CHAPTER 2

LITERATURE STUDY

2.1 Introduction

Multiplication involves two basic operations: the generation of the partial products and their accumulation. Therefore, there are two possible ways to speed up the multiplication: reduce the number of partial products or accelerate their accumulation [5]. A smaller number of partial products also reduces the complexity, and as a result, reduces the time needed to accumulate the partial products. Both solutions can be applied simultaneously.

2.2 High Speed Multiplier

2.2.1 Array Multiplier [6]

The array multiplier originates from the multiplication parallelogram. As shown in Figure 2.1, each stage of the parallel adders should receive some partial product inputs. The carry-out is propagated into the next row. The bold line is the critical path of the multiplier. In a non-pipelined array multiplier, all of the partial products are generated at the same time. It is observed that the critical path consists of two parts: vertical and horizontal. Both have the same delay in terms of full adder delays and gate delays. For an n-bit by n-bit array multiplier, the vertical and the horizontal delays are both the same as the delay of an n-bit full adder.

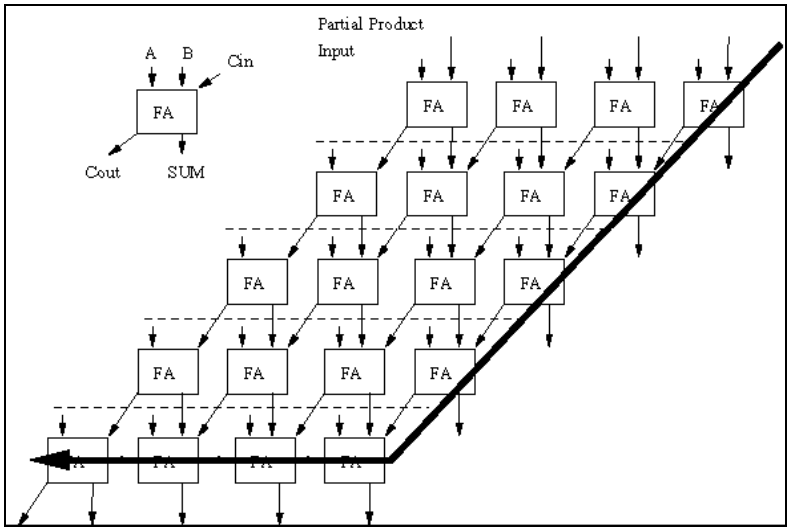

Figure 2.1 : 4-bits x 4-bits Array Multiplier [6]

One advantage of the array multiplier comes from its regular structure. Since it is regular, it is easy to layout and has a small size. The design time of an array multiplier is much less than that of a tree multiplier. A second advantage of the array multiplier is its ease of design for a pipelined architecture. The main disadvantage of the array multiplier is the worst-case delay of the multiplier proportional to the width of the multiplier. The speed will be slow for a very wide multiplier.

2.2.2 Tree Multiplier

In the multiplier based on Wallace tree, the multiplicand-multiples are summed up in parallel by means of a tree of carry save adders. A Carry Save Adder sums up three binary numbers and produces two binary numbers [6]. Figure 2.2 illustrates a block diagram of a multiplier based on Wallace tree. This consists of full adders, just like the array multiplier.

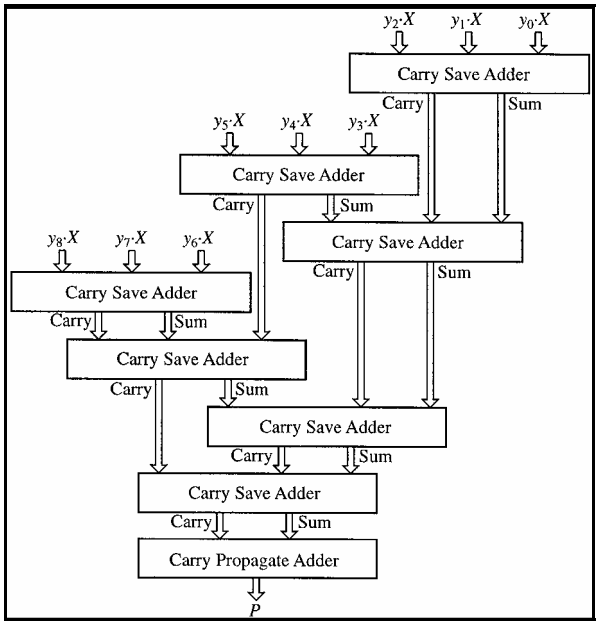

Figure 2.2 : A Multiplier with Wallace Tree [6]

One advantage of the Wallace tree is it has small delay. The number of logic levels required to perform the summation can be reduced with Wallace tree. The main disadvantages of Wallace tree is complex to layout and has irregular wires [1].

2.2.3 Booth Multiplier

The modified Booth recoding algorithm is the most frequently used method to generate partial products [8]. This algorithm allows for the reduction of the number of partial products to be compressed in a carry-save adder tree. Thus the compression speed can be enhanced. This Booth–Mac Sorley algorithm is simply called the Booth algorithm, and the two-bit recoding using this algorithm scans a triplet of bits to reduce the number of partial products by roughly one half. The 2-bit recoding means that the multiplier *B* is divided into groups of two bits, and the algorithm is applied to this group of divided bits. The Booth algorithm is implemented into two steps: Booth encoding and Booth selecting. The Booth encoding step is to generate one of the five values from the

adjacent three bits. The Booth selector generates a partial product bit by utilizing the output signals.

One advantage of the Booth multiplier is, it reduce the number of partial product, thus make it extensively used in multiplier with long operands (>16 bits) [7]. The main disadvantage of Booth multiplier is the complexity of the circuit to generate a partial product bit in the Booth encoding [9].

2.3 Modified Baugh-Wooley Two's Complement Signed Multiplier

2.3.1 Two's Complement System [15]

Two's complement is the most popular method of representing signed integers in computer science. It is also an operation of negation (converting positive to negative numbers or vice versa) in computers which represent negative numbers using two's complement. Its use is ubiquitous today because it does not require the addition and subtraction circuitry to examine the signs of the operands to determine whether to add or subtract, making it both simpler to implement and capable of easily handling higher precision arithmetic.

Two's complement and one's complement representations are commonly used since arithmetic units are simpler to design. Figure 2.3, shows two's complement and one's complement representations.

+N	Positive Integers	-N	Negative Integers		
			Sign & Magnitude	2's Complement	1's Complement
+0	0000	-0	1000	----	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
+8	----	-8	----	1000	----

Figure 2.3 : Two's Complement and One's Complement Representations

In an *n*-bit binary number, the most significant bit is usually the 2^{n-1} s place. But in the two's complement representation, its place value is negated; it becomes the -2^{n-1} s place and is called the sign bit.

If the sign bit is 0, the value is positive; if it is 1, the value is negative. To negate a two's complement number, invert all the bits then add 1 to the result.

If all bits are 1, the value is −1. If the sign bit is 1 but the rest of the bits are 0, the value is the most negative number, -2^{n-1} for an n-bit number. The absolute value of the most negative number cannot be represented with the same number of bits because it is greater than the most positive number that two's complement number by exactly 1.

A two's complement 8-bits binary numeral can represent every integer in the range −128 to +127. If the sign bit is 0, then the largest value that can be stored in the remaining seven bits is 27 − 1, or 127.

Using two's complement to represent negative numbers allows only one representation of zero, and to have effective addition and subtraction while still having the most significant bit as the sign bit.

2.3.2 Modified Baugh-Wooley Two's Complement Signed Multiplier

One important complication in the development of the efficient multiplier implementations is the multiplication of two's complement signed numbers. The Modified Baugh-Wooley Two's Complement Signed Multiplier is the best known algorithm for signed multiplication because it maximizes the regularity of the multiplier logic and allows all the partial products to have positive sign bits [3].

Baugh-Wooley technique was developed to design direct multipliers for two's complement numbers [9]. When multiplying two's complement numbers directly, each of the partial products to be added is a signed number. Thus, each partial product has to be sign-extended to the width of the final product in order to form the correct sum by the Carry Save Adder tree. According to the Baugh-Wooley approach, an efficient method of adding extra entries to the bit matrix is suggested to avoid having to deal with the negatively weighted bits in the partial product matrix. In Figure 2.4 partial product array's of 5-bits x 5-bits unsigned bit are shown below :

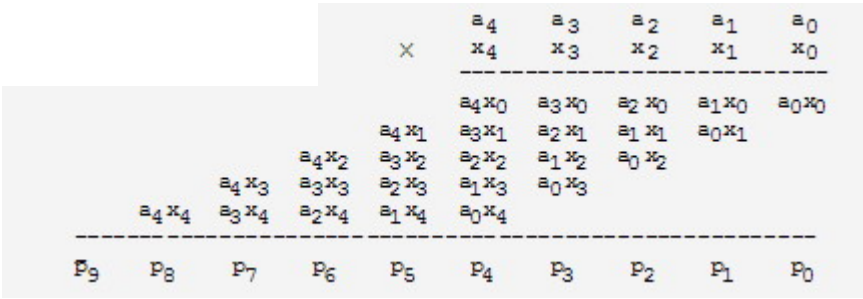

Figure 2.4 : Unsigned Multiplication [9]

Partial product array's of two's complement multiplication of 5-bits x 5-bits are shown in Figure 2.5 :

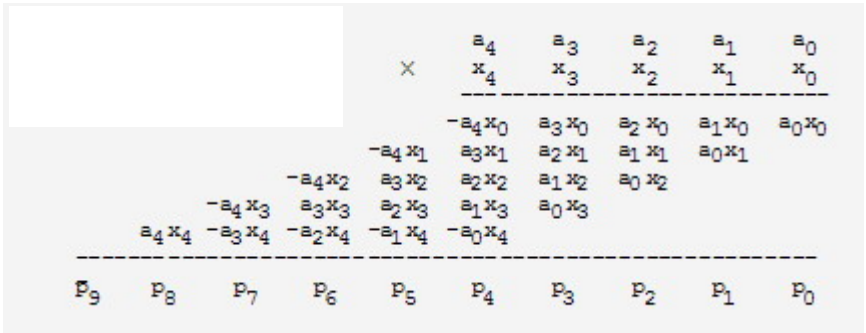

Figure 2.5 : Two's Complement Multiplication [9]

Here is how the algorithm works. Knowing that the sign bit in two's complement numbers has a negative weight, the entry the term can be written in terms of $-a_4x_0$.

$$-a_4x_0 = a_4(1-x_0) - a_4 = a_4\overline{x_0} - a_4 \tag{2.0}$$

Hence, the term $-a_4x_0$ is replaced with $a_4\overline{x_0}$ and $-a_4$. If a_4 is used instead of $-a_4$, the column sum increases by $2a_4$. Thus, $-a_4$ must be inserted in the next higher column in order to compensate the effect of $2a_4$. The same is done for $a_4\overline{x_1}$, $a_4\overline{x_2}$ and $a_4\overline{x_3}$. In each column, a_4 and $-a_4$ cancel each other out. The p_8 column gets a $-a_4$ entry, which is replaceable by $\overline{a_4}-1$. This can be repeated for all entries, yielding to the insertion of x_4 in the p_4 column, and $\overline{x_4}-1$ in the column p_8. There are two -1's in the eighth column now, which is equivalent to a -1 entry in p_9 and that can be replaced wit a 1 and borrow into the non-existing tenth column.

Baugh-Wooley method increases the height of the longest column by two, which may lead to a greater delay through the Carry Save Adder tree. In the given example of Figure 2.6 column height changes from 5 to 7, requiring an extra Carry Save Adder level.

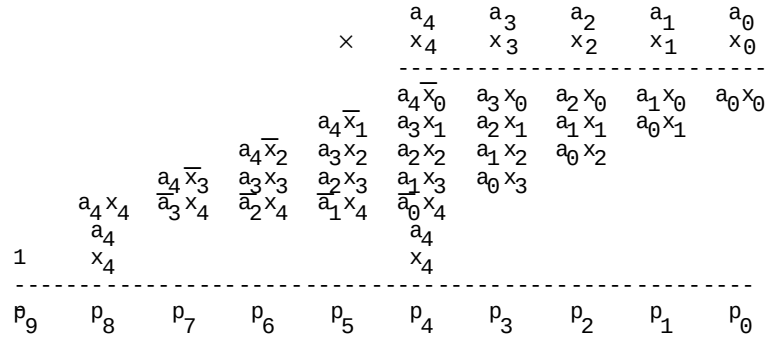

Figure 2.6 : Baugh-Wooley Two's Complement Signed Multiplication [9]

Removing x_4 from fourth column and writing two x_4 entries in the third column, which has only four entries, can reduce the extra delay caused by the additional Carry Save Adder level. Thus, the maximum number of entries in one column becomes six, which can be implemented with three level Carry Save Adder tree.

All negatively weighted a_4x_4 terms can be transferred to the bottom row, which leads to two negative numbers in the last two rows, where a subtraction operation from the sum of all the positive elements is necessary. Instead of subtracting a_4x_4 two's complement of a can be added x_4 times. This method is known as the Modified Baugh-Wooley algorithm as shown in Figure 2.7.

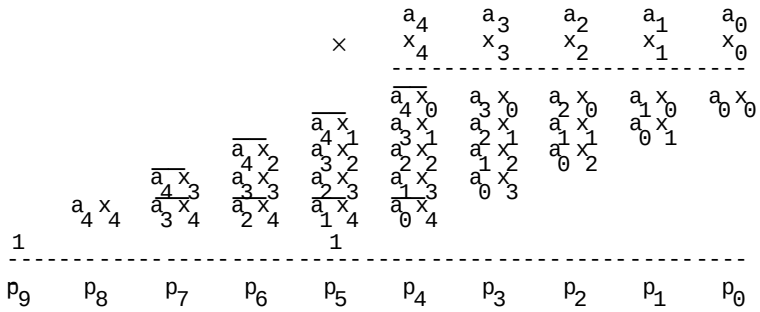

Figure 2.7 : Modified Baugh-Wooley Two's Complement Signed Multiplication

Modified form of the Baugh-Wooley method, is more preferable since it does not increase the height of the columns in the matrix. However, this type of multiplier is suitable for applications where operands with less than 32 bits are processed, like digital filters where small operands like 6, 8, 12 and 16 bits are used. Baugh-Wooley scheme becomes slow and area consuming when operands are greater than or equal to 32 bits.