Fast Multipliers

Shift and add Multiplication Algorithm vs Wallace Tree Multiplier

Team Members:

Karan Verma – 2023CSB1127
Neredumalli Likhit – 2023CSB1139
Parvinder Singh – 2023CSB1143
Sarvan Suthar – 2023CSB1159
Sumit Singh – 2023CSB1166
Vikash Kumar – 2023CSB1172

INTRODUCTION

Multiplication is one of the fundamental operations, utilized in several computing applications. Efficiency of a multiplication algorithm has impact on the performance of any system. Our Project deals with 2 multiplication algorithms:

1. Shift and Add Multiplication algorithm
2. Wallace Tree Multiplier

The shift and add algo uses shifting of bits and then add them step by step to calculate the product. However, the Wallace tree multiplier algorithm uses parallel processing to compute the partial products more quickly.

1. SHIFT AND ADD MULTIPLICATION ALGORITHM

The Shift-and-Add Multiplication Algorithm is an algorithm which sequentially computes the product of two integers. It multiplies each bit of multiplier with the multiplicand, the each product is summed up by shifting it bit by bit using step by step approach. The algorithm for simple hardware implementations, especially in systems with fewer resources, but it trades off performance for simplicity.

The Shift-and-Add Multiplication Algorithm operates by iterating through the bits of the multiplier, checking each bit's value. If the bit is 1, the algorithm shifts the multiplicand to the appropriate position and adds it to an accumulating product. This process repeats for all bits of the multiplier. While simple and resource-efficient, this sequential approach results in slower performance compared to more parallelized algorithms, making it more suitable for low-resource or embedded systems where hardware simplicity is prioritized over speed.

2. WALLACE TREE MULTIPLIER DESIGN

The Wallace Tree Multiplier is a high-speed multiplication algorithm that uses a parallel approach to efficiently compute the product of two integers. It generates partial products for each bit of the multiplier and organizes them into a tree structure. Using carry-save adders

(CSAs), the algorithm reduces the partial products in parallel, minimizing the number of sequential addition stages. This design significantly reduces delay, making it faster than traditional methods like Shift-and-Add. The key advantage of the Wallace Tree is its ability to perform multiple additions simultaneously, reducing the overall number of clock cycles needed to arrive at the final product. Each layer of the tree decreases the number of terms by half, which leads to a logarithmic reduction in the number of steps required for multiplication. This makes the Wallace Tree particularly advantageous in high-performance computing environments, where multiplication speed is critical. However, the complexity of the architecture, which involves the use of multiple stages of CSAs, and the higher resource utilization make it more suitable for performance-critical applications rather than resource-constrained systems. The increased gate count and the need for more complex routing make it less desirable for smaller, low-power devices or those with limited hardware resources. Despite these challenges, the Wallace Tree Multiplier's speed and efficiency in handling large numbers make it the go-to solution for applications like digital signal processing (DSP), graphics processing units (GPUs), and high-speed processors where the demand for fast computations outweighs the concerns about area and power consumption. Its ability to scale well with the bit-width of operands further enhances its suitability for systems requiring fast and efficient multiplication of large numbers.

3. COMPARISON

For the two algorithm of Shift and Add Multiplication and Wallace Tree multiplier design, we will compare both the algorithms based on some properties to provide a better understanding of their trade offs. Time taken tells about suitability for high speed applications. Similarly other properties can be used. This multifaceted comparison ensures a balanced assessment for selecting the right algorithm for specific applications.

Why the Wallace Tree Multiplier is Better:

The **Wallace Tree Multiplier** is better when considering **performance efficiency**. Despite its higher gate count and more complex control logic, the parallelism in the Wallace Tree structure allows it to perform multiplications faster, especially as the number of bits in the operands increases. The shift-and-add multiplier, while simpler and less hardware-intensive, becomes slower as the bit-width of the operands grows because it performs each operation sequentially. Therefore, for larger bit-widths or high-performance requirements, the Wallace Tree Multiplier's increased complexity is justified by its speed, making it a more efficient choice for hardware implementations.

Conclusion

After evaluating both the **Shift-and-Add Multiplier** and the **Wallace Tree Multiplier**, it is clear that both algorithms have their own advantages and limitations, making them suitable for different applications based on the requirements of the hardware and the multiplication task at hand.

The Shift-and-Add Multiplier stands out for its simplicity and ease of implementation. It requires fewer gates and has a more straightforward control logic due to its sequential nature. This makes it well-suited for environments with limited resources or where the multiplication involves relatively small operands. However, its performance suffers as the bit-width of the numbers increases, making it less efficient for large-scale computations. Its main limitation is the slower execution time due to its step-by-step processing approach.

On the other hand, the Wallace Tree Multiplier excels in speed and parallelism, making it a better choice for high-performance applications that require handling large numbers or fast multiplication. Despite its higher gate count and more complex control logic, it significantly reduces computation time by performing partial product additions in parallel. This makes it highly suitable for applications such as digital signal processing, computer graphics, and high-speed processors where time efficiency is critical. However, the added complexity in design and resource consumption can be a limiting factor for low-resource or simple systems.

In conclusion, the Wallace Tree Multiplier is generally the better choice for applications where speed and efficiency are crucial, particularly when handling large bit-width numbers. The Shift-and-Add Multiplier, while simpler and resource-efficient, is best suited for applications where simplicity and low resource usage are more important than speed, such as in small embedded systems or situations where operands are relatively small. Therefore, the selection between these two algorithms depends heavily on the specific needs of the project, including factors such as performance requirements, hardware constraints, and the scale of the multiplication task.