COMPARATIVE DESIGN OF REGULAR STRUCTURED MODIFIED BOOTH MULTIPLIER

Ram RackshaTripathi and S.G.Prakash

Department of Electronics & communication, University of Allahabad-211002

ABSTRACT

Multiplication is a crucial function and plays a vital role for practically any DSP system. Several DSP algorithms require different types of multiplications, specifically modified booth multiplication algorithm. In this paper, a simple approach is proposed for generating last partial product row for reducing extra sign (negative bit) bit to achieve more regular structure. As compared to the conventional multipliers these proposed modified Booth's multipliers can achieve improved reduction in area 5.9%, power 3.2%, and delay 0.5% for 8 x 8 multipliers. We can also observe that achievable improvement for 16 x 16 multiplier in area, power, delay are 4.0%, 2.3%, 0.3% respectively. These multipliers are implemented using verilog HDL and synthesized by using synopsis design compiler with an Artisan TSMC 90nm Technology

KEYWORDS

Digital Multiplier, Modified booth Multiplier (MBE), Power and Delay

1. INTRODUCTION

Expansive digital signal processing (DSP) capabilities are a major feature of a large number of System-on-Chip (SoC) designs mainly organize in embedded systems. DSP applications in SoCs range from audio or video processing to wireless communication or adaptive control systems and often make up a considerable part of the system's hardware resources and power consumption. Consequently, various different architecture concepts for their implementation exist, ranging from application-specific components to embedded reconFigureurable [1] hardware or digital signal processors. Each solution represents a specific trade-off between chip area, power consumption, performance, and design effort, currently the most important parameters in SoC design. Starting from our experience with processor data path extensions for wireless sensor network nodes, in this work we investigate the opportunities of reconFigureurable DSP components, which can be reused for different DSP tasks in a SoC. To enhance the processing performance and reducing the power dissipation of systems, designing of multipliers have most challenging task in multimedia and digital signal processing (DSP) applications. Multiplication and multiplication-accumulation (MAC) are very common mathematical operations in many digital signal processing (DSP) applications. For designing of parallel type multipliers there are three common steps to follow: the first one is generating the partial products. Second one is reducing the number of partial product rows. For example, Wallace tree [5]-[6] or Dadda tree. And the third one is adding the remaining two rows of partial products by using a carry propagate adder(e.g., carry lookahead adder) to obtain the final product.

DOI : 10.5121/vlsic.2016.7202

The conventional MBE algorithm generates n/2+1 partial product rows rather than n/2 due to the extra partial product bit at the least significant bit position of each partial product row for negative encoding, leading to an irregular partial product array and a complex reduction tree.

Booth encoding is a technique that leads to smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is the standard technique used in chip design, and provides significant improvements over the "long multiplication" technique. The widely used Booth algorithm is the radix-4 based modified Booth algorithm proposed by McSorley where it reduces the partial products into half. As the number of partial products reduces the number of CSAs required for the compression module, the height of the Wallace tree is also reduced.

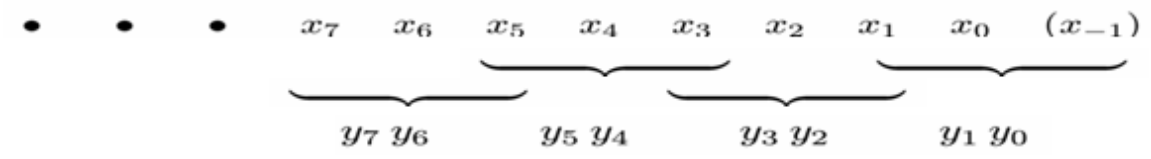

Figure 1. Modified Booth recoding pattern

In Figure.1.Modified Booth algorithm's basic idea is that the bits *Yi* and Y_{i-1} are recoded into Z_i and Z_{i-1}, while Y_{i-2} serves as reference bit. In a separate step, Y_{i-2} and Y_{i-3} recoded into Z_{i-2} and Z_{i-3} with, Y_{i-4} serving as reference bit. This signifies that the modified Booth's encoding partitions input *Y* into a group of 3-bits with 1-bit overlap and generates the following five signed digits, 2, 1, 0, -1 and -2. Encoding on the each group reduces the number of partial products by factor of 2.

Operations on the encoded digits performed with multiplier input *X* is illustrated in Table 1.
However, it is important to note that there are two unavoidable consequences of using MBE: sign extension prevention and negative encoding. The combination of these two unavoidable consequences results in the formation of one additional partial product row. So that, accumulation of the extra partial product row requires more hardware, and time.

Table 1 : Partial Product Selections and Operations

Recoded digit	*Booth's operation on X*	*Y2i-1 Y2i Y2i+1*
0	*Add 0 to PP*	*{0 0 0, 1 1 1}*
+1	*Add X to PP*	*{0 0 1, 0 1 0}*
+2	*Shift X left & add to PP*	*{0 1 1}*
-1	*Add 2's complementary X to PP*	*{ 1 0 1, 1 1 0}*
-2	*2's complementary X & shift-add*	*{1 0 0}*

To overcome the above consequences, the authors in [2] added the least significant bit of each partial product row with the *neg* bit of corresponding row to obtained a new least significant bit Y_{i0} and a carry c_i. Note that both Y_{i0} and c_i are generated no later than other partial product bits. Figure 2 depicts the 8 x 8 MBE partial product array generated by the approach proposed in [2].

Since c_i is at the left one bit position of neg_i, the required additions in the reduction tree reduced. However , the approach does not remove the additional partial product row.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					A_9	A_8	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	T_{00}
				1	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1	T_{10}	K_0	
		1	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	T_{20}	K_1			
	1	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	T_{30}	K_2				
									K_3						

Figure 2. Proposed MBE partial product array in [2] for 8 x 8 multiplication

The conventional method (complement a binary number and add 1 to the complemented number) will not work properly because the propagation delay of the carry linearly increases with the word size and it would be much greater than the delay to generate the partial products. Therefore, he proposed an extension of the well-known algorithm where all the bits after the rightmost "1" in the word are complemented but all the other bits are unchanged. The two's complement of a binary number 001010_2 is 110110_2. For this number, the rightmost "1" happens in bit position 1. Therefore, values in bit positions 2 to 5 can simply be complemented, while values in bit positions 0 and 1 are kept unchanged. Therefore, two's complementation now comes down to finding the conversion signals that are used for selectively complementing some of the input bit.

If the conversion signal at any position is "0", then the value is kept unchanged and, if the conversion signal is "1", then the value is complemented. The conversion signals after the rightmost "1" are always 1. They are 0 otherwise. Once a lower order bit has been found to be a "1," the conversion signals for the higher order bits to the left of that bit position should all be "1."

However, this searching for the rightmost "1" could be as time consuming as rippling a carry through to the MSB since the previous bits information must be transferred to the MSB Therefore, they find a method to expedite this detection of the rightmost "1."

As we will see, this search for the rightmost "1" can be achieved in logarithmic time using a binary search tree-like structure. They first find the conversion signals for a 2-bit group by grouping two consecutive bits (the grouping always starts from the LSB) from the input and finding the conversion signals in each group, as shown in Figure.3. Then, they find the conversion signals for a 4-bit group (formed by two consecutive 2-bit groups). Then, they find the conversion signals for a 8-bit group (formed by two consecutive 4-bit groups). This divide-and-conquer approach is pursued until the whole input has been covered.

								X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
							$\times$	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
					$\overline{PP_{80}}$	PP_{80}	PP_{80}	PP_{70}	PP_{60}	PP_{50}	PP_{40}	PP_{30}	PP_{20}	PP_{10}	PP_{00}
				1	$\overline{PP_{81}}$	PP_{71}	PP_{61}	PP_{51}	PP_{41}	PP_{31}	PP_{21}	PP_{11}	PP_{01}		neg_0
	1	1	$\overline{PP_{82}}$	PP_{72}	PP_{62}	PP_{52}	PP_{42}	PP_{32}	PP_{22}	PP_{12}	PP_{02}		neg_1		
$\overline{S_9}$	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0		neg_2				

Figure 3 Proposed partial products after removing the last neg.

However, the approach must additionally develop and design the 2's complement logic. It possibly enlarges the area and delay of the partial product generator and lessens the benefit contributed by removing the extra row.

To remove the extra partial product row the authors in [1] extends the methods proposed in [2] and [3]. To remove the extra partial product row $pp_{n/2}$ due to c_3 in Figure 2, they combine the c_i for i=n/2-1 with the partial product bit p_{i1} to produce a new partial product bit τ_{i1} and a new carry d_i can be incorporated into the sign extension bits of pp_0. The logic expressions of τ_{i1} and d_i can be written as

$$\tau_{i1} = one_i . \in + two_i . x_0 = \overline{(\overline{one}_i + \in).(\overline{two}_i + \overline{x}_0)}$$

$$d_i = \left(\overline{b_{2i+1} + a_0}\right).\left[\overline{(b_{2i-1} + a_1).(b_{2i} + a_1).(b_{2i} + b_{2i-1})}\right]$$

And the carry bit is incorporating into the sign extension bits of first partial product row. The maximal value of these sign extension bits are 100 so the addition of carry bit to sign extension bits will never produce an overflow.

2. PROPOSED WORK

For recording of MBE, We need at least three signal to represent the digit set {-2X, -1X, 0, 1X, 2X}. Many different ways have been developed. But the proposed selector circuit in[15] is not efficient to design the multiplier because it can not satisfy all the encoding conditions. For that purpose we redesign the selector circuit. The Booth encoder and selector circuits for the implementation of proposed MBE multiplier are shown in Figure. 4, Figure. 5 respectively.

To have a more regular least significant part of each partial product row , the authors in [2] added the least significant bit $_{i0}$ with neg_i in advance and obtained a new least significant bit γ_{i0} and a carry c_i. However, the approach does not remove the additional partial product row. i.e. , n/2+1 row still present. To remove this extra row, we are using the advantages of [3]. For generating a 2's complement of a number generally we have two methods. The first method is converting each bit of a binary number and add '1' to the least significant bit. And the second method is that search for a right most binary '1' and all the bits after right most '1' in the word are complemented but all the other bits are unchanged.

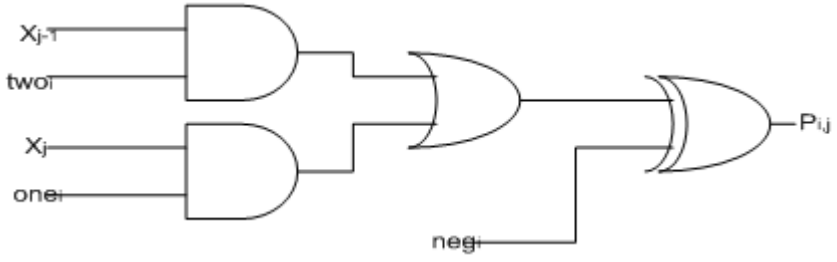

Figure 4. Proposed MBE selector circuit

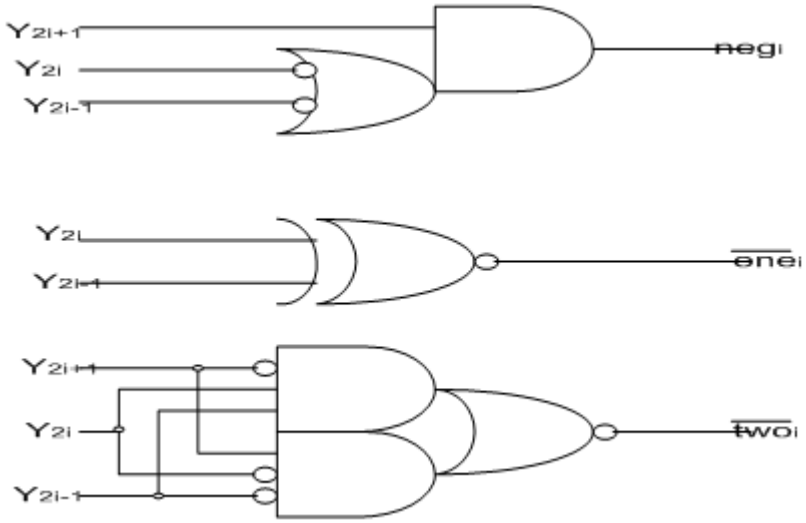

Figure 5. MBE encoder proposed in [15]

In this brief, for our proposed MBE multiplier, we combine the two methods for generating the last partial product row. For the generation of the first two least significant bits of last partial product row, we are using second method i.e. , searching for right most '1' in the two least significant bits and replace the bits according the Table 2. The corresponding circuits to generate p_{i1}, $\overline{si}$ for the proposed multiplier are depicted in Figure 6, Figure 7.

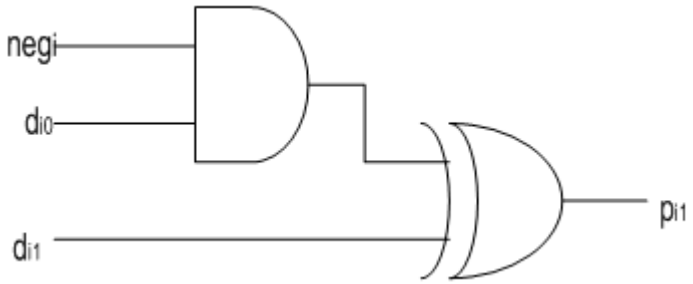

Figure 6. Proposed circuit to generate p_{i1}

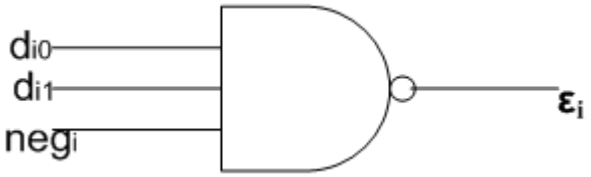

Figure 7. Proposed circuit to generate ε_i

Table. 2. Proposed MBE multiplier truth table for generation of partial product bits $p_{i1}\,p_{i0}$

neg_i	d_{i1}	d_{i0}	P_{i1}	P_{i0}	ε_i
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	0	1	1

Where ε_i is the carry bit and i=n/2-1. If any carry is generated, since the weight of ε_i is $2^{n,}$ which is equal to the weight of sign extension bit s_0 at bit position n, ε_i can be incorporated with the sign extension bits of first partial product row. So that the carry bit do not propagate up to the $2n^{th}$ bit position. Since the maximal value of sign extension bits $\bar{s}_0 s_0 s_0$ in the first partial product row is 100 so that the addition of these bits with ε_i will never produce an overflow. The total conversion of sign extension bits proposed in [1] is shown in Table 3 and corresponding circuit is given in Figure 8.

Table 3.Truth table for new sign extension bits

$\bar{S}_0$	S_0	$S_0\varepsilon_i$		b_2	b_1	b_0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0

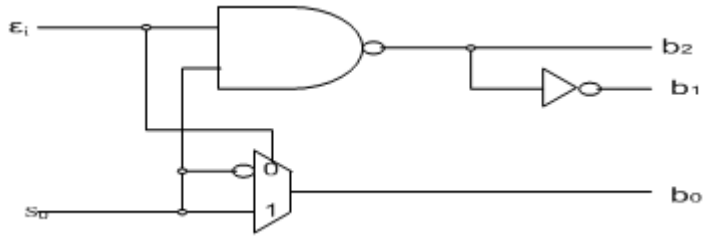

Figure 8. Circuit to generate $b_2b_1b_0$ proposed in [1]

3. SIMULATION RESULTS AND DISCUSSION

The simulation results for different numbers of bits for proposed modified Booth's multipliers for regular partial product array is shown in Figure. 9, 10, 11 and 12 below. Simulation results for 8-bit proposed multipliers for generating regular partial product array.

The partial products of multipliers are generally generated by using two-input AND gates or a MBE algorithm [7]–[9]. The latter has widely been adopted in parallel multipliers since it can reduce the number of partial product rows to be added by half, thus reducing the size and enhancing the speed of the reduction tree. However, the conventional MBE algorithm generates *n*/2 + 1 partial product rows rather than *n*/2 due to the extra partial product bit (*negative* bit) at the

least significant bit position of each partial product row for negative encoding, leading to an irregular partial product array and a complex reduction tree. Some approaches [1]-[3] have been proposed to generate more regular partial product arrays, for the MBE multipliers. The authors in [3] added the least significant bit of each partial product row with negative bit in advance, but the method does not remove the additional partial product row. In the proposed work in [2] the authors proposed a method to directly generates the two's complements of a negative row, but it requires extra hardware. In [1] author almost overcome the above problems by extending the methods proposed in [2] and [3]. The carry bit generated at the last partial product row in [3] is incorporating into the sign extension bits of first partial product row. But the proposed circuits were not satisfying all the conditions. So we redesigned the circuits by a simple method for reduction of last negative bit.

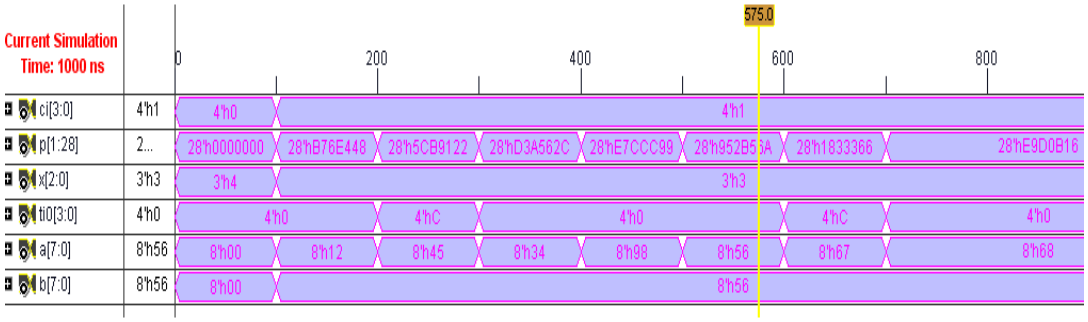

Figure 9. Simultion results for proposed8 x 8 multiplier PPA generation

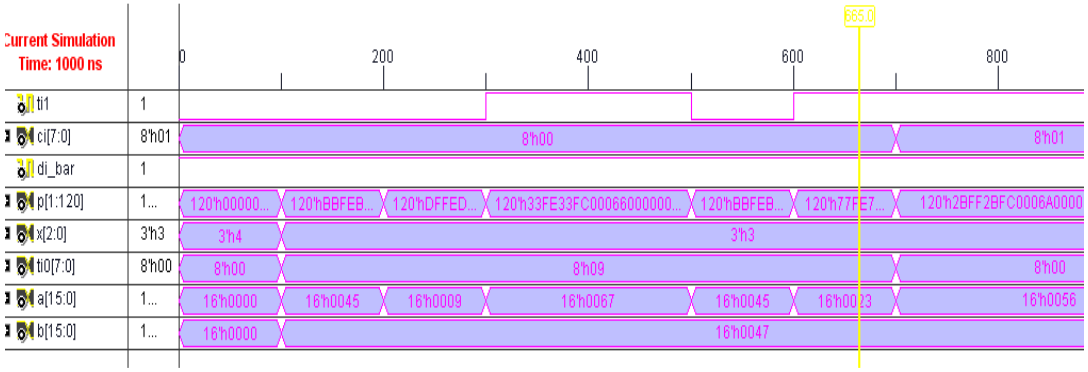

Figure10. Simultion results for proposed16 x 16 multiplier PPA generation

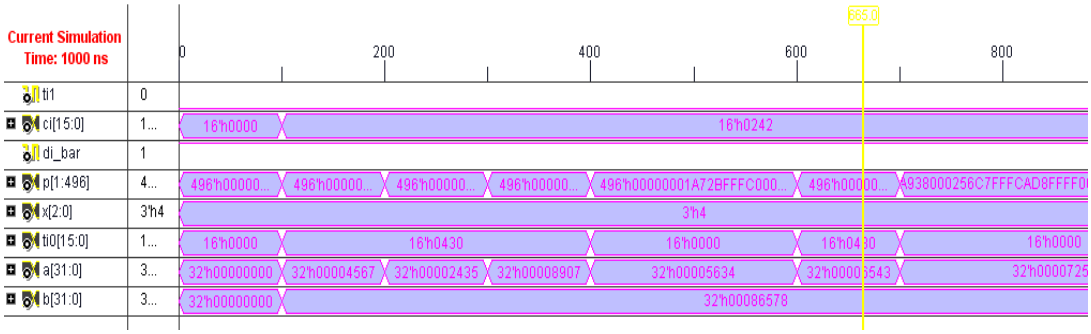

Figure 11. Simultion results for proposed32 x 32 multiplier PPA generation

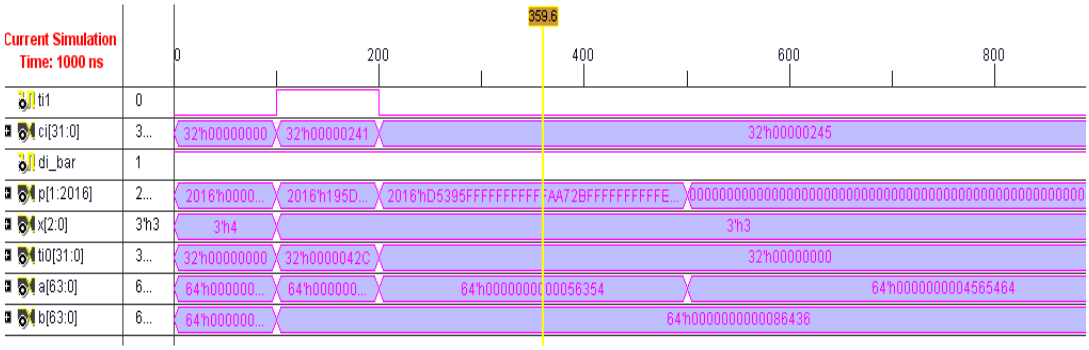

Figure 12. Simultion results for proposed 64 x 64 multiplier PPA generation

For comparison, we have implemented the multiplier proposed in [1]. For the implementation of the other partial product except for a few of partial product bits that are generated by different schemes to regularize the partial product array, the other partial product bits are generated using the proposed MBE selector circuits for all the multipliers. These multipliers are modelled in verilog HDL and synthesized by using synopsys design compiler with Artisan TSMC 90nm technology.

Table 4 : Experimental Results of Generation of Partial Product Bits

	Input (n-bit)	Area (μm^2)	Power (mw)	Delay (ns)
Ref_1	8	1046	0.306	0.68
	16	3550	1.176	0.72
	32	14029	4.328	0.86
	64	60918	19.007	5.62
Proposed	8	947	0.295	0.64
	16	3259	1.0796	0.71
	32	3988	4.254	0.84
	64	58275	19.007	5.61

The implementation results in Table 5 demonstrated that the improved reduction in area 5.9%, power 3.2%, and the delay is 0.5% as compared to proposed multiplier in [1] for8 x 8 multiplier. We can also observe that achievable improvement for 16 x 16 multipliers in area, power, and delay are 4.0%, 2.3%, and 0.3% respectively. For the comparison purpose for the generation of partial products for higher order bits we implemented up to 64 x 64 bit multiplier. As shown in Table 4. It shows that for higher order multipliers the percentage improvement with respect to the area, delay, and power is respectable.

Table 5 : Synthesis Report of Proposed Multiplier

	Input (n-bit)	Area (μm^2)	Power (mw)	Delay (ns)
Ref_1	8	2868	1.186	3.80
	16	10264	5.923	5.34
Proposed	8	2696	1.148	3.78
	16	10325	5.785	5.32

4. CONCLUSION

Multiplication is a frequently encountered operation, especially in signal processing applications. So the development of a multiplier is vital for applications in portable mobile devices such as personal multimedia players, cellular phones, digital cam coders and digital cameras. Many designs have been proposed for Booth encoder and selector logic using CMOS over the past decades. But those designs when implemented in CMOS resulted in higher transistor count. In our research, Booth encoder and selector logic occupies one third of the entire multiplier architecture. So careful optimization of these logic parts will result in a considerable reduction of hardware.

We proposed new circuits for removing the extra partial product array because of negative compensation bit for negative coding. The results obtained here are compared to proposed results in [1]. It shows that proposed multipliers are well regularly structured and complexity of the design is reduced. Also observed that the reduction in area, power, delay are 5.9%, 3.2%, 0.5% respectively for 8 x 8 multipliers and it is 4.0%, 2.3%, 0.3% for 16 x 16 bit multipliers. Apart from above we also presented the new circuit for MBE selector.

REFERENCES

[1] S. R. Kuang, J. P. Wang, C.Y. Guo, "Modified Booth multipliers with a regular partial product array," IEEE Trans.CircuitsSyst.II , vol. 56, pp.404 – 408, May 2009.

[2] W. C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, no. 7, pp. 692–701, Jul. 2000.

[3] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," IEEE Trans. Comput., vol. 55, no. 10, pp. 1253–1258, Oct. 2006.

[4] C. S. Wallace, "A suggestion for parallel multipliers," IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[5] O. Hasan and S. Kort, "Automated formal synthesis of Wallace tree multipliers," in Proc. 50th Midwest Symp. Circuits Syst., pp. 293–296, 2007.

[6] J. Fadavi-Ardekani, "M × N Booth encoded multiplier generator using optimized Wallace trees," IEEE Trans. Very Large Scale Integr. (VLSI)Syst., vol. 1, no. 2, pp. 120–125, Jun. 1993.

[7] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified Booth algorithm," IEEE Trans. Circuits Syst. II, vol. 47, no. 9, pp. 902–908, Sep. 2000.

[8] K. Choi and M. Song, "Design of a high performance 32 × 32-bit multiplier with a novel sign select Booth encoder," in Proc. IEEE Int. Symp.Circuits Syst., vol. 2, pp. 701–704, 2001.

[9] Y. E. Kim, J. O. Yoon, K. J. Cho, J. G. Chung, S. I. Cho, and S. S. Choi, "Efficient design of modified Booth multipliers for predetermined coefficients," in Proc. IEEE Int. Symp. Circuits Syst., pp. 2717–2720, 2006.

[10] O. Salomon, J.-M. Green, and H. Klar, "General algorithms for a simplified addition of 2's complement numbers," IEEE J. Solid-State Circuits, vol. 30, no. 7, pp. 839–844, Jul. 1995.

[11] E. de Angel and E. E. Swartzlander, Jr., "Low power parallel multipliers," in Workshop VLSI Signal Process. IX, pp. 199–208, 1996.

[12] A. A. Farooqui and V. G. Oklobdzija, "General data–path organization of a MAC unit for VLSI implementation of DSP processors," in Proc. IEEEInt. Symp. Circuits Syst., vol. 2, pp. 260–263, 1998.

[13] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, "Design of high-speed low-power 3–2 counter and 4–2 compressor for fast multipliers," Electron. Lett., vol. 34, no. 4, pp. 341–343, Feb. 1998.

[14] C.-H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4–2 and 5–2 compressors for fast arithmetic circuits," IEEE Trans. CircuitsSyst. I, Reg. Papers, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

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

[16] CIC Referenced Flow for Cell-based IC Design, 2008, Taiwan: Chip Implementation Center, CIC. Document no. CIC-DSD-RD-08-01.

[17] G.Goto et al., "A4.1ns compact 54 X54-b Multiplier Utilizing Sign-select Booth Encoders," IEEE J. Solid-state Circuits.,vol. 32, no. 11, pp. 1,676-1,682, Nov. 1997.

[18] P. F. STELLING, C. U. Martel, V. G. Oklobdzija, and R. Ravi, "Optimal Circuits for parallel Multipliers," IEEE Trans. Computers., vol. 47, no. 3, pp. 273-285, Mar. 1998.

[19] J. Cavanagh, "Digital Design and Verilog-HDL Fundamentals", CRC Press Taylor and Francis Group 2008.

[20] C. Senthilpari, A. K. Singh, K. Diwakar, "Design of a low power, high Performance, 8 × 8 bit multiplier using a Shannon-based adder cell", Microelectronics Journal, Vol. 39, Issue 5, pp.812-821, May 2008.

[21] William Stallings, "Cryptography and Network Security Principles and Practices", Fourth Edition, Prentice Hall, 2005.

[22] Shu Lin, Daniel J. Costello, "Error Control Coding: Fundamentals and Applications", Prentice Hall, 1983.

[23] Jean-Pierre Deschamps, Gery Jean Antoine Bioul, Gustavo D. Sutter, "Synthesis of Arithmetic Circuits- FPGA, ASIC, and Embedded Systems", A John Wiley & Sons, Inc., 2006.

[24] www.xilinx.com/

[25] https: //solvnet.synopsys.com