Design of Reversible Synchronous Sequential Circuits Using Pseudo Reed-Muller Expressions

Mozammel H A Khan, *Senior Member, IEEE*

***Abstract*— Reversible logic has become very promising for low-power design using emerging computing technologies. A number of good works have been reported on reversible combinational circuit design. However, only a few works reported on the design of reversible latches and flip-flops on the top of reversible combinational gates and suggested that sequential circuits be built by replacing the latches and flip-flops and associated combinational gates of the traditional irreversible designs by their reversible counter parts. This replacement technique is not very promising, because it leads to high quantum cost and garbage outputs. In this paper, we propose a novel approach of designing synchronous sequential circuits directly from reversible gates using pseudo Reed–Muller expressions representing the state transition and the output functions of the circuit. We present designs of arbitrary synchronous sequential circuit as well as practically important sequential circuits such as counters and registers. It is found that our direct designs save 1.54%–49.09% quantum cost and 51.43%–81.82% garbage outputs than the replacement design approach suggested earlier.**

***Index Terms*— Counters, pseudo Reed–Muller (PSDRM) expressions, registers, reversible logic, synchronous sequential circuit.**

I. Introduction

LANDAUER [1] states that irreversible logic operations dissipates $kT\ln 2$ J of heat energy for every bit of information loss, where k is Boltzmann's constant and T is the absolute temperature at which the operation is done. Zhirnov *et al*. [2] predict that this heat dissipation would become impossible to remove by 2020 if Moore's Law continues to be in effect by doubling the circuit density in every 18 months. Bennett [3] shows that from thermodynamic point of view, $kT\ln 2$ heat dissipation would not occur if a computation is done in reversible manner. This theoretical thermodynamic limit can be achieved only if the circuit is both logically and physically reversible. However, it has been shown that reversible logic leads to less heat dissipation than the thermodynamic limit of $kT\ln 2$ in physically irreversible low-power CMOS logic [4] and physically reversible superconductor flux logic (SFL) [5]. Thus, reversible logic may be a favorable logic by dissipating less heat than the thermodynamic limit of $kT\ln 2$ for the emerging computing technologies. Reversible logic has already found wide application in many emerging computing technologies such as SFL technology [5], [6], optical technology [7], [8], quantum dot cellular automata technology [9], [10], and nanotechnology [11]. Moreover, reversible logic plays a very important role in quantum computing and quantum information [12]. In summary, reversible logic has become a promising technology for power-efficient emerging computing technologies. Therefore, developing efficient methods for reversible logic synthesis and also designing practically important reversible circuits have become very important.

Most of the reversible logic synthesis attempts are concentrated on reversible combinational logic synthesis [13]–[22]. As feedback is considered as a restriction in reversible logic, some researchers argue that reversible sequential logic is not possible. However, in 1980, Toffoli [23] argued that if the feedback is provided through a delay element, then the feedback information will be available as the input to the reversible combinational circuit in the next clock cycle and sequential logic is possible. However, very recently, only limited attempts have been made in the field of reversible sequential logic synthesis [24]–[30]. References [24]–[28] present reversible designs of building blocks of sequential circuits such as latches and flip-flops on the top of reversible gates and suggest that sequential circuits be constructed by replacing the latches, flip-flops, and other combinational gates of traditional irreversible designs by their reversible counter parts. Following this replacement approach, [28] presents design of a four-bit falling-edge triggered universal shift register and paper [29] presents design of a four-bit level triggered up counter with synchronous parallel load. The first attempt of direct design of synchronous sequential circuit using reversible gates is given in [30], where we design level-triggered up counters using positive polarity Reed–Muller (PPRM) expression for representing the state transition of the counter. The designed up counter is more efficient than the replacement design of [29] in terms of both quantum cost and garbage outputs.

In [30], we express next state of the counter as a function of clock and present state. Then, we express the next state using PPRM expression and realize the PPRM expression using reversible gates. Similar PPRM-based reversible circuit synthesis is done in [31]–[33]. Fixed-polarity Reed–Muller (FPRM) expression requires less or at most same number of product terms than PPRM expression for a given function. Thus, FPRM-based reversible circuit synthesis is more efficient than PPRM-based reversible circuit synthesis. FPRM-based reversible circuit synthesis is done in [34] and [35]. Pseudo Reed–Muller (PSDRM) expression is a more generalized class of Reed–Muller expression and requires less or at most equal number of product terms than FPRM expres-

Manuscript received April 28, 2012; revised November 29, 2012, March 21, 2013, June 27, 2013, and August 22, 2013; accepted November 3, 2013. Date of publication November 22, 2013; date of current version October 21, 2014.

The author is with the Department of Computer Science and Engineering, East West University, Dhaka 1212, Bangladesh (e-mail: mhakhan@ewubd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2013.2290293

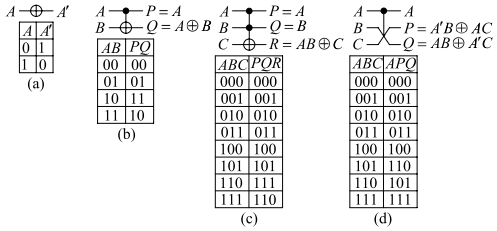

Fig. 1. Commonly used reversible gates. (a) NOT gate. (b) Feynman gate. (c) Toffoli gate. (d) Fredkin gate.

sion for a given function. Thus, PSDRM-based reversible circuit synthesis is more efficient than PPRM- and FPRM-based reversible circuit synthesis. In this paper, we propose PSDRM-based design of arbitrary level-triggered sequential circuit from reversible gates. Moreover, we propose method for providing asynchronous load facility and falling-edge triggering to any level-triggered sequential circuit. As a design example, we show design of an arbitrary two-bit sequential circuit with one external input and one external output, which requires 10.81% less quantum cost and 81.82% less garbage outputs than its replacement design. Then, we show designs of counters and registers. Our four-bit level triggered up counter with asynchronous load saves 1.54% quantum cost and 66.67% garbage outputs than that of [29]. Our four-bit universal register saves 49.09% quantum cost and 51.43% garbage outputs than that of [28].

The rest of the paper is organized as follows. In Section II, we provide a brief background on reversible logic. We present PSDRM expression and synthesis of reversible circuit using PSDRM expression in Section III. In Section IV, we introduce our direct design of synchronous sequential circuit from reversible gates using PSDRM expression. Then, we show specific designs of synchronous counters and registers in Sections V and VI, respectively. Finally, we conclude this paper in Section VII.

II. Background on Reversible Logic

A reversible gate (or a circuit) maps every input combination to a unique output combination. This unique mapping implies that a reversible circuit has the same number of inputs and outputs. A reversible circuit with n inputs/outputs is called an $n\times n$ reversible circuit. A reversible circuit is constructed as a network of reversible gates. Fig. 1 shows the commonly used reversible gates such as 1×1 NOT gate, 2×2 Feynman gate, 3×3 Toffoli gate, and 3×3 Fredkin gate. Toffoli gate may have more than three inputs/outputs and they are called multiple-controlled Toffoli gates.

The complexity of reversible circuit design is compared in terms of quantum cost (the number of primitive quantum gates required to realize the circuit) and the number of garbage outputs (the final outputs that are not used as the primary outputs). The 1×1 and 2×2 gates are technology realizable primitive gates and their quantum costs are assumed to be one. Thus, the quantum cost of NOT gate and Feynman gate is one each. Toffoli and Fredkin gates are macrolevel gates and need to be realized on the top of 2×2 gates. The 3×3 Toffoli gate and the Fredkin gate can be realized using five

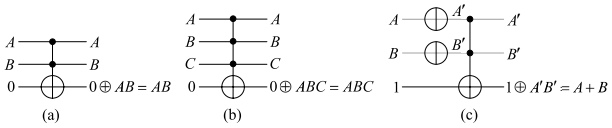

Fig. 2. Reversible realizations of classical (a) two-input AND gate, (b) three-input AND gate, and (c) two-input OR gate.

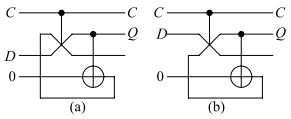

Fig. 3. Reversible realization of (a) level-triggered and (b) falling-edge triggered D flip-flops.

2×2 primitive gates [36], [37], and thus their quantum cost is five each. Realization of multiple-controlled Toffoli gates from primitive quantum gates are presented in [38], where quantum costs for up to 16×16 Toffoli gates are reported. The quantum costs for 4×4, 5×5, and 6×6 Toffoli gates are 14, 20, and 32, respectively.

Classical AND and OR gates can be realized using Toffoli gates. Reversible realization of two- and three-input AND gates are shown in Fig. 2(a) and (b), respectively. Reversible realization of two-input AND gate requires five quantum cost and two garbage outputs and that of three-input AND gate requires 14 quantum cost and three garbage outputs. Reversible realization of two-input OR gate is shown in Fig. 2(c), which requires seven quantum cost and two garbage outputs.

Reversible realizations of level-triggered and falling-edge triggered D flip-flops are shown in Fig. 3(a) and (b), respectively. In Fig. 3(a), the state output is copied using a Feynman gate and fed back to the second input of the Fredkin gate. When the clock C is zero, then the feedback is connected to the state output maintaining the state output unchanged. When C becomes one, then the D input is connected to the state output performing the level-triggered load operation. This realization requires six quantum costs and two garbage output. In Fig. 3(b), the feedback is connected to the third input of the Fredkin gate. When C is one, then the feedback is connected to the state output maintaining the state output unchanged. When C becomes zero, then the D input is connected to the state output performing the falling-edge triggered load operation. This realization requires six quantum costs and two garbage output.

III. Reversible Logic Synthesis Using PSDRM Expressions

An n-variable Boolean function $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$ can be expanded on the variable x_i using any of the following expansions [39]:

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = f_0 \oplus x_i f_2 \quad \text{(positive Davio, pD)} \tag{1}$$

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = f_1 \oplus x_i' f_2 \quad \text{(negative Davio, nD)} \tag{2}$$

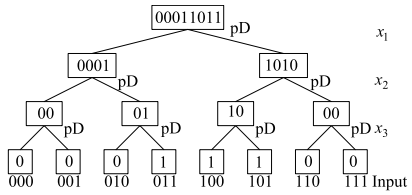

Fig. 4. Application of pD expansion on all variables of the function of (4).

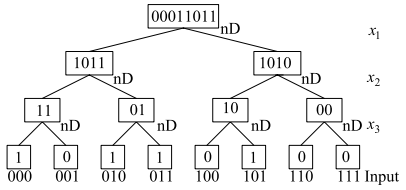

Fig. 5. Application of nD expansion on all variables of the function of (4).

where

$$f_0 = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$
$$f_1 = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

and

$$f_2 = f_0 \oplus f_1.$$

If we apply pD expansion on all variables of an n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$, then the resulting expression can be represented as [40]

$$f(x_1, x_2, \ldots, x_n) = f_{00\cdots00} \oplus f_{00\cdots01}x_n \oplus f_{00\cdots10}x_{n-1} \oplus f_{00\cdots11}x_{n-1}x_n \oplus \cdots \oplus f_{11\cdots11}x_1x_2\cdots x_{n-1}x_n \quad (3)$$

where the coefficients are $(\forall i \in \{0,1\}^n) f_i \in \{0,1\}$. If a subscript of a coefficient is one, only then the corresponding variable appears in the uncomplemented form in the associated product term. If a coefficient is one, only then the associated product term appears in the expression. The coefficient vector of the expression of (3) for a given n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$ can be computed directly from the output vector of the given Boolean function, as shown in the tree of Fig. 4 for a three-variable function

$$f(x_1, x_2, x_3) = \sum(3, 4, 6, 7). \quad (4)$$

The output vector of the function of (4) is 00011011. If we apply pD expansion on the variable x_1, then $f_0 = 0001$, $f_1 = 1011$, and $f_2 = 1010$. Now, f_0 goes to the left child of the root and f_2 goes to the right child of the root of the tree of Fig. 4. Similarly, the pD expansion is applied on the other internal nodes. The leaves represent the coefficient vector of the expression of (3). The resulting expression is determined from the ones of the coefficient vector and their corresponding input combinations. The resulting expression of the tree of Fig. 4 is

$$f(x_1, x_2, x_3) = x_2x_3 \oplus x_1 \oplus x_1x_3. \quad (5)$$

If we apply nD expansion on all variables of an n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$, then the resulting expression can be represented as [41]

$$f(x_1, x_2, \ldots, x_n) = f_{00\cdots00} \oplus f_{00\cdots01}x'_n \oplus f_{00\cdots10}x'_{n-1} \oplus f_{00\cdots11}x'_{n-1}x'_n \oplus \cdots \oplus f_{11\cdots11}x'_1x'_2\cdots x'_{n-1}x'_n. \quad (6)$$

The expression (6) is similar to (3) with the exception that variables appear in the complemented form. The computation of the coefficient vector of the expression of (6) for the function of (4) is shown in the tree of Fig. 5. As we apply nD expansion on the variable x_1, $f_1 = 1011$ goes to the left child of the root and $f_2 = 1010$ goes to the right child of the root of the tree of Fig. 5. Similarly, the nD expansion is applied on the other internal nodes. Determination of the resulting expression from the tree of Fig. 5 is similar to that from the tree of Fig. 4. The resulting expression of the tree of Fig. 5 is

$$f(x_1, x_2, x_3) = 1 \oplus x'_2 \oplus x'_2x'_3 \oplus x'_1x'_3. \quad (7)$$

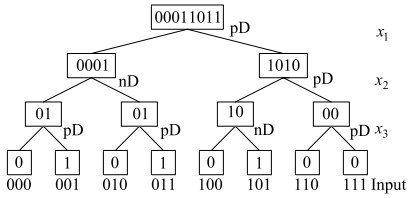

Fig. 6. Arbitrary PSDRM tree for the function of (4).

The trees of Figs. 4 and 5 have $2^n - 1$ internal nodes for an n-variable function. If we independently choose any of the pD or nD expansion for each of the internal nodes, then the resulting expression is called PSDRM expression [39]. There are 2^{2^n-1} PSDRM expressions for an n-variable function [39] and the expression with the minimum number of products is the minimum PSDRM expression. Exhaustive minimization of PSDRM expression is not possible and we need some sort of heuristics for this. In this paper, we develop our own heuristics tailored toward designing synchronous sequential circuit in Section IV. Before that, we explain here determination of PSDRM expression for a given set of expansions for the internal nodes. We show an arbitrary PSDRM tree for the function of (4) in Fig. 6. Determination of the PSDRM expression from the PSDRM tree is similar to that from Figs. 4 and 5. However, in this case, the expansion for each variable is separately considered by traversing the path from the root to a leaf with coefficient one. The resulting PSDRM expression from the tree of Fig. 6 is

$$f(x_1, x_2, x_3) = x_3 \oplus x'_2x_3 \oplus x_1x'_3. \quad (8)$$

The PSDRM expression of (8) can be realized using reversible gates, as shown in Fig. 7, which is self-explanatory. The circuit of Fig. 7 requires two NOT gates, one Feynman gate, and two 3×3 Toffoli gates. Therefore, its quantum cost is $2 \times 1 + 1 \times 1 + 2 \times 5 = 13$. The circuit of Fig. 7 has one primary output and three unused outputs. Therefore, it has three garbage outputs.

IV. Design of Synchronous Sequential Circuit Using PSDRM Expression

Design of synchronous sequential circuit involves design of next state logic and output logic. In this paper, we do not

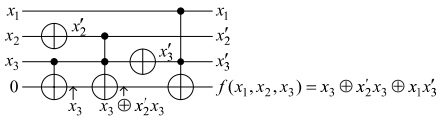

Fig. 7. Reversible realization of PSDRM expression of (8).

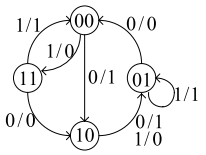

Fig. 8. State transition diagram of an arbitrary sequential circuit.

TABLE I
TRANSITION TABLE OF THE SEQUENTIAL CIRCUIT OF FIG. 8

$CxQ1Q0$	$Q1^+Q0^+$	$CxQ1Q0$	$Q1^+Q0^+$
0000	00	1000	10
0001	01	1001	00
0010	10	1010	01
0011	11	1011	10
0100	00	1100	11
0101	01	1101	01
0110	10	1110	01
0111	11	1111	00

use any flip-flop to store the present state; rather we take the feedback directly from the present state output as the input to the next state logic. This special design approach needs special method of designing the next state logic discussed in the following.

For designing the next state logic of a level-triggered sequential circuit, we construct transition table considering the clock (designated C), the present states (designated Q), and the inputs (if any) as the inputs and the next states (designated Q^+) as the outputs. State transition diagram of an arbitrary sequential circuit with two-bit states $Q1Q0$, one input x, and one output z is shown in Fig. 8 and the corresponding transition table is shown in Table I. Determination of the minimized PSDRM expression from the output vector of the next state $Q1^+$ from Table I is shown in the PSDRM tree of Fig. 9. Observation of the transition table of Table I shows that for $C = 0$, the next state relationship is simply $Q^+ = Q$ to provide the feedback needed for maintaining the state unchanged. In the PSDRM expression, this relationship can be maintained only if pD expansion is applied at the root and its left descendents, as shown at the left side of the cut shown in Fig. 9. For heuristic minimization of the PSDRM expression for the next state, at the right descendents of the root, we apply either pD or nD expansion, which produce the minimum number of ones in the next level, as shown at the right side of the cut of Fig. 9. For example, the subvector for the right child of the root of Fig. 9 is 1010 1011. The pD expansion produces two subvectors 1010 and 0001 in the next level, which have three ones. However, the nD expansion produces subvectors 1011 and 0001 in the next level, which have four ones. Therefore, we choose pD expansion

Algorithm 1 Algorithm for Determining Next State Expressions

1) Let the sequential circuit has m inputs and n-bit states. Construct a $(1 + m + n)$-input and n-output truth table representing the transition table of the sequential circuit considering the clock, the inputs, and the present states as inputs and the next states as outputs.
2) From the output vector of each of the next states, construct PSDRM tree using steps 3 and 4.
3) At the root and its left descendents, apply pD expansion.
4) At the right descendents of the root, apply either pD or nD expansion that produces the minimum number of ones at the next level of the tree. Break the tie by choosing pD expansion.
5) Determine PSDRM expressions for the next states from the constructed PSDRM trees.

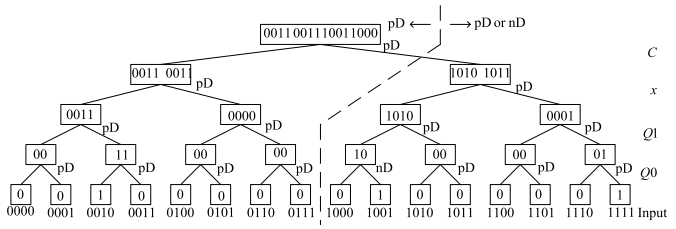

Fig. 9. Determination of the PSDRM expression for the next state $Q1^+$ of the transition table of Table I.

TABLE II
TRUTH TABLE OF OUTPUT z OF THE SEQUENTIAL CIRCUIT OF FIG. 8

$xQ1Q0$	z	$xQ1Q0$	z
000	1	100	0
001	0	101	1
010	1	110	0
011	0	111	1

for this node. The tie is broken by choosing pD expansion over nD expansion. This heuristic produces local minimum at every internal node with the hope to produce overall global minimum. Using this minimization technique, we constructed the PSDRM tree of Fig. 9. The PSDRM expression for the next state $Q1^+$ is determined from the PSDRM tree of Fig. 9, as shown in (9). The PSDRM expression for the next state $Q0^+$ can be determined in the similar manner and is determined, as shown in (10). The generic algorithm for determining next state expressions is given in Algorithm 1

$$Q1^+ = Q1 \oplus CQ0' \oplus CxQ1Q0 \tag{9}$$
$$Q0^+ = Q0 \oplus CQ0 \oplus CQ1Q0' \oplus CxQ1'. \tag{10}$$

For simplicity, we use minimized PSDRM expressions for synthesizing the output functions, which may not be the minimum. The truth table for the output function of the sequential circuit of Fig. 8 is shown Table II. In this case, we use the heuristic algorithm shown in Algorithm 2 for minimizing the PSDRM expression. The determined PSDRM

Algorithm 2 Algorithm for Determining the Output Expression

1) Let the sequential circuit has m inputs, n-bit states, and y outputs. Construct a $(m+n)$-input and y-output truth table representing the output functions of the sequential circuit considering the inputs and the present states as inputs and the y outputs as outputs.
2) From the output vector of each of the output functions, construct PSDRM tree using step 3.
3) At all nodes, choose pD or nD expansion that produces the minimum number of ones at the next level. Break the tie by choosing pD expansion.
4) Determine PSDRM expressions for the outputs from the constructed PSDRM trees.

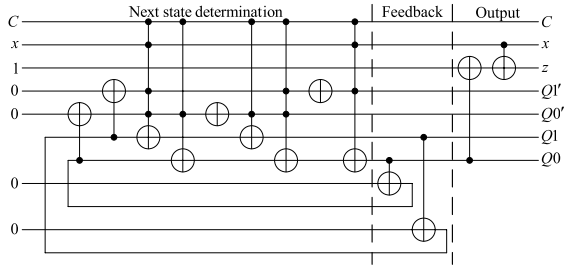

Fig. 10. Reversible realization of the level-triggered sequential circuit of Fig. 8 using PSDRM expressions of (9)–(11).

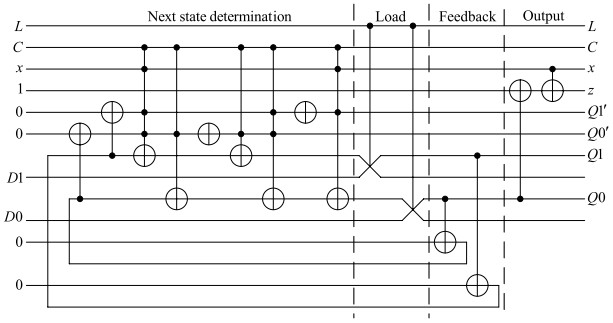

Fig. 11. Reversible realization of the synchronous sequential circuit of Fig. 8 with asynchronous parallel load facility.

expression for the output z is shown as

$$z = Q0' \oplus x. \tag{11}$$

The next state expressions of (9) and (10) are realized in the next state determination part of the circuit of Fig. 10. To provide feedback from the state outputs, the state outputs are copied using Feynman gates in the feedback part. The output expression of (11) is realized in the output part. The realized sequential circuit is a level-triggered circuit. This realization needs one 5×5 Toffoli gate, two 4×4 Toffoli gates, two 3×3 Toffoli gates, six Feynman gates, and two NOT gates. Therefore, its quantum cost is 66. The circuit has two garbage outputs.

Asynchronous parallel load facility may be incorporated in the sequential circuit of Fig. 10 by adding Fredkin gate in between the next state determination part and the feedback part of the circuit, as shown in Fig. 11. When $L = 0$, independent

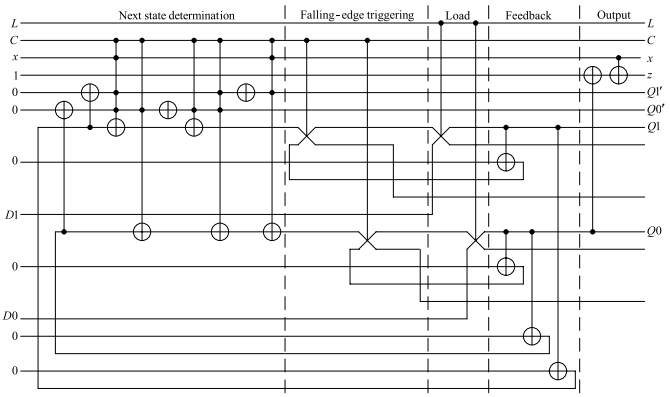

Fig. 12. Reversible realization of the synchronous sequential circuit of Fig. 8 with falling-edge triggering and asynchronous parallel load facility.

of the value of C, the state value available at the output of the next state determination part will pass to the state output. When $L = 1$, irrespective of the value of C, the data input D will be copied to the state output. However, when $C = 1$, the loaded value of the next state will be fed back and used in the next state determination circuit. It may happen that the next state determination may not be complete within the remaining part of the clock pulse making the next state ambiguous. Thus, load should be done asynchronously when $C = 0$. When $C = 0$, if the L input is changed back to zero after the parallel load is done, the loaded state will remain unchanged at the state output, since loaded state will be fed back through the next state determination circuit to the state output. The circuit of Fig. 11 has two more Fredkin gates than that of Fig. 10. Thus, the quantum cost of the circuit of Fig. 11 is 76. The circuit of Fig. 11 has five garbage outputs.

The level-triggered sequential circuit with parallel load facility of Fig. 11 can be made falling-edge triggered by adding falling-edge triggered D flip-flop of Fig. 3(b) in between the next state determination and the load parts of the circuit, as shown in Fig. 12, but taking the feedback from the state output of the feedback part. When $C = 0$, the next state determination part will simply pass the state feedback to its output and this output will then be passed to the state output through the D input of the D flip-flop of the falling-edge triggering part and the Fredkin gate of the load part (provided $L = 0$) to maintain the state output unchanged. When $C = 1$, the next state determination part will compute the next state based on the present state feedback to it and the external input, but the D flip-flop of the falling-edge triggering part will provide the feedback of the state output to maintain the state output unchanged. Now, when the C input goes back to zero after the next state determination is complete, the computed next state will be passed to the state output through the D input of the D flip-flop of the falling-edge triggering part and the Fredkin gate of the load part making the circuit falling-edge triggered. The circuit of the Fig. 12 has two more Fredkin gates and two more Feynman gates than that of Fig. 11. Thus, its quantum cost is 88. The circuit of Fig. 12 has seven garbage outputs.

The circuit of Fig. 12 is a general sequential circuit having input, output, and state changes. We have simulated the circuit

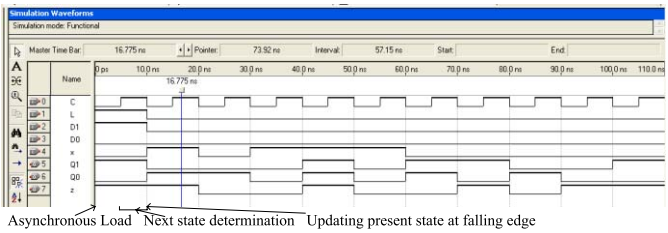

Fig. 13. Verilog HDL simulation of the sequential circuit of Fig. 12.

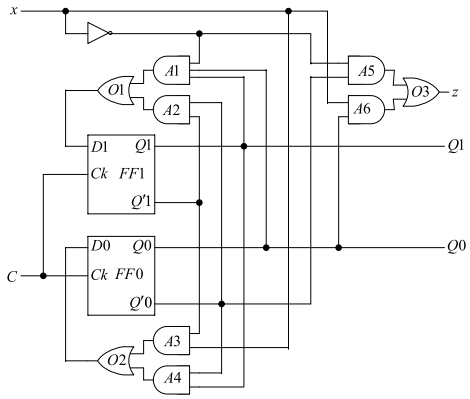

Fig. 14. Classical design of the synchronous sequential circuit of Fig. 8.

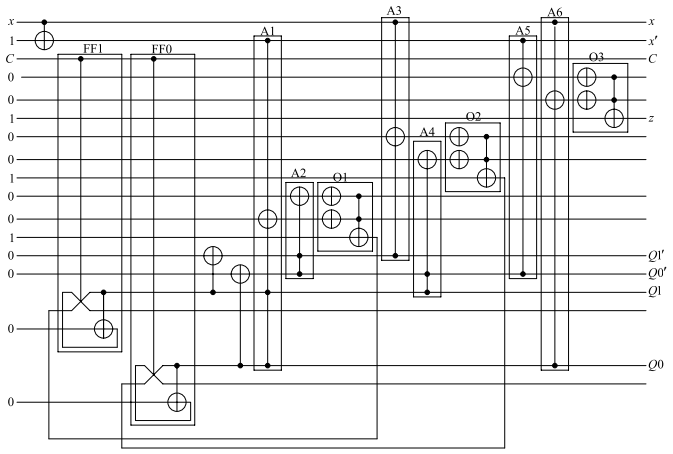

Fig. 15. Reversible realization of sequential circuit of Fig. 14 using replacement technique.

TABLE III
COMPARISON OF REVERSIBLE REALIZATIONS OF THE SEQUENTIAL CIRCUIT OF FIG. 8 USING THE DIRECT DESIGN METHOD AND THE REPLACEMENT DESIGN

	Direct design	Replacement design	% improvement over replacement design
Quantum cost	66	74	10.81
Garbage outputs	2	11	81.82

using Verilog HDL to test the correctness of the design. The simulation output is shown in Fig. 13. From Fig. 13, we see that when $C = 0$ and $L = 1$, the load data 10 is loaded to the state output. After that, when C becomes one, then the next state determination part determines the next state, but at the state output, the loaded output remains unchanged. As the present state is 10 and the input is zero, the next state will be 11. After the next state calculation is completed, when C goes back to zero, then the computed next state 11 is passed to the state output. From Fig. 13, it can be verified that state transitions and outputs are same, as shown in Fig. 8.

Any level-triggered sequential circuit may be provided with asynchronous parallel load and/or made falling-edge triggered using the techniques of Figs. 11 and 12. Therefore, complexity comparison of level triggered sequential circuit is sufficient.

For complexity comparison, we have designed the synchronous sequential circuit of Fig. 8 using classical technique using D flip-flop and the resulting circuit is shown in Fig. 14. The reversible circuit corresponding to the classical circuit of Fig. 14 is shown in Fig. 15. The level-triggered D flip-flops (FF0 and FF1) are replaced by their reversible counterparts, as shown in Fig. 3(a). The AND (A1–A6) and the OR gates (O1–O3) are replaced by their reversible counterparts, as shown in Fig. 2. The circuit of Fig. 15 needs five NOT gates, five Feynman gates, two Fredkin gates, one 4×4 Toffoli gates, and eight 3×3 Toffoli gates. Thus, its quantum cost is 74. The circuit has 11 garbage outputs. The complexity comparison of the direct design of Fig. 10 and the replacement design of Fig. 15 is shown in Table III, which shows that the direct design requires 10.81% less quantum cost and 81.82% less garbage outputs than the replacement design.

V. DESIGN OF COUNTERS

In this section, we discuss designs of practically important synchronous counters using the direct design technique presented in Section IV.

We have determined the PSDRM expressions for the next states of four-bit up counter as follows:

$$Q3^+ = Q3 \oplus CQ2Q1Q0 \tag{12}$$
$$Q2^+ = Q2 \oplus CQ1Q0 \tag{13}$$
$$Q1^+ = Q1 \oplus CQ0 \tag{14}$$
$$Q0^+ = Q0 \oplus C. \tag{15}$$

Reversible realization of four-bit level-triggered up counter using the PSDRM expressions of (12)–(15) is shown in Fig. 16. The circuit of Fig. 16 requires one 5×5 Toffoli gate, one 4×4 Toffoli gate, one 3×3 Toffoli gate, and five Feynman gates. Thus, its quantum cost is 44. The circuit has one garbage output. We can provide this counter with asynchronous parallel load using the technique of Fig. 11, which will require 64 quantum cost and six garbage outputs. Reference [29] presented reversible design of four-bit level-triggered up counter with synchronous parallel load facility using the replacement technique, which requires 65 quantum cost and 18 garbage outputs. Comparison of our direct design with the replacement design of [29] is shown in Table IV, which shows that our direct design requires 1.54% less quantum cost and 66.67% less garbage outputs than the replacement design of [29]. The level-triggered counter of Fig. 16 can also be made falling-edge triggered using

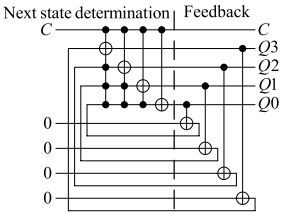

Fig. 16. Reversible realization of four-bit level-triggered up counter using the PSDRM expressions of (12)–(15).

TABLE IV
COMPARISON OF DIRECT DESIGN OF FOUR-BIT LEVEL-TRIGGERED UP COUNTER WITH PARALLEL LOAD WITH THAT OF THE REPLACEMENT DESIGN OF [29]

	Direct design	Design of [29]	% improvement over [29]
Quantum cost	64	65	1.54
Garbage outputs	6	18	66.67

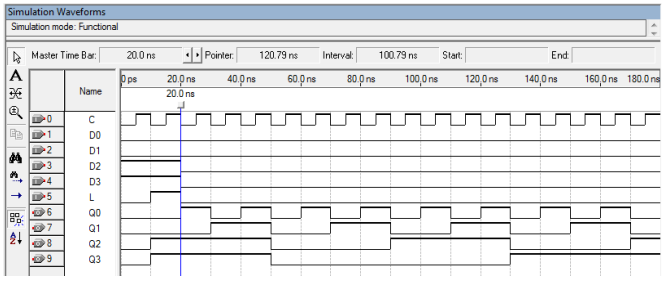

Fig. 17. Verilog HDL simulation of reversible four-bit falling-edge triggered up counter with asynchronous load facility.

the technique of Fig. 12, which will require 68 quantum cost and five garbage outputs. The four-bit falling-edge triggered up counter with asynchronous parallel load will require 88 quantum cost and 10 garbage outputs. Though the circuit of four-bit falling-edge triggered up counter with asynchronous parallel load is not shown here, it is simulated using Verilog HDL and the simulation result is shown in Fig. 17. From the simulation result, it is found that the parallel data 1100 is loaded correctly and the counter counts in the correct sequence.

We have determined the PSDRM expressions for the next states of four-bit down counter as follows:

$$Q3^+ = Q3 \oplus C\,Q2'Q1'Q0' \quad (16)$$
$$Q2^+ = Q2 \oplus C\,Q1'Q0' \quad (17)$$
$$Q1^+ = Q1 \oplus C\,Q0' \quad (18)$$
$$Q0^+ = Q0 \oplus C. \quad (19)$$

The PSDRM expressions of (16)–(19) can be realized as done in Figs. 10–12. The four-bit level-triggered down counter will require 44 quantum cost and one garbage output. The level-triggered down counter with asynchronous parallel load will require 64 quantum cost and six garbage outputs. The falling-edge triggered down counter will require 68 quantum cost and five garbage outputs. The falling-edge triggered down counter with asynchronous parallel load will require 88 quantum cost and 10 garbage outputs.

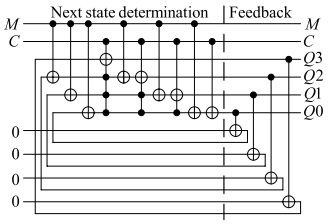

Fig. 18. Reversible realization of four-bit level-triggered up/down counter using the expressions of (20)–(23).

The PSDRM expressions of (12)–(15) for four-bit up counter and the PSDRM expressions of (16)–(19) for four-bit down counter can be combined together using a mode selector input M to construct four-bit up/down counter, as shown in (20)–(23). When $M = 0$, (20)–(23) become those of (12)–(15) implementing the up counter. When $M = 1$, (20)–(23) become those of (16)–(19) implementing the down counter

$$Q3^+ = Q3 \oplus C\,(M \oplus Q2)\,(M \oplus Q1)\,(M \oplus Q0) \quad (20)$$
$$Q2^+ = Q2 \oplus C\,(M \oplus Q1)\,(M \oplus Q0) \quad (21)$$
$$Q1^+ = Q1 \oplus C\,(M \oplus Q0) \quad (22)$$
$$Q0^+ = Q0 \oplus C. \quad (23)$$

Reversible realization of four-bit level-triggered up/down counter using the expressions of (20)–(23) is shown in Fig. 18, which requires one 5 × 5 Toffoli gate, one 4 × 4 Toffoli gate, one 3 × 3 Toffoli gate, and 11 Feynman gates. Thus, its quantum cost is 50. The circuit has two garbage outputs. The level-triggered up/down counter with asynchronous parallel load will require 70 quantum cost and seven garbage outputs. The falling-edge triggered up/down counter will require 74 quantum cost and six garbage outputs. The falling-edge triggered up/down counter with asynchronous parallel load will require 94 quantum cost and 11 garbage outputs.

One very important thing to note that the expressions of (20)–(23) can be made generalized as shown in (24) and (25) for designing n-bit up/down counter

$$Qi^+ = Qi \oplus C(M \oplus Q(i-1))(M \oplus Q(i-2)) \cdots (M \oplus Q0);\ i > 0 \quad (24)$$
$$Q0^+ = Q0 \oplus C. \quad (25)$$

VI. DESIGN OF REGISTERS

In this section, we discuss designs of practically important synchronous registers using the direct design technique presented in Section IV.

We have determined the PSDRM expressions for the next states of four-bit serial-in serial-out (SISO) right-shift register, as shown in (26)–(29), where D_R is the data input for right shift. Design of four-bit level-triggered SISO right-shift register by realizing the expressions of (26)–(29) will require

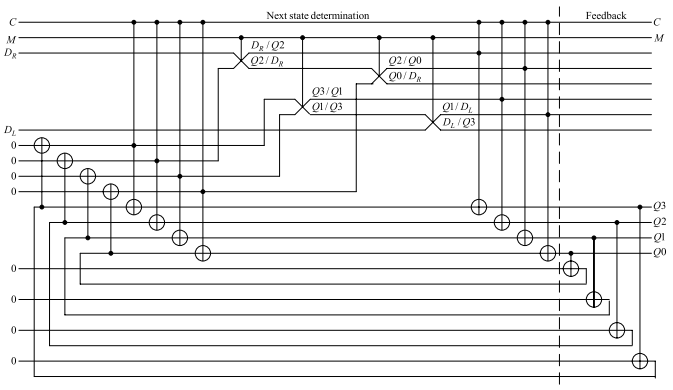

Fig. 19. Realization of four-bit level-triggered SISO right/left shift register using the expressions of (34)–(37).

48 quantum cost and six garbage outputs

$$Q3^+ = Q3 \oplus CQ3 \oplus CD_R \quad (26)$$
$$Q2^+ = Q2 \oplus CQ2 \oplus CQ3 \quad (27)$$
$$Q1^+ = Q1 \oplus CQ1 \oplus CQ2 \quad (28)$$
$$Q0^+ = Q0 \oplus CQ0 \oplus CQ1. \quad (29)$$

We have determined the PSDRM expressions for the next states of four-bit SISO left-shift register, as shown in (30)–(33), where D_L is the data input for left shift. Design of four-bit level-triggered SISO left-shift register by realizing the expressions of (30)–(33) will require 48 quantum cost and six garbage outputs

$$Q3^+ = Q3 \oplus CQ3 \oplus CQ2 \quad (30)$$
$$Q2^+ = Q2 \oplus CQ2 \oplus CQ1 \quad (31)$$
$$Q1^+ = Q1 \oplus CQ1 \oplus CQ0 \quad (32)$$
$$Q0^+ = Q0 \oplus CQ0 \oplus CD_L. \quad (33)$$

The expressions of (26)–(29) and those of (30)–(33) can be combined together using a mode selection input M, as shown in (34)–(37) for designing four-bit level-triggered SISO right/left shift register. The mode selection input M provides a multiplexing operation in the right-most parts of (34)–(37). When $M = 0$, (34)–(37) reduce to those of (26)–(29) realizing SISO right-shift register. When $M = 1$, (34)–(37) reduce to those of (30)–(33) realizing SISO left-shift register

$$Q3^+ = Q3 \oplus CQ3 \oplus C\left(M'D_R \oplus MQ2\right) \quad (34)$$
$$Q2^+ = Q2 \oplus CQ2 \oplus C\left(M'Q3 \oplus MQ1\right) \quad (35)$$
$$Q1^+ = Q1 \oplus CQ1 \oplus C\left(M'Q2 \oplus MQ0\right) \quad (36)$$
$$Q0^+ = Q0 \oplus CQ0 \oplus C\left(M'Q1 \oplus MD_L\right). \quad (37)$$

Realization of four-bit level-triggered SISO right/left shift register using the expressions of (34)–(37) is shown in Fig. 19. The multiplexing operation of the right-most parts of (34)–(37) can be implemented using Fredkin gates, as shown in Fig. 19. For ready understanding of the multiplexing operation, we explicitly write outputs of each Fredkin gate for $M = 0$ and 1 separated by slash (/). The circuit of Fig. 19 requires eight 3×3 Toffoli gates, four Fredkin gates, and eight Feynman gates. Thus, its quantum cost is 68. The circuit has eight garbage outputs. The level-triggered SISO right/left shift register of Fig. 19 can be made universal (that is, having both serial and parallel input and both serial and parallel output with right/left shift) by adding asynchronous parallel load using the technique of Fig. 11. The four-bit level-triggered universal register will require 88 quantum cost and 13 garbage outputs. The level-triggered SISO right/left shift register of Fig. 19 can be made falling-edge triggered SISO right/left shift register using the technique of Fig. 12. The four-bit falling-edge triggered SISO right/left shift register will require 92 quantum cost and 12 garbage outputs. The register of Fig. 19 can be made falling-edge triggered universal register using the techniques of Figs. 11 and 12. The four-bit falling-edge triggered universal register will require 112 quantum cost and 17 garbage outputs. Reference [28] presented design of a four-bit falling-edge triggered universal register using the replacement technique, which requires 220 quantum cost and 35 garbage outputs. The comparison of our direct design of four-bit falling-edge triggered universal register with that of replacement design of [28] is given in Table V. From Table V, we see that our direct design requires 49.09% less quantum cost and 51.43% less garbage outputs than the replacement design of [28].

TABLE V
COMPARISON OF DIRECT DESIGN OF FOUR-BIT FALLING-EDGE TRIGGERED UNIVERSAL REGISTER WITH THAT OF THE REPLACEMENT DESIGN OF [28]

	Direct design	Design of [28]	% improvement over [28]
Quantum cost	112	220	49.09
Garbage outputs	17	35	51.43

One very important thing to note that the expressions of (34)–(37) can be made generalized, as shown in (38)–(40) for designing n-bit SISO right/left shift register

$$Q(n-1)^+ = Q(n-1) \oplus CQ(n-1) \oplus C\left(M'D_R \oplus MQ(n-2)\right) \quad (38)$$
$$Qi^+ = Qi \oplus CQi \oplus C\left(M'Q(i+1) \oplus MQ(i-1)\right); \quad 1 \le i \le (n-2) \quad (39)$$
$$Q0^+ = Q0 \oplus CQ0 \oplus C\left(M'Q1 \oplus MD_L\right). \quad (40)$$

VII. CONCLUSION

Reversible logic has shown a good promise for low-power design using emerging computing technologies. A good number of design methods for reversible combinational circuits have been proposed [13]–[22]. However, only a very limited works have been reported on reversible sequential circuit design [24]–[30]. In this paper, we present a novel approach of direct design of sequential circuit with reversible gates using PSDRM expressions describing the state transitions and output functions of the circuit. Design examples show that our direct designs save 1.54%–49.09% quantum cost and 51.43%–81.82% garbage outputs than the replacement design approach suggested earlier. Thus, our proposed direct

design method outperforms the previously reported replacement design approach.

REFERENCES

[1] R. Landauer, "Irreversibility and heat generation in the computation process," *IBM J. Res. Develop.*, vol. 44, pp. 183–191, Jan. 2000.

[2] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to binary logic switch scaling—A Gedanken model," *Proc. IEEE*, vol. 91, no, 11, pp. 1934–1939, Nov. 2003.

[3] C. Bennett, "Logical reversibility of computations," *IBM J. Res. Develop.*, vol. 17, no. 6, pp. 525–532, 1973.

[4] A. D. Vos and Y. V. Rentergem, "Power consumption in reversible logic addressed by a ramp voltage," in *Proc. 15th Int. Workshop Power Timing Model., Optim. Simul.*, LNCS 3728. 2005, pp. 207–216.

[5] J. Ren and V. K. Semenov, "Progress with physically and logically reversible superconducting digital circuits," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 780–786, Jun. 2011.

[6] J. Ren, V. K. Semenov, Y. A. Polyakov, D. V. Averin, and J.-S. Tsai, "Progress towards reversible computing with nsquid arrays," *IEEE Trans. Appl. Supercond.*, vol. 19, no. 3, pp. 961–967, Jun. 2009.

[7] N. Kostinski, M. P. Fok, and P. R. Prucnal, "Experimental demonstration of an all-optical fiber-based Fredkin gate," *Opt. Lett.*, vol. 34, no. 18, pp. 2766–2768, 2009.

[8] C. Taraphdara, T. Chattopadhyay, and J. Roy, "Machzehnder interferometer-based all-optical reversible logic gate," *Opt. Laser Technol.*, vol. 42, no. 2, pp. 249–259, 2010.

[9] X. Ma, J. Huang, C. Metra, and F. Lombardi, "Reversible gates and testability of one dimensional arrays of molecular QCA," *J. Elect. Testing*, vol. 24, nos. 1–3, pp. 1244–1245, 2008.

[10] X. Ma, J. Huang, C. Metra, and F. Lombardi, "Detecting multiple faults in one-dimensional arrays of reversible QCA gates," *J. Electron. Test.*, vol. 25, no. 1, pp. 39–54, 2009.

[11] S. Bandyopadhyay, "Nanoelectric implementation of reversible and quantum logic," *Supperlattices Microstruct.*, vol. 23, nos. 3–4, pp. 445–464, 1998.

[12] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[13] V. V. Shende, A. Prasad, I. Markov, and J. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 710–722, Jun. 2003.

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

[15] K. N. Patel, J. P. Hayes, and I. L. Markov, "Fault testing for reversible circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 8, pp. 410–416, Aug. 2004.

[16] 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.

[17] V. V. Shende, I. L. Markov, and S. S. Bullock, "Synthesis of quantum-logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1000–1010, Jun. 2006.

[18] P. Gupta, A. Agarwal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 11, pp. 2317–2330, Nov. 2006.

[19] A. K. Prasad, V. Shende, I. Markov, J. Hayes, and K. N. Patel, "Data structures and algorithms for simplifying reversible circuits," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 2, no. 4, pp. 277–293, 2006.

[20] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 3, pp. 436–444, Mar. 2008.

[21] D. Große, R. Wille, G. Dueck, and R. Drechsler, "Exact multiple control Toffoli network synthesis with SAT techniques," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 5, pp. 703–715, May 2009.

[22] S. Mahammad and K. Veezhinathan, "Constructing online testable circuits using reversible logic," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 1, pp. 101–109, Jan. 2010.

[23] T. Toffoli, "Reversible computing," MIT Lab. Comput. Sci., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TM-151, 1980.

[24] J. E. Rice, "A new look at reversible memory elements," in *Proc. IEEE ISCAS*, May 2006, pp. 243–246.

[25] S. K. S. Hari, S. Shroff, S. N. Mohammad, and V. Kamakoti, "Efficient building blocks for reversible sequential circuit design," in *Proc. 49th IEEE MWSCAS*, Aug. 2006, pp. 437–441.

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

[27] M.-L. Chuang and C.-Y. Wang, "Synthesis of reversible sequential elements," *ACM J. Emerg. Technol.*, vol. 3, no. 4, pp. 1–19, 2008.

[28] H. Thapliyal and N. Ranganathan, "Design of reversible sequential circuits optimizing quantum cost, delay and garbage outputs," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 6, no. 4, pp. 14:1–14:35, 2008.

[29] M. Haghparast and M. S. Gharajeh, "Design of a nanometric reversible 4-bit binary counter with parallel load," *Austral. J. Basic Appl. Sci.*, vol. 5, no. 7, pp. 63–71, 2011.

[30] M. H. A. Khan and M. Perkowski, "Synthesis of reversible synchronous counters," in *Proc. 41st IEEE ISMVL*, May 2011, pp. 242–247.

[31] J. Hu, G. Ma, and G. Feng, "Efficient algorithm for positive-polarity Reed-Muller expansions of reversible circuits," in *Proc. ICM*, Dec. 2006, pp. 63–66.

[32] K. Takahashi and T. Hirayama, "Reversible logic synthesis from positive Davio trees of logic functions," in *Proc. IEEE Region Conf. TENCON*, Jan. 2009, pp. 1–4.

[33] Y. Pang, S. Wang, Z. He, J. Lin, S. Sultana, and K. Radecka, "Positive Davio-based synthesis algorithm for reversible logic," in *Proc. 29th ICCD*, Oct. 2011, pp. 212–218.

[34] Z. Jing and J. C. Muzio, "Improved implementation of a Reed-Muller spectra based reversible synthesis algorithm," in *Proc. IEEE Pac Rim Conf. Commun., Comput. Signal Process.*, Aug. 2007, pp. 202–205.

[35] M. Soeken, R. Wille, and R. Drechsler, "Hierarchical synthesis of reversible circuits using positive and negative Davio decomposition," in *Proc. 5th IDT*, Dec. 2010, pp. 143–148.

[36] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, *et al.*, "Elementary gates for quantum computation," *Phy. Rev. A*, vol. 52, no. 5, pp. 3457–3467, 1995.

[37] J. A. Smolin and D. P. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate," *Phys. Rev. A*, vol. 53, no. 4, pp. 2855–2856, 1996.

[38] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-controlled Toffoli gates," in *Proc. 41st IEEE ISMVL*, May 2011, pp. 288–293.

[39] T. Sasao, "AND-EXOR expressions and their optimization," in *Logic Synthesis and Optimization*, T. Sasao, Ed. Norwell, MA, USA: Kluwer, 1993, pp. 287–312.

[40] M. M. H. A. Khan and M. S. Alam, "Algorithms for conversion of minterms to positive polarity Reed-Muller coefficients and vice versa," *Inf. Process. Lett.*, vol. 62, no. 5, pp. 223–230, 1997.

[41] M. M. H. A. Khan and M. S. Alam, "Mapping of fixed polarity Reed-Muller coefficients from minterms and the minimization of fixed polarity Reed-Muller expressions," *Int. J. Electron.*, vol. 83, no. 2, pp. 235–247, 1997.

Mozammel H A Khan (M'06–SM'10) was born in Kushtia, Bangladesh. He received the B.Sc., M.Sc., and Ph.D. degrees from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 1984, 1986, and 1998, respectively.

He served as a Faculty Member of the Electrical and Electronic Engineering Department, Bangladesh Institute of Technology, Rajshahi, Bangladesh, and Computer Science and Engineering Department, Khulna University, Khulna, Bangladesh. Currently, he is a Full Professor with the Computer Science and Engineering Department, East West University, Dhaka, Bangladesh. He has authored or co-authored over 80 research papers published in international and national journals and conferences. His current research interests include logic synthesis, evolutionary algorithm, and quantum computing.