International Journal of Advanced Computer Research
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
Volume-5 Issue-18 March-2015

Design of reversible binary logarithmic multiplier and divider using optimal garbage

Arindam Banerjee*, Samayita Sarkar, Mainuck Das and Aniruddha Ghosh
Assistant Professor, Department of ECE, JIS College of Engineering, Kalyani, Nadia, W.B., India

Received: 27-January-2015; Revised: 16-March-2015; Accepted: 17-March-2015

Abstract

Energy efficient test-able binary logarithmic multiplier and divider architecture using reversible logic has been reported in this paper. The focus of this paper is to avoid multiplication and division stages to reduce large layout area and make the circuit efficiently test-able. Here all the computations have been performed in radix-2 basis. Moreover to avoid the usage of large number of constant inputs, an efficient technique has been adopted. Though the procedure demands errors in the result, efforts have been made to achieve higher accuracy.

Keywords

Logarithm, Antilogarithm, Addition, Subtraction, Logical Reversibility.

1. Introduction

Digital multipliers and dividers are indispensable in DSP processors and cryptography. There are so many well-known multiplication and division algorithms used in DSP, image processing and artificial neural network. One important aspect of improving the multiplier and divider efficiency is through the arrangement of adders and sub-tractors. Carry or borrow propagation is a major challenge in high speed VLSI circuit. Moreover conventional multiplication and division techniques provide larger and complex circuit layout and computational complexity.

*Author for correspondence

This research was performed under the CSIR sponsored project entitled "Synthesis and Testing of Reversible Circuits in application to Nano-technology and Quantum Computing" (Sanction No.- 22(0590)/12/EMRII) in the Department of CSE, Jadavpur University, Kolkata.

This issue can easily be resolved by using logarithmic processor. In addition to that in some complex DSP processors high speed achievement is prioritized than to get higher accuracy in results. To achieve faster operation and reduce the computational complexity and layout area, research has already been started on designing logarithmic processors [1], [2], [3], [4] from decades ago. Logarithmic processing technique has a wide range of application in Artificial Neural Network (ANN) [5].

As the number of transistors is increasing exponentially in VLSI circuits, the power consumption has become a great matter of concern. The situation is becoming worse for computation intensive logic circuits such as multipliers and dividers. To restrict large power consumption in designing VLSI circuits, logical reversibility is indispensable which was first proposed by Bennett [6]. Theoretically logical reversibility provides zero power consumption in the circuits. Therefore modern research is going on the reversible circuit synthesis [7], [8], and [9]. Though any Boolean function can be implemented by reversible logic, the problem is that this implementation may require large number of ancillary inputs and garbage outputs. Optimization of such parameters for an arbitrary reversible circuit is an open challenge to all the researchers. This paper proposes a test-able reversible logarithmic multiplier and divider using optimal ancillary inputs and garbage outputs. The numbers of ancillary inputs and garbage outputs have been greatly reduced using an efficient optimization technique. The quantum costs of the circuits have also been optimized by some novel techniques.

This paper has been organized in the following manner. Section 2 describes some basic well-known reversible gates. In Section 3 background mathematics of logarithmic multiplication and division technique has been explained elaborately.

Section 4 describes the architecture of logarithmic multiplier and divider. Section 5 describes the error calculation and accuracy checking of the algorithm. The optimization techniques used in the architecture has been elaborated in Section 6. The test-ability checking of the circuit has been discussed in Section 7. The optimized results have been tabulated in Section 8 and Section 9 is the conclusion.

2. Basic Reversible Gates and Circuits

Reversible gates are all n × n gates which have 1- to -1 mapping between input and output lines. The four basic gates shown in figure 1 that are used in arithmetic circuits are
(i) NOT gate or inverter - 1 × 1 circuit having one input and one output line,
(ii) CNOT (Controlled NOT) gate or Feynman Gate - 2 × 2 circuit having two input and two output lines,
(iii)Toffoli gate - n × n circuit (in figure-1 3×3 circuit) which performs the logical AND operation based on the target input,
(iv)Peres gate - 4 × 4 circuit consisting of double gate structure used for arithmetic addition based on the target line.

The above mentioned gates can be decomposed into some unitary quantum gates such as, (a) Controlled V gate and (b) Controlled V+ gate. Theoretically both the gates are considered to be the square root of CNOT gate. The decomposition is essential to calculate and optimize the quantum cost of the circuit. Theoretically they can be equated as,

$$V = \frac{(i+1)}{2}\begin{bmatrix} 1 & (-i) \\ (-i) & 1 \end{bmatrix} \quad (1)$$

Where $i = \sqrt{-1}$

$$V+ = \frac{(1-i)}{2}\begin{bmatrix} 1 & (+i) \\ (+i) & 1 \end{bmatrix} \quad (2)$$

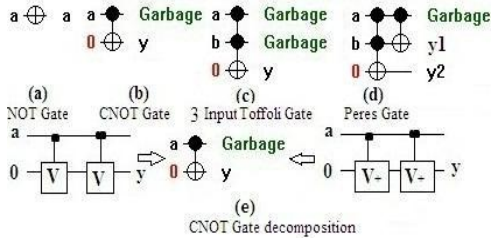

Figure 1: Basic Reversible gates

3. Description of Logarithmic Multiplication and Division Technique

A. Background Mathematics

Let us consider two 'n' bit binary numbers X and Y and $X \geq Y$. We need to determine the result P which is either $X \times Y$or $X \div Y$. Now X and Y can be expressed as,

$$X = 2^{k_1}\ (1 + m_1) \quad (3)$$

$$Y = 2^{k_2}\ (1 + m_2) \quad (4)$$

Where m_1 and m_2 can be expressed as,

$$m_1 = \sum_{i=1}^{k_1} x_i\, 2^{-i} \quad (5)$$

$$m_2 = \sum_{i=1}^{k_2} y_i\, 2^{-i} \quad (6)$$

It is suitable to call k_1 and k_2 as exponents or characteristics. So the logarithm of the result P can be expressed as,

$$\log_2 P = \log_2 X(Y)^{\pm 1} = \log_2 X \pm \log_2 Y \quad (7)$$

$$= \log_2[2^{k_1}(1 + m_1)] \pm \log_2[2^{k_2}(1 + m_2)] \quad (8)$$

$$= k_1 + \log_2(1 + m_1) \pm k_2 \pm \log_2(1 + m_2) \quad (9)$$

Here $k_1 \pm k_2$ is simply an addition or subtraction but $\log_2(1 + m_1) \pm \log_2(1 + m_2)$ is not at all a simple addition or subtraction. So the logarithmic portion of equation "(9)" can be expressed as,

$$\log_2(1 + m_1) \pm \log_2(1 + m_2) = \log_2[(1 + m_1)(1 + m_2)^{\pm 1}] \quad (10)$$

$$\cong (1 + m_1 \pm m_2 \pm m_1 \times m_2) \quad (11)$$

Here m_1 and m_2 are floating point binary numbers called mantissa. So $m_1 \times m_2$ must be a floating point number which is smaller than both $m_1$andm_2.

$$2^{m_1 \pm m_2 \pm m_1 \times m_2} = (1 + 1)^{m_1 \pm m_2 \pm m_1 \times m_2} \quad (12)$$

$$\cong 1 + m_1 \pm m_2 \pm m_1 \times m_2 \quad (13)$$

Here we are neglecting the term $m_1 \times m_2$. Therefore,

$$\log_2(1 + m_1 \pm m_2 \pm m_1 \times m_2) = m_1 \pm m_2 (14)$$

So equation (9) can be rewritten as,

$$\log_2 P = k_1 \pm k_2 + (m_1 \pm m_2) \quad (15)$$

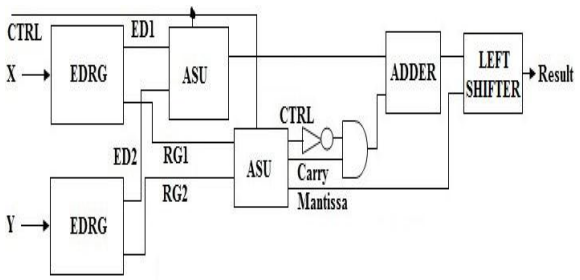

Figure 2: Schematic Diagram for Logarithmic Multiplier and Divider

To achieve higher accuracy equation "(15)" can be slightly modified. If $(m_1 \pm m_2) \geq 1$ then $\log_2 P = k_1 \pm k_2 + 1 + (m_1 \pm m_2 - 1)$ else $\log_2 P = k_1 \pm k_2 + (m_1 \pm m_2)$. Therefore, If $(m_1 \pm m_2) \geq 1$ then

$$P \cong 2^{(k_1 \pm k_2 + 1)} \times (m_1 \pm m_2 - 1) \quad (16)$$

else

$$P \cong 2^{(k_1 \pm k_2)} \times (m_1 \pm m_2) \quad (17)$$

B. Example of Binary to Logarithmic Number and Logarithmic to Binary Number Conversion

Binary to Logarithmic Number Conversion: -Assume a 16bit number X = 0110011100110101. (For the sake of simplicity we assume a 16 bit number; higher number of bit can be implemented using same manner). For this number, the highest power of 2 is 14_{10} which is the degree of the first non-zero bit. So, k = $(14)_{10}$ = $(1110)_2$ and m = $(1001110011010100)_2$.So the result Y = $\log_2$ X = $(1110.1001110011010100)_2$, which is a total of 20 bit.

Logarithmic to Binary Number Conversion: -Consider a 20 bit hybrid number Y = $(1110.1001110011010100)_2$(Logarithmic value of a number is given) whose antilogarithm is to be determined. For this number, exponent is E_x = $(1110)_2$= $(14)_{10}$ and mantissa is p = $(0.1001110011010100)_2$. Now place 1 at the position of $(14)_{10}$and shift the 16 bit mantissa towards left by $(14)_{10}$ times and the newly generated number is $(1100111001101010.0)_2$. Therefore antilogarithm of the number is equal to $(1100111001101010.0)_2$

4. Architecture of Logarithmic Multiplier and Divider

In this section, the proposed logarithmic multiplier and divider architecture has been described elaborately. The proposed architecture shown in figure 2 consists of three basic building blocks: (i) Exponent Determinant and Residue generator (EDRG), (ii) Adder/Sub-tractor (ASU), and (iii) Left Shifter (LS).

A. Exponent Determinant and Residue Generator (EDRG)

This is the main conversion block to extract the exponent and mantissa part of the binary input. Figure 3 shows the architecture for EDRG used here. The terms shown in equations "(3)" and "(4)" i.e. k_1, k_2, m_1 and m_2 are extracted using EDRG block. Here k_1 and k_2 known as exponents are the highest degrees of the binary numbers. The first part of figure4 is the exponent determinant. Here y_0 and y_1 are the outputs of exponent determinant. The circuit is modular and extensible for any number of bit streams. Residue Generator is used to extract the residues (m_1 and m_2) i.e. the floating point or mantissa part of the logarithmfrom the input. The output of exponent determinant is necessary to extract the mantissa. The second part of figure 3 is the residue generator. Here $l_0$to l_3 are the outputs of residue generator. The circuit is also modular and extensible for any number of bit streams.

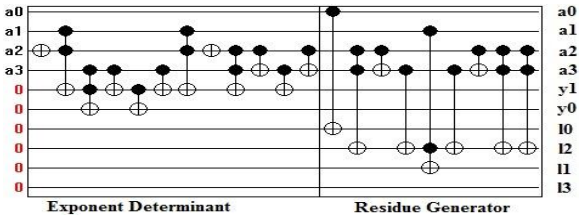

Figure 3: Architecture for Four bit Exponent Determinant and Residue Generator

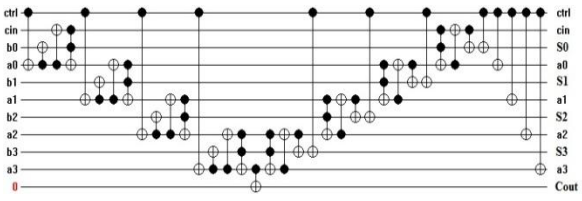

Figure 4: **Optimized Adder/Subtractor Circuit using only one constant input**

B. Addition/Subtraction Unit (ASU)

ASU is required to add or subtract the two exponents as well as two mantissas generated from EDRG. Figure 4 shows the optimized circuit for 4 bit ASU.

$$Result = a \oplus b \oplus c_{in} \quad (18)$$

$$k = ctrl \oplus a\ (say) \quad (19)$$

$$CB_{out} = k \oplus (k \oplus b)(k \oplus c_{in}) \quad (20)$$

The Boolean expressions shown in equations "(18)" and "(20)" are required to obtain the result and the carry/borrow output of the ASU. In this architecture we require three ASU blocks shown in figure 2. There are three operands m_1 and $m_2$shown in equations "(16)" and "(17)" which are added or subtracted by two ASU blocks. The third ASU is operating as an adder only to add the exponents (k_1, k_2) and the carry generated from the second ASU (in case of multiplication only). In figure 2, there is an AND gate followed by an Inverter which are required to put the carry from the second ASU to the adder in case of multiplication only. In case of multiplication 'CTRL' signal shown in figure 2 is low. Then the output of the AND gate is the carry output generated from the second ASU.

C. Left Shifter (LS)

This module is essential to achieve the antilogarithm of the result. Figure 5 shows the hardware implementation of LS in reversible logic. From equation "(16)" and "(17)", it is obvious that to retrieve P, we need to use the antilogarithm operation. From the exponential part shown in equation "(16)" and "(17)", the MSB of the result is set to the proper position and the remaining positions at the right hand side of the MSB are occupied by the floating point result. In figure 5 single bit shifting module which is elaborated in figure 6 has been used repetitively. In figure 5 we have shown 8 bit left shifter but it can be extended for large number of bits also. We have to keep in our mind that only one ancillary input is required for single bit shifting shown in figure 6 and the other ancillary inputs are bypassed for the next stage of operation which is implied and not shown in figure 5.

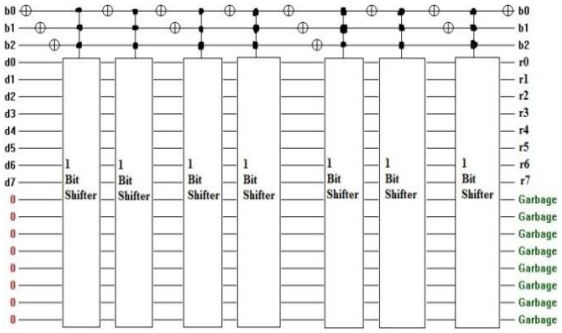

Figure 5: Circuit for 8bit left shifter

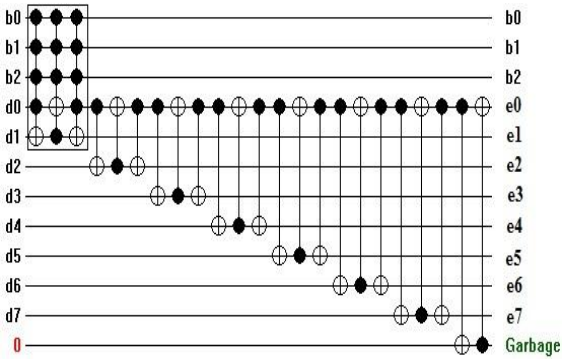

Figure 6: Circuit for Single bit left shifter

5. Error Calculation and Accuracy Checking

From equation (11) and (12) we get,

$$\log_2(1 + L) \cong L \quad (21)$$

where, $L = m_1 \pm m_2$. Now let us consider the error in calculation E can be expressed as,

$$E = \log_2(1 + L) - L \quad (22)$$

$$\frac{dE}{dL} = \frac{\log_2 e}{1+L} - 1 = \frac{1.442}{1+L} - 1 \quad (23)$$

$$\frac{d^2E}{dL^2} = \frac{1.442}{(1+L)^2} \quad (24)$$

From equations "(23)" and "(24)", it is clear that the error is minimized for small values of L and the minimum error occurs at L = 0.442. If $L = m_1 \pm m_2 \geq 1$, then error will increase. Table 1 shows the error in calculation based on the proposed algorithm.

Table 1: Error Deviation in Proposed Algorithm

Operand Value	Achieved Result	Exact Result	Error	Error Deviation
1.5	0.375	0.405	0.030	7.40%
1.75	0.468	0.559	0.090	16.27%
2.00	0.500	0.693	0.193	27.84%
2.25	0.531	0.810	0.279	34.44%
2.5	0.938	0.916	0.022	2.40%
3.5	1.281	1.252	0.144	2.31%
3.75	1.293	1.321	0.028	2.11%
4.5	1.460	1.504	0.044	2.92%
			Average	11.96%

6. Optimization of the Circuit

In reversible logic, there is always a trade-off between quantum cost, ancillary inputs and garbage outputs. Therefore our goal is to reduce the ancillary inputs and garbage outputs with a permissible increment in quantum cost. The design mentioned in Section 4 (particularly EDRG and ASU blocks) require many ancillary inputs and garbage outputs. Here we propose a technique to decrease these two parameters. In this technique few well known reversible gates are added to the circuits obtained in section 4 to make one constant input free to be reused later and thereby the number of ancillary inputs and garbage outputs are drastically reduced. This optimization technique is known as circuit duplication.

Optimization in ancillary inputs and garbage outputs:
Data: number of line, target reused
Whilenumberofline≠ 0 **do**
Scan the current target line;
Iftargetreused = true **then**
Number of line=number of line-1;
Go to next target line;
else
Use circuit duplication onto that target line;
number of line=number of line-1;
Go to next target line;
end
end

Algorithm 1: Algorithm for Circuit Duplication Technique

In this technique the target lines of the circuits obtained in Section 4 are scanned one by one. If one target line is not reused later again then the gates are duplicated on that target line to restore the original value. This line can be reused as one ancillary input for the future operation. So one ancillary input is reduced from the input side. Algorithm-1 shows the algorithm for circuit duplication technique.

Example: In the reversible circuit shown in figure 7(a), the number of ancillary inputs and garbage outputs are 3 and 6 respectively, which are reduced to 2 and 5 respectively in figure 7(b) by inserting a reversible circuit shown by the rectangular box. Notice that we could do this optimization, because the 1st ancillary input in figure 7(a) has the option to be reused.

Optimization in quantum cost: To decrease the quantum cost, we have adopted the decomposition technique of the reversible gates using NCV-v_1>[8] plus double gate library. Figure 8 shows a XOR-OR circuit used in EDRG block shown in figure 3. The circuit has been decomposed into few V, V+ and CNOT gates and then the adjacent V and V+ gates have been eliminated and finally the reduced quantum cost is 4 instead of 5.

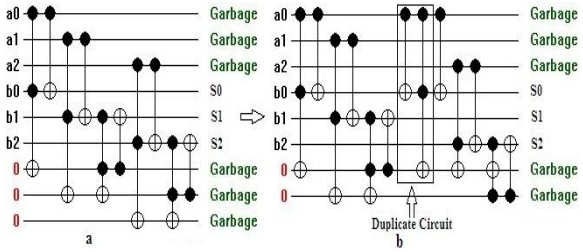

Figure 7: Architecture for duplicate circuit

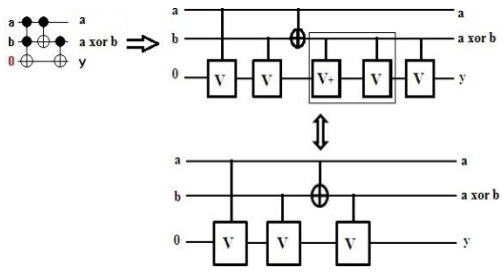

Figure 8: Quantum Cost reduction for XOR-OR Circuit

Figure 9 shows an adder circuit with optimal ancillary and garbage lines. This circuit has been used in ASU block shown in figure 4. At first the circuit has been partitioned into two parts (Part-I and Part-II). Then each part has been decomposed into V, V+ and CNOT gates. Apparently, the quantum cost of the circuit should be equal to 1+1+5+1+5+1+1 = 15. But we have shown that the transformation matrix of the circuit indicated by a box in figure 9(d) and (e) is a 4 × 4 unitary matrix for which its quantum cost is 1. The proof is given by following equations. We know the transformation matrix of a CNOT gate is as follows,

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (25)$$

Again the transformation matrix of a Controlled V gate is

$$CV = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix} \qquad (26)$$

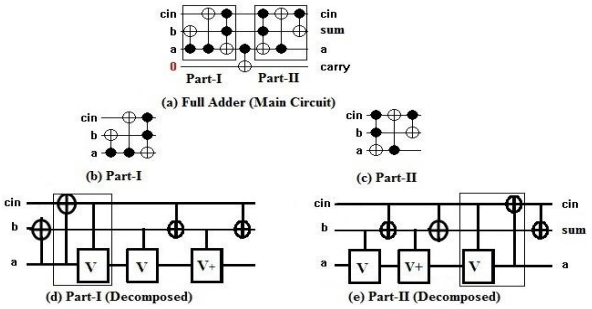

Figure 9: Quantum Cost reduction for the Adder with optimal garbage

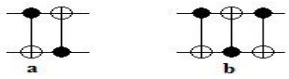

Figure 10: Cascaded CNOT gates for SWAP gate

So the transformation matrix of the circuit indicated by a rectangular box in figure 9(d) is given as,

$$CASCADE = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 \\ 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 \end{bmatrix} \qquad (27)$$

It can be easily be proved mathematically that the matrix shown in equation "(27)" is unitary and also follows a symmetric pattern with respect to the operation i.e. one of the two gates disappears virtually at exactly two different input combinations. So the circuit must have unit quantum cost. Also, we have verified the cost with the help of the technique discussed in [10]. Similar approach is applicable for figure 9(e). Therefore using circuit decomposition the cost has been reduced by 2 for a single stage of addition and the reduced cost becomes 6 + 6 + 1 = 13 instead of 15.The quantum cost of the LS shown in figure 5 and 6 has been drastically reduced by the following technique. The quantum cost of the circuit shown in figure 6 (indicated by a rectangular box) has been optimized by the technique shown by Soeken et. al. [8] using NCV v_1> plus double gate library. They have shown that for an n-controlled Toffoli gate, the quantum cost is 2 × n+1. But if we have m number of cascaded n-controlled Toffoli gates in which k number of control lines are common then the reduced quantum cost becomes m × (2 × n+1) – 2 ×(m - 1) × k. In figure 6 there are three 4-controlled Toffoli gates and three control lines are common. So the quantum cost of this part of the whole circuit is 3(2 × 4 + 1) - 2× 2 ×3 = 27 - 12 = 15 instead of 3(2 × 4 + 1) = 27. Figure 10 shows the cascade connection of two and three CNOT gates which have been used in figure 6. Apparently the quantum cost of the circuit shown in figure 6 is 2+7×3 = 23. Now if we study the transformation matrices of the gates then the idea is little bit changed. The transformation matrix of a CNOT gate is shown in equation "(27)". Now the transformation matrix of the circuit shown in figure 10(a) is,

$$CASCADE_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (28)$$

And similarly the transformation matrix of the circuit shown in figure 10(b) is,

$$CASCADE_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (29)$$

Mathematically it can be easily be shown that the matrices shown in equation "(28)" and "(29)" are unitary and thereby produce unit quantum cost and the total cost of the circuit is1 × 8 = 8 though the two gates do not follow any symmetric pattern discussed earlier. So the reduced quantum cost of single bit shifter is equal to 15 + 8 = 23 instead of 27 + 23 = 50. Therefore we are getting a significant reduction in quantum cost for the left shifter.

7. Test-Ability Checking of the Circuit

In this paper we have not only designed logarithmic multiplier and divider but we have also tested the whole architecture module wise. In this paper, transformation based fault detection technique has been introduced.

A. Transformation Based Fault Detection Technique

We have used two algorithms for the transformation based fault detection. The first algorithm has been developed to generate the transformation matrix of any reversible circuit and the second algorithm is for the generation of test pattern from the transformation matrices of the exact circuit and the faulty circuit. **Algorithm-2** has been developed to generate the transformation matrix of the exact circuit and the faulty circuit. Here we are creating an image of the exact circuit i.e. flipping the circuit horizontally by 180^0. Then the output of the exact circuit will be the input of the image circuit and the input will be the output. Now the matrices of the exact image circuit as well as the faulty image circuit are generated.

The test pattern of the circuit can easily be generated using **Algorithm-3**. Here TR_k and TR_{k+1} are the transformation matrices of the exact image circuit and the faulty image circuit respectively. Using this technique we have generated the test pattern of all the circuit modules.

Data:input bit pattern, output bit pattern,tmp
Input :maxrow,maxcolumn
Output:matrixTR_k[maxrow,maxcolumn]
Take the mirror image of the circuit;
for i ← 0 tomaxrow- 1 **do**
inputbitpattern = decimal to binary(i);
outputbitpattern = image circuit(inputbitpattern);
tmp = binary to decimal(output bit pattern);
for j← 0 to maxcolumn- 1 **do**
if j = tmp**then**
TR_k[i, j] = 1;
else
TR_k[i, j] = 0;
end
end
end

Algorithm 2: Algorithm for Transformation Matrix Generation

Example: Using the above mentioned technique we have generated the test pattern of the test circuit shown in figure 11.The transformation matrix for figure 11(a) is,

$$TR_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (30)$$

Data:tmp
Input :maxrow,maxcolumn,matrix
TR_k[maxrow,maxcolumn],matrix
TR_{k+1}[maxrow,maxcolumn]
Output:matrix pattern[maxcolumn, $\log_2$maxcolumn]
for i ←0 to maxrow- 1 **do**
for j ← 0 to maxcolumn- 1 **do**
ifTR_k[i, j] ≠TR_{k+1}[i, j] **then**
tmp = j;
for*l* ← 0 **to** $\log_2$(maxcolumn) - 1 **do**
pattern[i,*l*] = decimaltobinary(tmp);
end
end
end
end

Algorithm 3: Algorithm for Test Pattern Generation

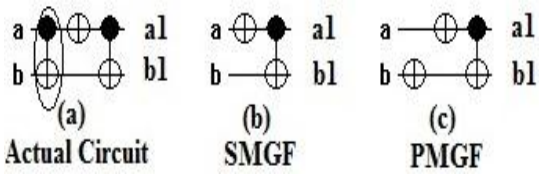

Figure 11: Test Circuit on which transformation based technique applied

The transformation matrix for figure 11(b) is,

$$TR_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (31)$$

The transformation matrix for figure 11(c) is,

$$TR_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (32)$$

From equations "(30)" and "(31)" we get the change of value in column number 2 and 3. So the test pattern (10 or 11) is sufficient to detect the SMGF fault occurred to the first gate of figure 11(a). Similarly from equations "(30)" and "(32)" the change in value is observed in column number 0 and 1. So the test pattern (00 or 01) is sufficient to detect the PMGF fault occurred to the first gate of figure 11(a).

B. Testing of EDRG

Table 2: Test Pattern for EDRG

Tests	a_3	a_2	a_1	a_0
T_1	0	0	1	0
T_2	0	1	0	1
T_3	1	0	0	0
T_4	1	0	1	0
T_5	1	1	1	1

From Table 2, we get five tests which are optimal to test all SMGF (Single Missing Gate Fault) and PMGF (Partial Missing Gate Fault).

Table 3: Test Pattern for a Single Bit ASU

Tests	a	b	c_i	ctrl
T_1	0	1	0	0
T_2	0	1	1	0
T_3	1	1	0	0

Table 4: Test Pattern for a Three Bit ASU

Tests	a_0	b_0	c_i	ctrl	a_1	b_1	ctrl	a_2	b_2	ctrl
T_1	0	1	0	0	0	1	0	0	1	0
T_2	0	1	1	0	0	1	0	0	1	0
T_3	1	1	0	0	0	0	0	1	1	0
T_4	0	0	1	0	1	1	0	0	0	0

C. Testing of ASU

Table 3 shows the three tests which are optimal to detect all SMGF and PMGF faults for a single bit ASU whereas Table 4 shows four tests to detect all SMGF and PMGF in a two bit ASU though the Table can be extended for n bit and the patterns are repetitive in nature.

D. Testing of LS

Table 5: Test Pattern for LS

Tests	b_0	b_1	b_2	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7
T_1	0	0	0	1	1	0	0	0	0	0	0
T_2	0	0	1	0	0	0	0	0	0	0	0
T_3	0	1	0	0	0	0	0	0	0	0	0
T_4	1	0	0	0	0	0	0	0	0	0	0
T_5	1	1	1	0	1	0	1	0	1	0	1
T_6	1	1	1	1	0	1	0	1	0	1	0
T_7	1	1	1	1	1	1	1	1	1	1	1

Table 5 shows 7 patterns to detect all SMGF and PMGF in the LS block shown in figure 6. Since it is nothing but a shifting circuit so we require testing all nodes of the multiple controlled Toffoli gates and thus we get the 7 tests which are essential for the circuit.

8. Result Analysis

From Table 6 and Table 7, we observe a significant amount of reduction of ancillary inputs and garbage outputs. Though the quantum cost has been increased but it exists in the tolerable limit. From the two Tables it is observed that all the parameter values for

4 bit architecture are almost getting doubled in case of 8 bit architecture. Therefore we have to observe that for 4n bit architecture, all the parameter values are approximately increased by n times or not. Moreover our future aim is to achieve success in designing higher order logarithmic circuit with optimal ancillary inputs and garbage outputs.

Table 6: Tabular Form of Calculated Parameters for Four bit Logarithmic Multiplication and Division

parameters	Without optimization	With optimization
Ancillary Input	60	28
Garbage Output	40	10
Quantum Cost	438	458

Table 7: Tabular Form of Calculated Parameters for Eight Bit Logarithmic Multiplication and Division

parameters	Without optimization	With optimization
Ancillary Input	112	52
Garbage Output	87	27
Quantum Cost	754	914

9. Conclusion

In this paper our aim is to exhibit multiplication and division technique using Logarithmic Number system. The technique is faster than any other technique for multiplication and division though the result produces some errors. This circuit is very helpful for the system where faster operation is prioritized than to achieve high accuracy in computation. Method of circuit duplication has been introduced to achieve the reduction in input and output pins which can drastically reduce the fabrication cost. Moreover circuit decomposition technique has been used to reduce the quantum cost of the circuits. Finally efforts have been given to make the circuits easily and efficiently test-able.

Acknowledgment

This research was performed under the CSIR sponsored project entitled “Synthesis and Testing of Reversible Circuits in application to Nanotechnology and Quantum Computing” (Sanction No.-22(0590)/12/EMRII) in the Department of CSE, Jadavpur University, Kolkata. We are heartily grateful to Professor Debesh Kumar Das, Dept. of CSE, Jadavpur University, for his precious cooperation and technical support.

References

[1] J. N. Mitchell, “Computer multiplication and division using binary logarithms”, IRE Transactions on Electronic Computers, 1962, pp.512–517.
[2] F. J. Taylor, R. Gill, J. Joseph, and J. Radke, “A 20 bit logarithmic number system processor”, IEEE Transactions on Computers, vol. 37, February 1988, pp. 190–200.
[3] P. Bulic, Z. Babic, and A. Avramovic, “A simple pipelined logarithmic multiplier”,IEEE International Conference on Computer Design, October 2010, pp. 235–240.
[4] E. L. Hall, D. D. Lynch, and S. J. Dwyer, “Generation of products and quotients using approximate binary logarithms for digital filtering applications”, IEEE Transactions on Computers, vol.C-19, February 1970, pp.97–105.
[5] U. Lotric and P. Bulic, “Logarithmic multiplier in hardware implementation of neural networks”, ICANNGA-2011, 2011, pp. 158–168.
[6] C. H. Bennett, “Logical reversibility of computation”, IBM Journal of Research and Development, vol. 17, November 1973, pp. 525–532.
[7] R. Wille, “Synthesis of reversible circuits”, 4th Workshop on Reversible Computation, July 2012.
[8] M. Soeken, Z. Sasanian, R. Wille, D. M. Miller, and R. Drechsler, “Optimizing the mapping of reversible circuits to four-valued quantum gate circuits,” in ISMVL 2012, 2012, pp. 173–178.
[9] H. B. Axelsen and M. K. Thomsen, “Garbage-free integer multiplication with constants”, 4th Workshop on Reversible Computation, July 2012.
[10] W. N. N. Hung, X. Song, G. Yang, and M. Perkowski, “Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, September 2006.

Arindam Banerjee is an Asst. Prof. in ECE Dept., JIS College of Engineering, Kalyani, Nadia, W.B, India. A. Banerjee finished his M.Tech from WBUT, WB, India in 2008. His area of specialization is Microelectronics and VLSI design. His areas of interest are Signal processing, Image processing and embedded system design.
Email: banerjee.arindam1@gmail.com.
.

Samayita Sarkar is an Asst. Prof., ECE Dept., JIS College of Engineering, Kalyani, Nadia, W.B, India. She has done her M.Tech in Microelectroninics and VLSI Design from WBUT, West Bengal. Her area of interest is VLSI Design.

Mainuck Das, Asst. Prof., ECE Dept., JIS College of Engineering, Kalyani, Nadia, W.B, India. His area of specialization is Microelectronics and VLSI design His areas of interest are Signal processing, Image processing and embedded system design.

Aniruddha Ghosh, Asst. Prof., ECE Dept., JIS College of Engineering, Kalyani, Nadia, W.B, India.A. Ghosh finished his M.Tech from Calcutta University, WB, India in 2011. His area of specialization is Radio-physics & Electronics. His areas of interest are Signal processing and Digital Communication.