Array Hybrid Multiplier versus Modified Booth Multiplier: Comparing Area and Power Consumption of Layout Implementations of Signed Radix-4 Architectures

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

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

Sergio Bampi
UFRGS,P.Alegre,Brazil
bampi@inf.ufrgs.br

João Baptista
UFSM/PPGEE – Santa Maria, Brazil
batista@inf.ufsm.br

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

Abstract- **In this paper, we describe the fully automated custom layout implementations of two architectures for signed multiplication. Performance comparisons between the two, namely in terms of their power consumption and area estimation are provided for 8 and 16-bit operands. The first architecture consists of a signed array multiplier that uses a radix-4 hybrid encoding to reduce the partial product lines and switching activity in the data buses. This new arithmetic operand encoding was recently proposed in [4], however only results at the logic level were presented. The second architecture implemented was the widely used modified Booth multiplier [9]. The layout of both multipliers was generated by an automatic layout synthesis tool called TROPIC [10]. We compare the layout implementations in terms of area and power, as well as provide comparisons to first-order area estimates done in the logic design phase. The results show that the new hybrid array multiplier can be significantly more efficient, with close to 30% power savings.**

I. INTRODUCTION

Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Wallace multiplier [13] is among the fastest. However, they do not have such a regular structure as the conventional array [8] or Booth [9] multipliers. Hence, when layout regularity, high-performance and low power are primary concerns, Booth multipliers tend to be the primary choice [2], [5], [7], [9], [11]. In this paper, we present layout implementations for both the Modified Booth multiplier and the new array multiplier using hybrid code proposed in [4]. We synthesize the multipliers by using an automatic synthesis tool named Tropic [10]. In order to compare the Modified Booth and the hybrid array architectures, both using radix-4, the ELDO which is a spice-like simulator and is part of the Mentor Graphics environment, was used. The results show that the new array multiplier is significantly more efficient, saving more than 30% in power consumption. The power reduction presented by the new array multiplier is mainly due to the lower logic depth, which has a large impact in the amount of glitching in the circuit. We should stress further that, in contrast to the architecture presented in [4], increasing 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 hybrid code, the hybrid array multiplier and the modified Booth multiplier. After this, we describe the design methodology, the fully automated layout synthesis and how area and power results were obtained. Comparisons between the radix-4 hybrid array multiplier architecture and the Modified Booth, for both switch level and electrical level are presented following the sequence. Finally we conclude this paper, discussing the main contributions and future work.

II. 2'S COMPLEMENT MULTIPLIERS

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 hybrid encoding.

A. Hybrid Code

The idea of the hybrid code is to split the operands in groups of m-bits, encode each group using the Gray code and to use the Binary approach to propagate the carry between the groups. In this way, a compromise between the minimum Hamming distance between consecutive values of the Gray code and the minimum bit dependency of the Binary code is achieved [3]. Table I shows the 2's complement Hybrid encoding for 4-bit numbers and m=2.

TABLE I
2'S COMPLEMENT HYBRID CODE REPRESENTATION FOR M=2.

Dec	Hyb	Dec	Hyb	Dec	Hyb	Dec	Hyb
0	0000	4	0100	-8	1100	-4	1000
1	0001	5	0101	-7	1101	-3	1001
2	0011	6	0111	-6	1111	-2	1011
3	0010	7	0110	-5	1110	-1	1010

B. Radix-2^m Hybrid Array Multiplier

The radix-2^m operation in 2's complement representation is given by Equation 1.

$$R \times Y = R' \times Y' - R' y_{W-1} y_{\frac{W}{m}-1} 2^{W-m} - r_{W-1} r_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} y_j 2^{W-m+j} \quad (1)$$

This operation is illustrated in Fig. 1. For the case of radix-4 we have a conversion from hybrid code to binary code at the input of values. After that, the bits are calculated to binary encoding. Finally, at the end of multiplication, the final value is converted to hybrid encoding.

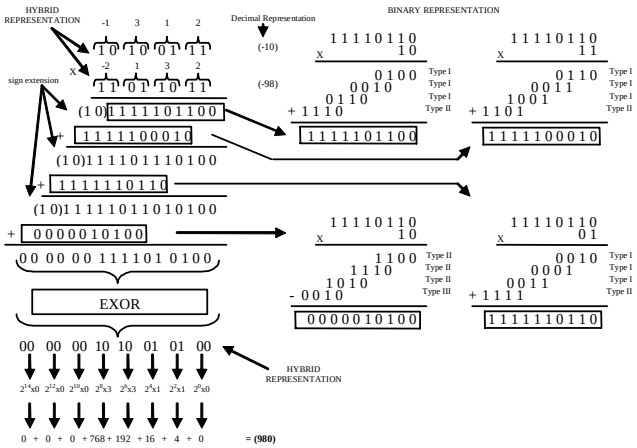

Fig. 1 2's complement 8-bit wide radix-4 hybrid multiplication

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 as shown in Fig. 3. 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.

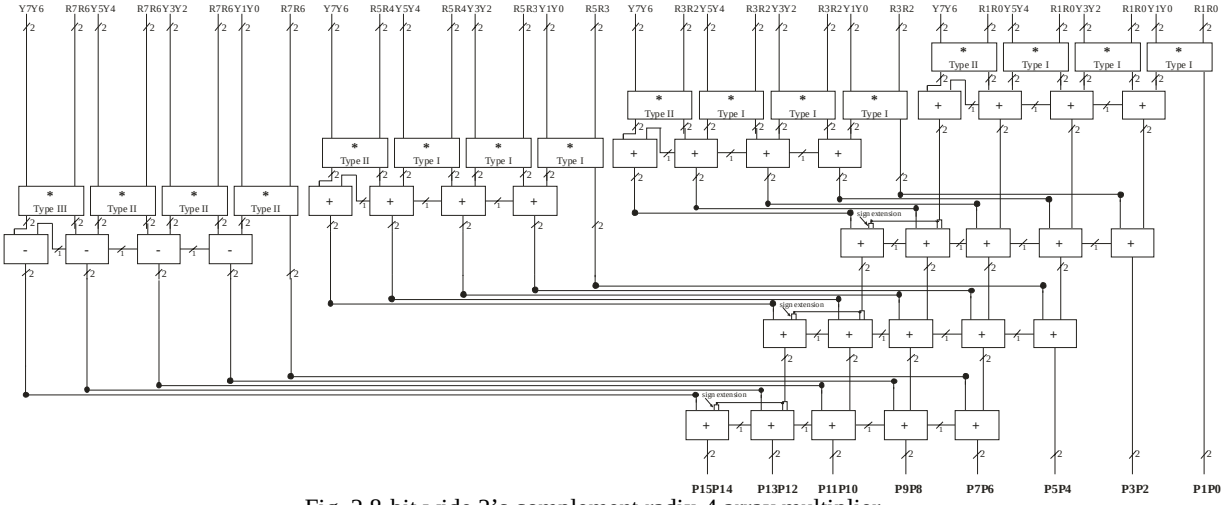

Fig. 2 8-bit wide 2's complement radix-4 array multiplier.

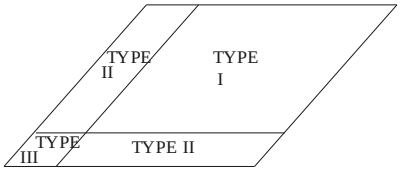

Fig. 3 General structure for a 2's complement radix-2^m multiplier.

Only one Type III module is required for any type of multiplier, whereas $\left(2\frac{W}{m}-2\right)$ Type II modules and $\left(\frac{W}{m}-1\right)^2$ Type I modules are needed. We present a concrete example for W=8 bit wide operands using radix-4 (m=2) in Fig. 2.

C. Modified Booth

The radix-4 modified Booth algorithm has been presented in [9]. 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. A multiplexer produces the partial product according to the encoded control signal.

One of the main problems in the modified Booth multiplier is the generation of the 2´s complement for the multiplicand term. This is calculated by each multiplexer shown in the example of Fig. 4 for an 8-bit wide.

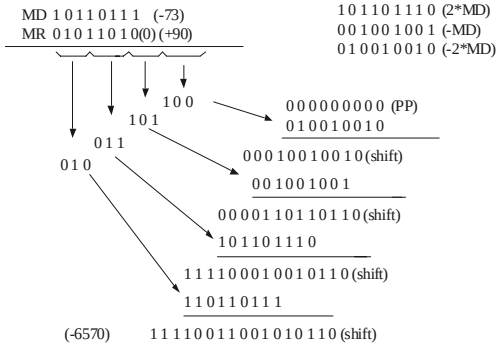

Fig. 4 Example of an 8-bit multiplication using Modified Booth algorithm

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. Besides taking more area, this complexity also makes it more difficult to increase the radix value in the Booth architecture.

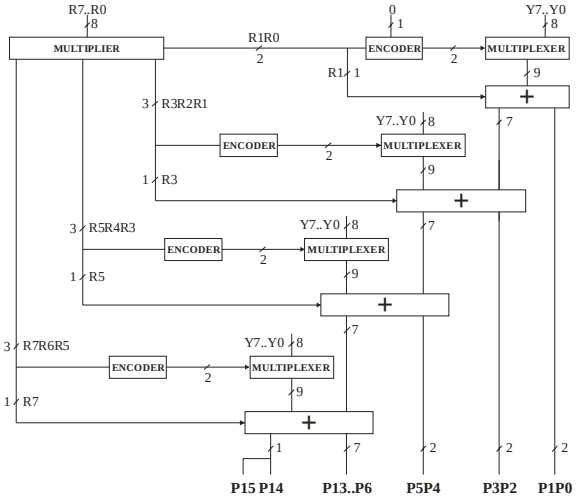

Fig. 5 8-bit Modified Booth Architecture.

As can be observed in Fig. 5, for an 8-bit architecture, 4 encoders and 4 multiplexers are necessary in order to calculate the partial product terms. The multiplexers produce the multiplicand term according to the 3 bits that are generated in the encoder circuit. The partial product terms are shifted by the adders, which are also used to produce the final result.

III. Design Methodology

In this section, we present the design flow for the layout implementation of the multipliers based on an automatic layout generation tool called TROPIC [10].

A. Design Flow

Fig. 6 shows the design flow used in the logic [3], in gray, and layout synthesis of the multipliers. In this figure we present both our methodology in black and the methodology used in [4], in grey, for the analysis of the multipliers.

The multiplier circuits 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 Fig. 6. In [3], where the multipliers were synthesized in a logic level, the

SIS tool [12] is used in order to estimate area of the multipliers. In the methodology of [3], the power consumption of the multipliers was estimated using the switch-level simulator SLS [6]. Thus, it was necessary to use a converter from *BLIF* to SLS format. For the power estimation, different types of vectors, in SLS format, were used in the primary inputs of the multipliers. For the synthesis of the multipliers implemented in this work, the Tropic tool is used. This tool uses a *sim* format 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. In the Tropic environment, the total area and the number of transistors of the circuits are also estimated. Since the Tropic tool generates the widely used *cif* format, the resulting circuit layout can be visualized with Mentor Graphics IC Station tool (Mentor Graphics Environment). Extracted SPICE netlist files were simulated in the Mentor Graphics environment for power consumption estimation at the back-annotated electrical level. The ELDO electrical simulator was used. Power simulation is dependent on a set of input vectors. In this paper we compare results for two sets of vectors: random and representing sinusoidal signal. These vectors were generated for the SLS tool binary digit format and converted into a set of input vectors in SPICE format for transient analysis also using a tool made especially for this project (Gerabits Converter).

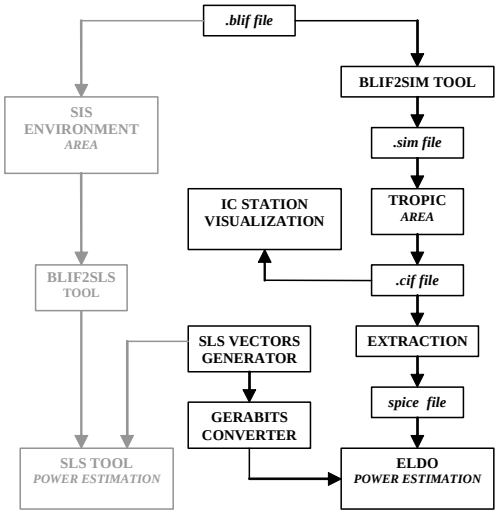

Fig. 6 Design tools for synthesis and power and area estimation.

B. *TROPIC Layout Generation Tool*

Tropic is an automatic synthesis tool that is used in this work in order to generate the multiplier layout circuits. It is a macro-cell generator, i.e., the complete circuit is generated. This tool uses a specific input syntax to describe transistor netlists. The static CMOS gates layout is automatically generated with up to 3 metal layers. The main features of the layout style are: linear matrix layout style, where each cell row is composed of two horizontal diffusion strips, and the width of the transistors is orthogonal to the diffusion strips; over the cell routing is implemented with 3 metal layers and stacked contacts, reducing the routing area; layout without compaction is used (cif format); the complete parasitic capacitance evaluation after layout synthesis is performed; simple technology files are used to describe design rules (28 rules) and parasitic capacitances (26 rules). 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. Fig. 7 shows the layout for the 8-bit array multiplier, which was generated automatically by Tropic tool.

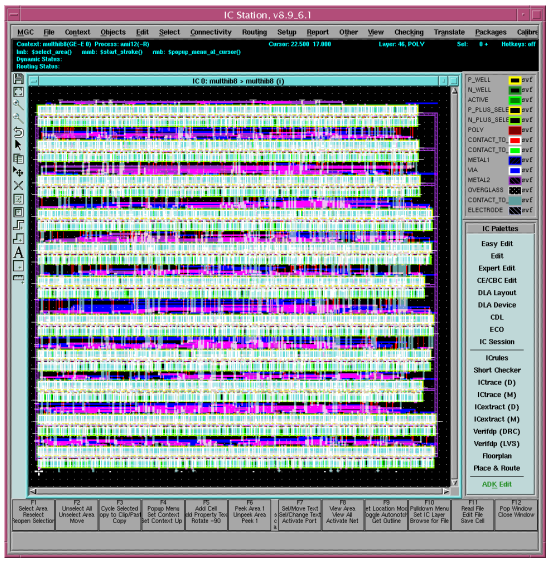

Fig. 7 Layout of an 8-bit hybrid multiplier generated automatically by TROPIC.

IV. Performance Comparisons

In this section, we present area and power results for the multiplier architectures after layout, both W=8 and 16 bit-wide operands. The modified Booth and the hybrid array multiplier proposed in [4] are compared. In the design a HCMOS 0.25µm technology is used. We set the number of cell bands generated by TROPIC such that the aspect ratio width/height close to 1 was obtained.

Area results were obtained using the TROPIC tool and are presented both in terms of total area and in terms of number of transistors. Automatic layout compaction is not used in this design. Total average power is presented and was obtained by ELDO simulations. Real trace input signals and a random pattern signal with 500 input vectors was used. The real trace signal represents two sinusoidal signals with 90 degrees phase difference. In these simulations we have used Vdd=2.5V and f=50MHz frequency operation. We have also compared our results with those reported in [4], considering 16-bit architectures and for random pattern input vectors.

A. *Area Results*

In the Booth multiplier, an encoding scheme is used in order to reduce the number of stages in multiplication. Two bits of multiplication are performed at once and thus the multiplier requires half the stages. In the hybrid array multiplier proposed in [4] the number of stages can be reduced by more than half while the regularity can be kept as in the pure array multiplier. Table II presents area results for 8-bit radix-4 Booth and the new hybrid array multiplier proposed in [4], both implemented in layout level.

TABLE II

AREA RESULTS FOR 8-BIT HYBRID ARRAY AND MODIFIED BOOTH RADIX-4 MULTIPLIER ARCHITECTURES.

Parameter	8-bit multiplier		
	Booth	Hybrid	Diff(%)
Number of transistors	2542.0	3418	+34.5
Width (µm)	225.4	276.45	+22.6
Height(µm)	221.0	258.6	+17.0
Total area (mm^2)	0.04982	0.07149	+43.5

As can be observed in Table II, the hybrid array multiplier presents the highest area and number of transistors values. The same results can be observed in Table III, for 16-bit multipliers. 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. As can be compared in Table II and Table III, a larger area results for both the 8 and 16-bit array multipliers we proposed.

TABLE III

AREA RESULTS FOR 16-BIT HYBRID ARRAY AND MODIFIED BOOTH RADIX-4 MULTIPLIER ARCHITECTURES.

Parameter	16-bit multiplier		
	Booth	Hybrid	Diff(%)
Number of transistors	10064	14066	+39.8
Width (μm)	468.7	520	+10.9
Height(μm)	463.4	594.4	+28.3
Total area (mm^2)	0.2172	0.30909	+42.3

B. *Power Results*

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 observed in Table IV and Table V, for 8 and 16-bit circuits respectively. Also noticeable in these tables, the array and Booth multipliers present more power reduction when using a sinusoidal waveform as input. This occurs due to the larger degree of correlation between input samples, leading to less abrupt variations, hence a smaller number of input bits toggling. As observed in [4], the new hybrid 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 IV

POWER RESULTS FOR 8-BIT HYBRID ARRAY AND MODIFIED BOOTH.

Vectors	8-bit multiplier		
	Booth (mW)	Hybrid (mW)	Diff(%)
Sinusoidal	6.5	3.8	-41.5
Random pattern	14.5	10.1	-30.3

TABLE V

POWER RESULTS FOR 16-BIT HYBRID ARRAY AND MODIFIED BOOTH..

Vectors	16-bit multiplier		
	Booth (mW)	Hybrid (mW)	Diff(%)
Sinusoidal	67.2	42.9	-36.2
Random pattern	83.0	56.4	-32.0

It is also important observe the results obtained in [4] with the results obtained in our work in terms of way of simulation. In [4], power consumption of the multipliers was estimated using the SLS switch-level simulator, where glitching activity is taken into account. Moreover, at the logic level power results were obtained by using a random pattern input signal with 10,000 input vectors. Area was estimated in SIS [12] environment and the results were presented in terms of number of literals, which is approximately half of the number of transistors

TABLE VI

COMPARISON BETWEEN ELECTRICAL/LOGIC SIMULATIONS (16-BIT).

Parameter	Logic Level	Electrical Level
Area (number of transistors)	+35.8%	+42.3%
Power (mW)	-31.3%	-32.0%

Table VI shows area and power percentage changes between the new hybrid array and modified Booth multipliers. The estimates at the logic level and after layout correlate extremely well for both area and power. Area estimates at the logic level is just the number of literals coming from logic synthesis. The relative power estimations are fairly close. The larger number of glitches generated in the modified Booth makes this architecture more power consuming, which is captured with the SLS simulator. This validates the results reported in [4] for the gate level design.

V. CONCLUSIONS

We have described the layout implementation of a new hybrid array multiplier that operates in 2´s complement numbers using radix-2^m encoding. The signed radix-2^m hybrid array multiplier we proposed exhibits much lower power consumption than the modified Booth multipliers. The power saving have reached up to 32% considering 16-bit architectures and a random pattern input vector. Comparison of the logic and electrical level estimates were done and support that the modified Booth multiplier consumes more power when glitching activity is correctly simulated.

We combined both academic and commercial tools in this design and power estimation. Delay and silicon results are being experimentally tested and will be shown in future work.

ACKNOWLEDGMENT

This research was partially supported by the CNPQ (Brasília, Brazil) and FCT (Portugal) foundations for R&D support.

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. Power Efficient Arithmetic Operand Encoding. In Proceedings Symposium on Integrated Circuits and Systems Design, pages 201–206, 2001

[4] 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

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

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