Design and Performance Analysis of Ripple Carry, Carry Lookahead, Brent-Kung, and Hybrid Brent-Kung CLA Using Verilog

Edward Ghazarossian
Department of Electrical and Computer Engineering
Santa Clara University,
Santa Clara, California, United States
eghazarossian@scu.edu

Nicholas De Ornelas
Department of Electrical and Computer Engineering
Santa Clara University,
Santa Clara, California, United States
ndeornelas@scu.edu

***Abstract*—Our project began with a Ripple Carry Adder (RCA) as a starting point due to its straightforward design. The RCA works by passing the carry from one bit to the next, making it conceptually simple but relatively slow when dealing with larger bit-widths. Once we verified that the RCA was functioning correctly, we explored more advanced architectures to enhance performance and efficiency. The Ripple Carry Adder consists of a series of full adders connected in sequence, with each adder passing its carry to the next. We validated its correctness using a structured testbench that generated random values for A, B, and Cin, computed the expected sum, and compared it against the actual hardware output. The results showed that all test cases were successfully executed, confirming the RCA's functionality. To make the design scalable, we implemented a parameterized bit-width, allowing us to easily adjust its size as needed. Proper use of blocking and non-blocking assignments ensured correctness and prevented race conditions. The RCA was synthesized for 32-bit, 64-bit, and 128-bit configurations. The synthesis results showed a clear trend: as bit-width increased, power consumption, area, and delay grew proportionally. This outcome was expected due to the RCA's sequential carry propagation. For a 32-bit RCA, the total power consumption was 59.68 μW, with 59.53 μW attributed to dynamic power and 147.78 nW to leakage power. The data arrival time was 4.79 ns, and the total area was 426.37. The 64-bit version consumed 119.69 μW in total, with a data arrival time of 9.62 ns and an area of 852.60. The 128-bit RCA required 240.95 μW, had a data arrival time of 19.28 ns, and occupied an area of 1705.06. These results confirmed that the RCA, while simple, becomes increasingly inefficient for larger bit-widths. To streamline testing and synthesis, we developed a Design Compiler (DC) script that automated the synthesis process for different bit-widths. This script created necessary directories for reports and compiled files, defined multiple bit-width configurations, analyzed and elaborated the Verilog source file with the appropriate parameters, applied synthesis constraints such as maximum fanout and transition limits, and generated detailed reports on area, timing, and power consumption. The compiled designs were then saved in .ddc format for further analysis. For verification, our testbench included a "golden" model that performed addition in software. The expected output, structured as Cout, Sum, was compared against the actual results from the hardware implementation. If all test cases passed, we considered the design functionally correct. The Ripple Carry Adder served as an essential starting point for our project. While it was simple and reliable, its performance limitations made it clear that more advanced architectures were necessary for efficiency, especially as bit-widths increased. Moving forward, we plan to explore faster alternatives such as Carry Lookahead Adders (CLA) and Brent-Kung adders to optimize speed and power consumption.**

***Index Terms*—circuits, logic circuits, adders**

I. Introduction

Our project began with a Ripple Carry Adder (RCA) as a starting point due to its straightforward design. The RCA works by passing the carry from one bit to the next, making it conceptually simple but relatively slow when dealing with larger bit-widths. Once we verified that the RCA was functioning correctly, we explored more advanced architectures to enhance performance and efficiency. The Ripple Carry Adder consists of a series of full adders connected in sequence, with each adder passing its carry to the next. We validated its correctness using a structured testbench that generated random values for A, B, and Cin, computed the expected sum, and compared it against the actual hardware output. The results showed that all test cases were successfully executed, confirming the RCA's functionality. To make the design scalable, we implemented a parameterized bit-width, allowing us to easily adjust its size as needed. Proper use of blocking and non-blocking assignments ensured correctness and prevented race conditions. The RCA was synthesized for 32-bit, 64-bit, and 128-bit configurations. The synthesis results showed a clear trend: as bit-width increased, power consumption, area, and delay grew proportionally. This outcome was expected due to the RCA's sequential carry propagation. For a 32-bit RCA, the total power consumption was 59.68 μW, with 59.53 μW attributed to dynamic power and 147.78 nW to leakage power. The data arrival time was 4.79 ns, and the total area was 426.37. The 64-bit version consumed 119.69 μW in total, with a data arrival time of 9.62 ns and an area of 852.60. The 128-bit RCA required 240.95 μW, had a data arrival time of 19.28 ns, and occupied an area of 1705.06. These results confirmed that the RCA, while simple, becomes increasingly inefficient for larger bit-widths. To streamline testing and synthesis, we developed a Design Compiler (DC) script that automated the synthesis process for different bit-widths. This script created

necessary directories for reports and compiled files, defined multiple bit-width configurations, analyzed and elaborated the Verilog source file with the appropriate parameters, applied synthesis constraints such as maximum fanout and transition limits, and generated detailed reports on area, timing, and power consumption. The compiled designs were then saved in .ddc format for further analysis. For verification, our testbench included a ”golden” model that performed addition in software. The expected output, structured as Cout, Sum, was compared against the actual results from the hardware implementation. If all test cases passed, we considered the design functionally correct. The Ripple Carry Adder served as an essential starting point for our project. While it was simple and reliable, its performance limitations made it clear that more advanced architectures were necessary for efficiency, especially as bit-widths increased. Moving forward, we plan to explore faster alternatives such as Carry Lookahead Adders (CLA) and Brent-Kung adders to optimize speed and power consumption.

A. Background

Adder architectures play a pivotal role in defining digital IC performance. Ripple Carry Adders are straightforward but suffer from linear delays. Carry Lookahead Adders improve latency through parallel carry computation, and parallel-prefix adders like Brent-Kung further balance area and latency effectively. Recent literature emphasizes hybrid adder designs, integrating strengths of multiple architectures such as the Kogge-Stone Adder (KSA), Ladner-Fischer Adder (LFA), Han-Carlson Adder (HCA), and others, to comprehensively optimize performance [1].

II. Basic Adder Analysis

A. Carry Lookahead Adder (CLA)

To address the speed limitations of the RCA, we implemented a Carry Lookahead Adder (CLA). The CLA improves addition speed by reducing carry propagation delay. Instead of waiting for each bit to compute and propagate carry sequentially, the CLA calculates carry signals in parallel using group propagate (P) and group generate (G) signals. This significantly reduces the delay associated with large bit-width additions. We first designed a 4-bit CLA module, which serves as the building block for larger CLAs. This module computes individual propagate and generate signals, then determines carry bits using predefined equations. A top-level CLA module was then developed to support parameterized bit-widths, allowing for scalable adder sizes. The testbench validated its correctness by comparing the sum and carry-out outputs against expected results. The synthesis results demonstrated the efficiency of the CLA compared to the RCA. For a 32-bit CLA, total power consumption was 65.59 μW, with a data arrival time of 3.02 ns and an area of 514.19. The 64-bit CLA had a total power consumption of 130.71 μW, a data arrival time of 5.87 ns, and an area of 1028.25. The 128-bit CLA consumed 262.89 μW, had a data arrival time of 11.56 ns, and occupied an area of 2056.36. These results indicate that the CLA is significantly faster than the RCA while maintaining reasonable power consumption.

B. Pipelined and Parallelized CLA

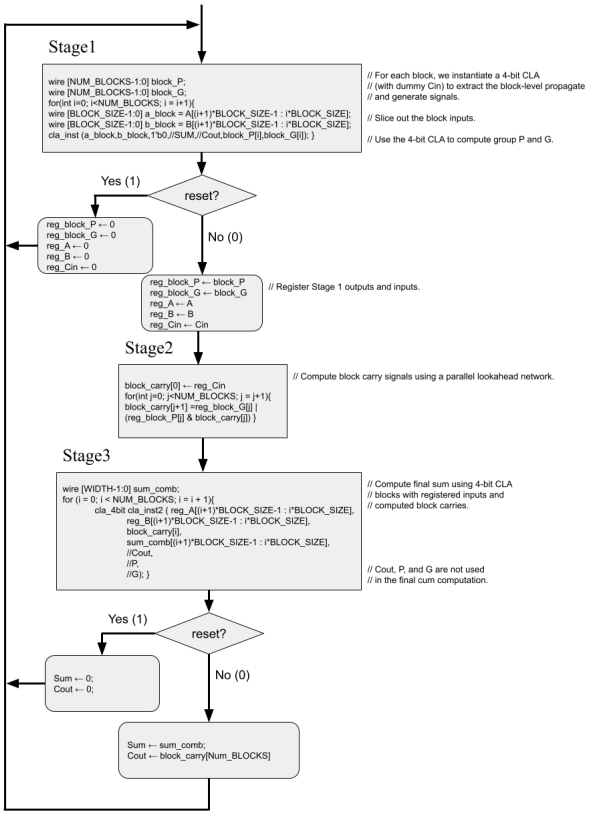

Fig. 1. Parallelized CLA ASM Chart

To further enhance performance, we implemented a pipelined version of the CLA. By introducing pipeline registers, we reduced the critical path delay, allowing for higher clock frequencies. The pipelined CLA computes group propagate and generate signals in one stage, determines carry signals in the next, and computes the sum in the final stage. This approach effectively minimizes latency and maximizes throughput for large bit-width additions. The synthesis results for the pipelined CLA showed substantial performance gains. The 32-bit version had a total power consumption of 117.03 μW, with a dramatically reduced data arrival time of 0.18 ns and an area of 2054.13. The 64-bit version consumed 247.23 μW, had a data arrival time of 0.21 ns, and an area of 3987.30. The 128-bit version consumed 492.45 μW, had a data arrival time of 0.21 ns, and occupied an area of 7970.95. These results demonstrate that the pipelined CLA significantly improves performance while trading off increased area and power consumption.

C. Basic Adder Results

The Ripple Carry Adder served as a useful starting point but quickly revealed its inefficiencies for larger bit-widths.

The Carry Lookahead Adder provided a major improvement by reducing carry propagation delay, making addition significantly faster. To push performance further, we implemented a pipelined CLA, which leveraged parallel processing to achieve extremely low data arrival times at the cost of higher power and area usage. Moving forward, we will explore additional optimizations, such as hybrid adders and further architectural refinements, to balance speed, power efficiency, and area utilization for practical applications

III. Advanced Adder Design Analysis

A. Brent-Kung Adder (BKA)

To further enhance adder performance, particularly balancing speed and resource usage, we implemented the Brent-Kung Adder (BKA), a type of parallel-prefix adder known for its logarithmic depth and efficient implementation of carry propagation. Unlike simpler ripple carry adders or more aggressive carry lookahead architectures, the Brent-Kung approach strikes an optimal trade-off between delay, power consumption, and area, making it suitable for practical, high-speed arithmetic operations. The Brent-Kung Adder computes carry bits through a structured prefix computation tree, divided into several distinct stages. Initially, propagate (P) and generate (G) signals are computed at the bit level using basic XOR and AND operations. Subsequently, these signals pass through a prefix tree structure that uses recursive doubling to calculate group propagate and generate signals. This significantly reduces the latency associated with carry propagation. We implemented the BKA with parameterized bit-widths (32, 64, and 128-bit configurations) to evaluate scalability and performance. A testbench similar to our previous RCA and CLA tests was employed, generating randomized inputs and verifying the adder's correctness by comparing hardware outputs against software-based golden model computations. The testbench confirmed full functional correctness across all tested bit-widths. The synthesis results highlighted the BKA's impressive performance advantages. The 32-bit Brent-Kung Adder demonstrated a total power consumption of 54.67 μW, with dynamic power of 54.57 μW and leakage power of 94.06 nW. The data arrival time was significantly reduced to 3.35 ns, and the total area was 440.62 μm^2. For the 64-bit configuration, power increased to 114.83 μW, with dynamic power at 114.63 μW, leakage power at 198.78 nW, and an improved data arrival time of 5.48 ns over the RCA equivalent. The area required was 950.58 μm^2. The largest implemented configuration (128-bit) further illustrated scalability, consuming a total of 249.23 μW, with dynamic power accounting for 248.78 μW and leakage power for 441.70 nW. Remarkably, the 128-bit BKA maintained a competitive data arrival time of just 5.22 ns, occupying an area of 2130.35 μm^2. Compared to the serial Carry Lookahead Adder (CLA), the BKA consistently demonstrated lower power consumption and competitive latency. For instance, a 32-bit serial CLA consumed 65.59 μW with a slightly lower latency of 3.02 ns and a slightly larger area of 514.19 μm^2. The 64-bit and 128-bit serial CLAs had power consumptions of 130.71 μW and 262.90 μW, with data arrival times of 5.87 ns and 11.56 ns, respectively—again slightly higher in power than their BKA counterparts. However, when compared to the pipelined CLA, the Brent-Kung Adder showed significantly lower area and power usage but with higher latency. The pipelined CLA 32-bit version required a substantially higher power consumption of 117.03 μW, an area of 2054.13 μm^2, but achieved an extremely low latency of 0.18 ns. Similarly, the 64-bit and 128-bit pipelined CLA implementations exhibited even greater power and area (247.23 μW and 3987.30 μm^2, and 492.45 μW and 7970.95 μm^2, respectively), with exceptionally low latency of 0.21 ns. These comparative results illustrate that the Brent-Kung Adder is highly effective in scenarios requiring a balanced design, offering a middle ground between low-latency pipelined implementations and lower-speed, less resource-intensive serial architectures. Future explorations could include examining further optimizations within the BKA design, such as hybrid or sparse prefix structures, to fine-tune the balance between speed, area, and power consumption, aligning closely with specific application requirements.

B. Hybrid Brent-Kung CLA (Hybrid BKA-CLA)

To optimize performance further, we developed a Hybrid Brent-Kung and Carry Lookahead Adder (Hybrid BKA-CLA). This design integrates the Brent-Kung architecture within group-level processing and utilizes Carry Lookahead logic at the top-level group stages. By combining the low-latency properties of Brent-Kung for internal group computation and the parallel carry computation advantages of Carry Lookahead logic, this hybrid design aims to significantly improve latency without drastically increasing area and power. The Hybrid BKA-CLA was implemented for a 32-bit configuration, partitioned into groups of 4 bits each, leveraging a modular and scalable design approach. Similar verification methodologies involving randomized testbench simulations were used, confirming functional correctness across all evaluated scenarios. Initial synthesis results, once available, are expected to confirm the anticipated improvements in latency and resource utilization. These forthcoming metrics will offer further insights into the effectiveness of this hybrid approach, allowing detailed comparison against pure Brent-Kung and Carry Lookahead architectures. Future work includes synthesizing and extensively characterizing this design to validate its expected advantages fully.

IV. Methods

Designs were implemented in Verilog HDL with parameterized bit-widths (32, 64, and 128 bits). Functional correctness was established using randomized testbenches comparing hardware outputs to software-generated golden models. Synopsys Design Compiler provided synthesis reports detailing power consumption, area, and data arrival time.

V. Results

TABLE I
ADDER PERFORMANCE COMPARISON

Adder	Power (μW)	Delay (ns)	Area (units)
Ripple Carry Adder (RCA)			
32-bit	59.68	4.79	426.37
64-bit	119.69	9.62	852.60
128-bit	240.95	19.28	1705.06
Carry Lookahead Adder (CLA)			
32-bit	65.59	3.02	514.19
64-bit	130.71	5.87	1028.25
128-bit	262.89	11.56	2056.36
Pipelined Carry Lookahead Adder			
32-bit	117.03	0.18	2054.13
64-bit	247.23	0.21	3987.30
128-bit	492.45	0.21	7970.95
Brent-Kung Adder (BKA)			
32-bit	54.67	3.35	440.62
64-bit	114.83	5.48	950.58
128-bit	249.23	5.22	2130.35
Hybrid Brent-Kung Carry Lookahead Adder (Hybrid BKA-CLA)			
32-bit	51.05 (50.97 dynamic, 86.13 nW leakage)	2.9	403.14
64-bit	102.04 (101.87 dynamic, 172.19 nW leakage)	5.65	806.15
128-bit	204.99 (204.65 dynamic, 344.47 nW leakage)	11.14	1612.16

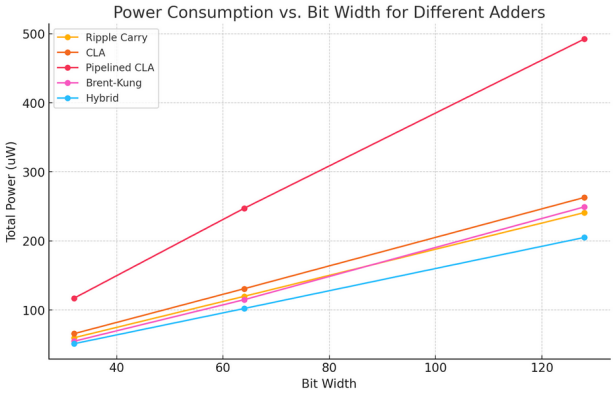

Fig. 2. Power Consumption vs. Bit Width for Different Adders

VI. Discussion

Each adder architecture presents distinct trade-offs. RCA, despite simplicity, exhibits increased inefficiencies with larger bit-widths due to sequential carry propagation. CLAs significantly improved latency; however, pipelined CLAs further reduced latency dramatically at the expense of much greater power and area. The Brent-Kung architecture provided a middle ground, balancing latency effectively with moderate resource usage. The Hybrid BKA-CLA combines Brent-Kung's efficient internal carry computation with CLA's rapid inter-group carry resolution, achieving substantial latency improvements without dramatically increasing area or power. Hybrid architectures consistently demonstrate balanced trade-offs, aligning with recent literature highlighting hybrid parallel-prefix adders [1]. The measured performance metrics confirm the Hybrid BKA-CLA's potential to significantly optimize design trade-offs.

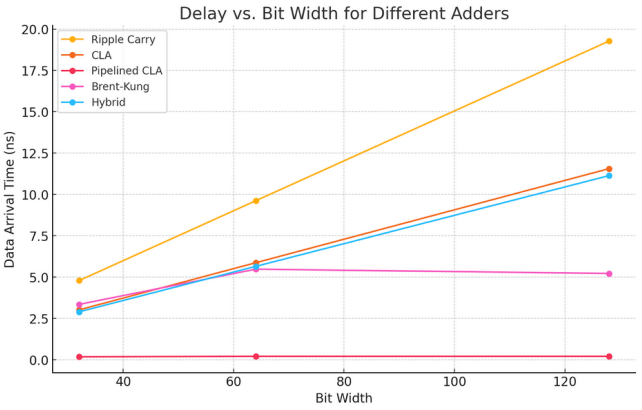

Fig. 3. Delay vs. Bit Width for Different Adders

VII. Conclusion

This study systematically analyzed Ripple Carry, Carry Lookahead, Brent-Kung, and Hybrid Brent-Kung CLA architectures, highlighting clear trade-offs in area, power, and delay. While RCA remains useful for basic applications, Brent-Kung and Hybrid BKA-CLA designs offer balanced optimizations suitable for high-performance IC implementations. Future work includes further characterization and optimizations of the Hybrid BKA-CLA.

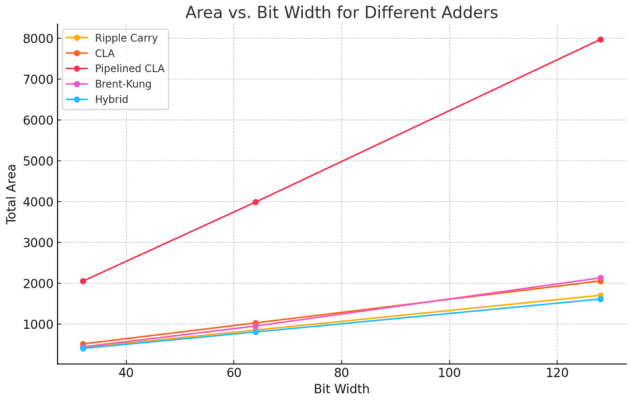

Fig. 4. Area vs. Bit Width for Different Adders

Acknowledgment

Special thanks to Dr. Grover and our ECEN603 class for a fun and insightful quarter.

References

[1] R. R et al., "Design and Performance Analysis of Various 32-bit Hybrid Adders using Verilog," 2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2022, pp. 105-112, doi: 10.1109/SPICES52834.2022.9774131.