Combined Radix-10 and Radix-16 Division Unit

Tomás Lang and Alberto Nannarelli*

Dept. of Electrical Engineering and Computer Science, University of California, Irvine, USA
*Dept. of Informatics & Math. Modelling, Technical University of Denmark, Kongens Lyngby, Denmark

***Abstract*— In this work we extend a previously proposed digit-recurrence radix-10 division unit to be able to perform also radix-16 division. The extension is simplified by the fact that in the radix-10 implementation the quotient digit is decomposed into two parts and that this decomposition is also appropriate for the radix-16 case. Moreover, to reduce the latency in the radix-10 the most-significant portion of the datapath, including the selection function, has been implemented in radix-2, so that the modifications of that part to include radix-16 consists mainly in combining the two modules to obtain the selection constants. The rest of the modifications relate to the generation of multiples, to the carry-save adder, to the carry-propagate adder, and to the on-the-fly conversion and rounding. The implementation results show that the delay of an iteration is similar to that of the radix-10 case and that the area is about thirty percent larger.**

I. Introduction

Hardware implementations of decimal arithmetic units have recently gained importance because they provide higher accuracy in financial applications [1]. Moreover, to reduce the required area, it is convenient to perform in the same unit the operation for both decimal and binary representations. Combined units of this type have been proposed for addition [2] and multiplication [3]. In this work we propose a combined unit for the division operation.

Previously we described a radix-10 division unit using the digit-recurrence approach [4]. Moreover, this approach has been used extensively for radix-2 representation [5].

Specifically, since the radix-10 unit produces one digit of the quotient per iteration and the radix-10 digit is represented in BCD by four bits, it seems appropriate to combine it with a radix-16 unit. This combination is simplified by the fact that in the radix-10 case we have decomposed the quotient digit into two parts, which is also the preferred method for implementing radix-16 division [5].

II. Division Algorithm

The expressions for the digit-recurrence iteration for the radix-10 case are [4]

$$\begin{aligned} v[j] &= 10w[j-1] - q_{Hj}(5d) \\ w[j] &= v[j] - q_{Lj}d \end{aligned}$$

with $q_{Hj} \in \{-1, 0, 1\}$ and $q_{Lj} \in \{-2, -1, 0, 1, 2\}$ for a redundancy factor $\rho_{10} = 7/9$.
Similarly, for the radix-16 case

$$\begin{aligned} v[j] &= 16w[j-1] - q_{Hj}(4d) \\ w[j] &= v[j] - q_{Lj}d \end{aligned}$$

with $q_{Hj} \in \{-2, -1, 0, 1, 2\}$ and $q_{Lj} \in \{-2, -1, 0, 1, 2\}$ for a redundancy factor $\rho_{16} = 10/16$.

Therefore, the two recurrences can be combined into

$$v[j] = rw[j-1] - q_{Hj}(kd) \quad (1)$$
$$w[j] = v[j] - q_{Lj}d \quad (2)$$

with quotient digit selection functions

$$q_H = SEL_H(\widehat{rw}, \widehat{d}) \quad (3)$$
$$q_L = SEL_L(\widehat{v}, \widehat{d}) \quad (4)$$

and quotient digit $q_j = kq_{Hj} + q_{Lj}$. The switch between the two radices is performed by setting a bit R such that

when $R = 0 \rightarrow r = 16$ and $k = 4$ (radix-16)
when $R = 1 \rightarrow r = 10$ and $k = 5$ (radix-10)

To ensure convergence, the recurrence is initialized as

$$w[0] = x/r^2 \ .$$

III. Divider Architecture

The scheme implementing the division recurrence of (1) and (2) is shown in Fig. 1. The divider is completed by a unit to convert the quotient-digit q_j from the signed-digit to the BCD, or to the binary unsigned, representation, and to perform the rounding.

As mentioned above, the radix selection is done with a signal R such that for radix-16 $R = 0$ and for radix-10 $R = 1$. Therefore, in the recurrence, we have to process data both in BCD (for radix-10) and in binary (for radix-16). This requires some modifications in the carry-save adders (CSAs) as explained in Section III-D.

In [4], to speed up the radix-10 division, we implement the most-significant slice (MS-slice) of the recurrence in radix-2 (two's complement). The conversion, one digit per iteration, from a BCD digit to a 4-bit binary digit is straightforward. When combining with radix-16, we need to apply only minor modifications in the MS-slice, as explained in Section III-C. One radix-16 digit is simply transferred from the dual-radix recurrence part every iteration.

In the following, we indicate with lower case letters (e.g. d) digit-vectors in the dual radix part of the recurrence and with upper case letters (e.g. D) bit-vectors in the MS-slice. When necessary to specify the radix, radix-10 digit-vectors are indicated with the subscript BCD (e.g. d_{BCD}).

We now discuss the implementation of the relevant blocks in Fig. 1 and the convert-and-round unit.

978-1-4244-2110-7/08/$25.00 ©2007 IEEE

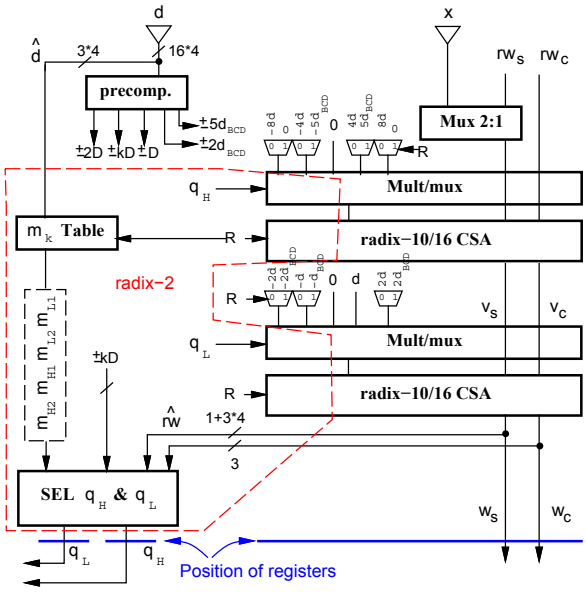

Fig. 1. Basic implementation of radix-10/radix-16 recurrence.

	-$q_H d$			-$q_L d$	
q_H	r-16	r-10	q_L	r-16	r-10
-2	$8d$	N/A	-2	$2d$	$2d_{BCD}$
-1	$4d$	$5d_{BCD}$	-1	d	d
0	0	0	0	0	0
1	$-4d$	$-5d_{BCD}$	1	$-d$	$-d_{BCD}$
2	$-8d$	N/A	2	$-2d$	$-2d_{BCD}$

TABLE I
OPERATION OF MULT/MUX.

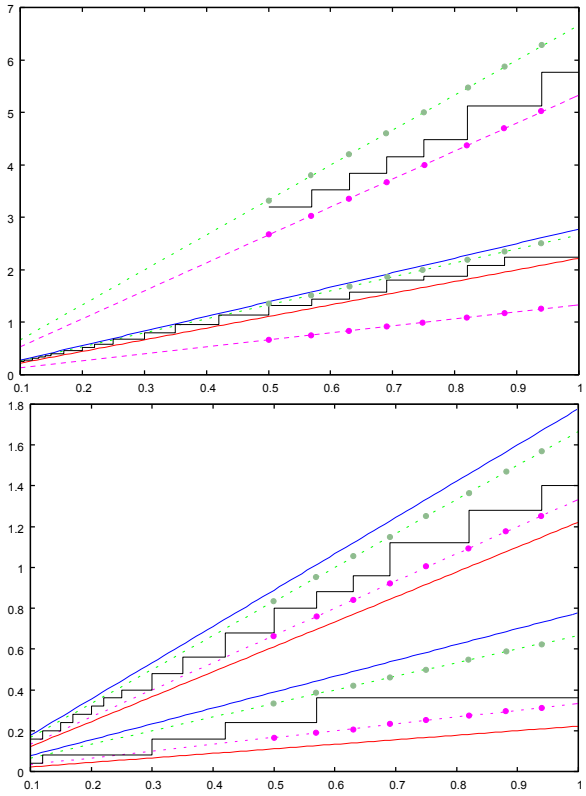

Fig. 2. PD plot for q_H (top) and q_L (bottom) (positive quadrant).

A. Precomputation of the multiples

This block computes the multiples of d necessary for both the recurrence and the selection function. In the radix-10 case the multiples five times the divisor ($5d_{BCD}$) and two times the divisor ($2d_{BCD}$), and their negatives, are precomputed. For the radix-16 the multiples required are eight, four, and two times the divisor ($8d$, $4d$, and $2d$); these are straightforward to compute (by shifting) and a selector is used to select among the multiples depending on the radix. The detail of the multiples selection for the dual radix recurrence is shown in Table I.

In the radix-2 MS-slice for the radix-16 division, we simply use a truncated representation of $8d$, $4d$, and $2d$. For radix-10 division, the multiples $5d_{BCD}$, $2d_{BCD}$ and their negatives are converted into two's complement.

B. Quotient-digit selection

In the quotient digit selection functions described by (3) and (4), the estimates $\widehat{rw}$ and $\widehat{v}$ are obtained by using a limited number of digits of the carry-save representation. Although in principle this number could be different, we use the same number to simplify the scheme.

The selection of the quotient-digit is done by preloading selection constants and comparison [6]. With respect to the radix-10 implementation of [4], in this dual radix divider we need to combine the radix-10 selection function with the radix-16 one. We explored two alternatives:

1) separate modules to generate the constants for each radix;
2) a combined module for both.

The combination of the radices can be done if the constants m_ks satisfy the conditions on the bounds of the selection intervals (see [5] and [4] for the detailed derivations). These bounds are shown in Fig. 2, for the positive quadrant. The dotted lines in the figure, represents the bounds for radix-16, and the solid lines the bounds for radix-10. The selection constants are then chosen by

$$L_k^{10} \leq m_k < U_{k+1}^{16} \qquad k = \{-1, 0, 1, 2\}$$

where L_k^{10} is the bound obtained for radix-10 (solid lines in Fig. 2) and U_{k+1}^{16} is the bound for radix-16 (dotted lines in Fig. 2).

Moreover, we choose constants which are symmetric with respect to the sign:

$$m_2 = -m_{-1} \quad \text{and} \quad m_1 = -m_0 \ .$$

For radix-16 (which selection function is decomposed into two radix-4 selection functions), 3 bits of the divisor d are sufficient to select the constant m_k

$$d = 0.1b_2b_3b_4\ldots$$

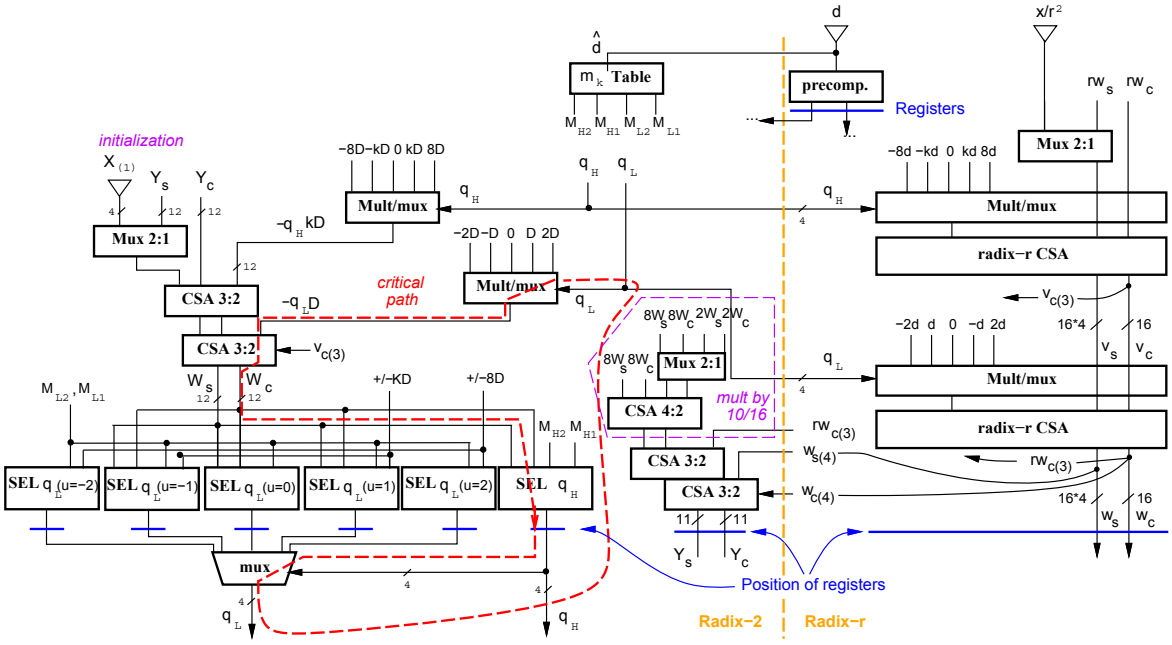

Fig. 3. Implementation of the dual-radix recurrence.

Therefore, to unify the intervals on d for both radices, we map the 8 configurations $0.1\ b_2b_3b_4$ into the closest 3 fractional digits BCD representation of d_{BCD}.

The resulting intervals on d and the constants for both q_H and q_L are reported in Table II. Their bounds are plotted in Fig. 2.

The values in Table II represent fractional numbers, that has to be implemented as integers in the radix-2 selection functions. This conversion fraction to integer is done by

$$M_k = m_k \cdot r^2$$

and therefore, we get two different encodings for radix-10 and radix-16.

Moreover, with respect to the radix-10 only implementation, the digit-set of q_H is extended from three to five values for radix-16. For this reason, the selection by comparison is modified by computing speculatively the five possible outcomes of $\hat{v}[j]$.

C. Radix-2 most-significant slice

With respect of the implementation of [4], the radix-2 MS-slice is modified as follows. A picture of the implementation of the recurrence is shown in Fig. 3.

1) To produce a 5-value quotient-digit q_H, the selection function for q_H is composed of four sign-detectors and the encoder of the quotient digit q_H is slightly changed.
2) As a consequence of 1), the multiplexer producing $q_H(kD)$ is changed into a 5:1 mux and two extra flip-flops are required to store q_H.
3) The speculative selection function for q_L is in this dual radix unit composed of five blocks computing speculatively

$$q_L = SEL_L(rW - \widehat{q_H(kD)}, \hat{d})\ ,\ q_H = \{-2, 1, 0, 1, 2\}$$

Consequently, a mux 5:1, controlled by q_H, must be used to select among the possible values of q_L.
4) The multiplication $rW[j]$ is performed by a CSA 4:2 and a multiplexer:

radix	R	inputs to CSA 4:2
16	0	$8W_s + 8W_c + 8W_s + 8W_c$
10	1	$8W_s + 8W_c + 2W_s + 2W_c$

D. Dual-radix carry-save adders

The carry-save adders (CSA) in the recurrence can be operated by selecting the radix with R. A scheme of the dual-radix CSA is shown in Fig. 4 for one digit, we indicate with $x_{(i)}$ the digit of weight r^{-i}.

E. Conversion and Rounding

The on-the-fly conversion and rounding implemented in [4] can be easily be adapted to the radix-16 case, with the exception of the normalization that in the binary case requires shifts of one bit, while in radix-10 the shifts is one BCD digit (4 bits). Moreover, the adder necessary to compute the sign of the final reminder, and to determine if it is zero, is implemented in dual-radix.

IV. Implementation and Comparisons

In this section we present the results of the evaluation of the dual radix division unit and a comparison with the decimal divider of [4] and a double-precision radix-16 digit-recurrence division unit.

$[d_i,\ d_{i+1})$	q_H				q_L			
	m_{H2}	m_{H1}	m_{H0}	$m_{H\overline{1}}$	m_{L2}	m_{L1}	m_{L0}	$m_{L\overline{1}}$
0.100, 0.106	-	0.26	-0.26	-	0.16	0.04	-0.04	-0.16
0.106, 0.120		0.28	-0.28					
0.12 , 0.13		0.32	-0.32		0.20	0.08	-0.08	-0.20
0.13 , 0.14		0.34	-0.34					
0.14 , 0.15		0.36	-0.36					
0.15 , 0.17		0.40	-0.40		0.24			-0.24
0.17 , 0.20		0.46	-0.46		0.28			-0.28
0.20 , 0.22		0.52	-0.52		0.32			-0.32
0.22 , 0.25		0.58	-0.58		0.36			-0.36
0.25 , 0.30		0.68	-0.68		0.40			-0.40
0.30 , 0.35		0.80	-0.80		0.48	0.16	-0.16	-0.48
0.35 , 0.42		0.96	-0.96		0.56			-0.56
0.42 , 0.50		1.14	-1.14		0.68	0.24	-0.24	-0.68
0.50 , 0.57	3.20	1.32	-1.32	-3.20	0.80			-0.80
0.57 , 0.63	3.52	1.44	-1.44	-3.52	0.88	0.36	-0.36	-0.88
0.63 , 0.69	3.84	1.58	-1.58	-3.84	0.96			-0.96
0.69 , 0.75	4.16	1.80	-1.80	-4.16	1.12			-1.12
0.75 , 0.82	4.48	1.88	-1.88	-4.48				
0.82 , 0.88	5.12	2.08	-2.08	-5.12	1.28			-1.28
0.88 , 0.94		2.24	-2.24					
0.94 , 1.00	5.76			-5.76	1.40			-1.40

TABLE II

CONSTANTS m_k FOR BOTH RADIX-10 AND RADIX-16 SELECTION.

Unit	cycle time $[ns]$	n. cycles	latency $[ns]$	speed-up	area $[\mu m^2]$	ratio
Radix-16 (standard)	1.00	16	16.0	1.00	38000	0.40
Radix-10 [4]	1.00	20	20.0	0.80	59700	0.60
					97700	1.00
Dual-radix (this work)	1.04	16/20	16.6/20.8	0.96/0.96	78500	0.80

TABLE III

SUMMARY OF RESULTS FOR THE SYNTHESIZED UNIT.

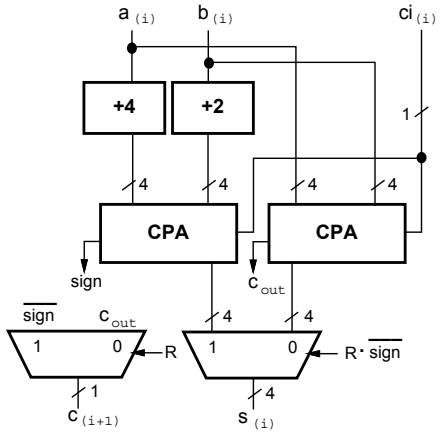

Fig. 4. Scheme of radix-10/radix-16 CSA (one digit).

We performed a synthesis of the unit of Fig. 3 (plus convert-and-round unit) using the STM 90 nm CMOS standard cells library [7] and Synopsys Design Compiler. From the synthesis we estimated the critical path (including estimations at netlist level of wire load) and the area. The critical path is highlighted in Fig. 3 (dotted line).

The results are compared with those of [4] for the radix-10 division and with those of [8] for radix-16.

The data in Table III show that the delay of the critical path for the dual radix unit is practically the same since the difference is about one INVFO4.

The additional area with respect to the implementation of [4] corresponds mainly to the following modules:

- multiplexers to select the multiples of the divisor in the precomputation block;
- module to compute the selection constants;
- the extra modules in the selection functions;
- the multiplexers for the CSAs in the dual radix recurrence (Fig. 4).

However, by comparing the area of the combined divider with separate units for each radix, we have about 20% less area.

V. CONCLUSIONS

We conclude that the combination of both radices in a single unit is feasible. The cycle time is similar to that of the radix-10 (and radix-16) implementation and the additional area can be justified by considering that the unit can perform both radix-10 and radix-16 divisions. The selection function might be simplified somewhat by modifying the implementation for radix-10 of [4] using a set for $q_H = \{-2, -1, 0, 1, 2\}$, since that is also required for radix-16.

REFERENCES

[1] M. F. Cowlishaw, "Decimal fbating-point: algorism for computers," in *Proc. of 16th Symposium on Computer Arithmetic*, June 2003, pp. 104–111.

[2] A. Vazquez and E. Antelo, "Conditional speculative decimal addition," *Proc. 7th Conference on Real Numbers and Computers (RNC 7)*, pp. 47–57, June 2006.

[3] A. Vazquez, E. Antelo, and P. Montuschi, "A new family of high-performance parallel decimal multipliers," *to appear in Proc. of 18th Symposium on Computer Arithmetic*, June 2007.

[4] T. Lang and A. Nannarelli, "A Radix-10 Digit-Recurrence Division Unit: Algorithm and Architecture," *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 727–739, June 2007.

[5] M. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic Publisher, 1994.

[6] N. Burgess and C. Hinds, "Design Issues in Radix-4 SRT Square Root and Divide Unit," *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, pp. 1646–1650, 2001.

[7] STMicroelectronics. 90nm CMOS090 Design Platform. [Online]. Available: http://www.st.com/stonline/prodpres/dedicate/soc/asic/90plat.htm

[8] E. Antelo, T. Lang, P. Montuschi, and A. Nannarelli, "Digit-recurrence dividers with reduced logical depth," *IEEE Transactions on Computers*, vol. 54, pp. 837–851, July 2005.