An Algorithm to Reduce Quantum Cost and Garbage Outputs in Reversible Logic Circuits

Guruprasad K H
M. Tech. VLSI design and embedded systems
T.John Institute of Technology
Bangalore

Sanjay Kumar Bhagat
Asst. Professor, Dept.of E&CE.
T.John Institute of Technology
Bangalore

***Abstract*— Reversible logic has become very promising for low power design using emerging computing technologies. Reversible sequential circuits constructed by replacing the latches, flip-flops, and other combinational gates of traditional irreversible designs by their reversible counter parts leads to more Garbage outputs and Quantum cost which in turn slowdowns the circuit. Here we propose an approach of designing sequential circuits directly from reversible gates using pseudo Reed-Muller expressions representing state transition and the output functions of the circuit. This approach reduces the Quantum cost and Garbage outputs. We present designs of arbitrary as well as practically important sequential circuits such as counters and registers.**

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

I. INTRODUCTION

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. In reversible logic circuits it is a very important thing to reduce this heat dissipation. This heat can be reduced if the circuits are reversible. Generally in physically reversible circuits the heat dissipation is much more reduced. Thus it helps a lot in reducing the heat dissipation in the new upcoming technologies such as SFL technology, optical technology, quantum dot cellular automata technology, and nanotechnology and Quantum computing and quantum information. Reversible logic synthesis attempts are mostly concentrated on reversible combinational logic synthesis, but till date very few reversible sequential logic are developed. These methods 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. This method increases the Quantum cost and Garbage outputs. In this paper an attempt is made to design the reversible sequential circuits directly from reversible gates using pseudo Reed-Muller expressions.

II. BACKGROUND OF REVERSIBLE LOGIC

In case of reversible circuits number of inputs are equal to number of outputs. Here we not only get outputs from inputs, but also can recover inputs from 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.

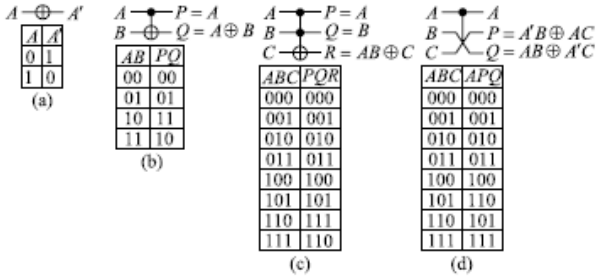

Fig.1 Commonly used reversible gates (a) NOT gate, (b) Feynman gate, (c) Toffoli gate, (d) Fredkin gate.

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 macro level 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 2×2 primitive gates, and thus their quantum cost is five each. The quantum costs for 4×4, 5×5, and 6×6 Toffoli gates are 14,20, and 32,respectively.

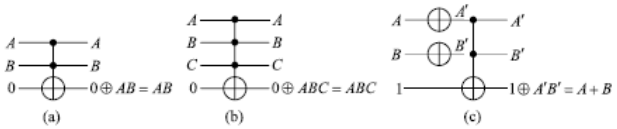

Fig.2 Reversible realizations of classical (a) two-input AND gate,(b) three input AND gate, and (c) two-input OR gate.

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.

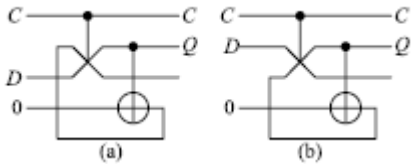

Fig.3 Reversible realization of (a) level-triggered and (b) falling edge triggered D flip-flops.

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:

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

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

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

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

Where the co-efficients are $(\forall i \in \{0,1\}^n) f_i \in \{0,1\}$.

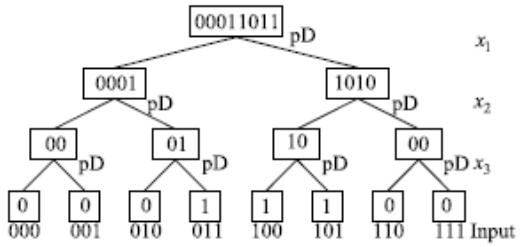

Fig.4 Application of pD expansion on all variables of equation 4.

If a subscript of a coefficient is one, only then the corresponding variable appears in the un-complemented 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

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

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

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

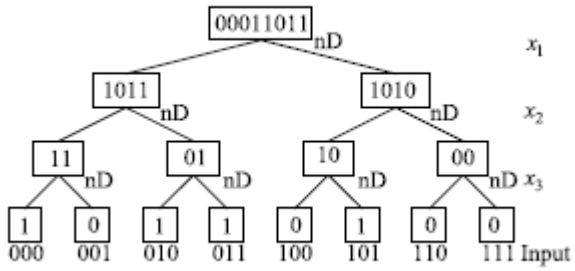

Fig.5 Application of nD expansion on all variables of equation 4.

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

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

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=1011goes 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, then D 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

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

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. There are 2^{2^n-1} PSDRM expressions for an n-variable function 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.

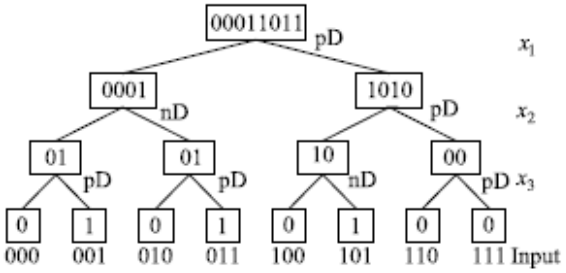

Fig.6 Arbitrary PSDRM tree for the equation 4.

The resulting PSDRM expression from the tree of Fig.6 is

$$f(x_1,x_2,x_3)= x_3\oplus x'_2\,x_3\oplus x_1x'_3. \quad (8)$$

The PSDRM expression of (8) can be realized using reversible gates, as shown in Fig.7, which is self-explanatory.

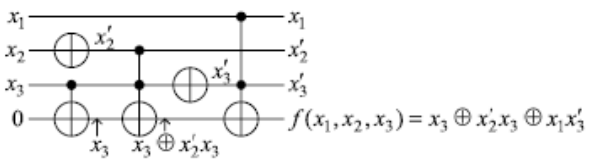

Fig.7 Reversible realization of PSDRM equation 8.

The circuit of Fig.7 requires two NOT gates, one Feynman gate, and two 3×3 Toffoli gates. Therefore, its quantum cost is 2×1+1×1+2×5=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 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

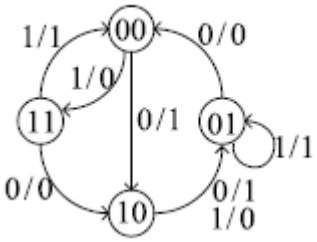

Fig.8 State transition diagram of an arbitrary sequential circuit

The corresponding transition table is shown in Table I.

$CxQ1Q0$	$Q1^+Q0^+$	$CxQ1Q0$	$Q1^+Q0^+$
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

Table I. State transition table.

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.

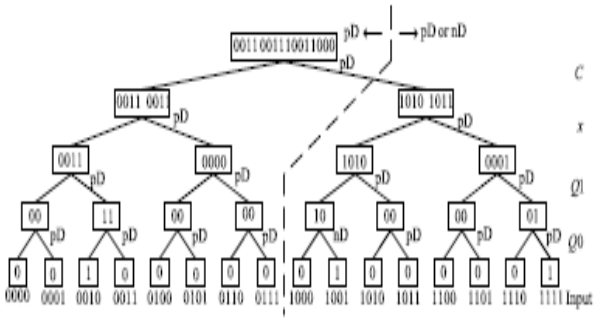

Fig.9 Determination of the PSDRM expression for the next state $Q1^+$ of the transition table.

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 sub vector for the right child of the root of Fig.9 is 10101011. The pD expansion produces two sub vectors 1010and0001 in the next level, which have three ones. However, the nD expansion produces sub vectors 1011 and 0001 in the next level, which have four ones. Therefore, we choose pD expansion 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.

$$Q_1^+ = Q_1 \oplus CQ'_0 \oplus CxQ_1Q_0 \qquad (9)$$

$$Q_0^+ = Q_0 \oplus CQ_0 \oplus CQ_1Q'_0 \oplus CxQ'_1 \qquad (10)$$

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. Construct PSDRM tree using steps 2 and 3.
2) At the root and its left descendents, apply pD expansion.
3) 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.
4) Determine PSDRM expressions for the next states from the constructed PSDRM trees.

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).

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

Table II. Truth table of output z of the sequential circuit of fig.8

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.

The generic algorithm for determining next state expressions is given in Algorithm1. 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 in Table II. In this case, we use the heuristic algorithm shown in Algorithm2 for minimizing the PSDRM expression. The determined PSDRM expression for the output z is shown as

$$z = Q'_0 \oplus x. \qquad (11)$$

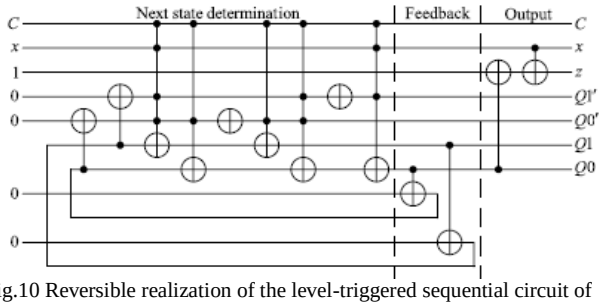

Fig.10 Reversible realization of the level-triggered sequential circuit of Fig.8 using equations of (9)-(11)

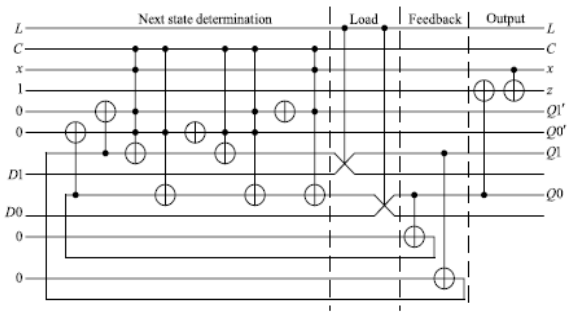

Fig.11 Reversible realization of the synchronous sequential circuit of Fig.8 with asynchronous parallel load capacity.

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 2×3 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 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.

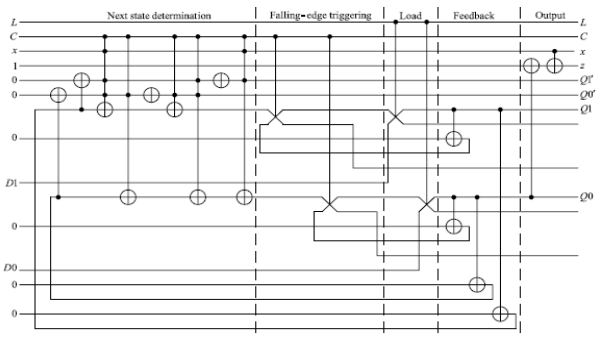

Fig.12 Reversible realization of the synchronous sequential circuit of Fig.8 with falling-edge triggering and asynchronous parallel load facility.

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 using Verilog HDL to test the correctness of the design and the simulation output is shown below.

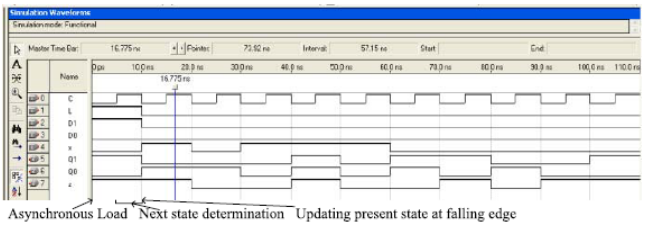

Fig.13 Verilog HDL simulation of the sequential circuit of Fig. 12

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.

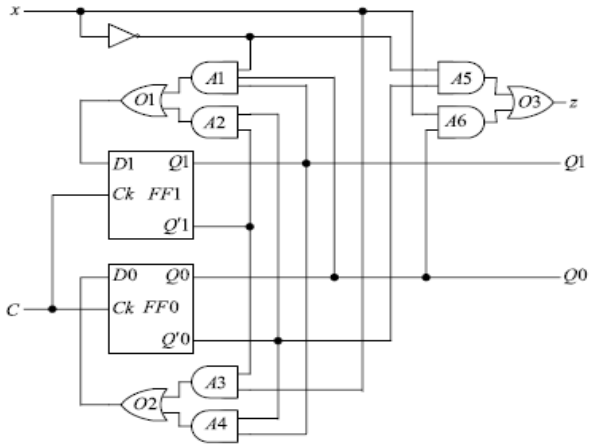

Fig.14 Classical design of the synchronous sequential circuit of Fig.8

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

Table III. Comparison of realization of Fig. 8 using the direct design method and the replacement design.

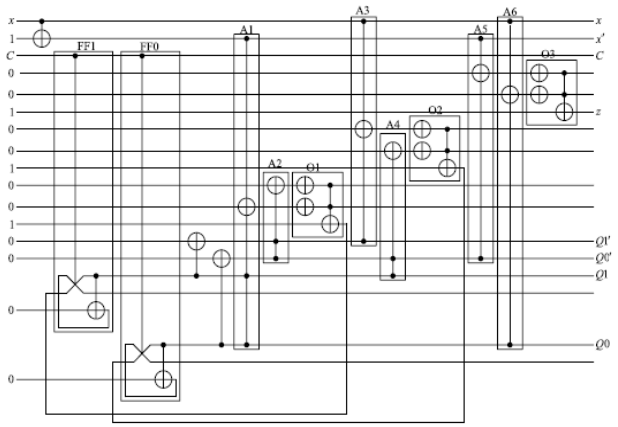

Fig.15 Reversible realization of sequential circuit of Fig.14 using replacement technique

V. DESIGN OF COUNTERS

Let us consider the PSDRM expressions for the next states of four-bit up counter as follows:

$$Q_3^+ = Q_3 \oplus CQ_2Q_1Q_0 \quad (12)$$
$$Q_2^+ = Q2 \oplus CQ_1Q_0 \quad (13)$$
$$Q_1^+ = Q_1 \oplus CQ_0 \quad (14)$$
$$Q_0^+ = Q_0 \oplus C \quad (15)$$

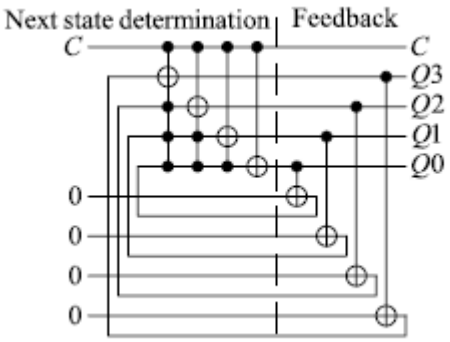

Fig.16 Reversible realization of four –bit level-triggered up counter using the PSDRM expressions (12)-(15).

Reversible realization of four bit level-triggered up counter using the PSDRM expressions of (12)–(15) is shown in Fig.16.This realization requires one 5×5 Toffoli gate, one 4×4 Toffoli gate, one 3×3 Toffoli gate, and five Feynman gates. Thus the Quantum cost is 44. It 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.

VI. DESIGN OF REGISTERS

We have determined the PSDRM expressions for the next states of four-bit level-triggered SISO shift register as follows:

$$Q_3^+ = Q_3 \oplus C\,Q_3 \oplus C\,(M'D_R \oplus M\,Q_2) \quad (16)$$
$$Q_2^+ = Q_2 \oplus C\,Q_2 \oplus C\,(M'Q_3 \oplus M\,Q_1) \quad (17)$$
$$Q_1^+ = Q_1 \oplus C\,Q_1 \oplus C\,(M'Q_2 \oplus M\,Q_0) \quad (18)$$
$$Q_0^+ = Q_0 \oplus CQ_0 \oplus C(M'Q1 \oplus MD_L) \quad (19)$$

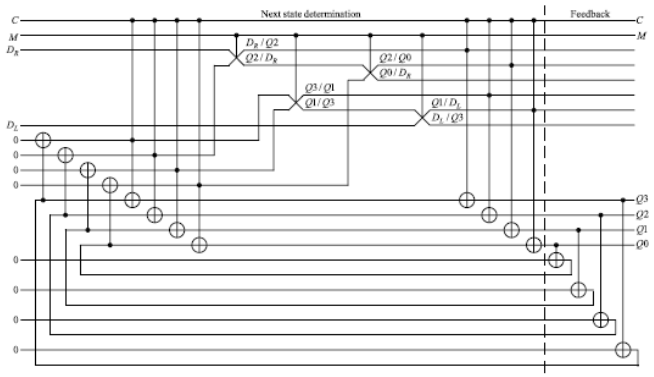

Fig.17 Realization four-bit level-triggered SISO right/left shift register using the expressions of (16)-(19).

Realization of four-bit level-triggered SISO shift register using the expressions of (16)-(19) is shown in Fig.17. The multiplexing operation of right-most parts of (16)-(19) can be implemented using Fredkin gates. Therefore the circuit of Fig.16 requires eight 3×3 Toffoli gates, four Fredkin gates, and eight Feynman gates. Thus the Quantum cost is 68. The circuit has eight Garbage outputs.

VII. CONCLUSION

Reversible logic plays an important role in emerging computing technologies due to its low power consumption. However, only a very limited works have been reported on reversible sequential circuit design. 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. With this approach we can reduce Quantum cost and Garbage outputs.

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] J. Ren and V. K. Semenov, "Progress with physically and logically reversible superconducting digital circuits," IEEE Trans. Appl. Super-cond., vol. 21, no. 3, pp. 780–786, Jun. 2011.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] T. Toffoli, "Reversible computing," MIT Lab. Comput. Sci., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TM-151, 1980.
[16] J. E. Rice, "A new look at reversible memory elements," in *Proc. IEEE ISCAS*, May 2006, pp. 243–246.
[17] 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.
[18] H. Thapliyal and A. P. Vinod, "Design of reversible sequential elements with feasibility of transistor implementation," in *Proc. ISCAS*, 2007, pp. 625–628.
[19] M.-L. Chuang and C.-Y. Wang, "Synthesis of reversible sequential elements," ACM J. Emerg. Technol., vol. 3, no. 4, pp. 1–19, 2008.
[20] 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.
[21] 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.
[22] M. H. A. Khan and M. Perkowski, "Synthesis of reversible synchronous counters," in *Proc. 41st IEEE ISMVL*, May 2011, pp. 242–247.
[23] 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.
[24] 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.
[25] 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.
[26] 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.