Design of Reversible Arithmetic Logic Unit with Built-In Testability

Hari Mohan Gaur, Ashutosh Kumar Singh, and Umesh Ghanekar
National Institute of Technology (NIT) Kurukshetra

Editor's note:
This article presents an advanced reversible arithmetic logic unit with built-in testability for single bit-flip errors.

—Shreyas Sen, Purdue University

The minimization of power levels is an important and growing area of research due to a faster decay of the natural sources of energy. Reversible logic is one of the alternative techniques that minimizes the current rate of energy consumption globally with its application to several emerging technologies. These circuits are theoretically proven for providing nearly energy-free computation by preventing loss of information in the form of heat [1]. A remarkable development has also been noticed in the investigational advancement of reversible circuits, where the researchers are proposing several combinational and sequential logic circuits on the top of distinguished logic gates and families. Testability has also been a major concern to validate its functionality. A variety of reversible gates based arithmetic logic unit (ALU) realizations using quantum-dot cellular automata (QCA) [2], [4], deoxyribonucleic acid (DNA) [3], new/basic gates [5]–[7], and traditional CMOS transistors [8] have been proposed in the literature. To the best of our knowledge, none of the reversible ALUs are present in the literature, which is integrated with inbuilt testability. However, there are several testing methodologies available for reversible circuits [9]. Most of the approaches are based on parity preservation and generation, as reversible circuits have the same number of inputs and outputs. The presented approaches require a large amount of hardware to induce testability in an original circuit, which drastically increases power consumption levels and the principle cost of manufacturing.

When parity-preserved architecture is used with an arbitrary design methodology, it ensures detection of faults that occurred due to single bit-flip in logic circuits without additional hardware [10]. However, the method is not sufficient for the detection of all types of fault models in reversible circuits, viz, stuck-at, bridging, missing gate, cross-point, and cell faults. At this point of unsatisfiability, there are two different viewpoints that can be used to acknowledge the applicability of a method for the detection of these faults using parity preservation. First, parity-preserving gates and parity-preserving architectures (considered as a single gate) are used for designing circuits, as shown in Figure 1. The circuit represents k individual blocks (PPG_1, PPG_2, … PPG_k). These blocks can be parity-preserving gates or a combination of gates that is treated as a single gate. The detection of the missing gate and cross-point faults cannot be detected in this case. Second, when nonparity-preserving gates (preferably Toffoli gates) are used to design parity-preserving architectures, it provides a better solution to a single occurrence of any type of fault

Digital Object Identifier 10.1109/MDAT.2019.2919017
Date of publication: 27 May 2019; date of current version: 1 October 2019.

model. However, online testing of reversible circuits deals with errors occurring due to change of logic values on the wires of the circuit, which are commonly known as bit-flip faults. Parity-preserving gates and architectures provide full coverage of single bit-flip faults in the circuit [11], [12].

In this context, we present a novel parity-preserving structure of ALU comprising two functional units. This circuit utilizes Fredkin gates and a two-fold gate placement method (wherever Toffoli gates are used) to attain full coverage of single bit-faults and efficient utilization of gates [12]. The circuit has scalability up to *N*-bit operations and provides full coverage of errors occurring due to single bit-flip faults.

Arithmetic logic unit

The basic building block in the proposed scheme is engaged in the development of a control unit (CU) and a full adder (FA), followed by a parity checker for the fault-detection process. The CU is designed using Fredkin gates, and an FA is recreated using Toffoli gates over a two-fold placement methodology.

Fredkin gate

A Fredkin gate comprises one control input (k) and two target inputs T_1 and T_2, as shown in Figure 2a, to form a 3×3 reversible gate. The control input follows the same input-to-output relationship. The output functions $f_1(k, T_1, T_2)$ and $f_2(k, T_1, T_2)$ can be calculated by

$$f_j(k,T_1,T_2) = k_1 \cdot T_1 + k_2 \cdot T_2 \begin{cases} \text{for } j=1\ ; k_1 = \bar{k},\ k_2 = k \\ \text{for } j=2\ ; k_1 = k,\ k_2 = \bar{k} \end{cases}. \tag{1}$$

Lemma 1: Fredkin gate is parity preserving.

Proof: Using (1), the two output functions of a Fredkin gate can be calculated by (2) and (3), respectively.

$$f_1(k,T_1,T_2) = \bar{k} \cdot T_1 + k \cdot T_2 \tag{2}$$

$$f_2(k,T_1,T_2) = k \cdot T_1 + \bar{k} \cdot T_2 \tag{3}$$

A gate or a circuit is said to be parity preserving when EXOR of all the inputs (I) and outputs (O) is 0. The calculation for the same is depicted in (4), which proves a parity-preserving property of Fredkin gates.

$$\begin{aligned} I \oplus O &= [k \oplus T_1 \oplus T_2] \oplus [k \oplus f_1(k,T_1,T_2) \\ &\quad \oplus f_2(k,T_1,T_2)] \\ &= [k \oplus T_1 \oplus T_2] \oplus [k \oplus (\bar{k} \cdot T_1 + k \cdot T_2) \\ &\quad \oplus (k \cdot T_1 + \bar{k} \cdot T_2)] \\ &= [k \oplus T_1 \oplus T_2] \oplus [k \oplus T_1 \oplus T_2] = 0 \end{aligned} \tag{4}$$

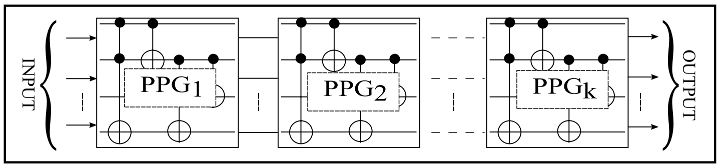

Figure 1. Example circuit.

Two-fold gate placement method

A Toffoli gate has two control inputs (k_1, k_2) and one target input T, as shown in Figure 2b, to form a 3×3 reversible gate. The control inputs also follow the same input-to-output relationship, and the output function $f(k_1, k_2, T)$ can be calculated by (5). The gate with one control (k), shown in Figure 2c, is called a CNOT gate (2×2 Toffoli gate), whose output $f(k, T)$ can be calculated by (6).

$$f(k_1,k_2,T) = (k_1 \cdot k_2) \oplus T \tag{5}$$

$$f(k,T) = k \oplus T \tag{6}$$

Lemma 2: A circuit containing two Toffoli gates having control inputs on the same wire and target input on different wires is parity preserving.

Proof: For a circuit containing two Toffoli gates with the same control input (k) and different target inputs (T_1 and T_2), the two generated output functions $f_1(k_1, k_2, T_i)$ and $f_2(k_1, k_2, T_i)$ are given by

$$f_i(k_1,k_2,T_i) = (k_1 \cdot k_2) \oplus T_i; i = \{1,2\}. \tag{7}$$

The EXOR of all the inputs (I) and outputs (O) of the circuit depicted in (8) is 0, which shows that the circuit is parity preserving.

$$\begin{aligned} I \oplus O &= [k_1 \oplus k_2 \oplus T_1 \oplus T_2] \oplus [k_1 \oplus k_2 \\ &\quad \oplus f_1(k_1,k_2,T_2) \oplus f_2(k_1,k_2,T_2)] \\ &= [k_1 \oplus k_2 \oplus T_1 \oplus T_2] \oplus [k_1 \oplus k_2 \\ &\quad \oplus \{(k_1 \cdot k_2) \oplus T_1\} \oplus \{(k_1 \cdot k_2) \oplus T_2\}] \\ &= [k_1 \oplus k_2 \oplus T_1 \oplus T_2] \\ &\quad \oplus [k_1 \oplus k_2 \oplus T_1 \oplus T_2] = 0 \end{aligned} \tag{8}$$

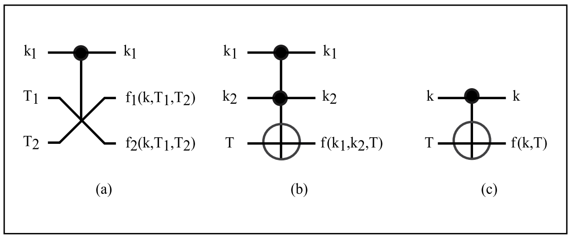

Figure 2. Reversible gates. (a) Fredkin gate. (b) Toffoli gate. (c) CNOT gate.

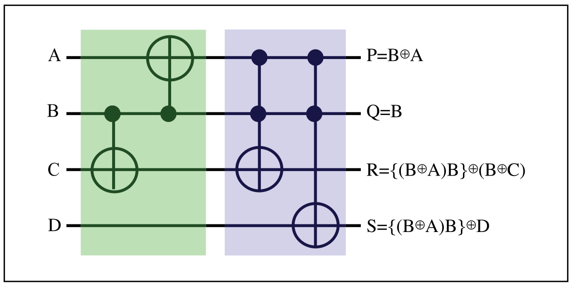

Figure 3. Two-fold gates placement model.

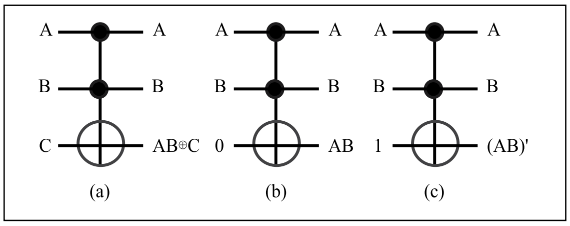

Figure 4. Boolean function generation using reversible gates. (a) Toffoli gate. (b) AND operation. (c) NAND operation.

The design methodology is also shown in Figure 3, where two sets of gates placements are shown on four wires with inputs A, B, C, D and outputs P, Q, R, S. In the first set (shown in green), a CNOT gate (2×2 Toffoli gate) is placed with its control input on wire B and target input on wire C. The control input of the second CNOT gate of this set is the same as that of the previous gate, and the target input can take any place except wire C to remove redundancy. Similarly, the second set (shown in blue) is created using 3×3 Toffoli gates. It can be calculated from the input–output of the circuit that the EXOR of inputs and outputs, i.e., $(A \oplus B \oplus C \oplus D) \oplus (P \oplus Q \oplus R \oplus S)$, is 0. It confirms the parity-preserving property of the circuit as well as of the utilized design methodology.

Proposed design of ALU

The implementation of a Boolean function using reversible gates is achieved using several ancilla inputs (AIs) and garbage outputs (GOs) for maintaining the same number of inputs and outputs. AIs are referred to as constant input wire and GOs are the unused outputs. Consider a 3×3 Toffoli gate, as shown in Figure 4a, with inputs A, B, C for producing AND and NAND functions. The input to the last wire C should be kept at constant logic 0 to produce AND operation of inputs A and B, as shown in Figure 4b. Similarly, the input to C should be kept at constant logic 1 to produce NAND operation, as shown in Figure 4c. In these two cases, the output to the first two wires of the gate remains unused. These AIs and GOs are two of the cost metrics that are used to analyze the efficiency of any circuit and design and methodology [13]. Several constant inputs are also utilized and some outputs remain unused to obtain desired logic function in the proposed ALU circuit. Full attention has been given to minimize the number of these cost metrics in present design methodology.

The proposed circuit of ALU is shown in Figure 5, where CU and FA are shown in two separate blocks. The three outputs of the CU (X, Y, Z) are fed into FA as input. Here, A and B denote the primary inputs to the circuit for which the operations are to be performed, and (S_0, S_1, S_2) are the three selection lines that are used to select the operation to be performed. ($C_1, C_2, \ldots, C_{12}$) are the constant inputs (AIs) that are assigned (0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0) logic values in the same sequence as they are appearing in the circuit. C_{in} is the carry input of the FA, which is also used as a selection line. ($G_1, G_2, \ldots, G_{16}$) denote the GOs of the circuit, $f(A, B, C_{in})$ is the final output of the ALU, and C_{out} represents the carry output that is utilized during addition operation. The intermediate outputs X, Y, and Z are given by (9), (10), and (11), respectively. The final output and the carry

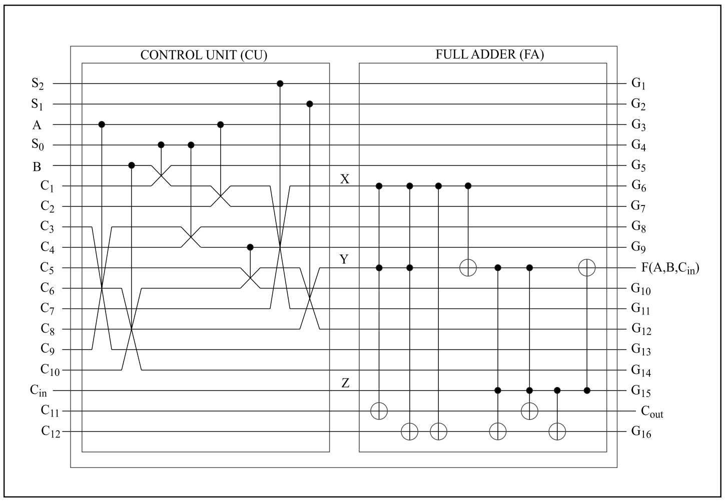

Figure 5. ALU circuit realization.

Table 1. Functional table of ALU.

S_2	S_1	S_0	C_{in}	X	Y	Z	F	Operation
0	0	0	0	A	B	0	A⊕B	XOR
0	×	1	0	A+B	AB / 0	0	A+B	OR
0	1	0	0	A	0	0	A	Transfer A
1	0	0	0	0	B	0	B	Transfer B
1	0	1	0	0	AB	0	AB	AND
1	1	×	0	0	0	0	0	Reset
0	0	0	1	A	B	1	A⊙B	XNOR
0	×	1	1	A+B	AB / 0	1	$\overline{A+B}$	NOR
0	1	0	1	A	0	1	$\overline{A}$	Invert A
1	0	0	1	0	B	1	$\overline{B}$	Invert B
1	0	1	1	0	AB	1	$\overline{AB}$	NAND
1	1	×	1	0	0	1	1	Set

can be calculated using (12) and (13), respectively. Based on the equations, the proposed ALU performs 12 arithmetic and logic operations, which are listed in Table 1, where $f(A, B, C_{\text{in}}) = F$ and EXNOR operation can also be generalized as a sum with carry.

$$X = \overline{S_2}\,(A + S_0 B) \tag{9}$$

$$Y = \overline{S_1}B\left(\overline{S_0} + S_0 A\right) \tag{10}$$

$$Z = C_{\text{in}} \tag{11}$$

$$f(A, B, C_{\text{in}}) = X \oplus Y \oplus Z \tag{12}$$

$$C_{\text{out}} = X \cdot Y + Y \cdot Z + Z \cdot X \tag{13}$$

Preposition 1: Proposed ALU is parity preserving.

Proof: The design consisted of two separate circuits, namely, CU and FA. CU is constructed using Fredkin gates and FA is designed using two-fold Toffoli gate structures. Consider a circuit containing M gates on n wires. If $(I_1, I_2, \dots I_n)$ are the inputs and $(O_1, O_2, \dots O_n)$ are the outputs of the circuit and the intermediate stages are $(O_{i1}, O_{i2}, \dots O_{\text{in}})$, where $i = (1–n)$.

The intermediate I/O mapping is given by

$$[O_{11}, O_{12}, \dots O_{1n}\ ] \rightarrow [O_{21}, O_{22}, \dots O_{2n}] \rightarrow \dots [O_{M1}, O_{M2}, \dots O_{Mn}].$$

Since intermediate stages are parity preserving, the EXOR of the previous stage (P_s) and the next stage (N_s) can be given by

$$P_s \oplus N_s = 0 \forall \begin{cases} P_s = O_{M1} \oplus O_{M2} \oplus \dots O_{Mn} \\ N_s = O_{(M+1)1} \oplus O_{(M+1)2} \oplus \dots O_{(M+1)n} \end{cases}. \tag{14}$$

For $N_s = (O_{11} \oplus O_{12} \oplus \dots O_{1n})$,

$P_s = (I_1 \oplus I_2 \oplus \dots I_n)$

If $P_s \oplus N_s = 0 \forall i = (1 \text{ to } N)$

Then, $(I_1 \oplus N_2 \oplus \dots I_n) \oplus (O_1 \oplus O_2 \oplus \dots O_n) = 0$.

It shows that the circuits formed using parity-preserving gates and structures are also parity preserving. Hence, CU and FA are parity preserving, which conclude that the proposed ALU is also parity preserving.

N-bit ALU: Scaling the bit-width

The scheme of designing can be extended for N-bit operations, where N is the pair of input variables in which the operations have to be performed. For instance, 1-bit means there are two input variables (say A_1, B_1). For Nth order operations, the input variables are $(A_1, A_2, \dots A_N)$ and $(B_1, B_2, \dots, B_N)$. The operations are performed between Nth values of A and B. Hence, N 1-bit ALU will be consumed. However, the selection line (S_0, S_1, S_2) will be common for each ALU block. The schematic representation for the same is shown in Figure 6. The corresponding outputs are given by $\{F_1 = F_1(A_1 + B_1), F_2 = F_2(A_2, B_2), \dots, F_N = F_N(A_N, B_N)\}$. During the intrusion of carry input (C_{in}), the circuit will become an N-bit reversible ripple carry adder (RCA) when carry output (C_{out}) of each ALU block is taken as carry input to the next ALU block. The C_{out} of the last block will provide the final carry out.

Fault detection

Fault detection can be accomplished by checking the input and output parity of the circuit.

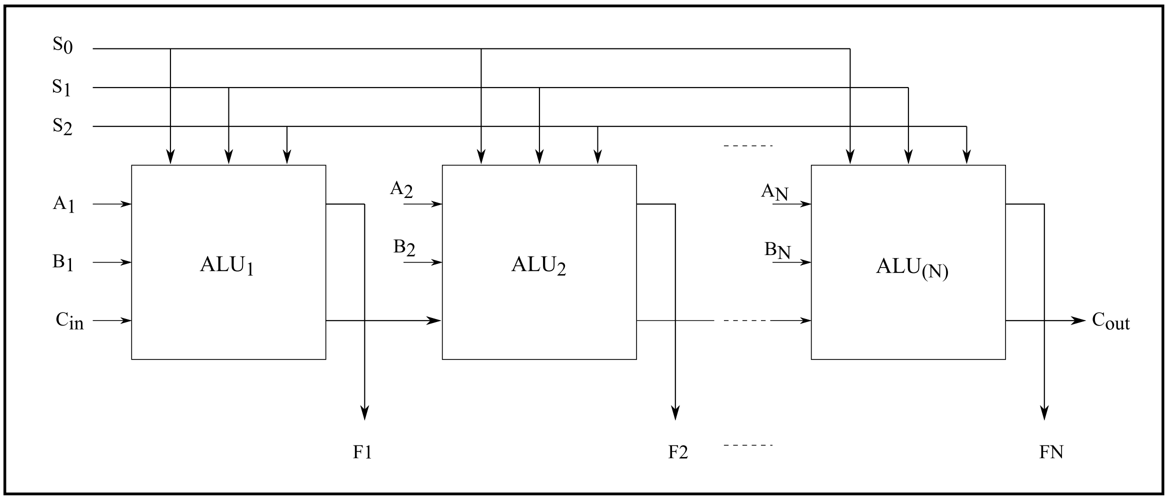

Figure 6. N-bit ALU.

In reversible circuits, it can be achieved by cascading CNOT gates from each wire of the circuit to a new constant input test wire (T_{in}) before and after the whole circuit. The illustration of parity checking for the proposed ALU is also shown in Figure 7. The output of the new wire (T_{out}) can be given by (15), where Σ symbolizes EXOR sum rather than a mathematical sum. Note that there is no requirement to preserve the parity of the input containing constant 0 as input. The gates can be removed from the input CNOT array for the reduction of gate cost (GC).

$$T_{\text{out}} = \left(\sum_{i=1}^{16} I_i\right) \oplus \left(\sum_{i=1}^{16} O_i\right) \oplus T_{\text{in}} \qquad (15)$$

Preposition 2: T_{out} will flip from logic 0 to 1 for any single bit-flip fault occurrence for the proposed ALU scheme when $T_{\text{in}} = 0$.

Proof: If $(I_1, I_2, \ldots I_n)$ input and $(O_1, O_2, \ldots O_n)$ output circuits are incorporated with a parity checker as per considered fault detection procedure, T_{out} can be given by (16) for $T_{\text{in}} = 0$. Since the circuit is parity preserving, $T_{\text{out}} = 0$ for fault-free conditions. Consider any bit-flip fault occurred at any level of the circuit. As each block of the circuit is also parity preserving, the same parity information will be transferred to the next level. Hence, the values at the output will be inverted in odd numbers. Considering $O_1 \rightarrow \overline{O_1}$, T_{out} will flip to 1 and the fault occurrence can be detected, as shown in (17).

$$T_{\text{out}} = \left[(I_1 \oplus I_2 \oplus \ldots I_n) \oplus (O_1 \oplus O_2 \oplus \ldots O_n)\right] \quad (16)$$

$$\begin{aligned} T_{\text{out}} &= \left[(I_1 \oplus I_2 \oplus \ldots I_n) \oplus \left(\overline{O_1} \oplus O_2 \oplus \ldots O_n\right)\right] \\ &= \left[O_1 \oplus \overline{O_1}\right] = 1 \end{aligned} \quad (17)$$

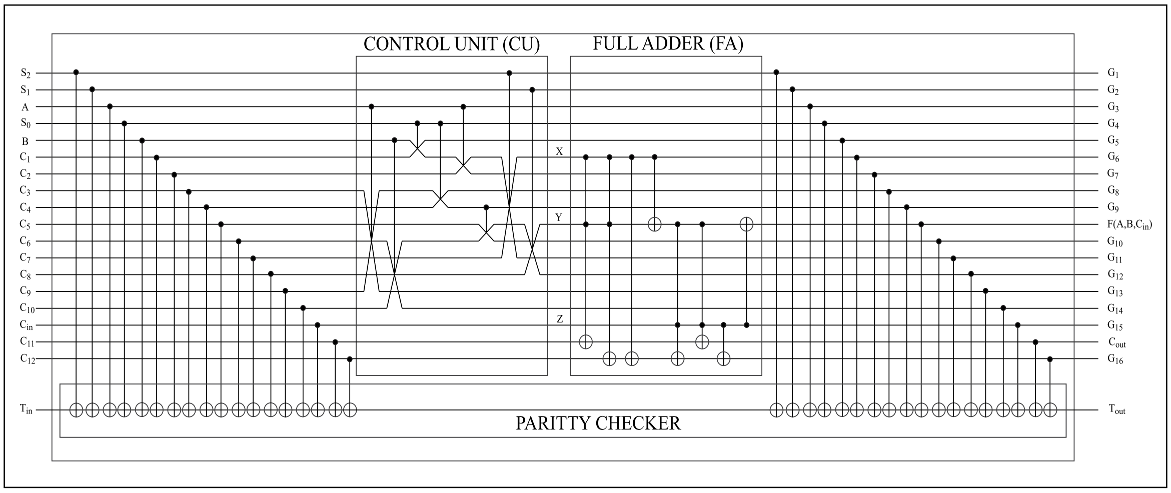

Figure 7. Testable ALU realization.

Table 2. Implementation results.

Proposed Circuit	n	GC	QC	GO	AI
CU	16	8	24	13	10
FA	5	8	24	3	2
1-bit ALU	18	16	48	16	12
4-bit ALU	60	64	192	52	48
8-bit ALU	116	128	384	100	96
16-bit ALU	228	256	768	196	192
32-bit ALU	452	512	1536	388	384
64-bit ALU	900	1024	3072	772	768

Hence, T_{out} will flip from logic 0 to 1 for any single bit-flip fault occurrence for the proposed ALU having inputs $(I_1, I_2, \ldots, I_{16})$, including constant input values and outputs $(O_1, O_2, \ldots, O_{16})$.

Performance analysis

The performance of the proposed design scheme is evaluated on the basis of measures defining the operating cost of reversible circuits, number of operations that can be performed, and testability incorporation. We have taken five existing ALU circuits (ALU1–ALU5) that were realized using reversible gates for comparison [2]–[6]. However, the technologies used for implementation in ALU1 [2] and ALU3 [4] is QCA, and ALU2 [3] is DNA cells, but the basic realization has been achieved using reversible gates.

Implementation

The circuits of 1-bit, 4-bit, 8-bit, 16-bit, 32-bit, and 64-bit are designed by creating Toffoli Fredkin cascade files and implemented using RC-viewer [14] for the calculations of operating costs in terms of number of inputs (n), GC, quantum cost (QC), GO, and AI. The corresponding results are listed in Table 2.

The work presented in ALU1 [2], ALU3 [4], and ALU4 [5] propose new reversible gates based structure, namely, RUG, RMG, and PAOG gates, respectively. These gates are first synthesized on Revkit [15] to produce their equivalent Toffoli circuits and then realized using RC-viewer [14] to obtain the results. It is found that RUG is realized using eight gates with QC 20. RMG and PAOG are realized using four gates with QC 20 and 6, respectively. The work presented in ALU2 [3] and ALU5 [6] utilizes basic Toffoli and Fredkin gates. For a fair comparison, corresponding 4-bit ALU architectures are implemented on the same platform and the results are compared with the present work, which is demonstrated in Table 3.

Table 3. Comparison with existing ALUs.

Parameter	ALU5 [6]	ALU4 [5]	ALU3 [4]	ALU2 [3]	ALU1 [2]	Proposed ALU
n	58	25	53	66	40	60
GC	80	48	164	80	160	64
QC	233	88	740	244	544	192
GO	49	14	41	71	29	52
AI	52	8	40	56	24	48

It can also be noticed that the number of gates and wires in the table is larger as they appear in the article for ALU2. Here, two extra gates for the primary inputs A and B, one gate for $\overline{B}$, and three gates for OR operation in the logic unit are required. It is calculated that, in the present work, a reduction in operating cost by 48%, 20%, and 60% in comparison to most recent methodologies is presented for designing ALU1, ALU2, and ALU3, respectively, when all the values are combined together.

Number of operations

The performance of an ALU is also defined by the number of operations that can be performed by its single unit. These operations include several binary transformations and mathematical evaluations that are governed by number of selection lines. For a typical ALU structure, the number of operations should be as large as possible, keeping minimum selection lines. There are a total of 12 operations listed in Table 1, which can be performed by proposed ALU using four selections lines. The comparison on this basis is provided in Table 4 with respect to considered existing ALU circuits. ALU1 [5] is also capable of performing 12 operations but requires five selection lines. The number of selection lines used in ALU3 [3] is minimum (=3), but it can be utilized to perform only seven operations.

Testability

The incorporation of testability features is one of the key components to justify the reliability. It is noticed that none of the existing techniques shows integrated testability features as shown in the last column of Table 4. However, they can be modified using prior testable techniques, but the operating cost will dramatically increase as a consequence. The proposed ALU circuit is parity preserving, which itself shows its in-built testability features. The procedure of inclusion of input and output CNOT arrays for

Table 4. Generalized comparison.

Parameter	ALU5 [6]	ALU4 [5]	ALU3 [4]	ALU2 [3]	ALU1 [2]	Proposed ALU	Max improvement*
n	13N+6	5N+5	12N+5	16N+2	9N+4	14N+4	9%
GC	20N	12N	41N	20N	40N	16N	61%
QC	57N+5	22N	185N	61N	136N	48N	74%
GO	12N+1	3N+2	9N+5	17N+3	6N+5	12N+4	27%
AI	13N	2N	10N	14N	6N	12N	14%
Operations	12	6	7	7	11	12	50%
Selection lines	5	5	3	4	4	4	20%
Testability	×	×	×	×	×	✓	-
%age Improvement in TOC w.r.t. proposed ALU*	12%	54%	60%	20%	48%	-	60%

*Calculations are based on 4-bit ALU

fault detection uses one extra wire and increases the gate and QC of the designed circuit by $2n$.

Comparison

The generalized equations for finding the operating costs on the basis of the number of gates used for designing respective ALUs are demonstrated in Table 4 along with the number of operations that can be performed by them and testability incorporation. The last column of this table gives the maximum improvement of the proposed ALU with respect to existing ALU circuits. Moreover, a parameter total operating cost (TOC) is included, which is the sum of n, GC, QC, GO, and AI. Number of operations, selection lines, and testability are ignored in this case. The percentage improvement in the TOC is also listed in the last row of the table. Specifically, the TOC consumed by the proposed ALU is $(102N+8)$, the highest consumption can be seen in the case of ALU3, which is (257+10), and the minimum values are for ALU4, i.e., $(44N+7)$. We have achieved a reduction of $(155N+2)$ with respect to ALU3. If we talk about the GC, which is the major metrics of comparing logic circuits, we have achieved a reduction of 60%, 20%, 61%, and 20% from ALU1, ALU2, ALU3, and ALU5, respectively. The operating cost of ALU4 is smaller than all existing and the presented work, but it is applicable for only six operations using five selections lines. However, the presented ALU is capable of performing 12 operations using four selection lines.

A NEW FREDKIN and Toffoli gates based built-in testable N-bit reversible ALU is presented to correspond with future generation microprocessors. A major role is played by the creation of parity preserving and efficient set of CU and FA for incorporating testability features in overall circuit realization. The methodology provides a detection of errors occurring due to single bit-flip in the circuit at lower overheads. The results also show a large reduction in operating costs as compared to prior work in this area. Diminution in the size, design complexity, and power requirements can be achieved as a consequence. Efficient designs of testable data path elements using reversible logic gates will be realized for further extension of the work. ■

References

[1] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Develop.*, vol. 17, no. 66, pp. 525–532, 1973.

[2] T. N. Sasamal, A. K. Singh, and A. Mohan, "Efficient design of reversible ALU in quantum-dot cellular automata," *Int. J. Light Elec. Opt.*, vol. 127, pp. 6172–6182, 2016.

[3] A. Sarker, H. M. H. Babu, and M. Rashid, "Design of a DNA-based reversible arithmetic and logic unit," *IET Nanobiotechnol.*, vol. 9, no. 4, pp. 226–238, 2015.

[4] B. Sen et al., "Modular design of testable reversible ALU by QCA multiplexer with increase in programmability," *Microelectron. J.*, vol. 45, no. 11, pp. 1522–1532, 2014.

[5] M. Morrison and N. Ranganathan, "Design of a reversible ALU based on novel programmable reversible logic gate structures," in Proc. *IEEE Comp. Soc. Ann. Symp. VLSI*, Chennai, India, 2011, pp. 126–131.

[6] Z. Guan et al., "An arithmetic logic unit design based on reversible logic gates," in *Proc. IEEE Pacific Rim Conf. Commun. Comput. Signal Process.*, Victoria, BC, 2011, pp. 925–931.

[7] M. Thomsen, R. Glck, and H. Axelsen, “Reversible arithmetic logic unit for quantum arithmetic,” *J. Phys, A*, vol. 43, no. 38, pp. 1–10, 2010.

[8] L. Gopal et al., “Design and synthesis of reversible arithmetic and logic unit,” in *Proc. IEEE Int. Conf. Comput. Commun. Contr. Technol.*, Langkawi, Malaysia, pp. 289–293.

[9] H. M. Gaur, A. K. Singh, and U. Ghanekar, “A review on online testability for reversible logic,” *Proc. Comput. Sci.*, NIT Kurukshetra, India, vol. 70, pp. 384–391, 2015.

[10] B. Parhami, “Fault-tolerant reversible circuits,” in *Proc. 40th Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, 2006, pp. 1726–1729.

[11] H. M. Gaur, A. K. Singh, and U. Ghanekar, “Testable design of reversible circuits using parity preserving gates, *IEEE Design & Test*, vol. 35, no. 4, pp. 56–64, 2018.

[12] H. M. Gaur and A. K. Singh, “Design of reversible logic circuits with high testability,” *IET Electron. Lett.*, vol. 52, no. 13, pp. 1102–1104, 2016.

[13] M. Mohammadi and M. Eshghi, “On figures of merit in reversible logic designs,” *J. Quant. Inf. Process.*, vol. 8, no. 4, pp. 297–318, 2009.

[14] D. Maslov, G. Dueck, and N. Scott, “Reversible logic synthesis benchmark.” Accessed: August 2017. [Online]. Available: http://www.cs.uvic.ca/~dmaslov

[15] *RevKit: An Open Source Toolkit for the Design of Reversible Circuits*. [Online]. Available: http://www.revlib.org

Hari Mohan Gaur is with the Department of Electronics and Communication Engineering, NIT Kurukshetra, Kurukshetra, India, and is currently working as an Associate Professor with the ABES Institute of Technology, Ghaziabad, India. His research interests include reversible logic, testing, and fault-tolerant digital design. Gaur has a PhD from the Department of Electronics and Communication Engineering, NIT Kurukshetra. He is the Student Member of the IEEE.

Ashutosh Kumar Singh is currently a Professor and Head of the Department of Computer Applications, National Institute of Technology (NIT) Kurukshetra, Kurukshetra, India, and a Chartered Engineer IET in the U.K. His research interests include verification, synthesis, design and testing of digital circuits, data science, cloud computing, machine learning, security, and big data. Singh has a Post-Doctoral from the Department of Computer Science, University of Bristol, Bristol, U.K. He is a Senior Member of the IEEE.

Umesh Ghanekar is currently a Professor with the Department of Electronics and Communication Engineering, National Institute of Technology (NIT) Kurukshetra, Kurukshetra, India. He has a PhD in computer engineering from NIT Kurukshetra. He is a Member of the IEEE.

■ Direct questions and comments about this article to Hari Mohan Gaur, Department of Electronics and Communication Engineering, NIT Kurukshetra, Kurukshetra, India; leoharimohan84@gmail.com.