An Optimised Add Multiply Operator Using S-MB Technique

Sachin Kumara M M
Department of Electronics and Communication
Don Bosco Institute of Technology
Bangalore, India

Mrs. Sindhuja R
Department of Electronics and Communication
Don Bosco Institute of Technology
Bangalore, India

***Abstract*— Optimizing the fused add multiply (FAM) operator increase the performance of Digital Signal Processing (DSP) Applications. This paper proposes the design and development of FAM operator and methods to develop the straight recoding of the sum of the two numbers in Modified Booth (S-MB) form. This proposed technique is evaluated with the existing methods. This method results in reduction of area of the device, power used.**

***Keywords*— Add-Multiply operator, Modified Booth recoding.**

I. INTRODUCTION

Optimization plays important role in communication media. With the help of this one can focus on increasing the performance of the device and also can reduce the system complexity and area used.

The evolution of the electronic devices simplifies the human life with less effort and hard work, and this evolution plays important role in the modern era. For example splitting the total work to a number of n processors will take less time than assigning total work to a single processor. Splitting the processors for each work results in high performance as well as reduction in area and power consumption.

Earlier in the field of arithmetic optimization many components were used to achieve the output. These methods leading to more utilization of resources with lot of efforts. Best example to explain this is the simple addition and subtraction operation with two methods. If A and B are inputs. In the first method separate processors are used for the addition and subtraction unit. If user wants to add inputs A and B will be given to Adder unit and the subtract unit will be unused. Similarly if the user wants to perform subtraction the inputs A and B will given to subtraction unit while the Adder unit remain unused. So in this method one unit will be remaining unused every time performing arithmetic operation. This leads to wastage of resources, increase devise area and also power consumption of the devise. Evolution of the arithmetic optimization overtook these drawbacks. In the second method optimization plays key role. Here instead of using two processors for separate addition and subtraction only one processor is used which containing both addition and subtraction units. By using select key we can perform either addition or subtraction by driving the inputs A and B. This method reduces the area used by eliminating one processor and also reduces the area, power compared to the first method.

The proposed method deals with the optimization of the FAM unit by direct recoding of the sum of the two numbers using S-MB method. Radix-4 algorithm is the key in this method to achieve the optimization. Radix-4 algorithm is used to reduce the number of partial products which in-turn increases the speed of the operation. Hence by this approach fusion of the FAM unit is done. This method is used to reduce the area, power and to increase the performance.

The objective of the proposed paper is summarized as follows

- Reducing the design of the Fused-Add Multiply (FAM) operator in terms of area.
- Straight recoding of the result of two numbers to its MB form

II. LITRATURE SURVEY

In [1], the main aim is to reduce the area consumption and power as well as less memory usage. Here two techniques are used to reduce above mentioned parameters. First one is parallel approach and second one is serial approach. In parallel approach, for example if addition and subtraction operations need to be perform, then parallel method will use separate adder and subtraction units. As using two arithmetic operation units will increase the area of the devise and consume more power also. To overcome this disadvantage serial approach is used. In this serial approach both adder and subtraction operation units are merged in to a single unit and with the help of selective key user can perform either addition or subtraction operation. In this method one of the arithmetic operator unit will be remain unused, if addition operation is operating means the subtraction unit will be unused. Hence this approach reduces the area of devise by reducing the number of arithmetic operators and consumes less power and memory.

In [2], this paper estimates the performance evaluation of the multi arithmetic operations or working of multi processors. The main aim of this paper is to reduce the area consumption and power as well as less memory usage. If a task or work is handled by many processors will automatically increase area, use more power and memory. This paper proposed a technique to overcome from this problem. Here also same two techniques are used to reduce above mentioned parameters, but this contains many arithmetic operations. First one is parallel approach and second one is serial approach. In parallel approach, for example if addition and subtraction and multiplication operations need to be perform, then parallel method will use separate adder, subtraction and multiplication units. As using three arithmetic operation units will increase the area of the devise and consume more power also. To overcome this disadvantage serial approach is used. In this serial approach adder, subtraction and multiplication operation units are merged in to a single unit and with the help of selective key user can perform either addition or subtraction or multiplication operation. In this method one of the arithmetic operator unit will be remain unused, if addition operation is operating means the remaining multiplication and subtraction unit will be unused. Hence this approach reduces the area of devise by reducing the number of arithmetic operators and consumes less power and memory.

In [3], this paper is focuses on increasing the speed of the multiplier and also to perform arithmetic operations with carry free addition. The higher version of radix is very useful in order to reduce number of partial products and also increases the performance of the devise. But the problem with higher radix is it produces hard multiples. To overcome from this problem a redundant binary method is used. This paper focuses on the designing of the redundant binary technique to solve problems while partial product generation and accumulation.

In [4], this paper focuses on recoding of the two numbers using booth recoder. In this different recoding methods are used to reduce the area of the devise and power consumed by the devise. The radix algorithm is used to reduce the number of partial products. Hence with the help of reduced partial products the performance the devise is increased. Different recoding methods are designed based on basic adders. The least path used to get the output will be considered to select the better recoding method.

III. METHODOLOGY

The proposed method focus on addition and multiplication units to design the operation Z=X. (A+B). As shown in Fig.1 inputs A and B are fed to the FAM unit. Another input X is given to the partial product generator. These partial products are driven through Carry Save Adder and Carry look ahead adder to get the final output Z=X. (A+B). The correction term (CT) is used to flip the corrupted bit.

The design of the MB recoder consisting of even and odd part. This design consisting of normal Full adder (FA) and FA* and two types of Half adder HA* and HA** to achieve the optimization. Here different types of adders are used to reduce the number of resources used. In [4] different MB recoders are designed using different adder units. Depending upon the less gates used and the critical path the best MB design will be selected. In the same way the proposed technique is designed with different adder units and this design is more optimized compared to the MB recoders used in [4]. The proposed even and odd MB recoders are shown if Fig.2 and Fig.3 respectively

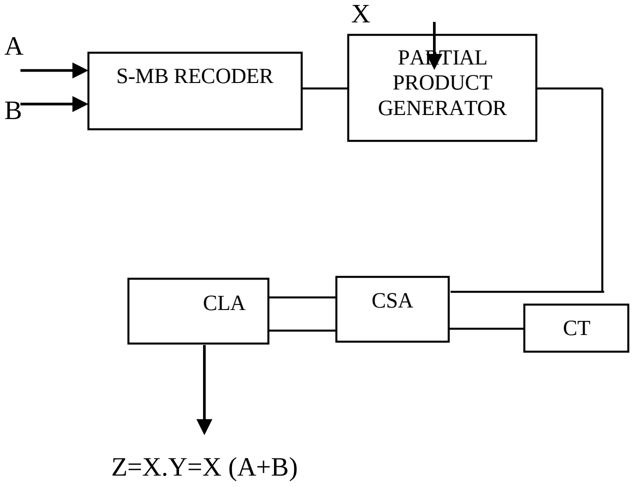


Fig 1: The proposed working block diagram

Captions:

MB: Modified Booth

CT: Correction Term

CSA: Carry save adder

CLA: Carry look ahead adder

Output of HA* = -a-b

Output of HA = -a+b**

Output of FA = -a-b+c**

A. *FAM IMPLEMENTATION*

In the Fused Add-Multiply (FAM) design the term Y is encoded and is multiplied with X. Both X and Y are having n=2k bits and they are in 2’s complement form. In the next level the generation of partial product is required. Each generated partial products now added to correction term (CT) for resolving any bit correction involved. After this the partial products are added through a Carry Save Adder (CSA). Finally this CSA output is fed to the Carry look ahead (CLA) adder to produce the final output Z=X.Y.

B. *MB RECODER*

In the existing system the MB recoder will be having separate adder and encoder units. Using adder unit in the begging will increase the area of the devise which also increases the power consumed by the devise. The proposed system merged these two adder and encoder unit making a single MB recoder unit in order to reduce the area of devise and reduce the power used by the devise.

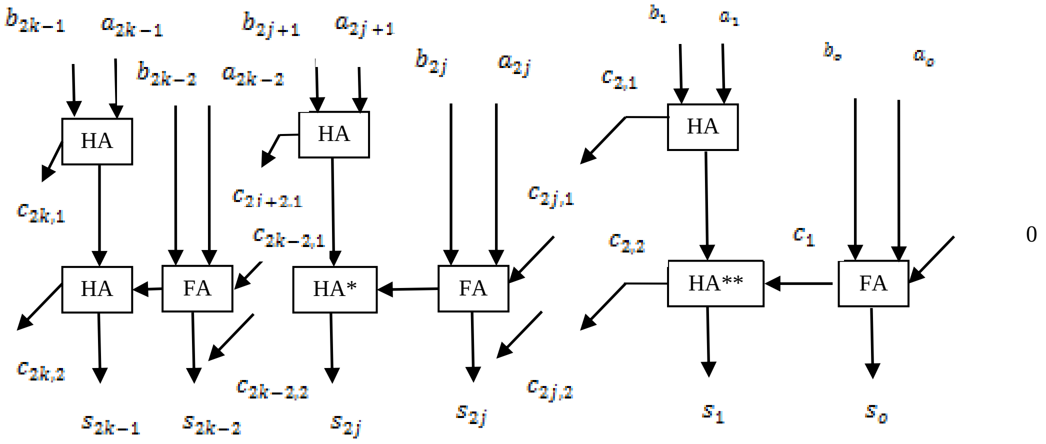


Fig 2: MB Recoder Even Part

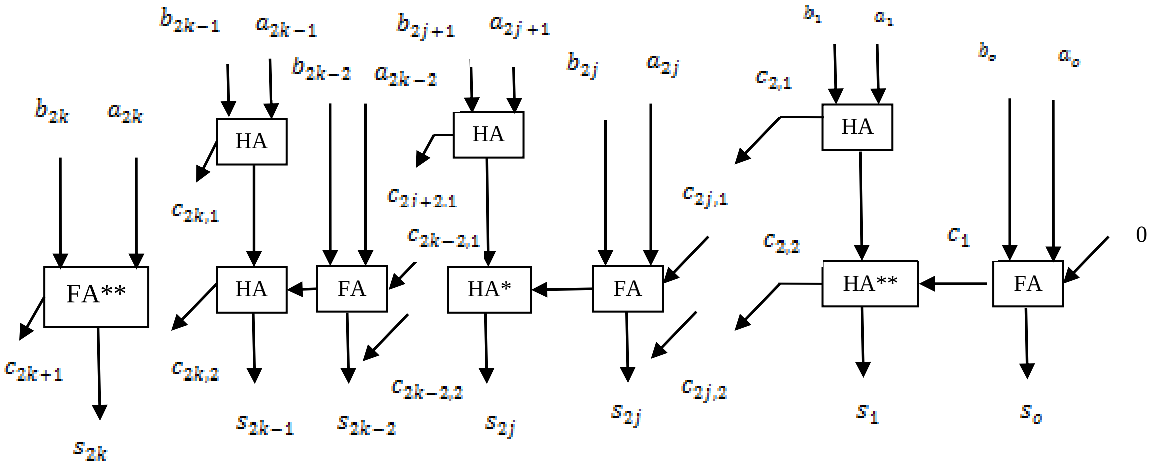


Fig 3: MB Recoder Odd Part

Captions

HA: HALF ADDER

FA: FULL ADDER

a,b: INPUTS

S: OUTPUT

C: CARRY

IV. RESULTS

In this section the proposed recoding scheme is explored. The fused Add-Multiply operator implemented for both even and odd bits. The results are shown below.

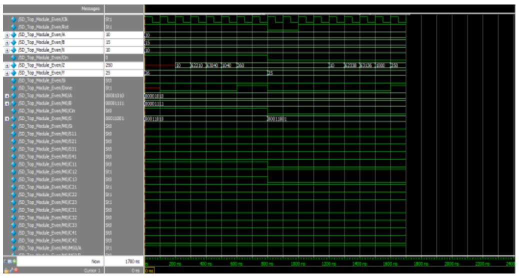


Fig 4: Output of Top module

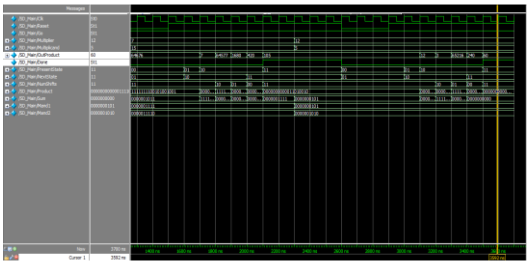


Fig 5: Output of partial product generator

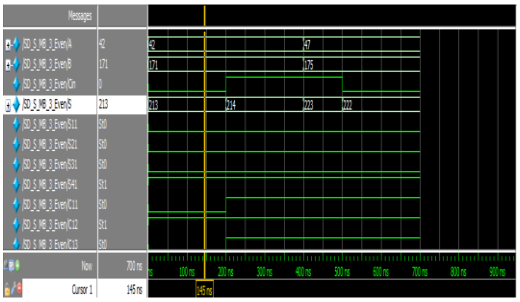


Fig 6: Output of S-MB Even

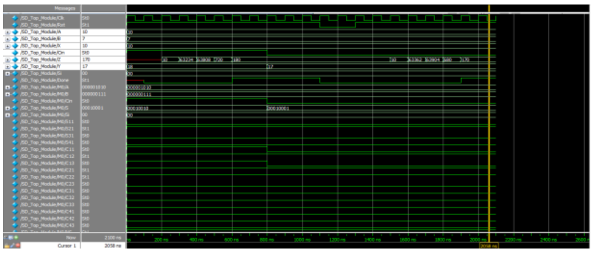
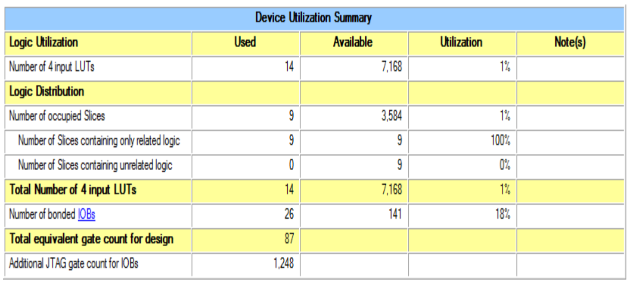
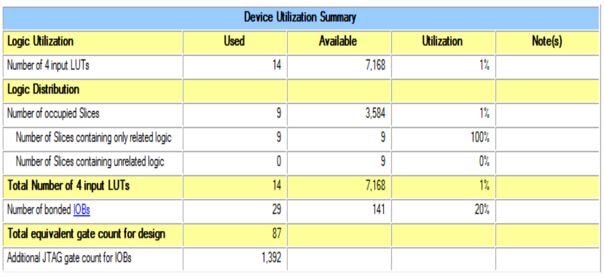


Fig.7 Output of S-MB Odd



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	14	7,168	1%	
Logic Distribution				
Number of occupied Slices	9	3,584	1%	
Number of Slices containing only related logic	9	9	100%	
Number of Slices containing unrelated logic	0	9	0%	
Total Number of 4 input LUTs	14	7,168	1%	
Number of bonded IOBs	26	141	18%	
Total equivalent gate count for design	87			
Additional JTAG gate count for IOBs	1,248			

Fig 8: Device Utilization Summary of Even Part



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	14	7,168	1%	
Logic Distribution				
Number of occupied Slices	9	3,584	1%	
Number of Slices containing only related logic	9	9	100%	
Number of Slices containing unrelated logic	0	9	0%	
Total Number of 4 input LUTs	14	7,168	1%	
Number of bonded IOBs	29	141	20%	
Total equivalent gate count for design	87			
Additional JTAG gate count for IOBs	1,392			

Fig 9: Device Utilization Summary of Odd Part

REFERENCES

[1] A. Amaricai, M. Vladutiu, and O. Boncalo, “Design issues and implementations for floating-point divide-add fused,” IEEE Trans. Circuits Syst. II–Exp. Briefs, vol. 57, no. 4, pp. 295–299, Apr. 2010.

[2] E. E. Swartzlander and H. H. M. Saleh, “FFT implementation with fused floating-point operations,” IEEE Trans. Comput., vol. 61, no. 2, pp. 284–288, Feb. 2012

[3] C. N. Lyu and D. W. Matula, “Constructing a low power multiplier using Modified Booth Encoding Algorithm in redundant binary number system” 1995, pp. 50–57.

[4] Kostas Tsoumanis and Kiamal Pekmestzi, “An Optimized Modified Booth Recoder for Efficient”, APRIL 2014
Design of the Add-Multiply Operator

[5] S. Nikolaidis, E. Karaolis, and E. D. Kyriakis-Bitzaros, “Estimation of signal transition activity in FIR filters implemented by a MAC architecture,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 19, no. 1, pp. 164–169, Jan. 2000.

[6] O. Kwon, K. Nowka, and E. E. Swartzlander, “A 16-bit by 16-bitMAC design using fast 5: 3 compressor cells,” J. VLSI Signal Process. Syst.,vol. 31, no. 2, pp. 77–89, Jun. 2002