IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 8, April 2023

VLSI Design of Approximate Baugh-Wooley Multiplier for Image Edge Computing

Gayathri R[1], Ajith Kumar A[2], Balaji P[3], Magesh P[4], Sridhar V[5]

Assistant Professor, Department of ECE[1]

Students, Department of ECE[2,3,4,5]

Dhanalakshmi Srinivasan Engineering College (Autonomous), Perambalur, India

Abstract*:* *A sort of digital circuit used to multiply two binary values is called a Baugh-Wooley multiplier. It is renowned for being more effective than other kinds of multipliers in terms of speed and gate count. You would normally start by creating a simple version of the circuit utilizing logic gates like AND gates, XOR gates, and half adders to simulate the Baugh-Wooley multiplier. The design would then be optimized utilizing methods like parallel processing, pipelining, and optimization algorithms to lower the overall gate count and boost the functionality of the circuit. Because of this, estimating a Baugh-Wooley multiplier would require a thorough knowledge of these elements as well as the specifications of the application. Verilog HDL is used to implement this design, and Modelsim 6.4 c is used to simulate it. The synthesis process tool from Xilinx measures performance.*

Keywords: Approximate computing, Edge detection, multiplying circuits

I. INTRODUCTION

Approximate circuits are a class of digital circuits that trade off accuracy for improved efficiency and reduced hardware requirements. They are particularly useful in applications where small errors can be tolerated, such as digital signal processing, multimedia, and machine learning.

One way to implement approximate circuits is by using approximate multipliers. Multiplication is a key operation in many computational tasks, including digital signal processing and machine learning. In conventional digital circuits, multiplication is implemented using high-precision arithmetic, which requires a significant amount of hardware resources. In contrast, approximate multipliers use simplified arithmetic operations to reduce the hardware requirements. An approximate multiplier typically consists of three stages: partial product generation, accumulation, and final addition. In the partial product generation stage, the input operands are decomposed into smaller sub-multiplications, which are then combined to form the full product. This stage is typically the most hardware-intensive, as it requires many multipliers to compute the partial products.

In the accumulation stage, the partial products are added together to form an intermediate result. This stage is less hardware-intensive than the partial product generation stage, as it only requires a relatively small number of adders.

Finally, in the final addition stage, any remaining error is corrected by adding a correction term to the intermediate result. This correction term is typically small compared to the intermediate result, so the hardware requirements for this stage are also relatively small. Overall, approximate circuits can significantly reduce the hardware requirements for certain types of computations, while still providing results that are accurate enough for many applications. By combining hardware and software approximations, it is possible to design complete RISC architectures that are optimized for specific types of applications, such as Convolution Neural Networks.

II. EXISTINGSYSTEM

Compressor designs are a popular approach to implementing practical multipliers with simple and regular structures. In this approach, the multiplication operation is broken down into smaller sub-multiplications that can be processed in parallel.

To illustrate the basic idea behind compressor designs, let's consider a simple example where we want to multiply a 2-bit multiplicand A by a 2-bit multiplier X. We can represent the multiplicand and multiplier as follows:

A = a1 a0, X = x1 x0

To perform the multiplication, we generate four partial products (PPs) by performing bitwise AND operations between the multiplicand and the multiplier:
PP0 = a0 * x0, PP1 = a0 * x1,
PP2 = a1 * x0, PP3 = a1 * x1
These partial products can then be combined to produce the result. However, the number of PPs generated in this way can quickly become unwieldy for larger inputs, so we need a way to reduce the number of PPs without sacrificing accuracy.
One way to achieve this is by using a compressor design. Compressors are circuits that take in multiple binary inputs and produce a smaller number of binary outputs based on a specific logic function. In the case of multipliers, compressors are used to reduce the number of PPs generated by combining multiple PPs into a smaller number of outputs.
For example, a 4-2 compressor is a circuit that takes in four binary inputs (denoted as ai, bi, ci, and di) and produces two binary outputs (denoted as yi and yi+1). The exact p-q compressor, on the other hand, takes in p binary inputs and produces q binary outputs, where q is smaller than p.
In a practical multi-stage multiplier, the 4-2 compressor is widely used to reduce the number of PPs generated in each stage. By grouping four PPs along the same i-th bit position, the compressor generates two PPs over two columns (yi and yi+1), which can then be used as inputs to the next stage of the multiplier.
Overall, compressor designs are a powerful tool for reducing the hardware requirements of multipliers by minimizing the number of PPs generated and processed. By using simple and regular structures, compressors can also be easily implemented in hardware, making them an attractive option for practical designs.
The Braun multiplier and Baugh-Wooley multiplier are two popular methods for implementing fast and efficient multipliers. Both methods use compressors to reduce the number of partial products generated, which can significantly reduce the hardware requirements for the multiplier.
The Braun multiplier is a type of multi-stage multiplier that uses a combination of exact and approximate compressors to reduce the number of partial products generated. The exact compressors are used to generate a small number of high-order bits, while the approximate compressors are used to generate a larger number of low-order bits. By combining these compressors, the Braun multiplier can achieve high-speed operation while minimizing hardware requirements.
The Baugh-Wooley multiplier, on the other hand, is a type of signed multiplier that uses a combination of exact and approximate compressors to reduce the number of partial products generated. The exact compressors are used to generate the most significant bits of the result, while the approximate compressors are used to generate the least significant bits. This approach can reduce the hardware requirements of the multiplier while still maintaining high accuracy.
Both the Braun and Baugh-Wooley multipliers can benefit from the use of 4-2 compressors to reduce the number of partial products generated. By using a 4-2 compressor to group four partial products along the same bit position, the multiplier can generate two outputs with reduced hardware requirements.
Overall, the use of compressors is a powerful technique for reducing the hardware requirements of multipliers and can be used with both the Braun and Baugh-Wooley multiplier designs. By using compressors to reduce the number of partial products generated, these designs can achieve high-speed operation while minimizing hardware requirements.

III. PROPOSED SYSTEM

The proposed method minimizes the average square of the absolute error of an approximate multiplier according to the probability distributions of operands extracted from the target application with consideration of input polarity, achieving low hardware cost and negligible application-level performance loss. The proposed method can generate unsigned multipliers (or signed multipliers) based on the Braun multiplier (or Baugh–Wooley multiplier). To demonstrate the effectiveness of the method, three different-scale quantized Edge detection including Matrix Multiplication, and Threshold Detection with 8 × 8 unsigned multiplications.
The proposed method is a technique for designing approximate multipliers that minimize the average square of the absolute error based on the probability distributions of operands from a target application. This approach takes into

account the input polarity of the operands and achieves low hardware cost with negligible performance loss at the application level.

The method can generate both unsigned and signed multipliers based on the Braun multiplier or the Baugh-Wooley multiplier. The Braun multiplier is a multi-stage multiplier that combines exact and approximate compressors to reduce the number of partial products generated, while the Baugh-Wooley multiplier is a signed multiplier that uses a combination of exact and approximate compressors to reduce the hardware requirements.

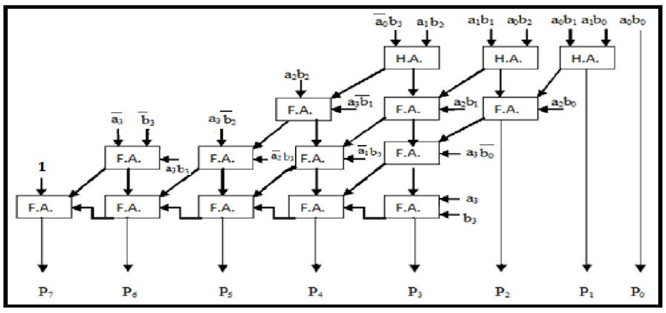


Fig. 1.Baugh-Wooley multiplier.

To demonstrate the effectiveness of the proposed method, the authors applied it to three different-scale quantized edge detection tasks, including matrix multiplication and threshold detection, using an 8x8 unsigned multiplication. The results showed that the proposed method can achieve significant improvements in hardware efficiency while maintaining high accuracy compared to traditional multiplier designs.

Overall, the proposed method is a promising technique for designing approximate multipliers that can reduce hardware requirements and improve performance without sacrificing accuracy. By considering the probability distributions of operands and input polarity, the method can generate efficient and accurate multipliers that are well-suited for a wide range of applications.

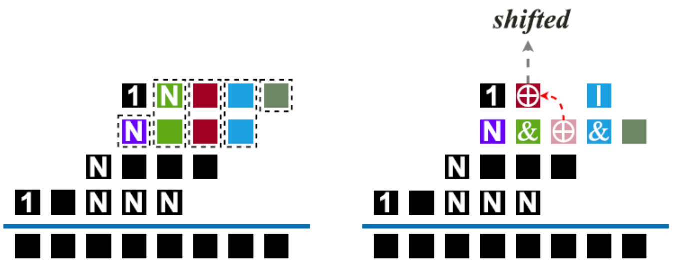


Fig. 2.Proposed System Block Diagram.

A. Image Edge Detection.

Edge Detection there are two masks, one mask identifies the horizontal edges, and the other mask identifies the vertical edges. The mask which finds the horizontal edges is equivalent to having the gradient in a vertical direction and the mask which computes the vertical edges is equivalent to taking in the gradient in the horizontal direction. In fig 3.

-1	-2	-1
0	0	0
1	2	1

1	0	-1
2	0	-2
1	0	-1

Fig. 3. horizontal edges and vertical edges.

IV. MAIN BLOCK DIAGRAM

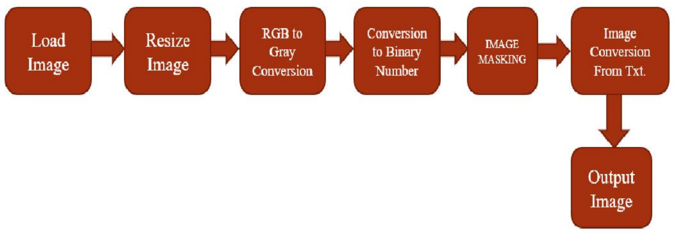


Fig. 4. Block Diagram of Process.

A. Process flow of RGB to binary image conversion (MATLAB Part)

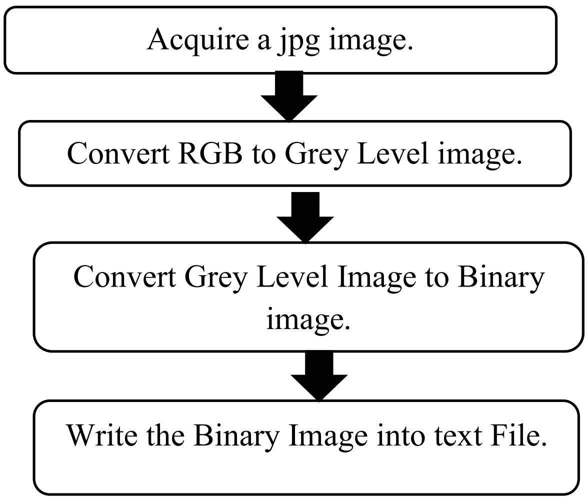

Acquire a jpg image.
Convert RGB to Grey Level image.
Convert Grey Level Image to Binary image.
Write the Binary Image into text File.

Fig. 5.Process flow of RGB to binary image conversion

B. VLSI PART (USING Model sim)

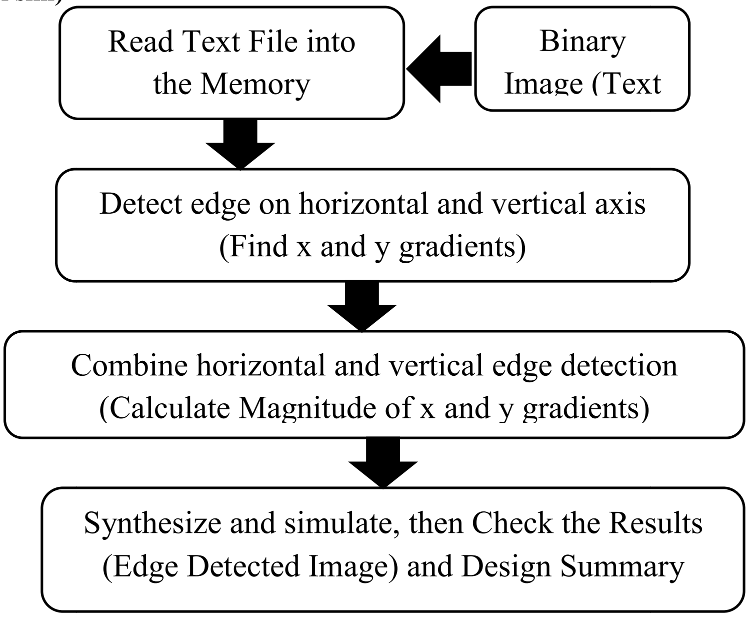

Read Text File into the Memory
Binary Image (Text
Detect edge on horizontal and vertical axis (Find x and y gradients)
Combine horizontal and vertical edge detection (Calculate Magnitude of x and y gradients)
Synthesize and simulate, then Check the Results (Edge Detected Image) and Design Summary

Fig. 6. VLSI PART (USING Modelsim)

IJARSCT

ISSN (Online) 2581-9429

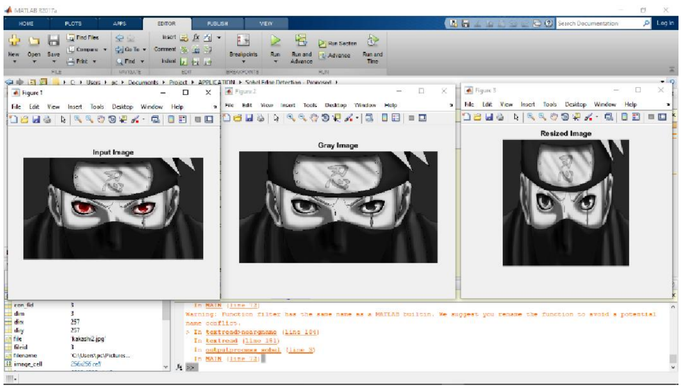
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 8, April 2023

V. FINAL OUTPUT



```
con_fid      3
dim          3
dimx         257
dimy         257
file         'kakashi2.jpg'
fileid       3
filename     'C:\Users\pc\Pictures...
image_cell   256x256 cell
```

```
  In MAIN (line 72)
Warning: Function filter has the same name as a MATLAB builtin. We suggest you rename the function to avoid a potential
name conflict.
> In textread>noargname (line 184)
  In textread (line 161)
  In outputprocess_sobel (line 3)
  In MAIN (line 72)
>>
```

Fig. 7. Input image.

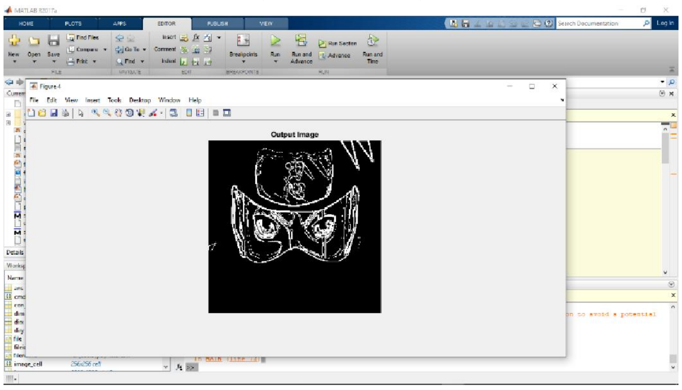


Fig. 8. Edge Detected image.

VI. COMPARISON TABLE

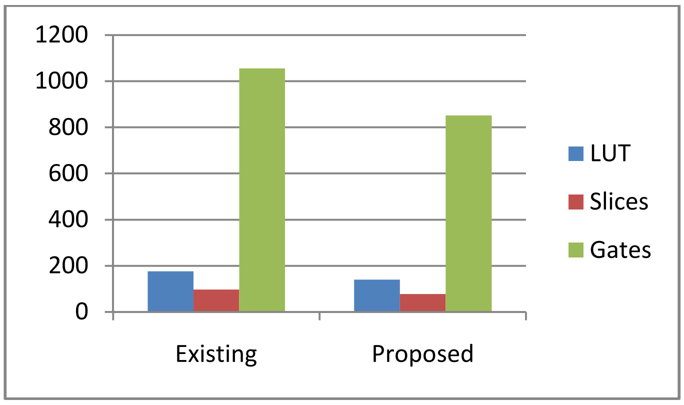


Fig 9. Comparison Table

VII. CONCLUSION

In this brief, Truncated Approximate Carry-based Baugh-Wooley Multiplier (TACBM) is presented. Multiplier is achieving an error compensation circuit designed by selective modification of the k-map to achieve the twin goal of energy and error minimization. Extensive error analysis is performed by applying different parts of the compensation circuit to the non-truncated part. This is done by Simulation by Modelsim and synthesis Done by Xilinx Tool. This Proposed Multiplier is used in Image Edge Detection Processing Applications using MATLAB. We are using this multiplier for image processing applications like edge detection schemes. This Edge Detection uses Sobel Operator in Digital Image Processing and implementation using Verilog HDL.

REFERENCES

[1]. S. Zheng, Z. Li, Y. Lu, J. Gao, J. Zhang, and L. Wang, "HEAM: High-efficiency approximate multiplier optimization for deep neural networks," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2022, pp. 3359–3363.

[2]. L. Eeckhout, "Is Moore's law slowing down? What's next?" IEEE Micro, vol. 37, no. 4, pp. 4–5, Jul. 2017.

[3]. W. Xu, S. S. Sapatnekar, and J. Hu, "A simple yet efficient accuracyconfigurable adder design," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 6, pp. 1112–1125, Jun. 2018.

[4]. T. Kong and S. Li, "Design and analysis of approximate 4–2 compressors for high-accuracy multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 29, no. 10, pp. 1771–1781, Oct. 2021.

[5]. J. Melchert, S. Behroozi, J. Li, and Y. Kim, "SAADI-EC: A qualityconfigurable approximate divider for energy efficiency," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 11, pp. 2680–2692, Nov. 2019.

[6]. K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low error fixed-width modified Baugh-Wooley multiplier," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[7]. H.-A. Huang, Y.-C. Liao, and H.-C. Chang, "A Self-Compensation Fixed-Width Baugh-Wooley Multiplier and Its 128-point FFT Applications," in Proc. IEEE Int. Symp. Circuits Syst., May 2006, pp. 3538–3541.

[8]. J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified Baugh-Wooley multipliers for lossy applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 1, pp. 52-60, Jan. 2011.

[9]. C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic estimation bias circuit for fixed-width Baugh-Wooley multiplier and its DCT applications," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 4, pp. 215-219, Apr. 2011.

[10]. Y.-H. Chen, "An accuracy-adjustment fixed-width Baugh-Wooley multiplier based on multilevel conditional probability," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 1, pp. 203-207, Jan. 2015.

[11]. W.-Q. He, Y.-H. Chen, and S.-J. Jou, "High-accuracy fixed-width Baugh-Wooley multipliers based on probability and simulation," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 8, pp. 2052-2061, Aug. 2015.

[12]. C. Meng, W. Qian, and A. Mishchenko, "ALSRAC: Approximate logic synthesis by resubstitution with approximate care set," in Proc. 57th ACM/IEEE Design Autom. Conf. (DAC), Jul. 2020, pp. 1–6.

[13]. Y.-J. Chang, Y.-C. Cheng, S.-C. Liao, and C.-H. Hsiao, "A low power Radix-4 booth multiplier with pre-encoded mechanism," IEEE Access, vol. 8, pp. 114842–114853, 2020.

[14]. V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 3, pp. 421–430, Mar. 2018.

[15]. 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.

[16]. A. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), 2014, pp. 1–4.

[17]. S. E. Ahmed, S. Kadam, and M. B. Srinivas, "An iterative logarithmic multiplier with improved precision," in Proc. IEEE 23rd Symp. Comput. Arithmetic (ARITH), Jul. 2016, pp. 104–111.

[18]. C. Chen, S. Yang, W. Qian, M. Imani, X. Yin, and C. Zhuo, "Optimally approximated and unbiased floating-point multiplier with runtime reconfigurability," in Proc. 39th Int. Conf. Comput. -Aided Design, Nov. 2020, pp. 1–9.

[19]. S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," ACM J. Emerg. Technol. Comput. Syst., vol. 14, no. 2, pp. 1–23, Apr. 2018.

[20]. Z. Vasicek, V. Mrazek, and L. S. Brno, "Towards low power approximate DCT architecture for HEVC standard," in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), Mar. 2017, pp. 1576–1581.