Dandao Xuebao/Journal of Ballistics
ISSN: 1004-499X
Vol. 38 No. 1 (2026)

FPGA Implementation and Comparative Analysis of 16-bit ALU Architectures Using Ripple Carry, Carry Look-Ahead, and Carry Select Adders

Yogendra Singh[1*], Dr. ChandraBhan[2]
[1*]Research Scholar,
Shambhunath Institute of Engineering & Technology, Prayagraj.
Email: Yogendrasingh945314@gmail.com.
[2]Associate professor,
Shambhunath Institute of Engineering & Technology, Prayagraj.
Email: cbjha1965@gmail.com.

Corresponding Author: Yogendra Singh, Research Scholar, Shambhunath Institute of Engineering & Technology, Prayagraj. Email: Yogendrasingh945314@gmail.com.

Abstract

The Arithmetic Logic Unit (ALU) is one of the most important units of the digital systems of the modern era, which performs arithmetic and logical operations. The efficiency of an inner adder architecture of an ALU has a significant effect on the performance of the latter. This paper is the design, FPGA implementation and comparative analysis of three Ripple Carry Adder (RCA) based 16-bit ALU architecture, Carry Look-Ahead Adder (CLA) based architecture, and Carry Select Adder (CSLA) architecture.

The suggested designs are programmed in Verilog HDL and synthesized on a Xilinx Kintex-7 FPGA (xc7k70tfbv676-1) with Vivado 2023.1. In addition, a hybrid CLA-CSLA architecture is implemented to improve computational speed, and operand isolation is applied to reduce switching activity and enhance power efficiency.

They have been experimentally demonstrated to be the smallest-area, consuming 73 LUTs, compared to 109 and 111 LUTs used by CSLA and CLA structures respectively. Nonetheless, Although the Simple ALU shows higher timing slack due to FPGA carry-chain optimization, CLA and CSLA architectures provide improved carry computation efficiency and better scalability for high-speed arithmetic operations. Power analysis shows that they all work at below 0.1 W power, with the CLA-based ALU having the lowest power consumption.

The paper discusses various trade-offs between area, speed, and power, and shows that optimized adder architectures can be used to achieve a high improved performance of the ALU in an FPGA-led system.

Keywords: FPGA; Arithmetic Logic Unit (ALU); Ripple Carry Adder (RCA); Carry Look-Ahead Adder (CLA); Carry Select Adder (CSLA); Verilog HDL; Vivado

1. INTRODUCTION

Arithmetic Logic Unit (ALUs) are crucial components associated with digital processors and computing systems and are used to implement fundamental arithmetic and logical functions, including addition, subtraction, and bitwise operations. The ALU performance directly affects the overall performance of microprocessors, digital signal processors, and embedded systems. As the use of digital systems is growing with the need to compute at high speed as well as to consume less energy, optimization of the ALU architecture has taken center stage in the design of the digital system (Gurjar et al., 2011).

Addition is one of the most common and basic operations carried out by an ALU. As such, a selection of adder architecture is of high relevance in deciding on the overall performance of the ALU. A number of adder architectures have been created that enhanced the speed whilst limiting the complexity of hardware (Sarkar et al., 2018). Ripple Carry Adder (RCA) is low in complexity and area efficient, but has a high propagation delay because of sequential transfer of carries.

In order to address these shortcomings, more sophisticated architectures like the Carry Look-Ahead Adder (CLA) and the Carry Select Adder (CSLA) have been suggested. The CLA minimizes latency by creating carry signals in parallel with propagate and generate logic (Boateng, 2019). Equally, CSLA enhances performance by computing results via multiple carry inputs in parallel and choosing the appropriate output via multiplexers (Anuraj

et al., 2024). These architectures are faster but with the trade-off of a greater complex hardware.

FPGAs provide an effective platform of implementing and evaluating a digital architecture owing to its reconfigurability, parallel processing and blistering prototyping functions. Implementation on FPGA enables designers to examine the performance characteristics, such as hardware use, timing performance, and power consumption, in realistic scenarios (Rout et al., 2024).

This work design and implements three 16 bit ALU architectures, derived on Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA) and Carry Select Adder (CSLA), in Verilog Hardware Description Language (HDL) and synthesized on a Xilinx Kintex-7 FPGA with the Vivado design suite. Moreover, a hybrid CLA-CSLA architecture is adopted in order to enhance the speed of computation. Moreover, operand isolation is implied to decrease the unnecessary switching activity and improve the power efficiency.

This work sets out to compare and offer an analytical discussion on various adder-based ALU architectures in terms of hardware use, timing, and energy use, as well as to discuss optimization methods to be used to achieve high efficiency. The findings are helpful to the selection of appropriate ALU architectures in digital system FPGA-based architecture. In addition, a hybrid CLA-CSLA architecture with operand isolation is proposed to improve both speed and power efficiency. (Tammana & Vardhan, 2024; Selvi, 2025).

2. LITERATURE REVIEW

Recent studies focus on the optimization of performance of ALU architectures by designing efficient adders on FPGA platforms. Jana et al. (2026) contrast 8-bit ALUs with Ripple Carry Adder (RCA) and Carry Look-Ahead Adder (CLA) noting that CLA is much slower than RCA but much lower in propagation delay, whereas RCA is area-efficient. Their results highlight the trade-off in digital design that is critical in terms of speed and resource consumption. Generalizing this to 16-bit ALUs, more advanced architectures like the Carry Select Adder (CSA) also increase the speed through parallel computation and thus comparative FPGA-based analysis becomes critical in determining the best design options in terms of delay, area, and power consumption.

In Chawla et al. (2016), the FPGA implementation of a power and speed efficient Carry Select Adder (CSLA) is introduced, and the authors note it has the benefit of a short propagation delay, unlike the traditional Ripple Carry Adders (RCA). Their work highlights the area optimization by using modified CSLA structures, and it is applicable to high-performance arithmetic units. The research is also applicable to 16-bit ALU design as it proves that the adder architecture can greatly influence the overall speed, power consumption, and resource usage on FPGA. The results give a solid foundation on comparative analysis of RCA, Carry Look-Ahead (CLA), and CSLA architectures in optimization of ALU performance of contemporary digital systems.

Sunitha et al. (2013) investigated ALU architectures in RISC processors with a focus on the variability in their performance, which was caused by the design of the adder. Their paper emphasizes that Ripple Carry Adders (RCA) are easy to implement, but have high propagation delay, and that Carry Look-Ahead Adders (CLA) are much faster due to parallel carry calculation, but more complex. Carry Select Adders (CSA) offer a trade-off which has a value of speed versus area. These lessons are very applicable to the implementation of 16-bit ALU using FPGA, where delay, area, and power optimization is paramount. The comparative methodology used in the research gives a good background to the analysis of the RCA, CLA, and CSA architecture within the contemporary FPGA settings.

Penchalaiah and Kumar (2022) introduce a low-power, area-efficient ALU architecture with a focus on high performance, implemented with the help of optimized design methods. Their work emphasizes the significance of choosing appropriate adder structures to strike the right balance between speed, power and silicon area in FPGA-based implementations. Though the research does not compare Ripple Carry, Carry Look-Ahead and Carry Select adders explicitly, it highlights the role of adder choice in influencing the efficiency of ALU to a great extent. This observation justifies the importance of comparative evaluation of various 16-bit ALU architectures implemented on these adders especially in FPGA platforms, where delay, resource consumption, and power consumption trade-offs are all essential to realizing optimal performance.

Balaji and Upadhyay (2016) studied the implementation of high-speed low power adder architectures in FPGA with special attention to carry save adder (CSA) to enhance computational efficiency. They emphasize the significance of minimizing carry propagation delay, which is a key element in the performance of an ALU. In spite of the fact that CSA is not similar to ripple carry, carry look-ahead, and carry select adders, the research offers a background perspective of parallel carry processing and power-delay trade-offs. This applies to 16-bit ALU design, in which the choice of adder architecture can have a strong influence on speed, area, and power. Their results justify the necessity of the comparative analysis of the various adders in the FPGA-based ALU implementation.

Gaur, Tyagi, and Mehra (2016) introduce a comparative analysis of adder architectures on an FPGA of 28 nm, where the metrics of performance that were considered included delay, power consumption, and resource utilization.

They point out in their work that Ripple Carry Adders (RCA) are simpler but slower since they propagate carry in a serial manner, and Carry Look-Ahead Adders (CLA) are much faster, but more complex. Carry Select Adders (CSA) provide trade off between speed and use of hardware. This research offers a solid basis to design and analyse a 16-bit ALU, where the choice of a favourable adder architecture is essential in obtaining high performance and efficiency.

A new article by IEEE researchers M. G. Venu et al. (2024) compares optimized parallel adder designs and points out the trade-offs between delay, power, and area. Ripple Carry Adders (RCA) are simple, but with a high propagation delay whereas Carry Look-Ahead Adders (CLA) are much lower in delay but are more complex. The Carry Select Adders (CSA) provide a middle ground between performance and the use of hardware. The findings are vital in the design of FPGA-based 16-bit ALU design because the choice of adder architecture has a direct influence on the performance, resource utilization, and the overall computational speed.

Thamizharasan and Kasthuri (2023) work is devoted to the implementation of a hybrid carry select adder based on FPGA to improve the work of digital-based arithmetic. The design presented is a blend of parallel carry computation methods to minimize propagation delay and maximize area and power. The results indicate a great speed enhancement when compared to other traditional adders like ripple carry adders and carry look-ahead adders. This paper has indicated that hybrid adder architectures can be more effective in designing high-performance ALU on FPGA and, therefore, these architectures can be effectively used to compare 16-bit ALU architecture designs based on RCA, CLA, and CSLA.

Recent articles point out that the design of FPGA-based ALU extensively relies on the efficiency of adder architecture. Ripple Carry Adders (RCA) are efficient in terms of area, but have high propagation delay because of the sequential carry transfer. Carry Look-Ahead Adders (CLA) are used to enhance speed by producing carry signals in parallel, which in turn means that delay is greatly reduced. Carry Select Adders (CSA) are an additional performance boosting technology that stores the output of several carry inputs in advance and picks the appropriate output. Recent papers, such as Maheswari and Abinaya (2024), highlight the use of FPGA-based ALU implementations with the aim of optimizing speed, area, and power of next-generation processors.

According to the research by Buzdar et al. (2016), the performance of ALU is extremely sensitive to the adder architecture used, and Ripple Carry Adder (RCA) is compared to advanced architecture, with a notable difference in delay, area, and power. RCA is a simple design, but has a high propagation delay, because sequential carry computation is used, whereas a faster design is possible with Carry Look-Ahead Adders (CLA), which precomputes carry signals. Recent studies highlight such trade-offs in FPGA/ASIC implementations, with faster adders such as CLA and Carry Select being faster than RCA, but being more complex and consuming more resources. This predetermines the necessity of comparative 16-bit ALU architecture 16-bit ALU architecture comparative FPGA-based analysis.

3. PROPOSED ALU ARCHITECTURE

This section details the design of the proposed 16-bit Arithmetic Logic Unit (ALU) and its implementation using different ways adder designs. It is designed to be modular and hierarchical in that multiple adder architectures will be provided on the same ALU. The architecture does permit ordered comparison and analysis of different implementations of arithmetic, which have identical functional behaviour.

The ALU architecture is implemented using the Verilog Hardware Description Language (HDL) and implemented on an FPGA platform. The proposed design has three versions of ALU that employ Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA) and Carry Select Adder (CSLA). The CSLA architecture is deployed in this work with inner CLA blocks that create a hybrid CLA-CSLA architecture to enhance the computation speed. The modular architecture allows each adder architecture to be independently analyzed with regards to hardware usage, timing, and power usage.

Overall System Architecture

Figure 1 is a diagram of the general architecture of the proposed ALU. The design is hierarchical being based on a testbench, wrapper module and a number of ALU architectures based on different adder engines

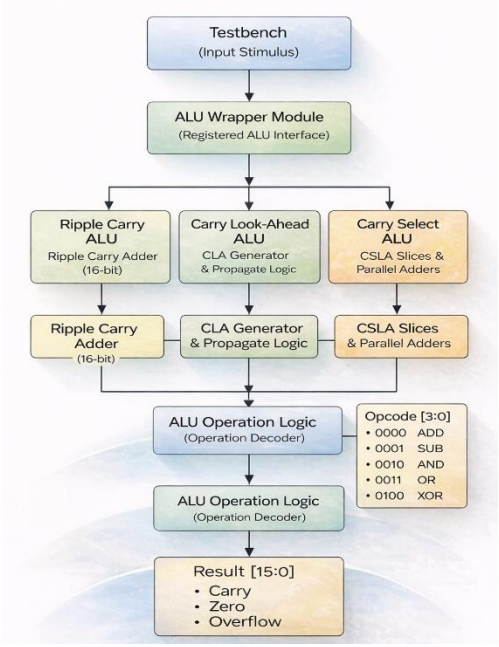


Figure 1: ALU System Architecture

To simulate the ALU, the testbench module (tb_alu16.v) is generated to drive the stimuli and tests into it to determine its functionality. It works through arithmetic and logical operations in various input operands and operation codes.

The top level module is the wrapper module (alu16_wrapper.v), which is used to synthesize and implement on the FPGA. It captures inputs and outputs of the ALU in order to provide a stable and synchronized operation to allow comparison of different architectures fairly.

The layer on architecture implementation consists of three implementations of ALUs:

- **alu16_simple** – ALU using Ripple Carry Adder
- **alu16_cla** – ALU using Carry Look-Ahead Adder
- **alu16_csla** – ALU using Carry Select Adder (hybrid CLA-CSLA)

The hierarchical design structure enhances the modularity and allows a good comparison between various arithmetic architectures within the same ALU design.

Simple ALU (Ripple Carry Architecture)

The original ALU design is a Ripple Carry Adder (RCA) design. Figure 2 is the architecture of this design.

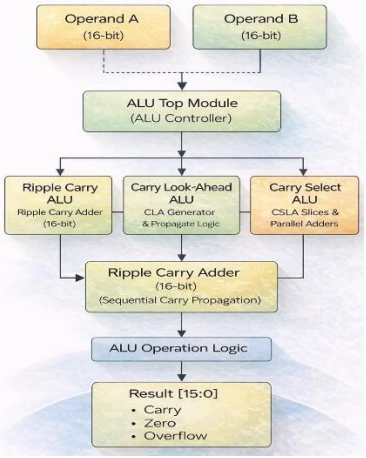


Figure 2: Ripple Carry ALU Architecture

Under the ripple carry architecture, the carry output produced at each stage of the full adder is passed on to the subsequent stage. The sum and carry is computed by each bit position in accordance with its input operands and the carry produced by the previous stage. Therefore, the carry signal has to pass across all the intermediate stages before the ultimate result can be generated.

Its popularity is due to the ease of construction and low hardware costs associated with it. The ripple carry architecture uses less logic and is thus area efficient when used in the FPGA devices.

The biggest weakness of this architecture is however based on its carry propagation delay. The carry signal has to flow in a time-dependent manner through each stage of the adder thus the general delay rises with the quantity of bits. This feature renders ripple carry adders inappropriate in high-speed arithmetic.

Although the ripple carry architecture is limited by delay the architecture is an important benchmark used to analyze the performance gains of other more advanced adder architectures.

Carry Look-Ahead ALU

The second ALU architecture is implemented on the design of Carry Look-Ahead Adder (CLA) as presented in Figure 3.

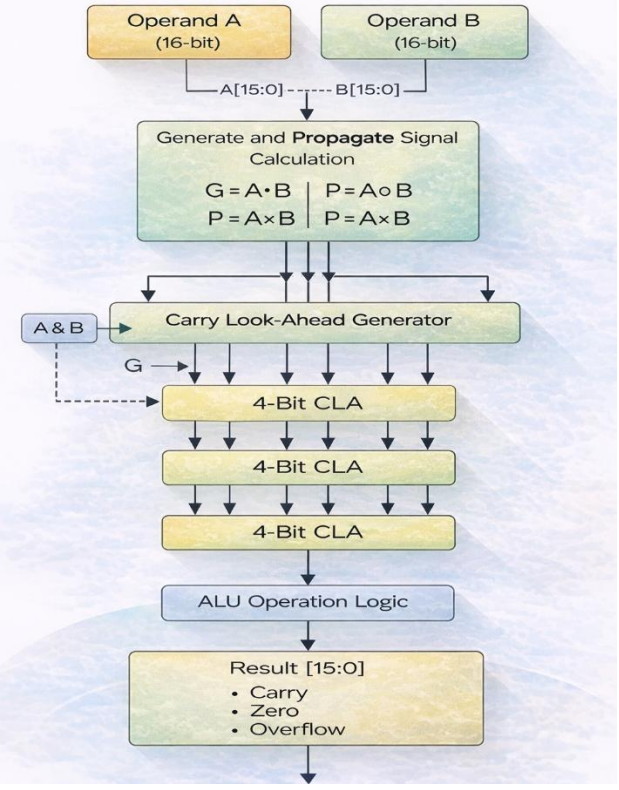


Figure 3: Carry Look-Ahead ALU Architecture

The CLA architecture enhances the speed of additions by computing carry signals simultaneously and not passing them through. This is accomplished by the generation and propagate signals which decide whether a stage will generate or propagate a carry or not.

The generate and propagate signals are defined as:

$G = A \cdot B$

$P = A \oplus B$

Where:

- (G) represents the **generate signal**, indicating that the stage will produce a carry regardless of the input carry.
- (P) represents the **propagate signal**, indicating that the stage will propagate the incoming carry to the next stage.

With these signals it is possible to compute carry outputs with combinational logic rather than with sequential propagation. This has a significant impact on a decrease in carry delay and an increase in overall speed of arithmetic operations.

Whereas the CLA architecture is more efficient than the ripple adders in terms of performance, it provides extra complexity to logic because of the carry generation network. This leads to a small increase in hardware usage in the case of implementation of the FPGA devices.

Carry Select ALU

The third architecture of ALU has the structure of Carry Select Adder (CSLA). Figure 4 is a depiction of the architecture of this design.

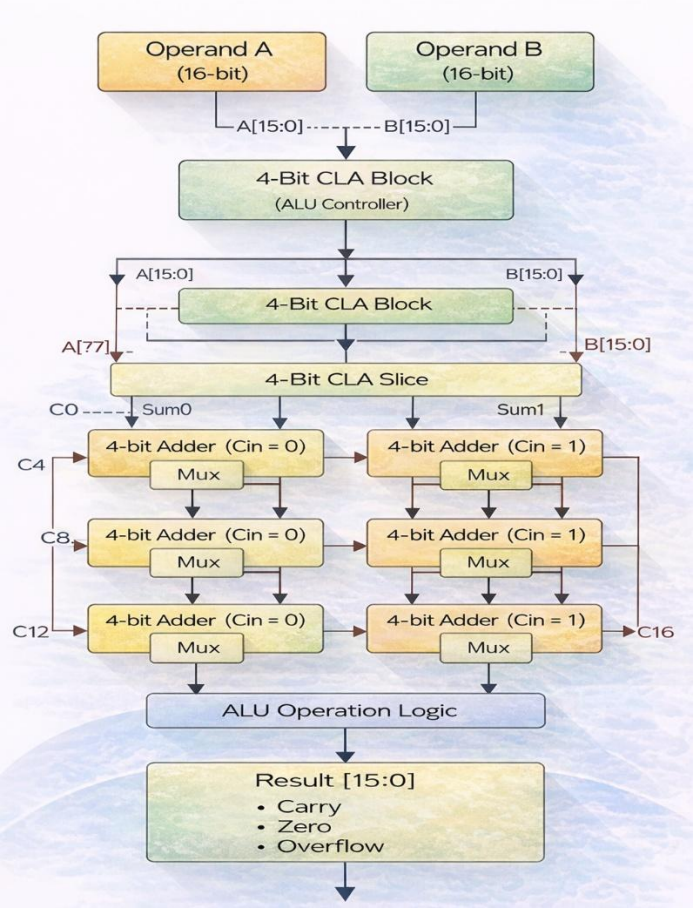


Figure 4: Carry Select ALU Architecture

CSLA architecture enhances the speed of addition by calculating multiple potential outcomes at the same time. Each block of the adder has two sum outputs, which are computed in parallel with two possible carry input values:

- Carry input = 0
- Carry input = 1

The two amounts are calculated at the same time and when the actual carry input is available, a multiplexer is used to choose the appropriate output.

The method greatly minimizes the delay caused by carry propagation since the sum outputs are already computed. Consequently, CSLA architecture gives faster arithmetic speed than ripple carry adders.

Nevertheless, CSLA design needs extra hardware facilities as they need duplicate adder blocks to calculate parallel sums. In spite of the trade-off between speed and hardware complexity is favorable in this architecture, regardless of the increased area overhead.

In general, the Carry Select architecture is a viable way to enhance the work of ALU in the digital system based on FPGA.

4. FPGA IMPLEMENTATION

This part gives a description of the implementation methodology to be employed to implement the proposed 16-bit ALU architectures on an FPGA platform. Ripple Carry, Carry Look-Ahead and Carry Select designs of ALU adders were designed in Verilog Hardware Description Language (HDL) and simulated in the Xilinx Vivado Design Suite. Evaluation of the architectures can be done with regard to hardware utilization, timing performance, and power consumption using the FPGA implementation process.

It was implemented with Vivado 2023.1, a popular synthesis, simulation and implementation environment of digital systems based on FPGAs. The actual hardware will be the Xilinx Kintex-7 FPGA, the xc7k70tfbv676-1 device, to be used in this research project. This FPGA platform has enough logic resources and execution capability to execute and debug more complex digital architectures like arithmetic logic units.

Verilog HDL was used in the design of the ALU architectures and offers a high-level hardware modeling language of digital circuits. Verilog allows development of designs in a modular and hierarchical way, with several adder architectures being developed in the same ALU design. The ALU architectures were all presented as independent modules and this makes it easy to compare during synthesizing and implementation.

The general FPGA implementation is in the normal digital design flow as illustrated in Figure 5.

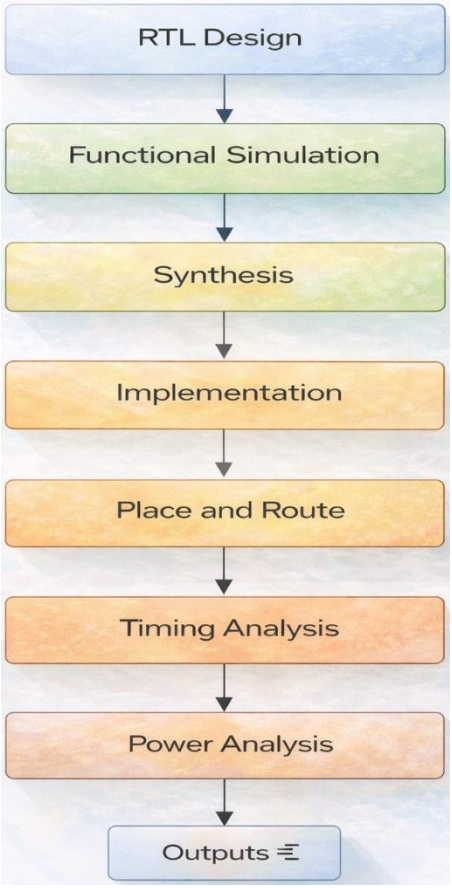
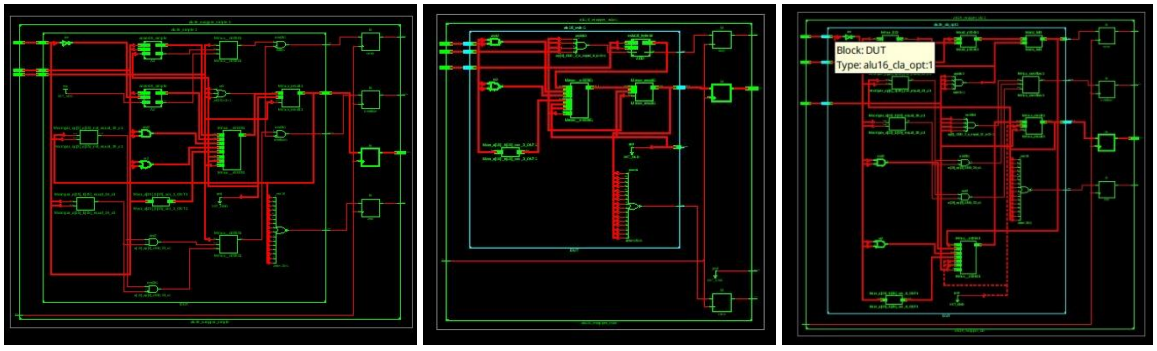


Figure 5: FPGA Design Flow

Design process starts with Register Transfer Level (RTL) design the ALU architectures as shown in figure 6 and other supporting modules are specified in Verilog HDL. Functional simulation follows RTL coding, and it is done with the help of testbench module to ensure that arithmetic and logical operations are done correctly.



(a) (b) (c)

Figure 6: Register Transfer Level (RTL) design of (a) Simple (b) CSLA (c) CLA

After the functionality is checked, the design is logic synthesized, where the RTL code is converted to a gate-level representation that can be used with the target FPGA device. The design is mapped to FPGA resources including Look-Up Tables (LUTs), flip-flops and routing resources in the synthesis process.

After synthesis, the design is sent to the implementation phase that involves placement and routing. The logic elements that are synthesized are placed in definite locations of the FPGA fabric during the placement. These elements are then linked by routing interconnect resources which are programmable to realize the necessary signal paths.

Once they are placed and routed, timing analysis is conducted to ensure that the design has the correct timing constraints. Parameters like worst negative slack, maximum operating frequency among others are analyzed to be certain of reliable operation of the circuit.

Besides timing analysis, power analysis is also performed to determine the approximate power consumption at the chip level of each ALU architecture. This analysis gives an insight with regard to the energy efficiency of the implemented designs.

In order to simplify the implementation process, TCL automation scripts were used in Vivado environment. These scripts automate the synthesis, implementation and report generation processes, allowing standard assessment of a wide range of ALU architectures. The automated execution will provide reproducibility of the results and make it much easier to extract reports on resource utilization, timing and power to compare them.

The framework of FPGA implementation created in this paper gives an opportunity to evaluate the various ALU architectures under the same conditions of hardware and appropriately compare their property of performance.

5. EXPERIMENTAL RESULTS

This part is a report on the results of the experiment of the FPGA implementation of the proposed 16-bit ALU architectures. The Ripple Carry, Carry Look-Ahead and Carry Select adders were designed and synthesized using the Xilinx Vivado design environment. The evaluation is done based on three key metrics of performance namely: hardware resource use, timing performance and power usage. These parameters give the information on the trade-offs between area efficiency, speed of computation and energy efficiency of the various ALU architectures.

These results were acquired upon the synthesis and implementation with the use of the Kintex-7 FPGA (xc7k70tfbv676-1) device. The vivado generated reports, which were analyzed to obtain the applicable performance metrics.

Resource Utilization

The use of hardware resources is a significant measure when assessing the efficiency of digital architectures on the FPGA devices. The major resources that are taken into account in this work are Look-Up Tables (LUTs), Flip-Flops (FFs), and Input/Output (IO) ports. The resources are the logical and physical components needed to execute the ALU architectures on the FPGA.

The summary of resource usage of the three ALU architectures is presented in Table 1.

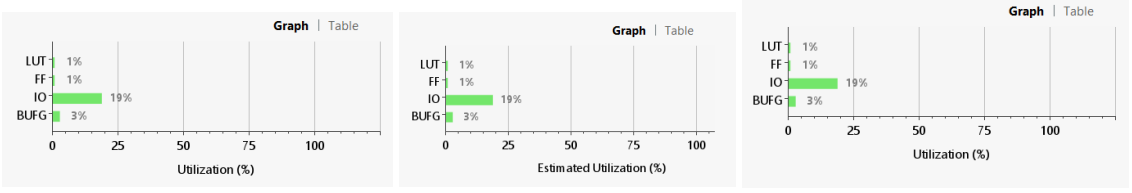
Table 1: FPGA Resource Utilization

Architecture	LUT	FF	IO
Simple ALU	73	55	56
CSLA ALU	109	55	56
CLA ALU	111	55	56

These findings indicate that the Simple ALU version which is based on the Ripple Carry Adder uses the fewest LUTs, thus, being the most area efficient. This is not surprising since the ripple carry architecture has an easy to implement sequential carry propagation model that only needs few extra logic.

By comparison, both the CSLA and CLA architectures have higher LUT resource requirements because they have extra carry computation logic to enhance performance. The CSLA architecture adds parallel add paths of computation, whereas the CLA architecture adds more generate and propagate logic of predicting the carries.

The graphical comparison of resource utilization of the three architectures is shown in Figure 7.



(a) (b) (c)

Figure 7: Resource Utilization Comparison of (a) Simple (b) CSLA (C) CLA

It is important to note that as shown in the graph, even though the advanced architectures need greater amounts of hardware resources, the growth in the amount of resources used is relatively low with respect to the total amount of FPGA resources available.

Timing Analysis

Performance measurement of digital circuits is also an important consideration, which is timing performance. The information on the slack values of the design is obtained by the timing reports obtained after implementation. In this analysis, two important parameters are taken into consideration:

- Worst Negative Slack (WNS)
- Worst Hold Slack (WHS)

A positive value of slack value shows that the design is within the stipulated timing constraints.

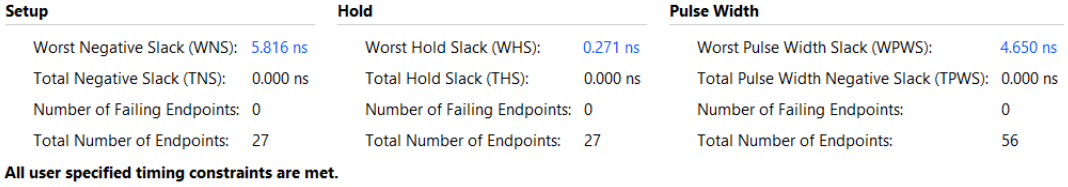
The results of a timing analysis of the three ALU architectures are given in Table 2.

Table 2: Timing Performance

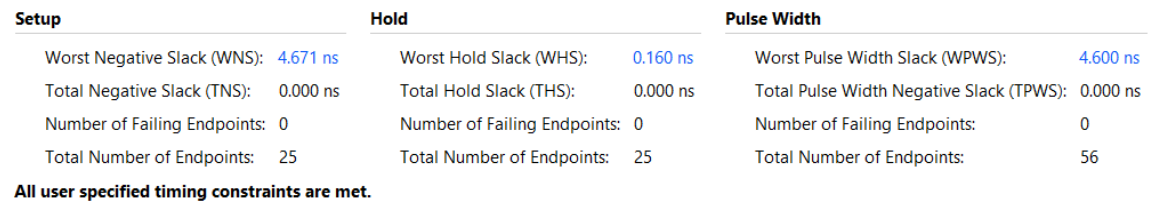
Architecture	WNS (ns)	WHS (ns)
Simple ALU	5.816	0.271
CSLA ALU	4.671	0.160
CLA ALU	4.873	0.263

The findings have shown that the architectures meet the timing requirements with non-negative slack values, which give them confidence in operating on the FPGA platform. The ripple carry architecture records the greatest slack value since it has a less complicated logic structure. More sophisticated architectures including CLA and CSLA are however aimed at minimizing the carry propagation delay as well as fast arithmetic operations in bigger systems.

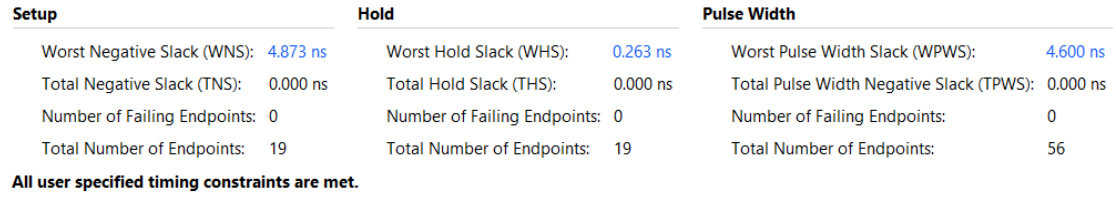
Graphical comparison of timing performance has been performed as shown in figure 8.



(a)



(b)



(c)

Figure 8: Timing Comparison of (a) Simple (b) CSLA (C) CLA

The timing results indicate the efficiency of carried-out optimization techniques in the CLA and CSLA architectures.

Power Analysis

The other important issue that needs to be considered in the contemporary design of digital systems, especially with respect to embedded and portable systems, is power consumption. An implementation was followed by power analysis through vivado power estimation tool.

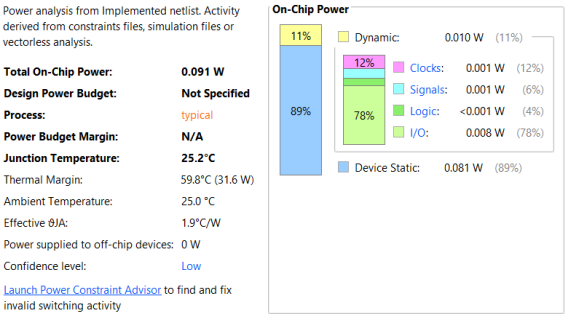
Table 3 shows the summary of the total on-chip power consumption of each ALU architecture.

Table 3: Power Consumption

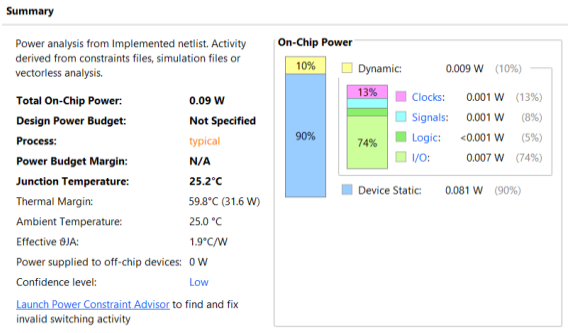
Architecture	Total Power
Simple ALU	0.091 W
CSLA ALU	0.090 W
CLA ALU	0.087 W

The findings indicate that all the three architectures have a very low power consumption below 0.1 W. The CLA-based ALU is the design with the lowest total power consumption, which implies a little more efficient energy saving.

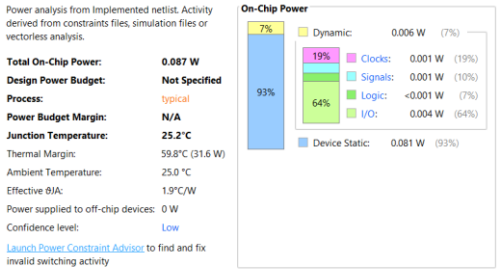
The comparison of the power consumption of the architectures is shown in Figure 9.



(a)



(b)



(c)

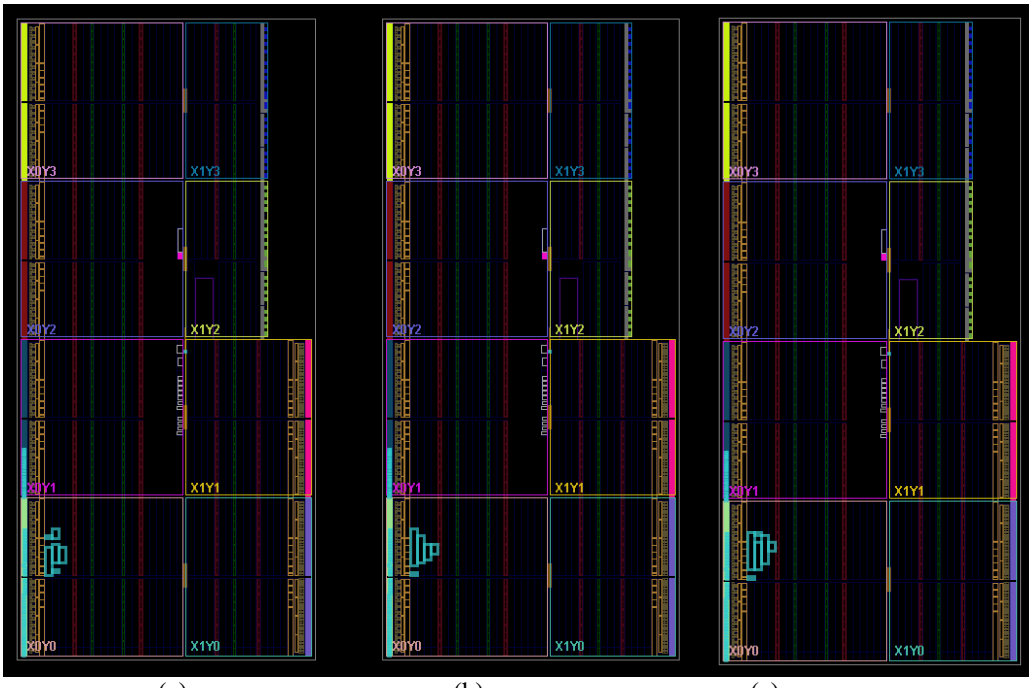
Figure 9: Power Consumption Comparison of (a) Simple (b) CSLA (C) CLA

Though the power consumption between the architectures is not very different, the findings show that optimized carry computation can bring marginal changes in terms of energy efficiency.

FPGA Placement and Routing

The design is then synthesized and implemented onto the FPGA fabric followed by placement and routing of the design. At this phase, the logic components are assigned onto physical FPGA resources and linked with programmable routing paths.

Figure 10 is the visualization of the placement and routing produced by Vivado implementation tool.



(a) (b) (c)

Figure 10: FPGA Floorplan After Placement and Routing of (a) Simple (b) CSLA (C) CLA

The floor plan represents the logical resource allocation in the FPGA device. The routing paths show that the ALU design does not consume much of the available FPGA resources, which means that the hardware fabric was efficiently used.

Effective placement and routing will help in better signal propagation and less congestion in the FPGA. The lack of routing conflicts and the completion of timing analysis by the designed ALU architectures are indicative that the provided ALU architectures are reliable in the context of the target FPGA device.

6. DISCUSSION

The experimental data (FPGA implementation) makes the three ALU designs (Ripple Carry, Carry Look-Ahead and Carry Select adders) well compared. The assessment was performed regarding the hardware resource usage, time performance, and power consumption. These parameters are significant in establishing the appropriateness of each architecture in the digital systems of FPGA.

Based on the resource utilization data, it can be seen that Ripple Carry ALU has the lowest hardware requirement of 73 LUTs and 55 flip-flops. This validates the fact that the ripple carry structures are very area-efficient as they have a naive way of propagating sequential carries. Nevertheless, more complex architectures like CLA and CSLA demand more minor hardware resources of 111 LUTs and 109 LUTs respectively, due to extra carry computation and parallel processing logic.

According to the timing analysis, all architectures meet the stated timing requirements, which is characterized by positive values of slack. Even though ripple carry adders are simpler in their structure, more optimized carry computations in CLA and CSLA architectures enhance computational efficiency of larger arithmetic systems. These architectures decrease the lag of carry propagation by either anticipating carries or paralleled adding up sums.

Analysis of power reveals that all the implementations are under 0.1 W of on-chip power. The CLA-based ALU has shown a very low power consumption that is a little lower than that of the other architectures during the evaluation. In general, the findings demonstrate the trade-off in the complexity of hardware and the computational performance in FPGA-based ALU designs.

The hybrid CLA-CSLA architecture improves performance by combining fast carry computation with parallel processing, while operand isolation enhances power efficiency.

7. CONCLUSION

The design and FPGA implementation of three 16-bit ALU architectures, which are Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA) and Carry Select Adder (CSLA) are presented in the paper. Verilog designs are used to implement the designs and synthesised using the Vivado design environment on a Xilinx Kintex-7 FPGA. The paper aims at comparing the effect of various adder architectures on the usage of hardware, timing performance, and power consumption.

Experimental findings indicate that the Ripple Carry ALU is the area-efficient design since it has a simple structure, but it has higher propagation delay, thus not as practical to high speed applications. Conversely, CLA and CSLA architecture enhances the available computational performance by minimizing carry propagation delay by using parallel carry computation and optimized logic structures.

Here, the CLA blocks are used as the implementation of the CSLA architecture which constitutes a hybrid CLA-CSLA design which is more efficient in terms of performance due to the combination of the benefits of the CLA and the CSLA. Also, the operand isolation is employed to minimize unnecessary switching action, hence better power efficiency.

Generally, the findings shows that despite the fact that the Ripple Carry ALU is ideal in curbing the space constraint applications, the hybrid CLA-CSLA architecture offers a more balanced approach in speed, power, and hardware exploitation. Thus, the suggested optimized ALU design is suitable to the high-performance FPGA-based digital systems.

Further work Liked higher order adder designs (parallel prefix adders, etc.) and additional techniques to optimize power through further low-power architectures in enhancing the ALU performance can be pursued.

REFERENCES

[1] Sarkar, S., Sarkar, S., & Mehedi, J. (2018, January). Comparison of various adders and their VLSI implementation. In 2018 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-9). IEEE. doi: 10.1109/ICCCI.2018.8441253.

[2] Gurjar, P., Solanki, R., Kansliwal, P., & Vucha, M. (2011, December). VLSI implementation of adders for high speed ALU. In 2011 Annual IEEE India Conference (pp. 1-6). IEEE. doi: 10.1109/INDCON.2011.6139396.

[3] Tammana, M., & Vardhan, M. (2024, June). Efficiency analysis of adder architectures: A comparative study across various bit lengths. In 2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS) (pp. 1-6). IEEE. doi: 10.1109/ICITEICS61368.2024.10625527.

[4] Rout, S. S., Patjoshi, R. K., Garnaik, S., & Rout, R. (2024). Comparative Analysis of Heterogeneous Adders: Evaluating Performance across 12-bit, 14-bit, and 16-bit Configurations. Journal of Information Assurance & Security, 19(4). https://reference-global.com/download/article/10.2478/ias-2024-0010.pdf

[5] Selvi, A. M. (2025). Efficient FPGA based Implementation of a 64-bit Parallel Prefix Adder. https://philpapers.org/rec/MUTEFB

[6] Anuraj, V., Verma, P., & Vaithiyanathan, D. (2024, November). Implementations and Analysis of Efficient Adders for 8, 16, and 32-Bit Design. In 2024 First International Conference for Women in Computing (InCoWoCo) (pp. 1-6). IEEE. doi: 10.1109/InCoWoCo64194.2024.10863541.

[7] Boateng, E. A. (2019, April). Design and Implementation of a 16 Bit Carry-Lookahead Adder. https://www.researchgate.net/profile/Emmanuel-Aboah-Boateng/publication/343982099_Design_and_Implementation_of_a_16_Bit_Carry-_Lookahead_Adder/links/5f4c04c892851c6cfd07206a/Design-and-Implementation-of-a-16-Bit-Carry-Lookahead-Adder.pdf

[8] Jana, S., Dutta, S., Das, R., Sinha, M., & Sarkar, M. (2026, February). Comparision Between 8 Bit ALU Using CLA & RCA. In 2026 9th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech) (Vol. 9, pp. 1-6). IEEE. doi: 10.1109/IEMENTech202669403.2026.11434177.

[9] Chawla, S. S., Aggarwal, S., Goel, N., & Bhatia, M. S. (2016, March). Design and implementation of a power and speed efficient carry select adder on FPGA. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 571-576). IEEE.

[10] Sunitha, M. S., Hegde, B. G., & Hegde, D. N. (2013). Design and Comparison of Risc Processors Using Different Alu Architectures. International Journal of Innovative Research in Science, Engineering and Technology, 2(7). https://d1wqtxts1xzle7.cloudfront.net/81768537/design-and-comparison-of-risc-processors-using-different-alu-architectures-libre.pdf?1646504230=&response-content-disposition=inline%3B+filename%3DDesign_and_Comparison_of_Risc_Processors.pdf&Expires=17762621

52&Signature=CszQyXDzOWj2kdU4YNuBqxiLGIcxA2pih-fHEUjZMIELfoiy6PRl9WJ5gbu56rAyggQ8rprzZ5~~UFt-2g4T1BR5jU2B6Ce-r26twRVi1NzU7hHWhlFwlcB-aWyw-m0mdDWG-IIfiYQHCBZrk-b1CoZ2XmElE-rXRAY-eRwcs5dgQ3WMf7mlbwVrEVuADsbm4WxUzhEX3JstIah~2fEqp3eJU0s4j1r9q1DKXan067FJCqMkSwE5xbtNOepf~KRd1DBikcvyv8gZM4bSRzGxaA3oLG7HBGUUX2m2DN6J5EKCUiiTMmwa-WeZ2w-f4fSsev~MyxFmK5ClcJKHc75Prw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[11] Penchalaiah, U., & Kumar, V. S. (2022). Design and implementation of low power and area efficient architecture for high performance ALU. Parallel Processing Letters, 32(01n02), 2150017. https://doi.org/10.1142/S0129626421500171

[12] Balaji, V. S., & Upadhyay, H. N. (2016). FPGA implemenation of high speed and low power carry save adder. IIOAB J, 7, 151-159.

[13] Gaur, N., Tyagi, D., & Mehra, A. (2016, September). Performance comparison of adder architectures on 28nm FPGA. In 2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall) (pp. 1-5). IEEE. doi: 10.1109/ICACCAF.2016.7748980.

[14] Venu, M. G., Dileep, B. N., Raghavendra, H., & Veena, M. B. (2024, October). Performance Analysis of Different Optimized Digital Parallel Adders. In 2024 5th IEEE Global Conference for Advancement in Technology (GCAT) (pp. 1-6). IEEE. doi: 10.1109/GCAT62922.2024.10924060.

[15] Thamizharasan, V., & Kasthuri, N. (2023). Design of proficient two operand adder using hybrid carry select adder with FPGA implementation. IETE Journal of Research, 69(12), 9152-9165. https://doi.org/10.1080/03772063.2022.2071771

[16] Maheswari, M., & Abinaya, A. (2024, July). Feasible Implementation of ALU for Next Generation Processors. In 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT) (pp. 1-9). IEEE. https://doi.org/10.1109/ICEEICT61591.2024.10718541

[17] Buzdar, A. R., Sun, L., & Buzdar, A. (2016). Comparative analysis of alu implementation with rca and sklansky adders in asic design flow. International Journal of Advanced Computer Science and Applications, 7(7). https://pdfs.semanticscholar.org/b049/e30f2d876ae190f73bb205c336fcbc26a6b3.pdf.