Volume 3, Issue 7, July 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

VHDL Implementation of Low-Power Sign and Unsigned 5-Bit Multiplier

A. M. Borkar[1], Mr. A. K. Sharma[4], Mr. Y. M. Gaidhane[5]
[1]Lecturer, ECE Dept., DBACER,
Nagpur, Maharashtra, India,

N. S. Panchbudhe[2], Mr. N. D. Bhomle[3],
[2]Lecturer, EN Dept., DBACER,
Nagpur, Maharashtra, India,

Abstract— Low power parallel array multiplier is proposed for both unsigned and two's complement signed multiplication. Modified Baugh-Wooley multiplier is further modified and if input numbers are not in two's complement form, proposed method makes the calculation of two's complement of the number redundant, thus reducing delay. Also power consumption has been found to be less than that of modified Baugh-Wooley multiplier.

Keywords---

I. Introduction

Multipliers are one of the most important arithmetic units in microprocessors and DSPs and also a major source of power dissipation. Reducing the power dissipation of multipliers is key to satisfying the overall power budget of various digital circuits and systems. Power consumed by multipliers can be lowered at various levels of the design hierarchy, from algorithms to architectures to circuits, and devices. Various algorithms and multiplier schemes have been proposed till date including Hoffman et al. [1], Burton and Noaks [2], De Mori [3], and Guilt [4] for positive numbers, and Baugh and Wooley [5] and Hwang [6] for numbers in two's complement form. References [7-9] give a good insight into the problem and design optimizations at all the hierarchy levels. In this paper, we focus on power reduction for both unsigned and signed multipliers. For Signed multiplier, the modified Baugh-Wooley algorithm (Fig. 1(b) and 2) is extended to obtain a power efficient multiplier. Inputs are the inverted bits of two's complement representation.

II. UNSIGNED PARALLEL ARRAY MULTIPLIER

The basic process of binary array multiplication involves the AND operation of multiplicand and multiplier bits and subsequent addition as shown in Fig. 1(a) for a 5 $*$ 5 multiplier. NOR gates are used instead of AND in accordance with the DeMorgan's Law:

$$\mathbf{A.B = (A' + B')'} \qquad \textbf{(1)}$$

From (1), it is clear that if NOR gates are used, the inputs have to be complimented. While it takes 6 transistors to build AND/OR gate, only 4 transistors are used for NOR/NAND gate. Also, AND gate neither has an extra delay of 1T compared to NOR gate.

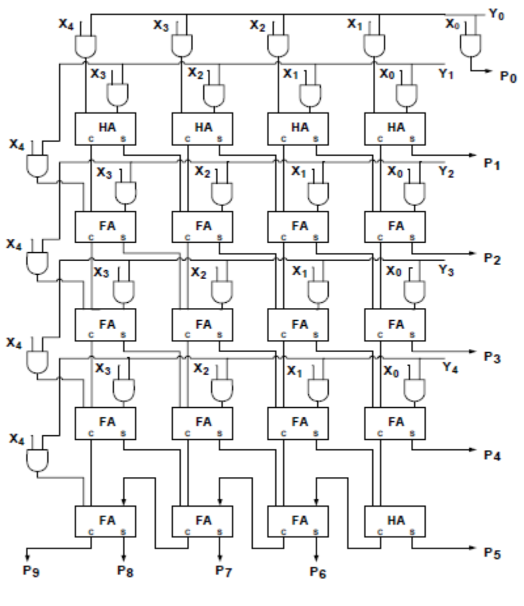


Figure 1(a). Conventional Unsigned Array Multiplier

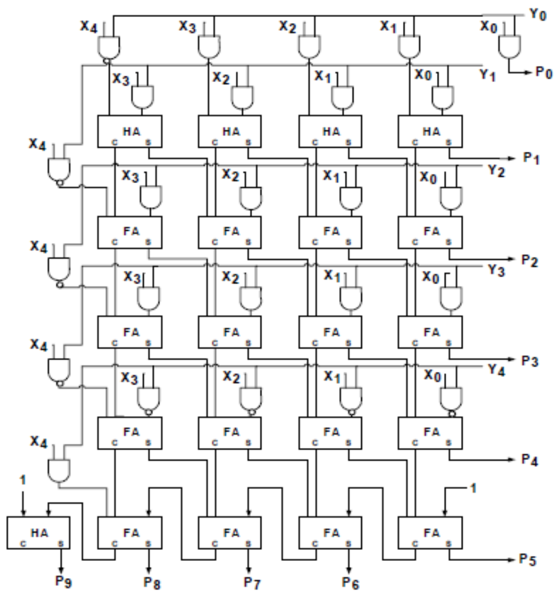


Figure 1(b). Modified Baugh-Wooley two's complement signed multiplier [7]

								y_{m-1}		.	.	.		y_4		y_3	y_2	y_1	y_0
										x_{n-1}		.	.		.	x_3	x_2	x_1	x_0
								1		x_0y_{m-2}	.	.	.	x_0y_4		x_0y_3	x_0y_2	x_0y_1	x_0y_0
								x_1y_{m-2}	.	.	.	x_1y_4		x_1y_3		x_1y_2	x_1y_1	x_1y_0	
							x_2y_{m-2}	.	.	.	x_2y_4	x_2y_3		x_2y_2		x_2y_1	x_2y_0		
						.											.		
					.											.			
				.											.				
	$x_{n-1}y_{m-1}$	0	$x_{n-2}y_{m-2}$			.		.		.	$x_{n-2}y_2$	$x_{n-2}y_1$		$x_{n-2}y_0$					
		$(x_{n-1}y_{m-2})'$	$(x_{n-1}y_{m-3})'$		.		.	.		$(x_{n-1}y_2)'$	$(x_{n-1}y_1)'$	$(x_{n-1}y_0)'$							
1		$(x_{n-2}y_{m-1})'$	$(x_{n-3}y_{m-1})'$	.	.	.	$(x_0y_{m-1})'$					1							
P_{m+n-1}	P_{m+n-2}	P_{m+n-3}	P_{m+n-4}	.	.	.	P_{m-1}	.	.	P_{n+1}	P_n	P_{n-1}	.	.	.	P_3	P_2	P_1	P_0

Figure 2. Tabular form for modified Baugh-Wooley two's complement signed multiplier

Thus, for a $m*n$ multiplier, the proposed method introduces $m + n$ extra inverters along with changing $m*n$ AND gates to $m*n$ NOR gates, effectively saving $(m*n - (m + n))$ inverters or $2*(m*n - (m+n))$ transistors (Fig. 2).

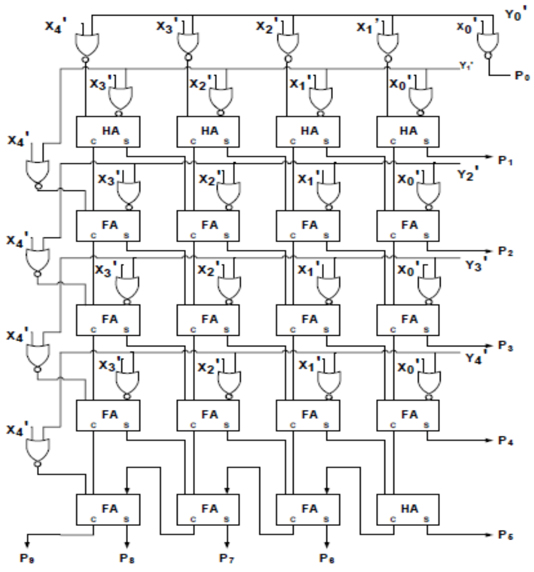


Figure 3. Proposed unsigned array multiplier

III. Parallel Two's Complement Signed Multiplier

For *m***n* parallel two's compliment signed multiplication, m-bit multiplier Yv is represented as

$$Y_v = -y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i$$

:

and n-bit multiplicand Xv is represented as:

$$X_v = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

Each of multiplier bits is ANDed to every multiplicand bit to produce partial products and then summed to form the product Pv,

$$P_v = -p_{m+n-1}2^{m+n-1} + \sum_{i=0}^{m+n-2} p_i 2^i = Y_v X_v$$

$$= \left(-y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i\right)\left(-x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i\right)$$

$$= \left(x_{n-1}y_{m-1}2^{m+n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} x_i y_j 2^{i+j}\right) - \left(\sum_{i=0}^{m-2} x_{n-1} y_i 2^{n-1+i} + \sum_{i=0}^{n-2} y_{m-1} x_i 2^{m-1+i}\right)$$

Modified Baugh-Wooley multiplier uses AND and NAND gates to generate partial products. The tabular form of modified Baugh-Wooley multiplier is shown in Fig. 2. Its architecture is shown in Fig. 1(b). Now we see, out of *m** *n* partial products, majority of the bits ($m*n - (m+n-2)$) are generated as a result of AND operation while only a few ($m+n-2$) are a result of NAND operation between multiplier and multiplicand bits. In the proposed multiplier, all the AND gates are not replaced with NOR gates.

According to DeMorgan's Law, **A.B = (A'+B')'.** **(2)**

This makes it necessary to use inverted inputs (addition of ($m + n$) inverters) and convert the remaining NAND gates to OR gates, as **(A.B)' = (A'+B').**

These changes reduce ($m*n - 2*(m+n-2) - (m+n)$) inverters in the proposed multiplier. (As an example, for a 16x16 multiplier, 164 inverters are reduced.) Thus, in comparison to the modified Baugh-Wooley multiplier (Fig. 1(b)), area is reduced. The probability of getting a 1 in both NOR and AND gate is same (i.e. 1/4) and that for NAND and OR is also same (i.e. 3/4). Thus the switching activity in the proposed multiplier remains the same. Since less number of transistors is used, power dissipation in the proposed multiplier is bound to decrease.

Complemented inputs, there is extra 1T delay. However, if the inputs are not in two's complement form, further modifications can be done to produce the complemented inputs as explained below.

Generation of Inverted Bits:

If the inputs are not in two's complement form, for modified Baugh-Wooley multiplier, they have to be initially converted into two's complement form. For this, first, sign extension to the modulus of the input is done irrespective of the sign of the input. This is done by appending 0s to the most significant side of the number. Then, if the number is positive, all the bits remain the same. If the number is negative, all the bits are complemented and 1 is added to the resulting number. The process is illustrated in example below and shown in Fig. 4(a).

Example: Consider two integer numbers 5 and -3. We shall use a 16 bit word length for illustration. Three bit binary representation of the modulus of given integers is

Multiplier, 5 = 101
Multiplicand, 3 = 011
Sign Extension done to 16 bits,
Multiplicand, 5 = 0000 0000 0000 0101
Multiplier, 3 = 0000 0000 0000 0011
Multiplier is positive so it would remain same i.e. 0000 0000 0000 0101.
Multiplicand is negative, so the bits are first inverted, and then 1 is added to give the two's complement form for -3 i.e. 1111 1111 1111 1101.

Instead of generating inverted bits simply by complementing two's complement form of input, a new way described below is proposed which makes the calculation of two's complement of a negative number redundant. First, sign extension is done to the modulus of the input irrespective of the sign of the input. This is done by appending 0s to the most significant side of the number. Now if the input number is positive, bits are complemented. If it is negative, the complement of its two's compliment form is obtained by adding **(2m - 1)** to the number. (Considering sign extension is done to*m* bits, proof given below.) The process is shown in Fig. 4(b) and illustrated in example given below. The tabular form representation shown in Fig. 2, now looks like as shown in Fig. 6.The block diagram is shown in Fig. 5. Example: Consider two integer numbers 5 and -3. We shall use a 16 bit word length for illustration. 3 bit binary equivalent of the modulus of given integers is ,Multiplier, 5= 101

Multiplicand, 3 = 011

Sign Extension is done to 16 bits,

Multiplicand, 5 = 0000 0000 0000 0101

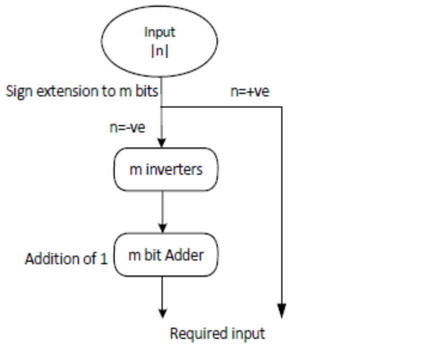


Figure 4(a). Conventional method of generating two's complement form

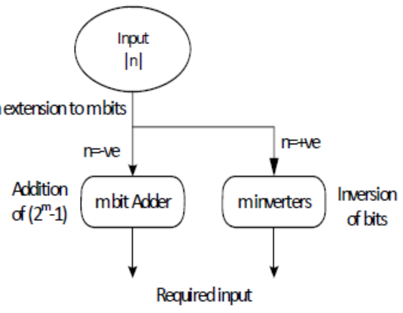


Figure 4(b). Generation of input of proposed method

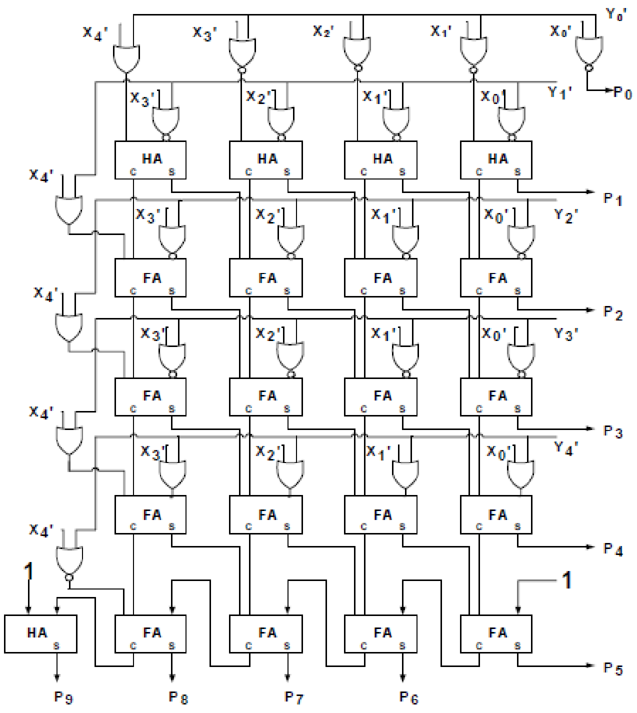
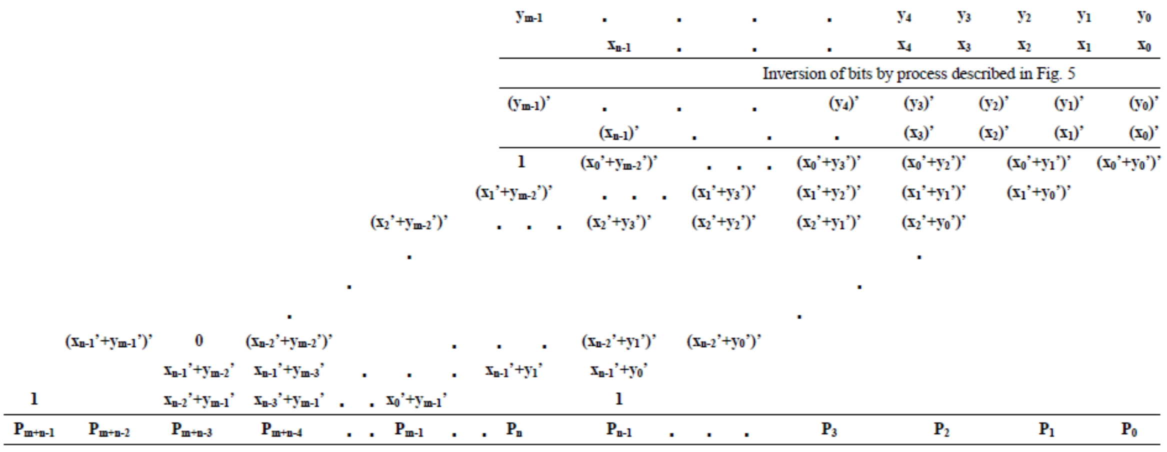


Figure 5. Proposed two's complement signed array multiplier


y_{m-1} . . . . y_4 y_3 y_2 y_1 y_0

x_{n-1} . . . x_4 x_3 x_2 x_1 x_0

Inversion of bits by process described in Fig. 5

$(y_{m-1})'$ . . . $(y_4)'$ $(y_3)'$ $(y_2)'$ $(y_1)'$ $(y_0)'$

$(x_{n-1})'$ . . . $(x_3)'$ $(x_2)'$ $(x_1)'$ $(x_0)'$

1 $(x_0'+y_{m-2}')'$ . . . $(x_0'+y_3')'$ $(x_0'+y_2')'$ $(x_0'+y_1')'$ $(x_0'+y_0')'$

$(x_1'+y_{m-2}')'$ . . . $(x_1'+y_3')'$ $(x_1'+y_2')'$ $(x_1'+y_1')'$ $(x_1'+y_0')'$

$(x_2'+y_{m-2}')'$ . . . $(x_2'+y_3')'$ $(x_2'+y_2')'$ $(x_2'+y_1')'$ $(x_2'+y_0')'$

. .

. .

. .

$(x_{n-1}'+y_{m-1}')'$ 0 $(x_{n-2}'+y_{m-2}')'$ . . . $(x_{n-2}'+y_1')'$ $(x_{n-2}'+y_0')'$

$x_{n-1}'+y_{m-2}'$ $x_{n-1}'+y_{m-3}'$ . . . $x_{n-1}'+y_1'$ $x_{n-1}'+y_0'$

1 $x_{n-2}'+y_{m-1}'$ $x_{n-3}'+y_{m-1}'$ . . $x_0'+y_{m-1}'$ 1

P_{m+n-1} P_{m+n-2} P_{m+n-3} P_{m+n-4} . . P_{m-1} . . P_n P_{n-1} . . . P_3 P_2 P_1 P_0

Figure 6. Tabular form for proposed two's complement signed array multiplier

Multiplier, 3 = 0000 0000 0000 0011

Multiplier is positive so the bits are inverted to give 1111111 1111 1010. Multiplicand is negative, so **(216 - 1)** is added to 0000 00000000 0011 to give the complement of two's complement form for -3 i.e. 0000 0000 0000 0010.

Proof for addition of (2m-1):

As mentioned earlier, required input for proposed multiplier is complement of two's complement form of input. Steps involved in obtaining two's complement form of a number (sign extended to m bits) are:

1) Complement the number.
2) Addition of 1.

As we know that, taking two's complement of a number twice gives the same number. Let **A** is the given m-bit sign extended number and **B** is its two's complement representation. **B'** is required number, then

$A \rightarrow A' \rightarrow (A'+1) = B$
$B \rightarrow B' \rightarrow (B'+1) = A$
$B' = A - 1 = A + (2m - 1)$

Thus for a negative number, calculation of its two's complement form and its complement are taken care of simultaneously avoiding the need to calculate its two's complement form. This not only improves time delay but also power dissipation is reduced.

IV. SIMULATION DETAILS AND RESULTS

The analysis has been carried out on the proposed multipliers by performing simulations on H Spice and compared with the existing multipliers. Simulations are performed for 16x16 bit multipliers at 1.2V and at afrequency of 50 MHz. Results shown in the Table I are for the particular inputs 1010101010101010x
01010010101010101. Similar results can also be obtained for other inputs.\

TABLE I. Power and Delay comparison of conventional and proposed multipliers

[illegible]			
	Conventional	Proposed	Proposed/Conventional
Power(inWatt)	12.2177E-04	9.6482E-04	0.92
Delay (in ns)	1.015	0.968	0.954
[illegible]			
	Modified	Proposed	Proposed/Modified
Power(inwatt)	13.6533E-04	10.5366E-	0.979
Delay (in ns)	1.1478	0.98	0.854

V. Conclusion

In this paper, a new approach for the design of parallel array multipliers has been suggested. AND gates in the existing designs have not been replaced with NOR gates. Where the numbers are not in two's complement form then they are inverted and given as input. Results of the simulation clearly show that the proposed multiplier architecture performs better than the existing modified Baugh-Wooley multiplier.

REFERENCES

[1] J. Hoffman, G. Lacaze, and P. Csillag, *"Iterative Logical Network for Parallel Multiplication"*, Electronics Letters, vol. 4, p. 178, 1968.
[2] P. Burton and D.R. Noaks, *"High-Speed Iterative Multiplier"*, *Electronics Letters*, vol. 4, p. 262, 1968.
[3] R. De Mori, "Suggestion for an IC Fast Parallel Multiplier," *Electronics Letters*, vol. 5, pp. 50-51, Feb. 1969.
[4] H. Guilt, *"Fully Iterative Fast Array for Binary Multiplication"*, *Electronics Letters*, vol. 5, p. 263, 1969.