Comparative Review of Approximate Multipliers

Sushree Sila P. Goswami*†, Bikram Paul†, Sunil Dutt†, Gaurav Trivedi†
†*Dept. of Electronics and Electrical Engg., Indian Institute of Technology Guwahati, India*
Email: *sushree@iitg.ac.in

***Abstract*—In the digital signal processing (DSP) system, multiplier is a significant arithmetic module. It contributes mainly in the power consumption and speed, and efficient multipliers are the need of the hour. Approximate computing has added a unique dimension in the area of digital design by reducing area, power and delay. The demand of efficient approximate multipliers is enhancing due to the high speed and fault tolerance as well as its power efficiency. This paper presents comparison of few recent approximate multiplier designs. Experimental results based on the accuracy and circuit parameters are presented. The accuracy parameters are amplitude data accuracy (ACC_amp), information data accuracy (ACC_inf), error rate (ER) and mean error distance (MED). The circuit parameters are delay, power consumption, area. Based on these parameters, the best design in terms of power consumption and area is datapath complexity reduction approximate multiplier which exhibits 58% less power and 61% less area as compared to broken array multiplier.**

Keywords*—Approximate, multiplier, Accuracy, Power, Error Rate.

I. Introduction

A descriptive view of the accuracy parameters are proposed in [1] and [2], which are: Accuracy for amplitude data (ACCamp) : It is used to quantify the errors in the amplitude, and is represented as $ACC_(amp) = 1 - (R_c R_e)/R_e$, where R_c is the correct result and R_e is the result of the approximate multiplier. Accuracy for information data (ACCinf) measures error significance and can be represented as $ACC_{inf} = (1 - B_e)/B_w$, where B_e is the number of error bits and B_w is the bit width of the data. The error rate is defined as the probability of occurrence of an error. Mean Error Distance means value of the error distances of all possible outputs for each input of a circuit is called the mean error distance. A descriptive view of the circuit parameters are: *Delay:* The propagation delay or delay is the time delay between the 50%-transition of the rising/falling input voltage and the 50% -transition of the falling/rising output voltage. *Power Consumption:* When the transistor makes a transition from one state to another, there is a consumption of power in the process, which is calculated in this analysis. *Area:* Area is measured according to three factors, area covered (μm^2), number of gates and number of transistors.

II. Categories of Approximate Multipliers

A. Approximate Error Tolerant Multipliers:

1) 2×2 Under designed Multiplier:: This design is proposed in [3]. It introduces error into the multiplier using the 2×2 multiplier as a building block. The motivation behind this design is to introduce a 4-bit result in 3-bits. So, the multiplication of 3×3 will be 7 instead of 9. Fig. 1 shows the gate level circuit of the multiplier. Partial products are produced by using the inaccurate 2×2 block and then adding the shifted partial products larger multipliers are built. Fig. 2 shows the description of the design.

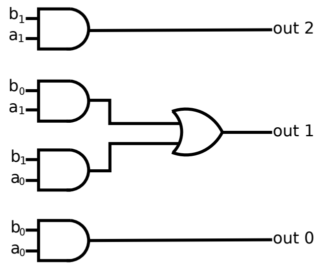


Fig. 1: Gate level circuit of the multiplier

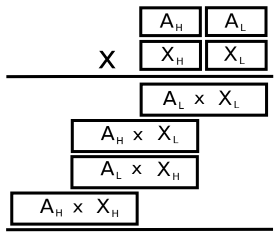


Fig. 2: Description of a large multiplier

2) Datapath complexity reduction approximate multiplier:: This multiplier design is proposed in [4]. Two changes have been carried out in [3] to design this approximate multiplier. First, by approximating out_0 to 0, we further approximate the 2×2 multiplier building block . Though the critical path of the resulting multiplier remains unchanged, the area has reduced. Fig. 3 shows the logic functions of the approximate 2×2 multiplier.

The second improvement that we propose is the manner in which a bigger multiplier is assembled. The

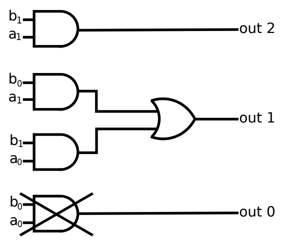


Fig. 3: Logic functions of the approx. 2×2 multiplier

multiplier in [3] utilizes the equivalent 2×2 surmised multiplier to figure every single incomplete item. Rather, three degrees of guess are presented inside the bigger multiplier. The least huge halfway item with greatest level of guess, the center incomplete items with medium estimation and the most huge fractional item with no guess are determined. The inspiration driving presenting three degrees of estimate is when two huge numbers are duplicated; just the lower fractional items are figured with guess. This improves the exactness of the multiplier contrasted with the reference multiplier.

3) Error Tolerant Multiplier: : The above mentioned approximate multiplier design is introduced in [5]. The multiplication algorithm can be explained as shown in Fig. 4. To begin with, the information operands are part into two sections: a duplication part in which various higher request bits are incorporated and a non-increase part which is comprised of the rest of the lower request bits. The duplication part experiences precise augmentation process. For the non-augmentation part, checking accomplished for each piece position from left to right (MSB to LSB) and the checking procedure is finished when either or both of the two operand bits are "1" and all the bit positions are set to "1" from that bit onwards. The situation when both operand bits are "0", the comparing item bit is set to "0".

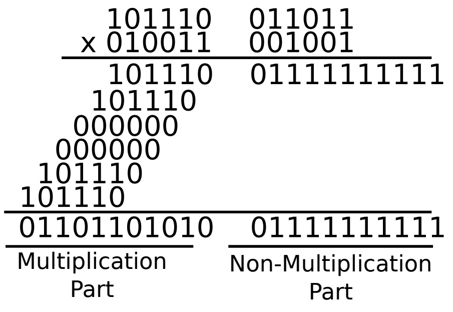


Fig. 4: Example of the approx. multiplication algorithm

4) Accuracy Configurable Multiplier(ACMA) :: The ACMA design is proposed in [6]. It improvises the recursive multiplication architecture and designs a new approximate pipelined architecture. The recursive multiplication builds a recursive tree. Let A be the multiplicand and X be the multiplier and both are assumed to be of $2b$ bits each. A and X can be written as $A = AHAL$ and $X = XHXL$, where each of them are of b bits each. The most critical $2b$ bits (out of an aggregate of $4b$ bits) considered as exact to a huge degree. Further, out of the least noteworthy $2b$ bits remaining, lower $b/2$ bits would be exact and upper $3b/2$ bits would be incorrect. The lower $b/2$ bits are kept exact in light of the fact that it doesn't require a great deal of equipment and simultaneously exactness increments. As it were, in the last item, out of $4b$ bits least huge $b/2$ bits will be exact and most huge $2b$ bits will be precise to a huge degree and remaining $3b/2$ bits will be off base. The approximate multiplication algorithm is shown in Fig. 5.

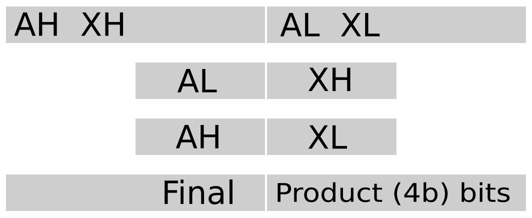


Fig. 5: The approx. multiplication algorithm of ACMA

B. Truncated Multipliers :

In [7], it is mentioned that multiplication of two numbers brings about an product with twice the original bit width. It is required to truncate the product bits to the desired accuracy to reduce area cost, leading to the structure of truncated multipliers, or fixed-width multipliers. By and large, there are two truncated multiplier structure techniques, in particular, steady and variable remedies, contingent upon how to repay the mistake acquainted due with the end of the least noteworthy halfway item (PP) bits. Consistent amendment structures diagnostically process the normal truncation mistake of the truncated PP bits and blunder is remunerated by including a line of steady into the lattice of the PP bits. Variable amendment plans diminish the all out truncation mistake by thinking about the least noteworthy piece of the PP bits.

Fixed-width multipliers is a subset of truncated multipliers which figure just n significant bits (MSBs) out of the $2n$-bit product for $n \times n$ increase and additional revision/remuneration circuits are utilized to diminish truncation errors. Fixed-width multipliers are generally utilized for advanced signal processor activities, for example, filtering, convolution, and fast Fourier or discrete cosine change. Along these lines, in this discussion, just the basic fixed width multiplier is examined with low error.

Low error fixed-width multiplier: The low error fixed-width multiplier is proposed in [8]. Parallel multiplier circuits consist of two parts: one for the higher order partial products (MP) and one for the lower order partial products (LP). The most basic fixed width multiplier removed the LP and substituted with a 0 to be carried to the residual part to generate MP. This reduced the area to half, but also generated a large amount of error. So, as to reduce the error, a circuit C_g is designed and is placed in place of 0. It consists of $n-2$ AO cells and one 2- input AND gate. Each AO cell consists of one 2-input AND gate and one 2-input OR gate. Each input of C_g is fed with a proper product term X_iY_j i.e. the partial products.

C. Bio-inspired imprecise multiplier

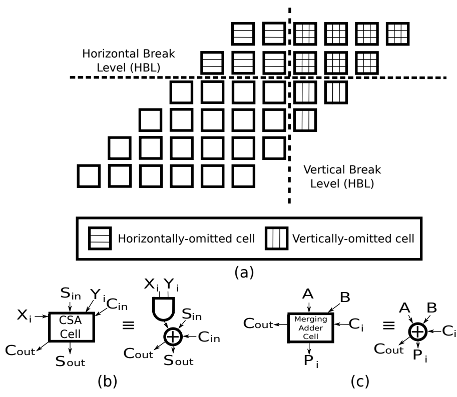
Instead of giving the precise value of the result, Bio-inspired Imprecise Computational blocks (BICs) [9] provide an applicable estimation of the same at a lower cost. In terms of area, speed, and power consumption, these novel structures are more efficient with respect to their precise counterparts. Here the Broken Array Multiplier or BAM is discussed which is a bio-inspired imprecise multiplier structure.

The Broken Array Multiplier (BAM): The broken-array multiplier (BAM), proposed in [9], is fundamentally the same as the design of an array multiplier. An Array multiplier with an m-bit multiplicand and n-bit multiplier comprises of mxn comparative cells called carry-save adder (CSA), trailed by a m-bit vector merging adder which converts carry-save redundant form to regular binary form. Every cell contains a 2-input AND gate to produce partial product and a full-adder (CSA) to add the partial products into the running sum.

A BAM breaks the CSA array and cells giving smaller and faster circuit while providing approximate results are eliminated. Depending on two introduced parameters namely Horizontal Break Level (HBL) and Vertical Break Level (VBL), the number and position of the eliminated cells are decided. The HBL=0 means that there is no horizontally eliminated CSA cell. If HBL increases, the horizontal dashed line moves downward and all cells falling above the dashed line are eliminated. So also the VBL=0 does not eliminate any CSA cells but as the VBL increases, the vertical dashed line moves left and eliminates all separated right-side cells. The respective outcome of the all eliminated cells are considered to be null. Fig. 6(a) shows the hardware structure of BAM, Fig. 6(b) shows the CSA cell and Fig. 6(c) shows the vector merging adder cell.

D. Inaccurate [4:2] Compressor based multipliers

A $[4:2]$ compressor accepts four equal weights as input and produces two outputs. It may be designed by



Horizontal Break Level (HBL)
Vertical Break Level (HBL)
Horizontally-omitted cell
Vertically-omitted cell
(a)
S_in Y_i C_in X_i CSA Cell C_out S_out
≡
X_i Y_i S_in C_in C_out S_out
(b)
A B C_out Merging Adder Cell C_i P_i
≡
A B C_in C_out P_i
(c)

Fig. 6: (a) BAM (b) CSA array (c) Vector merging Adder

using two $(3,2)$ counters. An intermediate carry, t_i is fed into the next column and accepts a carry, t_{i-1}, from the previous column, [10].

Inexact (or approximate) computing is an appealing paradigm for digital processing at nanometric scales. Approximate computing is especially interesting for computer arithmetic designs. The difference between an inaccurate computer and an ordinary counter is that an ordinary counter gives the outcome 1002 when all the 4 inputs are 1, but an inaccurate computer simplifies the outcome by representing the 3-bit outcome in 2-bits, [2]. This classification manages three inexact designs of inexact compressors to be applied on a Dadda Multiplier.

1) ***Design 1:****:* In an accurate compressor, out of 32 states, the carry output has the same value of the input C_{in} in 24 states. In Design 1, proposed in [11], by changing the value of the other eight outputs, the carry is simplified to:

$$C_{in} \times Carry = C_{in} \tag{1}$$

Being the higher weight of a binary bit, an erroneous value of Carry output signal will create a distinction estimation of two in the output. For example, if 01001 is given as the input pattern, then the right output is generated that is 010 which is equivalent to 2. By simplifying the carry output to Cin, the inexact compressor will produce the 000 pattern as output which is equivalent to value 0. This significant contrast can not be adequate. However, it has to be redressed or decreased by rearranging the C_{out} and Sum signals. Specifically, the difference between the approximate and the exact outputs can be reduced by simplifying the sum to a value of 0 and the the complexity of its design is also reduced. Additionally, as due to the presence of certain errors in the sum signal, the delay of creating the inexact sum reduces resulting in reduction of the overall delay of the design. Fig.7 shows the gate level structure of Design 1.

$$Sum = \overline{C_{in}}(\overline{(x_1 \oplus x_2)} + \overline{(x_3 \oplus x_4)}) \tag{2}$$

$$Cout = \overline{(\overline{(x_1x_2)} + \overline{(x_3x_4)})} \quad (3)$$

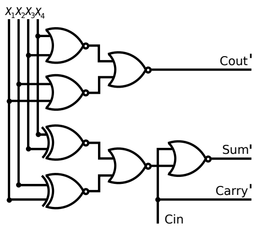


Fig. 7: The gate level structure of Design1

*2) **Design 2:**:* To additionally increase performance as well as to decrease the error rate, a subsequent design of an inexact compressor is also proposed in [11]. In the previous part, the proposed equations for the approximate $Carry$ and C_{out} can be interchanged as the $Carry$ and C_{out} outputs have the same weight,. In this new design, C_{out} is constantly equal to C_{in} and carry utilizes the right hand side of (3). As C_{in} is zero in the first stage, in rest of the stages, C_{out} and C_{in} will be zero. Therefore, in the hardware design, C_{in} and C_{out} can be overlooked. Fig. 8 shows the gate level structure of this inexact $4-2$ compressor and the expressions below gives its outputs.

$$Sum = \overline{C_{in}}(\overline{(x_1 \oplus x_2)} + \overline{(x_3 \oplus x_4)}) \quad (4)$$

$$Carry = \overline{(\overline{(x_1x_2)} + \overline{(x_3x_4)})} \quad (5)$$

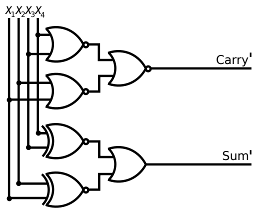


Fig. 8: The gate level structure of Design 2

*3) **Design 3:**:* In Design 3, proposed in [2], the main idea is to use a $2:1$ MUX in place of a XOR gate to reduce the delay and the power consumed by the multiplier. Let x_1, x_2, x_3 and x_4 are the four inputs and Sum and $Carry$ are the two outputs. The error occurs when all the four summands are 1 and the output 1112 is reduced to 102. It can be used to build an inaccurate 4×4 multiplier. It helps to reduce the adding stages from four to two to reduce the delay and power.

E. Approximate Logarithmic Multiplier :

A strategy of computer multiplication and division is proposed in [12], which utilizes binary logarithms. The logarithm of a binary number may be found approximately from the number itself by simple shifting and counting. A basic add or subtract and shift operation is needed to multiply or divide. Since the logarithms used are inexact, there can be errors in the outcome.

Simple Mitchell based logarithmic multiplier:: The simple Mitchell based logarithmic multiplier is proposed in [12]. It uses the concept of finding the multiplication of two numbers using binary logarithms. The arithmetic used is an approximation to the actual logarithm. To find the binary logarithm of a binary number, avoid the position of the most significant 1, and consider the number as a binary fraction. For example, approximate lg_{13} is 3.625 decimal and 11.101, where the largest characteristic is 3, since 13 consists of 4-bits. A step wise process is explained below.

Let A and B be two registers containing the two numbers, whose word size is of 8 bits. So, the largest characteristic will be 7.

Step 1: Shift A and B left until their most significant "one" bits are in the left-most positions. After the shifting is completed the counters will contain the characteristics of the logarithms of A and B.

Step 2: Shift bits $0-6$ of A and B into bit positions $0-6$ of registers C and D. C and D now contain the logarithms of the original numbers.

Step 3: Add $C + D \to E$. The result is now stored in register E.

III. Simulation Results:

The experimental results are divided in two parts:
(i) **Error Characteristics:** It discusses the accuracy parameters of all the aforesaid approximate multiplier designs architectures.
(ii) **Circuit Characteristics:** It discusses the circuit parameters of the five aforesaid approximate multiplier designs architectures from four of the five categories.

A. Error Characteristics:

The accuracy parameters are evaluated using MATLAB R2013a software for the approximate multiplication algorithms. It is observed that with the increase of bit width, the accuracy of the multiplier decreases. The approximate functions are simulated with 10000 random inputs. Some of the characteristic observations in the accuracy parameters are discussed here.

1) **Acc_amp:** The value of Acc_amp < 1. The maximum value is observed in 2×2 Under Designed Multiplier, of 0.98 for its 2-bit multiplier, while the minimum value is observed in Simple Mitchell based Logarithmic multiplier, of -27.018 for its 16-bit multiplier. It reaches a negative value for two designs, Datapath complexity reduction approximate multiplier and error tolerant multiplier.

Design Name	2×2 Under Designed Multiplier							
Parameters	2	4	6	8	10	12	14	16
Delay (ns)	10.0595	20.119	30.141	40.238	50.2975	60.357	70.4165	80.476
Power (μW)	2.6574	5.3148	7.9721	10.6295	13.2869	15.9443	18.6016	21.259
Area (μm^2)	15.6528	31.3055	46.9583	62.611	78.2638	93.9165	109.5693	125.222
Area (gates)	13	25	38	50	62	75	88	100
Area (transistors)	84	168	252	336	420	504	588	672
ER	0.054	0.882	0.972	0.991	0.999	1	1	1
MED	0.108	46.96	968.35	1.63E+04	2.63E+05	4.21E+06	6.70E+07	1.07E+09
Acc_amp	0.98	0.296	0.082	0.082	0.009	0.004	0.0022	0.0011
Acc_inf (E-04)	9.59	6.79	6.21	5.9	5.69	5.6	5.52	5.48

TABLE I: 2×2 Under Designed Multiplier

Design Name	Datapath complexity reduction approximate multiplier							
Parameters	2	4	6	8	10	12	14	16
Delay(ns)	10.0595	20.119	30.141	40.238	50.2975	60.357	70.4165	80.476
Power(μW)	2.6139	5.2278	7.8413	10.4555	13.0694	15.6832	18.2971	20.911
Area (μm^2)	15.4109	30.8219	46.2328	61.6437	77.0546	92.4656	107.8765	123.2874
Area (gates)	12	24	37	49	61	74	87	99
Area (transistors)	83	165	247	330	412	494	576	658
ER	0.553	0.842	0.974	0.992	0.998	1	1	1
MED	1.72	95.22	327.23	2.62E+04	1.61E+05	1.29E+07	6.08E+07	6.70E+09
Acc_amp	0.528	-0.0781	0.53	-0.156	0.287	-1.23	0.117	-3.187
Acc_inf(E-04)	7.82	6.19	6.15	5.66	5.61	5.41	5.45	5.28

TABLE II: Datapath complexity reduction

Design Name	Error tolerant Multiplier							
Parameters	2	4	6	8	10	12	14	16
ER	0.702	0.928	0.98	0.994	1	0.999	1	1
MED	41.42	115.65	253.68	1.29E+04	2.41E+05	4.21E+06	6.74E+07	1.03E+09
Acc_amp					0.052	0.015	0.0055	0.0024
Acc_inf(E-04)	2.81	5.34	5.89	5.33	5.4	5.57	5.41	5.41

TABLE III: Error tolerant Multiplier

Design Name	Data for ACMA multiplier							
Parameters	2	4	6	8	10	12	14	16
ER	0.262	0.855	0.972	0.993	0.996	1	1	1
MED	0.524	49.17	920.53	1.65E+04	2.57E+05	4.11E+06	6.84E+07	1.04E+09
Acc_amp	0.905	0.271	0.0937	0.331	0.0168	0.0063	0.0033	0.0015
Acc_inf(E-04)	8.89	6.77	6.24	5.89	5.8	5.64	5.5	5.52

TABLE IV: ACMA multiplier

Design Name	Data for Simple Mitchell based Logarithmic multiplier							
Parameters	2	4	6	8	10	12	14	16
ER	0.949	0.996	1	1	1	1	1	1
MED	2.89	45.68	710.46	1.17E+04	1.98E+05	2.98E+06	4.80E+07	7.29E+08
Acc_amp						-12.78	-7.61	-27.018
Acc_inf(E-04)	6.04	6.06	5.83	5.57	5.47	5.37	5.31	5.29

TABLE V: Simple Mitchell based Logarithmic multiplier

Design Name	Data for Low error fixed width multiplier							
Parameters	2	4	6	8	10	12	14	16
Delay(ns)	10.0695	20.139	30.2085	40.278	50.3475	60.417	70.4865	80.556
Power(μW)	4.5469	9.0938	13.6406	18.1875	22.7344	27.2813	31.8281	36.375
Area (μm^2)	27.898	55.7959	83.6939	111.5918	139.4898	167.3877	195.2857	223.1836
Area (gates)	20	40	60	80	100	120	140	160
Area (transistors)	146	291	437	582	728	873	1019	1164
ER	0.508	0.888	0.964	0.989	1	1	0.999	1
MED	1.39	43.92	783.96	1.17E+04	1.87E+05	2.98E+06	4.68E+07	7.62E+08
Acc_amp	0.662	0.306	0.189	0.188	0.192	0.2174	0.223	0.2438
Acc_inf(E-04)	8	6.52	6.01	6.02	5.81	5.81	5.71	5.66

TABLE VI: Low error fixed width multiplier

Design Name	Data for Broken Array Multiplier							
Parameters	2	4	6	8	10	12	14	16
Delay(ns)	10.0788	20.1575	30.2363	40.315	50.3938	60.4725	70.5513	80.63
Power(μW)	6.2617	12.5234	18.7851	25.0468	31.3085	37.5702	43.8319	50.0936
Area (μm^2)	43.5068	87.0135	130.5203	174.027	271.5338	261.0405	304.5473	348.054
Area (gates)	32	63	95	126	158	189	221	252
Area (transistors)	221	441	661	881	1101	1321	1541	1761
ER	0.421	0.873	0.97	0.986	0.999	1	1	1
MED	1.27	52.58	1.03E+03	1.67E+04	2.49E+05	4.29E+06	6.86E+07	1.09E+09
Acc_amp	0.729	0.192	0.042	0.016	0.0033	6.60E-05	1.19E-05	2.52E-06
Acc_inf(E-04)	8.78	6.54	6.02	5.78	5.7	5.54	5.46	5.4

TABLE VII: Broken Array Multiplier

Design Name	Data for Inaccurate [4:2] compressor based multiplier (Design 1)						
Parameters	2	4	6	8	10	12	14
ER	0.58	0.893	0.96	0.995	0.999	1	1
MED	2.31	59.55	971.93	1.62E+04	2.59E+05	4.35E+06	6.78E+07
Acc_amp	0.412	0.117	0.044	0.0065	0.0014	1.16E-04	2.72E-05
Acc_inf(E-04)	7.7	6.59	6.08	5.1	5.56	5.43	5.36

TABLE VIII: Inaccurate [4:2] compressor multiplier(1)

Design Name	Data for Inaccurate [4:2] compressor based multiplier (Design 2)						
Parameters	2	4	6	8	10	12	14
Delay(ns)	10.0588	20.1175	30.1763	40.235	50.2938	60.3525	70.4113
Power(μW)	4.5469	9.0938	13.6407	18.1875	22.7343	27.2813	31.8281
Area (μm^2)	30.9978	61.9955	92.99318	123.9909	154.9886	185.9863	216.984
Area (gates)	24	48	72	96	120	144	168
Area (transistors)	161	322	483	644	805	966	1127
ER	0.56	0.875	0.972	0.993	0.999	1	1
MED	2.26	52.1	967.3	1.57E+04	2.59E+05	4.27E+06	6.37E+07
Acc_amp	0.44	0.13	0.0319	0.0085	1.40E-03	1.26E-04	2.88E-05
Acc_inf(E-04)	7.81	6.71	6.04	5.72	5.58	5.38	5.43

TABLE IX: Inaccurate [4:2] compressor multiplier(2)

Design Name	Data for Inaccurate [4 : 2] compressor based multiplier (Design 3)						
Parameters	2	4	6	8	10	12	14
ER	0.55	0.881	0.977	0.991	1	1	1
MED	2.2	58.42	970.36	1.63E+04	2.56E+05	4.23E+06	6.89E+07
Acc_amp	0.45	0.125	0.02741	0.0104	4.13E-04	1.37E-04	2.84E-05
Acc_inf(E-04)	7.8	6.55	6.01	5.72	5.55	5.48	5.43

TABLE X: Inaccurate [4:2] compressor multiplier(3)

An interesting observation can be seen in this multiplier is that the value becomes negative after an interval of 4-bits. The value of Acc_amp is countable for this multiplier only from 10-bits onward and for Simple Mitchell based Logarithmic multiplier after 12-bits. This phenomenon occurs due to large amount of error present for the lower bits. For the approx. compressor based multiplier designs, all the designs show almost the same results. The results follow the order of Design 3, 2 & 1.

2) **Acc_inf:** The values of Acc_inf lie in $\times 10^{-4}$ range. All the designs follow the same trend and even almost the same values for various bit widths. The maximum value is 9.59×10^{-4} for 2×2 Under Designed Multiplier for its 2-bit multiplier, while the minimum value is 5.28×10^{-4} for Datapath complexity reduction approximate multiplier, for its 16-bit multiplier. For the approx. compressor based multiplier designs, all the designs show almost the same results. The results follow the order of Design 3, Design 2 and Design 1.
3) **ER:** The value of ER decreases as the bit width increases and the value saturates to 1 at higher bit widths. The least values of ERs are 0.054 for 2×2 Under Designed Multiplier and 0.262 for ACMA Multiplier for their respective 2-bit multipliers. For the approx. compressor based multiplier designs, all the designs show almost the same results. Although, Design 3 shows better results than Design 2 and Design 1, its ER reaches 1 for a 10×10 multiplier and the ERs of other two reach 1 only at 11×11 multiplier.
4) **MED:** For all the designs, the values of MED lie in a very large interval. It is seen that for higher bits (> 10), for each increment of 1 bit, the value of MED increases by 10 times. Two unusual observations are observed are:
   - For error-tolerant multiplier, at a 6×6 bit multiplier, MED reaches to a very low value of 253.68 compared to other designs.

- In broken array multiplier, also for 6×6 bit multiplier, MED is rather larger value of 1020, compared to other designs. For the approx. compressor based multiplier designs, all the designs show almost the same results. Design 3 shows the best result, followed by Design 2 and 1.

B. Circuit Characteristics:

Among the five categories presented, circuit characteristics of five approximate multipliers from four categories have been discussed. It is to be notified that the circuit characteristics are evaluated by using the TANNER EDA, software.

1) **Delay:** The delays of the designs are of the order of (ns). Considering a 15-bit multiplier, largest delay is observed in Broken Array Multiplier of $75.5906ns$, while the smallest delay is observed in Design 2 of approximate compressor based multipliers, of $75.446ns$. Since, Datapath complexity reduction approximate multiplier is designed from 2×2 Under Designed Multiplier, both have the same delay, of $80.476ns$ for a 16-bit multiplier.
2) **Power:** The power consumptions are in the order of (μW). Considering a 16-bit multiplier, largest power consumption is observed in Broken Array Multiplier of $50.0936\mu W$, while the smallest power consumption is observed in Datapath complexity reduction approximate multiplier of $20.911\mu W$. Power consumed in Design 2 of approximate compressor based multipliers is also very high of $34.1016\mu W$.
3) **Area :** Area is measured according to three factors, area covered in (μm^2), number of gates and the number of transistors. For a 16-bit multiplier, the largest number of gates and transistors are used in Broken Array Multiplier, 252 and 1761 respectively. Similarly, for a 16-bit multiplier, the smallest number of gates and transistors are used in Datapath complexity reduction approximate multiplier, 99 and 658 respectively. In terms of area covered, Datapath complexity reduction approximate multiplier covers the least area of $123.2874\mu m^2$ for a 16-bit multiplier, followed by 2×2 Under Designed Multiplier with an area of $125.222\mu m^2$, while the largest area covered by Broken Array Multiplier, with an area of $348.054\mu m^2$ for a 16-bit multiplier.

IV. Discussion and Conclusion:

In this paper, approx. multipliers are reviewed; their error and circuit characteristics are evaluated. Based on accuracy, the category of approx. error tolerant multipliers provide the best results; the two best designs will be Error tolerant Multiplier and 2×2 Under Designed Multiplier. For the lower bit multipliers, the design provided by 2×2 Under Designed Multiplier is very effective. But for higher bit multiplier structure, Error tolerant Multiplier provides a better performance. Based on circuit characteristics, the smallest delay is observed in the Design 2 of approximate compressor based multipliers, but its power consumption and area covered is pretty high. The least power consumed and area covered design is in Datapath complexity reduction approximate multiplier. Its delay is also not that large, so it is the best design based on circuit characteristics.

References

[1] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.

[2] C. Lin and I. Lin, "High accuracy approximate multiplier with error correction," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, Oct 2013, pp. 33–38.

[3] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *2011 24th Internatioal Conference on VLSI Design*, Jan 2011, pp. 346–351.

[4] M. Vasudevan and C. Chakrabarti, "Image processing using approximate datapath units," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2014, pp. 1544–1547.

[5] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo, "Low-power high-speed multiplier for error-tolerant application," in *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, Dec 2010, pp. 1–4.

[6] K. Bhardwaj and P. S. Mane, "Acma: Accuracy-configurable multiplier architecture for error-resilient system-on-chip," in *2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, July 2013, pp. 1–6.

[7] H. Ko and S. Hsiao, "Design and application of faithfully rounded and truncated multipliers with combined deletion, reduction, truncation, and rounding," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 5, pp. 304–308, May 2011.

[8] Jer Min Jou and Shiann Rong Kuang, "Design of low-error fixed-width multiplier for dsp applications," *Electronics Letters*, vol. 33, no. 19, pp. 1597–1598, Sep. 1997.

[9] 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 Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, April 2010.

[10] Weste, H. Neil H.E., and D. M. USA: Pearson, 2010, p. 486.

[11] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, April 2015.

[12] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, Aug 1962.