Efficient Function Approximation Using Truncated Multipliers and Squarers

E. George Walters III
Lehigh University
Bethlehem, PA, USA
waltersg@ieee.org

Michael J. Schulte
University of Wisconsin–Madison
Madison, WI, USA
schulte@engr.wisc.edu

Abstract

This paper presents a technique for designing linear and quadratic interpolators for function approximation using truncated multipliers and squarers. Initial coefficient values are found using a Chebyshev series approximation, and then adjusted through exhaustive simulation to minimize the maximum absolute error of the interpolator output. This technique is suitable for any function and any precision up to 24-bits (IEEE single precision). Designs for linear and quadratic interpolators that implement the reciprocal function, $f(x) = 1/x$, are presented and analyzed as an example. We show that a 24-bit truncated reciprocal quadratic interpolator with a design specification of ± 1 ulp error requires 24.1 % fewer partial products to implement than a comparable standard interpolator with the same error specification.

1. Introduction

Approximation of functions such as reciprocal, square root, cosine, logarithm, and others are important for a wide variety of applications, including general purpose computing, digital signal processing, and application specific processors such as for graphics acceleration. As transistor densities continue to increase, more of these functions are implemented in hardware. Piecewise polynomial function approximation, using coefficients stored in a lookup table, is a popular technique, with many papers written on the subject [1–3, 6, 7, 12].

Truncated multipliers and squarers are arithmetic units in which several of the least significant columns of partial products are not formed. Truncated multipliers and squarers offer area, delay, and power improvements, but introduce additional error into the computation. This paper presents a technique for designing function interpolators using truncated multipliers and squarers. Adjusting the coefficients through exhaustive simulation compensates for the nonlinear effects of finite precision arithmetic and minimizes the error due to the unformed partial products. Results show that significant area savings can be realized without exceeding design specifications for error.

This paper presents a brief overview of truncated multipliers and squarers in Section 2. Section 3 discusses function approximation and how initial coefficients are obtained. Section 4 describes general hardware designs, and presents an error analysis that is used to determine coefficient lengths and rounding precisions. Section 5 describes our method for optimizing coefficients to minimize lookup table size and to minimize output error. Section 6 presents results for several interpolator designs using our technique. Section 7 presents conclusions.

2. Truncated Multipliers and Squarers

Truncated multipliers and squarers are units in which several of the least significant columns of partial products are not formed [11]. Eliminating partial products from the multiplication or squaring matrix reduces the area of the unit by eliminating the logic needed to generate those terms, as well as reducing the number of adder cells required to reduce the matrix prior to the final addition. Additional area savings are realized because a shorter carry-propagate adder can be used to compute the final results, which often yields reduced delay as well. Eliminating adder cells, and thus their related switching activity, also results in reduced power consumption.

Figure 1 shows a 14×10-bit truncated multiplier, where r denotes the number of unformed columns and k denotes the number of columns that are formed but discarded in the final result. In this example, $r = 7$ and $k = 2$. Eliminating partial products introduces a reduction error, E_r, into the output. This error ranges from E_{r_low}, which occurs when each of the unformed partial product bits is a '1', to E_{r_high}, which occurs when each is a '0'. E_{r_low} is given by [9]

$$E_{r\_low} = -\big((r-1)\cdot 2^r + 1\big)\cdot 2^{-r-k} \text{ ulps} \ , \qquad (1)$$

where ulps is units in the last place of the product. Since

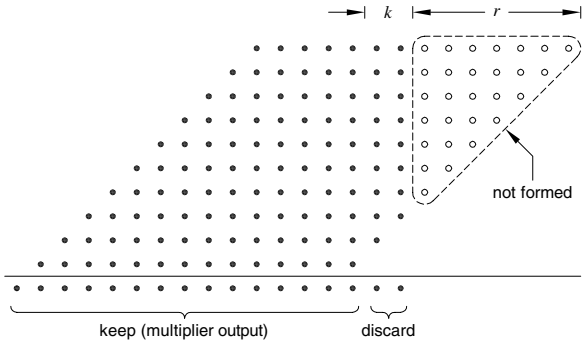

Figure 1. 14×10-bit truncated multiplier, $k = 2$, $r = 7$.

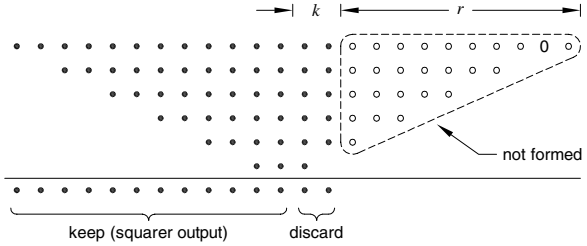

Figure 2. 12-bit truncated squarer, $k = 2$, $r = 10$.

E_{r_high} is zero, the range of the reduction error is

$$-\big((r-1)\cdot 2^r+1\big)\cdot 2^{-r-k}\text{ ulps} \le E_r \le 0\ . \quad (2)$$

In the example given in Figure 1, the range of reduction error is -1.502 ulps $\le E_r \le 0$ ulps. In comparison, the error due to rounding a product to nearest has a range of -0.5 ulps $< E_{rnd} \le 0.5$ ulps.

Figure 2 shows a 12-bit truncated squarer. Truncated squarers are an extension of specialized squarers, which are described in [14]. As with truncated multipliers, r denotes the number of unformed columns and k denotes the number of columns that are formed but discarded in the final result. Unlike truncated multipliers, it is impossible for each of the unformed partial product bits in a truncated squarer to be '1'. E_{r_low} for a truncated squarer is given by [13]

$$E_{r\_low} = \begin{cases} -\big(2^{r-1}\cdot r - 2^r + 1\big) \\ \quad \cdot 2^{-r-k}\text{ ulps, if } r \text{ is even} \\ -\left(2^{r-1}\cdot r - \dfrac{11}{8}\cdot 2^r + 2^{(\frac{1}{2}r-\frac{1}{2})} + 1\right) \\ \quad \cdot 2^{-r-k}\text{ ulps, if } r \text{ is odd}\ . \end{cases} \quad (3)$$

The range of reduction error for a truncated squarer is $E_{r\_low} \le E_r \le 0$. In the example given in Figure 2, $k = 2$, $r = 10$, and the range of reduction error is -1.000 ulps $\le E_r \le 0$ ulps.

For generality, k and r will be used in equations given in later sections to describe both standard and truncated multipliers and squarers. For a standard unit, $r = 0$ because all columns of partial products are formed.

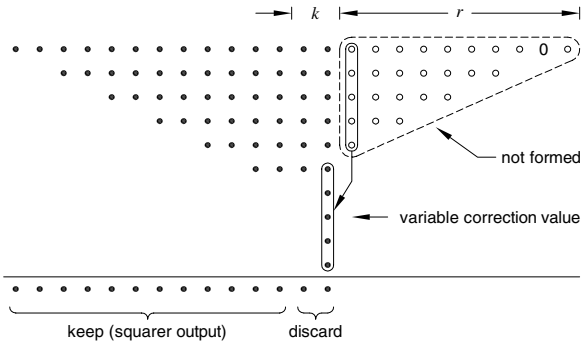

Figure 3. 12-bit truncated squarer with variable correction.

2.1. Constant Correction

From the previous discussion, it can be seen that the reduction error for both truncated multipliers and truncated squarers is always negative. One way to offset this error is to add a constant to the partial product matrix [9]. In some applications, it is desirable to select a constant that makes the average error of the multiplier or squarer as close to zero as possible. When designing a function interpolator, however, it is usually desirable to minimize the maximum absolute error. This is done by selecting a correction constant, C_{cc_abs}, equal to the additive inverse of the midpoint of the range of the error. This value is

$$C_{cc\_abs} = -\frac{E_{r\_low}}{2}\ , \quad (4)$$

where E_{r_low} is given by (1) for truncated multipliers and (3) for truncated squarers. In practice, C_{cc_abs} is rounded so that it does not have any bits in the r least significant columns, since those bits will have no effect on the final output.

2.2. Variable Correction

Another way to offset reduction error is through variable correction [5]. Figure 3 shows a truncated squarer with variable correction. With this technique, the most significant column of unformed partial products is added to the next most significant column. This can be thought of as rounding each row of partial products, such that the r least significant columns are eliminated.

3. Polynomial Function Approximation

Approximating a function $Y = f(X)$ is usually done in three steps. First, range reduction is performed to reduce X to x, where $x \in [x_{min}, x_{max})$. Next, $y = f(x)$ is found. Finally, $Y = f(X)$ is found by performing a reconstruction step. Range reduction and reconstruction techniques for common function approximations are well known [10], and are not discussed here. This paper presents a technique for computing $y = f(x)$ by piecewise polynomial approximation using truncated multipliers and squarers.

Polynomial approximations have the following form

$$\begin{aligned} f(x) &\approx a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1} \\ &\approx \sum_{i=0}^{N-1} a_i x^i \ , \end{aligned} \tag{5}$$

where N is the number of terms in the polynomial approximation, $f(x)$ is the function to be approximated, and a_i is the coefficient of the ith term.

In order to reduce the order of the polynomial while maintaining the desired output accuracy, the interval $[x_{min}, x_{max})$ is often partitioned into 2^m subintervals of equal size, each with a different set of coefficients. To implement this efficiently in hardware, the interpolator input is split into an m-bit most significant part, x_m, and an $(n-m)$-bit least significant part, x_l, where n is the number of bits input to the interpolator.

In the case where $x \in [0, 1)$,

$$x = x_m + x_l \quad \begin{cases} 0 \le x_m \le 1 - 2^{-m} \\ 0 \le x_l \le 2^{-m} - 2^{-n} \end{cases} . \tag{6}$$

In the case where $x \in [1, 2)$,

$$x = 1 + x_m + x_l \quad \begin{cases} 0 \le x_m \le 1 - 2^{-m} \\ 0 \le x_l \le 2^{-m} - 2^{-n} \end{cases} . \tag{7}$$

The coefficients for each subinterval are stored in a lookup table. x_m is used to select the coefficients, so (5) becomes

$$\begin{aligned} f(x) &\approx a_0(x_m) + a_1(x_m) x_l + a_2(x_m) x_l^2 \\ &\quad + \cdots + a_{N-1}(x_m) x_l^{N-1} \\ &\approx \sum_{i=0}^{N-1} a_i(x_m) x_l^i \ . \end{aligned} \tag{8}$$

With the approach presented in this paper, initial coefficient values are selected using a Chebyshev series approximation [4] for each subinterval. These coefficients are then quantized as described in Section 4, and optimized as described in Section 5. Schulte and Swartzlander [10] describe Chebyshev series approximation in detail. This section summarizes the equations used to generate initial coefficient values for linear and quadratic approximations.

First, the number of subintervals, 2^m, must be determined. When using infinite precision arithmetic, the maximum absolute error of a Chebyshev series approximation is [10]

$$E_{Chebyshev} = \frac{2^{-N(m+2)+1} \cdot |f^n(\xi)|}{N!} \ , \quad x_m \le \xi < x_m + 2^{-m} \ . \tag{9}$$

where ξ is the point on the interval being approximated such that the Nth derivative of $f(x)$ is at its maximum value. The maximum allowable error of the interpolator is 2^{-q}, which is selected as a design parameter. Since there will be error due to finite precision arithmetic in addition to $E_{Chebyshev}$, we limit $E_{Chebyshev}$ to 2^{-q-2}, and solve (9) for m. Since m must be an integer, this gives

$$m = \left\lceil \frac{q - 2N + 3 + \log_2(|f^N(\xi)|) - \log_2(N!)}{N} \right\rceil \ . \tag{10}$$

For a linear interpolator, the coefficients for the interval $[x_m, x_m + 2^{-m})$ are given by

$$a_0 = -\frac{1}{2} y_0 \left(\sqrt{2} - 1\right) + \frac{1}{2} y_1 \left(\sqrt{2} + 1\right) \tag{11}$$

$$a_1 = \sqrt{2} \left(y_0 - y_1\right) \cdot 2^m \ , \tag{12}$$

where

$$y_0 = f\left(x_m + 2^{-m-1} + \sqrt{2} \cdot 2^{-m-2}\right) \tag{13}$$

$$y_1 = f\left(x_m + 2^{-m-1} - \sqrt{2} \cdot 2^{-m-2}\right) \ . \tag{14}$$

For a quadratic interpolator, the coefficients for the interval $[x_m, x_m + 2^{-m})$ are given by

$$a_0 = \frac{1}{3} y_0 \left(2 - \sqrt{3}\right) - \frac{1}{3} y_1 + \frac{1}{3} y_2 \left(\sqrt{3} + 2\right) \tag{15}$$

$$a_1 = \frac{1}{6} y_0 \left(\sqrt{3} - 4\right) \cdot 2^{m+2} + \frac{1}{3} y_1 \cdot 2^{m+4} - \frac{1}{6} y_2 \left(\sqrt{3} + 4\right) \cdot 2^{m+2} \tag{16}$$

$$a_2 = \frac{1}{3} \left(y_0 - 2y_1 + y_2\right) \cdot 2^{2m+3} \ , \tag{17}$$

where

$$y_0 = f\left(x_m + \left(\frac{\sqrt{3}}{2} + 1\right) \cdot 2^{-m-1}\right) \tag{18}$$

$$y_1 = f\left(x_m + 2^{-m-1}\right) \tag{19}$$

$$y_2 = f\left(x_m + \left(1 - \frac{\sqrt{3}}{2}\right) \cdot 2^{-m-1}\right) \ . \tag{20}$$

Coefficients for cubic and higher order interpolators can be found using the equations presented in [10].

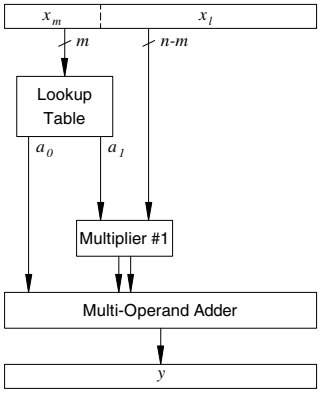

Figure 4. Linear interpolator block diagram.

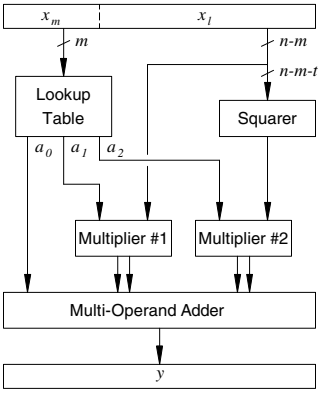

Figure 5. Quadratic interpolator block diagram.

4. Preliminary Hardware Design

4.1. Finite Precision Arithmetic Effects

In addition to the error of the Chebyshev approximation, there are errors due to quantization effects and multiplier/squarer rounding that affect the accuracy of the interpolator output.

Each coefficient a_i is rounded to nearest and quantized to n_i bits and stored in a lookup table. The least significant bit of each coefficient has a weight of $2^{-n_{fi}}$, where n_{fi} is the number of fractional bits in coefficient a_i. The difference between a quantized coefficient and its infinite precision value is defined as ε_i, so $|\varepsilon_i| \leq 2^{-n_{fi}-1}$.

In order to prevent excessive intermediate wordlengths, multiplier and squarer outputs are rounded. Rounding is accomplished by adding a '1' to the column immediately to the right of the rounding point, then discarding the k least significant bits at the output. The maximum rounding error, E_{rnd}, is 0.5 ulps.

4.2. Linear Interpolator

Figure 4 shows the block diagram of a linear interpolator. x_m is used to select coefficients a_0 and a_1 from a lookup table. Multiplier #1 computes $a_1 \cdot x_l$, which is then added to a_0 to produce the output. The multiplier output can be kept in carry-save form to reduce the overall area and delay of the interpolator.

Errors in the output due to the quantization of a_0 and a_1 are E_{ε_0} and E_{ε_1} respectively. Since a_0 contributes directly to the output, $E_{\varepsilon_0} = \varepsilon_0$, so

$$|E_{\varepsilon_0}| \leq 2^{-n_{f0}-1} \ . \tag{21}$$

a_1 is multiplied by x_l, which has a maximum value less than 2^{-m}, so

$$|E_{\varepsilon_1}| < 2^{-n_{f1}-1} \cdot 2^{-m} \ . \tag{22}$$

The design goal is to limit the interpolator output absolute error to 2^{-q}, where q is selected based on the overall accuracy requirements. We choose each coefficient length by setting $E_{\varepsilon_i} = 2^{-q-3}$ and solving for n_{fi}. This ensures that $E_{\varepsilon_0} + E_{\varepsilon_1} \leq 2^{-q-2}$. Section 5 describes how these lengths are then reduced to the minimum acceptable precision while maintaining the desired accuracy. In addition to the fractional bits, a sign bit is needed, so

$$n_0 = n_{f0} + 1 = q + 3 \tag{23}$$

$$n_1 = n_{f1} + 1 = q - m + 3 \ . \tag{24}$$

If additional bits to the left of the binary point are required (e.g., if $|a_0| \geq 1$ or $|a_1| \geq 1$), these values are incremented accordingly.

In addition to quantization errors, rounding the multiplier output introduces a rounding error E_{rnd_m1} at the interpolator output. The LSB (least significant bit) weight of a_1 is $2^{-n_{f1}}$ and the LSB weight of x_l is 2^{-n}, so the LSB weight of a full precision product would be $2^{-n_{f1}-n}$. Since r columns of partial products are not formed and k output bits are discarded, the LSB weight of the multiplier output is $2^{-n_{f1}-n+k_{m1}+r_{m1}}$, so $E_{rnd\_m1} = 2^{-n_{f1}-n+k_{m1}+r_{m1}-1}$, where k_{m1} and r_{m1} are k and r for the multiplier.

We start our design using standard multipliers. We want the rounding error to be less than the error due to coefficient quantization, so we choose k_{m1} by setting $E_{rnd\_m1} = 2^{-q-4}$ and solving for k_{m1}. Remember that for a standard multiplier, all columns of partial products are formed, so $r = 0$ and

$$k_{m1} = n - m - 1 \ . \tag{25}$$

4.3. Quadratic Interpolator

Figure 5 shows the block diagram of a quadratic interpolator. x_m is used to select coefficients a_0, a_1, and a_2 from

a lookup table. A specialized squarer computes x_l^2. Multiplier #1 computes $a_1 \cdot x_l$ and multiplier #2 computes $a_2 \cdot x_l^2$, both of which are then added to a_0 to produce the output. As with the linear interpolator, the multiplier outputs can be kept in carry-save form.

E_{ε_0} and E_{ε_1} for a quadratic interpolator are the same as for a linear interpolator, given by (21) and (22). a_2 is multiplied by the squarer output, which has a maximum value of 2^{-2m}, so

$$|E_{\varepsilon_2}| < 2^{-n_{f2}-1} \cdot 2^{-2m} \ . \tag{26}$$

As with the linear interpolator, the design goal is to limit the approximation error to 2^{-q}. In the case of the quadratic interpolator, however, there are three errors due to coefficient quantization as well as several rounding errors, so we initially set each E_{ε_i} equal to 2^{-q-4} rather than 2^{-q-3} to ensure that $\Sigma E_{\varepsilon_i} < 2^{-q-2}$, and the sum of all errors is less than 2^{-q}. Assuming a sign bit in addition to the fractional bits,

$$n_0 = n_{f0} + 1 = q + 4 \tag{27}$$
$$n_1 = n_{f1} + 1 = q - m + 4 \tag{28}$$
$$n_2 = n_{f2} + 1 = q - 2m + 4 \ . \tag{29}$$

As noted previously, these values are incremented accordingly if additional bits to the left of the binary point are required.

Analysis shows that for some configurations, x_l can be truncated at the input to the squarer to reduce the size of the squarer. Assume that the t least significant bits of x_l are truncated, such that $x_l = x_l' + \varepsilon_{x_l}$, where x_l' is the truncated version of x_l. The squarer output is then ${x_l'}^2$ rather than x_l^2, resulting in a squarer output error of $-2x_l' \cdot \varepsilon_{x_l} - \varepsilon_{x_l}^2$. Noting that $x_l' < 2^{-m}$, $|\varepsilon_{x_l}| < 2^{-n+t}$, and $\varepsilon_{x_l}^2$ is negligible, the magnitude of the squarer output error is less than $2^{-n-m+t+1}$. This error is then multiplied by a_2, so the error at the interpolator output due to ε_{x_l} is

$$|E_{\varepsilon_{x_l}}| < 2^{-n-m+t+1} \ , \tag{30}$$

assuming $|a_2| \leq 1$.

We set $E_{\varepsilon_{x_l}}$ equal to 2^{-q-4} to find the maximum value for t, which gives

$$t = n + m - q - 5 \ . \tag{31}$$

If $t \leq 0$, x_l cannot be truncated.

As with the linear interpolator, we set k for the multipliers and the squarer so that rounding error in each unit is less than each error due to quantization. Since we limited each quantization error to 2^{-q-4}, we limit each rounding error to 2^{-q-5}.

Like the linear interpolator, the LSB weight of the multiplier #1 output is $2^{-n_{f1}-n+k_{m1}+r_{m1}}$, resulting in $E_{rnd\_m2} = 2^{-n_{f1}-n+k_{m1}+r_{m1}-1}$. Setting this equal to 2^{-q-5} gives

$$k_{m1} = n - m - 1 \ . \tag{32}$$

The LSB weight of the squarer output is $2^{-2n+2t+k_{sq}+r_{sq}}$, so $E_{rnd\_sq} = 2^{-2n+2t+k_{sq}+r_{sq}-1}$, where k_{sq} and r_{sq} are k and r for the squarer. The squarer output is multiplied by a_2, where we assume $|a_2| \leq 1$, so the rounding error for the squarer is set equal to 2^{-q-5} to find k_{sq}

$$k_{sq} = 2n - 2t - q - 4 \ . \tag{33}$$

If $|a_2| > 1$, k_{sq} is increased accordingly.

The LSB weight of the multiplier #2 output is $2^{-n_{f2}-2n+2t+k_{sq}+r_{sq}+k_{m2}+r_{m2}}$, so $E_{rnd\_m2} = 2^{-n_{f2}-2n+2t+k_{sq}+r_{sq}+k_{m2}+r_{m2}-1}$. Setting the maximum rounding error equal to 2^{-q-5} gives

$$k_{m2} = q - 2m + 3 \ . \tag{34}$$

5. Coefficient Optimization

After the preliminary design is complete, the coefficients are optimized through exhaustive simulation and coefficient adjustment. First, simulation is done using standard multipliers and squarers to find the minimum coefficient lengths that can be used while maintaining a maximum absolute output error less than 2^{-q}. This reduces the lookup table size. Next, the standard multipliers and squarers are replaced with truncated units, and simulation is done to maximize the number of unformed columns while remaining within design specifications for error. This reduces the area of the interpolator unit.

5.1. Simulation Software

Exhaustive simulation is performed using a custom Java program. This program includes an interactive graphical user interface (GUI) that allows the user to change design parameters, then perform simulation of all possible input values on some or all of the 2^m subintervals. This simulation is bit-accurate, taking into account rounding effects of table lookups and arithmetic units, as well as accounting for the reduction error and correction techniques of truncated multipliers and squarers. The software also includes a routine for adjusting the coefficients to minimize the maximum absolute error of the output. Simplified pseudocode for this routine is as follows

```
i = 0;
while (i < N)
  for delta = -3 to +3 ulps
    simulate exhaustively, using a(i) + delta
    if error is improved, update a(i)
  next delta
  if a(i) was changed
```

```
    i = 0;
  else
    i++;
  end if
end while
```

This method for adjusting coefficients is similar to that presented by Schulte and Swartzlander [10]. One difference is that the coefficient a_i is varied by ± 3 ulps rather than ± 1 ulp, because it is possible for increasing/decreasing values of a_i to produce identical or worse error results before a better result is achieved, due to the nonlinear effects of the arithmetic units. Another difference is that if a_i changes, i is reset to 0 to readjust all lower order coefficients before higher order coefficients are adjusted. Thus, priority for adjustment goes in the order $a_0, a_1, \cdots, a_{N-1}$.

Because the software performs a large number of simulations, there are practical limits to the size of the interpolator that can be simulated. For this research, the program was run on a personal computer with a 2.4 GHz Pentium 4 processor. The time required to run the coefficient optimization routine for a 24-bit interpolator, for example, is on the order of one hour. Since the software was developed for research purposes rather than production purposes, there is a great deal of room for improvement in execution speed. For example, the code could be easily ported to C++, optimized, and run on a faster machine. Exhaustive simulation lends itself well to parallel processing, offering another approach to improve speed. Considering the time and money involved in developing commercial hardware that could benefit from this approach, a few hours of computer time is insignificant.

5.2. Optimization Using Standard Multipliers and Squarers

Using the simulation software just described, the initial hardware design is tested and adjusted to reduce the coefficient lengths as much as possible while maintaining desired accuracy. This is done through trial and error by reducing a single coefficient length by one bit, then running the coefficient optimization routine described above. If the resulting output error range is acceptable, the length of another coefficient is reduced by one bit. If the error is not acceptable, the coefficient is restored to its previous length. This process continues until each coefficient has been reduced to a minimum length. At this point, the size of the lookup table has been minimized, resulting in a final design for a standard interpolator.

If desired, the software allows the user to further explore the design space. Increasing the precision of the multipliers and the squarer by decreasing k below the values previously calculated may allow one or more coefficient lengths to be further reduced, at the expense of larger arithmetic units. The flexibility of the software allows such design tradeoffs to be easily quantified.

5.3. Optimization Using Truncated Multipliers and Squarers

After the size of the lookup table is minimized as described in the previous section, the multipliers and the squarer are replaced with truncated units. The goal is to maximize the number of unformed columns of partial product bits in each unit without exceeding the interpolator output error bounds. This reduces the area required for the interpolator.

As with finding the minimum coefficient lengths, the maximum value of r for each unit is obtained by trial and error. Note that for a standard multiplier or squarer, $r = 0$ because all partial products are formed. As r is increased for a truncated unit, k must be decreased by an equal amount in order to maintain the precision and weight of the output product or square.

A good place to start when using truncated units with constant correction is to set $k = 4$ and $r = k_{std} - 4$, where k_{std} is the value of k used for the original standard multiplier or squarer. For variable correction units, which are generally more accurate for equivalent values of r, one should start by setting $k = 3$ and $r = k_{std} - 3$. The coefficient optimization routine is run after each time k and r are changed, and the error bounds are found through exhaustive simulation. As with coefficient length reduction described above, k is increased and r is decreased for each unit until coefficient optimization fails to produce an acceptable range of output error. At this point the previous acceptable values for k and r are restored and the process continues until the maximum number of unformed columns has been found for each unit. When this is complete, the best truncated interpolator design has been found.

6. Results

In order to evaluate the methods presented in the preceding sections, a number of interpolators were developed. Each unit implements the reciprocal function, $y = f(x) = 1/x$. Table 1 presents results for 12-, 16-, and 20-bit linear interpolators, and Table 2 presents results for 16-, 20-, and 24-bit quadratic interpolators.

For each unit, the input x is assumed to be reduced to the interval $[1, 2)$. The most significant bit of x is always '1', so it is not input to the interpolator. Therefore, 1 ulp is 2^{-n-1}. Each unit is designed with an upper bound on the maximum absolute error of 1 ulp, so $q = n + 1$.

Each interpolator is first designed using standard multipliers and squarers, and the coefficient lengths are minimized as described previously. The results of each standard design are given, listed as "Standard" under type. The multipliers and squarers are then replaced by truncated units with constant correction, listed as "Constant" under type.

12-bit input. $n = 11$, $m = 5$, $n_0 = 14$, $n_1 = 11$ Lookup Table Size $= 2^5(12 + 10) = 704$ bits							
Type	k_{m1}	r_{m1}	E_{min} (ulps)	E_{max} (ulps)	σ_E^2 (ulps)	pp's	reduction
Standard	5	0	-0.832	0.757	0.101	66	n/a
Constant	0	5	-0.832	0.871	0.105	51	22.7 %
Variable	0	5	-0.832	0.921	0.102	56	15.2 %

16-bit input. $n = 15$, $m = 7$, $n_0 = 18$, $n_1 = 9$ Lookup Table Size $= 2^7(16 + 8) = 3072$ bits							
Type	k_{m1}	r_{m1}	E_{min} (ulps)	E_{max} (ulps)	σ_E^2 (ulps)	pp's	reduction
Standard	6	0	-0.892	0.964	0.113	72	n/a
Constant	2	4	-0.961	0.921	0.114	62	13.9 %
Variable	2	4	-0.958	0.932	0.112	66	8.3 %

20-bit input. $n = 19$, $m = 10$, $n_0 = 22$, $n_1 = 11$ Lookup Table Size $= 2^10(20 + 10) = 30720$ bits							
Type	k_{m1}	r_{m1}	E_{min} (ulps)	E_{max} (ulps)	σ_E^2 (ulps)	pp's	reduction
Standard	8	0	-0.961	0.978	0.114	99	n/a
Constant	4	4	-0.996	0.987	0.114	89	10.1 %
Variable	3	5	-0.962	0.968	0.113	89	10.1 %

Table 1. Linear interpolators, $f(x) = 1/x$.

Finally, constant correction units are replaced by variable correction units, listed as "Variable" under type. In addition to design parameters, the upper and lower output error bounds, E_{min} and E_{max} respectively, and the variance of error, σ_E^2, are listed for each interpolator.

The size of the lookup table is given for each interpolator. A close look at the coefficients in the tables shows that the first two bits of coefficient a_0 are constant and do not need to be stored. Likewise, the first bit of a_1 and the first bit of a_2 are constant. Since 2^m entries are required, the size of the lookup table is $2^m((n_0 - 2) + (n_1 - 1))$ for a linear interpolator, and is $2^m((n_0 - 2) + (n_1 - 1) + (n_2 - 1))$ for a quadratic interpolator.

In order to estimate the area reduction due to using truncated multipliers and squarers, the total number of multiplier and squarer partial products are given for each interpolator. Although there are several ways to add all the partial products to produce the output, y, each method will benefit from having fewer partial products. Tree multipliers and array multipliers, for example, would require fewer adder cells. As shown by Schulte, Stine, and Jansen [8], power reduction is comparable to area reduction for truncated multipliers. This is a result of reduced hardware as well as reduced switching activity, since no carries are propagated out of unformed columns.

From these results, it can be seen that a significant reduction in the number of the partial products can be achieved without exceeding the design specification of ± 1 ulp error. For the linear interpolator designs that were developed, a reduction of 8.3 % to 22.7 % is obtained, and for the quadratic designs a reduction of 16.8 % to 31.1 % is realized.

Constant correction truncated multipliers and squarers with r_{cc} unformed columns have the same number of unformed partial products as variable correction units with $r_{vc} = r_{cc} + 1$ unformed columns. For a given number of unformed partial products, constant correction and variable correction units have similar error characteristics. Due to the nonlinear nature of these units, however, the actual error bounds of an interpolator using them can only be determined through exhaustive simulation. Constant correction units are often the easiest to implement, because using variable correction can increase the maximum height of the multiplication or squaring matrix, potentially increasing overall delay of the unit if a tree structure is used to add the partial products. However, as shown in [5], variable correction is easily implemented in an array structure, and may offer better accuracy.

7. Conclusion

A technique for designing function interpolators using truncated multipliers and squarers has been presented. A number of linear and quadratic interpolators with less than 1 ulp error are developed and presented as an example. The technique is general, and can be used for any function, any error bound, and precision is limited only by simulation time. It can be easily adapted for use with other designs for function approximation, such as those presented in [1–3, 6, 7, 12].

The results show that truncated multipliers and squarers can be used in function interpolators to significantly reduce area. Although truncated interpolators have a slightly larger

16-bit input. $n = 15$, $m = 4$, $n_0 = 19$, $n_1 = 15$, $n_2 = 12$, $t_{sq} = 0$
Lookup Table Size $= 2^4(17 + 14 + 11) = 672$ bits

Type	k_{m1}	r_{m1}	k_{m2}	r_{m2}	k_{sq}	r_{sq}	E_{min} (ulps)	E_{max} (ulps)	σ_E^2 (ulps)	pp's	reduction
Standard	10	0	11	0	10	0	−0.926	0.993	0.102	375	n/a
Constant	4	6	4	7	3	7	−0.906	0.993	0.104	314	16.8 %
Variable	1	9	0	11	1	9	−0.967	0.972	0.103	268	29.5 %

20-bit input. $n = 19$, $m = 6$, $n_0 = 23$, $n_1 = 16$, $n_2 = 10$, $t_{sq} = 0$
Lookup Table Size $= 2^6(21 + 15 + 9) = 2880$ bits

Type	k_{m1}	r_{m1}	k_{m2}	r_{m2}	k_{sq}	r_{sq}	E_{min} (ulps)	E_{max} (ulps)	σ_E^2 (ulps)	pp's	reduction
Standard	12	0	11	0	14	0	−0.869	0.854	0.098	419	n/a
Constant	2	10	3	8	2	12	−0.945	0.952	0.100	292	31.1 %
Variable	1	11	2	9	1	13	−0.934	0.972	0.102	292	31.1 %

24-bit input. $n = 23$, $m = 7$, $n_0 = 26$, $n_1 = 20$, $n_2 = 13$, $t_{sq} = 1$
Lookup Table Size $= 2^7(24 + 19 + 12) = 7040$ bits

Type	k_{m1}	r_{m1}	k_{m2}	r_{m2}	k_{sq}	r_{sq}	E_{min} (ulps)	E_{max} (ulps)	σ_E^2 (ulps)	pp's	reduction
Standard	15	0	13	0	16	0	−0.951	0.967	0.099	622	n/a
Constant	4	11	4	9	4	12	−0.951	0.967	0.099	475	24.1 %
Variable	4	11	3	10	3	13	−0.997	0.947	0.099	486	22.3 %

Table 2. Quadratic interpolators, $f(x) = 1/x$.

range of error compared to standard interpolators, the additional error is small. The results show that for an error specification of $\pm$ 1 ulp, truncated interpolators can be designed that meet that specification while using the same size lookup table as a comparable optimized standard interpolator.

References

[1] M. G. Arnold and M. D. Winkel. A Single-Multiplier Quadratic Interpolator for LNS Arithmetic. In *Proceedings of the 2001 International Conference on Computer Design*, pages 178–183, Austin, TX, September 2001.

[2] J. Cao, B. W. Y. Wei, and J. Cheng. High-Performance Architectures for Elementary Function Generation. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pages 136–144, Vail, CO, June 2001.

[3] D. Das Sarma and D. W. Matula. Faithful Interpolation in Reciprocal Tables. In *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pages 82–91, Asilomar, CA, July 1997.

[4] J.H. Mathews. *Numerical Methods for Computer Science, Engineering, and Mathematics*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] E. J. King and E. E. Swartzlander, Jr. Data-Dependent Truncation Scheme for Parallel Multipliers. In *Proceedings of the 31st Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1178–1182, Pacific Grove, CA, November 1997.

[6] J. A. Pineiro and J. D. Bruguera. High-Speed Double Precision Computation of Reciprocal, Division, Square Root, and Inverse Square Root. *IEEE Transactions on Computers*, 51(12):1377–1388, December 2002.

[7] J. A. Pineiro, J. D. Bruguera, and J. M. Muller. Faithful Powering Computation Using Table Look-Up and a Fused Accumulation Tree. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pages 40–47, Vail, CO, June 2001.

[8] M. J. Schulte, J. E. Stine, and J. G. Jansen. Reduced Power Dissipation Through Truncated Multiplication. In *Proceedings of the IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, pages 61–69, Como, Italy, March 1999.

[9] M. J. Schulte and E. E. Swartzlander, Jr. Truncated Multiplication with Correction Constant. In *VLSI Signal Processing VI*, pages 388–396, Eindhoven, Netherlands, October 1993. IEEE Press.

[10] M. J. Schulte and E. E. Swartzlander, Jr. Hardware Designs for Exactly Rounded Elementary Functions. *IEEE Transactions on Computers*, 43(8):964–973, August 1994.

[11] E. E. Swartzlander, Jr. Truncated Multiplication with Approximate Rounding. In *Proceedings of the 33rd Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1480–1483, Pacific Grove, CA, October 1999.

[12] N. Takagi. Generating a Power of an Operand by a Table Look-Up and a Multiplication. In *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pages 126–131, Asilomar, CA, July 1997.

[13] E. G. Walters III, M. J. Schulte, and M. G. Arnold. Truncated Squarers with Constant and Variable Correction. In *Proceedings of the SPIE: Advanced Signal Processing Algorithms, Architectures, and Implementations XIV*, volume 5559, pages 40–50, Denver, CO, August 2004.

[14] K. E. Wires, M. J. Schulte, L. P. Marquette, and P. I. Balzola. Combined Unsigned and Two's Complement Squarers. In *Proceedings of the 33rd Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1215–1219, Pacific Grove, CA, October 1999.