International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE) Vol 3, Issue 5, May 2016

Design and Implementation of 4x4 Pipelined Iterative Logarithmic Multiplier using Reversible Logic

[1] Kavya Shree M S [2] Praveen Kumar Y G [3]Dr. M Z Kurian
[1] 14thsem, M.Tech (VLSI and Embedded Systems) [2] Assistant Professor, [3]HOD
Dept.of ECE, Sri Siddhartha Institute of Technology, Tumakuru
[1] kavyabhramara5@gmail.com

***Abstract*— As the demand increases for low power dissipation in digital computing system, a new technique called Reversible Logic was introduced. Reversible logic is one of the promising field which solves the problem of power dissipation and also it is the basic requirement for the field of quantum computing. The multiplication which plays a prior role in DSP applications. Some of the important operations in DSP are filtering, convolution and inner partial products. These are the processes which requires multipliers so the speed and performance of their operations depends on the speed of the multiplication and addition. The logarithmic multiplier which is designed based on the Mitchell's Algorithm proposed by Mitchell. The logarithmic multiplier convert multiplication and division problem into addition and subtraction.. This paper gives the design of pipelined iterative logarithmic multiplier using reversible logic.**

Key Words: **Reversible Logic, quantum Computing, Logarithmic Multiplier, Mitchell's Algorithm**.

I. INTRODUCTION

Reversible Logic has received great attention in the recent years due to their ability to reduce the power dissipation which is the main requirement in the low power VLSI design. Reversible logic is an n-input n-output logic device with one to one mapping. In the designing of the reversible logic circuits direct fan-out is not allowed and the concept of one-to-many is not entertained.

Digital arithmetic calculations are most important in the design of digital processors and application-specific systems. Arithmetic circuits plays an important class of circuits in the digital system. Multiplication is relevant since other arithmetic operators such as division or exponentiation, which they usually utilize multipliers as building blocks. Logarithmic multiplier converts given binary numbers into their equivalent logarithm numbers. In LNS (Logarithmic Number System) multiplication is done interms of addition.

II. DESIGN OF ALGORITHM

Logarithmic multiplication is an approximate multiplication technique that uses the fact that logarithm of the product is a sum of operand logarithms; therefore an operand conversion from integer number system into the logarithm number system (LNS) is used. The multiplication of the two operands N_1 and $N_2$is performed in three phases, calculating the operand logarithms, the addition of the operand logarithms and the calculation of the antilogarithm:

$$\mathbf{Log}\ (N_1N_2) = \mathbf{log}\ (N_1)\ .\mathbf{Log}\ (N_2)$$

The main advantage of this method is the substitution of the multiplication with addition. LNS multipliers can be generally divided into two categories, one based on methods that use lookup tables-where the log values are stored in memory tables and the pure LUT stores a (pre-calculated) value for the logarithm of every possible input value. The logarithm is obtained by looking for its value in the table. The LUT and Interpolator use a similar table to the LUT, but do not store every value. Instead it stores half, for example, and uses linear interpolation to estimate the values between the look-up points and the other based on Mitchell's algorithm (MA).

The binary representation of the number N can be written as:

$$N= 2^k\ (1+\sum_{i=j}^{k-1} 2^{i-k} z_i) = 2^k\ (1+x)\ldots\ldots\ (1)$$

Where,

K is place of the most significant bit which equals one, so called characteristic number,

z_i is a bit value at the ith position,

x is the fraction or mantissa,
j depends on the number's precision.

Because, computers works with binary number system, it is more appropriate to use 2 as logarithm basis, so it can be derive as:

$$\log_2(N)=\log_2\left(2^k\left(1+\sum_{i=j}^{k-1} 2^{i-k} z^i\right)\right) = \log_2\left(2^k(1+x)\right)$$
$$=k+\log_2(1+x) \ldots\ldots \quad (2)$$

The expression $\log_2(1 + x)$ is usually approximated; i.e, logarithmic based calculations are a trade-off between the accuracy and time consumption.

a. *Mitchell's Algorithm*

One of the most relevant multiplication methods in LNS is Mitchell's algorithm. Mitchell proposed a very low-cost approach to approximating the base-2 logarithm and antilogarithm without the need for a table or iteration, hence reducing hardware area. Mitchell's technique is accurate to about four bits. There are more accurate techniques but none are as economical as Mitchell's method.

An approximation of the logarithm and the antilogarithm is essential, and it is derived from a binary representation of the numbers.

The logarithm of the product is:

$$\log_2(N_1N_2)= k_1+k_2+\log_2(1+x_1) + \log_2(1+x_2) \ldots . (3)$$

The expression $\log_2(1 + x)$ is approximated with x and the logarithm of the two numbers' product is expressed as the sum of their characteristic numbers and mantissas:

$$\log_2(N_1N_2)\approx k_1+k_2+x_1+x_2 \quad \ldots\ldots(4)$$

The characteristic numbers k_1 and k_2 represent the places of the most significant operands' bits with the value of '1'. The final MA approximation for the multiplication (where $P_{true} = N_1 \times N_2$) depends on the carry bit from the sum of the mantissas and is given by:

$$P_{MA} = (N_1N_2)_{MA} = \begin{cases} 2^{k1+k2}(1+x_1+x_2), & x_1+x_2 < 1 \\ 2^{k1+k2+1}(x_1+x_2), & x_1+x_2 \geq 1 \end{cases} \quad (5)$$

If $x_1 + x_2 < 1$, the sum of mantissas is added to the most significant bit of product to complete the final result. Otherwise, the product is approximated only with the scaled sum of mantissas. The MA produces a significant error percentage. The maximum possible relative error for MA multiplication is around 11%, and the average error is around 3.8% .The error in MA is always positive so it can be reduced by successive multiplications.

Mitchell analyzed this error and proposed the following analytical expression for the error correction

$$(N_1.N_2)_{MAC}=P_{MA}= \begin{cases} p_{MA}+2^{k1+k2}(x_1.x_2), & x_1+x_2 < 1 \\ 2^{k1+k2}(1-x_1) . (1-x_2), & x_1+x_2 \geq 1 \end{cases} \quad (6)$$

Where, $2^{k1+k2}(x_1.x_2)$ and $2^{k1+k2}(1-x_1). (1-x_2)$ are the correction terms proposed by Mitchell.

To calculate the correction terms:

1. Calculate $x_1.x_2$ or $(1 - x_1) .(1 -x_2)$ depending on $x_1 + x_2$.
2. Scale the correction term by the factor 2^{k1+k2}.
3. Add the correction term to the product PMA.

Algorithm 1 (Mitchell's Algorithm)

```
1. N_1, N_2: n-bits binary multiplicands, P_MA = 0:2 n-bits approximate product
2. Calculate k_1: leading one position of N_1
3. Calculate k_2: leading one position of N_2
4. Calculate x_1: shift N_1 to the left byn - k_1bits
5. Calculate x_2: shift N_2 to the left byn - k_2 bits
6. Calculate k_12 = k_1 + k_2
7. Calculate x_12 = x_1 + x_2
8. IF x_12≥2^n (i.e. x_1 + x_2>=1):
   (a) Calculate k_12 = k_12 + 1
   (b) Decode k_12 and insert x_12 in that position of P_approx
ELSE:
   (a) Decode k_12 and insert '1' in that position of P_approx
   (b) Append x_12 immediately after this one in P_approx
9. Approximate N_1. N_2 = P_MA
```

One important observation (from Algorithm 1) is that the error correction can start only after the term $x_1 + x_2$ is calculated.

The proposed solution simplifies logarithm approximation and introduces an iterative algorithm with various possibilities for achieving the multiplication error as small as required and of achieving the exact result. High level of parallelism can be achieved by the principle of pipelining, thus increasing the speed of the multiplier with error representation of the numbers in (1), we can derive a correct expression for the multiplication:

$\mathbf{P_{true} = N_1 . N_2}$
$\mathbf{= 2^{k1} (1 + x_1) . 2^{k2}(1 + x_2)}$
$\mathbf{= 2^{k1+k2} (1 + x_1 + x_2) + 2^{k1+k2} (x_1 x_2)}$(7)

To avoid the approximation error, it would have to take into account the next relation derived from (1):

$\mathbf{x . 2^k = N - 2^k}$(8)

The combination of (7) and (8) gives:

$\mathbf{P_{true} = (N_1 . N_2)}$
$\mathbf{= 2^{k1+k2} + (N_1 - 2^{k1}) 2^{k2} + (N_2 - 2^{k2}) 2^{k1} + (N_1 - 2^{k1}).(N_2 - 2^{k2})}$
(9)

Let

$$\mathbf{P^{(0)}_{approx.} = 2^{k1+k2} + (N_1 - 2^{k1}) 2^{k2} + (N_2 - 2^{k2}) 2^{k1}}$$

be the first approximation of the product. It is evident that

$$\mathbf{P_{true} = P^{(0)}_{approx.} + (N_1 - 2^{k1}).(N_2 - 2^{k2})} \qquad (11)$$

The proposed method is very similar to MA. The error is caused by ignoring the second term in (11). The term $(N_1 - 2^{k1})$ $(N_2 - 2^{k2})$ requires multiplication. The difference between the proposed method and the method proposed by Mitchell is that the proposed method avoids the comparison of the addend $x_1 + x_2$ with 1. Hence the error correction can start immediately after removing the leading ones form the both input operands.

The absolute error after the first approximation is:

$$\mathbf{E^{(0)} = P_{true} - P^{(0)}_{approx.} = (N_1 - 2^{k1}).(N_2 - 2^{k2})} \qquad (12)$$

Note that $E^{(0)} >= 0$. The two multiplicands in (6) are binary numbers that can be obtained simply by removing the leading '1' in the numbers N_1 and N_2 so repeat the proposed multiplication procedure with these new multiplicands.

$$\mathbf{E^{(0)} = C^{(1)} + E^{(1)}}\ldots\ldots(13)$$

WhereC $^{(1)}$ is the approximate value of E $^{(0)}$ and E $^{(1)}$ is an absolute error when approximating E $^{(0)}$.The combination of (11) and (13) gives

$$\mathbf{P_{true} = P^{(0)}_{approx.} + C^{(1)} + E^{(1)}} \qquad (14)$$

Now add the approximate value of E $^{(0)}$ to the approximate product P_{approx} as a correction term by which decreases the error of the approximation.

$$\mathbf{P^{(1)}_{approx.} = P^{(0)}_{approx.} + C^{(1)}} \qquad (15)$$

If repeat this multiplication procedure with i correction terms, it can approximate the product as:

$$\mathbf{P^{(i)}_{approx.} = P^{(0)}_{approx.} + C^{(1)} + C^{(2)} + \ldots + C^{(i)} = P^{(0)}_{approx} + \sum_{j=1}^{i} C^{(j)}} (16)$$

The procedure can be repeated, achieving an error as small as necessary, or until at least one of the residues becomes a zero. Then the final result is exact: $P_{approx} = P_{true}$. The number of iterations required for an exact result is equal to the number of bits with the value of '1' in the operand with the smaller number of bits with the value of '1'

Algorithm 2 (Iterative MA-based algorithm with i correction terms)

```
1. N1, N2: n-bits binary multiplicands, $P^{(0)}_{approx.} = 0$ : 2n-bits first approximation $C^{(i)}=0$:2n-bits i correction terms,
$P_{approx} = 0$: 2n-bits product
2. Calculate $k_1$: leading one position of $N_1$
3. Calculate $k_2$: leading one position of $N_2$
4. Calculate $(N_1-2^{k1})2^{k2}$: shift $(N_1-2^{k1})$ to the left by $k_2$ bits
5. Calculate $(N_2-2^{k2})2^{k1}$ : shift $(N_2-2^{k2})$ to the left by $k_1$ bit.
6. Calculate $k_{12} = k_1 + k_2$
7. Calculate $2^{k1+k2}$: decode $k_{12}$
8. Calculate $P^{(0)}_{approx}$: add $2^{k1+k2}$, $(N_1-2^{k1})2^{k2}$ and $(N_2-2^{k2})2^{k1}$
9. Repeat i-times or until $N_1 = 0$ or $N_2 = 0$:
(a) Set: $N_1 = N_1-2^{k1}$, $N_2=N_2-2^{k2}$
(b) Calculate $k_1$: leading one position of $N_1$
(c) Calculate $k_2$: leading one position of $N_2$
(d) Calculate $(N_1-2^{k1})2^{k2}$: shift $(N_1-2^{k1})$ to the left by $k_2$ bits
(e) Calculate $(N_2-2^{k2})2^{k1}$ : shift $(N_2-2^{k2})$ to the left by $k_1$ bits
(f) Calculate $k_{12} = k_1 + k_2$
(g) Calculate $2^{k1+k2}$: decode $k_{12}$
(h) Calculate $C^{(i)}$: add $2^{k1+k2}$, $(N_1-2^{k1})2^{k2}$and $(N_2-2^{k2})2^{k1}$
10.$P^{(i)}_{approx}=P^{(0)}_{approx}+\sum_i C^{(i)}$
```

III. III. SYSTEM DESIGN:

i. Reversible Logic:

A reversible logic gate is an n-input, n-output logic device with one-to-one mapping, which helps to retrieve the inputs from the outputs. Also in the synthesis of reversible circuits direct fan-out is not allowed as one-to-many concept is not reversible. However fan-out in reversible circuits is achieved using additional gates.

There are many parameters for determining the complexity and performance of circuits.

- ➢ The number of reversible gates (N): The number of reversible gates used in the circuits.

- The number of constant inputs (CI): The number of inputs that are maintained constant at either 0 or 1in order to synthesize the given logical function.
- The number of garbage outputs (GO): The number of unused outputs present in a reversible logic circuits.
- Quantum cost (QC): The quantum cost refers to the cost of the circuit in terms of primitive gates.In a reversible logic circuit design two restrictions should be maintain strictly.
- Fan-out is not allowed.
- Feedback loops are also prohibited.Logic synthesis of reversible logic circuits should have the following objectives to achieve optimized structure.
- Design should use minimum number of logic gates.
- Constant inputs should be minimum.
- Quantum cost should be kept as low as possible.

IV. PROPOSED SYSTEM DESIGN

A basic block (BB) is the proposed multiplier with no correction terms. It calculates one approximate product. The basic block is presented in Figure 1, which consists of two leading-one detectors (LODs), two encoders, two barrel shifters, and decode unit decodes $k_1 + k_2$, i.e. it puts the leading one in the product. The left shifters are used to shift the residues. The two shifted residues are then added to form the approximate product.

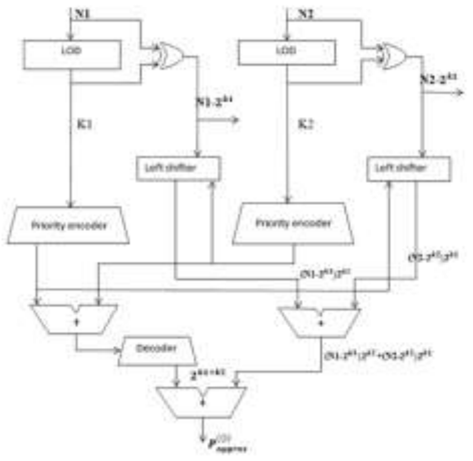

Figure 1: Basic Block Diagram of the Proposed Logarithmic Multiplier.

Implementation with Correction Circuits

The basic block is used in subsequent implementations to implement correction circuits and increase the accuracy of the multiplier. The error-correction circuit is used to calculate the term $C^{(1)}$ and thus approximates the term $(N_1 - 2^{k1}) (N_2 - 2^{k2})$.To implements the proposed multipliers, cascade the basic blocks. A block diagram of the proposed logarithmic multiplier with one error correction circuit is shown in Figure 2.The multiplier is composed of two basic blocks, of which the first one calculates the first approximation of the product $P^{(0)}_{approx.}$, while the second one calculates the error-correction term $C^{(1)}$.

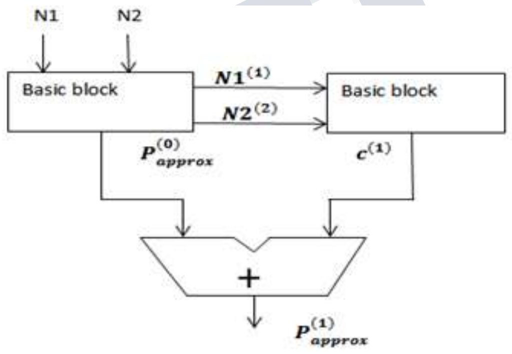

Figure 2:Block Diagram of the Proposed Multiplier with Error Correction Circuit.

The implementation with correction circuits shows substantial increase in combinational delay as each correction circuit is added. The two basic blocks cannot really work in parallel in real-time, because the correction block cannot start until the residues are calculated from the first basic block. But in the pipelined implementation of the basic block the residues are available after the first stage; the correction circuit can start to work immediately after the first stage from the prior block is finished.

V. IMPLEMENTATION OF PROPOSED MULTIPLIER

The RTL schematic of 4x4 reversible logarithmic multiplier without correction circuit is as shown in Figure 3.

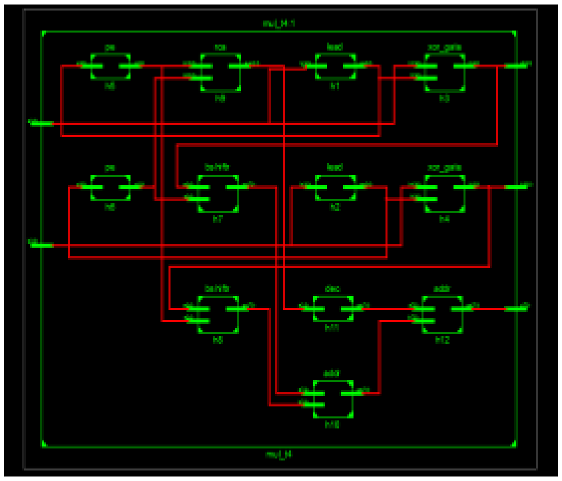

Figure 3: RTL Schematic of Reversible Log Multiplier Without Correction Terms.

The RTL schematic of 4x4 reversible logarithmic multiplier with correction circuit is as shown in Figure 4.

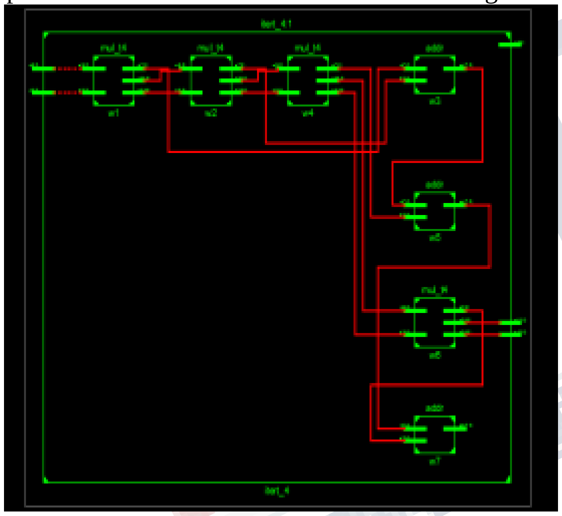

Figure 4: RTL Schematic of Reversible Log Multiplier With 3 Correction Terms.

The reversible implementation and simulation results of logarithmic multiplier is as shown in Figure 5.

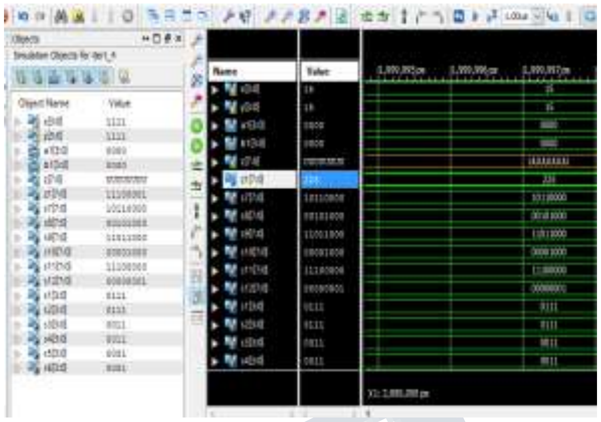

Figure 5: Simulation Results of 4x4 Log Multiplier.

VI. CONCLUSION

In this paper, the 4x4 logarithmic multiplier is designed using reversible logic.Power analysis is done for both conventional and reversible based 4×4 log multiplier and it is found that reversible based log multiplier consumes less power than conventional. As a future work, this design can be used for image enhancement, compression, video tracking, DSP filters etc..

REFERENCES

[1]Z. Babi'c, A.Avramovi'c, P. Buli'c, "*An iterative logarithmic multiplier"* ELSEVIER publication 2011, pp. 0141-9331.

[2] Landauer .R, "*Irreversibility and heat generation in the computing process"*, IBM Research and Development, pp.183-191, 1961.

[3] Bennett C.H., "*Logical reversiblility of Computatio"*, IBM Research and Development, pp 525-532, 1973.

[4] P.Vanusha, k.AmruthaVally, "*Low Power Computing Logic Gate design using Reversible logic*", International Journal of Application or Innovation in Engineering & Management, Vol. 3, N0. 10, pp. 2319-4847, OCT 2014.

[5] Ravish Aradhya H.V, Lakshmesha J, Muralidhara K.N, "*Design optimization of Reversible Logic Universal Barrel Shifter for Low Power applications*", International Journal of Computer Applications, Vol. 40, No. 15, pp. 0975-8887, Feb 2012.

[6]P.K.LALA,J.P.PARKSON,P.CHAKRABORTY, "*Adder Designs using Reversible Logic Gates*", WSEAS

transaction on circuits and systems, Vol. 9, No. 6, ISSN 1109-2734, June 2010.

[7] Sanjeev Kumar Patel, VinodKapse, "*Optimized Design and Implementation of an Iterative Logarithmic Signed Multiplier",* International Journal of Scientific & Engineering Research, Vol. 3, No. 13, pp. 2229-5518, Dec 2012.

[8]BhavaniPrasad.Y, Rajeev Pankaj.N, Samhitha.N.R, Shruthi.U.K, "Design of Reversible Multiplier by Novel ANU gate", International Journal of Engineering and Technology, Vol. 4, No, 6, pp. 2049-3444, June 2014.

[9]PragyanParamita Mohantly1, Mrs.Annapurana K.Y.2, "*FPGA Implementation of Iterative Log Multiplier using Operand Decomposition for Image Processing Application",* International Journal for Research in Applied Science and Engineering Technology, Vol. 2, No. 6, pp. 2321-9653, June 2014.

[10] Patricio Buli'c*, ZdenkaBabi'c and AleksejAvramovi'c, "*A Simple Pipelined Logarithmic Multiplier",* IEEE Transactions on Computers, pp. 978-1-4244-8935, Oct 2010.

[11] Arindam Banerjee*, SamayitaSankar, Mainuck Das and AniruddhaGhosh, "*Design of Reversible binary Logarithmic Multiplier and Divider using Optimal Harbage*", International Journal of Advanced Computer Research, Vol. 5, No. 18. pp. 2277-7970, March 2015.

[12] Rakshith.T.R, Rakshith Saligram, "*Design of High Speed Low Power Multiplier using Reversible Logic: a Vedic Mathematical Approach",* International Conference on Circuits, Power and Computing Technologies, pp.4673-492, ICCPCT-2013.

[13] Kavyashree M S, Praveen Kumar Y G, Dr.M.Z. Kurian, "*A Survey on Reversible Logic Based Logarithmic Multipliers"*, National Conference on Recent Trends in Information and Communication Technology, ISBN: 978-81-927765-3-8, PP:40-42, April-2016.