30-ns 55-b Shared Radix 2 Division and Square Root Using a Self-Timed Circuit

Gensoh Matsubara, Nobuhiro Ide, Haruyuki Tago, Seigo Suzuki and Nobuyuki Goto

ULSI Research Laboratories, Research and Development Center, Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan

Abstract

A shared radix 2 division and square root implementation using a self-timed circuit is presented. The same execution time for division and square root is achieved by using an on-the-fly digit decoding and a root multiple generation technique. Most of the hardware is shared, and only several multiplexers are required to exchange a divisor multiple and a root multiple. Moreover, quotient selection logic is accelerated by a new algorithm using a 3-b carry propagation adder. The implementation of the shared division and square root unit is realized by assuming 0.3 μm CMOS technology. The wiring capacitance and other parasitic parameters are taken into account. The execution time of floating point 55-b full mantissa division and square root is expected to be less than 30ns in the worst case of an input vector determined by an intensive circuit simulation.

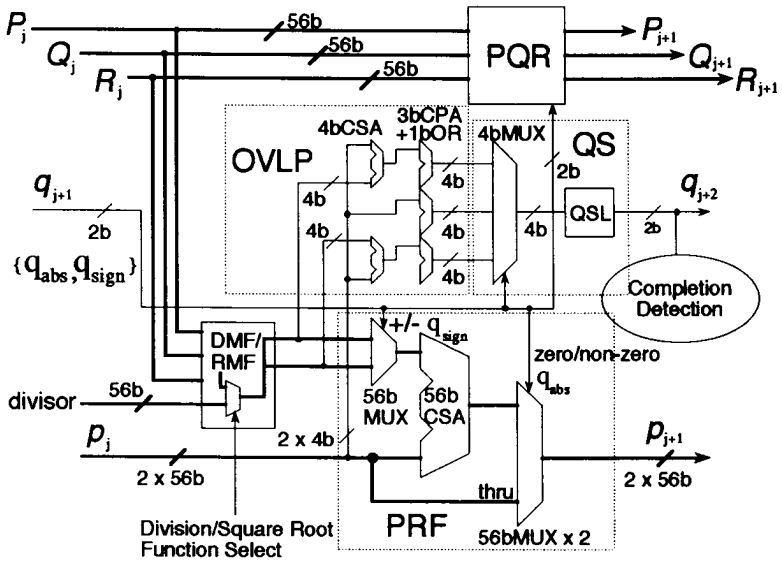
1 Introduction

Calculation time of a division and a square root is fairly long compared with other floating point operations such as a multiplication and an addition. So far, this long delay time of division and square root is suspected not to affect the total performance of computer systems because the appearance ratio between a multiplication and a division is about 4:1 and a square root calculation rarely occurs in ordinary applications. However, as clock frequency of microprocessors increases, the problem of data hazards due to a large number of pipeline stages becomes more prominent. Moreover, some applications such as three-dimensional graphics use division and square root much more frequently than ordinary applications. Thus, a speed-up technique other than improved device technology is needed for these applications. The clock period can be shortened by the faster device technology, but the latency clock count of division or square root remains unchanged.

Several algorithms to achieve a fast division and a square root calculation have been proposed such as the higher radix method, the Newton-Raphson method, and so on [1-8]. One of these speed-up methods, an adaptation of a self-timed circuit to the radix 2 SRT algorithm-based divider, was proposed to investigate a much faster division unit by Williams et al. [6]. A self-timed circuit implementation requires a somewhat larger silicon area in contrast with ordinary SRT algorithm-based systems. An implementation of the Newton-Raphson method also requires an even larger area for a multiplier and a read only memory (ROM). However, these two methods can achieve very high performance for calculations. Consequently, considering factors such as simplicity of the pattern layout, the silicon area, power consumption and other factors, we selected a self-timed circuit implementation for a shared division and square root calculation unit. In this paper, we report an implementation of 55-b shared radix 2 division and square root calculation unit by using a self-timed circuit. To evaluate the performance of the calculation unit quantitatively, 0.3 μm triple metal CMOS technology [9] was assumed so that an intensive circuit simulation could be performed. We also confirmed that the calculation unit could give correct results by a functional simulation using a high-level language.

Section 2 of this paper presents an overview of the calculation unit. The calculation unit uses a fast and simple quotient selection logic (QSL) and on-the-fly digit processing blocks called PQR (position, quotient, reverse quotient) to perform a square root function with a short

1063-6889/95 $4.00 © 1995 IEEE



$P_{j[n]}$ j: iteration index, n: bit position

P: position
Q: quotient (with carry propagation)
R: quotient (without carry propagation)
p: partial remainder/radicand
q: quotient (1 digit; +1,0,-1) represented by q_{abs} and q_{sign}

PQR: on-the-fly digit handling block
QS : quotient selection block
PRF: partial remainder/radicand formation block
OVLP: overlap execution block
DMF : divisor multiple formation
RMF : root multiple formation

Fig. 1 Configuration of the basic unit of the shared division and square root unit.

execution time. In Section 3, we focus on the QSL logic using a 3-b carry propagation adder (CPA) and a 1-b logical OR-gate. This QSL block can be used for both division and square root without any special hardware. The proof of the algorithm for this structure is also shown in Section 3. Section 4 shows an implementation of the root multiple generation block. Unlike the divider, the square root unit uses a result feedback mechanism to calculate a partial radicand. This feature usually affects the calculation speed of a square root. In our implementation, however, the execution time of a square root is the same as that of a division. In Section 5, we show the execution timing of the calculation unit. From the analysis of critical paths, it becomes clear that the execution time of a square root is equal to that of a division. Section 6 presents a comparison between our implementation and alternatives using synchronous circuit implementations is shown. In this section, an additional hardware cost to achieve the shared calculation unit from a division unit is also disscussed.

2 Implementation of Calculation Unit

Figure 1 shows the implementation of each SRT division / square root stage. In our calculation unit, a radix 2 SRT algorithm with the redundant digit representation (+1, 0, -1) is used. The stage consists of a partial remainder / radicand formation block (PRF), a divisor multiple formation block (DMF), a root multiple formation block (RMF), and a quotient selection logic (QSL) with an overlapped execution modules as in [1,6]. In our implementation, the RMF block is modified with an on-the-fly digit handling block (PQR). The PQR block is based on carry propagation considerations [4,5,7]. The signal "P" shows the bit position where the next digit is to be determined. The signal "Q" is the quotient with carry propagation at LSB, and "R" is the quotient without carry propagation. The RMF block requires a nonredundant representation of the quotient digit produced in the PQR block. There is no additional hardware to realize a shared calculation unit except for the divisor / root multiple generation block. Moreover, the delay time of the RMF block does not affect the critical path of the whole calculation unit as shown in the following section. Therefore, this implementation can realize the same latency in both division and square root at the same precision.

Figure 2 shows the block diagram for the whole division / square root unit. This construction uses 5 stages to achieve a zero-overhead self-timed calculation unit as proposed by Williams et al. [6]. The precharge timing controller which consists of a static flip-flop is used for dynamic circuits in each stage. The accumulation of the quotient digit is performed by the PQR block. The end of a calculation can be detected with the P signal. To make an IEEE754-compatible calculation unit, rounding and normalization of quotient digits are performed with the quotient from the Q signals (assumes a carry propagation) and the partial remainder / radicand. This implementation uses a dual-monotonic wire set as in [6]. The logic gate consists of a precharge transistor and an n-channel pull-down network.

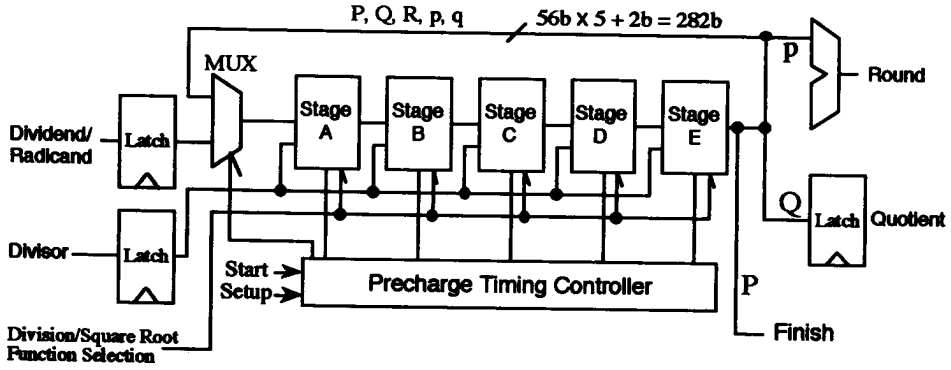


Fig. 2 Configuration of the shared division and square root unit.

Table I Quotient selection table and statistics for the shared radix 2 division and square root unit.

Approximated partial remainder / radicand				Digit selection			Value range of partial remainder / radicand	Statistics	
								Probability	
D_0	D_1	D_2	D_x	digit	q_{abs}	q_{sign}		Division	Square root
0	1	1	X	+1	0	1	[3/2 , 2), [-2, -3/2)	0.1%	0.05%
0	1	0	X	+1	0	1	[1, 2)	1.7%	1.8%
0	0	1	X	+1	0	1	[1/2, 3/2)	8.0%	7.9%
0	0	0	1	+1	0	1	[1/4, 1)	24.2%	25.5%
			0	0	1	1	[0, 1/2)	8.6%	8.1%
1	1	1	X	0	1	0	[-1/2, 1/2)	33.9%	35.0%
1	1	0	X	-1	0	0	[-1, 0)	20.0%	18.8%
1	0	1	X	-1	0	0	[-3/2, -1/2)	3.5%	2.8%
1	0	0	X	-1	0	0	[-2, -1)	0.0%	0.0%

'X' = don't care.

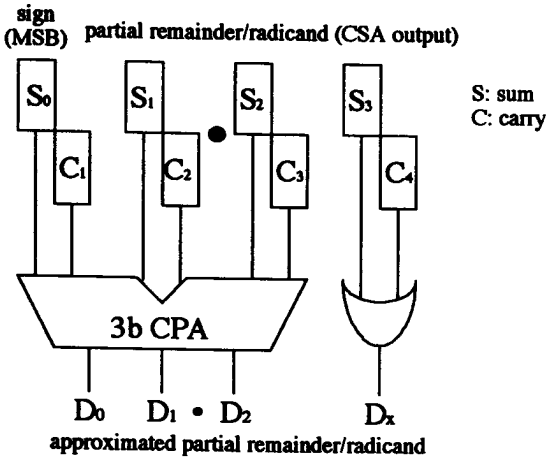


Fig. 3 The proposed implementation of the quotient selection logic (QSL) for the shared radix 2 division and square root unit.

Table II The code assignment for the quotient digit.

digit	q_{abs}	q_{sign}
0	1	X
+1	0	1
-1		0

3 Quotient Selection Logic Implementation

Figure 3 shows the configuration of the QSL block, and Table I shows the quotient selection rule and statistics from the functional simulation. When the digit code is assigned as shown in Table II, the code q_{sign}, which indicates a sign of a digit, is the inverse of D_0 exactly. This allows a simple logic circuit of the QSL, and faster execution of a partial remainder / radicand formation as shown in Fig. 1. Moreover, the probability of selecting a digit = 0 in our QSL is higher than that in a conventional QSL, that is, the probability of selecting a digit = 0 is about 42-43% for our division and square root in contrast with the conventional value of 35%. These figures were determined by our functional simulations. This feature may cause an additional speed-up, because an execution time of a partial remainder / radicand can be shortened when a quotient digit is 0. Besides, as proposed by several authors, the quotient selection logic for radix 2 SRT calculation unit can be formed by using 3-b CPA [3, 6]. The main problem appears when using a 3-b CPA and the result is +1.1 in binary. The QSL cannot discriminate a correct digit in this situation, because a true value of the partial remainder / radicand could lie in the range [+3/2, 2) and [-2, -3/2). This problem can be resolved by determining a latest quotient digit [3] or a latest partial remainder / radicand as in [6]. On the other hand, our design of the QSL does not use any former information. Only a 1-b OR-gate and a related QSL circuit were added. The proof of this algorithm is shown as follows.

(1) Division case

The equation of a partial remainder is defined as

$$p_{j+1} = 2p_j - q_{j+1} d \quad - (1)$$

where the symbols are defined as follows:

p_j j^{th} partial remainder,
q_j j^{th} quotient digit (redundant form),
d divisor.

The divisor d is normalized in [1/2, 1).

The problem occurs when

$$2p_{j+1} \in [-2, -3/2). \quad - (2)$$

Therefore, if $2p_{j+1}$ cannot be in the range [-2, -3/2), the problem has to be resolved. This situation of failure will occur only when a digit $q_{j+1} = 1$. Thus, the situation

$$-1 < 2p_j - d \leqq -3/4 \quad - (3)$$

is not wanted to appear in this case.

Then, the worst case is that $d = 1 - \varepsilon$ (ε is a very small number) from the assumption. From the relation (3), the QSL is not wanted to set the digit to 1 when the relation

$$0 < 2p_j < 1/4 \quad - (4)$$

is fulfilled.

Therefore, when the relation (4) is fulfilled, the QSL replies that a digit = 0 as shown in Table I. The other opportunity for fulfilling the relation (3) is the case that a digit = 0. However, if we assume d = 0, relation (3) becomes

$$-1 < 2p_j \leqq -3/4. \quad - (5)$$

In this case, the QSL always replies that a digit = -1. Then the assumption of d = 0 cannot be allowed.

Consequently, the QSL with a 3-b CPA and a 1-b OR-gate is proven to give correct results.

(2) Square root case

The equation of a partial radicand is defined as

$$p_{j+1} = 2p_j - q_{j+1} (2Q_j + q_{j+1} 2^{-(j+1)}) \quad - (6)$$

where the symbols are defined as follows:

p_j j^{th} partial radicand,
q_j j^{th} quotient digit (redundant form),
Q_j j^{th} quotient (nonredundant form, Q_0=0),

$$Q_{j+1} = Q_j + q_{j+1} 2^{-(j+1)} (j = 0, 1, 2, ...) \quad - (7)$$

The value of Q_j lies in the range [1/2, 1).

The problem occurs only when $q_{j+1} = 1$. Unlike in the case of a division, the QSL examines p_j. Thus, the same as the division case, the relation

$$-2 < 2p_j - (2Q_j + 2^{-(j+1)}) \leqq -3/2 \quad - (8)$$

gives a condition of failure.

On the other hand, the relation

$$1 < 2Q_j + 2 \cdot 2^{-(j+1)} < 2 \quad - (9)$$

is always fulfilled because of the normalization of a given radicand.

From the relations (8) and (9),

$$-1/2 - 2^{-(j+2)} < p_j < 1/4 - 2^{-(j+2)} \quad - (10)$$

is the failure condition when a digit = 1. As Table I clearly indicates, when the QSL selects a digit = 1, the partial radicand p_j does not exist in the range (10).

Finally, the QSL with a 3-b CPA and a 1-b OR-gate is proven to give correct results for both a division and a square root. Thus, when the QSL encounters the +1.1 of partial remainder / radicand, the QSL can always take a digit = +1, because the case of a digit = -1 never occurs in this situation. This feature also can be obtained by using a 4b CPA. However, our implementation obtains these results with a 3-b CPA and without any former information on the quotient selection. This QSL block can realize a fast quotient selection logic with very simple hardware.

4 Root Multiple Generation

To perform a square root calculation, the root multiple calculation must be taken into account. If the quotient Q is expressed by using a redundant digit form, the carry save adder (CSA) must have four inputs for the root multiple and former partial radicand indicated in a redundant form.

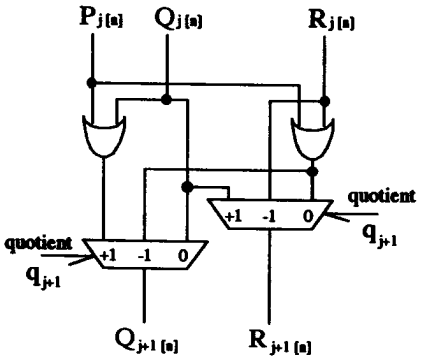


Fig. 4 On-the-fly quotient digit decoding logic.

		P	0000...	0 100 ...
digit	+1	Qnext =	0.QQQQ...	Q 1
		Rnext =	0.QQQQ...	Q 0
	0	Qnext =	0.QQQQ...	Q 0
		Rnext =	0. RRRR...	R 1
	-1	Qnext =	0. RRRR...	R 1
		Rnext =	0. RRRR...	R 0

Fig. 6 Operation of the on-the-fly digit decoding logic.

Thus, the delay time of the partial radicand formation (PRF: also used for partial remainder formation) becomes longer. This additional delay in the PRF affects the total delay of the calculation unit. Moreover, if the QSL is overlapped, it also needs the next PRF for a prediction of a quotient digit. This fact might cause an additional delay.

On the other hand, an on-the-fly digit decoding and a root multiple generation in a nonredundant digit form can be performed as reported in [4,5]. This technique can shorten delay time to make a root multiple digit in a nonredundant form, and merely the 3-input CSA in the PRF is needed. Figure 4 and Figure 5 show our implementation of the on-the-fly digit decoding block and the root multiple generation block respectively. $P_{[n]}$ is a bit position indicator at n^{th} bit position. At an initial state (0^{th} iteration), $P_{[0]} = 1$ and $P_{[n]}$ $(n \neq 0) = 0$. When the iteration proceeds, $P_{[j]} = 1$ and $P_{[k]}$ $(k \neq j) = 0$ in the j^{th} iteration. Thus, P is a kind of a shift register. As mentioned previously, Q and R are the current quotient assuming the carry propagation or without, respectively. At the initial state, Q and R are cleared to 0. Figure 6 shows how this on-the-fly block works. Figure 7 shows how the root multiple generation block works. In the root multiple generation process, a root multiple bit for a positive quotient digit is negated to perform a subtraction at the CSAs in PRF. In addition, a divisor multiple formation

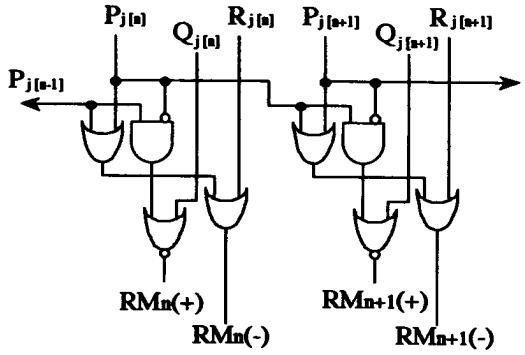


Fig. 5 Root multiple generation logic.

	P 0000...	0 100 ...
$\overline{RM(+)}$	0.QQQQ...	Q 0 1
RM(-)	0. RRRR...	R 1 1

Fig. 7 Operation of the root multiple generation logic.

(DMF) block and a function selector are provided to realize a shared division and square root calculation unit.

5 Calculation Timing

We have mentioned the construction of the shared division and square root unit. In this section, an execution timing of each block is discussed. Figure 8 illustrates the timing of the calculation unit. The lateral length of each rectangle in Fig. 8 means the delay time of each circuit block.

It can easily be seen in Fig. 8 that if the summation of the delay times of the PQR and RMF ($T_{PQR} + T_{RMF}$) is less than that of the PRF (T_{PRF}), the total execution time of a square root is the same as that of a division. In our implementation, this condition can be fulfilled. Namely, the delay time ($T_{PQR} + T_{RMF}$) is almost the same as (T_{PRF}).

Figure 9 shows an overview of the precharge timing control block. This block is controlled by a completion signal from the QSL block. The zero-overlapped execution is realized by using a 5-stage configuration as in [6]. If the perfect timing control is needed, completion signals from the QSL, PQR, and PRF are required. However, as the PQR and PRF block have many output signals, it is difficult to determine all these signals for a completion detection. For this reason, our design neglects a completion-check on these signals. As shown in Fig. 1, the QSL block completes the execution after the arrival of the signal q_{j+1}. The PQR and PRF are also brought to completion after the arrival of the signal q_{j+1}. We designed the circuit to ensure that the completion signal from the QSL block occurs within a certain range of time around the generation of the PQR and PRF signals.

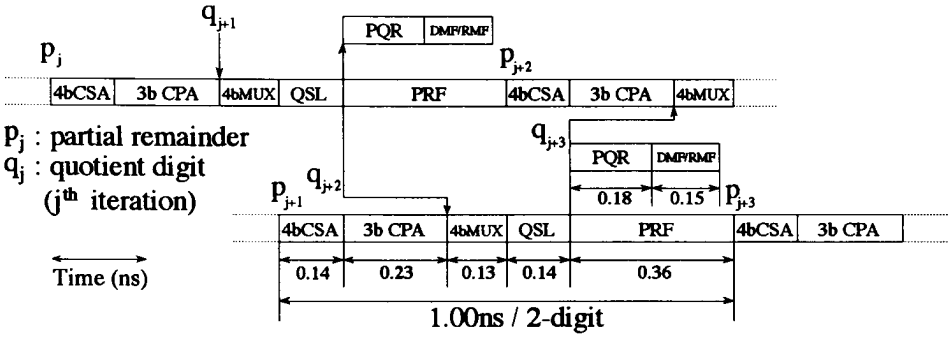


Fig. 8 Calculation timing of the shared division and square root unit.

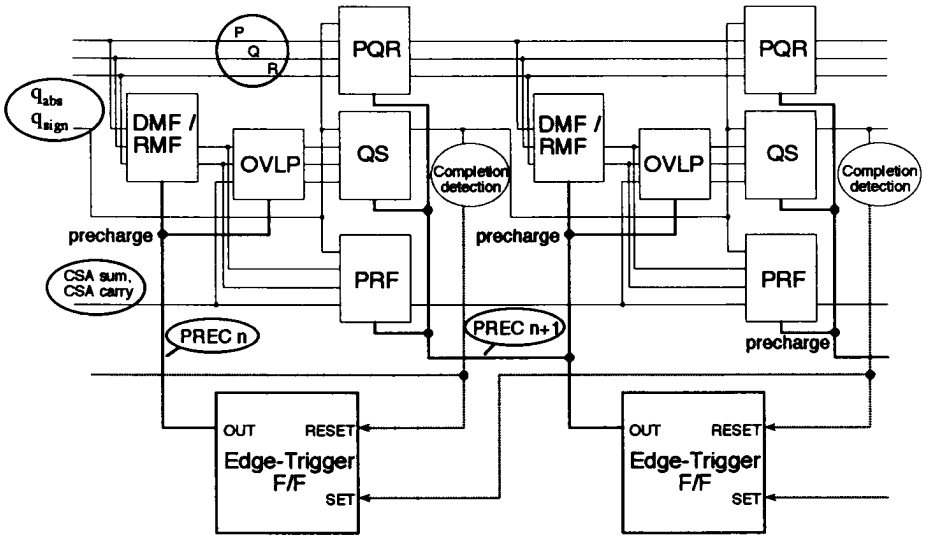


Fig. 9 Organization of the precharge timing control circuit.

6 Simulation Result and Comparison with Synchronous Implementations

First, we have confirmed that our division and square root unit computes correct results by using a functional simulation software written in a high-level language on an engineering workstation. The results from our calculation unit were compared with both the results from a commercial floating point co-processor and the results from the simulation of calculations using a classical nonrestoring method.

Following the functional simulation, we also performed a SPICE circuit simulation on our calculation unit. To obtain reliable results, we estimated a wiring capacitance by assuming a floor plan of the calculation unit. By considering the layout pattern for each bit partition of the calculation unit, we settled a width of bit slice to be 25 μm. As it stands, a total lateral width of the shared division and square root unit is about 1500 μm. A wiring capacitance is estimated to be about 0.2pF/mm from the specification of a device process. The device process technology is 0.3 μm triple metal CMOS [9].

Figures 10 and 11 show the simulation results of the signal timing. The signal names are defined in Fig. 9. In Fig. 10, the precharge control signal 'PREC n' is disabled and the circuit block is prepared for the evaluation before the data signals (such as P,Q,R and so on) reach the circuit block. Figure 11 also shows the timing relationship between the precharge signal 'PREC n+1' and data signal q_{abs} and q_{sign} (quotient digit). Thus, the circuit block is always prepared for evaluation when the data signals arrive at the circuit block. This fact ensures 'zero-overhead execution' in a self-timed circuit as in [6].

Next, we estimated a delay time to calculate one digit in the worst case. The worst case is that the digit = +1 or -1 is successively selected. As a result, the calculation time for one digit is 0.50ns at Vdd = 3.3V both for a division and a square root. The calculation time to obtain a 55-b full mantissa result is 29.5ns including 2.0ns loss of time in an initialize/loop control circuit.

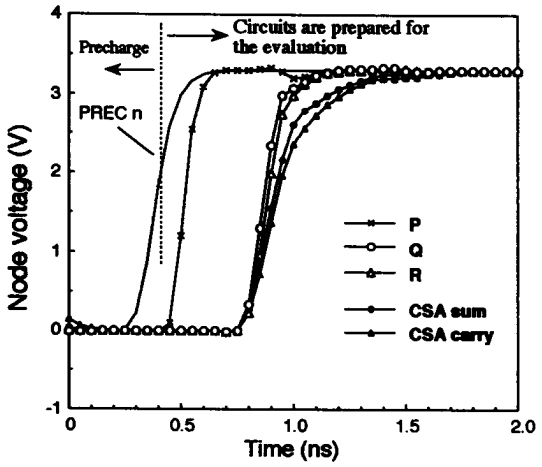


Fig. 10 The simulation result of node voltage around the precharge signal 'PREC n' in the calculation unit.

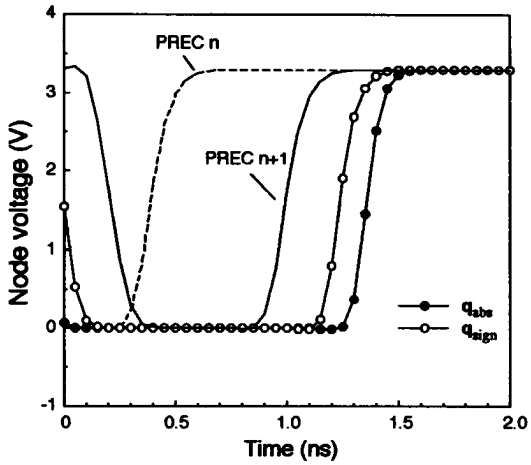


Fig. 11 The simulation result of node voltage around the precharge signal 'PREC n+1' in the calculation unit.

Table III Comparison of the latency of calculation and hardware cost between a conventional circuit design and the design using a self-timed circuit.

Circuit style Ovlp : with overlapped execution	Latency for 55b calculation	Hardware cost		
		Total cost	except for registers.	registers
Synchronous design Radix 2 Ovlp, 4-stage(4bit / iteration)	74.1ns	1270	688	291-b
Radix 4, 2-stage (4bit / iteration)	75.5ns	1060	470	295-b
Self-timed design (radix 2 Ovlp, 5-stage) shared division / square root unit	29.5ns	1837	1507	165-b
Division only — using on-the-fly block	29.5ns	1645	1315	165-b
Division only — without on-the-fly block	31.5ns	1545	1215	165-b

Hardware cost is normalized at the static 1-b CSA as 1.

Table III shows the comparison between conventional synchronous implementations and our design of a self-timed circuit implementation. In the synchronous circuit design, we considered two examples which are widely used for commercial floating point co-processors. One is using 4 stages of radix 2 quotient selection logic with overlapped execution. The other is using 2 stages of radix 4 quotient selection logic without overlapped execution. These examples can produce 4 digits per iteration cycle. In addition, we assume that a penalty delay time due to a latch loss is 0.5ns per iteration cycle. From Table III, the latency of the self-timed implementation is about 1/3 of the latency of the synchronous design. From the circuit simulation, most of the speed-up is accomplished by using a dynamic circuit with n-channel pull-down network. This feature is achieved most effectively with the zero-overlapped execution timing [6]. That is, the dynamic circuit can run without the penalty of the precharging time.

Next, considering the hardware cost, the silicon area of the self-timed implementation is larger than the conventional one, because the self-timed implementation uses many calculation stages to achieve fast execution and also uses a dual-monotonic wire set which requires a redundant circuit. However, the self-timed design requires less registers compared to the synchronous design, that is, the full bit width of the partial remainder p represented in the carry-save form, the quotient digit Q and R signals, as shown in Fig. 1, must be stored at the end of each iteration cycle in the synchronous design. In addition, the bit position signal P requires 14-b (1/4 of the full bit width) registers, because there are bit positions

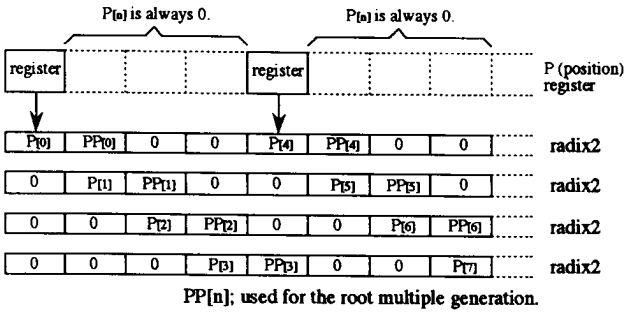


Fig. 12 Hardware reduction in the P register and related circuits. (radix 2, 4 - stage)

where the P signal never varies from 0 at the beginning of an iteration loop as shown in Fig.12. On the other hand, these signals are dynamically preserved at dynamic nodes in the self-timed design. Therefore, registers for the p, P and R signals are not needed. Only a register for the Q signal is needed for the mantissa rounding at the end of the 55-b iteration. When the implementation uses a register component with large layout size, the silicon area difference between the synchronous and the self-timed design becomes smaller. If we assume that the silicon area of a 1-b register is about twice that of a 1-b CSA circuit, the cost ratio between our self-timed implementation and the synchronous circuit in radix 4 and radix 2 with overlapped execution as shown in Table III is about 1.7 and 1.4, respectively. Moreover, considering other additional hardware, e.g. a final adder in full bit width, an incrementer for mantissa rounding, and so on, this ratio will be smaller than 1.7. In addition, our implementation of a shared division and square root unit requires a relatively small additional hardware compared with a division unit as shown in Table III. There are not any disadvantages in a calculation speed by adding a square root function. Consequently, the self-timed design can achieve a very fast division and square root unit by trading silicon area for speed at an acceptable rate.

7 Conclusions

(1) The shared division and square root unit with a self-timed circuit requires a latency of 29.5ns to calculate the mantissa of a 55-b result from the circuit simulation if a 0.3 μm triple metal CMOS technology is assumed.

(2) The new implementation of the QSL block using a 3-b CPA and a 1-b OR-gate allows for fast execution of the quotient selection for both division and square root calculations without any former information on the quotient selection.

(3) By using the on-the-fly digit decoding and the root multiple formation blocks, the latency of square root is shortened and becomes the same as that of division.

(4) The silicon area of the self-timed circuit implementation of the shared division and square root unit is estimated to be less than 1.7 times larger than the conventional one using a synchronous circuit. Considering the large advantage in latency of calculation, the self-timed circuit implementation is applicable to high-end computing systems which require a very high performance in floating point division and square root calculations.

Acknowledgment

The authors are grateful to Mr. Andrej Dolenc for his stimulating discussion.

References

[1] G.S.Taylor, "Radix 16 SRT Dividers With Overlapped Quotient Selection Stages", Proc. of 7th IEEE Symposium on Computer Arithmetic, pp.64-71 (Jun. 1985).
[2] P.Montuschi and M.Mezzalama, "Survey of square rooting algorithms", IEEE Proc., Vol. 137, Pt. E, No.1, pp.31-40 (Jan. 1990).
[3] S.Majeski, "Square-Rooting Algorithms for High-Speed Digital Circuits", IEEE trans. Computers, Vol. C-34, No. 8, pp.724-733 (Aug. 1985).
[4] J.Fandrianto, "Algorithm for High Speed Shared Radix 4 Division and Radix 4 Square-root", Proc. of 8th IEEE Symposium on Computer Arithmetic, Como, Italy, pp.73-79 (1987).
[5] J.Fandrianto, "Algorithm for High Speed Shared Radix 8 Division and Radix 8 Square-root", Proc. of 10th IEEE Symposium on Computer Arithmetic, Santa Monica, CA, pp.68-75 (1990).
[6] T.E.Williams, M.A.Horowitz, "A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider", IEEE J. of Solid-State Circuits, Vol. 26, No.11, pp.1651-1661 (Nov. 1991).
[7] M.D.Ercegovac and T.Lang, "On-the-fly conversion from redundant into conventional representation", IEEE Trans. Computers, No. 7, pp.895-897 (1987).
[8] H.Kabuo, T.Taniguchi, A.Miyoshi, H.Yamashita, M.Urano, H.Edamatsu and S.Kuninobu, "Accurate Rounding Scheme for the Newton-Raphson Method Using Redundant Binary Representation", IEEE trans. Computers, Vol. 43, No.1, pp.43-51 (1994).
[9] T.Yoshida, G.Matsubara, S.Yoshioka, H.Tago, S.Suzuki and N.Goto, "A 500MHz 1-stage 32bit ALU with self-running test circuit", to be appeared in 1995 Symposium on VLSI Circuits, 2-2 (June 1995).