International Journal of Electronics and Communication Engineering.
ISSN 0974-2166 Volume 6, Number 1 (2013), pp. 9-16

http://www.irphouse.com

A Simple Method to Improve the throughput of A Multiplier

Naga Mani Mendu

Asst. Professor, Dept. of ECE, SRK Institute of technology, Enikepadu, Vijayawada-512108, Andhra Pradesh, India
mani.mendu@gmail.com

Abstract

In this paper a simple method to improve the throughput of multipliers was presented. The method used is the double precision. In this method the operands of the multiplier will be made into narrow width operands. Hence the power dissipation and throughput of the multipliers will be improved. Here we will also describe how to apply this new method to the familiar multipliers such as Booth and Baugh-wooley. The proposed algorithm will be modeled using VHDL. Then the performance of these multipliers is compared with each other.

Keywords Double throughput, reduced power consumption, Double Precision, Modified-Booth, Baugh-Wooley, final adder.

I.INTRODUCTION

MULTIPLICATION is an important function in digital signal processing. For example, it is a needed in digital filtering and Fast Fourier Transform(FFT) processing. In the early stage, multiplier schemes were proposed by Hoffman(1986), Burton and Noaks(1968), De Mori(1969), and Guilt(1969) for positive numbers, and by Baugh-Wooley(1973) and Hwang(1979) for numbers in two's complement form. The performance requirements for the multipliers are increasing day by day and make it challenging to implement multipliers that are efficient in terms of throughput, delay, power, and area for a wide range of bit widths. A number of studies have been done on the bit widths used by the instructions used for integer applications. In most of the cases the bit width of the instructions used are the instructions with bit width less than 16 bits. Hence such operations are called narrow width operations. These narrow width operations are used to improve the throughput. This can be achieved by computing a number of narrow width operations in parallel on a full width data path.

There is a technique which is used for the narrow width operations to be done on a full width data path.

Double precision:

The technique used is the double precision. Precision is defined as the number of digits used to represent a number. Twin precision is nothing but using two different precisions with in a single multiplication. That is dividing the bit width of the operands into two equal bit widths. That means if we consider an operand with a precision of octal and we are using the twin precision technique then the precision of the operand is divided into two quad operands. This technique provides the same power reduction as the operand guarding and the area overhead is also reduced. But it is having a small delay penalty.

The twin precision technique can be applied for array multipliers, Baugh-Wooley multipliers and Modified-Booth multipliers to obtain double throughput. The precision of the operands will effect the throughput a lot i.e. for example if the precision is 4 then any number is represented by using 4 digits and this can be understood clearly by observing the below table:

Table1: Different types of Precisions

Precision	Number of digits	Example
Quad	4	9 = 1001
Octal	8	9 = 00001001
Hex	16	9=0000000000001001

II. TYPES OF MULTIPLIERS

A. Array multiplier:

The binary multiplication consists of two operands. First operand is called the multiplier and the second operand called the multiplicand. Each bit of the multiplier is multiplied with all the bits of the multiplicand to produce the partial products. The first row of partial products is produced by multiplying the first bit of multiplier with the multiplicand. The second row of partial products is produced by multiplying the second bit of the multiplier with the multiplicand. In this way all the partial products are generated. The illustration of an 8-bit array multiplier is shown in the below figure. Each partial product is shifted one position according to the position of the bit in the multiplier.

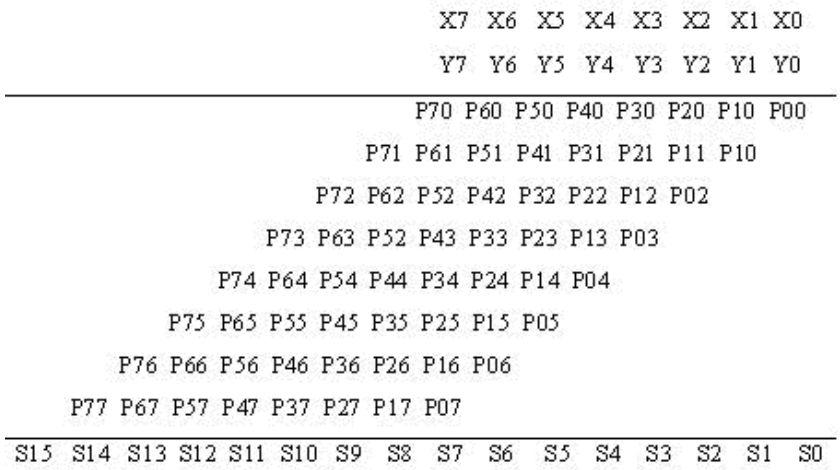

Fig. 1: Illustration of an Unsigned multiplier

All the partial products produced are together known as the partial product array. The final result is produced by adding all the bits present in each column. If the length of two binary numbers used in array multiplier A and B are m bits and n bits each, the number of summands produced will be mxn. An n x n multiplier requires n (n-2) full adders, n half-adders and n^2 AND gates. Array multipliers consume more power and occupy more area but they are easy. It requires large number of gates because of which area is increased. Thus, it is a fast multiplier but hardware complexity is high. The length of the final product will be double the length of the operands. That is if the length of the both the operands is 8 bit then the final product will be 16 bit length.

***B. Array multiplier using double precision*:**

Fig.2., shows the way of computing the two 4 bit multiplications simultaneously. The partial products shown in black and the partial products shown in white are the useful bits for the final results. But the remaining bits which are shown in gray color are not useful for any of the 4 bit multiplication.

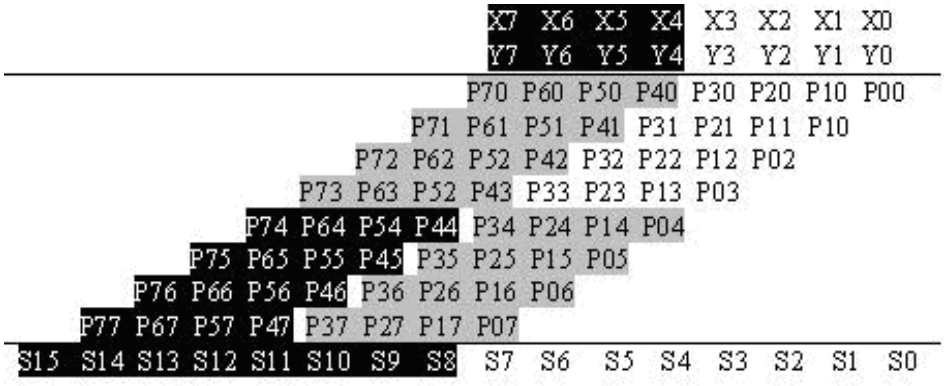

Fig. 2: Illustration of an unsigned multiplication, where a 4-bit multiplication shown in black, is computed in parallel with a second 4-bit multiplication, shown in white.

From the above figure we can observe that the eight bit of the product S7 is taken as the carry bit for the first 4 bit multiplication. So it is not used with the second 4 bit multiplication. Each bit of the partial product is obtained by using an AND gate. As can be seen from fig.2 only the bits which are in white color and black color are used for the multiplication so the remaining partial products can be made zero. So it is possible to perform two multiplications within the same partial-product array. The unwanted partial products can be made zero by using three input AND gates.

From above figure we can observe that the length of the two multiplications is equal. But it is not the mandatory. Each multiplication can have different precision. But the total precision must be less than or equal to the full precision, but not more than the full precision. That is

$$N_{FULL} \geq N_{LSP} + N_{MSP} \quad (1)$$

Where
N_{FULL} = Full precision
N_{LSP} = Precision of the least significant product.
N_{MSP} = Precision of the most significant product.

C. Baugh-Wooley Multiplier:

The array multiplier is most suitable for unsigned multiplication only, but signed multiplication the number of partial products and the length of the partial products will be very high. So an algorithm known as the Baugh-Wooley Algorithm was used for signed multiplication. Fig3 illustrates the algorithm for an 8-bit case. The creation of the reorganized partial-product array comprises three steps:

i. The most significant bit of the first N-1 partial products and all the bits of last partial product except the most significant bit are negated.
ii. A constant one is added to the Nth column.
iii. The most significant bit of the result is negated.

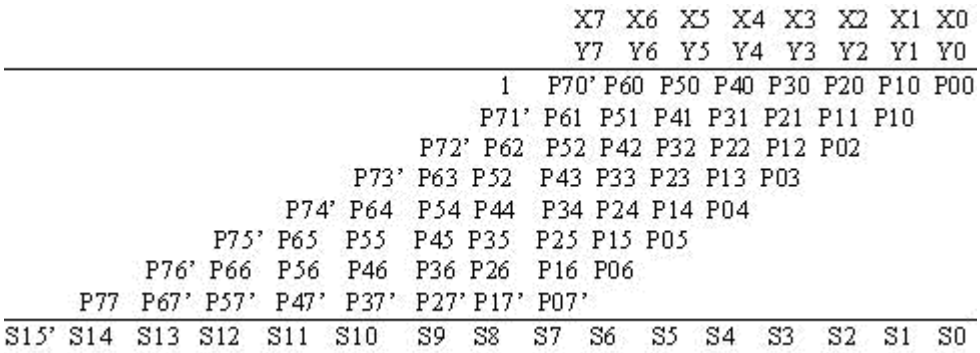

Fig.3: Illustration of signed 8-bit Multiplication using Baugh-Wooley Algorithm.

From the above figure we observe which bits of the partial products are negated. The eight bit appended to the first partial product is the sign bit.

D. Baugh-Wooley Multiplier using double-precision technique:

Applying double precision for Baugh-Wooley algorithm is not as easy as the array multiplier. In Baugh-Wooley algorithm some more changes are required to be done. As shown in the fig4, the fourth bit the first three partial products is negated and the first three bits of the fourth partial product is to be negated. When the double precision is applied the partial products which are in gray color are made zero. The same procedure is followed for the partial products which are in black color.

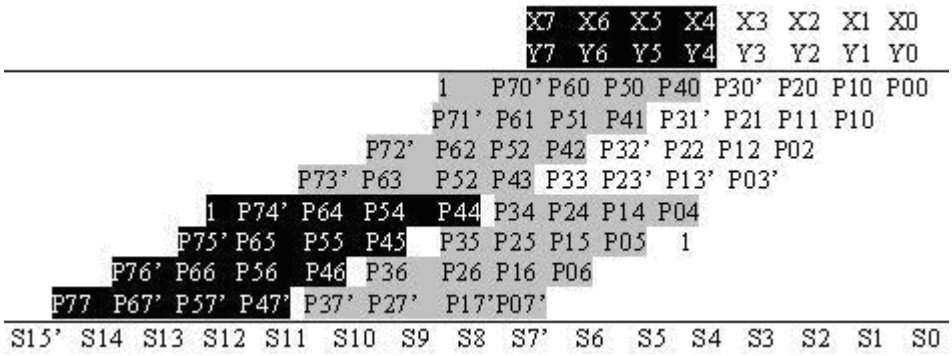

Fig. 4: Illustration of a signed multiplication using Baugh-Wooley Algorithm, where a 4-bit multiplication shown in black, is computed in parallel with a second 4-bit multiplication, shown in white.

After calculating the final product the eight bit and the sixteenth bit of the product are negated to obtain the two separate products of the N/2 multipliers.

E. Modified Booth Multiplier:

The Baugh-Wooley algorithm can be used for both signed and unsigned operand multiplication. But the number of partial p[roducts generated will be very high. This can be overcome by using the Modified-Booth algorithm. In this algorithm the recoded partial products are generated by grouping the bits in the Multiplier. Depending on the obtained group the partial product is generated.

Let us consider an 8-bit multiplier " . . . . . ". This is the dot representation of the operand. One more zero is appended at the right side of the multiplier and hence the multiplier will become ". . . . . 0".

Now grouping is done as follows. Consider the multiplier from the LSB side and group three bits and then start from the last bit of the previous group and end with the third bit from that bit.

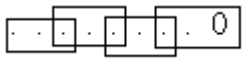

Fig. 5: Grouping of bits in the multiplier for encoding the partial products

The below table gives the operation to be performed on the multiplicand to generate the partial product depending upon the groups obtained from the multiplier.

Table 1: Generation of Partial products

Block	Operation on the multiplicand of partial product
000	No arithmetic operation, just add zeros to the product
001	1xMultiplicand, i.e. Add multiplicand to the product.
010	1xMultiplicand, i.e. Add multiplicand to the product.
011	2xMultiplicand. Shift left multiplicand by one position and add it to the product.
100	-2xMultiplicand. Take twos compliment of the multiplicand, shift left it by one position and add it to the product.
101	-1xMultiplicand. Take twos compliment of the multiplicand and add it to the product.
110	-1xMultiplicand. Take twos compliment of the multiplicand and add it to the product.
111	No operation, Just add multiplicand to the product.

Each partial product is generated by considering three bits of the multiplier. Therefore only half the number of partial products is obtained.

This way of generating partial products is easy and is used frequently. The sign bit is nothing but the MSB of the partial product and it is generated by just EXORing the MSB bit with zero. Second sign bit is obtained by EX-ORing the first sign bit with zero, third sign bit is obtained by EX-ORing second sign bit with zero and soon until the last sign bit. But applying twin precision for this scheme is not an efficient method, hence another method of generating partial products and sign bits is followed which makes the twin precision easy.

The mechanism of performing the multiplication using the Modified-Booth algorithm was given in the below figure:

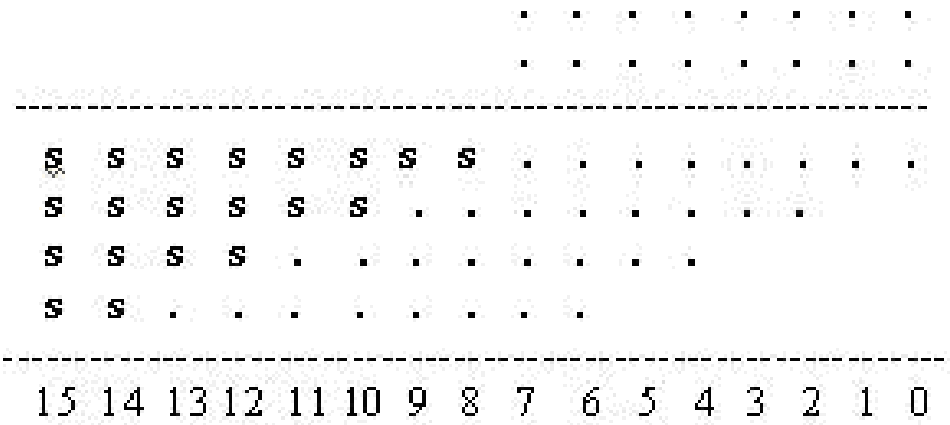

Fig. 6:Illustration of Signed 8-bit Multiplication using Modified Booth Algorithm.

F. Modified Booth Multiplier using Double precision:

It is not a straight forward method to implement the modified booth multiplier using twin precision technique because all the partial products are not computed the same way as the full precision technique.

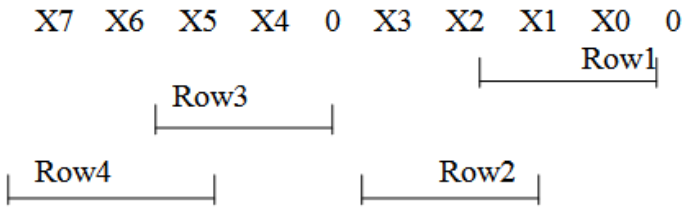

Fig 7: Encoding of 8-bit Modified Booth Algorithm

An Modified Booth multiplier works internally with two's complement representation of the partial products.

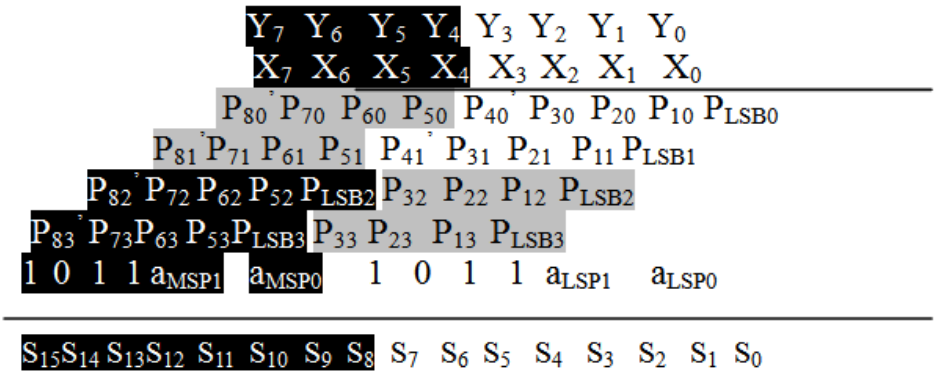

Fig. 8. Illustration of a signed multiplication using Modofied-Booth Algorithm, where a 4-bit multiplication shown in black, is computed in parallel with a second 4-bit multiplication, shown in white.

III.COMPARISON OF THE MULTIPLIERS

The comparison of the algorithms in terms of the number of partial products, Number of adders used and the number of shifts required is presented in the below table. It is clear that the Modified Booth algorithm is requiring less number of adders and less number of shifts. The number of partial products is reduced to half.

Table. 2. Comparison of the multipliers

Algorithm	Number of partial products	Number of adders	Number of shifts required	Final Adder
General Array multiplier	8	64	7	Ripple carry adder
Baugh-Wooley	8	64	7	Ripple carry adder
Modified Booth	4	36	6	Carry save adder

The plot for the above comparison table is obtained as shown below:

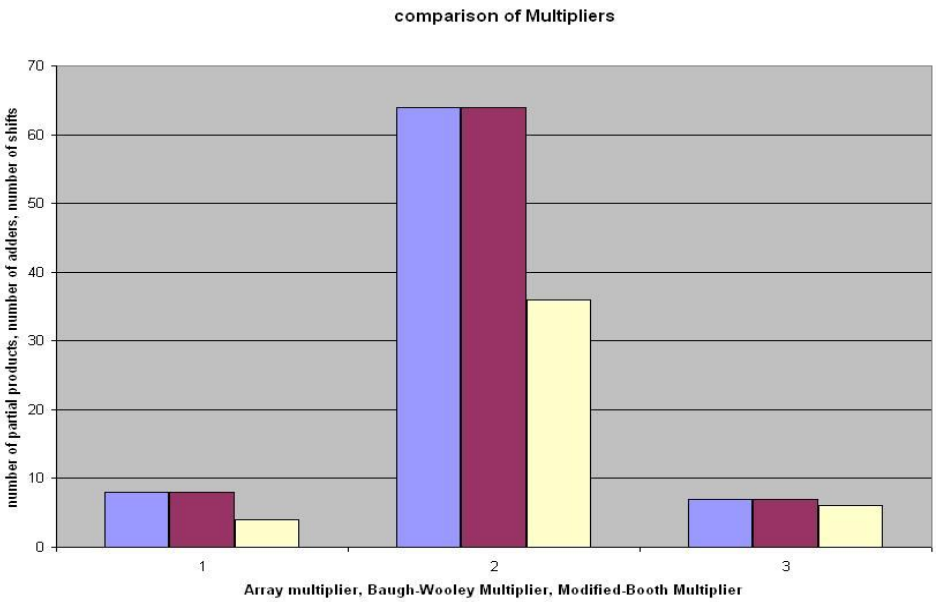

Fig. 8. Comparison of the Multipliers

IV.CONCLUSION

Array multiplier can also be used for both signed and unsigned multiplications. But the Modified Booth and Baugh-Wooley multipliers are performing better than the array. Among the three algorithms Modified Booth Multiplier is the best one.

REFERENCES

[1] M. Sjalander, P.Larsson-Edefors, "Multiplication Acceleration through Twin Precision", IEEE Trans. VLSI Systems, vol. 17, no.9, pp. 1233-1246, September 2009.

[2] A. D. Booth, "A signed binary multiplication technique," Quarterly J.Mechan. Appl. Math., vol. 4, no. 2, pp. 236–240, 1951.

[3] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, "Multiplier reduction tree with logarithmic logic depth and regular Connectivity," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2006, pp.4–8.

[4] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," IEEE Trans. Comput., vol. 22, pp. 1045–1047, Dec. 1973

[5] M. Sjalander, H. Eriksson, and P. Larsson-Edefors, *"An efficient twin-precision multiplier,"* in *Proc. IEEE Int. Conf. Comput. Des.* Oct.2004,pp. 30-33.