also shown through the simulation.

6.1. Normal ALU Mode

Multiplication operation is performed where,
NORMAL_TESTABLE_SEL = 0
indicating the ALU is in normal mode
INPUT_A = 60, INPUT_B = 80 indicating the operands
ALU_SELECTLINE_S = 0010 indicating the opcode for multiplication operation
OUTPUT_R = 4800 indicating the output of multiplication

Addition operation is performed where,
NORMAL_TESTABLE_SEL = 0
indicating the ALU is in normal mode
INPUT_A = 60, INPUT_B = 80 indicating the operands
ALU_SELECTLINE_S = 0000 indicating the opcode for addition operation
OUTPUT_R = 140 indicating the output of addition

![Multiplication Simulation](image)

![Addition Simulation](image)
OR Gate operation is performed where,
NORMAL_TESTABLE_SEL = 0
indicating the ALU is in normal mode
INPUT_A = 00111100,
INPUT_B = 01010000 indicating the operands
ALU_SELECTLINE_S = 0111 indicating the opcode for OR Gate operation
OUTPUT_R = 0000000001111100 indicating the output of OR Gate

Greater than operation is performed where,
NORMAL_TESTABLE_SEL = 0
indicating the ALU is in normal mode
INPUT_A = 80, INPUT_B = 60 indicating the operands
ALU_SELECTLINE_S = 1110 indicating the opcode for Greater than operation
As 80 > 60, OUTPUT_R = 1 indicating the output of Greater than

Shift Left operation is performed where,
NORMAL_TESTABLE_SEL = 0
indicating the ALU is in normal mode
INPUT_A = 00111100 indicating the operand
ALU_SELECTLINE_S = 1101 indicating the opcode for Shift Left operation
OUTPUT_R = 0000000001111000 indicating the output of Shift Left
6.2. Testable ALU Mode
Here NAND gate is used for testing whose opcode is 1010.

6.2.1. Non-Faulty
The NAND of the two of the LFSR generated pseudo random patterns are compacted and the signature response value produced is SIGNATURE_OUTPUT and ROM_OUTPUT is the memory containing the Golden Signature. Both the SIGNATURE_OUTPUT and ROM_OUTPUT are compared and as both are same, the fault free output is logic 1.

6.2.2. Faulty
The NAND of the two of the LFSR generated pseudo random patterns are compacted and the signature response value produced is SIGNATURE_OUTPUT and ROM_OUTPUT is the memory containing the Golden Signature. Both the SIGNATURE_OUTPUT and ROM_OUTPUT are compared and as both are not the same, the fault free output is logic 0.
7. Result and Performance Analysis

The total power consumed by the ALU is 75mW with 72mW in Static power and 3mW in Dynamic power.

In comparison with the paper Jamuna S et al.[1] which uses BIST Mechanism, consumes 1.061W, there has been a significant reduction in the power consumption in the proposed design with 75mW.

In comparison with the work by R Dhanabal et al.,[5], the combined power consumption for all of the ALU operations is 229.7mW, and the proposed design consumes much less power.

Now comparing the Dynamic Power which is 3mW with self-testable functionality in the proposed design with the work of B Pandey et al.,[8] which proposes Clock Gating technique to reduce power, consumes 3mW in Dynamic Power with clock gate on 100MHz.

| DESIGN                     | STATIC POWER | DYNAMIC POWER | TOTAL POWER |
|----------------------------|--------------|---------------|-------------|
| Jamuna S et al., [1]       | -            | -             | 75mW        |
| Dhanabal R et al., [5]     | -            | -             | 229.7mW     |
| B Pandey et al., [8]       | -            | 3mW           | -           |
| Proposed Design            | 72mW         | 3mW           | 75mW        |

8. Conclusion and Future Scope

It is very much required that the design of the ALU should be low powered. In this work, a low powered, fast, and testable ALU has been implemented. The ALU can be used for normal purposes and can detect if any fault is present or not. The design was targeted and implemented on Basys 3 Artix 7 FPGA and the overall power consumption is found out to be 75mW. The design has been simulated and is giving positive results.

One of the improvements which can be done is implementing the pipelining strategy to get more efficient results with less delay. Deep Learning algorithms can be implemented to improve the performance of the design by learning the frequently used operations.

References

[1] J. S, D. P, K. Shashikala and K. K. K, "Design and Implementation of BIST logic for ALU on FPGA," 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, 2018, pp. 1-5. Another reference

[2] T. E. Rani, M. A. Rani and R. Rao, "AREA optimized low power arithmetic and logic unit," 2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, 2011, pp. 224-228.

[3] AnkitMitra., “Design and implementation of low power 16 bit ALU with clock gating”
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 6, June 2013, ISSN: 2278 – 1323

[4] N. Sureka, R. Porselvi and K. Kumuthapriya, "An efficient high speed Wallace tree multiplier," 2013 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, 2013, pp. 1023-1026.

[5] R. Dhanabal & Bharathi, V. & Salim, Saira & Thomas, Bincy & Soman, Hyma & Sahoo, Sarat Kumar. (2013). Design of 16-bit low power ALU-DBGPU. Intern. J. Eng. Technol.. 5. 2172-2180.

[6] Kishore, P., Sridevi, P.V., Babulu, K., & Chandra, K. (2015). A Novel Low power and Area Efficient Carry-Lookahead Adder Using MOD-GDI Technique.

[7] T. Thubrikar, S. Kakde, S. Gaidhani, S. Kamble and N. Shah, "Design and implementation of low power test pattern generator using low transitions LFSR," 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2017, pp. 0467-0471.

[8] B. Pandey, J. Yadav, M. Pattanaik and N. Rajoria, "Clock gating based energy efficient ALU design and implementation on FPGA," 2013 International Conference on Energy Efficient Technologies for Sustainability, Nagercoil, 2013, pp. 93-97, doi: 10.1109/ICEETS.2013.6533362.

[9] A. K. Panigrahi, S. Patra, M. Agrawal and S. Satapathy, "Design and Implementation of a high speed 4bit ALU using BASYS3 FPGA Board," 2019 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 2019, pp. 1-6.

[10] B. Harish, K. Sivani and M. S. S. Rukmini, "Design and Performance Comparison among Various types of Adder Topologies," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 725-730, doi: 10.1109/ICCMC.2019.8819779

[11] U. Mandal, R. Banerjee and R. Mishra, "A Comparative Study of Arithmetic Multipliers," 2018 2nd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, 2018, pp. 1-5, doi: 10.1109/IEMENTECH.2018.8465371.

[12] B. Lamba and A. Sharma, "A review paper on different multipliers based on their different performance parameters," 2018 2nd International Conference on Inventive Systems and Control (ICISIC), Coimbatore, 2018, pp. 324-327, doi: 10.1109/ICISIC.2018.8399088.

[13] S. Abraham, S. Kaur and S. Singh, "Study of various high speed multipliers," 2015 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, 2015, pp. 1-5, doi: 10.1109/ICCCI.2015.7218139.

[14] B. Koyada, N. Meghana, M. O. Jaleel and P. R. Jeripotula, "A comparative study on adders," 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2017, pp. 2226-2230, doi: 10.1109/WiSPNET.2017.8300155.

[15] J. Saini, S. Agarwal and A. Kansal, "Performance, analysis and comparison of digital adders," 2015 International Conference on Advances in Computer Engineering and Applications,
Ghaziabad, 2015, pp. 80-83, doi: 10.1109/ICACEA.2015.7164650.

[16] K. B. Jaiswal, Nithish Kumar V, P. Seshadri and Lakshminarayanan G, "Low power wallace tree multiplier using modified full adder," 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, 2015, pp. 1-4, doi: 10.1109/ICSCN.2015.7219880.

[17] Shahzad Asif, Yinan Kong, "Low-Area Wallace Multiplier", VLSI Design, vol. 2014, Article ID 343960, 6 pages, 2014. https://doi.org/10.1155/2014/343960

[18] N. Ravindran and R. M. Lourde, "An optimum VLSI design of a 16-BIT ALU," 2015 International Conference on Information and Communication Technology Research (ICTRC), Abu Dhabi, 2015, pp. 52-55, doi: 10.1109/ICTRC.2015.7156419.

[19] A. Deeptha, D. Muthanna, M. Dhrithi, M. Pratiksha and B. S. Kariyappa, "Design and optimization of 8 bit ALU using reversible logic," 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2016, pp. 1632-1636, doi: 10.1109/RTEICT.2016.7808109.

[20] Dhanumjaya K., Kiran Kumar G., Giriprasad M.N., Raja Reddy M. (2010) Design and Modeling of Power Efficient, High Performance 32-bit ALU through Advanced HDL Synthesis. In: Das V.V., Vijaykumar R. (eds) Information and Communication Technologies. ICT 2010. Communications in Computer and Information Science, vol 101. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15766-0_3

[21] Shinde J.R., Sharma S., Dash L. (2020) An Optimization Design Approach for Arithmetic Logic Unit. In: Gunjan V., Garcia Diaz V., Cardona M., Solanki V., Sunitha K. (eds) ICICCT 2019 – System Reliability, Quality Control, Safety, Maintenance and Management. ICICCT 2019. Springer, Singapore. https://doi.org/10.1007/978-981-13-8461-5_81

[22] Shylashree, N., Venkatesh, B., Saurab, T.M. et al. Design and analysis of high-speed 8-bit ALU using 18 nm FinFET technology. Microsyst Technol 25, 2349–2359 (2019). https://doi.org/10.1007/s00542-018-4112-y

[23] Gary K. Yeap., “Practical Low Power Digital VLSI Design” 1998 Springer US Kluwer Academic Publishers

[24] Computer Architecture. A Quantitative Approach. Fourth Edition. John L. Hennessy. Stanford University. David A. Patterson. University of California at Berkeley.

[25] Modern Digital Electronics · R P Jain. Tata McGraw-Hill Education, Jun 1, 2003