Available online at www.elixirpublishers.com (Elixir International Journal)

Electrical Engineering

Elixir Elec. Engg. 38 (2011) 4533-4537

ISSN: 2229-712X

A new VLSI architecture for low power high performance parallel multiplier-accumulator based on radix-2 modified booth algorithm

R. Satya Veni and M. Prema Kumar

Department of Electrical and Communication Engineering, Shri Vishnu Engineering College for Women, Bhimavaram.

ARTICLE INFO

Article history:
Received: 22 August 2011;
Received in revised form:
26 August 2011;
Accepted: 31 August 2011;

Keywords
CSA,
FAs,
MBA,
MAC.

ABSTRACT

A new architecture of multiplier-and-accumulator (MAC) for high-speed arithmetic. In this by combining multiplication with accumulation and devising a hybrid type of carry save adder (CSA), the performance was improved. The CSA tree uses 1's-complement- based radix-2. Modified Booth's algorithm (MBA) and has the modified array for the sign extension in order to increase the bit density of the operands. The CSA propagates the carries to the least significant bits of the partial products and generates the least significant bits in advance to decrease the number of the input bits of the final adder. The MAC accumulates the intermediate results in the type of sum and carry bits instead of the output of the final adder, which made it possible to optimize the pipeline scheme to improve the performance.

Introduction

Power dissipation is recognized as a critical parameter in modern VLSI design field. To satisfy MOORE'S law and to produce consumer electronics goods with more backup and less weight, low power VLSI design is necessary.

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

The basic multiplication principle is twofold i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate bit of the 'multiplicand'. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation. To extend the multiplication to both signed and unsigned numbers, a convenient number system would be the representation of numbers in two's complement format.

Tele:
E-mail addresses: satya.genny@gmail.com

The MAC(Multiplier and Accumulator Unit) is used for image processing and digital signal processing (DSP) in a DSP processor. Algorithm of MAC is Booth's radix-4 algorithm, Modified Booth Multiplier, 34-bit CSA and improves speed. MIPS was implemented as micro processors and permitted high performance pipeline implementations through the use of their simple register oriented instruction sets.

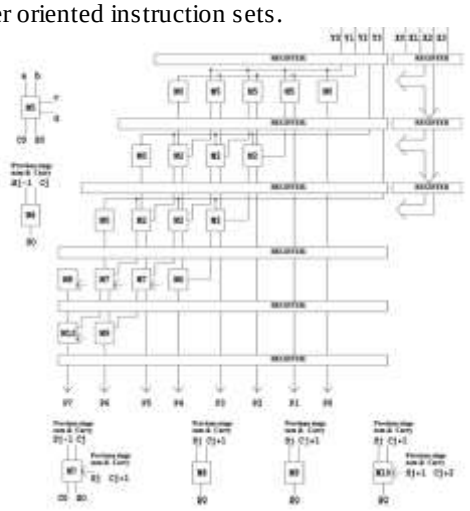

Figure 1: Array Multiplier

Although those algorithms (radix-4 algorithm, pipelining, etc) are widely used technique for speeding up each part, the MAC on specific processor cannot be run at 100% efficiency. Due to the reasons of lower speed of MAC, MIPS instruction "mul" (multiplication) takes longer time than any other instruction in our MIPS processor. To improve speed of MIPS, MAC needs to be fast and MIPS must have special algorithm for "mul" instruction. One of the methods we chose was to design multi-clock MAC instead of one-clock MAC which improved the speed of MIPS. In general, the instruction set of MIPS processor includes complex works like multiplication and floating point operation which has multi execution stage. Therefore, system clock of the processor was increased

efficiently. We applied 2 stage pipelining to the MAC to MIPS processor and as a result we were able to get the result of matrix multiplication which was used for image processing in our MIPS processor that supports MAC.

MAC

Overview of MAC

A multiplier can be divided into three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand X and the multiplier Y . The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry. If the process to accumulate the multiplied results is included, a MAC consists of four steps, as shown in Fig. 1, which shows the operational steps explicitly. General hardware architecture of this MAC is shown in Fig. 2. It executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y . This is added to the previous multiplication result Z as the accumulation step.

The N-bit 2's complement binary number can be expressed as

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, \quad x_i \in 0,1. \qquad \text{…..…..(1)}$$

If (1) is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm.

$$X = \sum_{i=0}^{N/2-1} d_i 4_i \qquad \text{…….……..(2)}$$

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}. \qquad \text{…….……(3)}$$

If (2) is used, multiplication can be expressed as

$$X \times Y = \sum_{i=0}^{N/2-1} d_i 2^{2i} Y. \qquad \text{…….…..(4)}$$

If these equations are used, the afore-mentioned multiplication–accumulation results can be expressed as

$$P = X \times Y + Z = \sum_{i=0}^{N/2-1} d_i 2^i Y + \sum_{j=0}^{2N-1} z_i 2^i. \qquad \text{…(5)}$$

Each of the two terms on the right-hand side of (5) is calculated independently and the final result is produced by adding the two results. The MAC architecture implemented by (5) is called the standard design [6].

If -bit data are multiplied, the number of the generated partial products is proportional to . In order to add them serially, the execution time is also proportional to .

The architecture of a multiplier, which is the fastest, uses radix-2 Booth encoding that generates partial products and a Wallace tree based on CSA as the adder array to add the partial products.

If radix-2 Booth encoding is used, the number of partial products, i.e., the inputs to the Wallace tree, is reduced to half, resulting in the decrease in CSA tree step. In addition, the signed multiplication based on 2's complement numbers is also possible. Due to these reasons, most current used multipliers adopt the Booth encoding.

Proposed MAC Architecture

If an operation to multiply two N –bit numbers and accumulate into a 2N -bit number is considered, the critical path is determined by the 2 -bit accumulation operation. If a pipeline scheme is applied for each step in the standard design of Fig. 1, the delay of the last accumulator must be reduced in order to improve the performance of the MAC. The overall performance of the proposed MAC is improved by eliminating the accumulator itself by combining it with the CSA function. If the accumulator has been eliminated, the critical path is then determined by the final adder in the multiplier. The basic method to improve the performance of the final adder is to decrease the number of input bits. In order to reduce this number of input bits, the multiple partial products are compressed into a sum and a carry by CSA. The number of bits of sums and carries to be transferred to the final adder is reduced by adding the lower bits of sums and carries in advance within the range in which the overall performance will not be degraded. A 2-bit CLA is used to add the lower bits in the CSA. In addition, to increase the output rate when pipelining is applied, the sums and carrys from the CSA are accumulated instead of the outputs from the final adder in the manner that the sum and carry from the CSA in the previous cycle are inputted to CSA. Due to this feedback of both sum and carry, the number of inputs to CSA increases, compared to the standard design and [17]. In order to efficiently solve the increase in the amount of data, a CSA architecture is modified to treat the sign bit.

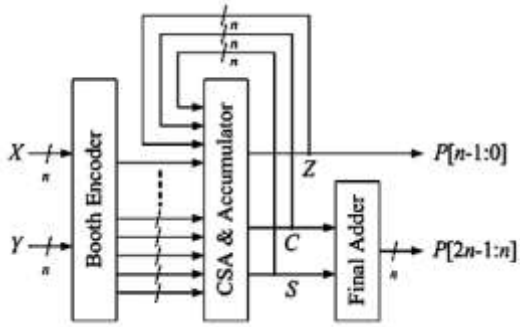

Figure 2: Hardware architecture of Proposed MAC

High-Speed Booth Encoded Parallel Multiplier Design: Modified Booth Encoder

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

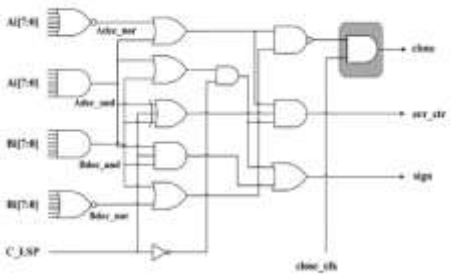

Figure 3: Detection logic circuits using an AND Gate

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The

basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Figure 3 shows the grouping of bits from the multiplier term for use in modified booth encoding.

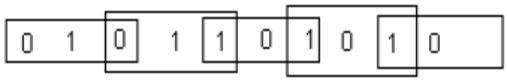

Figure 4: Grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 1

Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

For the partial product generation, we adopt Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half. For multiplication of 2's complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits.

Figure 4, shows a computing example of Booth multiplying two numbers "2AC9" and "006A". The shadow denotes that the numbers in this part of Booth multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals. According to the analysis of the multiplication shown in figure 4, we propose the SPST-equipped modified-Booth encoder, which is controlled by a detection unit. The detection unit has one of the two operands as its input to decide whether the Booth encoder calculates redundant computations. As shown in figure 9. The latches can, respectively, freeze the inputs of MUX-4 to MUX-7 or only those of MUX-6 to MUX-7 when the PP4 to PP7 or the PP6 to PP7 are zero; to reduce the transition power dissipation. Figure 10, shows the booth partial product generation circuit. It includes AND/OR/EX-OR logic.

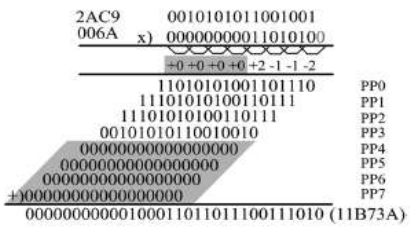

Figure 5: Illustration of multiplication using modified Booth encoding

The PP generator generates five candidates of the partial products, i.e., {-2A,-A, 0, A, 2A}. These are then selected according to the Booth encoding results of the operand B. When the operand besides the Booth encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree.

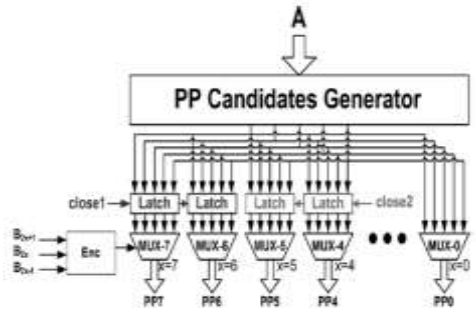

Figure 6: SPST equipped modified Booth encoder Partial product generator

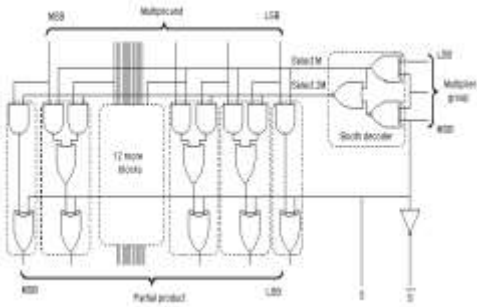

Figure 7: Booth Partial Product selector Logic

The multiplication first step generates from A and X a set of bits whose weights sum is the product P. For unsigned multiplication, P most significant bit weight is positive, while in 2's complement it is negative.

The partial product is generated by doing AND between 'a' and 'b' which are a 4 bit vectors as shown in fig. If we take, four bit multiplier and 4-bit multiplicand we get sixteen partial products in which the first partial product is stored in 'q'. Similarly, the second, third and fourth partial products are stored in 4-bit vector n, x, y.

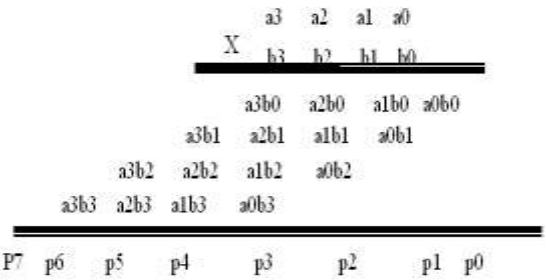

Figure 8: Booth partial products Generation

The multiplication second step reduces the partial products from the preceding step into two numbers while preserving the weighted sum. The sough after product P is the sum of those two numbers.

The two numbers will be added during the third step The "Wallace trees" synthesis follows the Dadda's algorithm, which assures of the minimum counter number. If on top of that we impose to reduce as late as (or as soon as) possible then the solution is unique. The two binary number to be added during the third step may also be seen a one number in CSA notation (2 bits per digit).

Spurious Power Suppression Technique

The former SPST has been discussed in [7] and [8].figure 2 shows the five cases of a 16-bit addition in which the spurious switching activities occur. The 1st case illustrates a transient state in which the spurious transitions of carry signals occur in the MSP though the final result of the MSP are unchanged. The 2nd and the 3rd cases describe the situations of one negative operand adding another positive operand without and with carry from LSP, respectively. Moreover, the 4th and the 5th cases respectively demonstrate the addition of two negative operands without and with carry-in from LSP. In those cases, the results of the MSP are predictable Therefore the computations in the MSP are useless and can be neglected. The data are separated into the Most Significant Part (MSP) and the Least Significant Part (LSP).To know whether the MSP affects the computation results or not. We need a detection logic unit to detect the effective ranges of the inputs. The Boolean logical equations shown below express the behavioral principles of the detection logic unit in the MSP circuits of the SPST-based adder/subtractor:

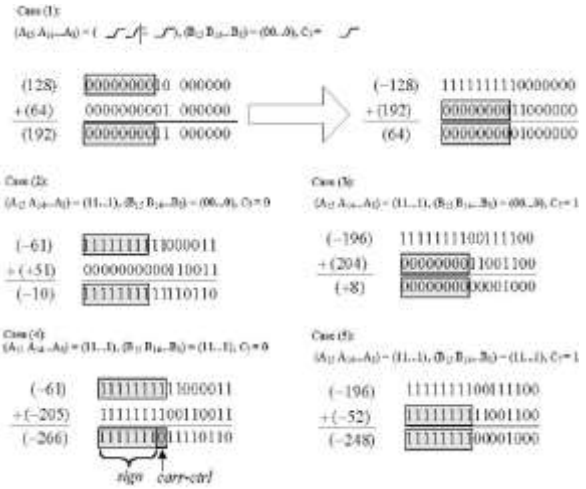

Figure 9: Spurious transition cases in multimedia/ DSP processing

AMSP = A[15:8]; BMSP = B[15:8] ;
Aand = A[15] A[14] A[8];
Band = B[15] B[14] B[8];]

$$A_{nor} = \overline{A[15] + A[14] + A[13] + \cdots + A[8]};$$

$$B_{nor} = \overline{B[15] + B[14] + B[13] + \cdots + B[8]};$$

$$Close = \overline{(Aand + Anor) \cdot (Band + Bnor)};$$

where A[m] and B[n] respectively denote the mth bit of the operands A and the nth bit of the operand B, and AMSP and BMSP respectively denote the MSP parts, i.e. the 9th bit to the 16th bit, of the operands A and B. When the bits in AMSP and/or those in BMSP are all ones, the value of Aand and/or that of Band respectively become one, while the bits in AMSP and/or those in BMSP are all zeros, the value of Anor, and/or that of Bnor respectively turn into one. Being one of the three outputs of the detection logic unit, close denotes whether the MSP circuits can be neglected or not. When the two input operand can be classified into one of the five classes as shown in figure 1, the value of close becomes zero which indicates that the MSP circuits can be closed. figure 1. also shows that it is necessary to compensate the sign bit of computing results Accordingly, we derive the Karnaugh maps which lead to the Boolean equations (7) and (8) for the Carr_ctrl and the sign signals, respectively. In equation (7) and (8), CLSP denotes the carry propagated from the LSP circuits.

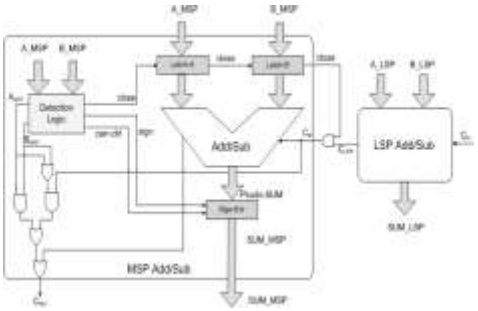

Fig. shows a 16-bit adder/subtractor design example adopting the proposed SPST. In this example, the 16-bit adder/subtractor is divided into MSP and LSP between the eighth and the ninth bits. Latches implemented by simple AND gates are used to control the input data of the MSP. When theMSP is necessary, the input data of MSP remain unchanged. However, when the MSP is negligible, the input data of the MSP become zeros to avoid glitching power consumption. The two operands of the MSP enter the detection-logic unit, except the adder/subtractor, so that the detection-logic unit can decide whether to turn off the MSP or not. Based on the derived Boolean equations (1) to (8), the detection-logic unit of SPST is shown in Fig. 6(a), which can determine whether the input data of MSP should be latched or not. Moreover, we propose the novel glitch-diminishing technique by adding three 1-bit registers to control the assertion of the close, sign, and carr-ctrl signals to further decrease the transient signals occurred in the cascaded circuits which are usually adopted in VLSI architectures designed for multimedia/DSP applications. The timing diagram is shown in Fig. 6(b). A certain amount of delay is used to assert the close, sign, and carr-ctrl signals after the period of data transition which is achieved by controlling the three 1-bit registers at the outputs of the detection-logic unit.

Hence, the transients of the detection-logic unit can be filtered out; thus, the data latches shown in Fig can prevent the glitch signals from flowing into the MSP with tiny cost. The data transient time and the earliest required time of all the inputs are also illustrated. The delay should be set in the range of, which is shown as the shadow area in Fig, to filter out the glitch signals as well as to keep the computation results correct. Based on Figs. 5 and 6, the timing issue of the SPST is analyzed as follows.

Conclusion

A 16x16 multiplier-accumulator (MAC) is presented in this work. A RADIX 4 Modified Booth multiplier circuit is used for MAC architecture. Compared to other circuits, the Booth multiplier has the highest operational speed and less hardware count. The basic building blocks for the MAC unit are identified and each of the blocks is analyzed for its performance. Power and delay is calculated for the blocks. 1-bit MAC unit is designed with enable to reduce the total power consumption based on block enable technique. Using this block, the N-bit MAC unit is constructed and the total power consumption is calculated for the MAC unit. The power reduction techniques adopted in this work. The MAC unit designed in this work can be used in filter realizations for High speed DSP applications. Table 12 summarizes the results obtained.

References

[1]A. D. Booth, "A signed binary multiplication technique," Quart. J.Math., vol. IV, pp. 236–240, 1952.

[2] C. S. Wallace, “A suggestion for a fast multiplier,” IEEE Trans. Elec- tron Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[3] A. R. Cooper, “Parallel architecture modified Booth multiplier,” Proc.
[4] Inst. Electr. Eng. G, vol. 135, pp. 125–128, 1988.
[5] N. R. Shanbag and P. Juneja, “Parallel implementation of a 4 4-bit multiplier using modified Booth’s algorithm,” IEEE J. Solid-State Cir- cuits, vol. 23, no. 4, pp. 1010–1013, Aug. 1988.
[6] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, “A 54 54 regular structured tree multiplier,” IEEE J. Solid-State Circuits, vol. 27, no. 9, pp. 1229–1236, Sep. 1992.