# Constructing Online Testable Circuits Using Reversible Logic

Sk. Noor Mahammad and Kamakoti Veezhinathan

***Abstract*—With the advent of nanometer technology, circuits are more prone to transient faults that can occur during its operation. Of the different types of transient faults reported in the literature, the single-event upset (SEU) is prominent. Traditional techniques such as triple-modular redundancy (TMR) consume large area and power. Reversible logic has been gaining interest in the recent past due to its less heat dissipation characteristics. This paper proposes the following: 1) a novel universal reversible logic gate (URG) and a set of basic sequential elements that could be used for building reversible sequential circuits, with 25% less garbage than the best reported in the literature; (2) a reversible gate that can mimic the functionality of a lookup table (LUT) that can be used to construct a reversible field-programmable gate array (FPGA); and (3) automatic conversion of *any* given reversible circuit into an online testable circuit that can detect *online* any single-bit errors, including soft errors in the logic blocks, using theoretically proved minimum garbage, which is significantly lesser than the best reported in the literature.**

***Index Terms*—Flip-flop, garbage, low power dissipation, online testing and digital circuits, reversible logic and gates.**

## I. Introduction

SPACE and mobile applications demand performance, low power dissipation, and fault tolerance. The major limitation in this type of applications is that the hardware executing the same has to survive on battery/solar power. Hence, these systems are to be extremely power aware. Traditional fault-tolerant systems that use techniques such as triple-modular redundancy (TMR) consume more logic and, hence, dissipate larger power than nonfault-tolerant versions of the same. As a result, the overall power dissipation of the chip significantly increases. This, in turn, makes the system power hungry and, hence, not suitable for portable applications. Another major application of hardware that is well reported in the literature is in the real-time/online safety critical system domain. For example, the safety logic systems for nuclear reactors are essentially hardware based and are greatly exposed to charged particles. Specifications of such systems demand online detection of faults to ensure correct and safe execution.

On the other hand, the semiconductor device density on chips continues to follow the Moore's law and currently enables packing of billions of transistors on very small die areas. With increasing device densities, the operating voltages and dimensions of the devices continue to shrink. Currently, the manufactured devices employ nanometer technology. These devices operate by storing very less electric charges when compared with the previous technologies. This has resulted in these devices being highly susceptible to being affected by the striking of higher charged particles in the external environments in which they are operating. The striking of externally charged particles normally causes a glitch in the combinational part of the circuit. This glitch, when it occurs closer to a clock edge, may propagate and get stored in a flip-flop, thereby changing the state of the circuit. This causes an error, which is called the *soft error*. The soft error does not damage the circuit and can be corrected by resetting the state of the circuit. Unlike manufacturing defects, the soft errors cannot be detected using conventional design-for-testability (DFT) techniques [1], although there are techniques reported in the literature, for example, the built-in soft-error resilience design paradigm [2], that *reuses* the existing on-chip DFT resources to reduce the soft-error rate. Putting all these together, in the next nanometer era, there is certainly a need for *a methodology to design power-aware systems that can detect online errors, including soft errors*.

From the point of view of power dissipation, researchers have looked into technologies based on carbon nanotubes [3], microelectromechanical system switches [4], and reversible logic [5]. Reversible logic offers a lot of promises in terms of practical realization [6]. As a justification to the foregoing statement, the Reversible Computing Research Group, Massachusetts Institute of Technology, has developed a *proof-of-concept* reconfigurable reversible chip, which is called the FlatTop [7].

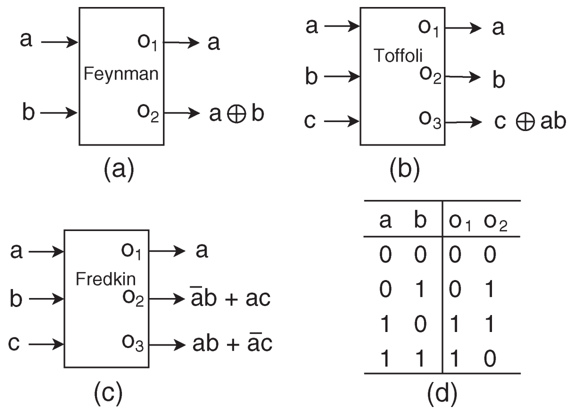
This paper presents techniques for the construction of reversible circuits that realize the functionality of any given arbitrary digital circuit, including a two-input lookup table (LUT), and automatically converting the aforementioned reversible circuits into online testable circuits. In essence, this paper collectively addresses the problems of *low power design* and *online error detection*. The rest of this paper is organized as follows: Section II presents the related literature survey. Section III proposes efficient building blocks for the construction of reversible circuits. Section IV details the automatic conversion of a reversible circuit into an online testable circuit. Section V illustrates the proposed methodology by constructing an online testable reversible decoder circuit. This section also presents SPICE-based simulation results to demonstrate the practical feasibility and the efficiency of the proposed approach. Section VI concludes this paper.

Manuscript received May 4, 2008; revised November 20, 2008. First published August 18, 2009; current version published December 9, 2009. This work was supported by the Reconfigurable and Intelligent Systems Engineering (RISE) Laboratory. The Associate Editor coordinating the review process was Dr. S. Demidenko.

Sk. N. Mahammad is with the RISE Laboratory, Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Madras, Chennai 600 036, India (e-mail: veezhi@gmail.com).

K. Veezhinathan is with the Department of Computer Science and Engineering, IIT Madras, Chennai 600 036, India (e-mail: kama@cse.iitm.ac.in).

Digital Object Identifier 10.1109/TIM.2009.2022103




| a | b | $o_1$ | $o_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

Fig. 1. Reversible gates. (a) Feynman. (b) Toffoli. (c) Fredkin.

## II. Literature Survey

An operation $p$ is said to be *adiabatic* or *physically reversible* if there is neither energy to heat conversion nor change in entropy due to $p$. In reversible logic, the state of the computational device just prior to an operation is uniquely determined by its state just after the operation. In other words, no information about the computational state is lost. Thus, the systems realized using reversible logic can be viewed as deterministic state machines. Landauer [8] showed that, for every bit of information that is erased during an irreversible logic computation, $kT \ln 2$ joules of heat energy are generated, where $k$ is the Boltzmann's constant, and $T$ is the temperature (in kelvins) at which the system operates. Bennett [9] showed that the $kT \ln 2$ amount of energy dissipation would not occur if a computation is carried out in a reversible way. Thus, a reversible operation ensures low energy dissipation [5], [10]. In the nanometer era of semiconductors, the size of a basic device is rapidly approaching the elementary particle level. At this level, matter is governed by principles of classical and quantum mechanics. This, in turn, states that the operations carried out by the devices at the particle level are *reversible* [11], indicating that the future computers that are to be realized using such reversible devices can be adiabatic/reversible [8], [10]. Hence, reversible logic is gaining grounds.

A reversible gate by definition is an $n \times n$ logical cell that has the same number $n$ of inputs and outputs, with a one-to-one mapping between the input and output vectors. Direct fan-outs from the outputs of reversible gates or connecting an output of gate $G$ directly to any input of $G$ are not permitted while constructing circuits with reversible gates [11].

A detailed elaborate list of reversible gates reported in the literature is presented in [12]. Some prominent among them are the Feynman gate [13] [Fig. 1(a)], the Toffoli gate [14] [Fig. 1(b)], the Fredkin gate [11] [Fig. 1(c)], the Kerntopf gate [12], and the Margolus gate [15]. An $n \times n$ reversible gate can uniquely be represented by the permutation of $2^n$ integers, as defined by its input–output bijection, along with the order of the inputs and outputs. For example, Fig. 1(d) shows the input–output bijection of the Feynman gate shown in Fig. 1(a). It is straightforward to see that the Feynman gate can uniquely be represented by the tuple $(((a, b) \rightarrow (o_1, o_2)), (0, 1, 3, 2))$. Many techniques for implementing the reversible circuits have been proposed until now, which include charge-recovery logic (CRL) [16], split-level CRL [17], reversible energy recovery logic (RERL) [18], [19], and NMOS RERL (nRERL) [20]. In addition, optoelectronic and nanometer-based implementations of reversible circuits are presented in [21] and [22]. However, there has been relatively little progress in the synthesis of digital designs into reversible circuits. The construction of reversible sequential logic elements is dealt with in [23]. One of the major issues in designing a reversible circuit is *garbage minimization*. Garbage is defined as the number of outputs added to make an $n$-input $k$-output Boolean function [$(n, k)$ function] reversible [24]. Realizing programmable logic using reversible gates is an interesting area of research. Reversible programmable logic arrays are proposed in [25].

Very little previous research has been done on testable reversible circuits. Conditions for a complete test set construction were discussed, and the problem of finding a minimum test set was formulated as an integer linear program with binary variables in [26]. A new fault model, which is called the *missing gate fault model*, was proposed to represent physical failure modes of quantum technologies [27]. An online testing technique for reversible logic circuits was proposed in [28]. The technique involved two testable reversible logic gates $R1$ and $R2$, which can be used to implement various reversible digital circuits. Another online testing technique for reversible logic circuits was given in [29] and [30]. This technique proposed three reversible logic gates $(R1, R2, R3)$. The gates $R1$ and $R2$ proposed in [28]–[30] are the same. A combination of these $R1$ and $R2$ gates was used to construct testable logic blocks. The $R3$ gate was used to construct a two-pair rail checker for detecting errors in any pair of blocks in the circuit.

The contributions of this paper are given as follows:

1) a novel universal reversible logic gate (URG) that can realize on the same outputs the following three pairs of two argument operators for different values of controlling signals: a) OR and AND; b) NOR and NAND; c) EXOR gate; and d) fan-out gate;
2) a set of basic sequential elements that could be used for building reversible sequential circuits, with 25% less garbage than those proposed in [23];
3) a reversible gate that can mimic the functionality of a LUT;
4) automatic conversion of *any* given reversible circuit into an online testable circuit that has theoretically proved minimum garbage and that can detect any single-bit errors, including soft errors in the logic blocks. The latest methodology reported in [30] consumes more garbage than the proposed technique and assumes that the complete circuit is realized in terms of a particular gate $R1$.

## III. Building Blocks for Reversible Circuits and Reversible LUTs

### A. Reversible Gates

Fig. 2 illustrates the proposed URG represented by the tuple $(((a, b, c) \rightarrow (o_1, o_2, o_3)), (0, 5, 6, 3, 4, 1, 7, 2))$. It is

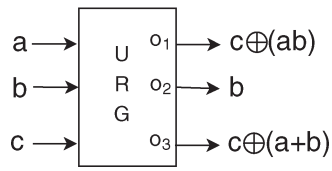



Fig. 2. URG.

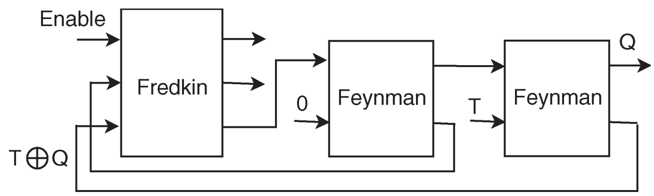



Fig. 3. Reversible positive-level-triggered T flip-flop.

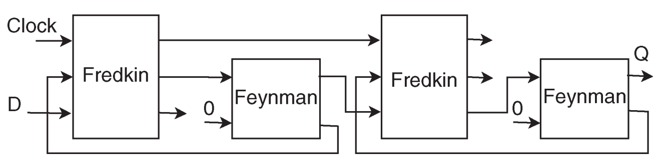



Fig. 4. Reversible positive-edge-triggered D flip-flop.

straightforward to see in Fig. 2 that:

1) by setting input $c$ to 0(1), the OR and AND (NOR and NAND) of inputs $a$ and $b$ are realized in $o_1$ and $o_3$, respectively;
2) by setting $b$ to 1, the XOR of $a$ and $c$ is realized;
3) by setting $a$ and $b$ to '0' a two-fan-out circuit of $c$ is realized in $o_1$ and $o_3$, respectively.

To the best of our knowledge, this is the first multifunctional reversible gate reported in the literature that realizes all the four basic functions, namely, AND, OR, NAND, and NOR. A similar attempt is reported in [12], which does not realize the minimal functionally complete functions [31], namely, NAND and NOR.

### B. *Reversible Sequential Elements*

As stated in Section II, not much research has been reported relating to the construction of sequential reversible circuits. The construction of latches and flip-flops using reversible gates is described in [23] and [32]. This section presents realizations of some sequential elements that are better than those reported in [23]. Fig. 3 proposes a reversible positive-level-triggered T flip-flop. Here, a Fredkin gate is used as a 2:1 multiplexer, and the Feynman gate is used for realizing a fan-out of 2. The constructions of the master–slave D flip-flop and the T flip-flop are shown in Figs. 4 and 5, respectively. All the flip-flops shown in Figs. 3–5 follow the respective truth tables, as stated in [31]. The comparison between the proposed designs and the most recent designs reported in [23] is shown in Table I. It is evident from Table I that the proposed designs lead to a 25% reduction in garbage and also a reduction in the number of 3 × 3 reversible gates, as compared with the best [23] reported in the literature.

### C. *Reversible LUTs*

A $k$-input LUT is a programmable logic element that can be configured to realize any one of the $2^{2^k}$ distinct $k$-input Boolean functions. Typically, these are realized using $2^k \times 1$ memory, which can store the truth table of any of the $k$-input Boolean function. The 16 distinct two-input Boolean functions can uniquely be represented by the integers 0–15. For example, the AND function $f(p,q) = p\&q$, whose truth table is (0, 0, 0, 1), can be represented by the integer 1. Of the 16 distinct two-input functions, the two constant functions (0, 0, 0, 0) and (1, 1, 1, 1), which are represented by the integers 0 and 15, respectively, do not need a logic element for realization. This section proposes a reversible gate $LUTG$ represented by the tuple $(((a,b,c) \rightarrow (O_2, O_1, O_0)), (1,5,3,0,4,2,7,6))$. The mapping is shown in Fig. 6(a). It is interesting to note that $LUTG$ can be used to realize all the 14 nonconstant and distinct two-input Boolean functions. Fig. 6(b) illustrates the procedure to use $LUTG$ for realizing the 14 distinct two-input Boolean functions. For example, it is seen from the third row of Fig. 6(b) that, by setting the inputs $(a,b,c)$ of $LUTG$ to $(p,0,q)$, the AND function $p\&q$ represented by the integer 1 can be realized on output $O_1$. Similarly, the other 13 nonconstant two-input Boolean functions can be realized using $LUTG$, as described in Fig. 6(b). It is straightforward to verify the correctness of Fig. 6(b) using the mapping shown in Fig. 6(a).

It is interesting to note that there are 408 such different permutations, including the above, each representing a different reversible gate, that can realize the same functionality of $LUTG$, as described in the previous paragraph. The major advantage offered by these reversible gates is the *memoryless realization* of an LUT.

## IV. Testing Reversible Circuits

This section proposes a technique for *automatically* converting *any* given reversible circuit into an online testable circuit. Unlike the technique proposed in [30], the proposed conversion mechanism is independent of the type of the reversible gates used. Every testable logic block in [30] comprises two 4 × 4 reversible gates $R1$ and $R2$ cascaded together, as shown in Fig. 7. The output $q$ of $R1$ is $q = u \oplus v \oplus w$. Similarly, the output $s$ of $R2$ is $s = x \oplus y \oplus z$. Thus, $q$ and $s$ are the parity outputs of $R1$ and $R2$, respectively. As stated in [30], *if $R1$ is fault free, its parity output $q$ and the parity output $s$ of $R2$ should be complementary; otherwise, the presence of a fault is assumed. A similar assumption is made about the faults in logic blocks in this paper.*

The conversion procedure has two steps: In the first step every reversible gate $G$ of the design is converted into a deduced reversible gate $DRG(G)$ *without* modifying its original functionality. In the second step, $DRG(G)$ is cascaded with a deduced identity gate to realize a testable version of $G$, namely, $TRG(G)$. The remaining of this section will explain in detail the aforementioned two steps.

*Step 1—Construction of $DRG(R)$:* Let the $n \times n$ reversible gate $R$ be shown in Fig. 8(a), with the input vector $I = [I_1, I_2, \ldots, I_n]$ and the output vector $O = [O_1, O_2, \ldots, O_n]$.

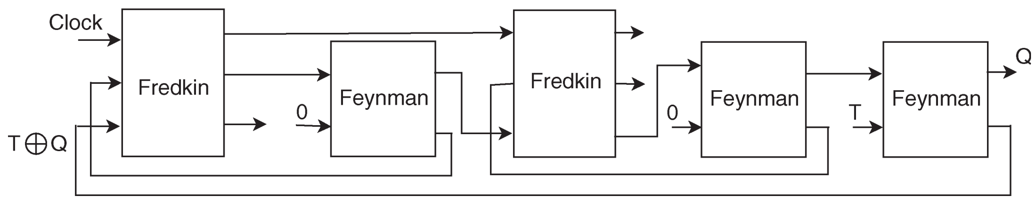



Fig. 5. Reversible positive-edge-triggered T flip-flop.

TABLE I
COMPARISON OF THE PROPOSED SEQUENTIAL DESIGNS WITH THE EXISTING DESIGNS

| Type | No. of Gates in Design | | | | No. of Garbage outputs | | % of Improvement in Garbage |
|---|---|---|---|---|---|---|---|
| | Existing [23] | | Proposed | | Existing Design [23] | Proposed Design | |
| | 2×2 | 3×3 | 2×2 | 3×3 | | | |
| T - Flip-Flop | 0 | 2 | 2 | 1 | 2 | 2 | 0 |
| Master Slave D - Flip-Flop | 3 | 2 | 2 | 2 | 4 | 3 | 25 |
| Master Slave T - Flip-Flop | 2 | 3 | 3 | 2 | 4 | 3 | 25 |

| a | b | c | $O_2$ | $O_1$ | $O_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

(a)

| a | b | c | $O_2$ | $O_1$ | $O_0$ |
|---|---|---|---|---|---|
| 0 | p | q | 4 | 2 | 14 |
| p | 0 | q | 6 | 1 | 12 |
| p | q | 0 | 3 | 5 | 13 |
| 1 | p | q | 11 | 7 | 2 |
| p | 1 | q | 3 | 11 | 10 |
| p | q | 1 | 9 | 3 | 8 |

(b)

Fig. 6. Bijection to realize all nonconstant two-input Boolean functions.

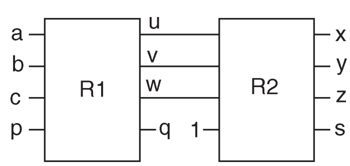



Fig. 7. Testable block of [30].

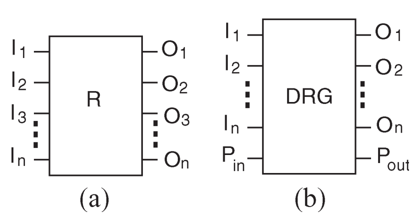



Fig. 8. (a) $n \times n$ reversible gate. (b) $DRG$.

As the gate is reversible, we have a one-to-one mapping between vector $I$ and vector $O$. Let $O_i = F_i(I_1, I_2, \ldots, I_n)$, shortly denoted by $F_i(I)$, $1 \leq i \leq n$. A $DRG$ of $R$, i.e., $DRG(R)$, is constructed by adding an extra input bit $P_{\rm in}$ and the corresponding output bit $P_{\rm out}$ to the gate $R$, as shown in Fig. 8(b). The bijection for $DRG(R)$ is derived as follows: *The outputs $(O_1, O_2, \ldots, O_n)$ remain the same, as defined by the original bijection of* $R$. The new output $P_{\rm out} = F \oplus P_{\rm in}$, where $F = F_1(I) \oplus F_2(I) \oplus \cdots \oplus F_n(I)$.

*Lemma 1:* $DRG(R)$ is reversible.

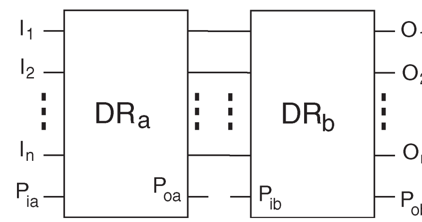



Fig. 9. Cascade of $DRG$s to form the $TRC$.

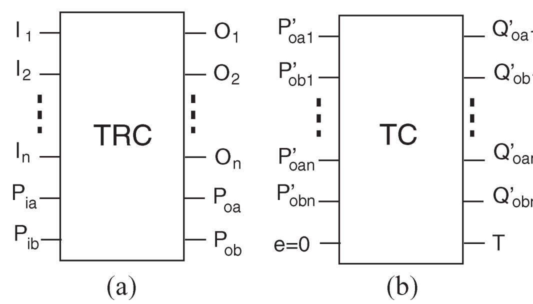



Fig. 10. (a) $TRC$. (b) $TC$.

*Proof:* Let $[I_1, I_2, \ldots, I_n]$ map to $[O_1, O_2, \ldots, O_n]$ in $R$. Given that $P_{\rm in} = 0$, it is easily seen that $[I_1, I_2, \ldots, I_n, 0]$ in $DRG(R)$ maps to $[O_1, O_2, \ldots, O_n, F]$, since $P_{\rm out} = F$ when $P_{\rm in} = 0$. Similarly, $[I_1, I_2, \ldots, I_n, 1]$ in $DRG(R)$ maps to $[O_1, O_2, \ldots, O_n, \sim F]$, since $P_{\rm out} = \sim F$ when $P_{\rm in} = 1$. The foregoing fact and given that $R$ is reversible imply that $DRG(R)$ is also reversible. ■

*Step 2—Construction of the Testable Reversible Cell $TRC(R)$:* Let $R$ be an $n \times n$ reversible gate. Consider an $n \times n$ gate $X$ such that all the inputs to the gate $X$ are mapped to the outputs without any change. It is obvious that the gate $X$ is reversible. Let $DR_a = DRG(R)$ and $DR_b = DRG(X)$. By Lemma 1, $DR_a$ and $DR_b$ are $(n + 1) \times (n + 1)$ reversible gates. Cascade the gates $DR_a$ and $DR_b$, as shown in Fig. 9, by connecting the first $n$ outputs of $DR_a$ to the first $n$ inputs of $DR_b$ in order. The resultant gate of Fig. 9 can be viewed as an $(n + 2) \times (n + 2)$ gate, as shown in Fig. 10(a), which forms the $TRC$ of $R$, i.e., $TRC(R)$. The input vector of $TRC(R)$ is defined as $[I, P_{ia}, P_{ib}]$, where $I$ is the input vector of gate

$R$, and $[P_{ia}, P_{ib}]$ are the additional input bits added to $R$ and $X$ while constructing the gates $DR_a$ and $DR_b$, respectively. Similarly, the output vector is defined as $[O, P_{oa}, P_{ob}]$, where $O$ is the output vector of gate $R$, and $[P_{oa}, P_{ob}]$ are the additional output bits added to $R$ and $X$ while constructing the gates $DR_a$ and $DR_b$, respectively.

*Lemma 2:* $TRC(R)$ is reversible.

*Proof:* The fact that $DR_a$ and $DR_b$ are reversible and the construction of $TRC(R)$ imply the lemma. ■

*Lemma 3:* $TRC(R)$ has two fault detection properties.

1) Setting $P_{ia} = P_{ib}$, then the output of $TRC(R)$ is erroneous if the parity bits $P_{oa}$ and $P_{ob}$ are complementary.
2) Setting $P_{ia} = \sim P_{ib}$, then the output of $TRC(R)$ is erroneous if the parity bits $P_{oa}$ and $P_{ob}$ are same.

*Proof:* From the construction of $TRC(R)$, note that $P_{oa} = P_{ia} \oplus O_1 \cdots \oplus O_n$ and $P_{ob} = P_{ib} \oplus O_1 \cdots \oplus O_n$. The proof for Case 1 is as follows: Given $P_{ia} = P_{ib}$ and if $P_{oa} \neq P_{ob}$, then one of the outputs of $DR_a$ or $DR_b$ is faulty. The proof for Case 2 follows in similar lines. ■

### A. Construction of an Online Testable Circuit

Algorithm 1 details the proposed approach for converting any reversible circuit to an online testable reversible circuit.

**Algorithm 1**: Construction of an Online Testable Circuit.
**Input**: Reversible Circuit $C$
**Output**: An online testable reversible circuit $C^T$

1 Construct $C^S$ by replacing every reversible gate $R$ in $C$ by $TRC(R)$. While constructing $TRC(R)$ for each $R$, set the parity input bits $P_{ia} = P_{ib}$. By Lemma 2, $C^S$ is reversible.

2 Let $n$ be the number of reversible gates in $C$. Construct a $(2n+1) \times (2n+1)$ test cell $(TC)$, as shown in Fig. 10(b), as follows: Let $P_{oak}$ and $P_{obk}$ be the output parity bits of the $k$th $TRC$ of $C^S$.
- The first $2n$ inputs of the $TC$ are the output parity bits from each of the $n$ $TRC$s of $C^S$, i.e., $P'_{oai} = P_{oai}$ and $P'_{obi} = P_{obi}$, $1 \leq i \leq n$.
- The last input bit of the $TC$, which is denoted by $e$, is set to either logic 0 or logic 1.
- The first $2n$ inputs of the $TC$ are transferred to the output without any change, i.e., $Q'_{oai} = P'_{oai}$ and $Q'_{obi} = P'_{obi}$, $1 \leq i \leq n$ [refer to Fig. 10(b)].
- The last output bit $T$ of the $TC$ is defined as follows: $T = [((P_{oa1} \oplus P_{ob1}) + (P_{oa2} \oplus P_{ob2}) + \cdots + (P_{oan} \oplus P_{obn})) \oplus e]$.

3 Cascade $C^S$ and $TC$, as stated in step 2, to obtain $C^T$.

*Theorem 1:* The cell $TC$ constructed in Algorithm 1 has three properties.

1) It is reversible.
2) If there is a single-bit error in the output of any $TRC$ in $C^T$, then $T = 1$, provided $e = 0$.
3) Function $T$ is implemented with minimum possible garbage.

*Proof:* Three observations can easily be seen.

1) $[P_{oa1}, P_{ob1}, \ldots, P_{oan}, P_{obn}, 0]$ maps to $[P_{oa1}, P_{ob1}, \ldots, P_{oan}, P_{obn}, T]$ and $[P_{oa1}, P_{ob1}, \ldots, P_{oan}, P_{obn}, 1]$ maps to $[P_{oa1}, P_{ob1}, \ldots, P_{oan}, P_{obn}, \sim T]$, where $T = [((P_{oa1} \oplus P_{ob1}) + (P_{oa2} \oplus P_{ob2}) + \cdots + (P_{oan} \oplus P_{obn})) \oplus e]$. Hence, $TC$ is reversible.
2) From the step 1 of Algorithm 1, it is seen that $P_{ia} = P_{ib}$ in the $TRC$. If there is a single-bit logical error in, e.g., the $i$th $TRC$, then by Lemma 3, its $P_{oai}$ will be complementary to $P_{obi}$. Therefore, $P_{oai} \oplus P_{obi} = 1$. Hence, $T = 1$.
3) The role of the function $T$ in the $TC$ is to test if the output parity bits of any of the $n$ $TRC$s differ. Thus, $T$ is a Boolean function with $2n$ inputs, which is realized as an OR of $n$ two-input EXOR functions. It is straightforward to see that $T$ evaluates to 0 if and only if all the $n$ two-input EXOR functions evaluate to 0. There are two possible inputs that set a two-input EXOR function to 0. This implies that there are $2^n$ possible inputs that set $T$ to 0, and the remaining $2^{2n} - 2^n$ inputs set $T$ to 1. For an $n$-input $k$-output function $f$, the minimum number of garbage bits required to make it reversible is $\lceil \log_2 M \rceil$, where $M$ is the maximum number of times an output pattern is repeated in the truth table of $f$ [33]. For the function $T$, $M = 2^{2n} - 2^n$, where $n$ is the number of $TRC$s in $C^T$. Therefore, $\log_2 M = \log_2(2^{2n} - 2^n) > \log_2(2^{2n}/2) = (2n-1)$, for $n > 1$. Therefore, $\lceil \log_2 M \rceil \geq 2n$. Hence, the minimum garbage that must be produced in the $TC$ that implements $T$ is $2n$. ■

### B. Construction of Hierarchical Online Testing Circuitry

The complexity (number of inputs) of the $TC$ gate linearly increases with $n$. The proposed hierarchical testing methodology is to have a $TC$ for every module (circuit of reasonably small size) of the chip. This section proposes a simple reversible gate, which is called the multimodular $TC$ $(MMTC)$, that will take as inputs the $T$ bits of the $TC$s and output an error (set one of its outputs to 1) if any one of the input $T$ bits is set to 1. Assume that there are $k$ TCs. The $MMTC$ is a $(k+1) \times (k+1)$ reversible gate, which is defined in the list that follows.

Inputs: $T_1, T_2, \ldots, T_k$ and $e$, where $e$ can be either 0 or 1.
Outputs: $T_1, T_2, \ldots, T_k$, where $MMT = (T_1 + T_2 + \cdots + T_n) \oplus e$.

*Lemma 4:* The cell $MMTC$ has three properties.

1) It is reversible.
2) $MMTC$ detects any single-bit error in $C_1, C_2, \ldots, C_k$, where $C_i$ is an online testable reversible module $\forall i$.
3) Function $MMT$ is implemented with minimum possible garbage.

*Proof:* Three observations can easily be seen.

1) $[T_1, T_2, \ldots, T_k, 0]$ maps to $[T_1, T_2, \ldots, T_k, MMT]$ and $[T_1, T_2, \ldots, T_k, 1]$ maps to $[T_1, T_2, \ldots, T_k, \sim MMT]$, where $MMT = (T_1 + T_2 + T_3 + \cdots + T_k) \oplus e$. Hence, $MMTC$ is reversible.

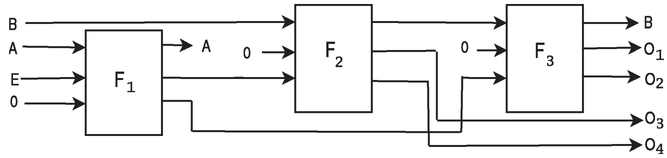



Fig. 11. Two-to-four reversible decoder $C$.

2) $T_i$ bit is the test bit from the $TC$ of module $C_i$. The multimodular test $(MMT)$ bit is the logical OR of $T_i$ bits for $i = 1, 2, \ldots, k$. Hence, if there is any error in any of the modules $C_i$, then $MMT$ will be logical 1, provided $e = 0$. Hence, the error is detected.

3) As mentioned earlier, the minimum number of garbage bits required to make the function $MMT$ reversible is $\lceil \log_2 M \rceil$, where $M$ is the maximum number of times an output pattern is repeated in the truth table of $MMT$ [33]. Given that the bit $e$ is set to 0, the function $MMT$ is a $k$-input OR function, which has $2^k - 1$ zeros in its truth table. Hence, $M = 2^k - 1$, where $k$ is the number of online testable reversible modules attached to the $MMTC$. Therefore, $\lceil \log M \rceil = \log(2^k - 1) = k$. Hence, the minimum garbage needed for making the function $MMT$ reversible is $k$. ■

To sum up, the hierarchical online testable reversible circuit can be constructed to realize a given Boolean function, as follows: 1) realize the Boolean function in terms of reversible gates; 2) convert each reversible gate $R$ in the design into $TRC(R)$; 3) partition the $TRC$s into several sets; 4) for each partition of $TRC$s, connect their output parity bits to a $TC$; 5) partition the $TC$s into several blocks; 6) for each partition of $TC$s, connect their $T$ outputs to an $MMTC$; 6) partition the $MMTC$ into several sets; 7) for each partition of $MMTC$s, connect their $MMT$ outputs to another $MMTC$; and 8) repeat steps 6 and 7 until the number of $MMTC$s becomes 1. *An error is detected when the $MMT$ bit of the last (root) $MMTC$ introduced in step 7 is set to 1. Given that $TRC$, $TC$, and $MMTC$ are reversible implies that the circuit built, as aforementioned, is also reversible.*

In the next section, it will be shown by an example that the proposed technique for constructing online testable reversible circuits yields circuits with much lesser garbage than those constructed using the latest technique reported in the literature [30].

## V. Illustration of the Proposed Technique

In this section, a reversible decoder circuit is converted to an online testable reversible decoder circuit using the proposed methodology. A decoder is a combinational circuit that converts binary information from $n$ input lines to a maximum of $2^n$ unique output lines. Consider a two-to-four decoder with the enable bit, as shown in Fig. 11. The construction of the reversible decoder circuit uses three Fredkin gates $(F_1, F_2, F_3)$. In Fig. 11, $A$ and $B$ are the one-bit inputs to the decoder, $E$ is the one-bit enable, and $O_1$, $O_2$, $O_3$, and $O_4$ are the output bits of the decoder. The functionality of the Fredkin gate shown in Fig. 1(c) implies the truth table shown in Table II, which captures the functionality of the decoder.

TABLE II
Truth Table for the Decoder Circuit

| E | A | B | $O_1$ | $O_2$ | $O_3$ | $O_4$ |
|---|---|---|---|---|---|---|
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

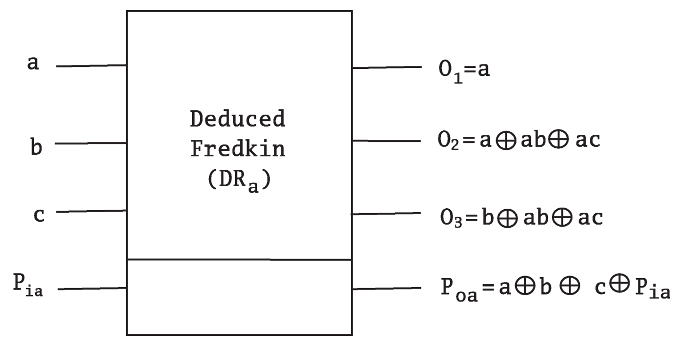



Fig. 12. Deduced Fredkin gate $DR_a$.

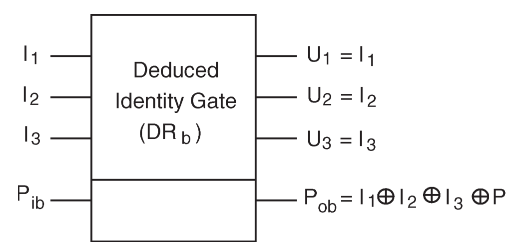



Fig. 13. Deduced gate $DR_b$.

The following steps detail the conversion of the reversible decoder, as shown in Fig. 11, to an online testable decoder.

Step 1) Each $3 \times 3$ Fredkin gate $F$ is replaced by a $4 \times 4$ $DRG$ $DR_a(F)$, as shown in Fig. 12.

Step 2) Each $DR_a(F)$ is connected to the deduced identity gate $DRG(X)$ to form $TRC(F)$. $DRG(X)$ is shown in Fig. 13.

Step 3) Fig. 14 shows the online testable version of the decoder shown in Fig. 11. The gate $TRC_i$ in Fig. 14 represents $TRC(F_i)$, $1 \le i \le 4$, $F_i$, as shown in Fig. 11.

Step 4) As shown in Fig. 14 the parity output bits of the $TRC$s are connected to the $TC$. The output $T$ of the $TC$ will be set to 1 if any one of the output of any reversible gate in the circuit is faulty.

### A. Simulation Results

Fig. 15 shows the two-to-four reversible decoder realized using the reversible gate $R1$, and Fig. 16 is its online testable version, which is constructed as stated in [30]. The gates $R2$ and the two-pair two-rail checker shown in Fig. 16 are as described in [30]. Transistor-level realizations of the reversible logic circuits are presented in [23] and [30]. These realizations are *functionally equivalent* to the gate-level descriptions and

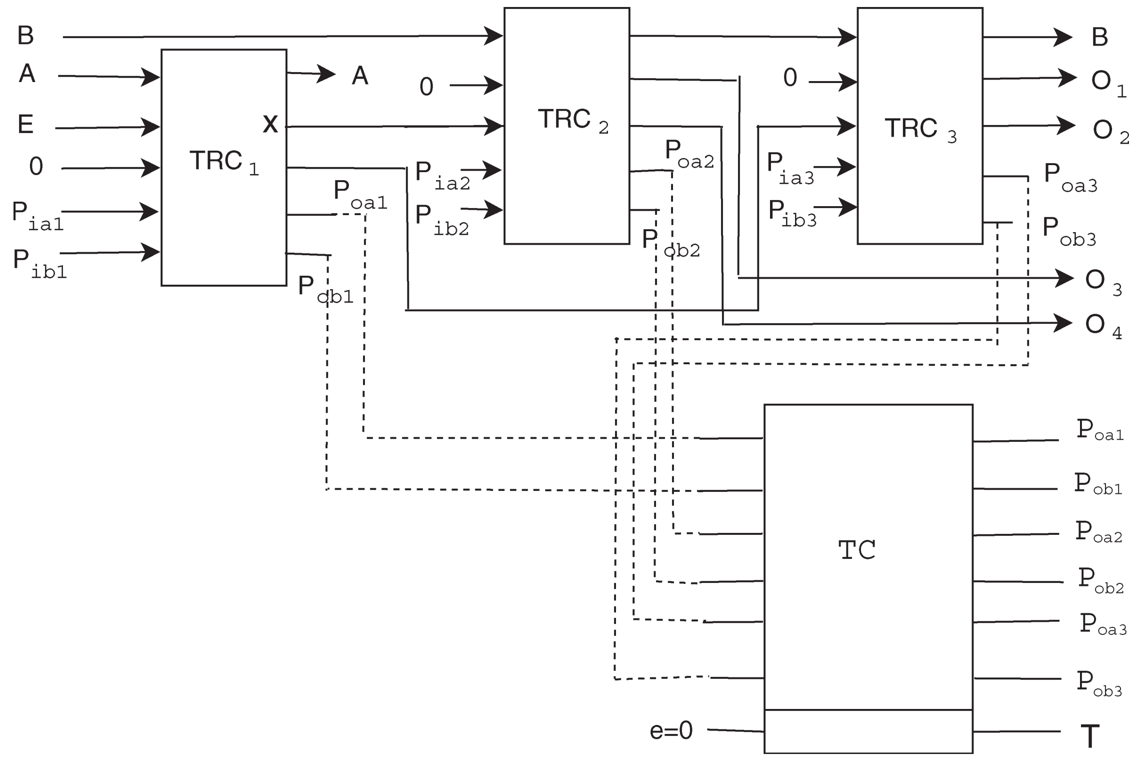



Fig. 14. Online testable reversible decoder circuit $C^T$.

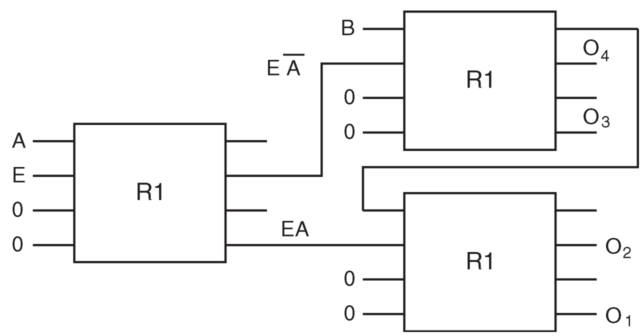



Fig. 15. Reversible decoder construction using the $R1$ gate.

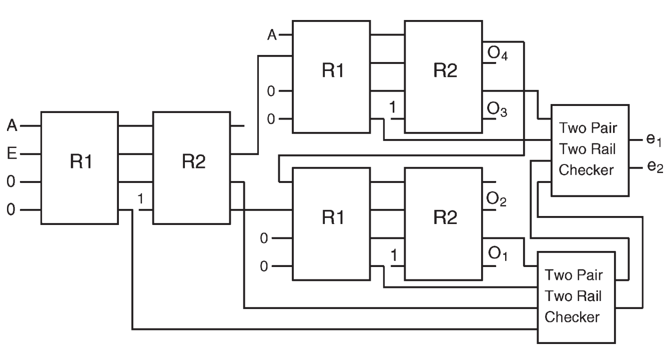



Fig. 16. Online testable $R1$-based reversible decoder.

do not employ any *reversible energy recovery* principles. The reversible energy recovery principles such as RERL, nRERL, etc., as mentioned in Section II, result in circuits that dissipate very less dynamic power when compared with their respective static CMOS implementations. In this paper, the RERL technique described in [34] is employed for constructing the

TABLE III
SIMULATION RESULTS FOR THE URG SHOWN IN FIG. 2

| Type | Dyn. Power (in $10^{-15}$ Watts) | Num. of Transistors | Delay (in $ns$) |
|---|---|---|---|
| Static CMOS | 12.8 | 56 | 1.563 |
| Reversible Logic | 0.44058 | 144 | 5.9 |

TABLE IV
SIMULATION RESULTS FOR THE TWO-TO-FOUR DECODER

| Type | Dyn. Power (in $10^{-15}$ Watts) | Num. of Transistors | Delay (in $ns$) |
|---|---|---|---|
| Static CMOS | 11.92 | 40 | 1.66 |
| R1-based (Fig. 15) [30] | 1.91375 | 744 | 8.1 |
| Proposed (Fig. 11) | 0.80928 | 384 | 5.7 |

reversible circuits using CMOS transmission gates. The circuits are simulated using SPICE to obtain the voltage and current waveforms. The 180-nm transistor model from TSMC is used for the simulation. The transistors used in the simulation are 240 nm in width and 180 nm in length. The employed power calculation method is similar to the method described in [35]. The results of the different simulations are summarized in the list that follows.

1) Table III summarizes the results obtained by simulating a static CMOS implementation of the URG gate shown in Fig. 2 and its reversible RERL implementation. *The reversible implementation consumes 29.23 times lesser power, 2.57 times more transistors, and 3.77 times more delay than its static CMOS counterpart.*
2) Table IV summarizes the results obtained by simulating three circuits, namely, a static CMOS implementation of a two-to-four decoder; an R1-gate-based RERL implementation of the same, as shown in Fig. 15; and the proposed

TABLE V
SIMULATION RESULTS FOR AN ONLINE TESTABLE TWO-TO-FOUR DECODER

| Type | Dyn. Power (in $10^{-15}$ Watts) | Num. of Transistors | Delay (in $ns$) |
|---|---|---|---|
| R1 based (Fig. 16) [30] | 4.115 | 3048 | 24.8 |
| Proposed (Fig. 14) | 2.249 | 1496 | 13.2 |

Fredkin-gate-based RERL implementation, as shown in Fig. 11. The R1-based implementation consumes 6.23 times lesser power, 18.6 times more transistors, and 4.88 times more delay than its static CMOS counterpart. The proposed Fredkin-gate-based implementation consumes 14.73 times lesser power, 9.6 times more transistors, and 3.43 times more delay than its static CMOS counterpart. *The proposed Fredkin-gate-based implementation consumes 2.36 times lesser power, 1.94 times lesser transistors, and 1.42 times lesser delay than the R1-based implementation*.

3) Table V summarizes the results obtained by simulating the two online testable decoders shown in Figs. 14 and 16. *The proposed online testable implementation consumes 1.83 times lesser power, 2.04 times lesser transistors, and 1.88 times lesser delay than the R1-based implementation* [30]. Random errors (with similar effects as single-event upsets) were induced by toggling the signal values on the interconnects within the testable cells such that they caused errors in their respective outputs. The values on the $T$ and $(e_1, e_2)$ lines in Figs. 14 and 16, respectively, were observed to find whether the circuits detect the error or not. In every case, the circuits detected the errors, hence justifying their online testing capabilities.

In all the aforementioned three cases, the power consumption factors of the proposed reversible designs, when compared with the respective static CMOS versions, is much more predominant than the area and the delay factors, suggesting that reversible logic is indeed a viable solution for designing futuristic power-aware systems with a reasonable compromise in area and performance. In Tables 4 and 5, it is clear that even the online testable version of the decoder circuits consumes much less power than the *untestable* static CMOS implementation. In the aforementioned Cases 2 and 3, the proposed designs consumed lesser power, used lesser number of transistors, and consumed lesser delay than their R1-based counterparts, thus illustrating the efficiency of the proposed technique over the recently reported technique in [30].

## VI. CONCLUSION AND FUTURE WORK

This paper has proposed several reversible circuits for realizing both combinational and sequential elements of a given digital circuit. A URG, which is shown to be advantageous for synthesizing multivalued reversible logic [12], was presented. This paper has also proposed a reversible gate that can mimic the functionality of a two-input LUT, thus enabling the memoryless realization of LUTs. This can be used as a programmable logic block in modern field-programmable gate array architectures. This paper has presented efficient realizations of reversible sequential elements. The proposed designs lead to a 25% reduction in garbage and a lesser number of $3 \times 3$ reversible gates when compared with the best reported in the literature. This paper has also proposed a methodology that automatically converts *any* circuit into an online testable reversible circuit with theoretically proved minimum garbage. The resultant testable circuit can detect online any single-bit errors in the logic blocks. An important advantage of the technique is that the design of a given reversible circuit need not be changed for the purpose of adding testability feature to it. This paper has discussed the construction of hierarchical multimodular online testable reversible circuits. The proposed technique was illustrated using an example that converts a Fredkin-based decoder circuit to an online testable reversible decoder circuit. An interesting future work would be to construct reversible circuits that can mimic the functionality of $k$-input LUTs, $k > 2$.

## ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their constructive feedback, which helped in significantly improving the technical quality of this paper, and also S. Potluri and V. Ashwin, Master's students at the RISE Laboratory, IIT Madras, for their timely help with the simulations.

## REFERENCES

[1] M. Abramovici, M. A. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*. New York: IEEE Press, 1995.
[2] T. Mak, S. Mitra, and M. Zhang, "DFT assisted built-in soft error resilience," in *Proc. 11th IEEE IOLTS*, Jul. 2005, p. 69.
[3] T. S. Cho, K.-J. Lee, J. Kong, and A. P. Chandrakasan, "A low power carbon nanotube chemical sensor system," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2007, pp. 181–184.
[4] J. Muldavin and G. Rebeiz, "Nonlinear electro-mechanical modeling of MEMS switches," in *Proc. IEEE MTT-S Int. Microw. Symp. Dig.*, 2001, vol. 3, pp. 2119–2122.
[5] R. W. Keyes and R. Landauer, "Minimal energy dissipation in logic," *IBM J. Res. Develop.*, vol. 14, no. 2, pp. 152–157, Mar. 1970.
[6] L. D. Paulson, "Reversible computing may improve mobile performance," *Computer*, vol. 37, no. 3, p. 21, Mar. 2004.
[7] M. P. Frank, "Reversibility for efficient computing," Ph.D. dissertation, EECS Dept., MIT, Cambridge, MA, Dec. 1999.
[8] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Develop.*, vol. 5, no. 3, pp. 183–191, Jul. 1961.
[9] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Develop.*, vol. 17, no. 6, pp. 525–532, Nov. 1973.
[10] C. H. Bennett, "Notes on the history of reversible computation," *IBM J. Res. Develop.*, vol. 32, no. 1, pp. 16–23, Jan. 1988.
[11] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theor. Phys.*, vol. 21, no. 3/4, pp. 219–253, Apr. 1982.
[12] P. Kerntopf, "Synthesis of multipurpose reversible logic gates," in *Proc. Euromicro Symp. DSD*, 2002, pp. 259–267.
[13] R. Feynman, "Quantum mechanical computers," *Opt. News*, vol. 11, pp. 11–20, Feb. 1985.
[14] T. Toffoli, "Reversible computing," in *Proc. Automata, Lang. Program. Conf.*, 1980, pp. 632–644.
[15] N. Margolus, "Physics and computation," Ph.D. dissertation, MIT, Cambridge, MA, 1988.
[16] S. G. Younis and T. F. Knight, "Practical implementation of charge recovering asymptotically zero power CMOS ," in *Proc. Symp. Res. Integr. Syst.*, 1993, pp. 234–250.
[17] S. G. Younis and T. F. Knight, "Asymptotically zero energy split-level charge recovery logic," in *Proc. Workshop Low Power Des.*, 1994, pp. 177–182.
[18] J. Lim, K. Kwon, and S. I. Chae, "Reversible energy recovery logic circuit without non-adiabatic energy loss," *Electron. Lett.*, vol. 34, no. 4, pp. 344–346, Feb. 1998.

[19] J. Lim, D. G. Kim, and S. I. Chae, "Reversible energy recovery logic circuits and its 8-phase clocked power generator for ultra-low-energy applications," *IEICE Trans. Electron.*, vol. E82-C, no. 4, pp. 646–653, Apr. 1999.

[20] J. Lim, D. G. Kim, and S. I. Chae, "nMOS reversible energy recovery logic for ultra-low-energy applications," *IEEE J. Solid-State Circuits*, vol. 35, no. 6, pp. 865–875, Jun. 2000.

[21] P. Picton, "Optoelectronic multi-valued conservative logic," *Int. J. Opt. Comput.*, vol. 2, pp. 19–29, 1991.

[22] S. Bandyopadhyay, "Nanoelectric implementations of reversible and quantum logic," in *Proc. Supperlattices Microstruct.*, 1998, vol. 23, pp. 445–464.

[23] H. Thapliyal and A. Vinod, "Design of reversible sequential elements with feasibility of transistor implementation," in *Proc. IEEE ISCAS*, May 2007, pp. 625–628.

[24] D. Maslov and G. W. Dueck, "Garbage in reversible design of multiple output functions," in *Proc. 6th Int. Symp. Represent. Methodology Future Comput. Technol.*, Mar. 2003, pp. 162–170.

[25] H. Thapliyal and H. R. Arabnia, *Reversible Programmable Logic Array (RPLA) Using Fredkin and Feynman Gates for Industrial Electronics and Applications*, 2006. vol.arXiv:cs/0609029v1[cs.AR]. [Online]. Available: http://aps.arxiv.org/abs/cs/0609029v1

[26] K. N. Patel, J. P. Hayes, and I. L. Markov, "Fault testing for reversible circuits," in *Proc. 21st IEEE VLSI Test Symp.*, Apr. 2003, pp. 410–416.

[27] J. P. Hayes, I. Polian, and B. Becker, "Testing for missing-gate faults in reversible circuits," in *Proc. 13th Asian Test Symp.*, Nov. 2004, pp. 100–105.

[28] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "A novel approach for on-line testable reversible logic circuit design," in *Proc. 13th Asian Test Symp.*, Oct. 2004, pp. 325–330.

[29] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "Online testable reversible logic circuit design using NAND blocks," in *Proc. 19th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Oct. 2004, pp. 324–331.

[30] D. P. Vasudevan, P. K. Lala, J. Di, and J. P. Parkerson, "Reversible logic design with online testability," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 2, pp. 406–414, Apr. 2006.

[31] S. D. Brown and Z. Vranesic, *Fundamentals of Digital Logic With VHDL Design*. New York: McGraw-Hill, 2004.

[32] S. K. S. Hari, S. Shroff, S. N. Mahammad, and V. Kamakoti, "Efficient building blocks for reversible sequential circuit design," in *Proc. 49th IEEE Int. Midwest Symp. Circuits Syst.*, 2006, pp. 437–441.

[33] D. Maslov and G. W. Dueck, "Reversible cascades with minimal garbage," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 11, pp. 1497–1509, Nov. 2004.

[34] J. Lim, D.-G. Kim, and S.-I. Chae, "A 16-bit carry-lookahead adder using reversible energy recovery logic for ultra-low-energy systems," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 898–903, Jun. 1999.

[35] Y. Ye and K. Roy, "Energy recovery circuits using reversible and partially reversible logic," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 43, no. 9, pp. 769–778, Sep. 1996.

**Sk. Noor Mahammad** is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India.

His areas of interests include VLSI design, reconfigurable computing, and fault-tolerant systems design.

**Kamakoti Veezhinathan** received the Ph.D. degree in computer science and engineering from the Indian Institute of Technology Madras, Chennai, India, in 1995.

He is currently a Professor with the Department of Computer Science and Engineering, Indian Institute of Technology Madras. His areas of interests include computer architecture, reconfigurable computing, high-performance computing, and fault-tolerant systems design.