Partial Product Reduction in Multipliers using Long Carry Lookahead Adders

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This paper provides a more efficient implementation of Wallace/Dadda multipliers, by replacing the half and full adders used in the partial product reduction stage with more efficient long carry lookahead adders (CLA). The use of long CLAs leads to a reduction in the partial products being generated, which reduced the number of reduction stages required, and leads to a more efficient design that is smaller in size and consumes less power. Thus making it more appealing for portable and low-powered applications. Several multipliers of various lengths have been designed with the long CLA approach, and there performance metrics are presented in this paper. The designs are implemented in verilog HDL using the FreePDK45 libraries, and simulated using synopsys VCS. The proposed design in general requires about 20% less power and is 5% smaller than a conventional Wallace or Dadda multiplier. Note: the long CLA based designs has approximately the same time delay as any conventional Wallace/Dadda multiplier, using non-parallel prefix adder in the final stage.**

***Index Terms*—Computer arithmetic, multiplier, partial product reduction, carry lookahead adder.**

I. Introduction

Multiplication is the second most commonly used arithmetic operation after addition, and thus warrants an implementation that offers high performance. Since many portable applications need low power consumption more than low latency, the primary focus of this approach is to reduce power requirements, with secondary attention to reducing the delay.

Many fast multipliers are variations of the original Wallace [2] and Dadda [3] multipliers. These multipliers generally consists of 3 main steps. The first step is the formation of partial products, also known as partial product generation (PPG). A total of N^2 AND gates are used to generate N^2 partial products as shown in figure 1. The second step is to reduce these partial products into an equivalent 2 rows of partial products, also known as partial product reduction (PPR). This step consumes the most power and requires the most delay in a multiplier. Therefore, this is where most of our research effort is focused. The final step is the addition of the final 2 rows of partial products using a carry propagating adder (CPA), also known as partial product summation (PPS). Therefore, the overall delay of a multiplier is defined as

$$Delay_{multiplier} = T_{PPF} + T_{PPR} + T_{PPS} \quad (1)$$

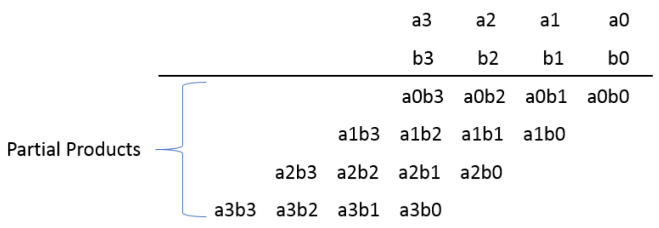

Fig. 1. Partial Product Generation

This paper is organized as follows; Section II provides a brief overview of Wallace/Dadda implementation, and previous approaches to using short CLAs within multipliers; while Section III presents the methodology and implementation of the new proposed multiplier design, along with its design complexity and gate delays. Section IV provides a performance comparison amongst various multiplier designs based on area, power consumption and timing delays. Finally, our conclusion is presented in Section V.

II. Background And State-Of-The-Art

The implementation of long CLA based multiplier is based on the systematic reduction of partial products proposed by Wallace/Dadda multipliers, and then extends the design of short CLA based multipliers to use long CLAs in a more efficient way.

A. Wallace and Dadda Multiplier

Wallace and Dadda are efficient implementations of parallel multipliers. Both multipliers uses the same approach to generate partial products using AND gates. However, they take different approaches for the reduction of partial products

The Wallace multiplier implements aggressive partial product reduction in every stage, as shown in figure 2. For each column of the partial product matrix, groups of 3 dots (partial products) form the inputs to a full adder (FA), while groups of 2 dots form the inputs to a half adder (HA). Both HAs and FAs generate a sum in the same column, while the carry out is added to the next column. Table I records the number of HAs and FAs used Wallace multipliers to help analyze the complexity of the design.

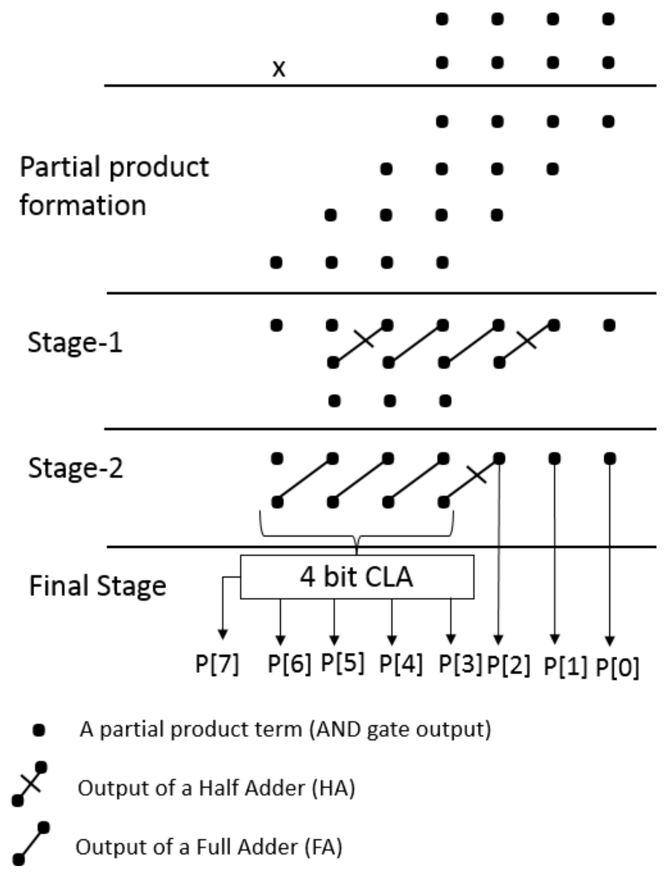
x
Partial product formation
Stage-1
Stage-2
Final Stage
4 bit CLA
P[7] P[6] P[5] P[4] P[3] P[2] P[1] P[0]
A partial product term (AND gate output)
Output of a Half Adder (HA)
Output of a Full Adder (FA)

Fig. 2. 4x4 Wallace Multiplier

	4x4	8x8	16x16	NxN
# of Half Adders (HA)	3	15	52	$>$ N-1
# of Full Adders (FA)	5	38	200	$N^2 - 4N + 2 + S$
Width of CPA	4	10	25	$\approx 2N - 1 - S$
Number of Stages (S)	3	5	6	$\approx log_{1.5}(N)$

TABLE I
TABLE OF COUNTING COMPONENTS IN A WALLACE MULTIPLIER [4]

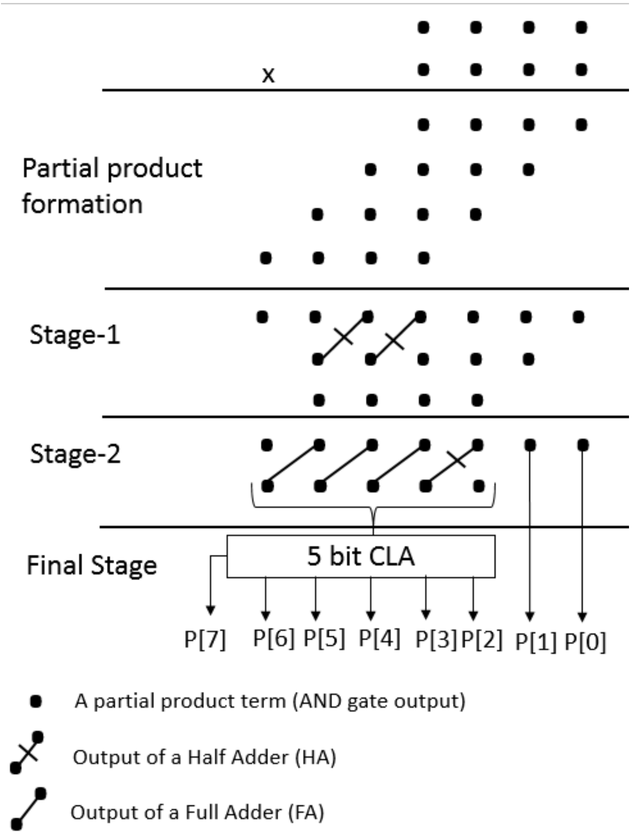
x
Partial product formation
Stage-1
Stage-2
Final Stage
5 bit CLA
P[7] P[6] P[5] P[4] P[3] P[2] P[1] P[0]
A partial product term (AND gate output)
Output of a Half Adder (HA)
Output of a Full Adder (FA)

Fig. 3. 4x4 Dadda Multiplier

	4x4	8x8	16x16	NxN
# of Half Adders (HA)	3	7	15	N-1
# of Full Adders (FA)	3	35	195	N^2-4N+3
Width of CPA	6	14	30	2N-2
Number of Stages (S)	3	5	6	$\approx log_{1.5}(N)$

TABLE II
TABLE OF COUNTING COMPONENTS IN A DADDA MULTIPLIER [4]

The Dadda multiplier on the other hand, does the minimum possible reduction necessary at each stage to achieve the predetermined height (number of partial products in each column), while using the same number of stages as a Wallace multiplier. Each stage in the Dadda multiplier is 1.5 times the height of the next stage, and can be expressed using equation 2. A 4x4 implementation of a Dadda multiplier is shown in figure 3, while table II records the number of HAs and FAs used within Dadda multipliers of different sizes.

$$Height_{Stage-Dadda} = Height_{Stage-1}/1.5 \qquad (2)$$

Based on the results from table I and table II, it is evident that Dadda multipliers requires the least number of HAs and FAs, and is therefore the most efficient type of multiplier, with the least time delay. One disadvantage of Dadda is that is uses a slightly wider CPA in the final stage as compared to Wallace. An approach similar to Dadda will be used in our proposed final design, where only minimal partial product reduction will be done in each stage.

B. Short Carry Lookahead Adder (CLA) based Multiplier

The use of CLAs have previously been proposed for partial product reduction [5] [6]. These implementations have limited the design to using up to 4 bit wide CLAs. Figure 4 shows such an implementation for an 8x8 multiplier; while table III records the number of different adders used in short CLA design for various multiplier lengths.

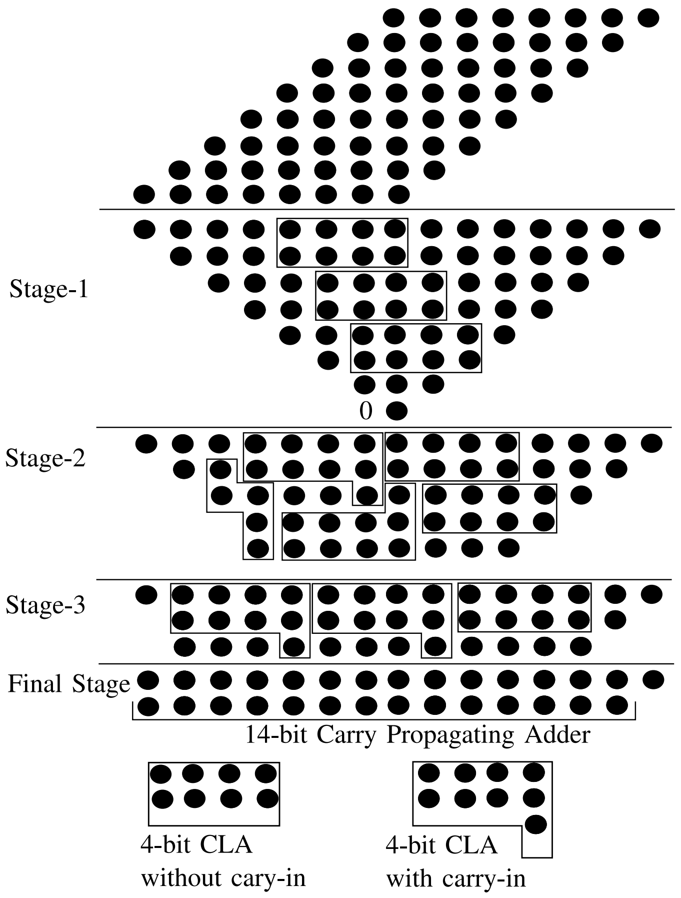

Fig. 4. Dot diagram for an 8x8 short CLA based Multiplier

	8x8	**16x16**	**32x32**	**64x64**
CLA4	10	51	244	1079
CLA3	0	0	9	85
CLA2	1	3	12	150
FA/HA	0	0	24	298

TABLE III
COMPONENTS USED IN SHORT CLA BASED MULTIPLIER

III. METHODOLOGY AND IMPLEMENTATION

This section focuses on the implementation of our proposed long CLA based multiplier. It briefly explains the implementation of a CLA, and why the use of long CLA is better than short CLA in achieving an efficient multiplier design. We also presented a detailed walkthrough of an 8x8 multiplier using the long CLA approach.

A. Carry Lookahead Adder (CLA)

The CLA can generate carries faster than a conventional multiple bit adder, by calculating carry signal in advance based on inputs. It is based on the logic that a carry signal can only be generated when 1) both inputs are high, or 2) one of the inputs is high and the carry in is also high. Therefore the resulting carry out (c_{i+1}) and sum (s_i) can be expressed using equations 3 and 4 respectively, where a_i and b_i are the two inputs, and c_i is the carry in.

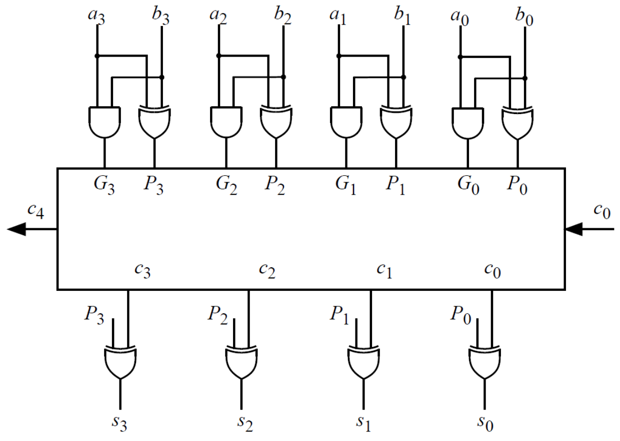

Fig. 5. 4 bit Carry Lookahead Adder

$$c_{i+1} = a_i b_i + (a_i \oplus b_i)c_i \tag{3}$$

$$s_i = (a_i \oplus b_i) \oplus c_i \tag{4}$$

$$P_i = a_i \oplus b_i \tag{5}$$

$$G_i = a_i b_i \tag{6}$$

In our experiments, the CLAs are constructed using 2-input AND and XOR gates, and inverters. For simplicity each gate is assumed to have one gate delay and also count as one gate for complexity. The above two equations can be used to generate two new signals P_i (Propagate) and G_i (Generate), as shown by equations 5 and 6 respectively. Since P_i and G_i signals are available after one gate delay, there is no need for the carry signal to ripple through all the previous stages. Figure 5 shows a 4 bit CLA with a 4 bit lookahead logic block and appropriate P_i and G_i signals. Therefore, a 4 bit CLA has a maximum gate delay of 6. This can be expressed using equation 7 ,where N is the width of the CLA and r is the width of the lookahead block; while the gate complexity is defined by equation 8

$$Delay_{CLA} = 2 + 4[log_r N], \tag{7}$$

$$Gates_{CLA} = 8N + 1/2[3r + r^2][(N-1)/(r-1)] + 2 \tag{8}$$

B. Reasoning for using Long CLAs

The original Wallace/Dadda multipliers used HAs and FAs for partial product reduction. The HA takes 2 input bits and generates 2 partial products consisting of a sum bit in the same column and a carry out bit in the next column; thus resulting in a 1:1 compression ratio. A FA on the other hand takes in 3 partial products and generates the same 2 partial products as a HA, giving it a 1.5:1 compression ratio. A higher compression ratio means a decrease in the number of generated partial products and the number of required reduction stages, which leads to a more efficient design.

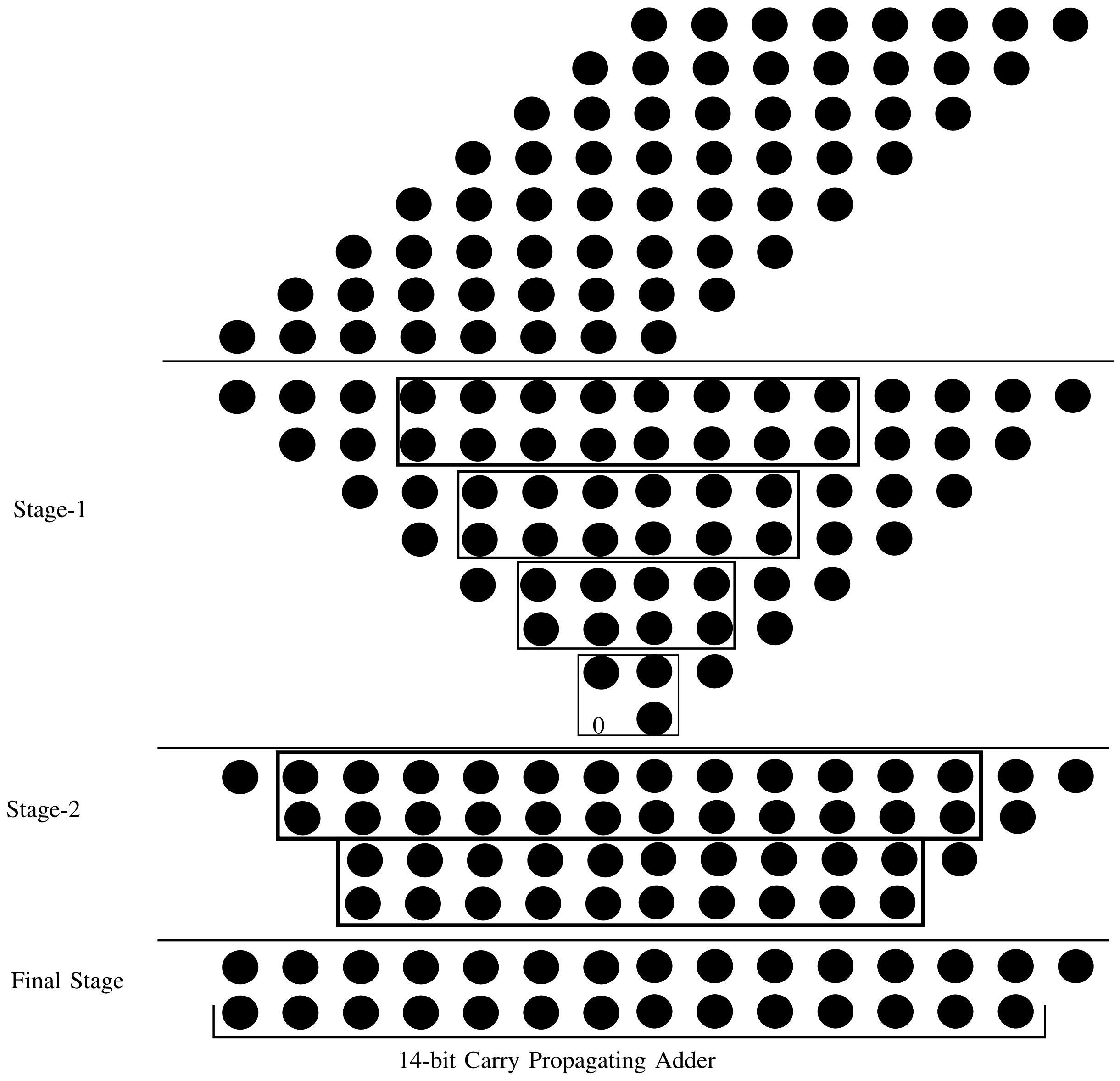

Fig. 6. Dot diagram for an 8x8 long CLA based Multiplier

In contrast a 4 bit CLA takes in 9 partial products (8 input bits and 1 carry in bit) and generates 5 partial products (4 sum bit and 1 carry out bit), giving it a 1.8:1 ratio. Whereas a 16 bit CLA takes 33 partial products (32 input bits and 1 carry in bit) and generates 17 partial products (16 sum bit and 1 carry out bit), giving it an approximate 2:1 compression ratio. Therefore, to achieve a more efficient design, we need to reduce the number of reduction stages, which can be achieved by maximizing the width of the CLA used in each stage.

C. Implementation of an 8x8 Long CLAs based Multiplier

Figure 6 shows an 8x8 multiplier with long CLA based reduction. The partial products are reduced based on a pre-determined matrix height (similar to Dadda multiplier). The matrix height can be calculated using a modified version of equation 2 that can be expressed as:

$$Height_{Stage-LongCLA} = Height_{Stage-1}/2 \tag{9}$$

Following is a walkthrough of the 8x8 multiplier using long CLAs. The goal is to minimize the number of reduction stages by maximizing the width of the CLAs used in each stage.

Stage 0: : The PPG step resulted in a total of 64 partial products, with a stage height of 8.

Stage 1: : The partial products are first rearranged in an inverted pyramid structure. This helps in analyzing the number of required CLAs for each stage, and their corresponding widths. A '0' is added to the Nth column to form symmetry within the partial product matrix, so that longer CLAs can be used.

Our goal is to reduce the height of the current stage to half it's original height. Since the stage height is 8, we need to reduce it to 4. We begin the reduction process from least significant to most significant bit (right to left). The columns that have partial products equal to or less than 4 are moved to the following stage as is. The first column from the right that has more than 4 partial products form the start of a CLA. The sum bit from each CLA can be moved straight to the following stage. Moving further left we continue to add more CLAs to achieve a maximum height of 4 for each column. An important thing to note is that each CLA generates a carry out bit in the column immediately following the last sum bit. Therefore the width of the CLAs must be adjusted to account for the extra bit. In total we used 1 of each CLA8, CLA6, CLA4 and CLA2.

Stage 2: : We are going to follow the same reduction process that we used in stage 1. However the current stage height is 4, and we need to reduce it to 2. We were able to achieve that by using CLA12 and CLA10.

Stage Final: : This is the PPS stage, where 2 rows of partial products are added together using the carry propagating adder.

IV. Results

This section analyzes how the new approach to partial product reduction compares with the original Dadda multiplier and the short CLA approach. Several multipliers are designed for various lengths using FreePDK45, which is an open source and open access based process design kit for the 45nm technology node. The final CPA for all multipliers will be implemented using the same size of CLA, i.e., the only difference in multipliers will be the method of partial product reduction. The multipliers are then compared based on area, power consumption and timing delay.

A. Timing Analysis

The delay of a long CLA based multiplier can be found by following the longest path from input to the output. The multiplication begins by forming partial products, which requires 1 AND gate delay. The next step is the PPR, which requires $log_2(N) - 1$ stages. Each stage has a different time delay due to the difference in size of the CLA adder. Therefore, we need to find an average width of the CLAs being used in the reduction stage. The longest CLA for stage 1 is N bit wide, which keeps increasing for all subsequent stages; while the longest CLA for the final reduction stage is 2N-4 bit wide. Taking an average of both CLAs, results in an average width of $3/2N - 2$. The last step is the summation of final two rows of PPs, which requires a 2N-2 bit wide CPA. Thus the total delay of a NxN multiplier using long CLA approach can be expressed as,

$$Delay_{longCLA} = 1 + [(log_2(N) - 1)(2 + 4log_4(3/2N - 2))] + 2 + 4(log_4(2N - 2)) \tag{10}$$

Table IV shows the timing delay for various CLA based multipliers, computed using Synopsys PrimeTime. The original Dadda multiplier and the long CLA based multiplier have the shortest time delay. This shows that the increased reduction in number of stages can in fact overcome the delay of a long CLA. The short CLA based multiplier has the worst delay of all CLA based multipliers, which means the delay of a short CLA (4 bit) in reality is not equal to that of a FA. The reduction in the number of stages also does not overcome this added wait time for the short CLA, which is why the short CLA multiplier has the longest time delay.

Multiplier Size	Dadda	Short CLA	Long CLA
4x4	0.6 (100.0%)	0.6 (100.0%)	0.6 (100.0%)
8x8	1.1 (100.0%)	1.2 (109.1%)	1.1 (100.0%)
16x16	2.1 (100.0%)	2.3 (109.5%)	2.2 (104.8%)
32x32	4.2 (100.0%)	4.3 (102.4%)	4.2 (100.0%)
64x64	8.3 (100.0%)	8.5 (102.4%)	8.4 (101.2%)
128x128	16.6 (100.0%)	16.8 (101.2%)	16.6 (100.0%)
256x256	33.1 (100.0%)	33.2 (100.3%)	33.2 (100.3%)

TABLE IV
TIMING DELAY (ns)

B. Area and Power Analysis

The complexity of a long CLA based multiplier can be found by evaluating the complexity of each multiplication step. The PPF requires N^2 AND gates, PPR requires $N - 2$ CLAs of an average length of $N - 1$, and the PPS requires a CLA of length $2N - 2$. The total number of gates needed for a long CLA based multiplier can be represented as,

$$Gates_{longCLA} = 1 + (N-2)(8(N-1) + \frac{1}{2}(3r + r^2)\frac{((N-1)-1)}{(r-1)} + 2) + (8(2N-2) + \frac{1}{2}[3r + r^2]\frac{((2N-2)-1)}{(r-1)} + 2) \tag{11}$$

Table V and Table VI shows the area and power consumption of the original Dadda, short CLA and long CLA based multipliers of various lengths.

Multiplier Size	Dadda	Short CLA	Long CLA
4x4	205 (100.0%)	221 (107.8%)	221 (107.8%)
8x8	1032 (100.0%)	1037 (100.5%)	1037 (100.5%)
16x16	4580 (100.0%)	4472 (97.6%)	4472 (97.6%)
32x32	19245 (100.0%)	19900 (103.4%)	18551 (96.4%)
64x64	78850 (100.0%)	89115 (113.0%)	75543 (95.8%)
128x128	319162 (100.0%)	374490 (117.3%)	304862 (95.5%)
256x256	1284192 (100.0%)	1526747 (118.9%)	1224840 (95.4%)

TABLE V
AREA CONSUMPTION (um^2)

The main difference between the use of FA versus short CLA versus long CLA is the number of carry bits that gets generated during the PPR step. A FA generates a carry bit for every 3 PPs, short CLA (4 bits) generates carry for every 9 PPs, while the long CLA (example 16 bits) generates a carry bit for every 33 PPs. Therefore, the use of CLA over FA reduces the total number of PPs that needs to be

Multiplier Size	Dadda	Short CLA	Long CLA
4x4	0.08 (100.00%)	0.08 (100.00%)	0.08 (100.00%)
8x8	0.55 (100.00%)	0.53 (96.36%)	0.51 (92.73%)
16x16	3.17 (100.00%)	2.98 (94.01%)	2.79 (88.01%)
32x32	15.72 (100.00%)	15.43 (98.16%)	13.14 (83.59%)
64x64	70.72 (100.00%)	77.25 (109.23%)	57.14 (80.80%)
128x128	295.41 (100.00%)	339.46 (114.91%)	237.97 (80.56%)
256x256	1200 (100.00%)	1425.1 (118.76%)	971.7 (80.98%)

TABLE VI
POWER CONSUMPTION (mW)

reduced during the PPR step. This results in fewer components being used, which leads to a reduction in area and power consumption. Since the long CLA based multiplier generates the least number of carry bits, it has the lowest area and power consumption. The short CLA has the highest power consumption because the decrease in the number of partial products is still unable to compensate for the increase in complexity of the CLA versus a FA.

V. CONCLUSIONS

The primary objective of this paper was to present a new approach to partial product reduction in multipliers, that provides a performance improvement for portable and low powered applications. Several multipliers using various adder topologies have been designed and implemented in 45nm CMOS technology using the proposed reduction approach. Design and simulation have demonstrated that long CLA based multipliers are competitive with other more commonly used multipliers. The logarithmic nature of the partial product reduction step makes it very difficult to achieve any significant improvements in time delay. However, modest improvements in area ($\approx 5\%$) and power ($\approx 20\%$) over Dadda multiplier have been demonstrated using this new approach.

REFERENCES

[1] Gerald B Rosenberger. Simultaneous carry adder, December 27 1960. US Patent 2,966,305.

[2] Christopher S Wallace. A suggestion for a fast multiplier.IEEE Transactions on electronic Computers, 13(1):14–17, 1964

[3] Luigi Dadda.Some schemes for parallel multipliers.Alta frequenza, 34:349–356, 1965.

[4] Whitney J Townsend, Earl E Swartzlander Jr, and Jacob A Abraham. A comparison of dadda and wallace multiplier delays. In Advanced signal processing algorithms, architectures, and implementations XIII, volume5205, pages 552–560. International Society for Optics and Photonics,2003.

[5] Wesley Chu, Ali I Unwala, Pohan Wu, and Earl E Swartzlander. Implementation of a high speed multiplier using carry lookahead adders. In2013 Asilomar Conference on Signals, Systems and Computers, pages400–404. IEEE, 2013.

[6] Sharma, Abhay. FPGA Implementation of a High Speed Multiplier Employing Carry Lookahead Adders in Reduction Phase.” International Journal of Computer Applications 116.17 (2015).

[7] Bickerstaff, KAndrea C., Michael Schulte, and Earl E. Swartzlander. ”Reduced area multipliers.” Proceedings of International Conference on Application Specific Array Processors (ASAP’93). IEEE, 1993.