New Algorithms and VLSI Architectures for SRT Division and Square Root

S.E. McQuillan, J.V. McCanny, R. Hamill

The Institute of Advanced Microelectronics
Department of Electrical and Electronic Engineering
The Queens University of Belfast
Belfast BT9 5AH, Northern Ireland

Abstract

In real time digital signal processing, high performance modules for division and square root are essential if many powerful algorithms are to be implemented. In this paper, new radix 2 algorithms for SRT division and square root are developed. For these new schemes, the result digits and the residuals are computed concurrently and the computations in adjacent rows are overlapped. Consequently, their performance should exceed that of the radix 2 SRT methods. VLSI array architectures to implement the new division and square root schemes are also presented.

1 Introduction

In general computation systems, the frequency of division and square root operations tends to be considerably lower than that of addition and multiplication. Consequently, many arithmetic processors do not include hardware support for these operations. Software routines for division and square root can be up to an order of magnitude slower than multiplication and addition and, as a result, the time-weighted impact of division and square root on processor utilisation can be significant. By way of illustration, consider an application requiring one division operation for every ten multiply-add operations. Assuming that the division takes 50ns and multiply-add 20ns then the percentage of the processor time devoted to division can be as much as 20%! In general computations, the speed of divide and square root operations is not critical. However, in real time digital signal processing, high performance division and square root are essential if many powerful algorithms are to be implemented.

In designing fast, hardware-oriented arithmetic algorithms, VLSI architectural design issues such as regularity, modularity and locality of interconnections must be addressed. This facilitates the mapping of the algorithm on to an architecture which is amenable to a VLSI implementation and assists in the test and complexity management of designs. The purpose of this paper is to describe the development of new radix 2 division and square root algorithms which are both fast and suited to a VLSI implementation.

The paper is structured as follows. Section 2 outlines the characteristics of the SRT division method and identifies the factors which inhibit performance. In section 3, the SRT method is modified to give a faster division algorithm and an array architecture to implement the new algorithm is presented. Section 4 focuses on square root while section 5 shows how the division and square root algorithms can be combined. This is followed by a discussion of the work in section 6.

2 The SRT Division Method

The well known SRT division method [1] is a digit-recurrence algorithm which utilises arithmetic redundancy [2] to reduce the required precision of comparisons between the divisor and the residual. The radix r recurrence for computing successive residuals is

$$Z_j = rZ_{j-1} - q_j \cdot D \ , \ j=1,2,3... \qquad (1)$$

where Z_j is the residual, $D \in [½,1)$ is the normalised divisor and $q_j \in \{-\rho..\rho\}$ is the jth quotient digit. Here, upper case variables refer to a complete word while lower case variables refer to individual digits of a word. A convention is adopted whereby the subscript i of a digit indicates a significance of r^{-i}. The initial residual Z_0 is set equal to the dividend and, to ensure convergence, the residual at the jth step must satisfy $|Z_j| < \rho D/(r-1)$.

The quotient is accumulated by appending successive quotient digits to the partial quotient Q_j, i.e., $Q_j = Q_{j-1} + q_j r^{-j}$. The representation of the quotient is

1063-6889/93 $03.00 © 1993 IEEE

redundant and usually employs the radix r signed digit number representation described by Avizienis [3]. One consequence of the redundant representation is that the quotient has alternative representations e.g., in the signed binary number representation (SBNR) with the digit set $\{-1,0,1\}$, the number $1011̄0$ is equivalent to 10010 where $\bar{1}$ denotes '-1'. Therefore, at each step, there may exist a degree of choice in selecting a valid quotient digit from the given digit set. Alternatively, quotient digits can be determined by examining a low precision estimate of the residual. For radix 2, this is typically the three most significant digits (MSDs) [14]. Any error introduced into the accumulating quotient can then be corrected on subsequent iterations. This is in contrast to conventional division where the full precision residual must be examined in determining a quotient digit. The SRT square root algorithm described by Majerski [4] is similar to SRT division and will not be outlined here.

There are two main factors which limit the performance of the SRT methods. Firstly, a serial dependency exists among the iterations. This is fundamental to the successive approximation method of computing the quotient. Secondly, the computation of the residual and the quotient digit are performed sequentially. This can be attributed to the need to examine an estimate of the residual in determining a quotient digit and is compounded by the occurrence of redundancy overflow when the *residual* is represented redundantly (redundancy overflow is the false overflow associated with redundant number systems and occurs when a n$-$digit operand occupies more than n significant digits). The aim of the work to be described has been to modify the radix 2 SRT algorithms to overlap the computation of the residual and the quotient digit whilst retaining the advantages of the SRT methods.

3 Modified SRT Division

3.1 The Algorithm

The key to this new radix 2 method of division is the reformulation of the recurrence in (1) as

$$Z_j^*=2Z_{j-1}+\alpha\ ,\ \alpha=\begin{cases}D & \text{if } q_j\in\{-1,0\}\\ -D & \text{if } q_j\in\{0,1\}\end{cases}$$
$$Z_j=\begin{cases}Z_j^* & \text{if } q_j=-1 \text{ or } 1\\ 2Z_{j-1} & \text{if } q_j=0\end{cases} \tag{2}$$

where $D\in[D_{\min},D_{\max})$ and q_j is chosen from the SBNR digit set. Clearly, the computation of the tentative residual, Z_j^*, can proceed *before* the full quotient digit has been computed. All that is required is that the quotient digit to be located to either the $\{0,1\}$ or $\{-1,0\}$ subsets. The quotient digit can then be computed *concurrently* and in a separate path to that of the tentative residual. To facilitate this, it must be possible to compute α as quickly as possible. Therefore, α should be dependent upon the MSD only of the residual and redundancy overflow in the residual should be avoided. This can be achieved by introducing a more stringent bound on the residual, namely $|Z_j|<D_{\min}$.

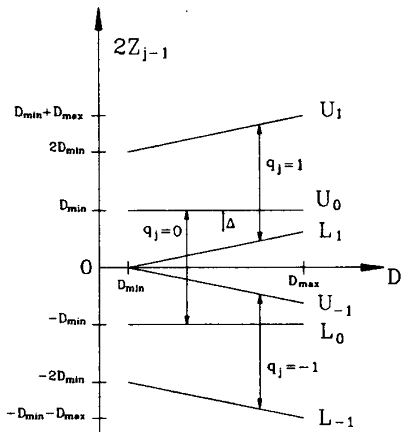


Figure 1 Selection Regions for Division

The analysis of the proposed algorithm follows the general method for on-line arithmetic described by Ercegovac and Lang [6]. The general quotient digit selection function can be written as:

$$q_j=k \text{ if } L_k<2Z_{j-1}<U_k\ ,\ k\in\{-1,0,1\} \tag{3}$$

where L_k and U_k denote the lower and upper bounds respectively of selection region k, a region in which the choice $q_j=k$ represents a valid selection. On substituting the residual bound into (1), the upper and lower bounds of the quotient digit selection regions are determined as:

$$\begin{aligned}U_k&=D_{\min}+kD\\ L_k&=-D_{\min}+kD\end{aligned} \tag{4}$$

for $k=-1,0,1$. The selection regions are illustrated in Figure 1. Note that adjacent selection regions overlap $-$ the manifestation of the redundancy in the representation of the quotient - and within the overlap regions, a degree of choice exists in selecting the quotient digit. In practice, a simple selection function based on examining a low precision estimate of the residual only is required. This simplified selection function, which must also facilitate the computation of α, is described by:

$$q_j=k \text{ if } l_k\le\hat{z}\le u_k\ ,\ k\in\{-1,0,1\} \tag{5}$$

where l_k and u_k define the lower and upper bounds respectively of the new, non-overlapping selection regions and $\hat{z}$ is the scaled residual truncated to t fractional digits. The residual and its estimate are related by:

$$-\Delta_{neg} \leq 2Z_{j-1} - \hat{z} \leq \Delta_{pos} \tag{6}$$

where $\Delta_{neg} > 0$ and $\Delta_{pos} > 0$ are the truncation errors. Valid result digit selection is achieved by ensuring that if $l_k \leq \hat{z} \leq u_k$ then $L_k < 2Z_{j-1} < U_k$. More formally, this is written as:

$$\begin{aligned} l_k &\geq (L_k)_{\max} + \Delta_{neg} \\ u_k &\leq (U_k)_{\min} - \Delta_{pos} \end{aligned} \tag{7}$$

where the subscripts *max* and *min* denote maximum and minimum. The selection thresholds, u_1 and l_{-1}, are special cases which must satisfy:

$$\begin{aligned} u_1 &< 2|Z_j|_{\max} + \Delta_{neg} \\ l_{-1} &> -2|Z_j|_{\max} - \Delta_{pos} \end{aligned} \tag{8}$$

This ensures that, for each residual value, it is possible to select a quotient digit from the signed binary digit set. The usable overlap of adjacent selection regions Δ is given by:

$$\Delta = (U_{k-1})_{\min} - (L_k)_{\max} = 2D_{\min} - D_{\max} \tag{9}$$

for $k=0,1$ and noting that $l_k = u_{k-1} + 2^{-t}$ (since the new selection regions are disjoint and both u_k and l_k are integer multiples of 2^{-t}), then from (7) it follows that:

$$\Delta \geq \Delta_{pos} + \Delta_{neg} - 2^{-t} \tag{10}$$

For a given number representation of the residual, the corresponding values of u_k, l_k and t can be determined by solving (7), (8) and (10).

$\hat{z}$	q	a/s	restore	compress	$\hat{z}-q.D_2$
00	0	x	1	0	00
$0\underline{1}$	0	x	1	0	$0\underline{1}$
$0\bar{1}$	0	x	1	0	$0\bar{1}$
10	1	0	0	x	00
$1\underline{1}$	1	0	0	x	01
$1\bar{1}$	0	x	1	1	01
$\underline{\bar{1}}0$	$\bar{1}$	1	0	x	$0\underline{0}$
$\underline{11}$	$\underline{0}$	x	1	1	$0\underline{\bar{1}}$
$\overline{11}$	$\bar{1}$	1	0	x	$0\bar{1}$

x=Don't Care

Table 1 Quotient Digit and Signal Generation for Modified SRT Division

Assuming a signed binary residual and a residual estimate with t fractional digits, then the truncation errors are $\Delta_{pos} < 2^{-t}$ and $\Delta_{neg} < 2^{-t}$ and, therefore, from (10), the overlap must satisfy $\Delta \geq 2^{-t}$. If $D_{\min}$ and $D_{\max}$ are chosen as ½ and ¾ respectively and the dividend is in the range [¼,½) – to avoid quotient overflow – then $|2Z_j| < 1$ and $\Delta = ¼$. With $D_{\min} = ½$ and $D_{\max} = 1$, the overlap is reduced to zero necessitating full precision comparisons between the divisor and the residual. The divisor range [¾,1) results in a more complex selection function and is dismissed at this stage. Consequently, t=2 i.e., the residual estimate is $\hat{z} = 0.z_1z_2$. The corresponding selection thresholds are $l_k = ¾k - ¼$ and $u_{k-1} = ¾k - ½$ for k=0,1. Also, $u_1 = 1$ and $l_{-1} = -1$. Equivalently, the selection function can be written as:

$$q_j = \begin{cases} 1 & \text{if } \tfrac{1}{2} \leq \hat{z} \leq 1 \\ 0 & \text{if } -\tfrac{1}{4} \leq \hat{z} \leq \tfrac{1}{4} \\ -1 & \text{if } -1 \leq \hat{z} \leq -\tfrac{1}{2} \end{cases} \tag{11}$$

In order to avoid redundancy overflow, $\hat{z} = 1$ and $\hat{z} = -1$ should not occur i.e. $\hat{z}$ should have no integer component. This requires the residual estimate to be compressed when a zero is selected as a valid quotient digit e.g. $0.1\bar{1}$ is compressed to 0.01.

Table 1 details the necessary control signals required by the algorithm for each digit-pair comprising the residual estimate. The **restore** signal is required when a zero is selected as a quotient digit in order to select the previous residual, rather than the tentative residual, as the new residual. The **compress** signal is required when the quotient digit is a zero and the MSD of the residual is non-zero i.e. when $\hat{z} = 0.\bar{1}1$ or $\hat{z} = 0.1\bar{1}$. The signal which determines whether the divisor multiple is D or –D is the add/subtract signal, a/s for brevity, and can be determined from the MSD of the residual as required. It is important to note that the first digit of the term $\hat{z} - q.D_2$, where D_2 is the two MSDs of the divisor, is always zero such that, when the new residual is scaled, no redundancy overflow occurs. This is due to both the particular divisor range and the value of the selection thresholds u_0 and l_0 and is an important property of the algorithm. Failure to avoid redundancy overflow would complicate the computation of α.

3.2 Division Array Architecture

An architecture to implement the modified SRT division algorithm is illustrated in Figure 2. The circuit comprises a regular array of type 1 and type 2 cells with quotient digits and control signals being determined by the S cells on the periphery of the array. The functional and gate level descriptions of the basic cells are given in Figure 3. The divisor digits $d_3d_4d_5$... and the dividend $N=0.0n_2n_3$... enter the array in a bit-parallel manner as shown. Since the two MSDs of the divisor are known

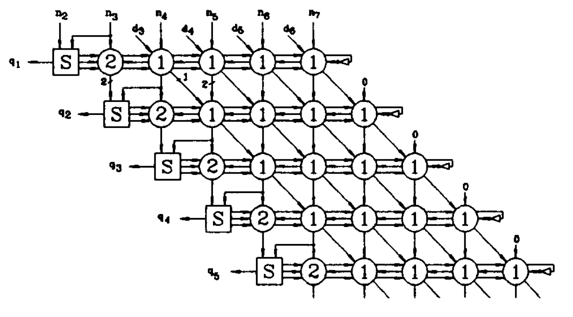


Figure 2 Radix 2 Division Array

implicitly, they are not input to the array. Each signed binary digit, z, comprising the residual is composed of two digits namely, $z^+ \in \{0,1\}$ and $z^- \in \{-1,0\}$, which are encoded as shown in Table 2 [13]. This coding enables conventional full-adders to be employed in adding/subtracting a signed binary operand and a binary operand. The four *interpretations* of a full adder employing this coding scheme are illustrated in Figure 4 where '+' denotes a positively weighted input i.e. {0,1} and '−' denotes a negatively weighted input i.e. {−1,0}.

z^+	Code
0	0
1	1

z^-	Code
−1	0
0	1

Table 2 Coding of z^+ and z^- digits

Each row of the architecture implements one iteration of the algorithm as follows. The first signal generated by the S cell is a/s. This is available as soon as z_1, the MSD of the scaled residual from the preceding row, becomes available. The a/s signal is then broadcast to all cells in the row to form the quotient digit extractor. When a/s = 1, the quotient digit extractor is simply the divisor D. When a/s = 0, the divisor must be negated. This is achieved by bit complementing the divisor, each *digit* being assigned a negative weight. The carry into the rightmost full adder is always interpreted as zero. The new tentative residual, Z_j^*, and the previous residual enter a multiplexer controlled by the restore signal. The previous residual is selected or "restored" only when the restore signal is set otherwise the tentative residual is selected.

The two bits comprising the second MSD, z_2, of the previous residual are inverted in the type 2 cell when the compress signal is set. This effectively compresses the

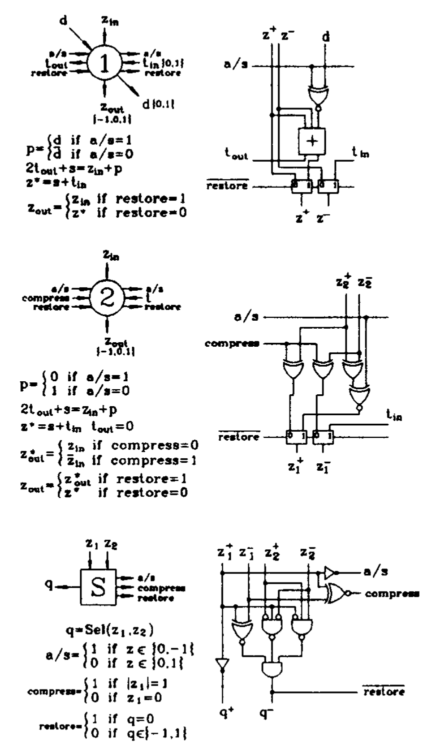

d
z_{in}
a/s
t_{out}
restore
1
$t_{in}\{0,1\}$
z_{out} $\{-1,0,1\}$
$d\{0,1\}$
$p=\begin{cases} d & \text{if } a/s=1 \\ \bar{d} & \text{if } a/s=0 \end{cases}$
$2t_{out}+s=z_{in}+p$
$z^*=s+t_{in}$
$z_{out}=\begin{cases} z_{in} & \text{if restore}=1 \\ z^* & \text{if restore}=0 \end{cases}$
z^+ z^-
+
$\overline{\text{restore}}$
t_{in}
2
compress
$p=\begin{cases} 0 & \text{if } a/s=1 \\ 1 & \text{if } a/s=0 \end{cases}$
$2t_{out}+s=z_{in}+p$
$z^*=s+t_{in}$ $t_{out}=0$
$z^*_{out}=\begin{cases} z_{in} & \text{if compress}=0 \\ \bar{z}_{in} & \text{if compress}=1 \end{cases}$
$z_{out}=\begin{cases} z^*_{out} & \text{if restore}=1 \\ z^* & \text{if restore}=0 \end{cases}$
z_2^+ z_2^-
z_1^+ z_1^-
z_1 z_2
q
S
$q=\text{Sel}(z_1,z_2)$
$a/s=\begin{cases} 1 & \text{if } z\in\{0,-1\} \\ 0 & \text{if } z\in\{0,1\} \end{cases}$
$\text{compress}=\begin{cases} 1 & \text{if } |z_1|=1 \\ 0 & \text{if } z_1=0 \end{cases}$
$\text{restore}=\begin{cases} 1 & \text{if } q=0 \\ 0 & \text{if } q\in\{-1,1\} \end{cases}$
q^+ q^-

Figure 3 Description of the Basic Cells

bit strings $0.1\bar{1}$ and $0.\bar{1}1$ to 0.01 and $0.0\bar{1}$ respectively ensuring no redundancy overflow. *Concurrently*, the S cell computes the quotient digit by examining the two MSDs of the previous residual.

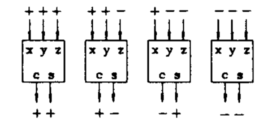


Figure 4 Full Adder Interpretations

Comparison with radix 2 SRT dividers indicates that this circuit should yield a higher performance for two main reasons. Firstly, there is an obvious reduction in the critical path due to the concurrent computation of the quotient and the residual. Secondly, the computations in adjacent rows are overlapped thereby reducing the average critical path per row. This is explained by noting that the MSD of the new residual is computed before the other residual digits and, therefore, the computation of the divisor multiple in the subsequent row can begin while the other residual digits are being determined.

3.3 Pre-scaling of the Divisor and the Dividend

The pre-scaling of the operands is effected by a simple shift-and-add operation. Assuming the divisor and dividend are available in the ranges $\frac{1}{2} \leq D < 1$ and $\frac{1}{4} \leq N < \frac{1}{2}$ respectively then the need to pre-scale the operands can be ascertained by examining the second MSD of the divisor, namely d_2. If $d_2 = 0$ then no pre-scaling is necessary. If $d_2 = 1$ then both N and D are pre-scaled by multiplying by $\frac{3}{4}$. This is achieved by adding $\frac{1}{2}D$ and $\frac{1}{4}D$ and the resultant quotient is always in the range $[\frac{1}{4},1)$. Whereas the scaled dividend can be computed in a signed binary format, the computation of the scaled divisor must be fast and yield the scaled divisor in a non-redundant form, as required by the array. Fast adder schemes, e.g. carry look-ahead adders [12], can be employed to do this. Pre-scaling of operands does not reduce the accuracy of the quotient but, as with the classical SRT scheme, the remainder does not conform to the IEEE 754 standard [11].

3.4 Comparison with Other Methods

A radix 2 division algorithm, based on pre-scaling the divisor into a suitable range and in which the quotient digit is determined concurrently with the computation of the residual at each step, has been proposed by Ercegovac and Lang [10]. This algorithm is also a modified SRT type scheme. However, the divisor is restricted to the range $7/8 \leq D \leq 9/8$ and, consequently, the pre-scaling of the operands is a little more complex than that required for the new algorithm. As the performance, area, power consumption etc of the circuits are highly technology dependent, it is not possible, at present, to give an absolute comparison of the relative merits of both methods. Fully objective comparisons require full-custom designs to be examined. Using gate equivalents as a measure, the critical path per row of both circuits is comparable while the gate count of the main cell of the circuit in [10] is less than that of the new divider. However, a greater throughput is expected with the new divider due to the overlapping of successive computations within adjacent rows. The objective of the approach described in [10] is to match the delays of the residual computation and the quotient digit selection and, therefore, overlapping of the computations in adjacent rows is not possible.

Recently, an array for implementing the radix 2 version of the Svoboda-Tung [7,8] division algorithm was reported by Burgess [9]. The array is similar to that illustrated in Figure 2 and should have a comparable throughput and hardware cost. On closer examination, this is not surprising since it can be shown that the SRT division method and the Svoboda-Tung algorithm share the same recurrence for computing successive residuals [15]. Unlike the scheme presented by Burgess, the division method described here can be extended to higher radices and square root.

4 Modified SRT Square Root

4.1 The Algorithm

The application of the above modification to the more demanding case of square root is described in this section. The scaled residual at the jth step of the square root algorithm is written as:

$$Z_j = 2^{j-1}(R - S_j^2) \tag{12}$$

where R is the radicand and S_j is the partial root accumulated at the jth step. At each step, the radicand and partial root are related by $|\sqrt{R} - S_j| < 2^{-j}$ which requires that the residual be bounded as:

$$2^{-j-1} - S_j < Z_j < S_j + 2^{-j-1} \text{ , } j = 1,2,3 \tag{13}$$

It can be shown that if $R \in [\frac{1}{4}, \frac{1}{2})$ then the bound on the residual can be replaced by $|Z_j| < \frac{1}{2}$ for $j = 2,3,4\ldots$ The initial residual is then $Z_2 = 2R - \frac{1}{2} = 0.0r_3r_4\ldots$ The recurrence for computing successive residuals is given by:

$$\begin{aligned} &Z_j^* = 2Z_{j-1} + \alpha - 2^{-j-1} \text{ , } \alpha = \begin{cases} S_{j-1} & \text{if } s_j \in \{-1,0\} \\ -S_{j-1} & \text{if } s_j \in \{0,1\} \end{cases} \\ &Z_j = \begin{cases} Z_j^* & \text{if } s_j = -1 \text{ or } 1 \\ 2Z_{j-1} & \text{if } s_j = 0 \end{cases} \end{aligned} \tag{14}$$

The general selection function in (3) applies where the upper and lower bounds of the selection regions are defined as:

$$\begin{aligned} U_k &= \tfrac{1}{2} + kS_{j-1} + k^2 2^{-j-1} \\ L_k &= -\tfrac{1}{2} + kS_{j-1} + k^2 2^{-j-1} \end{aligned} \tag{15}$$

for $k = -1,0,1$. The simplified selection function in (5) also applies here together with the associated constraints in (7), (8) and (10). As in the case of division, the residual is assumed to be a signed binary operand. Consequently, the truncation errors are $\Delta_{neg} < 2^{-t}$ and $\Delta_{pos} < 2^{-t}$ where t is the precision of the residual estimate $\hat{z}$. The overlap of adjacent selection regions, independent of j, is given by:

$$\Delta=(U_{k-1})_{\min}-(L_k)_{\max}=1-\frac{1}{\sqrt{2}} \qquad (16)$$

This implies $t \geq 2$. Choosing $t=2$, it is now possible to determine the selection thresholds, l_k and u_k. Remembering that both l_k and u_k are integer multiples of 2^{-t} then $l_k=\frac{3}{4}k-\frac{1}{4}$ and $u_{k-1}=\frac{3}{4}k-\frac{1}{2}$ for $k=0,1$. This is exactly the same as the selection function derived for the redundant restoring division algorithm and, therefore, the generation of the necessary control signals is the same as that in Table 1.

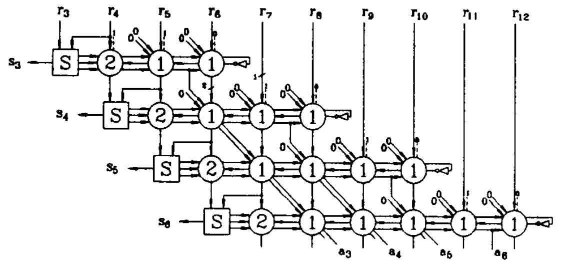
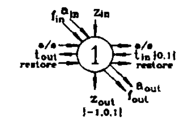


Figure 5 Square Root Architecture

4.2 Square Root Architecture

An architecture to implement the modified radix 2 SRT square root algorithm is illustrated in Figure 5. The circuit comprises a skewed triangular array of type 1 cells bounded at the left hand edge by type 2 and type S cells. The S cells generate the result digits and control signals as specified by Table 1. Note that only the $n-1$ least significant digits of the n-bit radicand R enter the array since the initial residual is given by $Z_2=0.0r_3r_4\ldots$ Note also that the two MSDs of the square root are known implicity due to the range restriction on the radicand, i.e., $S_2=\frac{1}{2}$. The functional and gate level descriptions of the type 1 cell is given in Figure 6 (see Figure 3 for cell types 2 and S).

The jth row operates as follows. The a/s signal, generated by the S cell, is broadcast to the row of type 1 and type 2 cells. This enables the square root digit extractor multiple to be formed. The extractor is composed of two parts, namely the partial root, S_{j-1}, and a subtractive term -2^{-j-1}. The subtraction of 2^{-j-1} is achieved by directing a zero into the spare negatively weighted input of the rightmost type 1 cell. Due to the coding of the residual digits, this is correctly interpreted as '-1'. The partial root, S_{j-1}, has been converted to a conventional representation, denoted by A_{j-1}, and is available at the jth step. When $a/s=1$, A_{j-1} is passed



$$p=\begin{cases} a_{in} & \text{if } a/s=1 \\ \bar{a}_{in} & \text{if } a/s=0 \end{cases}$$

$$2t_{out}+s=z_{in}+p$$

$$z^*=s+t_{in}$$

$$z_{out}=\begin{cases} z_{in} & \text{if restore}=1 \\ z^* & \text{if restore}=0 \end{cases}$$

$$f_{out}=\begin{cases} 1 & \text{if } f_{in}=1 \text{ or restore}=0 \\ 0 & \text{if } f_{in}=0,\ \text{restore}=1 \end{cases}$$

$$a^*=\begin{cases} a_{in} & \text{if } f_{in}=1 \text{ or } a/s=0 \\ \bar{a}_{in} & \text{if } a/s=1, f_{in}=0 \end{cases}$$

$$a_{out}=\begin{cases} a^* & \text{if restore}=0 \\ a_{in} & \text{if restore}=1 \end{cases}$$

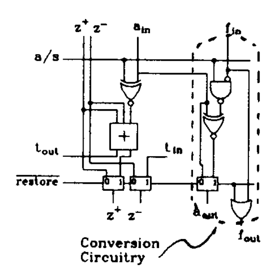


Figure 6 Description of the Type 1 Cell

unchanged to the row of full-adders. When $a/s=0$, the bit-complement of A_{j-1} is passed to the adder. The adder then computes the tentative residual Z_j^* which is passed to the multiplexer. If the restore signal is set, the previous residual is selected otherwise the tentative residual is selected as the new residual Z_j.

The conventional binary representation A_{j-1} of the signed binary partial root S_{j-1} is updated concurrently with the computation of the square root digit and the new residual. This is performed by the conversion circuitry in the type 1 cells and uses a modified version of the algorithm described in references [4,5]. The SBNR-to-binary conversion algorithm is modified by the introduction of a tentative conventional representation of each digit, a^*, and is described by the equations in Figure 6 (see reference [5]). The performance of this array is comparable to that of the division array.

5 Combined Square Root and Divide

Similarities between the radix 2 division and square root algorithms are apparent. It is possible to combine the recurrences for division and square root as:

$$Z_j^*=2Z_{j-1}+\alpha\cdot E\ ,\quad \alpha=\begin{cases} 1 & \text{if } v_j\in\{-1,0\} \\ -1 & \text{if } v_j\in\{1,0\} \end{cases}$$
$$Z_j=\begin{cases} Z_j^* & \text{if } v_j=-1 \text{ or } 1 \\ 2Z_{j-1} & \text{if } v_j=0 \end{cases} \qquad (17)$$

where v_j is either the jth quotient digit q_j or the (j+2)th square root digit s_{j+2} , as appropriate, and E is the result digit extractor given by:

$$E=\begin{cases} S_{j-1}+s_j2^{-j-1} & \text{if square root} \\ D & \text{if division} \end{cases} \qquad (18)$$

Assuming a signed binary residual and $|Z_j| < ½$ then the generation of result digits and control signals is as described in Table 1. The corresponding circuit is straightforward to derive.

6 Discussion

The development of new SRT-type algorithms and architectures for radix 2 division and square root has been described. The premise of the approach has been that concurrently computing the residual and the result digit at each step leads to an increase in the performance of the circuits compared to the SRT methods. The penalty of the new methods appears in the reduced range of the operands.

There are several features which make the algorithms interesting and should lead to an increase in speed over other SRT methods, namely, the concurrent computation of the residual and the quotient digit, the overlapping of computations in adjacent rows and the avoidance of redundancy overflow in the residual word. A totally objective assessment of the new algorithms and architectures and their benefits compared with others requires detailed circuit designs to be undertaken. This is necessary to examine practical VLSI design issues such as buffering, broadcasting, fan-in, fan-out etc. This is something which often appears to be ignored in the development of arithmetic algorithms. However, this is a key issue in systems which require the direct implementation of such algorithms in silicon.

The algorithm for square root, while being of theoretical interest, may have limited application in practice. For floating point arithmetic, the normalized range $[¼,½)$ introduces a post-scaling overhead when the exponent is odd namely the multiplication of the computed square root by $\sqrt{2}$. This problem does not arise in the classical radix 2 SRT square root. If the range of the radicand in the new square root algorithm was extended to $[¼,1)$ then the worst case overlap of adjacent selection regions would be zero thereby requiring full precision examination of the residual.

The arrays presented are very regular and require only a small number of simple cells, each comprising 20 to 30 logic gate equivalents. Extending the arrays for any operand precision is straightforward. Pipelining will increase the throughput rate at the expense of circuit latency. The throughput will, however, be independent of the operand precision.

VLSI implementations of existing division circuits and the division array described here are to be undertaken to objectively assess the benefits of the new division method. These will be reported at a later date.

Acknowledgements: This work has been supported by DRA Malvern.

7 References

[1] J.E. Robertson, 'A new class of digital division methods', *IRE Trans. on Elect. Compt.*, vol. EC-7, 1958, pp218-222.

[2] D.E. Atkins, 'Introduction to the Role of Redundancy in Computer Arithmetic', *IEEE Computer*, 1975, pp74-77.

[3] A.Avizienis, 'Signed Digit Number Representations for Fast Parallel Arithmetic', *IRE Trans. on Compt*, Vol. EC-10, 1961, pp389-400.

[4] S.Majerski, 'Square Rooting Algorithms for High Speed Digital Circuits', *IEEE Trans. on Compt.*, Vol. C-34, 1985, pp724-733.

[5] M.D. Ercegovac, T. Lang, 'On-the-fly conversion of redundant into conventional representations', *IEEE Trans. on Compt.*, Vol. C-36, No. 7, 1987, pp895-897.

[6] M.D. Ercegovac, T. Lang, 'On-line Arithmetic: A Design Methodology and Applications', *VLSI Signal Processing III*, 1988, pp252-263.

[7] A. Svoboda, 'An Algorithm for Division', *Information Processing Machines*, No.9, 1963, pp25-34.

[8] C. Tung, 'A Division Algorithm for Signed-Digit Arithmetic', *IEEE Trans. on Compts. (Short Notes)*, Vol. c-17, 1968, pp887-889.

[9] N. Burgess, 'A Fast Division Algorithm for VLSI', *IEEE Intl. Conf. on Compt. Design*, 1991, pp560-563.

[10] M.D. Ercegovac, T. Lang, 'Fast Radix-2 Division With Quotient-Digit Prediction', *Journal of VLSI Signal Processing*, 1, 1989, pp169-180.

[11] 'IEEE Standard for Floating Point Arithmetic', IEEE Standard 754, *IEEE Computer Society*, 1985.

[12] K. Hwang, Computer Arithmetic: Principles and Design, *J. Wiley and Sons*, 1979.

[13] M. Lapointe, P. Fortier, H.T. Huynh, 'A very fast realization of a time-domain block LMS filter', Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, 1991, pp2101-2104.

[14] S. Kuninobu, T. Nishiyama, T. Taniguchi, N. Takagi, 'Design of High Speed MOS Multiplier and Divider using Redundant Binary Representation', *Proc. 8th Symp. on Compt. Arith.*, 1987, pp80-86.

[15] S.E. McQuillan, 'Algorithms and Architectures for High Performance Arithmetic Processors', Ph.D. Thesis, The Queen's University of Belfast, 1992.