Implementation of Wallace Tree Multiplier Using Compressor

Naveen Kr. Gahlan  
Research Scholar  
Department of E&Ec  
PEC University of Technology  
Chandigarh, India

Prabhat Shukla  
Research Scholar  
Department of E&Ec  
PEC University of Technology  
Chandigarh, India

Jasbir Kaur  
Assistant Professor  
Department of E&Ec  
PEC University of Technology  
Chandigarh, India

Abstract—Designing multipliers that are of high-speed, low power, and regular in layout are of substantial research interest. Speed of the multiplier can be increased by reducing the generated partial products. Many attempts have been made to reduce the number of partial products generated in a multiplication process one of them is Wallace tree multiplier. Wallace Tree CSA structures have been used to sum the partial products in reduced time. In this paper Wallace tree construction is investigated and evaluated. Speed of traditional Wallace tree multiplier can be improve by using compressor techniques. In this paper Wallace tree is constructed by traditional method and with the help of compressor techniques such as 4:2 compressor, 5:2 compressor, 6:2 compressor, 7:2 compressor. Therefore, minimizing the number of half adders used in a multiplier reduction will reduce the complexity.

Keywords-Multiplier, Multiplication, Wallace Tree Multiplier, Compressor techniques

I. INTRODUCTION

The digital signal processing (DSP) is one of the core technologies in multimedia and communication systems. Many application systems based on DSP, especially the recent next-generation optical communication systems, require extremely fast processing of a huge amount of digital data [4]. Digital signal processor (DSP) systems incorporate a multiplication unit to implement algorithms such as convolution and filtering. In many DSP algorithms, the multiplier lies in the critical delay path and ultimately determines the performance of the algorithm [6].

Multiplication is the most critical operation in every computational system [8]. Graphics and Process control are two more domains where in the multiplier performance plays a crucial role. The bottlenecks posed by multiplication in the above mentioned application areas are both temporal and spatial in nature. Therefore it appears that, custom VLSI implementations in the form of Application Specific Integrated Circuits (ASIC) or the DSP processors are the only viable alternatives to address the latency demands of such computationally intensive applications, that too without compromising the spatial aspects in view of the propagation delays [2]. However, even in the FPGA paradigm, the custom multiplication hardware embedded within a reconfigurable array has shown promising results and hence is a preferred choice currently due to the cost effective rapid prototyping design cycles as well as the possibility of concurrency, distributed arithmetic and extensive specialization. Accordingly many researchers have implemented variants of the basic ‘Shift Addition’ method to realize the multipliers in FPGA. Some of the preeminent methods of multiplier implementation in the FPGA paradigm are Scaling Accumulator, Ripple Carry Array, Carry Save Array, Look-Up Table and Partial Product, Computed Partial Product, and Wallace Trees. Amongst the above mentioned multiplier implementations, the Wallace tree multipliers stimulate VLSI implementation interests reduce the depth of the adder chain thereby minimizing the time complexity. The most efficient multiplier structure will vary depending on the throughput requirement of the application. The first step of the design process is the selection of the optimum circuit structure. There are various structures to perform the multiplication operation starting from the simple serial multipliers to the complex parallel multipliers [1]. Any speed improvement in the multiplier will improve the operating frequency of the digital signal processors or can be traded for energy by optimizing circuit sizes and the voltage supply.

This paper addresses high-level optimization techniques for Wallace Tree Multiplier multipliers. High-level techniques refer to algorithm and architecture level techniques that consider multiplication’s arithmetic features and input data characteristics. One of the important algorithm presents in paper for VLSI implementable multiplication is Wallace Tree Multiplier using Xilinx and Cadence tool.

II. MULTIPLIER

A basic multiplier can consist of three parts (i) partial product generation (iii) partial product addition and (iii) final addition [9]. A multiplier essentially consist of two operands,
a multiplicand “Y” and a multiplier “X” and produces a product “P”. In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial products product. The second stage is the most important, as it is the most complicated and determines the speed of the overall multiplier to add these partial product to generate the Product “P”. This paper will be focused on the optimization of this stage, which consists of the addition of all the partial products. If speed is not an issue, the partial products can be added serially, reducing the design complexity. However, in high-speed design, the Wallace tree construction method is usually used to add the partial products in a tree-like fashion in order to produce two rows of partial products that can be added in the last stage. Although fast, since its critical path delay is proportional to the logarithm of the number of bits in the multiplier, the Wallace tree introduces other problems such as wasted layout area and increased complexity. In the last stage, the two-row outputs of the tree are added using any high-speed adder such as carry save adder to generate the output result.

III. WALLACE TREE MULTIPLIER

A fast process for multiplication of two numbers was developed by Wallace [7]. Using this method, a three step process is used to multiply two numbers; the bit products are formed, the bit product matrix is reduced to a two row matrix where sum of the row equals the sum of bit products, and the two resulting rows are summed with a fast adder to produce a final product. In the Wallace Tree method, three bit signals are passed to a one bit full adder (“3W”) which is called a three input Wallace Tree circuit, and the output signal (sum signal) is supplied to the next stage full adder of the same bit, and the carry output signal thereof is passed to the next stage full adder of the same no of bit, and the carry output signal thereof is supplied to the next stage of the full adder located at a one bit higher position[5].

![Fig.1. Logic used in 4 bit Wallace Tree Multiplier](image1.png)

![Fig.2. Wallace Tree Multiplier](image2.png)

![Fig.3. RTL Schematic of4 bit Wallace Tree Multiplier](image3.png)

![Fig.4. Logic used in 8 bit Wallace Tree Multiplier](image4.png)
A 8-bit multiplier is constructed by using Wallace tree architecture. The architecture has been shown in Figure 3. Partial products are added in 6 steps. In the Wallace Tree method, the circuit layout is not easy although the speed of the operation is high since the circuit is quite irregular [3]. The delay generated in Wallace tree multiplier can be further reduced by using modified tree structures called compressors.

IV. COMPRESSOR

Compressors are arithmetic components, similar in principle to parallel counters, but with two distinct differences: (1) they have explicit carry-in and carry-out bits; and (2) there may be some redundancy among the ranks of the sum and carry-output bits.

1) 4:2 Compressor

The 4:2 compressor has 4 input bits and produces 2 sum output bits (out0 and out1), it also has a carry-in (cin) and a carry-out (cout) bit (thus, the total number of input/output bits are 5 and 3); All input bits, including cin, have rank 0; the two output bits have ranks 0 and 1 respectively, while cout has rank 1 as well. Thus, the output of the 4:2 compressor is a redundant number; for example, out1 = 0 and cout = 1 is equivalent to out1 = 1 and cout = 0 in all cases.

Fig.5. (a) 4:2 compressor I/O diagram; (b) 4:2 compressor architecture;

2) 5:2 Compressor

The 5:2 compressor has 5 input bits and produces 2 sum output bits (sum and cout3), it also has a carry-in (cin1, cin2) and a carry-out (cout1, cout2, cout3) bit (thus, the total number of input/output bits are 7 and 4); All input bits, including cin1, have rank 0 and cin2 has rank 2 ; the two output bits have ranks 0 and 1 respectively, while cout2 has rank 1 and cout1 has rank 2 as shown in Fig

Fig.6. (a) 5:2 compressor I/O diagram
3) 6:2 Compressor

The 6:2 compressor has 6 input bits and produces 2 sum output bits (out\(_0\) and out\(_1\)), it also has a carry-in (C\(_{in0}\), C\(_{in1}\)) and a carry-out (C\(_{out0}\), C\(_{out1}\)) bit (thus, the total number of input/output bits are 8 and 4); All input bits, including C\(_{in0}\), have rank 0 and C\(_{in1}\) has rank 1; the two output bits have ranks 0 and 1 respectively, while C\(_{out0}\) has rank 1 and C\(_{out1}\) has rank 2 as shown in Fig. 7.

![6:2 compressor I/O diagram](image1)

![6:2 compressor architecture](image2)

![RTL Schematic of 6:2 Compressor](image3)


4) 7:2 Compressor

The 7:2 compressor has 7 input bits and produces 2 sum output bits (out\(_0\) and out\(_1\)), it also has a carry-in (C\(_{in0}\), C\(_{in1}\)) and a carry-out (C\(_{out0}\), C\(_{out1}\)) bit (thus, the total number of input/output bits are 9 and 4); All input bits, including C\(_{in0}\), have rank 0 and C\(_{in1}\) has rank 1; the two output bits have ranks 0 and 1 respectively, while C\(_{out0}\) has rank 1 and C\(_{out1}\) has rank 2 as shown in Fig. 8.

![7:2 compressor I/O diagram](image4)

![7:2 compressor architecture](image5)

![RTL Schematic of 7:2 Compressor](image6)
V. VERIFICATION OF SIMULATION

Xilinx is a powerful simulation tool for simulating and compiling Verilog/VHDL code efficiently. All the basic modules designed for Wallace tree multiplier are compiled and tested vigorously for functional correctness using waveforms. The complete circuit of 8x8 bit Wallace tree multiplier is described in Verilog, as a structural component. The hierarchical structure is created and the complete simulation can be observed. Fig.3 shows the waveform simulation result for the Wallace tree multiplier.

VI. MULTIPLIER PERFORMANCE AND COMPARISON

The performance analysis of Wallace Tree Multiplier using conventional method and using Compressor are listed in Tables 1-2 in terms of number of occupied slices, number of 4 input LUT using Xilinx and power, delay using Cadence tool. The simulation results of number of occupied slices and number of 4 input LUT are shown in Table-1. The simulation results of power and delay are shown in Table-2. Here Wallace Tree Multiplier represents as WTM in Table 1-2.

<table>
<thead>
<tr>
<th>Wallace Tree Multiplier 4x4(WTM)</th>
<th>Number of 4 input LUTs</th>
<th>Number of occupied Slices</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x4 WTM</td>
<td>27</td>
<td>15</td>
</tr>
<tr>
<td>4x4 WTM using 4:2 Compressor</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>4x4 WTM using 5:2 Compressor</td>
<td>37</td>
<td>20</td>
</tr>
<tr>
<td>8x8 WTM</td>
<td>139</td>
<td>75</td>
</tr>
<tr>
<td>8x8 WTM using 5:2 &amp; 4:2 Compressor</td>
<td>139</td>
<td>75</td>
</tr>
<tr>
<td>8x8 WTM using 6:2 &amp; 4:2 Compressor</td>
<td>139</td>
<td>76</td>
</tr>
<tr>
<td>8x8 WTM using 7:2 &amp; 4:2 Compressor</td>
<td>140</td>
<td>74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wallase Tree Multiplier 8x8(WTM)</th>
<th>Number of 4 input LUTs</th>
<th>Number of occupied Slices</th>
</tr>
</thead>
<tbody>
<tr>
<td>8x8 WTM</td>
<td>496</td>
<td>69.917</td>
</tr>
<tr>
<td>8x8 WTM using 4:2 Compressor</td>
<td>5.178</td>
<td>74.642</td>
</tr>
<tr>
<td>8x8 WTM using 6:2 &amp; 4:2 Compressor</td>
<td>5.390</td>
<td>74.309</td>
</tr>
<tr>
<td>8x8 WTM using 7:2 &amp; 4:2 Compressor</td>
<td>5.698</td>
<td>74.278</td>
</tr>
</tbody>
</table>

Table.1 Simulation Result using Xilinx

Table.2 Simulation Result using Cadence

VII. CONCLUSIONS

The Wallace tree multipliers can be solved & analyzed using a new modified method of Wallace tree construction using compressors. The modified tree has a slightly smaller critical path, a slightly larger wiring overhead but gives high speed. This modified design of multiplier which consist of 7:2 compressor ,6:2 compressor ,5:2 compressor ,4:2 compressor, 3:2 compressor, full adders and reduced no. of half adder and reduces the complexity and reduce the time delay.

Multiplier using Compressor have small increase in area and power but the time delay is less compare to conventional Wallace Tree Multiplier. As the Compressor order is increased the time delay reduces respectively. Hence for small delay requirement Wallace Tree Multiplier using compressor is suggested.

References


