4th Workshop on
Reversible Computation (RC)
July 2nd -3rd 2012, Copenhagen, Denmark


Program

Monday	July 2nd, 2012
from 8.00	Registration
	Morning coffee
8.50-9.00	Opening Session
9.00-9.50	Invited Talk **The Physics of Information: from Maxwell's Demon to Landauer** *Eric Lutz (Freie Universität Berlin, Germany)* *Chair: Alexis De Vos*
9.50-10.10	Coffee break
10.10-12.00	Session 1 **Theoretical Considerations** *Chair: Neil D. Jones*
	One-Way Reversible Multi-Head Finite Automata *Martin Kutrib, Andreas Malcher (Giessen University, Germany)* **A Deterministic Two-Way Multi-Head Finite Automaton can be Converted into a Reversible one with the same Number of Heads** *Kenichi Morita (Hiroshima University, Japan)* **Undecidability of the Surjectivity of the Subshift Associated to a Turing Machine** *Rodrigo Torres (University of Concepción, Chile), Nicolas Ollinger (University of Orléans, France), Anahí Gajardo (University of Concepción, Chile)* **WiP: On Radius 1 Nontrivial Reversible and Number-Conserving Cellular Automata** *Katsunobu Imai (Hiroshima University, Japan), Bruno Martin (University Nice-Sophia Antipolis, France), Ryohei Saito (Hiroshima City Hall, Japan)*
12.00-13.30	Lunch
13.30-15.20	Session 2

Workshop Organizer

Holger Bock Axelsen
University of Copenhagen

Department of Computer Sci
Universitetsparken 1
2100 Denmark
info@reversible-computation

	Reversible Software and Languages *Chair: Kenichi Morita*
	Isomorphic Interpreters from Logically Reversible Abstract Machines *Roshan P. James, Amr Sabry (Indiana University, USA)* **Synthesizing Loops for Program Inversion** *Cong Hou (Georgia Institute of Technology, USA), Daniel Quinlan, David Jefferson (Lawrence Livermore National Laboratory, USA), Richard Fujimoto, Richard Vuduc (Georgia Institute of Technology, USA)* **Frugal Encoding in Reversible MOQA: A Case Study for Quicksort** *Diarmuid Early, Ang Gao, Michel Schellekens (University College Cork, Ireland)* **WiP: Towards a General-Purpose, Reversible Language for Controlling Self-Reconfigurable Robots** *Ulrik Pagh Schultz (University of Southern Denmark, Denmark)*
15.20-15.50	Coffee break
15.50-17.10	Session 3 **Synthesis of Reversible Circuits** *Chair: Robert Wille*
	Reversible and Quantum Circuit Optimization: A Functional Approach *Zahra Sasanian, D. Michael Miller (University of Victoria, Canada)* **Optimal 4-bit Reversible Mixed-Polarity Toffoli Circuits** *Marek Szyprowski (Warsaw University of Technology, Poland), Pawel Kerntopf (Warsaw University of Technology and University of Łódź, Poland)* **WiP: A New Synthesis of Reversible and Quantum Realizations of Symmetric Boolean Functions** *Arighna Deb (Institute of Engineering and Management, India), Debesh K. Das (Jadavpur University, India), Hafizur Rahaman (Bengal Engineering and Science University, India), Bhargab B. Bhattacharya (Indian Statistical Institute, India)*
	Banquet
Tuesday	**July 3rd, 2012**
	Morning coffee
9.00-10.30	Session 4 **Quantum Circuits and Computation** *Chair: D. Michael Miller*
	Tutorial: Graphical Calculus for Quantum Circuits *Bob Coecke (Oxford University, UK), Ross Duncan (Université libre de Bruxelles, Belgium)* **A 2D Nearest-Neighbor Quantum Architecture for Factoring** *Paul Pham (University of Washington, USA), Krysta M. Svore (Microsoft Research, USA)*

Properties of Quantum Templates
Md. Mazder Rahman, Gerhard W. Dueck (University of New Brunswick, Canada)

10.30-10.50 Coffee break

10.50-12.00 Session 5
Physical Realizations and Design
Chair: Irek Ulidowski

Garbageless Reversible Implementation of Integer Linear Transformations
Stéphane Burignat, Kenneth Vermeirsch, Alexis De Vos (Ghent University, Belgium), Michael Kirkedal Thomsen (University of Copenhagen, Denmark)

WiP: Garbage-Free Integer Multiplication with Constants
Holger Bock Axelsen, Michael Kirkedal Thomsen (University of Copenhagen, Denmark)

WiP: Using π DDs in the Design for Reversible Circuits
Mathias Soeken, Robert Wille (University of Bremen, Germany), Shin-ichi Minato (Hokkaido University, Japan), Rolf Drechsler (University of Bremen, Germany)

12.00-13.30 Lunch

13.30-14.50 Session 6
Distributed Systems
Chair: Michel Schellekens

A Verification Technique for Reversible Process Algebra
Jean Krivine (Université Paris Diderot, France)

A Reversible Process Calculus and the Modelling of the ERK Signalling Pathway
Iain Phillips (Imperial College London, UK), Irek Ulidowski (University of Leicester, UK), Shoji Yuen (Nagoya University, Japan)

WiP: Controlled Reversibility and Compensations
Ivan Lanese (University of Bologna, Italy), Claudio Antares Mezzina (FBK, Italy), Jean-Bernard Stefani (INRIA, France)

14.50-15.20 Coffee break

15.20-16.40 Session 7
Testing, Verification and Fault Tolerance
Chair: Pawel Kerntopf

Property Checking of Quantum Circuits Using Quantum Multiple-Valued Decision Diagrams
Julia Seiter, Mathias Soeken, Robert Wille, Rolf Drechsler (University of Bremen, Germany)

An Efficient Approach to Design Reversible Fault Tolerant Plessey Logic Block of Field Programmable Gate Arrays
Md. Shamsujjoha, Nazma Tara, Lafifa Jamal, Hafiz Md. Hasan Babu (University of Dhaka, Bangladesh)

WiP: Design of an Online Testable Ternary Circuit from the Truth Table
Noor M. Nayeem, Jacqueline E. Rice (University of Lethbridge, Canada)

16.40-16.50 Closing Session

An Efficient Approach to Design Reversible Fault Tolerant Plessey Logic Block of Field Programmable Gate Arrays

Md. Shamsujjoha, Nazma Tara, Lafifa Jamal, and Hafiz Md. Hasan Babu**

Department of Computer Science and Engineering, University of Dhaka, Bangladesh.
{dishacse,taranaz2003,lafifa}@yahoo.com, and haifzbabu@hotmail.com

Abstract. This paper demonstrates the reversible logic synthesis of the logic element of Plessey **F**ield **P**rogrammable **G**ate **A**rray (FPGA). The circuits are designed using only parity preserving reversible gates. Thus the entire scheme inherently becomes fault tolerant. We have proposed algorithms to design compact reversible fault tolerant decoder, multiplexer, random access memory of Plessey FPGA. Comparative results show that the proposed method performs much better and has significantly better scalability in comparison to the previous approach.

1 Introduction

Maxwell and Szilard proved $KT\ ln(m)$ joules of energy is dissipated during every elemental enactment of computation, where K is Boltzmann's constant of $1.38\times10^{23}\ JK^{-1}$ and T is the absolute temperature of the environment and m is an integer number proportional to number of computed bits. Later Landauer showed that the energy dissipation is only avoidable if the system is made reversible [1]. In [2], Bennett showed that the reversible computation dissipate $KT\ ln(1)$ joule of energy, which is logically zero. It has also been pointed out that, an irreversible system has a fundamental lower limit to the energy dissipation during a computation which equals to $KT\ ln(2)$ for each erased bit. An irreversible system can also store the information that is produced during a computation rather than erasing it, but doesn't always provide a unique path from each state to its previous state. Energy used to store these information is unrecoverable. This assumption has a great importance in the future technological necessity that these power dissipations should be minimized. In this regard, reversible logic plays an extensively crucial role. Moreover, reversible circuit can be viewed as a special case of quantum circuit as quantum evolution must be reversible. On the other hand, reversible fault tolerant circuit allows to detect faulty signal in the primary outputs of the circuit through parity checking. Parity checking is the oldest and widely used mechanisms for detecting single level fault in communication. Its most common use is to detect errors in the storage or transmission of information, primarily because most arithmetic and other processing functions

** Corresponding Author

tend not to preserve the parity of the data [3]. If the parity of the input data is maintained throughout the computation then intermediate checking wouldn't be required and an entire circuit can preserve parity if its individual gate is parity preserving. Over the past few years, both reversible and fault tolerant circuitry gained remarkable interest in the field of DNA-technology, nano-technology, optical computing, program debugging and testing, database recovery, quantum dot cellular automata, discrete event simulation, modeling of biochemical systems, in the development of highly efficient algorithms etc.

Field **P**rogrammable **G**ate **A**rray (FPGA) has become an extremely useful medium for the digital designers as it capable of testing the products even before the fabrication. FPGA enables designers to avoid the pitfalls of nano-meter designs till the last second. Usually FPGA consists of an array of programmable logic blocks, routing channels and I/O cells. Logic blocks can be configured to implement sequential and complex combinational functions. Hence, influence both speed and density of an FPGA. As FPGAs are approximately 10 times less dense and three times slower than the mask programmed gate arrays, researchers are motivated to explore new logic blocks such that these density and speed gap become as minimum as possible. Most popular logic blocks of the FPGAs are based on look-up tables (LUTs) and design form Plessey. A look-up table can implement any function of its inputs and accordingly described by their number of inputs. With more inputs, a LUT can implement more logic, and hence fewer logic blocks are needed. This helps in routing by asking for less area, since there are fewer connections to route between the logic blocks. Main claim against the LUT is the exponential growth of complexity with respect to inputs. However, the Plessey logic block overcomes these difficulties through clusters of components, *e.g.*, NAND units, random access memories, multiplexers and latches. Therefore, this paper investigates the design methodologies of reversible fault tolerant Plessey FPGA[1].

2 Basic Definitions and Literature Review

This section provides basic definitions related to reversible and fault tolerant logic synthesis. The section ends with an introduction to the popular reversible and fault tolerant gates along with their quantum equivalent representations.

2.1 Reversible and Fault Tolerant Gates

An $n \times n$ reversible gate is a data stripe block which uniquely maps between input vector $I_v = (I_0, I_1, \ldots, I_{n-1})$ and output vector $O_v = (O_0, O_1, \ldots, O_{n-1})$. A Fault tolerant gate is a reversible gate that constantly preserves same parity between inputs and outputs. Specifically, it maintains the following property:

$$I_0 \oplus I_1 \oplus \ldots \oplus I_{n-1} = O_0 \oplus O_1 \oplus \ldots \oplus O_{n-1} \tag{1}$$

[1] Plessey FPGA and Plessey logic block of an FPGA are used interchangeably throughout the paper.

2.2 Garbage Output

Unwanted or unused output(s) of a reversible gate (or circuit) is known as garbage output(s), *i.e.*, outputs that neither used as primary outputs nor as input to another reversible gate are garbage outputs.

2.3 Popular Reversible and Fault Tolerant Gates

In this section we define some reversible fault tolerant gates and present their properties.

Feynman and Feynman Double Gate: Input vector, I_v and output vector, O_v for 2×2 reversible Feynman gate (FG) is defined as follows : $I_v = (a, b)$ and $O_v = (a, a \oplus b)$. Block diagram of FG is shown in Fig. 1(a). Input vector, I_v and output vector, O_v for 2×2 reversible Feynman double gate (FG) is defined as follows : $I_v = (a, b, c)$ and $O_v = (a, a \oplus b, a \oplus c)$. Block diagram of $F2G$ is shown in Fig. 1(b).

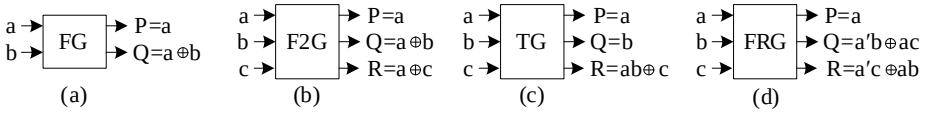

Fig. 1: Block diagram of popular reversible gates : (a) Feynman Gate (b) Feynman Double gate (c) Toffoli gate (d) Fredkin gate.

Toffoli Gate and Fredkin Gate: Input and output vectors for 3×3 Toffoli gate (TG) is defined as follows : Input vector $I_v = (a, b, c)$ and output vector $O_v = (a, b, ab \oplus c)$. Block diagram of TG is shown in Fig. 1(c). For 3×3 Fredkin gate (FRG), $I_v = (a, b, c)$ and $O_v = (a, a'b \oplus ac, a'c \oplus ab)$. Its block diagram is shown in Fig. 1(d). FRG can implement logical AND, OR, NOT and NAND operations like TG. FRG can also be used as 2-to-1 Multiplexer. Referring to the Fig. 1(d), when a=1 the inputs b and c will be swapped, produce the outputs as Q=c and R=b; whereas, if a=0, then outputs P, Q and R will be directly connected to inputs a(=0), b and c, respectively.

Reversible Fredkin and Feynman double gate comply the rule of Eq.1. Therefore, if a circuit is designed using only these two gates, the circuit will inherently become fault tolerant. The fault tolerant (parity preserving) property of Fredkin and Feynman double gate is shown in Table 1.

2.4 Quantum Equivalent Realization of Reversible Gates

Figs. 2(a-d) represent the quantum equivalent realization of FG, $F2G$, TG and FRG, respectively. In these figures, V is a square root-of-NOT (SRN) gate and V^+ is its hermitian. Thus, VV creates a unitary matrix of NOT gate and its cost

Table 1: Truth table for $F2G$ and FRG.

Input			Output of $F2G$			Output of FRG			
A	B	C	P	Q	R	P	Q	R	Parity
0	0	0	0	0	0	0	0	0	Even
0	0	1	0	0	1	0	0	1	Odd
0	1	0	0	1	0	0	1	0	Odd
0	1	1	0	1	1	0	1	1	Even
1	0	0	1	1	1	1	0	0	Odd
1	0	1	1	1	0	1	1	0	Even
1	1	0	1	0	1	1	0	1	Even
1	1	1	1	0	0	1	1	1	Odd

is zero. And $VV^+ = I$, an identity matrix which is describing just a quantum wire having zero quantum cost. Since quantum cost of a reversible gate or circuit is the total number of 2×2 quantum primitives, thus the quantum cost of FG, $F2G$, TG and FRG are 1, 2, 5 and 5, respectively (each rectangle in Fig. 2(d) is equivalent to 2×2 quantum primitives [4, 5]).

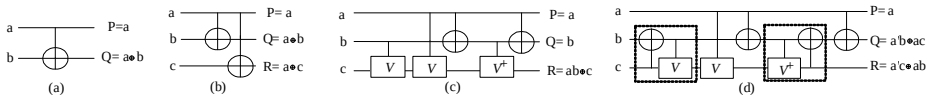

Fig. 2: Quantum equivalent realization of popular reversible gates : (a) Feynman Gate (b) Feynman Double gate (c) Toffoli gate (d) Fredkin gate.

3 Field Programmable Gate Arrays

As shown in Fig. 3(a), the routing network provides connections between the logic blocks of FPGA. The architecture of a plessey logic block is shown in Fig. 3(b). Here the basic block is a two-input NAND gate. The logic is formed by connecting NAND gate to implement the desired function(s). Other essential components of Plessey FPGA are a **R**andom **A**ccess **M**emory (RAM), 8-to-2 **Mu**ltiple**x**er (Mux) and a D-Latch. More details can be found in [6, 7].

An efficient reversible Plessey logic block was proposed in [8]. Authors [8] designed all the components of reversible Plessey FPGA using two new reversible gates (reversible NH and BSP gate) in addition to FG and TG gates.

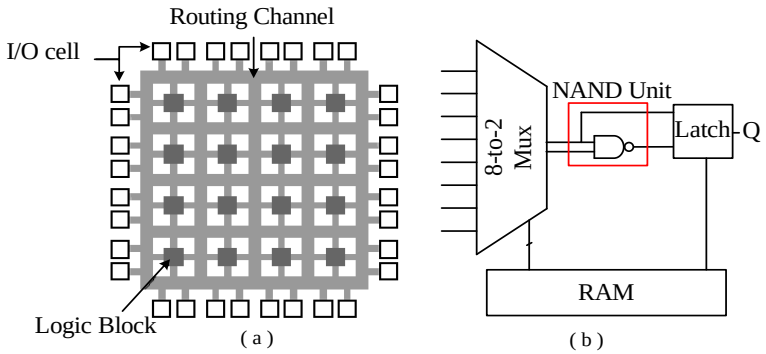

Fig. 3: Block diagram : (a) Basic irreversible FPGA (b) Plessey logic block

4 Proposed Reversible Fault Tolerant Plessey FPGA

This section has proposed reversible fault tolerant D-Latch, 8-to-2 Mux, RAM and NAND unit of Plessey FPGA. To design the RAM of Plessey FPGA, we extend our proposed reversible RAM [9] to reversible fault tolerant one.

4.1 Proposed Reversible Fault Tolerant D-Latch of Plessey FPGA

Fig. 4 represents the architecture of proposed reversible fault tolerant D-Latch of Plessey FPGA. From the figure, we find that the proposed Latch consists of reversible fault tolerant Fredkin and Feynman double gates.

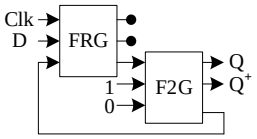

Fig. 4: Proposed reversible fault tolerant D-Latch

4.2 Proposed RFMux of Plessey FPGA

A $4n$-to-n **R**eversible **F**ault tolerant **Mux** (RFMux) has been proposed here. Considering the simplest case n=1, we have 4-to-1 multiplexer. According to the design procedure, a 4-to-1 RFMux requires a 2-to-4 **R**eversible **F**ault tolerant **D**ecoder (RFD). Increasing n from 1 to 2, we have 8-to-2 RFMux which demands a 3-to-8 RFD and so on.

Algorithm 1: Algorithm for the proposed reversible fault tolerant n-to-2^n decoder of Plessey FPGA, ***RFD(S, F2G, FRG)***

Input : Data input set $S(S_0, S_1, ..., S_{n-1})$
Feynman double gate ($F2G$) and Fredkin gate (FRG)
Output: n-to-2^n reversible fault tolerant decoder circuit

```
1  begin
2      i = input
3      o = output
4      for j ← 0 to n − 1 do
5          if j = 0 then
6              S_j → first.i.F2G, 1 → second.i.F2G, 0 → third.i.F2G
7          end if
8          else
9              S_j → first.i.FRG, 0 → second.i.FRG
10             if j=2 then third.o.F2G → third.i.FRG end if
11             else call (RFD(n − 1-to-2^(n−1))), RFD.o.j → third.i.FRG_j end if
12         end if
13     end for
14     return  FRG.o.3 and FRG.o.2 → desired output
15             remaining F2G.o & FRG.o → garbage output
16 end
```

Design Procedure of the Proposed RFD of Plessey FPGA: Figs. 5(a-b) show the architecture of 2-to-4 and 3-to-8 RFD of Plessey FPGA, respectively. From Fig. 5(b) we find that 3-to-8 RFD is designed using 2-to-4 RFD. Algorithm 1 presents the design procedure of the proposed n-to-2^n RFD. In the above algorithm (Algorithm 1) and all the following algorithms, $A \leftarrow b$ means value b is assigned to variable A. On the other hand, $a \rightarrow B$ means a is assigned to B.

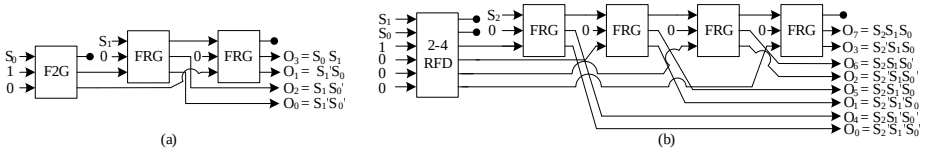

Fig. 5: (a) Proposed 2-to-4 RFD (b) Proposed 3-to-8 RFD

Design Procedure of the Proposed RFMux of Plessey FPGA: Fig. 6 shows the architecture of 4-to-1 RFMux of Plessey FPGA. From Fig. 6, we find that 4-to-1 RFMux requires a 2-to-4 RFD. So, according to this design procedure, a $4n$-to-n RFMux requires a m-to-2^m RFD (it is assumed that 2^m=$4n$). Algorithm for RFD design has already been shown in Algorithm 1, whereas Algorithm 2 presents the design procedure of the proposed $4n$-to-n RFMux.

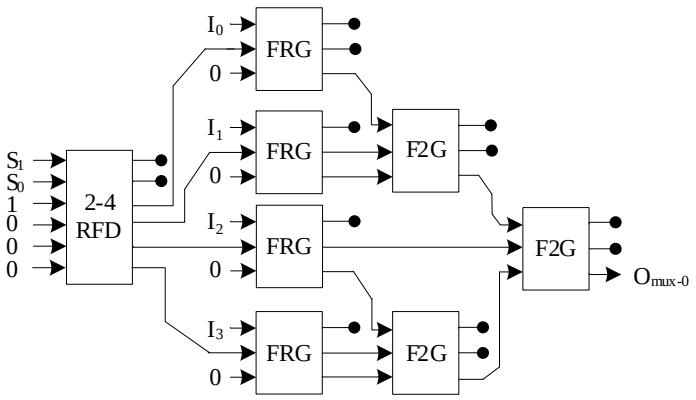

Fig. 6: Block diagram of the proposed 4-to-1 RFMux

Algorithm 2: Algorithm for the proposed reversible fault tolerant $4n$-to-n Mux of Plessey FPGA, ***RFMux(I, S, F2G, FRG)***

Input : Data input $I(I_0, I_1, \ldots, I_{4n-1})$, Control input $S(S_0, S_1, ..., S_{2n-1})$
Feynman double gate ($F2G$) and Fredkin gate (FRG)
Output: $4n$-to-n reversible fault tolerant Mux circuit

begin
  $i = input$, $o = output$
  for $j \leftarrow 0$ **to** $n-1$ **do**
    Call RFD(S_j,$F2G$,FRG)
    for $k \leftarrow 0$ **to** $4n-1$ **do**
      $first.i.FRG.k \leftarrow I_k$, $second.i.FRG.k \leftarrow RFD.(n-1)$,
      $third.i.FRG.k \leftarrow 0$
    end for
    for $l \leftarrow 0$ **to** $2n-1$ **do**
      $first.i.F2G.l \leftarrow o_3.FRG.j.second.i.F2G.l \leftarrow o_2.FRG.j+1$,
      $third.i.FRG.l \leftarrow o_3.FRG.j+1$
    end for
    for $m \leftarrow 2n$ **to** $3n-1$ **do**
      $first.i.F2G.m \leftarrow o_3.F2G.0$, $second.i.F2G.m \leftarrow o_2.F2G.1$,
      $third.i.F2G.m \leftarrow o_3.F2G.1$
    end for
  end for
  return
  for $j \leftarrow 3n-1$ **to** $2n$ **do**
    $F2G.o.3.j \rightarrow$ *desired output*
  end for
  remaining $F2G.o$ *&* $FRG.o \rightarrow$ *garbage output*
end

4.3 Proposed Reversible Fault Tolerant RAM of Plessey FPGA

RFRAM of Plessey FPGA requires write enable **M**aster-**S**lave **F**lip-**F**lops (MSFF). Fig. 7(a) shows the architecture of proposed **R**eversible **F**ault tolerant **MSFF** (RFMSFF) of FPGA. Proposed RFMSFF consists of a Fredkin gate, two Feynman double gates and two reversible **F**ault **T**olerant **D**-Latches (FTD).

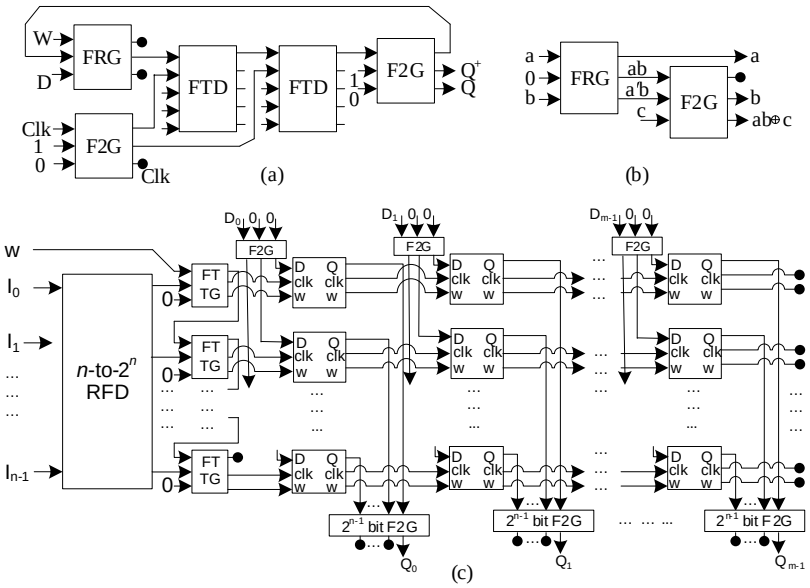

Fig. 7: Proposed architecture of the (a) RFMSFF (b) Fault tolerant realization of Toffoli gate (c) RFRAM

Design Procedure of the Proposed RFRAM of Plessey FPGA: We have proposed a reversible RAM in [9]. This design consists of reversible decoder, Toffoli gates, write enable master-slave flip-flops, 2 and 2^n bits Feynman gates. To design the RFRAM of Plessey FPGA, we extend the design [9] to reversible fault tolerant design. The extension is straight forward, we transformed each of the reversible component [9] to reversible fault tolerant components. An optimal design of reversible fault tolerant decoder is already proposed in Algorithm 1. By replacing Feynman gate of the design [9] with Feynman double gate and reversible MSFF with RFMSFF, we then convert the reversible Toffoli gate of [9] to fault tolerant Toffoli gate. Fig. 7(b) shows the architecture of a reversible **F**ault **T**olerant realization of **T**offoli **G**ate (FTTG). Block diagram of the proposed RFRAM of Plessey FPGA is shown in Fig. 7(c). Algorithm 3 demonstrates the design procedure of the proposed RFRAM of Plessey FPGA. From Fig. 7(c), we find that the proposed design demands an underneath RFD circuitry. Thus a call to RFD is specified. Complexity of this algorithm is $O(mn)$.

Algorithm 3: Algorithm for the proposed reversible fault tolerant RAM of Plessey FPGA, ***RFRAM(I, D, W, F2G, FRG)***

Input : Data $I(I_0, I_1, ..., I_{n-1})$, $D(D_0, D_1, ..., D_{m-1})$,$W$,$FRG$, $F2G$
Output: reversible fault tolerant RAM circuit of Plessey FPGA

1 **begin**
2 $i = input$, $o = output$
3 **call** *RFD(I, F2G, FRG)*
4 **for** $k \leftarrow 0$ **to** $n-1$ **do**
5 **if** $k = 0$ **then** $W \rightarrow first.i_k.FTTG$ **end if**
6 **else** $first.o_{k-1}.FTTG \rightarrow first.i_k.FTTG$ **end if**
7 $RFD.o_k \rightarrow second.i_k.FTTG$, $0 \rightarrow Third.i_k.FTTG$
8 **end for**
9 **for** $k \leftarrow 0$ **to** $m-1$ **do**
10 $D_k \rightarrow first.i_k.F2G$, $0 \rightarrow second$ & $third.i_k.F2G$
11 **for** $l \leftarrow 1$ **to** $n/2$ **do**
12 $F2G.o_k \rightarrow first.i_l.F2G$, $0 \rightarrow second$ & $third.i_l.F2G$
13 **end for**
14 **for** $l \leftarrow 0$ **to** $n-1$ **do**
15 $D_k(F2G.o_k l) \rightarrow first.RFMSFF.i$
16 $x = second$ & $third$
17 **if** $l = 0$ **then** $x.FTTG.o_l \rightarrow x.RFMSFF.i_l$ **end if**
18 **else** $x.RFMSFF.o_{l-1} \rightarrow x.RFMSFF.i_l$ **end if**
19 $first.RFMSFF.o.1 \rightarrow first.2^{n-1}bitF2G.i$
20 **end for**
21 **end for**
22 **return**
23 **for** $j \leftarrow 0$ **to** $m-1$ **do**
24 $2^{n-1}bitF2G.o.2^{n-2} \rightarrow$ *desired output*
25 **end for**
26 *all remaining* $F2G.o, FRG.o$ & $2^{n-1}bitF2G.o \rightarrow$ *garbage output*
27 **end**

4.4 Design Procedure of the Proposed Reversible Fault Tolerant Plessey FPGA

Fig. 8 shows the block diagram of the proposed Plessey logic block of FPGA. This architecture consists of the proposed reversible fault tolerant D-latch (Sec.4.1), RFMUX (Subsec.4.2), RFRAM (Sec.4.3) and an extra Fredkin and Feynman double gates. These two extra gates are needed to implement the NAND unit of Plessey FPGA (referring to Fig.3(b)). Algorithm 4 presents a design method of the proposed reversible fault tolerant Plessey FPGA. The algorithm considerably depends on RFRAM. The outputs of RFRAM are the selection bits for RFMux and the output of RFMux are used as the inputs to the NAND unit. Line 6 returns the primary output Q which actually is the input to the next logic block. Finally, line 7 returns all the garbage outputs of a reversible fault tolerant

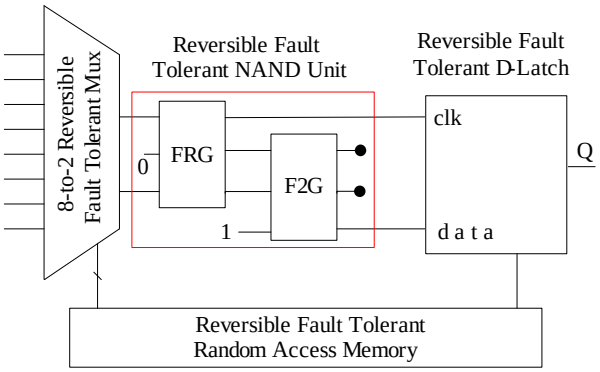

Fig. 8: Proposed Reversible Fault Tolerant FPGA

Plessey logic block for FPGA. As mentioned in Sec.1 and Sec.3, there are clusters of logic blocks in a FPGA, but the discussed procedure is capable to build a single reversible fault tolerant Plessey logic block. Thus, a recursive call has been declared in line 8 of Algorithm 4.

Algorithm 4: Algorithm for the proposed reversible fault tolerant logic block of Plessey FPGA

Input : Data input sets I, S, D, Write enable bit (W)
Fredkin gate (FRG) and Feynman double gate ($F2G$)
Output: Reversible fault tolerant Plessey FPGA circuit

1 **begin**
2 $i = input$, $o = output$
3 **call** $RFRAM(I, D, W, F2G, FRG)$
4 **call** $RFMux(I, S, F2G, FRG)$
where $S \leftarrow RFRAM.o$
5 $RFMux.o.1 \rightarrow FRG.i.1$, $0 \rightarrow FRG.i.1$, $RFMux.o.2 \rightarrow FRG.i.3$;
$FRG.o.2 \rightarrow F2G.i.1$, $FRG.o.3 \rightarrow F2G.i.2$, $1 \rightarrow F2G.i.2$;
$FRG.o.1 \rightarrow D - Latch.i.1$, $F2G.o.2 \rightarrow D - Latch.i.2$
6 **return** $Fourth.o.D - Latch \rightarrow Q$ *(input for next logic block)*
7 *remaining* $F2G.o$ *&* $FRG.o \rightarrow$ *garbage output*
8 ***if*** *this is not the last logic block*
goto *line 2*
9 **end**

4.5 Performance Evaluation of the Proposed Methods

There are two variables, *e.g.*, n and m, that fix the structure of the FPGA. Therefore, the comparative results are shown in contrast with three possible design approaches to prove the scalability of the proposed method:
(i) Varying m while keeping the constant n
(ii) Varying n while keeping the constant m
(iii) Varying both the m and n

Tables 2-5 show the performance of the proposed reversible fault tolerant scheme with the existing reversible one [8] in terms of numbers of gates, garbage outputs, constant inputs and quantum cost, respectively. In these tables, performance of the proposed method for case (i) can be found by looking at rows (1, 2) or (3, 4); rows (1, 3) or (2, 4) represent the performance of the proposed method for case (ii) and rows (1, 4) or (2, 4) represent the performance for case (iii). The results of Tables 2-5 show that the proposed scheme outperforms the existing work [8] by a high margin for all possible circumstances.

Table 2: Comparative study with respect to number of gates.

(n, m)	Proposed Design	Existing Design [8]
(8, 2)	417	4630
(8, 6)	661	12822
(16, 2)	65833	1179670
(16, 6)	66317	3276822

Table 3: Comparative study with respect to number of garbage outputs.

(n, m)	Proposed Design	Existing Design [8]
(8, 2)	286	4641
(8, 6)	770	12833
(16, 2)	33054	1179689
(16, 6)	99042	3276841

Table 4: Comparative study with respect to number constant inputs.

(n, m)	Proposed Design	Existing Design [8]
(8, 2)	428	713
(8, 6)	716	1065
(16, 2)	65860	131457
(16, 6)	66436	132161

Table 5: Comparative study with respect to quantum cost.

(n, m)	Proposed Design	Existing Design [8]
(8, 2)	1933	16000
(8, 6)	2957	47744
(16, 2)	361413	4063360
(16, 6)	428485	12189824

5 Conclusions

In this paper, we have presented the design methodologies of reversible fault tolerant Plessey FPGA. The fault tolerant designs of the proposed FPGA are first ever presented in the literature to our best knowledge. The proposed Plessey FPGA can be used in on-site hardware reconfiguration, logic emulation, network components, inexpensive prototype development, digital signal processing [6, 10] etc.

References

[1] Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**(3) (July 1961) 183–191

[2] Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**(6) (November 1973) 525–532

[3] Parhami, B.: Fault-tolerant reversible circuits. In: Fortieth Asilomar Conference on Signals, Systems and Computers. (2006) 1726 –1729

[4] Jamal, L., Shamsujjoha, M., Hasan Babu, H.M.: Design of optimal reversible carry look-ahead adder with optimal garbage and quantum cost. International Journal of Engineering and Technology **2** (2012) 44–50

[5] Biswas, A.K., Hasan, M.M., Chowdhury, A.R., Hasan Babu, H.M.: Efficient approaches for designing reversible binary coded decimal adders. Microelectron. J. **39**(12) (December 2008) 1693–1703

[6] Kuon, I., Tessier, R., Rose, J.: FPGA architecture: Survey and challenges. Found. Trends Electron. Des. Autom. **2** (February 2008) 135–253

[7] Kuon, I., Egier, A., Rose, J.: Design, layout and verification of an FPGA using automated tools. In: Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, New York, NY, USA, ACM (2005) 215–226

[8] Sayem, A.S.M., Polash, M.M.A., Babu, H.M.H.: Design of a reversible logic block of field programmable gate array. In: Silver Jubilee Conference on Communication Technologies and VLSI design, VIT University, Vellore, India (2009) 500–501

[9] Sharmin, F., Polash, M.M.A., Shamsujjoha, M., Jamal, L., Hasan Babu, H.M.: Design of a compact reversible random access memory. In: 4th IEEE International Conference on Computer Science and Information Technology. Volume 10., Chengdu, China (Jun. 2011) 103–107

[10] Voyiatzis, Gizopoulos, D., Paschalis, A.: Accumulator-based test generation for robust sequential fault testing in DSP cores in near-optimal time. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **14** (2005) 1079–1086