Exploiting Reversible Computing for Latent-Fault-Free Error Detecting/Correcting CMOS Circuits

Mustafa Altun, Sajjad Parvin, and Husrev Cilasun

Abstract—Unlike conventional CMOS circuits, reversible circuits do not have latent faults, so faults occurring in internal circuit nodes always result in an error at the output. This is a unique feature for online or concurrent fault tolerance and the main motivation of this study with an aim of achieving highly efficient fault-tolerant CMOS logic circuits. For this purpose, we first implement fault-tolerant reversible circuits. We develop two techniques to make a reversible circuit fault-tolerant by using multiple-control Toffoli gates. The first technique is based on single parity preserving, and offers error detection for odd number of errors at the output. The second technique is constructed on Hamming codes that results in circuits detecting any number of errors unless the number of errors at the output is the order of d, or correcting $(d-1)/2$ bit errors where d is the minimum Hamming distance between any pair of bit patterns. We select $d=3$ in this study. We also claim that 100% error detection is possible with conservative reversible gates such as a Fredkin gate. For this purpose we develop a greedy synthesis algorithm that implements an arbitrary reversible function with multiple-control Fredkin gates. As the next step, we utilize the proposed reversible circuits with conventional CMOS gates. This certainly approves the practical use of the proposed techniques. The effectiveness of our techniques is demonstrated on benchmark circuits, implemented by both reversible and CMOS gates, in terms of fault tolerance performances and area costs. Comparisons with the related studies in the literature as well as with dual modular redundancy and triple modular redundancy based circuits clearly favor the proposed designs.

Index Terms—Fault-Tolerant CMOS; Reversible Logic Synthesis; Latent Fault; Error Detection and Correction.

1 Introduction

In the literature, research on reversible computing has been mainly motivated by its low power capability that even allows zero power dissipation in theory [1], [2], and its direct relation with quantum computing constructed on unitary matrix based reversible operations [3]. Our motivation is different. We exploit fault tolerance capability of reversible computing to detect faults concurrently. Reversible gates do not have a "don't care" condition, and correspondingly any switching fault in a circuit node causing $0\rightarrow1$ or $1\rightarrow0$ transition should change the output logic values. Therefore, reversible circuits do not have latent switching faults, defined as faults not causing an error at the output for the current operation, but might be destructive for next operations. This inference is based on two properties: 1) a reversible circuit should satisfy one-to-one matching between its input and output assignments, and 2) a subcircuit of a reversible circuit is also reversible.

Different from reversible circuits, conventional CMOS logic circuits do have "don't care" conditions that results in latent faults. Consider a NAND gate having two inputs and an output. Suppose that a switching fault occurs in one of its inputs. Considering all of the four input assignments, we see that only in 50% of the cases we see a change at the output. It gets even worse for a three-input NAND gate; here, only 25% of the switching faults cause a change at the output. These low detection rates caused by latent faults are problematic, especially in online or concurrent fault tolerance for IoT and real-time applications as well as for reliability-critical aerospace and military applications; any problem should be immediately solved without necessarily waiting for an error occurrence at the output. Also such latent faults can ruin the used fault tolerance scheme [4]. For example, consider a system using dual modular redundancy (DMR) or triple modular redundancy (TMR). A permanent latent fault in one of the replicas would disrupt the detecting/correcting mechanism of DMR/TMR. To deal with this problem, N-modular redundancy (NMR) and similar techniques can be used [5], [6], [7]. However, these techniques do not fully solve the problem due to the existence of latent faults. Additionally, area cost increases significantly. We show that our reversible circuit based solutions with CMOS implementations are more efficient both in terms of area and fault tolerance performances.

Further investigating the literature, we see many works focusing on concurrent fault detection by the means of using various coding schemes such as Berger codes [8], weight-based codes [9], and Bose-Lin codes [10]. Even almost perfect fault detection (99.5% fault coverage) is achieved in [9]. However, this fault coverage is just for those observable at the output, so latent faults are neglected. A similar treatment is used in [11]. Also, there are some works partially detecting and masking faults for the most susceptible nodes in the logic network [12], [13]. Although these approaches

- *This work is supported by the TUBITAK-1002 project #215E268.*
- *Mustafa Altun and Sajjad Parvin are with the Department of Electronics and Communication Engineering, Istanbul Technical University, Istanbul, Turkey, 34469; Husrev Cilasun is with Aselsan A.S., Ankara, Turkey, 06370.*
- *E-mails: altunmus@itu.edu.tr, sajadparvin.stu93@gmail.com, and mhcilasun@aselsan.com.tr.*

are area-efficient, they offer poor fault coverage rates.

In contrary to the mentioned works, our approach do consider latent faults by utilizing the reversible bijective feature which allows faults occurring in any intermediate node to be reflected at the output. We first aim at achieving fault-tolerant reversible circuit implementations, and then replacing reversible gates with their proposed CMOS counterparts. Note that since CMOS gates are not reversible, our final CMOS circuits are not reversible as well.

We develop two techniques to make a reversible circuit fault-tolerant using multiple-control Toffoli gates. The first technique is for error detection and based on single parity preserving. The idea is preserving the input parity at the output, so any odd number of errors at the output can be detected by comparing input and output parities. This approach can be implemented in two possible ways. In the first way, the desired circuit can be synthesized by solely using custom parity-preserving building blocks that guarantees global parity preservation. For this purpose, different gates, such as Khan and Islam gates, are proposed [14], [15], [16], [17]. Although these gates work properly in theory by assuming that they are internally fault-free, this is not the case in practice. Indeed, these gates are generally too complex and large to be hardly assumed as simple fault-free gates. Therefore, any fault internally occurred in gate nodes rather than interconnections between gates can violate their fault-tolerant property. The second way of implementing parity-preserving circuits is to add an extra input and an output [18]. Although this approach offers better fault tolerance compared to the first one, its implementation with reversible gates is not given in the referred study and might cause extremely large area overheads. In this regard, using the same way, we introduce synthesis method that results in a fault-tolerant reversible circuit with doubled area.

Our second technique can be used either for error detection or for error correction. We exploit Hamming codes to achieve detection of any numbers of error at the output unless the number of errors is the order of d, or correction of $(d-1)/2$ bit errors where d is the minimum Hamming distance between any pair of bit patterns. We select $d = 3$ in this study. Indeed, the idea of using Hamming codes in fault tolerance of reversible circuits is previously introduced in [19], [20]. However, these studies focus on a constrained set of circuits for encoding and decoding purposes rather than presenting a generic method for converting any reversible circuit to a fault-tolerant one. In this paper, we satisfy this.

We also claim that 100% error detection is possible with conservative reversible gates such as a Fredkin gate. For this purpose we develop a greedy synthesis algorithm that implements an arbitrary reversible function with multiple-control Fredkin gates. Our algorithm first converts the truth table of a given function into a conservative form by adding 0 and 1 valued inputs and their corresponding outputs. Then row by row synthesis is performed with Fredkin gates. To our knowledge, there is no algorithm in the literature to synthesize any given reversible function with Fredkin gates. However, we realize that if we apply our initial conversion technique to a given function as a pre-processing step, and then the algorithm given in [21] could perform a synthesis with only Fredkin gates. Nevertheless, the resulted area results are generally much worse than those of ours.

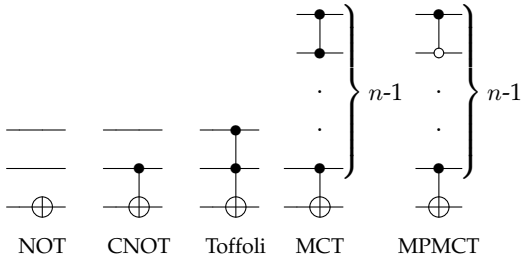


Fig. 1. Circuit representations of NOT, CNOT, Toffoli, MCT, and MPMCT.

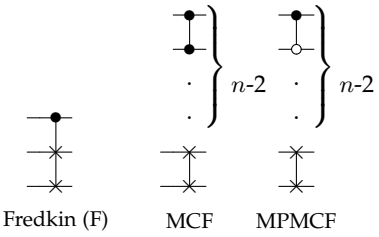


Fig. 2. Circuit representations of Fredkin (F), MCF, and MPMCF.

Apart from all of the mentioned studies, we utilize the proposed reversible circuits with conventional CMOS gates including NOT, NAND, and XOR gates to show the circuits' potential for practical use. The effectiveness of our techniques is demonstrated on benchmark circuits, implemented by both reversible and CMOS gates, in terms of fault tolerance performances and area costs. Comparisons with the related studies in the literature as well as with DMR and TMR based circuits clearly favor the proposed designs.

The rest of paper is organized as follow. In Section 2, we discuss basics of reversible logic and reversible cost measures used in this paper. In Section 3, we develop two techniques to make a reversible circuit fault-tolerant by using single parity preserving and Hamming code based approaches. Section 4 represents our synthesis technique for 100% error detection using Fredkin gates. In Section 5, we show how to utilize the proposed reversible circuits with CMOS logic gates. In Section 6, we give experimental results to evaluate the proposed circuits. Finally, Section 7 concludes this work with future directions.

2 PRELIMINARIES

While a conventional Boolean function always carries a one bit information (0 or 1) that is independent of the number of input bits, a **reversible Boolean function** carries information with using the same number of input and output bits. For reversible functions, each input bit combination results in a unique output bit combination; the reverse of this is also true because of the reversibility. This means that the input values can be deduced by looking at the output values of the reversible function. Bijection function in mathematics is also a great example to understand reversibility. In these functions, input and output sets have the same number of elements and each element has only one counterpart in the other set.

TABLE 1
Quantum Cost of Reversible Gates

Bit Size	Gate Name	Quantum Cost
1	NOT	1
2	CNOT	1
3	Toffoli	5
n	MCT	$2^n - 3$
n (at least one negative and one positive control)	MPMCT	$2^n - 3$
3 (all negative controls)	MPMCT	6
n (all negative controls)	MPMCT	$2^n - 1$
3	Fredkin	7
n	MPMCF	Cost of MPMCT + 2

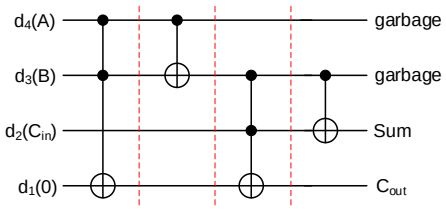


Fig. 3. Optimized 1-bit full adder.

2.1 Basics of Reversible Circuits

A reversible function can be realized by a **reversible circuit** consisting of reversible gates. In this study we use three types of gates: MCT (**M**ultiple **C**ontrol **T**offoli), MPMCT (**M**ixed **P**olarity **M**ultiple **C**ontrol **T**offoli), and MPMCF (**M**ixed **P**olarity **M**ultiple **C**ontrol **F**redkin). Definition of gates are as follows, with corresponding symbols given in Figure 1 and Figure 2 where symbols $\bullet$, $\circ$, $\oplus$, and $\times$ denote positive control, negative control, Toffoli target lines, and Fredkin target lines, respectively.

- **N**OT: a 1-bit gate performing NOT operation.
- **C**NOT: a 2-bit gate performing 1 bit NOT operation on its target bit iff its control bit is 1.
- **T**offoli: a 3-bit gate performing 1 bit NOT operation on its target bit iff its control bits are both 1.
- **M**ultiple **C**ontrol **T**offoli: an n-bit gate, $n = 1, 2, 3, 4, ...$, performing 1 bit NOT operation on its target bit iff all of its control bits are 1.
- **M**ixed **P**olarity **M**ultiple **C**ontrol **T**offoli: an n-bit gate, $n = 1, 2, 3, 4, ...$, performing 1 bit NOT operation on its target bit iff all of its positive control bits are 1 and all of its negative control bits are 0.
- **F**redkin: a 3-bit gate performing swap operation on its target bits iff its control bit is 1.
- **M**ultiple **C**ontrol **F**redkin: an n-bit gate, $n = 1, 2, 3, 4, ...$, performing swap operation on its target bits iff all of its control bits are 1.
- **M**ixed **P**olarity **M**ultiple **C**ontrol **F**redkin: an n-bit gate, $n = 1, 2, 3, 4, ...$, performing swap operation on its target bits iff all of its positive control bits are 1 and all of its negative control bits are 0.

2.2 Area Costs of Reversible Circuits

For **quantum cost**, we use a measure given in [22], [23] because it is the most commonly used and accepted one compared to other measures in the literature [24], [25]. Table 1 summarizes the quantum costs used in this study. One can also consider **reversible cost** or just simply gate count [26]. But this metric does not consider the complexity of a gate including the bit sizes.

2.3 Fault Tolerance in Reversible Circuits

The following lemma demonstrates why reversible circuits do not have **latent switching faults**. Such faults do not immediately cause an error at the output for the current operation, but they might be destructive for next operations [27].

Lemma 1. *A switching fault (0→1 or 1→0 transition) in a node of a reversible circuit always results in a change/transition at the output value.*

Proof. The proof is by contradiction. Suppose that a transition in a node does not cause any change at the output. Since subcircuit of a reversible circuit is also reversible, the node can be considered as an input node of a reversible circuit. Also we know that a reversible circuit has one-to-one matching between its inputs and outputs, so a change in an input should change the output. As a result, there is a contradiction. □

Consider a reversible circuit with inputs I_1, I_2,..., I_n and outputs O_1, O_2,..., O_n. The circuit is **parity preservative** iff $I_1 \oplus I_2 \oplus ... \oplus I_n$=$O_1 \oplus O_2 \oplus ... \oplus O_n$ where $\oplus$ represents an XOR logic operation. Also if a reversible circuit consists of parity preservative gates such as **Fredkin**, Feynman, and Peres gates then the circuit is parity preservative.

The following lemmas explain why we use a preservative gate based synthesis technique for 100% fault detection.

Lemma 2. *Consider a reversible circuit consisting of only preservative gates. For this circuit, 100% fault detection is possible.*

Proof. Since a transition does not scatter to multi transitions at the output. With XORing the outputs, one can always detect the fault. □

We select the Fredkin gate since it does not just preserve the XOR of inputs, but it also preserves the arithmetic summation of the input values. Therefore, along with XORing the outputs, one can also detect faults by counting the 1 or 0 valued outputs. Another reason of selecting the Fredkin is its synthesis friendly simple structure.

3 Making A Reversible Circuit Fault-Tolerant

In this section, we discuss our methods to make a given reversible circuit fault-tolerant in terms of error detection and correction. In order to elaborate them, an optimized 1-bit full adder synthesized in [20] is used as an example of a given circuit. The circuit is shown in Figure 3.

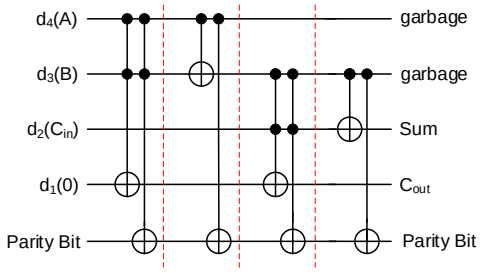


Fig. 4. A single parity fault-tolerant 1-bit full adder.

3.1 Single Parity based Error Detection

Single parity is basically based on the parity preservative property. In order to satisfy the property for circuits consisting of MCT or MPMCT gates which are not parity preservative gates, we add an extra bit line to a circuit and an extra gate for each gate of the circuit. The added gate shares the same control lines with those of the corresponding gate in the original circuit, with its target always in the added line. In this manner, we can satisfy parity preservative equation by doubling the circuit area cost. Thus, we could detect odd number of errors at the output. We elucidate our method with the following example.

Example 1. *Let's make the full adder in Figure 3 fault-tolerant. Firstly, we add an extra bit line "parity bit". Then for each of the four gates, we add an extra MCT gate. The resulted circuit is shown in Figure 4.*

Note that our method guarantees parity preserving property not only for the given circuit, but also for any subcircuit of it that can be used for determining the fault places. This cannot be done with the conventional DMR technique. Additionally, although area overheads are same in DMR and our technique, the resulted DMR circuit is not reversible, so there is no guarantee of keeping the fault information at the output.

3.2 Hamming 3 based Error Detection and Correction

Basic idea of Hamming 3 encoding is based on the following equations [28], [29].

$$p_1 \oplus d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 \oplus ... = 0 \quad (1)$$

$$p_2 \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus ... = 0 \quad (2)$$

$$p_4 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus ... = 0 \quad (3)$$

$$p_8 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus ... = 0 \quad (4)$$

$$p_{16} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus ... = 0 \quad (5)$$

Constructed on these equations, we present our two-step algorithm to make a reversible circuit fault-tolerant.

Input: A reversible circuit consisting of MPMCT gates having n bit/data lines $d_1, d_2, ..., d_n$.

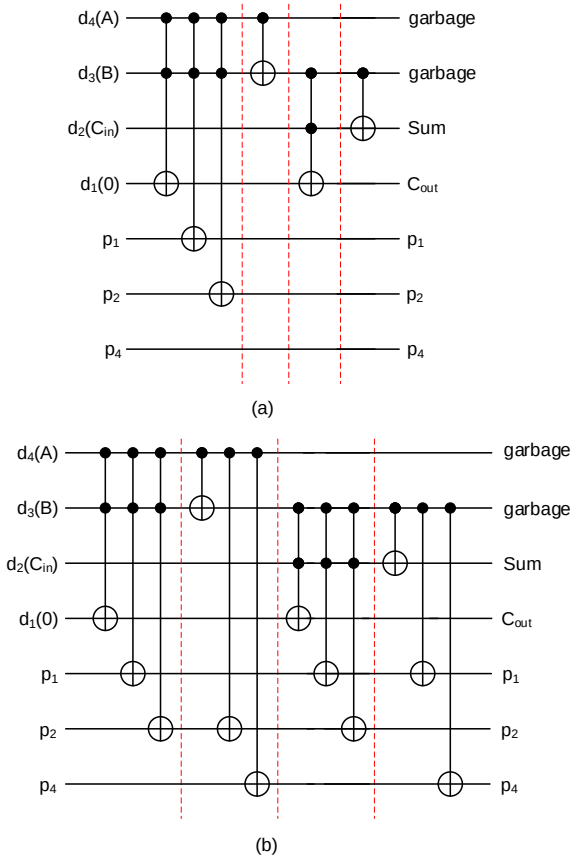


Fig. 5. Illustration of the algorithm for the given circuit in Figure 3: (a) first gate is considered, and (b) the final form.

Output: A fault-tolerant circuit that can detect any number of errors unless the number of errors at the output is the order of 3, or correct 1 bit error at the output.

1) Finalize Equations 1-5 by considering n. Thus, the needed parity lines are determined.
2) In order to satisfy the finalized equations, for each gate add extra gates having the same controls as those of the the corresponding gate and the targets on the parity lines.

We elucidate our method with the following example.

Example 2. *Again consider the full adder in Figure 3 as a given circuit. By using the first step we obtain the finalized equations:*

$$p_1 \oplus d_1 \oplus d_2 \oplus d_4 = 0; \quad (6)$$

$$p_2 \oplus d_1 \oplus d_3 \oplus d_4 = 0; \quad (7)$$

$$p_4 \oplus d_2 \oplus d_3 \oplus d_4 = 0. \quad (8)$$

In the second step, we start with the first gate on the left side. Since it has a target bit on d_1, by using Equations 6 and 7 we should add two targets on p_1 and p_2. Therefore, two extra gates are needed. This is illustrated in Figure 5 (a). After applying the procedure for the second, third, and the fourth gates, we obtain the final form as shown in Figure 5 (b).

The area overhead of our method is more or less the same with that of TMR. However, our method offers higher error correction rates. Also while our method can correct or detect errors, TMR is only for correction. Error detection performance of our method is much better than that of DMR. A final note is that similar to our single parity based method, the proposed error detection/correction scheme is valid for any subcircuit of the given circuit.

3.3 Simplified Single Parity and Hamming 3

For our methods introduced in the last two subsections, we add extra gates with their controls on parity bits. Here, we show that we can reduce the area overheads of the proposed techniques by investigating added gate pairs in adjacent stages, separated by dashed red lines in figures. We have two cases for simplification: 1) gates and their locations are identical such that both gates have the same target and control bit lines; and 2) one of the gates shares all control and target lines of the other one, plus having one extra control. For the first case, we remove both gates since switching a parity bit twice results in no change. For the second case, we remove the gate having one less control lines, and keep the other gate with negating its extra control. The reason is that a change in a parity occurs only if all of the shared controls are active and the extra control is inactive.

Figure 6 shows an example of simplification applied to the circuit in Figure 5. For the first two stages, there are two gates satisfying the second case. This is illustrated by Figure 6 (a). Also in the third and the fourth stages, there is a similar case. As a result, the simplified circuit is obtained as shown in Figure 6 (b).

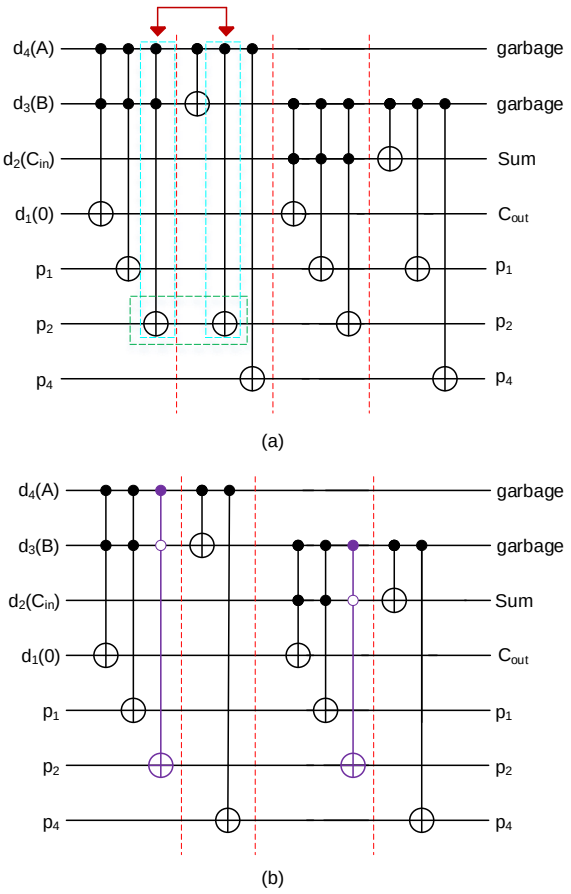


Fig. 6. Simplification: (a) locating proposer gate pairs of adjacent first and second stages; and (b) the final form of the simplified circuit.

4 Perfect Error Detection with Fredkin Gates

From Lemma 2 given in the preliminaries section, we know that 100% error detection is possible with Fredkin gates. For this purpose we develop a greedy synthesis algorithm that implements an arbitrary reversible function with MPMCF gates. Our algorithm has four steps as follows.

Input: A reversible function with its truth table having n inputs and n outputs.
Output: A reversible circuit consisting of MPMCF gates that implements the given function.

1) Make the given truth table conservative by adding 0 and 1 valued inputs and their corresponding outputs.
   - For each row of the truth table, find the difference value as the number of 1's in each output row minus the number of 1's in the corresponding input row.
   - The number of added 1 valued inputs is the highest positive value of the difference, and the number of added 0 valued inputs is the absolute of the lowest negative value of the difference.
   - Based on the added input values, the output values must be set in a way to achieve same number of 1's in each input/output row of the truth table.
2) Sort input and output columns of the table by considering the the number 1's in descending order.
3) Determine the unmatched bits between inputs and outputs for each row of the table.
4) Start from the row having the smallest unmatched bits, assign MPMCF gates row by row.
   - Select controls of MPMCF gates such that the gate only changes the bits in the corresponding row, without disturbing other rows. In the worst-case scenario, this is satisfied by using all bits controls except the two target bits.
   - The number of used MPMCF gates in a row is the number of unmatched bits over two.

As an example, we again use the reversible full-adder circuit in Figure 3 and its truth table; $n = 4$. Steps of the algorithm is summarized in Figure 7. First we determine the difference values, shown in Figure 3 (a). Since the highest positive value is 2, we add two 1 valued inputs I_{n1} and I_{n2} as well as the corresponding outputs O_{n1} and O_{n2}. This is illustrated in Figure 7 (b). Note that since there is no negative difference value, we do not need to add 0 valued inputs. After performing sorting, we map MPMCF gates as shown in Figure 7 (c) and (d).

5 CMOS Logic Implementations

Our algorithms given in Section 3 result in reversible circuits with MPMCT and MPMCF gates. We show how to convert these gates into CMOS gate based realizations.

Input				Output				Difference Value
A	B	C_{in}	d_1	A_{out}	B_{out}	S	C_{out}	
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	1	1	0	1
0	1	1	0	0	1	0	1	0
1	0	0	0	1	1	1	0	2
1	0	1	0	1	1	0	1	1
1	1	0	0	1	0	0	1	0
1	1	1	0	1	0	1	1	0

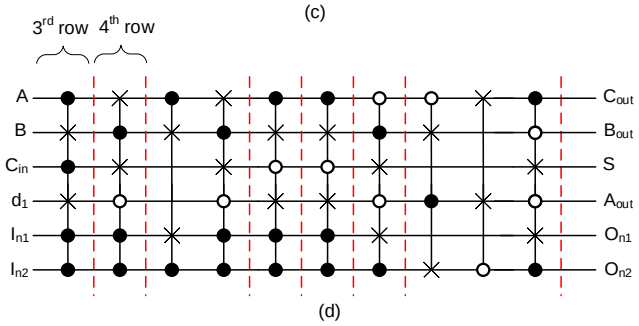
(a)

Input						Output					
A	B	C_{in}	d_1	I_{n1}	I_{n2}	A_{out}	B_{out}	S	C_{out}	O_{n1}	O_{n2}
0	0	0	0	1	1	0	0	0	0	1	1
0	0	1	0	1	1	0	0	1	0	1	1
0	1	0	0	1	1	0	1	1	0	0	1
0	1	1	0	1	1	0	1	0	1	1	1
1	0	0	0	1	1	1	1	1	0	0	0
1	0	1	0	1	1	1	1	0	1	0	1
1	1	0	0	1	1	1	0	0	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1

(b)

Input						Output						Fredkin Needed
I_{n1}	I_{n2}	C_{in}	B	A	d_1	O_{n1}	O_{n2}	S	B_{out}	C_{out}	A_{out}	
1	1	0	0	0	0	1	1	0	0	0	0	0
1	1	1	0	0	0	1	1	1	0	0	0	0
1	1	1	1	1	0	1	1	1	0	1	1	1
1	1	1	1	0	0	1	1	0	1	1	0	1
1	1	1	0	1	0	1	0	0	1	1	1	2
1	1	0	1	1	0	1	1	0	0	1	1	1
1	1	0	1	0	0	1	0	1	1	0	0	1
1	1	0	0	1	0	0	0	1	1	0	1	3

(c)



(d)

Fig. 7. Steps of the algorithm (a)-(b) first step; (c) second and third steps; and (d) fourth step.

Consider a conventional NAND gate. Since three input combinations are mapped to a single logical 1 value, the information regarding to a possible fault at one of the inputs can be lost. The same problem occurs in NOR, OR, and AND gates. This is indeed related to "don't care" conditions. On the other hand, NOT, XOR, and XNOR gates perfectly satisfy the awareness of an input fault. However, they do not form a universal set. We use NAND, XOR, and NOT gates for realizations such that any internal node of the resulted CMOS logic circuit should be an input of an XOR or a NOT gate. Also if an inverter is driving a NAND gate then we replace the inverter with a cascaded inverter pair in a loop to prevent "don't care" conditions. As a result, we guarantee of eliminating any latent switching faults at the nodes.

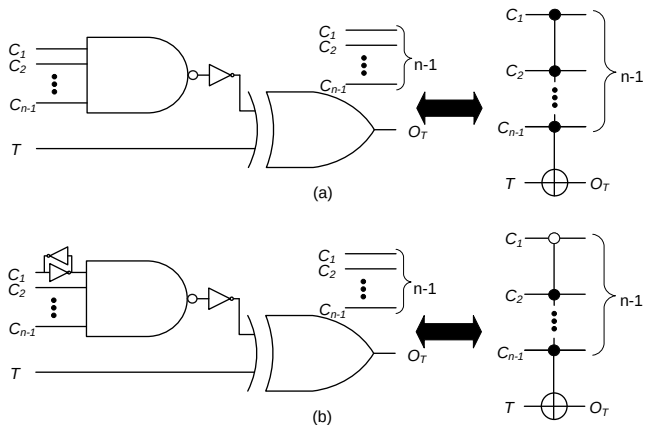


Fig. 8. CMOS gate implementations for (a) a MCT gate, and (b) a MPMCT gate.

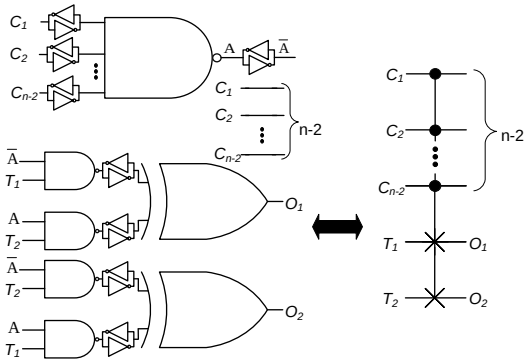


Fig. 9. CMOS gate implementation of a MPMCF gate.

Gate implementations of an MCT gate is shown in Figure 8 (a). For an MPMCT, we only add cascaded inverter pairs to the inputs having negative controls. This is shown in Figure 8 (b). Figure 9 shows the implementation for a MPMCF gate. Again in case of having negative controls, we add cascaded inverter pairs to the corresponding inputs.

Note that since CMOS logic is one directional, these implementations are not fully reversible anymore. They implement the reversible functions proceeding only from inputs to outputs.

6 Experimental Results

We use reversible benchmarks from [30]. We evaluate our methods in terms of area cost, power cost and fault tolerance performance. We consider three measures for area costs: 1) reversible cost, 2) quantum cost, and 3) CMOS cost. As previously explained in the preliminaries section, for reversible cost we basically use reversible gate counts, and for quantum cost we use a measure in Table 1. For CMOS cost, we report an estimation of occupied die area using TSMC 0.18 μ m technology. Beside that CMOS estimated power values are also reported. CMOS area and power results are obtained using the Genus tool in Cadence.

For fault tolerance analysis, in each try we inject a randomly placed switching fault into a circuit node that causes

TABLE 2
Area Comparison Between TBS Technique And Our Proposed Synthesis Technique.

Approach	Proposed Synthesis Technique Reversible-Quantum Cost	[31] TBS Technique Reversible-Quantum Cost
Single Parity	8-20	18-110
Hamming 3	12-32	124-5025

a bit flip, and check the resulted output errors. Detection and correction rates represent the ratio of the number of tries where errors are detected/corrected at the output to the total number of tries, using a Monte Carlo method. For the single parity scheme, errors are detected iff they occur in odd numbers at the output. And for the Hamming coded scheme, all of the errors are detected/corrected except those occurring in numbers multiplicand of d. In our study d is 3.

6.1 Fault Tolerance with Reversible Gates

For our single parity and Hamming 3 based methods, we directly use the synthesized benchmarks from [30]. Then we make them fault-tolerant. In the literature, to our knowledge, there is no similar study. As we discuss in the introduction section, although there are different fault-tolerant approaches proposed for reversible circuits, they lack implementations with reversible gates. If we implement them with the known reversible synthesis techniques suitable for don't care inputs (error detection/correction necessarily requires don't care conditions), then the area costs become excessively large. Table 2 shows an example for a reversible 1-bit full adder synthesized with our techniques and with the transformation based synthesis (TBS) technique [31]. Since area costs of TBS are much larger (even worse for larger benchmarks), we do not add further results of TBS in the following tables.

Area costs and error detection/correction rates of the proposed methods are shown in Table 3. By examining Table 3, we can conclude that for the Hamming 3, the simplification almost always reduces the area costs with a slight decrease in error detection/correction. On the other hand, for the single parity, the simplification causes a major decrease in error detection, so it might not be preferable. That is due to losing parity preservative feature of the simplified stages. Another inference is that in average our single parity and Hamming 3 based techniques make the original circuit area two and three times larger, respectively. This is similar to DMR and TMR area costs. One important point is that detection rates of the single parity method is as good as those for the Hamming 3 based method.

6.2 Fault Tolerance with CMOS Gates

To show practical usage of the proposed techniques, we perform CMOS implementations with NOT, NAND, and XOR-2 gates as explained in Section 5. Here, we extensively apply our techniques in comparison with DMR and TMR solutions to reversible benchmark functions by reporting area and power results.

Results are shown in Table 4. Since the single parity can only detect faults, we compare it with the DMR scheme. In most cases, the single parity has much higher error detection rates with similar or better area and power consumption in comparison to the DMR scheme. Since the Hamming 3 technique has a correction capability, we compare it with the TMR scheme. Again in most cases, the Hamming 3 proposes a better performance in both area cost and the error correction rates. However, power consumption of the Hamming 3 is generally more than that of the TMR. On average, the single parity consumes 8.2% less power and 59.4% less area in comparison to the DMR. Also, the Hamming 3 consumes 32.5% more power and 56.6% less area compared to the TMR.

6.3 Fault Tolerance with Fredkin Gates

Since Fredkin gate is a conservative gate and if a circuit is synthesized using only this gate it will yield 100% error detection. In the literature, synthesis with Fredkin gates has not been proposed. However, by making any truth table conservative and then performing Fredkin Enabled TBS scheme using the Mathias's approach [21], we can have a circuit constructed on just Fredkin gates. The results shown in Table 5, clearly favor our synthesis technique for each of the three area cost measures as well as for power consumption.

7 Conclusion

In this study, we have proposed methods to achieve latent-fault-free and error detecting/correcting CMOS circuits. For this purpose, we first implement fault-tolerant reversible circuits. Since our methods to make a reversible circuit fault-tolerant would not disturb the original circuit, it yields smaller area overhead in comparison to any other synthesis technique in the literature. Next, we convert our reversible circuits to CMOS realizations, and then compare our methods with conventional DMR and TMR techniques. On the quest to achieve perfect error detection, we also develop a greedy synthesis algorithm that implements an arbitrary reversible function with multiple-control Fredkin gates. As a future work, we aim to find a better, with much smaller CMOS area, Fredkin or other conservative gate based synthesis technique to achieve 100% error detection and correction.

References

[1] C. H. Bennett, "Logical reversibility of computation," *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[2] A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz, "Experimental verification of landauer/'s principle linking information and thermodynamics," *Nature*, vol. 483, no. 7388, pp. 187–189, 2012.

[3] M. M. Wilde, *Quantum information theory*. Cambridge University Press, 2013.

[4] M. Gabel, A. Schuster, R. Bachrach, and N. Bjørner, "Latent fault detection in large scale services," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, June 2012, pp. 1–12.

[5] E. Dubrova, *Fault-Tolerant Design*. Springer Verlag New York, 2013.

[6] T. Nakura, K. Nose, and M. Mizuno, "Fine-grain redundant logic using defect-prediction flip-flops," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, Feb 2007, pp. 402–611.

TABLE 3
Reversible-Quantum Costs and Error Detection and Correction Rates of the Proposed Hamming 3 and Single Parity based Methods

Benchmark Name	Proposed Hamming 3 Reversible-Quantum Cost	Proposed Simplified Hamming 3 Reversible-Quantum Cost	Proposed Single Parity Reversible-Quantum Cost	Proposed Simplified Single Parity Reversible-Quantum Cost	Proposed Hamming 3 Detection Rate	Proposed Simplified Hamming 3 Detection Rate	Proposed Hamming 3 Correction Rate	Proposed Simplified Hamming 3 Correction Rate	Proposed Single Parity Detection Rate	Proposed Simplified Single Parity Detection Rate
3_17	18-42	15-41	12-28	10-26	85.71%	87.17%	53.20%	53.38%	83.41%	54.51%
4_49	55-207	48-204	32-120	29-112	79.54%	79.47%	43.41%	41.08%	82.69%	53.24%
4b15g	48-156	45-153	30-94	27-91	80.36%	79.30%	50.89%	48.50%	83.50%	43.57%
adr4	187-2691	143-2665	110-1454	89-1488	94.76%	93.42%	65.64%	40.52%	94.53%	47.65%
aj-e11	43-143	39-143	26-90	24-84	84.08%	84.83%	52.43%	52.21%	83.02%	75.89%
alu-gupta	39-143	31-143	26-202	22-148	92.93%	89.20%	65.74%	62.70%	89.11%	62.46%
c17	27-297	23-297	18-198	16-170	95.41%	94.06%	72.45%	72.86%	89.43%	62.98%
con1	67-664	54-586	42-412	39-376	97.24%	96.67%	78.30%	76.13%	93.89%	43.24%
f2	62-836	49-577	38-510	32-339	97.36%	96.12%	73.95%	69.15%	95.40%	22.02%
Hamm.code (15)	245-1580	229-1520	140-906	132-833	77.95%	75.05%	35.70%	30.31%	81.64%	50.24%
Hidd.wei.bit	1084-8795	928-8364	662-4530	598-4466	76.93%	68.09%	32.79%	22.28%	81.18%	50.44%
indiv-pla-rec5	336-6062	250-5181	218-3146	177-2657	75.64%	72.66%	31.17%	23.70%	90.34%	45.01%
majority	24-363	24-360	16-214	14-212	96.44%	96.44%	86.62%	86.62%	94.57%	94.00%
misex1	180-3818	135-2859	110-1964	90-1772	93.15%	93.58%	73.76%	76.84%	95.62%	73.92%
mod5mils_18	17-41	15-39	10-26	9-25	90.08%	92.00%	64.99%	66.82%	85.07%	17.17%
ntd	187-2691	146-2665	110-1454	89-1328	94.74%	93.75%	65.48%	56.24%	94.46%	44.02%
rd73	255-3990	203-3716	160-2046	134-1944	96.88%	69.26%	67.66%	41.60%	93.80%	56.49%
sqrt8	148-3620	115-3462	80-1168	72-1111	96.42%	96.81%	73.82%	75.74%	93.19%	63.92%
sym9	840-17472	672-16497	420-8736	350-7579	97.31%	94.59%	72.77%	61.58%	94.37%	50.55%
xor5	21-21	21-21	14-14	14-14	88.04%	88.04%	54.60%	54.60%	82.14%	82.14%
1-2-3cntr	37-241	36-240	22-142	21-139	91.17%	91.28%	59.63%	59.03%	86.64%	53.06%
wim	84-713	59-511	50-434	37-343	94.19%	89.37%	75.84%	69.85%	95.18%	38.59%
z4	170-2868	139-2669	96-1284	84-	96.46%	89.79%	64.48%	49.42%	92.67%	38.03%

TABLE 4
CMOS Area-Power Costs and Error Detection and Correction Rates of the Proposed Hamming 3 and Single Parity Methods as well as for DMR and TMR

Benchmark Name	DMR Area-Power Cost μm^2 - μW	TMR Area-Power Cost μm^2 - μW	Proposed Hamming Area-Power Cost μm^2 - μW	Proposed Single Parity Area-Power Cost μm^2 - μW	DMR/TMR Detection/ Correction Rate	Proposed Hamming 3 Detection Rate	Proposed Hamming 3 Correction Rate	Proposed Single Parity Detection Rate
3_17	536-328	804-492	660-445	464-224	69.51%	85.71%	53.20%	83.41%
4_49	3568-726	5352-1089	2176-1578	1088-576	45.08%	79.54%	43.41%	82.69%
4b15g	4352-666	6528-999	1784-1274	1231-592	42.68%	80.36%	50.89%	83.50%
adr4	10808-2568	16212-3852	3585-1907	4619-1752	39.85%	94.76%	65.64%	94.53%
aj-e11	3318-524	4977-786	1694-1238	1124-426	41.19%	84.08%	52.43%	83.02%
alu-gupta	2212-500	3318-750	1801-1232	1356-391	47.51%	92.93%	65.74%	89.11%
c17	3104-382	4656-573	1249-549	820-320	43.41%	95.41%	72.45%	89.43%
con1	4852-946	7278-1419	2551-1170	1338-451	25.23%	97.24%	78.30%	93.89%
f2	4744-832	7116-1248	2658-1187	1605-446	31.29%	97.36%	73.95%	95.40%
Hamm.code (15)	19476-5282	29214-7923	10220-12286	6474-6920	50.06%	77.95%	35.70%	81.64%
Hidd.wei.bit	50904-10128	76356-15192	45196-49567	29928-27666	21.72%	76.93%	32.79%	81.18%
indiv-pla-rec5	207324-33614	310986-50421	18674-15128	13395-9471	13.12%	75.64%	31.17%	90.34%
majority	2176-456	3264-684	999-381	571-117	28.35%	96.44%	86.62%	94.57%
misex1	14910-3392	22365-5088	8650-3418	4798-1441	30.64%	93.15%	73.76%	95.62%
mod5mils_18	642-388	963-582	606-438	339-232	48.22%	90.08%	64.99%	85.07%
ntd	10808-2568	16212-3852	8133-4935	4619-1752	25.44%	94.74%	65.48%	94.46%
rd73	14446-2896	21669-4344	10844-7430	5957-2600	35.94%	96.88%	67.66%	93.80%
sqrt8	13662-3186	20493-4779	6153-3250	3068-1087	22.67%	96.42%	73.82%	93.19%
sym9	23686-3578	35529-5367	43448-30085	28466-16184	13.64%	97.31%	72.77%	94.37%
xor5	1248-548	1872-822	642-587	321-257	64.78%	88.04%	54.60%	82.14%
1-2-3cntr	4816-1048	7224-1572	1694-1271	1159-788	49.17%	91.17%	59.63%	86.64%
wim9	6742-1052	10113-1578	3157-1308	1891-561	40.30%	94.19%	75.84%	95.18%
z4	7598-1932	11397-2898	7438-4857	3763-1342	43.12%	96.46%	64.48%	92.67%

[7] W. G. Brown, J. Tierney, and R. Wasserman, "Improvement of electronic-computer reliability through the use of redundancy," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 407–416, Sept 1961.

[8] J. . Lo, S. Thanawastien, and T. R. N. Rao, "Concurrent error detection in arithmetic and logical operations using berger codes," in *Proceedings of 9th Symposium on Computer Arithmetic*, Sept 1989, pp. 233–240.

[9] D. Das and N. A. Touba, "Weight-based codes and their application to concurrent error detection of multilevel circuits," in

TABLE 5
Area and Power Costs of Fredkin Synthesis Techniques with 100 % Error Detection

Benchmark Name	Proposed Fredkin Synthesized Reversible Cost	Mathias Fredkin Synthesized Reversible Cost	Proposed Fredkin Synthesized Quantum Cost	Mathias Fredkin Synthesized Quantum Cost	Proposed Fredkin Synthesized CMOS Area Cost μm^2	Mathias Fredkin Synthesized CMOS Area Cost μm^2	Proposed Fredkin Synthesized CMOS Power Cost μW	Mathias Fredkin Synthesized CMOS Power Cost μW
3_17	12	27	483	1149	6325	14196	1410	3453
4_49	33	182	2303	24964	19249	103109	3917	22922
4b15g	21	45	665	1015	11069	22019	2489	5895
1-2-3-cntr	39	268	2850	20316	22358	145307	4848	34497
aj-e11	22	57	976	2205	11733	29760	2387	7191
alu by gupta	56	57	2712	2205	30897	45267	6603	10589
c17	139	702	15772	136553	89435	41836	18798	88829
mod5mils_18	61	75	2934	3529	33628	39156	7206	9265
Hidden weighted bit	105	144	4344	3622	55845	68789	13007	17515
mini alu	8	21	295	655	4276	13598	959	2505
4gt5-v1	37	52	2176	3194	20977	33772	436	6079

Proceedings 17th IEEE VLSI Test Symposium (Cat. No.PR00146), April 1999, pp. 370–376.

[10] ——, "Synthesis of circuits with low-cost concurrent error detection based on bose-lin codes," in *Proceedings. 16th IEEE VLSI Test Symposium (Cat. No.98TB100231)*, April 1998, pp. 309–315.

[11] K. Nepal, N. Alves, J. Dworak, and R. I. Bahar, "Using implications for online error detection," in *2008 IEEE International Test Conference*, Oct 2008, pp. 1–10.

[12] K. Mohanram and N. A. Touba, "Partial error masking to reduce soft error failure rate in logic circuits," in *Proceedings 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, Nov 2003, pp. 433–440.

[13] ——, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *International Test Conference, 2003. Proceedings. ITC 2003.*, vol. 1, Sept 2003, pp. 893–901.

[14] L. Jamal, M. M. Rahman, and H. M. H. Babu, "An optimal design of a fault tolerant reversible multiplier," in *2013 IEEE International SOC Conference*, Sept 2013, pp. 37–42.

[15] ——, "An optimal design of a fault tolerant reversible multiplier," in *2013 IEEE International SOC Conference*, Sept 2013, pp. 37–42.

[16] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz, "Fault tolerant reversible logic synthesis: Carry look-ahead and carry-skip adders," in *2009 International Conference on Advances in Computational Tools for Engineering Applications*, July 2009, pp. 396–401.

[17] N. M. Nayeem and J. E. Rice, "Online fault detection in reversible logic," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, Oct 2011, pp. 426–434.

[18] N. Farazmand, M. Zamani, and M. B. Tahoori, "Online multiple fault detection in reversible circuits," in *2010 IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct 2010, pp. 429–437.

[19] R. K. James, K. P. Jacob, S. Sasi *et al.*, "Fault tolerant error coding and detection using reversible gates," in *TENCON 2007-2007 IEEE Region 10 Conference*. IEEE, 2007, pp. 1–4.

[20] R. Wille, O. Keszocze, S. Hillmich, M. Walter, and A. Garcia-Ortiz, "Synthesis of approximate coders for on-chip interconnects using reversible logic," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1140–1143.

[21] M. Soeken and A. Chattopadhyay, "Fredkin-enabled transformation-based reversible logic synthesis," in *2015 IEEE International Symposium on Multiple-Valued Logic*, May 2015, pp. 60–65.

[22] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.52.3457

[23] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck, "Quantum circuit simplification using templates," in *Design, Automation and Test in Europe*, March 2005, pp. 1208–1213 Vol. 2.

[24] M. Amy, D. Maslov, and M. Mosca, "Polynomial-time t-depth optimization of clifford + t circuits via matroid partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1476–1489, Oct 2014.

[25] A. Kole, K. Datta, R. Wille, and I. Sengupta, "A nearest neighbor quantum cost metric for the reversible circuit level," in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Nov 2017, pp. 2943–2948.

[26] K. Datta, I. Sengupta, and H. Rahaman, "A post-synthesis optimization technique for reversible circuits exploiting negative control lines," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 1208–1214, April 2015.

[27] M. J. Iacoponi, "Optimal control of latent fault accumulation," in *[1989] The Nineteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, June 1989, pp. 382–388.

[28] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[29] A. B. Forouzan, *Data communications & networking (sie)*. Tata McGraw-Hill Education, 2006.

[30] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225, RevLib is available at http://www.revlib.org.

[31] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, June 2003, pp. 318–323.

Mustafa Altun received his BSc and MSc degrees in electronics engineering at Istanbul Technical University in 2004 and 2007, respectively. He received his PhD degree in electrical engineering with a PhD minor in mathematics at the University of Minnesota in 2012. Since 2013, he has served as an assistant professor at Istanbul Technical University and runs the Emerging Circuits and Computation (ECC) Group. Dr. Altun has been served as a principal investigator/researcher of various research projects including EU H2020 RISE, National Science Foundation of USA (NSF) and TUBITAK projects. He is an author of more than 50 peer reviewed papers and a book chapter, and the recipient of the TUBITAK Success, TUBITAK Career, and Werner von Siemens Excellence awards.

Sajjad Parvin received his B.Sc. in electronics engineering at Hormozgan University, Iran in 2016. He is currently working as researcher in TUBITAK. His current research interests include analog/digital circuit design, Electronic Design Automation(EDA), reversible and quantum computing.

M. Hüsrev Cılasun is currently an engineer at Aselsan A.Ş. His research interests include reversible and quantum circuits, FPGA/ASIC design, digital signal and image processing, and machine learning. He is author of several conference and journal papers on EEG processing and autonomous direction estimation, number theory, and robotics.