Mathematical Modelling of Engineering Problems
Vol. 12, No. 2, February, 2025, pp. 730-744
Journal homepage: http://iieta.org/journals/mmep

Design and Implementation of Low Power Generic Reversible Binary-Coded Decimal Adder on Artix-7 FPGA

Kanchan S. Tiwari[1*], Revati M. Wahul[2], Sagar D. Shinde[3], Manisha A. Dudhedia[4], Varsha P. Gaikwad[5], Pranav Bhalerao[1], Simrit Kaur Gill[1], Neeraj Chhajed[1], Christy Daniel[1], Shravan H. Gawande[6]

[1] Department of Electronics and Telecommunication, M. E. S. Wadia College of Engineering, S. P. Pune University, Pune 411001, India
[2] Department of Computer Engineering, M. E. S. Wadia College of Engineering, S. P. Pune University, Pune 411001, India
[3] Department of Mechanical Engineering, Shree Ramchandra College of Engineering, Pune, S. P. Pune University, Pune 411001, India
[4] Department of Electronics and Telecommunication, Marathwada Mitra Mandal's College of Engineering, Pune 411052, India
[5] Department of Information Technology, Government College of Engineering Aurangabad, Chh. Sambhajinagar 431005, India
[6] Department of Mechanical Engineering, M. E. S. Wadia College of Engineering, S. P. Pune University, Pune 411001, India

Corresponding Author Email: kanchan.tiwari@mescoepune.org

https://doi.org/10.18280/mmep.120235

Received: 5 October 2024
Revised: 18 December 2024
Accepted: 26 December 2024
Available online: 28 February 2025

Keywords:
reversible logic, generic Binary-Coded Decimal adder, quantum cost, garbage output, Artix-7 FPGA, power, delay

ABSTRACT

Reversible logic has emerged as a transformative approach to energy-efficient computing by enabling reversible computations and minimizing information loss, thereby reducing energy dissipation to negligible levels. This innovation is particularly relevant in the era of low-power and sustainable system design. Among its applications, the Binary-Coded Decimal (BCD) adder, a cornerstone of digital arithmetic, plays a vital role in domains such as financial modeling, signal processing, and embedded systems. This study introduces a novel reversible BCD adder that leverages Feynman, Toffoli, Haghparast–Navi Gate (HNG), Haghparast–Navi Full Gate (HNFG), Thapliyal–Sreenivas Gate (TSG), Peres, etc., gates to achieve exceptional energy efficiency and scalability. Designed using a generic programming methodology, the adder supports seamless adaptability to varying input sizes, scaling efficiently up to 512 bits without requiring redesign. The proposed design achieves a significant reduction in quantum cost (50), garbage outputs (14), and power dissipation (24 mW), while maintaining competitive delay (1.555 ns) and gate count (11), demonstrating superior efficiency compared to existing 8-bit reversible BCD adders. These findings have significant implications for power-sensitive applications, including portable devices, Internet of Things (IoT) systems, data centers, and quantum computing architectures. This work demonstrates the feasibility of high-performance, sustainable arithmetic operations for next-generation computing systems.

1. INTRODUCTION

In the era of energy-efficient computing, minimizing power consumption has become a critical priority, particularly for applications in portable electronics, IoT devices, and green technology. Reversible logic has emerged as a key enabler for achieving this goal, addressing energy dissipation by preventing information loss during computation.

Mollick [1] formulated a hypothesis suggesting that computing power would experience significant growth while costs would decrease proportionately. His forecasting predicted that all the transistors present on a semiconductor chip shall get two folded in every one and a half or two years, maintaining minimal cost increments. This forecast was known to be Moore's law and it also anticipated that substantial amount of heat would be generated as a result of certain information being lost. This heat dissipation, in turn, could lead to a drastic reduction in circuit performance and lifespan. Keeping that in mind, Landauer's [2] research elucidated that conventional gates dissipate energy at a minimum rate of kTln2 joules per bit of information loss, wherein k represents the Boltzmann constant (1.38064852×10^{-23} joule/kelvin) and T signifies absolute temperature in kelvin during operations. Following this, Bennett [3] demonstrated in 1973 that to circumvent this energy dissipation, circuits must be constructed using reversible logic. Consequently, various reversible gates namely the Fredkin Gate, Peres Gate, Toffoli Gate and Feynman Gate were devised with the intention of achieving this objective [4]. Unlike traditional logic circuits, which generate heat by erasing information, reversible gates allow input data to be fully reconstructed from outputs, adhering to Landauer's principle. This reduces energy dissipation to negligible levels, making reversible logic a cornerstone of sustainable computing.

A reversible gate establishes a bijective relationship between the input and output vectors. This capability allows for the generation of inputs from outputs and the ability to backtrack to any point in the computation. This characteristic is termed logical reversibility. However, the advantages of this are contingent upon achieving physical reversibility, which refers to a procedure that expends nil energy as heat. While achieving total perfect physical reversibility is nearly unattainable, efforts can be made to minimize or even eliminate energy dissipation by ensuring that computations are information lossless [5]. A qubit represents a fundamental unit of information, capable of existing in a state of 0, 1, or a superposition of both [6]. Any manipulation performed on individual qubits can be designed to be reversible, allowing for the preservation of information integrity. This is explained with an example given below in Figure 1.

Upon examining the AND Gate shown in Figure 1, which comprises two inputs and one output, and referring to the corresponding truth table given in Table 1, it is evident that an input combination, such as '00,' yields an output of '0,' as does '01' and '10.' Consequently, discerning the specific input combination resulting in the output '0' proves challenging.

Contrary to the CNOT Gate as depicted in Figure 2, this is additionally known as the Feynman Gate features two inputs and two outputs. Notably, when the inputs are '00,' the output is exclusively '00,' and no other input configuration produces this output. Similarly, for input '01,' the output is '01.' Thus, each input corresponds to a distinct output, unlike the conventional AND Gate, mitigating the complexity of prediction. This characteristic aligns with the principle of reversibility. Its truth table is shown in Table 2.

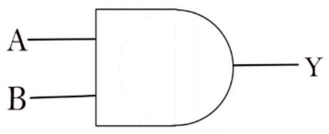


Figure 1. AND Gate

Table 1. AND Gate truth table

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

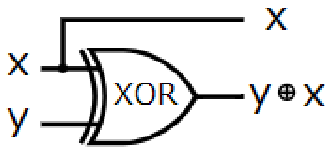


Figure 2. CNOT Gate

Table 2. CNOT Gate truth table

Input		Output	
X	Y	X	X⊕Y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

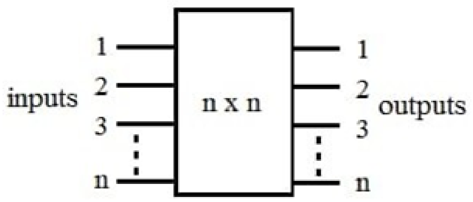


Figure 3. Basic reversible gate

In an N×N reversible gate, as shown in Figure 3, the input vector (Iv) and output vector (Ov) are represented as sequences of elements (I_1, I_2, I_3, ... I_N) and (O_1, O_2, O_3, ... O_N) respectively. To construct reversible circuits, a set of reversible gates is required, ensuring that each input vector corresponds uniquely to an output vector, and vice versa, thus maintaining a one-to-one mapping between them. This distinction between reversible and arbitrary logic circuits rests on the key properties:

(i) The equality of input and output bit counts,

(ii) The guarantee that each input pattern corresponds to a unique output pattern,

(iii) No fan-out allowed.

According to the study conducted by Nayak et al. [7], there exist various terminologies and parameters associated with reversible logics.

Constant Inputs (*CI*): Inputs set to 0 or 1 to synthesize non-reversible Boolean functions into reversible forms.

Garbage Outputs (*GO*): Unused outputs essential for achieving reversibility, ensuring the sum of inputs and constant inputs equals the sum of outputs and garbage outputs.

Gate Count: The number of reversible logic gates used in the circuit.

Quantum Cost (*QC*): Indicates the overall usage of 2×2 quantum primitives to form an equivalent circuit.

Delay. Based on the majority of gates from an input line to an output line, assuming that each gate operates in a single time unit and all inputs are available before computation starts.

Hardware Complexity. The number of logical operations (AND, OR, EX-NOR) within the circuit.

Flexibility. Indicates the universality of reversible logic gates, aiming for minimal garbage output, minimal delay, and no feedback/loops to achieve reversibility.

In hardware design, the preference for binary computing over decimal computing arises from the simplicity of constructing hardware based on the binary number system. Despite the convenience of building binary hardware, challenges emerge when representing fractional decimal numbers, like 0.110, precisely in binary form. Consequently, approximate values are employed for computational purposes within binary hardware. Acknowledging the limited tolerance of commercial, financial, and net-based platforms for blunders stemming from conversions among binary and decimal formats, there has been a significant shift towards prioritizing decimal arithmetic. Binary Coded Decimal (BCD) is a representation technique utilized for encoding decimal digits (0-9) in binary form. Ongoing efforts are aimed at accelerating the development of dedicated hardware specifically designed for decimal arithmetic. So, there is great emphasis on the design of low-power hardware implementation of decimal arithmetic circuits based on reversible logic.

The Binary-Coded Decimal (BCD) adder, a fundamental arithmetic component, is widely used in digital systems that require precise decimal computations, such as financial systems, digital clocks, IoT devices, quantum computing, and calculators. However, traditional BCD adders, designed using irreversible logic, are inherently energy-intensive and face scalability challenges. These limitations hinder their deployment in power-sensitive applications and emerging fields like quantum computing. A reversible BCD adder provides a compelling solution by combining the benefits of energy efficiency, scalability, and minimal hardware overhead.

This paper presents a low-power reversible BCD adder implemented on an Artix-7 FPGA, designed to address these limitations. Existing methods [8-13] have utilized various reversible gates such as the Toffoli Gate (TG) [14], New Gate (NG) [15], HNFG Gate [11], Feynman Gate [16], etc., for the implementation of a reversible BCD adder. The implementation of the BCD adder typically involves three key stages. First, a binary-centric 4-bit parallel adder performs addition on the two BCD numbers in their binary representation. Next, the result is evaluated by a Six Converter Module (SCM) to ensure accuracy [17]. If the outcome exceeds '1001' or generates a carry, it is identified as invalid and requires adjustment. Finally, the correction is made by adding '0110' to the sum using a 4-bit parallel adder.

The proposed architecture utilizes a combination of Fredkin, Toffoli, and Peres Gate, ensuring a bijective mapping between inputs and outputs while minimizing quantum cost (QC), garbage output (GO), and constant input (CI). The modular design is scalable to input sizes up to 512 bits, enabling seamless adaptation to various applications without frequent redesigns.

Compared to conventional designs [18, 19], the proposed BCD adder achieves a 70% reduction in power dissipation while maintaining comparable delay performance, enhanced scalability for large datasets, and applications requiring high precision, and simplified arithmetic processes for frequent decimal computations.

By implementing the design on an Artix-7 FPGA, this study combines energy efficiency with the benefits of high performance, flexibility, and rapid prototyping. The FPGA's architecture further enables potential parallel processing, making it ideal for applications such as financial systems, IoT devices, and advanced embedded systems.

The proposed design addresses the trade-offs in existing approaches, which often compromise on key metrics like QC and GC. It sets a benchmark for energy-efficient computing, paving the way for future advancements in reversible arithmetic circuits and their integration into low-power digital systems.

The structure of this paper is as follows: Section 2 provides an overview of reversible logic gates and prior work. Section 3 details the design methodologies of the proposed BCD adder. Section 4 discusses the verification and implementation using VHDL, and Section 5 concludes with key findings and future directions.

2. LITERATURE SURVEY

Numerous reversible gates have been developed and widely utilized by researchers [20-22] for designing various combinational and sequential circuits. This section provides a concise overview of the reversible gates employed in the proposed BCD adder design, followed by a discussion of the current state-of-the-art advancements in BCD adder designs [23-25].

2.1 Basic reversible gates

Exploring different reversible gates holds paramount importance in digital circuitry, given their role in enabling bidirectional information flow. This attribute holds significant importance in fields such as quantum computing, designing circuits with low power consumption, and applications involving cryptography. A comprehensive comprehension of the various types of reversible gates is fundamental for engineers and researchers, as they form the cornerstone for building reliable and efficient digital systems. A few of these basic gates and their types are mentioned below:

2.1.1 Feynman/CNOT Gate

The 2×2 Feynman Gate, also acknowledged as the controlled NOT (1-CNOT) Gate, depicted in Figure 4, operates based on logic operations where P is equivalent to B, and Q is identical to the XOR of A and B. This gate is highly efficient for managing a single bit, guaranteeing that if the second input is '0', the first input will be mirrored in both outputs. Consequently, the Feynman Gate is recognized as the utmost appropriate option for the single-bit operations, as it prevents the creation of unnecessary outputs, thereby ensuring processing efficiency. It will implement the logical functions: $P=A$ and $Q=A\oplus B$.

2.1.2 Toffoli Gate

A Toffoli Gate as depicted in Figure 5 is designed for 3 inputs and typically includes 3 outputs. In a three-input and three-output Toffoli Gate, inputs A and B correspond to the first as well as second outputs, and the third output depends on the states of inputs A and B, serving to invert input C and it executes the logical functions: $P=A$, $Q=B$, and $R=AB\oplus C$.

2.1.3 New Gate

The New Gate as depicted in Figure 6, is characterized by its ability to process multiple input variables and generate corresponding output variables based on predefined logical rules. The specifics of these logical rules, such as AND, OR, XOR operations, or more complex combinations, depend on the intended functionality of the gate within a given digital circuit. Designers often use NG Gates as building blocks for more complex digital systems. It executes the logical functions: $P=A$, $Q=AB\oplus C$, and $R=A'C'\oplus B'$. It works as a half adder when B is made '0'. The sum value will be obtained on R.

2.1.4 Haghparast–Navi Gate (HNG)

The HNG Gate as depicted in Figure 7, which is also known as the Hadamard-NAND Gate, combines two fundamental logic operations - the Hadamard operation and the NAND operation. It executes the logical functions: $P=A$, $Q=B$, $Sum =A\oplus B\oplus C$ in and $Cout =(A\oplus B)$. C in $\oplus AB\oplus D$. It works as a full adder when D is set to '0', in that case, R gives sum and S gives carry generated out of it.

2.1.5 Hadamard Nand Fanout Gate (HNFG)

HNFG as depicted in Figure 8, extends the functionality of the HNG gate by incorporating certain fanout capabilities. Each HNFG Gate functions effectively as two 2×2 Feynman gates and can also serve as a "Copying Circuit" to address the fan-out limitation in reversible circuits. It enables a single duplication of two bits without generating any garbage outputs, making it highly efficient for this purpose. It not only performs NAND operations but it also facilitates the distribution of output to multiple destinations within the circuit. It executes the logical functions: P=A, Q=A⊕C, R=B and S=B⊕D.

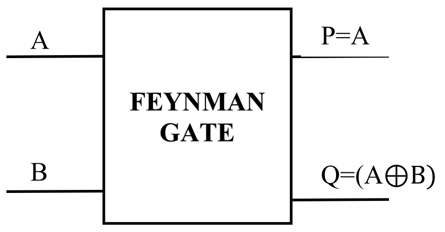


Figure 4. Feynman Gate

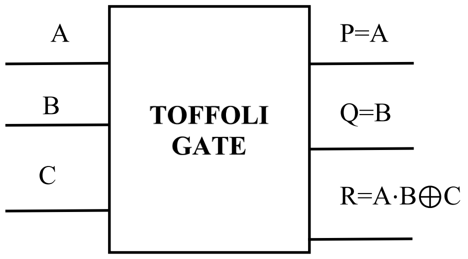


Figure 5. Toffoli Gate

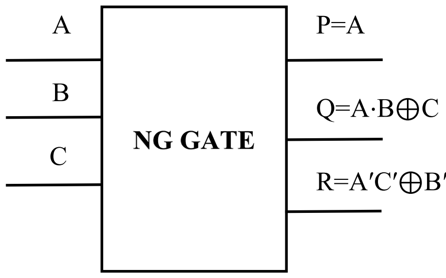


Figure 6. New Gate

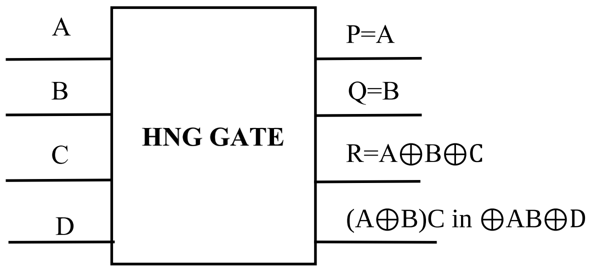


Figure 7. HNG Gate

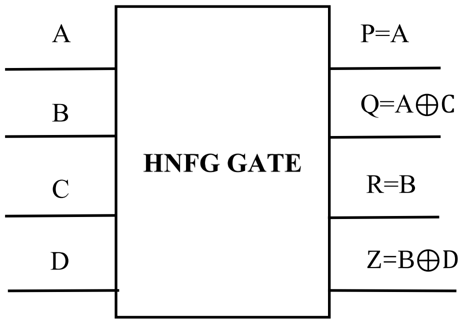


Figure 8. HNFG Gate

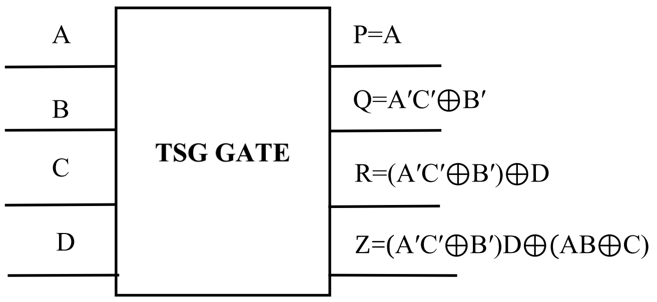


Figure 9. Thapliyal-Srinivas Gate

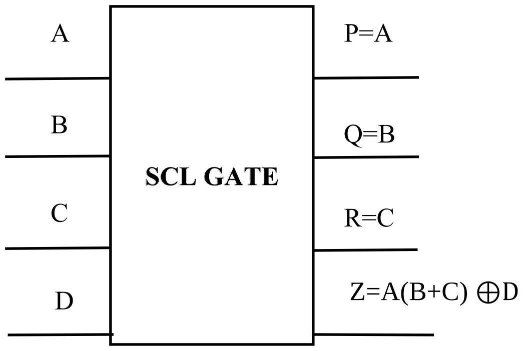


Figure 10. Six Correction Logic Gate

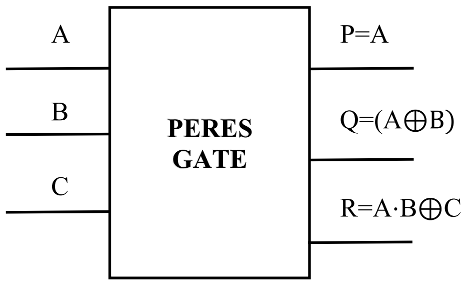


Figure 11. Peres Gate

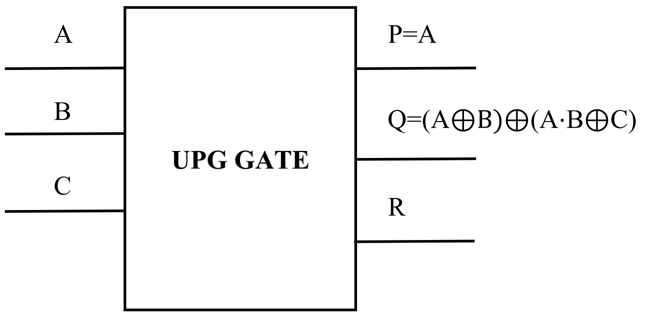


Figure 12. Universal Programmable Gate

2.1.6 Thapliyal-Srinivas Gate (TSG)

The TSG as shown in Figure 9 is a 4×4 reversible gate that functions as a versatile building block, capable of implementing all Boolean functions and operating as a reversible full adder, AND Gate, or half adder. The TSG is instrumental in designing efficient adder units, reversible 4:2 compressors, and 8×8 reversible Wallace tree multipliers. It offers significant optimization over existing architectures by reducing the number of reversible gates and garbage outputs, making it a highly efficient choice for advanced reversible computing designs. The equations for the TSG are P=A, Q=A′C′⊕B′, R=(A′C′⊕B′)⊕D, S=(A′C′⊕B′)D⊕(AB⊕C).

2.1.7 Six Correction Logic Gate (SCLG)

The SCLG is a 4×4 reversible gate specifically designed for correction in BCD addition. Its outputs are defined as P=A, Q=B, R=C, and S=A(B+C)⊕D. The primary output S is used for applying the correction factor of 6 ("0110") when the sum of two BCD digits exceeds 9 or generates a carry, ensuring valid BCD representation. The term A(B+C) detects conditions requiring correction, while ⊕D adjusts the output accordingly. The remaining three outputs (P, Q, R) propagate the inputs A, B, C, which is critical in reversible circuits where fan-out is not permitted. These outputs can be connected to other gates or used in subsequent stages to build complex circuits without requiring additional duplication logic. By facilitating correction and enabling seamless integration, the SCLG ensures efficient and reversible BCD addition. It is shown in Figure 10.

2.1.8 Peres Gate

The Peres Gate is a 3-input, 3-output reversible gate used in reversible logic circuits, particularly for BCD addition as shown in Figure 11. It operates as follows: the outputs are P=A, Q=A⊕B, and R=AB⊕C where A, B, C are the inputs, and P, Q, and R are the outputs. The gate computes the sum bit Q as the XOR of A and B, while the carry bit R is generated by the AND of A and B, adjusted by the previous carry C. The output P simply passes through the control input A. This functionality makes the Peres Gate efficient for BCD addition, enabling the simultaneous computation of sum and carry without additional resources, making it ideal for reversible, low-power arithmetic circuits.

2.1.9 Universal Programmable Gate (UPG)

A 3×3 UPG, as shown in Figure 12 implements the logical operations P=A, Q=(A⊕B)⊕(A·B⊕C), R=AB⊕C.

The quantum cost and delay for the UPG are both 4. This gate is termed the UPG because it can be configured as a programmable 3×3 gate that implements logical functions such as AND, NAND, OR, and NOR by using input C as the fixed selection line, and Q and R as fixed output lines. When C=0, the gate performs AND and OR operations. When C=1, it produces NAND and NOR operations.

The UPG offers a significant advantage over the Peres Gate in that when used in its programmable configuration, the UPG can produce four distinct logical operations, while the Peres Gate can only produce three (XOR, AND, and NAND). This flexibility makes the UPG a more versatile and efficient component in reversible logic circuits.

2.2 A survey on existing BCD adder

The Table 3, presents current state of art in design of BCD adder using reversible gates. Various researchers have designed optimized BCD adders [26-28]. Akilandeswari [29] implemented an 8-bit reversible ripple carry adder with a delay of 5.885 ns but did not discuss quantum cost or power dissipation, leaving a gap in comprehensive performance analysis. In the study [30], a novel approach for multi-digit Binary Coded Decimal (BCD) adders using quantum-dot cellular automata (QCA) technology was proposed. The delay values reported were 6.50 clock cycles for a 4-bit adder and 10.50 clock cycles for an 8-bit adder. As QCA designer was used, delay is the only parameter suitable for comparison. Borodzhieva et al. [31] implemented 1-, 2-, and 3-digit BCD adders without reversible logic and omitted any mention of performance metrics. Similarly, the work conducted by Ykuntam and Prasad [32] focuses on conventional BCD adders instead of reversible ones, reporting a delay of 17.7 ns for an 8-bit BCD adder.

Krishna and Uma [33] designed, simulated, and tested reversible logic-based full adders/subtractors for different sizes. While their 8-bit binary reversible adder achieved a delay of 3.2 ns, the gate count disparity between binary and BCD circuits explains the lower delays for binary adders. However, their proposed design for a 64-bit binary reversible adder has a high quantum cost of 640. As reported by Kumar et al. [34], a novel reversible BCD adder design was introduced using ASK and New Gate (NG) logic, achieving significant efficiency improvements with a gate count of 11, 13 ancilla inputs, and 22 garbage outputs, representing a 63.33% reduction in gate count compared to existing architectures. The design, validated using Xilinx Vivado 2022.2, demonstrates the effectiveness of reversible logic in optimizing resource utilization and energy efficiency.

Table 3. Survey of BCD adders

Authors	Observations
Akilandeswari [29]	This work explored the design of reversible logic-based adders in CMOS VLSI design. It investigates techniques for implementing adders using reversible logic, aiming to achieve efficient and low-power designs suitable for VLSI applications.
Chu et al. [30]	This paper presented various BCD adder implementations based on the three input XOR and other majority gates. By utilizing these gates, the design aims to achieve optimized performance in terms of speed and area efficiency for BCD addition operations.
Borodzhieva et al. [31]	This paper discusses the use of inquiry-based study for implementing BCD adders. It presents a pedagogical approach aimed at enhancing students' understanding and skills in designing BCD adders through hands-on learning experiences.
Ykuntam and Prasad [32]	The paper proposed a modified architecture for BCD adders utilizing a Mirror adder to enhance speed and reduce area. The approach aims to improve the efficiency of BCD addition operations by leveraging innovative design techniques.
Krishna and Uma [33]	Discusses the design, implementation, and analysis of low power-based adder/ subtractor circuits that are reversible.

3. PROPOSED GENERIC REVERSIBLE BCD ADDER

This paper discusses the design and implementation of a generic reversible BCD adder using four different architectures. BCD addition involves the addition of two BCD digits to yield an output BCD digit, each within the range of 0 to 9. This addition process adheres to specific rules of binary-centred addition, wherein correction logic is only required if the sum exceeds 9 or if a carry is present. Unlike traditional binary adders, BCD addition requires additional correction logic due to the limited range of BCD numbers (0 to 9). Nevertheless, challenges arise in the context of BCD addition, necessitating the incorporation of correction logic. The proposed design emphasizes modularity and scalability, making it adaptable for higher bit-widths beyond 8 bits. The foundation of the design lies in the use of a 4-bit reversible BCD adder as a building block, which can be combined iteratively to construct adders of larger sizes. For example, an 8-bit BCD adder is implemented by cascading two 4-bit BCD adders, as shown in Figure 13 and Figure 14. This hierarchical approach enables scalability to higher bit widths, such as 16-bit or 32-bit, by simply integrating additional blocks. The major advantages of scalability are:

Modular design: The use of 4-bit adders as primitive blocks simplifies the construction of higher-order adders without significant redesign.

Reusability: The correction logic and carry propagation mechanism remain consistent across different bit-widths.

Reduced development effort: The same design methodology can be extended to larger sizes, reducing implementation complexity.

The limitations and challenges in scaling are:

Delay accumulation: As shown in Table 4, the combinational path delay increases with the bit-width due to the propagation of carry signals across multiple blocks. For instance, the delay for a 64-bit adder is significantly higher compared to smaller sizes. This limits the practicality of the design for applications requiring ultra-low latency.

Power dissipation: Power consumption grows as additional blocks are added. Although reversible logic reduces overall power dissipation, the complexity of managing carry and correction logic in higher-order designs results in incremental power usage, as indicated in Table 5.

Increased resource utilization: Scaling requires more gates, leading to an increase in garbage outputs, constant inputs, and quantum costs. While the design strives for optimization, these parameters grow linearly or sub linearly with the number of bits, impacting efficiency. Table 6 highlights the resource usage for smaller designs, which would proportionally increase for larger sizes.

FPGA resource constraints: Larger bit-width designs may exceed the available hardware resources of certain FPGAs, such as the Artix-7 board used for implementation. Optimizing the design to fit within these constraints could require additional effort or design trade-offs.

To address these challenges and facilitate future improvements in scalability, the following strategies can be explored:

Optimized gate design: Developing more efficient reversible gates with reduced quantum costs and garbage outputs.

Parallel carry management: Employing carry-look ahead or carry-save methods in reversible logic to minimize delay.

Resource-aware mapping: Tailoring the design for specific FPGA architectures to maximize resource utilization efficiency.

Table 4. Combinational path delay for all design

Combinational Path Delay (ns)					
Adder Size	**Conventional Adder**	**Algorithm 1**	**Algorithm 2**	**Algorithm 3**	**Kogge Stone Adder**
4	2.267	2.537	2.34	2.256	0.948
8	3.468	3.743	3.96	3.721	1.555
16	5.605	8.682	7.781	6.653	2.901
32	9.905	15.51	10.192	12.517	6.001
64	18.494	30.492	19.015	24.244	12.11

Table 5. Power dissipation for all design

Sr. No.	Adder Type	Power Dissipation (mW)
1	Conventional (Irreversible)	80.2
2	Algorithm1	78
3	Algorithm 2	189
4	Algorithm 3	90
5	Kogge Stone BCD adder	55
6	Reversible Kogge Stone BCD adder	24

Table 6. Comparison of reversible adder algorithms

Algorithms	Garbage Output	Quantum Cost	Constant Inputs	Gate Count
Algorithm 1	24	60	24	16
Algorithm 2	22	55	30	12
Algorithm 3	10	45	21	8
Reversible Kogge Stone adder (Proposed)	14	37	14	11

Overall, the proposed design is inherently generic and scalable but requires careful consideration of resource usage, delay, and power as bit widths increase. By addressing these limitations through optimization strategies, the design can be effectively extended to higher-order BCD adders.

3.1 BCD adder design 1: Implementation details

Figure 13 illustrates the block diagram of the basic 4-bit reversible BCD adder. In this design, both the input and output have an equal bit count. The inputs, labelled as A, B, C, and D, each consist of four bits, while the outputs, represented by W, X, Y, and Z, also consist of four bits. In this configuration, Y denotes the sum, Z represents the carry, and W and X are garbage outputs. The implementation of the proposed reversible BCD adder follows a systematic, multi-stage design process. It begins with the construction of a single-bit adder using HNG gates, followed by the development of a 4-bit adder using four HNG gates. This initial 4-bit BCD adder serves as a foundational building block for higher-order BCD adders. In addition, a correction logic block, comprising NG, FG, and TG gates, is added to refine the output for BCD correctness.

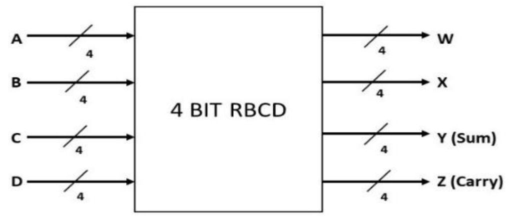


Figure 13. 4-bit reversible BCD adder

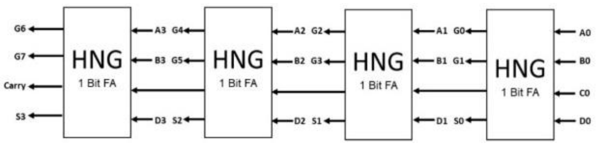


Figure 14. 4-bit parallel adder

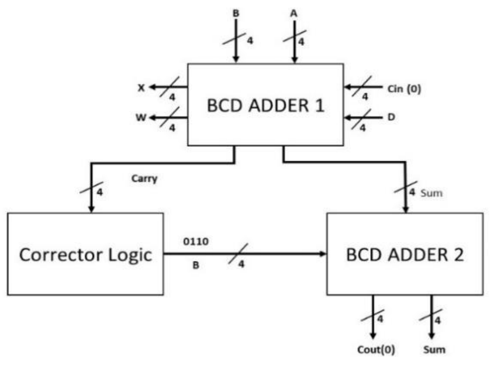


Figure 15. Decomposition of an 8-bit reversible Binary-Coded Decimal (BCD) using two 4-bit reversible BCD adders

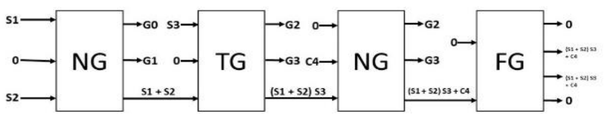


Figure 16. Correction logic

Each HNG gate functions as a full adder, producing both sum and carry outputs akin to a conventional full adder, thereby constituting a one-bit adder. By connecting four such one-bit adders, a four-bit adder is formed. This is shown in Figure 14. However, to adapt this binary adder into a BCD adder, integration of correction logic becomes necessary.

Block diagram shown in Figure 15 comprises of two 4-bit inputs labelled "A" and "B." The reversible adder block operates by adding the individual bits of the inputs. Specifically, it adds A (0) with B (0), and if there's a carry resulting from the addition, it's internally propagated, as illustrated in the "4-bit Parallel Adder" section. Any temporary carry generated is directed to the NG gate. Another 4-bit reversible adder block is employed, which takes inputs from the sum output of the first adder block and four inputs from the FG gate. The output comprises 4 bits of sum and 1 bit of carry. Furthermore, two NG gates and one TG gate function as correction logic for the BCD adder. Correction logic is designed using NG, TG and FG gates as shown in Figure 16. The main purpose of correction logic is to add “0110”, whenever sum is greater than 9 or carry is generated [34, 35].

The diagram shown in Figure 15 illustrates the decomposition of an 8-bit BCD adder using two 4-bit Reversible BCD adders. The initial adder computes the partial sum and carry, directing the partial carry to the subsequent BCD adder. Concurrently, the second BCD adder processes the inputs, generating the final sum and carry. The higher order adders can be subsequently created using this basic building block as shown in Figure 17.

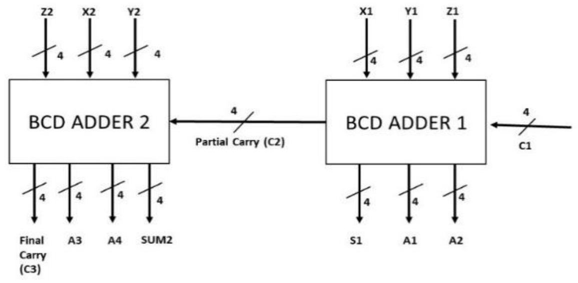


Figure 17. 8-bit reversible BCD adder using 4-bit BCD adder

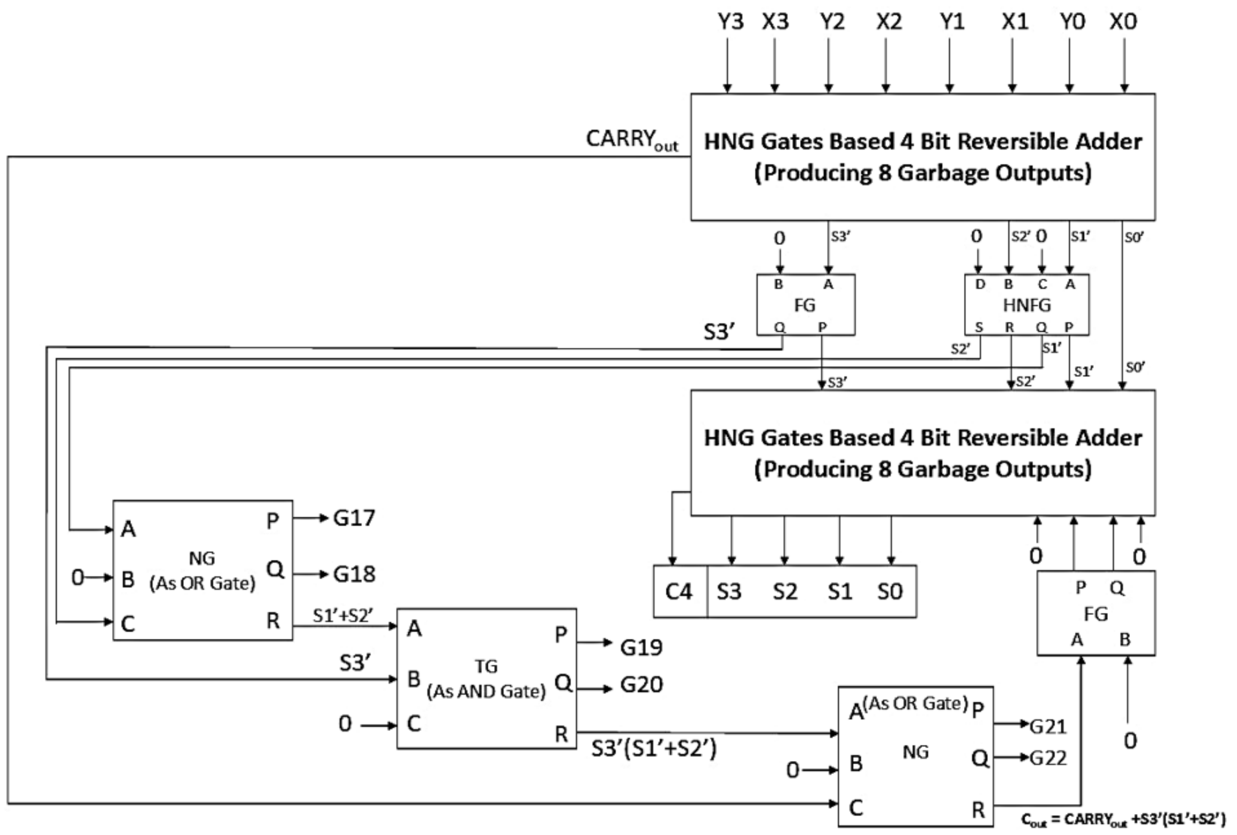

Y3 X3 Y2 X2 Y1 X1 Y0 X0
CARRY_out
HNG Gates Based 4 Bit Reversible Adder (Producing 8 Garbage Outputs)
FG
HNFG
S3′
S2′
S1′
S0′
HNG Gates Based 4 Bit Reversible Adder (Producing 8 Garbage Outputs)
C4 S3 S2 S1 S0
NG (As OR Gate)
G17
G18
S1′+S2′
TG (As AND Gate)
G19
G20
S3′(S1′+S2′)
NG
G21
G22
FG
C_out = CARRY_out +S3′(S1′+S2′)

Figure 18. Detailed schematic of 4-bit BCD adder

In Figure 18, a 4-bit reversible BCD adder block diagram is shown with all blocks integrated. The circuit consists of multiple HNG based adders, along with various auxiliary gates for managing BCD correction and carry propagation. The components description and working are as follows:

1) HNG Based 4-bit Reversible Adder (Upper Section):
   - Inputs: Y3, Y2, Y1, Y0, X3, X2, X1, X0
   - Outputs: Intermediate sums (S3', S2', S1', S0') and carry-out (CARRYout)
   - Function: This module performs the initial addition of two 4-bit BCD numbers and generates intermediate sum outputs along with the carry-out bit.
   - Logic Equations for Upper Section:
     - Sum Output: $Sn' = An \oplus Bn \oplus Cin$
     - Carry Output: $Cout = (An \cdot Bn) + (An \cdot Cin) + (Bn \cdot Cin)$ Where An, Bn are the n-th bits of inputs X and Y, Cin is the carry-in from the previous stage, and Sn' is the sum.
2) HNG Based 4-bit Reversible Adder (Lower Section):
   - Inputs: Intermediate sums from the upper adder (S3', S2', S1', S0')
   - Outputs: Final BCD sum (S3, S2, S1, S0) and carry-out (Cout)
   - Function: This module performs the final addition, ensuring that the result is a valid BCD number.
   - HNFG and FG gates: These gates ensure the reversible logic gates condition that fan-out is not permissible. It is acting as a buffer and simply replicates inputs at the outputs to facilitate further processing.
   - Logic Equations for Lower Section:
     - Final Sum Output: $Sn = Sn' \oplus Cout$ (corrected if necessary)
     - Carry Output: $Cout = (Sn' \cdot Cout)$
3) Correction Logic:
   - Components: OR Gate (NG), AND Gate (TG), and FG Gate (AS COPY Gate)
   - Function: This logic checks if the intermediate sum exceeds the BCD digit limit (9) and corrects the sum by adding "0110" (6) when necessary. The correction logic triggers the addition of "0110" to the binary sum if:
     - There is a carry-out from the initial addition (C′).
     - The two most significant bits of the sum (S3′ and S2′) are both 1.
     - The most significant bit and the second least significant bit of the sum (S3′ and S1′) are both 1.
   - Logic Equations for Correction logic

To determine when to add "0110", any one of these conditions being true is sufficient. The combined Boolean logic is $C' + (S3' \cdot S2') + (S3' \cdot S1') = 1$.

The correction logic implements this Boolean equation with the help above mentioned components. The NG gate works as an OR gate and generates the term $(S2' + S1')$. The TG gate then ANDs the term S3′ to it resulting in $(S3' \cdot S2') + (S3' \cdot S1')$. The NG gate further ORs this expression with C′ yielding the final term $C'\ (S3' \cdot S2') + (S3' \cdot S1')$ However, since fanout is prohibited in reversible circuits, an FG gate is used to copy this term and pass it on as the final carry.

The 4-bit reversible BCD adder described in the diagram exemplifies a sophisticated approach to BCD addition using reversible logic gates. The integration of HNG gates and dedicated correction logic ensures efficient and accurate BCD computation. The FG gate effectively manages fan-out issues, making the circuit practical for implementation. Future improvements could focus on further optimizing the quantum cost and garbage outputs to enhance the circuit's practical applicability.

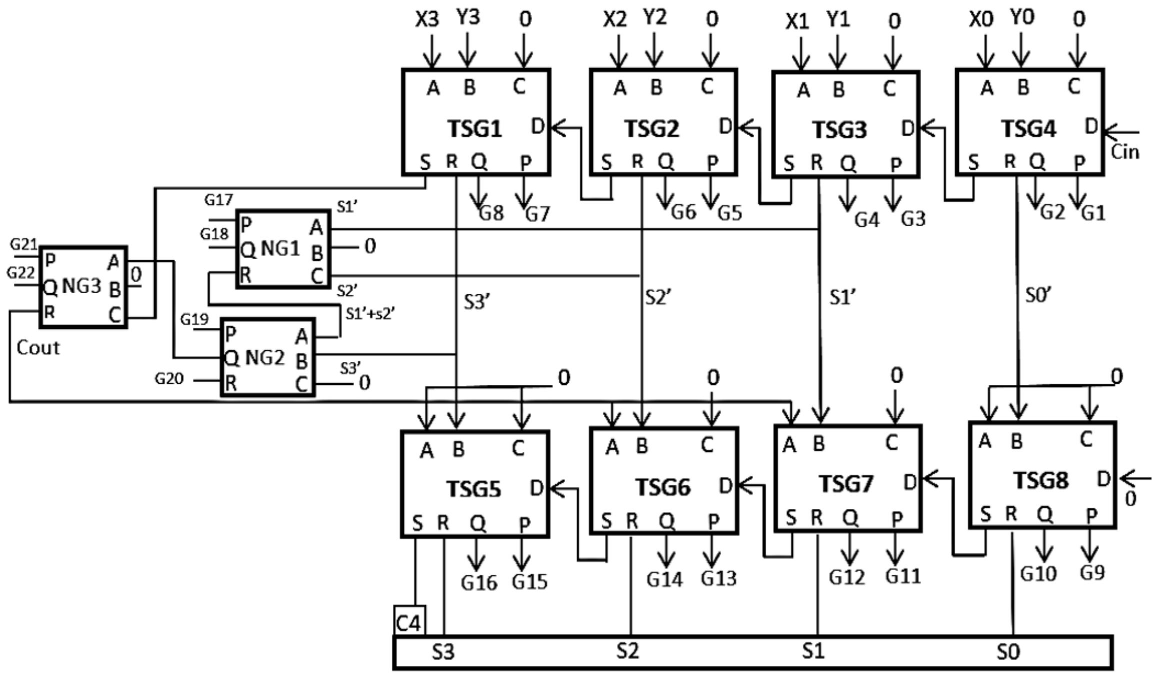


Figure 19. Algorithm 2 schematic

3.2 BCD adder design 2: Implementation details

Figure 19 represents a reversible 4-bit BCD adder circuit implemented using Thapliyal Srinivasan Gate (TSG) gates, supplemented by three NG gates. This proposed configuration includes eleven ancillary input lines and generates a total of twenty-two garbage output lines, which are considered non-essential to the main computational function but necessary for maintaining reversibility in logic operations.

The circuit has the following components:

1) Four TSG gates (TSG1, TSG2, TSG3, and TSG4) that perform the initial addition of the 4-bit BCD inputs X3 X2 X1 X0 and Y3 Y2 Y1 Y0, along with the carry input (Cin). These gates produce the following outputs:
   - S (Sum)
   - R (Garbage output)
   - Q (Garbage output)
   - P (Carry Out)
2) Five additional TSG gates (TSG5, TSG6, TSG7, and TSG8) that perform the BCD correction step by adding the necessary correction factor (e.g., 0×06 or 0×60) to the intermediate sum if it exceeds the BCD range.
3) The role of the new gates' module is critical in this architecture. It evaluates the carry bit status and decides whether the addition of "0110" to the intermediate sum is necessary. New gates used here implement the carry Boolean expression which basically is sum of products equation.

The operation of the circuit is as follows:

1) The 4-bit BCD inputs X3 X2 X1 X0 and Y3 Y2 Y1 Y0, along with the carry input (Cin), are fed into TSG1, TSG2, TSG3, and TSG4.
2) These gates perform the initial addition, producing the sum (S), carry out (P), and garbage outputs (R and Q).
3) The sum outputs (S) from the initial addition stage, along with the carry signals (CARRYout, G8, G7, G6, G5), are fed into TSG5, TSG6, TSG7, and TSG8.
4) These gates perform the BCD correction by adding the necessary correction factor to the intermediate sum if required.
5) The final BCD sum is obtained from the sum outputs (S) of TSG5, TSG6, TSG7, and TSG8.

When assessing the efficiency of the proposed design, it is observed that the quantum cost of the circuit is 55 and garbage outputs is 22. This metric provides an insight into the resource utilization and complexity of the circuit, highlighting its computational efficiency. However, a significant drawback of this design is its susceptibility to the fan-out problem. In reversible logic circuits, fan-out, or the duplication of output signals, is generally impermissible as it contradicts the principles of reversibility.

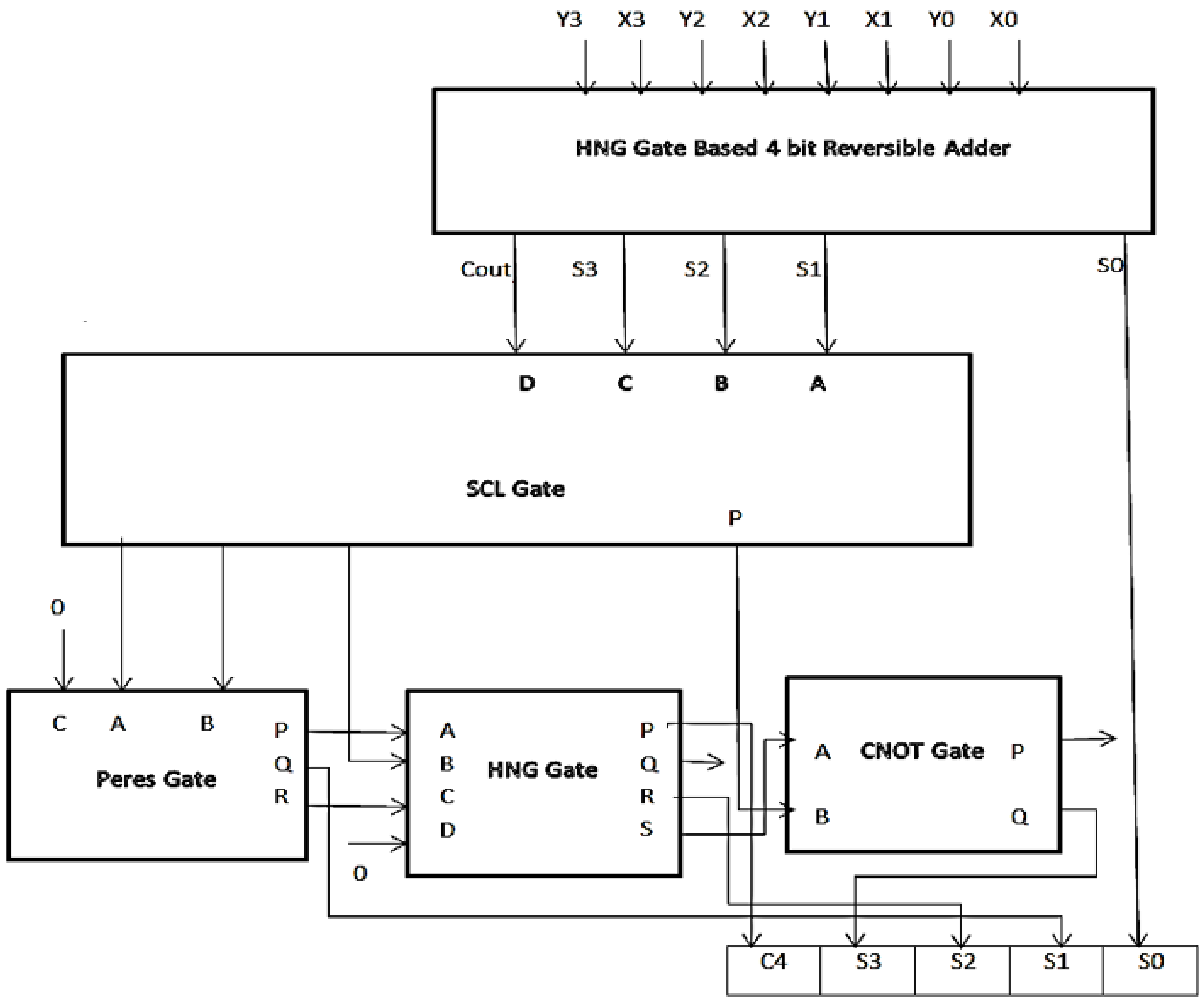


Figure 20. Algorithm 3

3.3 BCD adder design 3: Implementation details

The design in Figure 20 features a four-bit reversible adder, which employs four HNG gates, along with additional elements such as a Peres Gate, a Feynman Gate, and an SCLG. Moreover, it integrates an extra HNG Gate, expanding its capability to process information effectively. One notable advantage of this design is its ability to address the fan-out limitation. By utilizing six ancillary input lines and generating ten garbage output lines, it demonstrates a marked improvement in handling data flow and output distribution. It's worth mentioning that the quantum cost of this circuit remains unspecified, leaving room for further evaluation and optimization. Nonetheless, its innovative configuration showcases a promising approach to mitigating computational constraints and advancing quantum computing capabilities.

Components

1) HNG Gate Based 4-bit Reversible Adder:
   - Inputs: X3, X2, X1, X0 (first BCD number) and Y3, Y2, Y1, Y0 (second BCD number).
   - Outputs: S3', S2', S1', S0' (intermediate sum) and CARRYout (carry out from the 4-bit adder).
   - Description: This block consists of 4 HNFG gates, each taking four constant inputs and producing eight garbage outputs, to perform the 4-bit addition. The intermediate sum (S3', S2', S1', S0') and the carry output (CARRYout) are generated here.
2) SCL Gate:
   - Inputs: S3', S2', S1', S0' (intermediate sum) and CARRYout.
   - Outputs: Q, R, S, P.
   - Description: The SCL gate performs correction logic to ensure the sum is within the BCD range. If the intermediate sum exceeds 9, correction logic is applied.
3) Peres Gate (as Half Adder):
   - Inputs: S1', S2' and constant 0.
   - Outputs: P, Q, R.
   - Description: This gate is used to perform the half addition needed during the correction process. It generates a partial corrected sum and intermediate carry.
4) HNG Gate (as Full Adder):
   - Inputs: The outputs of the Peres Gate, the intermediate sum S2', and the carry output from the SCL gate.
   - Outputs: P, Q, R, S (final corrected sum).
   - Description: This gate finalizes the BCD correction by acting as a full adder, combining partial sums and carries to produce the final corrected BCD output.
5) CNOT Gate:
   - Inputs: Outputs from the HNG gate.
   - Outputs: Corrected sum bits.
   - Description: The CNOT gate is used for further logical operations necessary for the BCD correction and final sum generation.

Output

The final outputs of the circuit are the corrected BCD sum (C4, S3, S2, S1, and S0).

3.4 BCD adder design 4: Implementation details

Ripple Carry Adders (RCA) are simple but slow due to linear carry propagation. Their delay is linear, O(n). Carry-Lookahead Adders (CLA) improve speed by computing carries in advance, reducing delay but increasing complexity. Carry-Save Adders (CSA) are efficient for multi-operand addition, though additional steps are needed for the final sum. Parallel prefix adders, including Kogge-Stone, Brent-Kung, and Sklansky, offer logarithmic delay (O(logn)) by computing carries in parallel, with Kogge-Stone being the fastest due to minimal logic depth and balanced load, albeit at a higher gate count. Kogge-Stone is particularly superior for high-speed applications, offering significant performance improvements over other adders.

Figure 21 represents a 4-bit Kogge-Stone adder, which is a type of parallel prefix adder used for binary addition. The Kogge-Stone adder is known for its logarithmic depth and regular structure, making it efficient for high-speed arithmetic operations. The Kogge-Stone Adder (KSA) operates in three distinct stages: pre-processing, carry look-ahead computation, and post-processing.

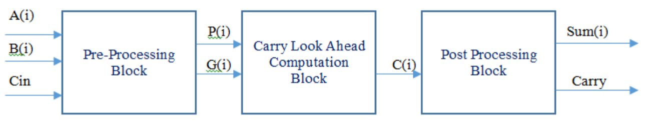


Figure 21. 4-bit Kogge Stone adder

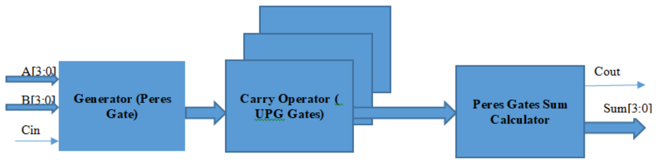


Figure 22. 4-bit reversible Kogge Stone adder

Each stage plays a critical role in the overall functionality and high performance of the adder. In this paper binary Kogge Stone adder is designed using reversible gates as shown in Figure 22. The output is converted in BCD form using a binary to BCD module. A detailed explanation of each stage is as follows:

1) Pre-processing

In the pre-processing stage, generate (G) and propagate (P) signals are computed for each bit pair in the input numbers A and B. This is implemented using Peres Gate.

- **Generate Signal (G)**: This signal indicates whether a given bit pair will generate a carry.

$Gi=Ai.Bi$

  - If both bits Ai and Bi are 1, a carry will be generated, thus Gi=1.
  - If either bit is 0, Gi=0.
- **Propagate Signal (P)**: This signal indicates whether a given bit pair will propagate a carry from a lower bit to a higher bit.

$Pi=Ai\oplus Bi$

- If Ai and Bi different, Pi=1 meaning the carry from the previous bit will be propagated.
- If Ai and Bi are the same, Pi=0.

2) Carry look-ahead computation

This stage is the core of the Kogge-Stone adder, enabling fast carry computation by combining generate and propagate signals using a parallel prefix network. The network consists of multiple levels of logic that combine these signals over increasing ranges of bits. This is implemented using three UPG gates.

- **Combine Propagate Signals (P)**:

$P_{i:j}$=Pi:k−1 and Pk:j

  - This formula computes the propagate signal for a range of bits from i to j.
  - If all bits in the range propagate a carry, then the overall propagate signal for the range is 1.

- **Combine Generate Signals (G)**:

Gi:j=Gi:k−1 or (Pi:k−1 and Gk:j)

  - This formula computes the generate signal for a range of bits from i to j.
  - A carry is generated for the range if either:
    - A carry is generated within the lower part of the range (Gi:k−1).
    - Or a carry is propagated through the lower part (Pi:k−1) and generated in the upper part (Gk:j).

3) Post-processing

In the post-processing stage, the final sum bits are calculated using the propagate signals and the carry-in values computed in the carry look-ahead stage. This is implemented using Peres Gates.

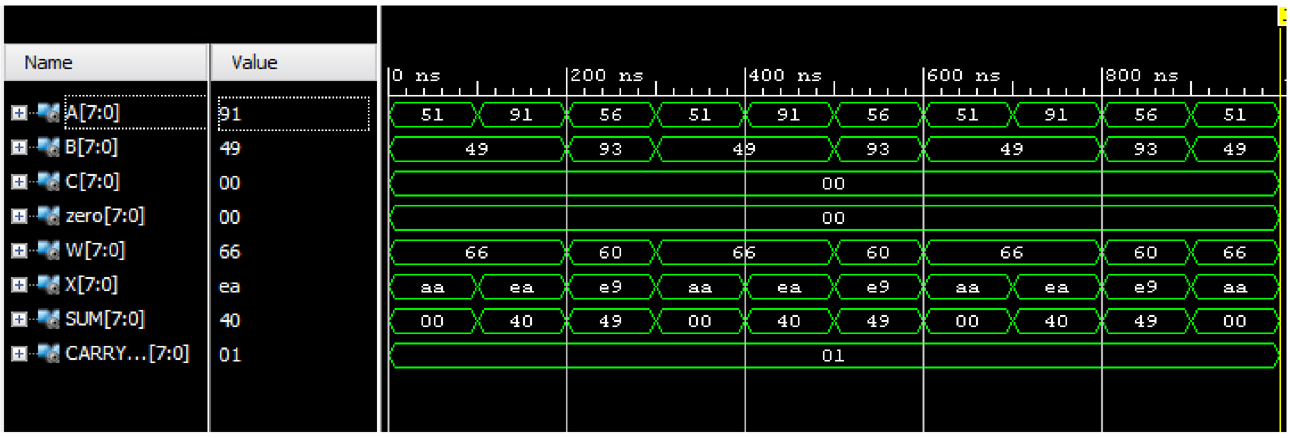
- **Sum Bit (S)**: Si=Pi$\oplus$Ci−1
  - The sum for each bit position i is calculated by XORing the propagate signal for that bit with the carry-in from the previous bit position.

Binary to BCD Converter: This module converts the final sum and carry in BCD form using state machine for segregating digits of different places.

The 4-bit Kogge-Stone adder is designed to perform fast binary addition through efficient carry propagation across multiple stages. Initially, in Stage 0, the adder computes the generate (G[i]=A[i] & B[i]) and propagate (P[i]=A[i]^B[i]) signals for each bit using Peres Gates. These signals indicate whether a carry is generated or propagated for each bit of the input operands A and B. In Stage 1, UPG gates are used to combine the generate and propagate signals from adjacent bits, producing new generate and propagate signals (G1 and P1) for pairs of bits. This process is further refined in Stage 2, where UPG gates combine the results from Stage 1 to handle carry propagation over wider groups of bits, producing G2 and P2 signals. In Stage 3, the final carry signals are determined using AND and OR gates based on the outputs from Stage 2 and the initial carry input (Cin). The final carry-out is produced from these signals. Lastly, in the Sum Calculation stage, the sum for each bit is computed using XOR gates. The propagate signals from Stage 0 are XORed with the carry-out signals to produce the final sum (Sum). This method ensures a fast and efficient addition process by minimizing the delay typically caused by carry propagation, making the Kogge-Stone Adder a highly effective design for high-speed arithmetic operations in digital circuits.

4. RESULTS AND DISCUSSION

The evaluation methodology for the proposed reversible BCD adder designs focuses on delay, power dissipation, and resource utilization. The design is implemented using Xilinx Vivado software, with functional verification performed using its inbuilt ISIM simulator. The FPGA device used was the Artix-7 (device name: xc7a35tcpg236-1), which operates at a frequency of 100MHz. The design was programmed using VHDL/Verilog and synthesized within Vivado, with configuration and bitstream generation executed through the Vivado toolchain. Testing was conducted at the specified 100MHz clock frequency to validate the performance of the design. To strengthen the analysis, experimental results are presented using waveform diagrams, FPGA implementation, and detailed comparisons against existing designs using Xilinx Vivado tool and Artix -7 FPGA board.



Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
A[7:0] 91 — 51 91 56 51 91 56 51 91 56 51
B[7:0] 49 — 49 93 49 93 49 93 49
C[7:0] 00 — 00
zero[7:0] 00 — 00
W[7:0] 66 — 66 60 66 60 66 60 66
X[7:0] ea — aa ea e9 aa ea e9 aa ea e9 aa
SUM[7:0] 40 — 00 40 49 00 40 49 00 40 49 00
CARRY...[7:0] 01 — 01

Figure 23. Waveform of 8 Bit BCD adder Algo 1

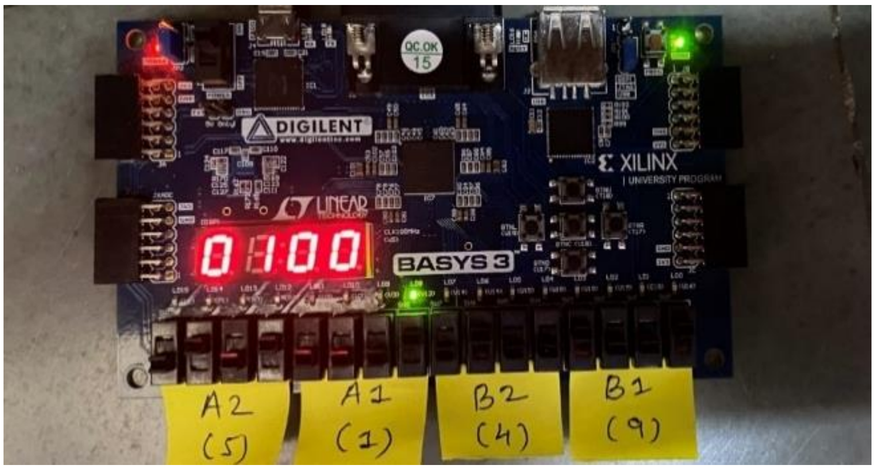


Figure 24. Results on Artix-7 FPGA board

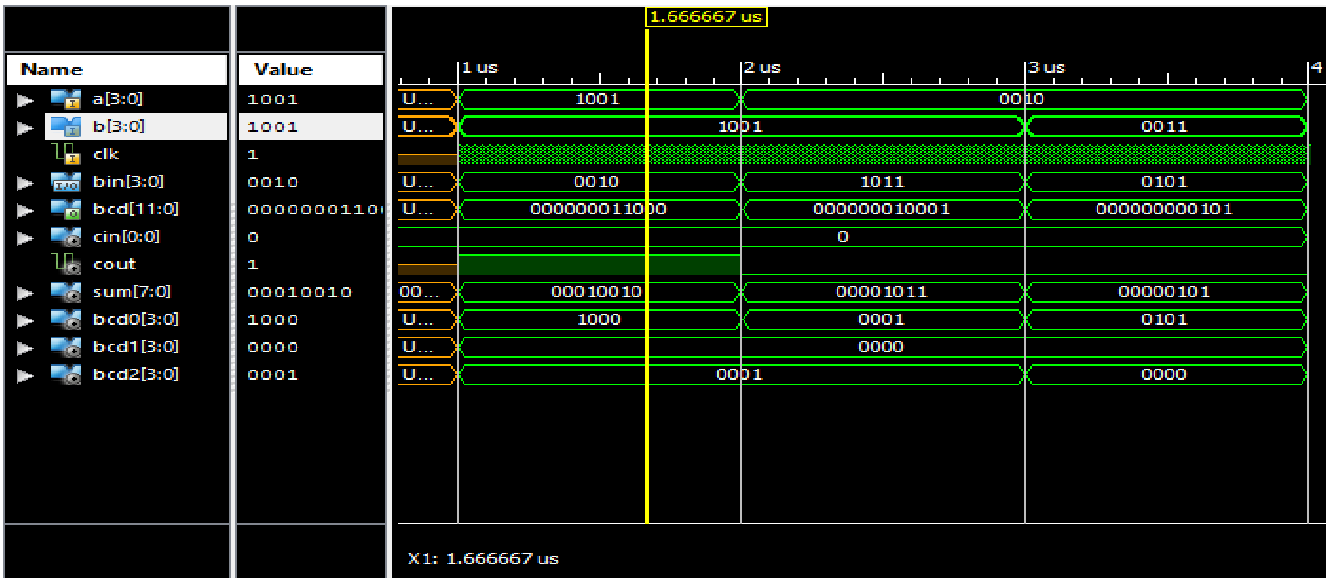


Figure 25. Kogge Stone adder 4 bit

From Figure 23, it is evident that register A contains the BCD value of 51 along with several other values, while register B also holds the BCD value of 49. Following BCD addition principles, the addition operation yields a sum of 00 and generates a carry of 1. Consequently, the final addition result is calculated as 100, as depicted in the waveform analysis. In this waveform A, B, C, zero all are inputs whereas W, X, Sum and carry are all outputs. Zero represents constant inputs set to 0 whereas W and X are garbage outputs. The results are tested on FPGA BASYS 3 [36] board hosting Artix-7 device. The results corresponding to inputs 51 and 49 are shown in Figure 24.

The Figure 25 shows the simulation of Reversible Kogge Stone adder of 4 bit. The inputs are a, b, cin and sum and cout are output. A is fed 9 and B is fed 9 so sum is 8 and cout is 1. This adder is using binary addition rules, but the sum obtained is converted into BCD form using binary to BCD converter module.

Table 4 presents the combinational path delay for varying adder sizes across different algorithms and designs.

The results highlight the following observations:

- The Kogge-Stone adder consistently achieves the lowest path delay due to its efficient parallel-prefix structure, with a delay of 0.948 ns for 4 bits and 12.11 ns for 64 bits.
- While other algorithms (1, 2, and 3) show a progressive increase in delay with the size of the adder, the percentage increase is not significant.
- Algorithm 3 demonstrates improved performance compared to Algorithm 1 and Algorithm 2 for larger bit sizes, making it a competitive option for scalable designs.
- By optimizing circuit design to reduce the number of garbage outputs and gate counts, the delay parameter is significantly minimized.

Table 5 presents the power dissipation of various adder designs as synthesized by Xilinx tools. The reversible Kogge-Stone BCD adder exhibits the least power dissipation, consuming only 24 mW. This is a 70% reduction compared to the conventional irreversible BCD adder, which consumes 80.2 mW. Among the proposed designs, Algorithm 2 has the highest power dissipation at 189 mW, while Algorithm 3 demonstrates better power efficiency at 90 mW. The use of reversible logic in the Kogge-Stone BCD adder drastically reduces power dissipation, aligning with the design's low-energy goals.

Table 6 summarizes resource utilization in terms of garbage values, quantum cost, and constant inputs. The Reversible Kogge-Stone Adder is the most efficient in terms of quantum cost (37) and garbage output (14), offering a good balance between complexity and resource use. Algorithm 3 also shows strong efficiency with low quantum cost (45), garbage output (10), and gate count (8), making it a well-balanced design. Algorithm 2 and Algorithm 1 are more complex, with higher quantum costs (55 and 60, respectively) and garbage outputs (22 and 24), indicating less optimization compared to the other designs.

The results shown in Table 7 shows that the Proposed 8-bit adder shows a significant improvement in quantum cost (50), garbage output (14), gate count (11), and power dissipation (24 mW) compared to the designs from [33, 34], while maintaining a similar delay (1.555 ns). This indicates a more efficient design in terms of both resource utilization and power consumption.

Table 7. Resource utilization for all design

Algorithms	Delay (ns)	Quantum Cost	Garbage Output	Gate Count	Power Dissipation
8-bit	3.623	165	40	64	0.107 W
8-bit	1.55	157	22	11	42 mW
Proposed	**1.555**	**50**	**14**	**11**	**24 mW**

Implementing the proposed reversible BCD adder on the Artix-7 FPGA significantly impacts design choices, scalability, and performance due to the specific characteristics of the FPGA. The Artix-7 FPGA provides high-performance, low-power logic resources, making it an ideal platform for validating the proposed design. However, certain considerations must be addressed during the design process. Resource utilization is a critical factor, as FPGA logic resources such as lookup tables (LUTs), flip-flops, and routing channels directly influence scalability. To ensure efficient utilization of these resources, especially for higher bit-width adders, it is essential to optimize gate counts and minimize garbage outputs. Delay optimization is another important aspect, as the FPGA's architecture introduces specific routing delays between logic blocks. Techniques such as placement constraints and pipelining can be employed to minimize critical path delays, thereby improving performance for larger designs.

Power dissipation is also a key consideration. The Artix-7 FPGA is optimized for low power, which makes it a suitable platform for evaluating the energy-saving benefits of reversible circuits. The proposed design, characterized by reduced gate complexity and fewer garbage outputs, aligns well with the FPGA's capabilities for efficient energy computation. While the design is theoretically scalable to higher bit-widths, practical limitations arise from the finite number of FPGA resources and increased routing congestion in larger circuits. Addressing these challenges may require partitioning the design into modular blocks or exploring hierarchical placement strategies. Furthermore, FPGA-specific optimizations such as utilizing DSP slices for arithmetic operations and customizing carry chains to accelerate carry propagation can enhance performance. Techniques like constraint-driven synthesis in Vivado 14.6 can also help tailor the design to the FPGA's architecture, improving its area, delay, and power metrics.

Testing and ensuring fault tolerance in reversible circuits pose unique challenges due to their inherent characteristics. One challenge lies in the input/output dependency, as reversible circuits require a one-to-one mapping between inputs and outputs. This necessitates testing all possible input combinations to ensure correctness across all output states. Additionally, the presence of garbage outputs complicates fault diagnosis, as it can be difficult to identify whether faults originate in the main functional logic or in the garbage output logic. Despite these challenges, reversible circuits offer opportunities for improved fault tolerance. Error detection is facilitated by the inherent preservation of information in reversible circuits, where faults in outputs often manifest as imbalances in garbage values or unexpected input-output mappings. Fault recovery is another advantage, as reversibility allows circuits to trace operations backward, enabling error correction.

To address these challenges, testing strategies can be employed. Built-in self-test (BIST) circuits embedded within the FPGA implementation can automate fault detection, leveraging the deterministic behavior of reversible circuits. Extensive simulation and validation, including corner-case testing and waveform analysis using Vivado 14.6, ensure functionality under diverse conditions. Physical testing on the Artix-7 FPGA further validates the circuit's fault tolerance and power-performance trade-offs in real-world scenarios. By addressing these testing and fault-tolerance considerations, the reliability and robustness of the proposed reversible design can be significantly enhanced. Although the current work primarily focuses on the design and implementation of the proposed reversible BCD adder, testing and fault tolerance have been identified as important areas for future exploration. These aspects will be thoroughly investigated and addressed in subsequent research efforts.

5. CONCLUSION

In this study, the performance of a Generic Reversible BCD adder, with a focus on the Kogge-Stone design, is thoroughly evaluated in terms of delay, power dissipation, and resource utilization. The results show that the Reversible Kogge-Stone Adder outperforms other algorithms across all key metrics. Specifically, it achieves a significant reduction in delay (1.555 ns for 8-bit design) compared to conventional designs (3.623 ns), and reduces power dissipation by 70%, achieving only 24mW compared to 0.107W for the conventional 8-bit design. Additionally, it demonstrates a quantum cost of 50, garbage output of 14, and a gate count of 11, all of which are more efficient than other algorithms such as Algorithm 1 (with quantum cost 60, garbage output 24, and gate count 16).

While the proposed design shows a slightly higher delay compared to conventional designs, especially as the adder size increases, the reduction in power consumption due to the reversible nature of the logic operations offers a substantial advantage. The results highlight the trade-offs between delay, power, and resource utilization, providing insight for optimizing designs based on specific application needs. The proposed design also offers a balance between performance and energy efficiency, paving the way for high-performance, sustainable computing systems.

REFERENCES

[1] Mollick, E. (2006). Establishing Moore's Law. IEEE Annals of the History of Computing, 28(3): 62-75. https://doi.org/10.1109/MAHC.2006.45

[2] Landauer, R. (1961). Irreversibility and heat generation

in the Computational Process. IBM Journal of Research and Development, 5(3): 183-191. https://doi.org/10.1147/rd.53.0183

[3] Bennett, C.H. (1973). Logical reversibility of computation. IBM Journal of Research and Development, 17(6): 525-532. https://doi.org/10.1147/rd.176.0525

[4] Rani, S., Bhandari, A.S. (2015). A survey on reversible logic gates. International Journal of Computer Applications, 7: 1-3. https://www.ijcaonline.org/proceedings/icaet2015/number7/22249-4086/.

[5] Yelekar, P.R., Chiwande, S.S. (2011). Introduction to reversible logic gates and its application. In 2nd National Conference on Information and Communication Technology (NCICT), pp. 5-9.

[6] Biswas, P., Gupta, N., Patidar, N. (2014). Basic reversible logic gates and it's QCA implementation. International Journal of Engineering Research and Applications, 4(6): 12-16.

[7] Nayak, S., Nayak, S., Singh, J.P. (2013). An introduction to basic logic gates for quantum computer. International Journal of Advanced Research in Computer Science and Software Engineering, 3(10): 163-171.

[8] Thapliyal, H., Ranganathan, N. (2010). Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs. ACM Journal on Emerging Technologies in Computing Systems, 6(4): 1-31. https://doi.org/10.1145/1877745.1877748

[9] Babu, H.M.H., Chowdhury, A.R. (2005). Design of a reversible binary coded decimal adder by using reversible 4-bit parallel adder. In 18th International Conference on VLSI Design Held Jointly with 4th International Conference on Embedded Systems Design, Kolkata, India, pp. 255-260. https://doi.org/10.1109/ICVD.2005.74

[10] Thapliyal, H., Kotiyal, S., Srinivas, M.B. (2006). Novel BCD adders and their reversible logic implementation for IEEE 754r format. In 19th International Conference on VLSI Design Held Jointly with 5th International Conference on Embedded Systems Design (VLSID'06), Hyderabad, India, p. 6. https://doi.org/10.1109/VLSID.2006.122

[11] James, R.K., Shahana, T.K., Jacob, K.P., Sasi, S. (2007). A new look at reversible logic implementation of decimal adder. In 2007 International Symposium on System-on-Chip Tampere, Finland, pp. 1-4. https://doi.org/10.1109/ISSOC.2007.4427442

[12] Haghparast, M., Navi, K. (2008). A novel reversible BCD adder for nanotechnology-Based systems. American Journal of Applied Sciences, 5(3): 282-288. https://doi.org/10.3844/ajassp.2008.282.288

[13] Islam, M.S., Hafiz, M.Z., Begum, Z. (2012). Quantum cost efficient reversible BCD adder for nanotechnology-Based systems. International Journal of Computer and Electrical Engineering, 4. https://doi.org/10.7763/IJCEE.2012.V4.443

[14] Toffoli, T. (1980). Reversible computing. Lab for Computer Science, Massachusetts Institute of Technology, Cambridge, MA. MIT/LCS/TM-151. https://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TM-151.pdf.

[15] Nagamani, A.N., Ashwin, S., Agrawal, V.K. (2015). Design of optimized reversible binary and BCD adders. In 2015 International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), Bengaluru, India, pp. 1-5. https://doi.org/10.1109/VLSI-SATA.2015.7050488

[16] Feynman, R.P. (1985). Quantum mechanical computers. Optics News, 11(2): 11-20. https://doi.org/10.1364/ON.11.2.000011

[17] Mamtaj, S., Saha, D., Banu, N. (2014). A review of reversible gates and its application in Logic Design. American Journal of Engineering Research, 3(4): 151-161.

[18] Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P. (2003). Synthesis of reversible logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 22(6): 710-722. https://doi.org/10.1109/TCAD.2003.811448

[19] Tiwari, K., Khopade, A., Jadhav, P. (2011). Optimized carry look-ahead BCD adder using reversible logic. In Technology Systems and Management: First International Conference, ICTSM 2011, Mumbai, India, pp. 260-265. https://doi.org/10.1007/978-3-642-20209-4_37

[20] NagaBabu, T., Sounder, D., Rao, B.S., Babu, P.B. (2012). A low power adder using reversible logic gates. International Journal of Research in Engineering and Technology, 1(3): 244-247. https://doi.org/10.15623/ijret.2012.0103007

[21] Khan, M.M.A. (2002). Design of full-adder with reversible gates. In International Conference on Computer and Information Technology, Dhaka, Bangladesh, pp. 515-519.

[22] Bruce, J.W., Thornton, M.A., Shivakumaraiah, L., Kokate, P.S., Li, X. (2002). Efficient adder circuits based on a conservative reversible logic gate. In Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002, Pittsburgh, PA, USA, pp. 83-88. https://doi.org/10.1109/ISVLSI.2002.1016879

[23] Kamalakannan, V., Shilpakala, V., Ravi, H.N. (2013). Design of adder/subtractor circuits based on reversible gates. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(8): 3796-3804.

[24] Vanusha, P., Vally, K.A. (2014). Low power computing logic gates design using reversible logic. International Journal of Application or Innovation in Engineering & Management, 3(10): 123-129.

[25] Lala, P.K., Parkerson, J.P., Chakraborty, P. (2010). Adder designs using reversible logic gates. WSEAS Transactions on Circuits and Systems, 9(6): 369-378.

[26] Babu, H.M.H., Islam, M.R., Chowdhury, A.R., Chowdhury, S.M.A. (2003). Reversible logic synthesis for minimization of full-adder circuit. In Euromicro Symposium on Digital System Design, 2003. Proceedings. Belek-Antalya, Turkey, pp. 50-54. https://doi.org/10.1109/DSD.2003.1231899

[27] Mishra, N., Wairya S., Sen, B., (2018). Design of conservative, reversible sequential logic for cost efficient emerging nano circuits with enhanced testability. Ain Shams Engineering Journal, 9(4): 2027-2037. https://doi.org/10.1016/j.asej.2017.02.005

[28] Krishnaveni, D., Geetha Priya, M. (2011). Novel design of reversible serial and parallel adder/subtractor. International Journal of Engineering Science and

Technology, 3(3). https://www.researchgate.net/publication/50984906.

[29] Akilandeswari, A. (2021). Reversible logic design for adder in CMOS VLSI design. International Journal of Creative Research Thoughts, 9(2): 118-139. https://ijcrt.org/papers/IJCRT2102016.pdf.

[30] Chu, Z., Li, Z., Xia, Y., Wang, L., Liu, W. (2020). BCD adder designs based on three-input XOR and majority gates. IEEE Transactions on Circuits and Systems II: Express Briefs, 68(6): 1942-1946. https://doi.org/10.1109/TCSII.2020.3047393

[31] Borodzhieva, A., Tsvetkova, I., Zaharieva, S., Dimitrov, D., Mutkov, V. (2021). Inquiry-Based learning used for implementation of BCD adders in the Course" digital electronics". In 2021 IEEE 27th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, pp. 150-155. https://doi.org/10.1109/SIITME53254.2021.9663683

[32] Ykuntam, Y.D., Prasad, S.H. (2021). A modified high speed and less area BCD adder architecture using Mirror adder. In 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, pp. 624-627. https://doi.org/10.1109/ICOSEC51865.2021.9591842

[33] Krishna, N., Uma, A. (2020). Design and analysis of low power reversible adder/subtractor circuits. International Journal of Engineering Research & Technology, 9(9): 576-585. https://doi.org/10.17577/IJERTV9IS090366

[34] Kumar, A.S., Pratap, N.L., Ramya, A., Upendra, V., Venugopal, E., Abhishek, P. (2024). An efficient BCD adder design utilizing reversible logic techniques. In 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, pp. 1-5. https://doi.org/10.1109/IACIS61494.2024.10721687

[35] Tiwari, K.S., Kadam, R.S., Dudhedia, M.A., Pansare, J.R., Khedkar, S.P., Gawande, S.H. (2024). Reversible logic gates and applications-A low power solution to VLSI chips. Mathematical Modelling of Engineering Problems, 11(3): 705-720. https://doi.org/10.18280/mmep.110315

[36] Basys3™ FPGA Board Reference Manual. https://www.amd.com/content/dam/amd/en/documents/university/aup-boards/XUPBasys3/documentation/Basys3_rm_8_22_2014.pdf, accessed on Dec. 20, 2023.