Novel Approach for Fast Partial Product Compressor structure using Quarter Module for 8x8 & Higher Multipliers

Nilay Nagdeve¹, Vishal Moyal²

¹ME IV Sem, Department of ET&T, FET, SSTC, Bhilai
² Assoc. Prof. & Head, Department of ET&T, SSITM, SSTC, Bhilai

ABSTRACT
It is very realistic that the FPGA and CPLD devices are used for Digital Signal Processing and signal testing purposes in this modern era of Digital Signal Handling. While realizing or testing any peculiar module, DSP processors needs a faster multiplication mechanism. The faster multiplication can be accomplished by using a faster multiplication algorithm with reflexions for parameters like Area and Power. Also, in another way, after generating a partial products using a simple multiplication process, use of faster compressors which will transfigures a partial products into a lawful data. This paper will show the efforts taken to generate a faster results using a faster code compressors of \[3:2\], and \[4:2\] value with Non-Booth tactic. Also this paper will throw a focus on designing and implementation of quarter module which is basically a combinational block having capability of 16:8 compression and can be used in symmetric multiplication.

Keywords – Multiplier, Compressor, Non Booth, Spartan, FPGA

I. INTRODUCTION
Some of Faster multiplication procedures focuses on generate a faster results using generation of less partial products and some on generating faster results using compressors. Multipliers consist of three fundamental parts: a partial product generator, a partial product reduction and a final fast adder part [1]-[4]. Generally, to create a valid data, partial products are simply added using Full adders and Half adders with a specific structure suggested like Wallace Tree. Full adder is fundamental unit in various circuits, especially, in performing arithmetic operations such as compressors, comparators, parity checkers, multipliers etc. It is the nucleus of many other useful operations such as subtraction, multiplication, division, exponentiation, address calculation and can significantly influence the overall achievable performances of the system [5].

The stages of multiplication can be categorized as follows: In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial products. The second stage is the most important, as it is the most complicated and determines the speed of the overall multiplier. The 4-2 compressor has been widely employed in the high speed multipliers for the construction of Wallace tree to lower the delay of the partial product accumulation stage [6], [7].

II. FUNDAMENTAL APPROACH
As there are so many patterns to design a same thing, in same way the faster multiplication can be done with various combinations. A general physicist named Andrew D. Booth proposed a multiplication algorithm first time (1950) in order to reduce the partial products using Booth Encoder and Booth Decoder. This paper will basically focuses on Conventional methodology with modern designing for partial product reduction rather than Booth’s Algorithm. Again Full adders were used as a basic building block for generating answer from the partial product

A. Single bit Full Adder
One bit full adder is nothing but a 3:2 compressor which takes three input, mainly two signals and one carry to produce two output bits, mainly sum and one carry output. While implementing a full adder operation, the carry generation is a task which take a delay in output. As only the carry transferred to next consecutive operation, it must be generated fast in order to make the next execution to be started and hence an overall fast execution. The implementation of first adder can be done through utilization of two simple Half-adders or by creating a new logic. Applying the knowledge of digital electronics, the equations are reduced to form a confident structure of Full Adder

B. Single bit 4:2 compressor
These compressors are basically takes 5 inputs and process to provide 3 output though the name shows 4 input and 2 output. It is called compressor, since it compress four partial products into two [8]. The figure 2 shows a simple 4:2 compressor in IN1, IN2, IN3, and IN4 are input and SUM, and Carry0 as output. As the
addition of 5 bits is taking place. The maximum length of output signal is of 3 bits. 0th bit will be the SUM bit, 1st bit will be the carry0 bit which is to be added to next addition and 2nd bit is Carry1 bit which is to be added to one next addition. The thoroughgoing value this compressor can generate is 101 when all the inputs are Logic High.

III. PROPOSED APPROACH
The proposed work is totally based on designing of new [3:2] and [4:2] compressor and Quarter Module by using those compressors.

A. [3:2] Compressor

The implementation of [3:2] can be done by using two [2:2] compressors with extra ORing for carry output. The implemented circuit is as shown:

Alternative calculation that may generate faster sum are shown above in figure 4(a), 4(b), & 4(c). The best way to generate faster carry is to generate carry outputs in which the sum output is logically selected by carry output. The carry output logic is drawn by solving k-map of truth table. Calculations are so done that the carry will select the sum. As the carry input is required for next block to start computing, carry will generate before sum. The circuit can be realized as shown above in figure
2. The timing diagram to verify the truth table by all of the above circuits for [3:2] compressor is as shown in figure 3. Figure also shows the values at intermediate nodes.

B. [4:2] Compressor
Moving towards higher compressors, [4:2] compressor can be made from lower compressors like [2:2] or [3:2]. From the conclusion drawn from previous section, design approaches IV and V can be preferred. Some of the designs for [4:2] compressor will use either IV-IV or V-V or IV-V design combinations. [4:2] compressor adds 4 inputs along with carry and produces sum and carry output. This compressor takes IN1, IN2, IN3, and IN4 with cin, adds them and produces one bit sum, one bit carry and one bit carry travel for second next module. Various possible implementations and proposed combinations are as shown in figure 4 (a), 4 (b), & 4 (c).

C. DESIGNING of a Quarter Module
While calculating a final answer from a partial products generated between the calculation, a symmetrical multiplication (here, symmetrical stands for (2x2, 4x4, 8x8, 12x12...and so on) generates a symmetrical products in a group that can be formed as shown in dot figure 3 for 8x8 multiplication.

The quarter module is nothing but a higher order compressor made up of best of 3:2 and 4:2 compressors. As the carries generated in 5:2 and higher compressors are not manageable, the compressors used here are 3:2 and 4:2.

After implementation of quarter module which is basically a quarter part of overall partial product reduction, the dot product with quarter module result can be realized as follows:

The implemented quarter module technology test bench waveform is as shown:
After getting the results from the quarter modules, the values can be added to get the final result again with the use for compressors discussed above. The multiplication is done with simple AND technique and the addition is done with faster compressors.

IV. COMPARISON & DISCUSSION

The above discussed [3:2] compressors have capabilities to produce sum and carry from inputs and carry input. Indeed all the designs have some faster outputs than others. The designing approach for design II and design III have same timing values, so mentioned in same column.

Table 1 comparison of various [3:2] compressor designs

<table>
<thead>
<tr>
<th>Source pad</th>
<th>Destination pad</th>
<th>Design I</th>
<th>Design II / Design III</th>
<th>Design IV</th>
<th>Design V</th>
</tr>
</thead>
<tbody>
<tr>
<td>In 1</td>
<td>Sum</td>
<td>7.024</td>
<td>7.142</td>
<td>6.785</td>
<td>6.940</td>
</tr>
<tr>
<td>In 2</td>
<td>Sum</td>
<td>7.064</td>
<td>6.976</td>
<td>6.578</td>
<td>6.888</td>
</tr>
<tr>
<td>Cin</td>
<td>Sum</td>
<td>5.990</td>
<td>5.985</td>
<td>6.411</td>
<td>7.060</td>
</tr>
<tr>
<td>In 1</td>
<td>Cout</td>
<td>6.812</td>
<td>6.803</td>
<td>6.945</td>
<td>6.386</td>
</tr>
<tr>
<td>In 2</td>
<td>Cout</td>
<td>6.852</td>
<td>6.637</td>
<td>6.738</td>
<td>6.334</td>
</tr>
</tbody>
</table>

From above bars and comparison table, following conclusions can be made
1) Design I can generate fastest cout from cin.
2) Design II and III can generate fastest sum from cin.
3) Design IV can generate fastest sum from any of the input.
4) Design V can generate faster cout from any one of the input.

Design IV and V are more preferable than other 3 designs while when the carry generation from input is a prime concern, design V will be more preferable. To design higher compressors with the help of [3:2] compressors, design IV and design V is preferable.
V. CONCLUSION
While comparing with the quarter module with and without fast compressors, the maximum path delay for the quarter module is getting as 14.7ns and minimum path delay is 11.0 to generate MSB of the result. The results are approximate 7% faster than normal compressor approach. This represents a Novel approach which is more beneficial than normal.

During the comparison of delay for producing a MSB result bit from LSB input bit, it has been found that, the proposed structure has a smaller delay of (maximum data path delay) 7.913ns than the Wallace tree multiplier (with delay of 8.504ns), which is approximately 7% (6.94%) reduced.

Again the result proves, higher order compressors increases the speed and reduces a delay.

REFERENCES