Optimizing Encoder and Decoder Blocks for a Power-Efficient Radix-4 Modified Booth Multiplier

1st Jean. C. Scheunemann
PGEEC
Catholic University of Pelotas
Pelotas, Brazil
ORCID 0000-0002-0277-171X

2nd Marlon S. Sigales
PPG SPAF
Federal University of Pelotas
Pelotas, Brasil
ORCID 0000-0001-8329-6430

3rd Mateus B. Fonseca
Engineering Center
Federal University of Pelotas
Pelotas, Brazil
ORCID 0000-0002-6086-2820

4th Eduardo. A. C. da Costa
PGEEC
Catholic University of Pelotas
Pelotas, Brazil

2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI) | 978-1-6654-2170-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SBCCI53441.2021.9529975

***Abstract*—The conventional modified Booth multiplier comprises an encoder and a decoder, which produce partial products by adjusting the multiplicand according to 3-bit windows generated in the segmentation of the multiplier. The partial products are added later with the necessary left shifts in each one. They produce accurate results, more useful in processes where the error is not acceptable, like divisions, transformations and filters. This paper proposes optimizing the radix-4 Modified Booth encoder (MBE) and decoder (MBD) for a power-efficient multiplier. Two new topologies are presented for optimizing the encoder and decoder set so that the partial product terms avoid unnecessary operations. The proposed encoder and decoder structures are highly regular, with few logic gates, and easily parallelized for any number of input bits. The results show that the proposed optimizations reveal gains in area and power compared to the conventional multiplexer-based multiplier. When applying the proposed multiplier in the butterflies of the Fast Fourier Transform, the proposed multipliers are more efficient with gains in the area, power, and power-delay-product (PDP) compared with the multipliers using the '*' operator from the literature.**

***Index Terms*—Radix-4 modified Booth multiplier, partial product terms, power-efficient design**

I. Introduction

Digital multipliers are widely used in Digital Signal Processing circuits such as digital filters and Fast Fourier Transform (FFT), or being part of other more complex digital arithmetic circuits, such as dividers [1]. In such circuits, the multipliers are often the most responsible for power consumption.

The fastest types of precise multipliers are the parallel ones. Among these, Wallace multipliers [2] are between the most immediate. However, when considering regularity, high-performance, and low power, Booth multipliers tend to be the first choice [3]. Notably, the Modified Booth algorithm achieves a significant performance improvement through radix-4 encoding.

Over the years, the literature's works have proposed more regular and suitable multiplier designs based on the Booth recoding techniques [4] [5], since the Modified Booth algorithm uses approximately half of the partial products. Although the Booth algorithm is straightforward, its design becomes difficult sometimes, mainly when using higher radixes. It occurs due to the complexity of pre-computing an increasing number of multiples of the multiplicand within the multiplier unit. This work proposes new topologies to generate the partial products in the radix-4 Modified Booth. The proposed strategy simplifies the encoder's internal logic that produces the partial products avoiding unnecessary operations, as well as simplifying the decoders differently in each proposed structure, reducing the critical path of operations. For this purpose, we offer two topologies for optimizing the encoders and decoders. A basic topology (Structure A) is generated from direct simplifications of the logic of the Booth algorithm, using multiplexers (MUX) for the selections of which output would be sent to the partial product. The first proposed topology (Structure B) reduces two multiplexers and connects the signal bits directly with the partial products. Beyond the simplifications of the first topology, the second one (Structure C) reduces one multiplexer more using controlled inverters with XNOR gate. The simplifications provided by the two proposed topologies show less area and power than the conventional MUX-based encoder's topology and in addition the proposed structures offer a reduced critical path. The benefits of using the proposed encoders in the Modified Booth multiplier is proved by showing gains in area and power compared with the multipliers from the literature. The proposed multipliers were tested in a FFT butterfly circuit for 16, 32 and 64 bits to validate our proposed approaches. The results show gains in the area, power, and power-delay-product (PDP) using our best multiplier than using the '*' multiplier from the tool. Therefore, the principal contributions of this work is propose new simplified and regular encoder and decoder topologies to generate the partial products in the Booth radix-4 multiplier.

This paper is organized as follows: Section II presents an overview of the Modified Booth multiplier. The principal works from the literature are shown in Section III. The proposed approach for the optimized Modified Booth multiplier is presented in Section IV. The synthesis results and comparison with the literature are offered in Section IV-B. A case study is presented in Section V. Finally, Section VI concludes the paper.

II. Modified Booth Overview

The Booth multiplication algorithm consists of a multiplication algorithm in 2's complement based on the reorganization of the numbers for the partial sums, reducing total sums. The original algorithm (radix-2) is very similar to a typical shift-add operation, except for containing some subtractions, with no advantages since the number of partial products is the same number of bits as the multiplier [3].

Figure 1 shows an overview of the Booth multiplication operation. The Multiplier (MR) is encoded in windows (multiplier Booth encoder - MBE). The partial products (PPs) are generated by the decoder (multipler Booth decoder - MBD) and finally the PPs are added (Adder Tree).

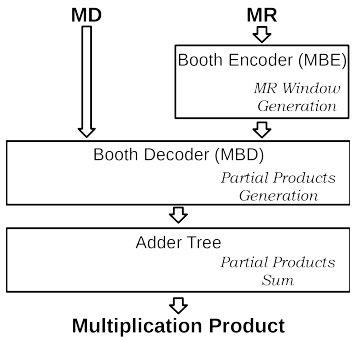

Fig. 1. Booth multiplication product scheme

The radix-4 Booths algorithm (also called Modified Booth) reduces the number of partial products by encoding the 2's complement multiplier. The multiplier operand generates control signals (0, +MD, +2MD, -MD, and -2MD) for each group of 3-bit from MR (windows). A multiplexer produces the partial products according to the encoded control signal. The PPs terms are sign-extended up. Table I shows the multiplicand operand values assumed according to the 3-bits of the multiplier word for generating the partial products.

TABLE I
Modified Booth Algorithm Partial Product Generation

Modified Booth Algoritm			
MR Window			Partial_product
MR(i+1)	MR(i)	MR(i-1)	
0	0	0	0
0	0	1	MD
0	1	0	MD
0	1	1	2*MD
1	0	0	-2*MD
1	0	1	-MD
1	1	0	-MD
1	1	1	0

The algorithm described in Table I represents the Modified Booth Algorithm and has the characteristic of reducing the number of sums by half the number of bits in the multiplier. We should stress further that, in contrast to the architectures presented in this work, raising the radix for the Booth architecture is a difficult task, thus not being able to leverage from the potential savings of higher radices. Other higher radices algorithms are more successful in this task.

In the radix-4 Modified Booth algorithm, the term multiplier (MR) is processed in into groups of three bits for processing the multiplication (MR window in the Table I). According to the three bits value, the multiplicand term (MD) assumes a value accordingly, as seen in Table I. Initially, a zero value with the least two significant bits composes the three first bits in operation, as shown in Fig. 2. After that, one uses the most significant bit of each group of three bits as the least significant bit for the next group as overlapping (Fig. 2). For each operation, we shift the partial product two positions left, representing a radix-4 operation. An 8-bit process is exemplified in Fig. 2, identifying four MR Windows generation.

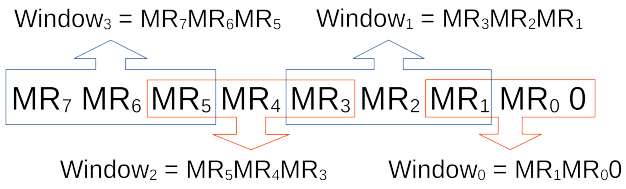

Fig. 2. Radix-4 Booth encoder from Table I for 8-bit example.

The windows from Fig. 2, obtained from MR, are then submitted to the logic of Table I using an MBD to generate the control signals. Thus, the MBE can act where the MD signals flow, obtaining the partial products added afterward. In each partial product, i zeros are inserted to the right, and its signal bit is replicated in the others MSB of a data with double the MD bits.

Finally, the number of partial product sums is equal to half the number of MR bits is made, obtaining the multiplication product result.

We propose regular and optimized structures for both Booth decoding and encoding (MBD and MBE) in this work. For a parallel generation of partial products, MBD and MBE are replicated the same number of times as the number of sums. Thus, the critical path, delay, and power are minimal and, in general, attributed to the sum trees. We can see the proposed structure in Section IV.

III. Related Work

Over the years, the literature offers several strategies to turn the Modified Booth multiplier more power-efficient. The work in [6] proposes the implementation of a 16×16 Modified Booth multiplier. The strategy consists of using Booth encoder multiplier with Carry-Select Adder (CSA). However, the work does not report area, power, and performance results. An area-efficient low power Modified Booth multiplier for FIR filter was proposed in [7]. Although the work promises a low power Modified Booth, there are no proposed strategies to optimize the multiplier. Furthermore, the presented power results are only FPGA-based. The work in [8] introduces a structured and efficient recoding technique for Modified Booth multiplier and explores three different schemes by incorporating them in

a fused Add-Multiply (FAM) operator. Although the authors claim that he proposed recoding schemes yield considerable performance improvements compared to the most efficient recoding schemes found in literature, there are no comparison results to prove it. An optimization in the encoder of the Modified Booth multiplier is proposed in [9], which rather than processing the 2's complement converter with NOT logic and one addition circuit, moves the addition part to the Wallace Tree Adder in the following step. The authors show gains in the delay results, comparing to the traditional Booth recoding, at the cost of a slight increase in FPGA resources, but with no power results report.

Regarding ASIC implementation, the work in [10] proposes the combination of partial products partition into two parts for independent parallel column compression and acceleration of the final addition using a new hybrid adder for Baugh-Wooley and Booth-encoded multipliers. The hybrid adder comprises carry lookahead and carry select adders. There are no optimizations in the Modified Booth encoder. The strategy in [11] consists of implementing a radix-4 8x8 Booth multiplier using four stages with a unique optimised stage-1 architecture. Instead of using adder/subtractor in stage-1, a novel binary-to-2's complement converter and a 2:1 MUX replace it. This work, like ours, works with the number of bits plus 1 bit of signal length only, instead of working with double the bits, as there are in the other bibliographies. Besides the technology being different (they work with 90nm), one of the differences is that it always selects at the input between 0, MD and 2MD leaving the 2's complement for the sums. In our Structure A, the selection was made between -MD, MD, -2MD and 2MD at the input, while 0 is result of MBE and MBD arrangements. For Structure B, only the selection between -MD and MD is made, and the selection between multiplying by 2 or null is done by MBE and MBD arrangements. Finally for Structure C, there are only the MD at the inputs, and the whole process is done by MBE and MBD, the sum of the 2's complement was reallocated with the sums.

The work in [4] offers a Booth encoded sign-digit-based conditional probability method for a fixed-width multiplier design. The work proposes a multiplexer-based estimation circuit to simplify the compensation logic, and reduces the critical path. Unlike our work, this one, as well as the [12] proposal, uses the approximate Booth algorithm, which works through some statistical inferences trying to achieve the smallest error possible with simplified hardware, being good for applications in video streaming, compression and other inaccurate applications, obtaining smaller areas, smaller delays and potentials compatible with the reduction. Which is not our proposal, since we deliver accurate results, more useful in processes where the error is not acceptable.

IV. Proposed Optimizations on Radix-4 Booth structure

The ways of optimizing the MBE and MBD are the principal targets of this work. The optimizations start from a careful analysis of Table I.

A first observation revealed that zero values fill the generated partial product if the three bits of the window are the same. We solved it with one combinational circuit, as shown in Fig. 3, containing one NAND3 gate, one OR3 gate, and one NAND2 gate. Therefore, one sets the output when at least one of the inputs differs from the others. This combinational circuit (called Structure A), controls a multiplexer that selects an input with zero value when the results should be low logic-level. On the other hand, in the proposed topologies of Fig. 4 (Structure B) and Fig. 5 (Structure C) one AND2 gate replicates this signal at each bit, which propagates zero values when the signal is low logic-level indicating that the inputs are the same.

Another observation considered the possibility of left shifts when the two least significant bits (LSB) of the windows are the same. If the input is zero value, there is no problem regarding the final result. For this operation, a XOR2 gate (Fig. 3) verifies if the two LSB bits, MR(i-1) and MR(i), are different or the same. If the values are the same, the multiplexer selects the value of the operand multiplied by two (2MD or -2MD). On the other hand, if the values are different, the original values of the operand are selected (MD or -MD). In the topologies of Fig. 4 and Fig. 5 the XOR2 gate selects the original MD operand with sign extension or shifted one position left (multiplied by two).

Finally, we verified that the MD values are multiplied by minus one when the MSB of the window is equal to one. Thus, this signal realizes this selection. The initial MUX between the MD and 2*MD inputs or their 2's complement selects the MBD topology of Fig. 3. For the topology in Fig. 4, only one MUX is used to select between MD and its 2's complement, since the multiplication by two happens with a shift of this signal in the following logic. A bitwise XNOR structure in MD replicates the same MSB in the topology of Fig. 5. It enables making a controlled inverter, leaving the sum of one as a carry-in for the adder, which is triggered when the window does not have all its bits equal and $MD_{[N-1]}$ is not null, leaving the 2's complement for the sum tree.

The proposed topologies are slightly different from each other, but they all share the regularity, are simple, and can be replicated in parallel for the generation of partial products. The topology presented in Fig. 3 is entirely composed of MUX, like the classic implementations, with the same number of entries and selection of entries through the MBE. This topology leaves the tool as possible simplifications, general in terms of higher timing, power dissipation, area, and higher critical path. The topology in Fig. 4 has some simplifications by eliminating 2 MUX and passing the signal bits directly to the partial product, and having only two inputs ANDs when the partial product will be zero. The topology of Fig. 5 uses all the topology improvements in Fig. 4, and also removes another MUX using XNOR controlled inverters, further reducing the critical path and power dissipated because it leaves the sum of 1 for the sum trees, represented by C_i in on Structure C, increasing one operator on final adder. These simplifications resulted in shorter critical paths and timing, reduction in area and power, avoiding previous multiplications, and placing

only one wire the shifting when the output is shifted in the multiplication by 2.

The design of Fig. 3 structure considered interpreting the information of Table I. The structure of Fig. 4 is a version optimized concerning the amount of MUX used and using only shifts, compared to the previous structure, which requires two multipliers by 2 and two adders. The structure of Fig. 5 removes the -MD generation circuit from the Encoder and places it together with the sum tree.

In the structures of Fig. 3, Fig. 4, and Fig. 5, the term i corresponds to twice the *window* index used in PPs generations.

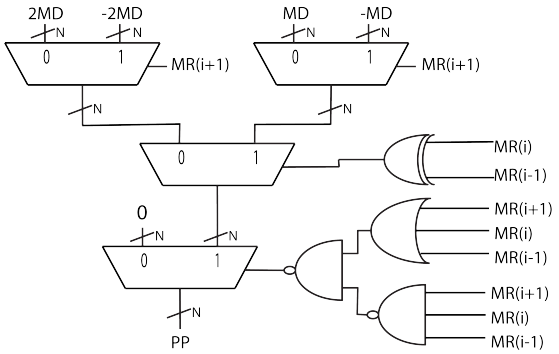

Fig. 3. Structure A - based on the algorithm from Table I. Partial Products generation with modified Booth encoder/decoder.

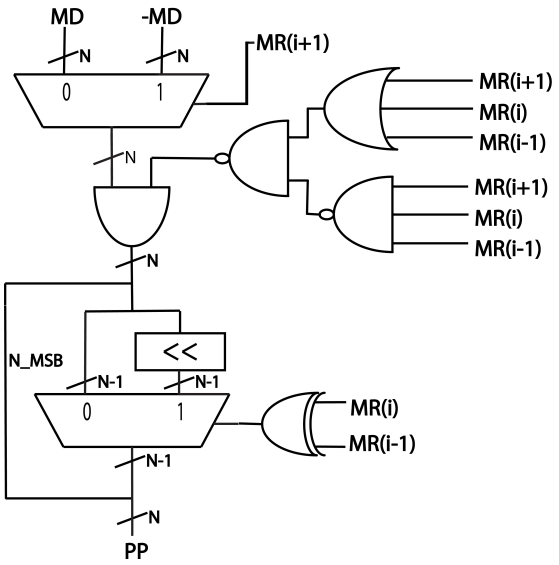

Fig. 4. Structure B - based on the algorithm from Table I. Partial Products generation with Modified Booth Encoder/Decoder.

The proposed circuit in Fig. 5 is new, regular, and simplified when compared with the literature review from Section III as to the best of our knowledge.

A. Partial product sum generation

During the generation of partial sums, the Booth algorithm presents some redundant elements, as highlighted in Fig. 6.

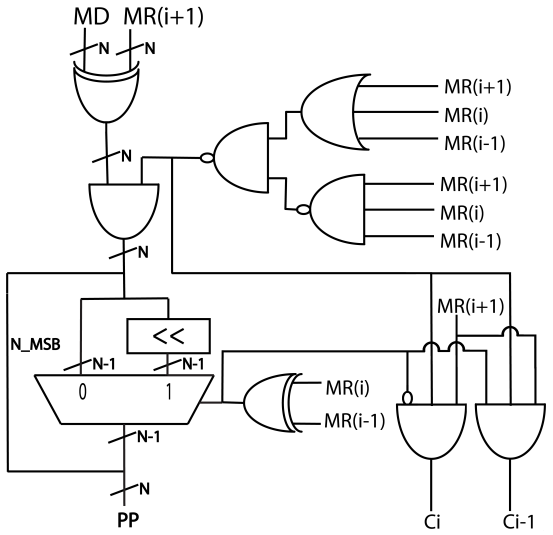

Fig. 5. Structure C - proposed modification. Partial Products generation with modified Booth encoder/decoder.

These elements can be removed in the optimization of the final structure, reducing the final area.

For the validation and comparison of the partial product generation topology, were used the adder tree in the array (Fig. 7), and Wallace (Fig. 8) structures, using standard adders from the synthesis tool. Notably, the Wallace topology is widely used in the Booth multiplier design [13], [14].

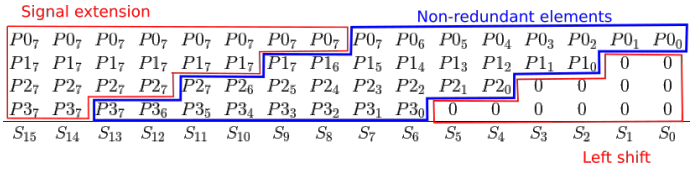

Fig. 6. Redundant elements in 8-bit Booth multiplication

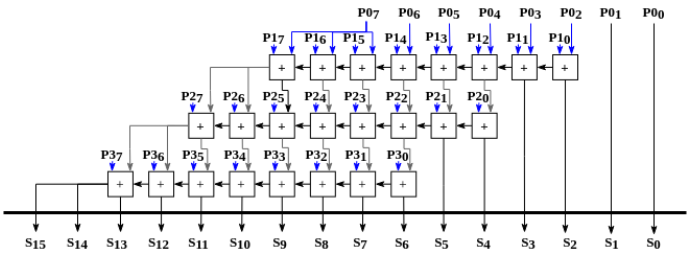

Fig. 7. Array adder example for an 8-bit Booth multiplier example with redundant sums optimizations

B. Multiplier Synthesis Results

Comparison results of the multiplier topologies presented in Section IV and the '*' operator of the synthesized Verilog are presented in Table II, using Cadence RTL compiler for the logical synthesis of Nangate's 45nm library. Stimuli of 10,000

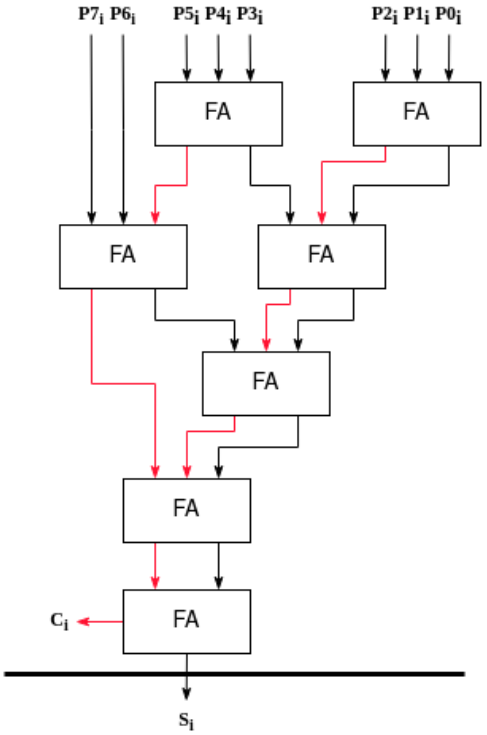

Fig. 8. i Bit-slice Wallace adder for 16 bit-width Booth multiplier. The red lines indicate the carry out propagation.

random combinations at 50MHz were applied to the inputs. The multiplier structure topologies are combined with different adder trees (Array and Wallace) to compare how changing the tree affects the multiplier. We present results regarding delay, area, power, and power delay product (PDP), for several bit-widths.

Equation 1 applies the ratio between the technologies E determining the power dissipation. The power changes quadratically as the voltage and current change in the proportion of $1/E$. This equation makes a fair comparison between the results presented and the results in other technologies present in the bibliography, as in Table II.

According to [15] relationships between different technologies can be established, and the parameters of one technology can be designed for another. In a scenario of integral scaling, where all parameters change according to the technology, one can use the Equation 1.

$$E = \frac{Tech_1}{Tech_2}, \forall E > 1$$
$$P_1 = \frac{1}{E^2} \cdot P_2 \quad (1)$$

The results from Table II shows the tool operator better in PDP than the other topologies for 8, 16 and 32 bits, but not for 64 bits, where structure B was smaller. When it is compared only the power consumption, structure B has always the better performance, regardless the use of the adder tree. Analysing the adder trees, for the same structure and bit-width, the array

TABLE II
BOOTH MULTIPLIER SYNTHESIS RESULTS COMPARISONS FOR VARIOUS BIT-WIDTHS, AND DIFFERENT MULTIPLIERS STRUCTURES WITH ARRAY AND WALLACE ADDER TREES, @45NM CMOS TECHNOLOGY.

Bits	Multiplier Structure	delay (ns)	area (μm^2)	Power (μW)	PDP (pJ)
8	Structure B Array	2353	1053	14,02	0,033
	Structure B Wallace	2159	857	15,32	0,033
	Structure C Array	2448	651	15,92	0,039
	Structure C Wallace	2084	622	17,07	0,036
	[11]*	1,040	-	108.98*	0.113
	Tool†	1807	564	16,81	0,030
16	Structure B Array	4953	3630	73,17	0,362
	Structure B Wallace	3933	3194	86,82	0,341
	Structure C Array	5099	2090	81,70	0,417
	Structure C Wallace	4086	2153	91,40	0,373
	Tool†	3765	1840	85,41	0,322
32	Structure B Array	10040	13467	143,17	1,44
	Structure B Wallace	7755	12025	177,15	1,37
	Structure C Array	10108	7543	156,01	1,58
	Structure C Wallace	7720	7822	184,66	1,43
	Tool†	7414	6948	176,69	1,31
64	Structure B Array	19445	51908	273,02	5,31
	Structure B Wallace	14636	46499	384,80	5,63
	Structure C Array	19607	28219	300,86	5,90
	Structure C Wallace	14612	29429	409,15	5,98
	Tool†	14499	27062	390,22	5,66

† Multiplier provided by synthesis tool
* Used (1)

type has less power consumption, but longer timing and bigger area, which leads to a bigger PDP, except for 64 bits, where the proposed structure C have lower total power, counterbalancing the increase in area with a lower timing.

It is noticeable that structure B, with the Wallace adder, presents a smaller PDP or close to the tool; however, presenting a larger total area for practically all bit-widths. It is also visible that structure C with the Array adder has very similar results to those extracted by the tool. Comparing B and C structures, the delay varies depending of the adder tree (having no relevant differences due to the used structure), the area is smaller for the C structure and B structure has less power consumption.

V. CASE STUDY

The Booth multiplier proposed in Section IV is inserted into the complex multiplier at the decimation in time (DIT) FFT butterfly in Fig. 9, and compared with the multiplier operator to prove the efficiency of our proposed approach. The FFT is a good case study because the line of multipliers is followed by adders and subtractors that can be easily aggregated in the optimizations in the adder tree of the Modified Booth multiplier.

A. FFT Butterfly Synthesis Results

Table III shows synthesis results in delay, area, power and PDP for the FFT butterfly from Fig. 9 with different multipliers comparisons.

Table III shows that structure B with Wallace adder tree have the best result in PDP for 16 bit-width, with the smallest delay and power consumption, but with the biggest area. For 32

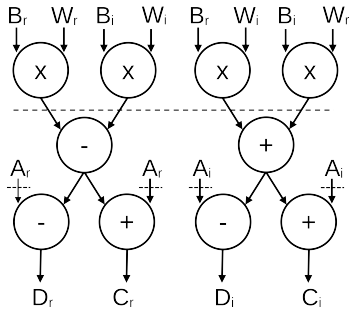

Fig. 9. Conventional FFT radix-2 DIT butterfly structure. The dot lines indicate the pipeline position

TABLE III
BOOTH MULTIPLIER APPLIED TO FFT. SYNTHESIS RESULTS COMPARISONS FOR VARIOUS BIT-WIDTHS, DIFFERENT MULTIPLIERS STRUCTURES WITH ARRAY AND WALLACE ADDER TREES, @45NM CMOS TECHNOLOGY.

Bits	Multiplier Structure	Delay (ns)	Area (μm^2)	Power (μW)	PDP (pJ)
16	Structure B Array	3559	8560	1847,90	6,58
	Structure B Wallace	2498	9539	1744,79	4,36
	Structure C Array	3838	6290	1930,47	7,41
	Structure C Wallace	2598	5056	1800,00	4,68
	Tool†	2512	4764	1758,10	4,42
32	Structure B Array	6828	25192	6693,34	45,70
	Structure B Wallace	4643	30754	6670,47	30,97
	Structure C Array	7097	17914	7187,48	51,01
	Structure C Wallace	4434	16675	6853,27	30,39
	Tool†	4615	18146	6855,07	31,64
64	Structure B Array	13072	91533	17615,46	230,27
	Structure B Wallace	8133	110556	18846,72	153,28
	Structure C Array	13274	60735	17915,66	237,81
	Structure C Wallace	8025	59496	18956,40	152,13
	Tool†	8177	64782	19236,43	157,30

† Multiplier provided by synthesis tool

bit-width, C structure with Wallace adder tree has the lowest PDP, with the best result in timing and area, the lowest power consumption is for the B structure with Wallace adder tree. For 64 bit-width, the results imitate the 32 bit pattern, for area and delay, C structure is better in PDP, delay and area, however, for power consumption, the structures with array adders presented the lowest values, being the lowest shown by B structure.

Notably, structure C presents results very close to the tool but with the lowest PDP for all sizes. Structure B shows a significantly larger area due to replicating the signals caused by -MD, which ends up reflecting in higher power dissipation and worst delay. It is interesting to see that Wallace brings a considerable benefit to structure B in terms of delay. Therefore, structure B with Wallace tree adder presents less PDP than this structure with Array tree adder. However, structure C is the most straightforward for an FFT design with a balance between area, delay, power, and PDP with the best values.

VI. CONCLUSION

This work presented different regular MBE/MBD topologies, easily replicated in parallel, with high efficiency and low complexity. Notably, the Structure B with Wallace tree proved to be the most efficient in terms of power. In addition, it was possible to verify the MBD/MBE arrangements applied in FFT structures. The adder tree of the FFT allied to the MBD/MBE arrangements in the Booth multiplier demonstrated their sovereignty about using the operator '*' of the tool. We intend to apply the proposed multipliers into other digital circuits such as FIR and IIR filters as future work. We also plan to compare the use of the MBE/MBD arrangement in different sum trees. Apply the methodology used in the present work for radix-4 in other bases of the Booth algorithm is also in our plan for future work.

REFERENCES

[1] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing*. Signal Processing Series. Prentice Hall, 2 edition, January 1999.

[2] C. Wallace. A suggestion for a fast multiplier. *IEEE Transactions on Electronic Computers*, 13:14–17, 1964.

[3] A. Booth. A signed binary multiplication technique. *Journal of Mechanics and Aplied Mathematics*, 4:236–240, 06 1951.

[4] Z. Zhang and Y. He. A low-error energy-efficient fixed-width booth multiplier with sign-digit-based conditional probability estimation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65:236–240, 2018.

[5] D. Villeger and V.G. Oklobdzija. Evaluation of booth encoding techniques for parallel multiplier implementation. *Electronics Letters*, 29:2016–2017(1), November 1993.

[6] V. Manjunath, K. Harikiran, S. Manikanta, and Sivanantham K. Design and implementation of 1616 modified booth multiplier. In *International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5, 2015.

[7] G. Haridas and D. George. Area efficient low power modified booth multiplier for fir filter. In *International Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015)*, pages 1163–1169, 2015.

[8] C. Ashok and M. Chandrasen. High-level optimization techniques for low-power modified booth multiplier design of fpga. *International Journal of Innovative Technology and Research*, 5:6132–6137, 2017.

[9] X. V. Luu, T. T. Hoang, T. T. Bui, and A. V. Dinh-Duc. A high-speed unsigned 32-bit multiplier based on booth-encoder and wallace-tree modifications. In *2014 International Conference on Advanced Technologies for Communications (ATC 2014)*, pages 739–744, Oct 2014.

[10] Kittur H. Ramkumar B. Faster and energy-efficient signed multipliers. *Hindawi Publishing Corporation VLSI Design*, pages 1–12, 2013.

[11] H. Xue, R. Patel, N. Boppana, and S. Ren. Low-power-delay-product radix-4 8*8 booth multiplier in cmos. *Electronics Letters*, 54:344–346, 2018.

[12] Liangyu Qian, Chenghua Wang, Weiqiang Liu, Fabrizio Lombardi, and Jie Han. Design and evaluation of an approximate wallace-booth multiplier. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2016.

[13] N. Sharma and R. Sindal. Modified booth multiplier using wallace structure and efficient carry select adder. *International Journal of Computer Applications*, 68:39–42, 2013.

[14] P. Tulasiram, D. Vaithiyanathan, and R. Seshasayanan. Implementation of modified booth recoded wallace tree multiplier for fast arithmetic circuits. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4:798–802, 2014.

[15] Jan Rabaey. *Low Power Design Essentials*. Springer Publishing Company, Incorporated, New York, NY, EUA, 2009.