IPHV8I2038X
International Journal Of Advanced Research and Innovation -Vol.8, Issue .II
ISSN Online: 2319 – 9253
Print: 2319 – 9245

A NOVEL APPROACH OF AREA-EFFICIENT APPROXIMATE WALLACE TREE MULTIPLIER

[#1]B.RAJENDAR, Associate *Professor & HOD,*
[#2]PEDDI ASHWINI,
Dept of ECE,
MOTHER THERESSA COLLEGE OF ENGINEERING & TECHNOLOGY, PEDDAPALLI, TS, INDIA.

Abstract: Today in sub-nanometer regime, chip/system designers add accuracy as a new constraint to optimize Latency-Power Area (LPA) metrics. In this paper, we present a new power and area-efficient Approximate Wallace Tree Multiplier (AWTM) for error-tolerant applications. We propose a bit-width aware approximate multiplication algorithm for optimal design of our multiplier. We employ a carry-in prediction method to reduce the critical path. It is further augmented with hardware efficient pre computation of carry-in. We also optimize our multiplier design for latency, power and area using Wallace trees. Accuracy as well as LPA design metrics are used to evaluate our approximate multiplier designs of different bit-widths, ***i.e***. 4 × 4, 8 × 8 and 16 × 16. The simulation results show that we obtain a mean accuracy of 99.85% to 99.965%. Single cycle implementation of AWTM gives almost 24% reduction in latency. We achieve significant reduction in power and area, ***i.e***. up to 41.96% and 34.49% respectively that clearly demonstrates the merits of our proposed AWTM design. Finally, AWTM is used to perform a real time application on a benchmark image. We obtain up to 39% reduction in power and 30% reduction in area without any loss in image quality.

Keywords: *Approximate Multiplier; Bit-Width Aware Multiplication Algorithm; Wallac*

I. INTRODUCTION

The International Technology Roadmap for Semiconductors (ITRS) [1] has anticipated imprecise/approximate designs that became a state-of-the art demand for the emerging class of killer applications that manifest inherent error-resilience such as multimedia, graphics, and wireless communications. In the error-resilience systems, adders and multipliers are used as basic building blocks and their approximate designs have attracted significant research interest recently. Conventional wisdom investigated several mechanisms such as truncation [2], over-clocking, and voltage over scaling(VOS) [3] which could not configure accuracy as well as Latency-Power-Area (LPA) design metrics effectively. Most of the other design techniques rely on functional approximations and a wide spectrum of approximate adders like [4], [5], [6] and [7] have been proposed in the past. However, very few research papers are reported on approximate multipliers in the literature. Most of the approximate multiplier designs reported shorten the carry chains in which error is configurable and the algorithms employed in the designs are for smaller numbers and give large magnitude of error as the bit-width of operands tree increases. In this paper, we present a new Approximate Wallace Tree Multiplier (AWTM) based on a bit-width aware algorithm. We design it specifically to give good results for large operands. Besides accuracy, the AWTM is also optimized for power and area. For single cycle implementation, AWTM gives significant reduction in latency as well. Our contributions are: We propose a new power and area-efficient AWTM based on a bit-width aware multiplication algorithm. We employ a novel Carry-in Prediction technique which significantly reduces the critical path of our multiplier. We further derive an efficient carry-in pre-computation logic to accelerate the carry propagation. We obtain a very high mean accuracy of 99.965% (mean error of only 0.035%) when the size of operands are 10 bits or more. However, if there is no lower bound on the size of operands, the mean accuracy varies from 99.85% to 99.9% (a very small mean error of 0.1% to 0.15%). We achieve a significant reduction in power and area, i.e. up to 41.96%, and 34.49% respectively for the 16- bit accuracy configurable AWTM design. For single cycle implementation of 16 × 16 AWTM, we also reduce the latency by around 24%. Our proposed AWTM, when used for a real time application on an image, achieved up to 39% reduction in power and up to 30% reduction in area with negligible loss in image quality. Rest of the paper is organized in various sections. In section 2 we discuss some background and related work reported in literature. We describe some preliminaries in Section 3. An approximate multiplier architecture is explained in Section 4. We propose a bit-width aware approximate multiplication algorithm in Section 5. We present AWTM design based on the proposed methodology and its optimization w.r.t. LPA design metrics in Section 6. The experimental results are given in Section 7. Finally, we conclude the paper in Section 8.

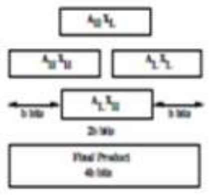
II. BACKGROUND AND RELATED WORK

Research on approximate arithmetic circuits mainly reported in the literature is on approximate adders. It is worthwhile to study these approximate adders in order to make research contributions on approximate multipliers. Lu [8] proposed ak-bit carry look-ahead adder in which only previous k bits are considered to estimate current carry signal. Lu‟s adder exhibits a low probability of getting correct sum and increases area overhead. Shin et al. [9] reduce data-path delay and re-design the data-path modules. It cuts the critical-path in carry-chain to exploit a given error rate to improve parametric yield.Zhu et al.[4] manifest an error-tolerant adder: ETA-I. ETA-I divides inputs into: 1) Accurate part, and 2) Inaccurate part. In the latter, no carry signal is considered at any bit position. Gupta et al. [10] target low-power and propose five different versions of mirror adder by reducing the number of transistors and internal node capacitance. Verma et al. [6] presented a Variable Latency Speculative Adder (VLSA) which provides approximate/accurate results but gives considerable delay and large area overhead. Kahng et. al [7] proposed an accuracy configurable adder with reduced critical-path and error rate. In contrast with the above work, very few researchers have reported work on approximate multipliers. Sullivan et al. [11] used *Truncated Error Correction* (TEC) to investigate an iterative approximate multiplier in which some amount of error correcting circuitry is added for each iteration. This circuitry replicates the effects of multiple pipeline iterations for the most problematic inputs quite inexpensively. Kulkarni et al. [12] proposed a 2 × 2 under designed multiplier block and built arbitrarily power aware inaccurate multipliers. Kyaw et al. [13] presented an Error Tolerant Multiplication (ETM) algorithm in which the input operands are split into two parts. a multiplication part consists of higher order bits and a non-multiplication part with the remaining lower order bits. The multiplication begins at the point where the bits split and move simultaneously towards the two opposite directions till all bits are taken care of. The ETM exhibited a significant reduction in delay, power and hardware cost for specific input combinations. Next, we explain the preliminary concepts so as to understand the proposed approximate multiplier.

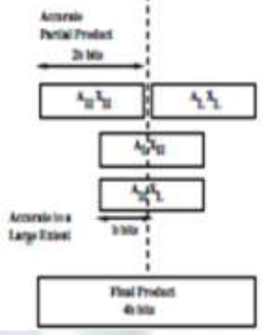
III. PRELIMINARIES

We make use of a simple recursive multiplication for our approximate multiplier design and use various accuracy design metrics [7], [13] for its evaluation. The recursive multiplication and the accuracy design metrics are described in the following subsections.

A. Recursive Multiplication: A given multiplication can be recursively broken down into several smaller-size multiplications, each of which can be performed in the same clock cycle. Let A be the multiplicand and X be the multiplier and both are of $2b$ bits each. A and X can also be written as $A = AHAL$ and $X = XHXL$ where AH, AL, XH, and XL are of b bits each. The multiplication $A \times X$, which is $2b \times 2b$, can be recursively carried out as shown in Fig. 1(a). In this multiplication, $AHXL$, $AHXH$, $ALXL$, and $ALXH$ are partial products, each of which is a $b \times b$ multiplication. Hence, a $2b{\times}2b$ multiplication is divided into four $b{\times}b$ multiplications followed by additions. Fig. 1(b) is derived from Fig. 1(a) for approximate multiplication.



(a)



(b)

Fig.1.a)Recursive multiplication, b) Approximate multiplication.

B. Accuracy Design Metrics: The accuracy design metrics are defined as follows:

1. Relative Error: Relative Error can be calculated as (1) Here, Rc is correct result and Re is approximate result. We denote accuracy as (2)

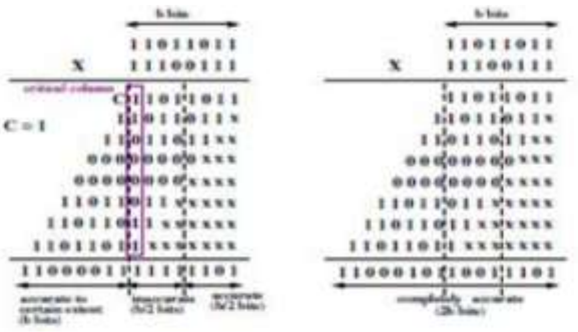
2. Mean Error: Mean error is the average of relative errors of all the combinations tested in an algorithm.

3. Minimum Acceptable Accuracy (MAA): Minimum Acceptable Accuracy is the minimum level of accuracy that an application can tolerate.

4. Acceptance Probability (AP): It is the probability that accuracy of the approximate arithmetic circuit is higher than the minimum acceptable accuracy. Its value is given by $AP = (ACCamp > MAA)$. In the next section, we discuss an approximate multiplier architecture to explain the bit-width aware algorithm proposed

IV. APPROXIMATE MULTIPLIER ARCHITECTURE

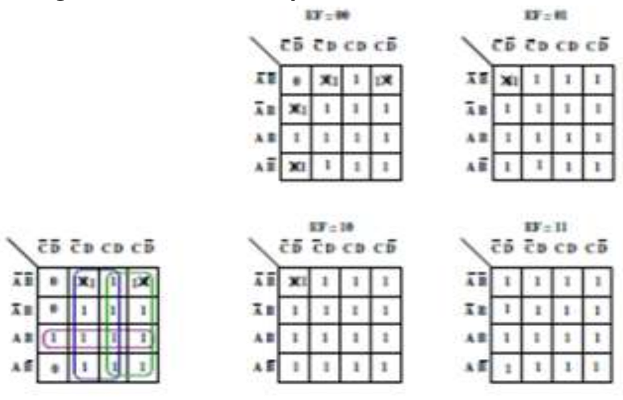
In order for the multiplier to exhibit high accuracy, the most significant bits (MSBs) of the final $4b$ bit product $(A \times X)$ should be accurate to high extent. Therefore, we make the multiplier $AHXH$ as $b \times b$ accurate multiplier and $AHXL$, $ALXH$, $ALXL$ as $b{\times}b$ approximate multipliers. As we shall see, the $b \times b$ approximate multipliers generate upper b bits as accurate to high extent, which further makes the pper $2b$ bits of final $4b$ bit product achieve high accuracy. The same is illustrated in Fig. 1(b). We have explained the design methodology of these approximate $b{\times}b$ multipliers in the Carry-in Prediction Logic[14]. We briefly explain this novel technique in the following subsection with the help of an example. A. The Carry-in Prediction − An Example Consider the unsigned multiplication of two 16-bit numbers (i.e. b = 8): A = ($AEDB$)16 = (44763)10 X = (B) Now, let us evaluate one approximate product out of $AHXL$, $ALXH$ and $ALXL$ using our algorithm. Say, we want to evaluate $ALXL$ i.e (DB)16 ×(E7)16. As shown in Fig. 2(a), we divide this multiplication in three independent parts: First, accurate computation of b/2 least significant bits (LSBs), followed by second part wherein b/2 bits are simply set to 1‟s. The third part is again accurate computation of remaining elements in the multiplication tree with an additional carry „C‟ arising from the inaccurate part at least significant position. So, the idea is to precompute „C‟ through some mechanism and begin multiplication simultaneously from both first and third part. At the same time, we reduce the number of addition operations involved by directly setting the bits in second part, thus significantly reducing the hardware costs. Fig. 2(a) further shows a critical column as the column containing maximum number of elements in the multiplication tree. Carry-in Prediction logic exploits the fact that if there are two or more 1‟s in the critical column, then a carry of at least 1 is definitely propagated to the next column. Next subsection discusses this prediction in more detail. In the second part, we set the b/2 bits in the inaccurate part as 1‟s because for such a large b (>= 5), it is very probable that carry propagated from critical column is more than 1. Therefore, setting those bits will reduce the error involved as it is analogous to the difference between 16 ($5'b$10000) and 15 ($5'b$01111) i.e. 16 just passes an extra carry. Fig. 2(b) shows the accurate $ALXL$ evaluation. As evident, out of 8 most significant bits (MSBs), 6 are correct in our approximate $ALXL$. Evaluating $AHXL$ and $ALXH$ in a similar fashion and adding all these as indicated in Fig. 1(b) gives approximate result as (7$CEBA$7FB)16. The correct answer is (7CED799D)16. The relative error in this case is merely 0.0056%. Precomputation of Carry-in „C‟ is described next in detail. B. Efficient Carry-in Pre-computation Carry-in Prediction necessitates the precomputation of carry-in. Since we are dealing with error resilient systems, We



(a) (b)

Fig.2. carry-in prediction example for b=8(i.e 16x16 multipplication).

Now, let us evaluate one approximate product out of $AHXL$, $ALXH$ and $ALXL$ using our algorithm. Say, we want to evaluate $ALXL$ i.e (DB)16 ×(E7)16. As shown in Fig. 2(a), we divide this multiplication in three independent parts: First, accurate computation of b/2 least significant bits (LSBs), followed by second part wherein b/2 bits are simply set to1‟s. The third part is again accurate computation of remaining elements in the multiplication tree with an additional carry „C‟ arising from the inaccurate part at least significant position. So, the idea is to precompute „C‟ through some mechanism and begin multiplication simultaneously from both first and third part. At the same time, we reduce the number of addition operations involved by directly setting the bits in second part, thus significantly reducing the hardware costs. Fig. 2(a) further shows a critical column as the column containing maximum number of elements in the multiplication tree. Carry-in Prediction logic exploits the fact that if there are two or more 1‟s in the critical column, then a carry of at least 1 is definitely propagated to the next column. Next subsection discusses this prediction in more detail. In the second part, we set the b/2 bits in the inaccurate part as 1‟s because for such a large b (>= 5), it is very probable that carry propagated from critical column is more than 1. Therefore, setting those bits will reduce the error involved as it is analogous to the difference between 16 ($5'b$10000) and 15 ($5'b$01111) i.e. 16 just passes an extra carry. Fig. 2(b) shows the accurate $ALXL$ evaluation. As evident, out of 8 most significant bits (MSBs), 6 are correct in our approximate $ALXL$. Evaluating $AHXL$ and $ALXH$ in a similar fashion and adding all these as indicated in Fig. 1(b) gives approximate result as (7$CEBA$7FB)16. The correct answer is (7CED799D)16. The relative error in this case is merely 0.0056%. Precomputation of Carry-in „C‟ is described next in detail. B. Efficient Carry-in Precomputation Carry-in Prediction necessitates the precomputation of carry-in. Since we are dealing with error resilient systems,



(a) (b)

Fig.3.carry-in precomputation for (a) b=4i.e 8x8 multiplier (b) b=6 i.e 12x12 multiplier.

We can further simplify and approximate the evaluation of carry in so that reduction in latency is not chieved at the cost of power. We consider the cases of b = 4 and b = 6 in order to better explain carry-in precomputation procedure. The precomputation is made hardware efficient by making minor changes in the K-Maps of the carry-in expressions as shown in Fig. 3. Here A,, . . . , F are the elements in critical column. The original K-Maps are obtained from the statement of Carry-in Prediction Logic i.e. Cin = 1 if 2 or more elements of critical column are 1. Fig. 3 further derives · for b = 4: By making changes in 2 cases out of 16, we can simplify the Carry-in expression to $Cin = A.B + C + D$ Similar results can also be derived for b < 4. · for b = 6: We make changes in 6 cases out of 64 and get $Cin = A + B + C + D + E + F$ This is same as OR operation of all the elements present in critical column. Therefore, in general, we can state that for large b (greater than 4), one should take the OR of all the elements present in critical column to get Cin. Next, we propose various accuracy configurations and a bitwidth aware approximate multiplication algorithm.

V. BIT-WIDTH AWARE APPROXIMATE MULTIPLICATION

In the approximate multiplication, we divide the $b \times b$ accurate multiplier $AHXH$ into 4 smaller components, each being a $b/2 \times b/2$ multiplier. This is because, when accurate $AHXH$ is performed in parallel with approximate $AHXL$, $ALXH$ and $ALXL$, the critical path will still be determined by the accurate multiplier. Therefore, recursively reducing it to smaller multipliers will make approximate $b{\times}b$ multipliers as deciding factors of critical path as they are more critical than accurate $b/2 \times b/2$ multipliers. In other words, the stage 1 of the pipelined approximate multiplier effectively consists of 7 multipliers. The designation

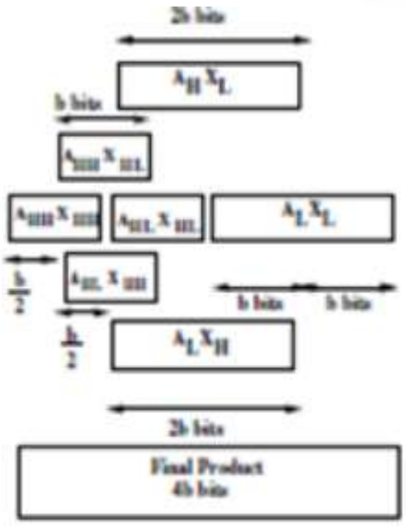
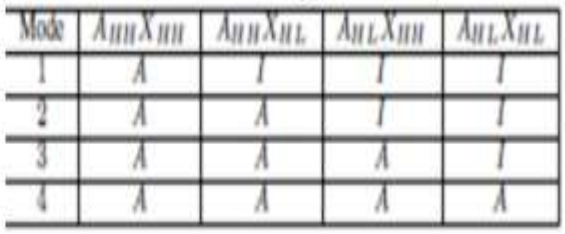


Fig.4.Latency-Driven pipelined approximate multiplier

TABLE I: Modes Operation Of Accuracy Configurable Muultiplier



Mode	$A_{HH}X_{HH}$	$A_{HH}X_{HL}$	$A_{HL}X_{HH}$	$A_{HL}X_{HL}$
1	A	I	I	I
2	A	A	I	I
3	A	A	A	I
4	A	A	A	A

set to 2 and not as 4. Further, the positions at which middle $b/2$ bits are set to 1 also changes with operand bit-width. This has already been indicated in Fig. 4. This is plausible because at a time, we will use a multiplier of fixed size depending on application and hence can program it accordingly. We present our algorithm (see Algorithm 1) of approximate $b \times b$ partial product computation (e.g. $AHXL$) in its most general form. This algorithm is coded later as a C-Program for simulation purposes. It should be noted that the index 0 is the Most Significant Bit in the Algorithm 1.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present power and area results obtained experimentally. All results have been produced as shown in Figs.

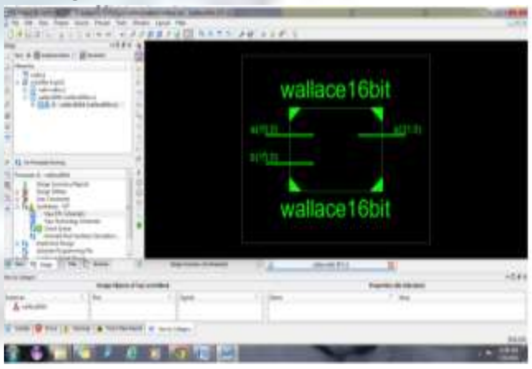


Fig.5. Modified Wallace Multiplier Block level of 16 bit

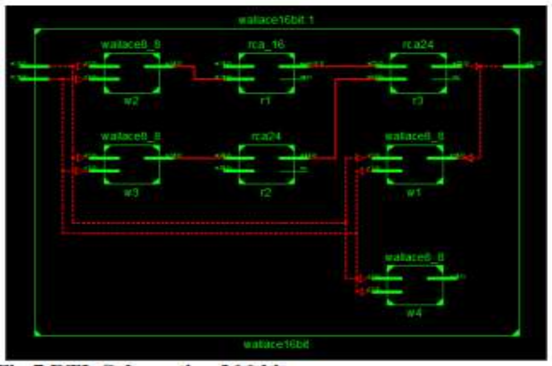


Fig.6 .RTL Schematic of 16 bit.

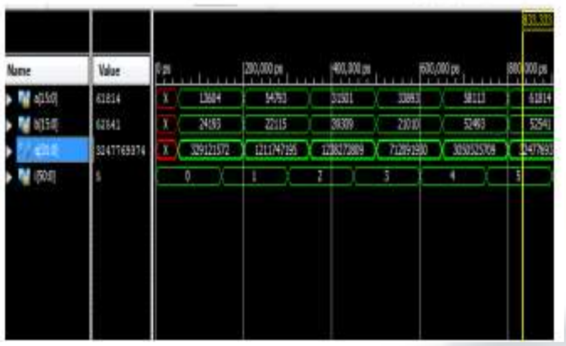


Fig.7.Simulation Results of 16 bit.

VII. CONCLUSION

We proposed a power and area-efficient Approximate WallaceTree Multiplier (AWTM) for error-resilient systems. A new bit-width aware approximate multiplication algorithm is also presented. The AWTM design is further empowered with Carry-in Prediction logic and its efficient precomputation to increase overall throughput. Our powerarea efficient AWTM is fast, particularly for operands size of 16-bit or more, and optimized w.r.t. LPA design metrics. Single cycle implementation of AWTM showed a 23.91% reduction in latency. We obtained the mean accuracy of 99.85% to 99.965% for 16-bit multiplication of different sized operands. We also achieved significant reduction in power and area of our multiplier design, up to 41.96% and 34.49% respectively for 16-bit multiplication which clearly demonstrates efficiency and effectiveness of AWTM. Finally, we demonstrated that AWTM produced images of almost the same quality as obtained by operations using accurate multipliers but with power and area savings of around 39% and 30% respectively.

REFERENCES

[1] "International technology roadmap for semiconductors, http://www.itrs.net."

[2] E. J. Swartzlander, "Truncated multiplication with approximate rounding," in Signals, Systems, and Computers, 1999. Conference Record of the Thirty-Third Asilomar Conference on, vol. 2, oct. 1999, pp. 1480–1483 vol.2.

[3] L. N. Chakrapani, K. K. Muntimadugu, L. Avinash, J. George, and K. V.Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: a mathematical foundation and preliminary experimental validation," CASES, pp. 187–196, 2008.

[4] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 18, no. 8, pp. 1225–1229, aug. 2010.

[5] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power highspeed adder for error-tolerant application," in SoC Design Conference (ISOCC), 2010 International, nov. 2010, pp. 323–327.

[6] A. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in Design, Automation and Test in Europe, 2008. DATE "08, march 2008, pp. 1250–1255.

[7] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, june 2012, pp. 820–825.

[8] S. L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67–73, mar 2004.

[9] D. Shin and S. Gupta, "A re-design technique for datapath modules in error tolerant applications," in Asian Test Symposium, 2008. ATS "08. 17th, nov. 2008, pp. 431–437.

[10] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 32, no. 1, pp. 124–137, jan. 2013.

[11] M. B. Sullivan and E. E. Swartzlander, "Truncated error correction for flexible approximate multiplication," in Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on, 2012, pp. 355–359.

[12] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in VLSI Design (VLSI Design), 2011 24th International Conference on, 2011, pp. 346–351.

[13] K. Y. Kyaw, W.-L. Goh, and K.-S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in Electron Devices and Solid-State Circuits (EDSSC), 2010 IEEE International Conference of, 2010, pp.1–4.

[14] K. Bhardwaj and P. S. Mane, "Acma: Accuracyconfigurable multiplier architecture for error-resilient systemon-chip," in Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop on, July 2013, pp. 1–6.

[15] C. S. Wallace, "A suggestion for a fast multiplier," in Electronic Computers, 1964 IEEE Transactions on, 1964, pp. 14–17.