FPGA Implementation of Low Power and High Speed Mixed Multiplier

K. Sahithiya[1] A. mohanapriya[2] S. Kannan[3] K. rajesh[4]
[1, 2, 3]UG Scholar [4]Assistant Professor
[1, 2, 3, 4]Electronics and Communication Engineering Department
[1, 2, 3, 4]Knowledge Institute Of Technology, Salem

Abstract— In this work, a new architecture was proposed to reduce the power dissipation of Multipliers. Low power digital Multiplier is designed based on array multiplier technique mainly used to reduce the switching power dissipation, while this new architecture offers great dynamic power savings mainly in array multipliers, due to their regular interconnection scheme, it misses the reduced area and high speed advantages of tree multipliers. Therefore, mixed style architecture, using a traditional tree based technique, combined with a booth multiplier, is proposed. Implementation of all these multiplier Architectures has been carried out on Spartan3E FPGA. By evaluating the performance of these Multiplier architectures using Xilinx ISE tool suite, it has been found that while the booth multiplier technique offers the minimum dynamic power consumption, the mixed architecture reduces the partial product and high speed operations can be performed when compared to all other architectures.

Keywords: FPGA (Field Programmable Gate Array), Booth multiplier, CSA (Carry save adder), DSP (Digital Signal Processing) and Wallace multiplier

I. INTRODUCTION

Low power issues have become an important factor in modern VLSI design. The limited power capacity systems had given rise to more power aware designs by designers. Now-a-days, power has become a crucial factor than area or speed. However, different implementation technologies present different power optimization opportunities. In technologies above 0.1m, dynamic power is the dominant contribution to the total power dissipated while in smaller technologies leakage power is gaining more importance. Dynamic power dissipation is the result of charging the load capacitances in a circuit [1]. It is given by equation where the output capacitance, the supply voltage is, E (sw) (called switching activity) is the average number of transitions and the clock frequency. Leakage power dissipation is divided in two major parts, the sub-threshold leakage and the gate oxide leakage. The sub-threshold leakage is caused by short channel effects and low threshold voltage (Vth), while the gate-oxide leakage is exponentially increasing with decreasing oxide thickness been applied in order to minimize dynamic power dissipation in arithmetic circuits, and especially in digital multipliers. Multipliers are very important devices in DSP applications, like for example FIR filters, leading to excessive power consumption. Consequently, the design of low power multipliers is a necessity for the design and implementation of efficient power-aware devices. There are two main choices for the design of a multiplier. The first is the Wallace tree form, which hast the advantage of a logarithmic circuit delay, and the second is the array multiplier form, like the Carry-Save array, where the delay is linear. On the other hand the array multiplier has a regular structure, which leads to a dense layout making it ideal for fabrication. The Wallace tree has irregular structure, occupies more area on the wafer, and needs greater wiring for cell interconnections [2]. This point is crucial for present and emerging IC technologies, where interconnection plays a predominant role. This factor makes the Wallace tree inappropriate for low power applications. In this paper, we present a technique to minimize power dissipation in digital multipliers, starting from dynamic power and concentrating on the switching activity. There have been proposed a lot of techniques to reduce the switching activity of a logic circuit. In the sequential world one of the most successful is clock gating, where an enable signal inhibits the clock, isolating a large block of the circuit. This technique eliminates activity and thus, power consumption. For example, in, clock gating is applied in pipelined array multipliers with significant power gains.

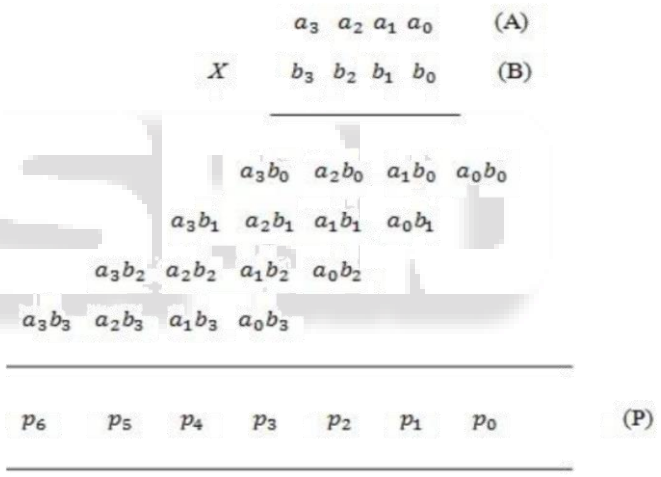

Fig. 1: Basic multiplier

Fig 1 shows the multiplication of two 4-bit numbers. The numbers are denoted by A and B where a0, a1, a2, represents the bits of multiplicand A with as its least significant bit and as its most significant bit and b0, b1, b2, represents the bits of multiplier B with as its least significant bit and as its most significant bit. The product of the two 4-bit numbers is denoted by P which is of 8-bit with as the least significant bit and as the most significant bit. Fig 1 shows the basic multiplication of two numbers and thus producing the result, P.

In pure combinational circuits the main effort of research has been to rearrange the tree of gates in a more profitable scheme, or to insert clock gated registers which provide the desired isolation. The contribution of this paper is the evaluation of a technique similar to clock gating. To isolate parts of a circuit, transmission gates are used as low delay and area overhead bypass switches. An array multiplier with bypass transmission gates can offer power reductions of more than 50%. Similar bypass ideas with different isolating components have also been reported in

the past, in. However, our contribution moves one step further and proposes a mixed architecture, to address both dynamic power dissipation and performance, by doing half of the multiplication through an array structure, with bypass transmission gates, and the other half through a Wallace tree. This mixed architecture offers a delay*power product improvement ranging from 1.2x to 6.5x, compared to all other architectures. Minimization of the switching activity in a digital circuit can be performed by isolating and blocking units producing non-valuable partial results, in order to save power [3]. To perform isolation, transmission gates can be used, as ideal switches with small power consumption, propagation delay similar to the inverter, and small area. Isolation by transmission gates can be applied to any kind of logic circuit. However in this paper we are considering digital multipliers, starting with the canonical and widely used Carry-Save array topology.

II. BASIC ARCHITECTURES

A. *Carry Save Array Multiplier*

The Carry-Save array multiplier is a straight forward implementation of vector multiplication. It is preferred in some implementations because of its canonical interconnect topology [4]. It consists of a partial product reduction tree, which is used to calculate partial products in Carry-Save redundant form, and a final chain adder to transform the redundant form in normal binary form.

The functionality of the Carry-Save array multiplier is as follows: We assume that and are unsigned numbers. The bits are fed into an array of FA* cells always 0. So, no transmission gate is used to block it. Fig 2 shows the basic carry save multiplier.

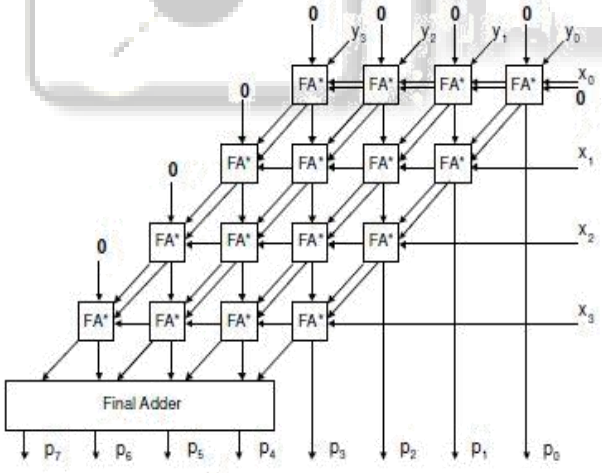

Fig. 2: Carry save multiplier

If, to fix any erroneous carry generated from previous computations, an AND gate is used before the final adder to make the final diagonal carry output 0.The whole structure of the modified multiplier is presented in figure 1.

B. *Wallace Tree Multiplier*

Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers [5]. Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers as shown in fig 3.

The Wallace tree has three steps:

1. Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding results.
2. Reduce the number of partial products to two by layers of full and half adders.
3. Group the wires in two numbers, and add them with a conventional adder.

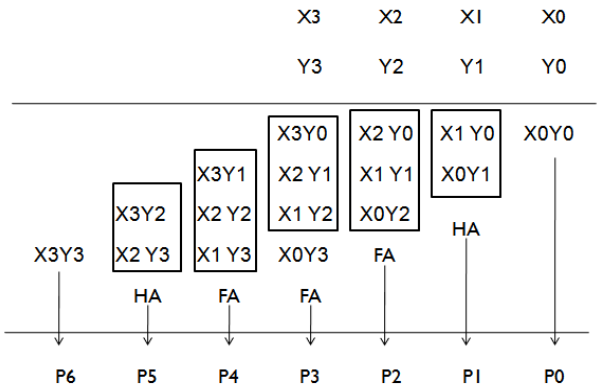

Fig. 3: Wallace Multiplier

The benefit of the Wallace tree is that there are only reduction layers, and each layer has propagation delay. As making the partial products is and the final addition is, the multiplication is only, not much slower than addition (however, much more expensive in the gate count). Naively adding partial products with regular adders would require time.

C. *Modified booth multiplier*

Booth encoding is a method of reducing the number of partial products required to produce the multiplication result. To achieve high-speed multiplication, algorithms using parallel counters like modified Booth algorithm has been proposed and used [6&7]. This type of fast multiplier operates much faster than an array multiplier for longer operands because it's time to compute is proportional to the logarithm of the word length of operands. By recoding the numbers that are to be multiplied, Modified Booth multiplier allows for smaller, faster multiplication circuits. The number of partial products is reduced to half, by using the technique of Booth recoding [8]. Reduction in the number of partial products depends upon how many bits are recoded and on the grouping of bits.

The following fig 4 represents the operation of booth multiplier. By using modified booth multiplier by radix algorithm we can reduce the number of partial products.

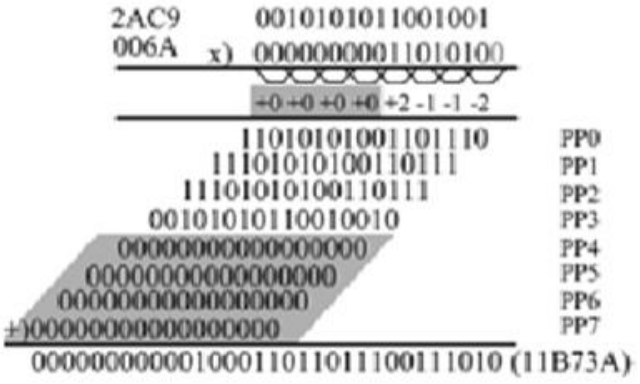

Fig. 4: Architecture of Booth multiplier

D. Existing modified multiplier

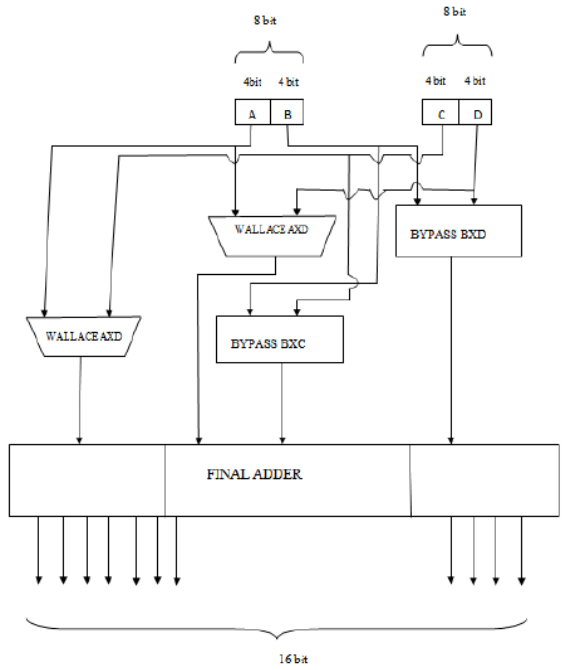

Fig. 5: Architecture of other existing modified multiplier

Fig 5 shows the other existing mixed architecture which is the combination of Wallace and bypass tree techniques. In general, array multipliers offers dynamic power saving but it fails to give a reduced area and fast speed advantages because of their regular interconnection pattern. So, tree multipliers are introduced to achieve reduced area and fast speed, to achieve both delay and power advantages it is better to use a different architecture that is having two parts one is tree based part and other is bypass architecture. The modified bypass technique offers minimum dynamic power compared to normal carry save array multiplier. The tree based part of the mixed architecture has enough timing slack to be implemented using high threshold voltage components. Two 8 bit values can be multiplied by splitting each one of them in two 4 bit parts.

III. PROPOSED ARCHITECTURE

The mixed architecture is nothing but the combination of Wallace tree and booth multiplier as shown in the fig 6. The great timing advantage of the Wallace tree along with the great power advantage of the booth multiplier can be combined in a mixed architecture. In general, array multipliers offers dynamic power saving but it fails to give a reduced area and fast speed advantages because of their regular interconnection pattern.

So, tree multipliers are introduced to achieve reduced area and fast speed, to achieve both delay and the power advantages it is better to use a different architecture that is having two parts one is tree based part and other is bypass architecture. The modified bypass technique offers minimum dynamic power compared to normal carry save array multiplier. The tree based part of the mixed architecture has enough timing slack to be implemented using high threshold voltage components.

Two 8 bit values can be multiplied by splitting each one of them in two 4 bit parts. If the first 8 bit value is (A, B) and the second is (C, D), four 8 bit products are generated. These four partial products shifted and added together generate the final 16 bit multiplication. The key point behind operand splitting is to use different multiplier architectures for each partial multiplication. In the example of figure 6, and are performed in Wallace multipliers while the others in booth architecture. So, from and performance is gained while from and power is gained. If half of one or both operands usually contain more 0s than 1s, this specific half should be passed through the bypass multiplier for greater power improvements. For example, the architecture of figure 5 gives better results when contains on average more 0s than 1s.all other multipliers in various aspects.

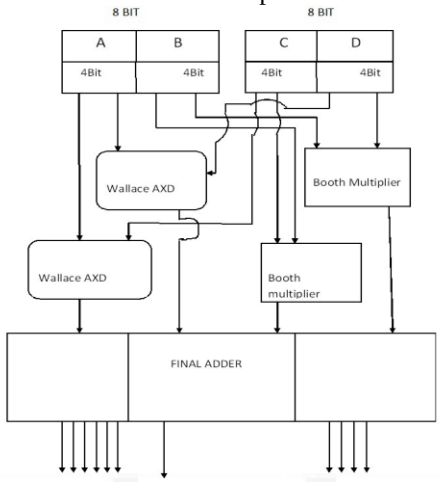

Fig. 6: Proposed Mixed Architecture

IV. RESULTS AND DISCUSSION

Functional Simulations of all the multipliers is carried out using Xilinx Simulator, ISIM Measurement and simulation results. Fig 7 shows the output of the Wallace tree.

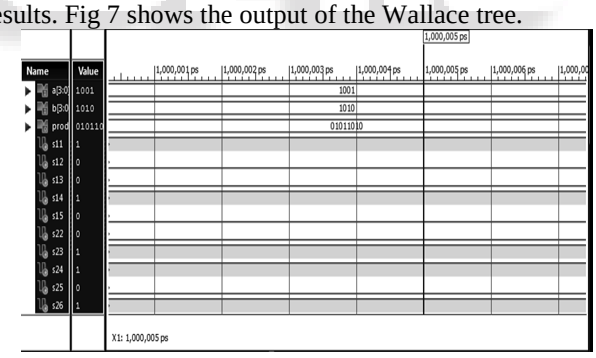

Fig. 7: Wallace output

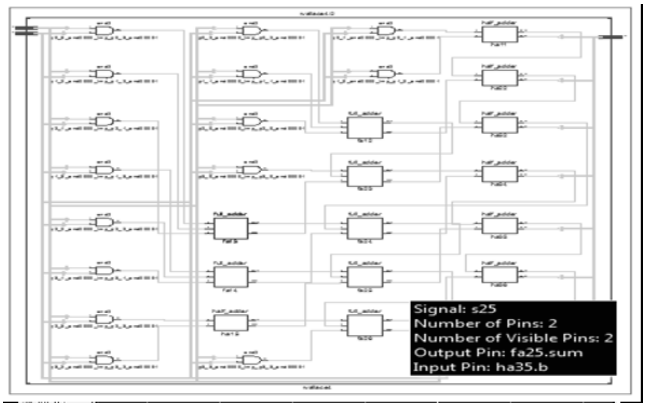

Fig. 8: Wallace RTL schematic

Fig 8 shows the RTL schematic of the Wallace tree.

Fig 9 shows the design summary of the Wallace tree.

wallace4 Project Status			
Project File:	wallace4.xise	**Parser Errors:**	No Errors
Module Name:	wallace4	**Implementation State:**	Programming File Generated
Target Device:	xc3s250e-5pq208	**•Errors:**	No Errors
Product Version:	ISE 12.1	**•Warnings:**	7 Warnings (7 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	33	4,896	1%	
Number of occupied Slices	19	2,448	1%	
Number of Slices containing only related logic	19	19	100%	
Number of Slices containing unrelated logic	0	19	0%	
Total Number of 4 input LUTs	33	4,896	1%	
Number of bonded IOBs	16	158	10%	
Average Fanout of Non-Clock Nets	3.13			

Fig. 9: Wallace Design Summary

Fig 10 shows the delay summary of the Wallace tree.

```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 586 / 8
--------------------------------------------------------
Delay:               12.162ns (Levels of Logic = 9)
  Source:            A<0> (PAD)
  Destination:       prod<6> (PAD)

  Data Path: A<0> to prod<6>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Na
    ----------------------------------------  ----------
     IBUF:I->O             6   1.106    0.721  A_0_IBUF
     LUT2:I0->O            3   0.612    0.603  p2_0_and0
     LUT4:I0->O            3   0.612    0.520  fa12/Mxor_
```

Fig. 10: Wallace Delay Summary

Fig 11 shows the output of the booth multiplier.

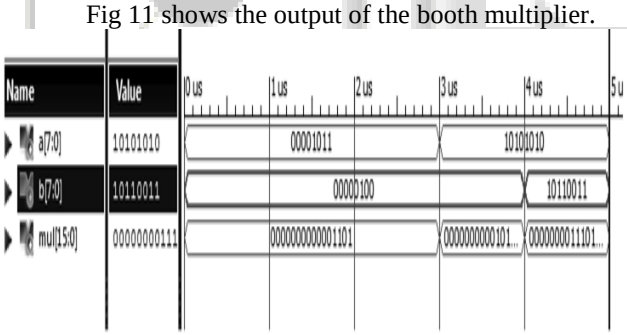

Fig. 11: Booth Multiplier output

Fig 12 shows the RTL schematic of the Wallace tree.

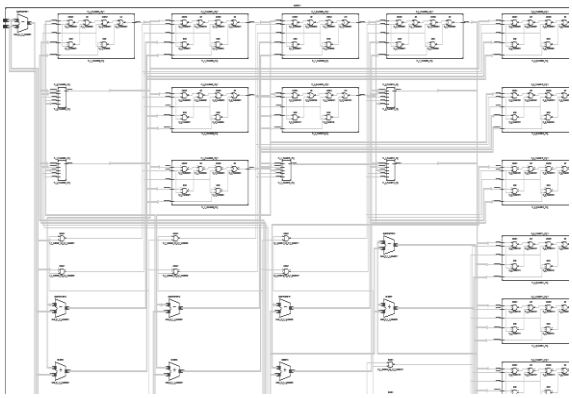

Fig. 12: Booth Multiplier schematic

Fig 13 shows the design summary of the booth multiplier.

booths Project Status (03/29/2014 - 18:37:37)			
Project File:	booths.xise	**Parser Errors:**	No Errors
Module Name:	booths	**Implementation State:**	Programming File Generated
Target Device:	xc3s250e-5pq208	**•Errors:**	No Errors
Product Version:	ISE 12.1	**•Warnings:**	29 Warnings (14 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	105	4,896	2%	
Number of occupied Slices	53	2,448	2%	
Number of Slices containing only related logic	53	53	100%	
Number of Slices containing unrelated logic	0	53	0%	
Total Number of 4 input LUTs	105	4,896	2%	
Number of bonded IOBs	32	158	20%	
Average Fanout of Non-Clock Nets	4.04			

Fig. 13: Booth Multiplier Design Summary

Fig 14 shows the delay summary of the booth multiplier.

```
Delay:               26.946ns (Levels of Logic = 26)
  Source:            a<0> (PAD)
  Destination:       mul<7> (PAD)

  Data Path: a<0> to mul<7>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical I
    ----------------------------------------  ---------
     IBUF:I->O             43   1.106    1.145  a_0_IBUF
     LUT4:I1->O             2   0.612    0.410  Msub_m_7_
     LUT3:I2->O             1   0.612    0.360  Msub_m_7_
     LUT4:I3->O             6   0.612    0.721  m_7_mux0(
     LUT3:I0->O             1   0.612    0.426  Madd_m_7_
     LUT4:I1->O             1   0.612    0.000  m_7_mux0(
```

Fig. 14: Booth Multiplier Delay Summary

Fig 15 shows the simulation output of the mixed architecture multiplier.

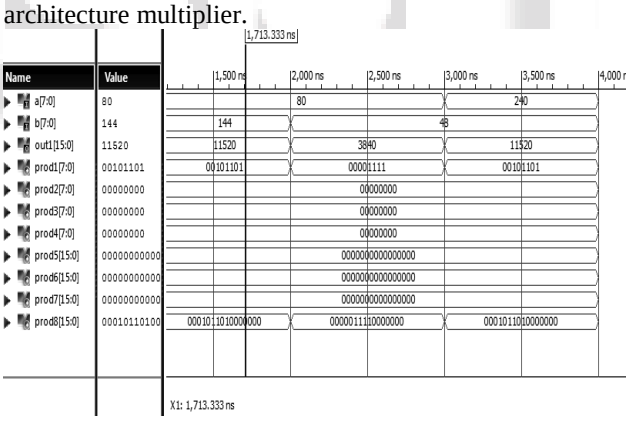

Fig. 15: Mixed Architecture output

Fig 16 shows the RTL schematic of the mixed architecture multiplier.

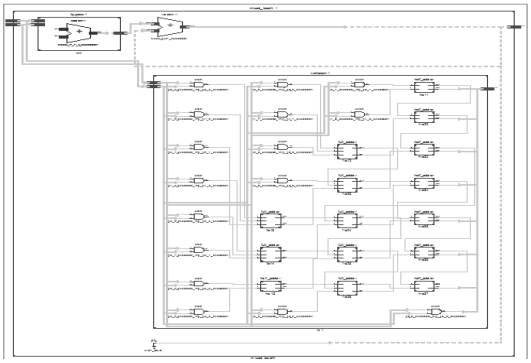

Fig. 16: Mixed Architecture Schematic

Fig 17 shows the design summary of the mixed architecture multiplier.

mixed_booth Project Status			
Project File:	mult4by4bit.xise	**Parser Errors:**	No Errors
Module Name:	mixed_booth	**Implementation State:**	Programming File Generated
Target Device:	xc3s250e-5pq208	**•Errors:**	No Errors
Product Version:	ISE 12.1	**•Warnings:**	31 Warnings (31 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	61	4,896	1%	
Number of occupied Slices	32	2,448	1%	
Number of Slices containing only related logic	32	32	100%	
Number of Slices containing unrelated logic	0	32	0%	
Total Number of 4 input LUTs	62	4,896	1%	
Number used as logic	61			
Number used as a route-thru	1			
Number of bonded IOBs	32	158	20%	
Average Fanout of Non-Clock Nets	3.14			

Fig. 17: Mixed Architecture Design Summary

Fig 18 shows the delay summary of the mixed architecture multiplier.

```
Delay:               14.033ns (Levels of Logic = 11)
  Source:            a<4> (PAD)
  Destination:       out1<15> (PAD)

  Data Path: a<4> to out1<15>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Nam
    ----------------------------------------  -----------
     IBUF:I->O             6   1.106    0.721  a_4_IBUF (a
     LUT2:I0->O            3   0.612    0.603  m1/p2_0_anc
     LUT4:I0->O            3   0.612    0.520  m1/fa12/Mxc
     LUT4:I1->O            3   0.612    0.603  m1/ha32/car
     LUT3:I0->O            3   0.612    0.603  m1/fa24/Mxc
     LUT3:I0->O            3   0.612    0.454  m1/fa25/Mxc
     LUT4:I3->O            1   0.612    0.509  m1/fa26/Mxc
```

Fig. 18: Mixed Architecture Delay summary

The following fig 19 shows various performance metrics of all the Multipliers.

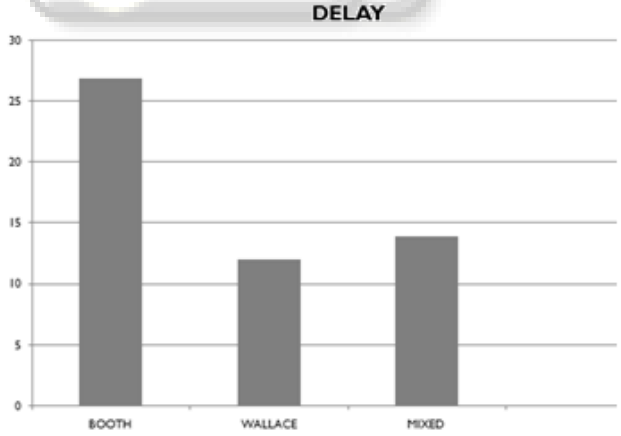

Fig. 19: Comparison between multipliers

V. CONCLUSION

In this work, we have implemented Multipliers for low power Applications. For power optimization, two basic algorithms, namely Wallace Tree and Array Multiplier, are combined to exploit the low power and High Performance gains from the individual architectures. This topology, called as mixed style Multiplier was implemented on Spartan 3E FPGA and it was observed that the power dissipation and the critical path delay are fairly lower than the normal multiplier. Power Measurements were performed using Xilinx "X-Power Analyzer", for a test case like random patterns. It is possible to extend the resolution of the mixed style multiplier, without affecting the performance. Also, this design can be implemented by using standard cell libraries designed for low power, with ASIC Development Kit, for improving the performance relatively.

REFERENCES

[1] Dimitris Bekiaris, George Economakos and Kiamal Pekmestzi, "A Mixed Style Multiplier Architecture for Low Dynamic and Leakage Power Dissipation," in International Symposium on VLSI Design Automation and Test (VLSIDAT).IEEE,2010.

[2] M. Karlsson, "A generalized carry-save adder array for digital signal processing," in 4th Nordic Signal Processing Symposium. IEEE, 2000, pp.287–290.

[3] P. Meier, R. A. Rutenbar, and L. R. Carley, "Exploring multiplier architecture and layout for low power," in Custom Integrated Circuits Conference. IEEE, 1996, pp. 513–516.

[4] N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective - Third Edition. Addison-Wesley, 2004.

[5] C. C. Wang and G. N. Sung, "Low-power multiplier design using a bypassing technique," Journal of Signal Processing Systems, 2008.

[6] S. Kim, S. Hong, M. Papaefthymiou, and W. E.Stark, "Low power parallel multiplier design for DSP applications through coefficient optimization," in 12th Annual International ASIC / SOC Conference. IEEE, 1999, pp. 286–290.

[7] G. Economakos and K. Anagnostopoulos, "Bit level architectural exploration technique for the design of low power multipliers," in International Symposium on Circuits and Systems. IEEE, 2006.

[8] C. N.Marimuthu1, P. Thangaraj2,"Low Power High Performance Multiplier" ICGST-PDCS, Volume 8, Issue 1, December 2008.

