Efficient Approach to design Reversible Fault Tolerant Cyclic Redundancy Check Circuit

Sajib Kumar Mitra, Tamzida Sultana, Shahed Anwar and Ahsan Raja Chowdhury

***Abstract*- Conventional circuitry is not able to recover bit loss because of their inability to regain the computed states in previous. Reversible circuits, having fault tolerance capability can recover bit-loss at input as well as bit-error at output bits. In this paper, we have proposed Reversible Fault Tolerant Cyclic Redundancy Check (RFT-CRC) circuit which is first proposed in this literature. The optimized realization of Fault Tolerant Master-Slave configured D-Flip Flop is also presented here. Finally, the cost effective design of (7, 4) reversible CRC having Fault Tolerant capability has been proposed.**

***Index Terms*— Fault Tolerant, D-Flip Flop, RFT-CRC, Reversible Computing.**

I. Introduction

In reversible logic, no information is lost unlike irreversible logic which consumes power to reload information. In logic computation, every bit of information loss generates $kTln2$ joules of heat energy where k is Boltzmann's constant of 1.38×10^{-23} J/K and T is the absolute temperature of the environment stated by author of reference [1]. At room temperature, the dissipating heat is around 2.9×10^{-21} J. Bennett [2] showed that if the network consists of reversible gates only, zero power dissipation would be possible. If zero power dissipation is possible then reversibility would be an essential property for the future circuit design. Reversible logic has several applications such as nanotechnology computation [3], DNA technology [4] and optical computing [5].

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data and has a way to make CRC for bit error correction in a certain level. It is used primarily in data transmission [6]. In the CRC method, a certain number of check bits, often called checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the correctness of data, to ascertain with a certain degree of tolerance.

Manuscript received November 15, 2010; revised November 30, 2010. The review of this paper was arranged by Convener Dr. Himanshu B. Soni.

S. K. Mitra is with the Computer Science and Engineering Department, University of Dhaka, Dhaka-1000, Bangladesh (corresponding author to provide phone: +8801710283239; fax: 88028615583; e-mail: sajibmitra.csedu@yahoo.com).

T. Sultana, S. Anawar and A. R. Chowdhury are with the Computer Science and Engineering Department, University of Dhaka, Dhaka-1000, Bangladesh (email: tamzida_du@yahoo.com, shahedanwar01@yahoo.com and farhan717@univdhaka.edu).

Rest of the paper consists of the following sections: Section II provides background knowledge about reversible logic with some popular Reversible Gates and Fault Tolerant. This section also presents sequential circuit design methodology of reversible D Flip-Flop. Section III reviews the methodology of CRC with structural design of general (n, k) CRC Encoder/ Decoder. In Section IV Reversible Fault Tolerant D Flip-Flop has been proposed followed by Reversible Fault Tolerant (7, 4) CRC Encoder/Decoder along with simulation. Section V ends the paper with concluding remarks.

II. Background Study

Basic definitions and ideas related to reversible logic, fault tolerant and sequential circuit design are illustrated in this section having a brief discussion of popular reversible and Fault Tolerant gates.

A. Reversible Logic

Reversible logic has a unique mapping between input and output bit patterns where unit logic entity is represented as gates and such gates/circuits don't lose information.

Definition 1: A **Reversible Gate** is a k-input, k-output (denoted by $k \times k$) circuit that produces a unique output pattern [7, 8] for each possible input sample. Alternatively, reversible circuits have equal number of input and output bits and there is a unique mapping between the input and output vectors.

Let us assume that input vector is I_v and output vector is O_v where, $I_v = I_0, I_1, I_2, \ldots, I_{k-1}$ and $O_v = O_0, O_1, O_2, \ldots, O_{k-1}$ then according to the definition, $I_v \leftrightarrow O_v$.

Fig. 1 shows the symbolic representation of conventional Ex-OR Gate and the mapping between Input and Output vectors which is not unique. So there is no way to regain input values from output. As a result, conventional logic is called irreversible and thus they can't determine the input vectors from output vectors uniquely. Reversible Ex-OR operation can be realized by using one Feynman Gate [9] where the number of inputs and number of outputs are equal and they can be uniquely mapped as shown in Fig. 2.

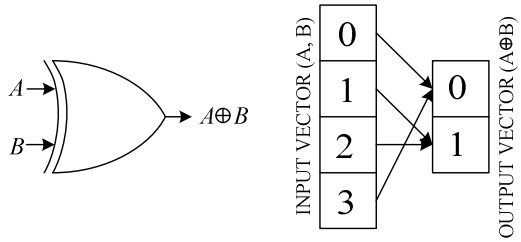

Fig. 1. Conventional EX-OR gate and its input-output vector mapping

ISBN: 978-1-6123-3002-0

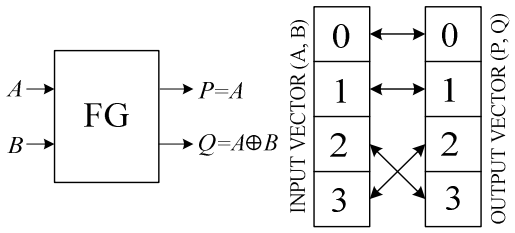

Fig. 2. Reversible Feynman Gate and the Input-Output unique mapping

Feynman Double Gate (F2G) [10], Peres Gate (PG) [11], Toffoli Gate (TG) [12], Fredkin Gate (FRG) [13], are the well-known reversible gates, shown in Fig. 3.

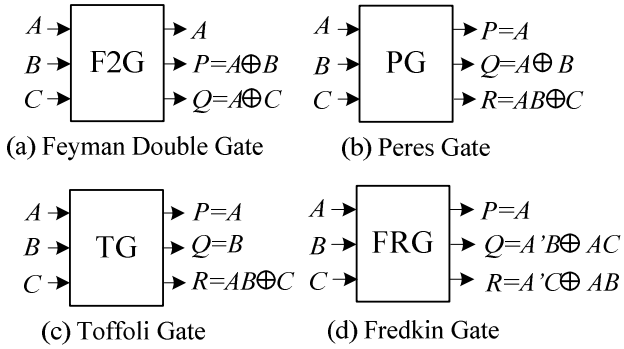

Fig. 3. Most popular Reversible Gates

B. Fault Tolerant Method

Reversible logic has a property called parity preserving where every mapped input and output vectors of a gate preserves same parity which can be even or odd.

Definition 2: Fault Tolerant Gate constantly preserves same parity between every mapped input and output and it is reversible. It is also called **Parity Preserving Gate**.

Let, I_v be the input vector and O_v be the output vector of a (*kxk)* reversible gate where, $I_v = I_1, I_2, I_3, I_4, \ldots, I_k$ and $O_v = O_1, O_2, O_3, O_4, \ldots, O_k$. According to the definition we get,

$$I_1 \oplus I_2 \oplus I_3 \oplus \cdots \oplus I_k = O_1 \oplus O_2 \oplus O_3 \oplus \cdots \oplus O_k$$

Fig. 4 shows the pictorial representation of Input-Output mapping of Parity Preserving FRG gate.

There are number of Reversible Fault Tolerant gates and the dimension of these gates can be of any number but lower dimension is preferable for designing efficient circuits [10].

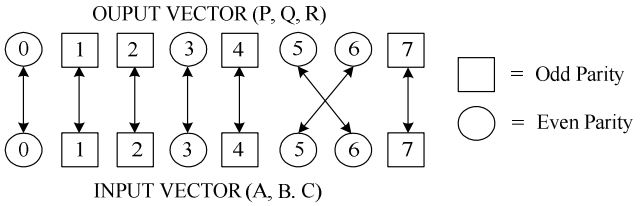

Fig. 4. Input-Output mapping of Fault Tolerant FRG gate

C. Quantum Cost Realization

To judge the competence of the reversible circuit Quantum Cost is another common factor.

Definition 3: Quantum Cost (QC) of any reversible circuit is the number of 2x2 Quantum gates (which is a 4x4 unitary matrix) used to realize equivalent quantum circuit where the cost of every 2x2 quantum primitives is 1 and there is no cost for 1x1 [14].

Every permutation of logic function can be constructed from 1x1 and 2x2 Quantum gates. There are four types of Quantum gates [15] and these are given in Fig. 5.

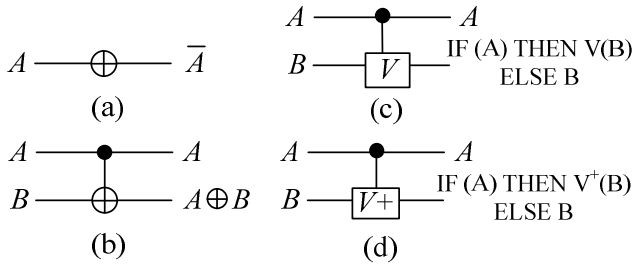

Fig. 5. Basic Quantum Primitives a. NOT operator, b. Ex-OR Gate, c. Square Root of NOT (SRN) and d. Hermitian of SRN

The operations of quantum gates are represented by unitary matrix which multiplies with the state of qubit [14]. The V gate is the square root of NOT gate and V^+ is the hermitian matrix of V, where $VV^+ = I$.

Following figure (shown in Fig. 6) shows the Quantum realization of few reversible gates.

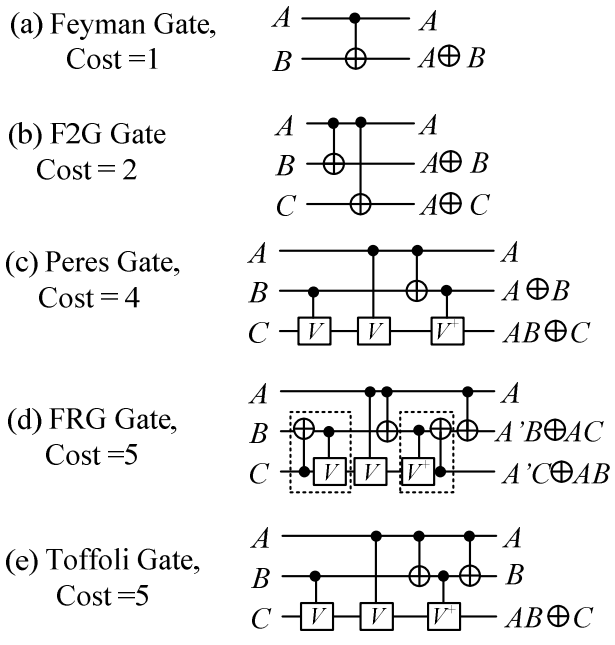

Fig. 6. Quantum Realization of popular Reversible Gates

D. Sequential Circuit Design

Reversible logic synthesis with respect to sequential logic is different from combinational logic because of the value of output of any sequential circuit depends not only on the present inputs but also the previous output.

The D Flip-Flop is a modification of the clocked RS Flip-Flop where the D input goes directly to the S input and its complement is applied to R input.

Fig. 7 shows the design of D Flip-Flop which is designed form irreversible gates [16].

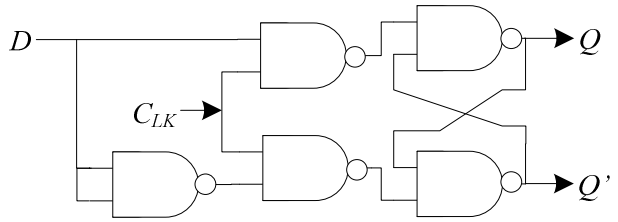

Fig. 7: Conventional [Irreversible] D Flip-Flop

In our proposed design we use edge triggered D Flip-Flops which are of two types: positive edge triggered and negative edge triggered.

III. Cyclic Redundancy Check Circuit

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, and making correction [6]. In CRC module consist of two parts: one is encoder and the other is decoder. Encoder is one which generates codeword from data-word and the decoder receives the codeword and generates the remainder of zero or non-zero syndrome. The term zero syndrome specifies the remainder contains only zeros and non-zero syndrome for others.

For example, suppose a data-word is of k bits and the codeword is of n bits. The data-word is augmented by adding (n-k) 0's to the right-hand side of the word. The augmented n-bit data-word is divided by the divisor of size (n-k+1). The quotient of the division is discarded and the remainder ($r_1, r_2, \ldots, r_{n-k}$) is appended to the data-word to create the codeword. The n-bit codeword is received by the decoder and the checker produces the remainder of syndrome of size (n-k) bits.

Fig. 8 shows the general (n, k) CRC encoder/ decoder [6]:

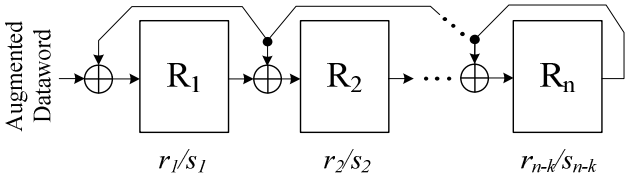

Fig. 8. General (n, k) CRC Encoder/ Decoder

If syndrome bits are all 0s then the data-word (excluding augmented 0 from codeword) is accepted otherwise discarded.

Mathematical Analysis:

Let, Data-word, $d(x)$, Codeword: $c(x)$, Generator: $g(x)$, Syndrome: $s(x)$, Error: $e(x)$

If $s(x)! = 0$, one or more bits are corrupted. More bits refer to the occurrence of odd numbers of errors or burst errors for which the codeword isn't divisible.

If $s(x) == 0$, either

a. No bit is corrupted. Or

b. Some bits are corrupted, but the decoder is failed to detect them

Received codeword = $c(x) + e(x)$

The checker divides the received codeword by $g(x)$ to get the syndrome. This can be written as,

Received codeword/g(x) = $c(x)/g(x) + e(x)/g(x)$

Those errors $e(x)$ that is divisible by $g(x)$ is never been considered.

IV. Proposed Design

A. Design of Fault Tolerant D-Flip Flop

The proposed design of Reversible Fault Tolerant D Flip-Flop (RFT-DFF) consists of two FRGs and two F2Gs gates which are fault tolerant gates.

ALGORITHM 1: Fault_Tolerant_D_FF__Design

INPUT : Data bit, D and Clock Pulse, C_{LK}
OUTPUT: Q
BEGIN

Step 1:

Set C_{LK} and D as 1st and 2nd input respectively of primary FRG and the 3rd input comes from the primary F2G gate which is attached in Loop (shown in Fig. 9).

Step 2:

Connect the 1st output (C_{LK}) of primary FRG to the 1st input of secondary FRG. 3rd input of secondary FRG gets data (D) from the 3rd output of primary F2G. Finally, 1st output of secondary F2G acts as 2nd input of secondary FRG and there is a crossing relation between secondary FRG and secondary F2G (shown in Fig. 9).

Step 3:

Secondary F2G generates Q as final output.

END

Fig. 9 shows the proposed design of RFT-DFF and Table I shows the comparison between Existing and Proposed design of D-FF. Table I shows the number of garbage outputs in proposed design is minimized and also the other designs are not fault tolerant. On the other hand, the difference of QC is so close.

ISBN: 978-1-6123-3002-0

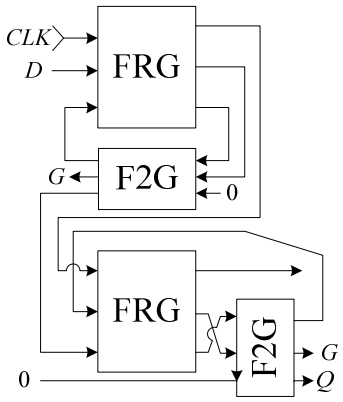

Fig. 9. Proposed Reversible Fault-Tolerant D Flip Flop (RFT-DFF)

TABLE I
COMPARISON BETWEEN PROPOSED AND EXISTING DESIGN OF D FLIP FLOP

Reversible D-Flip Flop	Number of Gates	Number of garbage outputs	Quantum Cost
Existing Design [17]*	5	3	12
Existing Design [18]*	4	3	12
Proposed Design (Fault Tolerant)	4	2	14

*Not Fault Tolerant

Proposed Fault Tolerant Reversible D Flip-Flop or RFT-DFF is used to design Fault Tolerant CRC which is illustrated in the following section.

B. Design of Fault Tolerant (7, 4) CRC

Proposed design of (*7*, *4*) Reversible Fault tolerant CRC or RFT-CRC is fundamentally based on serial CRC where checksum will be produced by insertion of input sequentially. The design of CRC encoder and decoder is identical.

Theorem 1: (*7*, *4*) RFT-CRC encoder/decoder can be realized by 3 RFT-DFFs and three F2G gates.

Proof: The size of Data-word is 4 and for Code-word is 7 where the size of redundant bit is 3, which will be calculated step by step by XOR operation. Consequent XOR operations check the Most Significant Bit (MSB) of intermediate state *s* where size is 4. In every stage of XOR operations the MSB of resultant value is always 0 and for this reason, the next bit is selected to make decision about the XOR operand (either 1111 or 0000). So, the total number of bits needed to store in every stage is 3. To store those 3 bits in CRC module, the required number of RFT-DFF is 3 and each three XOR operations can be realized by only 3 F2G gates.

So, 3 RFT-DFFs and 3 F2G gates are sufficient for the implementation of (7, 4) RFT-CRC. □

ALGORITHM 2: Fault_Tolerant_CRC_Design

INPUT : D (D_0, D_1, D_2, D_3, D_4, D_5, D_6) and C_{LK}
OUTPUT: Redundant Bits (r_1, r_2, r_3)

BEGIN

Step 1:
Set the clock pulse (C_{LK}) to every RFT-DFF. Connect all RFT-DFFs with one another.

Step 2:
Every RFT-DFF gets XORed data value from its prior RFT-DFF (shown in Fig. 10).

Step 3:
After few stages every intermediate F2G will generate remainders (r_1, r_2, r_3).

END

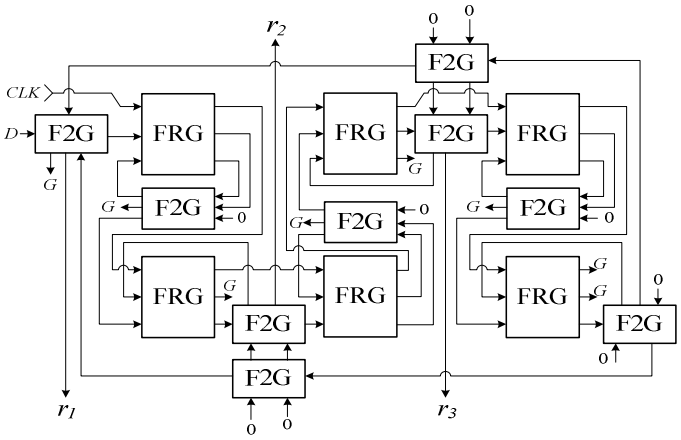

Fig. 10. Proposed Design of Reversible Fault-Tolerant CRC (RFT-CRC)

The realization of proposed design is shown in Table II.

TABLE II
COST CALCULATION OF (7, 4) RFT-CRC

Cost Factors	Proposed design of (7, 4) RFT-CRC
Number of Gates	15
Number of garbage outputs	8
Quantum cost	48

Fig. 11 shows the generalized structure of (*n*, *k*) Reversible Fault Tolerant CRC (RFT-CRC).

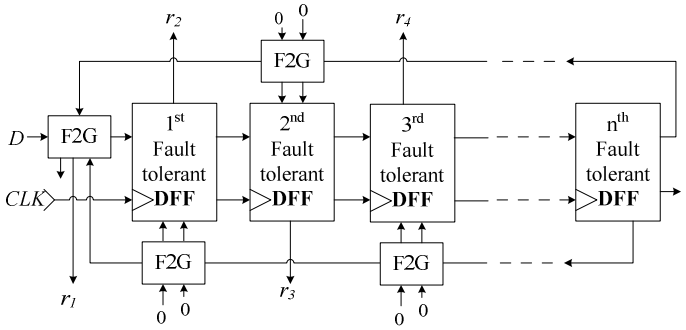

Fig. 11. Proposed Design of Reversible Fault-Tolerant CRC (RFT-CRC)

V. CONCLUSION

Error detection and correction methodology has been used over the year for the verification of data. Fault Tolerant Reversible CRC is first proposed in this research work and can also detect multiple bit errors. Error detection and correction methods become easier by introducing our proposed circuitry. Fault free and less power consumption circuitry of proposed Reversible Fault Tolerant CRC will add a new dimension in the field of Reversible Computing. Our future plan is to design parallel CRC where resultant circuit will give better performance than serial CRC.

REFERENCES

[1] R. Landauer, "Irreversibility and Heat Generation in the Computational Process", IBM Journal of Research and Development, Vol-5, pp. 183-191,1961.
[2] C. H. Bennett, "Logical reversibility of computation", IBM Journal of Research and Development, vol- 17, pp. 525–532, 1973.
[3] G. E. Moore, "Cramming more Component onto integrated Circuit", J. Electron, vol-38, 1965.
[4] M. Frank, "Physical Limits of Computing", CIX 4930.1194X/6930.1078 X, 2000.
[5] M. A. Perkowski, "Reversible Computation for Beginners", Lecture Series, 2000, Portland State University, http://www.ee.pdx.edu/~mperkows (last access date 02/07/10).
[6] Book of 'Data Communication and Networking',4/e by *Behrouz A. Forouzan.*
[7] H. M. H. Babu, M. R. Islam, A. R. Chowdhury, S.M. A. Chowdhury, "Reversible logic synthesis for minimization of full-adder circuit", IEEE Conf. Digital Syst. Des, pp. 50–54, 2003.
[8] H. M. H. Babu, M. R. Islam, A. R. Chowdhury, S.M. A. Chowdhury, "Synthesis of full-adder circuit using reversible logic", 17th International Conference on VLSI Design, January, pp. 757–760, 2004.
[9] R. Feynman,"Quantum mechanical computers", Optics News, Vol-11, pp. 11-20, 1985.
[10] B. Parhami, "Fault tolerant reversible circuits", Proc. of 40th Asimolar Conf. Signals, Systems, and Computers, Pacic Grove, CA, pp. 1726-1729, 2006.
[11] A. Peres, "Reversible logic and quantum computers", American Physical Society, Vol- 151, pp. 3266- 3276, 1985.
[12] T. Toffoli, "Reversible computing", Tech memo MIT/LCS/TM, Vol. 151, pp. 11-20, 1980.
[13] E. Fredkin and T. Toffoli, "Conservative logic", International Journal of Theoretical Physics, vol- 21, pp. 219-253, 1982.
[14] M. Perkowski, "A hierarchical approach to computer-aided design of quantum circuits, 6th International Symposium on Representations and Methodology of Future Computing Technology, pp. 201—209, 2003.
[15] W. Hung, X. G. Song, J. Yang, M. Perkowski, "Quantum Logic Synthesis by Symbolic Reachability Analysis", Design Automation Conference, pp .838-841, 2004.
[16] H. Thapliyal, M. B. Srinivas, and M. A. Zwolinski, "Beginning in the Reversible Logic Synthesis of Sequential Circuits", 8th MAPLD Conference (NASA office of Logic Design), Washington D.C, Sep, 2005.
[17] N. M. Nayeem, L. Jamal and H. M. H. Babu,"Efficient Reversible Montgomery Multiplier and Its Application to Hardware Cryptography", Journal of Computer Science, vol- 5, pp. 49-56, 2009.
[18] S. K. Hari, S. Shroff, N. Mahammad, and V. Kamakoti,"Efficient building blocks for reversible sequential circuit design", 49th IEEE International Midwest Symposium on Circuits and Systems MWSCAS, Vol- 1, pp. 437-441, 2006.