A Comparison of Layout Implementations of Pipelined and Non-Pipelined Signed Radix-4 Array Multiplier and Modified Booth Multiplier Architectures

Leonardo L. de Oliveira
UFSM, Santa Maria, Brazil
leonardo@mail.ufsm.br

Cristiano Santos, Daniel Ferrão
UFRGS, P. Alegre, Brazil
{clsantos,dlferrao}@inf.ufrgs.br

Eduardo Costa
UCPel, Pelotas, Brazil
ecosta@atlas.ucpel.tche.br

José Monteiro
IST/INESC, Lisboa, Portugal
jcm@inesc.pt

João Baptista Martins
UFSM, Santa Maria, Brazil
batista@inf.ufsm.br

Sergio Bampi, Ricardo Reis
UFRGS, UFPel, Brazil
{bampi,reis}@inf.ufrgs.br

Abstract

This paper presents performance comparisons, namely in power consumption, area and delay, between two multipliers architectures. The first architecture consists of a pure array multiplier that was modified to handle the sign bits in 2´s complement and uses a radix-4 encoding to reduce the partial product lines. It is extended for radix 2^m encoding, which leads to a reduction of the number of partial lines, enabling a significant improvement in performance and power consumption. The second architecture implemented was the widely used Modified Booth multiplier. These both architectures were still implemented in a pipelined form: new pipelined array architecture for signed multiplication and a pipelined Booth architecture. We describe a design methodology to physically implement these architectures and obtain area, power consumption and delay results. Up to now only results at the logic level were presented in previous work. The pipelined version of the radix-4 architecture was implemented in order to reduce both the critical path and useless signal transitions that are propagated through the array. The performance of pipelined array architecture is compared with the pipelined Modified Booth. We compare the layout implementations in terms of area, power and delay. The results show that the new pipelined array multiplier can be significantly more efficient, with close to 16% power savings.

1. Introduction

Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Wallace multiplier [15] is among the fastest. However, they do not have such a regular structure as the conventional array [10] or Booth [11] multipliers. Hence, when layout regularity, high-performance and low power are primary concerns, Booth multipliers tend to be the primary choice [2], [6], [8], [11], [13].

In this paper, we present layout implementations for both the Modified Booth multiplier and the new array multiplier in non-pipelined versions. We synthesize the multipliers by using an automatic synthesis tool, named Tropic [12]. In order to compare the Modified Booth and the array architectures, both using radix-4, the ELDO – a spice simulator, part of the Mentor Graphics environment, was used. The results show that the new array multiplier is significantly more efficient, saving more than 40% in power consumption. This result is very close to the results reported in [3], obtained at the logic level using a switch-level simulator and 16% power savings considering pipelined versions.

The power reduction presented by the new array multiplier is mainly due to the lower logic depth, which has a big impact in the amount of glitching in the circuit. We should stress further that, in contrast to the architecture presented in [3], rasing the radix for the Booth architecture is a difficult task, thus not being able to leverage from the potential savings of higher radices.

This paper is organized as follows. The next section briefly describes the radix-4 array multiplier, the Modified Booth multiplier and their pipelined forms. Section 3 describes the design methodology and how area, power and delay results are obtained. Comparisons between the radix-4 array multiplier architecture and the Modified Booth, for both switch level and electrical level are presented in Section 5. Finally, in Section 6 we conclude this paper, discussing the main contributions and future work.

2. Radix-2^m Array Multiplier

In this section, we summarize the methodology of [4] for the generation of regular structures for arithmetic operators using signed radix-2^m representation and extend it into a pipelined version [5].

For the operation of a radix-2^m multiplication, the operands are split into groups of *m* bits. Each of these groups can be seen as representing a digit in a radix-2^m. Hence, the radix-2^m multiplier architecture follows the basic multiplication operation of numbers represented in radix-2^m. The radix-2^m operation in 2´s complement representation is given by Equation 1.

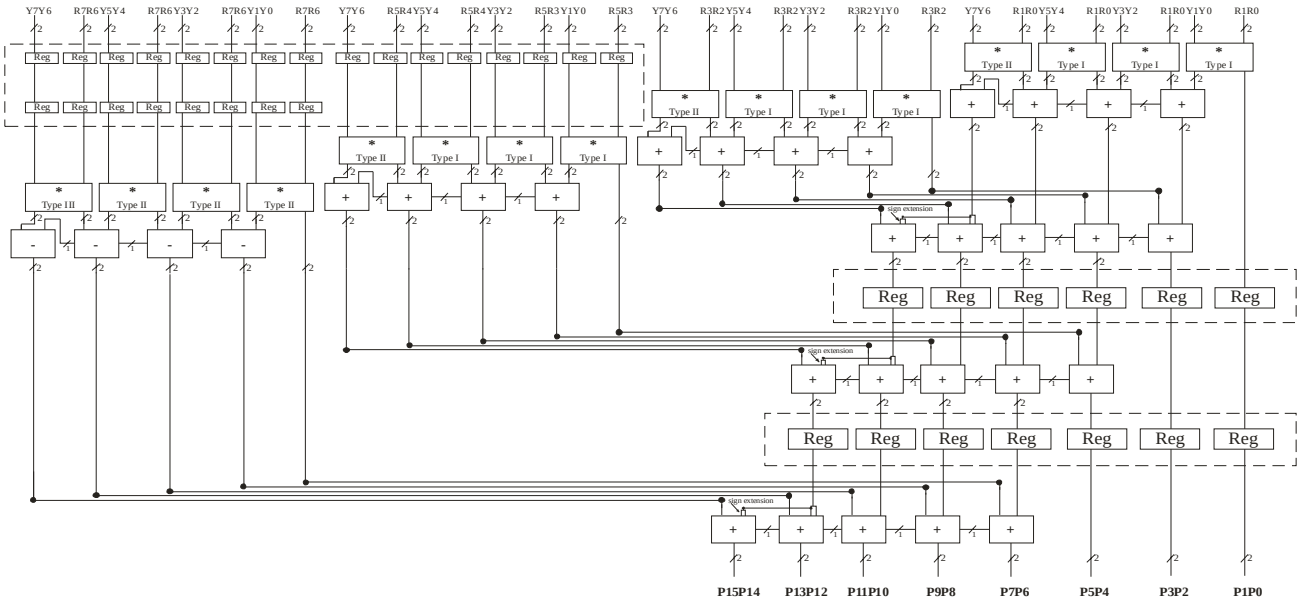

Figure 1 – Example of an 8-bit wide 2's complement radix-4 array multiplier

$$R \times Y = R' \times Y' - R' y_{W-1} y_{\frac{W}{m}-1} 2^{W-m} - \ldots$$

$$\ldots - r_{W-1} r_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} y_j 2^{W-m+j} \quad (1)$$

where R and Y are two operands W-bits wide; r_{W-1} is the most significant bit (is the sign bit); $R' = \sum_{i=0}^{W-2} r_i 2^i$.

This operation is illustrated in **Figure 2**. For the *W-m* least significant bits of the operands unsigned multiplication can be used. The partial product modules at the left and bottom of the array need to be different to handle the sign of the operands. For this architecture, three types of modules are needed. Type I are the unsigned modules. Type II modules handle the *m*-bit partial product of an unsigned value with a 2's complement value. Finally, Type III are modules that operate on two signed values. Only one Type III module is required for any type of multiplier, whereas $2\frac{W}{m}$ - 2 Type II modules and $(\frac{W}{m} - 1)^2$ Type I modules are needed. Figure 2 shows an example of an 8-bit wide 2's pipelined complement radix-4 array multiplier.

2.1. Pipelined Array Multiplier

The regularity of this architecture makes it suitable for the application of other power reducing techniques. A *pipelined* version was constructed in order to reduce the critical path and useless signal transitions that are propagated through the array. The doted lines in **Figure 1** show the pipelined version of the radix-4 array multiplier for 8-bit operands. As can be observed, the advantage of the layered structure of the array was taken into account and two layers of registers were introduced. Thus, 3 clock cycles are necessary to perform the computation considering 8-bit architectures.

3. Modified Booth multiplier

The radix-4 Booth's algorithm (also called Modified Booth) has been presented in [4]. In this architecture it is possible to reduce the number of partial products by encoding the two's complement multiplier. In the circuit the control signals (0, +Y, +2Y, -Y and -2Y) are generated from the multiplier operand Y for each 3-bit group, as shown in the example of **Figure 3**, for an 8-bit wide operation. A multiplexer produces the partial product according to the encoded control signal.

Common to both architectures is that, at each step of the algorithm, two bits are processed. However, the basic Booth cells are not simple adders as in the array multiplier, but must perform addition-subtraction-no operation and controlled left-shift of the bits of the multiplicand. **Figure 4** shows an example of an 8-bit modified Booth architecture.

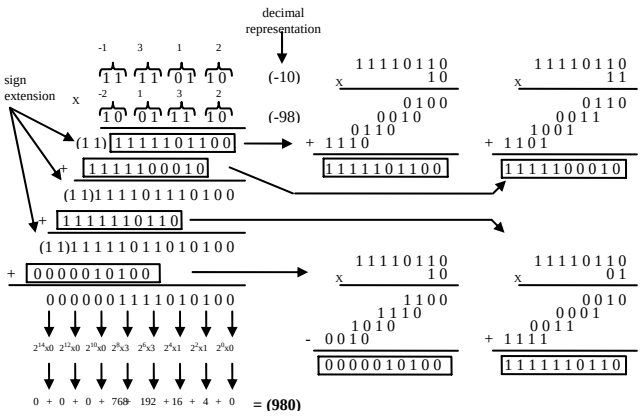

Figure 2 – Example of a 2's complement 8-bit wide radix-4 multiplication

3.1. Pipelined Modified Booth Multiplier

A pipelined Modified Booth by introducing registers along the layers of the array was implemented in and it is presented in **Figure 4**. As it can be observed in this figure, there are two layers of registers along the array as in the binary array multiplier with *m*=2. Again, 3 clock cycles are required to compute the final result in the 8-bit architecture and six cycles to the 16-bit one. Moreover, common to both architectures is that the registers are inserted at the output of the adders which are responsible for adding the partial product terms. However, in the Booth multiplier it is also necessary to introduce registers in the output of the encoders to perform the correct operation of each clock cycle as shown in **Figure 4**.

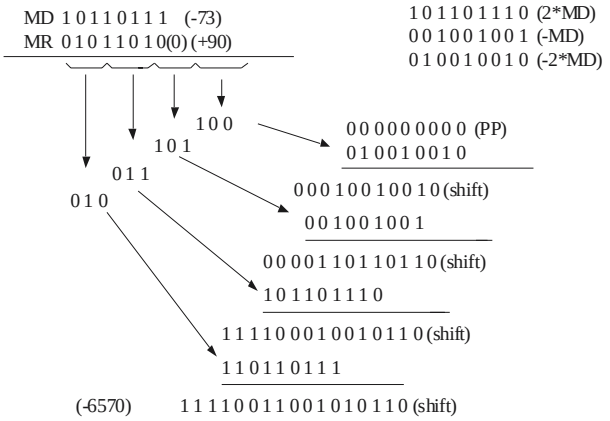

Figure 3 – Example of an 8-bit multiplication with Modified Booth algorithm

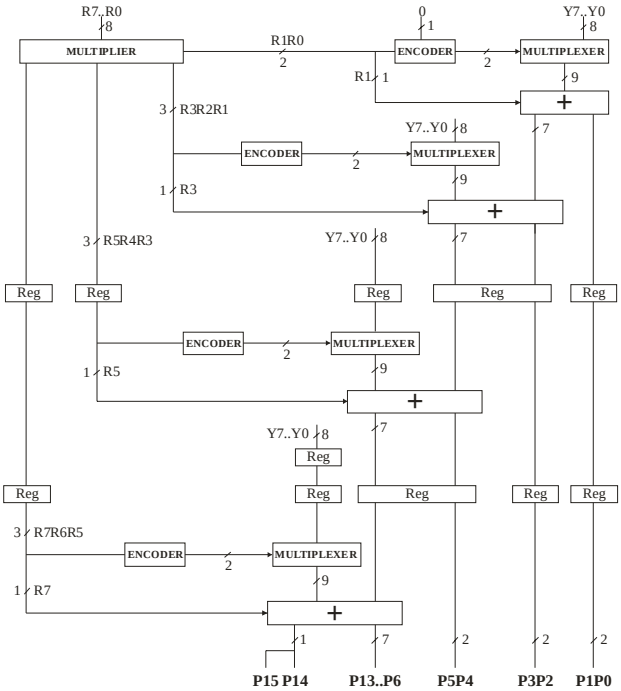

Figure 4 – 8-bit pipelined modified Booth architecture

4. Design Methodology

Figure 6 shows the design flow used in the implementation of the multipliers. In this figure we present both methodologies, our methodology is in black, and the methodology used in [7] and [8] with the SIS environment, in gray. The multipliers were described in a Berkeley Logic Interchange Format (BLIF). Thus, BLIF files are used as input of the design flow, as can be observed in **Figure 6**.

In [4] and [5], the multipliers were synthesized in a logic level and the SIS tool [14] was used to estimate area and delay of the multipliers. In the methodology of [4] and [5], the power consumption of the multipliers was estimated using the switch-level simulator SLS [7]. Thus, it was necessary to use a converter from BLIF to SLS format.

For the synthesis of the multipliers implemented in this work, the Tropic tool is used. This tool uses a *sim* format (specific input syntax to describe transistor netlists) as input and performs an automatic synthesis of the layout of the circuit. Thus, a converter from BLIF to *sim* format has been developed in this work. Tropic gives the total area and the number of transistors of the synthesized circuits. Before the layout synthesis of the circuits, it is necessary to set the width and length of the transistors in a cell library. Additionally, we have to specify parameters of the technology and the number of bands. This last parameter is useful to set the aspect ratio width/height. **Figure 5** shows the layout for the 8-bit array multiplier, which was generated automatically by Tropic tool.

Since the Tropic tool generates the widely used *cif* format, the resulting circuit layout can be visualized with Mentor Graphics IC Station tool. Once the *cif* file is generated, an electrical extraction can be performed using the Tropic tool. The extracted SPICE netlist files were simulated using the ELDO electrical simulator that gives power estimation at the back-annotated electrical level.

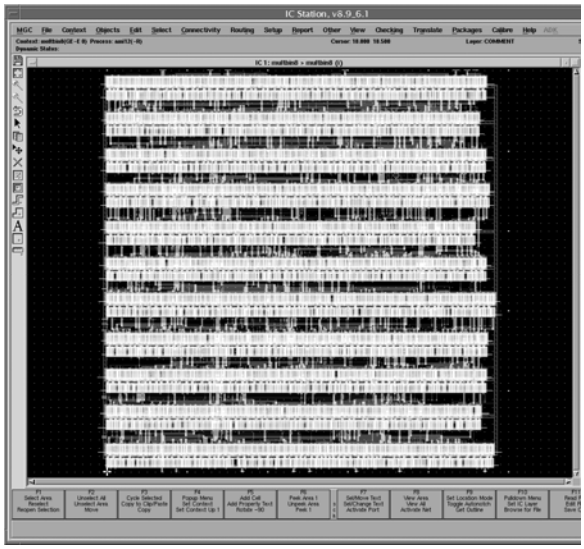

Figure 5 – Layout of an 8-bit array multiplier generated automatically by TROPIC

This simulator is part of the Mentor Graphics environment for power estimation, we have used a set of input vectors, which are firstly converted from SLS to SPICE format and then are used for transient analysis. A tool, named Gerabits Converter, was developed for this conversion step.

A timing analysis tool, called TicTac [9], was used to estimate the critical delay of the circuit. Topological delay was estimated. Topological Timing Analysis (Static Timing Analysis) is the standard approach used in the current designs complexity. TicTac tool uses a state-of-art sensitization criterion that considers logical and temporal information.

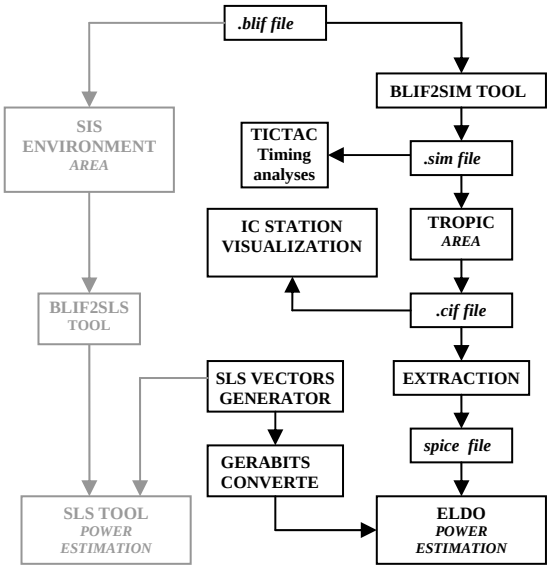

Figure 6 – Design tools for synthesis and power and area estimation

5. Performance Comparisons

In this section, we present area, delay and power results for the 16-bit multipliers after layout generation. For the implemented circuits, we have used HCMOS 0.25µm technology. Area results were obtained using the Tropic layout generation tool and are presented both in terms of total area and in terms of number of transistors. Power results are presented in terms of average power consumption and were obtained by using ELDO simulator. For the power simulation we have applied a random pattern signal with 100 input vectors. elay results were obtained by using the TicTac timing analysis tool. The topological delay is the longest path delay. We have not applied yet any transistor-level techniques which can further improve the efficiency of booth architectures.

5.1. Pipelined and Non-Pipelined Results

Table 1 presents area results for 16-bit radix-4 Booth and the new array multiplier proposed in [5], both implemented in layout level.

Table 1 - Area results for 16-bit parallel multipliers

	Parameter	Array	Booth	Diff(%)
non-pipelined	Number of transistors	12484	10064	-19.4
	Total area (mm^2)	0.2872	0.2172	-24.4
pipelined	Number of transistors	23014	21220	-7.8
	Total area (mm^2)	0.4829	0.4608	-4.6

As it can be observed in **Table 1**, the array multiplier presents the highest area and number of transistors. This occurs due to the fact that the partial product lines operate on group of *m* bits and the basic multiplier elements, which compose the modules for the product terms, are slightly more complex. The introduction of registers along the layers of the arrays increases the area of both architectures when compared to the non-pipelined architectures as shown in **Table 1**. Although the array multiplier presents the highest area value, this architecture can be slightly more efficient in terms of delay result as presented in **Table 2**. This is due to the lower logic depth presented by our proposed architecture.

Table 2 - Delay results (*ps*) for 16-bit parallel multipliers

	Array	Booth	Diff(%)
pipelined	3187ps	3740ps	+17.3
non- pipelined	7650ps	7558ps	-1.2

Figure 2 and **Figure 4** show that while in the pipelined array multiplier the critical path is given by a *m*=2 multiplier module and 2 full adders, in the pipelined Modified Booth, the critical path includes the encoder, an operand circuit composed by a multiplexer and a full adder. These circuits produce a large number of interconnections and a longer delay per row. Thus, the array multiplier presents less delay values than the Modified Booth even in the pipelined version as shown in **Table 2**.

As observed in [1], the major sources of power dissipation in multipliers are spurious transitions and logic races that flow through the circuit. Thus, the significantly less amount of spurious transitions in the new array multiplier justifies the gain in power when compared against the Booth multiplier as shown in **Table 3**. Moreover, the new array multiplier presents less logic depth due to the more balanced paths to the basic blocks that compose the array architecture. This contributes for improvement in power reduction because of the less generation of useless transitions. **Table 3** also shows that although our architecture consumes less power, it should be observed that the Modified Booth reduces more power when compared to non-pipelined version. This occurs because in the pipelined approach glitching is reduced significantly. This reduction will have a greater impact in the case where the glitching was higher. However, the reduced logic depth and delay presented by our architecture still makes it significantly more efficient, as shown in **Table 3**.

Table 3 - Power results for 16-bit parallel multipliers

	Array (*mW*)	Booth (*mW*)	Diff(%)
pipelined	14.76	17.12	+16.0
non-pipelined	10.76	16.75	+55.7

5.2. Comparison between Electrical and Logic Results

Table 4 shows area, delay and power percentage changes between the pipelined and non-pipelined array and Modified Booth multipliers. The estimates at the logic level and after layout correlate well for power. Area estimates at the logic level is just the number of literals coming from logic synthesis (SIS environment). Delay at the logic level was also estimated in SIS environment by using *mcnc* library. The relative power estimations are fairly close as shown in **Table 4**. In the logic level power results were obtained by using a random pattern input signal with 10,000 input vectors. The larger number of glitches generated in the Modified Booth makes this architecture more power consuming in both pipelined and non-pipelined version, which is captured with the SLS simulator. This validates the results reported in [4] and [5] at gate level design.

Table 4 - Comparison between parallel multipliers in electrical and logic simulations

	non-pipelined		pipelined	
Parameter	Logic Level	Electrical Level	Logic Level	Electrical Level
Area (n. of transistors)	-20.2%	-19.4%	-14.4%	-7.8%
Delay (ns)	+1.1%	-1.2%	+15.2%	+17.3%
Power (*mW*)	+35.7%	+16.0%	+18.7%	+55.7%

6. Acknowledgments

The support of CNPq, PDI-TI-CTINFO, FAPERGS is gratefully acknowledged.

7. Conclusions

We have described the layout implementation of a new array multiplier and Modified Booth multiplier both in pipelined and non-pipelined versions operating in 2's complement numbers using radix-2^m encoding. We have presented results that show significant improvement in power consumption in the new pipelined and non-pipelined array multiplier. We have compared the new array and Modified Booth multipliers both simulated in the logic and electrical level. The results showed that the relative values at the two levels of abstraction are similar when we compare the power consumption of the multipliers. As future work we hope to be able to prototype these architectures in order to experimentally validate these results.

8. References

[1] Callaway, T.; Swartzlander, E. Optimizing multipliers for WSI. In Fifth Annual IEEE International Conference on Wafer Scale Integration, pages 85-94, 1993.

[2] Cherkauer, B; Friedman, E. A Hybrid Radix-4/Radix-8 Low Power, High Speed Multiplier Architecture for Wide Bit Widths. In IEEE International Symposium on Circuits and Systems, volume 4, pages 53–56, 1996.

[3] Costa E. da; Monteiro J., and S. Bampi. A New Architecture for 2's Complement Gray Encoded Array Multiplier. In Proceedings Symposium on Integrated Circuits and Systems, pages 14-19, 2002.

[4] Costa, E., Monteiro, J., Bampi, S. A New Architecture for Signed Radix-2m Pure Array Multiplier. IEEE ICCD, September 2002.

[5] Costa, E., Bampi, S., Monteiro, J. A New Pipelined Array Architecture for Signed Multiplication. 16th SBCCI, September 2003.

[6] Gallagher, W. and Swartzlander, E. High Radix Both Multipliers Using Reduced Area Adder Trees. In Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, volume I, pages 545-549, 1994.

[7] Genderen, A. J. SLS: An Efficient Switch-Level Timing Simulator Using Min-Max Voltage Waveforms. Proceedings of VLSI Conference, pages 79-88, 1989.

[8] Goto, G.; et al. A 4.1-ns Compact 54 x 54-b Multiplier Utilizing Sign-Select Booth Encoders. IEEE Journal of Solid-State Circuits, 32:1676-1682, 1997.

[9] Santos, C.; Wilke, G.; Lazzari, C.; Reis, R.; Guntzel, J.L. A transistor sizing method applied to an automatic layout generation tool, 16th Symposium on Integrated Circuits and Systems Design, Proceedings, pages 303-307, Sept. 2003.

[10] Hwang, K. Computer Arithmetic - Principles, Architecture and Design. John Wiley & Sons, 1979.

[11] Khater, I.; Bellaouar, A.; Elmasry, M. Circuit Techniques for CMOS Low-Power, High-Performance Multipliers. IEEE Journal of Solid-State Circuits, 31:1535-1546, 1996.

[12] Moraes, F. A Virtual CMOS Library Approach for Fast Layout Synthesis. In: IFIP TC10 WG10.5 International Conference on Very Large Scale Integration, 10, pages 415-426, 1999.

[13] Seidel P., Mcfearin, L. and MATULA D. Binary Multiplication Radix-32 and Radix-256. In 15th Symposium. on Computer Arithmetic, pages 23–32, 2001.

[14] Sentovich, E. and et al. SIS: A System for Sequential Circuit Synthesis. Technical report, University of California at Berkeley, UCB/ERL – Memorandum nº M92/41, 1992.

[15] Wallace, C. A Suggestion for a Fast Multiplier. IEEE Transactions on Electronic Computers, 13:14–17, 1964.