Radix-8 New Svobota-Tung Divider with Carry Free Characteristic for Precsaling

Fun Ye*, Chih-Hao Kuei and Jen-Shiun Chiang

Department of Electrical Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.

Abstract

In recent years, computer applications have increased in the computational complexity. The speed requirement forces designers of general-purpose microprocessors to pay particular attention to implement the floating point unit (FPU). A new floating-point division architecture that complies with the IEEE 754-1985 standard is proposed in this paper. This architecture is based on the New Svoboda-Tung (NST) division algorithm and radix-8 MROR (maximally redundant optimally recoded) signed digit number system. In NST division, the dividend and divisor must be prescaled. For the divider implementation, a signed digit adder with carry free characteristic is proposed for addition and subtraction, and this adder can improve the cycle time significantly. A radix-8 MROR divider by TSMC 0.25 μm technology is thus designed and simulated. The simulation results show that the performance, hardware cost, and power consumption of the proposed divider is competitive to the conventional SRT divider.

Key Words: New Svoboda-Tung Division, Floating-Point Division, Prescaling, Radix-8, Signed Digit Number System

1. Introduction

For computer arithmetic division the most common format found in modern computers is the IEEE 754-1985 standard for binary floating point arithmetic [1]. This standard defines single and double precision formats. Most implementations of the division operation are based on digit-recurrence that one digit of the quotient is produced per iteration. There are two main classes of algorithms for digit-recurrence division, the SRT (Sweeney, Robertson, and Tocher) approach [2,3] and the ST (Svoboda and Tung) approach [4,5].

A lot of efforts have been expended in improving the design and implementation of SRT based dividers [6–11]. SRT stands for the initials of Sweeney, Robertson, and Tocher. They developed the algorithm independently at approximately the same time. The SRT division procedure iteratively employs the P-D plot [12] to determine the next quotient digit and an adder is used to update the partial remainder. The P-D plot approach is usually implemented by table look-up. However, the table size increases drastically with high radices. The Pentium FDIV bug is the most famous story of the Intel microprocessor bugs. It was caused by an error in a look-up table. Intel decided to use the SRT algorithm that can generate two quotient digits per clock cycle. The weakness is that the SRT algorithm needs to store quotient digit derived from a P-D plot. This look-up table was incorrectly entered into the Pentium FPU. Many researchers have presented several classes of SRT dividers that consider variations of the P-D plot, but the complexity of the P-D plot makes it difficultly to implement high-radix SRT dividers [13].

On the other hand, there are no P-D plot problems in Svoboda-Tung algorithm. However, Svoboda-Tung presents several drawbacks:

(1) As the Svoboda-Tung division theorem accurately points out, the radix should be greater than 4.

(2) Without compensation, the quotient digit may ex-

*Corresponding author. E-mail: fyee@mail.tku.edu.tw

ceed the radix base of the signed digit-set.

(3) Compared to the conventional high radix SRT division, the Svoboda-Tung division needs prescaling.

In order to overcome the drawbacks of ST division, in 1998, Montalvo et al. [14] presented the New Svoboda-Tung division algorithm, and it showed a way to rectify the original defect in the NST algorithm. They define a new radix to avoid drawbacks (1) and (2) of the ST division algorithm.

The iterative division can be implemented either using the NST approach that requires prescaling of the operands or the SRT approach that does not require prescaling. For NST the prescaling of dividend and divisor is needed so as to transform the divisor range from [1,2) to [1,1+δ), where δ is a fraction. In this work we summarize a general systematic method to accomplish the prescaling, and we also propose a hardware scheme such that the timing complexity is constant regardless of the bit length of the divisor. The delay of this prescaling architecture is constant, i.e. $O(1)$.

Based on [14], we propose a new radix-8 division by the MROR (maximally redundant optimally recoded) algorithm. Our division approach involves a simple recurrence with carry free addition and employs prescaling of the operands. The input operands comply with the IEEE 754-1985 floating-point standard. The result of the experiment shows that the performance of this divider is competitive to a conventional SRT divider.

The rest of this paper is organized as follows: Section 2 briefly reviews the digit recurrence. The New Svoboda-Tung division algorithm is described in Section 3. The prescaling scheme is described in Section 4. Section 5 analyzes the detailed architecture of the proposed NST radix-8 MROR divider. The comparison results of speed, area, and power dissipation are presented in Section 6. Finally, we make a concluding remark in Section 7.

2. Digit Recurrence Overview

The New Svoboda-Tung division takes its basis as the most basic division expression. The basic recursive division expression is as follows:

$$X = Q \times Y + R$$

where X, Y, Q and R, represent the dividend, divisor, quotient, and remainder, respectively. Division instructions are executed in most of today's digital computers via a digit-recurrence procedure. The time required for the digital division is spent primarily in repeating execution of this recursive procedure. Dividend X and divisor Y are normalized IEEE numbers with a range of $1 \le X, Y < 2$. The recursive expression of the NST division is defined as follows:

$$R^{(j+1)} = b \times R^{(j)} - q_{j+1} \times Y \tag{1}$$

The simplest and most widely implemented class of division algorithms is digit recurrence. Digit recurrence algorithms use subtractive methods to calculate quotients one digit in each iteration. Implementation of digit recurrence algorithms is typically of low complexity. In equation (1), $R^{(j)}$ is the current partial remainder; $R^{(j+1)}$ is the next partial remainder; b is the radix; q_{j+1} is the next quotient digit, and Y is the divisor. The dividend (or initial partial reminder) is $R^{(0)}$. Without loss of generality, we assume that both the dividend $R^{(0)}$ and divisor Y are fractions and so is the generated quotient Q,

$$Q = q_0 q_1 q_2 q_3 \ldots q_{n-1} q_n \tag{2}$$

The division procedure can be verified by applying the recursive equality (1) repeatedly.

Iteration 1: For $j = 0$, $R^{(1)} = b \times R^{(0)} - q_1 \times Y$

Iteration 2: For $j = 1$, $R^{(2)} = b \times R^{(1)} - q_2 \times Y$
$= b^2 \times R^{(0)} - (b \times q_1 + q_2)Y$

……

Iteration n: For $j = n\text{-}1$, $R^{(n)} = b^n \times R^{(0)}$
$- (b^{n-1} \times q_1 + \ldots + b \times q_{n-1} + q_n) \times Y$

3. NST Division Algorithm

In NST, the signed digit-set $D_{<b,\alpha>} = \{-\alpha, \ldots, -1, 0, 1, \ldots, \alpha\}$ such that $\frac{b}{2} \le \alpha \le b - 1$, where b is the radix, and α is the maximum digit in the balanced signed digit-set $D_{<b,\alpha>}$. It is used to represent the remainder $R^{(j+1)}$ and the quotient Q. The conventional digit-set, $D_{<b,\alpha>} = \{0, 1, \ldots, \alpha\}$, is used to represent the divisor Y.

In 1998, Montalvo [14] derived the range of divisor

Y in the NST algorithm and defined the range of divisor Y as follows:

$$1 \le Y < 1 + \frac{\alpha - \beta}{b \times t} \tag{3}$$

In equation (3), the strictest range is $t = \alpha$, and β *i*s a positive integer. Hence, the range of divisor Y in the NST algorithm becomes:

$$1 \le Y < 1 + \delta, \tag{4}$$

where $\delta = \frac{\alpha - \beta}{b \times t}$, and $\beta < \alpha$.

The NST algorithm is valid if and only if $\beta < \alpha$. Hence, if r_1^j and r_2^j have different signs, then r_2^j must meet the condition $|r_2^j| = \beta < \alpha$. If this condition does not meet, r_1^j and r_2^j must be recoded as r_{1a}^j and r_{2a}^j respectively to fulfill the following expression:

$$b \times r_1^j + r_2^j = b \times r_{1a}^j + r_{2a}^j$$

The recoded condition resolves into the following two cases.

Case 1: if r_1^j is positive but not zero, then r_{1a}^j is set to $r_{1a}^j = r_1^j - 1$, and $r_{2a}^j = r_2^j + b$.

Case 2: if r_1^j is negative but not zero, then r_{1a}^j is set to $r_{1a}^j = r_1^j + 1$, and $r_{2a}^j = r_2^j - b$.

As mentioned above, the purpose of recoding the two most significant digits of the residual is to ensure $r_0^{j+1} = 0$ [14].

In the NST algorithm, the range of the divisor Y is $1 \le Y < 1+\delta$. However, in the IEEE 754 floating point standard, the range of the divisor Y is $1 \le Y < 2$. We need a prescaling procedure to transform the divisor Y and dividend $R^{(0)}$ in the NST division. In other words, the original division defined by $X_{std} = Q \times Y_{std} + R$, where X_{std} and Y_{std} are IEEE 754 normalized significant, is replaced by the equivalent division defined by $K \times X_{std} = Q \times (K \times Y_{std}) + K \times R$, where K, $K \times Y_{std}$, and $K \times X_{std}$ are the scaling factor, the scaled divisor Y, and the scaled dividend X, respectively.

Montalvo classified the NST algorithm into four categories: MRMR (Maximally Redundant Maximally Recoded) algorithm, MROR (Maximally Redundant Optimally Recoded) algorithm, MRmr (Maximally Redundant minimally recoded) algorithm, and mr (minimally redundant) algorithm. The algorithms of the four categories are analyzed according to the possible digit-sets and recoded conditions. According to the range of the numerical values α, β and δ, we summarize the four algorithms in Table 1.

The NST division algorithm consists of two steps: Firstly, the input operands of the dividend and the divisor must be prescaled; secondly, the NST algorithm performs division recursion with partial reminder and quotient selection computation. In general, the basic procedure of the NST division algorithm can be summarized as follows:

(1) Prescale input operands of dividend $R^{(0)}$ and divisor Y such that $1 \le Y < 1 + \frac{\alpha - \beta}{b \times t}$.
(2) Select the most significant digit of the partial remainder $R^{(j)}$ as the $(j+1)$th quotient digit.
(3) Execute digit-recurrence $R^{(j+1)} = b \times R^{(j)} - q_{j+1} \times Y$.
(4) Recode partial reminder $R^{(j+1)}$.
(5) Repeat steps (2)~(4) till the division iteration is complete.

4. Prescaling Scheme of the NST Division

In the NST algorithm, the range of divisor Y is defined as $1 \le Y < 1+\delta$, where $\delta = \frac{\alpha - \beta}{b \times \alpha}$. After calculation δ is 1/14. Therefore, for a NST radix-8 division, we have to define the range of the divisor for the input in between

Table 1. δ for the special algorithms [14]

			$\delta = (\alpha - \beta)/(b \times \alpha)$					
Algorithm	α	β	Radix-b	Radix-2	Radix-4	Radix-8	Radix-16	Radix-32
MRMR	$b-1$	0	$1/b$	1/2	1/4	1/8	1/16	1/32
MROR	$b-1$	$b/2-1$	$1/2(b-1)$	1/2	1/6	1/14	1/30	1/62
MRmr	$b-1$	$b-2$	$1/b(b-1)$	1/2	1/12	1/56	1/240	1/992
mr	$b/2$	$b/2-1$	$2/b^2$	1/2	1/8	1/32	1/128	1/512

1 and 15/14, $1 \le Y < 15/14$. Since the IEEE 754 standard specifies a normalized range for the operands to be $1 \le X, Y < 2$, dividend X and divisor Y must be multiplied by a scaling factor K, such that $1 \le Y < 2$ becomes $1 \le Y \times K < 1+\delta$. The range of divisor Y is $1 \le Y < 2$, so the range of K is $1/2 \le K < 1$.

The prescaling unit is to find K corresponding to each segment of the divisor Y, and to multiply each K by dividend X and divisor Y. The design procedure of the prescaling unit is as follows:

(1) Set $Y = 1+d$ and $K = 1-e$, where $0 < d < 1$ and $0 < K < 1/2$. Because $1 \le Y \times K < 1+\delta$, Y and K are substituted to find:

$$1 \le (1-e) \times (1+d) < 1+\delta. \tag{5}$$

(2) Accumulate inequality (5) to obtain

$$\frac{e}{1-e} \le d < \frac{\delta+e}{1-e} \tag{6}$$

To simplify the hardware of the prescaling unit, δ, d, and e in expression (5) can be replaced by the approximate values. These approximate values are comprised by 2^{-h}, where h is an integer. The following procedures are used to find the values of d, e, and K in each range:

(a) Find the approximate value of δ, δ_t, where $\delta_t = 2^{-s} \le \delta$ and $s = [\log_2(1/\delta)]$.

(b) Find e_i and K_i by $e_i = i \times \delta_t/2$ and $K_i = 1 - i \times 2^{-(s+1)}$ respectively, where $0 \le i < 1/\delta_t$, in different segments.

(c) Use expression (6) and e_i to find the range of d_i.

The value d_i found in procedure (c) must satisfy $d_i = m \times 2^{-p}$, where m and p are integers.

In the NST radix-8 division, the parameters in the prescaling unit are listed as follows:

(a) $\alpha = b - 1 = 8 - 1 = 7$

(b) $\beta = (b/2) - 1 = 4 - 1 = 3$

(c) $1 \le Y_s < 1+\delta$

(d) $\delta = \dfrac{\alpha - \beta}{b \times \alpha} = \dfrac{7-3}{8 \times 7} = 1/14$

(e) $s = \log_2(1/\delta) = \log_2 14 = \log 14/\log 2 = (\log 2 + \log 7)/\log 2 = 3.8 \fallingdotseq 4$

(f) $\delta_t = 2^{-s} = 2^{-4}$

According to $0 \le i < 16$ and $K_i = 1 - i \times 2^{-(s+1)}$, we can find K_i of different prescales. The scaling factors are listed in Table 2, and the relationship of range Y, scaling factor K, and transition point d_i is graphically shown in Figure 1. The detailed architecture of our proposed prescaling scheme is discussed in next section.

5. Architecture of the NST Radix-8 MROR Divider

The proposed divider includes three sub circuits, prescaling, iteration, and registers, as shown in Figure 2. The digit set of the radix-8 MROR signed digit number system is $D_{<8,7>} = \{\overline{7}, \overline{6}, \overline{5}, \overline{4}, \overline{3}, \overline{2}, \overline{1}, 0, 1, 2, 3, 4, 5, 6, 7\}$. In order to represent a number in radix-8 signed digit format, it requires a sign bit and three magnitude bits, and totally 4 bits are used to represent a digit. The interpretation of these digits is shown in Table 3. In NST division, prescaling is to adjust the numerical values of the dividend and the divisor. Only once the adjustment has been executed that the dividend and the divisor correspond to the input operands of NST, then the partial remainder $R^{(j+1)}$ and the divisor Y in the recursive equation can be selected correctly. Only after selecting the correct partial remainder and divisor, the divider is able to do the operation in the iteration unit, $R^{(j+1)} = b \times R^{(j)} - q_{j+1} \times Y$.

A. Prescaling

Although the meaning of "prescale" is to multiply a scaling factor K, in actual hardware we use an addition/subtraction operation to replace the multiplication op-

Table 2. K scaling factor for radix-8 numbers

Range of Y	K	K in binary
$1 \le Y < 17/16$	1	1/2 + 1/2
$17/16 \le Y < 35/32$	0.96875	1/2 + 1/2 - 1/32
$32/35 \le Y < 9/8$	0.93750	1/2 + 1/4 + 1/8 + 1/16
$9/8 \le Y < 37/32$	0.90625	1/2 + 1/4 + 1/8 + 1/32
$37/32 \le Y < 19/16$	0.87500	1/2 + 1/4 + 1/8
$19/16 \le Y < 5/4$	0.84375	1/2 + 1/4 + 1/8 - 1/32
$5/4 \le Y < 41/32$	0.81250	1/2 + 1/4 + 1/16
$41/32 \le Y < 43/32$	0.78125	1/2 + 1/4 + 1/32
$43/32 \le Y < 45/32$	0.75000	1/2 + 1/4
$45/32 \le Y < 47/32$	0.71875	1/2 + 1/4 - 1/32
$47/32 \le Y < 49/32$	0.68750	1/2 + 1/8 + 1/16
$49/32 \le Y < 51/32$	0.65625	1/2 + 1/8 + 1/32
$51/32 \le Y < 27/16$	0.62500	1/2 + 1/8
$27/16 \le Y < 57/32$	0.59375	1/2 + 1/16 + 1/32
$57/32 \le Y < 15/8$	0.56250	1/2 + 1/16
$15/8 \le Y < 2$	0.53125	1/2 + 1/32

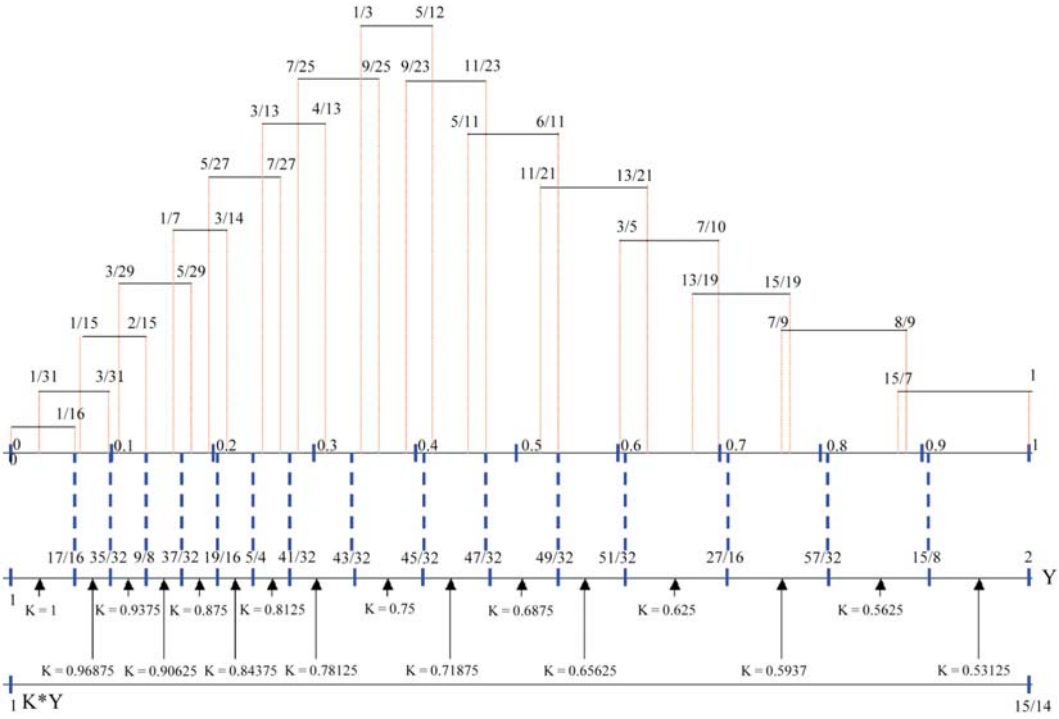

Figure 1. The relationship of range Y, factor K, and transition point d_i.

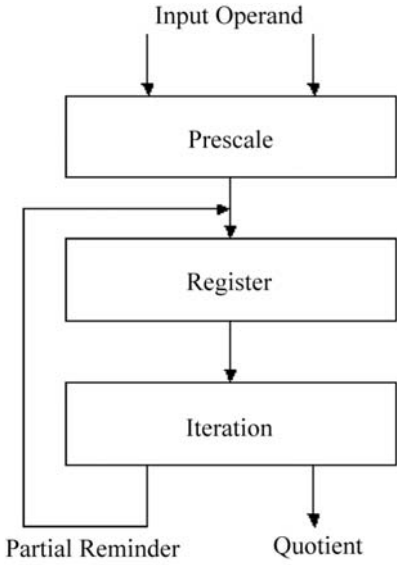

Figure 2. The operation flow of the proposed divider.

Table 3. The conversion table of Radix-8 MROR digit set

Digit number	Binary coding
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
$\overline{7}$	1001
$\overline{6}$	1010
$\overline{5}$	1011
$\overline{4}$	1100
$\overline{3}$	1101
$\overline{2}$	1110
$\overline{1}$	1111

eration. It is to facilitate the addition/subtraction operation to arrange the scaling factor K into various combinations of 2^n in the previous calculation. According to Table 2, Y is divided into 16 intervals, and each interval has the related scaling factor K. The first five most significant bits of the divisor Y can decide the interval location of the divisor Y and the details are shown in Table 4. In other words it decides the range of the input through $Y(n\text{-}1)$~$Y(n\text{-}5)$, and then decides the scaling factor K, and finally conducts addition and subtraction operations on the data selected by the multiplexer. The prescaling architecture of the prescaling block is shown in Figure 3.

B. Iteration

The iteration unit must satisfy the recurrence $R^{(j+1)} = b \times R^{(j)} - r_{1a}{}^{j} \times Y$. The iteration unit is composed of

Table 4. Deciding table for the divisor Y intervals

Range of Y	Transition point d_i	d_i value	d_i in binary $b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}$ $2^{-1}2^{-2}2^{-3}2^{-4}2^{-5}$
$1 \le Y < 17/16$	$0 \le d_0 \le 1/16$	$d_0 = 1/16$	00010
$17/16 \le Y < 35/32$	$1/31 \le d_1 \le 3/31$	$d_1 = 3/32$	00011
$32/35 \le Y < 9/8$	$1/15 \le d_2 \le 2/15$	$d_2 = 1/8$	00100
$9/8 \le Y < 37/32$	$3/29 \le d_3 \le 5/29$	$d_3 = 5/32$	00101
$37/32 \le Y < 19/16$	$1/7 \le d_4 \le 3/14$	$d_4 = 3/16$	00111
$19/16 \le Y < 5/4$	$5/27 \le d_5 \le 7/27$	$d_5 = 8/32$	01000
$5/4 \le Y < 41/32$	$3/13 \le d_6 \le 4/13$	$d_6 = 9/32$	01001
$41/32 \le Y < 43/32$	$7/25 \le d_7 \le 9/25$	$d_7 = 11/32$	01011
$43/32 \le Y < 45/32$	$1/3 \le d_8 \le 5/12$	$d_8 = 13/32$	01101
$45/32 \le Y < 47/32$	$9/23 \le d_9 \le 11/23$	$d_9 = 15/32$	01111
$47/32 \le Y < 49/32$	$5/11 \le d_{10} \le 6/11$	$d_{10} = 17/32$	10001
$49/32 \le Y < 51/32$	$11/21 \le d_{11} \le 13/21$	$d_{11} = 19/32$	10011
$51/32 \le Y < 27/16$	$3/5 \le d_{12} \le 7/10$	$d_{12} = 11/16$	10110
$27/16 \le Y < 57/32$	$13/19 \le d_{13} \le 15/19$	$d_{13} = 25/321$	11001
$57/32 \le Y < 15/81$	$7/9 \le d_{14} \le 8/9$	$d_{14} = 7/8$	11100
$5/8 \le Y < 2$	$15/17 \le d_{15} \le 1$	$d_{15} = 1$	11111

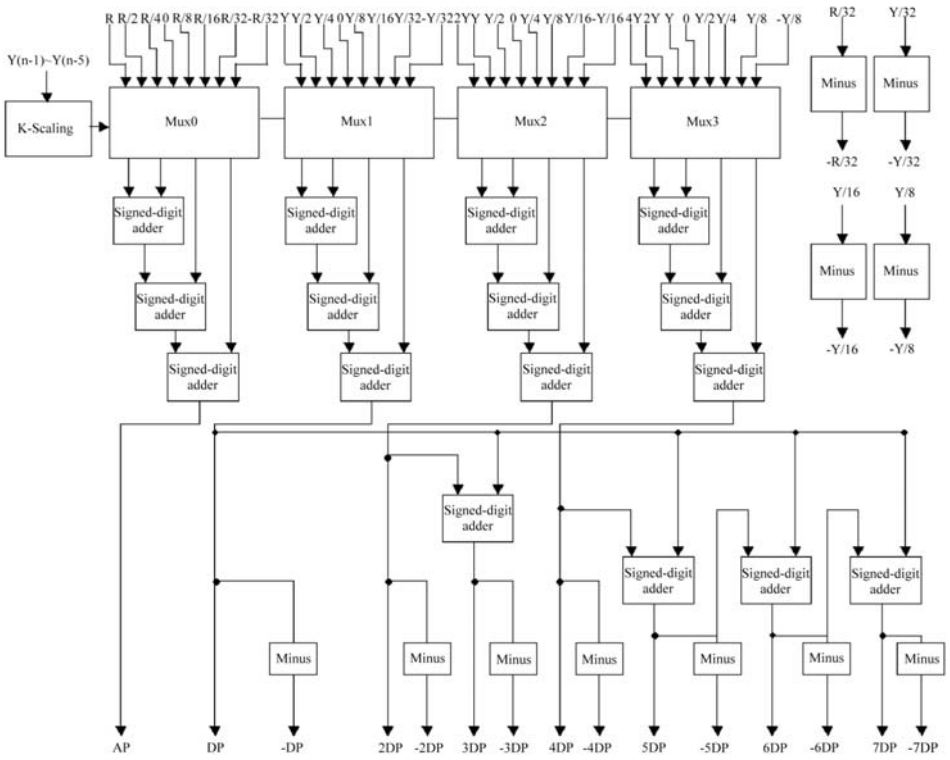

Figure 3. Architecture of the prescaling block.

recoded unit, shifter unit, compensation unit, signed digit adder, and quotient converter as shown in Figure 4. We can use an on-the-fly converter to convert the redundant quotient to binary form [14]. The detailed description of the various units is described in the following subsections.

C. Recoded Unit

The recoded unit can be implemented through simple logic circuits, and the design process is aimed to find

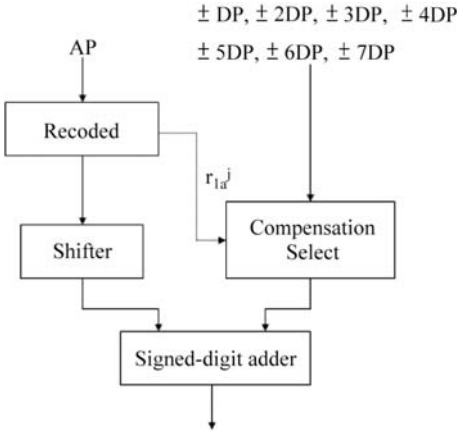

Figure 4. Architecture of the iteration block.

regularity in the recoded conditions. The logic diagram of the recoded unit is shown in Figure 5.

D. Shifter Unit

This unit implements the $b \times R^{(j)}$ operation of the recurrence, and in a radix-8 system it is to implement $8 \times R^{(j)}$. In the hardware implementation, it can simply be implemented through the shifter circuit.

E. Compensation Unit

This unit mainly implements $r_{1a}^{\ j} \times Y$ in the recurrence. This unit can be implemented by multiplexers.

F. Signed-Digit Adder

A new radix-8 MROR signed digit adder with carry free characteristic is proposed in this work. Usually, in conventional computers the addition is performed by employing the carry propagation technique, which essentially limits the computing speed. In order to eliminate the carry propagation and thereby speed up the addition process, the carry out signal should be independent of the carry in value. The logic diagram of the proposed signed-digit adder is shown in Figure 6.

The truth table of the final sum is shown in Table 5. Each sub circuit of the signed-digit adder described in Figure 6 represents a logic function. This partitioning allows the overall circuit to be easily expressed as follows:

$$\text{carry1} = (a_{i,s} \cdot b_{i,s})$$

$$\text{carrys1} = [\overline{(a_{i,s} \cdot b_{i,s})} \cdot i_{c2}] + [a_{i,s} \cdot b_{i,s} \cdot \overline{i_{c2}}] + [(a_{i,s} \cdot b_{i,s} \cdot i_{c2}) \cdot \overline{(a_{i,s} + b_{i,s} + i_{c2})}]$$

$$\text{its} = [(a_{i,s} \cdot b_{i,s}) \cdot (i_{s2} + i_{s1} + i_{s0})] + [\overline{i_{c2}} \cdot (a_{i,s} \oplus b_{i,s})]$$

$$\text{carrys2} = (i_{ts} \cdot \overline{i_{ts2}} \cdot \overline{i_{s1}} \cdot i_{s0})$$

$$\text{carry2} = (\overline{i_{ts}} \cdot i_{s2} \cdot i_{s1} \cdot i_{s0}) + (i_{ts} \cdot \overline{i_{s2}} \cdot \overline{i_{s1}} \cdot i_{s0}) = (\overline{i_{ts}} \cdot i_{s2} \cdot i_{s1} \cdot i_{s0}) + carrys\ 2$$

When performing the addition operation, the signed digit adder in [15], apart from generating the final sum,

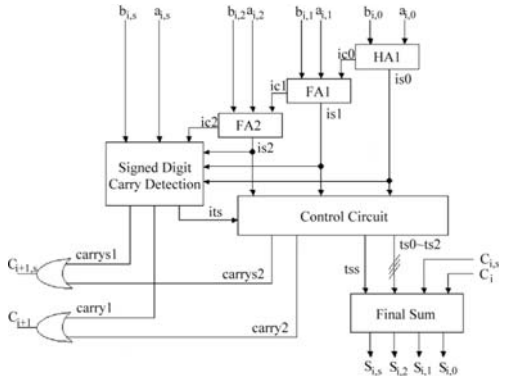

Figure 6. Signed-digit adder diagram.

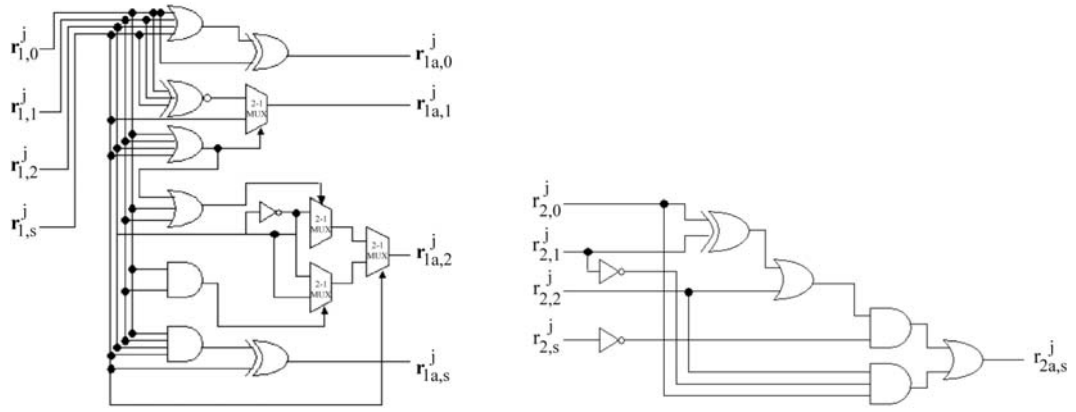

Figure 5. Recoded unit.

Table 5. Truth table of the final sum block

$C_{i-1,s} = 0$								$C_{i-1} = 1$							
tss	ts2	ts1	ts0	$S_{i,s}$	$S_{i,2}$	$S_{i,1}$	$S_{i,0}$	tss	ts2	ts1	ts0	$S_{i,s}$	$S_{i,2}$	$S_{i,1}$	$S_{i,0}$
1	0	1	0	1	0	1	1	0	0	0	1	0	0	1	0
1	0	1	1	1	1	0	0	0	0	1	0	0	0	1	1
1	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0
1	1	0	1	1	1	1	0	0	1	0	0	0	1	0	1
1	1	1	0	1	1	1	1	0	1	0	1	0	1	1	0
1	1	1	1	0	0	0	0	0	1	1	0	0	1	1	1
0	0	0	0	0	0	0	1								

$C_{i-1,s} = 1$								$C_{i-1} = 1$							
tss	ts2	ts1	ts0	$S_{i,s}$	$S_{i,2}$	$S_{i,1}$	$S_{i,0}$	tss	ts2	ts1	ts0	$S_{i,s}$	$S_{i,2}$	$S_{i,1}$	$S_{i,0}$
1	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	1
1	1	0	0	1	0	1	1	0	0	1	1	0	0	1	0
1	1	0	1	1	1	0	0	0	1	0	0	0	0	1	1
1	1	1	0	1	1	0	1	0	1	0	1	0	1	0	0
1	1	1	1	1	1	1	0	0	1	1	0	0	1	0	1
0	0	0	0	1	1	1	1								

also generates a control signal for the next level. The logic circuit for this adder is composed of two parts: a full-adder circuit and a sign and carry control circuit. The number of logic gates needed by the full-adder circuit is $2 \times (n+1)A_{FA}$, where A_{FA} is the area of a full adder. The second part, sign and carry control circuit, needs 24 logic gates. Therefore, the adder unit needs a total of 8 full adders and one sign and carry control circuit for the radix-8 number system. 8 full adders require 40 logic gates; the sign and carry control logic circuit needs 24 gates, and totally 64 logic gates are needed to comprise the adder unit in [15]. However, the proposed signed-digit adder only needs 52 logic gates, and in comparison with the area of the adder circuits in [15], ours makes a reduction of about 1/4.

6. The Design and Simulation of a Radix-8 Divider

Based on the proposed NST MROR radix-8 division approach, a NST radix-8 divider was designed and simulated by TSMC 1P4M 0.25 μm process. The performance of the designed divider is summarized in Table 6.

A comparison of the computation time of several divider designs is presented in [9]. The references do not consider quotient conversion time, and therefore the final quotient conversion time is ignored here. Table 7 is

Table 6. Characteristics of the designed divider

Process Technology	TSMC 1P4M 0.25 μm	
Voltage, Temperature	2.5 V, 25 °C	
Gate Count	Single precision	22780.6
	Double precision	30987.3
Number of Cycles	Single precision	11
	Double precision	21
Cycle Time (ns)	Single precision	6.01 ns
	Double precision	6.12 ns
Power Consumption (mW)	Single precision	97.4
	Double precision	138

Table 7. Speedup achieved by the proposed design over other designs (Single precision)

Single precision		
Design	Radix	Speedup
Proposed Divider	8	1
Prescaled NST MROR [17,29]	4	1.5
Prescaled NST MROR [9]	4	1.8
Regular [10]	4	2.1
Prescaled regular [18]	4	2.25
Prescaled over redundant (i) [19]	4	2.16
Prescaled over redundant (ii) [19]	4	2.05
Trade-off SRT [20]	4	1.5
Prescaled regular [21]	2	2.2
Regular (i) [22]	2	2.39
Regular (ii) [23]	2	2.7

the speedup comparison table among other single precision dividers. The performance of the proposed divider is better than others. We also compare the operation time of the proposed divider with other SRT dividers in the double precision system, and the experimental result is shown in Table 8. Gate count comparisons are shown is Table 9. The power consumption comparisons are shown in Table 10 for implementations of radixes 2, 4, 8, and 16, respectively.

7. Conclusion

A NST radix-8 divider complied with the IEEE 754-1985 is presented in this paper. The main architecture of this divider is composed of the prescaling scheme of the dividend and divisor and the signed-digit adder. For prescaling operation, we generate a table for *K* prescaling factors and the prescaling can be easily implemented by simply shifting and adding of the operands. For speed and hardware cost consideration we propose a new signed-digit adder. Compared to the conventional approach, the proposed signed-digit adder can save 1/4 hardware cost. The presented NST radix-8 divider is designed both for single precision and double precision operations by TSMC 1P4M 0.25 μm technology. The simulation results indicate that our divider has the best operation speed in the single precision operation among the SRT dividers, and good performance compared with other SRT dividers in double precision operations. The power consumption is competitive to other dividers.

Table 8. Speedup achieved by the proposed design over other designs (Double precision)

Double precision		
Design	Radix	Speedup
Proposed Divider	8	1
Regular (i) [16]	4	1.63
Regular (ii) [16]	8	1.24
Regular (iii) [16]	16	1.16
SRT Algorithm (i) [24]	4	1.32
SRT Algorithm (ii) [24]	16	0.9375
SRT Parallel Path Algorithm [25]	4	1.32

References

[1] "IEEE Standard for Binary Floating Point Arithmetic," *ANSI/IEEE Standard 754-1985, IEEE Computer Society* (1987).

[2] Robertson, J. E., "A New Class of Digital Division Methods," *IRE Trans. on Electronic Computers,* Vol. 7, pp. 218–222 (1958).

[3] Tocher, T. D., "Techniques of Multiplication and Division for Automatic Binary Computers," *Quarterly J. Mech. App. Math.,* Vol. 2, pp. 364–384 (1958).

[4] Svoboda, A., "An Algorithm for Division," *Information Processing Machines,* Vol. 9, pp. 183–190 (1963).

[5] Tung, C., "A Division Algorithm for Signed-Digit Arithmetic," *IEEE Trans. on Computers,* Vol. 17, pp.

Table 9. Gate count estimation of different NST division designs

Gate Count Estimation			
Design	Radix	Process Technology	Gate count
Proposed (Single precision)	8	TSMC 1P4M 0.25 μm	22780.6
Proposed (Double precision)	8	TSMC 1P4M 0.25 μm	30987.3
Area Efficient FDIV (Single precision) [26]	4	TSMC 0.18 μm	12241
Prescaled NST MROR (Single precision) [17]	4	TSMC 1P3M 0.6 μm	14494.5

Table 10. Power comparison between NST and SRT algorithm

Power Comparison			
Design	Radix	Process Technology	Power
Proposed (Single precision)	8	TSMC 1P4M 0.25 μm	97.4 mW
Proposed (Double precision)	8	TSMC 1P4M 0.25 μm	138 mW
Regular (i) (Double precision) [27]	4	Sun Ultra SPARC processor	108.4 mW
Regular (ii) (Double precision) [27]	8	Sun Ultra SPARC processor	133 mW
Parallel Single Precision [28]	4	TSMC 0.18 μm	154 mW

887–889 (1968).

[6] Oberman, S. F. and Flynn, M. J., "Design Issues in Division and Other Floating-Point Operations," *IEEE Trans. on Computers,* Vol. 46, pp. 154–161 (1997).

[7] Srinivas, H. R., Parhi, K. K. and Montalvo, L. A, "Radix 2 Division with Over-Redundant Quotient Selection," *IEEE Trans. on Computers,* Vol. 46, pp. 85–92 (1997).

[8] Burgess, N., "Prescaled Maximally-Redundant Radix-4 SRT Divider," *Electronics Letters,* Vol. 30, pp. 1926–1928 (1994).

[9] Srinivas, H. R. and Parhi, K. K., "A Fast Radix-4 Division Algorithm and Its Architecture," *IEEE Trans. on Computers,* Vol. 44, pp. 826–831 (1995).

[10] Montuschi, P. and Ciminiera, L., "Design of a Radix-4 Division Unit with Simple Selection Table," *IEEE Trans. on Computers,* Vol. 41, pp. 1606–1611 (1992).

[11] Fenwick, P., "High-Radix Division with Approximate Quotient-digit Estimation," *Journal of Universal Computer Science,* Vol. 1, pp. 2-23, http://www.jucs.org/jucs_1_1/high_radix_division_with/paper.html (1995).

[12] Kantabutra, V., "A New Algorithm for Division in Hardware," in *Proceeding of IEEE International Conference on Computer Design (ICCD),* pp. 551–556 (1996).

[13] Burgess, N. and Williams, T., "Choices of Operand Truncation in the SRT Division Algorithm," *IEEE Transactions on Computers,* Vol. 44, pp. 933–938 (1995).

[14] Montalvo, L. A., Parhi, K. K. and Guyot, A., "New Svoboda-Tung Division," *IEEE Trans. on Computers,* Vol. 47, pp. 1014–1020 (1998).

[15] Mekhallalati, M. C. and Ibrahim, M. K., "New High Radix Maximally-Redundant Signed Digit Adder," *IEEE International Symp. on Circuit and System,* Vol. 1, pp. 459–462 (1999).

[16] Nannarelli, A. and Lang, T., "Low-Power Division: Comparison among Implementation of Radix 4, 8 and 16," *14th IEEE Symp. on Computer Arithmetic,* p. 60 (1999).

[17] Tsai, M.-S., "The Design and Implementation of a High Speed Radix-4 Carry Free Division Architecture," *Master Thesis,* Tamkang University, June (2000).

[18] Ercegovac, M. D. and Lang, T., "Simple Radix-4 Division with Operands Scaling," *IEEE Trans. on Computers,* Vol. 39, pp. 1204–1208 (1990).

[19] Montuschi, P. and Ciminiera, L., "Over-Redundant Digit Sets and the Design of Digit-by-Digit Units," *IEEE Trans. on Computers,* Vol. 43, pp. 269–279 (1994).

[20] Wang, X. and Nelson, B. E., "Tradeoffs of Designing Floating-Point Division and Square Root on Virtex FPGAs," *Proc. 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines,* Vol. 2, pp. 195–203 (2003).

[21] Burgess, N., "A Fast Division Algorithm for VLSI," in Proc. *IEEE Int'l Conf. Computer Design: VLSI in Computers and Processors,* Boston, pp. 560–563 (1991).

[22] Kuninobu, S., Nishiyama, H. E. T., Tanaguchi, T. and Takagi, N., "Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation," *Proc. 8th Symp. Computer Arithmetic,* Como, Italy, pp. 80–86 (1987).

[23] Ercegovax, M. D. and Lang, T., *Division and Square Root.* Norwell, Mass.: Kluwer Academic (1994).

[24] Montuschi, P. and Lang, T., "Boosting Very-High Radix Division with Prescaling and Selection by Rounding," *IEEE Trans. on Computers,* Vol. 50, pp. 13–27 (2001).

[25] Rice, E. and Hughey, R., "A New Iterative Structure for Hardware Division: The Parallel Paths Algorithm," *16th IEEE Symposium on Computer Arithmetic (ARITH'03),* pp. 54–62 (2003).

[26] Moon, J.-S., Kwon, T.-J., Sondeen, J. and Draper, J., "An Area-Efficient Standard-Cell Floating-Point Unit Design for a Processing-In-Memory System," *Proc. of the Conference on European Solid-State Circuits,* pp. 57–60 (2003).

[27] Nannarelli, A. and Lang, T., "Low-Power Radix-8 Divider," *Proc. of International Conference on Computer Design,* pp. 420–426 (1998).

[28] Kwon, T.-J., Moon, J.-S., Sondeen, J. and Draper, J., "A 0.18 μm Implementation of a Floating-Point Unit for a Processing-In-Memory System," *IEEE International Symp. on Circuit and System,* Vol. 2, pp. 453–456 (2004).

[29] Chiang, J.-S. and Tsai, M.-S., "A Radix-4 New Svobota-Tung Divider with Constant Timing Complexity for Prescaling," *Kluwer Journal of VLSI Signal Processing,* Vol. 33, pp. 117–124 (2003).

Manuscript Received: Oct. 11, 2006
Accepted: Oct. 22, 2007