High speed design of fused add multiply operator Using SMB recoding technique

Christina Jesintha. R, Sudha. R.

***Abstract*— Add-Multiply operator is an operating unit which performs addition and then add the result to the input of the multiplier. It increases both area and critical path delay of the circuit, since the adder of AM unit inserts delay in its critical path and increases area occupation and power consumption. Hence the AM operator is optimized by employing fusion techniques. For the Design optimization of Fused- Add Multiply (FAM) unit recoding techniques are introduced. These techniques are used to implement direct recoding of the sum of two numbers in its modified booth form. Modified booth is a prevalent form used in multiplication; it reduces the number of partial products into half. In order to improve the performance of FAM unit different schemes of recoding technique called sum to modified booth recoding technique(S-MB) are introduced. By comparing the proposed recoding schemes with existing one the proposed technique yields considerable reductions in terms of delay, hardware complexity and power consumption of the FAM unit.**

***Index Terms*—Carry Look ahead adder, Carry Save adder, fused add multiply operator, modified booth recoding technique.**

I. INTRODUCTION

Digital signal processing (DSP) is the mathematical manipulation of an information signal to modify or improve it in some way. The goal of DSP is usually to measure, filter and/or compress continuous real-world analog signals. Usually, the first step is conversion of the signal from an analog to a digital form, by sampling and then digitizing it using an analog-to-digital converter (ADC), which turns the analog signal into a stream of discrete digital values. Often, however, the required output signal is also analog, which requires a digital-to-analog converter (DAC). Modern consumer electronics make extensive use of Digital Signal Processing (DSP) providing custom accelerators for the domains of multimedia, communications etc. Typical DSP applications carry out a large number of arithmetic operations as their implementation is based on computationally intensive kernels, such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Finite Impulse Response (FIR) filters and signals' convolution. As expected, the performance of DSP systems is inherently affected by decisions on their design regarding the allocation and the architecture of arithmetic units.

1.1 Fused Multiply-Add

A fused multiply–add is a floating-point multiply–add operation performed in one step, with a single rounding. That is, where an un fused multiply–add would compute the product b×c, round it to N significant bits, add the result to a, and round back to N significant bits, a fused multiply–add would compute the entire sum a+b×c to its full precision before rounding the final result down to N significant bits. .Fused multiply–add can usually be relied on to give more accurate results. However, Kahan has pointed out that it can give problems if used unthinkingly. If $x^2 - y^2$ is evaluated as ((x×x) − y×y) using fused multiply–add, then the result may be negative even when x = y due to the first multiplication discarding low significance bits.

1.2 Arithmetic Circuits

Arithmetic circuits are the most natural and standard model for computing polynomials. In this model the inputs are variables $x_1....x_n$, and the computation is performed using the arithmetic operations +, × and may involve constants from a field F. The output of an arithmetic circuit is thus a polynomial (or a set of polynomials) in the input variables. The complexity measures associated with such circuits are size and depth which capture the number of operations and the maximal distance between an input and an output, respectively.

1.3 Modified Booth Recoding

Booth's algorithm examines adjacent pairs of bits of the N-bit multiplier Y in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to N-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product. The multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at i = 0; the multiplication by 2^i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P.

ISSN: 2278 – 909X

1.4 Add Multiply Operator

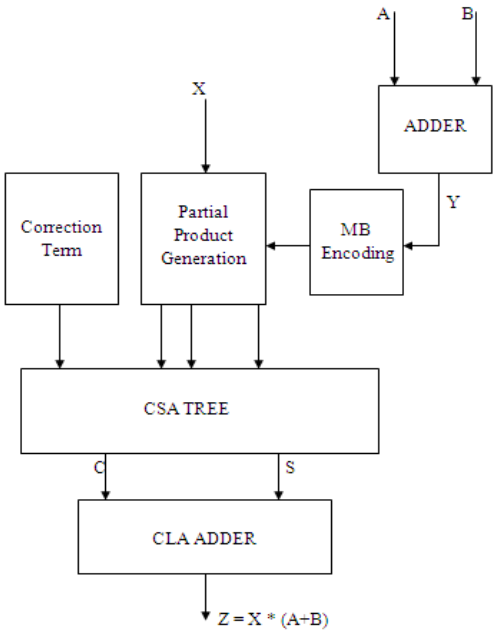

Fig .1 Conventional design of AM operator

Here we focus on AM units which implement the operation Z=X. (A+B). The conventional design of the AM operator requires that its input A and B are first driven into an adder and then the input X and the sum Y=A+B is 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 adder (CLA) which however increases the area occupation and power dissipation. The conventional design of AM operator is shown in fig.1.

1.5 Modified Booth Form

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:

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

$$\langle y_{k-1}y_{k-2}....y_1y_0\rangle_{MB} = \sum_{j=0}^{K-1} y_j^{MB}.2^{2j} \quad (1)$$

$$y_j^{MB} = y_{2j+1} + y_{2j} + y_{2j-1}$$

Digits y_j^{MB} ε {-2, -1, 0, +1, +2}, $0 \le j \le k-1$, correspond to the three consecutive bits y_{2j+1}, y_{2j} and y_{2j-1} with one bit overlapped and considering that $y_{-1} = 0$. Each digit is represented by three bits named s, *one* and *two*. The sign bit 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 a digit is equal to 2 (two = 1) or not (two = 0). Using these three bits calculates the MB digits y_j^{MB} by the following relation:

$$y_j^{MB} = (-1)^s * [one_j + 2 * two_j]. \quad (2)$$

1.6 FAM implementation

Let us consider the product X * Y. The term Y = $\langle y_{n-1}\ y_{n-2} \dots\ y_1\ y_0 \rangle_{2's}$ is encoded based on the MB algorithm and multiplied with X = $\langle x_{n-1}\ x_{n-2} \dots\ x_1\ x_0 \rangle_{2's}$. Both X and Y consist of n = 2k bits and are in 2's complement form. Equation (4) describes the generation of the partial products:

$$PP_j = X.y_j^{MB} = \bar{p}_{j,n}2^n + \sum_{i=0}^{n-1} p_{j,i}.2^i$$

The generation of the i-th bit $p_{j,i}$ of the partial product PP_j is based on the next logical expression while Fig 2. Illustrates its implementation at gate level:

$$p_{j,i} = ((x_i \oplus s_j) \wedge one_j) \vee ((x_{i-1} \oplus s_j) \wedge two_j). \quad (5)$$

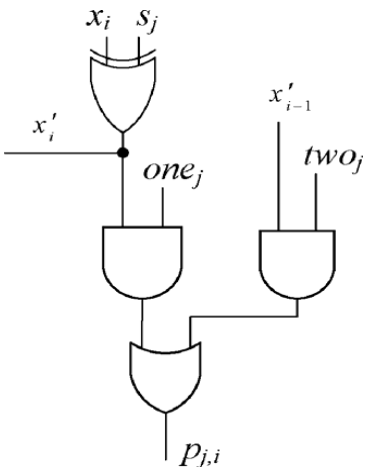

Fig.2 Generation of i-th bit $P_{j,i}$ of the partial product PP_j

For the computation of the least and the most significant bits of the partial product it considers $x_{-1} = 0$ and $x_n = x_{n-1}$respectively. Note that in case that n = 2k + 1, the number of the resulting partial products is [n/2] + 1 = k + 1 and the most significant MB digit is formed based on sign extension of the initial 2's complement number.

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

$$Z = X.Y = CT + \sum_{j=0}^{k-1} pp_j . 2^{2j} \quad (3)$$

$$CT = CT\ (low) + CT\ (high) = \sum_{j=0}^{k-1} c_{in,j} . 2^{2j} +$$

$$2^n(1+\sum_{j=0}^{k-1} 2^{2j+1}) \quad (4)$$

1.7 Fused Add-Multiply Operator

An optimized design of the AM operator is based on the fusion of the adder and the MB encoding unit into a single data path block by direct recoding of the sum 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, significant 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. Fig.4 represents the fused design of AM operator.

ISSN: 2278 – 909X

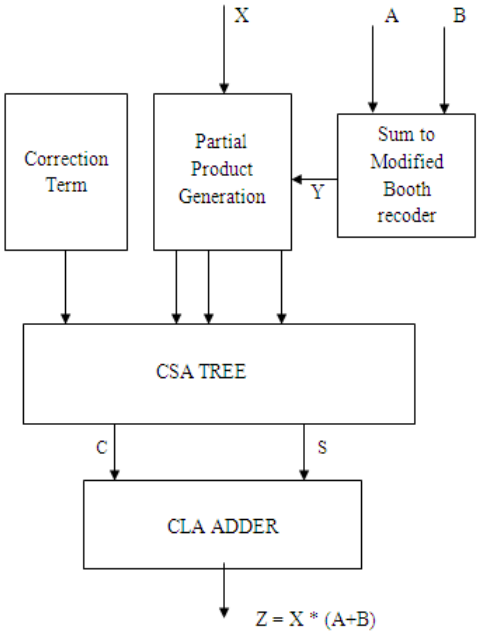

Fig.3 Fused design of AM operator

1.8 SMB-1 RECODING SCHEME

The first scheme of the proposed recoding technique is referred as *S-MB*1 and is illustrated in detail in Fig.4 for both even and odd bit-width of input numbers. As can be seen in Fig.4, the sum of A and B is given by the next relation:

$$Y = A+B = y_k \cdot 2^{2k} + \sum_{j=0}^{k-1} y_j^{MB} \cdot 2^{2j} \quad (4)$$

Where $y_j^{MB} = -2s_{2j+1} + s_{2j} + c_{2j}$

The encoding of the MB digits y_j^{MB}, $0 \le j \le k-1$, of is based on the analysis. Bits s_{2j+1} and s_{2j} are extracted from the j recoding cell of Fig.4. A conventional FA with inputs a_{2j}, b_{2j} and b_{2j-1} produces the carry $c_{2j+1} = (a_{2j} \Lambda b_{2j}) V (b_{2j-1} \Lambda (a_{2j} V b_{2j}))$ and the sum $s_{2j} = a_{2j} \oplus b_{2j} \oplus b_{2j-1}$. As the bit s_{2j+1} need to be negatively signed, use a FA* with inputs a_{2j+1}, b_{2j+1} (-) and c_{2j+1}, which produces the carry c_{2j+2} and the sum s_{2j+1} (-):

$$C_{2j+2} = (a_{2j+1} \wedge \bar{b}_{2j+1}) \vee (c_{2j+1} \wedge (a_{2j+1} \vee \bar{b}_{2j+1})) \quad (5)$$

Note that, based on equation $b_{2j+1} = 2 * b_{2j+1} - b_{2j+1}$, b_{2j+1} is driven to the FA* as negatively signed while it is also used with positive sign as an input carry of the subsequent recoding cell. It consider the initial values $b_{-1} = 0$ and $c_0 = 0$. When its form the most significant digit (MSD) of the *S-MB*1 recoding scheme, it distinguish two cases: In the first case, the bit-width of A and B is even, while in the second case, both A and B comprise of odd number of bits. In the first case, the MSD $y_{k,even}^{SD}$ is a signed digit and is given by the next algebraic equation:

$$y_{k,even}^{SD} = -a_{2k-1} + c_{2k} \quad (6)$$

The critical path delay of *S-MB*1 recoding scheme is constant in respect to the input bit-width and is given by the equation:

$$T_{S\text{-}MB1} = T_{FA,carry} + T_{FA^*,sum} \quad (7)$$

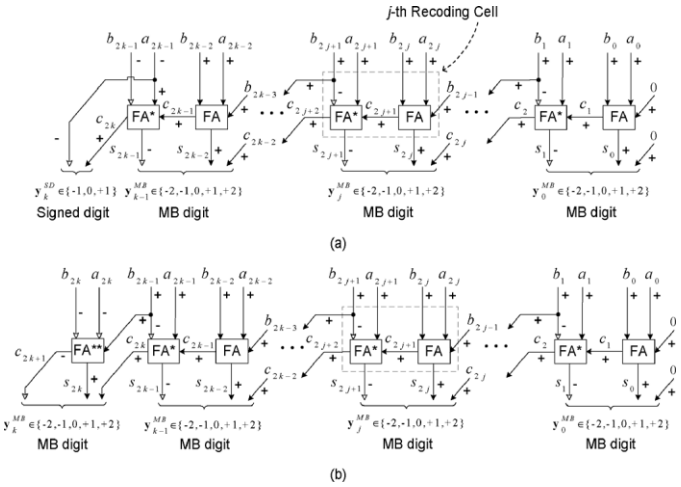

Fig.4*S-MB*1 recoding scheme for (a) even and (b) odd number of bits.

1.9 SMB recoding scheme 2

The second approach of the proposed recoding technique, S-MB2, is described in Fig.5 for even and odd bit-width of input numbers. It consider the initial values $c_{0,1} = 0$ and $c_{0,2} = 0$. The digits y_j^{MB}, $0 \le j \le k-1$, are formed based on s_{2j+1}, s_{2j} and $c_{2j,2}$. As in the S-MB1 recoding scheme, it uses a conventional FA to produce the carry c_{2j+1} and the sum s_{2j}. The inputs of the FA are a_{2j}, b_{2j} and $c_{2j,1}$. The bit $c_{2j,1}$ is the output carry of a conventional HA which is part of the (j-1) recoding cell and has the bits a_{2j-1}, b_{2j-1} as inputs. The bit s_{2j+1} is the output sum of a HA* in which it drives c_{2j+1} and the sum produced by a conventional HA with the bits a_{2j+1}, b_{2j+1} as inputs. The HA* is used in order to produce the negatively signed sum s_{2j+1} and its outputs are given by the following Boolean equations:

$$c_{2j+2,2} = c_{2j+1} V (a_{2j+1} \oplus b_{2j+1}) \quad (8)$$

$$s_{2j+1} = a_{2j+1} \oplus b_{2j+1} \oplus c_{2j+1}$$

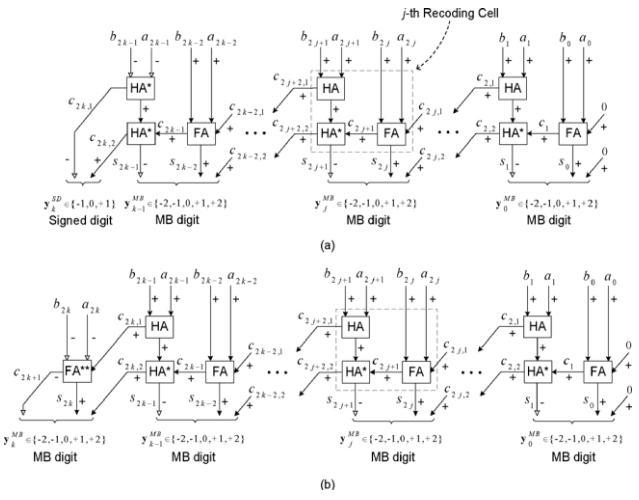

Fig.5*S-MB*2 recoding scheme for (a) even and (b) odd number of bits

In case that A and B comprise of even number of bits, a_{2n-1} and b_{2n-1} are negatively weighted and the conventional HA of the (n-1) recoding cell is replaced by the dual HA* The MSD $y_{k,even}^{SD}$ is a signed digit and is given by the relation:

ISSN: 2278 – 909X

$$y_{k,even}^{SD} = -c_{2k,1} + c_{2k,2} \qquad (9)$$

2. SMB 3 recoding scheme

The third scheme implementing the proposed recoding technique is *S-MB*3. It is illustrated in detail in for even and odd bit-width of input numbers. It consider that $c_{0,1} = 0$ and $c_{0,2}$. It builds the digits y_j^{MB} , $0 \le j \le k-1$, based on s_{2j+1}, s_{2j} and $c_{2j,2}$.

Once more, it uses a conventional FA to produce the carry c_{2j+1} and the sum s_{2j}. The bit $c_{2j,1}$ is now the output carry of a HA* which belongs to the (j-1) recoding cell and has the bits a_{2j-1}, b_{2j-1} as inputs. The negatively signed bit s_{2j+1} is produced by a HA** in which drive c_{2j+1} and the output sum (negatively signed) of the HA* of the recoding cell with the bits a_{2j+1}, b_{2j+1} as inputs. The carry and sum outputs of the HA** are given by the following Boolean equations:

$$c_{2j+2,2} = c_{2j+1} \wedge (\overline{a_{2j+1} \oplus b_{2j+1}})$$
$$s_{2j+1} = a_{2j+1} \oplus b_{2j+1} \oplus c_{2j+1}. \qquad (15)$$

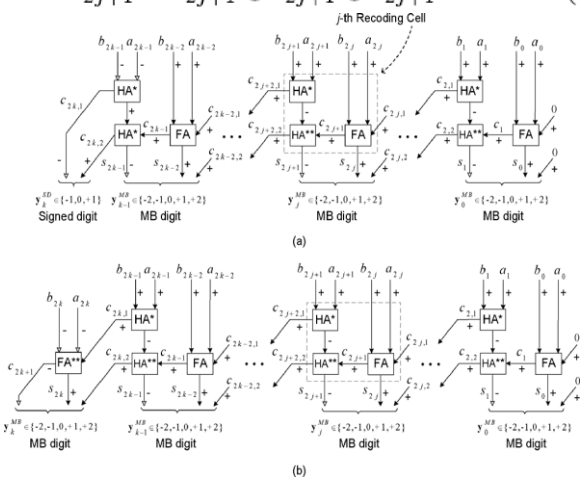

Fig.6 *S-MB*3 recoding scheme for (a) even and (b) odd number of bits.

2.1 SOFTWARE REQUIREMENTS

Modelsim SE 6.3f

In this paper the model sim software is used for simulation and verification. Model Sim is a verification and simulation tool for VHDL, Verilog, SystemVerilog, and mixed- language designs.

The best standards and platform support in the industry make it easy to adopt in the majority of process and tool flows. Model Technology's award-winning Single Kernel Simulation (SKS) technology enables transparent mixing of VHDL and Verilog in one design.

XILINX ISE

Xilinx ISE 7.1i & Xilinx Platform Studio 7.1 are the programming tools for the system. This is used for generate RTL & FPGA schematic. This is used for create logic design like micro blaze design XPS includes a graphical user interface (GUI), along with a set of tools that aid in project design. From the XPS GUI, you can design a complete embedded processor system for implementation within a Xilinx FPGA device.

2.2 RESULTS AND DISCUSSION

EVEN PART RESULTS

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilizatio**
Number of 4 input LUTs	187	26,624	
Logic Distribution			
Number of occupied Slices	100	13,312	
Number of Slices containing only related logic	100	100	
Number of Slices containing unrelated logic	0	100	
Total Number of 4 input LUTs	187	26,624	
Number of bonded IOBs	126	487	
Total equivalent gate count for design	1,191		
Additional JTAG gate count for IOBs	6,048		

Performance Summary		
Final Timing Score:	0	**Pinout Data:**
Routing Results:	All Signals Completely Routed	**Clock Data:**
Timing Constraints:	All Constraints	
Detailed Reports		

Fig.7Area of S-MB1- even

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		10875
Vccint 1.20V:	191	229
Vccaux 2.50V:	35	88
Vcco25 2.50V:	4224	10559
Inputs:	2	3
Logic:	30	35
Outputs:		
Vcco25	4224	10559
Signals:	114	137
Quiescent Vccint 1.20V:	45	54
Quiescent Vccaux 2.50V:	35	88

Thermal summary:	
Estimated junction temperature:	210C
Ambient temp:	25C
Case temp:	188C
Theta J-A:	17C/W

Fig.8 power consumption of S-MB1-even

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 38.817ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 710634 / 102
-------------------------------------------------------------------------
Delay:               38.817ns (Levels of Logic = 22)
  Source:            a<3> (PAD)
  Destination:       z_out<7> (PAD)
```

Fig.9 The total delay of S-MB1-even

ODD PART RESULTS

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilizatior**
Number of 4 input LUTs	190	26,624	
Logic Distribution			
Number of occupied Slices	100	13,312	
Number of Slices containing only related logic	100	100	
Number of Slices containing unrelated logic	0	100	
Total Number of 4 input LUTs	190	26,624	
Number of bonded IOBs	126	487	
Total equivalent gate count for design	1,218		
Additional JTAG gate count for IOBs	6,048		

Performance Summary		
Final Timing Score:	0	**Pinout Data:**
Routing Results:	All Signals Completely Routed	**Clock Data:**
Timing Constraints:	All Constraints	
Detailed Reports		

Fig.10 Area of S-MB-1 odd

ISSN: 2278 – 909X

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		7521
Vccint 1.20V:	146	175
Vccaux 2.50V:	35	88
Vcco25 2.50V:	2903	7258
Inputs:	2	3
Logic:	23	27
Outputs:		
Vcco25	2903	7258
Signals:	76	91
Quiescent Vccint 1.20V:	45	54
Quiescent Vccaux 2.50V:	35	88

Thermal summary:	
Estimated junction temperature:	153C
Ambient temp:	25C
Case temp:	138C
Theta J-A:	17°C/W

Fig.11The power consumption for S-MB1 odd

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 36.741ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 270387 / 101
-------------------------------------------------------------------------
Delay:               36.741ns (Levels of Logic = 21)
  Source:            b<1> (PAD)
  Destination:       z_out<7> (PAD)
```

Fig.12.The total delay for S-MB1-odd

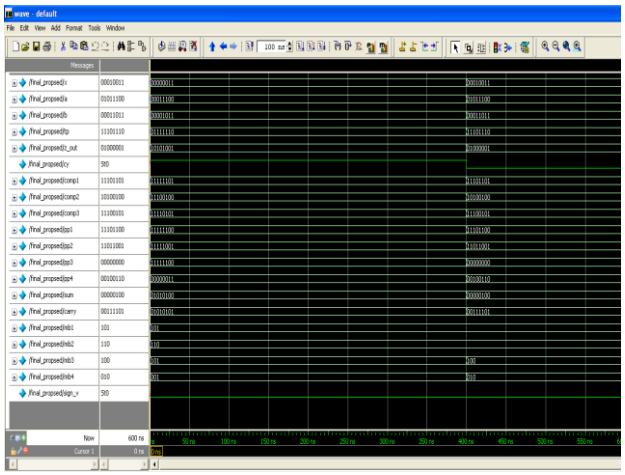

Fig.13 Simulated output of FAM using Model-Sim

2.3 CONCLUSION

In this paper, it presents a Modelsim simulated results of fused add multiply operator (FAM) using S-MB recoding technique. Our design consists of three schemes, S-MB 1 recoding technique, S-MB 2 recoding technique and S-MB 3 recoding technique. The area, power consumption and timing complexity are analyzed for FAM using Xilinx. The S-MB techniques are applied for both signed and unsigned bits and it has two cases namely, S-MB technique in case of odd and S-MB technique in case of even. For both the cases the area complexity, power consumption and timing complexity is analyzed and compared with existing recoding techniques.

REFERENCES

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

[2]Chen L-H. , Chen O. T.-C.T Wang T.-Y. , and Ma Y.-C., (2005) "A multiplication-Accumulation computation unit with optimized compressors and Minimized switching activities," in *Proc. IEEE Int, Symp. Circuits and Syst.*, Kobe, Japan, vol. 6, pp. 6118–6121.

[3]Daumas M.and Matula D. W. (2000) , "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*.

[4]Kwon O., Nowka K., and Swartzlander E.(2002), "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.

[5]Huang Z.,(2003) "High-Level Optimization Techniques for Low-Power Multiplier Design,"Ph.D., University of California, Department of Computer Science, Los Angeles, CA. pp. 205–214.

[6]Huang Z. and Ercegovac M. D.(2005), "High-performance low-power left-to right array multiplier design," *IEEE Trans. Comput.*, vol. 54, no. 3, pp.272–283.

[7]Lyu C. N. and Matula D. W. (1995), "Redundant binary Booth recoding," in *Proc. 12th Symp. Comput. Arithmetic*, 1995, pp. 50–57.

[8]Swartzlander E. E. and Saleh H. H. (2012),"FFT implementation with fused floating-point operations, *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 284–288.

[9]Seo Y.-H. And Kim D.-W. (2010), "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.

[10]]Wallace C. S. (1964), "A suggestion for a fast multiplier," *IEEE Trans. Electron Comput* ., vol. EC-13, no. 1, pp. 14–17.

[11]Xydis S., Triantafyllou I., Economakos G., and Pekmestzi K.(2009),"Flexible datapath synthesis through arithmetically optimized operation chaining," in *Proc. NASA/ESA Conf. Adaptive Hardware Syst.*

[12]Yeh W.-C. and Jen C.-W. (2003),"High-speed and low-power split-radix FFT," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874.

ISSN: 2278 – 909X

[13] [13]Zimmermann R. and Tran D. Q.(2003), "Optimized synthesis of sum-of-products, "in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, Washington, DC, pp. 867–872.

BIOGRAPHY

R. Christina Jesintha did her Bachelor of Engineering in Electronics and Communication Engineering at Jayam college of Engineering and Technology, Dharmapuri and doing Master of engineering in VLSI Design at Sri Shakthi Institute of Engineering and Technology, Coimbatore, India. Her research interests include digital electronics, VLSI design. She has attended a workshop on device modeling and simulation at KPR institute of engineering and technology and another workshop on ASIC Design and verification conducted by Vinchip.

ISSN: 2278 – 909X