Power Dissipation in Division

Wei Liu and Alberto Nannarelli

DTU Informatics, Technical University of Denmark, Kongens Lyngby, Denmark

***Abstract*— A few classes of algorithms to implement division in hardware have been used over the years: division by digit-recurrence, by reciprocal approximation by iterative methods and by polynomial approximation. Due to the differences in the algorithms, a comparison among their implementation in terms of performance and precision is sometimes hard to make. In this work, we use power dissipation and energy consumption as metrics to compare among those different classes of algorithms. There are no previous works in the literature presenting such a comparison.**

I. Introduction

The quotient q of the division

$$q = \frac{x}{d} + rem$$

can be computed directly or by multiplication of the reciprocal of d and the dividend x

$$q = \frac{1}{d} \times x$$

The digit-recurrence algorithm [1] is a direct method to compute the quotient q. On the other hand, the reciprocal of d can be computed by iterative approximation (Newton-Raphson) or by polynomial approximation [2]. Those algorithms differ in a number of aspects as explained later.

Power dissipation has become a major concern in the design on integrated circuits for its impact on costs (packaging, cooling systems, power bills) and battery lifetime for portable devices.

Division is implemented in hardware in all general purpose CPUs, in most of processors used in embedded systems and it is part of arithmetic co-processors used in advanced hearing aids. Therefore, having low power division is important to lower the costs of multicore chips powering servers, to increase their reliability and to extend the battery lifetime of portable and wearable devices.

In this work, we compare in terms of power dissipation and energy consumption the three main algorithms used to compute division in hardware: division by digit-recurrence and division by approximation of the reciprocal with the Newton-Raphson (NR) method and by quadratic polynomial approximation. We compare both single and double precision division units. For the digit-recurrence division, we also present a low-power version of the algorithm based on the methods of [3].

II. Division by Digit-Recurrence

The radix-r digit-recurrence division algorithm for double-precision significands described in detail in [1], for radix-4, which is a standard implementation of the algorithm, is implemented by the residual recurrence

$$w[j+1] = 4w[j] - q_{j+1}d \qquad j = 0, 1, \ldots, 28$$

with the initial value $w[0] = x$ and with the quotient-digit selection

$$q_{j+1} = SEL(d_\delta, \hat{y}) \qquad q_j = \{-2, -1, 0, 1, 2\}$$

where d_δ is d truncated after the δ-th fractional bit ($\delta = 3$ for radix-4) and the estimated residual, $\hat{y} = 4w_{St} + 4w_{Ct}$, is truncated after t fractional bits ($t = 3$ for radix-4). The residual $w[j]$ is kept in carry-save format to have a shorter cycle time. The divider is completed by a on-the-fly conversion unit, described in [1], which converts the quotient digits q_{j+1} from the signed-digit to the conventional representation, and performs the rounding based on the sign of the remainder computed by the sign-zero detect (SZD) block. The conversion is done as the digits are produced and does not require a carry-propagate adder. The scheme of the unit is depicted in Fig. 1 and the results of its implementation are listed as r4-std in Table I.

A. Low-power implementation

Starting from the scheme implemented in Fig. 1, we applied a number of design techniques [3] to reduce the power dissipation without increasing the cycle time. We consider two main portions: the recurrence and the conversion and rounding (C&R).

1) Retiming the recurrence: Retiming is the circuit transformation that consists in re-positioning the registers in a sequential circuit without modifying its external behavior [4]. By retiming the recurrence we limit the cells on the critical path to the most-significant bits of the recurrence. The idea is to create a slack in the timing paths to replace high speed (HS) gates with slower and

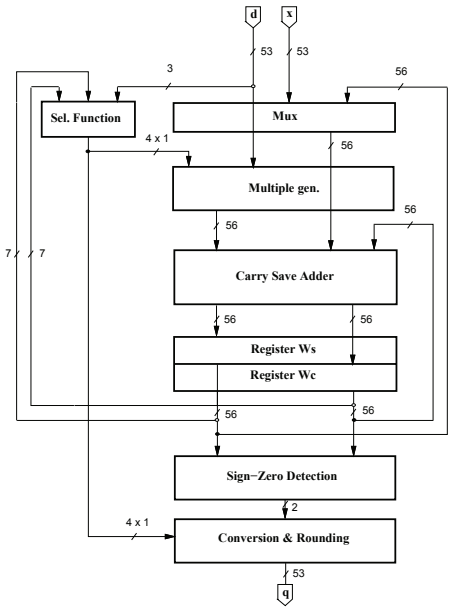

Fig. 1. Implementation of division by digit-recurrence (radix-4 double-precision).

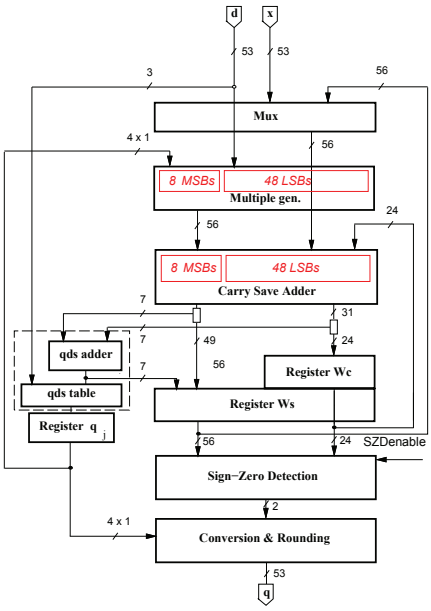

Fig. 2. Low-power implementation of radix-4 division (double-precision).

low power (LL) gates. The retiming is done by moving the selection function from the first part of the cycle to the last part of the previous cycle (cfr. Fig. 1 and Fig. 2). We have to introduce a new register to store the quotient digit, but the register q_j is quite small (4 bits) and it does not compromise the energy saving obtained by retiming. After the retiming, the critical path is limited to the 8 most-significant bits in the recurrence. Since the path through the least-significant bits of the multiple generator and the CSA does not include the selection function, these bits can be redesigned for low-power by changing HS cells into LL cells.

With this modifications, a significant reduction in power dissipation (-30% for dynamic and -70% for static power) is obtained. Fig. 3 shows the change in the HS and LL cell mix before and after the retiming.

2) Changing the Redundant Representation: Since the contribution of flip-flops to both energy dissipation and area is significant, it is useful to change the redundant representation of the residual (w_S and w_C) to reduce the number of flip-flops in the registers. By using a radix-4 carry-save representation with two sum bits and one carry bit for each digit (instead of two), we can reduce the number of flip-flops. With this modification we only need to store one carry bit for each digit.

The change in the redundant representation requires a redesign of the carry-save adder to propagate the carry inside the digit (Fig. 4). The propagation of the carry increases the delay so that this modification cannot be made for those cells (digits of w) that are in the critical

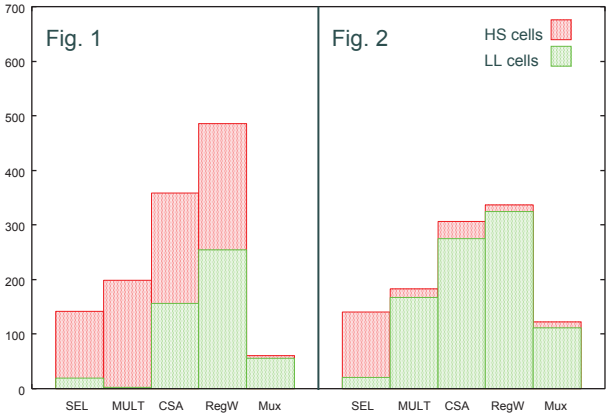

Fig. 3. Cell mix (HS and LL) in Fig. 1 and Fig. 2 recurrence.

path. After the recurrence has been retimed, the critical path is limited to the 8 MSBs and in the remaining 2-bit digits we can use radix-4 CSAs.

3) Disabling the SZD unit: The modification consists in switching-off blocks which are not active during several cycles. This is the case for the sign-zero-detection block (SZD), which is only used in the rounding step to determine the sign of the final remainder and if it is zero. The SZD can be switched off by forcing a constant logic value at its inputs during the recurrence steps.

4) On-the-fly conversion algorithm modification: The on-the-fly convert-and-round (C&R) algorithm [1] performs the conversion from the signed-digit representation to the conventional representation in 2's complement.

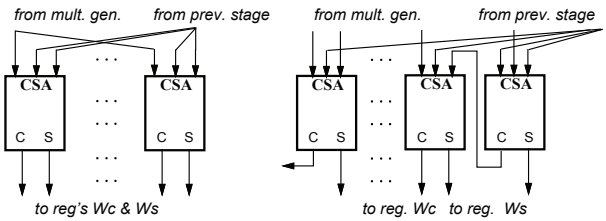

Fig. 4. Replacing CSAs with radix-4 CSAs.

The partial quotient is stored in two registers: Q holding the converted value of the partial quotient, and QM holding Q-1. The registers are updated in each iteration by shift-and-load operations, and the final quotient is chosen between those two registers during the rounding. The large amount of power dissipated in the unit is mainly due to the shifting during each iteration and to the number of flip-flops, used to implement the registers.

The power dissipation is reduced by:

1) We load each digit in its final position. In this way we avoid to shift digits along the registers. To determine the load position we use a 28-bit ring counter C, one bit for each digit to load.
2) We reduce the partial-quotient registers from two to one by eliminating Register QM and by including in Register Q a digit decrementer controlled by the ring-counter C (see [3] for more detail).
3) We switch off the clock signal (clock gating) for the flip-flops that do not have to be updated in a given iteration.

The results of this low power implementation are listed as r4-lp in Table I.

III. Division by Newton-Raphson $1/d$ Approximation

The division $q = x/d$ can also be implemented by the approximation of the reciprocal $R = 1/d$, followed by the multiplication $q = Rx$. By determining $R[0]$ as the first approximation of $1/d$, R can be approximated in m steps by the Newton-Raphson approximation

$$R[j+1] = R[j](2 - R[j]d) \qquad j = 0, 1, \ldots, m$$

Each iteration requires two multiplications and one subtraction. The convergence is quadratic and the number of iterations m needed depends on the initial approximation $R[0]$ (implemented by a look-up table in our case). The values of the initial approximation are reported in [8]. More detail on how to determine $R[0]$ and the approximation accuracy is given in [5].

The implementation of the division by NR is sketched in Fig. 5. The look-up table computing $R[0]$ is a 8-bit input and 7-bit output. Each iteration is implemented

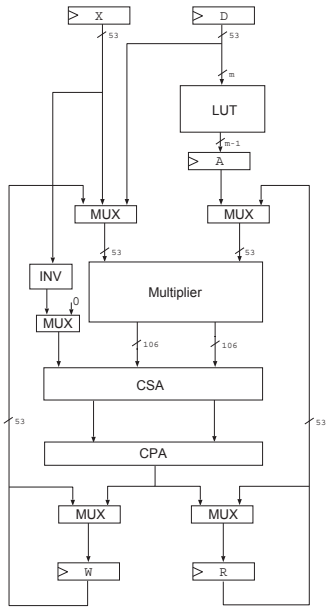

Fig. 5. Implementation of division by Newton-Raphson approximation.

by the multiply-add/subtract datapath of Fig. 5 every two cycles: in the first cycle $2 - R[j]d$ is computed, and in the second cycle $R[j+1]$ is computed. Once $R[m]$ has been computed, the quotient is obtained by an additional multiplication $Q = R[m]x$. For double-precision operands, $m = 3$ and the computation of the Q requires seven cycles to compute $R[3]$ plus another cycle to perform $Q = R[3]x$

To have rounding compliant with IEEE standard, an extra iteration (cycle) is required to compute the remainder and perform the rounding according to the specified mode [2]:

- $rem = Qd - x$
- $q = ROUND(Q, rem, mode)$.

IV. Division by $1/d$ Polynomial Approximation

Alternatively, the reciprocal $1/d$ can be obtained by polynomial approximation. This approximation is normally applied for operations in single-precision (or smaller) as the hardware complexity increases in excess for larger precisions. An example of unit implementing the piecewise quadratic polynomial approximation of $1/d$ is reported in [6].

A look-up table is used to retrieve optimized coefficients and the polynomial is evaluated by a high speed datapath. We followed the method proposed in [7] to generate the coefficients which can result in a smaller table than that of [6]. The function to compute $1/d$ is

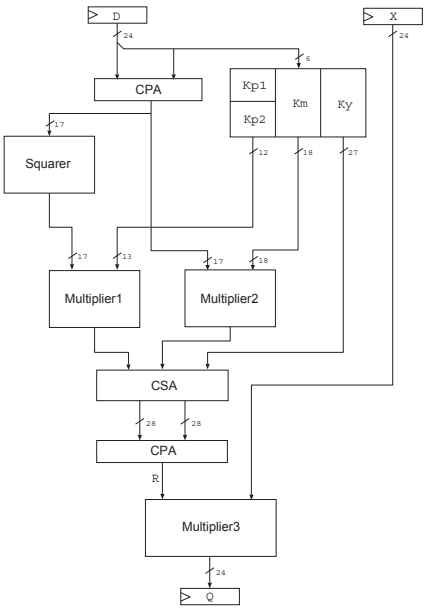

Fig. 6. Implementation of division by polynomial approximation.

approximated as follows:

$$f(d) \approx \begin{cases} K_y + K_m(d-d^*) + K_{p1}(d-d^*)^2 \\ \qquad\qquad \text{for } d < d^* \\ K_y + K_m(d-d^*) + K_{p2}(d-d^*)^2 \\ \qquad\qquad \text{for } d > d^* \end{cases}$$

where d^* is the mid-point in each interval.

The look-up table implements m, K_y, K_m and K_p with precision of 6, 12, 17, and 27 bits, respectively, corresponding to a table size of $64 \cdot (2 \cdot 12 + 17 + 27) = 4288$ bits. The error to approximate $1/d$ is smaller than $*2^{-24}$. The approximation tables with the values of K_y, K_m and K_p are reported in [8].

The approximation unit is depicted in Fig. 6. A squarer is used to compute $(d-d^*)^2$. Then two multipliers are used to implement $K_m(d-d^*)$ and $K_p(d-d^*)^2$. The multipliers recode the second operand into radix-4 digits before generating the partial products. Once the individual terms are ready, they are aligned and a 3-to-2 CSA sum them up. An additional multiplication of x and $1/d$ is required to obtain the quotient.

The results of this implementation are listed as poly in Table I.

V. Energy metrics

Because the algorithms are different and the latency of the operations varies from case to case, it is convenient to have a measure of the energy dissipated to complete an operation. This energy-per-operation is given by

$$E_{op} = \int_{t_{op}} vi \; dt \qquad [J]$$

where t_{op} is the time elapsed to perform the division. Divisions are usually performed in more than one cycle (in n cycles) of clock period T_C and the expression of t_{op} is typically $t_{op} = T_C \times n$. By dividing the energy-per-operation by the number of cycles we obtain the energy-per-cycle

$$E_{pc} = \frac{E_{op}}{n} \qquad [J]. \tag{1}$$

This term is proportional to the average power dissipation that can be expressed in its equivalent forms:

$$P_{ave} = \frac{E_{pc}}{T_C} = E_{pc} f = \frac{E_{op}}{t_{op}} = V_{DD} I_{ave} \quad [W] \tag{2}$$

where V_{DD} is the unit supply voltage and I_{ave} its average current. By combining (2) and t_{op} we obtain

$$E_{op} = P_{ave} \times T_C \times n \qquad [J] \tag{3}$$

The term P_{ave} has an impact on the sizing of the power grid in the chip and on the die temperature gradient, while the term E_{op} impacts the battery lifetime.

VI. Results of experiments

Because the polynomial approximation of Section IV can only be implemented for single-precision operands, we first perform a comparison for double-precision operands and then we compare division algorithms for single-precision operands.

The units are implemented in the STM 90 nm library of standard cells [9] and the power dissipation has been computed by Synopsys Power Analyzer based on the annotated switching activity of random generated vectors.

A. Double-precision division

The upper part of Table I shows the comparison of the units described in Fig. 1 (r4-std), Fig. 2 (r4-lp) and Fig. 5 (NR) for double-precision operands.

The results of the experiments show that the latency of the whole division (including rounding) is about 30% shorter for the Newton-Raphson approach at expenses of area

In terms of energy consumption, Table I shows that unit implementing the digit-recurrence algorithm consume less energy of unit implementing division by NR: the r4-lp consume about one fourth with respect to the NR per division. As for the average power dissipation, the digit-recurrence units are significantly better as well: the power dissipation of r4-lp is about one third than that of the NR unit.

DOUBLE-PRECISION DIVISION

Unit	T_C $[ns]$	n	t_{op} $[ns]$	E_{pc} $[pJ]$	ratio	E_{op} $[pJ]$	ratio	$P_{ave}{}^{(1)}$ $[mW]$	ratio	Area NAND2
NR	2.5	9	22.5	77.6	1.00	698.4	1.00	31.0	1.00	31000
r4-std	1.0	30	30.0	13.4	0.17	402.0	0.56	13.4	0.43	4500
r4-lp	1.0	30	30.0	8.7	0.11	256.9	0.37	8.7	0.28	4300

SINGLE-PRECISION DIVISION

Unit	T_C $[ns]$	n	t_{op} $[ns]$	E_{pc} $[pJ]$	ratio	E_{op} $[pJ]$	ratio	$P_{ave}{}^{(1)}$ $[mW]$	ratio	Area NAND2
NR	2.5	6	15.0	69.2	1.00	415.2	1.00	27.7	1.00	31000
r4-std	1.0	13	13.0	8.6	0.12	112.5	0.27	8.6	0.31	4500
r4-lp	1.0	13	13.0	7.2	0.10	93.6	0.23	7.2	0.26	4300
poly	3.5	1	3.5	88.6	1.28	88.6	0.21	25.3	0.91	25300

(1) P_{ave} is computed at $f_C = \frac{1}{T_C}$.

TABLE I
RESULTS OF EXPERIMENTS FOR DIVISION.

B. Single-precision division

For the digit-recurrence and the NR division, we estimated the power dissipation for single-precision operands by using the datapath for double-precision division. We reduced the operands size from double to single precision, we reduced the number of iterations, and eliminated the steps required for rounding.

Clearly, parts of the datapath keep switching even if the operands have reduced bit size, and therefore, the estimates for NR, r4-std, and r4-lp can be further optimized for single-precision operand.

The values in the lower part of Table I show that poly, the unit implementing division by polynomial approximation, has the shortest latency (1 clock cycle, corresponding to 3.5 ns), but larger area and power dissipation that the digit-recurrence division. Quite surprisingly, the energy-per-division (E_{op}) for the poly implementation is smaller than that of r4-lp ($88.6\ pJ$ vs. $93.6\ pJ$).

VII. CONCLUSIONS AND FUTURE WORK

The results of this survey on different approaches to the implementation of division in hardware show that methods based on the digit-recurrence algorithm gives the lowest power dissipation and energy-per-cycle. The implementation of division by polynomial approximation of the reciprocal has the highest power dissipation, but surprisingly, because of the reduced latency, consumes also the smallest energy for the whole single-precision division.

The method based on the approximation of the reciprocal by Newton-Raphson is the less favorable in terms of area and power/energy consumption. The division by NR has the shortest latency for double-precision, while for single-precision the radix-4 digit-recurrence implementation has a shorter latency than the NR.

The lower energy consumption per operation in the poly approach should be further investigated for very low energy devices, maybe, by trading off some speed for lower power dissipation.

REFERENCES

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

[2] ——, *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.

[3] A. Nannarelli and T. Lang, "Low-Power Divider," *IEEE Transactions on Computers*, pp. 4–17, Jan. 1999.

[4] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," *Proc. of 1993 International Conference on Computer-Aided Design (ICCAD)*, pp. 398–402, Nov. 1993.

[5] D. DasSarma and D. W. Matula, "Measuring the Accuracy of ROM Reciprocal Tables," *IEEE Transactions on Computers*, vol. 43, no. 8, pp. 932–940, Aug. 1994.

[6] S. F. Oberman and M. Y. Siu, "A High-Performance Area-Efficient Multifunction Interpolator," *Proc. of 17th Symposium on Computer Arithmetic*, pp. 273–279, June 2005.

[7] D. De Caro, N. Petra, and A. G. M. Strollo, "A high performance floating-point special function unit using constrained piecewise quadratic approximation," *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS 2008)*, pp. 472–475, May 2008.

[8] W. Liu and A. Nannarelli. Appendix to Power Dissipation in Division. IMM Technical Report 2008-15. [Online]. Available: http://orbit.dtu.dk/All.external?recid=228622

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