International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 05 | May 2019 www.irjet.net p-ISSN: 2395-0072

Physical Design of Approximate Multiplier for Area and Power Efficiency

N. Vidyaravali[1], Dr.C. Dharma Raj[2], M. Murali Krishna[3]

[1]M. Tech student, Dept. of ECE, GITAM University, Visakhapatnam, A.P, India
[2]Professor, Dept of ECE, GITAM University, Visakhapatnam, A.P, India
[3]Assistant Professor, Dept. of ECE, GITAM University, Visakhapatnam, A.P, India

---

Abstract - ***Approximate computing is a computation technique, where an approximate result is sufficient for its purpose like classification and recognition in data processing. It increases the efficiency in terms of power, area and decrease the design complexity A multiplier has a major impact on the speed and power dissipation of arithmetic processor. This transitory deals with a new design approach for approximation of multipliers. The Efficient Approximate multipliers are proposed by altering the partial product of the multiplier using generate and propagate signals. The partial products of the multiplier are altered to present varying probability terms. Logic complexity of approximation is wide-ranging for accumulation of altered partial products based on their probability. Approximate Half adder, approximate full adder and approximate 4:2 compressors are anticipated to reduce remaining partial products. The proposed approximation is exploited in two variants of 16 bit multipliers. It is analyzed for dadda multiplier. These multipliers are implemented in Verilog and synthesized using encounter Design Compiler and a TSMC 45 nm standard cell library at the slow gate process corner, Synthesis results tells that two proposed multipliers achieve better area and power saving than the exact and existing multipliers. A physical design of exact, existing and proposed multipliers are designed using cadence encounter tool***

***Key Words*: *Compressor, Dadda multiplier, Approximate computing, low power multipliers*.**

1. INTRODUCTION

In applications like multimedia signal processing and data mining which can tolerate error, exact calculating units are not all the time necessary. They can be changed with their approximate counterparts. Research on approximate computing for error tolerant applications is on the rise. Adders and multipliers form the main modules in these applications. Arithmetic units such as adders and multipliers are main components in a logic circuit. The speed and power consumption of arithmetic circuits suggestively influence the performance of a processor. High-performance arithmetic circuits such as carry look ahead adders (CLAs) and Wallace tree multipliers have been extensively operated. However, traditional arithmetic circuits that perform exact operations are encountering difficulties in performance improvement.

Approximate arithmetic that permits a loss of accuracy can reduce the critical path delay of a circuit. Since most approximate designs leverage simplified logic, they tend to have a reduced power consumption and area overhead. Thus, approximate arithmetic is supported as an approach to improve the speed, area and power efficiency of a processor due to the error-resilience of some algorithms and applications. As an important arithmetic module, the multiplier has been redesigned to many approximate versions. The often conflicting advantages and disadvantages of these designs make it difficult to select the most suitable approximate multiplier for a specific application. Thus, approximately redesigned multipliers are reviewed and a comparative evaluation is performed by considering both the error and circuit characteristic.

Approximate full adders [1] are proposed at transistor level and they are employed in digital signal processing applications. Full adders are used in accumulation of partial products in multipliers. Digital signal processing (DSP) blocks form the backbone of numerous multimedia applications used in portable devices. Most of these DSP blocks implement image and video processing algorithms, where the final output is either an image or a video for human consumption. Human beings have limited perceptual abilities when interpreting an image or a video. This permits the outputs of these algorithms to be numerically approximate rather than accurate. This relaxation on numerical exactness be responsible for some freedom to carry out imprecise or approximate computation. We can use this freedom to come up with low-power designs at different levels of design abstraction, i.e., logic, architecture, and algorithm.

To reduce hardware complexity of multipliers, truncation [2], [3] is widely employed in fixed-width multiplier designs. Then a constant or variable correction term is added to compensate for the quantization error introduced by the truncated part. Approximation techniques in multipliers focus on accumulation of partial products [4], which is crucial in terms of power consumption. Broken array multiplier is implemented, here the least significant bits of inputs are truncated, while forming partial products

to reduce hardware complexity. It saves few adder circuits in partial product accumulation [4].

Designs of [5] approximate 4-2 compressors are presented and used in partial product reduction tree of four variants of 8 × 8 Dadda multiplier. The major drawback of the proposed compressors in that they give nonzero output for zero valued inputs. The approximate design proposed in this brief overwhelms the existing drawback. This leads to better precision.

In the design of a fast multiplier, compressors have been widely used to speed up the partial product reduction tree and decrease power dissipation. Optimized designs of 4-2 exact compressors have been proposed, considering compression for approximate multiplication. An approximate signed multiplier has been proposed for use in arithmetic data value speculation (AVDS); multiplication is performed using the Baugh Wooley algorithm. However, no new design is proposed for the compressors for the inexact computation. Designs of approximate compressors have been proposed, however, these designs do not target multiplication. It should be noted that the approach of [5] improves by utilizing a simplified multiplier block that is amenable to approximate multiplication.

2. PROPOSED ARCHITECTURE

Operation of multiplier includes three steps: generation of partial products, reducing tree structure of partial products, and to end, a vector merge addition to produce final product from the sum and carry rows produced from reducing tree structure of partial products. Second step consumes extra power. In this brief, approximation is applied in reducing tree stage.

A 8-bit unsigned1 multiplier is used for design to describe the proposed method in approximation of multipliers. Let us take two 8-bit unsigned input operands $\alpha = \sum_{m=0}^{7} \alpha m 2^m$ and $\beta = \sum_{n=0}^{7} \beta_n 2^n$ .The partial product $a_{m,n} = \alpha_m \times \beta_n$ is the outcome of AND operation between the bits of α_m and β_n. From probability statistical point of view, the partial product $a_{m,n}$ has a probability of 1/4 of being 1. In the columns having more than three partial products, the partial products $a_{m,n}$ and $a_{n,m}$ are combined to form *propagate* and *generate* signals as given in (1). The resulting *generate and propagate* signals form altered partial products $p_{m,n}$ and $g_{m,n}$. From column 3 with weight 2^3 to column 11 with weight 2^{11}, the partial products $a_{m,n}$ and $a_{n,m}$ are substituted by altered partial products $p_{m,n}$ and $g_{m,n}$.The original and transformed partial Product matrices are shown in figure

$$p_{m,n} = a_{m,n} + a_{n,m}$$

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

The probability of the altered partial product $g_{m,n}$ being one is 1/16, which is expressively 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.

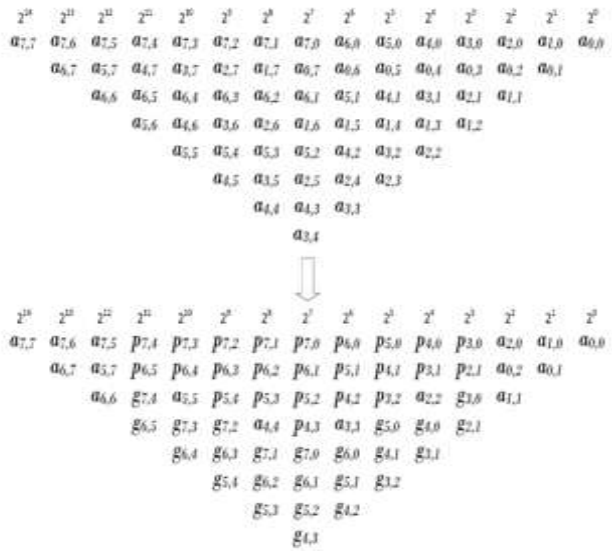


Figure 1: Transformations of generated partial products into altered partial products.

A. Approximation of Altered Partial Products $g_{m,n}$

The accumulation of *generate* signals is finished column wise. As every element has a probability of 1/16 of being one, two elements being 1 in the same column even decreases. As illustrated, in a column with 4 *generate* signals, probability of all numbers being 0 is $(1-pr)^4$ only one element being one is $4pr(1-pr)^3$, the probability of two elements being one in the column is $6pr^2(1-pr)^2$, three ones is $4pr^3(1-pr)^1$ and probability of all elements being 1 is pr^4 where pr is 1/16. The probability statistics for a number of *generate* elements *m* in each column are given in Table I

TABLE -1 PROBABILITY STATISTICS OF GENERATE SIGNALS

m	Probability of generated elements				p_{err}
	All zero	One 1	Two 1's	Three or more 1's	
2	0.8789	0.1172	0.0039	-	0.00390
3	0.8240	0.1648	0.0110	0.00024	0.01124
4	0.7725	0.2060	0.0206	0.00093	0.02153

Based on Table I, using OR gate in the accumulation of column wise *generate* elements in the altered partial product matrix provides exact outcome in most of the cases. The probability of error (*Perr)* while using OR gate for reduction of *generate* signals in every column is also listed in Table I. As can be perceived, the probability of misprediction is very low. Therefore, the number of *generate* signals increases, the error probability increases linearly. However, the value of error also rises. To avert this, the maximum number of *generate* signals to be grouped by OR gate is kept at 4. For a column having *m generate* signals, _*m*/4_ OR gates are used.

B. Approximation of Altered partial products using approximate Half adder, Full adder,4-2 compressor

The accumulation of other partial products with probability ¼ for $a_{m,n}$ and 7/16 for $p_{m,n}$ uses approximate counter parts. Approximate half-adder, full-adder, and 4-2 compressor are anticipated for their accumulation. *Carry* and *Sum* are the two outputs of these approximate circuits. Since *Carry* has higher weight of binary bit, error in *Carry* bit will provide more by producing error difference of two in the output. Approximation is processed 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. In adders and compressors, XOR gates tend to give high area and delay.

***Approximate Half adder*:** For approximating half-adder, XOR gate of *Sum* is changed with OR gate as given below equation. This results in one error in the *Sum* computation as shown in the truth table of approximate half-adder in Table II. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch.

$$Sum = x1 + x2$$

$$Carry = x1 \cdot x2. \qquad (2)$$

TABLE-2 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

***Approximate Full adder*:** In the approximation of full-adder, one of the two XOR gates is changed with OR gate in *Sum* calculation. This results in error in last two cases out of eight cases. *Carry* is modified as in equation shown below introducing one error. This provides more simplification, while keeping the difference between original and approximate value as one. The truth table of approximate full-adder is given in Table III

$$W = (x1 + x2)$$

$$Sum = W \oplus x3$$

$$Carry = W \cdot x3. \qquad (3)$$

TABLE -3 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: Two approximate 4-2 compressors in [5] gives nonzero output even for the cases where all inputs are zero. This causes high ED and 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 overwhelms this drawback. In 4-2 compressor, three bits are essential for the output only when all the four inputs are 1, which occurs only once out of 16 cases. This property is in use to eliminate one of the three output bits in4-2 compressor. To continue minimal error difference as one, the output "100" (the value of 4) for four inputs being one has to be substituted with outputs "11" (the value of 3). For *Sum* computation, one out of three XOR gates is changed with OR gate. 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.

$$W1 = x1 \cdot x2$$

$$W2 = x3 \cdot x4$$

$$Sum = (x1 \oplus x2) + (x3 \oplus x4) + W1 \cdot W2$$

$$Carry = W1 + W2 \quad (4)$$

TABLE -4 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

***Approximate Compressor Design1*:**

An approximate compressor is proposed Instinctively to design an approximate 4-2 compressor, it is possible to substitute the exact full-adder cells in Figure 2 by an approximate full-adder cell. However, this is not very efficient, because it produces at least 17 incorrect results out of 32 possible outputs, i.e. the error rate of this inexact compressor is more than 53% (where the *error rate* is given by the ratio of the number of erroneous outputs over the total number of outputs). Two different designs are anticipated next to reduce the error rate; these designs offer significant performance improvement compared to an exact compressor with respect to delay, number of transistors and power consumption.

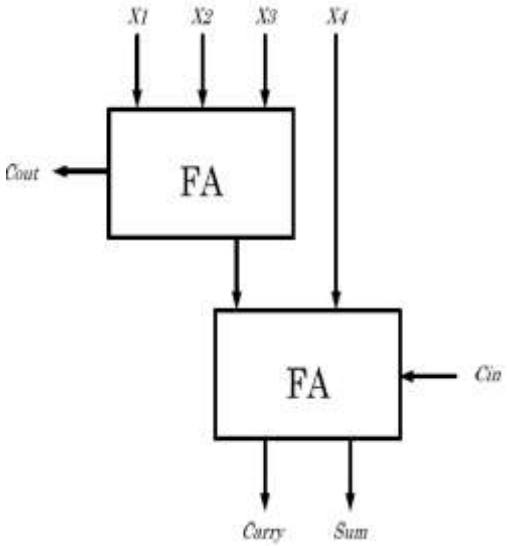


Figure 2 Implementation of 4-2 Compressor

As shown in Table IV the *carry* output in an exact compressor has the same value of the input *cin* in 24 out of 32 states. Hence, an approximate design must consider this feature. In Design 1, the *carry* is simplified to *cin* by changing of the other 8 outputs.

$$carry' = c_{in} \quad (5)$$

Since the *Carry* output has the higher weight of a binary bit, an erroneous value of this signal will give a difference value of two in the output. For example, if the input pattern is "01001" (row 10 of Table 3-8), the correct output is "010" that is equal to 2. By simplifying the *carry* output to *cin*, the approximate compressor will generate the "000" pattern at the output (i.e. a value of 0). This significant difference may not be acceptable; however, it can be compensated or reduced by simplifying the *cout* and *sum* signals. In specific, the simplification of *sum* to a value of 0 (second half of Table 3-8) reduces the difference between the approximate and the exact outputs as well as the complexity of its design. Also, the existence of some errors in the *sum* signal will results in a reductions of the delay of producing the approximate *sum* and the overall delay of the design (because it is on the critical path.

$$sum' = \overline{c_{in}}\,(\overline{x_1 \wedge x_2} + \overline{x_3 \wedge x_4}) \quad (6)$$

In the last step, the change of the value of *cout* in some states, may reduce the error distance provided by approximate *carry* and *sum* and also more simplification in the proposed design.

$$cout' = \overline{\overline{x_1 x_2} + \overline{x_3 x_4}}; \quad (7)$$

Although the above mentioned simplifications of *carry* and *sum* increase the error rate in the proposed approximate compressor, its design complexity and therefore the power consumption are considerably decreased.

TABLE -5 TRUTH TABLE OF APPROXIMATE COMPRESSOR DESIGN1

c_{in}	X_4	X_3	X_2	X_1	c_{out}'	$carry'$	sum'	$Difference$
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	-1
0	0	1	0	0	0	0	1	0
0	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	1	1	0	0	0
0	1	0	1	0	1	0	0	0
0	1	0	1	1	1	0	1	0
0	1	1	0	0	0	0	1	-1
0	1	1	0	1	1	0	1	0
0	1	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1	-1
1	0	0	0	0	0	1	0	1
1	0	0	0	1	0	1	0	0
1	0	0	1	0	0	1	0	0
1	0	0	1	1	0	1	0	-1
1	0	1	0	0	0	1	0	0
1	0	1	0	1	1	1	0	1
1	0	1	1	0	1	1	0	1
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	1	0	0
1	1	0	0	1	1	1	0	1
1	1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	0	0
1	1	1	0	0	0	1	0	-1
1	1	1	0	1	1	1	0	0
1	1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	0	-1

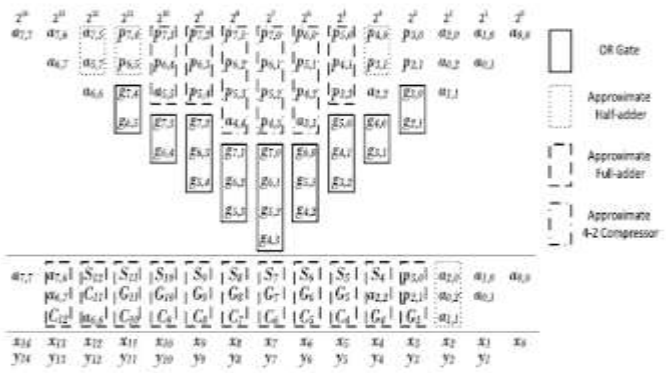


Figure 3: Reduction of Alterd Partial Products

Figure shows the reduction of altered partial product matrix of 8 × 8 approximate multiplier. It needs two stages to produce sum and carry outputs for vector merge addition step. Four 2-input OR gates, four 3-input OR gates, and one 4-input OR gates are required for the reduction of generate signals from columns 3 to 11. The resulting signals of OR gates are labeled as Gi corresponding to the column i with weight 2i . For reducing other partial products, 3 approximate half-adders, 3 approximate full-adders, and 3 approximate compressors are necessary in the first stage to produce Sum and Carry signals, Si and Ci corresponding to column i. The elements in the second stage are reduced using 1 approximate half-adder and 11 approximate full-adders produces final two operands xi and yi to be fed to ripple carry adder for the final computation of the result.

SIMULATION RESULTS

FIGURE- 4 SCHEMETIC DIAGRAM OF APPROXIMATE HALF ADDER

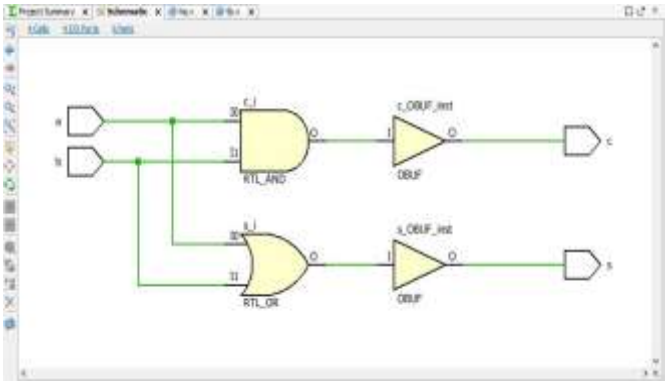


FIGURE-5 OUTPUT WAVEFORM OF APPROXIMATE HALF ADDER

FIGURE -6 SCHEMETIC DIAGRAM OF APPROXIMATE FULL ADDER

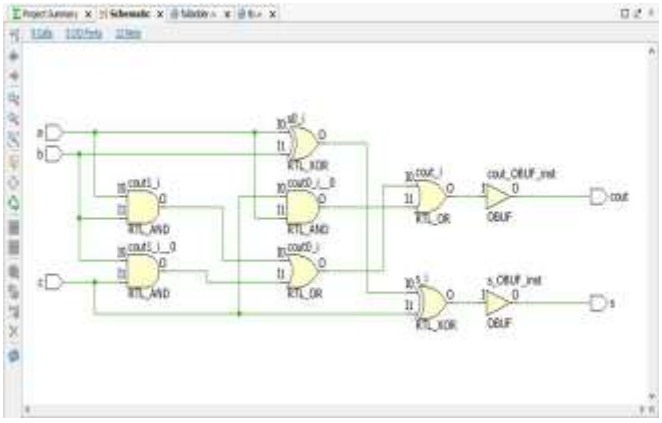


FIGURE- 7 OUTPUT WAVEFORM OF APPROXIMATE FULL ADDER

FIGURE - 8 SCHEMETIC DIAGRAM OF APPROXIMATE 4-2 COMPRESSOR

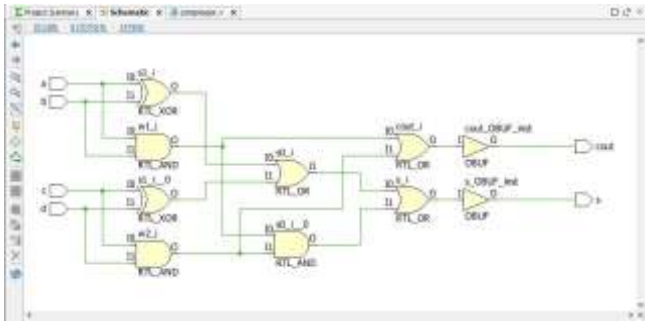


FIGURE- 9 OUTPUT WAVEFORM OF APPROXIMATE 4-2 COMPRESSOR

FIGURE - 10 OUTPUT WAVEFORM OF APPROXIMATE FULL ADDER

FIGURE -11 SCHEMETIC DIAGRAM OF APPROXIMATE COMPRESSOR DESIGN1

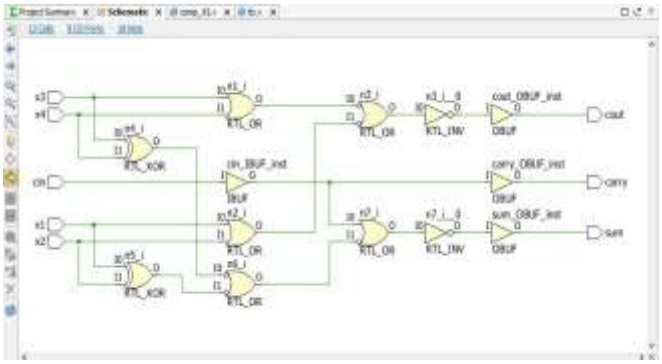


TABLE -6 SYNTHESIS RESULT OF EXACT,EXISTING AND PROPOSED MULTIPLIERS

Design	Area(μ m2)	Power(nW)	Delay (Ps)
Exact	1561	40651.027	700
Mul-1	1389	24828.175	1140
Mul-2	1406	28714.048	1190
ACM1	1461	45864.111	840
ACM2	1766	56314.544	1030

AREA & POWER GRAPHS OF EXACT,EXISTING AND PROPOSED

MULTIPLIERS

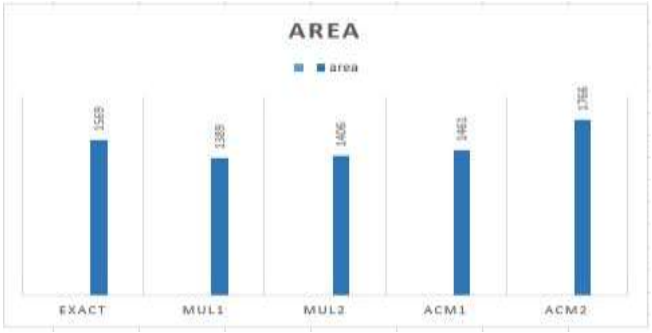


CHART-1 AREA GRAPH

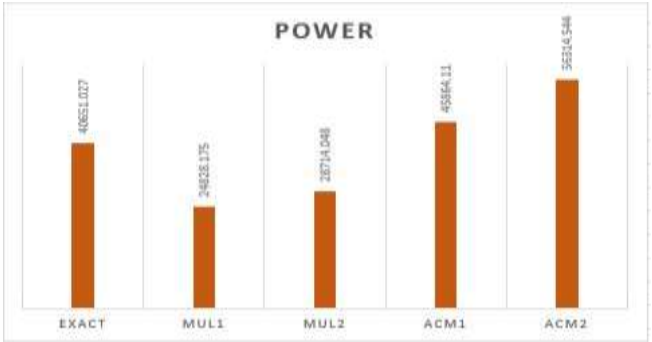
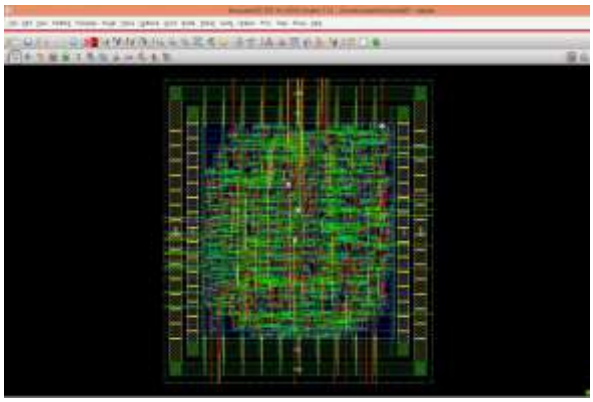


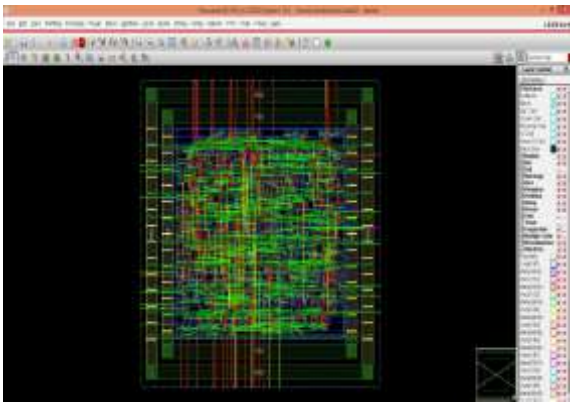
CHART-2 POWER GRAPH

The area and power graph reveals that multiplier1 & multiplier2 gains better area and power efficiency.

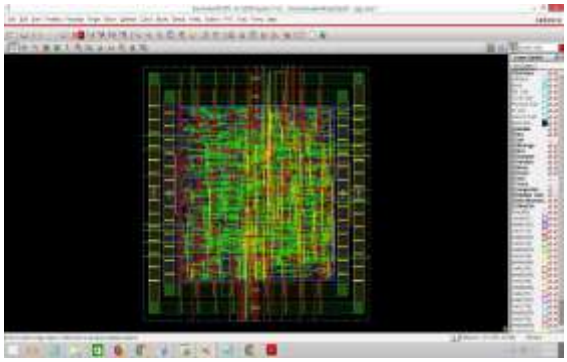
PHYSICAL CHIP DESIGNS OF EXACT, EXISTING AND PROPOSED MULTIPLIER



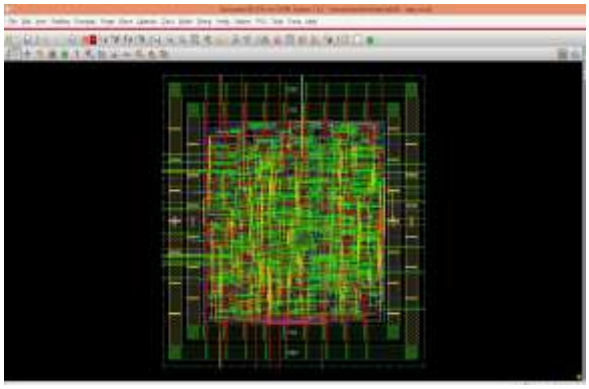
Physical chip design of Exact Multiplier



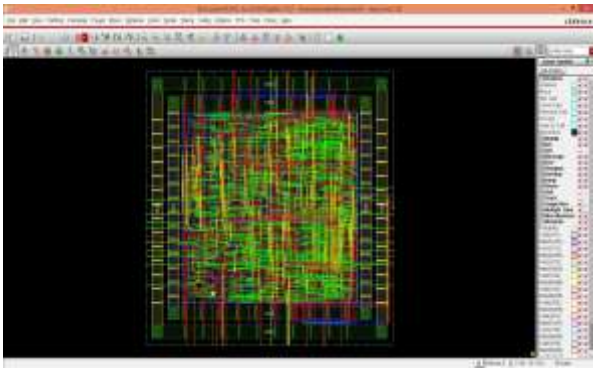
Physical chip design of Multiplier 1



Physical chip design of Multiplier 2



Physical chip design of ACM1



Physical chip design of ACM2

TABLE -7 PHYSICAL DESIGN RESULT OF EXACT,EXISTING AND PROPOSED MULTIPLIERS

Design	Memory(MB)	Power(nW)
Exact	727.05	53216.32
Mul1	702.16	33935.29
Mul2	695.5	37648.66
ACM2	676.36	74139.2
Acm1	668.64	45091

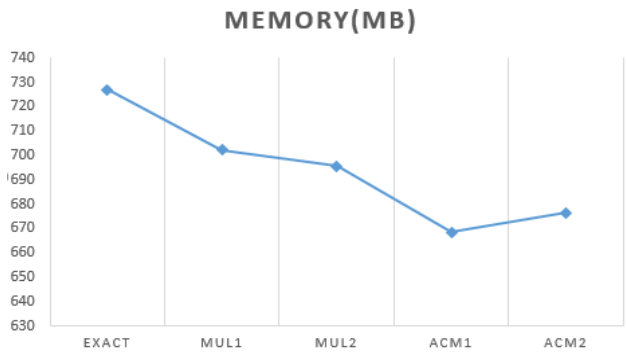


CHART -3 MEMORY & POWER GRAPHS OF EXACT,EXISTING AND PROPOSED MULTIPLIERS

The above graph represents the memory utilization of multiplier chip. Multiplier 1And Multiplier 2 utilizes less memory when compared with Exact Multiplier.

Power graph report of Exact, Existing and Proposed Multipliers depicts that power consumption by multiplier 1 and multiplier2 is less when compared with Exact multiplier, ACM1 and ACM2.

3. CONCLUSION

This project has presented the designs of exact multiplier, multiplier 1, multiplier2, ACM1, ACM2.In this brief, to propose efficient approximate multipliers, partial products of the multiplier are modified using generate and propagate signals. Approximation is applied using simple OR gate for altered generate partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products. These approximate half adder, full adder and compressor utilized in the reduction module of a Dadda multiplier Two variants of approximate multipliers are proposed, where approximation is applied in Multiplier1 and Multiplier2. This approximation technique enables the parameters like high area and power savings while retaining high accuracy. Approximation of Multiplier1 and Multiplier2 achieve significant reduction in area and power consumption compared with exact designs. We explored product perforation on a dadda multiplier architectures, evaluating its impact on different designs. The work is involved between various state-of the art approximation techniques; we showed that the approaches achieve significant gains in power, area compared with the remaining designs.

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] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in *Proc. 31st Asilomar Conf. Signals,Circuits Syst.*, Nov. 1998, pp. 1178–1182.

[3] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 522–531, May 2004.

[4] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[5] 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.

[6] 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.

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

[9] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. IEEE 31st Int. Conf. Comput. Design*, Sep. 2013, pp. 33–38.

[10] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance