DESIGN & IMPLEMENTATION OF COMPRESSOR BASED APPROXIMATE MULTIPLIER USING VLSI ARCHITECTURE

Mrs.M.Rojaramani, (rojaramani@gmail.com),Assistant Professor,AVS Engineering College,Salem, Tamilnadu.
Mr. G.Kanagaraj, (raj.techstorm@gmail.com), Assistant Professor,AVS Engineering College,Salem, Tamilnadu.
M.Masilamani,(masilasaravanan2017@gmail.com)PG Scholar,AVS Engineering College,Salem, Tamilnadu.

ABSTRACT

Approximate computing has received significant attention as a promising strategy to decrease power consumption of inherently error tolerant applications. In Existing system, they focus on hardware-level approximation by introducing the partial product perforation technique for designing approximate multiplication circuits. They prove in a mathematically rigorous manner that in partial product perforation, the imposed errors are bounded and predictable, depending only on the input distribution. Through extensive experimental evaluation, apply the partial product perforation method on different multiplier architectures and expose the optimal architecture perforation configuration pairs for different error constraints. To reduce this minimum error constraint and improve approximate difference, a novel compressor based approximate multiplier is proposed.Approximate compressor is proposed to further increase performance as well as reducing the error rate. This architecture is going to be implement in Xilinx ISE EDA to analyse area, power ,delay of the novel compressor design.

I.Introduction

Multipliers are among the fundamental components of many digital systems and, hence, their power dissipation and speed are of prime concern. For portable applications where the power consumption is the most important parameter, one should reduce the power dissipation as much as possible. One of the best ways to reduce the dynamic power dissipation, henceforth referred to as power dissipation in this paper, is to minimize the total switching activity, i.e., the total number of signal transitions of the system.

Many research efforts have been devoted to reducing the power dissipation of different multipliers. The largest contribution to the total power consumption in a multiplier is due to generation of partial product. Among multipliers, tree multipliers are used in high speed applications such as filters, but these require large area. The carry-select-adder (CSA)-based radix multipliers, which have lower area overhead, employ a greater number of active transistors for the multiplication operation and hence consume more power. Among other multipliers, shift-and-add multipliers have been used in many other applications for their simplicity and relatively small area requirement. Higher-radix multipliers are faster but consume more power since they employ wider registers, and require more silicon area due to their more complex logic. The multiplier shall then calculate the result using the shift and add method and provide the 16-bit result along with a Stop signal. As an application, this technique has been applied to different architectures of low power high order compressors such as 4-2 and 5-2 compressors unit, which are implements using static CMOS gates. The resulting modulo 2n + 1 multiplier and squarer have been implemented in standard CMOS cell technology and compared both qualitatively and quantitatively with the existing hardware implementations. The unit gate model analysis and the experimental results show that the proposed implementation is faster and consume less power than existing hardware implementations making it a viable option for efficient designs.

In the recent years, the number of internet and wireless communication nodes has grown rapidly, which involves the transmission of data over channels. The confidentiality and security requirements are becoming more and more important to protect the data transmitted and received. similarly, in the networked instrumentation and distributed measurement systems, secured communication is given the utmost priority. Various cryptographic systems have been studied and implemented to ensure the security of these systems. International Data Encryption Algorithm (IDEA) is one of the most reliable cryptographic algorithms used for transmission of the data.

The ability to perform fast encoding and decoding operations is then still a major issue for the implementation of IDEA, particularly from a hardware point of view. Numerous Residue Number System module architectures and different hardware implementations have been proposed Modulo 2n + 1 multiplier has been given much focus and it has found many applications in residue arithmetic, digital signal processing and cryptography. For example, three major operations that decide the overall delay and performance of IDEA cipher are modulo 2n addition, bitwise-XOR and modulo 2n+1 multiplication. As the first two operations take less time and are easy to implement, improving the delay and power efficiency of the modulo 2n + 1 multiplication operation leads to significant increase in the performance of the entire IDEA cipher. More recently, proposed an efficient algorithm for computing modulo 2n +1 multiplication, in which the partial product reduction block, which contributes most to the overall delay, is designed as a complex network of full-adders and carry chains.

II.Literature Review

Inexact (or approximate) computing is an attractive paradigm for digital processing at nanometric scales. Inexact computing is particularly interesting for computer arithmetic designs. This paper deals with the analysis and design of two new approximate 4-2 compressors for utilization in a multiplier. These designs rely on different features of compression, such that imprecision in computation (as

measured by the error rate and the so-called normalized error distance) can meet with respect to circuit-based figures of merit of a design (number of transistors, delay and power consumption). Four different schemes for utilizing the proposed approximate compressors are proposed and analyzed for a Dadda multiplier. Extensive simulation results are provided and an application of the approximate multipliers to image processing is presented. The results show that the proposed designs accomplish significant reductions in power dissipation, delay and transistor count compared to an exact design; moreover, two of the proposed multiplier designs provide excellent capabilities for image multiplication with respect to average normalized error distance and peak signal-to-noise ratio (more than 50dB for the considered image examples).

III.Existing System

Modern embedded electronic devices, power consumption are a first-class design concern. Considering that a large number of application domains are inherently tolerant to imprecise calculations, e.g., digital signal processing (DSP), data analytics, and data-mining approximate computing appear as a promising solution to reduce their power dissipation. Such applications process large redundant data sets or noisy input data derived from the real world, do not have a golden result, perform statistical/probabilistic computations, and/or demand human interaction, thus their exactness is relaxed due to limited human perception. Approximate computing can be applied at both software and hardware levels. Hardware-level approximation mainly targets arithmetic units, such as adders and multipliers, widely used in portable devices to implement multimedia algorithms, e.g., image and video processing. The most commonly used techniques for the generation of approximate arithmetic circuits are. Hardware circuits, contrary to software approximations, offer transistors reduction, lower dynamic and leakage power, lower circuit delay, and opportunity for downsizing. Motivated by the limited research on approximate
multipliers, compared with the extensive research on approximate adders, and explicitly the lack of approximate techniques targeting the partial product generation, we introduce the partial product perforation method for creating approximate multipliers. Inspired from they omit the generation of some partial products, thus reducing the number of partial products that have to be accumulated, they decrease the area, power, and depth of the accumulation tree.

More specifically, they apply the partial product perforation on 16 different multiplier architectures using industrial strength tools, i.e., Synopsys Design Compiler and Prime Time. Through extensive experimental evaluation, we present the optimal approximate multiplier configurations for various error constraints. They show that, compared with the accurate multiplier, the product perforation offers reductions of up to 50% in power consumption, 45% in area, and 35% in critical delay for 0.1% normalized mean error distance (NMED) . Moreover, it is compared with the state-of-the-art approximate computing works that use either VOS logic approximation or truncation outperforming them significantly in terms of power dissipation and error. Finally, we examine the scalability of our technique by applying it on different bit-width multipliers and show that the delivered savings increase with the width increase.

However, as stated in mneeds to be at least $n/2$ to attain acceptable accuracy, thus limiting the energy savings and the scalability of this approach. Hashemi *et al.* extended the idea of leading-one segments to enable dynamic range multiplication and added a correction term. Although delivers higher accuracy designs than using smaller values for m, its approach requires the allocation of extra complex circuitry, i.e., two leading-one detectors, two multiplexers for segment selection, one $\log(n)$-bit comparator, a $\log(n)$-bit adder, and one $2n$-bit barrel shifter. These extra components are expected to highly increase the circuit's complexity, introducing nontrivial delay, area, and energy overheads that may considerably decrease the benefits. This is expected to be more evident in designs targeting too small error values, in which need for larger m values is required.

IV.Proposed System

Design of an approximate compressor is proposed to further increase performance as Well as reducing the error rate. Multiplication is a fundamental operation in most signal processing algorithms. Multipliers have large area, long latency and consume considerable power. Therefore low-power multiplier design has an important part in low-power VLSI system design. A system is generally determined by the performance of the multiplier because the multiplier is generally the element and more area consuming in the system. Hence optimizing the speed and area of the multiplier is one of the major design issues. However , area and speed are usually conflicting constraints so that improvements in speed results in larger areas. Multiplication is a mathematical operation that include process of adding an integer to itself a specified number of times. A number (multiplicand) is added itself a number of times as specified by another number (multiplier) to form a result (product).play an important role in today's digital processing and various other applications. In this section, the impact of using the proposed compressors for multiplication is investigated. A fast (exact) multiplier is usually composed of three parts (or modules).

- Partial product generation.
- A carry save adder (CSA) tree to reduce the partial products' matrix to an addition of only two operands.
- A carry propagation adder (CPA) for the final computation of the binary result.

In the design of a multiplier, the second module plays a pivotal role in terms of delay, power consumption and circuit complexity. Compressors have been widely used to speed up the CSA tree and decrease its power dissipation, so to achieve fast and low-power operation. The use of approximate compressors in the CSA tree of a multiplier results in an approximate multiplier. 8x8 unsigned DADDA tree multiplier is considered to assess the impact of using the proposed compressors in approximate multipliers. The proposed multiplier uses in the first part AND gates to generate all partial products. In the second part, the

approximate compressors proposed in the previous section are utilized in the CSA tree to reduce the partial products. The last part is an exact CPA to compute the final result. It shows the reduction circuitry of an exact multiplier for n ¼ 8. The reduction part uses half-adders, full-adders and 4-2 compressors; each partial product bit is represented by a dot. In the first stage, two half-adders, two full adders and eight compressors are utilized to reduce the partial products into at most four rows. In the second or final stage, 1 half-adder, 1 full-adder and 10 compressors are used to compute the two final rows of partial products. Therefore, two stages of reduction and three half adders, three full-adders and 18 compressors are needed in the reduction circuitry of an 8x8 DADDA multiplier. In this paper, four cases are considered for designing an approximate multiplier.

- In the first case (Multiplier 1), Design 1 is used for all 4-2 compressors.
- In the second case (Multiplier 2), Design 2 is used for the 4-2 compressors. Since Design 2 does not have Cin and Cout, the reduction circuitry of this multiplier requires a lower number of compressors. Multiplier 2 uses six half-adders, one full-adder and 17 compressors.
- In the third case (Multiplier 3), Design 1 is used for the compressors in the n-1 least significant columns. The other n most significant columns in the reduction circuitry use exact 4-2 compressors.
- In the fourth case (Multiplier 4), Design 2 and exact 4-2 compressors are used in the n-1 least significant columns and the n most significant columns in the reduction circuitry respectively.

The objectives of the first two approximate designs are to reduce the delay and power consumption compared with an exact multiplier; However, a high error distance is expected. The next two approximate multipliers (i.e., Multipliers 3 and 4) are proposed to decrease the error distance. The delay in these designs is determined by the exact compressors that are in the critical path; therefore, there is no improvement in delay for these approximate designs compared with an exact multiplier. However, it is expected that the utilization of approximate compressors in the least significant columns will decrease the power consumption and transistor count (as measure of circuit complexity).

While the first two proposed multipliers have better performance in terms of delay and power consumption, the error distances in the third and fourth designs are expected to be significantly lower.

The DADDA multiplier was designed by the scientist Luigi Dadda in 1965. It looks similar to Wallace multiplier but slightly faster and requires less gates. DADDA Multiplier was defined in three steps:

- Multiply each bit of one argument with the each and every bit of other argument and continue until all arguments are multiplied.
- Reduce the number of partial products to two layers of full and half adders.
- Group the wires in two numbers, and add them with a conventional adder.

A 8x8 multiplier using dada multiplier design is designed. Instead of using conventional full adders and half adder for designing the multiplier, compressors which reduce the complexity of the multiplier is introduced.

Design of an approximate compressor is proposed to further increase performance as Well as reducing the error rate. Since the carry and Cout outputs have the same Weight, the proposed equations for the approximate carry and Cout in the previous part can be interchanged. In this new design, carry uses the right hand side and Cout is always equal to Cin; since Cin is zero in the first stage, Cout and Cin will be zero in all stages. So, Cin and Cout can be ignored in the design. Fig. shows the block diagram of this approximate 4-2 compressor and the expressions below describe its outputs.

The delay of the critical path of this approximate design is . So it is less than the previous designs; moreover, a further reduction in the number of gates is accomplished.

$$Sum' = (\overline{x1 \oplus x2} + \overline{x3 \oplus x4}) \quad \text{....(8)}$$

$$Carry' = \overline{(\overline{x1x2} + \overline{x3x4})} \quad \text{....(9)}$$

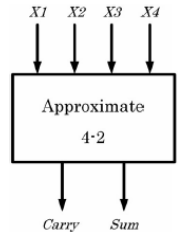

Fig 1. Optimized 4-2 compressor

Approximate design for a 4-2 compressor; this Table also shows the difference between the exact decimal value of the addition of the inputs and the decimal value of the outputs produced by the approximate compressor.

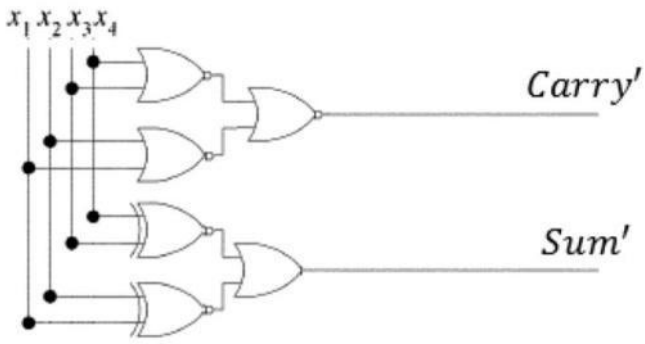

Fig 2. Gate level implementation

For example when all inputs are 1, the decimal value of the addition of the inputs is 4. However, the approximate compressor produces a 1 for the carry and sum. The decimal value of the outputs in this case is 3; shows that the difference is –1. This design has therefore four incorrect outputs out of 16 outputs, so its error rate is now reduced to 25 percent. This is a very positive feature, because it shows that on a probabilistic basis, the imprecision of proposed design is smaller than the other available schemes.

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. This results in one error in the *Sum* computation as seen in the truth table of

approximate half-adder . A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch

$Sum = x1 + x2$

$Carr\ y = x1 \cdot x2$

Table 1: Truth table of Approximate Compressor

X4	X3	X2	X1	Carry'	Sum'	Difference
0	0	0	0	0	1	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

Table 2: Truth table of Approximate Half Adder

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

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. *Carr y* 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

$W = (x1 + x2)$

$Sum = W \oplus x3 \quad Carr\ y = W \cdot x3.$

Table 3: Truth table of Approximate Full Adder

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

V.Results

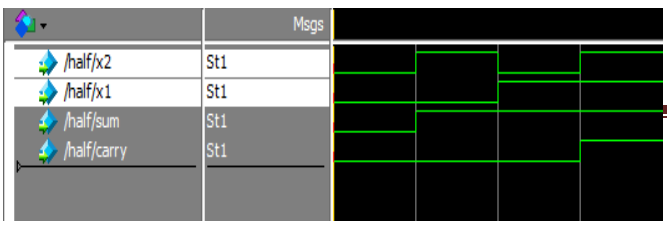

Fig3 approximate half adder

Fig4 approximate full adder

VI. CONCLUSION

In the proposed the partial product perforation technique for producing approximate hardware multipliers. The proposed technique omits a number of partial products enabling high area and power savings while retaining high accuracy. Through a rigorous error analysis, we analytically characterized the induced error metrics proving that the error is bounded and predictable and we proposed two error correction methods that trade a small increase in power for high error reduction. We explored product perforation on a large set of multiplier architectures, evaluating its impact on different architectures and error bounds. In comparison to the state-of the- art approximation techniques, we showed that the proposed approach achieves significant gains in power, area, and quality metrics of image processing and data analytics algorithms. Finally, we showed that our technique is scalable, offering results as the multiplier's bit width increases.

References

[1] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in Proc. 50th ACM/EDAC/IEEE Design Autom. Conf., May/Jun. 2013, pp. 1–9.

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

[3] S. T. Chakradhar and A. Raghunathan, "Best-effort computing: Re-thinking parallel software and hardware," in Proc. 47th ACM/IEEE
Design Autom. Conf., Jun. 2010, pp. 865–870.

[4] E. J. King and E. E. Swartzlander, "Data-dependent truncation scheme for parallel multipliers," in Proc. Conf. Rec. 31st Asilomar Conf. Signals, Syst. Comput., Nov. 1998, pp. 1178–1182.

[5] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant," in Proc. 6th VLSI Signal Process., Oct. 1993,
pp. 388–396.

[6] Y. Liu, T. Zhang, and K. K. Parhi, "Computation error analysis in digital signal processing systems with overscaled supply voltage," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 4, pp. 517–526, Apr. 2010.
[7] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: Imprecise adders for low-power approximate computing," in Proc. Int. Symp. Low Power Electron. Design, Aug. 2011, pp. 409–414.
[8] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in Proc. 24th Annu. Conf. VLSI Design, Jan. 2011, pp. 346–351.