A 1.5 GFLOPS Reciprocal Unit for Computer Graphics

Alberto Nannarelli, Morten Sleth Rasmussen and Matthias Bo Stuart

Dept. of Informatics & Math. Modelling, Technical University of Denmark, Kongens Lyngby, Denmark

***Abstract*— The reciprocal operation 1/d is a frequent operation performed in graphics processors (GPUs). In this work, we present the design of a radix-16 reciprocal unit based on the algorithm combining the traditional digit-by-digit algorithm and the approximation of the reciprocal by one Newton-Raphson iteration. We design a fully pipelined single-precision unit to be used in GPUs. The results of the implementation show that the proposed unit can sustain a higher throughput than that of a unit implementing the normal Newton-Raphson approximation, and its area is smaller.**

I. Introduction

Reciprocal is an arithmetic operation used in scientific computations and in computer graphics, and it is usually implemented in hardware in microprocessors.

There are several alternatives for the implementation of the reciprocal (see [1] for details): 1) digit-by-digit algorithms, 2) quadratic convergence algorithms, and 3) polynomial approximations. Methods 2 and 3 converge with less iterations, but require a dedicated parallel multiplier and additional tables. Method 1 is a cost effective alternative with a low hardware overhead.

An algorithm for the computation of the functions $1/d$ and $1/\sqrt{d}$, which consists of a digit-by-digit part followed by a linear approximation, is described in [2]. Moreover, in [2] the radix-4 implementation of a combined $1/d$ and $1/\sqrt{d}$ unit is also presented.

In this work, we apply the algorithm of [2] to a radix-16 $1/d$ unit. With respect to the radix-4 reciprocal unit, the radix-16 converges in a reduced (halved) number of iterations at cost of additional hardware (area).

A single-precision radix-16 $1/d$ unit for use in computer graphics is implemented by unrolling the sequential algorithm and by pipelining the unit to reach a good throughput-area tradeoff. The unit is compared with a different $1/d$ unit based on the Newton-Raphson algorithm.

The results of the implementations show that the proposed $1/d$ unit can sustain a higher throughput and has a reduced area with respect to the unit based on the Newton-Raphson approximation.

II. Algorithm

We now describe the algorithm of [2] for the computation of the reciprocal. The scheme is illustrated in Figure 1. The total delay is reduced by implementing the linear approximation as a digit-recurrence which is performed in an overlapped fashion with the digit-by-digit part. Because the linear approximation has quadratic convergence, roughly half of the iterations are required as compared to a conventional digit-recurrence algorithm.

For the reciprocal computation, the digit-by-digit part of the algorithm produces an approximation k of $1/d$. Then, as described by the Newton-Raphson iteration a better approximation is given by

$$E = k(2 - kd)$$

The relative order of convergence of this iteration is quadratic, that is, if the relative error of k is δ, then the relative error of E is $\epsilon = \delta^2$.

The algorithm of [2] can be summarized as follows:

- Obtain the approximation k using a digit-recurrence algorithm and performing g iterations, roughly half of final required precision.
- Since k is obtained digit-by-digit in most-significant digit first mode, perform the computation of $E = k(2 - d\ k)$ by means of a digit recurrence. In this way, there is a full overlap of the computation of k and the computation of the approximation (see Figure 1).

The radix-r $1/d$ algorithm, described in detail in [2], is implemented by the two recurrences

$$\begin{aligned} w[j+1] &= rw[j] - q_{j+1}d \\ E[j+1] &= r^2E[j] + q_{j+1}(2rw[j] - q_{j+1}d) \\ &\quad \text{with} \qquad j = 0, 1, \ldots, m \end{aligned} \tag{1}$$

The residual $w[j]$ and the approximation $E[j]$ are initialized to $rw[0] = 1.0$ and $E[0] = 0$. The quotient digit is computed at each iteration by the selection function

$$q_{j+2} = SEL(\widehat{rw}[j+1], \hat{d})$$

and the approximation $E[j+1]$ for n-bit operands converges after $g \simeq \frac{\lceil n/\log_2 r \rceil}{2}$ iterations to $1/d$.

For the radix-16 ($r = 16$) implementation, we define the quotient digit $q_j = 4q_H + q_L$, with $q_H = q_L \in [-2, 2]$ and $q_j \in [-10, 10]$. As usually done for radix-16 [3], the selection function is implemented by overlapping two radix-4 stages, as shown in Fig. 2. Moreover, the recurrence $w[j+1]$ is retimed [4]. Summarizing, the recurrences (1) for radix-16 are

$$\begin{aligned} w[j+1] &= 16w[j] - (4q_H + q_L)d \\ E[j+1] &= 256E[j] \\ &\quad +(4q_H + q_L)(32w[j] - (4q_H + q_L)d) \end{aligned} \tag{2}$$

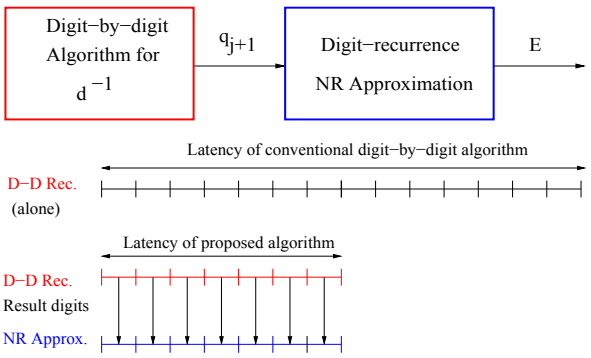

Fig. 1. Algorithm.

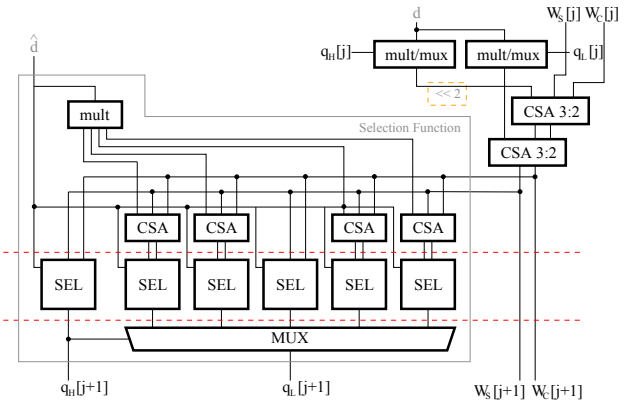

Fig. 2. Implementation of $w[j+1]$ recurrence with detail of the selection function.

III. Approximation Error

In order to evaluate the algorithm for radix-16 and the performance of the resulting unit, we compare the approximation error obtained by (2) with that obtained from the Newton-Raphson (NR) approximation of $1/d$

$$R[j+1] = R[j](2 - dR[j]) \qquad j = 0, 1 \ldots \tag{3}$$

where $R[0]$ is the initial approximation of $1/d$ [1].

We performed an exhaustive simulation for single-precision (2^{23} values of d) and computed the error for both the algorithm of (2) and the NR approximation of (3).

To ensure the convergence of the digit-recurrence algorithm, d is pre-shifted a few positions to the right to obtain $n = 26$ bits. Consequently, for the algorithm of (2)

$$\frac{\lceil n/\log_2 r \rceil}{2} = \frac{\lceil 26/\log_2 16 \rceil}{2} = 3.25 < 4$$

4 iterations are needed.

For the NR algorithm, by using the initial approximation $R[0]$ described in [5], 2 iterations are needed for single-precision $1/d$.

The approximation errors resulting from the simulation of the two alternatives are reported in Table I as the difference between the correctly rounded single-precision result and the obtained approximation. Both round-to-zero (truncation) and round-to-the-nearest rounding modes were simulated. For both algorithms the maximum difference is -2 *unit in last position (ulp)*, and for the Newton-Raphson method the error is smaller. However, for application in which a precision of 2^{-21} or smaller is required, such as computer graphics, the two methods can be considered as equivalent.

IV. Architecture

The first step in the design of the reciprocal unit is to find a suitable arrangement of the adders and multipliers in the two recurrences. Because of the chosen digit set for q_H and q_L, multipliers are implemented as multiplexers. Moreover, we want to keep the residual and the approximation in a redundant (carry-save) format to speed up the iteration time.

The approximation recurrence, E in (2), is rewritten in carry-save representation as

$$\begin{aligned} E_S[j+1] + E_C[j+1] = 256(E_S[j] + E_C[j]) + \\ +(4q_H + q_L)\{32(w_S[j] + w_C[j]) - (4q_H + q_L)d\} \end{aligned} \tag{4}$$

By expanding the term in { } of (4), and defining an intermediate value A

$$\begin{aligned} A_S + A_C &= 32(w_S[j] + w_C[j]) - (4q_H + q_L)d \\ &= 32w_S[j] + 32w_C[j] - 4q_H d - q_L d \end{aligned}$$

we obtain

$$\begin{aligned} E_S[j+1] + E_C[j+1] = 256E_S[j] + 256E_C[j] \\ +(4q_H + q_L)(A_S + A_C) \end{aligned}$$

which expands to

$$\begin{aligned} E_S[j+1] + E_C[j+1] &= 256E_S[j] + 256E_C[j] \\ &+ 4q_H A_S + 4q_H A_C \\ &+ q_L A_S + q_L A_C \end{aligned}$$

The corresponding architecture is depicted in Fig. 3.

First, we consider a sequential implementation of the algorithm. The sequential reciprocal unit is obtained directly by the scheme of Fig. 3 and by placing registers at the bottom of the figure to latch q_{j+1}, $w[j+1]$ and $E[j+1]$. If the number of bits required in the digit-recurrence is n, the number of bits required for $E[j+1]$ is $p \simeq 2n$.

The unit is completed either by an on-the-fly convert-and-round unit, or by a carry-propagate adder that from the carry-save representation of E computes the reciprocal $1/d = E_S[g] + E_C[g]$.

As a second step, we design the unit for high-throughput computer graphics applications. This is a single-precision fully pipelined unit based on the approximation error simulations of Table I. Consequently, the unit is implemented by unfolding 4 iterations of the sequential algorithm (4 stages of Fig. 3 as shown in Fig. 4) and by adjusting the bit-width of the datapath to minimize the area.

The unit is pipelined to achieve a reasonable tradeoff throughput/area. For this reason, each of the stages of Fig. 3

Difference	Algorithm of (2) round-to-zero		Algorithm of (2) round-to-nearest		Newton-Raphson round-to-zero		Newton-Raphson round-to-nearest	
-2 ulp	1508455	18.0%	42961	0.5%	65632	0.8%	5529	0.1%
-1 ulp	6523113	77.8%	5258594	62.7%	4731276	56.4%	657627	7.8%
0	357040	4.3%	3087044	36.8%	3591700	42.8%	7725452	92.1%
+1 ulp	0	0.0%	9	0.0%	0	0.0%	0	0.0%

TABLE I

ERRORS FOR THE TWO APPROXIMATIONS IN ROUND-TO-ZERO AND ROUND-TO-NEAREST MODES.

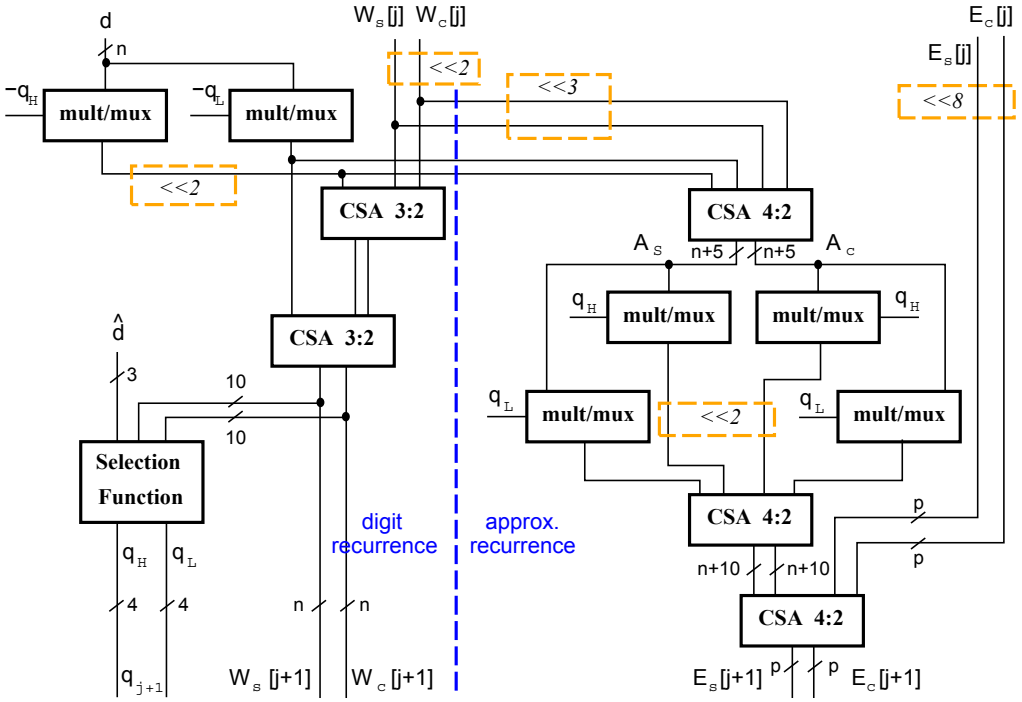

Fig. 3. Architecture of radix-16 reciprocal.

is broken down into two pipeline sub-stages as explained in Section V.

V. IMPLEMENTATION AND COMPARISONS

The implementation of the $1/d$ unit (single-precision unfolded and pipelined) is compared in terms of throughput, latency and area with a unit implementing the Newton-Raphson approximation. The two units have been synthesized using the STM 90 nm CMOS standard cells library [6] and Synopsys Design Compiler.

A. Digit-Recurrence and Approximation (DRA) unit

To achieve a reasonable tradeoff throughput/area in the digit-recurrence and approximation (DRA) unit, we first determined the critical path of the sequential unit, obtained by the direct implementation of Fig. 3. The critical path is through the digit-recurrence part:

mult	CSA 3:2	CSA 3:2	SEL_{r16} q_{j+1}	reg. (q_L)
0.10	0.05	0.09	0.62	0.12 = 0.98 ns

and the minimum clock period T_C, including set-up time, is about 1.1 ns. To increase the clock frequency, we need to introduce pipeline latches to reduce the critical path.

The radix-16 selection function (boxed area in Fig. 2) consists of 6 radix-4 selection functions (indicated as SEL in Fig. 2). The quotient digit $q_{j+1} = 4q_H + q_L$ is computed speculatively by selecting q_L among the 5 possible combinations of

$$q_L = SEL_{r4}\left(\widehat{4w[j]} - \widehat{4q_H d},\ \hat{d}\right) \qquad \text{for } q_H \in [-2, 2]$$

once q_H is known.

To avoid placing latches into the radix-4 selection functions, we have chosen a clock period such that $T_C \simeq t_{SEL} + t_{reg}$, where t_{SEL} is the maximum delay through the slowest of the radix-4 selection functions of Fig. 2 and t_{reg} is the sum of the propagation delay through the pipeline latches and the set-up time. The resulting clock period is $T_C = 0.66\ ns$ and the pipeline register placement is shown with the horizontal dotted lines in Fig. 5, and in Fig. 2 for the radix-16 selection function.

In the last pipeline stage the result is rounded by performing a carry-propagate addition: $1/d = E_S[4] + E_C[4] + \frac{1}{2}ulp$.

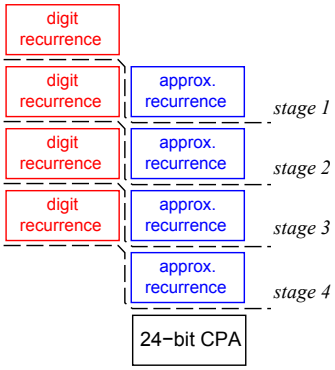

Fig. 4. Unit obtained by unrolling four iterations.

B. Newton-Raphson (NR) unit

We proceeded in a similar manner to design the Newton-Raphson (NR) unit.

First, we designed a sequential unit implementing the algorithm of (3)

$$R[j+1] = R[j](2 - dR[j])$$

with the initial approximation $R[0]$ implemented by a look-up table with values derived from [5]. To obtain the approximation error of Table I, two iterations of (3) are necessary. Each iteration requires two multiplications: $dR[j]$ and, by defining $T[j] = (2 - dR[j])$, $R[j]T[j]$. Because $dR[j] \simeq 1$, $T[j] = 2 - dR[j]$ is simply obtained by the complement of $dR[j]$. Therefore, the critical path of a sequential implementation of (3) is given by the following contributions:

$$t_{R[0]} + t_{mult} + t_{compl} + t_{mult}$$

where:

$t_{R[0]}$ is the delay of the approximation table. Its value is $0.19\ ns$ in our implementation.

t_{mult} is the delay of a 24×24 multiplier implemented as a radix-4 recoded tree multiplier. The delay to obtain the product in carry-save format is $t_{mult-CS} = 0.73\ ns$ plus an additional $0.30\ ns$ for the carry-propagate addition.

t_{compl} is the delay for the complementation, that can be done by inverting the bits of $T[j]$ and adding the '1' later in the multiplier's tree.

For the pipelined NR unit, we decided to place pipeline registers across the tree multipliers latching the carry-save product. In this way, the minimum clock cycle is

$$T_{min} \geq t_{mult-CS} + t_{reg} + t_{set-up} \simeq 0.90\ ns$$

By unfolding the two sequential iterations of the NR algorithm and pipelining the unit we obtain the scheme sketched in Fig. 6. It consists of 9 stages.

By synthetizing the unit with the register placement of Fig. 6, it results $T_C = 0.93\ ns$.

C. Comparison

The results of the implementations in the 90 nm library of standard cells for the DRA and NR units are shown in Table II.

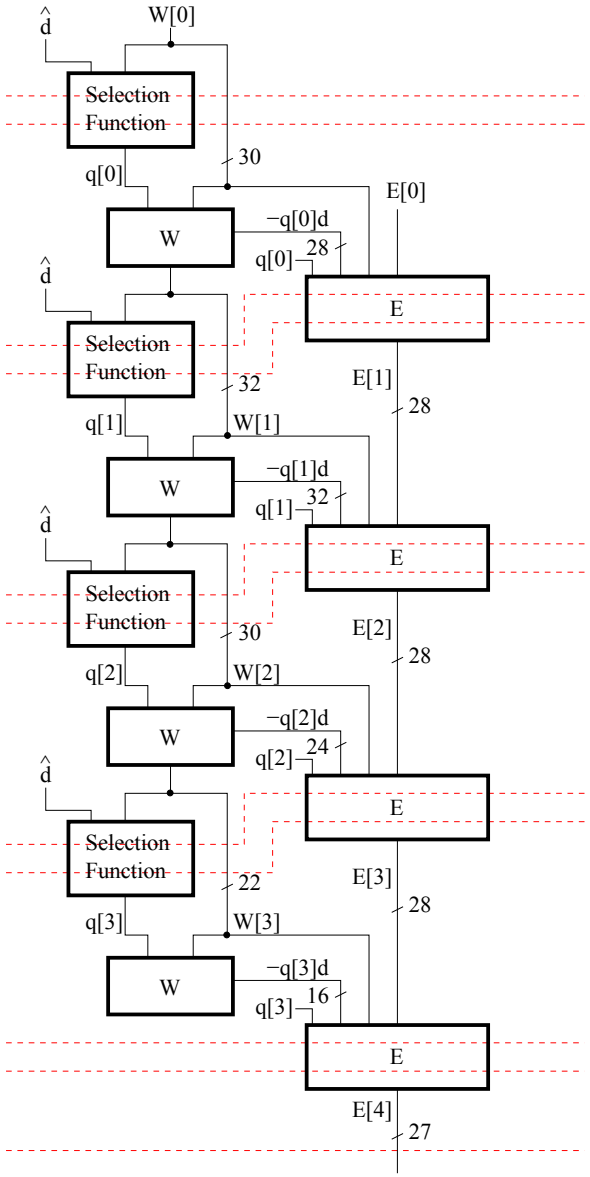

Fig. 5. Single-precision pipelined $1/d$ unit (overview).

The results show that the DRA unit has a shorter latency and can be clocked at a higher rate (throughput) than the NR unit. Furthermore the NR unit is more than twice as larger. On the other hand, the precision achieved with the Newton-Raphson algorithm is higher (Table I).

VI. Conclusions

In this work, we have designed a single-precision reciprocal unit suitable for computer graphics. The unit is based on the algorithm presented in [2] in which the digit-recurrence division algorithm is overlapped with the approximation of one Newton-Raphson iteration. Here, the algorithm has been applied to radix-16. As a consequence, the reciprocal unit computes $1/d$ with an maximum error of 2 ulp, a precision

stage	operation	block
1	$R[0]$	table
2	$T[0] = 2 - dR[0]$	mult-CS
3		CPA + compl.
4	$R[1] = R[0]T[0]$	mult-CS
5		CPA
6	$T[1] = 2 - dR[1]$	mult-CS
7		CPA + compl.
8	$R[2] = R[1]T[1]$	mult-CS
9		CPA (incl. round)

Fig. 6. Pipeline stages in NR unit.

considered sufficient for computer graphics, in 4 iterations. The unit is pipelined and clocked at 1.5 GHz to reach a throughput of 1.5 giga reciprocals per second.

The radix-16 digit-recurrence and approximation (DRA) unit is compared with one implementing $1/d$ based on the traditional Newton-Raphson approximation. The DRA unit can sustain a higher throughput and its area is significantly smaller than the NR unit.

	T_C	latency		Area	
unit	$[ns]$	cycles	$[ns]$	$[mm^2]$	ND2 eqiv.
DRA unit	0.66	11	7.26	0.081	18K
NR unit	0.93	9	8.37	0.192	43K

TABLE II
RESULTS OF IMPLEMENTATIONS.

ACKNOWLEDGMENTS

We thank Tomás Lang for the insightful discussion and comments.

REFERENCES

[1] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.
[2] E. Antelo, T. Lang, P. Montuschi, and A. Nannarelli, "Low latency digit-recurrence reciprocal and square-root reciprocal algorithm and architecture," in *Proc. of 17th Symposium on Computer Arithmetic*. IEEE, June 2005, pp. 147–152.
[3] M. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic Publisher, 1994.
[4] A. Nannarelli and T. Lang, "Low-power division: Comparison among implementations of radix 4, 8 and 16," *Proc. of 14th Symposium on Computer Arithmetic*, pp. 60–67, 1999.
[5] D. Das Sarma and D. W. Matula, "Measuring the accuracy of ROM reciprocal tables," *IEEE Transactions on Computers*, vol. 43, pp. 932–940, Aug. 1994.
[6] STMicroelectronics. 90nm CMOS090 Design Platform. [Online]. Available: http://www.st.com/stonline/prodpres/dedicate/soc/asic/90plat.htm