Design and Analysis of Approximate Multipliers

Priyadarshini.V[1], V. Sreelakshmi [2], S.Bhavani[3]
[1,2,3] Asst.prof Department of ECE, Gudlavalleru Engineering College, Gudlavalleru

Abstract— Approximate computing is a computation technique which returns a possibly inaccurate result rather than the guaranteed accurate result and can be used for applications where an approximate result is sufficient for its purpose. Approximate multipliers are widely being used for energy efficient computing in applications that exhibit an inherent tolerance to inaccuracy. Besides the performance of the multiplier, the area and delay makes the identification of suitable approximate multiplier is quite challenging. Hence, we identified the type of approximate full adder will be one of the major decision making factor for the selection of approximate multipliers. Approximate adders are used for the partial product summation in the multiplier. In this paper, an approximate multiplier is designed and analyzed with four different approximate adders. The design is simulated and synthesized using Xilinx Vivado. Compared with previously presented approximate multiplier, the proposed circuits provide significant reduction in area and power.

Keywords— Approximate computing, Approximate full adder, Partial product, Approximate multiplier.

I. INTRODUCTION

At the Nano scale era, improving performance of digital circuits and systems becomes increasingly difficult. Energy efficiency is of paramount concern in digital system design. Computing becomes increasingly heavy with multimedia processing (audio, video, graphics, and image), recognition, search, machine learning and data mining. Researchers and designers started to search novel solutions to compute efficiently. One of the most promising solutions is given by the approximate computing. A common characteristic is that a perfect result is not necessary and an approximate or less-than-optimal result is sufficient. Approximate computing reduces the hardware required for design of the system as compared to accurate computing.

Adders and multipliers are the basic fundamental units in each and every digital circuit used for performing the calculations. With the rapidly growing trends in scaling up to nanometer scale, the arithmetic circuits need to be implemented with low power, compact size, and less propagation delay. These are the reasons for realizing the adders and multiplier blocks using approximate computing.

Multipliers are key components of many high performance digital systems. A system's performance is generally determined by the performance of the multiplier as the multiplier is generally the slowest element in the system and generally consumes more area and power and long latency. Therefore, low-power multiplier design has been an important part in low-power VLSI system design.

An nxn multiplication is conventionally composed of three operational phases: Partial product generation, Carry-free reduction of partial products and Carry propagating addition. The Partial product reduction phase has been the subject of most research and design efforts on parallel multipliers mainly because it is the most area and power consuming part among the three.

In this paper we mainly focus on approximation in the partial product tree of approximate multiplier. For the partial product summation and carry free addition we proposed different approximate full adders and half adders. To improve the speed of operation we proposed a new approximate 4x2 compressor.

The rest of this paper is organized as follows. In Section II, approximate adders are briefly reviewed. Approximate 4x2 compressors is described in Section III. The Approximate multiplier using approximate adder and compressor is described in Section IV. Section V deals with the comparison results. Finally, this paper is concluded in Section VI.

II. APPROXIMATE ADDERS

In this section we discuss different approximations applied to conventional adders for designing approximate adders.

Approximation I

Approximation cannot do in arbitrary fashion. We need to make sure that the resulting simplification should introduce minimal errors in the FA truth table. Here, Approximation is done for both half adder and full adder by replacing XOR gate of sum with OR gate, as XOR gate tends to occupy more area and produces long delay. This approximation results in one error in sum computation. [7]

After approximating, the half adder equations are given in (1)

$$\text{Sum}= A+B$$
$$\text{Carry}=A.B \qquad \text{..........} \ (1)$$

In full adder, any one of the XOR gates is replaced by the OR gate in Sum calculation. This results error in last 2 cases out of 8 cases. Carry is modified as in (2)

$$P=A+B$$
$$\text{Sum}=P \text{ XOR } C$$
$$\text{Carry}= AB+BC \qquad \text{........} \ (2)$$

Approximation II

The truth table of Full adder shows that Sum= $\overline{Cout}$ for six out of eight cases, except for the input combinations A = 0,B = 0,Cin = 0 and A = 1,B = 1,Cin = 1.With this approximation sum has two errors and carry has no error for all the cases.[2]

Approximation III
A close interception of Full Adder truth table shows that Cout=A for six out of eight cases or Cout =B for 6 cases. so in Approximation III Sum is calculated as in Approximation I with three errors and Cout=A.

Approximation IV
For more error tolerant Applications, We can extend Approximation IV with Sum= B by allowing one more error. And Cout=A.[3]

Table I
Truth table of adders with Approximation I -IV

Inputs			Exact Outputs		Approximation I		Approximation II		Approximation III		Approximation IV	
A	B	Cin	Sum	Cout	Sum	Cout	Sum	Cout	Sum	Cout	Sum	Cout
0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	1	1	0	1	0	1	0	1	0	0	0
0	1	0	1	0	1	0	1	0	0	1	1	0
0	1	1	0	1	0	1	0	1	0	1	1	0
1	0	0	1	0	1	0	1	0	1	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	1

III. APPROXIMATE COMPRESSOR

The number of partial products in the multiplication process can be reduced by using the compressors. In general the compressor size can be represented as m:n ,where m represents the number of input bits and n represents the number of output bits. Compressors are of two types Low order compressors and High order compressors. In this approximate multiplier design we used a low order approximate compressor (4:2) for reducing the partial products. [3]
The low order approximate compressor is designed with the two approximate full adders.

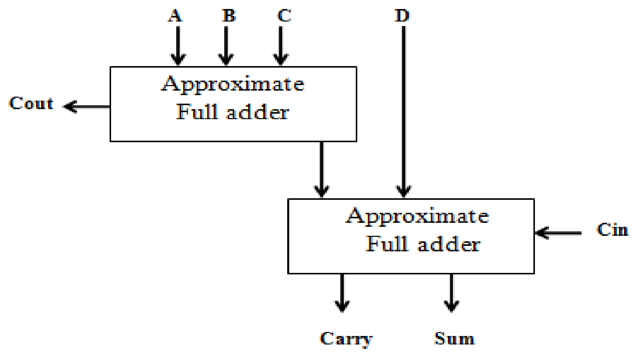

Fig 1: 4:2 Compressor

To maintain minimal error difference , In sum computation the XOR gate can be replaced by an OR gate. This approximation results 5errors out of 16 cases. Carry is considered same as carry in exact compressor. No approximation is applied for the Carry. [5]

$$\text{Carry} = \text{Cin}$$
$$\text{Sum} = (A \oplus B)' + (C \oplus D)'$$
$$\text{Cout} = A.B + C.D \qquad \text{.........} \ (3)$$

Table II
Truth table of 4:2 Approximate Compressor

INPUTS				EXACT COMPRESSOR		APPROXIMATE COMPRESSOR		ABSOLUTE DIFFERENCE
A	B	C	D	SUM	CARRY	SUM	CARRY	
0	0	0	0	0	0	1✗	0✓	1
0	0	0	1	1	0	1✓	0✓	0
0	0	1	0	1	0	1✓	0✓	0
0	0	1	1	0	1	1✗	1✓	1
0	1	0	0	1	0	1✓	0✓	0
0	1	0	1	0	1	0✓	0✗	0
0	1	1	0	0	1	0✓	0✗	0
0	1	1	1	1	1	0✗	1✓	1
1	0	0	0	1	0	1✓	0✓	0
1	0	0	1	0	0	0✓	0✓	0
1	0	1	0	0	0	0✓	0✓	0
1	0	1	1	1	1	0✗	1✓	1
1	1	0	0	0	0	1✗	1✗	1
1	1	0	1	1	1	1✓	1✓	0
1	1	1	0	1	1	1✓	1✓	0
1	1	1	1	0	1	1✓	1✓	0

Table III
Synthesis Results of 4 : 2 Compressor

Compressor Type	Delay in ns	Area in terms of LUT's
Exact 4: 2 compressor	6.582	4
Approximate compressor 4 : 2 compressor	5.103	2

IV. APPROXIMATE MULTIPLIER

Exact 8x8 multiplier:

Compared to other implementation techniques for multiplication process, Dadda technique for multiplication process decreases the number of adder stages. In this techniques half adder and full adders are used for summation of the partial products. Due to this hardware complexity is reduced.
The Dadda multiplication process is shown in Fig 2.

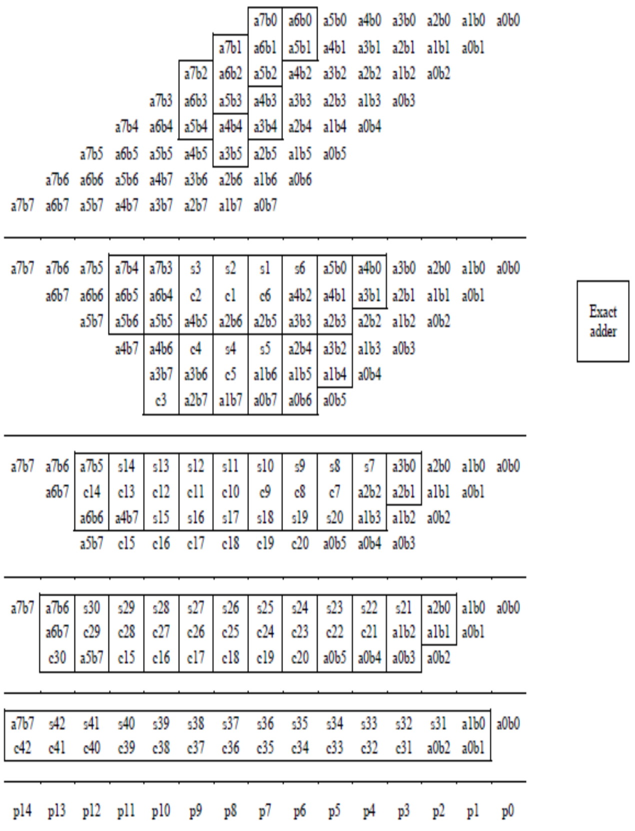

Fig 2: Exact Multiplier

Approximate 8x8 Mulitplier:

In this paper an 8x8 multiplier is proposed using dadda multiplication techniques. Approximation techniques are applied at different stages in the multiplication process.

Different approximation techniques are proposed for the summation of the partial products.

Multiplication process basically divided in to three steps
1. Generation of partial products
2. Reduction in the partial product tree by applying approximate compressor
3. Addition of the partial products

Generation of the partial product:
In this paper we considered an 8 bit multiplicand and an 8 bit multiplier. Let us say a is multiplicand and b is the multiplier. the mathematical representation of the multiplicand and multiplier are as follows

$$a=\sum^{7}_{m=0}a_m 2^m \quad .... (3)$$
$$b=\sum^{7}_{n=0}a_n 2^n \quad .... (4)$$

Partial product is generated by logical AND operation between Multiplier and Multiplicand bits.

The expression for the partial products of a and b is

$$a_{m,n}=a_{m.}\, b_n............................(5)$$

by equ(5) parial products can be obtained as in fig 3.

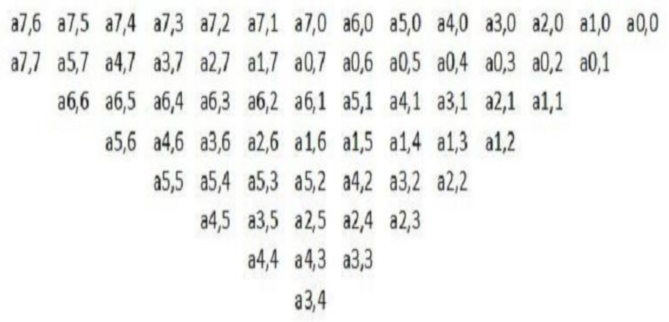
a7,6 a7,5 a7,4 a7,3 a7,2 a7,1 a7,0 a6,0 a5,0 a4,0 a3,0 a2,0 a1,0 a0,0
a7,7 a5,7 a4,7 a3,7 a2,7 a1,7 a0,7 a0,6 a0,5 a0,4 a0,3 a0,2 a0,1
a6,6 a6,5 a6,4 a6,3 a6,2 a6,1 a5,1 a4,1 a3,1 a2,1 a1,1
a5,6 a4,6 a3,6 a2,6 a1,6 a1,5 a1,4 a1,3 a1,2
a5,5 a5,4 a5,3 a5,2 a4,2 a3,2 a2,2
a4,5 a3,5 a2,5 a2,4 a2,3
a4,4 a4,3 a3,3
a3,4

Fig 3: Partial product generation for Dadda Multiplier

Reduction in partial product tree:

suppose if we have multiplier and multiplicand bits n =4, the partial products may be from 0-15. if n=8, partial product terms ranges from 0-63. for n=16, the partial products ranges from 0-255 i.e, as the number of bits in inputs are increasing the number of partial products are greatly increasing. For summation of all the partial products multiplier requires more number of half and full adders which in deed increase the hardware complexity of the multiplier circuit. So there is need to reduce the partial products in the multiplication process. The reduction in the terms can be done from the stastical point of view. The probability of getting partial product should be 1 is 1/4. Partial products reduction can be done in column wise. The column which is having more than three partial products can be altered. [1]
In this architecture column 3 to column 11 partial products are changed.
The partial products $a_{m,n}$ and $a_{n,m}$ are combined together to get propagate term and generate term. The propagate and generate terms are represented as $P_{m,n}$ and $g_{m,n.}$
The mathematical expressions for $P_{m,n}$ and $g_{m,n.}$ are

Propagate term= $a_{m,n}$ or $a_{n,m}$...................(6)
generate term= $a_{m,n}$ and $a_{n,m}$.....................(7)

The generate signal $g_{m,n.}$ has the probability of being 1/16 which is less than the probability of $P_{m,n}$ 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.
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. As the number of generate signals increases, the error probability increases linearly. For a column having m generate signals, m/4 OR gates are used.

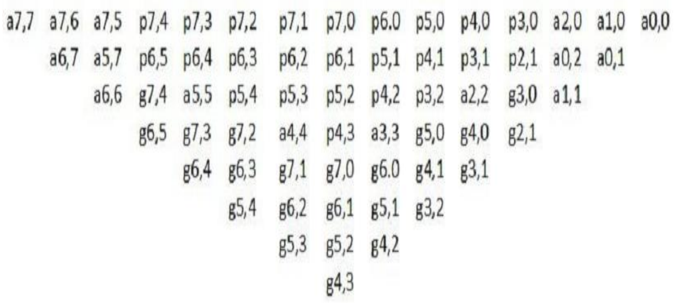
a7,7 a7,6 a7,5 p7,4 p7,3 p7,2 p7,1 p7,0 p6.0 p5,0 p4,0 p3,0 a2,0 a1,0 a0,0
a6,7 a5,7 p6,5 p6,4 p6,3 p6,2 p6,1 p5,1 p4,1 p3,1 p2,1 a0,2 a0,1
a6,6 g7,4 a5,5 p5,4 p5,3 p5,2 p4,2 p3,2 a2,2 g3,0 a1,1
g6,5 g7,3 g7,2 a4,4 p4,3 a3,3 g5,0 g4,0 g2,1
g6,4 g6,3 g7,1 g7,0 g6.0 g4,1 g3,1
g5,4 g6,2 g6,1 g5,1 g3,2
g5,3 g5,2 g4,2
g4,3

Fig 4: Reduction of partial Product Tree

Addition of the partial products:

To perform the addition of the partial products we used different approximate adders and approximate compressors. The structure is shown in Fig.5 [7]

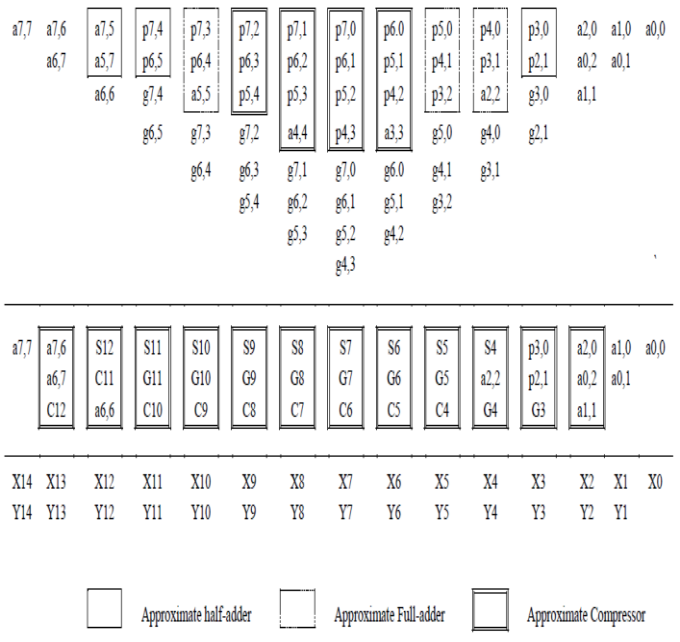
a7,7 a7,6 a7,5 p7,4 p7,3 p7,2 p7,1 p7,0 p6.0 p5,0 p4,0 p3,0 a2,0 a1,0 a0,0
a6,7 a5,7 p6,5 p6,4 p6,3 p6,2 p6,1 p5,1 p4,1 p3,1 p2,1 a0,2 a0,1
a6,6 g7,4 a5,5 p5,4 p5,3 p5,2 p4,2 p3,2 a2,2 g3,0 a1,1
g6,5 g7,3 g7,2 a4,4 p4,3 a3,3 g5,0 g4,0 g2,1
g6,4 g6,3 g7,1 g7,0 g6.0 g4,1 g3,1
g5,4 g6,2 g6,1 g5,1 g3,2
g5,3 g5,2 g4,2
g4,3
a7,7 a7,6 S12 S11 S10 S9 S8 S7 S6 S5 S4 p3,0 a2,0 a1,0 a0,0
a6,7 C11 G11 G10 G9 G8 G7 G6 G5 a2,2 p2,1 a0,2 a0,1
C12 a6,6 C10 C9 C8 C7 C6 C5 C4 G4 G3 a1,1
X14 X13 X12 X11 X10 X9 X8 X7 X6 X5 X4 X3 X2 X1 X0
Y14 Y13 Y12 Y11 Y10 Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1
Approximate half-adder
Approximate Full-adder
Approximate Compressor

Fig 5: Summation of Partial Products using Approximate Half adders and Full adders

In this structure just we required 3 approximate compressors and 3 approximate half and full adders in first stage. The number of full adders required is greatly reduced.

V. RESULT ANALYSIS

This section provides synthesis results to highlight the advantages of different approximate adders. All Approximation multipliers were designed for n=8.
For all experiments, synthesizable Verilog code was written and mapped to Xilinx Artix-7 FPGA using Xilinx Vivado. From the synthesis reports, we get area, delay, dynamic power and static power.

Table IV
Synthesis Results of Approximate Multipliers

Approximate Multipliers	Area in terms of LUT's	Delay in ns	Power in mw	PDP (p J)
Approximation I	23	3.977	78	310.206
Approximation II	26	4.003	87	348.261
Approximation III	19	3.879	56	217.224
Approximation IV	16	3.853	53	189.899

If high approximation can be tolerated for saving more power and delay Approximation IV has to used. Approximation IV offers 43% area savings and 68% power savings over Approximation I.
Table III gives a comprehensive comparison of approximate multipliers to get an idea of tradeoff between Area, power and delay.
For applications where high power savings are desired with less area, Approximation IV can be used. For moderate power savings with better performance, Approximation II is suggested.

VI. CONCLUSION

In this paper, we have presented different approximate multipliers, for partial product addition different approximation techniques have applied on adders. Approximation III and IV achieve significant reduction in area and power consumption compared with Approximation I. The proposed multiplier designs can be used in applications with minimal loss in output quality while saving significant power and area.

REFERENCES

References:

[1] suganthi venkatachalam and Seok -Bum ko,Senor Member,IEEE "Design of Power and Area efficient Approximate Multipliers"*IEEE transactions on Very Large Scale Integration(VLSI) systems,vol.25 No.5.May 2017.*

[2]v.Guptha,D.Mohapata,A.RaghuRathan,Fellow,IEEE and Kaushik Roy Fellow,IEEE
"Low Power Digital Signal Processing using Approximate adders" *IEEE transactions on Computer Aided design of Integrated Circuits and Systems,* vol.32 No.1.January 2013.

[3]S.Sowmiya.K.Stella and V.M.Senthil Kumar"Design and Analysis of 4-2 Compressor for Arithmetic Application",Asian Journal of Applied Science and Technology,volume1,Issue1,February 2017.

[4]Pooja Rathee,RekhaYadav "Approximate Compressors for Multiplication"International journal on Recent and Innovation

Trends in Computing and Communication, volume5,Issue5,May 2017.

[5] Jamuna RamaSwamy and Satish Kumar Nagarajan ,"Approximate MultiplierTechniques - A survey",IJREST ,volume4,Issue5,May 2017.

[6]ch.Lin and C.Lin "High Accuracy Approximate Multiplier with error Correction"in Proc.IEEE 31st International .conf.comput.Design.Sep-2013.

[7] 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.

[8] 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

[9] 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.

[10] P. Kulkarni, P. Gupta, and M. Ercegovac, “Trading accuracy for power with an underdesigned multiplier architecture,” in Proc. 24th IEEE Int. Conf. VLSI Design, Jan. 2011, pp. 346–351.

[11] C. Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” in Proc. Conf. Exhibit. (DATE), 2014, pp. 1–4.