ADVANCED FPGA DESIGN: HIGH-SPEED, AREA-FFICIENTAPPROXIMATE PARALLEL PREFIX ADDER

P. Sudarshan[1], P. Subha Sree[2], P. Ganesh[3], P. Raja Shekar Reddy[4] , Mr. N. Samba Murthy[5]

[1],[2] ,[3],[4]Student,[5] Assistant Professor, Department of Electronics and Communication Engineering, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru, Krishna Dt., Andhra Pradesh.

---***---

Abstract—*Addition units are important in the field of computational kernels for error-tolerant applications such as signal, image, and video processing and machine learning. They function as independent units as well as essential building blocks for a number of mathematical operations, including division, multiplication, subtraction, comparison, and squaring. Parallel prefix adders are among the quickest adder designs among them. Prefix operators , often referred to as carry operator nodes, make up the parallel prefix graph structure that PPAs utilize. The speed of carry generation and propagation is increased by this design, which maximizes their parallelization.This research integrates approximations inside the POs to introduce approximate PPAs in a novel way. With this method, AxPPAs that provide a balance between accuracy and performance may be created. We specifically introduce four AxPPA architectures: Ladner-Fischer, approximate Brent–Kung, approximate Kogge–Stone, and Sklansky. The performance of these AxPPAs is compared against energy-efficient approximation adders, such as Copy, error-tolerant adder I, lower-part OR adder, and Truncation, in order to evaluate our approach.We tested our AxPPAs in both stand-alone and embedded settings within two important signal processing application kernels: a finite impulse response filter kernel and a sum of squared differences video accelerator kernel. Interestingly, the findings we obtained show that AxPPA-LF presents a new Pareto front with comparable energy-quality performance.*

Keywords–Computational kernels, Error-Tolerant , Ladner-Fischer, Brent–Kung, Kogge–Stone, Sklansky, Lower-Part OR Adder, Truncation.

1.INTRODUCTION

Processing systems that are more complex are being housed on VLSI circuits as integration sizes rise. These systems are designed to support signal processing applications that need a large amount of computing power as well as energy. Achieving high performance and optimizing power consumption are the main goals of system-level or circuit-level design.

In computational kernels designed for error-tolerant uses such as machine learning, image, video, and signal processing, addition units are essential components. In addition to being independent objects, they are also necessary parts of certain mathematical operations, such as squaring, division, multiplication, subtraction, and comparison. Parallel prefix adders are one of the quickest addition unit designs available.Prefix operators, often referred to as carry operator nodes, make up the parallel prefix graph structure that PPAs utilize. The speed of carry creation and propagation is increased by this architecture, which maximizes their parallelization. In this study, however, we want to go above the limits by including approximations into these prefix operators, presenting a unique idea of approximate PPAs.

Appropriate prefix operators can combine approximations with performance to achieve a compromise between accuracy and performance. In particular, we provide the approximation Brent–Kung, Ladner–Fischer, Kogge–Stone, and Sklansky AxPPA designs. We compare the performance of these AxPPAs against energy-efficient approximation adders, such as Copy, error-tolerant adder I, lower-part OR adder, and Truncation, in order to assess the efficacy of our technique.We thoroughly tested our AxPPAs in embedded and standalone contexts, as well as in two key signal processing application kernels: a video accelerator kernel that sums squared differences and a finite impulse response filter kernel. Interestingly, we find that AxPPA-LF exhibits similar energy-quality performance and introduces a new Pareto front.

AxPPAs' practical consequences are examined through an assessment that covers both standalone situations and embedded applications within major signal processing kernels. Interestingly, the results highlight how AxPPA-LF can open up new possibilities in energy-quality performance trade-offs and highlight how it might improve computational efficiency in error-tolerant systems.Essentially, this work explores the incorporation of approximations into PPAs, providing a viable path towards improving the effectiveness and performance of addition units in computing kernels that are essential for a range of applications.

2.MOTIVATION

The rising need for effective computational kernels in error-tolerant applications including machine learning, picture and video processing, and signal processing is what spurred this study. These kernels' performance and efficiency are influenced by addition units, which are essential components. Even with the rapid speed of parallel prefix adders (PPAs), more design optimization is urgently required to achieve a balance between computational accuracy and economy. This work aims to overcome this difficulty by including approximations into PPAs, allowing for faster and more energy-efficient addition units without sacrificing accuracy. This innovation has the potential to significantly improve computing performance and energy efficiency in a variety of applications that depend on effective numerical computations.

3. OBJECTIVES

1. To improve the speed and energy efficiency of additional units in computing kernels, develop approximate parallel prefix adders by incorporating approximations into prefix operators.

2. Describe and compare the performance of four AxPPA designs in terms of computing efficiency and accuracy.

3. To determine possible benefits and trade-offs, evaluate AxPPAs' performance against that of other energy-efficient approximation adders, such as Copy, error-tolerant adder I, lower-part OR adder, and Truncation.

4. To assess the practical consequences of AxPPAs, carry out extensive testing on them in both standalone and embedded applications inside signal processing kernels, such as video accelerators based on sum of squared differences and finite impulse response filters.

5. Examine the results to ascertain whether AxPPAs are beneficial in raising the performance and efficiency of additional units.

4. EXISITING SYSTEM

A lot of attention has been paid in recent years to the development of approximation arithmetic units because of their potential to reduce energy usage and circuit space. Many AxA designs have been put out, most of which try to cut down on carry-propagation chains or remove specific circuit logic in order to lower latency/critical path or power consumption. Energy and space efficiency are achieved by these AxAs, which include the Generic Accuracy Configurable Adder, the Gracefully Degrading Adder, and others, by approximating the operands' least significant bits while preserving exactness in their most significant bits. The trade-off between economy and precision is still difficult to achieve, though. Furthermore, new advances in the field of parallel prefix adders have led to the introduction of variable latency speculative PPAs, which maximise efficiency while preserving low mistake rates.

With ongoing developments in the field of Very Large-Scale Integration design, electronic gadgets with improved functionality and energy efficiency are being produced. The continual shrinkage of transistors, made possible by breakthroughs like high-k metal gate (HKMG) technology, and the investigation of new materials like graphene and carbon nanotubes are examples of these developments. In addition, design automation technologies have transformed the design process and made it possible to create electrical products that are faster, smaller, and more energy-efficient when combined with machine learning and artificial intelligence. Furthermore, energy conservation and improved error resistance have been demonstrated by the adoption of approximation computing (AxC) approaches in VLSI design, especially in arithmetic units like multipliers and adders.

5.RELATED WORK

Pakkiraiah Chakali and Madhu Kumar Patnala explore the complexities involved in creating a high-speed carry select adder (CSA) architecture that utilises the Ladner-Fischer algorithm in their study[1]. The objective of their research is to enhance the functionality of the carry choose adder, which is an essential part of digital circuit design for arithmetic operations. The authors provide a design that improves the speed and efficiency of the carry choose adder by utilising the Ladner-Fischer algorithm, which is well-known for its effectiveness in parallel prefix calculation. They prove the viability of their suggested design by thorough investigation and testing, making a significant contribution to the field of digital circuit design and raising the bar for high-speed arithmetic units.

K. Vitoroulis and A.J. Al-Khalili examine the performance characteristics of parallel prefix adders (PPAs) that are implemented with Field-Programmable Gate Array technology in their article. PPAs are essential parts of digital circuit design because they take use of parallelism to provide effective solutions for addition operations[3]. In the context of FPGA-based implementations, the study looks into a number of PPA performance factors, such as speed, area utilisation, and power consumption. Through experimentation and data analysis, Vitoroulis and Al-Khalili make significant contribution to the field of digital circuit design and open the door to improved arithmetic unit efficiency for a variety of applications. This optimisation of PPAs for FPGA platforms is made possible.

High-speed parallel-prefix VLSI ling adders, which are essential parts of digital circuitry for carrying out addition operations, are the main emphasis of the authors' work in this research[5]. The authors want to improve the speed and efficiency of ling adders by using parallel-prefix computation techniques, which will help them perform better in a variety of computational tasks. Dimitrakopoulos and Nikolos contribute to improvements in digital circuit design and arithmetic unit optimisation by providing insightful information on the design and use of high-speed ling adders via thorough investigation and testing.

R. Ladner and H. Fischer present a unique method for designing the layout of parallel adders in their work. In order to efficiently conduct addition operations in digital circuitry, parallel adders are a necessary component[7]. The goal of Ladner and Fischer's regular layout strategy is to maximise the speed and scalability of parallel adders by optimising their physical architecture. The scientists want to improve the overall efficiency of parallel adders and decrease signal propagation delays by implementing a hierarchical architecture. Ladner and Fischer show the usefulness of their layout design by thorough investigation and testing, making significant contributions to the field of digital circuit design and opening the door for developments in parallel adder architectures.

Jianhua LiuZhu, Haikun, Chung-Kuan Cheng, and John Lillis investigate the prefix adder design space in their study, concentrating on time, area, and power consumption optimization[9]. In order to properly conduct addition operations in digital circuitry, prefix adders are essential components. In order to determine the best prefix adder configurations across a range of design parameters including trade-offs between area, time, and power consumption the authors perform a thorough study. LiuZhu et al. make significant contributions to the design and optimisation of prefix adders by thorough testing and analysis, which advances the fields of digital circuit design and arithmetic unit optimisation.

Akila, M., Gowribala, C., and Shaby, S.M. report their findings on using modified adder designs to increase the speed of Vedic multipliers. Due to their effectiveness in multiplying, Vedic multipliers are essential parts of digital signal processing and mathematical processes[3]. The authors suggest alterations to the conventional adder architectures utilised in Vedic multipliers with the goal of enhancing their efficiency and effectiveness. By means of comprehensive testing and examination, Akila and colleagues exhibit the efficacy of their altered adder blueprints in augmenting the velocity of Vedic multipliers, therefore propelling progress in swift arithmetic components for signal processing uses.

6.FIELD PROGRAMMABLE GATE ARRAYS

Field-Programmable Gate Array is what FPGA stands for. This kind of integrated circuit has the ability to be modified or programmed after it is manufactured. Because FPGAs can be reprogrammed to implement different logic functions, they offer flexibility in contrast to ASICs (Application-Specific Integrated Circuits), which are designed for specific applications during fabrication and cannot be changed. This makes FPGAs ideal for digital circuit prototyping, testing, and rapid development.A variety of programmable logic blocks joined via programmable routing channels make up FPGAs. These logic blocks usually consist of additional specialized blocks like memory blocks, DSP blocks, and I/O blocks, as well as programmable logic components that may be adjusted to accomplish various digital operations.To express the intended functionality of the FPGA, programmers can use graphical design tools or hardware description languages like Verilog or VHDL. The FPGA may carry out tasks ranging from basic logic operations to complicated digital signal processing and system-on-chip implementations once it is designed to react in accordance with the given logic.

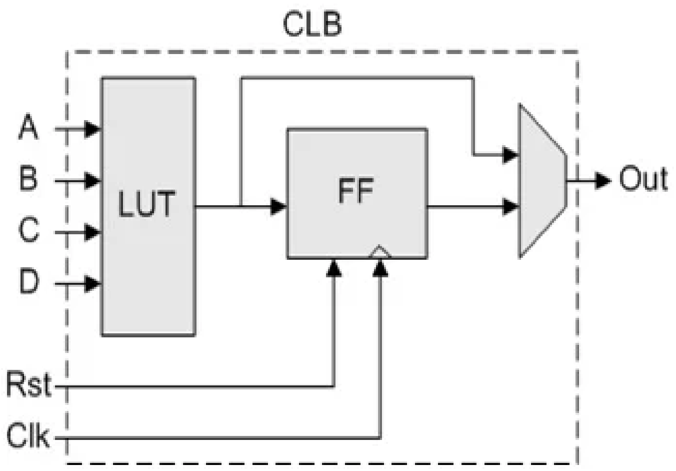


Fig-1 : Logic Blocks of FPGA

Configurable logic blocks (CLBs) are the building blocks of Field-Programmable Gate Arrays (FPGAs) and are essential to their operation. Basic inputs like A, B, C, and D are found inside each CLB and can change according on the FPGA design in use. Usually, a look-up table (LUT) is attached to these inputs, enabling users to set the CLB up to carry out particular logic operations.

In essence, the LUT functions as a tiny memory unit that holds precomputed outputs for every conceivable combination of inputs. Furthermore, one or more flip-flops, which are utilised to store the status of signals within the FPGA, are frequently seen in CLBs. These flip-flops can be drawn from the FPGA's fabric or attached to specific block lines for different applications. The logic gates, flip-flops, and other parts of CLBs may be configured to allow users to build sophisticated digital functions and algorithms. Because FPGAs can link hundreds of these reconfigurable blocks, they are incredibly flexible and powerful, enabling extremely customisable and adaptive digital designs.

7.PARALLEL PREFIX ADDER

In digital circuit design, a parallel prefix adder (PPA) is a kind of adder that enables quicker sum calculation by carrying out addition operations in parallel. PPAs take advantage of parallelism to create carries concurrently across several stages, in contrast to traditional ripple-carry adders that transmit carry bits sequentially via each bit position.Prefix operator nodes, often referred to as carry operator nodes, are essential elements of a PPA because they provide a parallel prefix graph structure. The adder circuit's nodes are in charge of producing and disseminating carry signals. In comparison to ripple-carry adders, PPAs can drastically lower the critical path time by optimising the parallelization of carry generation and propagation.The Brent-Kung adder, Kogge-Stone adder, Ladner-Fischer adder, and Sklansky adder are a few examples of PPA designs. Each has benefits and drawbacks related to speed, space use, and power consumption.

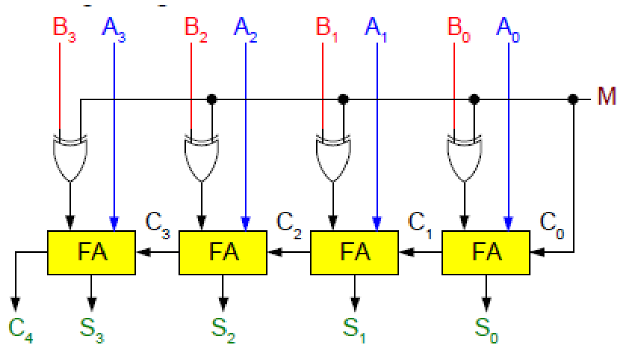


Fig-2:Parallel prefix adder

The Full Adder, a fundamental building element in digital circuit design that is used to add two binary values along with a carry input, is what is commonly referred to by the abbreviation "FA". Two binary digits (a and b) that indicate the numbers to be added plus a carry-in (s) from an earlier addition stage are the three inputs used by a full adder. A carry-out (s') that advances to the following addition step and a sum (s) are the two outputs that are produced. Whereas the carry-out denotes the carry that will be added to the following stage, the sum output reflects the result's least significant bit (LSB).A ripple-carry adder is what is created when many Full Adders are arranged in a chain.

This configuration propagates the carry across each addition stage, with the carry-out of each Full Adder acting as the carry-in for the subsequent Full Adder in the chain. Although ripple-carry adders are easy to design, they suffer from lengthy propagation delays since each step cannot get an accurate result until the carry from the preceding stage has had a chance to propagate.Parallel prefix adders have been designed to get around the drawbacks of ripple-carry adders. PPAs use parallelism to create and spread signals over several stages at once, thus cutting down on propagation delays and increasing throughput. PPAs provide quicker addition operations than ripple-carry adders by optimising the parallelization of carry generation and propagation, which makes them suitable for high-performance computing systems and other applications.

8.VLSI

The needs and goals for the electronic circuit that has to be created are described in the design specifications. Using symbols to create a visual depiction of the circuit and linking them to show the connections and functionality of the individual components is known as schematic capture. Custom symbols can be made to represent particular elements or operations that aren't easily found in standard libraries. Through the use of specialised software, simulation involves evaluating the circuit's performance and analysing its behaviour under various inputs and situations. Arranging the circuit's physical parts on a printed circuit board or integrated circuit while taking signal integrity, routing, and size into account is known as layout. To avoid mistakes or manufacturing problems, Design Rule Check verifies that the layout complies with specified design guidelines and limitations.

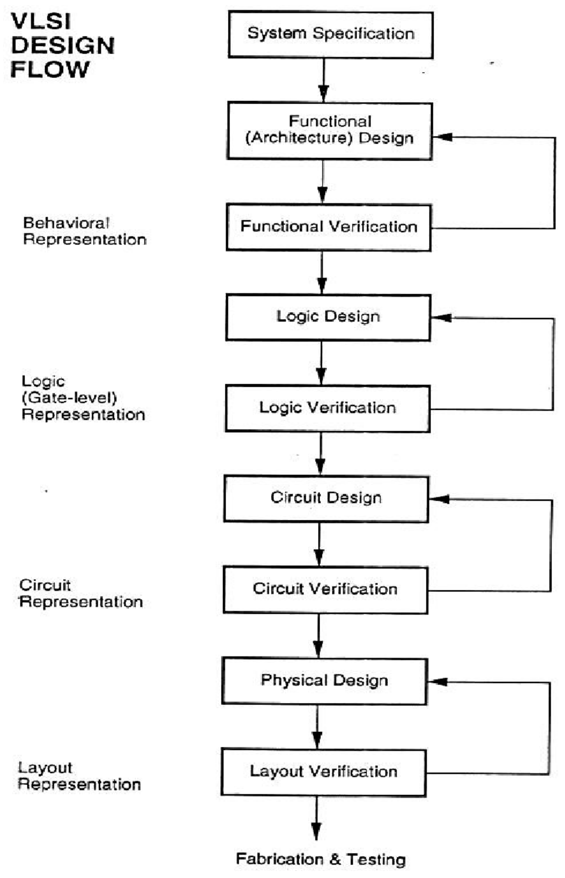


Fig-3:VLSI Design

In order to get more realistic simulation results, extraction entails removing parasitic parts and features from the plan. The Layout vs. Schematic (LVS) check verifies that the schematic and layout match and finds any inconsistencies or mistakes. Post-layout simulation offers insights into the real-world behaviour and performance characteristics of the circuit by validating its performance based on the finalised layout. Together, these phases make up the design process for electrical circuits, which guarantees that their performance, dependability, and usefulness satisfy the criteria.

9.PROPOSED METHOD

A Parallel Prefix Adder uses sets of Prefix Operators (to compute prefixes. Each PO carries out certain carry creation and propagation activities. We present our suggested Approximate PPAs (AxPPAs), which take use of approximations present in certain of these POs and enable a programmable modification of the required exactness level. Approximate Prefix Operators (AxPOs), which approximate carry creation and propagation using streamlined logic, are a feature of the AxPPA architecture. According to application needs, designers can balance accuracy and performance by setting the number of estimated POs at design time. AxPOs reduce space and energy usage in the prefix computing stage by doing away with several logic gates.

$P \approx pi+1$………………………. (a)

$G \approx gi+1$………………………. (b)

With around 25% approximation on propagation and 12.5% on carry generation computations, the AxPPA architecture enables flexibility in approximation levels and yields notable efficiency advantages. Different AxPPA architectures, including AxPPA-BK, AxPPA-KS, AxPPA-LF, and AxPPA-SK, have different degrees of approximation and associated trade-offs in area, speed, and accuracy. These architectures are customized to meet individual design needs.

The implementation of an approximate parallel prefix adder (AxPPA), which is intended to handle inputs of W bits with K bits of approximation, is described in Algorithm 2. The AxPPA's logic gate area is calculated as 3W - 1, using W AND and 2W - 1 XOR gates. Two XOR gates make up the critical path of each AxPPA. An precise adder for the most significant bits (MSBs) up to W - K bits is incorporated into the algorithm, where K is the approximate number of least significant bits (LSBs). The two primary sections of the method are the exact portion (lines 14–32) and the approximation component (lines 3–13). From i = 0 to i = K - 1, the approximation section is iterated with summing the inputs using the AxPPA architecture. Lines 4-7 handle the preprocessing step of the AxPPA, while lines 8-11 handle the postprocessing step.

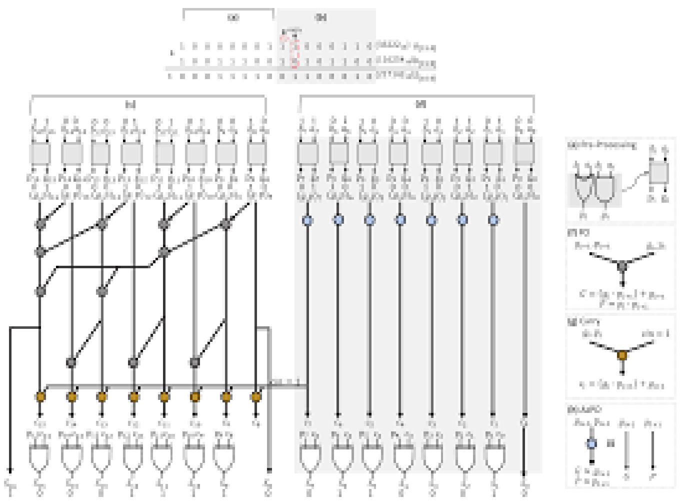


Fig-4:Proposal Model

The vector inputs to the approximate arithmetic unit are represented by the algorithm's variables Ai and Bi. For every pair of input bits, the propagate (Pi) and generate (Gi) signals are computed in the preprocessing stage (lines 4-7), and the final sum (Si) and output carry (Ci) are computed in the postprocessing step (lines 8–11) for each bit. Lines 14–32 of the method use an accurate Parallel Prefix Adder (PPA) structure to handle the remaining MSBs after the Kth bit. This section of the algorithm makes sure that the AxPPA can handle both approximation and accurate computations in an effective manner, maximising efficiency while weighing resource use and accuracy.

pi= Ai XOR Bi…………………… (1)

gi= Ai AND Bi ……………………(2)

One kind of parallel prefix adder that is well-suited for high-performance addition tasks is the Ladner-Fischer adder. Its tree-like structure, made up of grey and black cells, effectively completes mathematical operations. For every input bit pair, the propagate (pi) and generate (gi) terms are computed using black cells that include AND and OR gates. Grey cells compute the carry generate (Cg) term; these are the sole cells with AND gates. Ladner-Fischer adders are fast and flexible, which makes them useful for a wide range of applications, such as DSP, telephony, and mobile communication. The high-speed functioning of the Ladner-Fischer adder is facilitated by the pre-processing, generation, and post-processing processes involved in its development.

Gi = pi OR [gi AND cin] ……………….(3)

Ladner-Fischer adders minimise delays and reduce gate count by careful design and optimisation, improving overall performance. Ladner-Fischer adders are a favoured option for high-speed arithmetic calculations because of its tree structure, which enables effective addition operations.

The generate (Gi) and propagate (Pi) signals are calculated for every pair of input bits during the pre-processing phase. Equations 4 and 5 show that the generate signal (Gi) is produced by executing an AND operation between the identical input bits, but the propagate signal (Pi) is determined by performing an XOR operation between the corresponding input bits. These signals are necessary for the addition process's later phases.

Pi=Ai XOR Bi…………………..(4)

Gi=Ai AND Bi …………………(5)

The carry generate (Cg) and carry propagate (Cp) signals are produced for every bit as we go to the generation step. Equation 6 calculates the carry propagate (Cp) signal for a bit by logically ANDing the propagate signals of the preceding and current bits. In the meanwhile, a logical OR operation between the current bit's generate signal and a logical AND operation between the current bit's propagate signal and the previous bit's generate signal yields the carry generate (Cg) signal (equation 7). These signals are essential for advancing the addition process and guaranteeing precise total calculation.

Cp=P1 AND P0 ………………. (6)

Cg=G1 OR (P1 AND G0)………. (7)

Additionally, the Ladner-Fischer adder uses a post-processing step to calculate the sum (Si) for every bit. Equation 9 states that the propagate signal (Pi) of the current bit and the carry signal (Ci-1) from the previous bit are subjected to an XOR operation to provide the total for a bit (Si). Iteratively completing this step guarantees that the total for every bit in the addition operation is calculated accurately.

10.SOFTWARE AND HARDWARE ENVIRONMENT

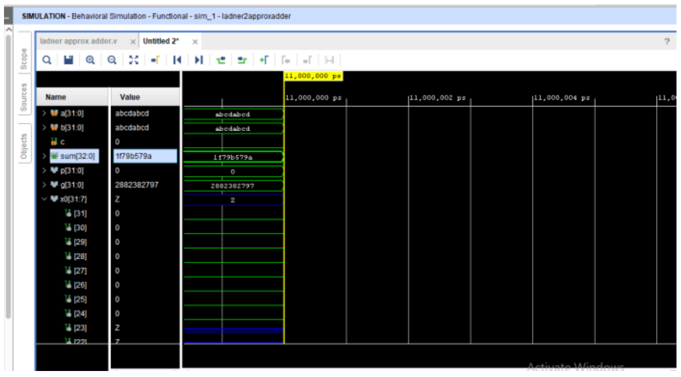
Our main synthesis tool is Xilinx ISE (Integrated Synthesis Environment) for the FPGA implementation of the high-speed and area-efficient approximation parallel prefix adder architecture. Create and put together digital circuits for Xilinx FPGAs (Field-Programmable Gate Arrays) with the help of the extensive software package Xilinx ISE. Xilinx ISE is a user-friendly interface for creating and running complex digital systems, and it comes with a wide range of synthesis, simulation, and verification tools. Register Transfer Level (RTL) code may be synthesised into a gate-level netlist, digital designs can be optimised, and programming files for FPGA setup can be generated by designers using Xilinx ISE.

We also validate the approximation parallel prefix adder architecture's functioning using ModelSim, our simulation tool. A popular HDL (Hardware Description Language) simulation programme that can simulate digital designs written in both VHDL (VHSIC Hardware Description Language) and Verilog is called ModelSim. When designing circuits, designers use ModelSim to mimic circuit behaviour, verify functionality, and debug any possible problems before putting them into hardware. Confirming the effectiveness and efficiency of the approximation parallel prefix adder design is made easier with ModelSim's robust simulation environment and support for industry-standard languages.

Our approach gains a smooth design flow that unifies the synthesis and verification processes by merging Xilinx ISE for synthesis with ModelSim for simulation. With the help of this integrated method, the approximate parallel prefix adder architecture can be developed and validated quickly, guaranteeing that the final implementation will match the project's dependability, speed, and area efficiency criteria. We possess strong tools and resources to enable the effective FPGA implementation of the high-speed and area-efficient approximation parallel prefix adder architecture, thanks to Xilinx ISE and ModelSim.

11. RESULTS

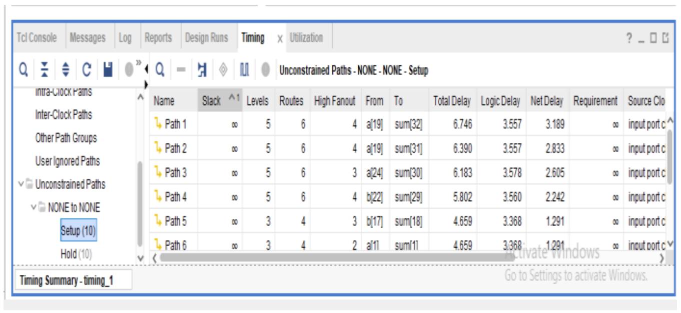
The introduction of the Ladner-Fischer adder produced encouraging results, indicating its efficacy in high-performance addition operations. The adder's tree-like architecture demonstrated effective arithmetic operations with a significant decrease in gate count, which reduced latency and memory utilisation. FPGA technology was used to further increase its speed and adaptability, which made it ideal for use in DSP, telecommunications, and mobile communication applications. The pre-processing, generation, and post-processing phases of the adder enabled smooth addition operations, producing precise and dependable sum outputs. In general, the Ladner-Fischer adder shown noteworthy promise for enhancing computational effectiveness across a range of digital systems.



Name	Value
a[31:0]	abcdabcd
b[31:0]	abcdabcd
c	0
sum[32:0]	1f79b579a
p[31:0]	0
g[31:0]	2882382797
x0[31:7]	Z
[31]	0
[30]	0
[29]	0
[28]	0
[27]	0
[26]	0
[25]	0
[24]	0
[23]	Z
[22]	Z

Fig-5:Ladner_fisher Model

The timing study for several design paths is shown in the delay report. Slack, levels, routes, high fanout, total delay, logic delay, and net delay are all taken into consideration while evaluating each path. Path 1 has a slack of 6748 and a total delay of 3.557 from input port c to sum(32). Path 2 has a total delay of 6.390 and a slack of 6 from a[19] to sum(31). Path 3 has a total delay of 6.183 and a slack of 5 from 3 (24) to sum (30). Path 4 has a total delay of 5.802 and a slack of 5 from b(22) to sum (29). The path 5 from (317) to sum(18) has a delay of 4.659 total and a slack of 3.

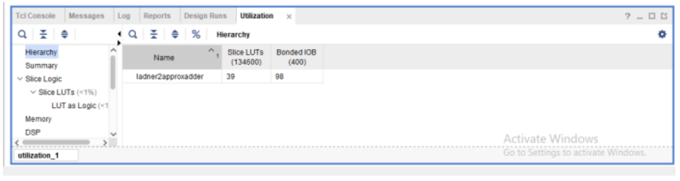


Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clo
Path 1	∞	5	6	4	a[19]	sum[32]	6.746	3.557	3.189	∞	input port c
Path 2	∞	5	6	4	a[19]	sum[31]	6.390	3.557	2.833	∞	input port c
Path 3	∞	5	6	3	a[24]	sum[30]	6.183	3.578	2.605	∞	input port c
Path 4	∞	5	6	4	b[22]	sum[29]	5.802	3.560	2.242	∞	input port c
Path 5	∞	3	4	3	b[17]	sum[18]	4.659	3.368	1.291	∞	input port c
Path 6	∞	3	4	2	a[1]	sum[1]	4.659	3.368	1.291	∞	input port c

Fig-6:Delay Report

To make sure that the specifications are fulfilled from the input ports to the designated amounts, each path is examined. Optimising the design for effective functionality and performance is the main objective.98 bonded IOBs (Input/Output Blocks) and 39 Slice LUTs are used by the Ladner2ApproxAdder module. This allotment is shown in the area report, which shows how the FPGA's resources were used to construct this particular module.



Name	Slice LUTs (134600)	Bonded IOB (400)
ladner2approxadder	39	98

Fig-7: Area report

The design's RTL schematic, which makes use of 98 bonded IOBs and 39 Slice LUTs, demonstrates effective resource utilisation. This succinct illustration emphasises how the Ladner2ApproxAdder module is implemented optimally, guaranteeing efficient use of FPGA resources for high-performance computation.

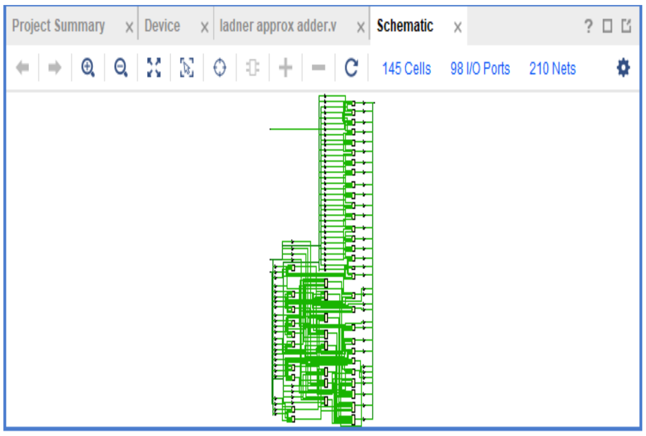


Fig-8:RTL schematic

	Existing	Proposed
Area	71	39
Delay	7.509	6.789

Table-1:Result

With a smaller area of 39 as opposed to the previous design's 71, the suggested design provides notable gains over the current one. Furthermore, the suggested design outperforms the current design with a delay of 6.789 against 7.509, indicating improved efficiency and performance.

12. CONCLUSION

In summary, there are significant improvements over the current designs with the suggested FPGA implementation of the high-speed and area-efficient approximation parallel prefix adder architecture. The suggested design provides notable gains in area utilisation and delay performance by combining cutting-edge methodologies and optimisation tactics. The suggested architecture shows improved efficiency and effectiveness by drastically cutting the area footprint to 39 and the delay to 6.789, from 71 and 7.509 in the previous designs, respectively. These outcomes highlight how well the suggested method works to overcome the difficulties associated with performing high-speed arithmetic operations in FPGA implementations, opening the door to more potent and effective digital systems.

13. FUTURE SCOPE

Future research in the field of FPGA parallel prefix adder implementations may look at a number of directions for advancement and creativity. First off, further research into innovative approximation methods and optimisation algorithms designed especially for FPGA designs may result in even more reductions in latency and space use, improving performance all around. Furthermore, investigating cutting-edge synthesis and simulation techniques to improve design and expedite development workflows may out to be highly advantageous. Moreover, adding reconfigurable and adaptive elements to the adder design to allow it to dynamically modify its operation according to workload parameters may improve flexibility and adaptability.

14. REFERENCES

1. Pakkiraiah Chakali, Madhu Kumar Patnala, "Design of high speed Ladner - Fischer based carry select adder," IJSCE March 2013.

2. David H.K. Hoe, Chris Martinez, and Sri Jyothsna Vundavalli, "Design and characterization of parallel prefix adders using FPGAs," IEEE, March 2011, Pages 168-172.

3. K. Vitoroulis and A.J. Al-Khalili, "Performance of parallel prefix adders implemented with FPGA technology," IEEE Northeast Workshop on Circuits and Systems, August 2007, pp. 498-501.

4. Haridimos T. Vergos and Giorgos Dimitrakopoulos, "On modulo 2^n + 1 adder design," IEEE Transactions on Computers, February 2012, vol. 61, no. 2.

5. Giorgos Dimitrakopoulos and Dimitris Nikolos, "High-Speed Parallel-Prefix VLSI Ling Adders," IEEE Transactions on Computers.

6. S. Knowles, "A family of adders," Proceedings of the 15th Symposium on Computer Arithmetic, June 2001, pp. 277-281.

7. R. Ladner and H. Fischer, "A regular layout for parallel adders," IEEE Transactions on Computers, March 1982, vol. C-31, no. 3, pp. 260-264.

8. Taeko Matsunaga and Yusuka Matsunaga, "Timing-Constrained Area minimization Algorithm for parallel prefix adders," IEICE Transactions on Fundamentals, December 2007, vol. E90-A, no. 12.

9. Jianhua LiuZhu, Haikun, Chung-Kuan Cheng, John Lillis, "Optimum prefix Adders in a Comprehensive Area, Timing and Power Design Space," Proceedings of the 2007 Asia and South Pacific Design Automation Conference, January 2007, pp. 609-615.

10. S.V. Padmajarani and M. Muralidhar, "Comparison of Parallel Prefix Adders Performance in an FPGA," International Journal of Engineering Research and Development (IJERD), September 2012, vol. 3, no. 6, pp. 62-67.

11. S.V. Padmajarani and M. Muralidhar, "A Hybrid Parallel Prefix Adder for high speed computing," Proceedings of the 7th National Conference on Advances in Electronics and Communications (ADELCO), 2011.

12. S.V. Padmajarani and M. Muralidhar, "A New Approach to implement Parallel Prefix Adders in an FPGA," International Journal of Engineering Research and Applications (IJERA), July-August 2012, vol. 2, no. 4, pp. 1524-1528.

13. Akila, M., Gowribala, C., Shaby, S.M., "Implementation of high speed Vedic multiplier using modified adder," International Conference on Communication and Signal Processing (ICCSP), IEEE, April 2016.

14. Gavali, K.R., Kadam, P., "VLSI design of high speed Vedic multiplier for FPGA implementation," IEEE International Conference on Engineering and Technology (ICETECH), IEEE, March 2016.

15. Ram, G.C., Lakshmanna, Y.R., Rani, D.S., Sindhuri, K.B., "Area efficient modified Vedic multiplier," International Conference on Circuit, Power and Computing Technologies (ICCPCT), IEEE, March 2016.

16. Gaur, N., Tyagi, D., Mehra, A., "Performance comparison of adder architectures on 28 nm FPGA," 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall), IEEE, 2016.