General Data-Path Organization of a MAC unit for VLSI Implementation of DSP Processors

Aamir A. Farooqui[1], Vojin G. Oklobdzija[2]
[1]Department of Electrical and Computer Engineering,
University of California, Davis, CA 95616.
e-mail: aamirf@ece.ucdavis.edu.
[2]Integration Berkeley, California.
email: vojin@nuc.berkeley.edu.
http://www.integr.com.

Abstract

This paper describes the data-path and VLSI implementation of a 32x32 bit signed/unsigned multiply accumulate (MAC) unit. In this design we have solved the problem of dealing with signed and unsigned numbers simultaneously, using modified Booth algorithm. This MAC unit can perform 32x32, 32x16, and two 16x16 multiplications, on signed/unsigned operands with a throughput of 2, 1, and 1 cycle, respectively. The Booth encoding technique reduces the number of partial products (PP) by half. Further increase in speed is achieved by using Three Dimensional reduction Method (TDM) to add the partial products. Special circuitry has been designed to accommodate sign/unsigned operands and to deal with sign extension. Modified Booth algorithm coupled with TDM and sign correction circuitry results in a multiplier, with a delay (partial product addition) equivalent to 6 XOR gates. The MAC unit has been modeled in VHDL, and it implements an algorithm which makes this data path organization fast and efficient in silicon.

1. Introduction

Nearly 90% of the multipliers are designed using Booth algorithm [1,2,3,4], because Booth encoding [5,6] reduces the number of PP by half and is good for Signed (2's complement) operands, but it has two drawbacks:

1. It requires the sign extension of the partial products to the m+n (where m = number of bits of the multiplier and n = number of bits the multiplicand) bit position.
2. It requires one extra bit to handle unsigned operands.

In this paper we have presented a design to deal with sign extension and accommodate sign/unsigned operands using Booth algorithm. The paper is organized as follows; Section 2 gives the Architectural overview of the MAC unit. Section 3 discusses the design of a 16x16 multiplier, after the brief review of Booth algorithm, Sign extension circuitry and TDM is described. In Section 4 the VHDL implementation and simulation results are presented.

2. Architectural Overview of MAC

Fig. 1 shows the complete overview of the MAC unit. The main building blocks are two 16x16-bit Booth multipliers and a 64-bit carry propagate adder. The inputs are applied using 32-bit registers A (multiplier) and B (multiplicand), while the result is stored in 64-bit register C. To balance the pipeline stages of the MAC, the 16x16 multiplier result is produced in carry save form and finally added in the second stage of the MAC. This reduces the clock cycle and power consumption of the MAC (fewer glitches due to a carry propagate adder) unit, at the cost of larger area used by pipeline registers.

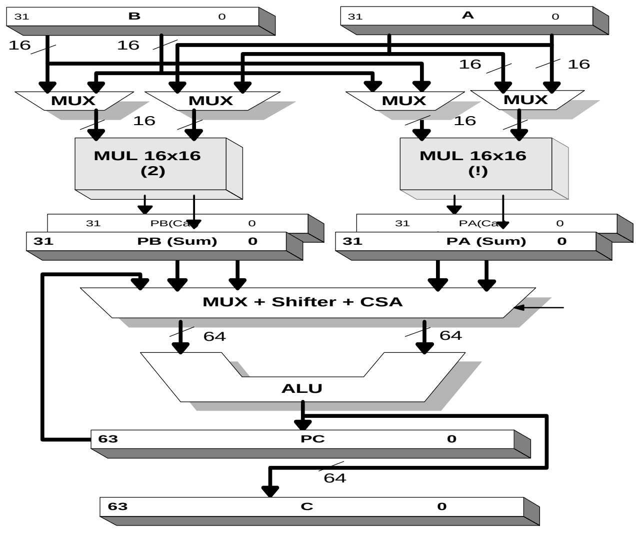


***Fig. 1**. Block diagram of the MAC unit.*

2.1 Mac operation

The operation of the MAC unit can easily be explained by taking the example of 32x32 signed multiplication. Let A1, B1 be the most significant 16 bits and A0, B0 be the least significant 16 bits of operand A and B, respectively. The 32x32 multiplication requires following four multiplications:

1. A0xB0 unsigned x unsigned multiplication,
2. A1xB0 signed x unsigned multiplication,
3. A0xB1 unsigned x signed multiplication, and

4. A1xB1 signed x signed multiplication.

The Mac operation during the three clock cycles (see Fig. 2) is as follows:

- During the first clock cycle, the products A1xB0 and A0xB1 are calculated by 16 bit multipliers 1 and 2 respectively (see Fig. 1). At the end of the first clock cycle the sum and carry result of these multiplications are stored in pipeline registers PA and PB.
- During the second clock cycle, the products A0xB0 and A1xB1 are calculated by the 16-bit multipliers 1 and 2 respectively in the first pipeline stage. The addition of A1xB0 and A0xB1 is performed in the second pipeline stage of the multiplier. At the end of second cycle the results of A0xB0 and A0xB1 are stored in registers PA and PB, and the sum of A1xB0 and A0xB1 is stored in second pipeline register PC.
- In the third cycle, the addition of concatenated PA and PB is performed with the 16-bit left shifted contents of register PC to obtain the 64 bit result.

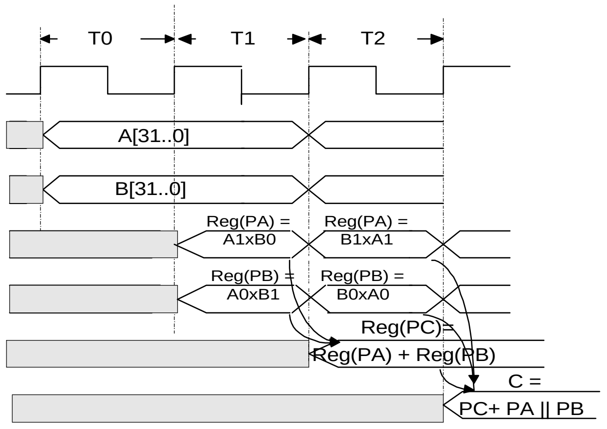


Fig. 2. *32x32 Multiplication operation.*

In the following section the design of the 16x16 bit multiplier is explained in detailed.

3. Basic Multiplier Design

Fig. 3 shows the block diagram of the 16x16 bit multiplier. The basic building blocks of this multiplier are modified Booth encoder and selector [1,4], TDM adder array for the carry free addition of the Partial products [7], and the Sign Correction circuit. The inputs to the multiplier are 16-bit multiplicand and multiplier, and Ub & Ua signals for the signed/unsigned multiplier and multiplicand, respectively. These signals are 0(1) when the operands are signed(unsigned).

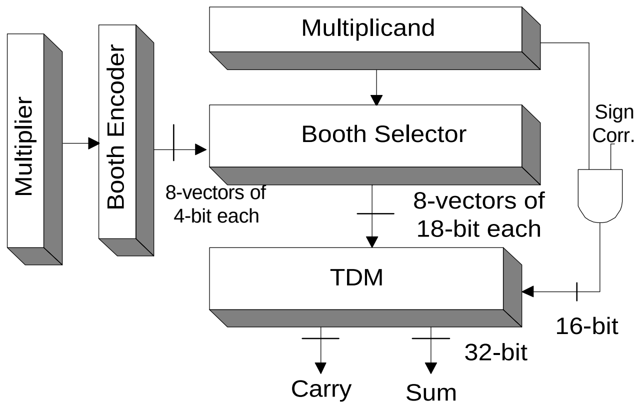


Fig. 3. *16x16 Signed / Unsigned Multiplier block diagram.*

3.1 *Modified Booth Recording*

The modified Booth algorithm scans an n-bit 2's complement multiplier A by the following equation [1,3].

$$\begin{aligned} A &= (-1)\, a_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \\ &= \sum_{i=0}^{n/2-1} (a_{2i-1} + a_{2i} + 2a_{2i+1}) 2^{2i} \end{aligned} \tag{1}$$

Where: $a_{-1} = 0$.

Booth algorithm divides the multiplier bits (including the sign bit) into sub strings of three bits, with the adjacent groups sharing the same bit. The Booth encoder scans three overlapped bits of the 16-bit multiplier and depending on these bits, produces four outputs (x1, x2, x-1, x-2, where x1 means multiply by 1, x-2 means multiply by 2, etc.) as shown in Table 1 [1,3].

a_{i+1}	a_i	a_{i-1}	Encoder Output
0	0	0	0
0	0	1	x1
0	1	0	x1
0	1	1	x2
1	0	0	x-2
1	0	1	x-1
1	1	0	x-1
1	1	1	0

Table 1. *Modified Booth Encoding Table, a_{i+1}, a_i, and a_{i-1}, represents the three bits of the multiplier A*

For m-bits m/2 Booth encoders and selectors are required, producing m/2 PP. Therefore, eight such encoders are required for 16 bits. For the first encoder a_{i-1} is 0, and for all the other encoders it is the overlapped bit with the previous encoder.

3.2 *Booth Selector*

The 16-bit multiplier is input to eight Booth encoders and the 16-bit multiplicand is applied to eight Booth selectors. Booth encoders control the outputs of the Booth selectors [4]. Each Booth selector produces 18-bit partial product. The Booth selector produces multiplicand times 1, 2, -1, or -2 depending on the output of the Booth encoder. The partial products generated by the Booth selector are then added using TDM.

3.3 Signed Unsigned operands

As we have seen in the above section that the number of Booth encoders and selectors required for m-bits is m/2 and Booth algorithm requires one extra bit to handle unsigned operands. Therefore in Booth multipliers (for number of bits in power of 2), one extra encoder and selector is required, to deal with unsigned numbers. In this paper we have solved this problem in a very simple manner, with minimum hardware and delay. In our design signed and unsigned multiplication is controlled by two signals Ua and Ub.

When the multiplier is unsigned (Ua =1) and the most significant bit (bit # 16) of the multiplier is one then multiplicand is added with the 8-partial products (means multiplication by 1), else a zero is added with the partial products. This can be accomplished by using And gates.

To deal with unsigned multiplicand, the multiplicand is sign extended to 17 bits. The value of this bit is controlled by Ub, when Ub=0 (signed multiplicand) then bit # 17 is the sign extension of the multiplicand else bit # 17 is zero.

3.4 Sign Extension

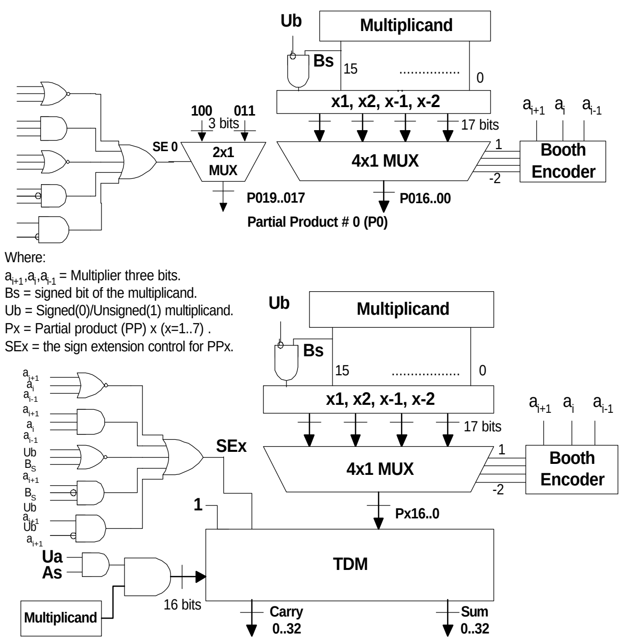
When adding partial products using CSA (carry save adders) each partial product must be sign extended to the m+n binary position. The addition of these extended signs becomes very costly. To prevent the sign extension of the PP, the following technique [8] has been used. In this technique it is supposed that the partial products are negative. Therefore, the sum of all the sign extensions can be pre-calculated using Equation 2 as a constant number = 101010....01011 [1]. If it turns out that a partial product is positive, then a one is added to undo the effect of the earlier assumption. The constant number is added to the (unextended) partial products, starting from the n+1 binary position. To deal with positive PP, a sign extension control bit (SE) is attached to each PP at bit position n+2. If it turns out that a partial product is positive then the sign extension control bit (SE)=1 [1,8].

$$\text{Sum of Negative Signs} = \sum_{i=0}^{m/2-1} \left((-1)2^{n}\right)4^{i} = 2^{n}(-1)\left(\frac{2^{m}-1}{3}\right) \qquad (2)$$

If this technique is applied number of operands to be added together increases (in this case it is 10). This number of operands is reduced by one, by combining the last three bits (011) of the constant 1010..01011 with the first PP and all the others with the rest of PP [8]. In the first PP a multiplexer controlled by SE has been used to select 011 (100) when SE 0 (1). The SE is dependent on sign bit of the multiplicand (Bs), signed/unsigned multiplicand (Ub), a_{i+1}, a_i, and a_{i-1}. The SE is calculated according to Table 2. The Sign extension bit SE is 1 when multiplied by zero i.e., when a_{i+1}, a_{i+1}, and a_{i+1} =111 or 000. The circuit for the generation of SE-bit is shown in Fig. 4.

a_{i+1}	Bs	Ub	SE
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 2. *Truth Table for Sign Extension bit (SE) generation.*



Ub
Multiplicand
Bs
15
0
x1, x2, x-1, x-2
100
011
3 bits
17 bits
SE 0
2x1 MUX
4x1 MUX
Booth Encoder
a_{i+1} a_i a_{i-1}
1
-2
P019..017
P016..00
Partial Product # 0 (P0)
Where:
a_{i+1}, a_i, a_{i-1} = Multiplier three bits.
Bs = signed bit of the multiplicand.
Ub = Signed(0)/Unsigned(1) multiplicand.
Px = Partial product (PP) x (x=1..7) .
SEx = the sign extension control for PPx.
SEx
Px16..0
TDM
Ua
As
Multiplicand
16 bits
Carry 0..32
Sum 0..32

Fig. 4. *Implementation of the 16x16 Signed/Unsigned Multiplier.*

3.5 TDM

Three Dimensional Reduction Method (TDM) is an algorithm for the generation of a parallel multiplier, optimized for speed [7]. The basic idea behind this method is to make proper connections globally so that the delay throughout each path is approximately the same. TDM uses the fact that, the Carry signal of the 1-bit full adder is generated earlier (1 XOR) than the Sum signal. Therefore the long delay path originating from previous adder should be connected to the short delay path of the next one, and so on. In general, this is not always possible since each output function has its unique characteristics and requires specific logic cells in its path. It is feasible to apply this idea to the partial product array using full adders since all of the partial products bits in the same bit position are logically the same and therefore interchangeable (for a detailed discussion please see [7]). By using TDM with Booth encoding, we are able to get fastest multiplier with the PP addition delay equivalent to 6-Xor gates.

4. VLSI Implementation

The MAC unit has been implemented in VHDL, using modular approach i.e., each module of the MAC unit has been implemented and simulated separately and then combined together, to form a MAC unit Fig. 5 shows the simulation results for 32x32 multiplication. The MULTIPLICAND_A1,A0 is the 32-bit multiplicand and MULTIPLIER_B1,B0 is the 32-bit multiplier. The result is produced in REG_C. It is clear from this figure that the first result (00040003 x 00020001= 00000008000A0003) is produced after a delay of 3 cycles. The next result (00080007 x 00060005 = 0000003000520023) is produced after a delay of 2 cycles from the previous result, showing the throughput of 2-cycles for 32-bit operands.

5. Conclusions

In this paper we have presented the design and VLSI implementation of a MAC unit, that can perform 32x32, 32x16, and two 16x16 (signed/signed, signed/unsigned, unsigned/unsigned) multiplications, with a throughput of 2, 1, and 1 cycle, respectively. In this multiplier modified Booth encoding and Three dimensional (TDM) technique has been used, to get the fastest multiplier. Special circuitry has been designed to accommodate sign/unsigned operands and to deal with sign extension. Modified Booth algorithm coupled with TDM and sign correction circuitry results in a multiplier, with a delay (partial product addition) equivalent to 6 XOR gates.

6. Acknowledgments

The Authors acknowledge support of Mr. Hourteillan and Mr. Kirby.

References

1 J. Fadavi-Ardekani. M x N Booth Encoded Multiplier Generator Using Optimized Wallace Trees,. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1(2):120--125, June 1993.

2 P. E. Madrid, B. Millar, and E. E. Swartzlander Jr. Modified Booth Algorithm for High Radix Fixed-Point Multiplication,. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1(2):164--167, June 1993.

3 Xiaoping Hung, Wen-Jung Liu, and Belle W.Y.Wei. A High Performance CMOS, Redundant Binary Multiplication and Accumulation (MAC) Unit,. IEEE Transactions On Circuits and Systems, 41(1):33--39, January 1994.

4 Atsuki Inoue et al. A 4.1ns Compact 54x54b Multiplier Utilizing Sign Select Booth Encoders,. IEEE International Solid-State Circuits Conference, pages 416--417, 1997.

5 Booth. A Signed Binary Multiplication Algorithm,. Quarterly Journal of Mechanics and Applied Mathematics, 4:236--240, 1951.

6 O. L. MacSorley. High Speed arithmetic in binary computers,. Proceedings IRE, 49, January 1961.

7 Vojin G. Oklobdzija, David Villeger, and Simon S. Liu. A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers using an Algorithmic Approach,. IEEE Transactions On Computers, March 1996.

8 Gary W. Bewick. Fast Multiplication Algorithms and Implementation,. Ph.D., Dissertation, Stanford University, February 1994.

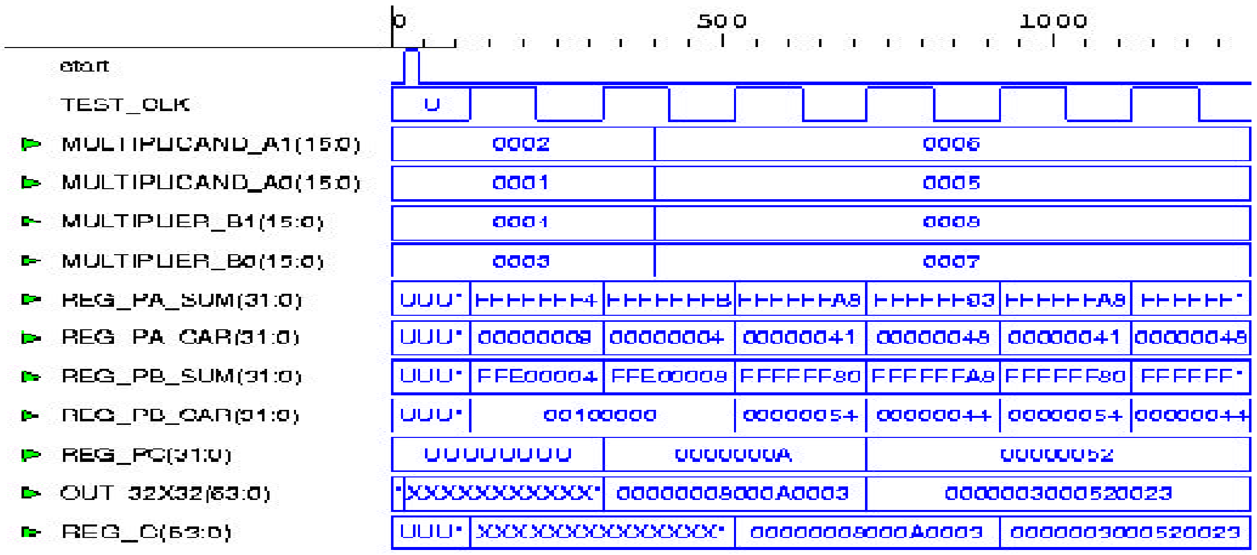


Fig. 5. *VHDL simulation results of 32x32 multiplication.*