International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 04 - Issue 06 || June 2019 || PP. 19-25

Low Power High Speed Multiplier for MAC Application

Dr. R. P. Meenaakshi Sundhari [1], S. Sriramsurya[2], M. Ramya[3], E. Revathi[4]
[1,2,3,4] *(Department of Electronics & Communication Engineering,*
P.A. College of Engineering and Technology, Coimbatore, India)

Abstract: In digital applications, multiplication is considered to be a coherent method for buying and selling off energy against performance and accuracy. In this paper, Multiply-Accumulate (MAC) unit is introduced. The MAC unit is designed the use of Dadda multiplier and hybrid adder. This paper proposes a multiplier whose ultimate product is developed by a ripple carry adder(RCA). The partial product tree of the multiplier is related to the proposed tree compressor. A multiplier design is two implemented by utilising the ripple carry adder and the compressor. The typical delay of MAC unit is reduced. Compared with the Wallace tree multiplier, the proposed multiplier (Dadda multiplier) has reduced the delay upto 18.5 and level of logic by 14 depending on the accuracy. The wide variety of paths has also reduced. The functional verification used to be done using ModelSim 5.7g. The MAC architecture with respect to area and delay were generated the use of Xilinx software.

Keywords: Dadda multiplier, Ripple carry adder, Wallace tree multiplier, MAC, compressor.

I. Introduction

Many computer arithmetic applications are resolved using digital logic circuits, as a result of operating with a high degree of accuracy and precision. Many flourishing and prominent applications like Multiply-Accumulator (MAC) unit have naturally progressive quantity of small inaccuracies. Approximate computing is an valuable approach for many applications because it can trade off accuracy for power and it presently plays an significant role in such application area. Distinctive applications have diverse precision necessities, as do diverse program stages in an application. If the accuracy of the multiplication is fixed, then power will be wasted when high accuracy is not required.

This paper aims on an approximate multiplier design that can control accuracy forcefully. A full adder is proposed that can be dynamically designed to function as ripple carry adder, a set of bit-parallel AND gates, or a combination of the two.

The carry maskable adder (CMA) in the final stage of the multiplier is replaced by using the proposed ripple carry adder (RCA).An approximate tree compressor is employed to decrease the accumulation layer depth of the partial product tree. Our approach propose a term representing the delay and efficiency requirements which reduces the partial product reduction (PPR) component as desired. An approximate multiplier is designed by using the ripple carry adder and compressor. This multiplier, together with a standard multiplier and the formerly studied approximate multipliers, was implemented in Verilog HDL using a 45-nm library to evaluate the critical path delay, and design area. Compared with the traditional Dadda tree multiplier, the proposed approximate multiplier decreased the critical path delay by between 29.9% and 60.5%, depending other appropriate computational accuracy. In addition, its design area was 44.6% smaller. Comparisons with the conventional approximate multipliers, none of which have any dynamic re-configurability, establishes that the proposed multiplier implemented the great trade-off of delay and area against accuracy. The multiplier design and the hybrid adder are then Figure out in a MAC unit application.

The remainder of this paper is standardized as follows. Section II analysis the previous works. Section III introduces the Dadda multiplier after explaining the tree compressor and the ripple carry adder. Section IV assess the multipliers experimentally and then evaluates the Dadda multiplier using MAC application. Section V affords our conclusions.

II. Releated Work

The adder is an essential factor of most multipliers. Tongxin Yang et al. [1] proposed the lower-part-OR adder, which utilizes AND gates for addition of the lower bits and OR gates for addition of the upper bits. It is similar to our proposed ripple carry adder (RCA) that it uses AND gates to generate the sum approximately, but our RCA is also energetically reconfigurable.

Liu et al. [2] employed an approximate adder to decrease carry propagation lengthen in partial product accumulation. The bit width of the blunder recuperation vector can be chosen with the aid of the originator to fulfill exactness prerequisites. Momeni et al.[3] The proposed design accomplish two significant reductions in power dissipation, delay and transistor count compared to an specific design. The deferral of the minimize hardware of a Dadda multiplier is reliant on the quantity of decrease stages and the postponement of each stage. Ravi et al. [4] This paper has proposed an area reduced full adder which is the key component in the design. It uses much less number of gates than the standard design and hence lesser area and delay. Both [3] and [4] allow a static trade off between delay and area. Darjn Esposito et. al [5] has implemented a technique for the design of Mac unit. The partial product terms of MAC unit are compressed by using simple OR gates. A remuneration term is presented in the proposed MAC, to minimize the conventional estimate blunder. Our proposed multiplier disables part of the combinational logic in the ripple carry adder (RCA) to obtain smaller area.

III. DADDA Multiplier

A standard multiplier consists of three parts: (i) partial product generation the usage of an AND gate; (ii) partial product reduction using an adder tree; and (iii) addition to produce the final result using ripple carry adder (RCA). Delay and circuit complexity are dominated by the PPR and the multiplier's critical path is influenced by way of the propagated carry chain in the RCA.

This section is prepared as follows. Section III-A provides how the partial product layer is simplified by the tree compressor. Section III-B introduces the ripple carry adder. Finally, Section III-C presents the general structure of the dadda multiplier, which uses the proposed adder and tree compressor.

A. Tree Compressor

Fig.1. shows compressor. Compressor configuration is proposed for two principle reasons, to enlarge the execution of the multiplier and to reduce the blunder rate. The carry and C_{out} outputs have the same weight, therefore the conditions for the carry and C_{out} in the past design can be traded.

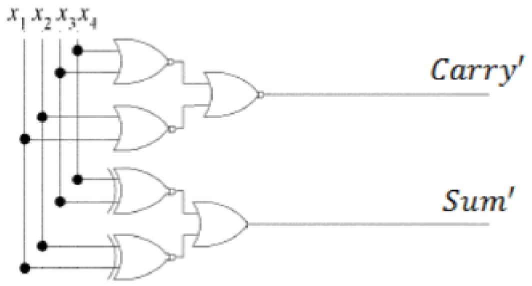

Fig.1.Approximate tree compressor

The value of sum can be obtained using (x1 XOR x2) OR (x3 XOR x4) and value of carry can be generated using (x1 NOR x2) NOR (x3 NOR x4).

Table 1: Truth Table for Compressor

INPUT				OUTPUT	
x1	x2	x3	x4	S	C
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	1

B. Ripple Carry Adder

Ripple carry adder is a digital circuit that produces the computation sum of two binary numbers. It can be formulated with full adders related in cascaded form. The carry out from each full adder linked to the carry input of the next full adder in the chain. The interrelationship of 4 full adder (FA) circuit is to provide a 4-bit ripple carry adder. The input is from the right side because the first cell consistently represents the least significant bit (LSB).

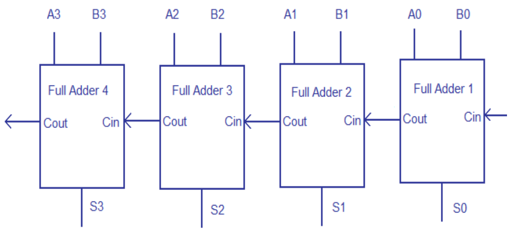

Fig.2. Ripple Carry Adder

In the ripple carry adder, the output is known after the carry executed by the previous stage is produced. They, the entirety of the most critical bit is just accessible after the carry signal has rippled through the adder from the least significant stage to the most considerable stage. As a result, the last sum and carry bits will be authentic after a noticeable delay.

C. Overall structure

A 8×8 Dadda tree multiplier is considered to assess the impact of using the presented compressors in approximate multipliers. The multiplier utilizes in the initial segment AND gates to generate all partial products. In the second part, the approximate tree compressors proposed in the previous section are utilized in the CMA tree to reduce the partial products. The last part is an exact RCA to compute the final binary result. Fig. demonstrates the reduce hardware of an accurate multiplier for n=8.In this fig. the minimization part employments half-adders, full-adders and 4-2 compressors; every fractional item bit is referred as dot. In this principle arrangement of 2 half-adders, 2 full-adders and 8 compressors are utilized to decreases the partial products into at most 4 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 phases of reduction and 3 half-adders, 3 full-adders and 18 compressors are required in the reduction hardware of an 8×8 Dadda multiplier.

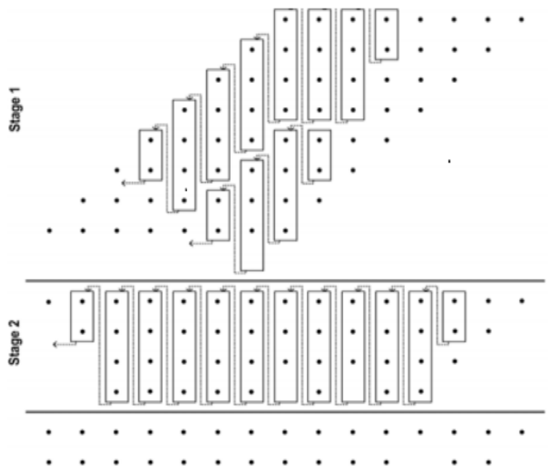

Fig.3. Dadda multiplier

IV. Experimental Results

A. Experimental Setup

n this section, the proposed multiplier is evaluated in terms of critical path delay, design area, and computational accuracy. To make clear the ability of the approximate multiplier to save power, shorten critical path delay and control the accuracy, a traditional Dadda multiplier and Wallace tree multiplier were implemented for comparison.

Wallace tree multiplier as well as the Dadda multiplier, were eight bits and coded using VHDL. The functional verification used to be finished through using ModelSim 5.7g. Delay and area was obtained using Xilinx software.

B. Critical path delay, Level of logic and Number of paths

Comparisons of number of paths, level of logic and critical path delay for Dadda multiplier and Wallace tree two multiplier are shown in Fig.4., Fig.5. and Fig.6. respectively.

From fig.4. the number of paths for Dadda multiplier has reduced when compared to the Wallace tree multiplier. Fig.5. represents the level of logic in which the proposed method has reduced number of logic when compared to the Wallace tree multilplier. Fig.6. shows the comparison results of Delay for Dadda and Wallace tree multiplier. It is referred to that Dadda multiplier has reduced delay of 18.573ns when compared to the Wallace tree multiplier

Table 2: Comparison of Wallace tree multiplier and dadda multiplier

Multipliers	**Number of paths**	**Level of logic**	**Delay**
Wallace tree multiplier (Existing method)	4610	18	29.694ns
Dadda multiplier (Proposed method)	2644	14	18.573ns

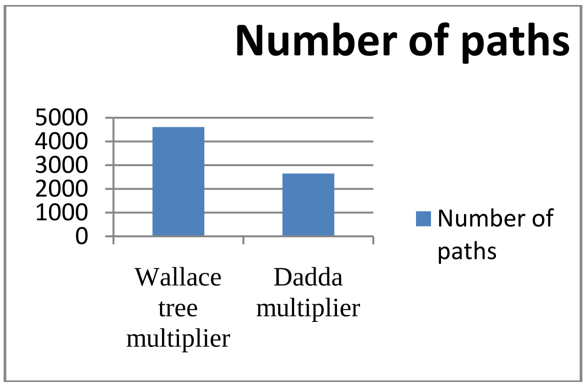

Fig..4. Analysis of number of paths

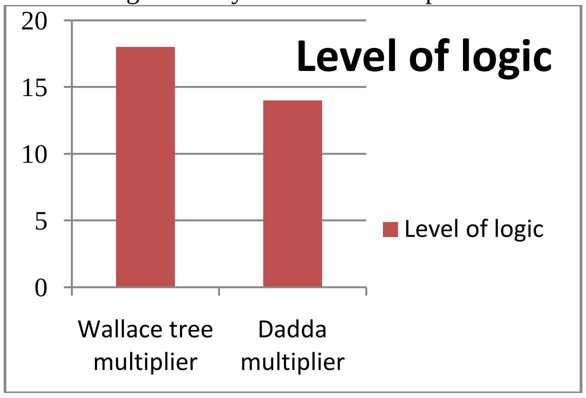

Fig.5. Analysis of level of logic

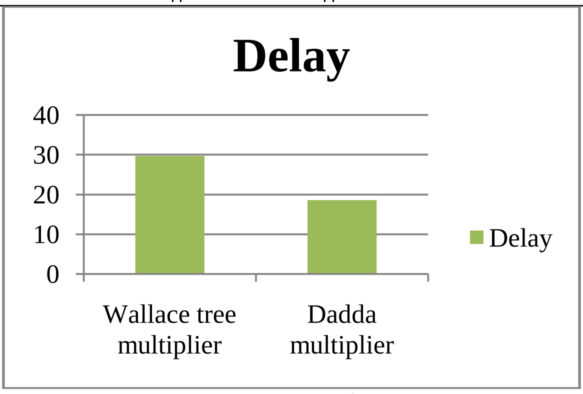

Fig.6. Analysis of delay

C. MAC Application

Multiply-Accumulate (MAC) unit is developed for various high performance applications. MAC unit is a fundamental block in the computing devices, especially in Digital Signal Processor (DSP). MAC unit model is designed by using incorporating the Dadda multiplier and hybrid adder. The performance evaluation of MAC unit models is done by designing the models in VHDL. Then, MAC unit models are simulated and synthesized in Xilinx ISE technology. The MAC unit performs multiplication and accumulation forms more than once in order to function continuous and complex operations in digital signal processing. Multiplier is the fundamental component in the MAC unit. Its most important goal is to expand the speed which in turn reduces the delay and consumes much less power..

Fundamental MAC unit consists of multiplier, adder, and accumulator. A standard n-bit MAC unit consists of an n-bit multiplier, 2n-bit adder, and 2n-bit accumulator. Capacity of addition and multiplication is performed by the MAC unit. MAC operates in two stages. Firstly, multiplier computes the given number output and the end result is forwarded to second stage i.e. addition/accumulation operates.

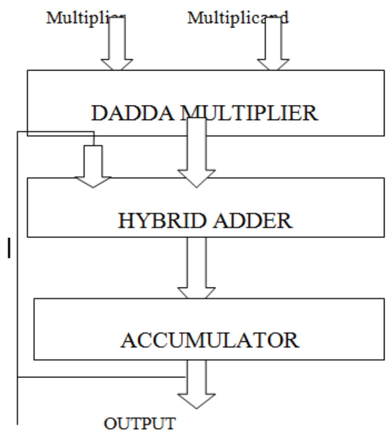

Fig.7. Block diagram of MAC unit

Speed of multiplier is essential in MAC unit which specifies critical path delay as nicely as area which is a great significance in designing of MAC unit. MAC unit is high in demand in Digital signal processing to provide the fundamental hardware for the systems. The functionality of the MAC unit enables high-speed filtering and other processing especially for DSP applications. The MAC using Dadda multiplier and Ripple carry adder is having better performance in terms of area and delay.

D. Hybrid adder

Ladner – Fischer adder supports parallelism, the requirement of majority gates is quite high. Therefore it is necessary for reducing the area. The ripple carry adders are simple and have low area requirement. This fact is taken advantage of in our design of a hybrid adder. A 16-bit hybrid adder requires 86 majority gates and sixteen inverters. In Fig.8, some of the carries computed in ripple carry fashion are shown enclosed in ellipses. For example, C5 and C6 in Stage 3 are computed in ripple carry style with one majority gate (delay) distinction between C5 and C6. However, C7 and C8 are computed in prefix style and in parallel with C4. Similarly, in Stage 4, C9 and C10 are computed in ripple carry style but C10 is computed in parallel with C11and 12(the latter are obtained in prefix style).The end result on majority gates for a 16-bit hybrid adder can be compared with that of the 16-bit Ladner–Fischer adder. In particular, 45 majority gates are decreased right here which is approximately reduction by 35%.

The latency reduction for 8-bit Hybrid adder in comparison to an 8-bit Ladner–Fischer adder. However, in the latter, one clock region delay is incurred in obtaining g0 and p0 (they can be got in parallel, for this reason just one region delay) whereas in the hybrid adder, g0's and p0's are not required at all thereby a reduction of 0.25 units (corresponding to one majority gate) of delay is possible.

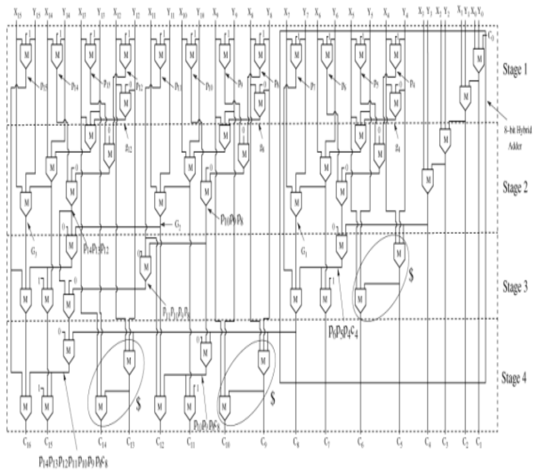

Fig.8. Hybrid Adder

It is cited that besides g0, computation of various different g0's is skipped in the case of a hybrid adder. These include g1, g2, g3, g5, g6, and g7 for an 8-bit hybrid adder. This leads to a substantial reduction in majority logic. However, the reduction is delay is only 0.25 units (due to the computation of g's happening in the "same" stage). Further reduction in delay (in a hybrid adder) is dependent on the clock region for wires.

Table 3**:** Comparison of MAC

Para meters	**Cell count**	**Available**	**Utilization**
Number of slices	112	960	11%
Number of flip flops	97	1920	5%
Number of input LUT	202	1920	10%
Numbers of IOs	35	-	-
Number of bonded IOB	35	108	32%
Number of GCLKs	1	24	4%

V. Conclusion

The core elements of all digital signal processors are the digital multipliers and adders. The speed of the multipliers and adders mostly determines the speed of the digital signal processors. The commonly used operation in various Digital signal processing applications is the MAC unit. MAC unit was designed efficiently using Dadda multiplier and hybrid adder(Ripple carry adder and Ladner fischer adder). The codingwas carried out by using VHDL and simulated the usage of Modelsim5.7g. The MAC architecture with appreciate to area and delay were generated using Xilinx software. The existing system was once implemented by using Wallace tree multiplier which has greater delay whereas the proposed system is implemented by using dadda multiplier which has reduced delay upto 18.573.

In future, the further development can be performed by using implementing the MAC unit in hardware for power reduction.

References

[1]. Tongxin Yang, Tomoaki Ukezono, Toshinori Sato." A Low Power High-Speed Accuracy-Controllable Approximate Multiplier Design" IEEE / ACM 2018.

[2]. C. Liu, J. Han, and F. Lombardi, "A Low-Power, High-Performance approximate multiplier with configurable partial error recovery," Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar.2014.

[3]. A. Momeni, J. Han, P. Montuschi, and F. Lombardi,"Design and analysis of approximate compressors for multiplication," IEEE Transactions on Computers, vol. 64, no. 4, pp. 984-994, Apr. 2015.

[4]. S. Ravi, Govind Shaji Nair, Rajeev Narayan and Harish M kittur,"Low Power and Efficient Dadda Multiplier"on Research Journal of Applied Science, Jan. 2015.

[5]. Darjn Esposito, Antonio G. M. Strollo, Massimo Alioto, "Low-Power Approximate MAC Unit"on Digital Circuits and Sub-Systems, 2017.

[6]. Priyanka. M, Balamurugan. V, "Design and performance analysis of a High speed MAC using different multipliers", on Fifth International Conference on Advances in Computing and Communications 2015.

[7]. Katsuhiko Wakasugi, "A-Low power High speed Accuracy-Controllable Multiplier Design" , IEEE Transaction on VLSI Design and Education Cente(vDEC) @2018IEEE.

[8]. Jayakodi and Kamalanathan. M, "Design and Analysis of Compressor based Dadda tree Multiplication", Second International Conference on Electrical, Information and Communication Technology 2016.

[9]. Maroju sasikumar, Ashok kumar.D, Samundiswary.P "Design and Performance Analysis of Multiply-Accumulate (MAC) Unit",on International Conference on Circuit, Power and Computing Technologies 2014.

[10]. Amir Momeni, JieHan, Paolo Montuschi, and Fabrizio Lombardi, "Design and Analysis of Approximate Compressors for Multiplication",IEEE Transaction on Computing Technologies 2015