IMPLEMENTATION OF LOW-POWER APPROXIMATE UNSIGNED MULTIPLIERS WITH CONFIGURABLE ERROR RECOVERY

G. Tejasree[1], P.M. Naseer Hussain[2], Dr.S.Siddeswara Reddy[3]

[1]PG scholar, Dept of ECE, VITS, Proddatur, Kadapa, AP, India

[2] Assistant Professor, Dept of ECE, VITS, Proddatur, Kadapa, AP, India

[3] Associate Professor & HOD , Dept of ECE, VITS, Proddatur, Kadapa, AP, India

Abstract: Inexact circuits have been considered for applications that can endure some deficiency of precision with further developed execution or potentially energy effectiveness. Multipliers are key number juggling circuits in a considerable lot of these applications including DSP. In this paper, a clever estimated multiplier with a low power utilization and a short basic way is proposed for elite execution DSP applications. This multiplier uses a recently planned estimated addition that restricts it convey spread to the closest neighbors for quick halfway item gathering. Various degrees of exactness can be accomplished by utilizing either OR entryways or the proposed surmised addition in a configurable blunder recuperation circuit. The approximate multipliers using these two error reduction strategies are referred to as AM1 and AM2, respectively. Both AM1 and AM2 have a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared with a Wallace multiplier optimized for speed, an 8×8 AM1 using four most significant bits for error reduction shows a 60% reduction in delay (when optimized for delay) and a 42% reduction in power dissipation (when optimized for area).

Index Terms— Approximate computing, multiplier, adder, error recovery, low-power, image processing.

1. Introduction

We have an abundance of new information, not just from big, efficient research and business machines, but also from billions of low-power devices of different sorts. While conventional workloads like transactional or database analysis continue to develop modestly, a variety of systems are explosively computerized to achieve greater insight in large quantities of structured and unstructured data. Traditional computing implies an accuracy which is not required in most types of information. Nevertheless, though, these computational systems stay on high-precision (and reliable) frameworks for general purposes (and accelerator). The goal of approximate computing is to loosen these restrictions in order to achieve substantial computational performance improvements-while preserving reasonable output.

A main research aim in estimated calculation is to define the estimation thresholds within various layers of the machine stack (from computers to circuits and electronic components) to enable for an appropriate but probably specific outcome from the results obtained using precise calculation. Approximate computational techniques researched by different researchers centred mainly on improving one component of the device stack. In this paper we explored how the use of several solution approaches spanning upwards of one layer of the device stack added value and whether such composite effects became widely available across various implementation areas.

We focused on three types of approximations to provide a practical demonstration: missing measurements, approximating numerical computations and approximating coordination between computing elements. We test loop perforation, decreased arithmetic precision and relaxing of synchronization as representations of each division. We have chosen applications which are computer-priced which can affect our lives dramatically if they are inexpensive and widespread. Digital image processing, robots and machine intelligence is our research fields. Our business. Our findings show in all of the applications tested that in the applications tested we were in a position to perforate hot loops by an average of 50%, thus that the total execution time proportionately while also delivering reasonable results efficiency. Furthermore, the data used to calculate the current 32 bits or even 64 bits could be reduced to 10-16 bits, with potential significant quality and power benefits. Truncated out the simultaneous applications we analyzed; the execution period was decreased by 50 percent by partial deletion of overhead synchronization.

Eventually, our results show that the advantages of these methods become amplified when used together. In other terms, the use of different strategies does not substantially decrease one another's efficacy. Seeing that the advantages of approximation computation are not confined to a particular class of programmers, these findings enable one to reconsider the design of the general function of the system such that various forms of approximations may be taken into account more effectively.

Rounding is one of the most effective methods to pack data input before processing. This approach will boost circuit features such as energy and usage, speed and range which are appropriate for estimated computation. Approximate computer technology works well with most of computer vision , image

processing, design recognition, image analysis, computer science, and machine learning applications that are resilient to errors. Work into these fields has created plenty of work resources Truncatedout the past few years. A multiplier is a basic computing block one of the most commodities consuming. Along this point, we are seeing endless studies along accuracy and energy loss dramatically compromised. Partial product production, partial product reduction and packaging are the fundamental building blocks of the converter. This paper suggests rounding as a modern input frame approach until partial product output. The precision curve plays a vital function in managing and raising the number of errors.

Guess it depends on requests, considerable. Specific algorithms on varying types of multiplier frames are applied. The input block has a 16bit and 32bit rounding technique based on precision levels. The intermediate results generated are categorized into active or partial products that are inactive. Both unused sections are nil and thus no compressors can be taken into account in the reduction process.

2. Design of proposed multiplier

A. The Approximate Adder

In this part, the plan of another inexact addition is introduced. This addition works on a bunch of pre-handled information sources. The information pre-handling (IPP) depends on the commutativity of pieces with similar loads in various addends. For instance, think about two arrangements of contributions to a 4-cycle addition: I) A = 1010, B = 0101 and ii) A = 1111, B = 0000. Unmistakably, the augmentations in I) and ii) produce a similar outcome. In this cycle, the two info bits Ai Bi = 01 are comparable toAi Bi = 10 (with I being the piece file) because of the commutativity of the relating pieces in the two operands. The fundamental principle for the IPP is to switch Ai and Bi if Ai = 0 and Bi = 1 (for any I), while keeping different blends (i.e., Ai Bi = 00, 10 and 11) unaltered. Thusly, more 1's are normal in An and more 0's are normal in B. Assuming that A˙i B˙ I are the ith bits in the pre-handled data sources, the IPP capacities are given by:

$$\dot{A}_i = A_i + B_i, \quad (1)$$

$$\dot{B}_i = A_i B_i . \quad (2)$$

Table 1: Truth table of the inexact addition cell. 'x' addresses that no such a mix happens due to the ipp

S_i/E_i		$\dot{B}_i\dot{B}_{i-1}$			
		00	01	11	10
$\dot{A}_i\dot{A}_{i-1}$	00	0/0	X	X	X
	01	0/0	1/0	X	X
	11	1/0	1/1	1/0	0/0
	10	1/0	X	X	0/0

Conditions (1) and (2) register the proliferate and produce signals utilized in

an equal addition, for example, the convey lookahead addition (CLA). The proposed addition can deal with information in equal by cutting the convey spread chain. Allow An and B to indicate the two info double operands of an addition, S be the total outcome, and E address the blunder vector. Simulated intelligence,Bi , Si and Ei are the ith least huge pieces of A, B, S and E, separately. A convey proliferation chain begins at the ith bit when B˙i = 1, A˙i+1 = 1, B˙i+1 = 0. In an exact addition, Si+1 is 0 and the convey spreads to the higher piece. In any case, in the proposed Truncated addition, Si+1 is set to 1 and aerror signal is produced as Ei+1 = 1. This forestalls the convey signal from spreading to the higher pieces. Consequently, a convey signal is created simply by the produce signal, i.e., Ci = 1 just when B˙i = 1, and it just spreads to the following higher piece, i.e., the (I +1)th position. Table I shows reality table of the surmised addition, where A˙i, B˙i and B˙i−1 are the contributions after IPP. The error signal is used for blunder pay purposes as talked about in a later segment. For this situation, the surmised addition is like an excess number framework [24] and the sensible elements of Table I are given by

$$Si = B˙i−1 + B˙i\, A˙i, \quad (3)$$

$$Ei = B˙i\, B˙i−1\, A˙i\,. \quad (4)$$

By supplanting A˙i and B˙i utilizing (1) and (2) individually, the rationale capacities regarding the first information sources are given by

$$Si = (Ai \oplus Bi) + Ai−1Bi−1, \quad (5)$$

$$Ei = (Ai \oplus Bi)Ai−1Bi−1, \quad (6)$$

where I is the piece record, i.e., I = 0, 1,⋯ , n for a n-digit addition. Let A−1 = B−1 = 0 when I is 0, consequently, S0 = A0 ⊕ B0 and E0 = 0. Likewise, Ei = 0 when Ai−1 or Bi−1 is 0. Consider a n-cycle addition, the data sources are given by A = An−1 ⋯ A1A0 and B = Bn−1 ⋯ B1B0, the specific total is S = Sn−1 ⋯S1S0. Then, at that point, Si can be figured as Si + Ei and hence, the specific amount of An and B is given by

$$S = S + E. \quad (7)$$

In (7) '+' signifies the expansion of two paired numbers rather than the 'OR' work. The error E is consistently non-negative and the surmised total is consistently equivalent to or more modest than the precise total. This is a significant component of this addition on the grounds that an extra addition can be utilized to add the blunder to the surmised total as a pay step. While this is instinctive in a addition plan, it is an especially valuable element in a multiplier plan as just a single extra addition is expected to lessen the blunder in the eventual outcome

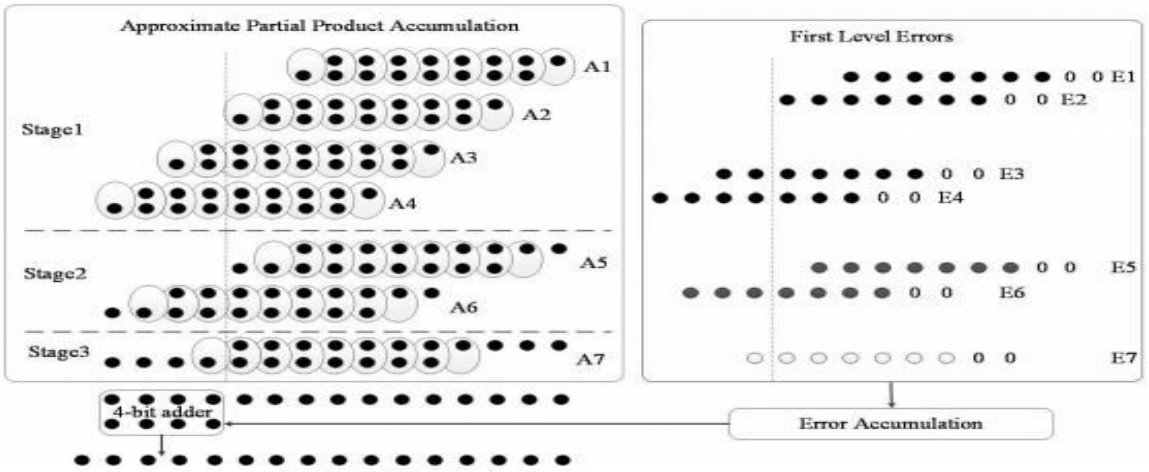

Fig. 1. A Truncated multiplier with incomplete blunder recuperation utilizing 5 MSBs of the error vector: a halfway item, aggregate or a blunder bit created at the primary stage; : a error bit produced at the subsequent stage; : a blunder bit created at the last stage.

B. Proposed Approximate Multiplier

A distinctive component of the proposed surmised multiplier is the effortlessness to utilize inexact adders in the fractional item gathering. It has been shown that this might prompt low precision [14], on the grounds that blunders might amass and it is hard to address errors utilizing existing surmised adders. In any case, the utilization of the recently proposed surmised addition conquers this issue by using the error signal. The subsequent plan has a basic way postpone that is more limited than a regular the slightest bit full addition, on the grounds that the new n-cycle addition can handle information in equal. The surmised addition has a fairly high error rate, however the component of creating both the total and blunder flags simultaneously diminishes blunders in the end result. A addition tree is used for fractional item amassing; the errorsignals in the tree are then used to repay the blunder in the result to create an item with a superior precision. The engineering of the proposed estimated multiplier is displayed in Fig. 1. In the proposed plan, the disentanglement of the incomplete item aggregation stage is cultivated by utilizing a addition tree, in which the quantity of fractional items is decreased by a variable of 2 at each phase of the tree. This addition tree is generally not executed utilizing precise multi-bit adders because of the long inertness. In any case, the proposed estimated Fig. 2. Images for (a) an OR entryway, (b) a full addition or a half addition and (c) a Truncated addition cell addition is reasonable for carrying out a addition tree, since it is less intricate than an ordinary addition and has a lot more limited basic way delay. Careful quick

multipliers frequently incorporate a Wallace or Dadda tree utilizing full adders (FAs) and half adders (HAs); blowers are additionally used in the Wallace or Dadda tree to additionally lessen the basic way with an expansion in circuit region. These plans require an appropriate determination of various circuit modules; for instance, 4:2 blowers, FAs and HAs are usually utilized in a Wallace tree and a wise association of these modules should be considered in a tree plan. This builds the plan intricacy, particularly when multipliers of various sizes are thought of; the proposed configuration is straightforward for different multiplier sizes.

III. ERROR REDUCTION

The inexact addition produces two signals: the surmised total S and the blunder E; the utilization of the error signal is viewed as close to lessen the error of the multiplier. As (7) is material to the amount of each and every surmised addition in the tree, a blunder decrease circuit is applied to the last increase result rather than to the result of every addition. Two stages are needed to decrease errors: I) blunder gathering and ii) blunder recuperation by the expansion of the collected blunders to the addition tree yield utilizing a CPA. In the blunder amassing step, error signals are aggregated to a solitary error vector, which is then added to the result vector of the fractional item collection tree. Two inexact blunder gathering strategies are proposed, yielding the Truncated multiplier 1 (AM1) and surmised multiplier 2 (AM2). Fig. 2 shows the images for an OR entryway, a full addition and half addition cell and an inexact addition cell utilized in the blunder amassing tree.

A. Error Accumulation for Approximate Multiplier 1

As displayed in Fig. 1, each surmised addition Ai produces a total vector Si and a error vector Ei, where I = 1, 2,⋯ , 7. In the event that the error signals are added utilizing precise adders, the gathered blunder can completely remunerate the erroneous item; but to lessen intricacy, an inexact blunder collection is presented.

Consider the perception that the error vector of each estimated addition will in general have more 0's than 1's. Thusly, the likelihood that the error vectors have a blunder bit '1'at a similar position, is tiny. Henceforth, an OR entryway is utilized to Truncated process the amount of the errors for a solitary cycle. In the event that m blunder vectors (signified by E1, E2, ..., Em) must be amassed, then, at that point, the amount of these vectors is acquired as Ei = E1i OR E2i OR ... Or then again Emi.

To lessen blunders, an aggregated error vector is added to the addition tree yield utilizing an ordinary CPA (for example a convey lookahead addition).

Notwithstanding, just a few (for example k) MSBs of the error signals are utilized to remunerate the results to additionally lessen the general intricacy. The quantity of MSBs is chosen by the degree that blunders should be redressed. For instance in a 8 × 8 addition tree, there are a sum of 7 blunder vectors, produced by the 7 estimated adders in the tree. Nonetheless, not every one of the pieces in the 7 vectors should be added, on the grounds that the MSBs of certain vectors are less huge than the most un-critical pieces of the k MSBs. In the case of Fig. 1, 5 MSBs (for example the (11 − 14)th pieces, no blunder is produced at the fifteenth piece position) are considered for error recuperation and in this manner, 4 error vectors are thought of (i.e., the error vectors E3, E4, E6 and E7). The error vectors of the other three adders are less huge than the eleventh piece, so they are not thought of. The gathered error E is acquired utilizing (8); then, at that point, the end-product is found by adding E to S utilizing a quick precise CPA. The error collection conspire is displayed in Fig. 3. As no error is created essentially critical two pieces of each inexact addition Ai (I = 1, 2,⋯ , 7), the most un-huge two pieces of every blunder vector Ei are not aggregated.

B. Blunder Accumulation for Approximate Multiplier 2

The error gathering plan for AM2 is displayed in Fig. 4. To present the plan of AM2, a 8 × 8 multiplier with two data sources X and Y is thought of. For instance, consider the initial two incomplete item vectors X0Y7, X0Y6, ..., X0Y0 and X1Y7, X1Y6, ..., X1Y0 amassed by the primary inexact addition (A1 in Fig. 1), where Xi and Yi are the ith least critical pieces of X and Y , individually.

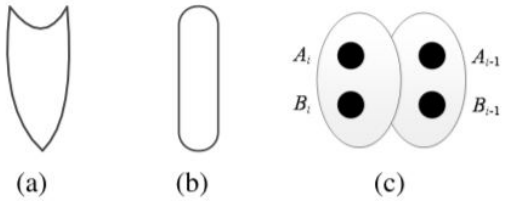

Fig. 2. Symbols for (a) an OR gate, (b) a full adder or a half adder and (c) an approximate adder cell

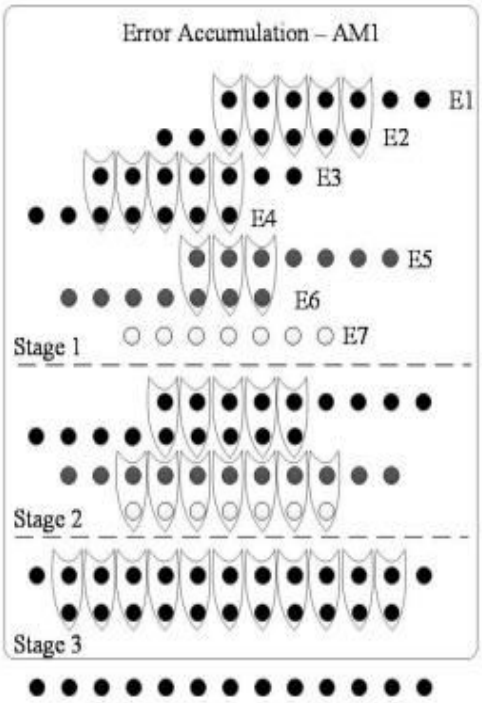

Fig. 3. Error accumulation tree for AM1.

Review from (6) for the estimated addition, the condition for Ei = 1 is Ai−1 = Bi−1 = 1 and Ai = Bi . (9) For the principal Truncated addition in the fractional item aggregation tree, its bits of feedbacks are A = X0Y7, X0Y6, ..., X0Y0 and B = X1Y7, X1Y6, ..., X1Y0. Subsequently, the ith least huge pieces for An and B are Ai = X0Yi and Bi = X1Yi−1, separately.

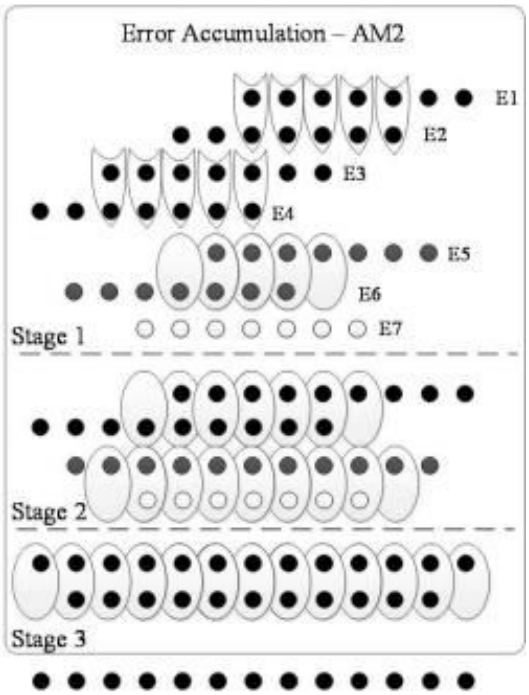

Fig. 4. Error accumulation tree for AM2.

On the off chance that X0 or X1 is 0, there will be no error in this Truncated addition in light of the fact that either An or B is zero. In this manner, no blunder happens except if X0X1 = 11. At the point when X0X1 = 11, Ai and Bi are rearranged to Yi and Yi−1, individually. Then, at that point, to ascertain Ei , Ai−1, Bi−1, Ai and Bi are supplanted by Yi−1, Yi−2, Yi and Yi−1, separately. For Ei to be 1, YiYi−1Yi−2 = 011 as indicated by (9). Hence, aerror possibly happens when the info has "011" as a piece succession. In light of this perception, the "distance" between two errors in a Truncated multiplier is no less than 3 pieces. In this manner, two adjoining estimated adders in the primary phase of the fractional item tree can't have blunders at a similar section, on the grounds that the errors in a lower inexact addition are those in the upper addition moved by 2 pieces when the two errors exist. The blunders in two adjoining inexact adders would then be able to be precisely amassed by OR entryways, e.g., an OR door can be utilized to aggregate the two pieces in the error vectors E1 and E2 in Fig. 1. Subsequent to applying the OR entryways to aggregate E1 and E2 just as E3 and E4, the four error vectors are compacted into two. For E5, E6 and E7, they are created from the surmised amount of the fractional items rather than the incomplete items. Hence, they can't be precisely collected by OR additions.

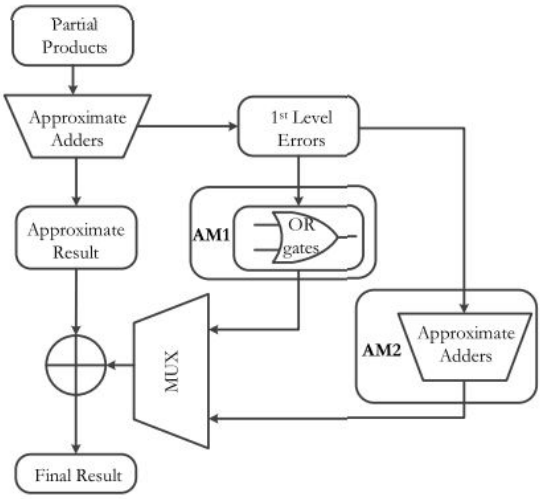

Fig. 5. Block graph of the proposed multipliers.

One more fascinating component of the proposed Truncated addition is as per the following. Accept Ei = 1 in (6), then, at that point, Ai−1 = Bi−1 = 1 and Ai = Bi. Since Ai−1 = Bi−1 = 1, i.e., Ai−1 ⊕ Bi−1 = 0, it is not difficult to show that Ei−1 = 0. Additionally, as Ai = Bi, i.e., Ai Bi = 0, then, at that point, Ei+1 = 0. In this way, once there is aerror in the slightest bit, its adjoining pieces are sans blunder, i.e., there are no continuous error bits in a single line. Along these lines, there is no convey proliferation way longer than two pieces when two error vectors are gathered, and the blunder vectors are precisely collected by the proposed Truncated addition. In view of the above examination, E5 and E6 are precisely aggregated by one estimated addition in the primary phase of the error amassing. After the main phase of blunder collection, three vectors are produced, and one more two inexact adders are then used to gather these three vectors just as the error vector staying from the past stage (E7). Recreation results (found in later areas) show that the changed error aggregation outflanks the OR-door blunder collection with minimal overhead on deferral and power. In the future, the proposed n × n inexact multiplier with k-MSB OR-door based blunder decrease is alluded to as a n/k AM1, while a n × n estimated multiplier with k-MSB surmised addition basederror decrease is alluded to as a n/k AM2. The constructions of AM1 and AM2 are displayed in Fig. 5.

C. 16 × 16 Approximate Multipliers

In both AM1 and AM2, all the error vectors are packed to one blunder vector, which is then added back to the Truncated result of the halfway item tree. Contrasted with 8 × 8 plans, 16 × 16 multipliers produce more blunder vectors, and a lot of data would be disregarded if a similar error decrease techniques are utilized. That is, utilizing just one packed blunder vector doesn't make a decent gauge of the general error. In the altered plans, the blunder vectors created by the inexact adders are packed to two last error vectors. Take a 16 × 16 AM1 for instance, he eight blunder vectors created at the primary phase of the fractional item gathering tree are packed to one error vector, EV1, utilizing OR entryways. The leftover seven error

vectors from the second, third and fourth stages are packed to another blunder vector EV2. Then, at that point, both EV1 and EV2 are added back to the result of the halfway item at the fourth stage. Essentially, the proposed inexact adders are utilized in a 16×16 AM2 to pack the eight error vectors from the principal stage to one blunder vector and the leftover error vectors to another blunder vector. Truncation can likewise be applied to the proposed plans for huge info operands. Accordingly, 16 LSBs of the incomplete items are Truncated in 16×16 AM1 and AM2, coming about in Truncated AM1 (TAM1) and Truncated AM2 (TAM2).

4.Simultation Results

Figure 6: Simulation outcome

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	4982	204000	2%
Number of fully used LUT-FF pairs	0	4982	0%
Number of bonded IOBs	131	600	21%

Figure 7: Design summary

The above result represents the synthesis implementation by using the Xilinx ISE software. From the above table, it is observed that only 4982 look up tables are used out of available204000. It indicates very less area (2%) was used for the proposed design.

```
LUT2:I0->O            1   0.043   0.000  div1/Madd_GND_49_o_GND_49_o_a
MUXCY:S->O            1   0.230   0.000  div1/Madd_GND_49_o_GND_49_o_a
XORCY:CI->O           2   0.251   0.347  div1/Madd_GND_49_o_GND_49_o_a
LUT4:I2->O            1   0.043   0.000  div1/Msub_n0258_Madd_lut<30>
MUXCY:S->O            0   0.230   0.000  div1/Msub_n0258_Madd_cy<30> (
XORCY:CI->O           1   0.251   0.289  div1/Msub_n0258_Madd_xor<31>
LUT5:I4->O            1   0.043   0.279  Mmux_out110 (out_0_OBUF)
OBUF:I->O                 0.000          out_0_OBUF (out<0>)
----------------------------------------
Total                      54.238ns (31.895ns logic, 22.343ns route)
                                    (58.8% logic, 41.2% route)
```

Figure 8: Time summary

Figure 8 represents the time consumed such as path delays by using the Xilinx ISE software. The consumed path delay is 54.238ns.Figure 9represents the power consumed by using the Xilinx ISE software. The consumed power is 0.143uw.

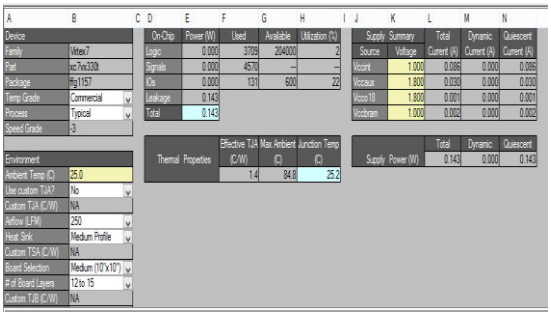

Figure 9:Power summary

Table 1: Performance comparison

Parameter	Existing Method	Proposed Method
Time delay	59.110 ns	54.238 ns
Power	1.293uw	0.143 uw
LUTs	5277	4982

5. Conclusion

This paper proposes a high-performance and low-power approximate partial product accumulation tree for a multiplier using a newly designed approximate

adder. The proposed approximate adder ignores the carry propagation by generating both an approximate sum and an error signal. OR gate and approximate adder based error reduction schemes are utilized, yielding two different approximate 8 × 8 multiplier designs: AM1 and AM2. Moreover, modifications are made on the error reduction schemes for 16 × 16 multiplier designs, such that TAM1 and TAM2 are obtained by truncating 16 LSBs of the partial products. The proposed approximate multipliers have been shown to have a lower power dissipation than an exact Wallace multiplier optimized for speed. Functional analysis has shown that on a statistical basis, the proposed multipliers have very small error distances and thus, they achieve a high accuracy.

References

[1] M. Alioto, "Ultra-low power VLSI circuit design demystified and explained: A tutorial," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 1, pp. 3–29, Jan. 2012.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[3] 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.

[4] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in Proc. Int. Conf. Comput.-Aided Design, Nov. 2011, pp. 667–673.

[5] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie, "New approximate multiplier for low power digital signal processing," in Proc. 17th Int. Symp. Comput. Archit. Digit. Syst. (CADS), Oct. 2013, pp. 25–30.

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

[7] D. R. Kelly, B. J. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in Proc. Conf. Design Archit. Signal Image Process., 2009, pp. 97–104.

[8] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in Proc. IEEE Int. Conf. Electron Devices Solid-

State Circuits (EDSSC), Dec. 2010, pp. 1–4.

[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] K. Bhardwaj and P. S. Mane, "ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip," in Proc. 8th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip, 2013, pp. 1–6.

[11] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in Proc. 15th Int. Symp. Quality Electron. Design (ISQED), 2014, pp. 263–269.

[12] J. N. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electron. Comput., vol. EC-11, no. 4, pp. 512–517, Aug. 1962.

[13] V. Mahalingam and N. Ranganathan, "Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition," IEEE Trans. Comput., vol. 55, no. 12, pp. 1523–1535, Dec. 2006.

[14] Nangate 45nm Open Cell Library, accessed on 2010. [Online]. Available: http://www.nangate.com/

[15] H. R. Myler and A. R. Weeks, The Pocket Handbook of Image Processing Algorithms in C. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.

[16] 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.

[17] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Austin, TX, USA, 2015, pp. 418–425.

[18] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in Proc. 31st Int. Conf. Comput. Design (ICCD), 2013, pp. 33–38.

[19] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in Proc. 49th Design Autom. Conf. (DAC), Jun. 2012, pp. 820–825.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, Apr. 2004.

[21] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders,"

IEEE Trans. Comput., vol. 62, no. 9, pp. 1760–1771, Sep. 2013.