

Architectural Level Power Optimization Techniques for Multipliers

V.Alekhy^{#1}, B.Srinivas^{*2}

Department of ECE, MVGR College Of Engineering
Vizianagaram, AP, India

Abstract— In this work, a new topology was proposed to optimize the power dissipation of Multipliers. Low power digital Multiplier Design based on bypassing technique mainly used to reduce the switching power dissipation. While this technique 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 part, combined with a bypass, array based part, is proposed. Prototyping 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 bypass technique offers the minimum dynamic power consumption, the mixed architecture offers a delay*power product improvement , compared to all other architectures.

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 0.1 μ m, 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. It is given by equation $P_{dynamic} = C_L V_{DD}^2 E(sw) f_{clk}$ where C_L is the output capacitance, V_{DD} the supply voltage, $E(sw)$ (called switching activity) is the average number of transitions and f_{clk} 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 (V_{th}), while the gate-oxide leakage is exponentially increasing with decreasing oxide thickness.

In the past, different architectural optimizations have 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 has 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. 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. 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. 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 straightforward implementation of vector multiplication. It is preferred in some implementations because of its canonical interconnect topology. 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. A 4x4 Carry-Save multiplier is shown in figure 1.

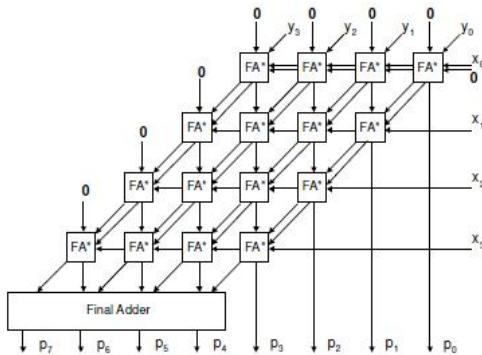


Figure1. A 4x4 Carry-Save multiplier

The functionality of the Carry-Save array multiplier is as follows: We assume that $X = (x_{n-1}, \dots, x_1, x_0)$ and $Y = (y_{m-1}, \dots, y_1, y_0)$ are unsigned numbers. The bits are fed into an array of FA* cells (figure 2(a)).

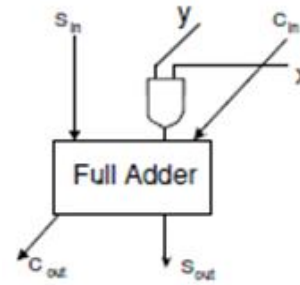


Figure2(a). FA* cell

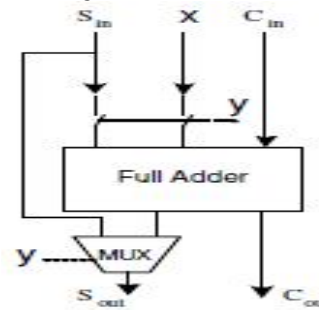


Figure2(b). FAB Cell

Each FA* cell performs the multiplication $x_i \times y_j$ using an AND gate and then adds the result with the incoming carry bits, to produce an output sum and an output carry. All FA* cells are appropriately connected (sums and carries) as shown in figure 1, to perform the multiplication. The final adder shown in figure 1 is used to merge the sums and carries from the last row of the array, since in every row the carry bits are not immediately added but rather propagated to the row below.

We can see that when y_j is 0 then the corresponding diagonal cells are functioning unnecessarily. In all these cells the partial products $x_i \times y_j$ and the carry inputs are zero for $i = 0, 1, \dots, n - 1$ and this chain does not contribute to the formation of the product. Consequently, the sum output of the above cell can bypass this unimportant diagonal with the use of transmission gates. To achieve all of the above we can replace the FA* cell with the cell in figure 2(b) (called the FAB cell).

B. CSA with Bypassing Technique

The transmission gates in the FAB cell lock the inputs of the full adder to prevent any transitions. When $y = 0$, and a multiplexer propagates the sum input to the sum output. When $y = 1$, the sum output of the full adder is passed. The carry input does not generate any new value since the initial diagonal carry input is

always 0. So, no transmission gate is used to block it. If $y = 0$, 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 3.

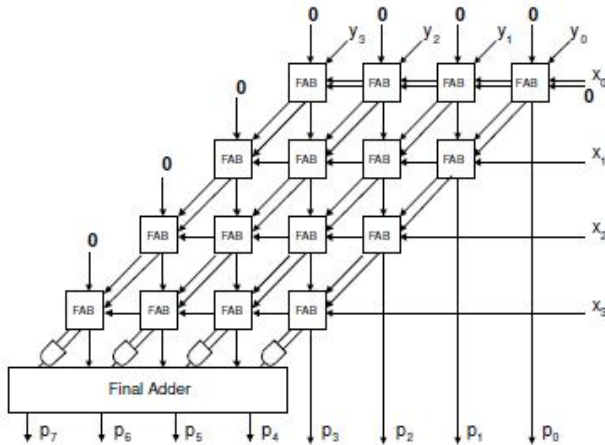


Figure3. 4 x 4 Carry Save Array Multiplier with Bypass

C. Wallace Tree Multiplier

Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers.

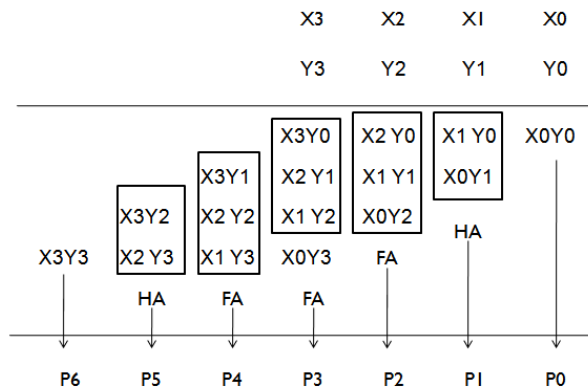


Figure4. Wallace Tree

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 n^2 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.

The benefit of the Wallace tree is that there are only $O(\log n)$ reduction layers, and each layer has $O(1)$ propagation delay. As making the partial products is $O(1)$ and the final addition is $O(\log n)$, the multiplication is only $O(\log n)$, not much slower than addition (however, much more expensive in the gate count). Naively adding partial products with regular adders would require $O(\log^2 n)$ time.

D. Mixed Architectures

The mixed architecture is nothing but the combination of Wallace tree and Array structure. The great timing advantage of the Wallace tree along with the great power advantage of the bypass scheme can be combined in a mixed architecture. The figure for mixed architecture is shown in figure 5.

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.

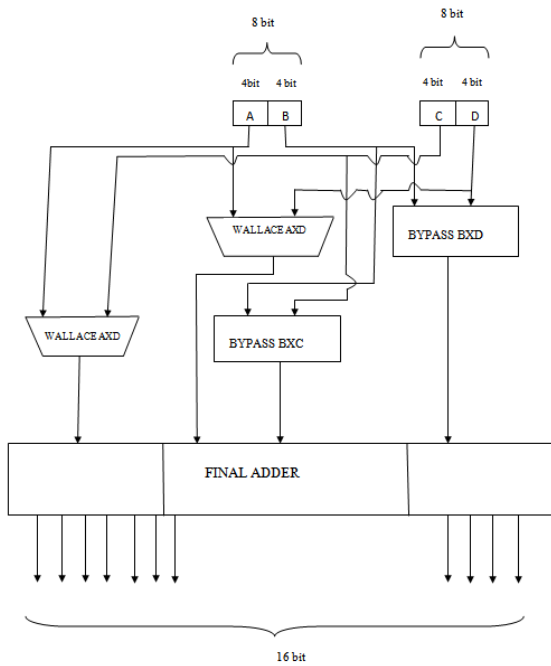


Figure5. A 16 bit multiplication split in parts

If the first 8 bit value is (A,B) and the second is (C,D), four 8 bit products are generated, $X = A \times D$, $Y = B \times D$, $Z = A \times C$ and $W = B \times C$. 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 5, $X = A \times D$ and $Z = A \times C$ are performed in Wallace multipliers while the others in a bypass architecture. So, from $X = A \times D$ and $Z = A \times C$ performance is gained while from $Y = B \times D$ and $W = B \times C$ 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.

III. Performance Evaluation

The Performance Evaluation of Bypass Multipliers Was performed for the resolution of 4x4 using Xilinx tool suite. When compared to other architectures, such as Wallace and normal multipliers, it is power efficient. The test cases that were used for this purpose are generated in a random fashion. By combining 4x4 Wallace tree and bypass multipliers, mixed style multiplier was implemented with a resolution of 8x8. The Performance of this topology was compared with

all other multipliers in various aspects. Mainly from the performance point of view, it was observed they provide low switching power dissipation with a slight increase in the delay.

The following table (Table I) shows various performance metrics of all the Multipliers.

Multiplier Topology	Bits	Switching Power Dissipation (mW)	Critical Path Delay (ns)	Area (in terms of LUTs)	Power Delay Product (picoJoules)
Carry save Array Multiplier	4x4	2.89	14.15	26	40.89
Carry save Array Multiplier	8x8	4.11	22.34	124	91.81
Carry save Array Multiplier	16x16	5.06	45.9	494	232.25
Carry save Array Multiplier	32x32	6.7	87.6	2014	586.92
Bypass	4x4	2.65	12.131	27	32.14
Bypass	8x8	3.02	22	117	66.44
Bypass	16x16	4	42.82	493	171.28
Bypass	32x32	4.88	83.98	2013	409.82
Wallace	4x4	2.67	12.782	31	34.12
Wallace	8x8	3.13	24.68	132	77.24
Wallace	16X16	6.34	46	532	291.64
Wallace	32X32	8.14	88.6	2100	721.2
Mixed Style Multiplier	8x8	3.31	19.8	135	65.53
Mixed Style Multiplier	16x16	3.57	31.8	549	113.52
Mixed Style Multiplier	32X32	5.22	53.42	2170	278.85
Mixed Style Multiplier	64X64	6.58	96.92	8457	637.73

Table I: Performance Metrics

Functional Simulations of all the multipliers is carried out using Xilinx Simulator, Isim.

IV. 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.

V. 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 (VLSI-DAT). IEEE, 2010, pp. 258-261
- 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.