ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2026.0388

FPGA AWARE HYBRID APPROXIMATE PARALLEL PREFIX ADDERS FOR ERROR TOLERANT APPLICATIONS

Shyam Peraka, Vinod Babu Pusuluri, Shanmuga Shruthi Alluru and Vamsi Krishna Sai Avvaru
Department of Electronics and Communication Engineering, Rajiv Gandhi University of Knowledge Technologies, India

Abstract

Traditional parallel prefix adders ensure faster carry propagation but may cause routing issues and additional overhead in interconnect when implemented on FPGAs. In order to overcome such limitations, this paper proposes an FPGA-friendly 32-bit Hybrid Approximate Parallel Prefix Adder (HAPPA), which utilizes routing-balanced prefix adder design approaches as well as placement-based methods to implement the low-power multimedia and image processing systems. The suggested design architecture uses three different computation regions: 13-bit Kogge-Stone prefix adder network for high-speed carry propagation in most significant bits, 15-bit Knowles prefix adder to compromise between routing complexity and delay performance and reference- inspired 4-bit approximate adder for reduced logic and switching activities in least significant bits computation. Localized PBLOCK-based floor planning as well as hierarchy-preserving placement strategy were applied using Xilinx Vivado Design Suite 2023.2. The proposed design architecture was then simulated and implemented on a Xilinx Artix-7 FPGA platform. As a result of this implementation, the propagation delay value of 12.465 ns was obtained by implementing the design using only 165 Lookup Tables (LUTs). The approximate computing framework provides a 99.56% arithmetic accuracy trading minimal precision to eliminate critical path delay and area overheads.

Keywords:

Approximate Computing, Parallel Prefix Adder (PPA), FPGA-Aware Design, Image Fusion, P-Block Floor Planning

1. INTRODUCTION

The growing demand for fast and energy-efficient arithmetic circuits in multimedia, image processing, and digital signal processing (DSP) applications has generated a lot of interest in optimized adder architectures. Binary adders are the basic building blocks of modern digital systems, where the overall performance of the system depends mainly on the arithmetic computation speed and hardware efficiency. Parallel prefix adders (PPAs) are one of the most preferred adder architectures because of their fast carry generation and logarithmic delay nature. The conventional PPAs, like Kogge-Stone, Brent-Kung, and Knowles adders, are known for their high computational speed as they generate the carry in parallel [1], [3], [5]. However, these architectures lead to large interconnection complexity, routing congestion, and wiring overhead during implementation on modern FPGA devices. Routing delay is generally more dominant than logic delay in many FPGA applications, leading to reduced implementation efficiency and increased difficulty in achieving timing closure [10], [11].

Approximate computing has proven to be an effective method to lower the complexity of hardware designs, decrease power dissipation, and reduce switching activity due to computational inaccuracy [6]. In approximate arithmetic circuits, small inaccuracies are introduced into computations to minimize delay, area, and energy usage instead of producing accurate results from the computations performed. Applications like image processing, multimedia processing, and image fusion may tolerate some amount of arithmetic inaccuracy without affecting the final result [8]. Several approximate parallel prefix adders have been proposed in recent years with the goal of improving hardware efficiency and reducing switching activity [4], [7]. However, most existing methods focus on optimization at the logic level and pay less attention to FPGA-aware physical implementation challenges such as routing locality, placement optimization, and congestion-aware floorplanning [9]–[11].

To address these issues, this paper introduces a new Hybrid Approximate Parallel Prefix Adder (HAPPA) architecture for low-power multimedia and image fusion applications. The proposed 32-bit architecture includes a 13-bit Kogge-Stone prefix network, a 15-bit Knowles prefix network, and a 4-bit approximate adder designed for low-significant-bit operations. Additionally, in our design, PBLOCK-based floorplanning and hierarchy-preserved placement methods are used to improve placement locality and optimize critical path delay during FPGA implementation. The HAPPA architecture is implemented using a Xilinx Artix-7 FPGA platform and the Vivado 2023.2 tool. As shown in experimental results, the proposed architecture considerably minimizes delay time and maximizes LUT usage without compromising on accuracy. This clearly shows that the proposed architecture is very reliable in providing efficient solutions in hardware and can be considered for multimedia and image processing.

The proposed HAPPA architecture uses a 32-bit Datapath to align with modern industry standards. Modern computer vision and machine learning inference engines heavily standardize on 32-bit precisions to avoid data underflow and truncation errors during multi-stage matrix math. Using 32-bit precision for our hardware means we can achieve seamless compatibility with current GPUs and accelerators.

1.1 MAIN CONTRIBUTIONS

- A 32-bit FPGA Aware Hybrid Approximate Parallel Prefix Adder (HAPPA) combining Kogge-Stone and Knowles prefix structures.
- A hybrid prefix partitioning methodology that balances logic depth, placement locality, and timing performance during FPGA implementation.
- Localized placement organization based on PBLOCK algorithm to enhance timing of data path in FPGA-based systems.
- FPGA implementation and experimental results on Xilinx Artix-7 using Vivado 2023.2 tool for evaluation.

2. RELATED WORK

2.1 EXACT PARALLEL PREFIX ADDER

Due to their excellent carry propagation abilities, outstanding logarithmic delays, and high efficiency in carry generation, PPAs are frequently utilized in contemporary high-speed arithmetic circuits. A widely utilized architecture for constructing prefix adders today is The Kogge–Stone Adder (KSA), widely recognized for its minimal logic depth and high-speed carry propagation among standard parallel prefix topologies [3]. Nonetheless, the significant number of prefix nodes and interconnections in KSA designs leads to considerable routing overhead and wiring congestion when deployed in FPGAs. Brent-Kung Adders (BKA) employ a balanced prefix computation tree to minimize routing overhead and fanout among their prefix node interconnections, resulting in significant area efficiency compared to the nearly identical propagation time of KSA [5]. Likewise, Knowles adders present a space-efficient option to BKAs by allowing for flexible configurations of their prefix structure, facilitating a balance between minimized fan-out and processing speed [1]. Numerous scholars have assessed the performance of exact prefix adders by comparing Kogge-Stone, Brent-Kung, Han-Carlson, and hybrid prefix architectures. For instance, [9] offered a comparative analysis of the delay, area, and power traits of these architectures. Although parallel prefix adders exhibit best possible precision of computation and fastest carry propagation speeds when implemented using an ASIC technology, they tend to perform poorly on FPGAs. Due to their highly interconnected nature, parallel prefix adders experience heavy routing congestion and difficult placement issues when mapped into standard logic (instead of utilizing FPGA vendor-specific carry chains).

2.2 HYBRID APPROXIMATE ADDERS

Hybrid prefix adders combine various parallel prefix architectures to balance processing speed, silicon area, power dissipation, and routing complexity. Unlike standard adders that use a single prefix network throughout, hybrid designs divide the carry path into distinct functional segments. The architectural splitting reduces wiring congestion and logic depth while keeping carry propagation fast across the critical paths.

Many hybrid prefix structures have been developed to improve arithmetic performance. Because interconnect delays often dominate overall timing in modern FPGA fabrics, embedding hardware-aware routing constraints early in the design cycle is vital [2]. Following this paradigm, [13] designed hybrid prefix adders for Residue Number System (RNS) applications by blending Ling and Kogge–Stone topologies, achieving better area-delay trade-offs than conventional exact parallel prefix adders (PPAs). While their approach focuses on optimizing a Ling-based carry look-ahead structure, the architecture proposed in this work combines Knowles and Kogge–Stone structures to target specific prefix truncation goals. This custom hybrid configuration fits well with FPGA architectures because it localizes carry propagation and minimizes long-line interconnect requirements.

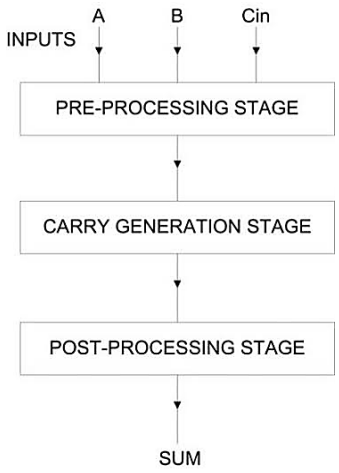


Fig.1. Parallel Prefix Adder Architecture

The creation of approximate computing circuits has become popular because of its efficiency in hardware design for applications that can tolerate errors, including multimedia processing, machine learning, and image fusion tasks [6]. Utilizing managed inaccuracies in arithmetic functions results in reduced hardware complexity, decreased switching frequency, lower power consumption, and minimized propagation delay. Multimedia applications, for instance, can withstand minor inaccuracies in arithmetic operations without considerable quality degradation, thus highlighting the significance of designing approximate adders in contemporary low-power computing. [4] introduced an AxPPA approximate parallel prefix adder circuit that minimizes logic complexity and energy consumption by simplifying the prefix computation process.[7] introduced approximate prefix circuits that are energy-efficient for applications in low-power computing. Approximate prefix architectures have been utilized in image processing to enhance hardware performance while maintaining an acceptable quality of images. Many previous approximate adders were developed by focusing on logic optimization instead of considering FPGA-specific physical concerns.

2.3 FPGA AWARE ARCHITECTURES

In contemporary FPGA systems, routing latency plays a major role in the total propagation delay due to extensive interconnection networks and uneven placement designs. As a result, techniques for FPGA-aware optimization, like timing-driven placement, congestion-aware routing, PBLOCK floorplanning, and hierarchy-preserved implementation, have gained significance in realizing high-performance arithmetic circuits [10], [11]. [10] introduced the Flow-Map algorithm aimed at optimizing delays in FPGA implementations that use lookup tables, highlighting the significance of routing-aware synthesis to enhance timing performance. In a similar vein, [11] proposed timing-driven placement and routing techniques aimed at enhancing routing locality and achieving timing closure in FPGA designs. Multiple recent FPGA designs of parallel prefix adders have shown that placement locality and floorplanning greatly affect propagation delay and routing congestion [12]. Despite these advancements, there is a lack of research on FPGA-aware approximate hybrid parallel prefix adders that concurrently enhance implementation time, area efficiency, and computational

precision. This study introduces an FPGA-focused Hybrid Approximate Parallel Prefix Adder (HAPPA) design that integrates hybrid prefix calculations, approximate arithmetic, and P-Block based localized organization methods to enhance the efficiency of FPGA implementations for low-power multimedia and image fusion tasks.

In this regard, the hybrid architecture using both Kogge-Stone Adder and Knowles Adder is faster in carry propagation than typical hybrid adders, with improved placement and balanced prefix distribution during FPGA implementation. Though KS architecture guarantees fast performance by low-depth logic design, KW architecture cuts down on interconnection overhead as well as the number of prefix nodes.

3. PROPOSED ARCHITECTURE

3.1 PROPOSED HAPPA ARCHITECTURE

The current work presents an FPGA friendly architecture of a Hybrid Approximate Parallel Prefix Adder (HAPPA). This architecture has been designed using carry propagation with timing efficient and approximate operations to enhance performance and reduce energy consumption on FPGAs. The hybrid architecture combines three computational functions: 13-bit Kogge-Stone Adder, 15-bit Knowles Adder and Existing 4-bit Approximate Unit [4].

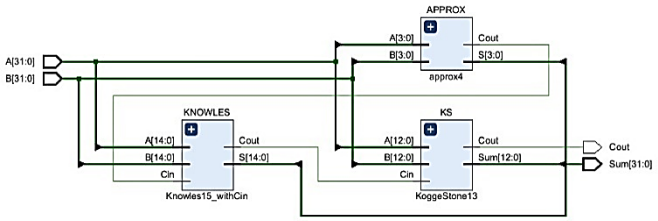


Fig.2. RTL of Proposed Architecture

The complete architecture is the implementation of a hybrid prefix structure consisting of a total of 32 bits, with 2 adder topologies designated per the importance of the computation. To balance parallel carry generation speed with localized routing density, the structural architecture of the 32-bit adder has been explicitly partitioned into three operational bit-ranges: Bits 31–19 use the 13-bit Kogge-Stone Adder, Bits 18–4 use the 15-bit Knowles Adder and Bits 3–0 use the 4-bit Approximate Unit.

The overall purpose of this hybrid partitioning technique is to provide improved speed associated with propagation while improving the balance between logic depth, placement locality and overall timing performance. The elements designated as the most significant are based on the use of the Kogge-Stone Adder due to its minimal logic depth and rapid carry propagation. However, Kogge-Stone structures can generate a substantial amount of routing overhead associated with the dense prefix interconnections.

Therefore, the intermediate bits are implemented using the Knowles Adder due to its balanced fanout and decreased wiring complexity compared to the standard Kogge-Stone structure. To reduce the area utilization of circuits and switching activity, the four least significant bits (LSBs) can be approximated because image processing and multimedia applications are capable of tolerating minor computational errors [15].

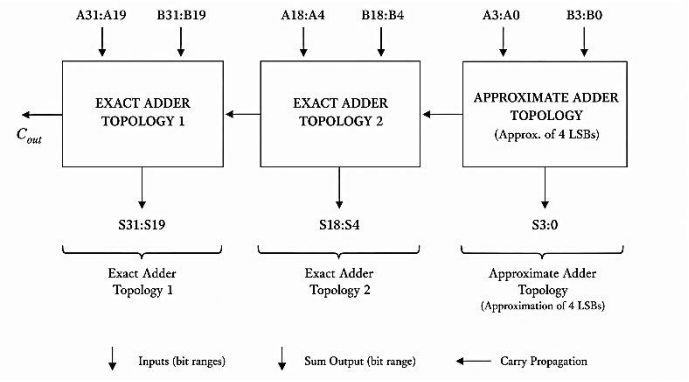


Fig.3. Bit Partition of Proposed Architecture

The existing approximate computation [4] applies strictly to the bits of lower significance because the arithmetic error does not visibly degrade the reconstructed image quality of lower significance. The approximation unit also decreases the complexity of carry propagation logic, along with reducing the number of logical transitions, therefore increasing overall power efficiency and delay performance. Propagation signal: $P_i = A_i \oplus B_i$. and Generation signal: $G_i = A_i \cdot Bi$. Approximate the sum equation as follows: $S_0 = P_0$, $S_1 = P_1 \oplus G_0$, $S_2 = P_2 \oplus G_1$, $S_3 = P_3 \oplus G_2$ and $C_{out} = G_3$.

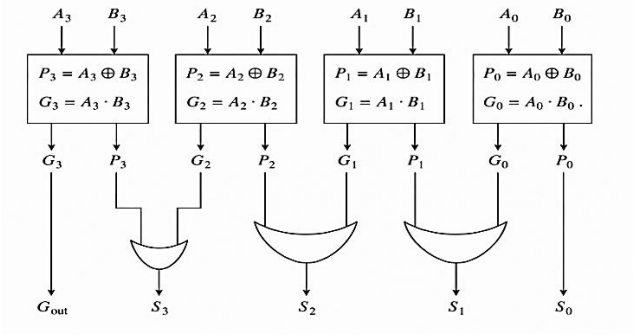


Fig.4. Approximation Block

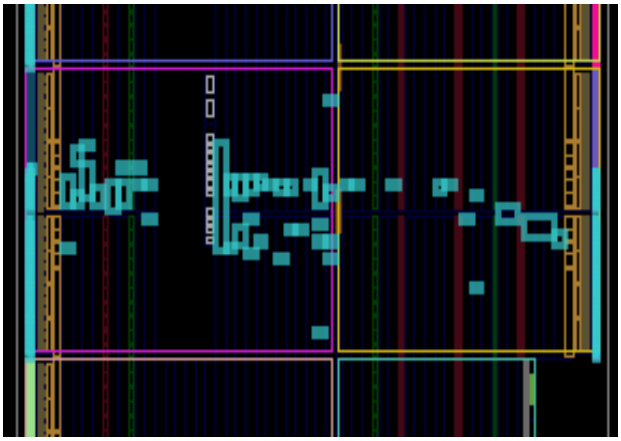
The carry outputs from each computational block will be routed through to the next stage, and final sum outputs will be generated by performing an XOR on both propagate and carry signals. The overall hybrid architecture thus offers an integrated FPGA oriented towards a combination of high-speed exact computation, balanced routing structures, and approximate arithmetic.

3.2 FPGA AWARE FLOOR PLANNING

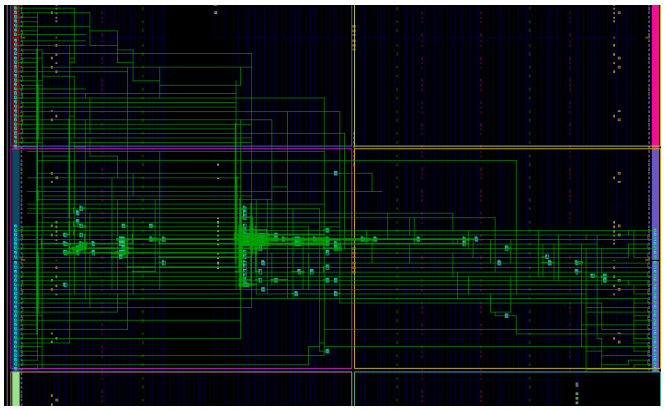
The routing delay plays a large part in the total propagation delay in FPGA designs due to long interconnect paths and irregular placement structures. To enhance routing locality and to lower the amount of interconnect overhead, FPGA-awareness was introduced into the proposed HAPPA architecture through optimization techniques. PBLOCK-based floorplanning constraints were implemented within the Xilinx Vivado environment to localize independent computational modules into compact Configurable Logic Block (CLB) regions.

The Kogge-Stone, Knowles, and Approximation modules were placed using hierarchy-preserving implementation techniques to minimize long routing tracks between prefix computation stages. The localized approach helps to improve placement locality as well as helps in organizing essential computational blocks in smaller regions within the FPGA. Even though there is a natural need for denser connections within the parallel prefix architecture, localization of the P-blocks aids in improving the timing closure due to reduced inefficiencies in placing them far apart on the FPGA structure.

Datapath architectures, the proposed FPGA-aware floorplanning methodology can be extended using TCL-based PBLOCK constraint automation in the Vivado design environment. Instead of manually defining placement regions for individual arithmetic modules, hierarchical design blocks can be programmatically assigned to localized PBLOCK regions through automated Xilinx Design Constraint (XDC) generation. This methodology enables scalable placement-aware implementation for higher word-length arithmetic architectures such as 64-bit and 128-bit adders while reducing manual floorplanning complexity.



(a)



(b)

Fig.5. (a) Unconstrained placement generated by Vivado implementation tools, (b) Routing visualization of the proposed HAPPA architecture on Artix-7 FPGA

The physical constraints are represented by Xilinx Design Constraints (XDC) through the use of the resize_pblock command to specify clear physical boundaries on the physical chip. By confining these multi-bit hybrid adders within fixed physical spaces, we force a different routing behavior from the Vivado environment. Although limiting these separate nets to compile within defined physical locations causes a definite increase in routing paths, the effect on the cascading internal gates is to compact them. Compacting the gate-level logic results in less overall gate logic delays, enabling the entire system to reach timing closure much more easily.

The architecture proposed above was created using the Vivado 2023.2 toolset to synthesize a design that runs on a Xilinx Artix-7 FPGA platform and performs an experimental verification of the new design outputs. The experimental verification results achieved a lower routed delay and less LUT usage than was achieved with traditional free implementations. Therefore, the proposed floorplan-testing techniques improve both timing performance and implementation efficiency of high-speed approximate arithmetic architectures on FPGA. The PBLOCK floorplanning method is used by FPGA implementation software to override the usual placement mechanism to gain more control over the important hardware blocks. While automatic placement software tries to optimize the design as much as possible, it sometimes spreads the blocks across different parts of the FPGA, making them more difficult to route. The PBLOCK method helps in keeping critical blocks closer together, resulting in faster routing and better timing.

4. IMPLEMENTATION

4.1 FPGA DESIGN FLOW

The Hybrid Approximate Parallel Prefix Adder (HAPPA) has been developed in Verilog HDLs and built in the Xilinx Vivado 2023.2 design environment. The complete implementation flow of an HAPPA on an FPGA includes six main stages: RTL design; functional simulation; synthesis; placement; routing; and timing verification.

The Kogge-Stone, Knowles, and Approximation modules were initially created as separate modules and verified via behavioral simulation. Following verification, these three modules were combined to make up a complete 32-bit hybrid approximate parallel prefix adder architecture. Functional verification of carry propagation and the final generated sum was achieved through simulation waveforms.

Once the simulation was complete, RTL synthesis of the HAPPA was executed with the intent to create a netlist for placement and routing on a Xilinx Artix-7 FPGA platform. During this step, FPGA-aware PBLOCK floorplanning constraints were applied to the implementation to facilitate improved routing proximity and reduced interconnect overhead between computational blocks. Next, timing reports, congestion analysis reports, and utilization reports were created to evaluate the performance of the implementation.

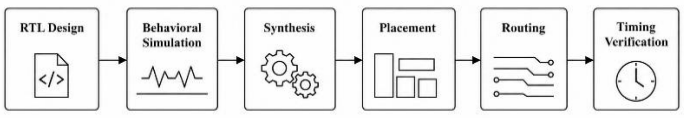


Fig.5. FPGA Design Flow

4.2 EXPERIMENTAL SETUP AND FPGA CONFIGURATION

The proposed architecture has been realized on the Xilinx Artix-7 FPGA using the Vivado 2023.2 tools. The hybrid architecture has been modeled in RTL using Verilog HDL. The implementation target has been the XC7A100TCSG324-1 FPGA device. Constraints on timing and PBLOCK floorplans have been applied during the implementation to assist in improving placement locality and routing quality.

The Kogge-Stone, Knowles, and Approximate modules were synthesized separately from each other and integrated later during top-level implementation. An implementation method that preserved the hierarchy has been used in order to maintain routing regularity and minimize disturbances to placement during optimization.

The experimental data generated from the Vivado implementation tools were used to assess the delay, logic utilization and timing performance.

Also, besides the FPGA implementation, the proposed approximate architecture has been assessed in an application for the purpose of image fusion. The input images have been initially processed using MATLAB to extract the pixel values for placement in the BRAM blocks of the FPGA. The approximate hybrid adder has performed the pixel fusion operations in hardware on the FPGA, and then the resultant fused pixels from the FPGA were returned to MATLAB for the purpose of reconstructing the fused image. The image fusion setup demonstrates that the proposed HAPPA architecture is a valid system in practical multimedia and image processing systems.

Table.1. FPGA Device Specifications

Parameter	Value
FPGA	XC7A100TCSG324-1
Family	Artix-7
Tool	Vivado 2023.2
HDL	Verilog
Carry Primitive	Carry4

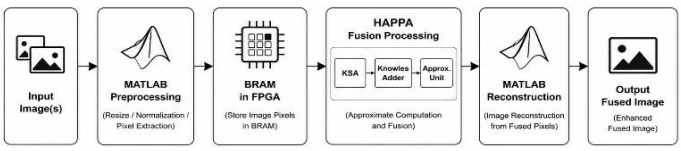


Fig.7. Image Fusion Flowchart

5. RESULTS AND DISCUSSION

5.1 FPGA IMPLEMENTATION RESULTS

Vivado 2023.2 was used to synthesize and implement the proposed FPGA-aware Hybrid Approximate Parallel Prefix Adder (HAPPA) architecture on the Xilinx Artix-7 FPGA. Each implementation was evaluated against LUT utilization, propagation delay, routing congestion, timing characteristics, and approximate computing accuracy. The FPGA-aware PBLOCK floorplanning methodology used in the implementation helped to improve routing locality and improve placement locality and overall timing organization

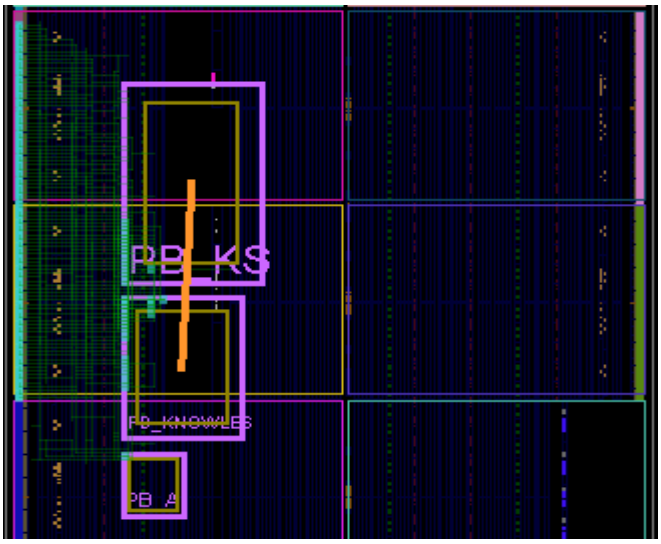


Fig.8. Exploratory PBLOCK-Based Placement Organization of the Proposed HAPPA Architecture on Artix-7 FPGA

The timing analysis indicated that the propagation delay for the proposed FPGA-aware HAPPA architecture was 12.465 ns. In addition, the resulting implementation reports confirmed that timing closure was successful by showing positive timing slack with respect to applied virtual clock constraints. The proposed architecture achieved improved overall Datapath timing characteristics through FPGA aware implementation.

5.2 COMPARATIVE PERFORMANCE ANALYSIS

Performance comparison of the Proposed FPGA-Aware HAPPA architecture with respect to other architectures such as RCA, CLA, and actual Parallel Prefix Adders is carried out. The HAPPA architecture offers considerable savings in terms of area and delay by providing a routed propagation delay of 12.465 with an area utilization of just 165 LUTs. The traditional Exact Kogge-Stone architecture produces high-speed computation, but has a high routing overhead and complexity for the connections when someone uses the FPGA for its implementation. The FPGA-aware implementation of the proposed architecture combines approximate computation with hybrid prefix structures to yield a more balanced performance between delay, area, and routing efficiency.

However, an implementation that is aware of FPGAs enhances timing behavior using a placement organization approach and even distribution of prefixes. While the parallel prefix architecture inherently relies on dense interconnection structures, the use of PBLOCK placements enhances critical path locality and also the timing behavior of the entire Datapath.

Through the design implementation results, it was shown that by utilizing FPGA-aware optimizations, the timing of the implementation was significantly improved due to a reduction of the interconnection overhead. Therefore, the proposed architecture achieves an acceptable trade-off of accuracy, complexity of the hardware, routing efficiency, and implementation delay.

Table.2. Performance Comparison of 32-bit Adder Architectures

Architecture	LUTs	Delay (ns)	Fmax (MHz)	Power (W)	PDP (nJ)
RCA	169	18.552	53.90	0.171	3.1724
CLA	171	17.878	55.93	0.115	2.0059
KSA	229	17.712	56.46	0.165	2.9224
Hybrid	188	13.321	75.07	0.193	2.5709
FPGA-aware Hybrid	165	12.465	80.23	0.115	1.4334

It is observed from the results shown in Table 2 that although the novel architecture of FPGA-Aware HAPPA consumes power similar to conventional Carry Look-Ahead Adder (CLA), which is 0.115W, it has been found that the critical path delay of the architecture is much lower i.e., 12.465ns, thanks to the optimized PBLOCK-based routing. The optimized routing provides reduced delay due to better routing locality and therefore contributes in improved timings when implemented on FPGA. Moreover, it is noted that the Power Delay Product (PDP) achieved by the architecture is 1.433nJ, which is 28.5% more efficient than the conventional CLA approach.

5.3 ERROR METRICS

To evaluate the precision of the suggested approximate design, we utilize these standard points for evaluating approximate computing performance: Mean Error Distance, Normalized Error Distance, and Error Rate. These standard points will also help us provide a quantification of the loss in arithmetic precision that occurs when approximating the least significant bits with our evaluations, which include a sample of randomized 100,000 cases.

In this particular study, ER was determined to be 0.355500, MED to be 3.459080, and NED to be 8.053798×10^{-10}. Nevertheless, it is MRED that should be considered crucial for this particular case, as MRED amounts to 8.1×10^{-7}. The relative value is extremely important when working with 32-bit systems, as the error is estimated relatively to the output value. From the data provided above, it can be seen that, in general terms, the error represents less than 0.0001% of the output value. Since the magnitude of the error obtained with the help of the approximation technique is negligible in comparison with the possible variety of output values in the case of a 32-bit output, one may conclude that the approximate solution obtained is rather precise.

Table.3. Error Metrics of Proposed HAPPA

Metrics	Value
Error Rate (ER)	0.355500
Mean Error Distance (MED)	3.459080
Normalized Error Distance (NED)	8.053798×10^{-10}
Mean Relative Error Distance (MRED)	8.1×10^{-7}
Accuracy	99.56%

Error estimate analysis results indicate that using approximations on the least significant digits will create very small differences between estimated and actual data values, and indicates that the HAPPA Design is suitable for media applications where there will be some degree of error, such as multimedia and image processing.

6. IMAGE FUSION APPLICATION

6.1 HARDWARE-IN-THE-LOOP (HIL) IMAGE FUSION FRAMEWORK

To verify the practical effectiveness of the proposed HAPPA design using real image data, an image fusion system was chosen as a target multimedia application. Image fusion combines multiple source images, such as panchromatic and multispectral data into a single, higher-quality image with enhanced visual characteristics and higher information content. Because most image processing applications are highly tolerant of minor arithmetic errors, approximate hardware techniques are perfectly suited for this type of system. To test this layout, a Hardware-in-the-Loop (HIL) co-simulation setup was created using MATLAB coupled with a physical Xilinx Artix-7 FPGA. This method isolates the core arithmetic verification from external board-level memory access limitations, providing a highly precise measurement of the hardware logic on actual silicon. Figure 7 outlines the data flow process:

- **Preprocessing:** MATLAB handles the initial preprocessing and data sourcing by reading theromatic and multispectral images to extract their raw pixel array information. Each pixel consists of standard 8-bit Red, Green, and Blue (RGB) color values. To fit these into our 32-bit adder, MATLAB combines the three 8-bit color channels into a single 32-bit word by placing them side-by-side and adding zeros to fill the remaining empty bits at the front.
- **Hardware Execution:** The binary vectors are streamed from the PC into the internal Block RAM (BRAM) modules of the FPGA. The 32-bit HAPPA hardware core reads these values synchronously and runs the pixel fusion calculations on the physical fabric. During this process, the HAPPA architecture operates in approximate mode.
- **Post-Processing:** The resulting fused pixels are stored in a secondary destination BRAM block within the FPGA. These values are then read back out to the MATLAB environment. MATLAB runs the final reconstruction scripts to display the fused image and compute its Peak Signal-to-Noise Ratio (PSNR).

Implementing the HAPPA architecture in this HIL environment successfully decreases overall hardware complexity and processing overhead on the FPGA fabric. Because the approximation is strictly confined to the lower-significant bits (LSBs) of the pixel values, the final output image maintains adequate visual quality with very low amount of distortion.

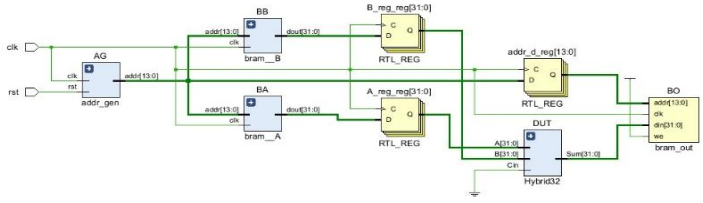


Fig.9. RTL Block Diagram of the FPGA-Based HAPPA Image Fusion Processing System

6.2 EXPERIMENTAL RESULTS

For assessing the performance and accuracy of the proposed HAPPA architecture, an image fusion experiment was performed using exact arithmetic baseline as the basis of comparison for evaluation of the reconstruction accuracy. The results obtained from this experiment clearly indicate that the HAPPA architecture achieves a high level of structure and visual fidelity at significantly lower hardware overhead when compared to traditional exact arithmetic implementation.

From Experiment 1, the PSNR value of the panchromatic (PAN) image with the software fused output (MATLAB simulation) was measured to be around 34 dB. From the hardware implementation of the FPGA image fusion, the PSNR was found to be around 30.38 dB. Since the PSNR is above the critical 30 dB value, the approximation errors can be considered negligible and thus the architecture would prove effective for implementation in a multimedia system.

To evaluate the image processing performance of the proposed HAPPA architecture, its Peak Signal-to-Noise Ratio (PSNR) was compared against several existing approximate adder designs. Traditional error-free adders, such as the Kogge–Stone Adder, yield optimal image quality profiles but suffer from severe routing congestion and high-power consumption implemented on Artix-7 FPGAs. Conversely, low-power alternatives like the AxPPA approach utilize aggressive prefix truncation, which introduces significant arithmetic error and degrades the output PSNR to approximately 26–28 dB [4].

The proposed 32-bit HAPPA design mitigates this quality degradation by confining approximation strictly to the 4 least significant bits (LSBs). By maintaining the precision of the remaining 28 bits through a hybrid Knowles–Kogge–Stone architecture, the HAPPA design achieves a PSNR of 30.38 dB. This demonstrates that the architecture preserves high structural signal integrity while successfully optimizing hardware efficiency for error-tolerant multimedia applications.

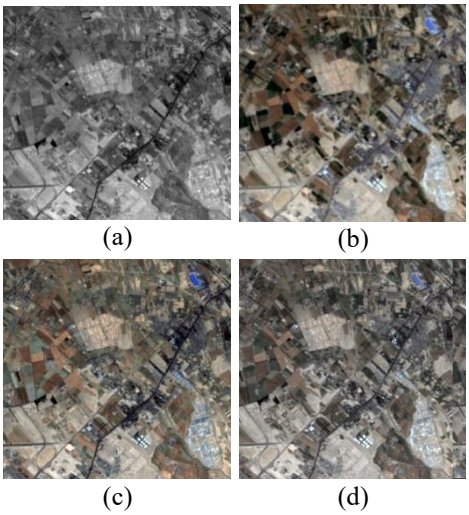


Fig.10. (a) Panchromatic Image, (b) Multispectral Image, (c) MATLAB Generated Fusion Image, (d) Fusion Image generated using HAPPA (FPGA)

7. CONCLUSION

This work presented an FPGA-aware 32-bit Hybrid Approximate Parallel Prefix Adder (HAPPA) architecture for low-power multimedia and error-tolerant applications. The proposed design combines a high-speed Kogge-Stone prefix network, a routing-balanced Knowles structure, using a existing low-complexity approximate computation block to achieve improved timing performance and hardware efficiency during FPGA implementation.

Through the use of PBLOCK-enabled floorplanning and hierarchy-preserving placement techniques, the entire timing performance of the Datapath within the HAPPA architecture was enhanced through placement locality and balanced logic placement.

Accuracy of approximate computing was assessed out by means of four different measures of error: Mean Error Distance (MED), Normalized Error Distance (NED), Error Rate (ER), and Mean Relative Error Distance (MRED). The results of quantitative analysis include ER = 0.355500, MED = 3.459080, and the extremely small NED value of 8.053798×10^{-10}. Importantly, the relative value of MRED equals to 8.1×10^{-7}, taking into account that error is normalized against extremely wide dynamic output range in 32 bits. Additionally, experimental results from image fusion show that the architecture maintains an acceptable Peak Signal-to-Noise Ratio (PSNR), confirming its viability for error-tolerant multimedia tasks. Thus, the HAPPA architecture which has been presented is a good answer for creating FPGA-based systems of arithmetic that can be used for image processing (multimedia) types of applications as well as other types of error tolerant embedded applications. Future work could expand on this architecture by developing higher bit-width implementations, using adaptive approximation methods, and by developing more advanced FPGA routing optimizations.

Future work can extend the proposed FPGA-aware floorplanning methodology toward higher bit-width architectures.

REFERENCES

[1] S. Knowles, “A Family of Adders”, *Proceedings of IEEE Symposium on Computer Arithmetic*, pp. 1-6, 1999.

[2] S. Gupta, A. Ranjan and V.K. Sharma, “FPGA Based Low Power Approximate Hybrid Parallel Prefix Adders with Less Area”, Proceedings of International Conference on Advance Microelectronics, pp. 1-7, 2024.

[3] P.M. Kogge and H.S. Stone, “A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations”, *IEEE Transactions on Computers*, Vol. 22, No. 8, pp. 786-793, 1973.

[4] M.M. Azevedo da Rosa, “AxPPA: Approximate Parallel Prefix Adders”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 31, No. 1, pp. 17-28, 2023.

[5] R.P. Brent and H.T. Kung, “A Regular Layout for Parallel Adders”, *IEEE Transactions on Computers*, Vol. 31, No. 3, pp. 260-264, 1982.

[6] J. Han and M. Orshansky, “Approximate Computing: An Emerging Paradigm for Energy-Efficient Design”, *Proceedings of IEEE Symposium on European Test*, pp. 1-6, 2013.

[7] Z. Yang, "Approximate Parallel Prefix Adders for Energy Efficient Computing", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 25, No. 5, pp. 1789-1800, 2017.

[8] H. Jiang, "Approximate Arithmetic Circuits: A Survey, Characterization and Recent Applications", *Proceedings of the IEEE*, Vol. 108, No. 12, pp. 2108-2135, 2020.

[9] S. Rakesh, "A Comprehensive Review on the VLSI Design Performance of Parallel Prefix Adders", *Materials Today Proceedings*, Vol. 18, pp. 4567-4575, 2019.

[10] J. Cong and Y. Ding, "Flow Map: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 1, pp. 1-12, 1994.

[11] S. Chatterjee, "Timing Driven Placement and Routing for FPGA Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No. 11, pp. 1349-1360, 2002.

[12] K. Vitoroulis and A.J. Al-Khalili, "Performance of Parallel Prefix Adders Implemented with FPGA Technology", *Proceedings of IEEE Northeast Workshop on Circuits and Systems*, pp. 498-501, 2007.

[13] K. Ch and S. Varadarajan, "Area and Delay Efficient Hybrid Prefix Adders for Residue Number System Applications", *Proceedings of International Conference on Communication Systems and Network Technologies*, pp. 1-8, 2023.

[14] J. Anirudh, A. Ravi Sankar Babu, K. Manoj Gopi, S. Karishma and S. Musala, "FPGA Based Low Power Approximate Hybrid Parallel Prefix adders with Less Area", *Proceedings of IEEE International Conference on Information, Implementation, and Innovation in Technology*, pp. 1-6, 2025.

[15] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy and A. Raghunathan, "SALSA: Systematic Logic Synthesis of Approximate Circuits", *Proceedings of 49th Annual Conference on Design Automation*, pp. 796-801, 2012.