IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AN EFFICIENT APPROXIMATE COMPRESSORS BASED MULTIPLIER

Esakkiammal.S
UG Student, Dept. of ECE
Sri Ramakrishna Institute of Technology
Coimbatore, India

Savithaa.N
Asssistant professor, Dept. of ECE
Sri Ramakrishna Institute of Technology
Coimbatore, India

Sangeetha. S
UG Student, Dept. of ECE
Sri Ramakrishna Institute of Technology
Coimbatore, India

Akshaya.S
UG Student, Dept. of ECE
Sri Ramakrishna Institute of Technology
Coimbatore, India

Abstract: This paper presents a novel approach to enhance the efficiency of a multiplier for image blending applications by employing the Dadda multiplier architecture. The proposed design incorporates an efficient approximate compressors-based module and integrates an error-correcting module, contributing to a judicious trade-off between computational efficiency and accuracy. Notably, the error-correcting module achieves substantial compensation, reducing errors from 36.30% to 28.69%. Leveraging the inherent high-speed and low-power characteristics of the Dadda multiplier, the proposed architecture optimizes hardware complexity while maintaining an acceptable level of accuracy. Through comprehensive simulations and analysis, this validates the multiplier's effectiveness in real-time image blending scenarios, highlighting its potential for applications where computational speed and precision are critical.

Keywords: Dadda multiplier; Approximate Compressors; Error Correcting Module.

INTRODUCTION

In the rapidly advancing field of digital image processing, the demand for efficient and high-performance algorithms is paramount. One critical operation in this domain is image blending, where seamless integration of multiple images is essential for achieving visually appealing and realistic results. To address the challenges associated with image blending, this research introduces an innovative approach—a novel multiplier employing approximate compressors coupled with an error-correcting module. Inmedia processing applications, multiplication is the fundamental operation that is performed inthree phases [1].

1) Partial product generation (PPG);
2) Partial product reduction (PPR) and
3) Final accumulation.

The primary goals of this research are:

1) To lower hardware use in a multiplier at the PPR stage, a new approximate 4:2compressor and constant correction term are provided.

2) An easy-to-use but effective error correcting circuit is used to lower the multiplier'serror.

3) For the 8-bit and 16-bit multipliers, we parametrized the suggested multiplier by changing the number of columns in the approximation region.

Traditional multipliers are often resource-intensive, consuming substantial computational power and memory bandwidth. In contrast, our proposed design leverages the power of approximate compressors, which exploit the inherent trade-off between accuracy and efficiency. By strategically incorporating these compressors into the multiplier architecture, we aim to achieve a fine balance between computational accuracy and processing speed, making our approach well-suited for real-time applications.

The inclusion of an error-correcting module further enhances the robustness of the multiplier, mitigating the potential drawbacks associated with approximate computations. This module intelligently identifies and rectifies errors, ensuring that the final blended images maintain the desired quality and fidelity. The synergy between the approximate compressors and the error-correcting module not only optimizes computational efficiency but also guarantees the reliability of the image blending process.

This research contributes to the growing body of work dedicated to accelerating image processing tasks by introducing a tailored solution for image blending applications. Our proposed multiplier architecture not only promises substantial gains in computational efficiency but also introduces a mechanism for error correction, elevating the overall reliability of image blending processes. As we navigate the complexities of modern image processing, this research serves as a pioneering step towards the development of more efficient and robust algorithms for various applications in the digital imaging domain.

PROPOSED METHODOLOGY

In the proposed methodology for the Efficient Approximate Compressors-Based Multiplier with Error-Correcting Module for Image Blending Application, a key component is the integration of a 4:2 approximate compressor. This specific type of compressor is strategically employed in regions of the multiplier where hardware complexity reduction, time delay reduction, and spatial efficiency are prioritized over absolute precision. The 4:2 approximate compressor performs a compression operation on four binary inputs, generating two outputs with an intentional level of inaccuracy. By leveraging approximation techniques, such as dropping least significant bits or using simplified logic functions, the 4:2 approximate compressor aims to reduce hardware complexity and enhance computational efficiency. The introduction of this approximate compressor aligns with the overarching goal of achieving an energy-efficient multiplier for image blending applications. Through careful simulation and iterative optimization, the 4:2 approximate compressor is fine-tuned to strike an optimal balance between computational accuracy and efficiency, contributing to the overall effectiveness of the proposed methodology.

In digital signal processing, the suggested approximate 4:2 compressor circuit is used to lower a signal's dynamic range. It reduces a 4-bit input toa 2-bit output by compressing it.

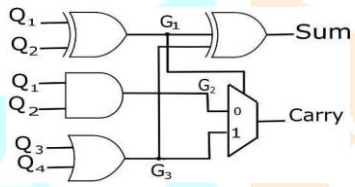


Fig 1: Proposed approximate compressor

Q1	Q2	Q3	Q4	CARRY	SUM
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Fig 2: Truth table of approximate compressor

8 X 8 DADDA MULTIPLIER

The Dadda multiplier is a hardware multiplier architecture that efficiently performs binary multiplication in digital circuits. Luigi Dadda introduced this technique in a seminal paper in 1965. The Dadda multiplier is known for its ability to generate partial products in parallel, reducing the overall multiplication time. It is commonly used in digital signal processing (DSP) applications and other areas where fast multiplication is crucial.

Here are the key steps in the operation of a Dadda multiplier:

1. Partial Product Generation: The multiplication process begins by generating partial products. Each bit of one operand is multiplied with each bit of the other operand.

2. Partial Product Reduction: The partial products are then arranged in a tree-like structure, where each level corresponds to a specific bit position in the operands. The tree structure facilitates parallel reduction of the partial products.

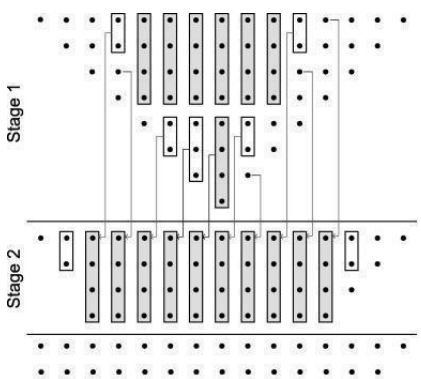
3. Carry Propagation and Summation: Carry-save adders are used at each level of the tree to accumulate partial products and propagate carries. The final result is obtained by adding up all the partial products and carries.

The key advantage of the Dadda multiplier is its parallel processing capability, which makes it suitable for applications where speed is crucial. The structure of the Dadda multiplier helps reduce the critical path delay, leading to faster multiplication. The efficiency of the Dadda multiplier comes from the use of compressors, such as 2:1 compressor, 3:2 compressors, and 4:2 compressors, which reduce the number of partial products by combining multiple bits. This compression technique helps in achieving a more compact and efficient hardware implementation. The Dadda multiplier has been widely used in the design of hardware multipliers for various applications, including digital signal processing, image processing, and communication systems. Implementing a Dadda multiplier involves careful consideration of factors such as speed, power consumption, and hardware complexity, making it a versatile choice for different scenarios.

Fig 3: Dadda multiplier

Fig 4: Dadda Multiplier in Accurate compressor

Fig 5: Dadda Multiplier Time Delay in Exact compressor



```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 1529 / 16
-------------------------------------------------------------------------------
Delay:               5.792ns (Levels of Logic = 11)
  Source:            a<2> (PAD)
  Destination:       p<14> (PAD)

  Data Path: a<2> to p<14>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             11   0.001   0.672  a_2_IBUF (a_2_IBUF)
     LUT4:I0->O             2   0.097   0.758  approx1/m1/Mmux_op111 (approx1/m1/Mmux_op11)
     LUT6:I0->O             2   0.097   0.748  approx10/Mxor_G<0>_xo<0>1 (approx10/G<0>)
     LUT5:I0->O             3   0.097   0.367  fa7/x11_xo<0>11 (fa7/x11_xo<0>1)
     LUT6:I5->O             2   0.097   0.360  fa8/out1 (y<6>)
     LUT4:I3->O             3   0.097   0.367  fa9/out1 (y<7>)
     LUT6:I5->O             3   0.097   0.367  fa11/out1 (y<9>)
     LUT5:I4->O             3   0.097   0.583  fa13/out1 (y<11>)
     LUT5:I2->O             1   0.097   0.355  fa15/x11_xo<0>11 (fa15/x11_xo<0>1)
     LUT6:I5->O             1   0.097   0.339  fa15/x11_xo<0>2 (p_14_OBUF)
     OBUF:I->O                  0.000          p_14_OBUF (p<14>)
    ----------------------------------------
    Total                       5.792ns (0.874ns logic, 4.918ns route)
                                        (15.1% logic, 84.9% route)
```

```
    ----------------------------------------
    Total                       5.172ns (0.874ns logic, 4.298ns route)
                                        (16.9% logic, 83.1% route)
```

Fig 6: Dadda Multiplier in Approximate compressor

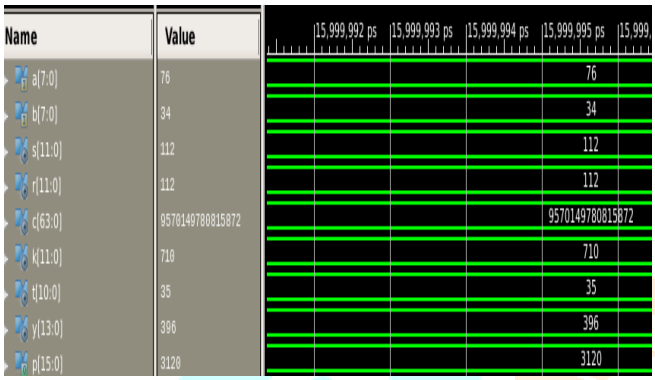


Fig 7: Dadda Multiplier Time Delay in approximate compressor

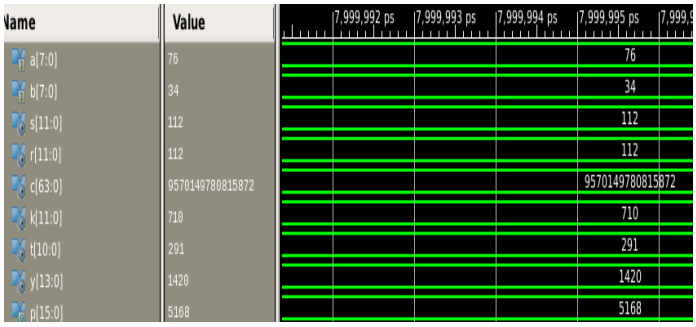


Fig 8: Dadda multiplier in approximate compressor

```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 1529 / 16
-------------------------------------------------------------------------------
Delay:               5.172ns (Levels of Logic = 11)
  Source:            a<2> (PAD)
  Destination:       p<14> (PAD)

  Data Path: a<2> to p<14>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             11   0.001   0.602  a_2_IBUF (a_2_IBUF)
     LUT4:I0->O             2   0.097   0.697  approx1/m1/Mmux_op111 (approx1/m1/Mmux_op11)
     LUT6:I0->O             2   0.097   0.688  approx10/Mxor_G<0>_xo<0>1 (approx10/G<0>)
     LUT5:I0->O             3   0.097   0.305  fa7/x11_xo<0>11 (fa7/x11_xo<0>1)
     LUT6:I5->O             2   0.097   0.299  fa8/out1 (y<6>)
     LUT4:I3->O             3   0.097   0.305  fa9/out1 (y<7>)
     LUT6:I5->O             3   0.097   0.305  fa11/out1 (y<9>)
     LUT5:I4->O             3   0.097   0.521  fa13/out1 (y<11>)
     LUT5:I2->O             1   0.097   0.295  fa15/x11_xo<0>11 (fa15/x11_xo<0>1)
     LUT6:I5->O             1   0.097   0.279  fa15/x11_xo<0>2 (p_14_OBUF)
     OBUF:I->O                  0.000          p_14_OBUF (p<14>)
    ----------------------------------------
    Total                       5.172ns (0.874ns logic, 4.298ns route)
                                        (16.9% logic, 83.1% route)
```

Fig 9: Dadda Multiplier Time Delay in approximate compressor

Fig 10: Time delay of Dadda Multiplier using Approximate Compressor- Error Compensated

Fig 11: Dadda Multiplier using Approximate Compressor- Error Compensated.

ERROR COMPENSATION TECHNIQUE

In Dadda multipliers or any multiplier architecture, error compensation refers to techniques used to mitigate or correct errors that may arise during the multiplication process, especially in approximate or low-power designs. Error compensation techniques are important in scenarios where a certain level of error is tolerable, and the goal is to achieve faster or more power-efficient multiplication.

The use of AND logic in error compensation within a Dadda multiplier context might involve introducing intentional errors and then using AND gates to selectively correct or mask those errors. However, the specific implementation details can vary based on the design goals and constraints. The error compensation technique described in the scenario is based on the use of AND logic gates as error recovery modules. Specifically, when errors are generated in the proposed compressor due to the input bits Q3 and Q4 being '1', AND logic gates are employed as error correction modules. In the architecture described, these error correction modules are strategically placed in the most significant part (MSP) of the approximate region. Two AND logic gates are used in level 1, and one AND logic gate is used in level 2. These gates generate error correction terms, which act as carry-ins to exact 4:2 compressors in the same levels, thereby improving the accuracy of the multiplier. The error correction modules are selectively deployed in the approximate portion of the multiplier, ensuring that the hardware overhead is kept small. The number of error correction modules varies based on the size of the multiplier and the number of columns approximated.

IMAGE BLENDING

Image blending application is used by converting the Verilog code. Image blending is a technique used to combine two or more images into a single composite image. It involves merging the pixel values of multiple images in a visually appealing way to create a seamless transition or integration between them. Alpha Blending also known as cross-dissolve blending, uses an alpha channel to specify the transparency or opacity of each pixel in the input images. A blending parameter, typically ranging from 0 to 1, determines the contribution of each image to the final result. This method is commonly used in graphics and compositing to create smooth transitions between images.

In alpha blending, each pixel in the blended image is calculated as a weighted sum of the corresponding pixels from the input images, where the weights are determined by the alpha channel. The alpha channel, typically represented as an additional channel alongside the color channels (e.g., red, green, blue), stores the transparency or opacity value for each pixel. A value of 0 indicates complete transparency (fully transparent), while a value of 1 indicates complete opacity (fully opaque).

The blending formula for alpha blending is often expressed as:

Blended Pixel=α×Foreground Pixel+(1−α) ×Background Pixel

Where,

α is the transparency value of the foreground pixel.

Foreground Pixel: Foreground Pixel is the pixel value of the foreground image.

Background Pixel: Background Pixel is the pixel value of the background image.

Higher alpha values make the foreground image more visible, while lower alpha values make the foreground image more transparent, allowing more of the background image to show through. By

	1	2	3	4	5	6	7	8	9
1	99	202	183	4	9	83	208	81	
2	124	130	253	199	131	209	239	163	
3	119	201	147	35	44	0	170	64	
4	123	125	38	195	26	237	251	143	
5	242	250	54	24	27	32	67	146	
6	107	89	63	150	110	252	77	157	
7	111	71	247	5	90	177	51	56	
8	197	240	204	154	160	91	133	245	
9	48	173	52	7	82	106	69	188	
10	1	212	165	18	59	203	249	182	
11	103	162	229	128	214	190	2	218	
12	43	175	241	226	179	57	127	33	
13	254	156	113	235	41	74	80	16	
14	215	164	216	39	227	76	60	255	
15	171	114	49	178	47	88	159	243	
16	118	192	21	117	132	207	168	210	

adjusting the alpha value for each pixel, you can control the degree to which the foreground image is overlaid on the background image.

Taking into account two grayscale input pictures, each having a goal of 256 x 256. Subsequent to mixing the two pictures, the comparing result will likewise be in grayscale, bringing about an estimated and precise picture. Following the mixing system, decide the Mean Squared Error (MSE) and Peak Signal to-Noise Ratio (PSNR) for the estimated and precise pictures; the inexact picture ought to have a Mean Squared Error of under 30 dB, and the precise picture ought to have a PSNR of in excess of 30 dB. In view of these calculations, it very well may be presumed that the picture is being upgraded and that its quality is being assessed

Fig 12: Blended Images using exact, approximate compressor implemented in Dadda multiplier

Fig 13: 8 x 8-bit multiplication of the blended images.

CONCLUSION

When compared to regular full-precision multipliers, compressor-based approximation multipliers may dramatically minimize delay. When utilized on integrated circuits, compressor-based designs, which are designed to have a minimal delay in general, may speed up the process,allowing for additional characteristics or components on the same chips. The compressor-basedapproximation multiplier reduces the time delay by 36%, resulting in quicker multiplication. Compressor-based systems have the capacity to process data at higher throughputs, making them suited for applications that operate in real time or systems that demand quick data processing. This project may be improved further by including appropriate error correction modules to guarantee that the final outcomes are within a suitable range of accuracy for the particular application. Error compensation technique is introduced to minimize the time delay of the approximate compressor. AND logic is used as an error compensating component in order to minimize the delay. After implementing the error compensation technique to Dadda multiplier, the error distance is reduced from 36.30% to 28.69%. image blending application is used in order to enhance the quality and accuracy of the image by calculating PSNR (Peak Signal to Noise Ratio) and MSE (Mean Square Error). These two parameters are calculated for accurate and approximate compressor implemented in Dadda multiplier.

ACKNOWLEGDEMENT

We express our gratitude to Sri Ramakrishna Institute of Technology for providing us with the necessary resources and guidance to successfully complete our project.

Ms. N. Savithaa, received her Master's Degree in VLSI Design from Sri Ramakrishna Engineering College, Coimbatore and B.E Degree in Electronics and Communication Engineering from Sri Shakthi Institute of Engineering and Technology under Anna University, Chennai. Now she is Pursuing Ph.D. under Anna University in the area of Low Power VLSI design. She has 5 years of teaching experience. Currently she is working as an Assistant Professor in the Department of Electronics and Communication

Engineering at Sri Ramakrishna Institute of Technology, Coimbatore.

Ms Akshaya S is currently pursuing her Final year in Bachelor of Degree in Sri Ramakrishna Institute of Technology, Coimbatore.

Ms Esakkiammal S is currently pursuing her Final year in Bachelor of Degree in Sri Ramakrishna Institute of Technology, Coimbatore.

Ms Sangeetha S is currently pursuing her Final year in Bachelor of Degree in Sri Ramakrishna Institute of Technology, Coimbatore.

REFERENCES

[1] U. Anil Kumar, Sumit K. Chatterjee, and Syed Ershad Ahmed, "Low-Power Compressor-Based Approximate Multipliers with Error Correcting Module," IEEE EMBEDDED SYSTEMS LETTERS, VOL. 14, NO. 2, JUNE 2022

[2] H. Pei, X. Yi, H. Zhou, and Y. He, "Design of ultra-low power consumption approximate 4–2 compressors based on the compensation characteristic," IEEE Trans.Circuits Syst. II, Exp. Briefs, vol. 68, no. 1, pp. 461–465, Jan. 2021.

[3] T.-D. Ene and J. E. Stine, "A comprehensive exploration of the parallel prefix adder tree space," in Proc. IEEE 39th Int. Conf. Comput. Design (ICCD), Oct. 2021, pp. 125–129.

[4] R. Roy et al., "PrefixRL: Optimization of parallel prefix circuits using deep reinforcement learning," in Proc. 58th ACM/IEEE Design Automat. Conf. (DAC), Dec.2021, pp. 853–858

[5] H. Waris, C. Wang, W. Liu, J. Han, and F. Lombardi, "Hybrid partial product- based high-performance approximate recursive multipliers," IEEE Trans. Emerg. Topics Comput., early access, Aug. 4, 2020.

[6] Antonio Giuseppe Maria Strollo, Senior Member, IEEE, Ettore Napoli, Senior Member, IEEE, Davide De Caro, Senior Member, IEEE, Nicola Petra, Member, IEEE, and Gennaro Di Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers," IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–I: REGULAR PAPERS, VOL. 67, NO. 9, SEPTEMBER 2020

[7] X. Yi, H. Pei, Z. Zhang, H. Zhou, and Y. He, "Design of an energy efficient approximate compressor for error-resilient multiplications," in Proc. IEEE ISCAS, Sapporo, Japan, 2019, pp. 1–5.

[8] L. Soares, M. da Rosa, C. Machado, E. da Costa, and S. Bampi, "Design methodologyto explore hybrid approximate adders for energy-efficient image and video processingaccelerators," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 6, pp. 2137–2150,Jun. 2019.

[9] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high- speedadders: A Pareto driven machine learning approach," IEEE Trans. Comput. -Aided Design Integr. Circuits Syst., vol. 38, no. 12, pp. 2298–2311, Dec. 2019.

[10] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," IEEE Embedded Syst. Lett., vol. 10, no. 1, pp. 6–9, Mar. 2018.

[11] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "Low-power approximate multipliers using encoded partial products and approximate compressors," IEEE J. Emerg. Sel. Topics Circuits Syst., vol. 8, no. 3 pp. 404–416, Sep. 2018.

[12] Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," IEEE Trans. Very Large-Scale Integration. (VLSI) Syst., vol. 25, no. 5, pp. 1782–1786, May 2017.

[13] V. Pudi, K. Sridharan, and F. Lombardi, "Majority logic formulations for parallel adderdesigns at reduced delay and circuit complexity," IEEE Trans. Comput., vol. 66, no. 10, pp. 1824–1830, Oct. 2017.

[14] Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximatecompressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[15] Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error resilient multiplier design," in Proc. IEEE DFTS, Amherst, MA, USA, 2015, pp. 183–186.

[16] Harish Rao. B, Ramesh Kumar. V, "Implementation of 8 x 8 Dadda multiplier using approximate compression for image enhancement," (IJAER), Vol. No. 10, Issue No. I,July 2015