A DESIGN OF RECURSIVE BASED APPROXIMATE MULTIPLIERS

V.Priya Darshini V. Naga Lakshmi , P. Sandhya Rani , P. Sushma Lavanya , S. Manoj Murthy
Assistant Professor Department of Electronics and Communication Engineering, Gudlavalleru Engineering College
ECE DEPT Gudivada ,India

Abstract: Approximate computing is a technique which will be best suited for error resilient applications. In applications like multimedia signal processing and data mining which can tolerate error, exact computing units are not always necessary. They can be replaced with their approximate counterparts. Approximate computing reduces accuracy, but they are still provides faster results with usually low power consumption this is particularly suited for arithmetic circuits. In this paper, a new design is proposed to utilize the partitions of partial products using recursive multiplication for approximate multipliers. The simulation results shows that the proposed design achieve significant accuracy improvement together with power and delay reductions compared to previous approximate designs.

***Keywords*—Approximate computing, multipliers, low power.**

I. INTRODUCTION

Scientific and engineering problems are computing uses accurate, precise and deterministic algorithms. However, in many applications involving signal/image processing and multimedia, accurate computations are not always necessary, because these applications are error tolerant and producing results those are good enough for human perception[1]. In these error resilient applications, the circuit complexity is reducing, and thus, area, power and delay is very important for the operation. And also still providing meaningful results faster and/or with lower power consumption[2]. Approximation techniques in multipliers focus on accumulating partial products, which is crucial in terms of power consumption.The partial products are altered to introduce terms with different probabilities. Probability statistics of the altered partial products are analyzed, which is followed by systematic approximation. Simplified arithmetic units (half-adder, full-adder, and 4-2 compressor) are proposed for approximation. The arithmetic units are not only reduced in complexity, but care is also taken that error value is maintained low. While systemic approximation helps in achieving better accuracy, reduced logic complexity of approximate arithmetic units consumes less power and area.

In this paper, the three major decision making factors for the selection of an approximate multipliers the type of approximate full adder ,half adders and compressors used to construct the multiplier, the architecture, i.e., structure of the multiplier and the placement of sub-modules of approximate and exact multipliers in the main multiplier module. Based on these factors, we explored the design for circuit level implementations of approximate multipliers[2]. A common conclusion from these works is that it is feasible to develop a generic analysis with reasonable complexity for components constructed from similar type of basic functional unit. In this regard, analysis of approximate multipliers has attracted relatively on area and delay parameters. Therefore, in this paper, for the first time, we present the low-power recursive approximate multipliers with approximate partial products. This represents a major class of low-power approximate multipliers.

II . EXISTING ARCHITECTURE

Implementation of multipliers comprises three levels i.e., generating the partial products, partial products reduction tree, and finally, a vector merge addition to produce final product from the sum and carry rows generated from the reduction tree. Second step consumes more power. In this brief, approximation is applied in reduction tree stage in [3]. The partial products $a_{m,n}$ and $a_{n,m}$ are combined to form propagate and *generate* signals as given in (1). The resulting propagate and *generate* signals form altered partial products $p_{m,n}$ and $g_{m,n}$. The original and transformed partial product matrices.

$$p_{m,n} = a_{m,n} + a_{n,m}$$
$$g_{m,n} = a_{m,n} \cdot a_{n,m} \qquad (1)$$

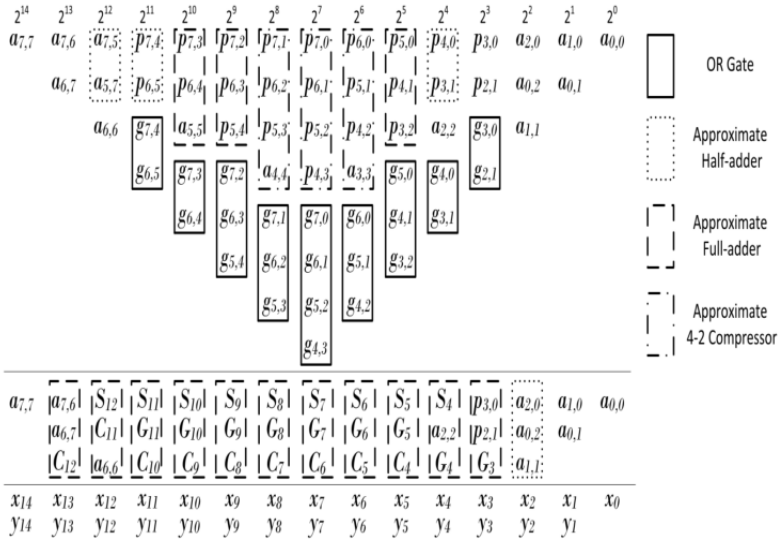

Fig.1: Reduction of altered partial products.

The probability of the altered partial product g*m,n* being one is 1/16, which is significantly lower than 1/4 of a*m,n* . The probability of altered partial product p*m,n* being one is 1/16 3/16 3/16 7/16, which is higher than g*m,n* . These factors are considered, while applying approximation to the altered partial product matrix in [4]. The accumulation of generate signals is done column wise. Using OR gate in the accumulation of column wise generate elements in the altered partial product matrix provides exact result in most of the cases. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. Carry and Sum are two outputs of these approximate circuits. Since Carry has higher weight of binary bit. Approximation is handled in such a way that the absolute difference between actual output and approximate output is always maintained as one. Hence Carry outputs are approximated only for the cases, where Sum is approximated[5].In adders and compressors, XOR gates tend to contribute to high area and delay. For approximating half-adder, XOR gate of Sum is replaced with OR gate as given in (2). This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table I. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch.

$$\text{Sum} = x1 + x2$$
$$\text{Carry} = x1 \cdot x2. \qquad (2)$$

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. Carry is modified as in (3) introducing one error. This provides more simplification, while maintaining the difference between original and approximate value as one. The truth table of approximate full-adder is given in Table II.

$$W=(x1+x2)$$
$$\text{Sum}=(W \oplus x3)$$
$$\text{Carry}=(W.\ x3). \qquad (3)$$

Table I

TRUTH TABLE OF APPROXIMATE HALF ADDER

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
x1	*x2*	*Carry*	*Sum*	*Carry*	*Sum*	
0	0	0	0	0 ✔	0 ✔	0
0	1	0	1	0 ✔	1 ✔	0
1	0	0	1	0 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✗	1

Table II

TRUTH TABLE OF APPROXIMATE FULL ADDER

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
x1	*x2*	*x3*	*Carry*	*Sum*	*Carry*	*Sum*	
0	0	0	0	0	0 ✔	0 ✔	0
0	0	1	0	1	0 ✔	1 ✔	0
0	1	0	0	1	0 ✔	1 ✔	0
0	1	1	1	0	1 ✔	0 ✔	0
1	0	0	0	1	0 ✔	1 ✔	0
1	0	1	1	0	1 ✔	0 ✔	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✔	0 ✗	1

Approximate 4-2 compressor produces nonzero output even for the cases where all inputs are zero. This results in high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the reduction tree. The proposed 4-2 compressor overcomes this drawback. In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases which is shown in Table III.

$$W1 = x1 \cdot x2; W2 = x3 \cdot x4$$
$$Sum = (x1 \oplus x2) + (x3 \oplus x4) + W1 \cdot W2$$
$$Carry = W1 + W2. \quad (4)$$

This property is taken to eliminate one of the three output bits in 4-2 compressor. To maintain minimal error difference as one, the output “100" (the value of 4) for four inputs being one has to be replaced with outputs “11" (the value of 3). For Sum computation, one out of three XOR gates is replaced with OR gate in[3].

Table III

TRUTH TABLE OF APPROXIMATE 4-2 COMPRESSOR

Inputs				Approximate outputs		Absolute Difference
x1	*x2*	*x3*	*x4*	*Carry*	*Sum*	
0	0	0	0	0 ✔	0 ✔	0
0	0	0	1	0 ✔	1 ✔	0
0	0	1	0	0 ✔	1 ✔	0
0	0	1	1	1 ✔	0 ✔	0
0	1	0	0	0 ✔	1 ✔	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✔	1 ✔	0
1	0	0	0	0 ✔	1 ✔	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✔	1 ✔	0
1	1	0	0	1 ✔	0 ✔	0
1	1	0	1	1 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✔	0
1	1	1	1	1 ✗	1 ✗	1

Also, to make the Sum corresponding to the case where all inputs are ones as one, an additional circuit $x1 \cdot x2 \cdot x3 \cdot x4$ is added to the Sum expression. This results in error in five out of 16 cases. Carry is simplified as in (4). The corresponding truth table is given in Table III.

III . PROPOSED ARCHITECTURE:

The technique proposed which is used in this paper for designing 8× 8 multipliers using 4× 4 multipliers is known as recursive multiplication. Suppose there are 2 numbers A and B of 2a bits each. It is possible to break the two numbers into 2 halves, i.e. most significant a bits and least significant a bits .So A_K denotes the upper a bits of A, A_L denotes the lower a bits of A and similarly, B_K and B_L denote the upper and the lower a bits of B respectively. Then, instead of performing a 2a × 2a multiplication, four a × a multiplications are performed and added to get the final output, as shown in figure 2.

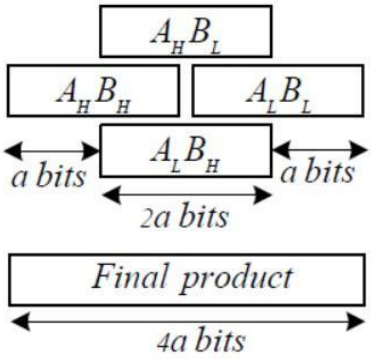

Fig.2: Recursive form of multiplication

3.1 4X4 MULTIPLIER DESIGNS:

The designing of 8× 8 multipliers using 4× 4 multipliers is known as recursive multiplication. Four 4x4 multipliers have been implemented for the and further used in the 8x8 implementation. The partial products are obtained by multiplying the taken input bits. The advantage of breaking the products is to obtain smaller multiplication blocks that are performed in parallel and thus faster. Then, they merely need to be added, according to Fig.2 to obtain the final product.

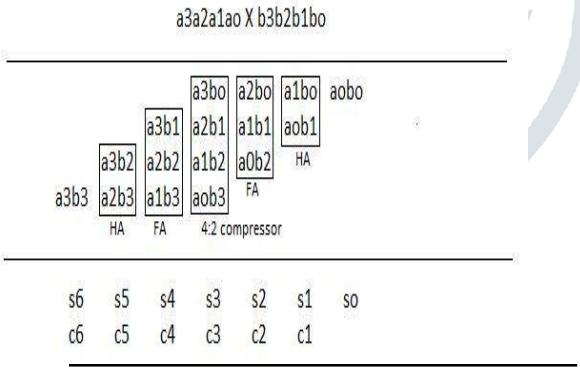

Fig3. four by four multiplier design

3.2 **8X8 MULTIPLIER DESIGNS:**

The 8 by 8 bit multiplier is designed by taking four parts of 4 by 4 bit multiplications using recursive multiplication In many cases, accuracy is not a strict requirement and hence, it can be traded off with power, delay and area. The multiplication process is shown in the below fig3. When designing power efficient, low accuracy multipliers, the four by four multipliers are used for calculating the significant partial products.

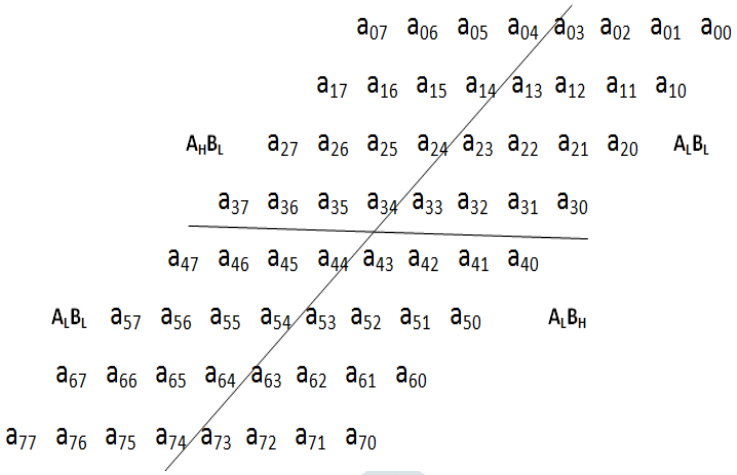

Fig4. 8 by 8 multiplier design using four by four multipliers

IV. SYNTHESIS AND SIMULATION RESULTS:

Mainly we are dealing with area ,delay and power constraints. As we are not considering with the exact results .By comparing the area and delay constraints of proposed work and existing work ,we observed that the area, delay and power of the multiplier are reduced such that the designed multipliers are area efficient with low delay and less power. The simulation and synthesis results are shown below and also power report is also given in below.

```
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s50pq208-5

 Number of Slices:                14  out of    768     1%
 Number of 4 input LUTs:          24  out of   1536     1%
 Number of IOs:                   16
 Number of bonded IOBs:           16  out of    124    12%
```

fig4.synthesis report of four by four multiplier

Table IV. Power , area and delay parameters of 4x4 multiplier

Area (LUT's)	**Power (mW)**	**Delay (nS)**	**Power Delay Product(PDP) x10^-12**
11	45	2.04	91.8

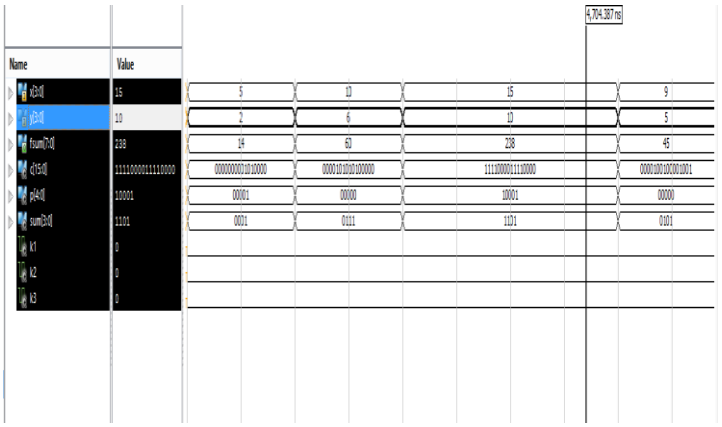

fig5.simulation report of four by four multiplier

```
==========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s50pq208-5

 Number of Slices:                       51  out of    768     6%
 Number of 4 input LUTs:                 89  out of   1536     5%
 Number of IOs:                          32
 Number of bonded IOBs:                  32  out of    124    25%
```

fig6.synthesis report of 8 by8 multiplier

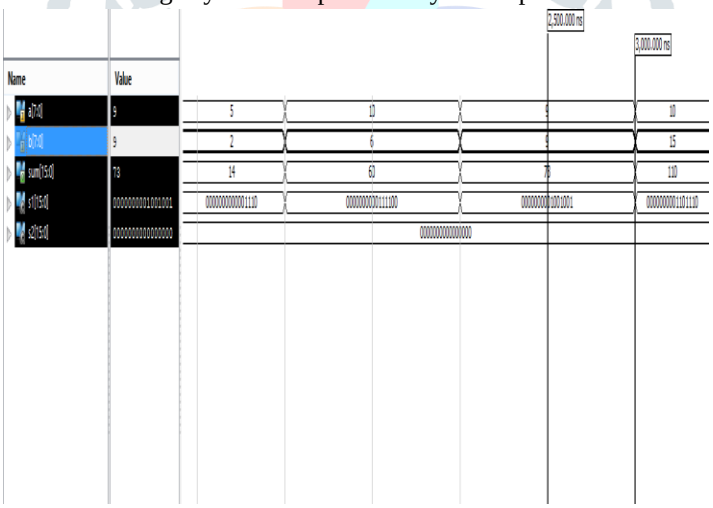

fig6.simulation report of 8by 8 multiplier

Table V. Power , area and delay parameters of existing and proposed multiplier

Parameter	Area (LUT's)	Delay (nS)	Power (mW)	Power Delay Product(PDP) x10^-12
Existing Multiplier	39	4.003	92	368.27
Proposed Multiplier	26	2.715	122	331.24

V . CONCLUSION AND FUTURE WORK:

This approximate multipliers, will lead to an estimated error probability, which can be considered a tight lower bound. The improvement we have made from the existing architecture is elimination of propagate and generate terms. As they are eliminated delay parameter is reduced . This paper can be further improved for finding error probability. When error probabilities are known more accurate multipliers can be designed which eliminates the concept of approximation.

REFERENCES:

[1] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. vol. 32, no. 1, pp. 124–137,Jan. 2013.

[2] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim,"Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[3] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi,"Design-efficient approximate multiplication circuits through partial product perforation," IEEE Trans. Very Large Scale Integr.(VLSI) Syst.,vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[4] P. Kulkarni, P. Gupta, M. Ercegovac, "Trading accuracy for power with an Underdesigned Multiplier architecture," 24th International Conference on VLSI Design, 2011.

[5] K. Bhardwaj, P.S. Mane, J. Henkel, "Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems," 15th International Symposium on Quality Electronic Design (ISQED), 2014.

[6] C.H. Chang, J. Gu, M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE Transactions on Circuits and Systems, vol.51, no.10, pp.1985-1997, 2004.

[7] Harpreet Singh Dhillon, Abhijit Mitra, "A ReducedBit Multiplication Algorithm for Digital Arithmetic",International Journal of Computational andMathematical Sciences, Waset, Spring 2008.