Asian Journal of Convergence in Technology
ISSN NO: 2350-1146 I.F-5.11

Volume VI Issue II

Significance-Driven Logic Compression for Energy Efficient Multiplier Design

Sindhu S[1], Spoorthy[2], Suchitha K M[3], Annapurna K Y[4]

Student[1,2,3] , Assistant Professor[4]

Department of Electronics and Communication, PES University, Bengaluru-560085.

***Abstract*— Approximate Arithmetic is a new design paradigm that is being used in many applications which are tolerant to imprecision and do not require accurate results. It can reduce circuit complexity, delay and energy consumption by relaxing accuracy requirements. The partial product bit matrix can be reduced based on their progressive bit significance using a Significance Driven Logic Compression(SDLC) approach. Further, the complexity of the approximate multiplier can be reduced by using Approximate adders in place of exact adders in the accumulation method. Removing some of the transistors from an accurate adder will result in an approximate adder. By using approximate adders which have less number of transistors, the power, propagation delay, and the switching capacitance can be reduced. In this paper, approximate multipliers are implemented using different approximate adders and they are compared with an exact multiplier in terms of power, delay and energy savings.**

Keywords*— *Approximate Arithmetic, SDLC, Approximate Adders.

I. INTRODUCTION

Approximate Computing is the method which gives approximate result rather than accurate result by replacing the complex processing blocks by low complexity ones. As a result, there will be a reduction in complexity, chip-area, power consumption and it also improves performance. Approximate computing can be used in many applications like digital signal processing, robotics, multimedia, computer visions, and so on where accurate results are not necessary and approximate results are sufficient. Arithmetic units for example adders and multipliers are key parts in a logic circuit that can permit the loss of exactness and can reduce the delay of the circuit. Hence approximate arithmetic can be used to improve speed, power efficiency and area of the processor.

Many applications commonly make use of a number of multiplication operations to get results. Multipliers consume a large amount of energy because of their complex logic design. Therefore more research has been carried out in case of the multipliers.

The paper is organized as follows: Section II introduces implemented approximate multiplier design which is proposed in [3]. Section III explains about approximate adders which are proposed in [6]. Gaussian blur filter application is discussed in section IV. The results obtained for different approximate multipliers are compared in Section V. Section VI concludes the paper.

II. APPROXIMATE MULTIPLIER

Conventional multiplication consists of three stages: i) partial product formation, ii) partial product accumulation, and iii) carry propagation adder. Let us consider two 8-bit binary inputs A and B for an (8 x 8) multiplier, the product P can be expressed as P=A.B. 8^2 AND gates are used in parallel to generate the product terms. Since the hardware complexity and power consumption depend largely on the height of the accumulation tree, the main aim is to reduce the number of vertical products in partial product bit matrix(PPM) thereby reducing the height of the accumulation tree for which the SDLC approach can be used [3].

SDLC Approach: Since the accuracy of a product depends mainly on higher significant bits, the higher significant bits are preserved and the lower significant bits are compressed. This can be done using three steps: i)Logic clustering, ii)Logic compression, and iii)Commutative remapping. The block diagram is shown in Fig 1.

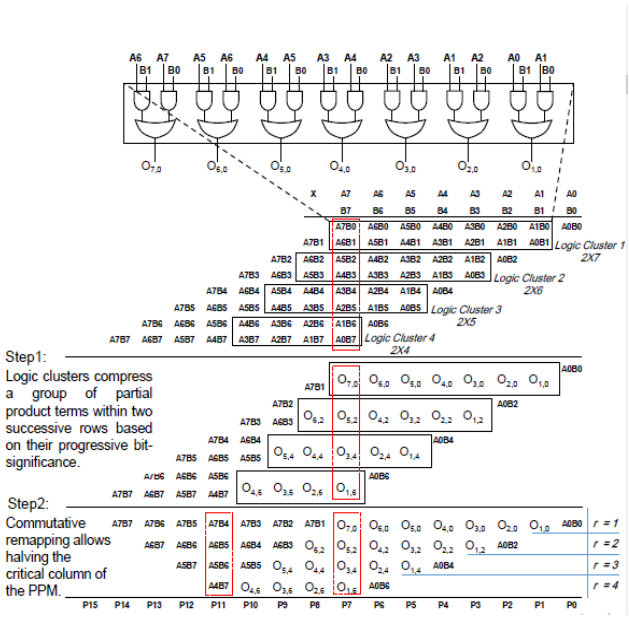


Fig 1:[3] Block diagram indicating SDLC approach

i) ***Logic clustering***: The partial product terms are arranged using different sizes of logic cluster. The size of the logic cluster is (d×L), where d is the depth of the logic cluster i.e. the number of rows and L indicates the number of columns that logic cluster can contain. To make the result more accurate lower significant bits are compressed using logic cluster. Therefore L decreases as we go down in the PPM. This logic cluster targets the column containing two bits starting from the least significant bit and proceeds further to its successive bits. L is given by:

$$L_{d-bit}(r) = \begin{cases} (N+d-2)-r, & 1 \le r < \lceil N/d \rceil \\ (2N-3)-(d+1)(r-1), & r = \lceil N/d \rceil \end{cases}$$

Here r is the row index of the compressed and remapped PPM, N represents the number of bits, d is the depth of the logic cluster. It ranges from 1 to N/d. After multiplication 8 rows of PPM are obtained which can be reduced using SDLC. Since depth d is equal to 2 we will get four rows of PPM after SDLC approach. We will have 4 logic clusters arranged into d×L size.

ii) ***Logic compression***: After arranging the PPM using logic clusters, the two rows in each logic cluster can be further reduced into a single row using a 2-input OR gate. Let us consider two vertically aligned bits $a_i b_j$ and $a_{i-1} b_{j+1}$ of (i+j)th column in PPM then output is given by:

$$O_{i+j}^{2-bit} = a_i b_j \vee a_{i-1} b_{j+1}$$

Using these array of OR gates, the height of the partial product bit matrix is reduced to half.

iii) ***Commutative remapping***: After reducing the partial product matrix using OR gates, these bits are rearranged according to the binary weights. Bits with the same binary weights are arranged in the same column. Now the height of PPM is reduced to half.

***Variable logic cluster*:** An N row PPM can be reduced into N/d rows using Significance driven logic compression method. Since the performance depends largely on the height of PPM, by increasing the depth of logic cluster the height of PPM can further be reduced. But this results in error since OR gate is used to compress the logic cluster. OR gate results in 1 when both inputs are high but in case of adder 1+1 results in sum 0 and carry 1. This error will propagate resulting in higher inaccuracy.
In Fig. 2 a, b and c represents PPM with depth two. An eight rows PPM is reduced to four rows using SDLC method. Fig d, e and f represent PPM with depth three.Fig g, h and i represent PPM with depth four. Since the number of bits for OR gate is increasing, error will propagate further resulting in larger inaccuracy.

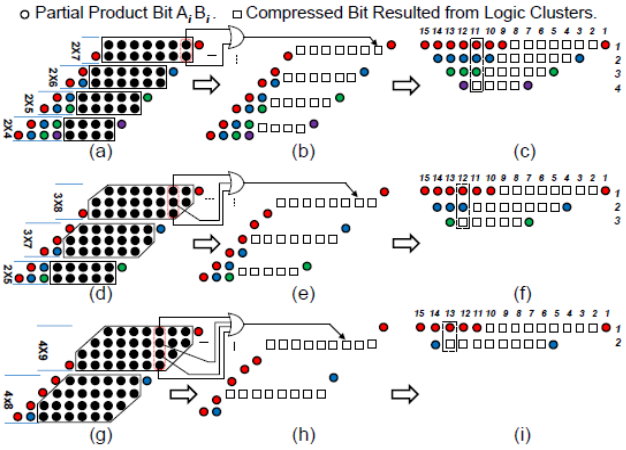


Fig .2: [3] Diagram to show the impact of increasing the depth of logic cluster

After applying SDLC method, the Wallace accumulation method is used to further reduce the PPM to two rows as shown in Fig 3. All the addition happens simultaneously and gives sum and carry which are then rearranged in the second stage. A ripple carry adder is used to get the final 16-bit product.

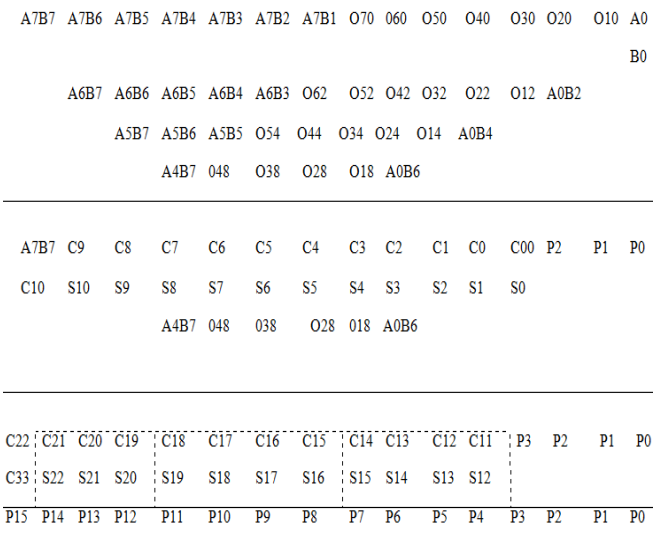


Fig.3 : Wallace accumulation method used after SDLC approach, approximate adders are applied for least significant bits and exact adders for most significant bits. Here C S and P represents carry, sum, and product terms respectively.

III. APPROXIMATE ADDERS

The main component of approximate computing is approximate adders. The mirror adder circuit is most widely used to implement full adder. Approximate computing consumes less power, delay and the number of transistors are reduced which leads to reduction in the circuit complexity.

There are several approximate adders discussed in [5] and [6].

Conventional Mirror Adder: Mirror adder circuit is used to implement full adder because not more than two transistors are in series in the carry generator circuit. The pull up and pull down networks are not dual to each other, so some of the transistors can be removed. The mirror adder consists of 24 transistors. Expressions for SUM and CARRY are given by:

$SUM=AB'C_{in}'+A'BC_{in}'+A'B'C_{in}+ABC_{in}$

$CARRY=AB+BC_{in}+AC_{in}$

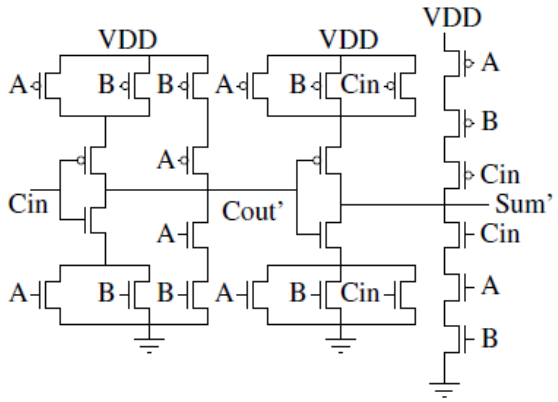


Fig.4: Conventional Mirror Adder circuit

Static and dynamic power are the two main reasons for power dissipation in the circuit. When the system is in an inactive state there will be power dissipation in form of leakage current. This leakage current and reverse-biased PN junction cause static power dissipation. Static power dissipation equation is given by:
P (static) =I (leakage)*Vdd.

Dynamic power dissipation occurs because of switching capacitance i.e. charging and discharging of capacitor at gate output. Hence it will also depend on the number of times capacitor charges and discharges.
$P_{dynamic}=\alpha_{0\text{-}1}CV_{dd}^2f$
α indicates the number of switching activity and C is the load capacitance being charged and discharged.
Total power dissipated P=P (static) +P (dynamic)

Larger delay will be caused due to the series connected transistors in the circuit, therefore removal of some of these transistors leads to faster charging and discharging of capacitor. Hence the delay will decrease. αC reduces which further reduces power. But the removal of transistors cannot be done randomly, we must make sure that this should not result in short/open circuit or any direct path between supply and ground which will lead to an increase in static power. We should also take care that it will result in a minimum error in the full adder truth table and care should be taken to minimize error in carry so that error should not propagate further in the circuit.

Approximate adder 1

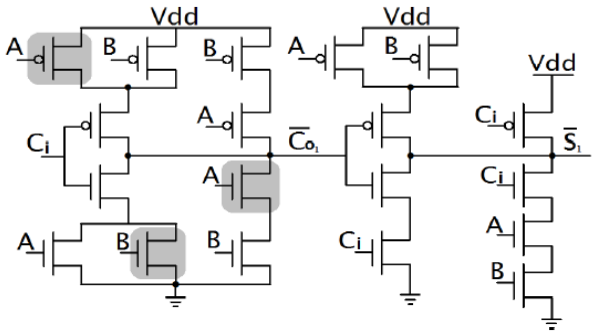


Fig.5: [6] Circuit diagram of Approximate adder 1

No transistors are removed from the carry generation circuit and 5 transistors are removed from the sum generation circuit which will result in 2 errors in sum. This approximate adder consists of 19 transistors. The logical equations are given by:

SUM=A'B'Ci+ABCi
CARRY=AB+BCi+ACi

Approximate adder 2

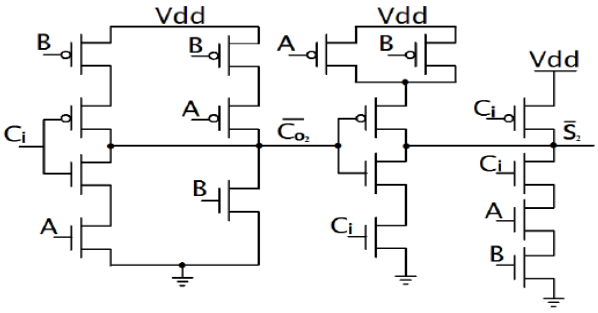


Fig.6: [6] Circuit diagram of Approximate adder 2

For approximate adder 2, 3 transistors are removed from the carry generation circuit and the sum generation circuit is not modified and kept the same as approximate adder1. This will result in 1 error in carry and 2 errors in sum. This adder consists of 16 transistors. Its logical equations are given by:
SUM=A'B'Ci+ABC
CARRY=B+ACi

Approximate adder 3

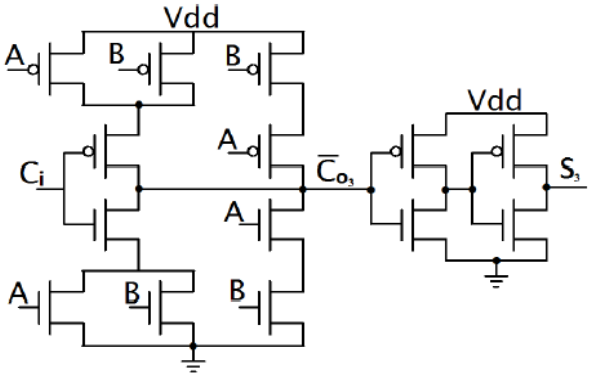


Fig.7: [6] Circuit diagram of Approximate adder3

We can see from the truth table of full adder that sum is compliment of carry for 6 cases out of 8. So SUM is taken as a compliment of carry and no transistors are removed from the carry generation circuit. This circuit consists of 14 transistors. This will result in 2 errors in case of sum. The logical equations of SUM and CARRY are given by:
CARRY=AB+BCi+ACi
SUM = ~CARRY

Approximate adder 4

Approximate adder 4 consists of 11 transistors. From the truth table of approximate adder 2 it can be seen that sum is compliment of carry for 6 cases out of 8. Therefore, carry generation circuit is kept same as that in approximate adder 2 and sum is taken as compliment of carry.

This results in approximate adder 4. The logical equations of SUM and CARRY are given by:
CARRY=B+ACi
SUM=~CARRY

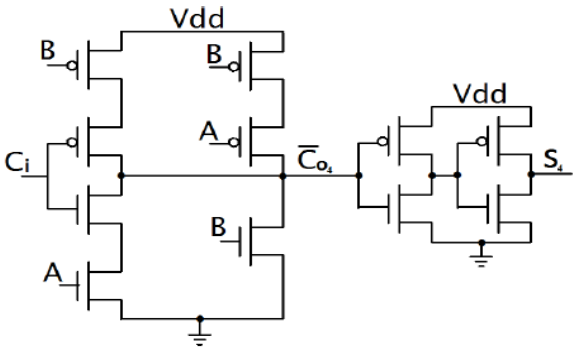


Fig.8: [6] Circuit diagram of Approximate adder 4

IV.CASE STUDY

Gaussian filter is widely used in image processing typically to reduce image noise and remove useless details which gives a blurring effect to the image. Gaussian filter involves the convolution between the image and the Gaussian kernel. A grayscale image is represented by a matrix of pixels with values ranging from 0 to 255.These pixel values are converted to hexadecimal values and stored in a text file. These hexadecimal values are multiplied with the Gaussian mask.

Inputs			Outputs									
A	B	Ci	*S*	*Co*	S_1	Co_1	S_2	Co_2	S_3	Co_3	S_4	Co_4
0	0	0	0	0	0	0	0	0	1×	0	1×	0
0	0	1	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	0×	0	0×	1×	1	0	0×	1×
0	1	1	0	1	0	1	0	1	0	1	0	1
1	0	0	1	0	0×	0	0×	0	1	0	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	0×	1	0×	1

TRUTH TABLE

The gaussian mask is chosen to be a 3x3 matrix whose values are obtained using Gaussian function which is defined by:

$$G(x,y) = \frac{1}{2\Pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Where x and y indicate the coordinates and σ is the standard deviation. Gaussian mask for σ=1 is given by:

G(x,y) =

78	125	78
125	203	125
78	125	78

Convolution of the image with a 2D Gaussian mask is given by:

$$f(x,y) = \sum_{i=0}^{h-1}\sum_{j=0}^{w-1} G(i,j)I(x-i,y-j)$$

For multiplication, we are using approximate multiplier. The resultant hexadecimal values are stored in a file and converted back to an image. MATLAB and ModelSim are the software used for implementing the Gaussian filter.

Peak signal to noise ratio (PSNR)

The PSNR is used to measure image quality. The higher the PSNR value, the image obtained by the Gaussian filter has better quality. The PSNR can be calculated using:

PSNR= $10log_{10}(\frac{255^2}{MSE})$

where MSE is the Mean Square Error between the actual image and the resultant filtered image with p , q are pixel of image P, Q.

$$MSE = \frac{1}{\text{P X Q}}\sum_{p=1}^{P}\sum_{q=1}^{Q}(I_1(p,q) - I_2(p,q))^2$$

Fig 9:Original image

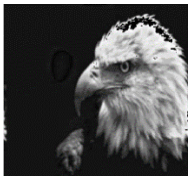


Fig 10:Exact multiplier

Fig11: Approximate Adder1

Fig 12: Approximate adder2

Fig 13: Approximate adder3

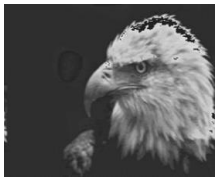
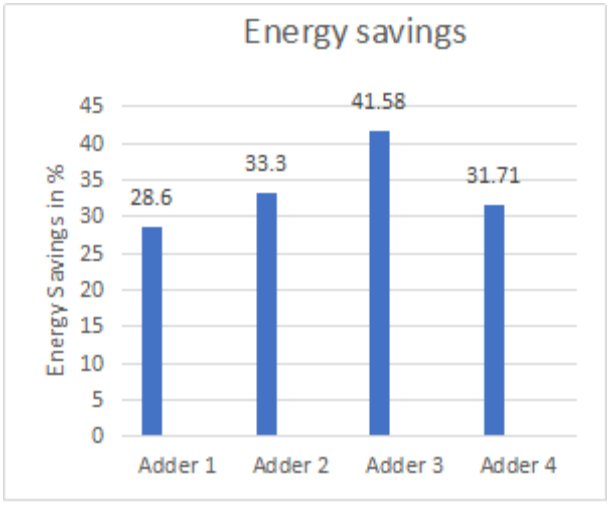


Fig 14 : Approximate adder 4

V. RESULTS

Table 1: Comparison Table

Multiplier	No of Transistors	LUT	Power (mW)	Delay (ns)	Energy (pJ)	Energy Saving (%)	Error 100*100	PSNR
Exact multiplier	24	102	163	11.498	1872.87		0	30.9058
Approximate Adder1	19	74	129	10.361	1336.569	28.6%	0.44	30.8297
Approximate Adder2	16	70	133	9.177	1248.072	33.3%	0.44	30.828
Approximate Adder3	14	68	129	8.481	1094.049	41.58%	2.36	30.744
Approximate Adder4	11	66	154	8.305	1278.97	31.71%	3.32	30.5851



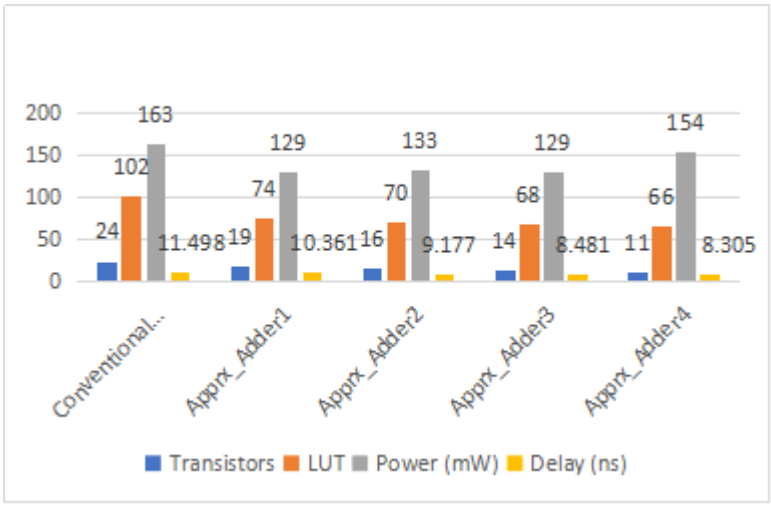


Fig15: Graph indicating energy saving

Table1 shows the comparison of different approximate multipliers using approximate adders and the exact multiplier in terms of various parameters.
From the Energy savings graph as in Fig 15, we can see that by using Approximate multiplier using Approximate adder 3, more amount of energy can be saved. Thus, it can be concluded that Approximate multiplier using approximate adder 3 is the efficient one when compared to other approximate multipliers.

VI. CONCLUSION AND FUTURE SCOPE

Since the main principle is to reduce the height of the accumulation tree the SDLC approach is used. An 8-bit approximate multiplier which uses exact adder in accumulation method is implemented. Four different approximate multipliers are implemented which use different approximate adders instead of exact adders in the accumulation method. The implementation was done in Verilog in Xilinx ISE. Exact multiplier was implemented and the delays of exact multiplier and the approximate multipliers are compared.
The low delay and area-efficient multiplier can be used for FIR filter design. We have implemented an unsigned 8-bit multiplier. The hardware implementation of an approximate multiplier for signed operation can be done. It can be downloaded into FPGA for further improvement. We can further implement for different number of bits and compare the results.

VII. REFERENCES

[1] L. Sekanina, "Introduction to approximate computing: Embedded tutorial," in DDECS, 2016, pp. 1–6.

[2] Sparsh Mittal, "A Survey of Techniques for Approximate Computing".

[3]Issa Qiqieh , Rishad Shafik ,Ghaith Tarawneh , Danil Sokolov,Shidhartha Das,Alexandre Yakovlev,"Significance-DrivenLogic Compression for Energy-Efficient Multiplier Design" in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2018.

[4] Bhupender Pratap Singh, Rakesh Kumar. "Design and Implementation 8-bit Wallace Tree Multiplier". International Journal of Advanced Reseach in Electrical, Electronics and Instrumentation Engineering. Volume 5, Issue 4, April 2016.

[5] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: Imprecise adders for low-power approximate computing," in ISLPED, 2011, pp. 409–414.

[6] Manikantta Reddy K, Nithin Kumar Y.B, Dheeraj Sharma3, Vasantha M.H. "Low Power, High Speed Error Tolerant Multiplier Using Approximate Adders".

[7] Leila kabbai, Anissa Sghaier, Ali Douik and Mohsen Machhout. "FPGA implementation of filtered image using 2D Gaussian filter". International Journal of Advanced Computer Science and Applications, Volume 7, No.7, 2016.

[8] Frank Cabello, Julio Leon, Yuzo lana , Rangel Arthur , "Implementation of a Fixed-Point 2D Gaussian Filter for Image Processing based on FPGA"

Fig16: Graph indicating no. of transistor, lut , power and delay