Design and FPGA Implementation of Radix-10 Algorithm for Division with Limited Precision Primitives

Miloš D. Ercegovac
Computer Science Department
Univ. of California at Los Angeles
California

Robert McIlhenny
Computer Science Department
California State University, Northridge,
California

***Abstract*— We present a radix-10 digit-recurrence algorithm for division using limited-precision multipliers, adders, and table-lookups. We describe the algorithm, a design, and its FPGA implementation. The proposed scheme is implemented on the Xilinx Virtex-5 FPGA device and we obtained the following characteristics: for $n = 7$, delay is $\approx 105ns$ and the cost is 782 LUTs. For $n = 14$, the implementation has a delay of $\approx 197ns$ and the cost of 1263 LUTs. The proposed scheme uses short operators which may have an advantage at the layout level and in power optimization.**

INTRODUCTION

Several digit-recurrence schemes for decimal division have been presented in recent papers [1], [7], [8], [9], intended for ASIC implementations. In this paper we present a decimal fixed-point division algorithm and design and implementation with the Xilinx Virtex-5 FPGA device [11]. The main feature of the algorithm is its use of short-precision primitive operators including adders, multipliers, and table-lookups. The corresponding scheme is highly modular. The algorithm is of the digit-recurrence type [4], that is, in each iteration, one digit of the quotient is obtained based on the shifted residual. The algorithm, proposed in [2], unlike a conventional digit-recurrence division, is characterized by

1) Single digit residual,
2) Digit-by-digit use of the dividend like in online division [4],
3) Quotient-digit selection by multiplying 2-digit divisor reciprocal by a 3-digit auxiliary residual,
4) Update of the residual by a convolution of quotient and divisor digits,
5) The final remainder is not produced; it can be obtained in extra steps; the quotient can be rounded.

The scheme in many respects corresponds to the conventional digit-recurrence division. It also has features of on-line division [4] methods in which the digits of the dividend and divisor are serially introduced to minimize the amount of computation and input communication bandwidth. In the proposed scheme, the digits of the dividend are used serially to keep the precision of the residual short. The divisor digits are used in parallel to have a short cycle time as discussed later. In on-line algorithms the error in the residual due to the incremental use of operands is compensated in each step by adding a term missed in prior steps. Specifically, this term is $Q[j] \cdot d_{j+\delta}$. Here a different method of error compensation is used. It is based on convolution which has been proposed by Fourier [10] for very long division. We call the proposed scheme F-division. The use of a short divisor to estimate the quotient digit has also been used in division algorithms which use redundant quotient digit set [5], [6]. The scheme is easily extended to evaluate square roots which can be combined with F-division [3].

I. OVERVIEW OF F-DIVISION METHOD

The dividend and the divisor are fixed-point numbers

$$x = \sum_{i=0}^{n-1} x_i 10^{-i},\ d = \sum_{i=0}^{n-1} d_i 10^{-i} \tag{1}$$

where $x_i, d_i \in \{0, \ldots, 9\}$, and $x < d \in [1, 2)$. As customary to avoid quotient overflow, x is shifted right, its precision extended by one digit and one extra iteration is performed. For implementation efficiency, the operands' digits are recoded into the digit set $\{-5, \ldots, 5\}$, prior to the operation. The algorithm begins with a truncated dividend and a divisor, and introduces one more digit of the dividend in each iteration. The divisor is used in parallel to have a short cycle time as discussed later. Here we choose two-digit initial dividend and divisor

$$x^* = 10x_0 + x_1 = x_1 \tag{2}$$

$$d^* = 10d_0 + d_1 = 10 + d_1 \quad (short\ divisor) \tag{3}$$

The quotient is $q = \sum_{i=0}^{n} q_i 10^{-i}$, $q_i \in \{-9, \ldots, 9\}$. For efficiency, the quotient digits are recoded internally as $q_j = 10q_j^H + q_j^L$, $q_j^H \in \{-1, 0, 1\}$ and $q_j^L \in \{-5, \ldots, 5\}$. The redundant quotient form can be converted to a conventional representation using on-the-fly conversion.

The residual recurrence is similar to a recurrence used in on-line division

$$\begin{aligned} w[j+1] &= 10w[j] + x_{j+1} - C[j] - q_{j+1}d^* \\ &= v[j+1] - q_{j+1}d^* \end{aligned} \tag{4}$$

$$C[j] = \sum_{i=1}^{j} q_i d_{j+2-i} \tag{5}$$

A. Compensation by convolution

The term

$$C[j] = \sum_{i=1}^{j} q_i d_{j+2-i} \tag{6}$$

compensates for the error caused by the digit-serial handling of the divisor. The computation of term C is a convolution of the divisor and quotient digits, that is, a summation of digit by digit products. For example, $C[3] = q_1 d_4 + q_2 d_3 + q_3 d_2$.

The maximum absolute value of $C[j]$ is

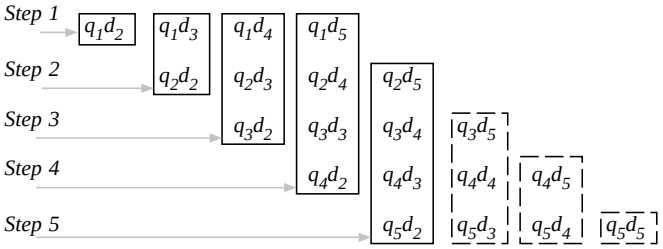
$$C_{max} = \sum_{i=1}^{n-1} max(q_j) \times max(d_k) \leq 25(n-1) \tag{7}$$

This implies that the maximum precision of the compensation factor C in radix-10 digits is

$$prec(C_{max}) \leq 3 \tag{8}$$

for $n \leq 40$. However, no precise bound has been found. The experiments indicate that the worst case may not happen.

Figure 1 illustrates the compensation in F-division.



Compensation in F-division (shaded boxes needed for full remainder):

a term consists of equal-weight digit-by-digit products

Fig. 1. Compensation by convolution

II. Division Algorithm

The proposed algorithm and an example are given next.

1. $[Initialize]$
   $v[1] = 10x_0 + x_1 = x_1; d^* = 10d_0 + d_1 = 10 + d_1$
   $g \leftarrow T_{REC}(d^*)$
   $q_1 = SEL(v[1], g)$
   $w[1] \leftarrow v[1] - q_1 d^*; C[1] \leftarrow q_1 d_2$
2. $[Recurrence]$
   for $j = 1 \ldots n$
   $v[j+1] = 10w[j] + x_{j+1} - C[j]$
   $q_{j+1} = SEL(v[j+1], g)$
   $w[j+1] \leftarrow v[j+1] - q_{j+1} d^*$
   $C[j+1] \leftarrow \sum_{i=1}^{j+1} q_i d_{j+3-i}$;
   $Q[j+1] \leftarrow OFC(Q[j], q_{j+1})$

j	x_{j+1}	$v[j+1]$	q_{j+1}	$w[j+1]$	$C[j+1]$	$Q[j+1]$
0	7	7	0	7	0	0
1	1	71	5	1	5	0.5
2	9	14	1	0	46	0.51
3	0	-46	-3	-4	16	0.507
4	5	-51	-4	5	16	0.5066
5	1	35	2	7	-6	0.50662
6	7	83	6	-1	9	0.506626
7	8	-11	-1	3	39	0.5066259
8	0	-9	-1	5	-3	0.50662589
9	0	53	4	-3	27	0.506625894
10	0	-57				

III. Design and Implementation

We discuss a sequential implementation of the proposed algorithm. We give the delay of the critical path, the total execution time, and the cost obtained from implementing the scheme on the Xilinx Virtex-5 FPGA device for $n = 7$ (single precision significand) and $n = 14$ (double precision). A block-diagram of digit-recurrence implementation is shown in Figure 2.

We now discuss the main modules of this implementation.

The quotient-digit selection function uses a short 2-digit reciprocal $g = 1/d^* = 1/(10 + d_1) = (1, g_2, g_3)$, obtained from a table. The quotient digit q_j is then obtained as the integer part of the rounded product $v[j] \times g$:

$$q_j = SEL(v[j], g) = sign(v[j]) \cdot int(|v[j] \cdot g| + 0.5) \tag{9}$$

It produces $q_j \in \{-9, \ldots, 9\}$ which is recoded for efficient multiplier implementation to the digit set $\{-5, \ldots, 5\}$.

Short Reciprocal Lookup Table

The table T_{REC} containing short reciprocals of the divisor produces the output
$g(d_0, d_1) = 10^{-3} int(10^3/(10d_0 + d_1)) = \sum_{i=1}^{3} g_i 10^{-i}$
Because $d_0 = 1$ and $d_1 \in \{0, \ldots, 9\}$ the lookup table has 10 entries, each entry having 3 digits recoded to the set

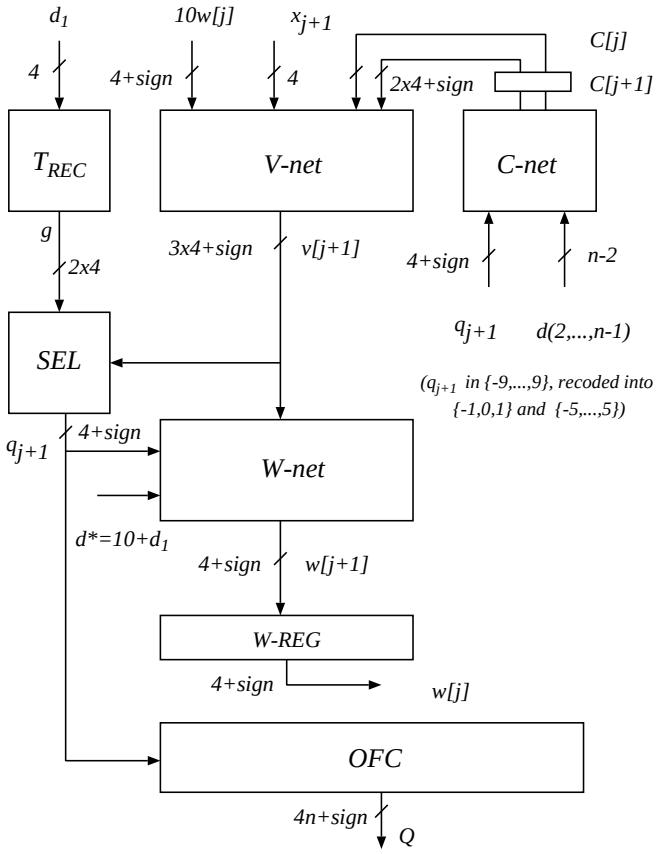


Fig. 2. Digit-recurrence implementation of decimal divider.

TABLE I

T_{REC} TABLE LOOKUP

d_1	rec	g_1	g_2	g_3
0	0.1	1	0	0
1	0.09	1	-1	0
2	0.083	1	-2	3
3	0.076	1	-2	-4
4	0.071	1	-3	1
5	0.066	1	-3	-4
6	0.062	1	-4	2
7	0.058	1	-4	-2
8	0.055	1	-4	-5
9	0.052	1	-5	2

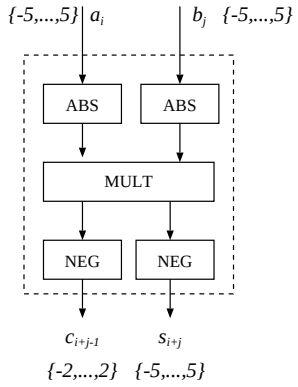


Fig. 3. Digit-by-digit multiplier

$\{-5, \ldots, 5\}$. Because $g_1 = 1$, it is not stored and the width of the able is $4 + 4$ bits (2 decimal digits)

Selection function

Note that $|v * g| + 0.5 < 10 \Rightarrow |v|_{max} < 9.5/0.052 < 193$. Therefore, $|v| = |\sum_{i=2}^{0} v_i 10^i|$, $v_1 = 0$ *or* 1. The partial product matrix for $v*g+(-1)^{sign(v)}0.5$ is shown below where the signed "5" in the last row performs the rounding.

v_2	v_1	v_0		
	$v_2 g_2$	$v_1 g_2$	$v_0 g_2$	
		$v_2 g_3$	$v_1 g_3$	$v_0 g_3$
		$(-1)^{sign(v)}5$		
q_a	q_b			
	q_i			

All input digits in the matrix are in the set {-5,....,5}. The quotient digit $q_i = 10q_a + q_b \in \{-9, \ldots, 9\}$. This is implemented with digit-by-digit multipliers and multi-operand signed-digit adders of the type [6:2], [4:2], and [3:2].

Digit-by-digit Multiplier

A digit-by-digit multiplier uses absolute values of the operands and adjusts the sign of the product accordingly.

The digit-by-digit multipliers are used in the selection function and the network for computing the compensation factor.

C-Net for Compensation

To have a short cycle, we evaluate the compensation factor $C[j] = \sum_{i=1}^{j} q_i d_{j+2-i}$ recursively in j cycles. For example,

$$C[4] = q_4 d_2 + (q_3 d_3 + (q_2 d_4 + q_1 d_5))$$

is computed in cycles 1,2,3, and 4 by doing a digit-by-digit multiplication and accumulation.

The recursive computation is defined by the following expression

TABLE II

DIGIT-BY-DIGIT MULTIPLICATION TABLE

$\|x_i\|, \|y_j\|$	1	2	3	4	5
1	0,1	0,2	0,3	0,4	0,5
2	0,2	0,4	1,-4	1,-2	1,0
3	0,3	1,-4	1,-1	1,2	1,5
4	0,4	1,-2	1,2	2,-4	2,0
5	0,5	1,0	1,5	2,0	2,5

c_{i+j-1}, $s_{i,j}$

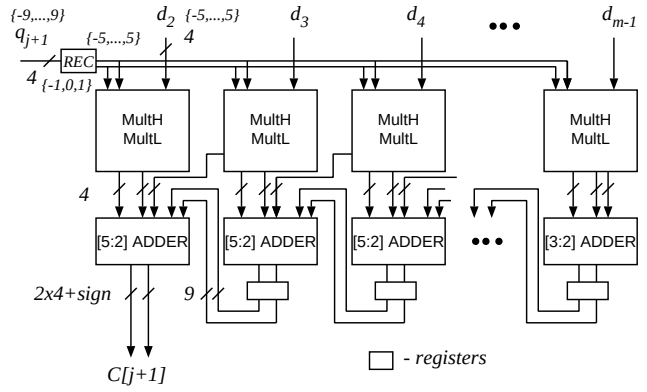


Fig. 4. Implementation of C-net

$$A[j,k] = q_j d_{2+k-j} + A[j-1,k]$$

$j = 1,\ldots,n-1$, $k = j, j+1,\ldots,n-1$, and $A[0,k] = 0$.

At iteration j, the value of the compensation factor is $C[j] = A[j,j]$.

To simplify the digit-by-digit multiplications, the selected quotient digit $q_{j+1} \in \{-9,\ldots,9\}$ is recoded as $10q^H + q^L$, resulting in high and low multiplier digits $q^H \in \{-1,0,1\}$ and $q^L \in \{-5,\ldots,5\}$. Now we perform digit-by-digit multiplication $q_{j+1} \times d_j \in \{-5,\ldots,5\}$ as

$$10q^H * d_k + q^L * d_k = (c^H, s^H, c^L, s^L)$$

producing the product in carry-save form.

The high and low digit multipliers operate in parallel:

- MultH: $\{-5,....,5\} \times \{-1,0,1\}$ trivially produces : $-d_j$, 0, or d_j
- MultL : digit-by-digit multiplier with operand digits in {-5,....,5}.

In each digit slice of the compensation network we add:

- Output of MultH: $-d_j$, 0, or d_j;
- Output of MultL: c_j, s_j (carry from the previous slice);
- Previous 2 decimal digits of $A[j-1,k]$.

Note that all signal connections are local except for the transmission of q^H and q^L.

V-Net (Auxiliary residual)

The V-Net computes the auxiliary residual

$$v[j+1] = 10W[j] + x_{j+1} - C[j] \tag{10}$$

with the inputs:

- Shifted residual: $10w[j]$, consisting of a single signed decimal digit (shifted left).
- A dividend digit $x_{j+1} \in \{0,\ldots,9\}$.
- The compensation factor $C[j]$ - redundant (carry,sum) as produced by the [5:2] and [3:2] adders in the C-net. It consists of 2 decimal digits and a sign.
- The output is $v[j+1] = (v_2, v_1, v_0)$.

Since the output precision is 3 decimal digits, this is done fast.

W-Net (Next residual)

This module computes next residual

$$w[j+1] = v[j+1] - q_{j+1}(10 + d_1)$$

The auxiliary residual $v[j+1]$ has 3 decimal digits (leading digit 0 or 1); it is in a nonredundant form. Multiplication $q_{j+1}(10+d_1)$ is performed asas $q_{j+1} \times d_1 + 10q_{j+1}$. The output is the next residual $w[j+1]$, consisting of a single signed decimal digit.

Quotient conversion

The computed signed-digit quotient has to be converted to conventional representation. This can be done with the on-the-fly conversion (OFC) algorithm [4] in parallel with the computation of quotient. To avoid propagation of carries, two forms Q and QM are maintained such that at every iteration the following condition holds: $QM[j] = Q[j] - 10^{-j}$. The two forms are updated as follows

$$Q[j+1] = \begin{cases} (Q[j], q_{j+1}) & \text{if } q_{j+1} \geq 0 \\ (QM[j], (10 - |q_{j+1}|)) & \text{if } q_{j+1} < 0 \end{cases} \tag{11}$$

$$QM[j+1] = \begin{cases} (Q[j], q_{j+1} - 1) & \text{if } q_{j+1} > 0 \\ (QM[j], ((9 - |q_{j+1}|)) & \text{if } q_{j+1} \leq 0 \end{cases} \tag{12}$$

with the initial conditions $Q[0] = QM[0] = 0$ (for a positive quotient). Note that concatenations are used to update the forms. At step j, $Q[j]$ corresponds to the j-digit conventional representation of the quotient. Consequently, after the last iteration, $Q[n+1]$ represents the quotient in the conventional form. One extra iteration is due to the initial right shift of the dividend, needed to avoid the quotient overflow. Note that the OFC is not in the critical path.

IV. FPGA Implementation

The radix-10 divider was designed, implemented, synthesized, and tested using the Xilinx Foundation Design Suite 10.1 tool [11] and mapped onto a Xilinx Virtex 5 xc5vlx50-2ff324 FPGA. The cost was measured in terms of 4-input look-up-tables (LUTs) and delay was measured for individual components, as well as for the entire design in nanoseconds (ns). Optimization of the layout was not performed. As a consequence, the routing delays are very large.

The delay of the routing network produced by the synthesis tool is

$$\begin{aligned} T_{route} &= tr_{V-net} + tr_{Sel} + tr_{W-reg} \\ &= 1.795 + 4.395 + 3.765 + 0.91 \\ &= 10.865ns \end{aligned}$$

The delay through the logic of the synthesized divider is

$$\begin{aligned} T_{logic} &= t_{V-net} + t_{Sel} + t_{C-net} + t_{W-reg} \\ &= 0.471 + 0.688 + 0.688 + 0.396 \\ &= 2.243ns \end{aligned}$$

TABLE III

IMPLEMENTATION CHARACTERISTICS

Module	LUTs $n = 7$	LUTs $n = 14$	Delay* [ns]	Period [ns]
T_{Rec}	8	8	3.904	n/a
Sel	96	96	11.659	n/a
$C - net$	313	684	–	3.835
$V - net$	35	35	7.481	n/a
$W - net$	63	63	13.812	n/a
OFC	62	115	–	1.086
Routing	205	262	–	n/a
Divider	782	1263	–	13.108

* Delay of combinational modules implemented individually.

Since *W-net* is not part of the critical path, we included the individual cost and delay for it. For the actual design, the delay starts from outputing $w[j]$ from the *W-reg* and then inputing it into *V-net*, since the delay goes from one clocked output to the next. Note that *V-net*, *W-net*, Reciprocal and Selection are not clocked. Therefore, the period is determined by the path going from *W-reg* to *C-net*. These are the only two modules that are clocked.

V. SUMMARY

We presented a decimal division scheme which uses modules of short precision – 1 to 3 decimal digits. The algorithm and its implementations are described for specific choices of the initial dividend and divisor precision. The proposed scheme is implemented on the Xilinx Virtex-5 FPGA device and we obtained the following characteristics: for $n = 7$, delay is $(7 + 1) \times 13.108 \approx 105ns$ and the cost is 782 LUTs. For $n = 14$, the implementation has a delay of $(14+1)\times 13.108 \approx 197ns$ and the cost of 1263 LUTs. The proposed scheme uses short multipliers which may have an advantage at the layout level. The delay of routing in the layout produced by the synthesis tool is very large compared to logic delays: we expect that routing delays can be significantly reduced by controlling the layout process. This is the primary goal of future work. Future work includes FPGA implementation of a combined scheme for decimal division and square root. Another extension is developing a floating-point version. We also plan a more comprehensive comparisons with other design approaches.

REFERENCES

[1] F.Y. Busaba, C.Y. Krygowski, W.H. Li, E.M. Schwarz, and S.R. Carlough, "The IBM z900 Decimal Arithmetic Unit", *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, pp. 1335-1339, 2001.

[2] M.D. Ercegovac and T. Lang, Division with Limited Precision Primitive Operations. *Proc. 35th Asilomar Conference on Signals, Systems and Computers*, pp. 841-845, 2001.

[3] M.D. Ercegovac and J.-M. Muller, Digit-recurrence algorithms for division and square root with limited precision primitives. *Proc. 37th Asilomar Conference on Signals, Systems and Computers*, pp. 1440-1444, 2003.

[4] M.D. Ercegovac, T. Lang, *Digital Arithmetic*, San Francisco, Morgan Kaufmann, 2004.

[5] D.E. Atkins, "Higher-radix division using estimates of the divisor and partial remainders." IEEE Trans. Computers, Vol.C-17(10):925-934, October 1968.

[6] M.D. Ercegovac and T. Lang. *Division and Square Root: Digit-Recurrence Algorithms and Implementations.* Norwell, MA: Kluwer Academic Publishers, pps. 230, 1994.

[7] T. Lang and A. Nannarelli, "A Radix-10 Digit-Recurrence Division Unit: Algorithm and Architecture", IEEE Trans. Comput., 56(6):727-739, June 2007.

[8] T. Lang and A. Nannarelli, "Combined Radix-10 and Radix-16 Division Unit", *Proc. 41-st Asilomar Conference on Signals, Systems and Computers*, pp. 967-971, 2007.

[9] H. Nikmehr, B. Phillips, and C.-C. Lim, "Fats Decimal Floating-Point Division", IEEE Trans. VLSI., 14(9):951-961, September 2006.

[10] J.V. Uspensky, *Theory of Equations*, McGraw-Hill, 1948.

[11] Xilinx Corporation. Xilinx ISE Foundation Design Suite 10.1. 2008.