

Reduction of Power and Area using Appropriate Multiplier in FPGA

Ramaraju Sowjanya¹, Neelima.G²

¹ Pursuing M.Tech (Es), Newton's Institute of Engineering College Alugurajupally, Macherla, Guntur Dist., Ap, India

² Assistant Professor, Newton's Institute of Engineering College Alugurajupally, Macherla, Guntur Dist., Ap, India

Abstract- Inexact figuring can diminish the outline many-sided quality with an expansion in execution and power proficiency for blunder strong applications. This short manages another plan approach for estimate of multipliers. The fractional results of the multiplier are adjusted to present differing likelihood terms. Rationale unpredictability of estimation is shifted for the gathering of adjusted incomplete items in light of their likelihood. The proposed guess is used in two variations of 16-bit multipliers. Amalgamation comes about uncover that two proposed multipliers accomplish control funds of 72% and 38%, individually, contrasted with a correct multiplier. They have better exactness when contrasted with existing surmised multipliers. Mean relative mistake figures are as low as 7.6% and 0.02% for the proposed rough multipliers, which are superior to the past works. Execution of the proposed multipliers is assessed with a picture handling application, where one of the proposed models accomplishes the most noteworthy pinnacle flag to commotion proportion.

I. INTRODUCTION

In applications like media flag preparing and information mining which can endure blunder, correct figuring units are not generally essential. They can be supplanted with their inexact partners. Research on rough figuring for mistake tolerant applications is on the ascent. Adders and multipliers frame the key segments in these applications. In rough full adders are proposed at transistor level and they are used in advanced flag preparing applications. Their proposed full adders are utilized as a part of gathering of halfway items in multipliers. To lessen equipment unpredictability of multipliers, truncation is broadly utilized in settled width multiplier outlines. At that Point a steady or variable amendment term is

added to adjust for the quantization mistake presented by the truncated part. Estimate strategies in multipliers center on gathering of fractional items, which is significant as far as power utilization.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM:

2.1.1 Dadda multiplier: The Dadda multiplier is an equipment multiplier configuration created by PC researcher 'Luigi Dadda' in 1965. It is like the 'Wallace multiplier', however it is marginally speedier (for all operand sizes) and requires less entryways (for everything except the littlest operand sizes). Actually, Dadda and Wallace multipliers have a similar 3 stages:

1. Multiply (coherent AND) each piece of one of the contentions, by each piece of the other, yielding n^2 comes about. Contingent upon position of the duplicated bits, the wires convey diverse weights, for instance wire of bit conveying after effect of a_2b_3 is 32 bits.
2. Reduce the quantity of fractional items to two by layers of full and half adders.
3. Group the wires in two numbers, and include them with a customary snake.

Be that as it may, dissimilar to Wallace multipliers that decrease however much as could reasonably be expected on each layer, Dadda multipliers do as couple of diminishment as would be prudent. Along these lines, Dadda multipliers have a more affordable decrease stage; however the numbers might be a couple of bits longer, accordingly requiring marginally greater adders. To accomplish this, the structure of the second step is administered by marginally more unpredictable guidelines than in the

Wallace tree. As in the Wallace tree, another layer is included if any weight is conveyed by at least three wires. The diminishment rules for the Dadda tree, be that as it may, are as per the following:

- Take any three wires with similar weights and info them into a full viper. The outcome will be a yield wire of a similar weight and a yield wire with a higher weight for every three info wires.
- If there are two wires of a similar weight left, and the present number of yield wires with that weight is equivalent to 2 (modulo 3), input them into a half viper. Something else, go them through to the following layer.
- If there is only one wire left, interface it to the following layer.

3. DESIGN ANALYSIS OF PROPOSED SYSTEM

Proposed system:

Execution of multiplier involves three stages: age of fractional items, halfway items decrease tree, lastly, a vector consolidate expansion to deliver last item from the aggregate and convey push created from the lessening tree. Second step devours more power. In this short, guess is connected in decrease tree organize. A 8-bit unsigned1 multiplier is utilized for delineation to depict the proposed technique in estimate of multipliers. Consider two 8-bit unsigned information operands $\alpha = _7m=0 \alpha m2m$ and $\beta = _7n =0 \beta n2n$. The incomplete item $\alpha m, n = \alpha m \beta n$ in Fig. 1 is the consequence of AND task between the bits of αm and βn . The proposed surmised procedure can be connected to marked duplication including Booth multipliers also, aside from it isn't connected to sign augmentation bits.

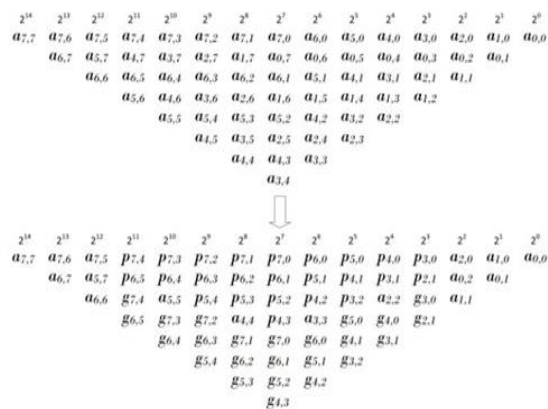


Fig.3.1: Transformation of generated partial products into altered partial products.

| m | Probability of the generate elements being | | | | P_{err} |
|---|--|--------|---------|--------------------|-----------|
| | all zero | one 1 | two 1's | three 1's and more | |
| 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 |

Table 3.1: Probability statistics of generate signals

From statistical point of view, the partial product a_m, n has a probability of 1/4 of being 1. In the segments containing in excess of three incomplete items, the fractional items a_m, n and a_n, m are joined to shape propagate and produce motions as given in (1). The subsequent propagate and create signals shape changed halfway items p_m, n and g_m, n . From segment 3 with weight 23 to segment 11 with weight 211, the halfway items a_m, n and a_n, m are supplanted by modified fractional items p_m, n and g_m, n . The original and transformed partial product matrices are shown in Fig. 1

$$P_m, n = a_m, n + a_n, m$$

$$G_m, n = a_m, n \cdot a_n, m \tag{1}$$

The probability of the altered partial product g_m, n being one is 1/16, which is significantly lower than 1/4 of a_m, n . The probability of altered partial product p_m, n being one is $1/16 + 3/16 + 3/16 = 7/16$, which is higher than g_m, n . These factors are considered, while applying approximation to the altered partial product matrix. A. Approximation of Altered Partial Products g_m, n . The accumulation of generate signals is done column wise. As each element has a probability of 1/16 of being one, two elements being 1 in the same column even decreases. For example, 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 $4pr^3(1 - pr)$ 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. Based on Table I, using Or on the other hand door in the gathering of section astute create components in the changed fractional item grid gives correct outcome in the majority of the cases. The likelihood of mistake (Perr) while utilizing OR entryway for lessening of produce motions in every section is additionally recorded in Table I. As can be

seen, the likelihood of miss prediction is low. As the quantity of create signals expands, the mistake likelihood increments directly. Be that as it may, the estimation of mistake likewise rises. To keep this, the most extreme number of produce signs to be gathered by OR door is kept at 4. For a section having m create signals, $_m/4_OR$ entryways are utilized.

B. Guess of Other Partial Products

The gathering of other fractional items with likelihood $1/4$ for am, n and $7/16$ for pm, n utilizes inexact circuits. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. Carry and Sum are two outputs of these approximate circuits. Since Carry has higher weight of binary bit, error in Carry bit will contribute more by producing error difference of two in the output. Approximation is handled 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 contribute to high area and delay. For approximating half-adder, XOR gate of Sum is replaced with OR gate as given in (2). This results in one error in the Sum computation as seen in the truth table of approximate half adder in

| 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 |

Table 3.2: Truth table of approximate half adder
In Table 3.2. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch

Sum = x1 + x2

Carry y = x1 · x2. (2)

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. Carry is modified as in (3) introducing one error. This provides more simplification, while maintaining 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 x3

Carry = W · x3 (3)

Two approximate 4-2 compressors in [5] produce nonzero output even for the cases where all inputs are zero. This results in high ED and high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the result.

The proposed 4-2 compressor overcomes this drawback. In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases. This property is taken to eliminate one of the three output bits in 4-2 compressor.

| 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 |

| 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 |

To maintain minimal error difference as one, the output “100” (the value of 4) for four inputs being one has to be replaced with outputs “11” (the value of 3). For Sum computation, one out of three XOR gates is replaced 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. This results in error in five out of 16 cases. Carry is simplified as in (4). The corresponding truth table is given in Table IV

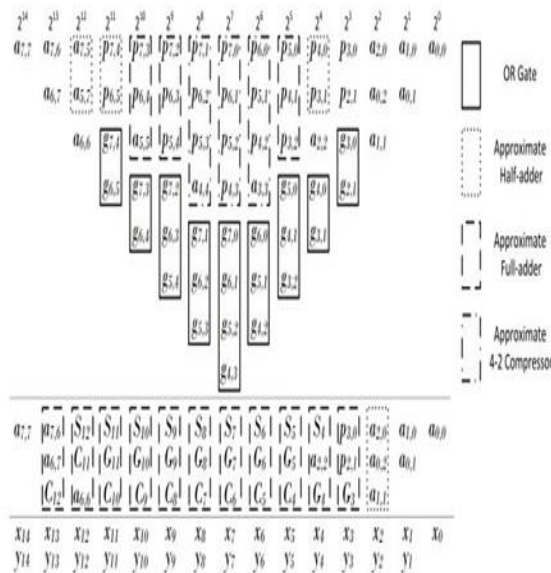
$$W1 = x1 \cdot x2$$

$$W2 = x3 \cdot x4$$

$$\text{Sum} = (x1 \cdot x2) + (x3 \cdot x4) + W1 \cdot W2$$

$$\text{Carry} = W1 + W2 \quad (4)$$

Fig.3.4 shows the reduction of altered partial product matrix of 8×8 approximate multiplier. It requires 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 resultant signals of OR gates are labeled as G_i corresponding to the column i with weight 2^i . For reducing other partial products, 3 approximate half-adders, 3 approximate full-adders, and 3 approximate compressors are required in the first stage to produce Sum and Carry y signals, S_i and C_i corresponding to column i . The elements in the second stage are reduced using 1 approximate half-adder and 11 approximate full-adders producing final two operands x_i and y_i to be fed to ripple carry adder for the final computation of the result. C. Two Variants of Multipliers Two variants of multipliers are proposed. In the first case (Multiplier1), approximation is applied in all columns of partial products

Table 3.5: Synthesis results of exact, existing, and proposed approximate multipliers

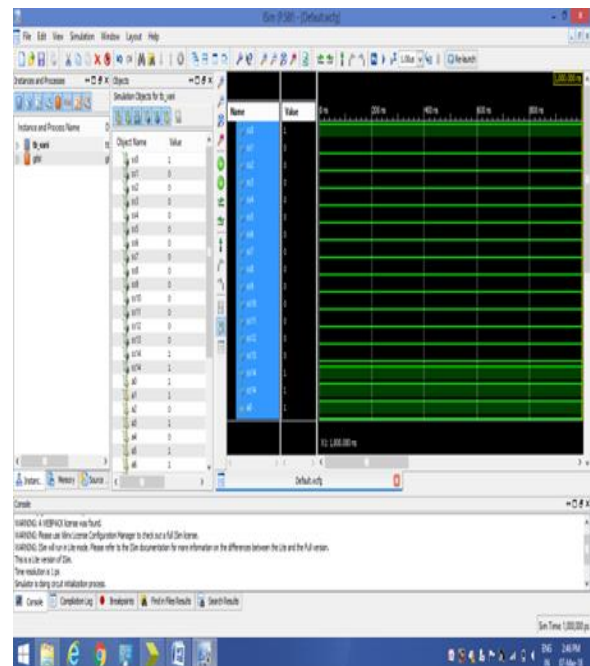
| Multiplier Type | Area (μm^2) | Delay (ns) | Power (μW) | PDP (fJ) | APP ($\mu m^2 \cdot \mu W$)(10^5) |
|-----------------|--------------------|------------|-------------------|----------|---|
| Exact | 4859.28 | 0.68 | 1776.49 | 1208.01 | 86.32 |
| Multiplier1 | 2158.56 | 0.47 | 503.15 | 236.48 | 10.86 |
| Multiplier2 | 3319.20 | 0.66 | 1102.03 | 727.34 | 36.57 |
| ACM1 [5] | 2871.72 | 0.4 | 435.31 | 174.12 | 12.50 |
| ACM2 [5] | 3782.16 | 0.63 | 1250.70 | 787.94 | 47.30 |
| SSM [6] | 3953.88 | 0.69 | 1225.29 | 845.45 | 48.44 |
| PPP [7] | 4547.52 | 0.64 | 1570.79 | 1005.31 | 71.43 |
| UDM [8] | 3938.00 | 0.67 | 1318.51 | 883.40 | 51.92 |

Table 3.5: Synthesis results of exact, existing, and proposed approximate multipliers

| Multiplier | Mean Relative Error | Normalized Error Distance |
|-------------|-----------------------|---------------------------|
| Multiplier1 | 7.63×10^{-2} | 1.78×10^{-2} |
| Multiplier2 | 2.44×10^{-4} | 7.10×10^{-6} |
| ACM1 [7] | 16.6 | 4.96×10^{-2} |
| ACM2 [7] | 2.30×10^{-3} | 6.36×10^{-6} |
| SSM [8] | 6.34×10^{-4} | 1.07×10^{-4} |
| PPP [9] | 8.98×10^{-4} | 4.58×10^{-5} |
| UDM [10] | 3.32×10^{-2} | 1.39×10^{-2} |

Table 3.6: Error metrics for 16-bit multiplier

4. SIMULATION RESULT



5. CONCLUSION

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. Here we are designing 8x8 approximate multipliers it is having less design complexity, occupies low area and power consumption will be more in case of this approximate multiplier, the speed also increases. They are also found to have better precision when compared to existing approximate multiplier designs. The proposed multiplier designs can be used in applications with minimal loss in output quality while saving significant power and area.

FUTURE SCOPE:

In this project we are implementing 8x8 approximate multiplier. 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, it can be extended up to 16-bit,32-bit,64-bit,etc.

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. Lord and E. E. Swartzlander, Jr., "Information subordinate truncation conspire for parallel multipliers," in *Proc. 31st Asilomar Conf. Signs, Circuits Syst.*, Nov. 1998, pp. 1178– 1182.
- [3] K.- J. Cho, K.- C. Lee, J.- G. Chung, and K. K. Parhi, "Outline of low-mistake settled width changed stall multiplier," *IEEE Trans. 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-propelled uncertain computational squares for productive VLSI execution of delicate registering applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850– 862, Apr. 2010.
- [5] Z. Vasicek and L. Sekanina. Evolutionary approach to approximate digital circuits design. *IEEE Tr. on Evolutionary Computation*, 19(3):432–444, 2015. [19] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. Axnn: Energy-efficient neuromorphic systems using approximate computing. In *ISLPED '15*, 2014.
- [6] S. Venkataramani, K. Roy, and A. Raghunathan. Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits. In *DATE'13*, 2013. [21] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan. Salsa: systematic logic synthesis of approximate circuits. In *DAC '12*, pages 796–801.
- [7] S.-C. Wang. *Interdisciplinary Computing in Java Programming*, chapter Artificial Neural Network, pages 81–100. Springer US, Boston, MA, 2003.
- [8] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.
- [9] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu. Approxann: An approximate computing framework.
- [10] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits.