Compressors Based High Speed 8 Bit Multipliers Using Urdhava Tiryakbhyam Method

Dr. M. Pushpavalli¹, C. Ragapriya², B. Praveena³

¹,²,³ Department of Electronics and Communication Engineering, Bannari Amman Institute of Technology, INDIA

ABSTRACT

Arithmetic operations are becoming a bigger concern in the digital system for applications like ALU (Arithmetic and Logic Unit) and DSP (Digital Signal Processing). Our work focuses on novel 4:2 and 5:2 Compressors (CM) applied in multiplication architectures such as Unsigned Wallace tree multiplier, Vedic mathematics using Urdhva Tiryakbhyam sutra, and Wallace tree multiplier, Signed Booth multipliers. The proposed compressors architectures have shown better results when compared with the existing compressors. In our project, a design of 8-bit x 8-bit signed multipliers for high speed Digital Signal Processing (DSP) applications was done. Urdhava Tiryakbhyam method is used in the architecture of multipliers to implement high speed and it also has less power consumption and delay. Our proposed work was compared with booth, array and Wallace multiplier in terms of area, delay and power consumption. Our techniques produces better result than the existing multipliers.

Keywords: compressors, multipliers, urdhava tiryakbhyam method

I. INTRODUCTION

High speed multiplication has always been a fundamental requirement of high performance processors and systems. Multipliers play an important role in today’s digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. This work is based on one of the ancient Vedic algorithms (sutras) called Urdhava Tiryakbhyam method.

The next generation of wireless network requires high-throughput and low power Digital Signal Processing (DSP) System on-Chip (SOC). Amongst the building blocks of a DSP system a multiplier is an essential component that has a significant role in both speed and power performance of the entire system. Therefore to enhance the performance of DSP SOCs designing of a high-performance and power efficient multiplier is crucial. In our work we have discussed about the three different multipliers and their working process.

A Wallace tree multiplier is an efficient hardware implementation of a digital circuit that multiplies two integers devised by an Australian computer scientist Chris Wallace in 1964. Wallace tree reduces the number of partial products and use carry select adder for the addition of partial products.

Booth multiplication algorithm gives a procedure for multiplying binary integers in signed -2's complement representation. Booth algorithm requires examination of the multiplier bits, and shifting of the partial product (P).

II. LITERATURE SURVEY

The application which we are considering describes about the information about all the basic Vedic mathematics techniques used for various operations. Among these techniques more preferable method is Urdhva Tiryakbhyam method to describe Urdhava Tiryakbhyam methodology and their hardware architecture details and implementation presented. The algorithm to architecture mapping using floating point number representation Consumes more hardware which tends to be expensive. Fixed point number representation is a good option to implement at silicon level. Hence our
focus in this work is to develop optimized hardware modules for multiplication operation.

Fast and low power MAC unit is most extreme prerequisite of today's VLSI frameworks and advanced sign handling applications like FFT, Finite motivation reaction channels, convolution and so on. creators have examined diverse sorts of multipliers like stall multiplier, combinational multiplier, Wallace tree multiplier, cluster multiplier and consecutive multiplier. Every multiplier has its own particular points of interest and impediments. Finally compressors are used to reduce the latency but use of many adders resulted in many partial products.

Xuan-Vy Luu, Trong-Thuc Hoang, Trong-Tu Bui: High speed multiplication has always been a fundamental requirement of high performance processors and systems. With MOS scaling and technological advances there is a need for design and development of high speed data path operators such as adders and multipliers to perform signal processing operations at very high speed supporting higher data rates. Based on this review suitable modifications are suggested in the design for high speed and low power multipliers[1].

Nilay Nagdeve, Vishal Moyal, Ms. Archana Fande: This work is based on one of the ancient Vedic algorithms (sutras) called Urdhava tiryakbhyam method. These sutras are meant for faster calculation. Though faster when implemented in hardware, it consumes less area[2].

Leonardo L. de Oliveira, Eduardo. C. Sergio B: Present work is an effort to design and implement the multiplier using Urdhava- Tiryakbhyam Vedic Sutra and Nikhilam Sutras and compare their performance with multipliers developed with booth algorithm and Array algorithm specifically in terms of area and speed[3].

Rao, M.J. Dubey: The Wallace tree basically multiplies two unsigned integers. The conventional Wallace tree multiplier architecture comprises of an AND array for computing the partial products, a carry save adder for adding the partial products so obtained and a carry propagate adder in the final stage of addition. In the proposed architecture, partial product generation and reduction is accomplished by the use of booth algorithm[5].

Sureka N, Porselvi R and Kumuthapriya K: Wallace tree sums up all the bits of same weights in a merged tree unlike completely adding the partial products in pairs like the ripple adder does. Usually full adders are used so that three equally weighted bits are combined to produce two bits: the carry with the weight n+1 and sum with the weight n. The Slansky and Kogge Stone adders can be used but these give more delay and power consumption than the carry save adders[10].

III. URDHAVA TIRYAKBHYAM METHOD

Urdhava Tiryakbhyam is a Sanskrit word which means vertically and crosswire in English. The method is a general multiplication formula applicable to all cases of multiplication. It is based on a novel concept through which all partial products are generated concurrently.

The least significant bit (LSB) of the multiplier is multiplied with least significant bit of the multiplicand (vertical multiplication). This result forms the LSB of the product. In step 2 next higher bit of the multiplier is multiplied with the LSB of the multiplicand and the LSB of the multiplier is multiplied with the next higher bit of the multiplicand (crosswire multiplication). These two partial products are added and the LSB of the sum is the next higher bit of the final product and the remaining bits are carried to the next step. The Partial products and their sums for every step can be calculated in parallel.

With \( c6r6r5r4r3r2r1r0 \) being the final product.

\[ AH \ AL \]
\[ BH \ BL \]

\[ (AH \times BH) + (AH \times BL + BH \times AL) + (AL \times BL). \]

Thus we need four 4-bit multipliers and two adders to add the partial products and 4-bit intermediate carry generated. Since product of a 4 x 4 multiplier is 8 bits long, in every step the least significant 4 bits correspond to the product and the remaining 4 bits are carried to the next step which is shown in the fig.3.2. This process continues for 3 steps in this case. Similarly, 16 bit multiplier has four 8 x 8 multiplier and two 16 bit adders with 8 bit carry. Therefore we see that the...
multiplier is highly modular in nature. Hence it leads to regularity and scalability of the multiplier layout.

IV. COMPRESSORS

A compressor is a device which is used to reduce the operands while adding terms of partial products in multipliers.

High speed multipliers use 3-2, 4-2 and 5-2 compressors to lower the latency of partial product reduction part. Compressors are used to minimize delay and area which leads to increase the performance of the overall system. Compressors are generally designed by XOR-XNOR gates and multiplexers.

The most widely and the simplest used compressor is the 3-2 compressor which is also known as a full adder. A 3-2 compressor has three inputs X1, X2, X3 and generates two Outputs they are sum and the carry bits which is shown in fig 3.

V. WALLACE TREE MULTIPLIER

A Wallace tree multiplier is an efficient hardware implementation of a digital circuit that multiplies two integers devised by an Australian computer scientist Chris Wallace in 1964. Wallace tree reduces the no. of partial products and use carry select adder for the addition of partial products.

VI. STEPS IN WALLACE TREE

Wallace tree has three steps:-
1. Multiply each bit of multiplier with same bit position of multiplicand. Depending on the position of the multiplier bits generated partial products have different weights.
2. Reduce the number of partial products to two by using layers of full and half adders.
3. After second step we get two rows of sum and carry, add these rows with conventional adders.

Explanation of second step:-
1. Take any three rows with the same weights and input them into a full adder. The result will be an output row of the same weight i.e sum and an output row with a higher weight for each three input wires i.e carry.
2. If there are two rows of the same weight left, input them into a half adder.
3. If there is just one row left, connect it to the next layer.

VII. ADVANTAGES OF WALLACE TREE

The advantage of the Wallace tree is that there are only $O(\log n)$ reduction layers (levels), and each layer has $O(1)$ propagation delay. As making the partial products is $O(1)$ and the final addition is $O(\log n)$, the multiplication is only $O(\log n)$, not much slower than addition (however, much more expensive in the gate count). For adding partial products with regular adders would require $O(\log n^2)$ time.

Though the process seems to be complex it yields multipliers with delay proportional to the logarithm of the operand word length $n$. The Wallace tree multiplier belongs to a family of multipliers called column compression multipliers.

VIII. ARRAY MULTIPLIERS

In array multiplier the two basic functions, partial product generation and summations are combined. For unsigned $N \times N$ multiplication, $N2+N-1$ cells (where $N2$ contain an AND gate for partial product generation, a full adder for summing and $N-1$ cells containing a full adder) are connected to produce a multiplier. This array generates $N$ lower product bits directly and uses a carry-propagate adder, in this case a ripple carry adder, to form the upper $N$ bits of the product.
IX. FUNCTIONS OF ARRAY MULTIPLIER

The two basic functions of array multiplier, partial product generation and summation are combined. For unsigned N x N multiplication, N^2+N-1 cells are connected to produce a multiplier, where N^2 contain an AND gate for partial product generation, a full adder for summing and N-1 cells containing a full adder. The array generates N lower product bits directly and uses a carry-propagate adder, in this case a ripple carry adder, to form the upper N bits of the product. Replacing full adder with half adders, possibly reduces the complexity to N^2 AND gates, N half adders, and N(N-2) full adders. The worst case delay is (2N-2) Δc , where Δc is the adder delay.

In order to design an array multiplier for two's complement operands, Booth algorithm can be employed. This algorithm computes the partial products by examining two multiplicand bits at a time. Except for enabling usage of two's complement operands, this algorithm offers no performance or area advantage in comparison to the basic array multiplier. Better delays, though can be achieved by implementing a higher radix modified Booth algorithm calculation, thereby increasing overall delays.

X. ALGORITHM FOR ARRAY MULTIPLIER

Step 1: Define variables for multiplier and multiplicand.
Step 2: Use the shift and add method.
Step 3: Cascade 4 blocks of 2x2 bit blocks together to form 8x8 bit multiplier.
Step 4: The result of 8x8 multiplier will be 16 bits.

XI. BOOTH MULTIPLIER

Booth multiplication algorithm gives a procedure for multiplying binary integers in signed -2’s complement representation.

XII. STEPS TO IMPLEMENT BOOTH ALGORITHM

Following steps are used for implementing the booth algorithm:-
Let X and Y are two binary numbers and having m and n numbers of bits (m and n are equal) respectively.

Step 1 Making booth table: In booth table we will take four columns one column for multiplier second for previous first LSB of multiplier and other two (U and V) for partial product accumulator (P).
1. From two numbers, choose multiplier (X) and multiplicand (Y).
2. Take 2’s complement of multiplicand (Y).
3. Load X value in the table.
4. Load 0 for X-1 value.
5. Load 0 in U and V which will have product of X & Y at the end of the operation.
6. Make n rows for each cycle because we are multiplying m and n bits numbers.

Step 2 Booth algorithm: Booth algorithm requires examination of the multiplier bits, and shifting of the partial product(P). Prior to the shifting, the multiplicand may be added to P, subtracted from the P, or left unchanged according to the following rules:
1. Xi Xi-1
    0 0 Shift only
    1 1 Shift only
    0 1 Add Y to U and shift
    1 0 Minus Y from U and shift
2. Take U & V together and shift arithmetic right shift which preserves the sign bit of 2’s complement number. So, positive numbers and negative numbers remains positive and negative respectively.
3. Circularly right shift X because this will prevent us from using two registers for the X value. Repeat the same steps until n no. of cycles are completed. In the end we get the product of X and Y.

XIII. RESULTS AND DISCUSSIONS

SIMULATION RESULTS

Output wave forms of different Multiplier implementations are shown. Different multipliers like Booth multiplier, Wallace Tree multiplier functionality is verified by writing verilog code. And these three multipliers are implemented using xilinx simulator as shown below:

Fig.5 8 Bit booth multiplier result

From the above fig.6.1 it is observed that keeping an input constant and changing the another input results in reduced execution time. Therefore, delay is minimal when switching of inputs are high.

Fig.6 8 Bit Wallace Tree Multiplier result

From the above fig.6.1 it is observed that using reduced number of gates in Wallace tree multiplier...
architecture, we can reduce the delay time which probably results in reduced latency period.

From the above fig.6.1 it is observed that for negative numbers and mixed numbers multiplication in array multiplier shows higher processing speed than normal by using 2’s complement’s succession.

XIV. PERFORMANCE AND COMPARISON

1. Area wise, array multiplier consumes less area as compared to Wallace tree and Booth multiplier.
2. Power wise, Booth multiplier consumes less power compared to Wallace tree and array multiplier.
3. Delay wise, Wallace tree has less delay as compared to Booth and array multiplier.

<table>
<thead>
<tr>
<th>Multiplier</th>
<th>Delay (ns)</th>
<th>Area (μm²)</th>
<th>Power (Mw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>3.02</td>
<td>853</td>
<td>5.23</td>
</tr>
<tr>
<td>Booth</td>
<td>2.81</td>
<td>979.7</td>
<td>5.16</td>
</tr>
<tr>
<td>Wallace</td>
<td>2.73</td>
<td>910</td>
<td>5.39</td>
</tr>
</tbody>
</table>

Table.1 Performance and comparison table

XV. CONCLUSION AND FUTURE WORK

Wallace Tree Multiplier, Booth Multiplier and Array multiplier are implemented and power dissipation, area and propagation delay are calculated. Hence the results are simulated to improve the overall performance of these multipliers.

In our work, we have identified the techniques for designing the selected three 8 bits multipliers namely Booth, Wallace tree and array multiplier by analyzing their performance in terms of delay, area and power characteristics with particular emphasis on designing the cells for optimum power using layout design techniques.

These three multipliers are implemented and the constraints such as area, power and timing are optimized using Verilog codes based on software resources Xilinx ISE and we improved the overall performance of these multipliers in terms of area, power, delay by urdhava Tiryakabyam method using compressors.

REFERENCES