International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

IJIEMR Transactions, online available on 8 June 2017. Link :
http://www.ijiemr.org/downloads.php?vol=Volume-6&issue=ISSUE-4

Title: A Design Technique For Faster Dadda Multiplier.

Volume 06, Issue 04, Pages: 719 – 728.

Paper Authors

*** P. SOUJANYA, M. KOTESWARA RAO, D.NAGA RAVIKIRAN (Ph.D.)**
*DEPT OF ECE , CHALAPATHI INSTITUTE OF TECHNOLOGY GUNTUR.
.

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

A DESIGN TECHNIQUE FOR FASTER DADDA MULTIPLIER

P. SOUJANYA
PG SCHOLAR
Dept. of E.C.E
Chalapathi institute of technology
GUNTUR

M. KOTESWARA RAO
Assistant professor
Dept. of E.C.E
Chalapathi institute of technology
GUNTUR

D.NAGA RAVIKIRAN (Ph.D.)
Associate professor, H.O.D
Dept. of E.C.E
Chalapathi institute of technology
GUNTUR

ABSTRACT: In this paper, faster column compression multiplication has been achieved by using a combination of two design techniques: partition of the partial products into two parts for independent parallel column compression and acceleration of the final addition using a hybrid adder proposed in this paper. Based on the proposed techniques 8, 16, 32 and 64-bit Dadda multipliers are developed and compared with the regular Dadda multiplier.

The performance of the Dadda multiplier is analyzed by evaluating the delay, area and power.The resultanalysis shows that the 64-bit regular Dadda multiplier is asmuch as 41.1% slower than the proposed multiplier andrequires only 1.4% and 3.7% less area and power respectively.Also the power-delay product of the proposed design issignificantly lower than that of the regular Dadda multiplier.

Index Terms- Column compression, Dadda multiplier, Faster,Hybrid final adder.

I. INTRODUCTION

High speed multiplication is a primary requirement ofhigh performance digital systems. In recent trends thecolumn compression multipliers are popular for high speedcomputations due to their higher speeds [1-2]. The firstcolumn compression multiplier was introduced by Wallacein 1964 [3]. He reduced the partial product of *N* rows bygrouping into sets of three row set and two row set using(3,2) counters and (2,2) counters respectively. In 1965,Dadda altered the approach of Wallace by starting with theexact placement of the (3,2) counters and (2,2) counters inthe maximum critical path delay of the multiplier [4]. Since2000's, a closer reconsideration of Wallace and Daddamultipliers has been done and proved that the Daddamultiplier is slightly faster than the Wallace multiplier andthe hardware required for Dadda multiplier is lesser than theWallace multiplier [5-6]. Since the Dadda multiplier has afaster performance, we implement the proposed techniquesin the same and the improved performance is compared withthe regular Dadda multiplier.The column compression multipliers have total delays thatare proportional to the logarithm of the

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

operand word lengthswhich is unlike the array multipliers which have speedsproportional to the word length [7-8]. The total delay of themultiplier can be split up into three parts: due to the PartialProduct Generation (PPG), the Partial Product SummationTree (PPST), and finally due to the Final Adder [9]. Of thesethe dominant components of the multiplier delay are due tothe PPST and the final adder.

The relative delay due to thePPG is small. Therefore significant improvement in thespeed of the multiplier can be achieved by reducing the delayin the PPST and the final adder stage of the multiplier. In thiswork the delay introduced by the PPST is reduced by usingtwo independent structures in the partial products. proposed hybrid final adder computes the final productsmuch faster.

This paper is structured as follows: Sections II and IIIdescribe the design of parallel structures for the PPST andthe design of hybrid final adder structure respectively.Section IV reports the ASIC implementation details and thesimulation results. Finally, Section V summarizes theanalysis. Throughout the paper, it is assumed that the numberof bits in the multiplier and multiplicand are equal.

II. DESIGN OF PARALLEL STRUCTURES

The multiplication process begins with the generation ofall partial products in parallel using an array of AND gates.The next major steps in the design process are partitioning ofthe partial products and their reduction process. Each ofthese steps are elaborated in the following subsections.

A. Partitioning the partial products:

We consider two *n*-bit operands *an-1an-2…a2a1a0* and *bn-1bn-2…b2b1b0* for *n* by *n* Baugh-Wooley multiplier, the partialproducts of two *n*-bit numbers are *aibj* where *i,j* go from0,1,..*n*-1. The partial products form a matrix of *n* rows and2*n*-1 columns as show in Fig. 1(a). To each partial productwe assign a number as shown in Fig. 1 (a), e.g. *a0b0* is givenan index 0, *a1b0* the index 1 and so on. For convenience werearrange the partial products as shown in Fig 1(b). Thelongest column in the middle of the partial productscontributes to the maximum delay in the PPST.

Therefore in this work we split-up the PPST into two partsas shown in the Fig. 1(c), in which the Part0 and part1consists of *n* columns. We then proceed to sum up eachcolumn of the two parts in parallel. The summationprocedure adopted in this work is described in the nextsection.

B. The Dadda based reduction: Next the partial products of each part are reduced to tworows by the using (3,2) and (2,2) counters based on theregular Dadda reduction algorithm as shown in Fig. 2 andFig. 3. The grouping of 3-bits and 2-bits indicates (3,2) and(2,2) counters respectively and the different colors classifythe difference between each column, where *s* and *c* denote*partial sum* and *partial carry* respectively. E.g. the bitpositions of *6* and *13* in part0 are added using a (2,2) counterto generate sum *s0* and *c0*. The *c0* is carried to the nextcolumn where it is to be added up with the sum *s1* of a (3,2)

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

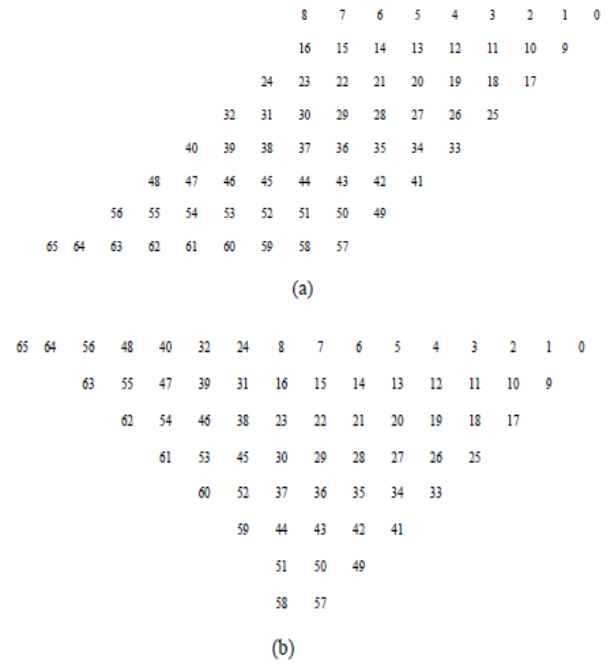


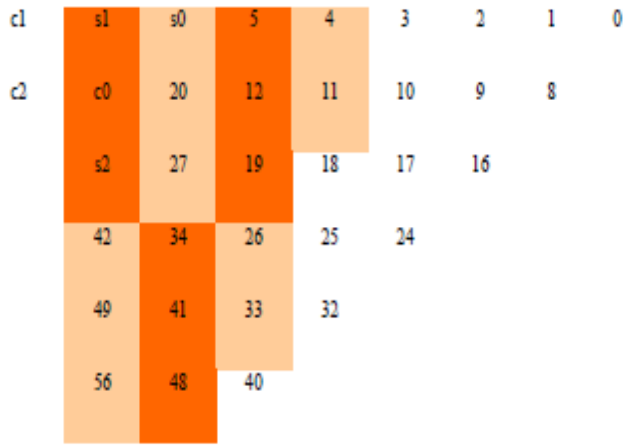
Fig. 1. Partitioning the partial products:
(a) Partial product arraydiagram for 8*8 multiplier,
(b) An Alternative Representation,
(c)Partitioned structure of multiplier showing part0 and part1.

counter adding *7, 14* and *21*. The carry *c1* of (3,2) counter isadded to the next column. The final two rows of each partare summed using a Carry Look-ahead Adder (CLA) to formthe partial final products of a height of one bit column whichindicated at the bottom of Fig. 2 and Fig. 3.

The two parallel structures for Fig. 2 and Fig. 3 based onthe Dadda approach are shown in Fig. 4, where HA, FA, p0,p1 and p denote Half Adder ((2,2)counter)), Full Adder((3,2)counter) , partial final product from part0, partial finalproduct from part1 and final product respectively. Thenumerals residing on the HA and FA indicates the position ofpartial products.

The output of part0 and part1 are computedindependently in parallel and those values are added using ahigh speed hybrid final adder to get the final product.However, before we proceed to carry out the final additionwith the proposed hybrid adder, we first carry out the finaladdition with the CLA for both the unpartitioned Daddamultiplier and the partitioned Dadda multiplier.

This enablesus to evaluate and analyze the effect of partitioning the PPSTinto two parts. The simulation results are listed in Table I andTable II. The comparison between the Table I



International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

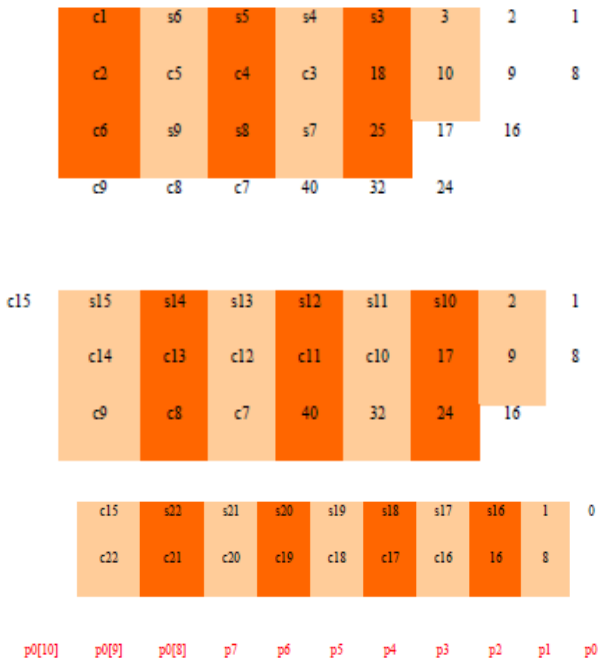


Fig. 2. Reduction of the partial products of part1 based on the Dadda approach.

and Table II gives that the percentage improvement in delay,area and power of the partitioned multipliers with respect tothe regular Dadda multiplier.

It can be seen that for the 8-bit multiplier, there is noimprovement in the speed, area and power. But with theincrease in the word size, the improvement in the speed, areaand power of the partitioned multipliers increases. There is amaximum of 10.5% improvement in delay for the 64-bitmultiplier with only a slight increase in the area and powerof 1% and 1.8% respectively.

Having clearly demonstrated the reduction in the delay ofthe Dadda multipliers due to the partitioning of the partialproducts we now proceed to further enhance the speed of theproposed multiplier.

The further improvement in theperformance can be achieved by replacing the CLA with theproposed hybrid final adder structure which is elaborated inthe next section.

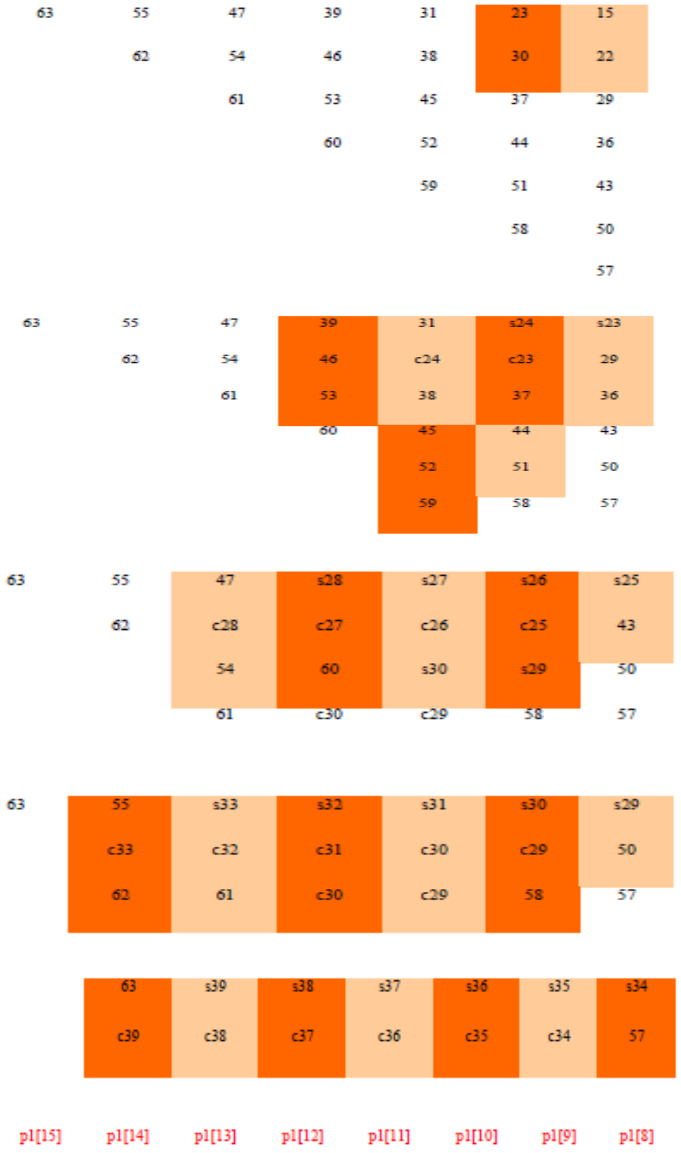


Fig. 3. Reduction of multiplier partial products of part2 based on the Dadda reduction tree.

III. THE HYBRID FINAL ADDER DESIGN

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

In previous works the hybrid final adder designs used toachieve the faster performance in parallel multipliers weremade up of CLA (Carry Lookahead Adder) and CSLA(Carry Select Adder) [9-11]. But due to the structure of theCSLA, it occupies more chip area than other adders.

Thus toachieve the optimal performance, the proposed hybrid adderin this work uses MBEC (Multiplexers with Binary toExcess-1 Converters) and Ripple Carry Adder (CLA) for fastsummation of uneven input arrival time of the signalsoriginating from the PPST. The MBEC adder provides fasterperformance than Carry Save Adder (CSA) and Carry LookAhead (CLA) adder [12]. Also it consumes less area andpower than the Carry Select Adder (CSLA) [13].

A. Hybrid Adder for 8 by 8Multiplier:

Once each part of the partial products has been reduced to a height of one bit column, we get the final partial products asfollows,

p0[10] p0[9] p0[8] p[7] p[6] p[5] p[4] p[3] p[2] p[1]

p1[15] p1[14] p1[13] p1[12] p1[11] p1[10] p1[9] p1[8]

The p0[10:8] are the exceeding carry bits of part0 andp1[15] is the carry bit of part1. The p[7:0] of part0 aredirectly assigned as the final products. To find the remainingp[15:8], we use the CLA and the MBEC shown in Fig. 5.

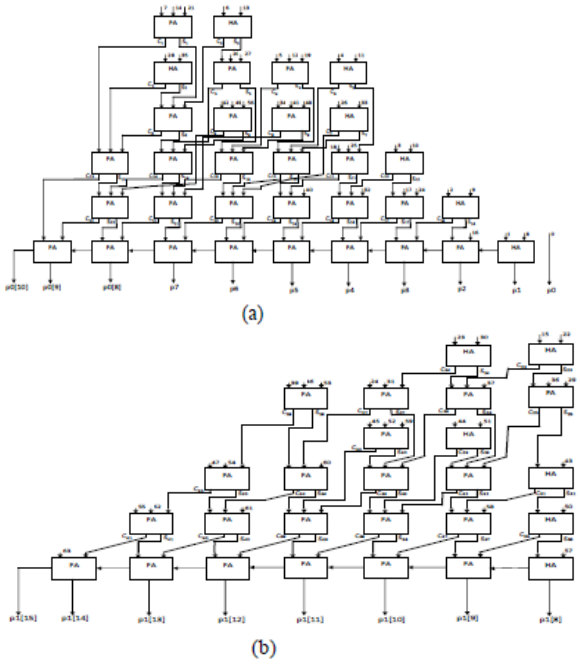


Fig. 4. The Dadda based implementation:
(a) Implementation of part1,
(b) Implementation of part2

TABLE I
REGULAR DADDA MULTIPLIER WITH CLA

Multiplier N by N	Area (μm^2)	Delay (ns)	Power (μW)
8 by 8	8,428	3.40	6.32
16 by 16	29,169	4.71	33.09
32 by 32	105,237	5.92	210.50
64 by 64	397,146	7.54	925.92

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

TABLE II
PARTITIONED DADDA MULTIPLIER WITH CLA

Multiplier *N* by *N*	Area ($\mu m2$)	Delay (*ns*)	Power (μW)
8 by 8	8,957	3.51	6.85
16 by 16	30,241	4.61	35.22
32 by 32	107,362	5.47	218.76
64 by 64	386,629	6.94	952.59

The p0[10:8] and p1[10:8] are added using 3-bit CLA whichfinds p[10:8]. To obtain the remaining p[15:11], thep1[15:11] are assigned to the input of 5-bit MBEC, whichproduce the two partial results p1[15:11] with *Cin* of '0' andthe 5-bit BEC output with the *Cin* of '1'. Depending on the*Cout* of CLA(c[10]), the mux provides the final p[15:11]without having to ripple the carry through p1[15:11].

The 8-bit multiplier uses a single 5-bit MBEC in the finaladder. But the large bit sized multipliers requires multipleMBEC and each of them requires the selection input fromthe carry output of the preceding MBEC. Therefore togenerate the carry output from the MBEC, an additionalblock is developed which is called MBECWC (MBEC WithCarry).

The detailed structures of the 5-bit BEC withoutcarry (BEC) and with carry (BECWC) are shown Fig. 6(a)and Fig. 6(b). The BEC gets *n* inputs and generates *n* output;the BECWC gets *n* input and generates *n+1* output to givethe carry output as the selection input of the next stage *mux*used in the final adder design of 16, 32 and 64-bitmultipliers. The function table of BEC and BECWC areshown in Table III.

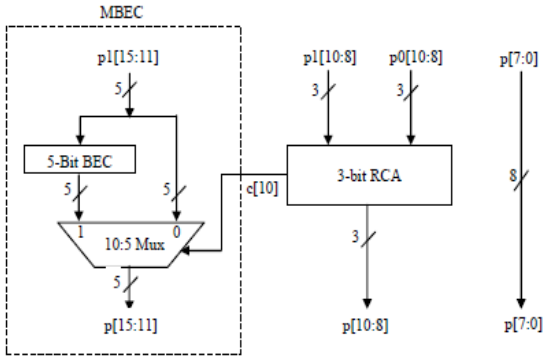


Fig. 5. Hybrid final adder of 8 by 8 multiplier

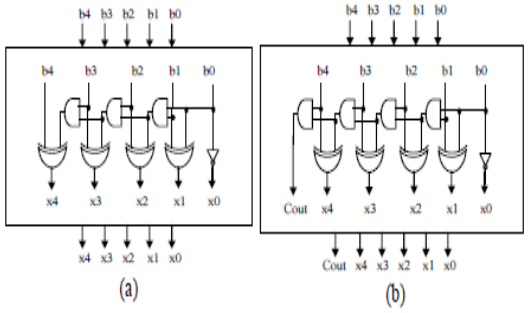


Fig. 6. The 5-bit Binary to Execss-1 Code Converter: (a) BEC (without carry), (b) BECWC (with carry).

B. Variable Block Hybrid Adder

The variable size of adder blocks always leads to fasteradders than fixed size block adder [14]. Thus to furtherimprove the speed of addition, we breakdown the ripple ofgates in the MBEC into multiple size groups of size 2n,where n □ 2. Based on this approach the final adder designfor 16, 32 and 64-bit multipliers are shown in Fig. 7.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

InMBECWC, the mux is getting *n*-bits of data input "as it is"input for selection input '0' and *n*+1-bits of data input fromthe BECWC output for selection input '1'.

Thus to makeequal the size of the inputs to the mux, the one bit '0' isappended as the MSB (Most Significant Bit) to the *n*-bits ofinput. E.g. In Fig. 7(a), the 10:5 mux of MBECWC gets thetwo inputs: 4-bits (*n*-bits) of p[23:20] for selection input '0'and 5-bits (*n*+1-bits) from the 4-bit BECWC for selectioninput '1' respectively. Thus to make equal the size of theinputs, the one bit '0' is appended as the MSB to the inputof p[23:20] is like {0,p[23:20]}

.

TABLE III
FUNCTION TABLE OF 5-BIT BEC & BECWC

Input	BEC without carry	BEC with carry	
b[4:0]	x[4:0]	cy	x[4:0]
00000	00001	0	00001
00001	00010	0	00010
00010	00011	0	00011
00011	00100	0	00100
00100	00101	0	00101
⋮	⋮	⋮	⋮
11011	11100	0	11100
11100	11101	0	11101
11101	11110	0	11110
11110	11111	0	11111
11111	00000	1	00000

To analyze independently the effect of the proposedhybrid adder, the partitioned multiplier with CLA final adderis compared with the partitioned multiplier along with theproposed hybrid adder. The simulation results are listed inTable IV and Table V. The comparison between the TableIV and Table V gives that the percentage improvement in thedelay, area and power of the proposed multiplier (partitionedmultiplier with hybrid final adder) with respect to thepartitioned multiplier with CLA final adder.

The plot clearly shows that the performance improvementin delay increases with the word size of the multiplier. Thespeed of the 8, 16, 32 and 64-bit multipliers are improved14.9%, 21.1%, 25.2% and 27.7% respectively. The area andpower overhead for all word sizes is only slightly higher.

IV. ASIC IMPLEMENTATION AND SIMULATION RESULTS

The ASIC implementation of proposed design follows thecadence design flow.

The design has been developed usingVerilog-HDL and synthesized in Encounter RTL compilerusing typical libraries of TSMC 180nm technology. TheCadence SoC Encounter is adopted for Placement & Routing(P&R) [15]. Parasitic extraction is performed using

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

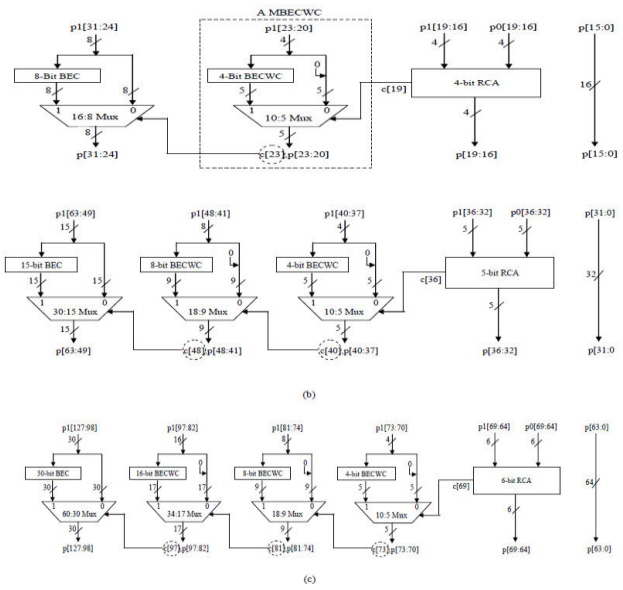


Fig. 7. Variable block hybrid final adder: (a) For 16-bit multiplier, (b) For 32-multiplier, (c) For 64-bit multiplier.

TABLE IV
PARTITIONED DADDA MULTIPLIER WITH CLA

Multiplier *N* by *N*	Area (μm^2)	Delay (ns)	Power (μW)
8 by 8	8,957	3.51	6.85
16 by 16	30,241	4.61	35.22
32 by 32	107,362	5.47	218.76
64 by 64	386,629	6.94	952.59

TABLE V
PARTITIONED DADDA MULTIPLIER WITH HYBRID ADDER

Multiplier *N* by *N*	Area ($\mu m2$)	Delay (ns)	Power (μW)
8 by 8	9,144	3.38	7.07
16 by 16	30,577	4.13	35.99
32 by 32	107,491	4.71	221.01
64 by 64	381,776	5.51	966.45

TABLE VI
PERFORMANCE OF THE REGULAR WITH REFERENCE TO THE PROPOSED DADDA MULTIPLIER

Multiplier *N* by *N*	Area %	Delay %	Power %
8 by 8	-8.5	+ 0.5	-11.8
16 by 16	-4.8	+ 12.21	-8.76
32 by 32	-2.1	+ 20.40	-4.99
64 by 64	3.8	+ 26.91	-2.21

Encounter Native RC extraction tool. The extracted parasiticRC (SPEF format) is back annotated to Common TimingEngine in Encounter Platform for static timing analysis. Foreach word size of the multiplier, the same VCD (ValueChanged Dump) file is generated for possibleinput conditions and imported the same to CadenceEncounter. Power Analysis to perform the powersimulations. The similar design flow is followed for both thedesigns in this work.

V. RESULT SUMMARY

The comparison between the Table I (regular Daddamultiplier with CLA) and Table V (partitioned multiplierwith hybrid adder) summarizes the enhanced performance ofthe proposed multiplier in

terms of percentages which arelisted in Table VI. It exhibits that the area of the regularDadda multiplier is only slightly lesser, ranging from 7.7%to 1.4% for the 8, 16, 32 and 64-bits respectively, than thearea of the proposed multiplier. It is clear that the areaoverhead of the proposed multiplier continuously decreaseswith increasing word size and is only 1.4% for the 64-bitmultiplier.

The power consumption of the regular Dadda multiplier is5.2% less than the proposed multiplier for the 8-bit wordsize. With increasing word size the difference in powerrequirement of the proposed and the Dadda multiplierdecreases. Thus the 64-bit Dadda multiplier requires only3.7% less power than the proposed multiplier.

The delay values clearly indicate that the proposedmultiplier is always faster than the regular Dadda multiplier,also with increasing word size the percentage reduction ofthe delay increases. The speed enhancement is significantfor the 64-bit where the regular Dadda requires 41.1% moretime than the proposed multiplier.

VI. CONCLUSION

The faster multiplication byusing a combination of two design techniques; partitioningof the partial products into two parts to perform independentparallel column compression and fast final addition usinghybrid final adder structure has successfully achieved. The result analysis shows thatthe power and area overheads are not significant. But thespeed and power-delay product improvements aresignificant compared to the regular Dadda multipliers. The proposed multiplier design technique can be implementedwith any type of parallel multipliers to achieve fasterperformance.

REFERENCES

[1] B.Parhami, "Computer Arithmetic", Oxford University Press, 2000.

[2] E. E. Swartzlander, Jr. and G. Goto, "Computer arithmetic," *TheComputer Engineering Handbook*, V. G. Oklobdzija, ed., BocaRaton, FL: CRC Press, 2002.

[3] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEETransactions on Electronic Computers*, Vol. EC-13, pp. 14-17,1964.

[4] Luigi Dadda, "Some Schemes for Parallel Multipliers," *AltaFrequenza*, Vol. 34, pp. 349-356, August 1965.

[5] K.C. Bickerstaff, E.E. Swartzlander, M.J. Schulte, Analysis of columncompression multipliers, Proceedings of 15th IEEE Symposium onComputer Arithmeitc,2001.

[6] W. J. Townsend, Earl E. Swartzlander and J.A. Abraham, "Acomparison of Dadda and Wallace multiplier delays", AdvancedSignal Processing Algorithms, Architectures and ImplementationsXIII. Proceedings of the SPIE, vol. 5205, 2003, pages 552-560.

[7] P. R. Cappello and K Steiglitz: A VLSI layout for a pipe-lined Daddamultiplier, ACM Transactions on Computer Systems,pp. 157-174,1983.

[8] Bickerstaff, K.C."Optimization of Column Compression Multipliers"Doctoral Dissertation, Dept. of Electrical and Computer Engineering,University of Texas at Austin, Austin, Texas, 2007.

[9] V. G. Oklobdzija and D.Villeger, "Improving Multiplier Design byUsing

Improved Column Compression Tree and Optimized FinalAdder in CMOS Technology", IEEE transactions on Very LargeScale Integration (VLSI) systems, Vol. 3, no. 2, June 1995.
[10] Paul F.Stelling, "Design strategies for optimal hybrid final adders inparallel multiplier",Journal of VLSI signal processing, vol 14,pp,321-331,1996.

Pamulapati. Soujanya studied B.Tech at tenali engineering college and at present she is pursuing M.Tech at chalapathi engineering college. Her area of interest is V.L.S.I Design.

Project Guide: M. Koteswararao Assistant professor,ECE Department

HOD: D.NAGA RAVIKIRAN,M.TECH,(Ph.D), Associate professor, ECE Department