Generating a Power of an Operand by a Table Look-up and a Multiplication

Naofumi Takagi
Department of Information Engineering, Nagoya University
Nagoya 464-01, Japan
email: ntakagi@nuie.nagoya-u.ac.jp.

Abstract

An efficient method for generating a power of an operand, i.e., X^p for an operand X and a given, fixed p, is proposed. The method 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 non-negative integer. The reciprocal, the square root, and the reciprocal square root are included as special cases. It is a modification of the piecewise linear approximation based on the first-order Taylor expansion. The same accuracy is achieved. A power of an operand is generated through a table look-up and a multiplication with operand modification. No addition is required. The required table size is reduced, because only one coefficient instead of two has to be stored.

1 Introduction

With the increasing availability of large ROM's (read only memories) and high-speed multipliers, generating functions by table look-up and multiplication has become attractive. In this paper, we focus on generation of a power of an operand.

The piecewise linear approximation based on the first-order Taylor expansion [1, 2] is an efficient method for generating powers of an operand in rather low precision. 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 $2m$-bits accuracy is obtained.

In this paper, we propose a new method for generating the p-th power of an operand. The method 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 non-negative 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 cube $X^{2^1+2^0}$, and the fourth power X^{2^2} are all included as special cases.

The method is a modification of the piecewise linear approximation based on the first-order Taylor expansion. 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 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 [3, 4, 5, 6]. The required table size is reduced, because only one coefficient instead of two has to be stored. One clock cycle may be saved, because the addition is removed.

In the next section, we explain the piecewise linear approximation based on 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 discuss on the details of the method in several practical applications.

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

By the first-order Taylor expansion, X^p of X in the range $X_1 \le X < X_1 + 2^{-m}$ can be approximated as

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

Table 1. A computation rule for the redundant binary Booth recoding

Step 1

	t_{j-1},u_j			
	$a_{2j-1}a_{2j}$			
$b_{2j-1}b_{2j}$	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	$\times$	-2	-1
-2	-2	-1	0
-1	-1	0	1
0	0	1	2
1	1	2	$\times$

$\times$: Never occur.

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 $p(p-1) \cdot (X_1 + 2^{-m-1})^{p-2} \cdot (X_2 - 2^{-m-1})^2/2$. Therefore, about $(2m + 3 - \log_2 |p| - \log_2 |p-1| - \max\{0, p-2\})$-bits accuracy is obtained. (We may refine C_1 and C_0 for each interval so that we may obtain slightly better accuracy.) The required table size is about $2^m \times (m + 2m)$ bits.

2.2 Redundant binary Booth recoding

In the method to be proposed, the redundant binary Booth recoding [3, 4, 5, 6] 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 summing up u_j and t_j. In the first step, we determine u_{j-1} and t_j by looking into 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. Namely, we can perform 2-bit Booth recoding of A by letting B be 0.

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.

3 A new method for generating a power of an operand

3.1 A New Method

Now, we propose a new method for generating X^p. We can rewrite eq. (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. The look-up table keeps C for 2^m intervals of X. The required table size is about $2^m \times 2m$ bits. Only multiplication with modification of the operand is required besides a table look-up.

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

Case (1): $p = 2^{-k}$ (k: non-negative integer)

The square root, $X^{2^{-1}}$, is included as a special case [7].

Substituting $p = 2^{-k}$ to eq. (3), we get $X' = X_1 + 2^{-m-1} + 2^{-k} \cdot (X_2 - 2^{-m-1}) = X_1 + 2^{-m-1} - 2^{-m-k-1} + 2^{-k} \cdot X_2$. Therefore, $X' = [1.x_1 x_2 \cdots x_m x_{m+1} \tilde{x}_{m+1} \cdots \tilde{x}_{m+1} x_{m+2} x_{m+3} \cdots x_n]$, where $\tilde{x}_i$ is the complement of x_i. There are k $\tilde{x}_{m+1}$'s between x_{m+1} and x_{m+2}.

Table 2. A modified computation rule for the redundant Booth recoding (Step 1)

	t_{j-1},u_j			
	$a_{2j-1}a_{2j}$			
$b_{2j-1}b_{2j}$	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	0, −2	0, −1	0, 0	*1, −3/0, 1
−11	0, −1	0, 0	*1, −3/0, 1	1, −2

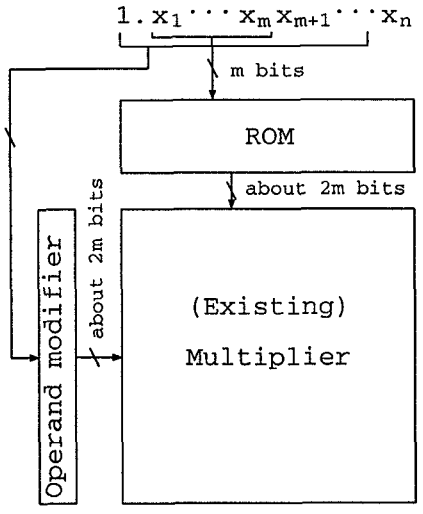

Figure 1. An implementation for case (1)

We can form X' by only inserting k $\tilde{x}_{m+1}$'s between x_{m+1} and x_{m+2}. (We throw several lower bits away, when they are unnecessary.)

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. The operand modifier consists of only an inverter. (k-bit shift may be implemented by wiring.) No other dedicated hardware is required when we use an existing multiplier. The multiplier must be $2m$-bits by $2m$-bits or larger.

Case (2): $p = -2^{-k}$ (k: non-negative integer)

The reciprocal, X^{-2^0}, and the reciprocal square root, $X^{-2^{-1}}$, are included as special cases [7].

Substituting $p = -2^{-k}$ to eq. (3), we get $X' = X_1 + 2^{-m-1} - 2^{-k} \cdot (X_2 - 2^{-m-1}) = X_1 + 2^{-m-1} + 2^{-m-k-1} - 2^{-k} \cdot X_2$. Therefore, $X' = [1.x_1x_2 \cdots x_m\tilde{x}_{m+1}x_{m+1} \cdots x_{m+1}\tilde{x}_{m+2}\tilde{x}_{m+3} \cdots \tilde{x}_n] + 2^{-n-k}$. There are k x_{m+1}'s between $\tilde{x}_{m+1}$ and $\tilde{x}_{m+2}$.

We can form X' by only complementing X_2 bitwise and inserting k x_{m+1}'s between $\tilde{x}_{m+1}$ and $\tilde{x}_{m+2}$. (We ignore the last term $+2^{-n-k}$. We throw several lower bits away, when they are unnecessary.)

An implementation of the method for this case is the same as that for case (1) shown in Fig. 1, except that the operand modifier consists of $n - m$ inverters. (The number of inverters may be fewer when several lower bits are unnecessary.)

Case (3): $p = 2^k$ (k: positive integer)

The fourth power, X^{2^2}, is included as a special case.

Substituting $p = 2^k$ to eq. (3), we get $X' = X_1 + 2^{-m-1} + 2^k \cdot (X_2 - 2^{-m-1})$. Therefore, $X' = [1.x_1x_2 \cdots x_m1] + 2^{-m+k} \cdot [.\bar{\tilde{x}}_{m+1}x_{m+2}x_{m+3} \cdots x_n]$. $\bar{\tilde{x}}_i$ is -1 or 0 accordingly as x_i is 0 or 1.

We can form X' by only a k-bit left shift of X_2 and a redundant binary Booth recoding. Since the most significant digit of the second term $\bar{\tilde{x}}_{m+1}$ may be -1, we modify the calculation rule of Step 1 for this position. Table 2 shows a modified computation rule for Step 1.

Fig. 2 illustrates an implementation of the method for this case. The operand modifier is a redundant binary Booth recoder, which can be obtained by modifying an ordinary Booth recoder. No other change is required on hardware when we use a multiplier with a Booth recoder.

Case (4): $p = -2^k$ (k: positive integer)

The reciprocal square, X^{-2^1}, is included as a special case.

Substituting $p = -2^k$ to eq. (3), we get $X' = X_1 + 2^{-m-1} - 2^k \cdot (X_2 - 2^{-m-1})$. Therefore, $X' = [1.x_1x_2 \cdots x_m1] + 2^{-m+k} \cdot [.\bar{x}_{m+1}\tilde{x}_{m+2}\tilde{x}_{m+3} \cdots \tilde{x}_n 1 \cdots 1] + 2^{-n}$. $\bar{x}_i$ is 0 or -1 accordingly as x_i is 0 or 1.

We can form X' by only a bitwise complementation and a k-bit left shift of X_2, and a redundant binary Booth recoding.

An implementation of the method for this case is the same as that for case (3) shown in Fig. 2. Inverters are required at the input of the redundant binary Booth recoder.

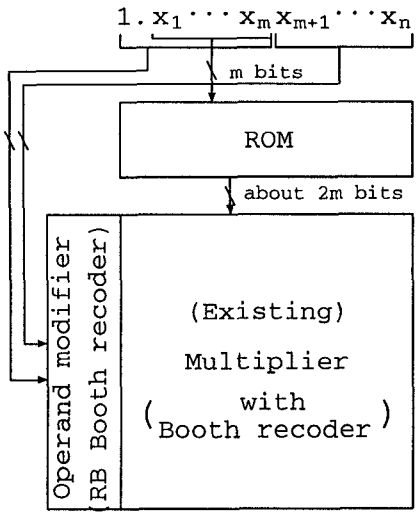

Figure 2. An implementation for case (3)

Case (5): $p = \pm 2^{k_1} \pm 2^{-k_2}$
(k_1: integer, k_2: non-negative integer)

The cube, $X^{2^1+2^0}$, is included as a special case.

Substituting $p = \pm 2^{k_1} \pm 2^{-k_2}$ to eq. (3), we get $X' = X_1 + 2^{-m-1} + (\pm 2^{k_1} \pm 2^{-k_2}) \cdot (X_2 - 2^{-m-1}) = (X_1 + 2^{-m-1} \pm 2^{-k_2} \cdot (X_2 - 2^{-m-1})) \pm 2^{k_1} \cdot (X_2 - 2^{-m-1})$. Since k_2 is a non-negative 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 redundant binary Booth recoding.

An implementation of the method for this case is similar to the former cases. The operand modifier is a combination of the operand modifier for case (1) or (2) and a redundant binary Booth recoder.

3.2 Refinement of the Coefficient

Here, we refine the coefficient C to C' in order to reduce the maximum absolute error.

The error of $C \times X'$ is $C \times X' - X^p = X' \times (C - X^p \times X'^{-1}) = X' \times ((X_1 + 2^{-m-1})^{p-1} - (X_1 + X_2)^p \cdot (X_1 + 2^{-m-1} + p \cdot (X_2 - 2^{-m-1}))^{-1}) = X' \times ((X_1^{p-1} + (p-1) \cdot 2^{-m-1} \cdot X_1^{p-2} + (p-1)(p-2) \cdot 2^{-2m-3} \cdot X_1^{p-3} + \cdots) - (X_1^p + p \cdot X_1^{p-1} \cdot X_2 + p(p-1) \cdot 2^{-1} \cdot X_1^{p-2} \cdot X_2^2 + \cdots) \cdot (X_1^{-1} - X_1^{-2}(p \cdot X_2 - (p-1) \cdot 2^{-m-1}) + X_1^{-3}(p \cdot X_2 - (p-1) \cdot 2^{-m-1})^2 - \cdots)) \simeq X' \times (-p(p-1) \cdot 2^{-1} \cdot X_1^{p-3}(X_2 - 2^{-m-1})^2)$. Therefore, C' should be $C + p(p-1) \cdot 2^{-1} \cdot X_1^{p-3} \cdot 2^{-2m-3} = (X_1 + 2^{-m-1})^{p-1} + p(p-1) \cdot 2^{-2m-4} \cdot X_1^{p-3} \simeq X_1^{p-1} + (p-1) \cdot 2^{-m-1} \cdot X_1^{p-2} + (p-1)(3p-4) \cdot 2^{-2m-4} \cdot X_1^{p-3}$.

Then, when calculations are carried out with infinite precision, the maximum absolute error becomes $|p(p-1) \cdot 2^{-2m-4} \cdot X_1^{p-2}|$. Therefore, about $(2m + 4 - \log_2 |p| - \log_2 |p-1| - \max\{0, p-2\})$-bits accuracy is obtained. Namely, one more bit accuracy is obtained by the refinement of the coefficient.

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 $2m$-bits accuracy.

The proposed method requires a look-up table of size about $2^m \times 2m$ bits, while the conventional method requires a 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.

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.

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 $2m$-bits 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 reduced the table size of direct approximation by bipartite tables and a redundant binary Booth recoding (addition) [8]. When we want to obtain $2m$-bits accuracy by this method, we require two tables of size about $2^{4m/3} \times 2m$ bits and of size $2^{4m/3} \times 2m/3$ bits.

5 Applications

5.1 Reciprocal

Computation of the reciprocal of an operand, i.e., X^{-2^0}, is not only important by itself, but also important for generation of an initial approximation to the reciprocal of the divisor in multiplicative division. It belongs to case (2) of Section 3.1.

The refined coefficient C' is $X_1^{-2} - 2^{-m} \cdot X_1^{-3} + 7 \cdot 2^{-2m-3} \cdot X_1^{-4}$, and satisfies $2^{-2} < C' < 1$. The modified operand X' is $X_1 + 2^{-m} - X_2 = [1.x_1x_2\cdots x_m\tilde{x}_{m+1}\tilde{x}_{m+2}\cdots\tilde{x}_n] + 2^{-n}$. We can form X' by only complementing X_2 bitwise. (We ignore the last term $+2^{-n}$.)

When calculations are carried out with infinite precision, the maximum absolute error is about $2^{-2m-3} \cdot X_1^{-3}$. When we use an m-bits-in t-bits-out table for C', the total absolute error considering the rounding error of C' is smaller than $2^{-2m-3} \cdot X_1^{-3} + 2^{-t-1} \cdot X_1$. Note that we do not need whole X' but need down to about the t-th position of X'. The size of multiplication is about $(t+1)$-bits by t-bits.

The required hardware is a ROM of size $2^m \times t$ bits and an operand modifier which consists of only $t - m$ inverters. Note that the operand modifier may be a modification of the complementer which is used for the multiplicative division algorithms, such as Newton-Raphson method. No other dedicated hardware is required when we use an existing multiplier which is also used for the multiplicative division. When we prepare a dedicated multiplier, its size should be $(t+1)$-bits by t-bits.

For generating the reciprocal in single-precision (24-bits) accuracy, m should be 11 and t should be 25, and the table should be of size $2^{11} \times 25 = 50K$ bits.

When we apply the method to generation of an initial approximation to the reciprocal for double-precision multiplicative division by Newton-Raphson method, the table should be of size $2^3 \times 8 = 64$ or $2^6 \times 14 = 896$ or $2^{13} \times 28 = 224K$ bits, accordingly as followed by 3 or 2 or 1 Newton-Raphson iterations [7]. (We assume that we compute the reciprocal in 54-bits accuracy.)

5.2 Square Root

The square root, i.e, $X^{2^{-1}}$, belongs to case (1) of Section 3.1.

The refined coefficient C' is $X_1^{-1/2} - 2^{-m-2} \cdot X_1^{-3/2} + 5 \cdot 2^{-2m-6} \cdot X_1^{-5/2}$, and satisfies $2^{-1/2} < C' < 1$. The modified operand X' is $X_1 + 2^{-m-2} - 2^{-1} \cdot X_2 = [1.x_1x_2\cdots x_mx_{m+1}\tilde{x}_{m+1}x_{m+2}\cdots x_n]$. We can form X' by only a complementation of x_{m+1} and a 1-bit right shift.

When calculations are carried out with infinite precision, the maximum absolute error is about $2^{-2m-6} \cdot X_1^{-3/2}$. When we use an m-bits-in t-bits-out table for C', the total absolute error considering the rounding error of C' is smaller than $2^{-2m-6} \cdot X_1^{-3/2} + 2^{-t-2} \cdot X_1$.

For generating the square root in single-precision accuracy, m should be 10 and t should be 24, and the table should be of size $2^{10} \times 24 = 24K$ bits.

5.3 Reciprocal Square Root

Computation of the reciprocal square root of an operand, i.e., $X^{-2^{-1}}$, is important for generation of an initial approximation to the reciprocal square root of the operand in multiplicative square rooting. It belongs to case (2) of Section 3.1.

The refined coefficient C' is $X_1^{-3/2} - 3 \cdot 2^{-m-2} \cdot X_1^{-5/2} + 33 \cdot 2^{-2m-6} \cdot X_1^{-7/2}$, and satisfies $2^{-3/2} < C' < 1$. The modified operand X' is $X_1 + 2^{-m-1} + 2^{-m-2} - 2^{-1} \cdot X_2 = [1.x_1x_2\cdots x_m\tilde{x}_{m+1}x_{m+1}\tilde{x}_{m+2}\cdots\tilde{x}_n] + 2^{-n-1}$. We can form X' by only a bitwise complementation of X_2 and a 1-bit right shift. (We ignore the last term $+2^{-n-1}$.)

When calculations are carried out with infinite precision, the maximum absolute error is about $3 \cdot 2^{-2m-6} \cdot X_1^{-5/2}$. When we use an m-bits-in t-bits-out table for C', the total absolute error considering the rounding error of C' is smaller than $3 \cdot 2^{-2m-6} \cdot X_1^{-5/2} + 2^{-t-1} \cdot X_1$.

The required hardware is a ROM of size $2^m \times t$ bits and an operand modifier which consists of only $t - m$ inverters. Note that the operand modifier may be a modification of the complementer which is used for the multiplicative square rooting algorithms, such as Newton-Raphson method.

For generating the reciprocal square root in single-precision accuracy, m should be 10 and t should be 25, and the table should be of size $2^{10} \times 25 = 25K$ bits.

When we apply the method to generation of an initial approximation to the reciprocal square root for double-precision multiplicative square rooting by Newton-Raphson method, the table should be of size $2^6 \times 14 = 896$ or $2^{13} \times 28 = 224K$ bits, accordingly as followed by 2 or 1 Newton-Raphson iterations [7]. (We assume that we compute $\sqrt{X}$ or $\sqrt{2X}$ accordingly as the exponent is even or odd. We also assume that we compute the reciprocal square root in 54-bits accuracy.)

5.4 Reciprocal Square

The reciprocal square, i.e., X^{-2^1}, belongs to case (4) of Section 3.1.

The refined coefficient C' is $X_1^{-3} - 3 \cdot 2^{-m-1} \cdot X_1^{-4} + 15 \cdot 2^{-2m-3} \cdot X_1^{-5}$, and satisfies $2^{-3} < C' < 1$. The modified operand X' is $X_1 + 2^{-m-1} - 2 \cdot (X_2 - 2^{-m-1}) = [1.x_1 x_2 \cdots x_m 1] + 2^{-m+1} \cdot [.\bar{x}_{m+1} \tilde{x}_{m+2} \tilde{x}_{m+3} \cdots \tilde{x}_n 1] + 2^{-n}$. We can form X' by a bitwise complementation of X_2, a 1-bit left shift and a redundant binary Booth recoding.

When we use an m-bits-in t-bits-out table for C', the total absolute error considering the rounding error of C' is smaller than $3 \cdot 2^{-2m-3} \cdot X_1^{-4} + 2^{-t-1} \cdot X_1$.

For generating the reciprocal square in single-precision accuracy, m should be 12 and t should be 25, and the table should be of size $2^{12} \times 25 = 100K$ bits.

5.5 Cube

The cube, i.e., $X^{2^1+2^0}$, belongs to case (5) of Section 3.1.

The refined coefficient C' is $X_1^2 - 2^{-m} \cdot X_1 + 5 \cdot 2^{-2m-3}$, and satisfies $1 < C' < 4$. The modified operand X' is $X_1 + X_2 + 2 \cdot (X_2 - 2^{-m-1}) = [1.x_1 x_2 \cdots x_n] + 2^{-m+1} \cdot [.\bar{\tilde{x}}_{m+1} x_{m+2} x_{m+3} \cdots x_n]$. We can form X' by a 1-bit left shift and a redundant binary Booth recoding.

When we use an m-bits-in t-bits-out table for C', the total absolute error considering the rounding error of C' is smaller than $3 \cdot 2^{-2m-3} \cdot X_1 + 2^{-t+1} \cdot X_1$.

For generating the cube in single-precision accuracy, m should be 12 and t should be 28, and the table should be of size $2^{12} \times 28 = 112K$ bits.

5.6 Fourth Power

The fourth power, i.e., X^{2^2}, belongs to case (3) of Section 3.1.

The refined coefficient C' is $X_1^3 + 3 \cdot 2^{-m-1} \cdot X_1^2 + 3 \cdot 2^{-2m-1} \cdot X_1$, and satisfies $1 < C' < 8$. The modified operand X' is $X_1 + 2^{-m-1} + 2^2 \cdot (X_2 - 2^{-m-1}) = [1.x_1 x_2 \cdots x_m 1] + 2^{-m+2} \cdot [.\bar{\tilde{x}}_{m+1} x_{m+2} \cdots x_n]$. We can form X' by a 2-bit left shift and a redundant binary Booth recoding.

When we use an m-bits-in t-bits-out table for C', the total absolute error considering the rounding error of C' is smaller than $3 \cdot 2^{-2m-2} \cdot X_1^2 + 2^{-t+2} \cdot X_1$.

For generating the fourth power in single-precision accuracy, m should be 13 and t should be 29, and the table should be of size $2^{13} \times 29 = 232K$ bits.

6 Concluding Remarks

We have proposed a new method for generating the p-th power of an operand for several useful p's. It requires only one multiplication with a slight modification of the operand and produces an approximation with the same accuracy as the conventional piecewise linear approximation based on the first-order Taylor expansion which requires one multiplication and one addition. One clock cycle may be saved, and the required ROM size is reduced.

The proposed method can be used for direct generation of several powers, e.g., reciprocal, square root, reciprocal square root, reciprocal square, cube, fourth power, and etc., of an operand in single-precision (24-bits) or less accuracy. It is also efficient for generation of reciprocal and reciprocal square root used for initial approximations for multiplicative division and square rooting, respectively.

References

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

[2] P. M. Farmwald: 'High bandwidth evaluation of elementary functions,' Proc. of 5th Symposium on Computer Arithmetic, pp. 139–142, 1981.

[3] T. Nishimoto: 'Multiple/Divide Unit,' U.S. Patent 4337519, June 1982.

[4] N. Takagi: 'Studies on Hardware Algorithms for Arithmetic Operations with a Redundant Binary Representation,' Doctoral dissertation, Dept. of Information Science, Kyoto University, Aug. 1987.

[5] N. Takagi: 'Arithmetic unit based on a high speed multiplier with a redundant binary addition tree,' Proc. SPIE, vol. 1566, pp. 244–251, July 1991.

[6] C. N. Lyu and D. W. Matula: 'Redundant binary Booth recoding,' Proc. 12th Symposium on Computer Arithmetic, pp. 50–57, July 1995.

[7] M. Ito, N. Takagi, and S. Yajima: 'Efficient initial approximation for multiplicative division and square root by a multiplication with operand modification,' IEEE Transactions on Computers, vol. 46, no. 4, April 1997.

[8] D. DasSarma and D. W. Matula: 'Faithful bipartite ROM reciprocal tables,' Proc. 12th Symposium on Computer Arithmetic, pp. 17–28, July 1995.