Performance analysis of parallel prefix adders developed with field programmable gate array technology

Masood Ahmad Mahammad[1], Appala Raju Uppala[2], Suggala Ram Prasad[1], Anusha Marouthu[3]
[1]Department of Electrical, Electronics, and Communication Engineering, GITAM Deemed to be University, Hyderabad Campus Telangana, India
[2]Department of Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Hyderabad, India
[3]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vijayawada, India

Article Info

Article history:

Received Jun 29, 2023
Revised Aug 18, 2024
Accepted Aug 30, 2024

Keywords:

Adder structure
Area
Field programmable gate array
Performance (speed)
Power
Prefix adders

ABSTRACT

In many digital systems like high-performance computing and digital signal processing, parallel prefix adders are vital. Field programmable gate array (FPGA) technology is a well-known platform for developing parallel prefix adders. FPGA performance depends on bit size of the adder, the adder structure chosen, and the implementation specifications. An examination of the performance and area of parallel prefix adders developed using FPGA technology is presented in this research work. We look into how different design factors, such as the adder structure and the number of input bits, affect the performance and area of parallel prefix adders. The different adders used are Sklansky, Kogge-Stone, Brent-Kung, Han-Carlson, and Ladner-Fisher adders. These adders are implemented using Verilog hardware description language (Verilog HDL) on FPGA boards. The performance is significantly influenced by choice of adder structure and design factors optimized for area or performance. The suggestions for choosing the best adder structure and design factors for the best performance or optimized area are obtained from the synthesis results. Ladner-Fisher adders is best parallel prefix adder with respect area and performance compared with the Sklansky, Kogge-Stone, Brent-Kung and Han-Carlson. Our synthesis can be used as a guide for designers looking to construct specific hardware on FPGA.

Corresponding Author:

Masood Ahmad Mahammad
Department of Electrical, Electronics, and Communication Engineering
GITAM Deemed to be University, Hyderabad Campus
Hyderabad, Telangana, India
Email: mmahamma@gitam.edu, masoodahmad80@gmail.com

1. INTRODUCTION

Parallel prefix adders are a type of adder that can execute logarithmic add on multiple operands. In high-speed arithmetic circuits, parallel prefix adders are commonly utilized. The primary reason is that parallel prefix adders can handle big data sets rapidly and efficiently.

A parallel prefix adder comprises a series of complete adders connected in a tree-like layout. The bottom row of the tree receives the input operands, and the outputs of each level are supplied as inputs to the following level. Each level computes a partial sum and a carry bit, which are propagated to the next level until the final total and carry bits are computed at the tree's top. The Sklansky, Kogge-Stone, Brent-Kung, Han-Carlson, and Ladner-Fisher algorithms can be used to build parallel prefix adders. Each technique employs a unique approach to optimizing the carry propagation network and reducing the number of full adders required. The high area and delay of exact parallel prefix adders restrict their performance in high-speed applications. As a result, parallel prefix adders are critical components in high-speed arithmetic circuits. They are useful for applications such as digital signal processing, multimedia, encryption, and scientific computing because they allow efficient and quick addition of multiple operands.

In 1960, Robert E. Sklansky invented the Sklansky adder, a parallel prefix adder [1]. It is a very efficient and expandable adder topology that works well with parallel technologies such as field programmable gate array (FPGA). The Sklansky adder groups input bits and process them in a series of steps, each performing a partial addition. The basic structure of the Sklansky adder comprises several stages, each with its carry lookahead logic. Each group created from the input bits comprises a power of two bits. Eight bits can be split into two groups of four bits each if the adder accepts eight bits as input. The carry lookahead circuitry calculates the carry bits for each group during the first stage's independent processing of each group. The partial sums for each group are combined with the carry bits from the previous stage, and processing is repeated. The process is repeated until all partial sums have been added to determine the total amount.

Scalability is an advantage of the Sklansky adder. As long as the bit count is a power of two, it can accept inputs of any size. Sklansky adder with bit count is a power of two, making it a popular option for hardware implementations that use parallelism to achieve high performance, like FPGAs. The Sklansky adder carry lookahead logic is used at every stage. Therefore, the Sklansky adder has low latency. The Sklansky adder, however, has a few drawbacks. One drawback is that it necessitates a lot of carry lookahead logic, which might make the adder bigger and use more power. The sequential treatment of each group of bits also diminishes the adder's parallelism. The Sklansky adder, in conclusion, is an extremely efficient and scalable adder structure that is perfect for hardware implementation in parallel. It uses carry lookahead logic and partial addition steps to produce minimal latency and great performance. Conversely, employing carry lookahead logic and sequential processing may restrict parallelism, increase system size, and use more power. Figure 1 shows the 16-bit Sklansky adder schematic diagram.

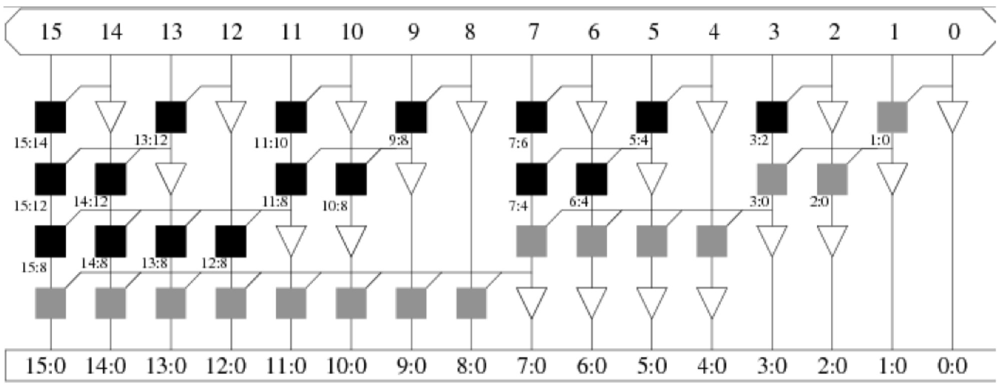


Figure 1. 16-bit Sklansky adder schematic diagram

The parallel prefix adder, the Kogge-Stone adder, was first introduced in 1973 by Kogge and Stone [2]. This adder topology is effective, scalable, and functions well with similar technology like FPGAs. The Kogge-Stone adder's basic structure comprises several stages, each performing a partial addition on the input bits. Each stage is organized as a binary tree with a unique lookahead mechanism. The input bits are split into pairs in the first step, and each pair is added to provide a partial sum and carry output. In the second stage, the first stage's partial sums are connected in pairs and added once more to produce another set of partial sums and carry outputs. This process is repeated until the output and final result are acquired by adding all component sums. One advantage of the Kogge-Stone adder is its high level of parallelism. The parallel processing of numerous bits at each stage allows for rapidly adding large quantities. Additionally, each stage's binary tree structure and carry lookahead logic offer minimal latency and excellent performance. However, the Kogge-Stone adder has a lot of limitations. It has the disadvantage of requiring a lot of carry lookahead logic, which could increase the size and power consumption of the adder. Implementing the binary tree structure in hardware could also be difficult. In conclusion, the Kogge-Stone adder is a scalable adder structure that is extremely effective and well-suited for implementation on parallel hardware. Although it enables fast performance and little delay, the adder's use of a binary tree structure with carry lookahead logic increases its size and power usage.

Brent and Kung [3] invented the Brent-Kung adder in 1982, a parallel prefix adder. It is extremely effective and scalable for hardware implementations, like in FPGAs. The input bits are initially split into smaller groups for the Brent-Kung adder's recursive divide-and-conquer strategy, which combines the input bits gradually to get the final total and carry output. A high level of parallelism is possible with the divide-and-conquer strategy since each stage can process several bits concurrently. Each level of the Brent-Kung adder uses half as many bits as before. The input bits are split into pairs and added in the first stage to create a collection of partial sums and carry outputs. The first stage's partial sums are joined in pairs and added again in the second stage to create another set of partial sums and carry outputs. Each subsequent stage of this process is repeated until the final sum and output are produced by adding all the component sums.

The effectiveness and scalability of the Brent-Kung adder are two benefits. High levels of parallelism are possible using the divide-and-conquer strategy, enabling quick addition of big numbers. The adder's structure is also very regular, which makes it simple to construct in hardware. The Brent-Kung adder has some drawbacks, including the need for extensive wiring to link the multiple stages together. The extensive wiring may increase the adder's size and energy usage, especially for greater quantities of bits. In conclusion, the Brent-Kung adder is a parallel prefix adder that is extremely effective, scalable, and ideally suited for hardware implementation. Due to the intricacy of the wiring, even though it uses a recursive divide-and-conquer strategy to achieve high levels of parallelism, the adder's size and power consumption may also increase.

Han and Carlson [4] first presented the Han-Carlson adder, a parallel prefix adder, in 1993. It is made effective and scalable for use in hardware implementations like FPGAs. In the Han-Carlson adder's recursive technique, the input bits are divided into groups and merged in stages to produce the final sum and carry output. The recursive approach has a high level of parallelism because each stage can handle many bits simultaneously. A sequence of steps makes up the Han-Carlson adder. Each of them uses a certain subset of the input bits. The input bits are divided into groups at each level and processed concurrently to produce a collection of partial sums and carry outputs. After combining all of the component's sums to produce the final sum and carry output, the separate sums are blended again in the following stage. The effectiveness and scalability of the Han-Carlson adder are two advantages. The recursive approach allows for great parallelism and rapid addition of large numbers. Furthermore, the structure of the adder is fairly regular, making it straightforward to implement in hardware. However, one shortcoming of the Han-Carlson adder is that it necessitates extensive wiring to connect the various stages. The extensive wiring can increase the adder's space and power consumption, especially for greater bits. Finally, the Han-Carlson adder is a parallel prefix adder that is very effective, scalable, and well-suited for hardware implementation. Although adopting a recursive approach allows for high levels of parallelism due to the wiring complexity, it can also increase the adder's size and power consumption.

In 1960, Ladner and Fischer [5] developed the Ladner-Fischer adder, a parallel prefix adder [6]-[19]. It is made effective and scalable for use in hardware implementations like FPGAs. The Ladner-Fischer adder uses a recursive method: the input bits are split into groups and then sequentially joined to create the final sum and carry output. Due to each stage's ability to process several bits simultaneously, the recursive technique provides a high parallelism level. The Ladner-Fischer adder is a series of steps, each employing a different subset of the input bits. The input bits are divided into two groups and processed concurrently in each stage to produce a set of partial sums and carry outputs. Once all partial sums have been merged to generate the final sum and carry output, the different sums are blended again in the following stage. The Ladner-Fischer adder has two advantages: efficacy and scalability. The recursive approach allows for great parallelism and rapid addition of large numbers. Furthermore, the structure of the adder is fairly regular, making it straightforward to implement in hardware. On the other hand, the Ladner-Fischer adder's capacity to generate redundant carry signals may increase the adder's size and energy usage. Because a lot of wire is needed to connect the adder's various stages, adding more bits can increase the size and power consumption.

Last, the parallel prefix adder [20]-[25], the Ladner-Fischer adder, is effective, scalable, and well-suited for hardware implementation. Last, the parallel prefix adder, the Ladner-Fischer adder, is effective, scalable, and well-suited for hardware implementation. Because of its recursive method, it can achieve high degrees of parallelism, although wiring complexity might result in redundant carry signals, increasing the adder's size and power consumption. Finally, the Ladner-Fischer adder is a parallel prefix adder that is efficient, scalable and well-suited for hardware implementation. Because of its recursive method, it can achieve high degrees of parallelism, although wiring complexity might result in redundant carry signals, increasing the adder's size and power consumption. Finally, the Ladner-Fischer adder is a parallel prefix adder that is efficient, scalable and well-suited for hardware implementation. Because of its recursive method, it can achieve high degrees of parallelism, although wiring complexity might result in redundant carry signals, increasing the adder's size and power consumption.

2. METHOD

The Xilinx Vertex 5 device is selected as the target FPGA device for the parallel prefix adder implementation. Due to its numerous internal resources and cutting-edge technology, this device is picked. The XC5VLX220 has several features, including 65 nm technology. There are six input look-up tables in its configurable blocks. These configuration blocks are referred to as slices. Synthesis and implementation are done using the Xilinx ISE software. Two runs of the software tools were organised to allow for the proper optimization. The synthesis and implementation strategies were designed to optimise for area in the first run, and speed in the second. The optimization attempts of the tools were "high" in both circumstances. The findings are presented in the results and discussion section.

3. RESULTS AND DISCUSSION

The results are shown in Tables 1 to 4. The terms BK is short form of the Brent-Kung adder, HC is short form of the Han-Carlson adder, KS is short form of the Kogge-Stone adder, LF is short form of the Ladner-Fischer, and SK is short form of the Sklansky adder are used throughout the table. The area is measured in slice lookup tables units, which are programmable logic units in the FPGA. The time difference is measured in nanoseconds. Experimental results: Series 1 is a 16-bit adder. Series 2 is a 32-bit adder. Series 3 is a 64-bit adder. Series 4 is a 128-bit adder. Series 5 is a 256-bit adder.

Table 1. Area results (area optimization)

Bit size	Prefix adders LUTs used in FPGA					
	SK	KS	BK	HCK=1	HCK=2	LF
16	54	54	65	58	57	61
32	140	298	124	225	166	140
64	282	898	322	545	389	317
128	619	2295	624	1335	934	605
256	1447	5891	1366	3229	2072	1207

Table 2. Area results (performance optimization)

Bit size	Prefix adders LUTs used in FPGA					
	SK	KS	BK	HCK=1	HCK=2	LF
16	19	19	19	19	35	18
32	51	136	34	100	83	52
64	134	449	72	236	195	141
128	296	1111	204	614	432	301
256	649	2584	586	1335	995	652

Table 3. Performance results in nano seconds (area optimization)

Bit size	Parallel prefix adders					
	SK	KS	BK	HCK=1	HCK=2	LF
16	12.7	12.9	12.9	13.2	10.8	12.2
32	17.3	17.8	20.3	19.8	15.6	17.1
64	25.8	30.5	29.3	28.3	34.5	21.5
128	41.6	57.3	70	67.3	64.7	36.3
256	74.7	106.7	92.7	105.8	118	69

Table 4. Performance results in nano seconds (speed optimization)

Bit size	Parallel prefix adders' type					
	SK	KS	BK	HCK=1	HCK=2	LF
16	10.5	10.5	10.2	11.1	10.8	11
32	11.8	13.4	17.5	16.2	13.2	11.5
64	17.3	19	26.6	19.6	16.7	15.4
128	26.1	29.2	41.8	31.4	28.3	31.3
256	30.1	42.7	62.2	45.1	36.2	27.3

Tables 1 and 2 show the results of the area and speed optimization software, respectively. These tables show how the area optimization method produces significantly smaller adders than the speed optimization method. Table 1 shows how frequently adder zones match the characteristics of their type. The Brent kung is the smallest adder, while the Kogge stone is the largest. The results of the Han Carlson adder, which fall between the Brent kung and the Kogge stone, show a more compact k=2 structure. The Sklansky and Ladner fisher types are unexpected in this chart because the Lend fisher was predicted to have less area. Figure 1 is created using data from Table 1. Figure 2 is created using data from Table 2.

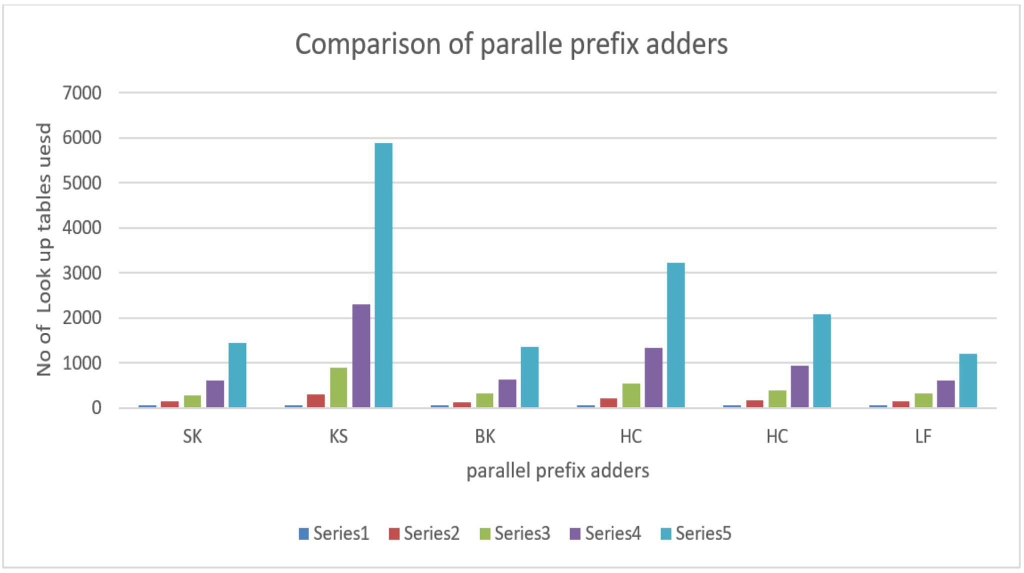


Figure 2. Number of lookup tables used in FPGA with area optimization

Tables 3 and 4 show that when the optimization target was set to speed, the critical delay of each adder was greatly reduced. Table 3 shows that the adders have significant critical path delays as a result of the software's efforts to fit the synthesised circuits as tightly as possible within the FPGA's limits. The Brent kung adder outperforms the Han Carlson and Kogge stone adders in this table, which is an unusual case. The prefix tree of the Brent kung adder, on the other hand, has a more complex structure. Another surprising result is that, despite having shallow prefix trees, the Ladner-Fischer and Sklansky adder adders outperform the Kogge-Stone adders. Adders frequently perform worse as the number of interconnects in their prefix networks grows. Table 3 data is used to generate Figure 3.

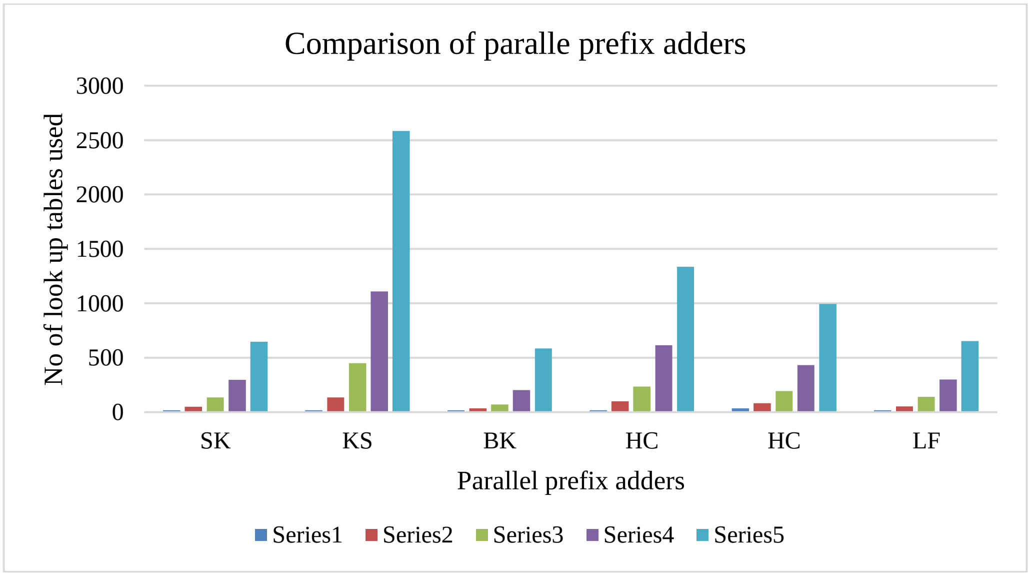


Figure 3. Number of lookup tables used in FPGA with speed optimization

Table 4 shows the rate at which all adders improve their speed. Given their common characteristics, each adder in this table performs as expected. We can see that for 256-bit adders, adders with a high interconnect count perform slower than adders with the same critical path length but fewer interconnects. According to the table, the 256-bit Ladner-Fischer adder is slightly faster than the 128-bit version. Nothing in the two adders' prefix tree networks can account for this result. As a result, it is possible that it is linked to tool-dependent resource allocation and optimizations. Table 4 data was used to create Figure 4. Table 5 is used to develop the Figure 5. The Figure 5 shows delay measured in nano seconds in parallel prefix adders when they are optimized for the speed while synthesized in the FPGA.

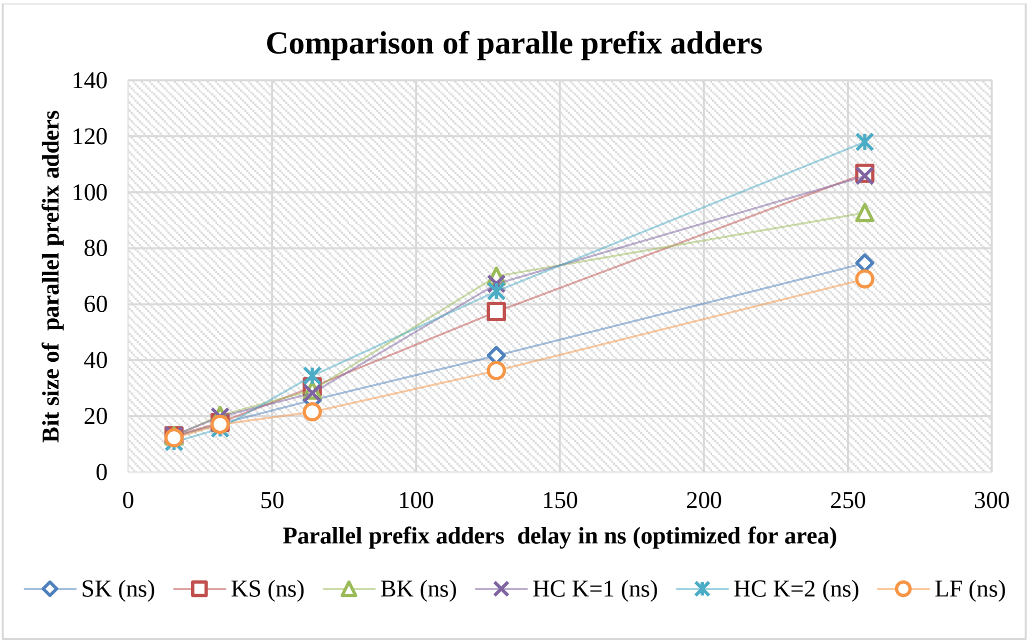


Figure 4. Performance of parallel prefix adders with area optimization

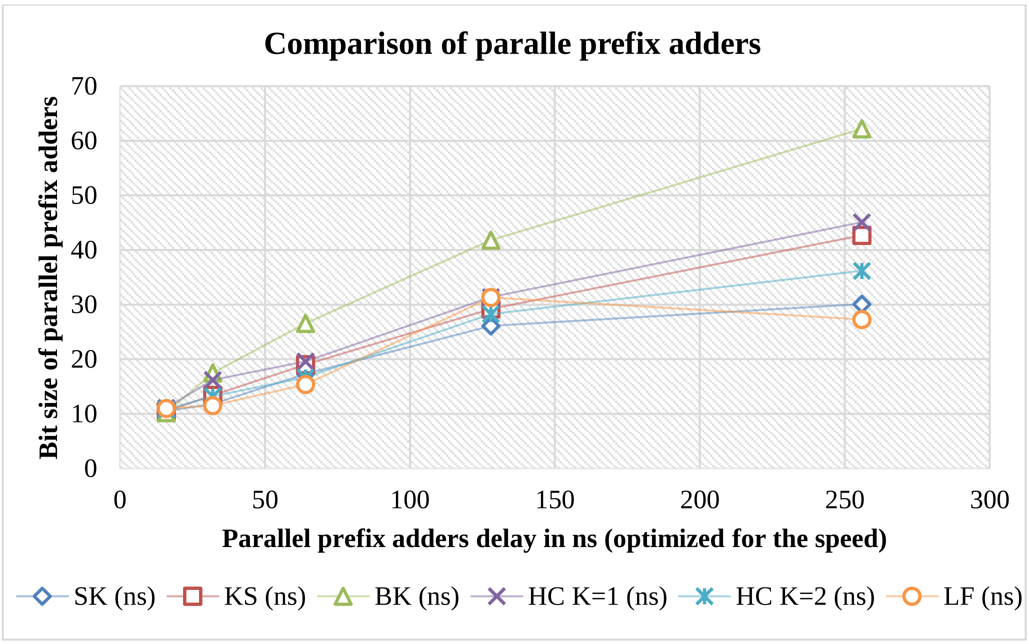


Figure 5. Performance of parallel prefix adders with speed optimization

4. CONCLUSION

In conclusion, parallel prefix adders are widely used in computationally demanding applications, including digital signal and image processing. The parallel prefix adder's performance is obtained with implementation on FPGA. The Sklansky adder, Brent-Kung adder, Kogge-Stone adder, Han-Carlson adder, and Ladner-Fischer adder are the parallel prefix adders we covered in this study. We evaluated the performance of parallel FPGA-based prefix adders in this study. The results suggest that the key factors influencing adder performance were software optimization settings, effective resource allocation, and the use of specialised FPGA resources. In several cases, software tool optimizations caused some adders to lose their algorithm dominance. PPA design elements include area, logic depth, connection count, and fan-out. The results of parallel prefix adders demonstrate that the number of interconnects has the largest influence on both area and speed performance. The 256-bit Ladner-Fischer is the area efficient and fastest parallel prefix adder compared to the Sklansky, Kogge-Stone, Brent-Kung, and Han-Carlson.

REFERENCES

[1] J. Sklansky, "Conditional-sum addition logic," *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, 1960, doi: 10.1109/TEC.1960.5219822.

[2] 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. C–22, no. 8, pp. 786–793, 1973, doi: 10.1109/TC.1973.5009159.

[3] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers,* vol. C–31, no. 3, pp. 260–264, 1982, doi: 10.1109/TC.1982.1675982.

[4] T. Han and D. A. Carlson, "Fast area-efficient VLSI adders," in *1987 IEEE 8th symposium on computer arithmetic (ARITH)*, 1987, pp. 49–56, doi: 10.1142/9789814651578.

[5] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 831–838, 1980, doi: 10.1145/322217.322232.

[6] D. Harris, "A taxonomy of parallel prefix networks," *The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 2213–2217, 2003, doi: 10.1109/acssc.2003.1292373.

[7] G. Dimitrakopoulos and D. Nikolos, "High-speed parallel-prefix VLSI ling adders," *IEEE Transactions on Computers*, vol. 54, no. 2, pp. 225–231, 2005, doi: 10.1109/TC.2005.26.

[8] A. Beaumont-Smith and C. C. Lim, "Parallel-prefix adder design," in *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001*, 2001, pp. 218–225.

[9] H. Zhu, C.-K. Cheng, and R. Graham, "Constructing zero-deficiency parallel prefix adder of minimum depth," in *Proceedings of the 2005 Asia South Pacific Design Automation Conference*, 2005, pp. 883–888, doi: 10.1109/aspdac.2005.1466481.

[10] X. Zhang, X. Zhou, and X. Wu, "Design and implementation of parallel prefix adder on FPGA," in *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, 2016, pp. 157–162.

[11] Q. Zheng and Y. Wu, "A high-performance hybrid parallel prefix adder for FPGAs," *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1253–1262, 2018.

[12] X. Gao and L. Xie, "Design and analysis of a high-performance parallel prefix adder based on FPGA," in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 3127–3130.

[13] B. Majhi and S. Dehuri, "FPGA implementation of parallel prefix adders: A comparative analysis," *International Journal of Electronics*, vol. 105, no. 4, pp. 652–669, 2018.

[14] Z. Ling and H. Du, "Implementation of a high-performance parallel prefix adder on FPGA," in *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2017, pp. 1–6.

[15] W. Wu, C. Xu, X. Li, and Y. Liu, "Design and optimization of high-performance parallel prefix adders on FPGAs," *Journal of Systems Architecture*, no. 107, p. 101774, 2020.

[16] 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)*, Kyoto, Japan, 2019, pp. 125-134, doi: 10.1109/ARITH.2019.00034.

[17] Y. Shao, X. Zhang, and C. Wang, "Design and implementation of a high-performance parallel prefix adder on FPGA," *Journal of Electronic Science and Technology*, vol. 17, no. 2, pp. 162–169, 2019.

[18] F. Jafarzadehpour, A. S. Molahosseini, A. A. E. Zarandi, and L. Sousa, "New energy-efficient hybrid wide-operand adder architecture," *IET Circuits, Devices and Systems*, vol. 13, no. 8, pp. 1221–1231, Nov. 2019, doi: 10.1049/iet-cds.2019.0084.

[19] S. Muthyala Sudhakar, K. P. Chidambaram and E. E. Swartzlander, "Hybrid Han-Carlson adder," *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Boise, ID, USA, 2012, pp. 818-821, doi: 10.1109/MWSCAS.2012.6292146.

[20] D. Esposito, D. De Caro, M. De Martino, and A. G. M. Strollo, "Variable latency speculative Han-Carlson adders topologies," *2015 11th Conference on Ph.D. Research in Microelectronics and Electronics, PRIME 2015*, pp. 45–48, 2015, doi: 10.1109/PRIME.2015.7251090.

[21] R. Adusumilli and V. Kumar, "Design and Implementation of a high speed 64 bit Kogge-Stone adder using Verilog HDL," *International Journal of Electrical and Electronic Engineering and Telecommunication (IJEETC)*, vol. 4, no. 1, pp. 2319–2518, 2015.

[22] S. Banerjee and W. Rao, "A general design framework for sparse parallel prefix adders," *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, pp. 231–236, 2017, doi: 10.1109/ISVLSI.2017.48.

[23] D. Esposito, D. De Caro, and A. G. M. Strollo, "Variable latency speculative parallel prefix adders for unsigned and signed operands," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1200–1209, 2016, doi: 10.1109/TCSI.2016.2564699.

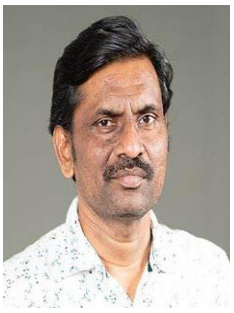
[24] D. Esposito, D. De Caro, E. Napoli, N. Petra, and A. G. M. Strollo, "Variable latency speculative Han-Carlson adder," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 5, pp. 1353–1361, 2015, doi: 10.1109/TCSI.2015.2403036.

[25] P. Singh, A. Kumar, and A. Kumar, "Low power high-speed approximate parallel prefix adder for multimedia applications," *Journal of Computational Electronics*, vol. 19, no. 4, pp. 1139-1154, Dec. 2020.

BIOGRAPHIES OF AUTHORS

Masood Ahmad Mahammad received his B.E. degree in electronics and communication engineering from VTU, Karnataka. He received M.Tech. in electronics and instrumentation from Andhra University College of Engineering, Andhara University. He received Ph.D. in low power VLSI from JNTU Hyderabad, Hyderabad, Telangana, India. He is currently working as Assistant Professor in GITAM University, Hyderabad, Telangana, India. His research includes low power VLSI, digital system design, FPGA's and internet of things. He has 15 technical research publications. He has six international journals, three international conference proceedings, one national conference, two books and three book chapters. He has 18 years of teaching experience. He is a senior member of IEEE, life member of IETE, ISTE, Institute of Engineers India, semiconductor society of India and VLSI society of India. He can be contacted at email: mmahamma@gitam.edu, masoodahmad80@gmail.com.

Dr. Appala Raju Uppala received the Diploma in electronics and communication engineering (ECE) from Govt. Polytechnic College, Narsipatnam, India, in 1997, the A.M.I.E degree in electronics and communication engineering from Institution of Engineers India, Kolkata, the M.Tech. degree with specialization of digital system and computer electronics from Jawaharlal Nehru Technological University (JNTU), Hyderabad, India, in 2007, and Ph.D. degree from Department of Electronics and Communication Engineering, Jawaharlal Nehru Technological University (JNTU), Kakinada, India, in 2022. His areas of interest analog electronics and design, cognitive radio system, signal processing, adaptive signal processing, analog, and digital communications. He published 20 technical research publications in international journals, 4 conference proceedings, 1 patent and 1 book chapter. He has 18 years of teaching experience. Currently, he is an Associate Professor in the Department of Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Hyderabad, India. He is a life member of IETE, India. He can be contacted at email: raju.mdl@gmail.com.



Suggala Ram Prasad received his B.E. degree in instrumentation engineering and M.Tech. in embedded systems and persuing Ph.D. in sensors technology at GITAM Visakhapatnam Campus. He is working as Assistant professor in GITAM University, Hyderabad. His research includes sensors, instrumentation, and VLSI. He can be contacted at email: rsuggala@gitam.edu.

Anusha Marouthu obtained her Ph.D. degree in Department Computer Science and Engineering (CSE) at KL University, India, 2019. She currently works as an Associate professor in the Department of Computer Science and Engineering at the KL University of India and research interests include wireless sensors networks and MAC protocol problems in cognitive radio networks. She can be contacted at email: anushaaa9@kluniversity.in.