CHAPTER 5

THE MULTIPLIER

5.1 INTRODUCTION

A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis (Ayman.A et al (2001)). Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints. Since multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design.

Multiplications are very expensive and slows the overall operation. The performance of many computational problems are often dominated by the speed at which a multiplication operation can be executed. Consider two unsigned binary numbers X and Y that are M and N bits wide, respectively. To introduce the multiplication operation, it is useful to express X and Y in the binary representation
\[ X = \sum_{i=0}^{M} X_i 2^i \]  
\[ Y = \sum_{j=0}^{N} Y_j 2^j \]  
(5.1)  
(5.2)

With \( X_i, Y_j \in \{0,1\} \). The multiplication operation is then defined as follows:

\[ Z = X \times Y = \sum_{k=0}^{M+N-1} Z_k 2^k \]  
\[ = (\sum_{i=0}^{M} X_i 2^i) (\sum_{j=0}^{N} Y_j 2^j) \]  
\[ = \sum_{i=0}^{M-1} X_i Y_j 2^{i+j} \]  
(5.3)  
(5.4)  
(5.5)

The simplest way to perform a multiplication is to use a single two input adder. For inputs that are \( M \) and \( N \) bits wide, the multiplication tasks \( M \) cycles, using an \( N \)-bit adder. This shift–and-add algorithm for multiplication adds together \( M \) partial products. Each partial product is generated by multiplying the multiplicand with a bit of the multiplier – which, essentially, is an AND operation – and by shifting the result in the basis of the multiplier bit’s position. Similar to the familiar long hand decimal multiplication, binary multiplication involves the addition of shifted versions of the multiplicand based on the value and position of each of the multiplier bits. As a matter of fact, it’s much simpler to perform binary multiplication than decimal multiplication. The value of each digit of a binary number can only be 0 or 1, thus, depending on the value of the multiplier bit, the partial products can only be a copy of the multiplicand, or 0. In digital logic, this is simply an AND function.

A faster way to implement multiplication is to resort to an approach similar to manually computing a multiplication. The entire partial product are generated at the same time and organized in an array. A multioperand addition is applied to compute the final product. The approach is illustrated in the figure 5.1. This set of operation can be mapped directly into hardware. The resulting structure
is called an array multiplier and combines the following three functions: partial-
product generation, partial-product accumulation and final addition.

```
1 0 1 0 1 0  # Multiplicand
X 1 0 1 1    # Multiplier

1 0 1 0 1 0  # Partial product
1 0 1 0 1 0
0 0 0 0 0
+ 1 0 1 0 1 0

1 1 1 0 0 1 1 0  # Result
```

Figure 5.1 Example of manual multiplication

So the adder unit is very important for designing any multiplier (John Rabaey (2003)). The different types of adders and their functions were discussed in (Oklobdzija.V.G et al (1995)), (Pucknell (2004)), (Shalem.R et al (1999)) and (Zimmermann.R and Fichtner.W (1997)). From the results of (Shalem.R et al (1999)), we came to know that the new improved 14 Transistor full adder cell shows better result in Threshold loss problem, power dissipation and speed by sacrificing MOS transistor count.

In this research paper, the following four important types of multipliers (Array, Baugh wooly, Braun and Wallace tree) are constructed using different types of adder cells presented in (Shalem.R et al (1999)), we then find out the best one in the performance characteristics like power dissipation, speed and area.
5.2 ARRAY MULTIPLIER

The composition of an array multiplier is shown in the figure 5.2. There is a one-to-one topological correspondence between this hardware structure and the manual multiplication shown in figure 5.1. The generation of n partial products requires N*M two-bit AND gates. Most of the area of the multiplier is devoted to the adding of n partial products, which requires N-1, M-bit adders. The shifting of the partial products for their proper alignment is performed by simple routing and does not require any logic. The overall structure can be easily be compacted into rectangle, resulting in very efficient layout.

5.3 BAUGH WOOLEY MULTIPLIER

Baugh-Wooley algorithm for the unsigned binary multiplication is based on the concept shown in figure 5.3. The algorithm specifies that all possible AND terms are created first, and then sent through an array of half-adders and full-adders with the Carry-outs chained to the next most significant bit at each level of addition. Negative operands may be multiplied using a Baugh-Wooley multiplier.

5.4 BRAUN MULTIPLIER

The simplest parallel multiplier is the Braun array. All the partial products are computed in parallel, then collected through a cascade of Carry Save
Adders. The completion time is limited by the depth of the carry save array, and by the carry propagation in the adder. Note that this multiplier is only suited for positive operands. The structure of the Braun algorithm for the unsigned binary multiplication is shown in figure 5.4.

5.5 WALLACE TREE MULTIPLIER

The partial-sum adders can also be rearranged in a tree like fashion, reducing both the critical path and the number of adder cells needed. The presented structure is called the Wallace tree multiplier and its implementation is shown in figure 5.5. The tree multiplier realizes substantial hardware savings for larger multipliers. The propagation delay is reduced as well. In fact, it can be shown that the propagation delay through the tree is equal to $O(\log_{3/2}(N))$. While substantially faster than the carry-save structure for large multiplier word lengths, the Wallace multiplier has the disadvantage of being vary irregular, which complicates the task of an efficient layout design.
Figure 5.2 Array Multiplier Architecture
Figure 5.3 Baugh Wooly Multiplier Architecture
Figure 5.4 Braun Multiplier Architecture
The given four types of multiplier architecture are designed with seven different kinds of adder cell namely 14 transistors full adder cell, 20 transistors full adder cell, 28 transistors full adder cell, conventional full adder cell, New improved 14T full adder cell, transmission functional full adder cell and transmission gate full adder cell.

Based on the extensive simulations, Except the Wallace tree multiplier, the other multipliers do not provide low power dissipation for all possible input combinations. Thus the Wallace tree multiplier using new improved
14T adder improves the power dissipation by 27% when compared to the other multipliers architecture. Wallace tree multiplier using new improved 14T adder cell dissipates small amount of power, which shall be called as low power multiplier. As mentioned earlier, the performance of many larger circuits is strongly dependent on the performance of the multiplier circuits that have been used. Also its Speed is improved by 47.8% when compared to the other multiplier types. The Wallace tree multiplier using new improved 14-transistor adder circuits presented in this research are good candidates to build these large systems, such as high performance FIR filters with low power consumption. The small increase in transistor count of these adders can significantly reduce the latency of the systems built upon them. Also the area occupied by Wallace tree multiplier using new improved 14T adder is reduced significantly as compared to other types of multiplier architectures.

Thus, after the multiplier analysis, it is concluded that the implementation of the NEW adder in the Wallace tree multiplier structure give the demanding results.