Design Of Approximate Multiplier to Reduce Delay and Area

SULAKSHANA W
Faculty Department of Electrical and Electronics Engineering, GNDEC Bidar, KA- India

Abstract- ***The key idea of approximate computing is to trade off accuracy in computation, for better performance and energy efficiency. Many important applications such as, multimedia signal processing and image processing applications etc. can essentially tolerate inaccurate computation. Approximate com- putting has the ability to tolerate lack of accuracy and soft errors found in many applications to gain remarkable energy. It can provide considerable reductions in circuit complexity, delay, area and energy consumption by easing accuracy requirements. In this paper, we propose approximate multiplier design with 32-bits using Vedic multiplier and altered partial product, that demonstrate reduced area and shorter critical path delay than the conventional multiplier. The proposed design is analyzed using Xilinx.***

Indexed Terms- ***Approximate computing, approximate multi- plier, approximate compressors, approximate adders.***

I. INTRODUCTION

In different arithmetic blocks, multipliers are the main blocks, which are widely used in different applications such as signal processing and image processing applications. In today's modern world power consumption is the first concern. Considering this the approximate computing is emerged as a promising solution to save the power [1]. For Digital signal processing (DSP) applications, approximate multiplication proves to be energy efficient compared to exact multipliers [4]. Most of the DSP blocks are used in video and image processing algorithms, and most of the outputs are image or either video for human consumption. The human eye neglects the minute error in image or video [2]. As approximate adders and compressors are utilized in multiplier designs. These designs depend on different characteristics of compression, such that lack of accuracy in computation can meet with respect to circuit-based figures of merit of a design (number of transistors, delay and power consumption) [3]. There are many different design approaches for approximation. Approximation methods in multipliers focus on accumulation of partial products, which is pivotal in terms of power utilization [5]. To reach the power, area and speed requirements, many different methods at different design concept levels have been suggested [7]. Approximate computing can be applied at both software and hardware levels. Error distance (ED) [8] can be dened as the arithmetic distance between a correct output and approximate output for a given input [6]. In [5], approximate adders are evaluated and normalized. ED (NED) is proposed as nearly unchanged metric independent of the size of the approximate circuit [9]. Also, basic error analysis, MRE is calculated for existing and proposed multiplier designs. The mean error distance (MED) and normalized error distance (NED) are then proposed. In this paper we are using altered partial product method [6] to implement 32-bit multiplier using Vedic multiplier [10].

II. EXISTING ARCHITECTURE

The existing system consists of generating the partial products, reduction of generated partial products tree, and producing the final product by vector merger addition. The second step consumes more power compared to first step. In the second step approximation is applied.

An 8-bit unsigned multiplier is used to explain the existing system in approximation of multipliers. Consider two 8-bit unsigned input operands am and an. The partial products are altered column wise as shown in figure 1.

$$a_{7,7}a_{7,6}a_{7,5}a_{7,4}a_{7,3}a_{7,2}a_{7,1}a_{7,0}a_{6,0}a_{5,0}a_{4,0}a_{3,0}a_{2,0}a_{1,0}a_{0,0}$$
$$a_{6,7}a_{6,6}a_{6,5}a_{6,4}a_{6,3}a_{6,2}a_{6,1}a_{5,1}a_{4,1}a_{3,1}a_{2,1}a_{1,1}a_{0,1}$$
$$a_{5,7}a_{5,6}a_{5,5}a_{5,4}a_{5,3}a_{5,2}a_{4,2}a_{3,2}a_{2,2}a_{1,2}a_{0,2}$$
$$a_{4,7}a_{4,6}a_{4,5}a_{4,4}a_{4,3}a_{3,3}a_{2,3}a_{1,3}a_{0,3}$$
$$a_{3,7}a_{3,6}a_{3,5}a_{3,4}a_{2,4}a_{1,4}a_{0,4}$$
$$a_{2,7}a_{2,6}a_{2,5}a_{1,5}a_{0,5}$$
$$a_{1,7}a_{1,6}a_{0,6}$$
$$a_{0,7}$$

Figure 1

Columns containing three or more partial products, the partial products $a_{m,n}$ and $a_{n,m}$ are combined to form generate and propagate signals as per equation 1.The resulting signals form altered partial products. From column 3 to column 11, the partial products $a_{m,n}$ and $a_{n,m}$ are replaced by $p_{m,n}$ and $g_{n,m}$ as shown in figure 2.

$$a_{7,7}a_{7,6}a_{7,5}p_{7,4}p_{7,3}p_{7,2}p_{7,1}p_{7,0}p_{6,0}p_{5,0}p_{4,0}p_{3,0}a_{2,0}a_{1,0}a_{0,0}$$
$$a_{6,7}a_{6,6}p_{6,5}p_{6,4}p_{6,3}p_{6,2}p_{6,1}p_{5,1}p_{4,1}p_{3,1}p_{2,1}a_{1,1}a_{0,1}$$
$$a_{5,7}g_{5,6}a_{5,5}p_{5,4}p_{5,3}p_{5,2}p_{4,2}p_{3,2}a_{2,2}g_{1,2}a_{0,2}$$
$$g_{4,7}g_{4,6}g_{4,5}g_{4,4}p_{4,3}g_{3,3}g_{2,3}g_{1,3}g_{0,3}\,g_{3,7}g_{3,6}g_{3,5}g_{3,4}g_{2,4}g_{1,4}g_{0,4}$$
$$g_{2,7}g_{2,6}g_{2,5}g_{1,5}g_{0,5}$$
$$g_{1,7}g_{1,6}g_{0,6}$$
$$g_{0,7}$$

Figure. 2 the transformation of altered partial products

Columns containing three or more partial products, the partial products $a_{m,\,n}$ and $a_{n,\,m}$ are combined to form generate and propagate signals as per equation 1 and equation 2. The resulting signals form altered partial products. From column 3 to column 11, the partial products $a_{m,\,n}$ and $a_{n,\,m}$ are replaced by $p_{m,\,n}$ and $g_{n,\,m}$ as shown in Fig. 2.

$$p_{m,\,n} = a_{m,\,n} + a_{n,\,m} \quad (1)$$
$$g_{n,\,m} = a_{m,\,n} \cdot a_{n,\,m} \quad (2)$$

Altered partial product $g_{m,n}$ has probability of one in sixteen, which is lower than one of four in $a_{m,\,n}$. Probability of $p_{m,\,n}$ being one is seven of sixteen which is higher than $g_{m,\,n}$. These factors are considered while applying approximation.

III. APPROXIMATION OF ALTERED PARTIAL PRODUCTS $g_{m,\,n}$

The accumulation of generate signals is done column wise. As each element has a probability of 1/16 of being one, two elements being 1 in the same column even decreases [6]. 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. The probability of error while using OR gate for reduction of generate signals in each column is also listed [6]. As the number of generate signals increases, the error probability also increases linearly. However, the value of error also rises. To prevent this, the maximum number of generate signals to be grouped by OR gate is kept at 4. OR gates are used in the columns having generate signals.

IV. APPROXIMATION OF REMAINING PARTIAL PRODUCTS

The accumulation of partial product uses approximate circuits. Approximate adders and 4:2 approximate compressors are proposed to accumulate. The truth tables are drawn for half adder full adder and compressor. Approximation is done in such a way the absolute difference between accurate and approximate is always maintained one. Therefore, carry outputs are approximated where sum is approximated. In adders and compressors XOR gates are replaced with OR gates because XOR gates contribute more area and delay. Accurate half adder equation is given in equation (3) and equation (4), and approximate half adder is given in equation (4) and equation (5).

$$Sum = A \wedge B \quad (3)$$
$$Carry = A1 \cdot A2 \quad (4)$$
$$Sum = A1 + A2 \quad (5)$$
$$Carry = A1 \cdot A2 \quad (6)$$

For approximate full adder, one XOR gate is replaced with OR gate in sum calculations which are given in equations (7), (8) and (9).

$$W = A1 + A2 \quad (7)$$
$$Sum = W \wedge A3 \quad (8)$$
$$Carry = W \cdot A3 \quad (9)$$

TABLE I
TRUTH TABLE FOR APPROXIMATE HALF ADDER

Inputs	Correct outputs	Approximate outputs	Absolute Difference
A1A2	SumCarry	sum Carry	

00	00	0 0	0
01	10	1 0	0
10	10	1 0	0
11	01	1× 1	1

TABLE II
TRUTH TABLE FOR APPROXIMATE FULL ADDER

Inputs			Correct outputs	Approximate outputs		Absolute Difference
A1	A2	A3	SumCarry	Sum	Carry	
0	0	0	00	0	0	0
0	0	1	10	1	0	0
0	1	0	10	1	0	0
0	1	1	01	0	1	0
1	0	0	10	1	0	0
1	0	1	01	0	1	0
1	1	0	11	1×	0×	1
1	1	1	10	0×	1	0

The 4-2 compressor are used where three bits are required for the output. To calculate sum, one XOR gate out of three is replaced with OR gate. This is illustrated with equations (10), (11), (12) and (13) and the truth of 4:2 approximate is shown in table III.

$$w1 = A1 \cdot A2 \quad (10)$$

$$w2 = A3 \cdot A4 \quad (11)$$

$$Sum = (A1 \wedge A2) + (A3 \wedge A4) + W1 \cdot W2 \quad (12)$$

$$Carry = w1 + w2 \quad (13)$$

TABLE III
TRUTH TABLE FOR APPROXIMATE FULL ADDER

Inputs				Approximate outputs	Absolute Difference
A1	A2	A3	A4	Sum Carry	
0	0	0	0	00	0
0	0	0	1	10	0
0	0	1	0	10	0
0	0	1	1	01	0
0	1	0	0	10	0
0	1	0	1	1×0×	1
0	1	1	0	1×0×	1
0	1	1	1	11	0
1	0	0	0	10	0
1	0	0	1	1×0×	1
1	0	1	0	1×0×	1
1	0	1	1	11	0
1	1	0	0	01	0
1	1	0	1	11	0
1	1	1	0	11	0
1	1	1	1	1×1×	1

Figure.3 shows the reduction of partial products of 8x8 approximate multiplier. It needs steps to produce sum and carry outputs. Here one four input OR gates, four two input OR gates, and four three input OR gates are required to generate signals. For reducing other partial products three approximate half adders, three approximate full adders and three approximate compressors are required to produce sum and carry outputs.

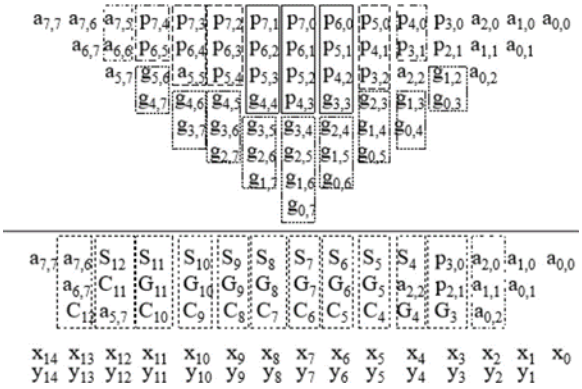

Figure. 3. Reduction of Altered Partial Products

V. ENHANCEMENT OF THE PROPOSED ARCHITECTURE

Existing system is implemented with 8-bit and 16-bit un-signed multipliers architecture by altering the partial products. Proposed Architecture is implemented with 32-bit by following the partial product alteration method. These 8-bit and 16- bit approximate multipliers are called in the implementation of 32-bit multiplier using Vedic multiplier block.

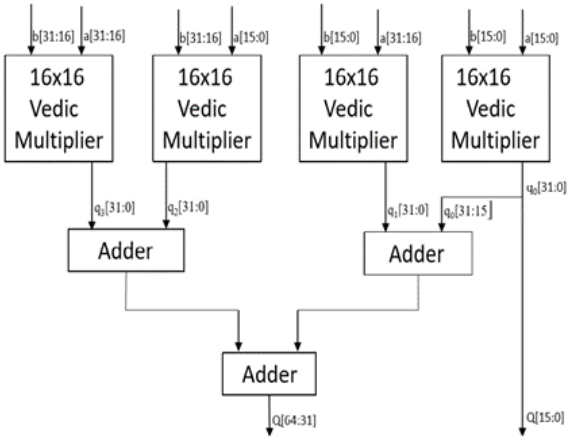

Fig. 4. 32-bit Vedic Multiplier

Figure 5 shows the resulting bits

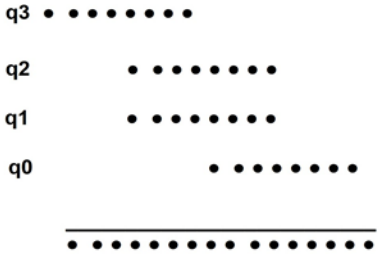

Fig. 5. shows the resulting bits

VI. RESULTS AND DISCUSSION

The 32-bit approximate multiplier is designed using 4 16 × 16 approximate multipliers and this multiplier is implemented using Verilog code. Both synthesis and simulation are also done using Xilinx ISE Design Suit 14.7. During synthesis and simulation delay and device utilization summary is calculated which is listed in the table below IV. Area is calculated in the form of number of LUTs used. Error distance and mean relative error is calculated. Implemented 32-bit approximate multiplier is used in an image processing application. The algorithm 32-bit multiplier is coded and implemented in MATLAB. Some random image is processed to compare the accurate and approximate images. The results are shown in Image 1 and image 2 as shown in Figure.6.

Figure. 6. Random Image Processed to Compare the Accurate (Image 1) and Approximate (Image 2) multiplier Images

TABLE IV
PARAMETRS FOR ACCURATE AND APPROXIMATE MULTIPLIERS

Parameter	Accurate Multiplier	Approximate Multiplier
Delay(ns)	17.461ns	12.01ns
No. of Slice LUTs	1883	1626
MRE	-	0.000082
ED	-	0.000006

CONCLUSION

From the results it is analyzed that approximate multiplier achieves significant reduction in area and delay compared with exact designs. Results shows that the approximate multiplier has less delay compared to accurate multiplier. The proposed approximate multiplier designs can be used in applications where output quality loss is minimum while saving notable area. Proposed approximate multipliers are use in multimedia applications like image processing

REFERENCES

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

[2] 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.

[3] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, *Design and analysis of approximate compressors for multiplication*, IEEE Trans. Comput., vol. 64, no. 4, pp. 984 - 994, Apr. 2015.

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

[5] Naman Maheshwari, Zhixi Yang, Jie Han, and Fabrizio Lombardi, A *Design Approach for Compressor Based Approximate Multipliers*, 2015 28th conference on VLSI Design, 3 – 5 Janauary 2015.

[6] Venkatachalam, S., and Ko, S.B. *Design of power and area efficient approximate multipliers*. IEEE Trans. Very Large Scale Integr. VLSI Syst. 25(5), 1782 -1786, 2017.

[7] P. Kulkarni, P. Gupta, and M. D. Ercegovac, T *rading accuracy for power in a multiplier architecture*, J. Low Power Electron., vol. 7, no. 4, pp. 490 - 501, 2011.

[8] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, and C. Lucas, B *io-Inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications*, IEEE Transactions on Circuits and Systems, vol. 57 no. 4, 2010.

[9] Amir Momeni, Jie Han, Paolo Montuschi, and Fabrizio Lombardi. *Design and Analysis of Approximate Compressors for Multiplication*.IEEE Trans. Computers, PP (99):1 - 1,2014.

[10] Abhyarthana Bisoyi, Mitu Baral, and Manoja Kumar Senapati, A *pprox- imation techniques in multipliers focus on accumulation of partial prod- ucts, which is crucial in terms of power consumption*. IEEE Xplore.26 January 2015