FPGA Implementation of a High Speed Multiplier Employing Carry Lookahead Adders in Reduction Phase

Abhay Sharma
M.Tech Student
Department of ECE
MNNIT Allahabad, India

ABSTRACT
Tree Multipliers are frequently used to reduce the delay of array multipliers. The objective of tree multipliers is to utilize the concept of carry save adders in reducing the partial product. Two well known tree multipliers Wallace and Dadda uses full adders and half adders for the aforesaid purpose. This paper implements a multiplier which will perform reduction of partial products using 4 bit Carry Lookahead Adders primarily instead of Full adders. This will result in fewer reduction stages as Full adders reduces 3 partial products bits to 2 giving a 1.5 to 1 ratio whereas 4 bit CLA will reduce 9 partial products bits to 5 giving 1.8 to 1 ratio. Xilinx Spartan 3E FPGA board is used for implementation of structural verilog code for the multiplier design.

Keywords
FPGA, Multiplier, Wallace, Dadda, Carry Lookahead Adders.

1. INTRODUCTION
High speed multiplication is required in almost all computing systems. Various methods have been proposed which generally are some forms of Wallace/Dadda tree multipliers, they performs the multiplication in three stages [1, 2]. In the first stage partial products are formulated utilizing collection of AND gates. Second stage concentrates on reducing the partial product bits using the principle of carry save adders [3]. Carry save addition is the idea of utilizing addition without carries connected in series but just to count. So a full adder can take 3 partial product bits as input and produce 2 partial product bits as output. Hence, can be thought of 3:2 compressor. Number of such full adders can be employed in parallel to form a tree structure which reduces the partial product bits down to sum and carry vectors. In the final stage sum and carry vectors are added using carry propagate addition to produce the resultant product. The total delay of such multiplication is the sum of delays of all three stages. This paper focuses on reducing the delay of the second stage which will significantly reduce the overall delay of the multiplier.

FPGA (Field Programmable Gate Array) is used to verify the design as it offers a rapid design cycle with the flexibility of reprogramming [4]. It consists of logic blocks which can be connected together using programmable interconnects. FPGAs can be configured to perform any logical operation with the help of HDL (Hardware Description Language). Here, we will be using Verilog HDL for describing the multiplier architecture.

2. WALLACE TREE MULTIPLIER
Wallace multiplier implements an aggressive approach to the reduction process where maximum possible reduction is done at every stage. Partial Product matrix rows are grouped into non-overlapping sets of three. Within each three row set, Full Adders reduce columns with three bits and Half Adders reduce columns with two bits. Rows that are not the part of three row set are transferred to the next stage for reduction. When used in multiplier trees, Full Adders and Half Adders are often referred to as (3, 2) and (2, 2) counters. This reduction method is applied to each successive stage until only two rows remain. This process is illustrated by 8 bits by 8 bits Wallace reduction through dot diagram shown in Fig. 1.

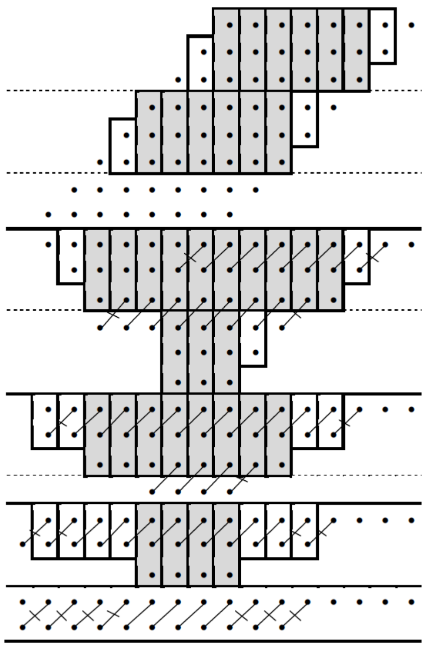


Fig. 1. 8 bits by 8 bits Wallace reduction

3. DADDA TREE MULTIPLIER
Dadda proposed a sequence of matrix heights that are predetermined to give minimum number of reduction stages. The reduction process is formulated using the following recursive algorithm:-

1. Let $d_1 = 2$ and $d_{j+1} = [1.5 \bullet d_j]$, where d_j is the matrix height for the jth stage from the end. Then, proceed to find the smallest j such that at least one column of the original partial product matrix has more than d_j bits.
2. In the jth stage from the end, employ (3, 2) and (2, 2) counters to obtain a reduced matrix with no more than d_j bits in any column.

3. Let j = j-1 and repeat step 2 until a matrix with only two rows is generated.

This method of reduction is called column compression because it tries to compress each column. One of the major advantage of using Dadda is that it utilizes minimum number of (3, 2) counters. The stage height limits (starting from the end) are: 2, 3, 4, 6, 9, 13, 19, 28, 42, etc. 8 bits by 8 bits Dadda reduction is shown in Fig. 2.

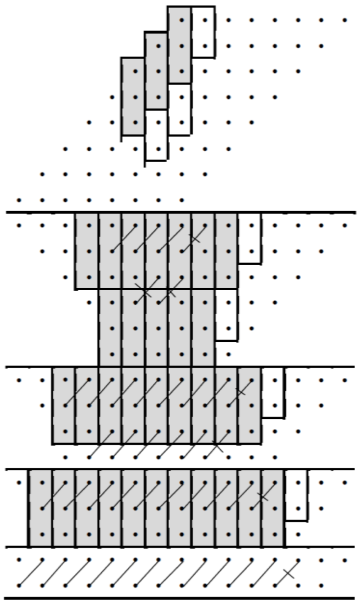


Fig. 2. 8 bits by 8 bits Dadda reduction

4. CARRY LOOKAHEAD ADDER

In case of Ripple Carry Adders (RCA) carry is computed and forwarded to the subsequent adders. This algorithm is improved by pre-computing the carries ahead of time. This modification results in an implementation known as Carry Lookahead Adder (CLA) [5]. In this method the recursive equation for carry-out relating to carry-in is recursively utilized to form all necessary carries. As a result, implementation has logarithmic ordered delay.

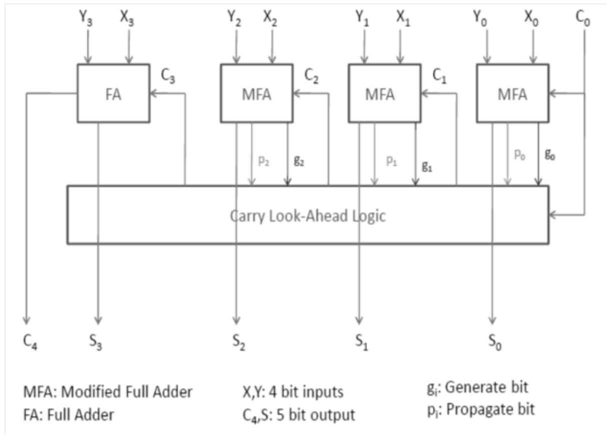


Fig. 3. 4 bit Carry Lookahead Adder

The carry lookahead adders in this paper are constructed with 2-input to 4-input AND and OR gates and inverters. Each gate is assumed to have one gate delay and counts as one gate for complexity. Using these gates, CLA4s are constructed with 4 modified full adders (MFA) and one 4-bit lookahead logic block.

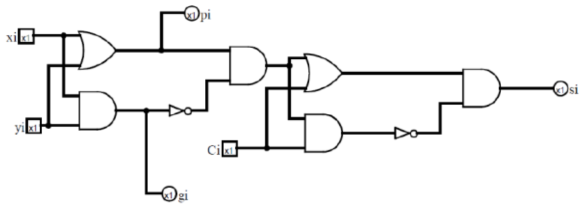


Fig. 4. Modified Full Adder (MFA)

The CLA critical path can be determined by beginning at any of the MFAs input through to the final sum bit, S4. After 1 gate delay, Propagate and Generate bits are available. Carry bits are available after 2 gate delays as they require one AND operation and one OR operation. Moreover, 3 gate delays are consumed from the carry input to the sum output in a full adder. This results in a total of 6 gate delays which is equivalent to a 9 gate full adder.

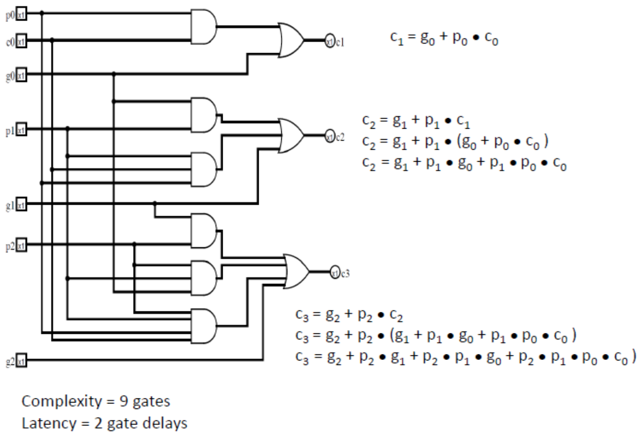

p0
c0
g0
p1
g1
p2
g2
c1
c2
c3
$c_1 = g_0 + p_0 \bullet c_0$
$c_2 = g_1 + p_1 \bullet c_1$
$c_2 = g_1 + p_1 \bullet (g_0 + p_0 \bullet c_0)$
$c_2 = g_1 + p_1 \bullet g_0 + p_1 \bullet p_0 \bullet c_0$
$c_3 = g_2 + p_2 \bullet c_2$
$c_3 = g_2 + p_2 \bullet (g_1 + p_1 \bullet g_0 + p_1 \bullet p_0 \bullet c_0)$
$c_3 = g_2 + p_2 \bullet g_1 + p_2 \bullet p_1 \bullet g_0 + p_2 \bullet p_1 \bullet p_0 \bullet c_0)$
Complexity = 9 gates
Latency = 2 gate delays

Fig. 5. Carry Lookahead Logic

5. MULTIPLIER DESIGN WITH CLA4 IN REDUCTION PHASE

The performance of the multiplier can be improved by using CLA in reduction process as compared to Wallace and Dadda multiplier which uses full adders and half adders for partial product reduction [6]. Four bit CLA can take 9 partial product bits as input and give 5 partial product bits as output. Hence, such 4 bit CLAs (CLA4) can be implemented in parallel during multiplication reduction phase giving the same delay when using full adders and half adders instead. CLAs taking more than 4 bits as input have deteriorating effect on the performance in reduction phase. This is due to fact that CLAs taking more than 4 bits as inputs require greater than or equal to 10 gate delays to generate the outputs. This nullifies the benefit of using CLAs with larger input size. For instance, a stage that has 10 gate delays means that it should perform roughly twice the reduction that Wallace/Dadda does on every stage which is not possible even with the best CLA partial product reduction ratios. Realizing a faster design with CLA4s is viable because CLA4s have the same delay as a Wallace/Dadda stage (6 gate delays). Due to fewer reduction stages with CLA4s, the overall speed of the multiplier is faster than a Wallace/Dadda multiplier.

Best performance can be achieved by employing as many CLA4 as possible. Though, after first stage of reduction the structure of CLA4 multiplier becomes less regular. Greedy placement approach may not be applicable for all cases as it may incur an extra stage of delay. This is generally observed in the final stages wherein combination of Full/Half adders can be employed to restrict the complexity.

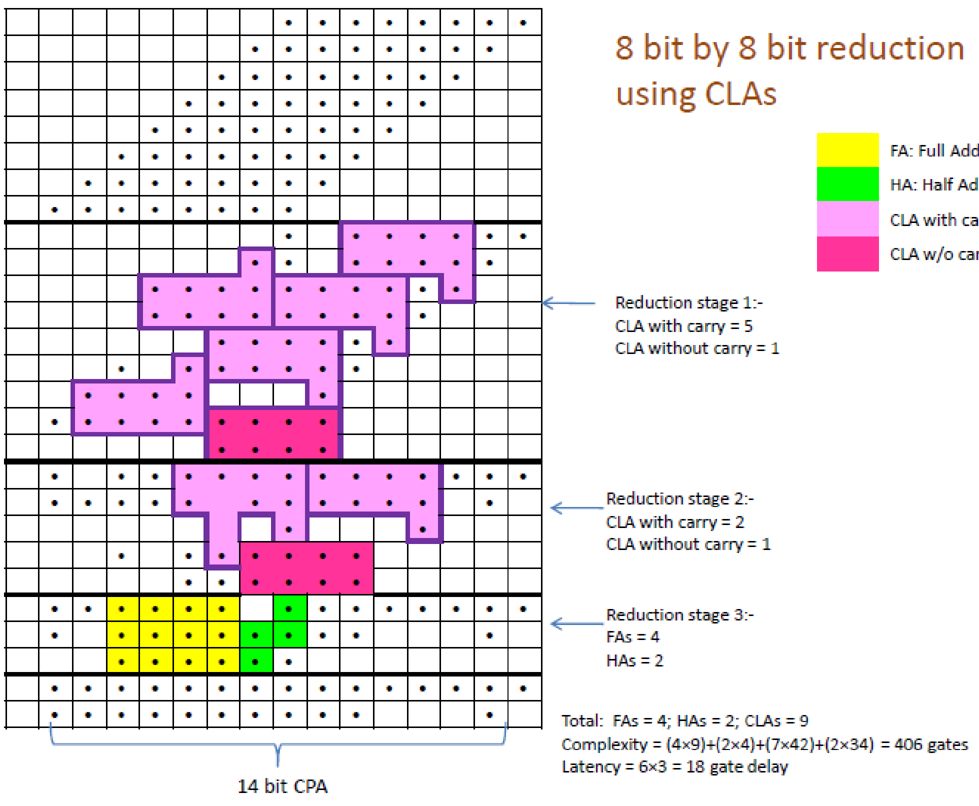


Fig.6. Dot Diagram for 8 bit reduction using 4 bit CLA

6. FPGA IMPLEMENTATION

A field programmable gate array (FPGA) is a logic device that contains a two dimensional array of generic logic cells and programmable switches. Hardware description languages such as Verilog are used to describe and model a digital system [7]. Here we are using verilog to model multiplier design and perform synthesis to transform HDL to generic gate level components. Design is then implemented on Spartan 3E starter board by Digilent. Xilinx ISE 10.1 is used for design entry and implementation.

6.1 Verilog Description

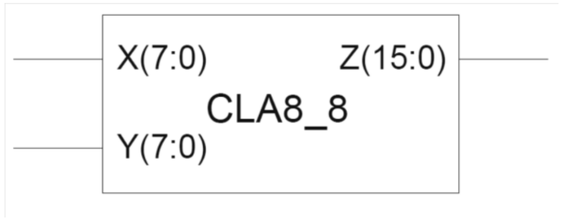
Verilog description of 8 bit by 8 bit reduction using carry Lookahead adder is given below.

```
module CLA8_8(Z, X, Y);
input [7:0] X, Y; // 8 bit inputs
output [15:0] Z; // 16 bit output
wire [7:0] P7,P6,P5,P4,P3, P2, P1, P0;//for partial products
//instantiate Partial Product Generation
PP pp1 (P7,P6,P5,P4,P3, P2, P1, P0, X, Y);
//Level one Reductions
buf b1(Z[0],P0[0]);
CLA4
a0(s00,s01,s02,s03,c0,P0[2],P0[3],P0[4],P0[5],P1[1],P1[2],P1[3],P1[4],
P2[0]);
CLA4
a1(s10,s11,s12,s13,c1,P2[2],P2[3],P2[4],P2[5],P3[1],P3[2],P3[3],P3[4],
P4[0]);
CLA4
a2(s20,s21,s22,s23,c2,P4[2],P4[3],P4[4],P4[5],P5[1],P5[2],P5[3],P5[4],
P6[0]);
CLA4
a3(s30,s31,s32,s33,c3,P0[6],P6[1],P6[2],P6[3],P1[5],P7[0],P7[1],P7[2],
1'b0);
```

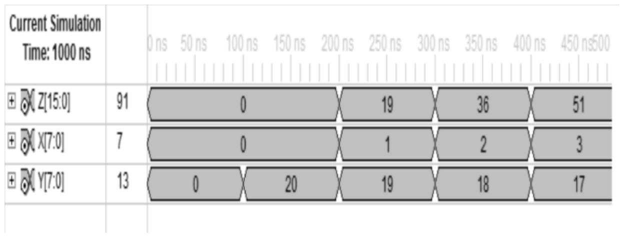
```
CLA4
a4(s40,s41,s42,s43,c4,P2[6],P2[7],P3[7],P4[7],P3[5],P3[6],P4[6],
P5[6],P1[7]);
CLA4
a5(s50,s51,s52,s53,c5,P6[4],P6[5],P6[6],P7[7],P7[3],P7[4],P7[5],
P7[6],P5[5]);
//Level Two Reductions
CLA4
a6(s60,s61,s62,s63,c6,s01,s02,s03,c0,P2[1],s10,s11,s12,P3[0]);
CLA4
a7(s70,s71,s72,s73,c7,P4[1],s20,s21,s22,P5[0],s30,s31,s32,1'b0);
CLA4
a8(s80,s81,s82,s83,c8,P0[7],s40,s41,s42,s13,c1,s23,s50,P1[6]);
//Level Three Reductions
HA ha1(c6,s72,s0,cc0);
HA ha2(s73,s81,s1,cc1);
FA fa1(s33,c7,s82,s2,cc2);
FA fa2(c2,c3,s83,s3,cc3);
FA fa3(s43,s51,c8,s4,cc4);
FA fa4(c4,s52,P5[7],s5,cc5);
endmodule
```

Module CLA8_8 desribes the reduction phase of multiplier. Here partial product (PP) generation was instantiated to generate the PP matrix using module named as PP which uses AND gate arrays for PP generation. Modules for Half Adder, Full Adder and 4 bit Carry Lookahead Adder were also instantiated for the use in reduction process. For the multiplier implementation 14 bit CPA module was written separately.

6.2 RTL Schematic



6.3 ISIM Simulation output waveform


Current Simulation Time: 1000 ns
0 ns 50 ns 100 ns 150 ns 200 ns 250 ns 300 ns 350 ns 400 ns 450 ns 500
Z[15:0] 91 0 19 36 51
X[7:0] 7 0 1 2 3
Y[7:0] 13 0 20 19 18 17

7. RESULTS

Verilog code for Array Multiplier, Wallace Multiplier, Dadda Multiplier and Multiplier using CLA were synthesized and implemented on the FPGA Board and their results were compared in Table 1 and Table 2.

Table 1. Estimated Logic Utilization

Multiplier Type	**Input Size (Bits)**					
	4		8		12	
	No. of slices	No. of 4 input LUT	No. of slices	No. of 4 input LUT	No. of slices	No. of 4 input LUT
Array	21	36	139	248	361	638
Wallace	22	39	116	205	294	520
Dadda	23	39	124	218	293	517
CLA4	19	32	95	167	247	432

Table 2. Maximum Combination Path Delay

Multiplier Type	**Input Size (Bits)**		
	4	8	12
Array	16.023ns	42.398ns	65.456ns
Wallace	18.534ns	34.629ns	46.873ns
Dadda	17.864ns	30.750ns	40.686ns
CLA4	14.287ns	29.403ns	37.602ns

7.1 Observations

It can be observed from Fig.7 that the 8 bit multiplier employing CLA4 provides the least delay as compared to earlier tree structures. This is due to the fact the number of reduction stages were reduced when implementing the multiplier with CLA4. Moreover, as the input word size increases CLA4 provides faster multiplier structures. Another interesting observation is that not only the latency reduced, estimated logic utilization of the FPGA also decreased which will certainly help in power optimization goals.

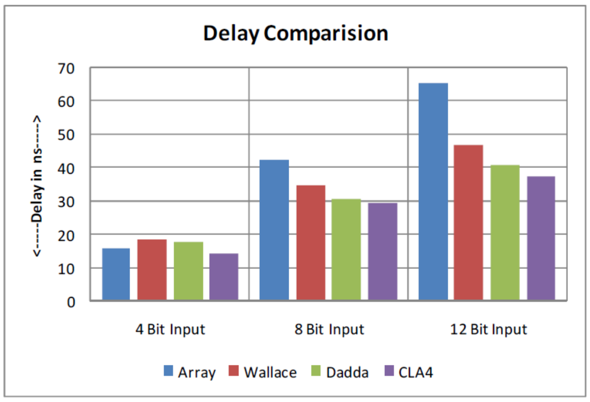


Fig.7. Delay assessment of various tree structures for different input word size

From the analysis of various structures it was found that the possible drawback of employing CLA4 in the reduction phase may result in large carry propagating adder (CPA) in the final stage of multiplication. CPA has a delay which increases every time its input size increases by a power of four. One possible solution to overcome this problem is the use of carry select adder in the final summation stage [8].

8. CONCLUSION

In this paper we have illustrated the use of 4 bit Carry Lookahead Adders in the reduction phase of a multiplier design which resulted in lesser number of reduction stages. Due to fewer stages overall delay of the multiplier computation were reduced. CLA with input sizes more than 4 bits are not used as their latency will not be same as that of Full Adder hence even though it will reduce the stages but overall delay of the multiplier will be higher. We used dot diagrams for the analysis of various multiplier designs and verified the same by implementing the verilog code on FPGA using integrated simulation environment provided by Xilinx. Moreover from the synthesis results it was observed that maximum combinational path delay for multiplier with CLA4 in the reduction phase is the least as compared to Array, Wallace and Dadda Multiplier. Device Logic utilization for the design was also better than other structures. Using the proposed concept we can design tree structures for various datapath units such as Multiply-Accumulate unit, mantissa multiplication in Floating Point Multiplier, Fused Dot Product unit.

9. REFERENCES

[1] C.S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Transactions on Electronic Computers, vol. 13, pp. 14-17, 1964.

[2] L.dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, 1965.

[3] Behrooz Prahami, *Computer Arithmetic Algorithms and Hardware Designs, New* York: Oxford University Press, 1999.

[4] Pong P. Chu, *FPGA Prototyping by Verilog examples: Xilinx Spartan 3 Version*, John Wiley & Sons Publication, 2008.

[5] M. Morris Mano, Ciletti, *Digital Design*, 4th edition, Pearson, 2008.

[6] W. Chu, A.I. Unwala, P. Wu & E.E. Swartzlander, Jr., “Implementation of a High Speed Multiplier Using Carry Lookahead Adders,” Asilomar, pp. 400-404, 2013.

[7] Samir Palnitkar, “Verilog HDL: A Guide to Digital Design and Synthesis,” 2nd Edition, Pearson, 2003.

[8] Shruti Dixit, Praveen Kumar Pandey, “FPGA implementation of Wallace tree multiplier using CSLA/CLA,” International Journal of Science and Research, vol. 2, issue 12, pp. 314-318, Dec. 2013.