International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Sum to Modified Booth Recoding Techniques For Efficient Design of the Fused Add-Multiply Operator

D.S. Vanaja[1], S. Sandeep[2]

[1]M. Tech scholar in VLSI System Design, Department of ECE, Sri VenkatesaPerumal College of Engineering & Technology, Puttur, Chittoor (dt), A.P.

[2]Assistant Professor , Department of ECE, Sri VenkatesaPerumal College of Engineering & Technology, Puttur, Chittoor (dt), A.P.

Abstract: ***Digital Signal Processing (DSP) applications carry out a large number of complex arithmetic operations. Multiplier plays an important role in high performance of the system, reduce the power and area. In this paper we focus on optimizing the design of Fused Add Multiply (FAM) operator for increasing performance.We introduce new techniques to implement the direct recoding of the sum of two numbers in its Modified Booth (MB) form. We propose a structured and efficient recoding technique and explore three different schemes by incorporating them in Fused Add Multiply (FAM) designs. Comparing the proposed FAM designs with existing recoding schemes, the proposed technique gives considerable reductions in terms of critical delay, power consumption and hardware complexity of the FAM unit.***

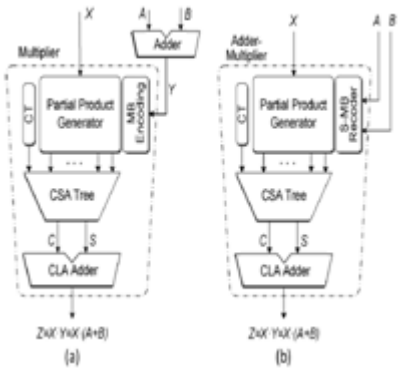
Keywords: Add-Multiply operation, Arithmetic circuits, Modified Booth recoding, VLSI design.

1. 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 decisions on their design regarding the allocation and the architecture of arithmetic units. Using of the large large computation leads to increases in power and area of the system.

Recent research activities in the field of arithmetic optimization have shown that the design of arithmetic components combining operations which share data, can lead to significant performance improvements. 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 leading to more efficient implementations of DSP algorithms compared to the conventional ones, which use only primitive resources. Several architectures have been proposed to optimize the performance of the MAC operation in terms of area occupation, critical path delay or power consumption. Except the MAC/MAD operations, many DSP applications are based on Add-Multiply (AM) operations (e.g., FFT algorithm). The straightforward design of the AM unit, by first allocating an adder and then driving its output to the input of a multiplier, increases significantly both area and critical path delay of the circuit. Targeting an optimized design of AM operators, fusion technique are employed based on the direct recoding of the sum of two numbers (equivalently a number in carry-save representation in its Modified Booth (MB) form. Thus, the carry-propagate (or carry-look-ahead) adder of the conventional AM design is eliminated resulting in considerable gains of performance. Lyu and Matula presented a signed-bitMBrecoder which transforms redundant binary inputs to their MB recoding form. A special expansion of the preprocessing step of the recoder is needed in order to handle operands in carry-save representation. In , the author proposes a two-stage recoder 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 so that it matches the form that the MB digits request. Recently, the technique of has been used for the design of high performance flexible coprocessor architectures targeting the computationally intensive DSP applications.

Although the direct recoding of the sum of two numbers in its MB form leads to a more efficient implementation of the fused Add-Multiply (FAM) unit compared to the conventional one, existing recoding schemes are based on complex manipulations in bit-level, which are implemented by dedicated circuits in gate-level. This work 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 (Sum to MB – S-MB). More specifically, we propose a new encoding technique which decreases the critical path delay and reduces area and power consumption. The proposed S-MB algorithm is structured, simple and can be easily modified in order to be applied either in signed (in 2's complement representation) or unsigned numbers, which comprise of odd or even number of bits. We explore three alternative schemes of the proposed S-MB approach using conventional and signed-bit Full Adders (FAs) and Half Adders (HAs) as building blocks.



2. Modified Booth Multiplier

A. Motivation

In the existing system, we focus on AM units which implement the operation $Z=X.(A+B)$. The conventional design of the AM operator (Fig. 1(a)) requires that its inputs A and B are first driven to an adder and then the input X and the sum $Y=A+B$ are driven to a multiplier in order to get Z. The drawback of using an adder is that it inserts a significant delay in the critical path of the AM. As there are carry signals to be propagated inside the adder, the critical path depends on the bit-width of the inputs. In order to decrease this delay, a Carry-Look- Ahead (CLA) adder can be used which, however, increases the area occupation and power dissipation. An optimized design of the AM operator is based on the fusion of the adder and the MB encoding unit into a single datapath block (Fig. 1(a)) by direct recoding of the sum $Y=A+B$ to its MB representation. The fused Add -Multiply (FAM) component contains only one adder at the end (final adder of the parallel multiplier). As a result, signify-cant area savings are observed and the critical path delay of the recoding process is reduced and decoupled from the bit- width of its inputs. In this work, we present a new technique for direct recoding of two numbers in the MB representation of their sum. The proposed architecture which is shown in below figure.1(c), in this we implement both partial product generation and addition in a single unit for SMB-3 technique to reduce the critical path delay and area occupation compared to existing system.

Figure 1: AM operator based on the (a) conventional design and (b) fused design with direct recoding of the sum of and in its MB representation(c)Proposed architecture for SMB-3 technique.

B. Review of the Modified Booth Form

Modified Booth (MB) is a prevalent form used in multiplication. It is a redundant signed-digit radix-4 en-coding technique. Its main advantage is that it reduces by half the number of partial products in multiplication comparing to any other radix-2 representation. Let us consider the multiplication of 2's complement numbers X and Y with each number consisting of n=2k bits . The multiplicand Y can be represented in MB form as

$$Y=\langle y_{n-1}y_{n-2}\cdots y_1y_0\rangle_{2's}=-y_{2k-1}\cdot 2^{2k-1}+\sum_{i=0}^{2k-2} y_i\cdot 2^i$$

$$=\langle \mathbf{y}_{k-1}^{MB}\mathbf{y}_{k-2}^{MB}\cdots\mathbf{y}_1^{MB}\mathbf{y}_0^{MB}\rangle_{MB}=\sum_{j=0}^{k-1}\mathbf{y}_j^{MB}\cdot 2^{2j} \quad (1)$$

$$\mathbf{y}_j^{MB}=-2y_{2j+1}+y_{2j}+y_{2j-1}. \quad (2)$$

Digits , correspond to the three consecutive bits with one bit overlapped and considering that . Table I shows how they are formed by summarizing the MB encoding technique. Each digit is represented by three bits named , one and two. The sign bit shows if the digit is negative ( ) 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 a digit is equal to 2 (two=2) or not (two=0). Using these three bits we calculate the MB digits by the following relation:

$$\mathbf{y}_j^{MB}=(-1)^{s_j}\cdot[one_j+2\cdot two_j].$$

Figure 2(a) shows the Boolean equations on which the implementation of the MB encoding signals is based (Fig. 2(b))

C. FAM Implementation

In the FAM design presented in Fig. 1(b), the multiplier is a parallel one based on the MB algorithm. Let us consider the product X.Y . The term is en-coded based on the MB algorithm (Section II.B) and multiplied with .Both and consist of n=2k bits and are in 2's complement form.below Equation describes the generation of the partial products:

TABLE I
MODIFIED BOOTH ENCODING TABLE.

Binary				MB Encoding			Input Carry
y_{2j+1}	y_{2j}	y_{2j-1}	$\mathbf{y}_j^{MB}$	sign=s_j	×1=one_j	×2=two_j	$c_{in,j}$
0	0	0	0	0	0	0	0
0	0	1	+1	0	1	0	0
0	1	0	+1	0	1	0	0
0	1	1	+2	0	0	1	0
1	0	0	-2	1	0	1	1
1	0	1	-1	1	1	0	1
1	1	0	-1	1	1	0	1
1	1	1	0	1	0	0	0

$$PP_j=X\cdot\mathbf{y}_j^{MB}=\overline{p}_{j,n}2^n+\sum_{i=0}^{n-1}p_{j,i}\cdot 2^i.$$

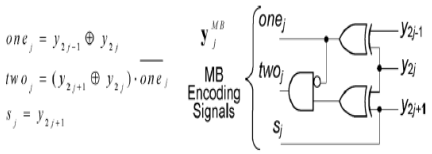


Figure 2: (a) Boolean equations and (b) gate-level schematic for the implementation of the MB encoding signals

The generation of the i-th bit p i,j of the partial product PPj is based on the next logical expression

$$p_{j,i}=((x_i\oplus s_j)\wedge one_j)\vee((x_{i-1}\oplus s_j)\wedge two_j).$$

After the partial products are generated, they are added, properly weighted, through aWallace Carry-Save Adder (CSA) tree along with the Correction Term (CT) which is given by the following equations:

$$Z=X\cdot Y=CT+\sum_{j=0}^{k-1}PP_j\cdot 2^{2j}$$

$$CT = CT(low) + CT(high) = \\ = \sum_{j=0}^{k-1} c_{\text{in},j} \cdot 2^{2j} + 2^n \left(1 + \sum_{j=0}^{k-1} 2^{2j+1}\right)$$

where $c_{\text{in},j} = (one_j \vee two_j) \wedge s_j$ (see Table I).

3. Sum to Modified Booth Recoding Technique (S-MB)

A. Defining Signed-Bit Full Adders and Half Adders for Structured Signed Arithmetic

In S-MB recoding technique, we recode the sum of two consecutive bits of the input $A\ (a_{2j},\ a_{2j+1})$ with two consecutive bits of the input $B\ (b_{2j},\ b_{2j+1})$ into one MB digit y_j^{MB}. As we observe from (2), three bits are included in forming a MB digit. The most significant of them is negatively weighted while the two least significant of them have positive weight. Consequently, in order to transform the two aforementioned pairs of bits in MB form we need to use signed-bit arithmetic. For this purpose, we develop a set of bit-level signed Half Adders (HA) and Full Adders (FA) considering their inputs and outputs to be signed.

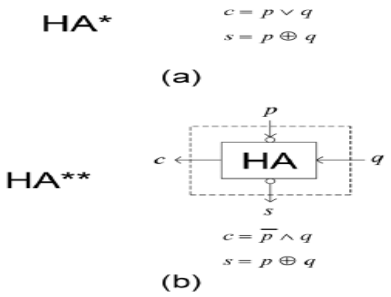


Figure 4: Boolean equations and schematics for signed (a) HA* and (b) HA**.

More specifically, in this work, we use two types of signed HAs which are referred as HA* and HA**. Tables II - IV are their truth tables and in Fig. 4 we present their corresponding Boolean equations. Considering that , are the binary inputs and , are the outputs (carry and sum respectively) of a HA* which implements the relation 2.c-s=p+q where the sum s is considered negatively signed (Table II, Fig. 4(a)), the output takes one of the values {0,+1,+2}. In Table III, we also describe the dual implementation of HA* where we inversed the signs of all inputs and outputs and, consequently, changed the output values to {-2,-1,0}. Table IV and Fig. 4(b) show the operation and schematic of HA** which implements the relation 2.c-s=-p+q and manipulates a negative (p) and a positive (q) input resulting in the output values {-1,0,+1}.

TABLE II
HA* BASIC OPERATION.

Inputs		Output Value[1]	Outputs	
p (+)	q (+)		c (+)	s (-)
0	0	0	0	0
0	1	+1	1	1
1	0	+1	1	1
1	1	+2	1	0

[1] $Output\,Value = 2 \cdot c - s = p + q$

TABLE III
HA* DUAL OPERATION.

Inputs		Output Value[2]	Outputs	
p (-)	q (-)		c (-)	s (+)
0	0	0	0	0
0	1	-1	1	1
1	0	-1	1	1
1	1	-2	1	0

[2] $Output\,Value = -2 \cdot c + s = -p - q$

TABLE IV
HA** OPERATION.

Inputs		Output Value[3]	Outputs	
p (-)	q (+)		c (+)	s (-)
0	0	0	0	0
0	1	+1	1	1
1	0	-1	0	1
1	1	0	0	0

[3] $Output\,Value = 2 \cdot c - s = -p + q$

Also, we design two types of signed FAs which are presented in Table V and VI and Fig. 5. The schematics drawn in Fig. 5(a) and (b) show the relation of FA* and FA** with the conventional FA. Assuming that p q,and ci are the binary inputs and Co,s are the output carry and sum respectively.

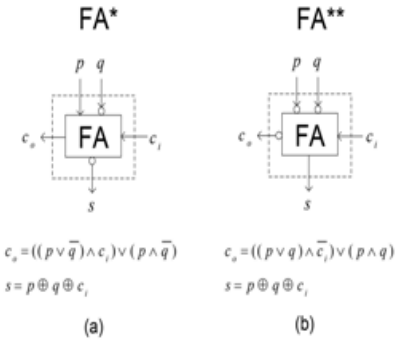


Figure 5: Boolean equations and schematics for signed (a) FA* and (b) FA**

TABLE V
FA* OPERATION.

Inputs			Output Value[1]	Outputs	
p (+)	q (-)	c_i (+)		c_o (+)	s (-)
0	0	0	0	0	0
0	0	1	+1	1	1
0	1	0	-1	0	1
0	1	1	0	0	0
1	0	0	+1	1	1
1	0	1	+2	1	0
1	1	0	0	0	0
1	1	1	+1	1	1

[1] $Output\,Value = 2 \cdot c_o - s = p - q + c_i$

TABLE VI
FA** OPERATION.

Inputs			Output Value[2]	Outputs	
p (-)	q (-)	c_i (+)		c_o (-)	s (+)
0	0	0	0	0	0
0	0	1	+1	0	1
0	1	0	-1	1	1
0	1	1	0	0	0
1	0	0	-1	1	1
1	0	1	0	0	0
1	1	0	-2	1	0
1	1	1	-1	1	1

[2] $Output\,Value = -2 \cdot c_o + s = -p - q + c_i$

B. Proposed S-MB Recoding Techniques

We use both conventional and signed HAs and FAs of Section 3.A in order to design and explore three new alternative schemes of the S-MB recoding technique. Each of the three schemes can be easily applied in either signed (2's complement representation) or unsigned numbers which consist of odd or even number of bits. In all schemes we consider that both inputs A and B are in 2's complement form and consist of 2k bits in case of even or 2k+1 bits in case of odd bit-width.

1)S-MB1 Recoding Scheme: The first scheme of the proposed recoding technique is referred as *S-MB*1 and is illustrated in detail in Fig. 6 for both even (Fig. 6(a)) and odd (Fig. 6(b)) bit-width of input numbers. As can be seen in Fig. 6, the sum of A and B is given by the next relation:

$$Y = A + B = \mathbf{y}_k \cdot 2^{2k} + \sum_{j=0}^{k-1} \mathbf{y}_j^{MB} \cdot 2^{2j}$$

$$\text{where} \quad \mathbf{y}_j^{MB} = -2s_{2j+1} + s_{2j} + c_{2j}.$$

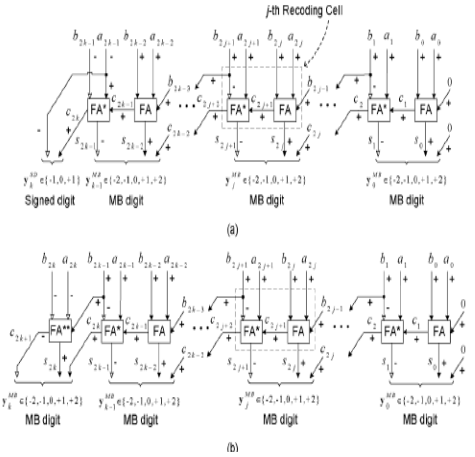


Figure 6: S-MB1 recoding scheme for (a) even and (b) odd number of bits.

2)S-MB2 Recoding Scheme: The second approach of the proposed recoding technique, S-MB2, is described in Fig. 7 for even (Fig. 7(a)) and odd (Fig. 7(b)) bit-width of input numbers. We consider the initial values $c_{0,1} = 0$ and $c_{0,2} = 0$.

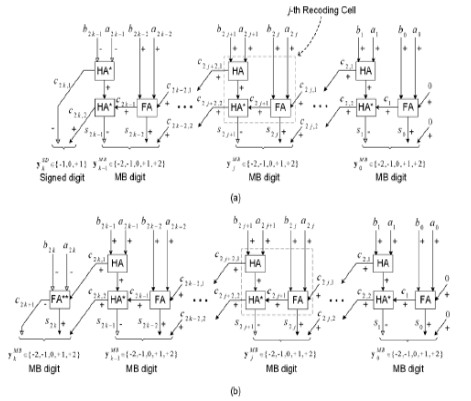


Figure 7: S-MB2 recoding scheme for (a) even and (b) odd number of bits.

3) S-MB3 Recoding Scheme: The third scheme implementing the proposed recoding technique is S-MB3. It is illustrated in detail in Fig. 8 for even (Fig. 8(a)) and odd (Fig.8(b)) bit-width of input numbers. We consider that $c_{0,1} = 0$ and $c_{0,2} = 0$.

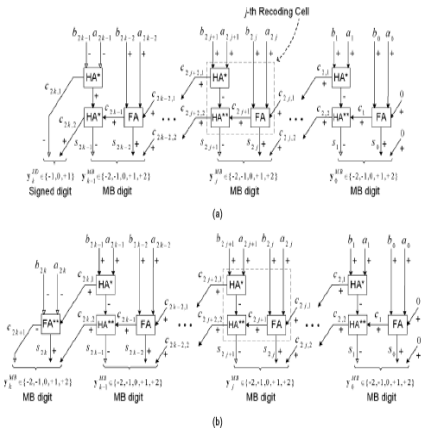


Figure 8: S-MB3 recoding scheme for (a) even and (b) odd number of bits.

4) Unsigned Input Numbers: In case that the input numbers A and B are unsigned, their most significant bits are positively signed. Figs. 9–11 present the modifications that we have to make in all S-MB schemes for both cases of even (the two most significant digits change) and odd (only the most significant digit change) bit-width of A and B , regarding the signs of the most significant bits of A and B. The basic recoding block in all schemes remains unchanged.

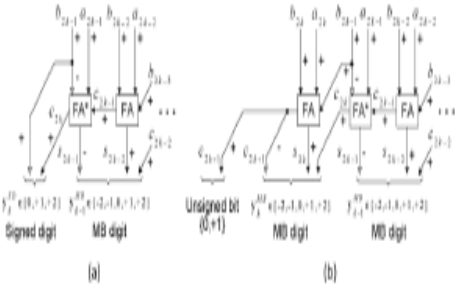


Figure 9: Implementation of the MSD of the S-MB1 recoding scheme in case of unsigned input numbers for (a) even and (b) odd bit-width.

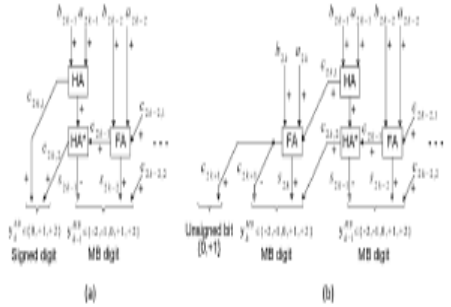


Figure 10: Implementation of the MSD of the S-MB2 recoding scheme in case of unsigned input numbers for (a) even and (b) odd bit-width.

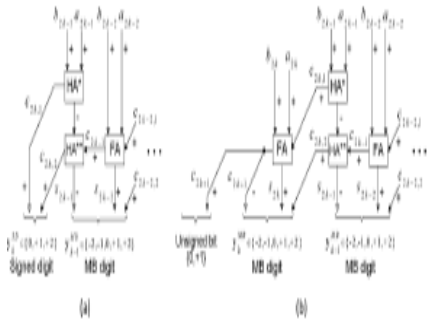


Figure 11: Implementation of the MSD of the S-MB3 recoding scheme in case of unsigned input numbers for (a) even and (b) odd bit-width.

4. Simulation Results

The proposed Fused add unit is implemented, 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.12 FAM Unit with CLA adder for even bit and Fig.13 FAM unit with CLA 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 CLA adder .

The below figure shows the Block diagram of the fused add-multiply(FAM)operator

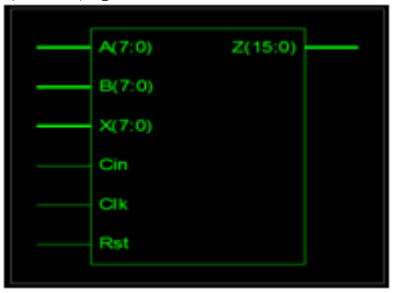


Figure 12: Blockdiagram of FAM operatorS-MB3 Recoding scheme(for EVEN bits)

RTL Schematic:

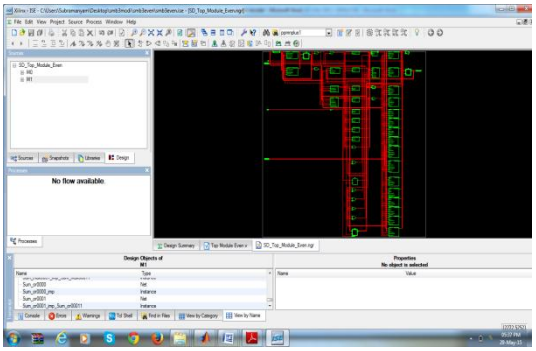


Figure 13: RTL Schematic for SMB3 EVEN

Technology schematic:

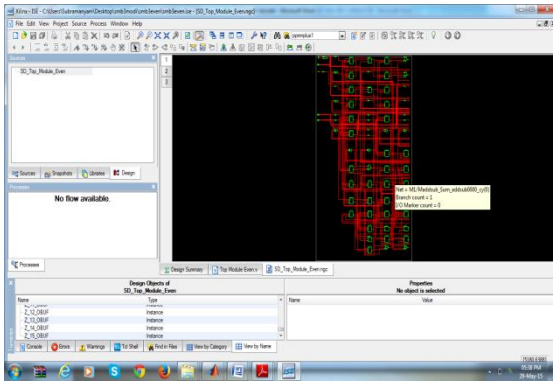


Figure 14: Technology schematic for SMB3(EVEN)

Simulation results:

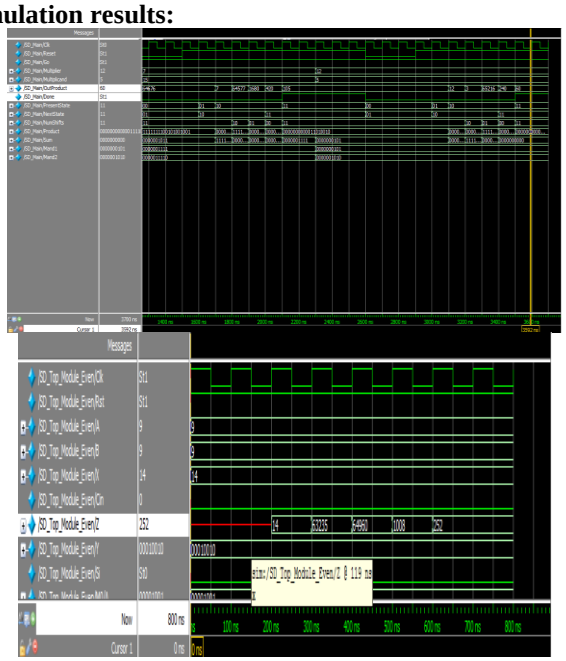


Figure 15: Simulation output waveform for the SMB3(EVEN)S-MB3 Recoding scheme (for ODD bits):

RTL schematic:

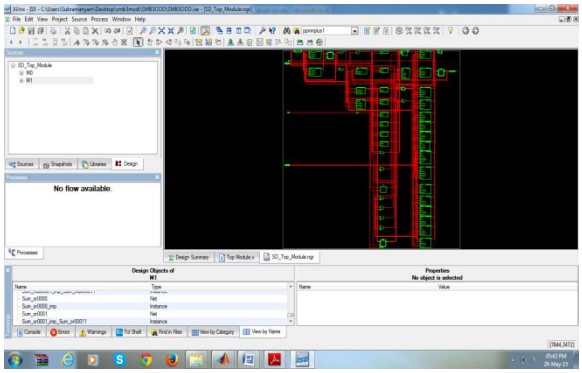


Figure 16: RTL Schematic for SMB3 (ODD)

Technology schematic:

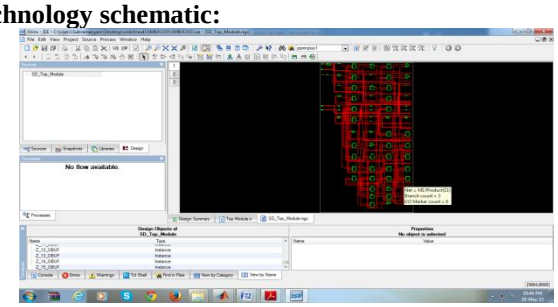


Figure 17: Technology schematic for SMB3 (ODD)

Simulation Results :

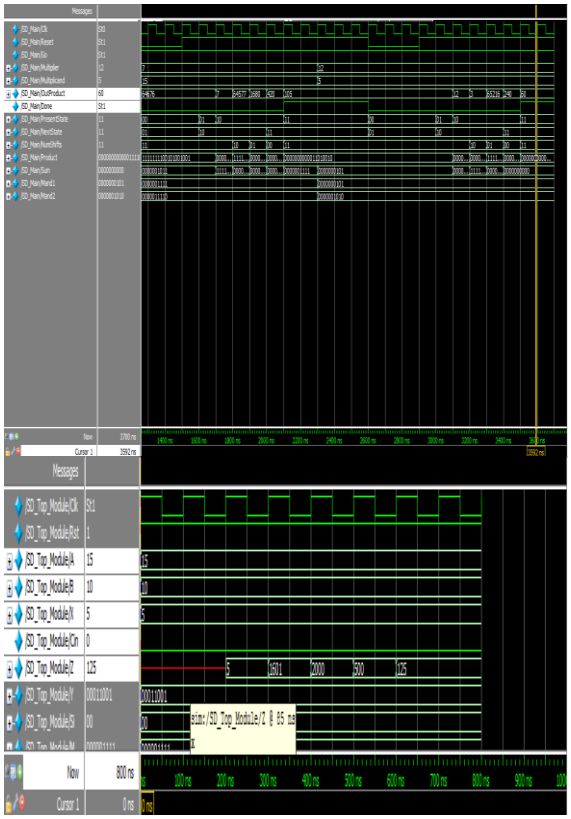


Figure 18: Simulation output waveform for the SMB3(ODD)

5. Conclusion

This paper focuses on optimizing the design of the Fused-Add Multiply (FAM) operator. We propose a structured technique for the direct recoding of the sum of two numbers to its MB form. We explore three alternative designs of the proposed S-MB recorder. We show that the adoption of the proposed recoding technique delivers optimized solutions for the FAM design 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 occupation and power consumption, thus outperforming the existing S-MB recoding solutions.

References

[1] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, 1964.
[2] [Online].Available: http://www.faraday-tech.com/main/IPonline/category.do?method=showProcessIPList&process=90&categoryID=3089&categoryName=Standard%20Cell
[3] M. Daumas and D. W. Matula, "A Booth multiplier accepting both a redundant or a non redundant input with no additional delay," in Proc.IEEE Int. Conf. on Application-Specific Syst., Architectures, and Processors,2000, pp. 205–214.
[4] 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.
[5] 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.
[6] 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.