Journal of Engineering Science and Technology
21st EURECA 2024 Special Issue August (2025) 76 - 84

DESIGN A HIGH PERFORMANCE 32-BIT ADDER USING VERILOG

SERENE ZOE-YUC CHIN, WAI-LEONG PANG*

School of Engineering, Taylor's University, Taylor's Lakeside Campus,
No. 1, Jalan Taylor's, 47500, Subang Jaya, Selangor DE, Malaysia
*Corresponding Author: waileong.pang@taylors.edu.my

Abstract

The usage of adder in RISC-V functions to perform various mathematical operations such as address calculation, division, multiplication and Fast Fourier Transform, that are essential for computer program's arithmetic computations. The efficiency and design of the adder are key factors that influence the overall efficiency of the processor. Most adders nowadays are constructed using CMOS standard digital cell library; however, CMOS transistors exhibit several limitations such as short channel effect, inadequate gate control and high-power leakage. The Hardware Description Language, Verilog modelled adder can act as an alternative to existing transistor level adder. The 32-bit Kogge-Stone Adder (KSA), the Carry Select Adder (CSA), and the Han-Carlson Adder (HCA) are modelled using Verilog in this paper. These adders designed are simulated in Intel Quartus Prime Software. The adders are implemented on an Intel Cyclone IV Field-Programmable Gate Array (FPGA). Extensive simulation results are carried out to evaluate the performance of these three adders. The performance metrics, delay, logic utilization and power consumption are evaluated. The simulation results shown that HCA provides the lowest delay (8.16 ns), but HCA has the highest power consumption (0.91 mW). HCA is suitable for the high-speed computing applications. However, KSA has the best power efficiency with 7% lower power consumption and 2.2% higher delay compared to HCA. Therefore, KSA is appropriate for applications that are sensitive to energy use. CSA only provides the average performance compared to HCA and KSA in various aspects.

Keywords: Carry select adder, Han Carlson adder, Kogge stone adder.

1. Introduction

The adder circuit serves as a fundamental building block within the Arithmetic Logic Unit (ALU), a core component of processing units. Essential for executing a vast array of complex mathematical computations, including multiplication, division, and address generation, adders are indispensable in modern computing. Their critical role extends to logarithmic operations, such as those employed in Fast Fourier Transforms (FFT). Consequently, the pursuit of low-power, high-speed adder circuits across various design parameters is paramount for advancing Very-Large-Scale Integration (VLSI) capabilities [1].

Adder circuits are classified into distinct categories based on their underlying logic and architectural configurations. To optimize performance metrics such as speed, area, and power consumption, specific adder types have been developed. Conventional adders and Parallel Prefix Adders (PPAs) represent the primary classifications. Examples of conventional adders include Carry Lookahead Adders (CLAs), Carry Select Adders (CSLAs), and Ripple Carry Adders (RCAs). In contrast, PPAs employ a hybrid of sequential and parallel computations to expedite carry generation [2].

RCAs form the foundational adder structure, characterized by sequential bitwise addition and inherent carry propagation delays. In contrast, CLAs prioritize speed through precomputed carry signals, albeit with increased circuit complexity. The Carry Select Adder (CSA) balances speed and area efficiency by partitioning the addition process into parallel blocks [3]. PPAs exemplified by the KSA, excel in speed but demand substantial hardware resources. HCA emerges as a compromise, blending elements from various designs to meet specific performance constraints.

To address the escalating demand for accelerated computing, specialized hardware processors have emerged. While general-purpose central processing unit (CPU) offers flexibility, their efficiency pales in comparison to domain-specific architectures like graphics processing units (GPUs), which excel in data-parallel workloads such as deep learning. Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs) surpass both CPUs and GPUs in terms of power efficiency and computational throughput [4, 5].

The RISC-V ISA has rapidly gained prominence since its inception in 2010, finding applications across diverse domains. This widespread adoption is attributable, in part, to the burgeoning ecosystem of open-source RISC-V hardware implementations [6]. RISC-V ISA-based processor designs remove barriers in the semiconductor industry by not requiring licensing fees for patented intellectual property (IP) core blocks. With only 47 basic instructions, RISC-V's modular architecture allows it to be easily customized to meet a wide range of design requirements [7].

The RISC-V ISA does not specify implementation details or necessary subsets. The RISC-V processors can improve additions to save memory usage, conserve power, and reduce code size. The way instructions are implemented has a big impact on processor efficiency. Single-cycle processors only carry out the subsequent instruction once the current one has been completed. Efficiency declines with instruction complexity, and latency rises when the cycle time has to accommodate the slowest instruction.

CMOS scaling technology has been a major driver of processor performance progress over the past forty years [8]. But CMOS technology is getting close to its physical limits, which makes further scaling more challenging. The design and production of CMOS transistors are labour-intensive and intricate procedures that can be carried out using silicon-on-insulator, twin-well, or N-or P-well technology. The fact that every transistor manufacturing is unique makes adjustments more difficult. Despite this, CMOS standard digital cell libraries have been used to create a wide range of adders, from ripple carry to parallel-prefix adders [9].

Verilog is programmable Hardware Description Language (HDL), it can reduce the design cycle of transistor-level adders, which makes circuit modifications easier. Nevertheless, because most designs concentrate on the transistor level, there is a dearth of study on Verilog modelling for high-performance adders. Considerations like speed, power consumption, and resource usage are important when choosing an adder for modelling. The CSA, KSA, and HCA is especially well-suited for this paper because of their unique benefits. By computing sums and carrying them in parallel for both possible carry-in values, the CSA improves computational speed over sequential approaches such as the RCA and minimizes propagation time.

KSA excels in high-speed operation due to its deeply parallel prefix computation and minimal logic stages. This architectural approach enables efficient parallel processing, leading to exceptional performance scalability across varying bit widths. By distributing computational load across multiple stages, the KSA mitigates power consumption and signal integrity issues commonly associated with large-scale adder designs.

The HCA prioritizes resource efficiency without compromising performance by optimizing gate count and reducing circuit depth compared to the KSA. Its balanced computational approach contributes to lower power consumption, making it suitable for power-sensitive applications.

High-performance 32-bit adders offer substantial advantages in resource-constrained environments demanding low power consumption and rapid computation. Their suitability for edge computing and IoT devices is evident in applications such as power-efficient smart sensors. The adder's lower latency and power consumption can speed up operations like convolutional neural networks and big data analytics, which improves processing efficiency for large datasets and speeds up the training of AI models in fields like artificial intelligence and machine learning, where quick arithmetic computations are essential [10].

2. Methods

2.1. Carry selects adder (CSA)

To improve computational performance and properly control carry propagation, a 32-bit CSA is first implemented by splitting the adder into multiple smaller blocks, usually consisting of 4-bit sections. Each block will carry out the addition operation twice, under the assumptions that a carry-in of 0 and a carry-in of 1 respectively.

Since this is the first stage, the sum and carry in the least significant block (Block 1) are determined independently of any earlier carries. Two separate 4-bit adders are used in each block for the following blocks (Block 2 onwards); one

assumes a carry input of 0 and the other a carry input of 1. For every block, this dual calculation produces two sets of outputs: the sums (sum0 and sum1) and the associated carry outputs (cout0 and cout1).

A 2-to-1 multiplexer is utilized to determine the proper sum and carry outputs for each block. Sum0 and cout0 are picked if the carry out from the previous block is 0, while sum1 and cout1 are chosen if it is 1. The final carry out decides the overall carry for the full 32-bit addition operation. This selection method is repeated for all blocks. Equations (1) and (2) can be used to determine the sum and carry.

$$Sum = (A \oplus B \oplus Cin) \tag{1}$$

$$Carry = (A \cdot B) + (Cin \cdot (A \oplus B)) \tag{2}$$

2.2. Han Carlson Adder (HCA)

As seen in Fig. 1, the 32-bit HCA works by breaking the adder up into smaller blocks, usually 4-bit segments. This method, which resembles the conventional CSA, aims to streamline the design procedure and provide operational clarity. Every segment computes the sum (sum0 and sum1) and carry-out (cout0 and cout1) for two alternative carry-in conditions simultaneously, grouping adjacent blocks (e.g., 0-3, 4-7, 8-11, etc.).

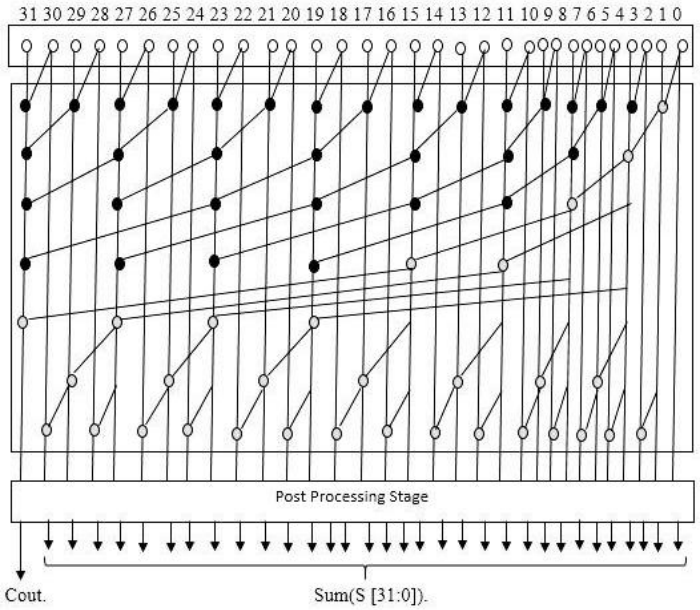

Fig. 1. Schematic of 32-bit Han Carlson Adder [11].

HCA improves performance by managing possible outcomes simultaneously. Furthermore, by effectively sharing multiplexers across neighbouring block pairs. The adder streamlines the design while retaining fast computing speed, reducing hardware complexity. Equations (3) through (6) contain the specific logic equations that control the HCA.

Generate Signal

$$Gi = Ai + Bi \tag{3}$$

Propagate Signal

$$Pi = Ai \oplus Bi \tag{4}$$

Black Cell Calculation

$$(Gi, Pi) = (Gi + (Pi \cdot Gi - 1), Pi \cdot Pi - 1) \tag{5}$$

Gray Cell Calculation

$$Gi = Gi + (Pi \cdot Gi - 1) \tag{6}$$

2.3. Kogge Stone Adder (KSA)

The 32-bit KSA makes use of a binary tree structure that shown in Fig. 2. Each level of the tree processes increasingly larger groups of bits. This structure enables effective parallel computation. Level 0 is where the process starts, with initial partial sums and carry being calculated using nearby bit pairs. These partial results are integrated as the calculation moves through following levels, each of which integrates a bigger portion of the binary input.

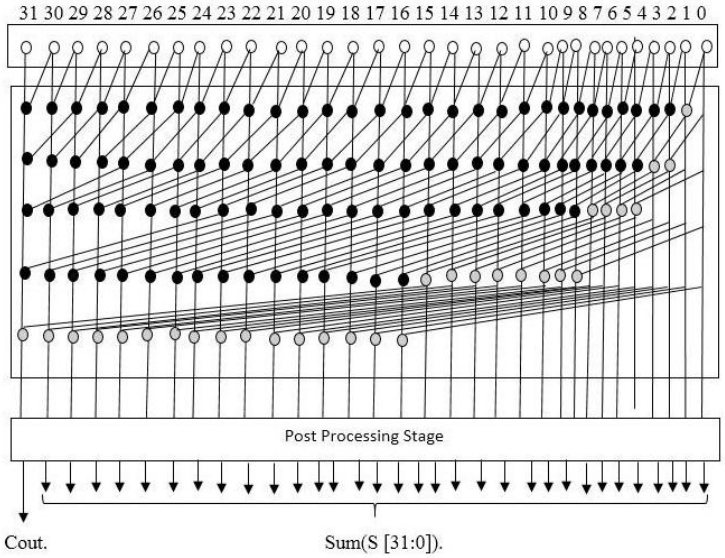

Fig. 2. Schematic of 32-bit Kogge Stone Adder [11].

The main benefit of the KSA is its parallel carry computation, which minimizes propagation delay compared to sequential techniques like the ripple carry adder by enabling carries to be computed throughout the whole structure at once. At the top of the tree (Level $\log_2(n)$ for 32 bits), the total and execution come from the combined outcomes that have been spread through the lower levels. The exact logic Eqs. (7) to (10), which regulate this procedure, are given.

Generate Signal

$$Gi = Ai + Bi \tag{7}$$

Propagate Signal

$$Pi = Ai \oplus Bi \tag{8}$$

Black Cell Calculation

$$(Gi, Pi) = (Gi + (Pi \cdot Gi - 1), Pi \cdot Pi - 1) \tag{9}$$

Final Sum

$$\text{Si} = \text{Pi} \oplus \text{Ci} - 1 \tag{10}$$

2.4. Verilog modelling of the adders

The Model Specification phase of the project kicks off with a thorough determination and documentation of the adders' functionality, criteria, inputs, and outputs. This initial phase guarantees a comprehensive comprehension of the particular goals that the finished design has to accomplish. The flow chart for the Adder Circuit Verilog Modelling Process is shown in Fig. 3. Intel Quartus Prime is used to carry out the simulation works.

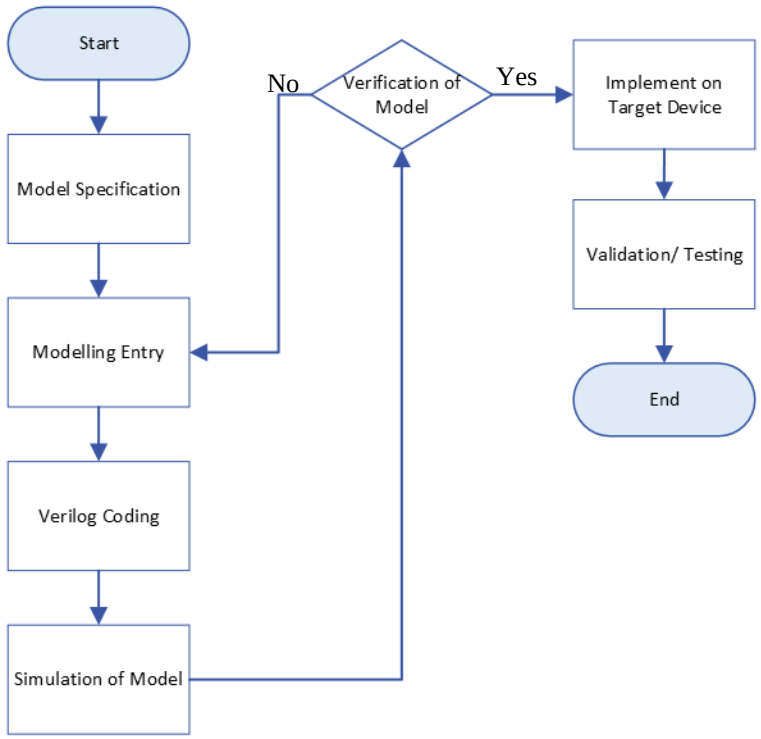

Fig. 2. Verilog modelling process for adder circuit.

The process begins with model specification that defines the functionality of the adders modelled in Verilog. The essential information such as the input and output specifications, functional operations, requirements and constraints. In modelling entry, the adder's architecture is designed and represented using a suitable design entry method based on the specification. This often involves

creating a block diagram or schematic to visualize the circuit's components and their interconnections.

The behaviour and structure of the adder is modelled in Verilog, which provides a textual representation of the hardware design. The Verilog code is simulated to verify its functionality and identify any errors or discrepancies. This step helps ensure that the adder circuit operates as intended before proceeding to hardware implementation. ModelSim is used for the performance and functional verification. Testbenches are built to evaluate the functionality of the adders.

Extensive analyses are carried out to evaluate the performance of the adder modelled in Verilog in order to ensure the design meets the specified requirements with correct functionality. Any issues identified during this stage necessitate returning to the previous steps for corrections or modifications. The adder modelled in Verilog is synthesized and implemented onto a Field-Programmable Gate Array (FPGA) for experimental test.

3. Results and Discussion

The adders' simulated waveform, shown in Fig. 4, offers a thorough analysis of three different adder architectures: HCA, KSA and CSA. After analysis, performance indicators such as logic use, power consumption, and delay were compiled into Table 1.

With the lowest delay of 8.16 ns, the HCA proved to be the fastest in terms of speed. HCA performs better due to its parallel prefix structure, which successfully reduces carry propagation delay. With a delay of 8.34 ns, the KSA trailed closely after, providing a well-balanced strategy that blends speed with controlled complexity. With a delay of 8.53 ns, the CSA offered an effective speed profile even if it was not as quick as the KSA or HCA.

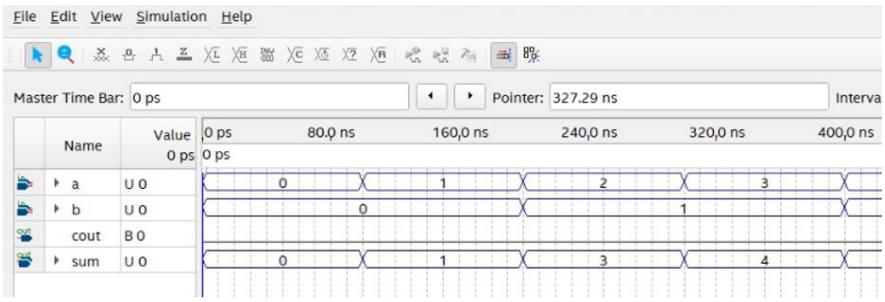

Fig. 4. The example of the simulated waveform.

Table 1. Test model specifications and test conditions.

Adders	Delay (ns)	Logic Utilization (in ALMs)	Power consumption (mW)
Carry Select Adder	8.53	98	0.89
Han Carlson Adder	8.16	78	0.91
Kogge Stone Adder	8.34	189	0.85

The KSA was the most energy-efficient with a power consumption of just 0.85 mW. KSA works especially well in situations where power efficiency is crucial, including in systems that are energy-conscious or battery-powered. However, because of its substantial parallel processing, the HCA had the highest power usage at 0.91 mW, even though it had a speed advantage. With a moderate power consumption of 0.89 mW, the CSA falls in between the KSA and HCA, making it appropriate for situations where power consumption and speed are both moderately important.

KSA required the most amount of logic resources with 189 Adaptive Logic Modules (ALMs). This is due to the parallel prefix structure's complexity. On the other hand, the HCA showed the best resource efficiency, using just 78 ALMs because of its more straightforward architecture with lowers hardware overhead. The CSA used 98 ALMs, providing a resource-use strategy that was balanced.

HCA is suite for the applications that value speed, like high-performance computing environments where cutting down on processing time is essential. The HCA's exceptional speed comes at the cost of increased power consumption, rendering it less suitable for power-constrained environments. However, the KSA's lower power profile and acceptable latency make it the preferred choice for energy-critical applications. The CSA strikes a balance between speed, power, and resource efficiency, making it a versatile option for general-purpose computing tasks.

4. Conclusions

The behavioural and operations of 3 adders, i.e. KSA, CSA, and HCA are modelled in Verilog. A comprehensive simulation study was conducted in Intel Quartus Prime Software to assess the performance of these adders across the crucial performance metrics of delay, resource utilization, and power consumption. The HCA exhibited the lowest delay (8.16 ns), making it ideal for high-speed applications. However, its power consumption and resource demands limit its suitability for power-constrained scenarios. In contrast, the KSA demonstrated exceptional power efficiency (0.85 mW), rendering it well-suited for battery-powered devices. CSA strikes a balance between speed, power, and resource efficiency, making it a versatile option for general-purpose computing tasks.

Acknowledgement

This research was supported by Taylor's University, Malaysia through Taylor's Matching grant (International) (PARTNERSHIP/QIS/2025/FIT/001).

References

1. Panda, S.K.; Achyut, K.; and Panda, D.C. (2023). *Synthesis and time analysis of FPGA-Based DIT-FFT module for efficient VLSI signal processing applications*. In Tripathi, S.L.; and Mahmud, M. (Eds.), *Explainable Machine Learning Models and Architectures*. Wiley, 65-79.
2. Brzozowski, I. (2024). Comparative analysis of dynamic power consumption of parallel prefix adder. *ACM Transactions on Design Automation of Electronic Systems*, 29(3), 1-22.
3. Kadam, D.B.; Pandyaji, K.K.; and Liyakat, K.K.S. (2022). Implementation of carry select adder (CSLA) for area, delay and power minimization. *Telematique*, 21(1), 5461-5474.

4. Kalapothas, S.; Flamis, G.; and Kitsos, P. (2022). Efficient edge-AI application deployment for FPGAs. *Information*, 13(6), 279.
5. Kalapothas, S.; Flamis, G.; and Kitsos, P. (2021). Importing custom DNN models on FPGAs. *Proceedings of the* 2021 10th *Mediterranean Conference on Embedded Computing* (*MECO*), Budva, Montenegro, 1-4.
6. Lu, T. (2021). A survey on risc-v security: Hardware and architecture. *arXiv*:2107.04175 [*cs.CR*], 1-39.
7. Palmer, C. (2022). Simplified instruction set architecture accelerates chip development and wins the 2022 draper prize. *Engineering*, 17, 7-9.
8. Chakraborty, S.; and Joshi, R.V. (2024). Cryogenic CMOS design for Qubit control: Present status, challenges, and future directions [Feature]. *IEEE Circuits and Systems Magazine*, 24(2), 34-46.
9. Balasubramanian, P.; and Mastorakis, N.E. (2022). High-speed and energy-efficient carry look-ahead adder. *Journal of Low Power Electronics and Applications*, 12(3), 46.
10. Niknejad, N.; Ismail, W.; Ghani, I.; Nazari, B.; and Bahari, M. (2020). Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation. *Information Systems*, 91, 101491.
11. Naik, V.M. (2015). Performance analysis of parallel prefix adder. *International Journal of Electrical Electronics and Data Communication*, 3(7), 74-77.