یازدهمین کنفرانس بین المللی کامپیوتر انجمن کامپیوتر ایران، پژوهشگاه دانشهای بنیادی، پژوهشکده علوم کامپیوتر، ۴ تا ۶ بهمن ۱۳۸۴، تهران، ایران.
11th International CSI Computer Conference (CSICC'2006), School of Computer Science, IPM, Jan. 24-26, 2006, Tehran, Iran.

A Novel Implementation of Radix-4 Floating-Point Division/Square-Root Using Comparison Multiples

Hooman Nikmehr
School of Electrical and Electronic Engineering,
The University of Adelaide, Australia

Department of Computer Engineering,
Bu Ali Sina University, Hamedan, Iran

nhooman@eleceng.adelaide.edu.au

Braden Phillips and Cheng-Chew Lim
School of Electrical and Electronic Engineering,
The University of Adelaide, Australia

phillips,cclim@eleceng.adelaide.edu.au

ABSTRACT
A new implementation for minimally redundant radix-4 floating-point SRT division/square-root (division/sqrt) with the recurrence in the signed-digit format is introduced. The implementation is developed based on the comparison multiples idea. In the proposed approach, the magnitude of the quotient (root) digit is calculated by comparing the truncated partial remainder with 2 limited precision multiples of the divisor (partial root). The digit sign is determined by investigating the polarity of the truncated partial remainder. A timing evaluation using the logical synthesis (Synopsys DC with Artisan 0.18 μm typical library) shows a latency of 2.5 ns for the recurrence of the proposed division/sqrt. This is less than of the conventional implementation.

Keywords
digit recurrence algorithms, SRT division, redundant arithmetic

1 INTRODUCTION
To achieve high performance in carrying out massive mathematical computations, almost all recent microprocessors and digital signal processors, perform in hardware all four fundamental arithmetic operations, namely addition, subtraction, multiplication and division [1]. Studying the processors' architectures and implementations reveals that of the four operations, division is not performed as fast as addition, subtraction and multiplication [2].

In 1994, Intel lost $475 Million due to an error in the division part of the Pentium microprocessor's floating-point unit [3, 4]. This fiasco highlights that the algorithms, architectures and realisations proposed for division are still immature, requiring more investigation and attention especially when designing modern high performance processors.

The general objective of this work is to develop an algorithm for more robust radix-4 (floating-point) FP SRT division with less chance of error when being implemented. The algorithm is then modified to be used for FP sqrt as well. It is attempted to achieve more efficient implementation of radix-4 FP SRT division/sqrt by increasing the parallelism among the functional units. Having employed more efficient functioning modules with higher concurrency among them, a quicker circuit for radix-4 FP SRT division/sqrt is obtained. The time delay estimations carried out using the method of logical effort [5] and the logic synthesis show considerable decrease in the execution time with respect to conventional implementations.

2 BACKGROUND
Some surveys [6, 2] shows that most VLSI implementations of FP division are based on digit recurrence division algorithms known as SRT (SRT division algorithm was introduced independently by D. Sweeney [7], J. E. Robertson [8] and T. D. Tocher [9]). SRT division is an iterative algorithm with linear convergence toward the quotient. In this algorithm, the *quotient digit selection* (QDS) function calculates a fix number of the quotient bits every iteration. The speed of a FP SRT divider is mainly determined by the complexity of the *quotient digit selection* (QDS) function [1], which is traditionally implemented using the *lookup table* method. In this method, the QDS function is realised in a table totally implemented with a PLA or a ROM [10]. Since the QDS function is the main part of the critical path, any improvement in this circuit effectively decreases the delay of FP SRT division.

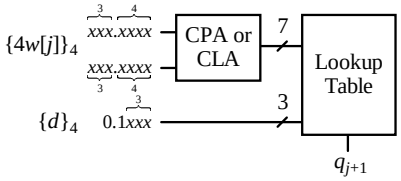

Figure 1: Implementation of the QDS function. CPA and CLA denote carry-propagate and carry-lookahead adders, respectively.

Ercegovac and Lang [10] cover almost all issues in designing and implementing SRT FP division including QDS techniques. A radix-4 SRT division compliant with the IEEE 754 standard [11] for double precision is implemented using the recurrence

$$w[j+1] = 4w[j] - q_{j+1}d, \text{ where } j = 0, 1, \cdots, 27 \quad (1)$$

and $w[0] = \frac{x}{4}$, plus one more cycle to produce the quotient in the IEEE 754 standard. In the recurrence (1), the dividend x and the divisor d are two normalised binary numbers in the range $[0.5, 1)$. Also, q_{j+1} represents the quotient digit in the signed-digit (SD) redundant format [12] selected from the minimally redundant radix-4 digit set

$$\left\{\overline{2}, \overline{1}, 0, 1, 2\right\}, \text{ where } \overline{m} = -m. \quad (2)$$

In (1), the next partial remainder (PR) $w[j+1]$ is represented in carry-save (CS) redundant format [12]. To select the correct q_{j+1}, the QDS function uses truncated $4w[j]$ and d as shown in Figure 1. In this figure, a 7-bit carry propagating adder (CPA) or a 7-bit carry lookahead adder (CLA) assimilates the 7 most significant sum and carry bits of the shifted PR to generate a 7-bit result. This result along with 3 bits from d are applied to a lookup table in order to fetch a value for q_{j+1}. In Figure 1, $\{X\}_c$ is the binary (redundant) number X truncated to c-th fractional bit (digit). The quotient digit q_{j+1} in (1) should be selected such that $w[j+1]$ is always bounded as

$$-\frac{2}{3}d \le w[j+1] \le \frac{2}{3}d, \quad (3)$$

which is called the *convergence condition*.

The general structure of radix-4 FP SRT division is shown in Figure 2. In this figure, the factor generator is a circuit delivering either d, $2d$, $-d$ or $-2d$ to CSA, based on the value of q_{j+1} retrieved by the QDS function. As shown in Figure 2, the quotient digits are delivered to a circuit for on-the-fly conversion. This circuit converts the SD quotient q represented to 2's complement and meanwhile, rounds and normalises the final result to comply with the IEEE 754 standard.

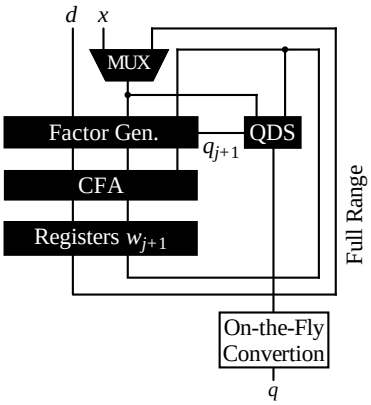

Figure 2: Conventional structure of radix-4 FP SRT division. CSA denotes a CS adder. The critical path passes through the black components.

After the well-published Pentium FDIV bug in 1994 [3], a considerable effort has been put to analysing the QDS function lookup table [13], studying its implementation [14] and verification [3, 15]. In addition, developing alternative approaches to implementing FP SRT division has been another agenda [10]. Recently, to implement the QDS function, a method using *selection constants* is widely employed [10]. This approach is more practical than the original lookup table method [2]. For the minimally redundant radix-4 design, the divisor's range $[0.5, 1)$ is partitioned into $2^3 = 8$ equal size intervals $[d_i, d_{i+1})$, where $d_1 = \frac{1}{2}$ and $d_{i+1} = d_i + 2^{-4}$. Then, a set of selection constants $m_k(i)$, where $k = -1, 0, 1, 2$, is associated to each interval. Using the divisor truncated to 4 fractional bits, the appropriate interval and consequently the correct $m_k(i)$ set is found.

In this method, the QDS function is defined through a set of the selection constants as for $d \in [d_i, d_{i+1})$, if $m_k(i) \leq 4w[j] < m_{k+1}(i)$, then $q_{j+1} = k$, where q_{j+1} is in the SD set (2). Like the traditional method, the comparison $m_k(i) \leq 4w[j] < m_{k+1}(i)$ can be performed in limited precision as $m_k(i) \leq \{4w[j]\}_4 < m_{k+1}(i)$, where $\{4w[j]\}_4$ is $4w[j]$ truncated to 4 fractional digits. The QDS function can be implemented using the selection constants method as shown in Table 1. Although the QDS function is defined very simple in the selection constants approach, the comparison of $\{4w[j]\}_4$ with $m_k(i)$ and $m_{k+1}(i)$ is still performed using a table similar to that used in the traditional implementation. Therefore, the design still suffers the complexity of the lookup table and the faults that may happen when designing and fabricating the chip.

Another step to optimising the implementation of FP SRT division is to remove the lookup table, either totaly or partly. This method still uses the selection constants method, however, it implements the QDS function using the comparators. In the traditional implementation, the truncated $4w[j]$ and d fetch a value for the quotient digit directly by addressing a specific location inside a lookup table. However, in the new approach, as the implementation of radix-4 QDS function in Figure 3 shows, all the selection constants are kept in a smaller lookup table and the appropriate set of $m_k(i)$ is selected using $\{d\}_4$ and then, $\{4w[j]\}_4$ and $m_k(i)$ take part in the subtractions in the form $\{4w[j]\}_4 - m_k(i)$, for $k = -1, 0, 1, 2$ followed by sign detections. Finally, the signs are manipulated by a coder to obtain the correct value for q_{j+1}. One recent implementation for the QDS function using comparators and selection constants is disclosed by Burgess and Hinds [17]. The QDS function is part of the divide/square root unit used in a vector processing chip called ARM VFPJ. As shown in Figure 3, to prevent carry propagation while comparing, $w[j]$ is represented in CS format. Therefore, CSAs can be employed. However, the constants are kept in the 2's complement format. This enhances the divider speed effectively.

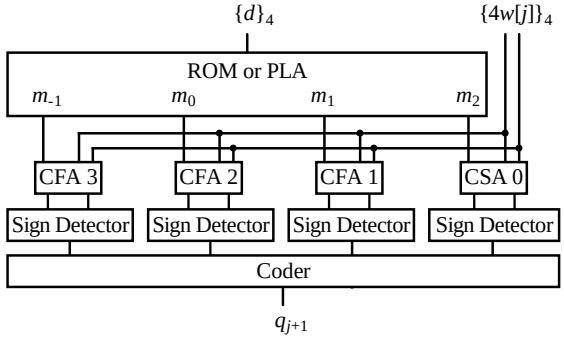

Figure 3: Implementation for the QDS function using comparators and selection constants.

3 COMPARISON MULTIPLE QDS

Although the latest techniques of implementing the QDS function contributing to increase the speed of FP SRT division are well studied in the literature, still the *comparison multiples* method [10] is not seriously considered by designers. For example, Ercegovac and Lang [10] claim:

> *Since the resulting implementation is still complicated because of the need for the multiples (of the divisor) and the comparisons, we develop the following alternative, which has more flexibility and results in a simpler implementation.*

A similar discussion on the comparison multiples method is given by Antelo et al [16] as:

> *An alternative implementation is based on comparisons of the residual estimate with truncated multiples of the divisor, however, this implementation is rarely used in practice because it requires assimilation of the truncated redundant residual and comparison, so no advantage is obtained with respect to the implementation with selection constants.*

Despite these quotes, there are reports of radix-2 [18] and radix-8 [19] SRT division, based on the comparison multiples idea, that show relatively improved response time. Moreover, Jensen [20] reports that although a highly optimised divider implemented using the conventional approach is just slightly faster than its un-optimised counterpart implemented using the comparison multiples idea, optimising the components of the critical path of the latter design may result in a faster circuit. Jensen also recounts other advantages of the comparison multiples approach such as simpler implementation, which allows the designer to produce the divider in less time and with less risk.

This section establishes a new approach for implementing minimally redundant radix-4 FP SRT division based on the comparison multiples method. The basic hypothesis as well as the mathematical framework of the proposed approach are discussed.

3.1 New Definition

Substituting (1) in (3) and adding dq_{j+1} results in

$$d(q_{j+1} - \frac{2}{3}) \leq 4w[j] \leq d(q_{j+1} + \frac{2}{3}). \quad (4)$$

Now, considering (3), since $q_{j+1} = k$ can take any of the 5 values in the SD set $\{\overline{2}, \overline{1}, 0, 1, 2\}$, the range (4) can be sliced into 5 intervals in the general form of $d(k - \frac{2}{3}) \leq 4w[j] \leq d(k + \frac{2}{3})$, where each interval is associated with a member of the SD set. So, to find an appropriate value for q_{j+1}, it is sufficient to investigate which interval contains $4w[j]$. This means that, the QDS function can be expressed as

$$q_{j+1} = k \in \{\overline{2}, \overline{1}, 0, 1, 2\}, \text{if } d(k - \frac{2}{3}) \leq 4w[j] \leq d(k + \frac{2}{3}). \quad (5)$$

Table 2 shows the QDS function (5) in an expanded form. Since (3) must be always satisfied and also because there is no possible choice for q_{j+1} larger (smaller) than 2 ($\overline{2}$), interval $\frac{4}{3}d \leq 4w[j] \leq \frac{8}{3}d$ is replaced by $\frac{4}{3}d \leq 4w[j]$ and $-\frac{8}{3}d \leq 4w[j] \leq -\frac{4}{3}d$ by $4w[j] \leq -\frac{4}{3}d$ in

Table 1: Minimally redundant radix-4 QDS function using the selection constant method [16].

	$\{d\}_4$							
m_k	0.1000	0.1001	0.1010	0.1011	0.1100	0.1101	0.1110	0.1111
m_2	00.1100	00.1110	00.1111	01.0000	01.0010	01.0100	01.0100	01.0110
m_1	00.0100	00.0100	00.0100	00.0100	00.0110	00.0110	00.1000	00.1000
m_0	11.1100	11.1010	11.1010	11.1000	11.1000	11.1000	11.1000	11.1000
m_{-1}	11.0011	11.0001	11.0000	10.1110	10.1100	10.1100	10.1010	10.1000

Table 2: The QDS function with an alternative expression.

q_{j+1}	Condition
$\overline{2}$	$4w[j] \le -\frac{4}{3}d$
$\overline{1}$	$-\frac{5}{3}d \le 4w[j] \le -\frac{1}{3}d$
0	$-\frac{2}{3}d \le 4w[j] \le \frac{2}{3}d$
1	$\frac{1}{3}d \le 4w[j] \le \frac{5}{3}d$
2	$\frac{4}{3}d \le 4w[j]$

the table. Investigation reveals that the intervals in Table 2 always have some overlaps, where there are two choices for the q_{j+1}. Therefore, to make the QDS function one-to-one, every two conjunct intervals are detached using separating points, namely the comparison multiples. The comparison multiple M_k is defined as

$$d(k-\frac{2}{3}) \le M_k \le d(k-1+\frac{2}{3}), \tag{6}$$

where $k \in \left\{\overline{2}, \overline{1}, 0, 1, 2\right\}$. The 2's complement number M_k can be represented as $M_k = A_k d$, where A_k is a rational number. The QDS function shown in Table 2 can be expressed as

$$q_{j+1} = \begin{cases} \overline{2}, & \text{if } 4w[j] < M_{-1} \\ \overline{1}, & \text{if } M_{-1} \le 4w[j] < M_0 \\ 0, & \text{if } M_0 \le 4w[j] < M_1 \\ 1, & \text{if } M_1 \le 4w[j] < M_2 \\ 2, & \text{if } M_2 \le 4w[j]. \end{cases} \tag{7}$$

Fortunately, existence of the overlaps means that there are sometimes more than one possible value for q_{j+1}. This allows the QDS function to compare just the most significant c fractional digits of $4w[j]$ with M_k truncated to c fractional bits. Therefore, (7) can be rewritten into the truncated form

$$q_{j+1} = \begin{cases} \overline{2}, & \text{if } \{4w[j]\}_c < \{M_{-1}\}_c \\ \overline{1}, & \text{if } \{M_{-1}\}_c \le \{4w[j]\}_c < \{M_0\}_c \\ 0, & \text{if } \{M_0\}_c \le \{4w[j]\}_c < \{M_1\}_c \\ 1, & \text{if } \{M_1\}_c \le \{4w[j]\}_c < \{M_2\}_c \\ 2, & \text{if } \{M_2\}_c \le \{4w[j]\}_c, \end{cases} \tag{8}$$

which requires 4 comparisons $\{4w[j]\}_c - \{M_k\}_c$ followed by 4 sign detections and a coder to be implemented.

3.2 Further Optimization

The two major ideas to speed up the QDS function and consequently, the recurrence cycle time, are breaking the critical path into two or more concurrent but shorter paths and decreasing the fan out of the circuits delivering $\{4w[j]\}_c$ to the QDS function.

A close look at Table 2 exposes a symmetry among the margins. This allows (8) to be redefined as

$$q_{j+1} = \begin{cases} \overline{2}, & \text{if } 4w[j] < 0 \quad \text{and} \quad \{4w[j]\}_c < -\{M_2\}_c \\ \overline{1}, & \text{if } 4w[j] < 0 \quad \text{and} \quad -\{M_2\}_c \le \{4w[j]\}_c < -\{M_1\}_c \\ 0, & \text{if } 4w[j] < 0 \quad \text{and} \quad -\{M_1\}_c \le \{4w[j]\}_c \\ 0, & \text{if } 4w[j] \ge 0 \quad \text{and} \quad \{4w[j]\}_c < \{M_1\}_c \\ 1, & \text{if } 4w[j] \ge 0 \quad \text{and} \quad \{M_1\}_c \le \{4w[j]\}_c < \{M_2\}_c \\ 2, & \text{if } 4w[j] \ge 0 \quad \text{and} \quad \{M_2\}_c \le \{4w[j]\}_c \end{cases} \tag{9}$$

since $\{M_{-k+1}\}_c = \{-M_k\}_c = -\{M_k\}_c$ for $k = 1, 2$.

The number of comparisons in (9) decreases to 2, however, (9) cannot be fulfilled unless the sign of the shifted PR is known. Checking the sign of $4w[j]$, while represented in the SD format, needs a full-length time consuming carry generation. This causes an impracticable response time proportional to $\log_2$ of the width of the division operands. It is shown later that existence of the overlaps helps to convert the full-length sign detection to a limited-range. In other words, to make the decision whether $\{M_k\}_c$ or $-\{M_k\}_c$ should participate in the comparisons, it is not necessary to know the exact sign of $4w[j]$, but the approximate sign, obtained from $4w[j]$ truncated to c' fractional digits. So, (9) changes to

$$q_{j+1} = \begin{cases} \overline{2}, & \text{if } \{4w[j]\}_{c'} < 0 \quad \text{and} \quad \{4w[j]\}_c < -\{M_2\}_c \\ \overline{1}, & \text{if } \{4w[j]\}_{c'} < 0 \quad \text{and} \quad -\{M_2\}_c \le \{4w[j]\}_c < -\{M_1\}_c \\ 0, & \text{if } \{4w[j]\}_{c'} < 0 \quad \text{and} \quad -\{M_1\}_c \le \{4w[j]\}_c \\ 0, & \text{if } \{4w[j]\}_{c'} \ge 0 \quad \text{and} \quad \{4w[j]\}_c < \{M_1\}_c \\ 1, & \text{if } \{4w[j]\}_{c'} \ge 0 \quad \text{and} \quad \{M_1\}_c \le \{4w[j]\}_c < \{M_2\}_c \\ 2, & \text{if } \{4w[j]\}_{c'} \ge 0 \quad \text{and} \quad \{M_2\}_c \le \{4w[j]\}_c. \end{cases} \tag{10}$$

The main modules involved in the implementation of the QDS function (10) are as follows.

3.2.1 *Comparison Multiple Generator*

This module produces the comparison multiples $\{M_1\}_c$ and $\{M_2\}_c$.

3.2.2 *Limited-Range Comparator*

For every $k \in \{1, 2\}$, a binary SD (BSD) adder [12] performs

$$\{4w[j]\}_c - \{M_k\}_c, \text{ if } \{4w[j]\}_{c'} \ge 0 \tag{11a}$$

$$\{4w[j]\}_c + \{M_k\}_c, \text{ if } \{4w[j]\}_{c'} < 0 \tag{11b}$$

or equivalently

$$\{4w[j]\}_c + \neg\{M_k\}_c + \neg S_{4w[j]}, \text{if } S_{4w[j]} = 0 \tag{12a}$$

$$\{4w[j]\}_c + \{M_k\}_c + \neg S_{4w[j]}, \text{if } S_{4w[j]} = 1, \tag{12b}$$

where '$\neg$' is 1's complement (invert), and

$$S_{4w[j]} = \begin{cases} 0, & \text{if } \{4w[j]\}_{c'} \ge 0 \\ 1, & \text{if } \{4w[j]\}_{c'} < 0. \end{cases} \tag{13}$$

3.2.3 *Limited-Range Comparison Sign Detector*

Each comparator in the QDS function is followed by a sign detector finding the sign of the result obtained from (12).

3.2.4 *Limited-Range PR Sign Detector*

The comparisons (12) cannot be accomplished unless the sign of $\{4w[j]\}_{c'}$ is already known. Therefore, to prevent any additional delay to the iteration response time, the QDS function should be implemented in such a way that the sign of $\{4w[j]\}_{c'}$ becomes available before (12) starts. Since $x > 0$, it is found that the sign of $\{4w[0]\}_{c'}$ is already known before the first iteration begins. Therefore, selecting a value for q_1 using (10) is independent of the sign detection operation. This observation can be extended to the j-th iteration to find the polarity of $\{4w[j+1]\}_{c'}$. The PR sign detector uses the function (13). This operation is performed in parallel to the rest of the QDS function and therefore, does not affect the iteration delay.

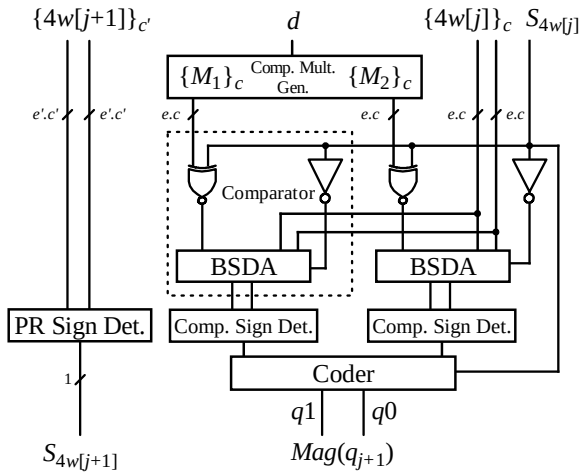

Figure 4: Structure of the radix-4 QDS function based on the comparison multiples idea. Notation *m.n* ia a BSD (2'complement) number represented in *m* integer and *n* fractional digits (bits).

3.2.5 *Coder*

As in the previous design, the results of the comparison sign detectors are applied the coder, which outputs the quotient digit in an eligible format. However, since the implementation of the QDS function is changed, the value for q_{j+1} is formed differently, this time using $Sign(q_{j+1})$ and $Mag(q_{j+1})$. In this representation,

$$Sign(q_{j+1}) = \begin{cases} S_{4w[j]}, & \text{if } q_{j+1} \in \{\overline{2}, \overline{1}, 1, 2\} \\ \text{don't care}, & \text{if } q_{j+1} = 0 \end{cases} \tag{14}$$

determines the sign and $Mag(q_{j+1})$ indicates the magnitude (absolute) of the value selected for q_{j+1}. In other words, $Mag(-q_{j+1}) = Mag(q_{j+1})$. Figure 4 shows the general structure of the QDS function.

3.3 Operand Precisions

3.3.1 *Determining c and e*

To determine c, which denotes the number of fractional digits of $4w[j]$ (as well as the number of fractional bits of M_k) involved in the BSD additions, the comparison intervals shown in (10) are studied. For simplicity, only the general interval $\{M_k\}_c \le \{4w[j]\}_c < \{M_{k+1}\}_c$ is investigated. However, the other cases can be derived in the same way. The interval is divided into

$$\{M_k\}_c \le \{4w[j]\}_c \tag{15a}$$

$$\{4w[j]\}_c < \{M_{k+1}\}_c\,. \tag{15b}$$

It is known that if a 2's complement number X is truncated to t fractional bits, then

$$\{X\}_t \le X < \{X\}_t + 2^{-t}. \tag{16}$$

However, for a BSD number Y the same truncation results in

$$\{Y\}_t - 2^{-t} < Y < \{Y\}_t + 2^{-t}. \tag{17}$$

Using (16) and (17), (15a) changes to $M_k - 2^{-c} < \{M_k\}_c \le \{4w[j]\}_c < 4w[j] + 2^{-c}$ or simply

$$M_k - 2^{-c+1} < 4w[j]. \tag{18}$$

Since $q_{j+1} = k$, adding $-kd$ to (18) and using (1) results in $M_k - 2^{-c+1} - kd < w[j+1]$. To maintain the convergence condition (3), the inequality $-\frac{2}{3}d \le M_k - 2^{-c+1} - kd$ or equivalently

$$d(k - \frac{2}{3}) + 2^{-c+1} \le M_k, \tag{19}$$

must be complied with. For interval (15b), a similar derivation can be used to obtain

$$M_{k+1} \le d(k + \frac{2}{3}) - 2^{-c}. \tag{20}$$

Table 3: Most convenient for generating values for M_1 and M_2.

k	Range	Selected A_k	M_k
1	$\frac{1}{3}d + \frac{1}{16} \le M_1 \le \frac{2}{3}d - \frac{1}{32}$	$\frac{1}{2}$	$\frac{1}{2}\,d$
2	$\frac{4}{3}d + \frac{1}{16} \le M_2 \le \frac{5}{3}d - \frac{1}{32}$	$\frac{3}{2}$	$\frac{3}{2}\,d$

Replacing $k+1$ with k in (20) and combining with (19) gives

$$d(k - \frac{2}{3}) + 2^{-c+1} \le M_k \le d(k - 1 + \frac{2}{3}) - 2^{-c}. \tag{21}$$

This inequality gives tighter ranges than condition (6). This is because rather than full-range, truncated comparison multiples are used in the comparisons. To make sure that finding a value for M_k using (21) is always possible, the inequality $d(k - \frac{2}{3}) + 2^{-c+1} < d(k - 1 + \frac{2}{3}) - 2^{-c}$, should be always maintained. This leads to $2^c > \frac{9}{d}$. Since $d \ge \frac{1}{2}$, it follows that $2^c > 18$ or

$$c \ge 5. \tag{22}$$

Inequality (22) gives the lower bound on c for the minimal redundant radix-4 QDS function based on the comparison multiples idea.

In addition to c, it is required to determine e, the width of the integer section of the operands involved in the BSD additions (11). Since $0 < M_k < \frac{8}{3}d$ and $\frac{1}{2} \le d < 1$, then M_k is a binary number with at most $\log_2 4 = 2$ integer bits. If $w[j]$ already has i integer digits, then the inputs to the BSD adders in Figure 4 have no more than $2 + i$ integer bits/digits. Therefore, $e = 2 + i$. For the particular implementation of radix-4 FP division shown in Figure 6, $i = 0$ and $e = 2$.

3.3.2 *Determining c' and e'*

From (17), it can be derived that $0 \le \{4w[j]\}_{c'} < 4w[j] + 2^{-c'}$ or equivalently

$$-2^{-c'} < 4w[j]. \tag{23}$$

To make (23) comply with the convergence condition (3) in the neighborhood of 0 with $q_{j+1} = 0$, the inequality

$$2^{-c'} \le \frac{2}{3}d \tag{24}$$

must be satisfied. Since $d \ge \frac{1}{2}$, inequality (24) can be changed to a tighter condition independent to d as $2^{c'} \ge 3$ or $c' \ge 2$.

Unlike 2's complement numbers, a BSD number has various representations. Therefore, $w[j+1]$ may be represented differently when appearing in different paths of the iteration. However, they all have a unique value. Considering i' as the number of integer digits of $w[j+1]$ when being applied to the PR sign detectors, using the same approach employed to determine e, the number of integer digits of the operand involved in the PR sign detection (13), can be expressed as $e' = 2 + i'$. For the radix-4 FP divider shown in Figure 6, $i' = 3$ and so, $e' = 5$.

3.4 Comparison Multiples Generator

Having substituted $c = 5$ in (21), the ranges, where M_1 and M_2 are defined, can be determined. The ranges along with the values selected for M_1 and M_2 are listed in Table 3. While $\{M_1\}_5 = \left\{\frac{1}{2}\,d\right\}_5$ is easily calculated just by shifting d one bit to the right and then keeping only the 5 most significant fractional bits, generating $\{M_2\}_5 = \left\{\frac{3}{2}\,d\right\}_5$ requires more operations. Mathematically, there are an infinite number of approaches to calculate $\{M_2\}_5 = \left\{\frac{3}{2}\,d\right\}_5$. However, investigation shows that the fastest approach is to perform $\{M_2\}_5 = \{2d\}_5 - \left\{\frac{1}{2}\,d\right\}_5$ using a 6-bit binary adder.

3.5 Comparators

In order to perform the comparisons (12), two 7-digit BSD adders can be employed. As shown in Figure 4, the adders follow the XNOR gates delivering either the set $\{M_1\}_5$ and $\{M_2\}_5$, or the set $\neg\{M_1\}_5$ and $\neg\{M_2\}_5$, to the adders. The comparison results are two BSD numbers with up to 8 digits. However, investigation shows that no representation overflow [21] happens when calculating (12). Therefore, the results could be represented in 7 digits instead. This makes the size of

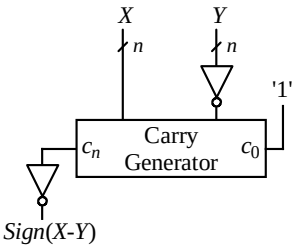

Figure 5: An architecture for n-digit BSD sign detectors using carry generators. For the sign detectors used in the proposed radix-4 QDS function $n = 7$.

Table 4: Values of q_{j+1} constructed by $Mag(q_{j+1})$ and $Sign(q_{j+1})$.

S_{M_1}	S_{M_2}	$Sign(q_{j+1})$	$Mag(q_{j+1}) = q1q0$	q_{j+1}
1	1	1	11	$\overline{2}$
1	0	1	10	$\overline{1}$
0	0	1	00	0
1	1	0	00	0
0	1	0	10	1
0	0	0	11	2

the comparison sign detectors smaller.

3.6 Comparison and PR Sign Detectors

According to the architecture in Figure 4, three circuits determine the polarities of three 7-digit BSD operands. The straightforward approach to determining the sign is to convert the BSD number to 2's complement format and check the most significant bit (sign bit).

The identity between BSD to binary (2's complement) conversion and the binary addition processes is now clearly understood [22,23,24,25]. Consider binary subtraction $Z = X - Y$, where X and Y are two n-bit 2's complement numbers. Using the BSD representation definition, the composite number $(X - Y)$ can be interpreted as a BSD number, where each BSD digit (x_i, y_i) has a value $(x_i - y_i)$. This means that any algorithm developed for the binary subtraction can be directly used for the BSD to binary conversion. However, since the exact value of the number is not important, employing methods that find the sign bit directly may result in a more efficient implementation for the sign detectors.

Figure 5 represents an architecture for such sign detectors. The architecture is derived from the fundamental definition of the binary subtraction [26]. According to this definition, when performing n-bit subtraction $Z = X - Y$, the n-th carry sent out from the subtractor can be recognised as the inverse of the sign of Z. This sign is equal to the polarity of the BSD number (X, Y).

The carry generator is defined as a part of parallel-prefix addition algorithms with overall delay of $O(\log_2 n)$ [27]. In parallel-prefix adders, a carry processing network precomputes all the carry-in signals required for the final calculation of the sum bits. However, in the three carry generators operating in the proposed FP divider, those parts of the circuit that generate c_1 to c_6 are ignored. This makes the implementation of the sign detectors simpler, smaller and probably faster.

3.7 Coder

Based on the value of $\{4w[j]\}_5$, the comparison sign detectors produce two bits, namely S_{M_1} and S_{M_2}, which construct $Mag(q_{j+1})$. The values represented by S_{M_1} and S_{M_2} as well as their relationship with the exact value of q_{j+1} are shown in Table 4. Investigating the table reveals that $Mag(q_{j+1})$ can be generated using the equations

$$q0 = S_{M_2} \text{ XNOR } Sign(q_{j+1}) \quad (25)$$

$$q1 = S_{M_1} \text{ XNOR } Sign(q_{j+1}). \quad (26)$$

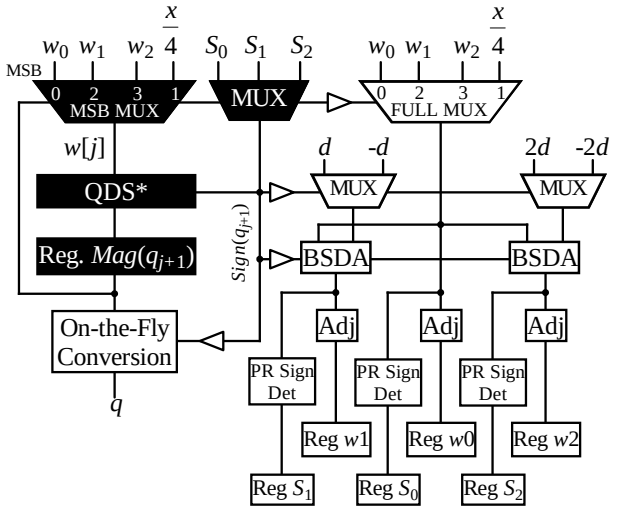

Figure 6: New implementation of the recurrence of radix-4 FP SRT division. QDS* refers to the QDS function without the PR sign detector.

3.8 Buffers

Sice the comparators, the comparison sign detectors and the coder are on the FP divider critical path [28], one way to decrease the recurrence critical path delay might be to minimise the fan out of the circuit supplying the QDS function's comparators with $\{4w[j]\}_5$ and $\{M_k\}_5$. So, as shown in Figure 4, buffers are inserted to limit the load by isolating the circuits producing $q1$ from the counterpart circuits generating $q0$. However, this means that $q0$ is delivered to the $Mag(q_{j+1})$ register earlier than $q1$. It can be fixed by clock skewing the $q1$ register by as much as one buffer propagation delay.

3.9 Evaluation of the QDS Function

The proposed QDS function can be evaluated as follows.

- In the new approach, the number of comparators and comparison sign detectors is halved compared to some recent designs [17,16]. This makes the recurrence operate faster by decreasing the fan out of the circuits delivering input signals to the QDS function.
- The new coder contributes less delay to FP SRT division because it produces $Mag(k)$ based on 3 inputs rather than 4. Also, compared to the previous coder, $Mag(k)$ is represented in fewer bits.
- In the new design, the PR sign detector, which determines $Sign(q_{j+1})$, operates in parallel to the rest of the QDS function. Therefore, the concurrency exposed by the redefined implementation reduces the critical path delay of FP SRT division.

4 RADIX-4 SRT DIVISION RECURRENCE

Traditionally, the recurrence of radix-r FP SRT division is based on the scheme represented in Figure 2. However, this conventional structure can be altered in order to achieve less delay. For radix-r FP SRT division, Nikmehr et al. [28] introduce a recurrence, which is optimised for the QDS function displayed in Figure 4. In dividers using the QDS function shown in Figure 3, a factor generator precalculates $\pm d$ and $\pm 2d$, and the quotient digit q_{j+1} determined by the QDS function selects the appropriate value for $-q_{j+1}d$ using a 5:1 multiplexer. This value is applied to the CFA to perform (1). In the new design, the PR formation does not wait for the quotient digit. While the QDS function is operating, the PR formation computes all possible values for the next PR in advance and then adjusts them. This causes more overlap between the QDS function and the PR formation. It is schematically shown in Figure 6. Two distinct paths can be found in the figure. While the shaded one is considered to be the critical path, the other

Table 5: Reformatting process carried out by the adjust unit.

w'_k<57:53>= $ABC.XY$	w_k<54:53>= $.xy$
$000.XY$	$.XY$
$001.0\overline{1}$	$.11$
$001.\overline{1}X$	$.1X$
$00\overline{1}.01$	$.\overline{11}$
$00\overline{1}.1X$	$.\overline{1}X$
$01\overline{1}.0\overline{1}$	$.11$
$01\overline{1}.\overline{1}X$	$.1X$
$0\overline{1}1.01$	$.\overline{11}$
$0\overline{1}1.1X$	$.\overline{1}X$
$1\overline{11}.0\overline{1}$	$.11$
$1\overline{11}.\overline{1}X$	$.1X$
$\overline{1}11.01$	$.\overline{11}$
$\overline{1}11.1X$	$.\overline{1}X$

with shorter delay runs in parallel.

4.1 The Recurrence Components

The main components involved in the implementation of the new recurrence are as follows.

4.1.1 *MSB and FULL MUX*

Two wide 4:1 multiplexors, namely MSB MUX and FULL MUX, can be found in the recurrence shown in Figure 6. However, since $w0$, $w1$, and $w2$ are BSD numbers, each multiplexer represents two similar multiplexors with binary inputs. While the BSD inputs to MSB MUX are 7 digits wide, FULL MUX deals with 55-digit BSD numbers. To comply with the PR initialisation $w[0] = \frac{x}{4}$, the $Mag(q_{j+1})$ register is initialised to 01. According to Table 4, this combination is an invalid code, which is never generated during division.

4.1.2 *MUX 3:1*

As shown in Figure 6, the correct sign of the current shifted PR, $S_{4w[j]}$, is selected by a 3:1 multiplexer. Since $w[0] = \frac{x}{4}$ is a positive number, the S_0, S_1 and S_2 register are initialised to 0.

4.1.3 *PR Formation*

There are three candidates for the next PR, namely w'_0, w'_1 and w'_2. While generating $w'_0 = 4w[j]$ needs no hardware, producing

$$w'_1 = \begin{cases} 4w[j] + d, & \text{if } S_{4w[j]} = 1 \\ 4w[j] - d, & \text{if } S_{4w[j]} = 0 \end{cases} \tag{27}$$

and

$$w'_2 = \begin{cases} 4w[j] + 2d, & \text{if } S_{4w[j]} = 1 \\ 4w[j] - 2d, & \text{if } S_{4w[j]} = 0 \end{cases} \tag{28}$$

requires two full length BSD additions.

4.1.4 *Adjust Unit*

As shown in Figure 6, three instances of the adjust unit operate in parallel to fix representation overflow of the three candidates of the next PR, w'_0, w'_1 and w'_2. Since $\frac{1}{2} \le d < 1$, for $k = 0, 1, 2$, (3) implies

$$0.\overline{1}0\overline{1}0\overline{1}0\cdots = -\frac{2}{3} < w'_k < \frac{2}{3} = 0.101010\cdots. \tag{29}$$

This means that it is always possible to reformat the widened w'_k into 55-digit arrays with no integer digit. Having named the adjusted values as w_0, w_1 and w_2, Table 5 shows the reformat process. While the left column indicates all the possible combinations, which may appear in the 5 most significant digits of w'_0, w'_1 and w'_2,[1] the right column shows the adjusted values, which have no nonzero digit in their integer part. The reformatting process can be summarised as

[1] Other combinations of digits never happen in the integer parts of w'_0, w'_1 and w'_2 otherwise, condition (29) is not fulfilled.

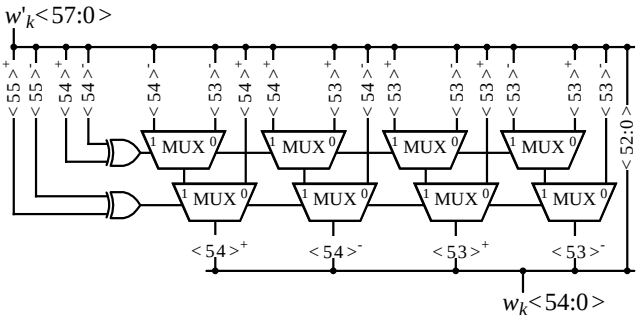

Figure 7: The adjust unit based on function 30, where w'_k <55 : 53>= $C.XY$ and w_k<54:53>= $.xy$, and $k = 0, 1, 2$.

$$xy = \begin{cases} XY, & \text{if } C = 0 \\ (-Y)(-Y), & \text{if } C \neq 0 \text{ and } X = 0 \text{ and } Y \neq 0 \\ (-X)Y, & \text{if } C \neq 0 \text{ and } X \neq 0 \text{ and } Y \neq 0. \end{cases} \tag{30}$$

Using (30), a simple implementation for the adjust unit is developed and shown in Figure 7. As shown in the figure, the original and the adjusted digit arrays share the BSD digits numbered <52:0> and only three digits are involved in the reformatting process. This means that after digits w_k <53 : 54> are generated by the adjust unit, they are concatenated to the rest of the digits, which are remained unchanged.

4.1.5 *PR Sign Detector*

Since three possible values for the next PR are generated by the PR formation, three instances of the PR sign detector are required so that all the possible polarities of the next PR are determined in advance. Although the PR sign detector is originally part of the QDS function, due to the new structure of the radix-4 recurrence, it is moved out of the QDS function to operate in parallel and off the critical path.

4.1.6 *Convert, Round and Normalise Unit*

The convert, round and normalise (CRN) unit converts q from SD to 2's complement representation, rounds the result based on the RTNE scheme [11] and normalises the final quotient, on-the-fly [10].

4.2 Evaluation of the Recurrence

The design shown in Figure 6 can be evaluated as follows.

- A new register, namely $Mag(q_{j+1})$, is introduced to store the magnitude of q_{j+1}. In addition, the registers tagged S_0, S_1 and S_2, and $w0$, $w1$ and $w2$ are used to keep all the 3 possible values calculated for $S_{4w[j+1]}$ and $w[j+1]$, respectively.
- Duplicating the wide multiplexer to MSB and LSB MUX minimises the fan out of the $Mag(q_{j+1})$ register. This technique not only balances the load by distributing it over the $Mag(q_{j+1})$ and the set of S_0, S_1 and S_2 registers, but also divides the critical path into 2 concurrent shorter paths. Therefore, recurrence cycle time is reduced.
- The MUX 2:1, which in Figure 2 selects between the PR and x, is embedded in the MSB MUX in Figure 6. This decreases the critical path delay, since one multiplexer arm is retired. This change requires the $Mag(q_{j+1})$ register be initialised such that in the first iteration, the input corresponding to x is selected.
- LSB MUX, the PR sign detectors, the PR formation and the adjust units are isolated from the critical path using buffers. Having the noncritical components detached from the rest of the design improves the recurrence cycle time since the critical circuit bears less load.
- The new recurrence has different critical path comparing to the conventional implementation.

5 COMPARISON MULTIPLE SQRT

Section 5 describes SRT FP sqrt and the issues related to its implementation. Due to sharing the same characteristics, sqrt $s = \sqrt{x}$ can be represented using digit recurrence algorithms as in division $q = \frac{x}{d}$

Table 6: Bit patterns representing $-\left(2S[j]s_{j+1} + s_{j+1}^2 4^{-(j+1)}\right)$ (adapted from [10]).

s_{j+1}	Bit Pattern
$\overline{2}$	$F_{\overline{2}} = b\cdots b1100$
$\overline{1}$	$F_{\overline{1}} = b\cdots bb111$
0	$F_0 = 0\cdots 00000$
1	$F_1 = \overline{a}\cdots\overline{aa}111$
2	$F_2 = \overline{a}\cdots\overline{a}1100$

[10]. This implies that with minor changes, the comparison multiple technique developed for SRT FP division can be used for implementing SRT FP sqrt. To keep the discussion concise, this section covers only the changes that need to be applied to Sections 3 and 4.

5.1 Sqrt Recurrence

To meet the IEEE standard requirements, it is assumed that the radicand x is bounded on the interval $[\frac{1}{4}, 1)$. Therefore, the root s is in the region $\frac{1}{2} \le s < 1$.

For minimally redundant radix-4 sqrt, the value of the root after the j-th iteration is represented as $S[j] = \sum_{i=0}^{j} s_i 4^{-i}$, where $s_0 = 1$ and $s_i \in \left\{\overline{2}, \overline{1}, 0, 1, 2\right\}$. Therefore, the final result is $s = S[n] = \sum_{i=0}^{n} s_i 4^{-i}$. In this algorithm, the convergence condition is represented as

$$-\frac{4}{3}S[j] + \frac{4}{9}4^{-j} < w[j] < \frac{4}{3}S[j] + \frac{4}{9}4^{-j} \tag{31}$$

with the initial condition $w[0] = x - 1$. The SRT sqrt recurrence is

$$w[j+1] = 4w[j] - \left(2S[j]s_{j+1} + s_{j+1}^2 4^{-(j+1)}\right), \tag{32}$$

which is carried out by the PR formation using a BSD adder. Since the digits of $S[j]$ are originally produced in SD format, it should be converted into 2's complement form using on-the-fly conversion. In on-the-fly conversion, two variables A and B are required. They are updated, in every iteration, as $A[j] = S[j]$ and $B[j] = S[j] - 4^{-j}$. Now, the recurrence (32) can be performed as

$$w[j+1] = \begin{cases} 4w[j] + F[j] + 1, & \text{if } s_{j+1} > 0 \\ 4w[j] + F[j], & \text{otherwise,} \end{cases} \tag{33}$$

where $F = -\left(2S[j]s_{j+1} + s_{j+1}^2 4^{-(j+1)}\right)$ is a 2's complement number generated as shown in Table 6. In the table, $a\cdots aa$ and $b\cdots bb$ are the bit strings representing $A[j]$ and $B[j]$, respectively.

5.2 Root Digit Selection Function

Having replaced q_{j+1} with s_{j+1}, (10) can be used as the root digit selection (RDS) function for the minimally redundant radix-4 FP sqrt. However, the values of c, e, c', e' M_1 and M_2 for SRT FP sqrt may be different from those calculated for SRT FP division.

Having applied the technique developed in Section 3.3 along with the convergence condition (31) and the recurrence (32), the precisions $c = 4$, $e = 3$, $c' = 1$ and $e' = 6$ are calculated. Also, the ranges where M_1 and M_2 can be selected from are obtained as

$$\frac{2}{3}S[j] + \frac{1}{9}4^{-(j+1)} \le M_1 \le \frac{4}{3}S[j] + \frac{4}{9}4^{-(j+1)} \tag{34}$$

$$\frac{8}{3}S[j] + \frac{16}{9}4^{-(j+1)} \le M_2 \le \frac{10}{3}S[j] + \frac{25}{9}4^{-(j+1)}. \tag{35}$$

For $j > 1$, the comparison multiples for the RDS function are $\{M_1\}_4 = \{S[j]\}_4$ and $\{M_2\}_4 = \{3S[j]\}_4$. In case $j = 0$, since $S[0] = 1$, the comparison multiples can be $\{M_1\}_4 = 1$ and $\{M_2\}_4 = \frac{7}{2}$. For $j = 1$, $\{M_1\}_4 = \{S[1]\}_4$ and $\{M_2\}_4 = \{3S[1]\}_4 + \frac{1}{8}$. As appeared, unlike SRT division, the comparison multiples, have to be computed every iteration. While $\{M_1\}_4 = \{S[j]\}_4$ is obtained from on-the-fly conversion, $\{M_2\}_4 = \{4S[j]\}_4 - \{S[j]\}_4$ can be calculated by a 6-bit binary adder.

6 COMBINED DIVISION AND SQRT

The IEEE 754 standard requires the designers to implement both division and sqrt in the FP units of the microprocessors [11]. Given the number of similarities between the recurrences of these two, it is very normal to implement a combined circuit that perform both operations. Examples of such implementations can be found in [29, 30].

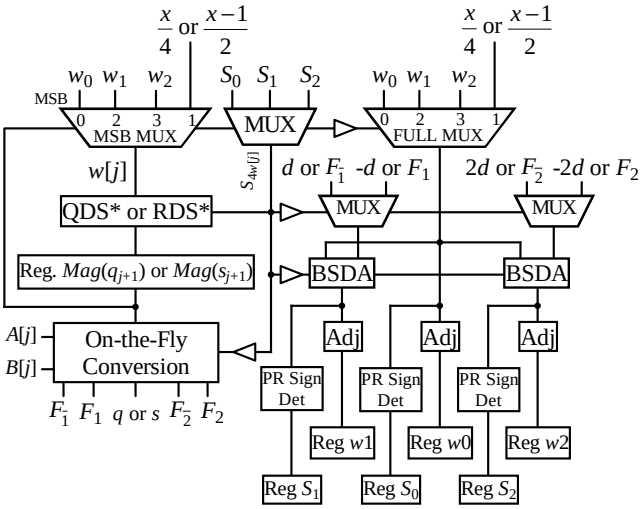

Figure 8: Combined FP SRT division/sqrt unit based on the comparison multiples idea. For sqrt, $A[j]$ and $B[j]$ are sent to the circuits calculating M_1, M_2, $F_{\overline{2}}$, $F_{\overline{1}}$, F_1 and F_2 every iteration.

Comparing the specification of FP SRT division and sqrt reveals that to match the recurrences, it is sufficient to modify the sqrt PR such that

$$\text{new } w[j] = \frac{\text{old } w[j]}{2}. \tag{36}$$

Having called new $w[j]$ just $w[j]$, (32) can be rewritten as

$$w[j+1] = 4w[j] - \frac{\left(2S[j]s_{j+1} + s_{j+1}^2 4^{-(j+1)}\right)}{2}. \tag{37}$$

Accordingly, the precisions are changed to $c = 5$, $e = 2$, $c' = 2$ and $e' = 5$. Also, the comparison multiples are altered to $\{M_1\}_5 = \left\{\frac{S[j]}{2}\right\}_5$ and $\{M_2\}_5 = \left\{\frac{3S[j]}{2}\right\}_5$. Figure 8 displays the combined radix-4 FP SRT division/sqrt unit based on the comparison multiples idea.

7 TIMING ANALYSIS

Using the method of logical effort, the critical path delay of the proposed radix-4 FP divider is obtained as 15.63 FO4[2]. Using the same timing evaluation technique, the critical path delay of the conventional radix-4 FP divider with a PD plot type QDS function [10] is calculated as 22.53 FO4. Comparing these values shows that the proposed radix-4 FP divider is almost 28% faster than the traditional divider. This comparison is supported by the logic synthesis carried out using Synopsys design compiler (DC) with Artisan 0.18 μm typical library on the new and traditional radix-4 FP dividers. This timing evaluation shows that the critical path delays of the comparison multiple and the conventional radix-4 FP dividers are 2.34 ns and 3 ns, respectively. Comparing these values shows 22% speedup for the new divider.

Having applied the logic synthesis to the combined radix-4 FP SRT division/sqrt unit, the latency of 2.5 ns is estimated for a cycle. The critical path delay for the combined unit is almost 7% more than that for the divider only. This is mainly due to the more complex circuit generating F_1, $F_{\overline{1}}$, F_2 and $F_{\overline{2}}$.

8 CONCLUSION

A new realisation of the QDS function based on the new comparison multiples approach is proposed. The QDS function is then used for implementing a minimally redundant radix-4 FP SRT division. In this method, which is mathematically and architecturally described, instead of searching for the quotient digit in a lookup table, the quo-

[2] Fan out of 4.

tient digit is directly calculated in sign and magnitude format. Using the new representation for the quotient digits, the fan out of some components on the critical path is almost halved making them operate faster. The QDS function receives a truncated PR and investigates the range to which the PR belongs. It performs the examination by comparing the truncated PR with the truncated multiples of the divisor produced once at the beginning of division. The result of the comparisons are delivered to a coder in order to produce the magnitude of the quotient digit. Meanwhile, another part of the QDS function, called the PR sign detector, calculates the polarity of the quotient digit by inspecting the sign of the truncated PR. In the comparison multiple QDS function, the PR sign detector operates off the critical path because the quotient digit sign and magnitude are calculated separately. These changes make the QDS function faster and consequently, reduce the divider critical path delay. The new idea can be used also to implement a combined division/sqrt unit. While a timing estimation using the method of logical effort shows that the comparison multiple divider is almost 28% faster than the conventional implementation, another evaluation using the logic synthesis supports the improvement by reporting 22% speedup for the comparison multiple divider comparing to the traditional approach. The logic synthesis shows the critical path delay of 2.5 ns for the combined division/sqrt unit.

8 REFERENCES

[1] S. F. Oberman and M. J. Flynn, "Division Algorithms and Implementations," *IEEE Transactions on Computers*, vol. 48, no. 6, pp. 833–854, August 1997.

[2] P. Soderquist and M. Leeser, "Area and performance tradeoffs in floating-point divide and square-root implementations," *ACM Computing Surveys (CSUR)*, vol. 28, no. 3, pp. 518–564, 1996.

[3] R. E. Bryant, "Bit-Level Analysis of an SRT Divider Circuit," in *Proceedings of the 33rd Annual Conference on Design Automation*. Las Vegas, NV, USA: ACM Press, June 1996, pp. 661–665.

[4] C. B. Moler, "A Tale of Two Numbers," *SIAM News*, vol. 28, no. 1, pp. 16–16, January 1995.

[5] I. E. Sutherland, R. F. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA, USA: Morgan Kaufman Publishers, Inc., 1999.

[6] S. F. Oberman, N. Quach, and M. J. Flynn, "The Design and Implementation of a High-Performance Floating-Point Divider," Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA, USA, Tech. Rep. CSL-TR-94-599, January 1994.

[7] J. Cocke and D. W. Sweeney, "High Speed Arithmetic in a Parallel Device," IBM Corp., Tech. Rep., February 1957.

[8] J. E. Robertson, "A New Class of Digital Division Methods," *IRE Transactions on Electronic Computers*, vol. EC-7, no. 3, pp. 88–92, September 1958.

[9] K. D. Tocher, "Techniques of Multiplication and Division for Automatic Binary Computers," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 11, pp. 364–384, 1958.

[10] M. D. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Norwell, MA, USA, and Dordrecht, The Netherlands: Kluwer Academic Publishers Group, 1994.

[11] IEEE, *Std 754-1985 IEEE Standard for Binary Floating-Point Arithmetic*, Standards Committee of The IEEE Computer Society. 345 East 47th Street, New York, NY 10017, USA, 1985.

[12] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Walton Street, Oxford OX2 6DP, UK: Oxford University Press, 2000.

[13] N. Burgess and T. Williams, "Choices of Operand Truncation in the SRT Division Algorithm," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 933–938, 1995.

[14] S. F. Oberman and M. J. Flynn, "Measuring the Complexity of SRT Tables," Computer Systems Laboratory, Stanford University, Stanford, CA, USA, Tech. Rep. CSL-TR-95-679, November 1995.

[15] H. Rueß, N. Shankar, and M. K. Srivas, "Modular Verification of SRT Division," in *Computer-Aided Verification, CAV'96*, ser. Lecture Notes in Computer Science, vol. 1102. New Brunswick, NJ, USA: Springer-Verlag, July/August 1996, pp. 123–134.

[16] E. Antelo, T. Lang, P. Montuschi, and A. Nannarelli, "Fast Radix-4 Retimed Division with Selection by Comparisons," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'02)*, San Jose, CA, USA, 17–19 July 2002, pp. 185–196.

[17] N. Burgess and C. Hinds, "Design Issues in Radix-4 SRT Square Root and Divide Unit," in *Proceedings of the 35th Asilomar Conference on Signals, Systems and Computers (ASILOMAR'01)*, Pacific Grove, CA, USA, 4–7 November 2001, pp. 1646–1650.

[18] V. Kantabutra, "A New Algorithm for Division in Hardware," in *Proceedings of the 1996 International Conference on Computer Design*. Austin, TX, USA: IEEE Press, October 1996, pp. 551–556.

[19] ——, "A New Theory for High-Radix Division in Hardware: Two Direct, Comparison-Based Radix-8 Cases," August 1997, unpublished http://math.isu.edu/∼vkantabu/radix8.pdf.

[20] T. A. Jensen, "Alternative Implementations of SRT Division and Square Root Algorithms," Master's thesis, Department of Mathematics and Computer Science (IMADA), University of South Denmark, Odense, Denmark, June 1998.

[21] N. Burgess, "Radix-2 SRT Division Algorithm with Simple Quotient Digit Selection," *Electronics Letters*, vol. 27, no. 21, pp. 1910–1911, October 1991.

[22] M. Uya, K. Kaneko, and J. Yasui, "A CMOS Floating-Point Multiplier," *IEEE Journal of Solid State Circuits*, vol. SC-19, no. 5, pp. 697–702, October 1984.

[23] A. Vandemeulebroecke, E. Vanzieleghem, T. Denayer, and P. G. A. Jespers, "A New Carry-Free Division Algorithm and its Application to a Single-Chip 1024-b RSA Processor," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 3, pp. 748–755, June 1990.

[24] H. R. Srinivas and K. K. Parhi, "A Fast VLSI Adder Architecture," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 5, pp. 761–768, May 1992.

[25] K. K. Parhi, "Fast Low-Energy VLSI Binary Addition," in *Proceedings of IEEE Conference on Computer Design*, Austin, TX, USA, October 1997, pp. 676–684.

[26] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design*, 1st ed. New York, NY, USA: John Wiley & Sons, 1979.

[27] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 3rd ed. Addison-Wesley, 2004.

[28] H. Nikmehr, "Architectures for Floating-Point Division," PhD Dissertation, School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, Asutralia, June 2005.

[29] M. D. Ercegovac and T. Lang, "Module to Perform Multiplication, Division, and Square Root in Systolic Arrays for Matrix Computations," *Journal of Parallel and Distributed Computing*, vol. 11, no. 3, pp. 212–221, 1991.

[30] S. E. McQuillan, J. V. McCanny, and R. F. Woods, "High performance vlsi architecture for division and square root," *Electronics Letters*, vol. 27, no. 1, pp. 19–21, Jan. 1991.