VLSI DESIGN
2001, Vol. 12, No. 3, pp. 365–376
Reprints available directly from the publisher
Photocopying permitted by license only

A Novel Low-power Shared Division and Square-root Architecture Using the GST Algorithm*

MARTIN KUHLMANN† and KESHAB K. PARHI‡

Broadcom Corporation, Irvine, CA 92619, USA

(Received 20 June 2000; In final form 3 August 2000)

Although SRT division and square-root approaches and GST division approach have been known for long time, square-root architectures based on the GST approach have not been proposed so far which do not require a final division/multiplication of the scale factor. A GST square-root architecture is developed without requiring either a multiplication to update the scaled square-root quotient in each iteration or a division/multiplication by the scaling factor after completing the square-root iterations. Additionally, quantitative comparison of speed and power consumption of GST and SRT division/square-root units are presented. Shared divider and square-root units are designed based on the SRT and the GST approaches, in minimally and maximally redundant radix-4 representations. Simulations demonstrate that the worst-case overall latency of the minimally-redundant GST architecture is 35% smaller compared to the SRT. Alternatively, for a fixed latency, the minimally-redundant GST architecture based division and square-root operations consume 32% and 28% less power, respectively, compared to the maximally-redundant SRT approach.

Keywords: Square-root; Division; Scaling; Low-power architecture; SRT; GST

1. INTRODUCTION

Among the four basic arithmetic functions, division is the most difficult algorithm to be implemented in silicon. Assuming the simplest implementation for adder, multiplier and divider, the addition using a carry-ripple adder requires a critical path of n full-adder and the multiplication employing a carry-save scheme and a final carry-ripple adder requires $2 \cdot n$ full-adder delays (the multiplexer delays are neglected). However, the division using the paper and pencil method, in which a carry-ripple adder is used in each iteration, requires a critical path of $n \cdot n$ full-adder

* This work was carried at the University of Minnesota and supported by the Defense Advanced Research Projects Agency under contract number DA/DABT63-96-C-0050.
† Corresponding author. e-mail: kuhlmann@broadcom.com
‡ Present address: Department of Electrical and Computer Engineering of the University of Minnesota, Minneapolis, MN 55455. e-mail: parhi@ece.umn.edu

delays. Hence, it is obvious that there is the demand of obtaining fast, area and power efficient divider algorithms and architectures. The computation of the square-root can be based on the division, where most of the hardware can be shared among the two operations. Division and Square-root operations are required in pocket calculators as basic arithmetic operations. Hence, division and SQRT have to be included in every micro-processor, mostly implemented in the form of a co-processor. Nevertheless, due to the introduction of smaller technologies and the trend of implementing an entire systems on a single chip (SOC), division and square-root operations are nowadays implemented on the main processor. Bose *et al.* [1] indicate that the total number of division operations can range from one-third to one-half the number of multiplications in computations. Oberman [2] concludes that even though the division is an infrequent operation, it can result in performance degradation if the implementation is being ignored.

In general, division and square-root are defined as

$$N = Q_d \cdot D + R, \tag{1}$$

$$X = Q_s \cdot Q_s + R, \tag{2}$$

where N, D, Q_d, R, X and Q_s are dividend, divisor, division quotient, remainder, square-root operand and square-root quotient, respectively. Generally divider algorithms can be separated into two different kinds of algorithms, the Multiplicative Algorithms (MA) [3–5] and the Iterative Digit Recurrence Algorithms (IDRA), which is also known as the paper-and-pencil method. A general overview of existing divider algorithm is given in Figure 1.

The IDRA performs the division by repeated subtractions

$$r_{i+1} = r \cdot R_i - q_i \cdot D, \tag{3}$$

while the MA employs repeated multiplications to obtain the final quotient

$$Q_{\text{division}} = \frac{N}{D} = \frac{N \cdot R_0 \cdot R_1 \cdots R_{m-1}}{D \cdot R_0 \cdot R_1 \cdots R_{m-1}} \to \frac{Q}{1}. \tag{4}$$

The Newton–Raphson and the Goldschmidt algorithm are the most popular among the multiplicative algorithms. Among the IDRA, there are the two main algorithms, the SRT (due to the accomplishments of Sweeney, Robertson and Tocher [6, 7]) and the GST (the g represents *generalized*, ST due to the accomplishments of Svoboda and Tung [8, 9]). The case of higher radix division with redundant digit sets which also implies similar considerations for the corresponding higher radix square-root, has been extensively studied in [10]. Other significant work on division and division/square-root can be found in [11–17].

The number of iterations of the Multiplicative Algorithm is proportional to $\log_2$(word-length). However, since every iteration doesn't consist of simple addition/subtraction operations, but of two multiplications and one addition, very fast multipliers are needed to obtain the same efficiency for small and medium word-lengths as the IDRA. The large area requirement of this approach is a major drawback.

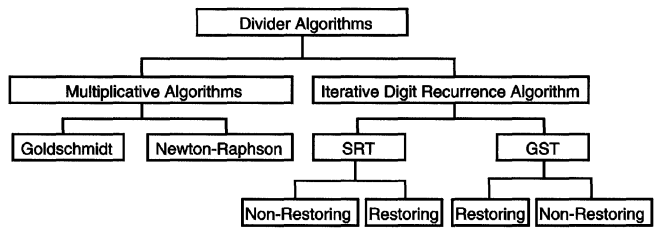

FIGURE 1 An overview of different divider algorithms.

According to IEEE standard 754, the unsigned operands of the division and the square-root have to be in the range [1, 2) and [0.25, 1), respectively. In case that the inputs do not correspond to this standard, they must be normalized and the their exponents be adjusted. Assuming binary operands of word-length n, it becomes obvious that the division/square-root computation requires n additions/subtractions to obtain the entire quotient. The computation can be accelerated by either reducing the number of iteration (using higher radices) or by reducing the critical path of an iteration. The SRT and GST algorithms take advantage of a redundant number system and a higher radix r. This leads to a word-length independent critical path of the addition and subtraction and reduces the number of iterations to $n/(\log_2 r)$. In [18] a novel shared architecture for division and square-root was presented, which developed a GST square-root architecture without requiring an additional division by the scaling factor after the square-root operation as in [19].

This paper is organized as follows. Section 2 presents the mathematical background for GST division and GST square-root operations. Section 3 presents the architecture of the GST divider/square-root unit and the estimated power consumption of the SRT and GST square-root architectures. Section 4 presents the simulation results while Section 5 concludes the paper.

2. THE GST DIVISION AND SQUARE-ROOT

2.1. Mathematical Background

The quotient digit selection is a 2-dimensional function depending in every iteration i on the partial remainder R_i and the denominator D for the division and the partial remainder R_i and the square-root quotient $Q_{s,i}$ for the square-root computation. By employing the recurrence equation for the division and square-root, the new remainder R_{i+1} can be computed according to

$$R_{i+1} = r \cdot R_i - q_i \cdot D \tag{5}$$

and

$$\begin{aligned} R_{i+1} &= r \cdot R_i - q_i \cdot (2 \cdot Q_{i-1} + r^{-i} \cdot q_i) \\ &= r \cdot R_i - q_i \cdot 2Q_i^1, \end{aligned} \tag{6}$$

where q_i is the newly computed square-root quotient digit of iteration i and Q_{i-1} the square-root quotient from the previous iteration. Note that these recurrence equations are also used for the SRT division. The decision criteria can be simplified by restricting the range of D and $2Q_i^1$ to a certain range $[1, 1+\delta)$, in which the decision function is independent of D or $2Q_i^1$. This can be performed by a multiplication. Pre-scaling of both operands does not alter the sought quotient, since:

$$Q_{\text{division}} = \frac{N}{D} = \frac{N \cdot k}{D \cdot k} \tag{7}$$

and

$$Q_{\text{square-root}} = \frac{X}{Q_s} = \frac{X \cdot k}{Q_s \cdot k} = \frac{X'}{Q_s'}. \tag{8}$$

Before the first iteration can be performed, the scaling of the operands is required. Additionally, the arithmetic condition which guarantees that the most significant digit equals zero after the subtraction/addition of the multiple of the denominator, has to be satisfied. Additionally, since the square-root quotient is unknown at the beginning of the computation, it has to be updated and scaled according to the quotient digits q_i and k.

Since Q_s changes every iteration, kQ_s has to be updated according to the quotient digits q_i and k. The sought result is obtained after every iteration by:

$$Q_{s,i} = Q_{s,i-1} + r^{-i} \cdot q_i. \tag{9}$$

Scaling Eq. (6) results in:

$$\begin{aligned} R_{i+1} &= r \cdot R_i - q_i \cdot (2 \cdot Q_{s,i-1} + r^{-i} \cdot q_i) \cdot k \\ &= r \cdot R_i - q_i \cdot 2Q_{s,i}^{1'}. \end{aligned} \tag{10}$$

The goal of this section is to obtain an equation of $Q^{1'}_{s,i+1}$ in terms of $Q^{1'}_{s,i}$. Therefore, increasing i by 1 in (10) leads to

$$\begin{aligned} R_{i+2} &= r \cdot R_{i+1} - q_{i+1} \cdot (2 \cdot Q_{s,i} + r^{-(i+1)} \cdot q_{i+1}) \cdot k \\ &= r \cdot R_{i+1} - q_{i+1} \cdot 2Q^{1}_{s,i+1} \cdot k. \end{aligned} \tag{11}$$

Recalling that in (6) $2Q^{1}_{s,i}$ was defined as

$$2Q^{1}_{s,i} = 2 \cdot Q_{s,i-1} + q_i \cdot r^{-i}, \tag{12}$$

and by comparing Eqs. (10), (11) and (12), the sought equation is obtained as:

$$\begin{aligned} 2Q^{1'}_{s,i+1} &= (2 \cdot Q_i + q_{i+1} \cdot r^{-(i+1)}) \cdot k \\ &= (2 \cdot Q_{s,i-1} + 2q_i \cdot r^{-i} + q_{i+1} \cdot r^{-(i+1)}) \cdot k \\ &= 2 \cdot Q^{1'}_{s,i} + q_i \cdot r^{-i} \cdot k + q_{i+1} \cdot r^{-(i+1)} \cdot k. \end{aligned} \tag{13}$$

The next scaled square-root quotient can now be obtained by just adding the quantities $q_i \cdot r^{-i} \cdot k$ and $q_{i+1} \cdot r^{-(i+1)} \cdot k$ to the previous scaled square-root quotient [18].

2.2. The Arithmetic Condition and the Range of the Radicand

Let us assume the same symmetric redundant digit set $D_{\langle r,\alpha \rangle}$ as used for the division [20]. The maximum and the minimum valid remainders are given by

$$R_i^{\max} = 0.\alpha\alpha\alpha \ldots \alpha \tag{14}$$

$$R_i^{\min} = 0.\bar{\alpha}\bar{\alpha}\bar{\alpha} \ldots \bar{\alpha}, \tag{15}$$

which can be rewritten as

$$-\frac{\alpha}{r-1} < R_{i+1} < \frac{\alpha}{r-1}. \tag{16}$$

By employing the arithmetic condition in the square-root recurrence equation

$$R_{i+1} = r \cdot R_i - q_i \cdot (2Q_i + q_i \cdot r^{-i}) \tag{17}$$

an upper and lower bound can be computed in which the square-root quotient has to be located. By obtaining this upper and lower bounds for the square-root quotient, the limitations for the operand X can be computed.

By introducing the same rewrite condition as in the division [20], the square-root quotient has to be scaled in the range of

$$2Q_s < \frac{1}{\alpha}\left(\frac{\alpha}{r-1} + \alpha - \frac{\beta}{r} - \frac{\alpha}{(r-1)r}\right) = 1 + \frac{\alpha - \beta}{r \cdot \alpha} \tag{18}$$

and

$$2Q_s > \frac{1}{\alpha}\left(\frac{\alpha}{r-1} + \alpha - \frac{\alpha}{r-1}\right) = 1. \tag{19}$$

This leads to

$$1 \le 2Q_s < 1 + \frac{\alpha - \beta}{r \cdot \alpha}. \tag{20}$$

This corresponds to the same range as for the division. By obtaining the upper and lower bounds for the square-root quotient Q_s, the corresponding bounds for the operand X can be computed according to

$$X_{l,u} = 0.25 \cdot (2Q_{l,u})^2, \tag{21}$$

where X_l, X_u, $2Q_l$ and $2Q_u$ represent the lower and upper bound of the operand X and the square-root quotient, respectively.

An even distribution of the scale intervals does not lead to an optimal solution as shown in Table I.

Besides the fact that eight bits of the square-root operand X have to be examined, the shown bounds of X lead to a longer critical path in the update of the square-root quotient since the word-length of k corresponds to seven bits. To minimize the number of bits to be examined for the prediction of the scale factor k and to limit the scale factor to a multiple of 1/16, the upper and lower bounds of X have to be slightly altered (see Tab. II). This also leads to a useful symmetry which can be used for simplifying the scale factor selection.

TABLE I Square-root K-selection table for the digit set $DS_{\langle 4,2\rangle}$

Range of 2Q	Range of 2Q	K	$T_1+T_2+T_3$
$1 \le Q_s \le 9/8$	$1/4 \le X \le 81/256$	1	$1-1/8+1/8$
$9/8 \le Q_s \le 10/8$	$81/256 \le X \le 25/64$	0.890625	$1-1/8+1/64$
$10/8 \le Q_s \le 11/8$	$25/64 \le X \le 121/256$	0.8125	$1-1/8-1/16$
$11/8 \le Q_s \le 12/8$	$121/256 \le X \le 9/16$	0.75	$1-1/8-1/8$
$12/8 \le Q_s \le 13/8$	$9/16 \le X \le 169/256$	0.6875	$1/2+1/8+1/16$
$13/8 \le Q_s \le 14/8$	$169/256 \le X \le 49/64$	0.625	$1/2+1/8+1/128$
$14/8 \le Q_s \le 15/8$	$49/64 \le X \le 225/256$	0.59375	$1/2+1/8-1/32$
$15/8 \le Q_s \le 16/8$	$225/256 \le X \le 1$	0.5625	$1/2+1/8-1/16$

TABLE II Optimized square-root K-selection table for the digit set $DS_{\langle 4,2\rangle}$

Range of X	K	$T_1+T_2+T_3$
$1/4 \le X \le 5/16$	1	$1-1/8+1/8$
$5/16 \le X \le 11/32$	0.9375	$1-1/8+1/16$
$11/32 \le X \le 13/32$	0.875	$1-1/8-0$
$13/32 \le X \le 15/32$	0.8125	$1-1/8-1/16$
$15/32 \le X \le 9/16$	0.75	$1/2+1/8+1/8$
$9/16 \le X \le 21/32$	0.6875	$1/2+1/8+1/16$
$21/32 \le X \le 51/32$	0.625	$1/2+1/8-0$
$51/64 \le X \le 1$	0.5625	$1/2+1/8-1/16$

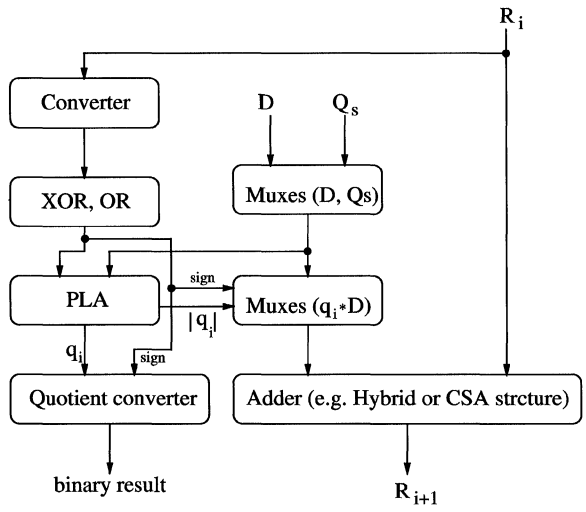

FIGURE 2 Block diagram of the SRT minimally redundant radix-4.

3. THE ARCHITECTURE OF THE SRT AND GST DIVISION AND SQUARE-ROOT

3.1. Minimally Redundant Radix-4 SRT

In [21] a minimally redundant radix-4 architecture for a shared division and square-root implementation was introduced (see Fig. 2).

Every iteration requires a decision criteria, multiplexers which select a multiple of the denominator D or the square-root quotient Q_i, a redundant adder and a part of the on-line converter. As already mentioned entirely 10 bits of the remainder and the denominator/square-root quotient have to be examined to predict the next quotient digit. Depending on the choice of adder, the redundant remainder has to be converted to a two's complement number by employing a fast seven bit adder or an on-line converter for four digits. In order to reduce the costs for the decision criteria, this 2's compliment representation is converted into a sign-magnitude representation. This guarantees a smaller implementation of the decision criteria. Thus, the decision criteria only distinguishes between the quotient digit $q_i=0$, $q_i=1$ and $q_i=2$. The MSB of the remainder determines if the multiple of the denominator $q_i \cdot D$ is added or subtracted to/from the previous remainder. Nevertheless, the combinational logic to implement the decision criteria is quite costly even though, it only has to differentiate between three quotient digits. Most efficiently, the decision criteria of $q_i=0$ is implemented using random logic while the distinction of the quotient digit $q_i=1$ is rather implemented using a Programmable Logic Array (PLA). The addition is either

performed by carry-save adders or by a Hybrid-adder. The on-line converter has to simultaneously operate in two different modes. The first one computes the correct square-root quotient Q_i while the second one updates the previous square-root quotient Q_{i-1} by half of the new quotient digit. The architecture presented in [22] and [23] use a radix-8 and radix-2 digit set, respectively, and are not compared to the radix-4 implementations due to the different number of iterations. In [17], an architecture is presented which does not require an initial PLA. However, after the initial iteration, the design still uses a PLA to predict the new quotient digit.

3.2. Maximally Redundant Radix-4 SRT

As already mentioned, the digit set $DS_{\langle 4,3\rangle}$ has larger overlapping regions and hence, requires less bits to be examined to predict the new quotient digit. The drawback of the triple multiple of the denominator can be resolved by employing two stages of CSA adders, the first one adds a multiple from the digit set $q_{i,1} \in \{\bar{2}, 0, 2\}$, the second one adds a multiple from the digit set $q_{i,2} \in \{\bar{1}, 0, 1\}$. By adding $q_{i,1}$ and $q_{i,2}$, the quotient digit $q_i = q_{i,1} + q_{i,2}$ is obtained which is performed in *DD*. The corresponding circuitry for one iteration is shown in Figure 3 was introduced in [24].

The quotient selection function is divided into two parts *LS* and *NS*, the first one predicts the digit $q_{i,1}$ the latter one $q_{i,2}$. The on-line converter (*OC*) converts the redundant quotient digit into a binary format. Like in the architecture of the minimally redundant radix-4 architecture, the redundant remainder is converted into a binary representation by a fast binary adder (*MXA7*) and then converted into a sign-magnitude representation (*EOL*) to guarantee a smaller implementation of the decision criteria. The on-line converter has to simultaneously operate in two different modes. The first one computes the correct square-root quotient Q_i while the second one updates the previous square-root quotient $Q_{s,i-1}$ by half of the new quotient digit.

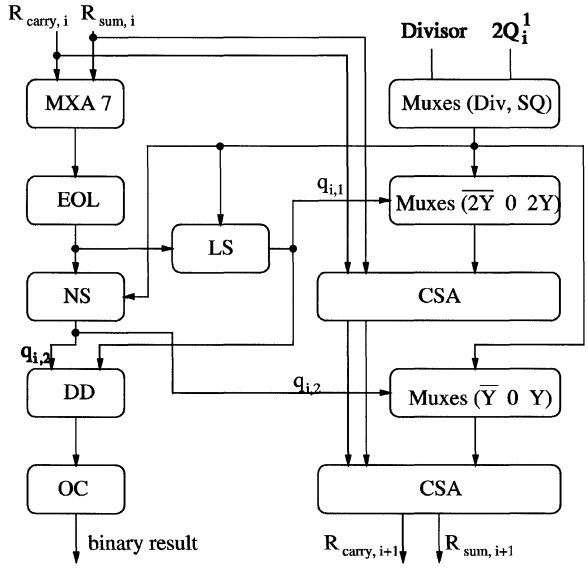

FIGURE 3 Block diagram of the SRT maximally redundant radix-4.

3.3. Minimally Redundant Radix-4 GST

In [25] and [20] two minimally redundant radix-4 division architectures have been presented. The architecture has been expanded according to (13). The first remainder R_0 can be obtained by the summation of the three partial sums by a row of Carry-Save adders and an additional binary tree adder. The final scaled numerator can be -without any hardware costs- converted into a redundant representation. Alternatively, T_1 and T_2, which own a smaller critical path than T_3, can be added by a Hybrid-adder resulting in $S_{\text{red},1}$. Hence, either T_1 or T_2 have to be converted into an redundant representation. This is done without any hardware costs. T_3 is added to the $S_{\text{red},1}$ by another Hybrid-adder.

In the first iteration, the scaled denominator $k \cdot D$ and the scaled square-root operand $k \cdot X$ are not yet available. However, in case of division, the first quotient digit is restricted to either $q_0 = 1$ or $q_0 = 2$. This is caused by the restricted range of the operands. Furthermore, $q_0 = 1$ covers the range from $Q_s \in [1/2,\ 5/3)$. Thus, $q_0 = 2$ is only selected, if the numerator is close to 2 and the denominator is

close to 1. Hence, by checking the second to forth significant bits of numerator and denominator, the most significant quotient digit can be predicted. To obtain $q_0 = 2$, the numerator has to be larger than $N \geq 1.875$ which corresponds to $n_1 = n_2 = n_3 = 1$ and the denominator has to be smaller than $D < 1.125$ which corresponds to $d_1 = d_2 = d_3 = 0$. The output of the first decision criteria selects the multiple of the two partial sums $D_{\text{scaled,carry}}$ and $D_{\text{scaled,sum}}$ that is subtracted from the first remainder R_0. The required modified Hybrid-adder consists of a critical part of 2.5 full-adder and does not lengthen the overall critical path of an iteration. In case of a square-root operation, the most significant quotient digit is always one while the following quotient digit is either $q_1 = \bar{1}$ or $q_1 = \bar{2}$. This covers the entire range from 0.5 to 1. To overcome the additional iteration due the leading $q_0 = 1$ and the need to compute 13 fractional digits to meet precision requirements, the first subtraction can be performed by replacing h, which is zero, by the scale factor k. The second quotient digit can be obtained by applying random logic to the non-scaled radicand. The combination of $q_0.q_1 = 1.\bar{1}$ covers the range [0.584, 1). Hence, similar to the division, it is sufficient to examine 3 bits $(X_1X_2X_3)$ of the unscaled input operand to predict the correct digit for q_1.

In every iteration, a decision has to be made, a selection between division and square-root performed (kD or kQ^i), a multiple of kY be selected (using a multiplexer structure) and subtracted from the previous remainder (Hybrid-adder) (see Fig. 4).

RW/DC represents the rewrite and decision criteria of the most significant two digits. The most significant two digits have to be rewritten for the following cases: $2\bar{2} \rightarrow 12$, $1\bar{2} \rightarrow 02$, $\bar{1}2 \rightarrow 0\bar{2}$, $\bar{2}2 \rightarrow \bar{1}\bar{2}$. This step insures the conversion of the algorithm [20]. The residual bounds can also be found in [20] and easily applied to the square-root architecture. Rewriting the most significant two digits which is performed by random logic results in a quotient digit which corresponds exactly to the most significant digit. The three select signals *neg*, *val* and *zero* selects the correct multiple of the denoninator or square-root quotient according to following criterias. If $R_0 \leq 0$ (the bit with the weight -2 in the coding of the minimally redundant digit set is 1), *neg* is set to high. The control signal *val* indicates which value the new quotient digit has (either 1 or 2). This can be achieved by simply XORing the bits of weight $+1$ of the most significant digit. The control signal *zero* is set to high if all three bits of the most significant digit are all zero or one. The control signal *div* distinguished between the division and square-root operation.

The square-root iteration also consists of a four bit adder that adds twice the value of the corresponding multiple of the scale factor k to the previous scaled square-root quotient $(2q_i \cdot r^{-i} \cdot k)$. In case of a negative multiple, the addition leads to a wrong result, since the most significant bits of the negative multiple of k are ones. Normally, this calls for a full-length addition increasing the critical path tremendously. Nevertheless, this bottleneck can be solved by modifying

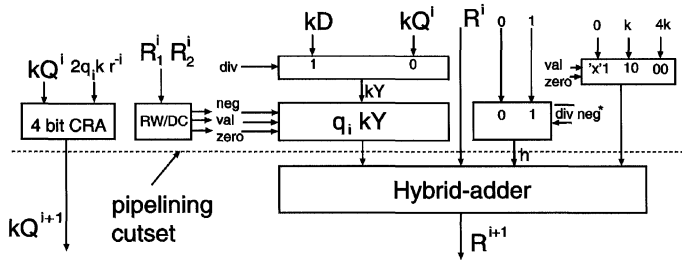

FIGURE 4 The architecture of the division/square-root algorithm based on the GST.

the Hybrid-adder in such a way, the not one but two hybrid-additions are performed. The first Hybrid-addition subtracts the possible wrong scaled square-root quotient while the second addition corrects the result by adding ones up to the bit position $2i$, where i corresponds to the current iteration. In case that there are l negative quotient digits, entirely l different words with leading ones have to be added. This bottleneck can be eliminated by realizing that the addition of all those correcting terms can be simplified by using an on-the-fly converter which uses the scale-factor k and the quotient digit q_i as its inputs.

The signal $\overline{divneg}$ selects the correct value for h in case of square-root operation and a negative quotient digit has been predicted. The result of the updated square-root quotient and the correcting term are pipelined for the next iteration. To perform the update of the term $q_{i+1}^2 \cdot r^{-(i+1)} \cdot k$, a simple multiplexer structure can be chosen which selects between the 0, k and $4k$. This term is always positive due to q_{i+1}^2. The most significant bit of this term has the weight 2^{-1} smaller than the correcting term h and can be added in parallel to h. Figure 5 indicates the scheme of the update of h. Depending on a positive or negative quotient digit, either a word with all zeros or ones is added to the previous value of h. However, the ones are only placed upto the bit position $2 \cdot i$, where i corresponds to the iteration number.

q_i	h
$1.$	0 0
$\overline{1}$	1 1 1 1
$\overline{1}$	1 1 1 0 1 1
0	1 1 1 0 1 1 0 0
2	1 1 1 0 1 1 0 0 0 0
$\overline{1}$	1 1 1 0 1 0 1 1 1 1 1 1
2	1 1 1 0 1 0 1 1 1 1 1 1 0 0
0	1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0
$\overline{1}$	1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1
1	1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0
$\overline{1}$	1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1

FIGURE 5 The selection of the correcting term h required for the square-root computation.

In [13] a very high radix square-root architecture which utilizes prescaling and rounding is presented. The shown architecture indicates that two multiplications per iteration have to be performed. These multiplications are in the critical path and increase the iteration delay.

4. SIMULATION RESULTS

The algorithm has been implemented using 32-bit operands, 24 bits for the mantissa and 8 bits for the exponent using a 0.5 µm CMOS technology. The building blocks are designed using minimal transistor widths of $W_n = 3\lambda$. However, $W_p = 6\lambda$ is chosen to guarantee equal slew rates. Only drivers make use of larger transistor widths. HSPICE simulation have shown a 35% ($t_d = 4\,\text{ns} \cdot 13$) smaller latency of the GST compared to the maximally redundant SRT implementation ($t_d = 6.2\,\text{ns} \cdot 13$), and 44% smaller latency compared to the minimally redundant SRT implementation ($t_d = 7.1\,\text{ns} \cdot 13$) [26, 18]. Other power consumption studies have been published in [27, 28], however, they are limited to dividers.

In the Tables III, IV and V, the results of the power estimation are shown for a frequency of 100 MHz and $V_{dd} = 3.3\,\text{V}$. All elements are simulated separately. However, the load capacity of the next stage and wiring capacity between the two blocks are considered in the power simulations. Registers have been implemented after each iteration due to pipelining.

The total power consumption of the three presented architectures are $P_{GST} = 86.9\,\text{mW}$, $P_{SRT,mr} = 63.6\,\text{mW}$ and $P_{SRT,MR} = 70.7\,\text{mW}$, respectively, all at the frequency of 100 MHz and for a operand word-length of 24 bits. The supply voltage of the GST architecture can be reduced to obtain a low power implementation. By reducing the supply voltage to $\beta \cdot V_{dd}$, the faster implementation can be used at the same frequency as the slower implementation. Assuming $V_{dd} = 3.3\,\text{V}$, $V_t = 0.67\,\text{V}$, $\beta_{GST,MR}^2 = 0.5802$ and $\beta_{GST,mr}^2 = 0.4977$, respectively, are obtained [29]. The supply voltage may accordingly be reduced to

TABLE III Power estimation of the minimally redundant radix-4 GST architecture

Minimally redundant radix-4 GST architecture	
K-selector	0.15 mW
Full-adder	0.035 mW · 14 = 0.5 mW
D-MXA	1.2 mW
N-scale by red. adder	1.95 mW
Register for D	0.02 mW · 28 · 13 = 7.25 mW
Register for R_i	0.02 mW · 42 · 13 = 10.9 mW
random logic for RW and DC	0.15 mW · 13 = 1.95 mW
Muxes,Hybrid-adder	0.220 mW · 14 · 12 = 37 mW
On-line converter	0.021 mW · 13 = 0.25 mW
On-line converter Muxes	0.011 mW · 14 · 13 = 1.9 mW
Register of converter	0.02 mW · 24 · 13 = 6.25 mW
Muxes for Div/SQRT selection	0.011 mW · 28 · 13 = 3.8 mW
Muxes for h selection	0.011 mW · 21 · 13 = 3 mW
Register for h	0.02 mW · 28 · 13 · = 7.3 mW
Muxes for $q_i^2 \cdot k$ selection	0.017 mW · 7 · 13 = 1.5 mW
4-bit CRA	0.172 mW · 12 = 2 mW
total for SQRT	86.9 mW

TABLE IV Power estimation of the minimally redundant radix-4 SRT architecture

Minimally redundant SRT architecture	
Converter	0.18 mW · 13 = 2.35 mW
XOR	0.012 mW · 8 · 13 = 1.3 mW
OR	0.07 mW · 6.13 = 0.6 mW
PLA	0.05 mW · 13 = 0.65 mW
Register for D	0.02 mW · 28 · 13 = 7.25 mW
Register for R_i	0.02 mW · 42 · 13 = 10.9 mW
Muxes, Hybrid-adder	0.162 mW · 14 · 12 = 27.3 mW
On-line converter	0.02 mW · 13 = 0.25 mW
On-line converter Muxes	0.011 mW · 13 · 14 = 1.9 mW
Register of converter	0.02 mW · 24 · 13 = 6.25 mW
Muxes	0.011 mW · 28 · 13 = 3.8 mW
Extra register of converter	0.02 mW · 13 · 4 = 1.05 mW
total for SQRT	63.6 mW

TABLE V Power estimation of the maximally redundant radix-4 SRT architecture

Maximally redundant SRT architecture	
MXA 7 bit	0.253 mW · 13 = 3.3 mW
XOR	0.012 mW · 7 · 13 = 1.65 mW
OR	0.007 mW · 6 · 13 = 0.6 mW
NS	0.1 mW · 13 = 1.3 mW
LS	0.05 mW · 13 = 0.65 mW
DD	0.035 mW · 13 = 0.45 mW
Register for D	0.02 mW · 24 · 13 = 6.25 mW
Register for R_i	0.02 mW · 48 · 13 = 12.5 mW
Muxes to select $q_{i,1} \cdot D$	0.015 mW · 24 · 13 = 4.7 mW
Carry-save adders to add $q_{i,1} \cdot D$	0.035 mW · 24 · 13 = 10.9 mW
Muxes to select $q_{i,2} \cdot D$	0.015 mW · 24 · 13 = 4.7 mW

TABLE V (Continued)

Maximally redundant SRT architecture	
Carry-save adder to add $q_{i,2} \cdot D$	$0.035\,\text{mW} \cdot 24 \cdot 13 = 10.9\,\text{mW}$
On-line converter	$0.021\,\text{mW} \cdot 13 = 0.25\,\text{mW}$
On-line converter Muxes	$0.011\,\text{mW} \cdot 13 \cdot 14 = 1.9\,\text{mW}$
Register of converter	$0.02\,\text{mW} \cdot 24 \cdot 13 = 6.25\,\text{mW}$
Muxes	$0.011\,\text{mW} \cdot 28 \cdot 13 = 3.8\,\text{mW}$
Extra register of converter	$0.02\,\text{mW} \cdot 13 \cdot 4 = 1.05\,\text{mW}$
total for SQRT	70.7 mW

TABLE VI Comparison between GST and SRT_{MR} Algorithm

	GST	SRT_{MR}	GST	SRT_{MR}
Latency	52 ns	79.3 ns	0.65	1
$\text{Power}_{SQRT@4\,\text{ns}}$	218 mW	–	1	–
$\text{Power}_{SQRT@6.2\,\text{ns}}$	140.5 mW	114 mW	1	0.8
$\text{Power}_{SQRT} \cdot \text{Delay}$	11.3 nJ	9 nJ	1	0.8
$\text{Energy}_{SQRT} \cdot \text{Delay}$	589 nJ ns	717 nJ ns	0.82	1
$\beta^2 \cdot \text{Power}_{SQRT@6.2\,\text{ns}}$	81 mW	114 mW	0.71	1

TABLE VII Comparison between GST and SRT_{mr} Algorithm

	GST	SRT_{mr}	GST	SRT_{mr}
Latency	52 ns	92.3 ns	0.56	1
$\text{Power}_{SQRT@4\,\text{ns}}$	218 mW	–	1	–
$\text{Power}_{SQRT@7.1\,\text{ns}}$	121 mW	89.6 mW	1	0.73
$\text{Power}_{SQRT} \cdot \text{Delay}$	11.3 nJ	8.3 nJ	1	0.73
$\text{Energy}_{SQRT} \cdot \text{Delay}$	589 nJ ns	763 nJ ns	0.68	0.77
$\beta^2 \cdot \text{Power}_{SQRT@7.1\,\text{ns}}$	60.2 mW	89.6 mW	0.67	1

$V_{dd,GST} = 2.51\,\text{V}$ by comparing the GST to the maximally redundant SRT and to $V_{dd,GST} = 2.32\,\text{V}$ by comparing the GST to the minimally redundant SRT architecture. Hence the total power consumption of the GST-SQRT architecture with same latency as the maximally redundant SRT-SQRT, is $P_{GST} = 81\,\text{mW}$, and $P_{GST} = 60.2\,\text{mW}$ compared to the minimally redundant SRT (see Tabs. VI and VII). This corresponds to 29% and 33% less power, respectively.

Another way of showing the superiority of the GST over the SRT is by calculating the Power-Delay and Energy-Delay products of the circuits. Those results are also shown in Tables VI and VII. Many improvements to the divider implementations have been suggested in [30]. Since these improvements are applicable to both SRT and GST dividers in an identical way, so these don't change the overall ratio between the GST and SRT behavior of speed and power.

5. CONCLUSION

The GST division algorithm has been successfully applied to the square-root operation in a hardware-efficient manner. The additional hardware costs increase critical path only slightly, so that the benefits in speed of the GST algorithm over the SRT algorithm are maintained. The power consumption increases by a significant amount (plus 36%), however, by operating the architectures at the same speed, the supply voltage of the GST architecture can be reduced so that the critical

path match the clock frequency. Simulations have shown that the overall latency of the GST is 35% smaller compared to the fastest implementation of the SRT. Alternatively, by fixing the latency, the GST division/square-root implementation requires 29% less power compared to the SRT using a maximally redundant radix-4 digit set. To conclude, the GST approach leads to a superior design for division, square-root and shared division/square-root architectures for latency and power critical applications.

References

[1] Bose, B. K., Pei, L., Taylor, G. G. and Patterson, D. A., "Fast Multiply and Divide for a VLSI Floating-Point Unit", In: *Proc. of 8th IEEE Int. Symposium on Computer Arithmetic* (Como, Italy), pp. 87–102, May, 1987.

[2] Oberman, S. F. and Flyn, M. J., "Design issues in Division and Other Floating-Point Operations", *IEEE Trans. on Computers*, **46**, 154–161, Feb., 1997.

[3] Kunz, K. S., *Numerical analysis*. New York, McGraw-Hill, 1957.

[4] Southworth, R. W. and Deleeuw, S. L., *Digital computation on numerical methods*. New York, McGraw-Hill, 1965.

[5] Morsund, D. G. (1967). "Optimal starting values for Newton-Raphson calculation of $\sqrt{(x)}$", *Communication ACM*, **10**(7), 430–432.

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

[7] Tocher, K. (1958). "Techniques of multiplication and division for automatic binary computers", *Quart. Journ. Mech. Appl. Math.*, **2**(3), 364–384.

[8] Svoboda, A. (1963). "An algorithm for division", *Information Processing Machines* (*Prague, Czech Republic*), No. 9, pp. 25–32.

[9] Tung, C., "A division algorithm for signed-digit arithmetic", *IEEE Trans. Computers* (*short notes*), **C-17**, 887–889, Sept., 1968.

[10] Atkins, D. E., "High-radix division using estimates of the divisor and partial remainders", *IEEE Trans. on Computers*, **C-17**, 925–934, Oct., 1968.

[11] Gosling, J. B. and Blaeley, C. M. S., "Arithmetic unit with integral division and square-root", *IEE Proc. Pt E*, **134**, 17–23, Jan., 1987.

[12] Ercegovac, M. D., "An on-line square-root algorithm", In: *Proc. of 4th IEEE Symposium on Computer Arithmetic* (Santa Monica, CA), pp. 183–189, Oct., 1978.

[13] Lang, T. and Montuschi, P., "Very High Radix Square Root with Prescaling and Combined Division/Sqare Root Unit", *IEEE Trans. on Computers*, **48**, 827–841, Aug., 1999.

[14] Montuschi, P. and Lang, T., "Boosting Very-High Radix Division with Prescaling and Selection by Rounding", In: *Proceedings. 14th IEEE Symposium on Computer Arithmetic* (Adelaide, Australia), pp. 52–59, Aug., 1999.

[15] Ercegovac, M. D., Lang, T. and Montuschi, P., "Very-High Radix Division Root with Prescaling and Selection by Rounding", *IEEE Trans. on Computers*, **43**, 909–918, Aug., 1994.

[16] Tenca, A. F. and Ercegovac, M. D., "On the Design of High Radix On-line Division for Long Precision", In: *Proceedings. 14th IEEE Symposium on Computer Arithmetic* (Adelaide, Australia), pp. 44–51, Aug., 1999.

[17] Ercegovac, M. D. and Lang, T., "Radix-4 Square Root Without Initial PLA", *IEEE Trans. on Computers*, **39**, 1016–1024, Aug., 1990.

[18] Kuhlmann, M. and Parhi, K. K., "Fast Low-Power Shared Division and Square-Root Architecture", In: *Proc. of IEEE Int. Conf. on Computer Design*, (*ICCD*) (Austin, TX), pp. 128–135, Oct., 1998.

[19] Lang, T. and Montuschi, P., "Higher radix square-root with prescaling", *IEEE Trans. on Computer*, **41**, 996–1009, Dec., 1992.

[20] Montalvo, L. (1995). *Number systems for high performance dividers, Ph.D. Thesis.*

[21] Fandrianto, J., "Algorithm for high speed shared radix-4 division and radix-4 square-root", In: *IEEE 1987 8th Symposium on computer arithmetic* (Como, Italy), pp. 73–79, May, 1987.

[22] Fandrianto, J., "Algorithm for high speed shared radix-8 division and radix 8 square-root", In: *Proc. 9th IEEE Symposium on Computer Arithmetic* (Santa Monica, CA), Sep., 1989.

[23] Srinivas, H. R. and Parhi, K. K., "A Floating Point Radix 2 Shared Division/Square-Root Chip", *Journal of VLSI Signal Processing*, **21**, 37–60, May, 1999.

[24] Montuschi, P. and Ciminiera, L., "Design of a radix 4 division unit with simple selection table", *IEEE Trans. on Computers*, **41**, 1606–1611, Dec., 1992.

[25] Srinivas, H. and Parhi, K., "A Fast Radix-4 Division Algorithm", *IEEE Trans. on Computers*, **44**, 826–831, Jun., 1995.

[26] Kuhlmann, M. and Parhi, K. K., "Power comparison of SRT and GST dividers", In: *SPIE Proc. of the International Symposium on Optical Science, Engineering and Instrumentation* (San Diego, CA), Jul., 1998.

[27] Nannarelli, A. and Lang, T., "Low-Power Radix-4 Divider", In: *Inter. Conf. on Low-Power Electronics and Design* (Monterey, CA), pp. 205–208, Aug., 1996.

[28] Nannarelli, A. and Lang, T., "Power-Delay Tradeoffs for Radix-4 and Radix-8 Dividers", In: *Inter. Conf. on Low-Power Electronics and Design* (Monterey, CA), pp. 109–111, Aug., 1997.

[29] Chandrakasan, A., Sheng, S. and Brodersen, R., "Low power CMOS digital design", *IEEE Journal of Solid State Circuits*, **27**, 473–485, April, 1992.

[30] Harris, D., Oberman, S. and Horowitz, M., "SRT division architectures and implementations", In: *Proc. 13th Symp. Computer Arithmetic*, pp. 18–24, July, 1997.

Authors' Biographies

Martin Kuhlmann received his Diplom Ingenieur and PhD degree in electrical engineering from the University of Technology Aachen, Germany in 1997 and from the University of Minnesota, Minneapolis in 1999, respectively. Currently, he

is a staff design engineer at Broadcom Corporation, Irvine, CA. His reserach interests include computer arithmetic, digital communication, VLSI design and deep-submicron crosstalk. He is a member of IEEE.

Keshab K. Parhi is a distinguished McKnight University Professor of Electrical and Computer Engineering at the University of Minnesota, Minneapolis where he also holds the Edgar F. Johnson Professorship. He is currently a senior principal scientist at Broadcom Corp., Irvine, on leave of absence. He received the B.Tech., M.S.E.E., and Ph.D. degrees from the Indian Institute of Technology, Kharagpur (India) (1982), the University of Pennsylvania, Philadelphia (1984), and the University of California at Berkeley (1988), respectively.

His research interests include all aspects of VLSI implementations of broadband access networks. He is currently working on VLSI adaptive digital filters, equalizers and beamformers, error control coders and cryptography architectures, low-power digital systems, and computer arithmetic. He has published over 300 papers in these areas. He has authored the text book *VLSI Digital Signal Processing Systems* (Wiley, 1999) and coedited the reference book *Digital Signal Processing for Multimedia Systems* (Dekker, 1999). He received a Golden Jubilee medal from the IEEE Circuits and Systems Society in 1999, a 1996 Design Automation Conference best paper award, the 1994 Darlington and the 1993 Guillemin-Cauer best paper awards from the IEEE Circuits and Systems society, the 1991 paper award from the IEEE signal processing society, the 1991 Browder Thompson prize paper award from the IEEE, and the 1992 Young Investigator Award of the National Science Foundation.

He has served on editorial boards of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS – PART II: ANALOG AND DIGITAL SIGNAL PROCESSING, the IEEE TRANSACTIONS ON VLSI SYSTEMS, and the IEEE SIGNAL PROCESSING LETTERS. He is an editor of the JOURNAL OF VLSI SIGNAL PROCESSING. He is the guest editor of a special issue of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and two special issues of the JOURNAL OF VLSI SIGNAL PROCESSING. He served as technical program cochair of the 1995 IEEE Workshop on Signal Processing and the 1996 ASAP conference. He also serves on numerous technical program committees. Dr. Parhi was a distinguished lecturer of the IEEE Circuits and Systems society and is a fellow of IEEE.