# Powering by a Table Look-Up and a Multiplication with Operand Modification

Naofumi Takagi, *Member*, *IEEE*

**Abstract**—An efficient method for generating a power of an operand, i.e., $X^p$ for an operand $X$ and a given $p$, is proposed. It is applicable to $p$s in the form of $\pm 2^k$, where $k$ is any integer and of $\pm 2^{k_1} \pm 2^{-k_2}$, where $k_1$ is any integer and $k_2$ is any nonnegative integer. The reciprocal, the square root, the reciprocal square root, the reciprocal square, the reciprocal cube, and so forth are included. The method is a modification of the piecewise linear approximation. A power of an operand is generated through a table look-up and a multiplication with operand modification. The same accuracy is achieved as the piecewise linear approximation. The multiplication and an addition required for the piecewise linear approximation are replaced by only one double-sized multiplication with a slight modification of the operand and, hence, one clock cycle may be reduced. The required table size is reduced because only one coefficient instead of two has to be stored.

**Index Terms**—Computer arithmetic, powering, division, square rooting, multiplier.

## 1 Introduction

With the increasing availability of fast ROMs (read only memories) and high-speed multipliers, generation of functions by table look-up and multiplication has become attractive. In this paper, we propose a new method for generating a power of an operand, i.e., $X^p$ for an operand $X$ and a given $p$, by table look-up and multiplication.[1] It is applicable to $p$s in the form of $\pm 2^k$, where $k$ is any integer and of $\pm 2^{k_1} \pm 2^{-k_2}$, where $k_1$ is any integer and $k_2$ is any nonnegative integer. The reciprocal $X^{-2^0}$, the square root $X^{2^{-1}}$, the reciprocal square root $X^{-2^{-1}}$, the reciprocal square $X^{-2^1}$, the reciprocal cube $X^{-2^2+2^0}$, the cube $X^{2^1+2^0}$, the fourth power $X^{2^2}$, and so forth are included.

The piecewise linear approximation [2], [3] is an efficient method for generating a power of an operand by table look-up and multiplication. The two coefficients of the linear function are read out of a look-up table. A multiplication and an addition are required besides a table look-up. When the $m$ most significant bits of an operand are used as the index of the look-up table, about $2m$-bit accuracy is obtained. The required table size is about $2^m \times (m + 2m)$-bits. The size of multiplication is about $m$-bits by $m$-bits, and the size of addition is about $2m$-bits.

The method to be proposed in this paper is a modification of the piecewise linear approximation. The same accuracy is achieved. The multiplication and an addition required for the piecewise linear approximation are replaced by only one multiplication with a slight modification of the operand. The modification of the operand is a bitwise inversion, a shift and/or a redundant binary Booth recoding [4], [5], [6], [7], and is implemented by a very simple circuit with small delay. One clock cycle may be saved because the addition is removed. The required table size is reduced to about $2^m \times 2m$-bits, because only one coefficient instead of two has to be stored. The size of multiplication is doubled, i.e., about $2m$-bits by $2m$-bits.

As the piecewise linear approximation, the proposed method can be applied to powering with rather low precision, i.e., up to about 24-bit accuracy. It is efficient for generating powers which require a lot of computation, e.g., $X^{-\frac{5}{8}}$. Note that $-\frac{5}{8} = -2^{-1} - 2^{-3}$. Another important application of it is generation of initial approximations to the reciprocal and the reciprocal square root, which are required for multiplicative division and square rooting, respectively, as shown in [8]. This paper gives a theoretical foundation to the methods proposed in [8].

The next section is an introductory section, where we explain the piecewise linear approximation based on the first-order Taylor expansion and the redundant binary Booth recoding. We propose the new method in Section 3, and compare it with conventional methods in Section 4. In Section 5, we show several practical applications of the method.

1. This paper is a revision of [1]. Some applications are added to and some are omitted from [1].

• *The author is with the Department of Information Engineering, Nagoya University, Nagoya 464-8603, Japan.*
*E-mail: ntakagi@nuie.nagoya-u.ac.jp.*

*Manuscript received 16 Jan 1998.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 106167.*

## 2 Preliminaries

### 2.1 Piecewise Linear Approximation Based on Taylor Expansion

We assume that the operand $X$ is an $n + 1$-bit binary number in the range $1 \le X < 2$. Namely, $X$ is represented as $[1.x_1x_2 \cdots x_n]$ $(x_i \in \{0, 1\})$. We split $X$ into two parts, the

upper part $X_1$ and the lower part $X_2$, where $X_1 = [1.x_1x_2 \cdots x_m]$ and $X_2 = [.x_{m+1}x_{m+2} \cdots x_n] \times 2^{-m}$.

$X^p$ of $X$ in the range $X_1 \le X < X_1 + 2^{-m}$ can be approximated as follows by the first-order Taylor expansion at the mid point, $X_1 + 2^{-m-1}$, of the range:

$$(X_1 + 2^{-m-1})^p + p \cdot (X_1 + 2^{-m-1})^{p-1} \cdot (X_2 - 2^{-m-1}). \quad (1)$$

The piecewise linear approximation based on the first-order Taylor expansion adopts a linear function $C_1 \times X_2 + C_0$, where $C_1 = p \cdot (X_1 + 2^{-m-1})^{p-1}$ and $C_0 = (X_1 + 2^{-m-1})^p - 2^{-m-1} \cdot p \cdot (X_1 + 2^{-m-1})^{p-1}$. The two coefficients $C_1$ and $C_0$ are read through table look-up addressed by $X_1$ (without the leading 1). The look-up table keeps the coefficients for $2^m$ intervals of $X$. One multiplication and one addition are required besides a table look-up. The error is about $2^{-1} \cdot p \cdot (p-1) \cdot (X_1 + 2^{-m-1})^{p-2} \cdot (X_2 - 2^{-m-1})^2$. Therefore, about $(2m + 3 - \log_2 |p| - \log_2 |p-1| - \max\{0, p-2\})$-bit accuracy is obtained. Note that $(X_1 + 2^{-m-1})^{p-2}$ is bounded by 1 when $p - 2 < 0$ and by $2^{p-2}$ otherwise. We can obtain 1-bit better accuracy by adjusting $C_0$ for each interval. The required table size is about $2^m \times (m + 2m)$-bits. The size of multiplication is about $m$-bits by $m$-bits, and the size of addition is about $2m$-bits.

### 2.2 Redundant Binary Booth Recoding

In the method to be proposed, in some cases, the redundant binary Booth recoding [4], [5], [6], [7] is used for the modification of the operand. It converts a binary number in the carry save form into a radix-4 signed-digit (SD4) number with the digit set {–2, –1, 0, 1, 2}. Namely, it calculates the sum of two binary numbers in the SD4 representation.

Let us consider conversion of two binary numbers $A$ (= $[.a_1\ a_2 \cdots a_n]_2$) and $B$ (= $[.b_1\ b_2 \cdots b_n]_2$) into an SD4 number $S$ (= $[s_0.s_1\ s_2 \cdots s_{\lceil n/2 \rceil}]_{SD4}$), where $S = A + B$. The conversion process consists of two steps. In the first step, at each position of $S$, we determine $t_{j-1}$ ($\in \{0, 1, 2\}$) and $u_j$ ($\in \{-3, -2, -1, 0, 1\}$), satisfying $4t_{j-1} + u_j = (2a_{2j-1} + a_{2j}) + (2b_{2j-1} + b_{2j})$. In the second step, at each position of $S$, we calculate $s_j$ by adding $u_j$ and $t_j$. In the first step, we determine $t_{j-1}$ and $u_j$ by examining $a_{2j+1}$ and $b_{2j+1}$ so that $s_j$ satisfies $-2 \le s_j \le 2$ in the second step. Table 1 shows a computation rule for the redundant binary Booth recoding.

The redundant binary Booth recoding is an extension of 2-bit Booth recoding. By the computation rule in Table 1, when $B$ is 0, $S$ is simply the 2-bit Booth recoded representation of $A$.

Since the computation can be performed in parallel at each position, the depth of a redundant binary Booth recoder is a small constant independent of $n$.

TABLE 1
A COMPUTATION RULE FOR THE REDUNDANT BINARY BOOTH RECODING

Step 1

| | $t_{j-1}, u_j$ | | | |
|---|---|---|---|---|
| | $b_{2j-1}b_{2j}$ | | | |
| $a_{2j-1}a_{2j}$ | 00 | 01 | 10 | 11 |
| 00 | 0, 0 | *1, –3/0, 1 | 1, –2 | 1, –1 |
| 01 | *1, –3/0, 1 | 1, –2 | 1, –1 | 1, 0 |
| 10 | 1, –2 | 1, –1 | 1, 0 | *2, –3/1, 1 |
| 11 | 1, –1 | 1, 0 | *2, –3/1, 1 | 2, –2 |

* : *Both $a_{2j+1}$ and $b_{2j+1}$ are 1. / Otherwise.*

Step 2

| | $s_j$ | | |
|---|---|---|---|
| | $t_j$ | | |
| $u_j$ | 0 | 1 | 2 |
| –3 | × | –2 | –1 |
| –2 | –2 | –1 | 0 |
| –1 | –1 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 1 | 1 | 2 | × |

× : *Never occur*

## 3 A New Method for Powering

Now, we propose a new method for generating $X^p$. We can rewrite (1) as follows:

$$(X_1 + 2^{-m-1})^{p-1} \times (X_1 + 2^{-m-1} + p \cdot (X_2 - 2^{-m-1})). \quad (2)$$

Therefore, $C \times X'$ produces the same value as $C_1 \times X_2 + C_0$, where $C = (X_1 + 2^{-m-1})^{p-1}$ and

$$X' = X_1 + 2^{-m-1} + p \cdot (X_2 - 2^{-m-1}). \quad (3)$$

$C$ can be read through table look-up addressed by $X_1$ (without the leading 1). For special $p$s, $X'$ can be obtained by modifying $X$ as explained later. The look-up table keeps $C$ for $2^m$ intervals of $X$. The required table size is about $2^m \times 2m$-bits. Only a multiplication with modification of the operand is required besides a table look-up. The size of multiplication is about $2m$-bits by $2m$-bits.

The error of $C \times X'$ is as follows:

$$\begin{aligned}
&C \times X' - X^p = X' \times \left(C - X^p \times X'^{-1}\right) \\
&= X' \times \left( \left(X_1 + 2^{-m-1}\right)^{p-1} - \left(X_1 + X_2\right)^p \cdot \right. \\
&\quad \left. \left(X_1 + 2^{-m-1} + p \cdot \left(X_2 - 2^{-m-1}\right)\right)^{-1} \right) \\
&= X' \times \left( \left( X_1^{p-1} + (p-1) \cdot 2^{-m-1} \cdot X_1^{p-2} + (p-1)(p-2) \right. \right. \\
&\quad \left. \cdot 2^{-2m-3} \cdot X_1^{p-3} + \cdots \right) - \\
&\quad \left( X_1^p + p \cdot X_1^{p-1} \cdot X_2 + p(p-1) \cdot 2^{-1} \cdot X_1^{p-2} \cdot X_2^2 + \cdots \right) \\
&\quad \cdot \left( X_1^{-1} - X_1^{-2}\left(p \cdot X_2 - (p-1) \cdot 2^{-m-1}\right) \right. \\
&\quad \left. \left. + X_1^{-3}\left(p \cdot X_2 - (p-1) \cdot 2^{-m-1}\right)^2 - \cdots \right) \right) \\
&\simeq X' \times \left( -p(p-1) \cdot 2^{-1} \cdot X_1^{p-3}\left(X_2 - 2^{-m-1}\right)^2 \right).
\end{aligned}$$

Therefore, we can obtain 1-bit better accuracy by adjusting $C$ to $C'$ as follows:

$$C' \tilde{=} C + p(p-1)\cdot 2^{-1}\cdot X_1^{p-3}\cdot 2^{-2m-3}$$
$$= \left(X_1 + 2^{-m-1}\right)^{p-1} + p(p-1)\cdot 2^{-2m-4}\cdot X_1^{p-3}$$
$$\tilde{=} X_1^{p-1} + (p-1)\cdot 2^{-m-1}\cdot X_1^{p-2} + (p-1)(3p-4)\cdot 2^{-2m-4}\cdot X_1^{p-3}.$$

Note that the error of $C' \times X'$ is about

$$X' \times p(p-1)\cdot 2^{-1}\cdot X_1^{p-3}\cdot\left(2^{-2m-3} - \left(X_2 - 2^{-m-1}\right)^2\right).$$

The same accuracy is obtained as the piecewise linear approximation with the adjusted coefficients.

Now, we show how we can produce $X'$ by modifying $X$.

### 3.1 Case (1): $p = 2^{-k}$ ($k$: Nonnegative Integer)

The square root, $X^{2^{-1}}$, belongs to this case [8].

Substituting $p = 2^{-k}$ to (3), we get

$$X' = X_1 + 2^{-m-1} + 2^{-k}\cdot\left(X_2 - 2^{-m-1}\right)$$
$$= X_1 + 2^{-m-1} - 2^{-m-k-1} + 2^{-k}\cdot X_2.$$

Therefore,

$$X' = [1.x_1x_2\cdots x_m x_{m+1}\overline{x}_{m+1}\cdots\overline{x}_{m+1}x_{m+2}x_{m+3}\cdots x_n],$$

where $\overline{x}_{m+1}$ is the complement of $x_{m+1}$. There are $k$ $\overline{x}_{m+1}$s between $x_{m+1}$ and $x_{m+2}$. We can form $X'$ by only inserting $k$ $\overline{x}_{m+1}$s between $x_{m+1}$ and $x_{m+2}$. We may throw the lower about $n + k - 2m$ bits away.

Fig. 1 illustrates an implementation of the method for this case. The required hardware is a ROM of size about $2^m \times 2m$-bits and an operand modifier which consists of only an inverter. The $k$-bit shift may be implemented by wiring. No other dedicated hardware is required when we use an existing multiplier. The multiplier must be about $2m$-bits by $2m$-bits or larger.

### 3.2 Case (2): $p = -2^{-k}$ ($k$: Nonnegative Integer)

The reciprocal, $X^{-2^0}$, and the reciprocal square root, $X^{-2^{-1}}$, belong to this case [8].

Substituting $p = -2^{-k}$ to (3), we get

$$X' = X_1 + 2^{-m-1} - 2^{-k}\cdot\left(X_2 - 2^{-m-1}\right)$$
$$= X_1 + 2^{-m-1} + 2^{-m-k-1} - 2^{-k}\cdot X_2.$$

Therefore,

$$X' = [1.x_1x_2\cdots x_m\overline{x}_{m+1}x_{m+1}\cdots x_{m+1}\overline{x}_{m+2}\overline{x}_{m+3}\cdots\overline{x}_n] + 2^{-n-k}.$$

There are $k$ $x_{m+1}$s between $\overline{x}_{m+1}$ and $\overline{x}_{m+2}$. We can form $X'$ by only complementing $X_2$ bitwise and inserting $k$ $x_{m+1}$s between $\overline{x}_{m+1}$ and $\overline{x}_{m+2}$. We may ignore the last term $+2^{-n-k}$ and throw the lower about $n + k - 2m$ bits away.

An implementation of the method for this case is the same as that for Case (1), except that the operand modifier consists of about $m - k$ inverters.

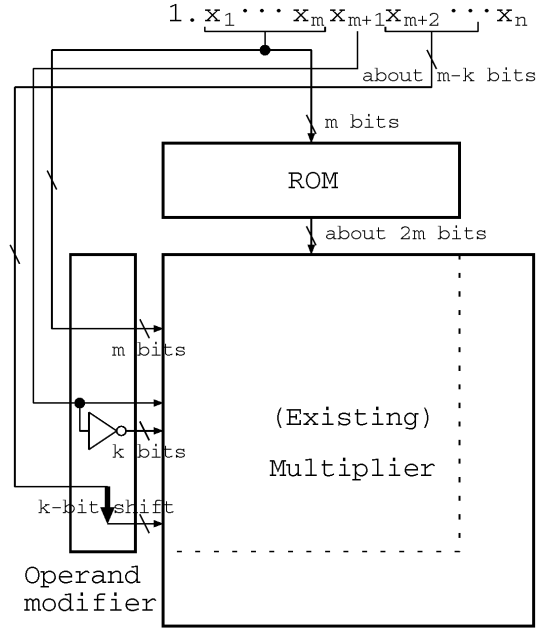


Fig. 1. An implementation for Case (1).

### 3.3 Case (3): $p = 2^k$ ($k$: Positive Integer)

The fourth power, $X^{2^2}$, belongs to this case.

Substituting $p = 2^k$ to (3), we get

$$X' = X_1 + 2^{-m-1} + 2^k\cdot(X_2 - 2^{-m-1}).$$

Namely, $X'$ is the sum of $X_1 + 2^{-m-1}$, i.e., $[1.x_1x_2\cdots x_m1]$ and the $k$-bit left shifted $X_2 - 2^{-m-1}$, i.e., $2^{-m+k}\cdot[.\tilde{\overline{x}}_{m+1}x_{m+2}x_{m+3}\cdots x_n]$, where $\tilde{\overline{x}}_{m+1}$ is $-1$ or 0 accordingly as $x_{m+1}$ is 0 or 1. They overlap each other in $k + 1$ bit positions, as shown in Fig. 2. We can form $X'$ in SD4 representation by the redundant binary Booth recoding. Since the most significant digit of the latter, i.e., $\tilde{\overline{x}}_{m+1}$, may be $-1$, we modify the calculation rule of Step 1 for the corresponding position as shown in Table 2. We may throw the lower about $n - k - 2m$ bits of the latter away.

Fig. 3 illustrates an implementation of the method for this case. When we use a multiplier with a Booth recoder, we just modify a part ($k + 1$ bit positions) of the Booth recoder. (The part indicated by a shadowed rectangle labeled "RB" in the figure.) The $k$-bit shift may be implemented by wiring.

### 3.4 Case (4): $p = -2^k$ ($k$: Positive Integer)

The reciprocal square, $X^{-2^1}$, belongs to this case.

Substituting $p = -2^k$ to (3), we get

$$X' = X_1 + 2^{-m-1} - 2^k\cdot(X_2 - 2^{-m-1}).$$

Namely, $X'$ is the sum of $X_1 + 2^{-m-1}$, i.e., $[1.x_1x_2\cdots x_m1]$, the bitwise complemented and $k$-bit left shifted $X_2 - 2^{-m-1}$, i.e., $2^{-m+k}.[\tilde{\overline{x}}_{m+1}\overline{x}_{m+2}\overline{x}_{m+3}\cdots\overline{x}_n]$, where $\tilde{\overline{x}}_{m+1}$ is 0 or $-1$ accordingly as $x_{m+1}$ is 0 or 1, and $2^{-n+k}$. The former two overlap each

$$\begin{array}{llllllllllll} & 1. & x_1 & \cdots & x_{m-k} & x_{m-k+1} & x_{m-k+2} & \cdots & x_m & & 1 & \\ + & & & & & \tilde{\tilde{x}}_{m+1} & x_{m+2} & \cdots & x_{m+k} & x_{m+k+1} & x_{m+k+2} & \cdots \quad x_n \end{array}$$

Fig. 2. $X'$ for Case (3).

other in $k + 1$ bit positions. We can form $X'$ in SD4 representation by the redundant binary Booth recoding, as in Case (3).

An implementation of the method for this case is similar to that for Case (3). Inverters are required for complementing $X_2$ bitwise.

### 3.5 Case (5): $p = \pm 2^{k_1} \pm 2^{-k_2}$ ($k_1$: Integer, $k_2$: Nonnegative Integer)

The reciprocal cube, $X^{-2^2+2^0}$, and the cube, $X^{2^1+2^0}$, belong to this case.

Substituting $p = \pm 2^{k_1} \pm 2^{-k_2}$ to (3), we get

$$\begin{aligned} X' &= X_1 + 2^{-m-1} + \left(\pm 2^{k_1} \pm 2^{-k_2}\right) \cdot \left(x_2 - 2^{-m-1}\right) \\ &= \left(X_1 + 2^{-m-1} \pm 2^{-k_2} \cdot \left(X_2 - 2^{-m-1}\right)\right) \pm 2^{k_1} \cdot \left(X_2 - 2^{-m-1}\right). \end{aligned}$$

Since $k_2$ is a nonnegative integer,

$$X_1 + 2^{-m-1} \pm 2^{-k_2} \cdot (X_2 - 2^{-m-1})$$

can be obtained by the way for Case (1) or (2). We can add $\pm 2^{k_1} \cdot (X_2 - 2^{-m-1})$ to this by a redundant binary Booth recoding.

An implementation of the method for this case is a combination of that for Case (1) or (2) and that for Case (3) or (4).

## 4 Comparison

The proposed method generates a power in the same accuracy as the conventional piecewise linear approximation based on the first order Taylor expansion when the same part of the operand is used as the table index. When we use the upper $m$ bits (without the leading 1) of the operand as the table index, both methods generate a power in about $2m$-bit accuracy.

The proposed method requires a look-up table of size about $2^m \times 2m$ bits, while the conventional method requires one of size about $2^m \times 3m$ bits. Namely, the look-up table of the proposed method is about the two thirds of that of the conventional method in size.

The proposed method does not require an addition, but requires an operand modification which is carried out by a bitwise inversion, a shift, and / or a redundant binary Booth recoding. The operand modifier consists of inverters and / or a redundant binary Booth recoder, which is a modification of an ordinary Booth recoder. It is simple and small and has a very small constant delay independent of $m$ and $n$. Each powering requires different operand modification logic.

The size of multiplication of the proposed method is about $2m$-bits by $2m$-bits, while that of the conventional method is about $m$-bits by $m$-bits. Therefore, when we prepare a dedicated multiplier, the proposed method requires a larger multiplier. The proposed method is more attractive when we may use an existing multiplier.

DasSarma and Matula proposed the faithful interpolation method, which reduces the table size required for the piecewise linear approximation [9]. In the method, only the function values at end points of intervals are stored. The coefficient of the first-order term of each interval is calcu-

TABLE 2
A Modified Computation Rule for Step 1 of the Redundant Booth Recoding

(A) $m - k$ is even ($k \geq 3$)

| | $t_{j-1}, u_j$ | | | |
|---|---|---|---|---|
| | $\tilde{\tilde{x}}_{m+1} x_{m+2}$ | | | |
| $x_{m-k+1} x_{m-k+2}$ | 00 | 01 | –10 | –11 |
| 00 | 0, 0 | *1, –3/0, 1 | 0, –2 | 0, –1 |
| 01 | *1, –3/0, 1 | 1, –2 | 0, –1 | 0, 0 |
| 10 | 1, –2 | 1, –1 | 0, 0 | *1, –3/0, 1 |
| 11 | 1, –1 | 1, 0 | *1, –3/0, 1 | 1, –2 |

*: *Both* $x_{m-k+3}$ *and* $x_{m+3}$ *are* 1. */ Otherwise.*

(B) $m - k$ is odd ($k \geq 2$)

| | $t_{j-1}, u_j$ | |
|---|---|---|
| | $\tilde{\tilde{x}}_{m+1}$ | |
| $x_{m-k} x_{m-k+1}$ | 0 | –1 |
| 00 | 0, 0 | 0, –1 |
| 01 | *1, –3/0, 1 | 0, 0 |
| 10 | 1, –2 | *1, –3/0, 1 |
| 11 | 1, –1 | 1, –2 |

*: *Both* $x_{m-k+2}$ *and* $x_{m+2}$ *are* 1. */ Otherwise.*

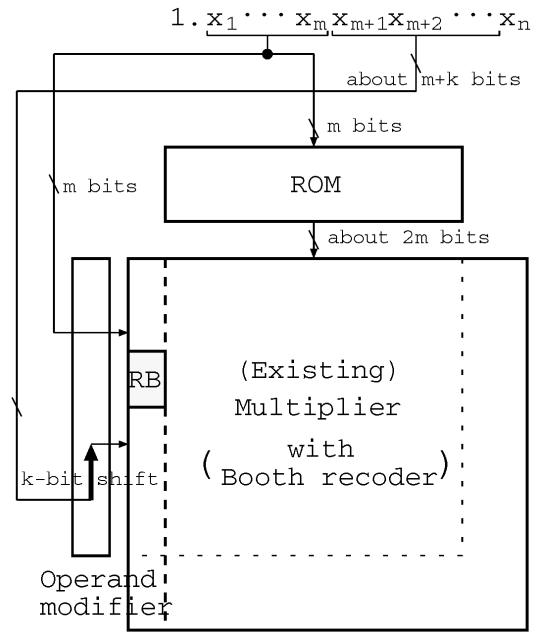


Fig. 3. An implementation for Case (3).

TABLE 3
COMPARISON OF POWERING BY TABLE LOOK-UP FOR OBTAINING $2m$-BIT ACCURACY

| Method | Table size (bits) | Multiplier (bits) | Adder (bits) | Others |
|---|---|---|---|---|
| Proposed | $2^m \times 2m$ | $2m \times 2m$ | – | (RB Booth) |
| Linear | $2^m \times (m + 2m)$ | $m \times m$ | $2m$ | |
| Faithful [9] | $2^m \times 2m$* | $m \times m$ | $2m$ | RB Booth |
| Direct | $2^{2m} \times 2m$ | – | – | |
| Bipartite [10] | $\left(2^{\frac{4}{3}m} \times 2m\right) + \left(2^{\frac{4}{3}m} \times \frac{2}{3}m\right)$ | – | – | RB Booth |

*: *Reading two consecutive contents at once.*

lated from the function values of the both end points of the interval by the redundant binary Booth recoding (subtraction). The required table is of size about $2^m \times 2m$ bits. Two consecutive contents of the table have to be read out at once. A multiplication of size about $m$-bits by $m$-bits and an addition of size about $2m$-bits are required as the conventional method.

Compared with the direct approximation, i.e., directly reading the approximation through table look-up, the proposed method additionally requires a multiplication with operand modification. It requires a much smaller table. When we want to obtain $2m$-bit accuracy by the direct approximation, we have to use the upper about $2m$ bits (without the leading 1) of the operand as the index to a table of size about $2^{2m} \times 2m$ bits.

DasSarma and Matula also reduced the table size of direct approximation by bipartite tables and a redundant binary Booth recoding (subtraction) [10]. To obtain $2m$-bit accuracy, this method requires two tables of size about $2^{\frac{4}{3}m} \times 2m$ bits and of size $2^{\frac{4}{3}m} \times \frac{2}{3}m$ bits.

Table 3 shows the comparison of powering by table look-up for obtaining $2m$-bit accuracy.

## 5 APPLICATIONS

### 5.1 Reciprocal

The calculation of the reciprocal of an operand, i.e., $X^{-2^0}$, is important, not only as it is but also as initial approximation for multiplicative division through Newton method or Goldschmidt's algorithm. It belongs to Case (2) of Section 3.

The coefficient $C'$ should be about

$$X_1^{-2} - 2^{-m} \cdot X_1^{-3} + 7 \cdot 2^{-2m-3} \cdot X_1^{-4}.$$

$2^{-2} < C' < 1$ holds. The modified operand $X'$ is

$$X_1 + 2^{-m} - X_2 = \left[1.x_1x_2\cdots x_m\overline{x}_{m+1}\overline{x}_{m+2}\cdots\overline{x}_n\right] + 2^{-n}.$$

We can form $X'$ by only complementing $X_2$ bitwise. (We ignore the last term $+2^{-n}$.)

When we use an $m$-bit-in $t$-bit-out table, the maximum absolute error is bounded by $2^{-2m-3} \cdot X_1^{-3} + 2^{-t-1} \cdot X_1$ except the error caused by the truncation of the result of the multiplication. We can truncate $X'$ at the $t$th position (and add (concatenate) 1 at the $(t + 1)$th position). (When $n \le t + 1$, we have to take the error caused by the ignored term $+2^{-n}$ into account. Note that, in such case, we can add $+2^{-n}$ into $X'$, if the multiplier is with a Booth recoder.) The table is of size $2^m \times t$ bits. The operand modifier consists of $t - m$ inverters. The multiplication is of size $t$-bits by $t + 2$-bits.

When we calculate the reciprocal with $2^{-24}$ accuracy, $m$ and $t$ should be 11 and 25, respectively, and the table is of size $2^{11} \times 25 = 50K$ bits. When we generate the initial approximation to the reciprocal for double precision division through Newton method, the table is of size $2^3 \times 8 = 64$ bits or $2^6 \times 14 = 896$ bits or $2^{13} \times 28 = 224K$ bits, accordingly as followed by three or two or one Newton iterations [8].

### 5.2 Square Root

The square root, i.e., $X^{2^{-1}}$, belongs to Case (1).

The coefficient $C'$ should be about

$$X_1^{-\frac{1}{2}} - 2^{-m-2} \cdot X_1^{-\frac{3}{2}} + 5 \cdot 2^{-2m-6} \cdot X_1^{-\frac{5}{2}}.$$

$2^{-\frac{1}{2}} < C' < 1$ holds and, therefore, the bit at the first binary position of $C'$ is always 1. The modified operand $X'$ is

$$X_1 + 2^{-m-2} + 2^{-1} \cdot X_2 = \left[1.x_1x_2\cdots x_mx_{m+1}\overline{x}_{m+1}x_{m+2}\cdots x_n\right].$$

We can form $X'$ by only inserting the complement of $x_{m+1}$ between $x_{m+1}$ and $x_{m+2}$.

When we use an $m$-bit-in $t$-bit-out table, the maximum absolute error is bounded by $2^{-2m-6} \cdot X_1^{-\frac{3}{2}} + 2^{-t-2} \cdot X_1$. We can truncate $X'$ at the $(t + 1)$th position. The operand modifier consists of only one inverter.

When we calculate the square root with $2^{-24}$ accuracy, $m$ and $t$ should be 10 and 24, respectively, and the table is of size $2^{10} \times 24 = 24K$ bits.

### 5.3 Reciprocal Square Root

The calculation of the reciprocal square root of an operand, i.e., $X^{-2^{-1}}$, is important as initial approximation for multiplicative square rooting through Newton method or Goldschmidt's algorithm. It belongs to Case (2).

The coefficient $C'$ should be about

$$X_1^{-\frac{3}{2}} - 3 \cdot 2^{-m-2} \cdot X_1^{-\frac{5}{2}} + 33 \cdot 2^{-2m-6} \cdot X_1^{-\frac{7}{2}}.$$

$2^{-\frac{3}{2}} < C' < 1$ holds. The modified operand $X'$ is

$$X_1 + 2^{-m-1} + 2^{-m-2} - 2^{-1} \cdot X_2 =$$
$$\left[1.x_1x_2\cdots x_m\overline{x}_{m+1}x_{m+1}\overline{x}_{m+2}\cdots\overline{x}_n\right] + 2^{-n-1}.$$

We can form $X'$ by complementing $X_2$ bitwise and inserting $x_{m+1}$ between $\overline{x}_{m+1}$ and $\overline{x}_{m+2}$. (We ignore the last term $+2^{-n-1}$.)

When we use an $m$-bit-in $t$-bit-out table, the maximum absolute error is bounded by $3 \cdot 2^{-2m-6} \cdot X_1^{-\frac{5}{2}} + 2^{-t-1} \cdot X_1$. We can truncate $X'$ at the $t$th position. The operand modifier consists of $t - m$ inverters.

When we calculate the reciprocal square root with $2^{-24}$ accuracy, $m$ and $t$ should be 11 and 25, respectively, and the table is of size $2^{11} \times 25 = 50K$ bits. When we generate the initial approximation to the reciprocal square root for double precision square rooting through Newton method, the table is of size $2^6 \times 14 = 896$ bits or $2^{13} \times 28 = 224K$ bits, accordingly, as followed by two or one Newton iterations [8].

### 5.4 Reciprocal Square

The reciprocal square, i.e., $X^{-2^1}$, belongs to Case (4).

The coefficient $C'$ should be about

$$X_1^{-3} - 3 \cdot 2^{-m-1} \cdot X_1^{-4} + 15 \cdot 2^{-2m-3} \cdot X_1^{-5}.$$

$2^{-3} < C' < 1$ holds. The modified operand $X'$ is

$$X_1 + 2^{-m-1} - 2 \cdot \left(X_2 - 2^{-m-1}\right) =$$
$$\left[1.x_1x_2\cdots x_m1\right] + 2^{-m+1} \cdot \left[.\widetilde{x}_{m+1}\overline{x}_{m+2}\overline{x}_{m+3}\cdots\overline{x}_n\right] + 2^{-n+1}.$$

We can form $X'$ in SD4 representation by complementing bitwise and 1-bit left shifting $X_2$ and adding it to $X_1$ by means of the redundant binary Booth recoding. Note that we can use the ordinary 2-bit Booth recoding except the overlapping two bit positions.

When we use an $m$-bit-in $t$-bit-out table, the maximum absolute error is bounded by $3 \cdot 2^{-2m-3} \cdot X_1^{-4} + 2^{-t-1} \cdot X_1$. We can truncate $X'$ at the $t$th position.

When we calculate the reciprocal square with $2^{-24}$ accuracy, $m$ and $t$ should be 12 and 25, respectively, and the table is of size $2^{12} \times 25 = 100K$ bits.

### 5.5 Reciprocal Cube

The reciprocal cube, i.e., $X^{-2^2+2^0}$, belongs to Case (5).

The coefficient $C'$ should be about

$$X_1^{-4} - 2^{-m+1} \cdot X_1^{-5} + 13 \cdot 2^{-2m-2} \cdot X_1^{-6}.$$

$2^{-4} < C' < 1$ holds. The modified operand $X'$ is

$$X_1 + X_2 - 2^2 \cdot \left(X_2 - 2^{-m-1}\right) =$$
$$\left[1.x_1x_2\cdots x_n\right] + 2^{-m+2} \cdot \left[.\widetilde{x}_{m+1}\overline{x}_{m+2}\overline{x}_{m+3}\cdots\overline{x}_n\right] + 2^{-n+2}.$$

We can form $X'$ in SD4 representation by complementing bitwise and 2-bit left shifting $X_2$ and adding it to $X$ by means of the redundant binary Booth recoding.

When we use an $m$-bit-in $t$-bit-out table, the maximum absolute error is bounded by $3 \cdot 2^{-2m-2} \cdot X_1^{-5} + 2^{-t-1} \cdot X_1$. We can truncate $X'$ at the $t$th position.

When we calculate the reciprocal cube with $2^{-24}$ accuracy, $m$ and $t$ should be 13 and 25, respectively, and the table is of size $2^{13} \times 25 = 200K$ bits.

### 5.6 Others

The cube, i.e., $X^{2^1+2^0}$, the fourth power, i.e., $X^{2^2}$, the fifth power, i.e., $X^{2^2+2^0}$, the seventh power, i.e., $X^{2^3-2^0}$, the eighth power, i.e., $X^{2^3}$, and so forth can be generated by the proposed method. They belong to Case (3) or (5). However, generating these powers by the proposed method is not so attractive because they can be generated easily through a couple of multiplications. Generation of the reciprocals of these powers by the proposed method is attractive. They belong to Case (4) or (5) and are generated in similar ways to those shown in the previous two subsections.

The fourth root, i.e., $X^{2^{-2}}$, the eighth root, i.e., $X^{2^{-3}}$, and so forth, and reciprocals of them can also be generated by the proposed method efficiently. They belong to Case (1) or (2), and are generated in similar ways to those shown in Sections 5.2 and 5.3.

The proposed method is relatively more efficient for generating powers which require a lot of computation including division and square rooting, e.g., $X^{-\frac{5}{8}}$ which requires two square rootings, a multiplication, and a division. $X^{-\frac{5}{8}} = X^{-2^{-1}-2^{-3}}$ and belongs to Case (5).

## 6 Concluding Remarks

We have proposed a new method for powering by a table look-up and a multiplication. It produces powers with the same accuracy as the conventional piecewise linear approximation based on the first-order Taylor expansion. The proposed method requires only one multiplication with a slight modification of the operand besides a table look-up, while the conventional piecewise linear approximation requires one multiplication and one addition. One clock cycle may be saved because the addition is removed. The required ROM size is also reduced, because only one coefficient instead of two is stored for each interval.

The proposed method is applicable to the generation of the reciprocal, the square root, the reciprocal square root, the reciprocal square, the reciprocal cube, the cube, the fourth power, and so forth. It is efficient for the direct generation of these powers in single precision (24-bit accuracy) or less accuracy. It is relatively more efficient for the generation of powers which require a lot of computation including division and square rooting, e.g., $X^{-\frac{5}{8}}$. It is also efficient for the generation of initial approximations to the reciprocal and the reciprocal square root for multiplicative division and square rooting, respectively.

## References

[1] N. Takagi, "Generating a Power of an Operand by a Table Look-Up and a Multiplication," *Proc. 13th Symp. Computer Arithmetic*, pp. 126-131, July 1997.

[2] *Numerical Methods*, G. Dahlquist, A. Bjorck, and N. Anderson, eds. Prentice Hall, 1974.

[3] A.S. Noetzel, "An Interpolating Memory Unit for Function Evaluation: Analysis and Design," *IEEE Trans. Computers*, vol. 38, no. 3, pp. 377-384, Mar. 1997.
[4] T. Nishimoto, "Multiple/Divide Unit," U.S. Patent 4337519, June 1982.
[5] N. Takagi, "Studies on Hardware Algorithms for Arithmetic Operations with a Redundant Binary Representation," Doctoral dissertation, Dept. of Information Science, Kyoto Univ., Aug. 1987.
[6] N. Takagi, "Arithmetic Unit Based on a High Speed Multiplier with a Redundant Binary Addition Tree," *Proc. SPIE*, vol. 1,566, pp. 244-251, July 1991.
[7] C.N. Lyu and D.W. Matula, "Redundant Binary Booth Recoding," *Proc. 12th Symp. Computer Arithmetic*, pp. 50-57, July 1995.
[8] M. Ito, N. Takagi, and S. Yajima, "Efficient Initial Approximation for Multiplicative Division and Square Root by a Multiplication with Operand Modification," *IEEE Trans. Computers*, vol. 46, no. 4, pp. 495-498, Apr. 1997.
[9] D. DasSarma and D.W. Matula, "Faithful Interpolation in Reciprocal Tables," *Proc. 13th Symp. Computer Arithmetic*, pp. 82-91, July 1997.
[10] D. DasSarma and D.W. Matula, "Faithful Bipartite ROM Reciprocal Tables," *Proc. 12th Symp. Computer Arithmetic*, pp. 17-28, July 1995.

**Naofumi Takagi** received the BE, ME, and PhD degrees in information science from Kyoto University, Kyoto, Japan, in 1981, 1983, and 1988, respectively. He joined the Department of Information Science, Kyoto University, as an instructor in 1984 and was promoted to an associate professor in 1991. He moved to the Department of Information Engineering, Nagoya University, Nagoya, Japan, in 1994, where he is now a professor. His current interests include computer arithmetic, hardware algorithms, and logic design. He served as a program co-chair of ARITH13. He has been an associate editor of the *IEEE Transactions on Computers* since October 1996. He is a member of the IEEE.