

## Design of novel high speed parallel prefix adder

Deepak Kumar Athur<sup>1</sup>, Bhuvanesh Narayanan<sup>2</sup>, Amshuman Gopalakrishnan<sup>2</sup>, Sasipriya Palanisamy<sup>3</sup>,  
Anita Angeline Augustine<sup>3</sup>

<sup>1</sup>School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, United States

<sup>2</sup>School of Electronics Engineering, Vellore Institute of Technology, Chennai, India

<sup>3</sup>Centre for Nanoelectronics and VLSI Design, Vellore Institute of Technology, Chennai, India

### Article Info

#### Article history:

Received Jun 30, 2022

Revised Nov 15, 2022

Accepted Nov 18, 2022

#### Keywords:

Brent-kung  
Han-carlson  
Kogge-stone  
Parallel prefix  
Sparse adder

### ABSTRACT

Adders are crucial logical building blocks found almost in all the modern electronic system designs. In the adder architecture design, the fundamental issue is the propagation latency in the carry chain. As the length of the input operands increases, the length of the carry chain along with it. Parallel prefix adders, which address the problem of carry propagation in adders, are the most efficient adder topologies for hardware implementation. However, delay reduction still could be achieved for very high speed applications. Hence, in this paper design of 16bit novel parallel prefix adder is proposed and compared against the existing parallel prefix adder architectures. The design and simulation are carried out using xilinx vivado for field-programmable gate array (FPGA) simulation and Cadence® for ASIC. The results of ASIC implementation demonstrate 17.8% delay reduction while compared to sparse kogge-stone adder.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



### Corresponding Author:

Anita Angeline Augustine  
Centre for Nanoelectronics and VLSI Design, Vellore Institute of Technology  
Vandalur-Kelambakkam Road, Chennai, India  
Email: anitaangeline.a@vit.ac.in

## 1. INTRODUCTION

Very large scale integrated (VLSI) system design comprise of subsystems that are split into various modules. One such subsystem is the data path unit. Within a microprocessor, the data path elements are the operational unit that performs computing operation based on the instruction. These tasks include memory related access, arithmetic operations like addition, multiplication and logical operations. Although performance optimization can be handled at any design level, it has the greatest influence at a higher level of abstraction/algorithmic level [1]. Practically in all modern processing units, addition is a time-critical operation. The choice of adders for diverse applications such as digital signal processor (DSP) is determined by performance criteria such as the area, adder delay, and power dissipation. The primary purpose of any design is to reduce power consumption while enhancing performance. To achieve performance and limit power dissipation, the adder topology must be deliberately crafted.

The ripple carry adder calculates the carry bit along with the sum bit, and each bit must wait until the previous carry has been computed before it can calculate its own result and carry bits [2], [3]. A carry-look ahead adder boosts performance by minimizing the time it takes to determine carry bits. The parallel prefix adder (PPA) utilizes a three-step structure of the carry-look ahead adder [4]. The carry generation stage is enhanced so that it is parallelizable to reduce time [5]. The advantage of the operation depends on the initial inputs. It involves performing an operation in parallel and segmenting it into smaller chunks that are computed in parallel giving it the name parallel prefix adder. This results in various research avenues for designing parallel prefix adder architectures with reduced power consumption and increased speed of operation. Owing to the wide range of design options for achieving area, power, delay efficiency, as well as optimization without compromising on trade-offs the parallel prefix adders are a good choice [6].

## 2. METHOD

### 2.1. Existing parallel prefix adders

The parallel prefix adder has three stages, the first of which is the pre-processing stage, where Propagate ( $P_i$ ) and Generate ( $G_i$ ) signals are generated as depicted in Figure 1. When the inputs A and B are given, the generate and propagate are calculated as in (1) and (2).  $G_i$  and  $P_i$  indicates whether the carry is generated from that bit or whether carry is propagated from that bit respectively.

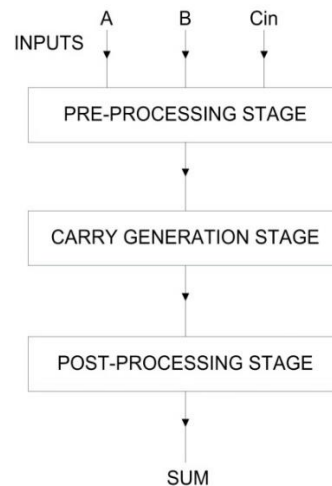


Figure 1. Block diagram of parallel prefix adders

$$G_i = A_i \text{ AND } B_i \quad (1)$$

$$P_i = A_i \text{ XOR } B_i \quad (2)$$

Prefix graph based tree structure is utilized for the PPA carry generation stage [7]. Pairs of generate and propagate signals ( $G_x, P_x$ ), ( $G_y, P_y$ ) are fed into the second stage (carry generation stage). This leads to calculating group generates and group propagate signals ( $G_{x:y}, P_{x:y}$ ), as shown in as in Figure 2 and as depicted in (3) and (4) [8].

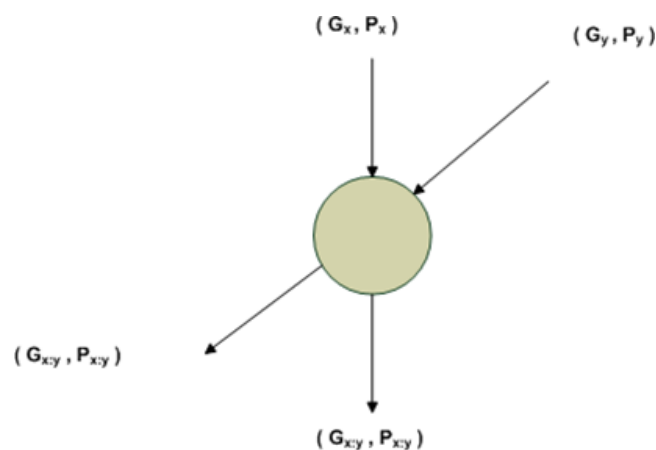


Figure 2. Calculation of carry bit in a prefix graph

The final cell in each bit functions is to provide the carry bit. The last carry bit is enabled in a simultaneous summation of the next bit until the last bit. The carry generate and carry propagate are given in the (3), (4) [9].

$$Gx:y = Gx + (Px.Gy) \tag{3}$$

$$Px:y = Px.Py \tag{4}$$

In post processing stage, the sum is calculated as given in the (5).

$$S_i = P_i XOR G_{i-1:cin} \tag{5}$$

The critical route in parallel adders is determined by the carry from the least significant bit adder to the most significant bit adder, hence reduction of the critical path for the carry to reach the most significant bits (MSB) is necessary [10].

**2.1.1. Brent-kung adder**

A Brent-kung adder is a parallel adder with a layout intended to save chip size and make manufacturing easier. Its symmetry and regular construction structure greatly decreases production costs and allows it to be employed in pipelined topologies. With a chip size of  $O(n \log n)$ , the addition of  $n$ -bit numbers can be done in time  $O(\log n)$ , making it a good choice when there are area limitations while maximizing performance [11].

Figure 3 depicts Brent-kung parallel prefix adder which consists of pre-processing stage (stage 1), carry generation stages (stages 2-7) and post-processing (stage 8). It is one among the advanced adder architecture, simpler in construction and offer significantly less wiring congestion. Hence, it has the lowest wiring tracks, which reduces the amount of space needed to implement the architecture. Furthermore, because there are fewer wires crossing or overlapping each other, routing becomes much easier. However, the penalty is the increase in delay due to increased number of stages. In addition, the fan out of this adder increases, which further increases the delay [12].

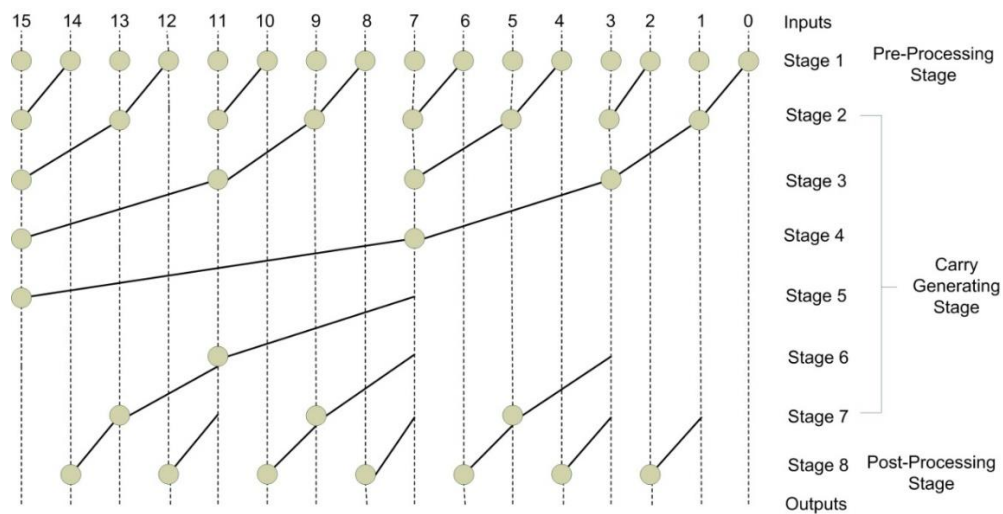


Figure 3. Architecture of Brent-Kung adder

**2.1.2. Kogge-stone adder**

Kogge-stone adder is a fast adder, which produces carry signals in  $O(\log n)$  time [13]. The 16-bit adder is implemented in four layers and hence incurs less delay. The Kogge-stone adder's high speed of operation is due to its minimal logic depth and low fan-out. Because it has the lowest fan-out compared to other approaches, it has the shortest latency, making it ideal for high-speed industrial applications [14].

As in the architecture diagram shown in Figure 4, pre-processing stage (stage 1), carry generating stages (stages 2,3,4), and post-processing stage (stage 5) make up the Kogge-stone parallel prefix adder (5). While compared to other algorithms, low fan-out lowers delay, making it ideal for fast commercial applications. Kogge-stone adders take up a lot of space and have a bunch of overlapping circuitry [15]. With a small rise in the number of bits, the number of dot operators rapidly grows. The carry processing stage provides the carries equivalent to each bit in the carry propagation step. These actions on bits are performed in parallel. This block sets Kingdom of Saudi Arabia (KSA) apart from other adders and is responsible for its outstanding performance [16].

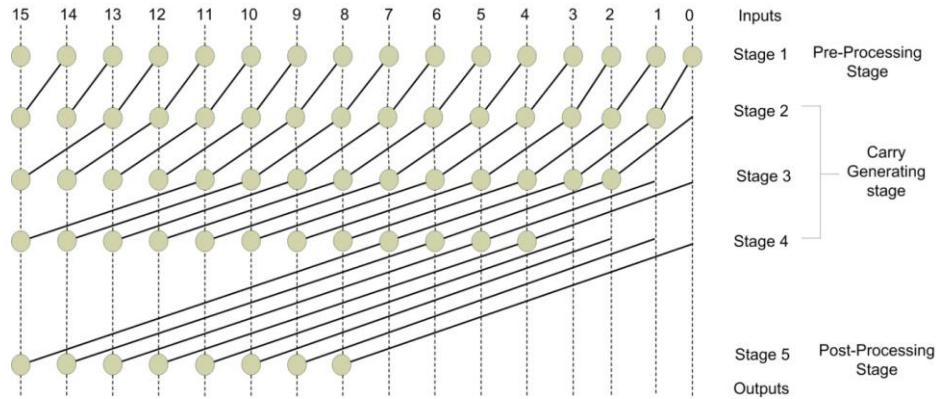


Figure 4. Architecture of kogge-stone adder

**2.1.3. Han-carlson adder**

The Brent-Kung and Kogge-Stone adder principles are combined in the Han-Carlson adder [17]. Carry-merge operations are only performed on even bits in this system. Odd-bit generate and propagate signals are forwarded downwards as depicted in Figure 5. At the end, they recombine with even bits carry signals to generate true carry bits. In comparison to Kogge-Stone, it performs better with lower bit adders. This adder offers an optimal speed with comparatively minimal area and power usage. It's simple to create, with a well-balanced maximum fan out and logic levels [18]. As shown in the architecture diagram shown in Figure 2, the Han-Carlson parallel prefix adder consists of pre-processing stage (stage 1), carry generation stages (stage 2-5) and post-processing (stage 6).

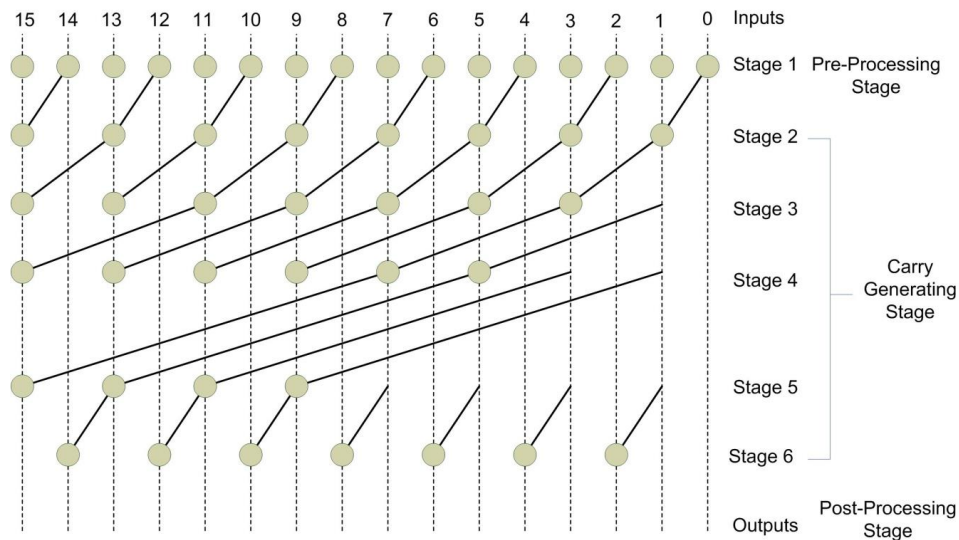


Figure 5. Architecture of han-carlson adder

**2.1.4. Sparse-4 kogge-stone adder**

The tree structure of the 16 bit sparse Kogge-Stone adder is similar to that of the Kogge-Stone adder, with the exception that every fourth carry bit is generated while the rest of the tree is skipped. The terminology "adder sparsity" refers to the number of carry bits produced by the carry-tree. Since in our sparse Kogge-Stone every 4th bit is generated, it is called sparse-4 Kogge-Stone adder. It ends with ripple carry adders that provide the final sum [19]-[21]. In Figure 6, the black cells calculate both generate and propagate whereas the grey cells calculate only the generate bit.

As we can see from Figure 6 stage 1 is the pre-processing stage where generate and propagates are calculated using (1) and (2) respectively. The stages 2 to 5 are carry generation stages which make use of the

(3) and (4). The final computation stage provides the sum with help of 16 ripple carry adders. The Sparse Kogge-Stone adder reduces the amount of wiring junctions required for implementation without sacrificing much of processing speed. It also enhances the adder’s power and area consumption [22], [23].

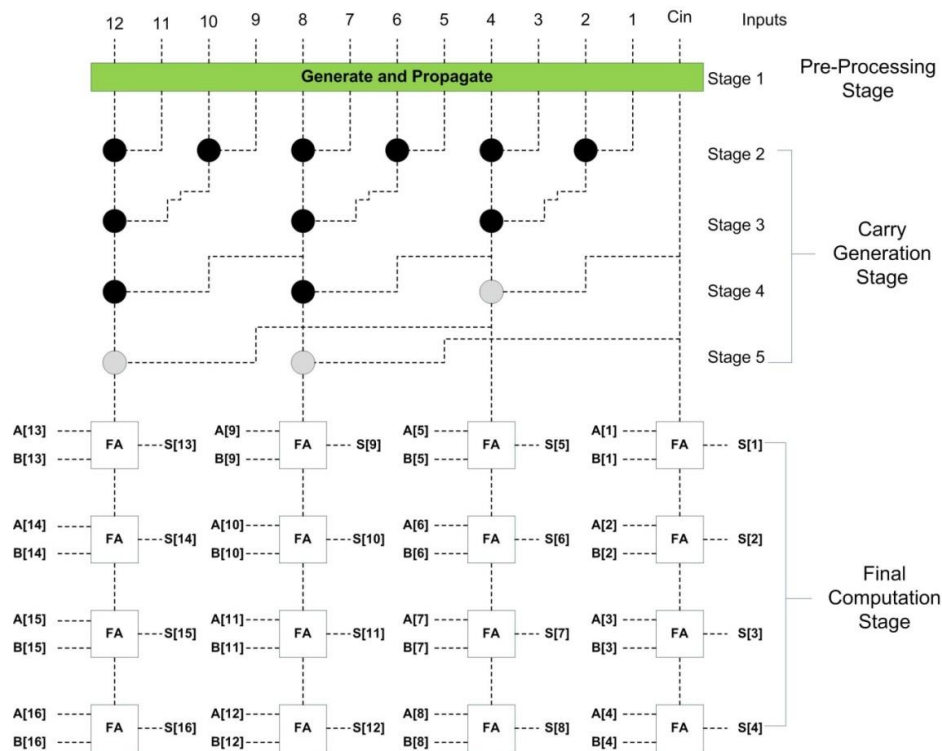


Figure 6. Tree structure of sparse-4 kogge-stone adder

**2.2. Proposed delay optimized sparse-4 kogge-stone adder**

The design’s significant potential to improve area and power consumption was demonstrated by the sparse-4 kogge- stone adder. The goal was to attain performance characteristics that were in between those of the kogge-stone and han- carlson adders. The kogge-stone adder’s high speed could be partially sacrificed for better power, area and wire congestion [24]. This is achieved by improving the speed of sparse-4 kogge-stone adder which already exhibits low area and power consumption. The number of stages in a parallel prefix adder directly influences the speed of the adder. Therefore, as seen in Figure 7 the first stage of the sparse-4 kogge-stone adder was removed and the black cell 1, 2 and 3 were calculated directly. The black cell calculates both generates and propagates whereas the grey calculates only the generates. The equation for generate and propagate of black cell 1 is given as:

$$G4: 1 = G4 + P4 . G3 + P4 . P3 . G2 + P4 . P3 . P2 . G1 \tag{6}$$

$$P4: 1 = P4 . P3 . P2 . P1 \tag{7}$$

Similarly the other 2 black cells in this stage can be calculated and they all have a fan in of 4. The third stage of the sparse-4 kogge-stone adder was also removed except gray cell 1. The gray cell 2 was calculated by using black cell 2 and grey cell 1 with a fan in of two instead of calculating with the help of black cell 2, black cell 1 and cin (as done in original sparse-4 KSA)which would have had a fan in of 3. The equation for gray cell 2 is given as:

$$G8: cin = G8: 5 + ( P8: 5 . G4: cin ) \tag{8}$$

And the equation of gray cell 3 is given as:

$$G12: cin = G12: 9 + P12: 9 . G8: 5 + P12: 9 . P8: 5 . G4: cin \tag{9}$$

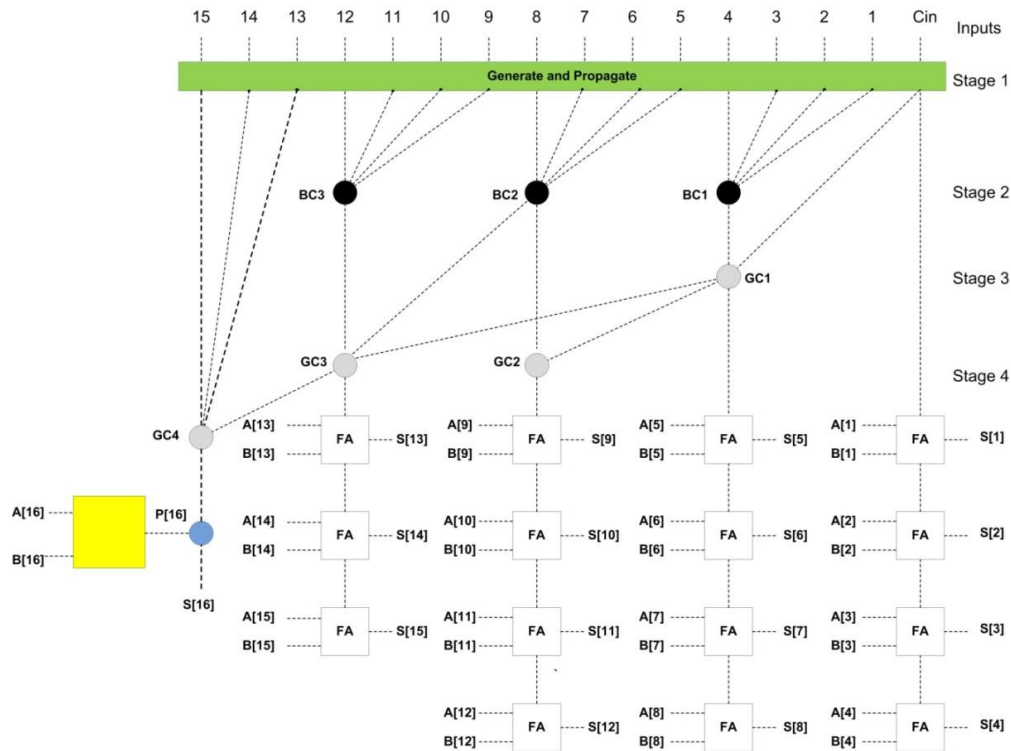


Figure 7. Structure of proposed delay optimized sparse-4 kogge-stone adder

The calculation of the 16<sup>th</sup> sum bit was problematic as it had the highest delay being at the end of the carry chain and ripple carry adders. To tackle this, grey cell four was implemented which directly calculated the carry required for calculation of sum 16. It is calculated after stage 4 using grey cell 3 and generates and propagates from preprocessing stage. The equation for gray cell four is given as:

$$G15: cin = G15 + P15 . G14 + P14 . G13 . P15 + P15 . P14 . P13 . G12: cin \tag{10}$$

After this equation is obtained, the P [16] required for the calculation of sum using (5) is calculated in the yellow box. This and the carry given by the (10) are used to calculate sum 16 directly which reduces delay significantly.

### 3. RESULTS AND DISCUSSION

The parallel prefix adders mentioned were simulated in xilinx vivado using spartan-7 field-programmable gate array (FPGA) xc7s50fpga484-2. For the power simulation, default values were used in the software. The outputs of the simulated 16-bit adders using ModelSim are shown in Figures 8-12.

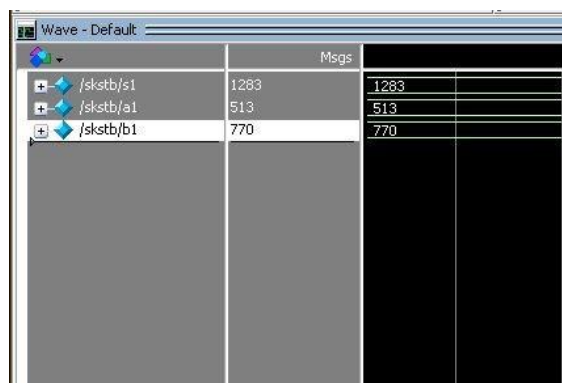


Figure 8. FPGA Simulation result of sparse 4 kogge-stone adder

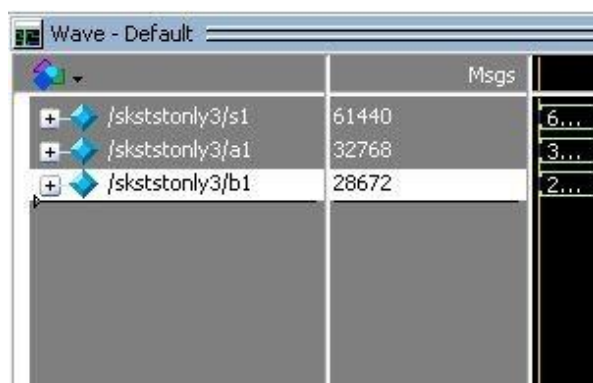


Figure 9. FPGA Simulation result of proposed delay optimized sparse 4 kogge-stone adder

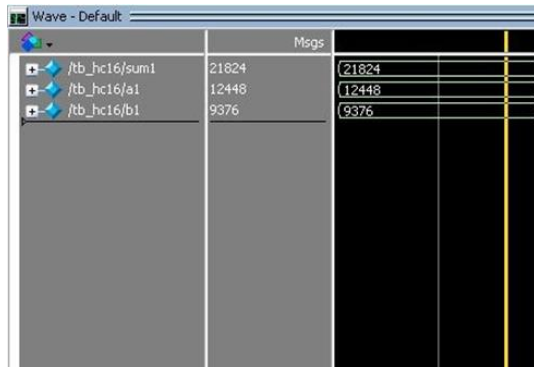


Figure 10. FPGA simulation result of han-carlson adder



Figure 11. FPGA simulation result of kogge-stone adder

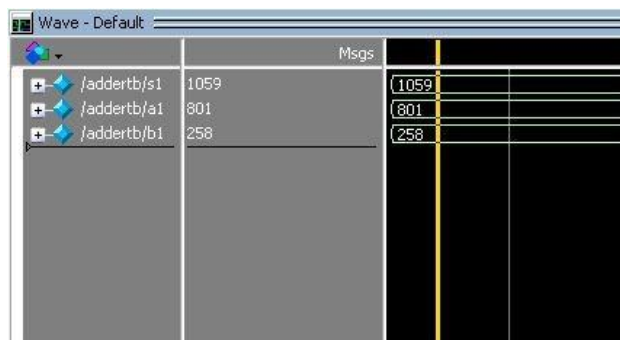


Figure 12. FPGA Simulation result of kogge-stone adder

The results depicted in Table 1 show that in the FPGA implementation, kogge-stone is the fastest adder which is 13.3% faster than han-carlson and 13.1% faster than brent-kung. However, it consumes 55.5% more area than han-carlson and 44.8% more than brent-kung. It also consumes 44.3% more power than the other two adders. The FPGA implementation of sparse 4 kogge-stone adder is actually faster than the kogge-stone adder while it is expected to be slower. This could be likely due to the impact of the routing overhead [25]. The modified sparse 4 kogge-stone adders is faster than sparse kogge-stone by 5.4% while consuming the same area and power as that of the sparse kogge-stone adder. ASIC implementation results are being shown below.

Table 1. FPGA Results of 16-BIT parallel prefix adders

Parallel Prefix Adder	Slice LUTs	Logic Power (mW)	Total Delay (ns)
Kogge-Stone	42	257	10.501
Han-Carlson	27	178	12.122
Brent-Kung	29	178	12.078
Sparse-4 Kogge-Stone	28	168	10.123
Proposed Sparse-4 Kogge-Stone	28	168	9.570

The ASIC circuits have also been built and analyzed using Cadence® Genus using 180nm technology. The results are shown in Figures 13-15. As we can see kogge-stone is faster than han-carlson by 49% and brent-kung by 50.9%. However, it consumes 54.3% more area than han-carlson and 50.7% than brent-kung. It also consumes 31.5% more power than han-carlson and 28.2% more than brent-kung. The sparse kogge-stone adder is slower than the kogge-stone adder by 87.5% while consuming 26.5% less area and 8.8% less power as expected. The proposed delay optimized sparse kogge-stone adder is faster than sparse kogge-stone adder by 17.8% and slower than kogge-stone adder by 54% while consuming 2.2% less power than sparse kogge-stone and 10.7% less power than kogge-stone. It also consumes 9.7% area than sparse kogge-stone and 19.3% less area than the kogge-stone. The proposed delay optimized sparse-4 kogge-stone adder performance is in between kogge-stone and han-carlson adder.

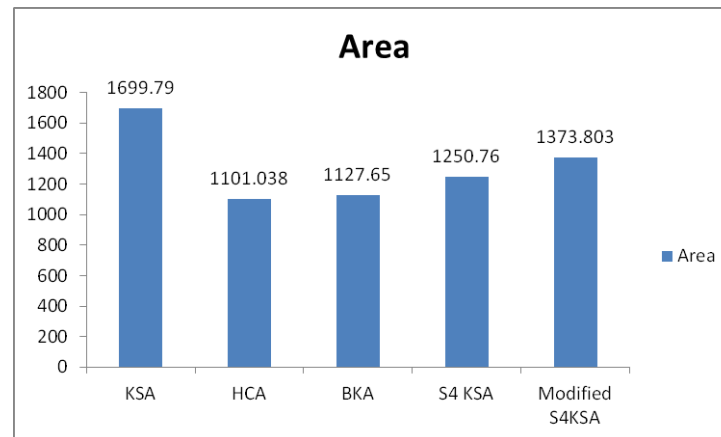


Figure 13. Comparison of 16-bit adder area - ASIC implementation

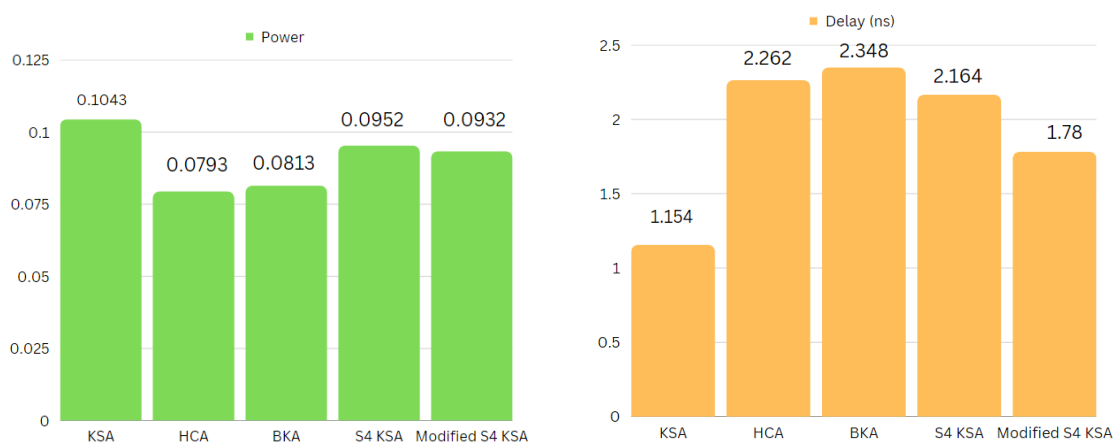


Figure 14. Comparison of 16-bit adder power (mW)- ASIC implementation

Figure 15. Comparison of 16-bit adder delay (ns)-ASIC implementation

#### 4. CONCLUSION

The performance analysis of various 16-bit parallel prefix adders such as han-carlson, brent-kung, and kogge-stone are performed using FPGA platform and ASIC synthesis. The ASIC was performed using TSMC 180nm technology. The simulation results depict Kogge-Stone adder is the fastest adder with increased power and area consumption. Han-carlson adder occupied less area than brent-kung but was slower in the FPGA implementation. The proposed delay optimized sparse-4 kogge-stone adder demonstrates 17.8% increase in speed performance over sparse-4 kogge-stone adder and 10.7% power reduction while compared with kogge-stone adder. It would be worthwhile to implement custom adders using sparsity-2 and implementing sparsity on other adders.

#### REFERENCES





- [1] H. E. Neil, D. M. Weste, and Harris, "Datapath subsystems," in *CMOS VLSI Design: A Circuits and Systems Perspective*, Pearson Publications 2004.
- [2] D. H. K. Hoe, C. Martinez, and S. J. Vundavalli, "Design and characterization of parallel prefix adders using FPGAs," in *2011 IEEE 43rd Southeastern Symposium on System Theory*, pp. 168–172, 2011, doi: 10.1109/SSST.2011.5753800.
- [3] N. Zamhariu, P. Voon, and K. Kipli, "Comparison of parallel prefix adder," *Proceedings of the World Congress on Engineering*, vol. 2, 2012.
- [4] S. K. Yezerla and B. Rajendra Naik, "Design and estimation of delay, power and area for Parallel prefix adders," *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014, pp. 1-6, doi: 10.1109/RAECS.2014.6799654.
- [5] S. Gedam, P. Zode, and P. Zode, "FPGA implementation of hybrid Han- Carlson adder," *2014 2nd International Conference on Devices, Circuits and Systems (ICDCS)*, 2014, pp.1–4, doi: 10.1109/ICDCSyst.2014.6926185.






- [6] Y. Choi and E. E. Swartzlander, "Parallel prefix adder design with matrix representation," *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*, 2005, pp. 90-98, doi: 10.1109/ARITH.2005.35.
- [7] K. C. Shilpa, M. Shwetha, B. C. Geetha, D. M. Lohitha, and N. V. Pramod, "Performance analysis of parallel prefix adder for datapath VLSI design," *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 1552-1555, 2018, doi: 10.1109/ICICCT.2018.8473087.
- [8] A. Raju and S. K. Sa, "Design and performance analysis of multipliers using Kogge Stone Adder," *2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2017, pp. 94-99, doi: 10.1109/ICATCCCT.2017.8389113.
- [9] G. Thakur, H. Sohal, and S. Jain, "A novel parallel prefix adder for optimized Radix-2 FFT processor," *Multidim Syst Sign Process*, vol. 32, pp. 1041-1063, 2021, doi: 10.1007/s11045-021-00772-1.
- [10] N. Banerjee, C. Augustine, and K. Roy, "Fault-Tolerance with Graceful Degradation in Quality: A Design Methodology and its Application to Digital Signal Processing Systems," *IEEE International Symposium on, Defec and Fault Tolerance of VLSI Systems*, pp. 323-331, 2008, doi: 10.1109/DFT.2008.43.
- [11] K. S. Pandey, D. K. B. N. Goel, and H. Shrimali, "An ultra-fast parallel prefix adder," *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 125-134, 2019, doi: 10.1109/ARITH.2019.00034.
- [12] R. A. Patel, M. Benaissa, and S. Boussakta, "Fast parallel-prefix architectures for modulo  $2n-1$  addition with a single representation of zero," *IEEE Transactions on Computers*, vol. 56, pp. 1484-1492, 2007, doi: 10.1109/TC.2007.70750.
- [13] S. Roy, "Towards optimal performance-area trade-off in adders by synthesis of parallel prefix structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 1517-1530, 2014, doi: 10.1145/2463209.2488793.
- [14] S. Banerjee and W. Rao, "A general design framework for sparse parallel prefix adders," *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 231-236, 2017, doi: 10.1109/ISVLSI.2017.48.
- [15] B. Harish, K. Sivani, and M. S. S. Rukmini, "Design and performance comparison among various types of adder topologies," *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 725-730, 2019, doi: 10.1109/ICCMC.2019.8819779.
- [16] M. Bhavani, M. S. Kumar, and K. S. Rao, "Delay comparison for  $16 \times 16$  vedic multiplier using RCA and CLA," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, no. 3, p. 1205, 2016, doi: 10.11591/ijece.v6i3.pp1205-1212.
- [17] B. Harish and M. S. S. Rukmini, "Ultra high speed full adder for biomedical applications," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 10, no. 1, p. 25, 2021, doi: 10.11591/ijres.v10.i1.pp25-31.
- [18] Y.-T. Pai and Y.-K. Chen, "The fastest carry lookahead adder," *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, pp. 434-436, 2004, doi: 10.1109/DELTA.2004.10071.
- [19] C.-J. Fang, C.-H. Huang, J.-S. Wang, and C.-W. Yeh, "Fast and compact dynamic ripple carry adder design," *Proceedings. IEEE Asia-Pacific Conference on ASIC*, 2002, pp. 25-28, doi: 10.1109/APASIC.2002.1031523.
- [20] R. W. Doran, "Variants of an improved carry look-ahead adder," in *IEEE Transactions on Computers*, vol. 37, no. 9, pp. 1110-1113, 1988, doi: 10.1109/12.2261.
- [21] R. A. Javali, R. J. Nayak, A. M. Mhetar, and M. C. Lakkannavar, "Design of high-speed carry save adder using carry lookahead adder," *International Conference on Circuits, Communication, Control and Computing*, pp. 33-36, 2014, doi: 10.1109/CIMCA.2014.7057751.
- [22] A. Naik, D. Deka, and D. Pal, "ASIC implementation of high-speed adaptive recursive karatsuba multiplier with square-root-carry-select-adder," *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, pp. 1-4, 2020, doi: 10.1109/LASCAS45839.2020.9068973.
- [23] G. C. Ram, D. S. Rani, R. Balasaikesava, and K. B. Sindhuri, "Design of delay efficient modified 16 bit Wallace multiplier," *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2016, pp. 1887-1891, doi: 10.1109/RTEICT.2016.7808163.
- [24] Y. D. Ykuntam, K. Pavani, and K. Saladi, "Design and analysis of High speed wallace tree multiplier using parallel prefix adders for VLSI circuit designs," *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.2020.9225404.
- [25] K. Nehru, A. Shanmugam, and S. Vadivel, "Design of 64-bit low power parallel prefix VLSI adder for high-speed arithmetic circuits," *2012 International Conference on Computing, Communication and Applications*, 2012, pp. 1-4, doi: 10.1109/ICCCA.2012.6179204.

## BIOGRAPHIES OF AUTHORS






**Deepak Kumar Athur**     is Pursuing is master's degree in computer engineering (Electrical Engineering) at Arizona State University, with his undergraduate degree Electronics and Communication Engineering graduate from VIT Chennai. His interests include VLSI Physical design and Computer Architecture. He can be contacted at email: deepak.deepak851@gmail.com.






**Bhuvanesh Narayanan**    is a B.Tech. Electronics and Communication Engineering graduate from VIT Chennai. His interests include VLSI and chip design. He can be contacted at email: bhuvaneshnarayanan11@gmail.com.



**Amshuman Gopalakrishnan**    is a B.Tech Electronics and Communication Engineering graduate from VIT Chennai. He is interested in VLSI and Embedded Systems. He can be contacted at email: amshuman.gopal2018@vitstudent.ac.in.



**Sasipriya Palanisamy**    is an Associate Professor, Centre for Nanoelectronics and VLSI Design at VIT, Chennai. She obtained her undergraduate degree from Bharathiar University (Tamilnadu) and her M.E degree in Applied Electronics from PSG College of Technology and Ph.D. from VIT Chennai. She has 20 years of teaching and research experience. Her specializations include Low Power VLSI Circuit Design, Computer Architecture and Embedded system design. She can be contacted at email: sasipriyap@yahoo.com.



**Anita Angeline Augustine**    is currently working as an Associate Professor, in Centre for Nanoelectronics & VLSI Design, VIT Chennai. She obtained her BE degree and ME degree in Applied Electronics from Karunya Institute of Technology, Coimbatore and PhD in the area of VLSI Design from VIT, Chennai, India. She has teaching experience and research experience of more than 2 decades. Her research areas include the design of high speed arithmetic structures using dynamic logic circuits and architectural modifications. She can be contacted at email: anitaangeline.a@vit.ac.in.