COMPARING 2'S COMPLEMENT MULTIPLIERS WITH BINARY AND HYBRID OPERAND ENCODING

Leonardo L. de Oliveira[1], Marcos Boschetti[2], Eduardo da Costa[3], Sergio Bampi[2], João Baptista[1]
leonardo@mail.ufsm.br, marcosrb@inf.ufrgs.br, ecosta@atlas.ucpel.tche.br, bampi@inf.ufrgs.br, batista@inf.ufsm.br

[1]Universidade Federal de Santa Maria – UFSM – PPGEE
[2]Universidade Federal do Rio Grande do Sul – UFRGS – GME
[3]Universidade Católica de Pelotas – UCPEL

ABSTRACT

In this paper we present the analysis and design flow of new architectures for signed multiplication. These architectures maintain the pure form of an array multiplier and are extended for radix-2 encoding, which leads to a reduction in the number of partial lines.. Layout results for the multipliers using Binary and a Hybrid encoding representation for the operands are shown. The proposed architectures have been synthesized using SIS environment and the layouts were generated using Tropic tool in a 0.25um technology. Area and power comparisons between Binary and Hybrid architectures are also presented.

1. INTRODUCTION

Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. In this category, the Wallace multiplier [1] is among the fastest. However, it suffers from bad layout regularity. In this paper, we propose a new approach to handle operands in 2's complement. We use exactly the same structure of an array multiplier, with the same unsigned bit products for all the bits except those that involve sign bits. This new approach enables the implementation of regular and compact layouts. Besides the Binary representation, we use a Hybrid operand encoding [2] to obtain arithmetic modules that operate directly on these codes.

2. RELATED WORK

A substantial amount of research work has been put in developing efficient architectures for multipliers given their widespread use and complexity. Early multiplier schemes such as bi-section, Baugh-Wooley and Hwang [3] propose the implementation of a 2's complement architecture using repetitive modules with uniform interconnection patterns. However, the irregular tree-array form used does not permit an efficient VLSI realization. More suitable multiplier designs based on the Booth encoding technique have been proposed [4], [5]. The main purpose of these designs is to increase the performance of the circuit by the reduction of the number of partial products. As the Booth architecture, in our work the improvements in the Binary and Hybrid multipliers are based on the reduction of the partial product terms, while keeping the regularity of an array multiplier [3].

3. 2'S COMPLEMENT MULTIPIER ARCHITECTURES

For the operation of a radix-2m multiplication, the operands are split into groups of m bits. Each of these groups can be seen as representing a digit in a radix-2m. For the Hybrid code we encode each group using the Gray code for operands that operate in 2's complement representation [2]. 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

For the Binary and Hybrid multipliers, three types of modules are needed. `Type I` are the unsigned modules used in the previous section. `Type II` modules handle the *m*-bit partial product of an unsigned value with a 2's complement value. Finally, `Type III` 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. We present a concrete example for *W*=8 bit wide operands using radix-16 (*m*=4) in Figure 1. The example shown in this figure represents a Binary multiplier, since the sign extension shown in the dotted lines is represented in Binary mode. For the Hybrid circuit, the modules and the sign extension are performed in Hybrid encoding representation [6].

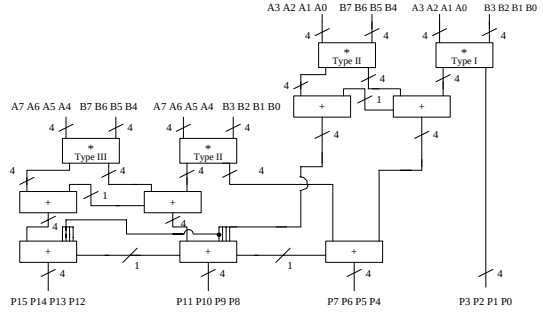

Fig. 1. Example of an 8-bit wide 2's complement radix-16 array mult.

4. DESIGN METHODOLOGY

In order to simplify and make faster the electrical implementation and verification of both multipliers, many tools were used in this project. The design flow is showed bellow:

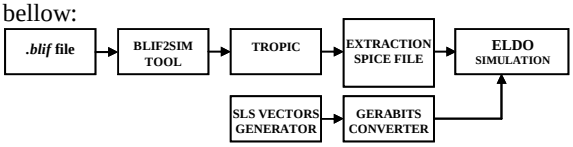

Fig. 2. Flowchart project

The multipliers were described in SIS environment [SIS] by using blif files. The files were converted to input Tropic language with the Blif2sim tool (a converter developed for this project). The Tropic3 tool was used for the generation of the cif files and extraction into a spice language file. The last step was the simulation in the ELDO simulator. For the simulation of the multipliers, the input vectors were generated by a SLS to ELDO converter.

5. RESULTS

In this section we present comparison results for Binary and Hybrid multipliers using radix-4(m=2). Area results were obtained in the Tropic environment. Power results were obtained with the ELDO simulator tool. For the power simulation we have applied a real trace input signal and a random pattern signal, with 50 input vectors. The real trace signal represents two sinusoidal signals with 90 degree phase difference. A HCMOS 0.25µm technology from a specific *techfile* was used.

5.1. Area Results

For the layouts generation, all transistors have the minimum channel length (0.25µm) e 5µm for the width. Table II shows the area comparisons between Binary and Hybrid multipliers. As can be observed in this table, Binary multiplier presents the less area values. This due the simplest structure of the basic modules.

TABLE II. Area table for 16 and 8 multipliers

	16 bits				8 bits			
Multiplier	**number of transistors**	**with (µm)**	**height (µm)**	**total area (µm2)**	**number of transistors**	**with (µm)**	**height (µm)**	**total area (µm2)**
BINARY	12484	554.45	518	0.28721	2976	240.45	248.6	0.05978
HYBRID	14066	520	594.4	0.30909	3418	276.45	258.6	0.07149
diff bin to hyb	-11.25%	6.63%	-12.85%	-7.08%	-12.93%	-13.02%	-3.87%	-16.38%

5.2. Power Results

Table III shows the power results for both multipliers. Table IV shows the comparisons between the multipliers. As can be observed in Table IV, Binary and Hybrid multiplier present almost the same values, although the Binary presents slightly less values. This occurs due to the simplest structure of the Binary multiplier.

TABLE III. Power Table

			POWER (W)					
			16 bits			8 bits		
	vector	*freq. MHz*	*rms*	*average*	*global cpu time*	*rms*	*average*	*global cpu time*
binay	*RAN 50P*	*1*	1.5595E-02	-1.0590E-03	9h 12mn 13s	3.3250E-03	-1.7212E-04	46mn 2s
		10	4.8449E-02	-1.0251E-02	8h 53mn 31s	1.0527E-02	-1.6869E-03	47mn 31s
		50	1.1077E-01	-5.2390E-02	8h 36mn 20s	2.3625E-02	-8.4372E-03	45mn 5s
	SIN 50P	*1*	1.3357E-02	-8.5578E-04	7h 55mn 31s	2.5128E-03	-1.2016E-04	39mn 43s
		10	4.2230E-02	-8.4592E-03	8h 0mn 24s	7.9132E-03	-1.1668E-03	40mn 43s
		50	9.4789E-02	-4.2136E-02	7h 25mn 10s	1.7783E-02	-5.8075E-03	37mn 33s
hybrid	*RAN 50P*	*1*	1.6792E-02	-1.1427E-03	10h 12mn 45s	3.6489E-03	-1.9298E-04	53mn 59s
		10	5.3153E-02	-1.1267E-02	10h 25mn 10s	1.1546E-02	-1.8894E-03	52mn 1s
		50	1.1923E-01	-5.6258E-02	9h 45mn 35s	2.5852E-02	-9.4290E-03	52mn 30s
	SIN 50P	*1*	1.3924E-02	-9.0360E-04	8h 46mn 14s	2.7290E-03	-1.3694E-04	45mn 14s
		10	4.4053E-02	-8.9035E-03	8h 34 mn 28s	8.5778E-03	-1.3246E-03	46mn 49s
		50	9.8734E-02	-4.4367E-02	8h 2mn 32s	1.9307E-02	-6.6018E-03	43mn 54s

TABLE IV. Comparison table

			16 bits		8 bits	
	vector	*freq. MHz*	*rms*	*average*	*rms*	*average*
hybrid to binary	*RAN 50P*	*1*	7.68%	7.90%	9.74%	12.12%
		10	9.71%	9.91%	9.68%	12.00%
		50	7.64%	7.38%	9.43%	11.76%
	SIN 50P	*1*	4.24%	5.59%	8.60%	13.96%
		10	4.32%	5.25%	8.40%	13.52%
		50	4.16%	5.29%	8.57%	13.68%

6. CONCLUSIONS

In this paper we presented layouts for new multiplier architectures for 2's complement operation. The regularity of the multipliers was explored in the layout design implementation. As could be observed, the new Binary multiplier presented less area and power values. However, it should be observed that the new Hybrid multiplier can be used to reduce the power consumption in data buses with Hybrid code. Table IV shows that as we increase the number of bits of the multiplier we decrease the difference between the power consumption of the hybrid and binary multipliers. Finally, it was shown that it is feasible to combine academicals and commercials tools to get good design results.

7. ACKNOWLEDGMENT

This research was partially supported by the CNPQ (Brasília/Brazil).

8. REFERENCES

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

[2] E. da Costa, J. Monteiro, and S. Bampi. Power Optimization Using Coding Methods on Arithmetic Operators. In IEEE International Symposium on Signals Circuits and Systems, pages 505–508, 2001.

[3] K. Hwang. Computer Arithmetic - Principles, Architecture and Design. School of Electrical Engineering, 1979.

[4] B. Cherkauer and E. Friedman. 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.

[5] P. Seidel, L. McFearin, and D. Matula. Binary Multiplication Radix-32 and Radix-256. In 15th Symp. on Computer Arithmetic, pages 23–32, 2001.

[6] E. da Costa, J. Monteiro, and S. Bampi. A New Architecture for 2's Complement Gray Encoded Array Multiplier . In Proceedings of the XV Symposium on Integrated Circuits and Systems Design., September 2002.

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