National Journal of Multidisciplinary Research and Development
ISSN: 2455-9040; Impact Factor: RJIF 5.22
Received: 18-11-2018; Accepted: 20-12-2018
www.nationaljournals.com
Volume 4; Issue 1; January 2019; Page No. 37-40

Improving the performance of dadda multiplier using higher order compressors

Julie Antony Roselin J[1], Augusta Angel M[2], Sujitha A[3]
[1-3] Department of ECE, VV College of Engineering, Tirunelveli, Tamil Nadu, India

Abstract
In this paper, For high performance digital systems, high speed multiplication is the primary requirement. Multiplier plays vital role in order to perform the arithmetic operations in both digital signal processors and microprocessors. Several adder and multiplier designs have been proposed for implementations in exact computing. In this paper, different compressors such as 5:2, 6:2, 7:2 and 8:2 compressors are used for partial product reduction in a dadda multiplier is analyzed and designed. Dadda multiplier is a hardware multiplier and it requires fewer gates. It is more efficient compared to other multipliers. These designs rely on different features of compression such as precision in computation. It is also used to reduce the error rate and provides significant performance improvement compared to an existing methods with respect to delay, number of gates and power consumption.

Keywords: compressor, dadda multiplier, high speed, delay, power

1. Introduction
Multipliers play an important role in today's digital signal processing, micro processors and various other applications. Multipliers are designed to achieve the design targets such as high speed, low power consumption, regularity of layout and less area or even combination of them in one multiplier. It makes the multiplier to achieve various high speed, low power and compact VLSI implementation.
Multipliers are frequently used in DSP, Image processing architectures and microprocessors. Fast Fourier Transform (FFT), Discrete Wavelet Transform (DWT) and auto-correlation are the few important areas where multipliers are mostly used. Multipliers are designed as low power, high speed to reduce latency and power dissipation of a processing system. Most of the multipliers require more gates and power but the efficiency of the multiplier is increased with the usage of less number of gates. The parallel architecture includes dadda multiplier, which is mainly used in high performance applications.
The common multiplication method is "add and shift" algorithm. In parallel multipliers, in order to determine the performance of the multiplier, the main parameter is number of partial products to be added. To achieve high speed, Wallace Tree algorithm can be used to reduce the number of sequential adding stages [8]. If the amount of shifts between the partial products and intermediate sums to be added quickly then it may result in reduced speed, increase in silicon area due to irregular structure and also increased power consumption due to increase in interconnect resulting from complex routing.
On the other hand "dadda" multipliers are used to achieve better performance for area and power consumption. Dadda multipliers provide high speed and requires few gates compared to other multipliers. It is a column compression multiplier and with the help of compressors, the partial products are reduced efficiently. The multiplication operation includes the following three parts.

- Partial product generation.
- Partial product reduction.
- Final computation.

Partial products result from the logical AND of multiplicand bit X with a multiplier bit Y. A carry save adder tree is used to reduce the partial product matrix by adding only two operands. A carry propagation adder is used for the final computation of the binary result.

2. Overview
The dadda multiplier is a hardware multiplier design invented by computer scientist Luigi Dadda in 1965. It is similar to Wallace tree multiplier but it requires fewer gates. Multiplication process can be stated as repeated addition process of partial products. By adding the multiplicand to itself number of times specified by another multiplier repeatedly, then the final product is obtained.
Dadda multiplier is a hardware multiplier designed similar to Wallace multiplier. Unlike Wallace multipliers that perform reductions as much as possible on each layer, Dadda multipliers do as few reductions as possible. Due to this, Dadda multipliers have less expensive reduction phase, but the numbers to be reduced are few bit longer, thus they require somewhat larger adders. This implies that some of the columns are compressed in the initial stages of the column compression tree, and more columns in the later levels of the multiplier.
Dadda multiplier is more efficient compared to other multipliers. It is faster and requires only few gates. The multiplier architecture consists of a partial product generation stage, partial product reduction stage and final addition stage. The partial product reduction stage is responsible for a significant portion of the total multiplication delay, power and area. Therefore in order to accumulate partial products, compressors usually implement this stage because they contribute to the reduction of the partial products and also contribute to reduce the critical path which is important to maintain the circuit's performance. The Wallace tree multiplier is similar to dadda multiplier but it requires more gates. The speed is also slow compared to dadda multiplier. These multipliers are also used in image processing applications such as sharpening, smoothing and image multiplication.

In a parallel multiplier, by using array of AND gates, the partial products are generated. The main problem is the summation of the partial products. Because the time taken to perform this summation only determines the maximum operating speed of multiplier. The Dadda multiplier reduces the number of adder stages in the summation of partial products. To reduce the number of rows in the summation stage, full and half adders are used**.**

Algorithm

Dadda multipliers minimize the number of gates used as well as input and output delay. Maximum height d_j defined by d=2 and it is given by,

$$d_{j+1} = \text{floor}\ (1.5 * d_j)$$

The floor function is the function that takes as input as real number x and gives as output the greatest integer less than equal to x denoted floor(x) = x

This yields a sequence like so, d_1=2, d_2=3, d_3=4, d_4=6, d_5=9, d_6=13.

$$d_j < \max(n_1, n_2)$$

Where n_1 and n_2 represents the number of bits in the input multiplicand and multiplier. To reduce the height of each column equal to the value of d_j is performed by the following ways.

- If height (c_j) ≤ d_j, the column does not require reduction and it moves to the next column c_{i+1}.
- If height (c_j) = d_j+1, add the top two elements in a half adder and placing the result at the bottom of the column and the carry at the top of column c_{i+1}.
- Else if add the top three elements using full adder, placing the result at the bottom of the column and the carry at the top of column c_{i+1}.
- Else add the top elements using compressors, placing the result at the bottom of the column and the carry at the top of column c_{i+1}. Again restart at step c_i.

The higher order compressors such as 5:2, 6:2, 7:2 and 8:2 compressors are used to reduce the partial products. Depends on the size of column, these compressors are used.

3. Module Description

Compressor is a digital modern circuit which is used for high speed with minimum gates requires designing technique [4]. A compressor is a device which is mostly used in multipliers to reduce the operands while adding terms of partial products. The compressors are classified into two types.

- Lower Order Compressor – The number of partial product reduction stages cannot be reduced as much as using the lower order compressors. Hence the delay of multipliers also was not reduced much. It provides uneven signal transition to the multiplier during partial product reduction stage.
- Higher Order Compressor – The higher order compressors were used to improve the performance of the multipliers. They have minimized vertical critical path delay and have further reduced partial product stages and power consumption.

3.1 Dadda Multiplier

The dadda multiplier is a hardware multiplier design. It is invented by Computer scientist Luigi Dadda. It is similar to Wallace tree multiplier but it needs only fewer gates. Multiplication process can be stated as repeated sum of partial products. The final product is obtained by repeatedly adding the multiplicand to itself number of times specified by another multiplier.

Dadda multiplier is more efficient compared to other multipliers. It is faster and requires only few gates. A dot diagram for an 8 by 8 dadda multiplier is shown in figure 1. Two reduction stages are required with matrix heights 8 and 6. In first stage the partial product contains 8 rows and in second stage it will be reduced into 2 rows and the final stage the output is produced. Depends on the height of the column, the compressors are applied to each column. For example, if the height of the column is five then 5:2 compressor is applied, which produces two outputs such as sum and carry.

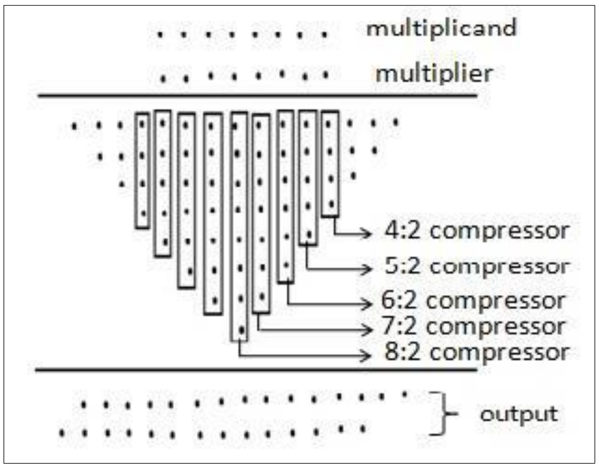


Fig1: Dot diagram of 8×8 Dadda multiplier

The carry bit is added to top of the next column. After first stage of reduction, the height of the column will be reduced into two. Finally, the half adder is used to get the final result. The flow chart describes the partial product reduction of 8×8 bit dadda multiplier is shown in figure

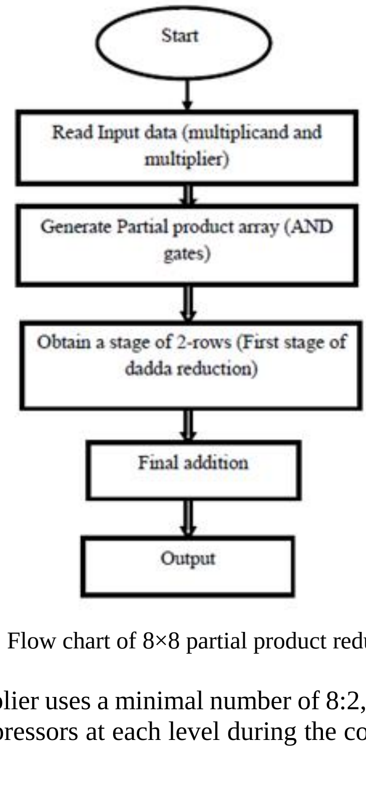

Start
Read Input data (multiplicand and multiplier)
Generate Partial product array (AND gates)
Obtain a stage of 2-rows (First stage of dadda reduction)
Final addition
Output

Fig 2: Flow chart of 8×8 partial product reduction

Dadda multiplier uses a minimal number of 8:2, 7:2, 6:2, 5:2, and 4:2 compressors at each level during the compression to achieve the required reduction. The reduction procedure for Dadda compression trees is developed by the following recursive algorithm.

- Let d_1=2 and dj+1=[1.5*dj] where dj is the height of the matrix for the jth stage. Repeat until the largest jth stage is reached in which the original N height matrix contains at least one column which has more than dj dots.
- In the jth stage from the end, place 8:2, 7:2, 6:2, 5:2, 4:2, 3:2 and 2:2 compressors as required to achieve a reduced matrix. Only columns with more than dj dots as they receive carries from less significant 8:2, 7:2, 6:2, 5:2, 4:2, 3:2 and 2:2 compressors are reduced.
- Let j=j-1 and repeat step 2 until a height of two is generated. This should occur when j=1.

Dadda multipliers perform few reductions only when compared to Wallace multiplier. Because of this, Dadda multipliers have less expensive reduction phase, but if the numbers having longer bits, thus slightly bigger adders are required [10]. To achieve this, the second step is performed by slightly more complex rules than in the Wallace tree multipliers. Dadda multiplier consists of three stages. The partial product matrix is formed in the first stage by N^2AND gates. In the second stage, the partial product matrix is reduced to a height of two.

Dadda replaced Wallace Pseudo adders with parallel (n, m) counters. A Parallel (n, m) counter is a circuit which has n inputs and produce m outputs which provide a binary count of the Ones present at the inputs. A full adder is an implementation of a (3, 2) counter which takes 3 inputs and produces 2 outputs. Similarly a half adder is an implementation of a (2, 2) counter which takes 2 inputs and produces 2 outputs. The 4:2 compressor [6] or (4, 2) counter takes four input and produces two output. The Dadda multipliers reduce the number of rows as much as possible on each layer using the compressors, full adder and half adder circuits.

Table 1: Number of reduction stages for Dadda multiplier

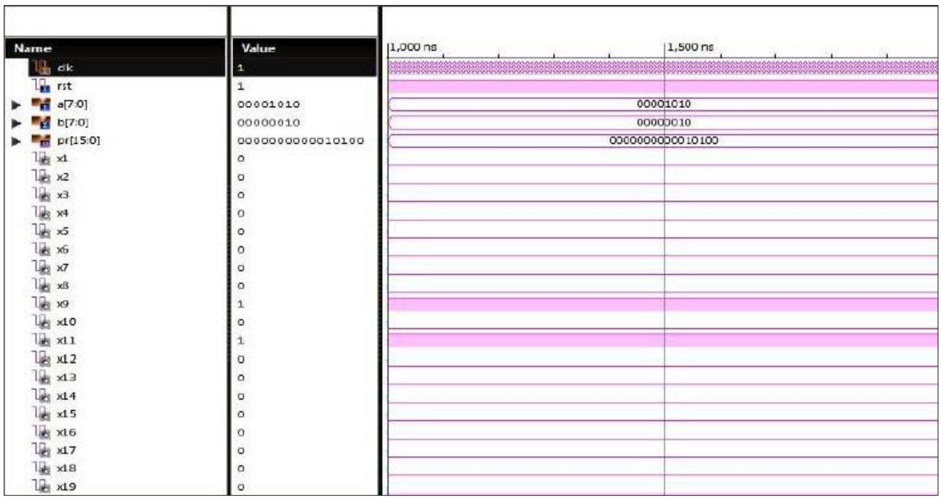
Bits in multiplier(N)	Number of stages
3	1
4	1
$5 \le N \le 6$	2
$7 \le N \le 9$	2
$10 \le N \le 13$	3
$14 \le N \le 19$	4
$20 \le N \le 28$	5
$29 \le N \le 42$	6
$43 \le N \le 63$	7
$63 \le N \le 94$	8

The table 1. Shows the number of reduction stages required to implement Dadda architecture for various numbers of bits. It consists of number of bits in the multiplier and number of stages. The dadda multiplier requires less stages compared to the Wallace tree multiplier. Dadda multiplier reduces the usage of number of gates.

4. Results and Discussion

The design of dadda multiplier is coded in Verilog language. The simulation and synthesis results are obtained in Xilinx ISE 13.2. The power analysis results are obtained in Quartus II 9.0.

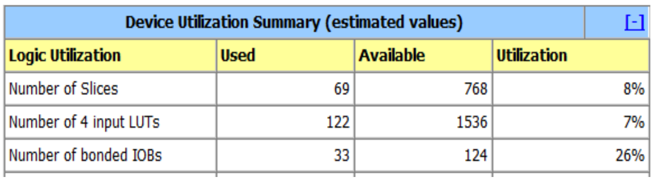
For 8×8 binary multiplication using dadda multiplier, the simulation is performed in Xilinx ISE 13.2 and the output waveform is shown in figure 3.



```
Name        Value
clk         1
rst         1
a[7:0]      00001010
b[7:0]      00000010
pr[15:0]    0000000000010100
x1          0
x2          0
x3          0
x4          0
x5          0
x6          0
x7          0
x8          0
x9          1
x10         0
x11         1
x12         0
x13         0
x14         0
x15         0
x16         0
x17         0
x18         0
x19         0
```

Fig 3: Simulation result of Dadda multiplier

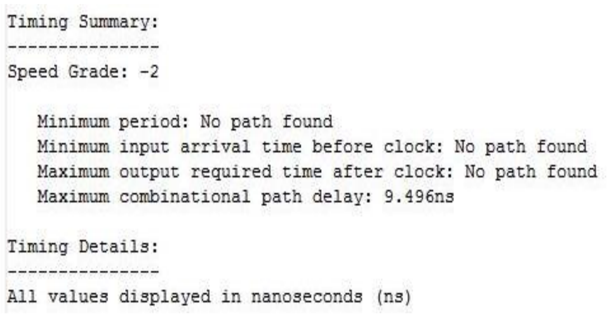
From the figure 3, the two values a and b i.e., 00001010 and 00000010 are given as the input and it produces the output p has the value of 0000000000010100. Synthesis results are obtained using Xilinx ISE 13.2. The figure 4 shows the number of slices utilized by the dadda multiplier is 69. And it also shows the number of 4 input LUTs are 122 and the number of bonded IOBs are 33.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	69	768	8%
Number of 4 input LUTs	122	1536	7%
Number of bonded IOBs	33	124	26%

Fig 4: Device utilization result of Dadda multiplier

Figure 5 shows the power analysis of dadda multiplier. The power analysis of dadda multiplier is obtained using Quartus II 9.0. Power analysis summary contains the details of static thermal power dissipation and dynamic thermal power dissipation.
Figure 6 shows the synthesis summary of dadda multiplier. The total delay of the dadda multiplier is 9.496ns which is shown in the synthesis summary report.



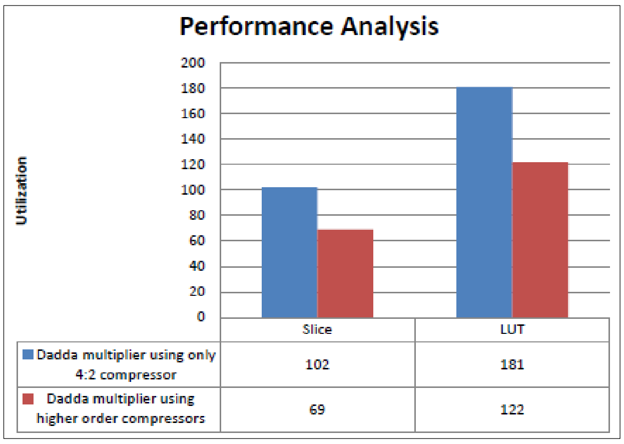
```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 9.496ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)
```

Fig 6: Synthesis summary of Dadda multiplier

Performance analysis of multiplier is plotted in a graph based on their slice and LUT utilization in figure 7.



	Slice	LUT
🟦 Dadda multiplier using only 4:2 compressor	102	181
🟥 Dadda multiplier using higher order compressors	69	122

Fig 7: Performance analysis of multiplier

5. Conclusion

In this paper, the dadda multiplier is designed with the help of different compressors such as 5:2, 6:2, 7:2 and 8:2 compressors. The higher order compressors are used to reduce the partial products efficiently than the lower order compressors. This multiplier is suitable for multiplication of large number of bits. The simulation results demonstrate that the dadda multiplier delivers less area, delay and low power utilization. From the obtained results, the dadda multiplier has the best results in the reduction of slices, LUTs, power consumption and delay. In future work, approximate compressor is used to design a dadda multiplier and to reduce the partial products efficiently.

6. References

1. Akbari O, Kamal M, Afzali-Kusha M, Pedram M. RAP-CLA: A reconfigurable approximate carry look-ahead adder, IEEE Trans. Circuits Syst. II, Express Briefs DOI: 10.1109/TCSII.2633307,2016
2. Dandapat A, Ghosal S, Sarkar P, Mukhopadhyay D, A 1.2ns 16*16-Bit Binary Multiplier Using High Speed Compressors. International Journal of Electronics and Communication Engineering. 2010; 4(3).
3. Baran D, Aktan M, Oklobzija V. Energy Efficient Implementation of Parallel CMOS Multipliers with Improved Compressors, in Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD), Austin, Texas, USA. 2010; pp. 978-4503.
4. Marimuthu, Bansal D, Balamurugan S, Mallick S. Design of 8-4 and 9-4 Compressors for High Speed Multiplication. American Journal of Applied Sciences, ISSN: 1546-9239, 2013.
5. Momeni A, Han J, Montuschi P, Lombardi F. Design and analysis of approximate compressors for multiplication, IEEE Trans Comput. 2015; 64(4):984-994.
6. Akbari O, Kamal M, Afzali-Kusha A, Pedram M. Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers, IEEE Transaction. VLSI, 2017.
7. Radhakrishnan D, Preethy AP. Low Power High Speed 3-2 Compressor, IEEE Trans. Computers. 2010; 3:1296-1298.
8. Masumdar RE. Design of High Performance Wallace tree Multiplier Using Compressors and Parallel Prefix Adder. International Journal of Electrical, Electronics and Data Communication, ISSN: 2320-2084. 2016; 4(10).
9. Shetty R, Neelagar MB. Design and Implementation of High Performance 4-bit Dadda Multiplier Using Compressor", International Journal of Computer Science and Mobile Computing (IJCSMC). 2017; 6(7):249-254.
10. Ramesh AP. Implementation of Dadda and Array Multiplier Architectures Using Tanner Tool, International Journal of Computer Science & Engineering Technology (IJCSET), ISSN: 2229-3345. 2011; 2(2).