CMOS IMPLEMENTATION OF A HYBRID RADIX-4 DIVIDER

Luis MONTALVO, Alain GUYOT.

Integrated Systems Design Group, TIMA - INPG,
46 Av. Félix Viallet, 38031 Grenoble Cedex, France.
Phone: +(33) 76 57 48 34, Fax: +(33) 76 47 38 14,
e-mail: montalvo@verdon.imag.fr

ABSTRACT

A 1.2 um CMOS combinational implementation of a new hybrid radix-4 division algorithm is presented. The algorithm is based on Svoboda's division and is named hybrid because: the dividend, the quotient, and the remainder are represented using the signed-digit-set {2, 1, 0, 1, 2}; while the divisor is represented using the conventional digit-set {0, 1, 2, 3}. The divider requires the divisor Y to be pre-scaled to the range 1 ≤ Y < 1 + 1/8. For 16 bit accuracy, the radix-4 divider is about 50 % less expensive but 12 % slower than a corresponding radix-2 divider.

INDEX TERMS

Radix-4 division, signed-digit-sets, hybrid arithmetic.

1. INTRODUCTION

A radix-2 add/subtract-shift division algorithm produces one bit of the quotient at each iteration while a radix-4 division produces two bits. This fact might lead to suppose that radix-4 division is twice as fast as radix-2 division; however the overall division time depends also on the latency per iteration, which is higher for radix-4 than for radix-2 and may turn the radix-4 division even slower than the radix-2 one. This paper presents a 1.2 um CMOS combinational implementation of a new hybrid radix-4 division algorithm [1] developed at the TIMA/INPG Laboratory and compares its results to those of the CMOS ver-sion of the radix-2 division algorithm reported in [2]. A radix-4 division algorithm, in some aspects similar to ours, has independently been developed by Srinivas and Parhi and is to appear in [3]

2. THE HYBRID RADIX-4 DIVISION ALGORITHM

The hybrid radix-4 division algorithm is a variant of Svoboda's division [4] and is based on the recurrence: $R^{(j+1)} = 4 * R^{(j)} - q_{j+1} * Y$. Y is the divisor, $R^{(j)} = \sum_{i=0}^{n-1} (r_i^{(j)} * 4^{-i}) * 4^{-j}$ is the remainder, and $Q = \sum_{j=0}^{n-1} q_j * 4^{-j}$ is the quotient [5], [6]. The notation in use is the following: j = 0, 1, ..., n-1 is the recursion index, i = 0, 1, ..., n-1 is the digit-weight-index of the remainder, n is the word length of the quotient, $r_i^{(j)}$ is the i_{th} digit of the j_{th} remainder, q_{j+1} is the $(j+1)_{th}$ quotient digit to the right of the radix point, $R^{(0)}$ is the dividend X, and $R^{(n-1)}$ is the final remainder.

At each iteration of the recurrence, the q_{j+1} quotient-digit is selected by examining the two most significant digits of the $R^{(j)}$ remainder only, according to the rules summarized in Table 1. Fig. 1 shows graphically the relation between the range of the remainder and the quotient digit selection function. The arithmetic bounds are given by: $-Y < R^{(j+1)} < Y$, and $1 \le Y < 1 + 1/8$.

Table 1: q_{j+1} selection function.

$r_1^{(j)}$ \ $r_2^{(j)}$	$\bar{2}$	$\bar{1}$	0	1	2
$\bar{2}$	$\bar{2}$				$\bar{1}$
$\bar{1}$	$\bar{1}$				$\bar{0}$
0	$\bar{0}$	0 or $\bar{0}$			0
1	0	1			
2	1	2			

The hybrid quality of the algorithm comes from the fact that the dividend X, the quotient Q and the remainder $R^{(j+1)}$ are represented using the signed-digit-set {2, 1, 0, 1, 2}; while the

divisor Y is represented using the conventional digit-set {0, 1, 2, 3}. This quality differentiates our algorithm from the standard SRT radix-4 division, and from Srinivas et. al.'s algorithm [3], where the partial remainder is represented with the digit-set {$\bar{3}$, $\bar{2}$, $\bar{1}$, 0, 1, 2, 3}.

The r_i digits of the remainder are encoded with three bits according to the equation: $r_i = r_i^{+} + r_i^{++} - 2 * r_i^{2}$. For the q_j digits of the quotient we use the special sign and magnitude code of three bits (add & u1u2) specified in Table 2, where: add denotes the sign and, u1u2 the magnitude. Here, the (+, *) symbols are used to indicate arithmetic addition and multiplication.

Table 2: Binary encoding of q_{j+1}.

Control signals			
add	u1	u2	q_{j+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	2
1	0	0	$\bar{1}$
1	0	1	$\bar{0}$
1	1	0	$\bar{2}$
1	1	1	$\bar{0}$

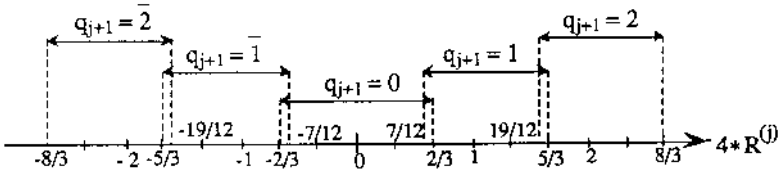

Fig. 1: Correspondance between the range of the remainder and q_{j+1}.

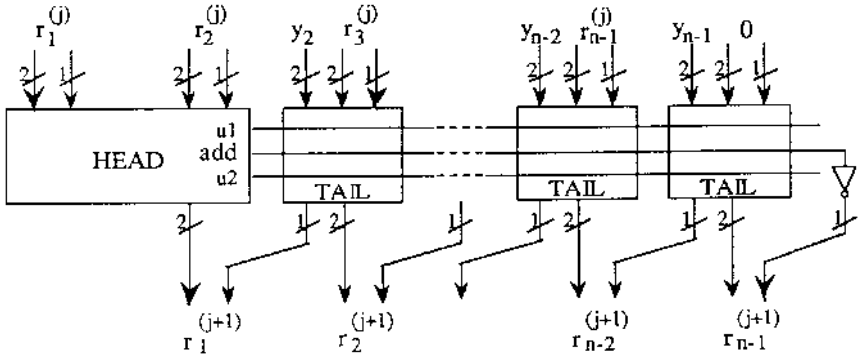

Fig. 2: Structure of a slice of the radix-4 DIVIDER.

3. RANGE REDUCTION OF THE DIVISOR AND QUOTIENT CONVERSION

Since the IEEE 754 standard specifies a normalized range for the operands ($1 \le Y < 2$), the originals divisor and dividend are multiplied by a constant K so that the divisor fits into the scaled range $1 \le Y < 1 + 1/8$. The constant K is such that it can be computed by the addition of three terms of the form 2^{-i}, i being an integer, hence a full adder can be used as a multiplier.

The signed-digit quotient Q is converted back to the conventional notation in parallel with the generation of the q_j signed-digits (On the fly conversion [7]). The details of these mechanisms can be found in [1].

4. LOGIC DESIGN OF THE DIVIDER

The hybrid radix-4 DIVIDER is organized as a stack of n-1 identical slices, each one performing an iteration of the recurrence. A slice of the DIVIDER is composed by a HEAD

cell and an iterative logic array (ILA) of n-1 TAIL cells (Fig. 2).

The HEAD computes the quotient digit q_{j+1}, and part of the most significant digit of the remainder $r_1^{(j+1)}$. The logic equations of the QUOTIENT SELECTOR of the HEAD are as follows (The $\wedge$, $\vee$ and $\oplus$ symbols are used to indicate the logical AND, OR, and XOR operations):

$$m2 = r_2^{-2} \wedge \overline{r_2^{+}} \wedge \overline{r_2^{++}}; \quad p2 = \overline{r_2^{-2}} \wedge r_2^{+} \wedge r_2^{++}$$

$$add = r_1^{-2} \wedge (\overline{r_1^{++}} \vee \overline{r_1^{+}}) \vee m2 \wedge (\overline{r_1^{-2}} \vee \overline{r_1^{+}} \wedge \overline{r_1^{++}})$$

$$u1 = \overline{p2} \wedge \overline{r_1^{+}} \wedge \overline{r_1^{++}} \vee \overline{m2} \wedge r_1^{+} \wedge r_1^{++}$$

$$u2 = \overline{m2} \wedge \overline{r_1^{-2}} \wedge (\overline{r_1^{++}} \vee \overline{r_1^{+}}) \vee p2 \wedge (r_1^{+} \oplus r_1^{++}) \vee m2 \wedge r_1^{+} \wedge r_1^{++} \vee \overline{r_1^{-2}} \wedge \overline{r_1^{+}} \wedge \overline{r_1^{++}}$$

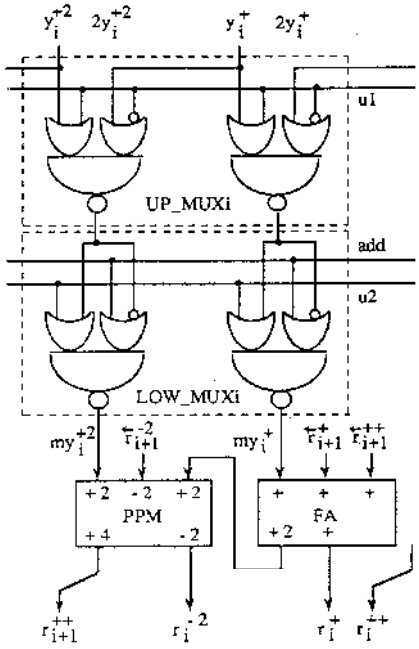

Fig. 3: Detailed circuitry of the TAIL cell.

The ILA of TAIL cells selects a multiple of the divisor Y; according to add, u1, u2; and computes the rest of the $r_1^{(j+1)}$ digits. Fig. 3 shows the details of the logic structure of a TAIL cell. The UP_MUXi multiplexer outputs $\overline{y_i}$ or $2\overline{y_i}$ depending on the value of u1. The LOW_MUXi places in my_i (my_i^{+2}, my_i^{+}), one of the variables $\{2y_i, 2\overline{y_i}, y_i, \overline{y_i}, 0, 1\}$, depending on the values of the add and u2 variables. The PPM (Plus Plus Minus) and FA (Full Adder) circuits constitute the needed hybrid radix-4 adder.

5. LAYOUT STRATEGIES

Préforme, a symbolic layout design tool [8], was used to draw the DIVIDER's layout. Préforme's basic principle is to make the designer draw on a grid that corresponds to the real layout grid but that is not identical to it as it is the case in a conventional full custom design tool. The designer still has to compose gates manually, but using symbols; the program then verifies the symbolic design according to the chosen grammar and, if succesful, generates the layout. Such tools liberate the designer from learning the fairly complicated and ever-changing layout design rules imposed by the ever-improving fabrication technologies; the designer only needs to learn the grammar, containing a few basic rules pertaining the relative distances of the sticks.

Fig. 3 shows that a PPM and a FA circuits are required to build-up the hybrid radix-4 adder in the TAIL cell. The same structure is also needed in the HEAD cell to calculate part of the $r_1^{(j+1)}$ digit of the remainder. In order to save draftsman time, a symetrical FA (one which has non-inverting inputs and inverting outputs, so-called because the N and the P transistor arrangements are exactly the same) plus two inverters are used as a basis for the FA as well as the PPM circuits. This technique implies very simple rewiring of the layout of the FA to get the PPM, no transistor rearrangment is needed.

6. PERFORMANCE OF THE DIVIDER AND CONCLUSIONS

In this section, three categories of comparison between the radix-2 and the radix-4 dividers for 16-bit accuracy are presented [9]: 1) theoretical estimates based on simplified models of the critical path and complexity, 2) spice simulation based on the transistor schematics and 3) spice simulation based on the extracted layouts (the layout of the radix-4 divider is shown in Fig. 4). The results are given, in terms of delay and cost in Table 3. The theoretical estimate category assumes that the logic functions are implemented using AND and OR gates only, considering a fan-in of 4: delay is measured in terms of gate delays, assigning a unit gate delay to any AND and OR gate; cost is measured in terms of the number of

gates needed to implement a logic function. Inverters are not considered in either delay or cost estimates. The transistor schematic and the extracted layout categories consider 1.2 um CMOS technology. In the case of the transistor schematic one: delay is measured in terms of ns, and cost is measured in terms of the number of transistors needed to implement a logic function; as for the extracted layout one: delay is measured in terms of ns, and cost is measured in terms of mm^2.

From Table 3, we can observe that the theoretical estimates for the cost are nearly the same as the one obtained; the radix-4 divider is about 50 % less expensive than the radix-2 one. As regards delay, the theoretical estimates differ from the spice simulation results; theoretically the radix-4 divider should be about 17 % faster than the radix-2 divider, however the spice simulation of the corresponding transistor schematics show that the radix-4 divider is about 12 % slower than the radix-2 divider. No data is available for the extracted layout of the radix-2 divider and then no comparison is possible.

The 1.2 um CMOS implementation of our hybrid radix-4 divider turns out to be good with regard to cost and satisfactory with regard to speed, which could be substantially improved, without major area increase, by logic redesign of the selection and arithmetic circuitry of the TAIL cell.

Table 3: Delay and cost of the 16-bits accuracy radix-2 and radix-4 dividers.

	Delay			Cost		
Divider type (Delay units - Cost units)	Radix 2	Radix 4	Ratio	Radix 2	Radix 4	Ratio
Theoretical estimate (# gates - # gates)	96	80	0.83	3,968	1,976	0.498
Transistor schematic (ns - # transistors)	40	44.8	1.12	11,264	9,196	0.816
Extracted layout (ns - mm^2)	-	71.2	-	2.30	1.07	0.465

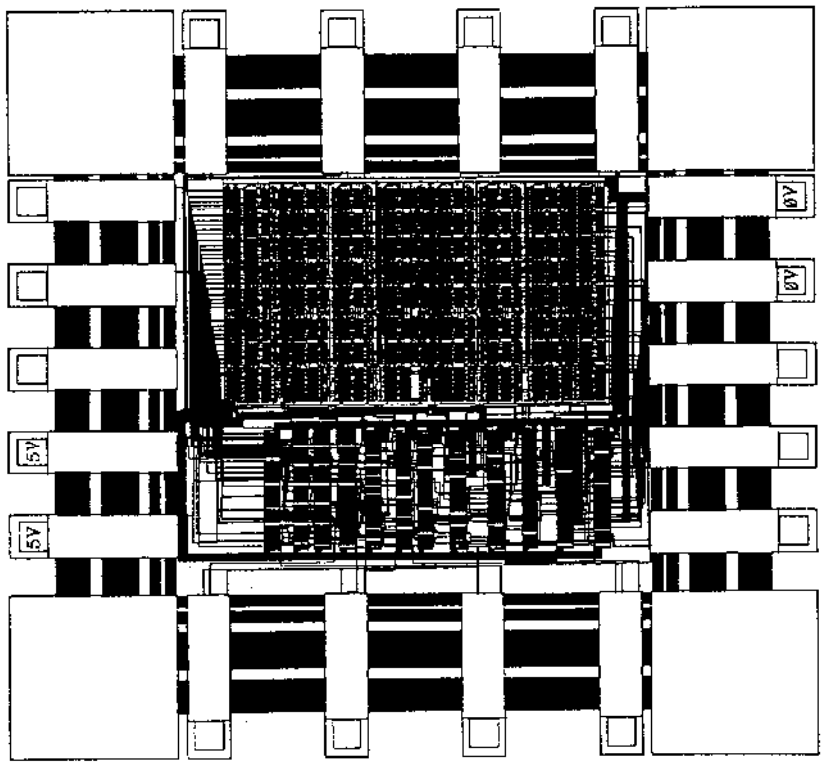

Fig. 4 Layout of the hybrid radix-4 divider for 16 bits accuracy.

REFERENCES

[1] L. Montalvo and A. Guyot, "A hybrid radix-4 divider with operands scaling," *TIMA/INPG Technical Report*, Oct. 1993.

[2] I. Moussa, A. Guyot and P. Rost, "Design and comparison of GaAs and CMOS redundant divider," in *Proc. ESSCIRC '93*, Sevilla - Spain, 1993.

[3] H. Srinivas and K. Parhi, "A fast radix-4 division algorithm," to appear in *Proc. IEEE International Conference on Circuits and Systems*, London - Great Britain, May 30 - June 2 1994.

[4] A. Svoboda, "An algorithm for division," *Information Processing Machines* (Prague, Czechoslovakia), no. 9, pp. 25 - 32, 1963.

[5] D. E. Atkins, "Higher-radix division using estimates of the divisor and partial remainders," *IEEE Trans. Comp.*, vol. C-17, no. 10, pp. 925 - 934, Oct. 1968.

[6] J. E. Robertson, "A new class of digital division methods," *IRE Trans. Electron. Comp.*, vol. EC-7, pp. 218 - 222, Sept. 1958.

[7] M. D. Ercegovac and T. Lang, "On the fly conversion of redundant into conventional representation," *IEEE Trans. Comp.*, vol. C-36, no. 7, pp. 895 - 897, July 1987.

[8] J. C. Dufourd and J. F. Naviner, "An optimizable model for process independant symbolic design," in *Proc. EDAC - ETC - Euroasic '94*, Paris - France, 1994.

[9] B. Behnam, "Hybrid radix-2 and radix-4 dividers / design, implementation and comparison," UJF - INPG, Grenoble - France, D.E.A Microélectronique Thesis, Sep. 1993.

BIOGRAPHIES

Alain Guyot was born in Trifouilly Les Oies, France, on Septembre 11, 1945. He received the Ph. D. degree from the University of Grenoble, France, in 1975; and the "Habilitátion à Diriger des Recherches" degree from the Institut National Polytechnique de Grenoble, France, in 1991.

Since 1986 to present, he has been an assistant professor in the Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble at the Institut National Polytechnique de Grenoble, Grenoble - France. He is the author or co-author of more than 50 scientific publications in Journals, Conference Proceedings and Research Reports. He has been a reviewer for ESSIRC, VLSI, Computer Arithmetic, EUROASIC, IEEE TC, CAVE and other conferences. He is also the Head of the Integrated Systems Design Group at the TIMA/INPG Laboratory.

Luis Montalvo was born in Quito - Ecuador, on August 3, 1955. He received the Engineering degree on Electronics and Telecommunications from the Escuela Politécnica Nacional, Quito - Ecuador, in 1982; and the M.S.E.E. degree from Ohio University, Athens - Ohio, in 1986.

He is currently pursuing his Ph. D. degree at the Institut National Polytechnique de Grenoble, France.

Between 1981 and 1993 he was a professor at the Electrical Engineering Department of the Escuela Politécnica Nacional, Quito - Ecuador. His present research interest is on computer arithmetic with enphasis in its application to public - key cryptography. He is a student member of the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers.