# Redundant Binary Partial Product for Fused add Multiplier to Optimize the Power

**Dr.C.Venkatesh, S.Poonkuzhali, M.Moortheeswari**

*Abstract*— **Many Digital Signal Processing (DSP) applications carry out a large number of complex arithmetic operations. Multiplier takes a important role in the performance of the system, thus by optimizing the multiplier the power and area can be reduced. This paper focus on optimizing the power using redundant binary partial product. This implements a new technique called RB encoding, in which the sum of two numbers and there product is multiplied with the multiplier in Modified Booth (MB) form. The RB encoding is used in multipliers with operand, without increasing the delay of partial product accumulation. It is used for both signed and unsigned Radix-4, which is a parallel multiplier. its An efficient multiplier which reduces partial product by N/2, where N is the number of multiplicand. The proposed FAM unit with RB encoding is coded in Verilog, simulated and synthesized using Xilinx ISE tool. The performance of FAM unit with RB encoding is compared with other existing technique in terms of power consumption and critical path. The proposed FAM unit with RB encoding yields considerable reduction in terms of critical delay and power consumption.**

***Index Terms*— Digital Signal Processing (DSP), Multiply-Accumulator (MAC), Add-Multiply (AM) operation, Modified Booth (MB) algorithm, Redundant Binary (RB), Fused Add Multiplier (FAM), Non binary (NB)**

## I. INTRODUCTION

Multiplier plays an important part in digital signal processing (DSP) system. It is used in implementation of recursive, transverse filters and discrete Fourier transforms. The performance of DSP systems is inherently affected by decision on their design regarding the allocation and the architecture of arithmetic units. Using of the large computation leads to increases in power and area of the system.

In [1], a two-stage recorder which converts a number in carry save form to its MB representation. The first stage transforms the carry-save form of the input number into signed-digit form which is then recoded in the second stage and it matches the form that the MB digits request. Recently, this technique had been used for the design of high performance flexible coprocessor architecture targeting the computationally intensive DSP applications. In [2], the recoding of a redundant input form its carry-save form to the corresponding borrow-save form keeping the critical path of the multiplication operation fixed. The multiplication operation product is obtained by adding partial products presented in [3], thus the final speed of the multiplier circuits depends on the speed of the adder circuits and the number of partial products generated. Radix-8 booth encoded technique used then there are only 3 partial products and only one CSA and CLA is required to produce the final product. Through this algorithm it increases in power of the system.

Based on the observation that an addition can often be subsequent to a multiplication (e.g., in symmetric FIR filters), the Multiply-Accumulator (MAC) and Multiply- Add (MAD) units were introduced [4] leading to more efficient implementations of DSP algorithms compared to the conventional ones, which use only primitive resources [5]. Several architectures have been proposed to optimize the performance of the MAC operation in terms of area occupation, critical path delay or power consumption [6], [7].

In [8], MAC components increase the flexibility of DSP data path synthesis as a large set of arithmetic operations can be efficiently mapped on the system. The fused operations are a two-term dot product and add-subtract unit. The FFT processors [9] use “butterfly” operations that consist of multiplications, additions, and subtractions of complex valued data. Both radix-2 and radix-4 butterflies are implemented efficiently with the two fused floating-point operations.

Although the direct recording of the mutiplier in its MB form leads to a more efficient implementation of the fused add-multiply (FAM) unit with RB encoding compared to the conventional one. The existing recoding schemes are based on complex manipulations in bit-level, which are implemented by dedicated circuits in gate-level. This paper focuses on the efficient design of FAM unit with RB encoding and targeting the optimization of the recoding scheme for direct shaping of the MB form of multipliers. The adoption of the proposed recoding technique delivers optimized solutions for the FAM design using RB encoding enabling the targeted operator to be timing functional (no timing violations) for a larger range of frequencies. Also, under the same timing constraints, the proposed designs deliver improvements in both area and power consumption, thus outperforming the existing Modified Booth Scheme recoding solutions.

## II. EXISTING METHOD

### A. FAM Architecture

The direct recoding of the sum of two numbers in its MB form produce more efficient implementation of the FAM unit compared to the conventional one. The existing recoding schemes are based on complex manipulations in bit-level, which are implemented by circuits in gate-level. This paper

**Dr.C.Venkatesh** , Associate professor, Department of Electronics and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore-641202, Tamil Nadu, India.

**S.Poonkuzhali,** PG Student, Department of Electronics and Communication Engineering,Sri Eshwar College of Engineering, Coimbatore-641202, Tamil Nadu, India.

**M.Moortheeswari**, GET, Caliber Embedded Technologies

focuses on the efficient design of FAM operators, targeting the optimization of the recoding scheme for direct shaping of the MB form of the sum of two numbers. More specifically, this paper proposes a new recoding technique which decreases the critical path delay, reduces area and power consumption. The proposed Modified Booth Recoding Scheme algorithm is structured, simple and can be easily modified in order to be applied either in signed or unsigned numbers, which comprise of odd or even number of bits. In this technique, explore an alternative schemes of the proposed Modified Booth Recording Scheme approach using conventional and signed-bit Full Adders (FAs).

In the FAM design presented in Fig. 1. It shows the multiplier is a parallel one based on the MB algorithm. Consider the product X·Y, where X is the multiplier and Y is the output of adder A and B.

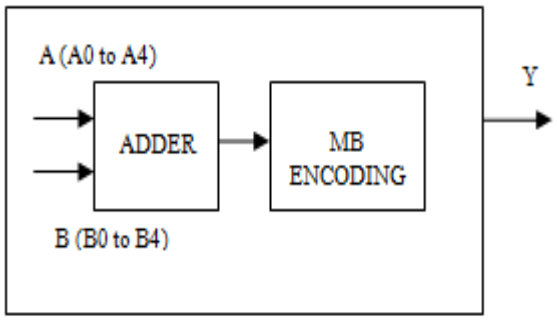


**Fig.1.** FAM UNIT DESIGN

The term Y is expressed in terms of ($y_{n-1}$ $y_{n-2}$ ……. $y_1$ $y_0$) is encoded based on the MB algorithm and X in terms of ($x_{n-1}$ $x_{n-2}$ ….. $x_1$ $x_0$). Both X and Y consist of n=2k bits where the value of k is 4 and are in 2's complement form.

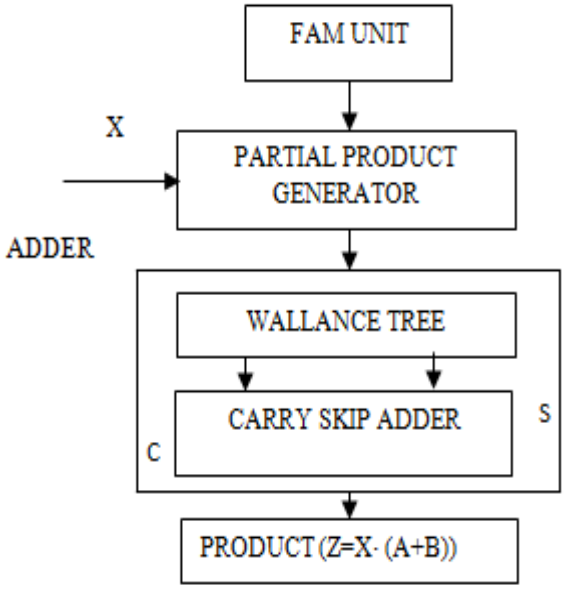


**Fig.2.** FAM ARCHITECTURE

The basic FAM architecture is shown in Fig.2. The generation of k partial products can be written as,

$$PP = X \cdot Y_j^{MB} = \bar{p}\, 2^n + \sum_{i=0}^{n-1} p_{j,i} \cdot 2^i \qquad (1)$$

Where PP is the generation of partial products and digit $Y_j^{MB}$ is correspond to the three consecutive bits, $Y_{2j+1}$, $Y_{2j}$ and $Y_{2j-1}$. Table I shows how they are formed by summarizing the MB encoding technique. Each digit is represented by three bits named S, one and two. The sign bit S shows if the digit is negative (S=1) or positive (S=0). Signal one shows if the absolute value of a digit is equal to 1 (one=1) or not (one=0). Signal two shows if the absolute value of digit is equal to 2 (two=1) or not (two=0). Using these three bits we calculate the MB digits $Y_j^{MB}$ by the following relation:

$$Y_j^{MB} = (-1)^{sj} \cdot [\, one_j + two_j] \qquad (2)$$

TABLE – 1
Modified Booth Encoding Table.

| Binary | | | MMB Encoding | | | carry |
|---|---|---|---|---|---|---|
| $Y_{2j+1}$ | $Y_{2j}$ | $Y_{2j-1}$ | Sign= sj | one j | two j | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

After the partial products are generated, they are added through a Wallance Carry Save Adder (CSA) tree along with the correction term. Finally the carry save output of the Wallance CSA tree is leaded to a fast Carry skip adder to form the final result Z = X·Y. The drawback of using CSA adder is to increase the area occupation and power dissipation. In order to reduce this drawback, RB encoding algorithm has been proposed in which partial product generator, wallence tree, carry skip adder is replaced by redundant binary partial product generator is shown in the Fig.3. The use of RB encoding will not only reduce area, but will also reduce power.

## III. PROPOSED METHOD

A multiplier essentially consists of two operands, a multiplicand 'Y' and a multiplier 'X', and produces a product 'P'. In a conventional multiplier, a number of partial products are formed first by multiplying the multiplicand with each bit of the multiplier. These partial products are then added together to generate the product 'P'. In short, we can break down multiplication into two parts, namely partial product generation and partial product accumulation. Speeding up multiplication therefore must aim (i) speeding up partial product generation (PPG), (ii) reduce the number of partial products, (iii) speeding up partial product summation .

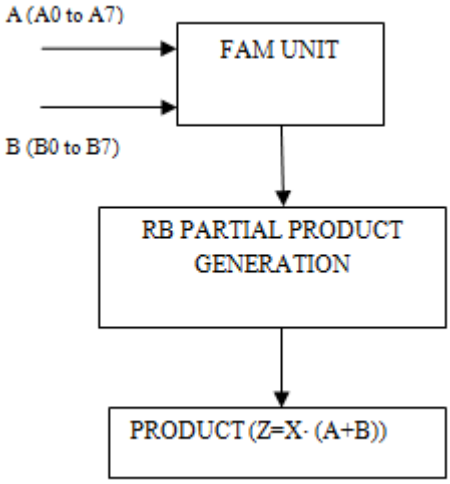


**Fig. 3**- Block diagram of RB encoding

The proposed partial product generator generates RB partial products, without any carry propagation delay. For a multiplicand 'y' the radix-4 booth encoder will have five NB different partial products {-2y,-y,0,y,2y}. Instead of generating -2y and –y in two's complement form the multiplier retains the partial products in their one's complement form and introduces an extra bit '1' along with the partial products. The NB partial product –y obtained from booth encoder is expressed as $(\overline{y}, 1)$, where $\overline{y}$ is the one's complement of y. the set of partial product obtained from booth encoder is represented as {( $\overline{2y}, 1$) , $(\overline{y}, 1), (0.,0), (y, 0), (2y, 0)$}.

A NB partial product X can be represented as,

$X = (X^* + x)$ (3)

Where $X^* = \overline{X}$ and x=1, when X is negative and X* =X and x=0, when X is positive.

The sum of two NB partial product X and Y can be expressed as

$X + Y = (X^*,Y^*) -1+x+y$ (4)

For different positive and negative number X and Y, the values of x and y will be chosen according to Table-2. It can be observed that x and y are nothing but the sign bits of X and Y respectively.

If Z= x + y- 1, equation (4) can be modified as,

$X + Y = (X^*,Y^*) + Z$ (5)

Where Z can be coded according to Table 2.

TABLE – 2
Encoding for NB partial products

| X | Y | x | y | Z |
|---|---|---|---|---|
| + | + | 0 | 0 | -1 |
| + | – | 0 | 1 | 0 |
| – | + | 1 | 0 | 0 |
| – | – | 1 | 1 | 1 |

The extra RB digit from each RB operand form an extra operand, which can be fed into the next partial product accumulation stage. This correction word will be having the format …0Z000Z000Z, where Z = {1, 0, -1}. The addition of two NB partial products A = -10 and B = -20 according to Table II encoding is shown in Fig.1. The two partial products are grouped along with the extra bit. In this case both numbers are expressed in their one's complement representations. The extra bits are '1' for both, and are shown separately in Fig. 1. The bit pair (A,B) is also shown in Fig 1. These bit pairs represent the sum A+B in RB notation. The equivalent RB number can be obtained using Encoding 2 in Table I and is shown at the bottom. The extra bit position is also assigned unit weight. The RB result obtained can be reconverted into its equivalent decimal value using a negative weight for the MSB bit. This results in the final sum of –30.

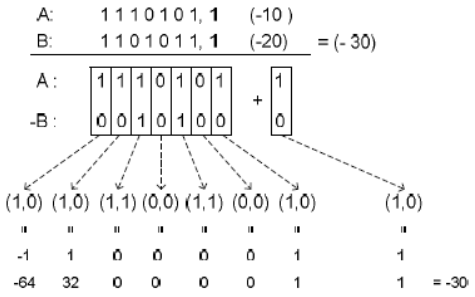


**Fig. 4** RB encoding using one's complement airthematic

The RB encoding method avoid carry propogation operation during partial product generation, and simply express the partial products in one's complement NB fromat for a negative number. The extra bit for each NB partial product is same as thesign bit of each operand. The correction bit Z is found directly from the grouping, instead of a combination of RB and booth recoding terms. Thus the final result of Z=(X.Y) is obtained from RB partial product generator.

## IV. Power Analysis

The power of FAM unit using Carry skip adder and FAM unit with RB encoding are compared and tabulated in table 4. It also shows that the proposed fused add multiple unit with RB encoding algorithm consumes less power than the existing algorithm like modified booth recoder, basic recoding algorithm etc. The Fig. 5 shows a bar chart with power in y-axis and various techniques in x- axis. From the fig.5 it is found that the FAM with RB encoding posses the least power of 0.032W, while the basic recoding algorithm posses the maximum of 0.122W.

**Table 3 - Power analysis**

| **Various technique** | **Power (W)** |
|---|---|
| FAM+RB encoding | 0.032 |
| FAM+CSA adder | 0.034 |
| FAM+CLA adder | 0.038 |

| | |
|---|---|
| MB ALGORITHM | 0.109 |
| BASIC RECORDING ALGORITHM | 0.122 |

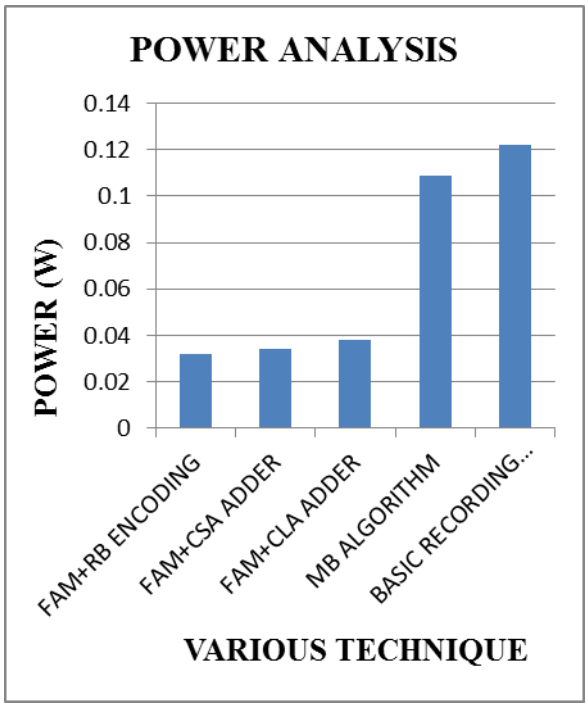


**Fig 5.** Power Vs Various technique

The Fused add multiply unit with RB encoding posseses lower power compared to the fused add multiply unit with carry skip adder. The RB encoding uses the advantage of reducing the critical path in the circuit and operates faster with lower delay overheads.

## V. Simulation Results

The proposed Fused add unit is implemented in the Verilog, simulated with Xilinx and modelsim. Testing is carried out using various inputs. This is an used to reduced the computation in the circuits. The simulation results of fused add multiply unit with carry look ahead adder is shown in Fig.6 FAM Unit with CSA adder for even bit and Fig.7 FAM unit with CSA adder for odd bit. An two 8-bit input A and B is given to the FAM unit. Then produced output 8-bit from FAM unit is multiplied with the 8-Bit multiplicand X and partial products are generated. Then the generated 16-bit Z output is taken from CSA adder .

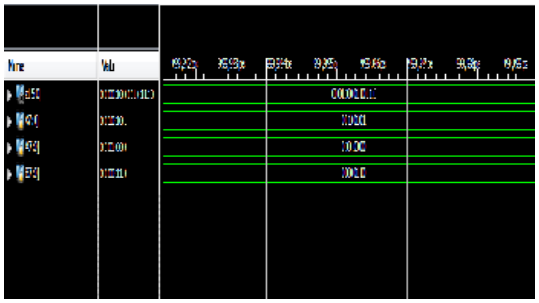

**Fig. 6** FAM unit with CSA adder for even bit

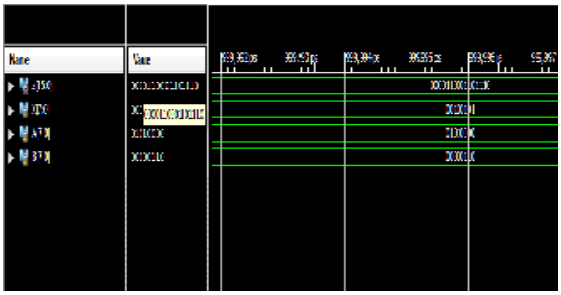

**Fig.7** FAM unit with CSA adder for odd bit

The simulation result of FAM with RB encoding is shown in Fig. 8 FAM unit with CSA adder for even bit and Fig 9 FAM unit with RB encoding for odd bit. Similar to the first architecture two 8-bits inputs of A and B is given to the FAM unit. Then produced output 8-bit from FAM unit is multiplied with the 8-Bit multiplicand X and partial products are generated. Then the generated 16-bit Z output is taken from RB partial product generator.

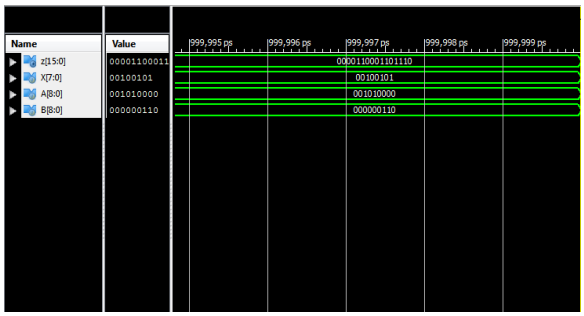

**Fig.8** FAM unit with RB encoding for even bit

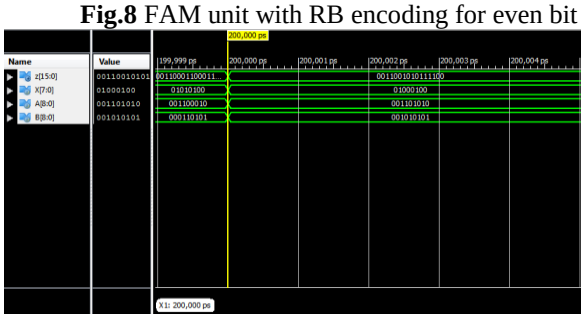

**Fig.9** FAM unit with RB encoding for odd bit

The power consumption of the two algorithm is analysed in Xilinx using XP power analyser. The values of power in case of FAM with CSA is same for both even and odd bit shown in Fig. 10 and 11 of FAM unit with CSA adder.

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Logic | 0.000 | 8 | 1920 | 0 |
| Signals | 0.000 | 52 | --- | --- |
| MULTs | 0.000 | 1 | 4 | 25 |
| IOs | 0.000 | 40 | 66 | 61 |
| Leakage | 0.034 | | | |
| Total | 0.034 | | | |

**Fig. 10** Power of FAM with CSA adder

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Logic | 0.000 | 9 | 1920 | 0 |
| Signals | 0.000 | 55 | --- | --- |
| MULTs | 0.000 | 1 | 4 | 25 |
| IOs | 0.000 | 42 | 66 | 64 |
| Leakage | 0.032 | | | |
| Total | 0.032 | | | |

**Fig.11** Power of FAM unit with RB encoding

From the fig.11 provides the power value in case of FAM unit with RB encoding algorithm. From the simulation results and power analysis it is found that the FAM with RB encoding consumes 0.032 W which is 0.002 W less than that of the FAM with CSA adder.

## VI. CONCLUSION

The proposed fused add multiply algorithm is a efficient airthematic algorithm in which the delay and complexity of the system is reduced. The proposed algorithm uses fused add multiply unit with RB encoding which possess binary addition and multiplication is simpler by generating the partial product. The modified booth algorithm use large area for airthematic process and computation time is longer whereas the fused add multiply algorithm generate the partial products with less computation time. A RB encoding is used in fused add multiply algorithm to increase the speed of multiplication process and reduce in area.

The fused add multiply unit with RB encoding consumes less power than the other existing codes. The major component of FAM is multi-bit adder whose design will have a significant impact on the overall performance of the FAM system. The FAM with CSA adder uses 0.034W POWER while the FAM with RB encoding adder uses only 0.032W of power. From the proposed models, it is found that the FAM with RB encoding consumes power which is 0.002W less than the one with CSA adder.

## REFERENCES

[1]Kostas Tsoumanis, Sotiris Xydis and Kaimal Pekmestzi "An Optimized Modified Booth Recoder for efficient design of the Add- Multiply Operator, " IEEE Trans. Vol. 61,No. 4, April 2014.

[2]Paladugu Srinivas Teja " Design of radix-8 Booth Multiplier Using Koggestone Adder For High Speed Airihmetic Applications " EEIEJ, Vol.1, No. 1, February 2014.

[3]Minu Thomas "Design and Simulation of Radix-8 Booth Encoder Multiplier for Signed and Unsigned Numbers, " IJSRD, vol.1, Issue 4, 2013.

[4]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.

[5]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.

[6]Y.-H. Seo and D.-W. Kim, "A new VLSI architecture of parallel multiplier–accumulator based on Radix-2 modified Booth algorithm," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 2, pp. 201–208, Feb. 2010.

[7]N. H. E. Weste and D. M. Harris, "Datapath subsystems," in CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Readington: Addison-Wesley, 2010, ch. 11.

[8]Z. Huang andM. D. Ercegovac, "High-performance low-power left-toright array multiplier design," IEEE Trans. Comput., vol. 54, no. 3, pp.272–283, Mar. 2005.

[9]R. Zimmermann and D. Q. Tran, "Optimized synthesis of sum-of-products," in Proc. Asilomar Conf. Signals, Syst. Comput., Pacific Grove, Washington, DC, 2003, pp. 867–872.