Reciprocal Approximation Theory with Table Compensation

Albert A. Liddicoat and Michael J. Flynn

Technical Report No. CSL-TR-00-790

January 2000

Reciprocal Approximation Theory with Table Compensation

by

Albert A. Liddicoat and Michael J. Flynn

Technical Report No. CSL-TR-00-790

January 2000

Computer Systems Laboratory

Stanford University

Gates Building 3A, Room 332

Stanford, California 94305-9030

pubs@shasta.stanford.edu

Abstract

[Sch93] demonstrates the reuse of a multiplier partial product array (PPA) to approximate higher order functions such as the reciprocal, division, and square root. Schwarz generalizes this technique to any higher order function that can be expressed as $A * B = C$. Using this technique, the height of the PPA increases exponentially to increase the result precision. Schwarz added compensation terms within the PPA to reduce the worst case error.

This work investigates the approximation theory of higher order functions without the bounds of multiplier reuse. Additional techniques are presented to increase the worst case precision for a fixed height PPA.

A compensation table technique is presented in this work. This technique combines the approximation computation with a compensation table to produce a result with fixed precision. The area-time tradeoff for three design points is studied. Increasing the computation decreases the area needed to implement the function but also increases the latency.

Finally, the applicability of this technique to the bipartite ROM reciprocal table is discussed. We expect that this technique can be applied to the bipartite ROM reciprocal table to significantly reduce the hardware area needed at a minimal increase in latency.

In addition, this work focuses on hardware reconfigurability and the ability of the hardware unit to be used to perform multiple higher order functions efficiently. The PPA structure can be used to approximate several higher order functions that can be expressed as a multiply.

Key Words and Phrases: Approximation Theory, PPA, Divide, Reciprocal,

Contents

1 Introduction **1**

2 Compensation Table **3**

2.1 Area and Latency Analysis 5

3 Computation complexity versus table size study **6**

3.1 Determining optimal design points 6

3.2 Maximizing the precision from a fixed height PPA 8

3.3 Applying the precision enhancement techniques to the three design points 16

4 PPA Computation Technique applied to the Bipartite Reciprocal table **22**

5 Conclusions **23**

List of Figures

1 Partial Product Array (PPA) in dot notation for the reciprocal function approximation as proposed by [Sch93] for a booth 2 multiplier with height 27. The worst case accuracy computed is 9.17 bits . . . 2

2 The error distribution for the reciprocal approximation technique using a PPA height of 27 and a 13-bit input producing a 12-bit result. . . . 3

3 The PPA in dot notation for the reciprocal function after the constant adjustments have been made to bias error. . . . 4

4 A (3,2) Counter and CPA combine the PPA and the compensation table outputs to produce the final result. . . . 4

5 Wallace tree for reducing PPA column with nine partial products. . . . 7

6 Delay and worst case precision as a function of PPA height. . . . 7

7 Maximum height reduction technique. . . . 9

8 Reciprocal PPA structure with Boolean elements [Sch93]. The dashed line represents the desired PPA height while the circles indicate which terms maybe manipulated by the proposed technique. . . . 9

9 Reciprocal PPA structure with Boolean elements after height manipulation technique. 10

10 Reciprocal PPA structure with Boolean elements moved to less significant columns and duplicated. . . . 11

11 Reciprocal PPA structure with Boolean elements after logical manipulations and the column q_8 excess term approximation. . . . 12

12 The difference between the computed and the round to nearest result. The compensation table must account for the difference between the maximum and minimum points on each curve. . . . 13

13 Distribution of the difference between the computed result and the round to nearest result for all 2^13 inputs. . . . 15

14 Final PPA configuration for the reciprocal approximation. . . . 15

15 Final hardware configuration for reciprocal unit with PPA of height 9. . . . 16

16 Final hardware configuration for reciprocal unit with PPA of height 18. . . . 18

17 Final hardware configuration for reciprocal unit with PPA of height 36. 20

18 The Area-Time tradeoff for a reciprocal Unit with 12-bit output precision. An increase in computation increases the reciprocal latency but decreases the compensation table size and therefore decreases the area needed implement the unit. 21

19 A $j+2=3k+1$ bits-in $j=3k-1$ bits-out Faithful Reciprocal table. [DM95] . . . 22

1 Introduction

Renato Stefanelli [Ste72] expressed division and the reciprocal function as the inverse of multiplication ($Q = A/B -> Q * B = A$ and $Q = 1/B -> Q * B = 1$). The unknown input Q is multiplied with the input operand B and set equal to the input operand A or 1. Then by solving a set of linear equations the quotient Q can be determined. Each quotient digit is dependent on all of the more significant quotient digits. The latency for this technique grows linearly with the number of digits to compute.

David Mandelbaum [Man90] enhanced Stefanelli's technique by removing the recursion between digits with back substitution. The latency for the enhanced technique grows logarithmically with the number of digits to compute since all the digits are computed simultaneously. Mandelbaum expressed his equations in the form of a partial product array (PPA). He also applied this technique to other higher order functions such as the square root, natural log, and exponential functions.

Eric Schwarz [Sch93] further enhanced these techniques by developing a method to implement the approximation theory on a standard direct multiplier or booth 2 multiplier PPA. Schwarz also developed compensation terms to improve the average and worst case error.

Figure 1 depicts the reciprocal function approximation PPA structure proposed by Schwarz for implementation on a booth 2 multiplier. The PPA includes the compensation terms used for the reciprocal approximation. 9.17 bits of accuracy or better can be achieved using this booth 2 multiplier PPA with height of 27. Schwarz also shows that 12.0 bits of accuracy can be achieved using a direct multiplier PPA of height 53.

In this work we propose several techniques, including a compensation table, to increase the worst case accuracy of a result produced by a fixed PPA height. In section 2, the compensation table technique is applied to the PPA shown in figure 1. Using the compensation table technique the worst case accuracy is improved from 9.17 bits to 12 bits. In section 3, the PPA computation versus table size is studied. By increasing the PPA height the size of the compensation table required to obtain a fixed precision result decreases. However increasing the PPA height increases the latency of the approximation computation. Therefore, there is an area-time tradeoff to arrive at a fixed precision result. Section 4, then looks at applying the technique discussed in this paper to the primary table in the faithful bipartite ROM reciprocal tables proposed by Das Sarma and Matula [DM95]. The final section of this paper provides some conclusions based on this work.

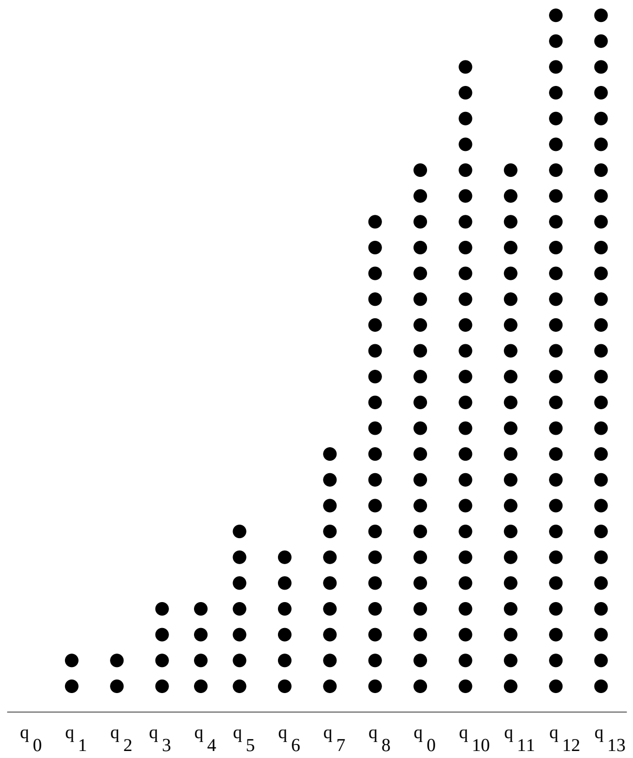

Figure 1: Partial Product Array (PPA) in dot notation for the reciprocal function approximation as proposed by [Sch93] for a booth 2 multiplier with height 27. The worst case accuracy computed is 9.17 bits

2 Compensation Table

In this work we propose several techniques to increase the worst case accuracy of the reciprocal approximation produced by a fixed height PPA. In this section, the compensation table technique is proposed and applied to the booth 2 PPA of height 27 as shown in figure 1. The compensation table of size $2^{13}x5$ bits is required to achieve a 12-bit accurate result rounded to the nearest digit.

Figure 2 shows the histogram of the difference between the reciprocal approximation technique computed on a PPA of height 27 and the round to nearest 12-bit result. The error is normally distributed with a minimum value of -10 and a maximum value of 9. By subtracting the maximum error from the constants in the PPA, the error can be shifted to the range of -19 to 0. Therefore, a positive compensation table value of 5-bits may be directly added to the computed result to produce an output result with 12 bits of accuracy. The 5-bit compensation table stores an offset between 0 and $(2^5 - 1 = 31)$ for each of the 2^{13} possible input values.

Figure 3 shows the PPA for the reciprocal function in dot notation after the changes have been made to the PPA constants to bias the error. The computed approximation from the PPA and the compensation table word can be summed together by one (3,2) counter stage and one final carry propagate add (CPA). The final result is the reciprocal of the input accurate to 12 bits of precision. Figure 4 shows the proposed hardware structure to combine the PPA output with the compensation table, and then reduce the result with one (3,2) counter stage and a 12-bit CPA.

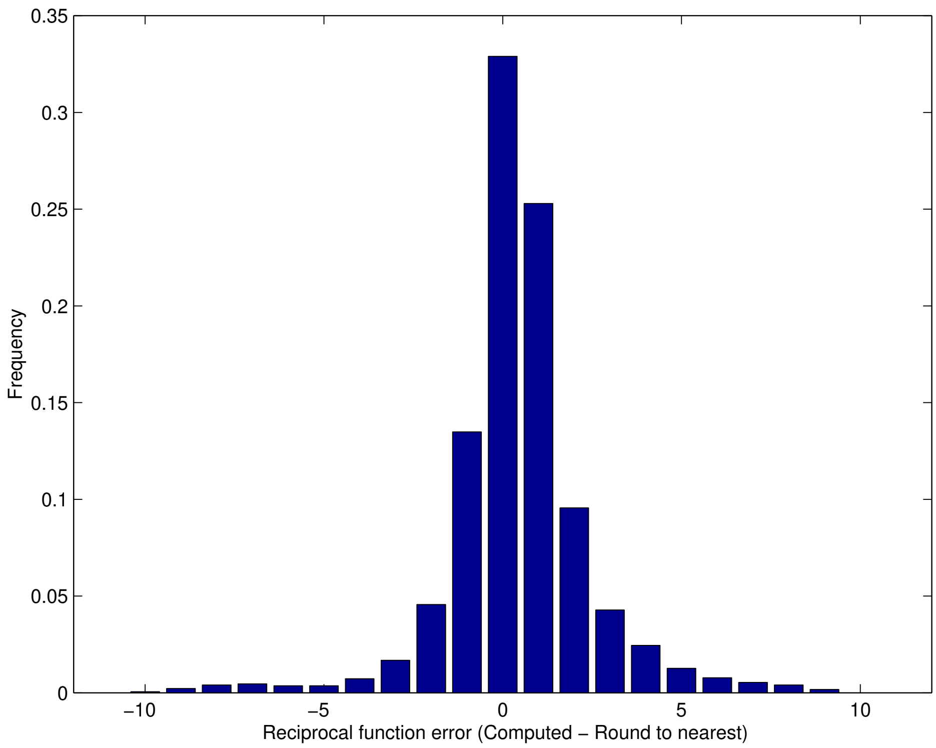

Figure 2: The error distribution for the reciprocal approximation technique using a PPA height of 27 and a 13-bit input producing a 12-bit result.

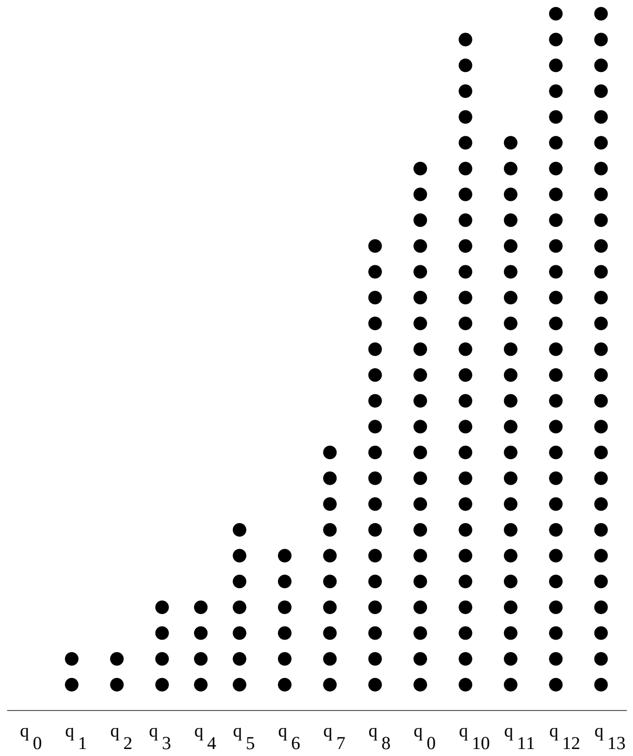

Figure 3: The PPA in dot notation for the reciprocal function after the constant adjustments have been made to bias error.

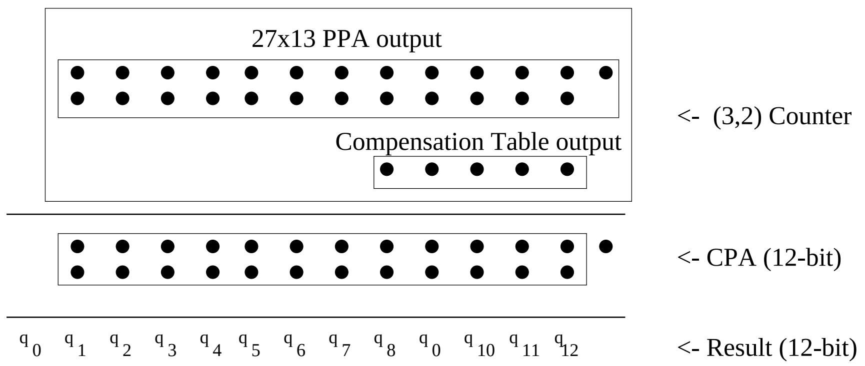

Figure 4: A (3,2) Counter and CPA combine the PPA and the compensation table outputs to produce the final result.

2.1 Area and Latency Analysis

Comparing the approximation theory with compensation table technique to a single lookup table we find that the proposed method has two main benefits. First the area required to implement the proposed technique is significantly smaller than the area of a single ROM table. Additionally, the PPA hardware required for the approximation can be reconfigurable and used to approximate a variety of functions such as square root, division, and the transcendental functions. Therefore, the compensation table is the only area exclusively dedicated to perform one specific arithmetic function. The disadvantage however is performance. It requires more time to compute the approximation than to simply access a large table. This design point with a PPA of height 27 and a $2^{13}x5$ bit compensation table of was chosen to demonstrate the compensation table technique. The design point was not selected as an optimum design point based on area or performance. Table 2.1 summarizes the area required and latency for the reciprocal function approximation technique and compares it to a single ROM lookup table.

In the approximation theory technique 1179 logic gates replace more than 57kbits of ROM. The latency of the approximation theory technique with a compensation table is approximately 28 gate delays. This is approximately 18 gate delays greater than the latency of a single lookup table.

Table 1: Reciprocal Function Area requirements and Latency

Technique	Functional Block	Area (gates)	ROM (bits)	Gate Delays
Approx	Input Logic	200		2
	PPA's	895		16
	Compensation Table		40,960	9 (overlapped)
	(3,2) Counter	60		2
	CPA (12 bits)	24		8
	Total	1179	40,960	28
Single Tbl	Total		98,304	10

3 Computation complexity versus table size study

3.1 Determining optimal design points

In this section we evaluate the tradeoff between increasing the PPA height, and therefore the precision of the computation, and decreasing the table width while producing a fixed precision result. To understand the area latency tradeoff we identified three design points that optimize the area latency product.

We selected a Wallace tree PPA implementation. The delay of a Wallace tree is proportional to $log_{\frac{3}{2}}h$. Figure 5 shows a Wallace tree that can reduce nine partial products in four CSA delays. The tree can be doubled by combining two 9 input trees with two CSA's to produce a tree that can reduce 18 partial products in 6 CSA stage delays. The 18 input tree requires 16 CSA's per column in the PPA. An 18 input partial product tree can be doubled in a similar fashion, again by combining two 18 input trees with two additional CSA's. The 36 partial product tree requires 8 CSA delays and a total of 32 CSA's per column in the PPA. Figure 6 shows the Wallace tree delay in terms of CSA stages for varying PPA heights. One can observe the discrete changes in delay that increase logarithmically with an increase in the PPA height.

The reciprocal function PPA is determined by back-solving the PPA for the equation $QxB = 1$. The equations for quotient bits $q(0)$ to $q(13)$ are then found using the technique described in [Sch93]. The worst case accuracy and PPA height is then determined for approximations using bits $q(0)$ to $q(n)$. As n increases the worst case accuracy improves and the PPA height required to compute the approximation increases. The worst case precision as a function of the required PPA height is shown in Figure 6. Only the six Pareto points are plotted. The worst case precision also grows logarithmically with the PPA height. Alternately stated, the required PPA height grows exponentially with the worst case result precision. The figure also shows the ratio of the CSA delay to the worst case precision of the result. Here we notice that the delay per bit for PPA with height greater than five is fairly constant at about 1.4 CSA delays per bit of precision. Since the precision is increasing monotonically and the delay increases discretely, the optimal performance points (CSA delays per bit of precision) are at the corners of the CSA delay curve. The corners are where the largest precision can be obtained on the PPA delay plateau. In the following subsections the approximation technique with table compensation is evaluated for the three corner points where the PPA heights are 9, 18, and 36.

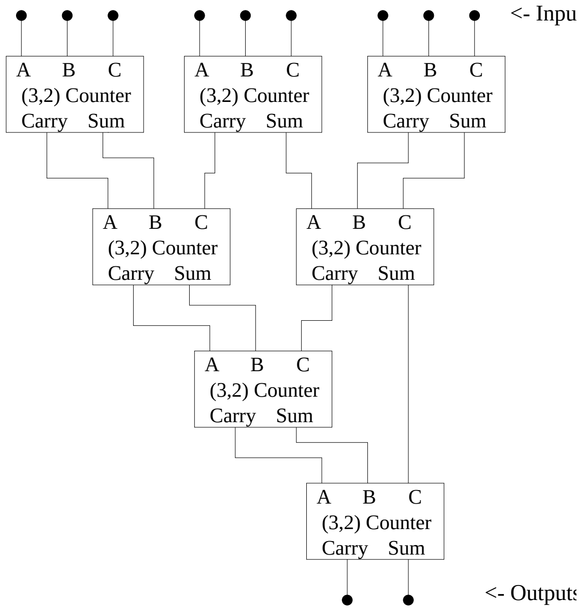

Figure 5: Wallace tree for reducing PPA column with nine partial products.

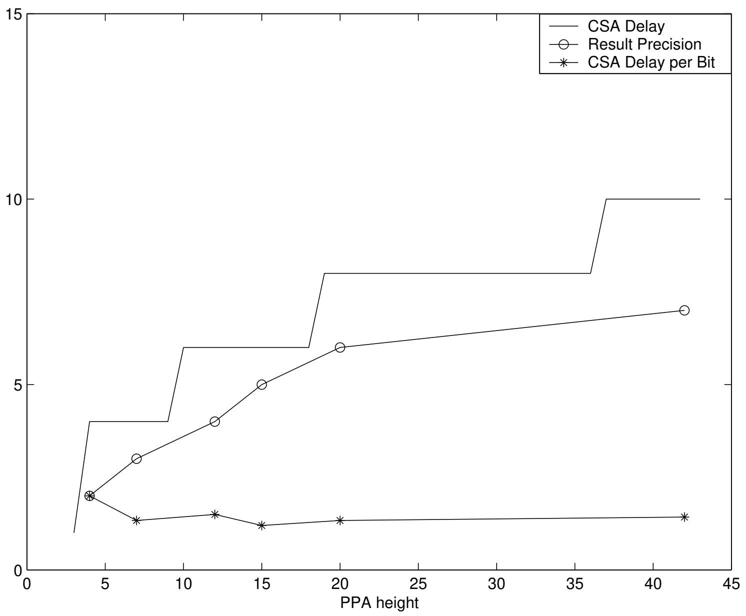

Figure 6: Delay and worst case precision as a function of PPA height.

3.2 Maximizing the precision from a fixed height PPA

We propose several techniques to reduce the worst case error of the approximation. Schwarz proposes using boolean compensation terms within the PPA to reduce the error but recommends that future work be done in the area of reducing the worst case error.

We have developed three additional techniques to reduce the worst case error. First logically equivalent manipulations can be used to reduce the height of the largest columns. Reducing the height of columns that exceed the desired PPA height reduces the PPA error as opposed to truncating the boolean terms. Second, a technique is proposed that decreases the worst case error by approximating the boolean terms that would otherwise not fit into a PPA of a given height. Finally, selective boolean term elimination can be used as a very fine grain manipulation to accommodate the error biasing constants and the boolean compensation terms. Each of these techniques is discussed below and they are applied to the design point with PPA height of 9. Then the proposed techniques to optimize the approximation precision and the table compensation are applied to the remaining two design points. The area and latency results from the three design points are then presented.

3.2.1 Logically equivalent manipulations

Schwarz identifies two boolean and two algebraic techniques to reduce the number of terms within a column. Since the PPA algebraically adds the terms in a column, standard boolean reductions do not work. Schwarz also proposed a technique to shift terms from a column to a less significant column by doubling the terms since the weight of the adjacent less significant column is 1/2 that of the current column. For example the boolean term $b_1\&b_2$ can be moved from column 7 into column 8 if the term is duplicated in column 8. This manipulation is equivalent since algebraically $b_1\&b_2 = \frac{1}{2}(b_1\&b_2) + \frac{1}{2}(b_1\&b_2)$. All of the techniques proposed by Schwarz allow the array to be reduce without affecting the computed result.

Here we propose a new technique that allows terms to be combined across columns so that terms in the largest columns can be migrated to more significant columns. This technique can be used to reduce the height of the largest columns. For example the q_7 column has ten terms as can be seen in figure 1 and we are attempting to fit the design into a PPA with a maximum height of nine. Figure 7 a) uses a Karnaugh map to show that the min-terms from two of the boolean terms in the q_7 column overlap. With a straight boolean reduction one could eliminate the $b_3\&b_4'\&b_5$ term. However, since the terms are summed together algebraically the weight of each term must be considered. Figure 7 b) shows that the terms can be separated into two independent boolean terms, $b_3'\&b_4'\&b_5$ and $b_3\&b_4'\&b_5$. The first term has weight one and the second term has weight two. Now the two terms can be represented by the first term with weight one in the q_7 column and the second term with weight two in the q_6 column. This manipulation is possible since the q_6 column implicitly has twice the weight of the q_7 column. In this example the total number of terms remains the same but one of the terms was moved to a more significant column in a less populated portion of the PPA.

A dual of this manipulation can be made with the boolean 'OR' terms. For example in the column for q_5 the terms $(d_3' + d_5)$ and $(d_2' + d_3' + d_5)$ can be replaced by the term $(d_3' + d_5)$ in the q_4 column,

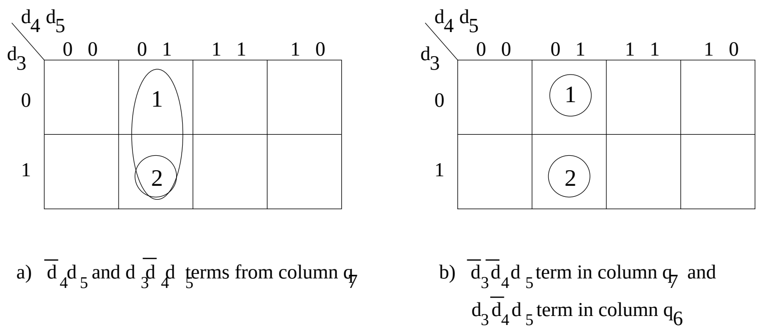

Figure 7: Maximum height reduction technique.

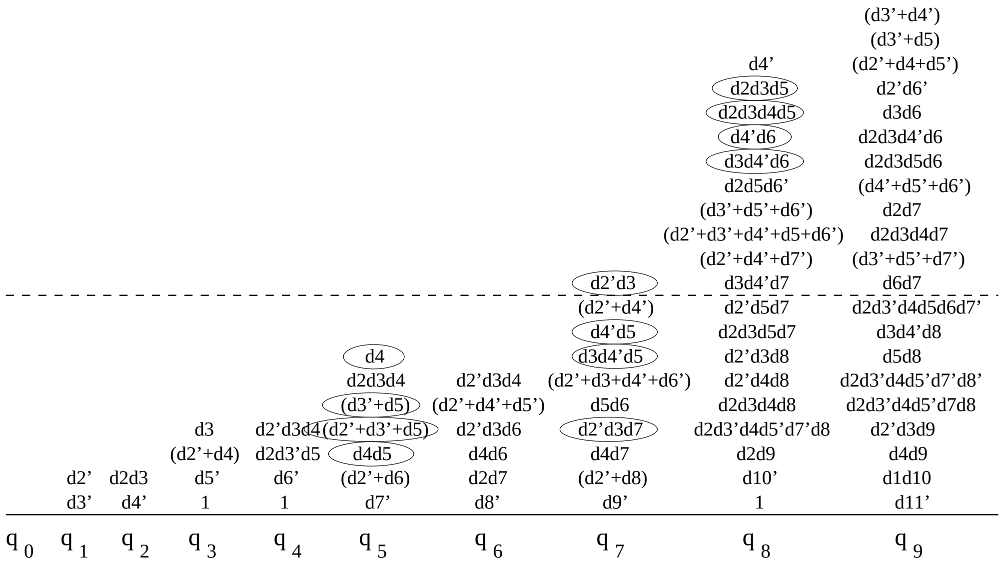

Figure 8: Reciprocal PPA structure with Boolean elements [Sch93]. The dashed line represents the desired PPA height while the circles indicate which terms maybe manipulated by the proposed technique.

since if either d_3' or d_5 is true both terms will be true in the q_5 column. Both of the terms in the q_5 column when true contribute a '1' to the PPA q_5 column which is equivalent to one term contributing a '1' in the q_4 column. Additionally, the case when d_3' and d_5 are false but d_2' is true needs to contribute a '1' to the q_5 column in the PPA. Therefore, the term $(d_2' \& d_3 \& d_5')$ must remain in the q_5 column. Effectively column q_5 was reduced by one term and column q_4 was increased by one term, similarly to the 'AND' boolean term manipulation described in the previous paragraph.

This technique does not change the error of the result since the manipulations are both logically and algebraically equivalent. Figure 8 shows the boolean terms for the quotient bits zero through nine. The terms in the q_5 to q_8 columns that can be rearranged with the proposed technique have been circled. The dashed line indicates the desired PPA height. Figure 9 shows the array after applying the PPA column height reduction technique. Now quotient bits q_0 through q_7 will fit within the PPA of height 9 and the height of the q_8 column has been reduced.

This technique can be generalized to manipulate several terms from various columns to both reshape

q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
									(d3’+d4’)
									(d3’+d5)
									(d2’+d4+d5’)
								d4’	d2’d6’
								d2d3d5	d3d6
								d2d3d4d5	d2d3d4’d6
								d3’d4’d6	d2d3d5d6
								d2d5d6’	(d4’+d5’+d6’)
								(d3’+d5’+d6’)	d2d7
								(d2’+d3’+d4’+d5+d6’)	d2d3d4d7
								(d2’+d4’+d7’)	(d3’+d5’+d7’)
								d3d4’d7	d6d7
							d3d4’d6	d2’d5d7	d2d3’d4d5d6d7’
						d3d4’d5	(d2’+d4’)	d2d3d5d7	d3d4’d8
						d2’d3d7	d3’d4’d5	d2’d3d8	d5d8
					d4	d2’d3d4	(d2’+d3+d4’+d6’)	d2’d4d8	d2d3’d4d5’d7’d8’
				(d3’+d5)	d2d3d4	(d2’+d4’+d5’)	d5d6	d2d3d4d8	d2d3’d4d5’d7d8
			d3	d2’d3d4	d2’d3d5’	d2’d3d6	d2’d3d7’	d2d3’d4d5’d7’d8	d2’d3d9
			(d2’+d4)	d2d3’d5	d4d5	d4d6	d4d7	d2d9	d4d9
	d2’	d2d3	d5’	d6’	(d2’+d6)	d2d7	(d2’+d8)	d10’	d1d10
	d3’	d4’	1	1	d7’	d8’	d9’	1	d11’

1

Figure 9: Reciprocal PPA structure with Boolean elements after height manipulation technique.

the PPA. In addition, this technique can be expanded to look for reductions across multiple columns simultaneously. For example for or five terms from adjacent columns maybe able to be replaced by fewer terms with more significant weights. A CAD tool can be designed that uses the proposed technique to methodically optimize the number of terms in the PPA as well as the shape of the PPA.

3.2.2 Partial Product Array boolean term approximation

Since we are using a compensation table the difference from the maximum error and the minimum error for the PPA computation from the round to nearest result must be less than 2^n where n is the bit width of the compensation word stored in the table. For example, a compensation table with 6-bit entries requires that the difference between the maximum error and minimum error be less than $2^6 = 64$ units. The constant terms in the PPA later must be chosen so that the error is biases from $-2^n - 1 to 0$

If the PPA is truncated after q7 the difference from the maximum error to the minimum error is 138 units, thus requiring a compensation table of 8 bits. Effectively the PPA would only be computing 4-bits of precision. If the table was truncated at the dashed line in figure 9 then the difference would be 103 units, requiring a compensation table with a 7-bit word size. Clearly, neither of these approaches will result in the highest precision from a PPA with a fixed height.

Some of the terms in column q_8 that exceed the PPA height limit may be moved to columns q_9 and q_{10} by doubling and quadrupling the terms respectively. This technique is discussed in [Sch93]. Moving some of the terms to the less significant columns is a logically equivalent operation. However, not all of the boolean terms in column q_8 that exceed the height limit of 9 can be moved to columns q_9 and q_{10} since these columns are also limited to a height of 9. Therefore, some of the boolean terms contribute the proper value to the result while other boolean terms are truncated, thus increasing the worst case and average error. Figure 10 shows the PPA after several of the terms from column q_8 have been moved to column q_9 and q_{10}. The boxes indicate the terms that have been doubled and quadrupled. The difference between the maximum error and the minimum error for the PPA shown in figure 10 is 100 units.

If all the terms in column q_8 were included in the PPA the maximum difference would be 76 unit. However, this would require a PPA with height greater than 9. But by including all the terms in column q_8 the difference between the maximum and minimum error is 25% less than that which is obtained using the technique of doubling and quadrupling some of the terms as described in [Sch93].

								d4'		
								d2d3d5		
						d3d4'd6		d2d3d4d5	d2d3d5d7	d2d3d5d7
					d3d4'd5	(d2'+d4')		d3'd4'd6	d2d3d4d8	d2'd3d8
					d2'd3d7	d3'd4'd5		d2d5d6'	d2d3d4d8	d2'd3d8
				d4	d2'd3d4	(d2'+d3+d4'+d6')		(d3'+d5'+d6')	d2d3'd4d5'd7'd8	d2'd3d8
			(d3'+d5)	d2d3d4	(d2'+d4'+d5')	d5d6		(d2'+d3'+d4'+d5+d6')	d2d3'd4d5'd7'd8	d2'd3d8
		d3	d2'd3d4	d2'd3d5'	d2'd3d6	d2'd3d7'		(d2'+d4'+d7')	d2d9	d2'd4d8
		(d2'+d4)	d2d3'd5	d4d5	d4d6	d4d7		d3d4'd7	d2d9	d2'd4d8
d2'	d2d3	d5'	d6'	(d2'+d6)	d2d7	(d2'+d8)		d2'd5d7	d10'	d2'd4d8
d3'	d4'	1	1	d7'	d8'	d9'		1	d10'	d2'd4d8
q_0 q_1	q_2	q_3	q_4	q_5	q_6	q_7		q_8	q_9	q_{10}

1

Figure 10: Reciprocal PPA structure with Boolean elements moved to less significant columns and duplicated.

							d3d4'd6	d4'	d3d4'd7	d3d4'd7
						d3d4'd5	(d2'+d4')	d2d3d5	d2'd5d7	d2'd5d7
						d2'd3d7	d3'd4'd5	d2d3d4d5	d2d3d5d7	d2d3d5d7
					d4	d2'd3d4	(d2'+d3+d4'+d6')	d3'd4'd6	d2'd3d8	d2'd3d8
				(d3'+d5)	d2d3d4	(d2'+d4'+d5')	d5d6	d2d5d6'	d2'd4d8	d2'd4d8
			d3	d2'd3d4	d2'd3d5'	d2'd3d6	d2'd3d7'	(d3'+d5'+d6')	d2d3d4d8	d2d3d4d8
			(d2'+d4)	d2d3'd5	d4d5	d4d6	d4d7	(d2'+d3'+d4'+d5+d6')	d2d3'd4d5'd7'd8	d2d3'd4d5'd7'd8
	d2'	d2d3	d5'	d6'	(d2'+d6)	d2d7	(d2'+d8)	(d2'+d4'+d7')	d2d9	d2d9
	d3'	d4'	1	1	d7'	d8'	d9'	1	d10'	d10'
q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}

1

Figure 11: Reciprocal PPA structure with Boolean elements after logical manipulations and the column q_8 excess term approximation.

We propose a new technique to reduce the difference between the maximum and minimum error. This technique approximates the terms in the column that exceeds the PPA height. Rather than replicating some of the terms to maintain logically equivalence and truncating other terms, this technique approximates all of the terms that exceed the fixed PPA height. Let BTq_i represent a boolean term in the column q_i and BTq_{i+1} represents the same boolean term in the column q_{i+1}. Equation 1 indicates that a term in the ith column can be replaced by an infinite series of the same term in columns $i + 1to\infty$. Equation 2 shows that the infinite series of terms can be approximated by a finite series of terms in the columns $i + 1ton$.

$$BTq_i = BTq_{i+1} + BTq_{i+2} + BTq_{i+3} + ... + BTq_\infty \tag{1}$$

$$BTq_i \approx BTq_{i+1} + BTq_{i+2} + BTq_{i+3} + ... + BTq_n \tag{2}$$

The above relation allows us to move the 9 boolean terms that exceed the desired PPA height in column q_8 to columns q_9 through q_n. Figure 11 shows the PPA after approximating the excess terms in column q_8 with $n = 10$. After applying this technique the worst case difference is now 75 units. We found that only two columns were needed to approximate the terms. The difference between the maximum and minimum error did not decrease for PPA's with $n < 10$. Therefore the boolean terms in column q_8 were represented by $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$ of their weight. For larger PPA's more columns are required to obtain the desired accuracy.

3.2.3 Compensation Terms

After applying the logically equivalent manipulations and the term approximation technique to excess terms in column q_8, the maximum error difference is 75 units for the PPA shown in figure 11. This would require a table with 7-bit words since $2^7 = 128 > 75 > 2^6 = 64$. We can now apply individual compensation terms described in [Sch93]. By looking at the error as a function of the input we identified three compensation terms to add to column q_8. After adding the three compensation terms the difference between the maximum and minimum errors has been reduced to 59 units. Since the maximum difference is less than 64 a compensation table with 6 bits can be

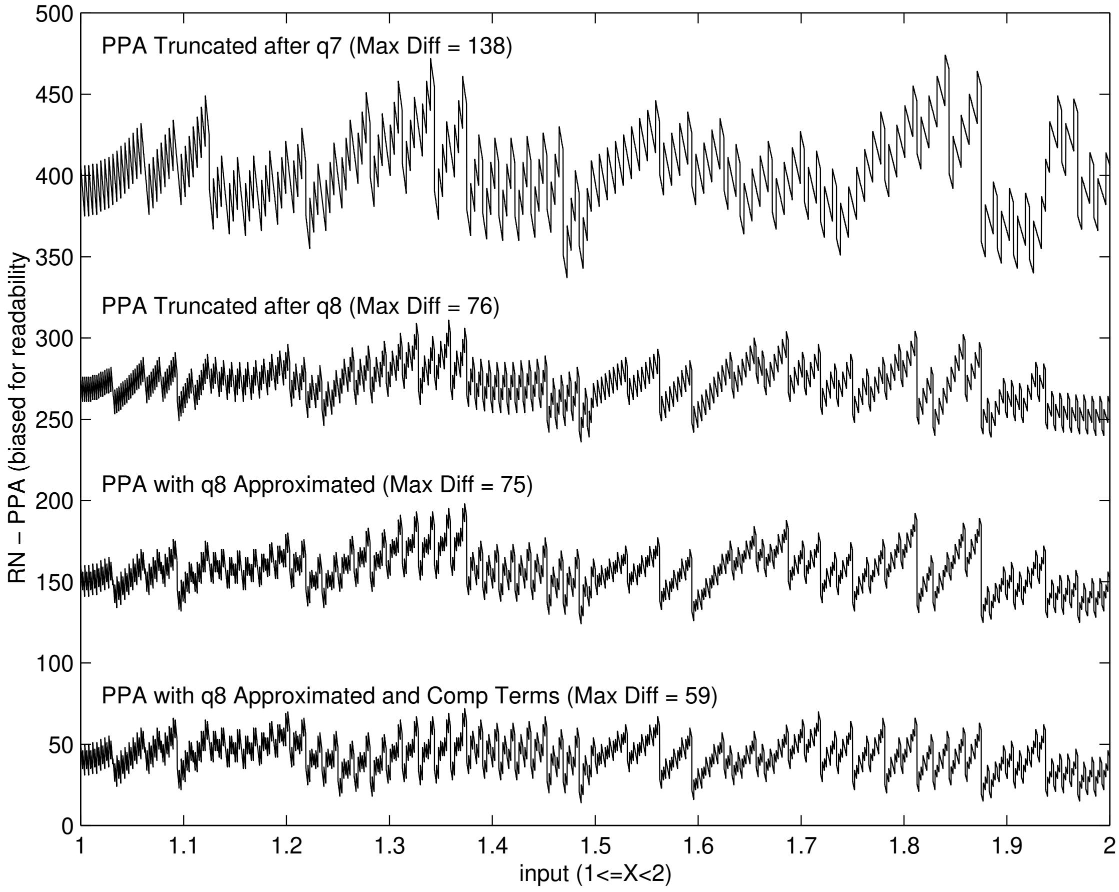

Figure 12: The difference between the computed and the round to nearest result. The compensation table must account for the difference between the maximum and minimum points on each curve.

used. Figure 12 shows the relative difference between the round to nearest result and the computed result. The top curve shows the result difference between the computed result and the round to nearest result when the PPA is truncated at q_7 with a height of 9. The second curve shows the difference when the PPA is truncated at q_8 including all 18 terms in column q_8. The third curve shows the difference when ppa column q_8 is restricted to 9 terms and the remaining 9 terms are approximated by columns q_9 and q_{10} as shown in figure 11. The bottom curve shows the difference after the compensation terms have been added. The curves have been biased so they can be viewed on one graph. The bias is removed by adjusting the constant terms in the PPA. The compensation table must account for difference from between the maximum point and the minimum point on each curve.

3.2.4 Boolean Element Elimination

After adding the three compensation terms to column q_8 the PPA height exceed the desired height of 9. In addition, we need to store positive constants in the compensation table so that the PPA computation and compensation table lookup can be added using one (3,2) counter stage followed by a CPA. A constant value must be subtracted from the constant terms in the PPA to properly bias the error. The result calculated by the PPA must always be equal to or less than the round to nearest result. Furthermore, the computed result should not be less than $2^n - 1$, where n is the number of bits in the compensation table.

Figure 13 shows that the difference between the computed result and the round to nearest result is normally distributed. Very few of the input values would require a compensation table entry near the maximum value (tails of the distribution). Furthermore, several of the boolean terms in the PPA do not affect the tails of the distribution and will not affect the difference between the maximum and minimum error. These boolean terms will affect the average error but since the compensation table eliminates the average error in the computation eliminating the terms will not affect the accuracy of the final result.

The boolean terms with the most literal variables are the terms that are less likely to affect the worst case difference. For the 'AND' terms, each literal variable in the expression reduces the percent of results that will be affected by one half. For example an 'AND' term with 1,2,3, or 4 literal variables will affect 1/2, 1/4, 1/8, and 1/16 of the possible results respectively. For the 'OR' terms each literal variable in the boolean term reduces the number of results not affected by one half. For example an 'OR' term with 1,2,3, or 4 literal variables will affect 1/2, 3/4, 7/8, and 15/16 of the possible results. 'OR' terms that affect most of the possible result can be approximated by a constant. These constants will be absorbed into the other constants in the PPA and do not increase the height of the PPA.

Additionally, we noted that boolean terms with only one literal affect half of the results which also may not affect the worst case difference. Either the term does not affect one of the tails of the distribution or it may affect both of the tails by the same amount which does not increase the difference in the maximum and minimum error. Eliminating the boolean terms with more literal variables reduces the complexity of the input functions to the PPA.

Two terms were identified in column q_8 that did not affect the worst case difference and three terms in column q_9. After these terms were removed and the constant array readjusted the final PPA shown in figure 14 was obtained. The difference between the round to nearest result and the computed results is less that 64 for all input values. Therefore, a compensation table with a 6-bit word size is required for the PPA in figure 14. The PPA shown in figure 14 effectively calculates 6-bits of the result while the table contributes the other 6-bits of the 12-bit result.

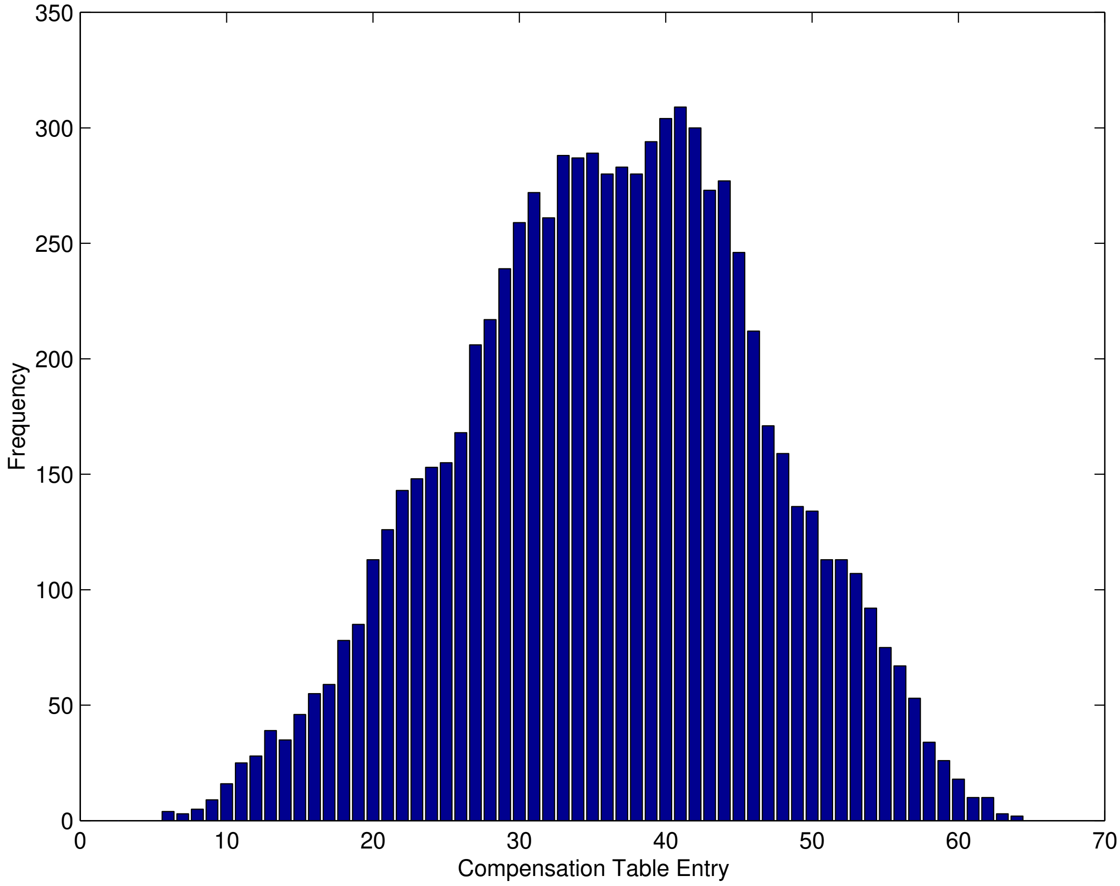

Figure 13: Distribution of the difference between the computed result and the round to nearest result for all 2^13 inputs.

							d3d4’d6	d1d2’d3d4’	d3d4’d7	d3d4’d7
						d3d4’d5	(d2’+d4’)	d1d2d3’d4d5’	d2’d5d7	d2’d5d7
					d4	d2’d3d7	d3’d4’d5	d1d2d3d4’d6	d2’d3d8	d2d3d5d7
					d2d3d4	d2’d3d4	(d2’+d3+d4’+d6’)	d4’	d2’d4d8	d2’d3d8
				(d3’+d5)	d2’d3d5’	(d2’+d4’+d5’)	d5d6	d2d3d5	d2d9	d2’d4d8
			d3	d2’d3d4	d4d5	d2’d3d6	d2’d3d7’	d3’d4’d6	d10’	d2d3d4d8
			(d2’+d4)	d2d3’d5	(d2’+d6)	d4d6	d4d7	d2d5d6’	1	d2d3’d4d5’d7’d8
	d2’	d2d3	d5’	d6’	d7’	d2d7	(d2’+d8)	(d3’+d5’+d6’)	1	d2d9
	d3’	d4’	1	1	1	d8’	d9’	(d2’+d3’+d4’+d5+d6’)	1	1
q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}

1

Figure 14: Final PPA configuration for the reciprocal approximation.

3.3 Applying the precision enhancement techniques to the three design points

In section 3.1 three PPA structures were identified that have the best precision per delay. These were PPA's of height 9, 18, and 36. In this section we will apply the precision enhancement techniques discussed in section 3.2 to the three design points that optimize result precision per delay. First the hardware structure, area and latency are analyzed for each design. Then the area versus time tradeoff is studied for the three design points.

3.3.1 Reciprocal computation with a PPA of height 9

The hardware structure to implement a 13-bit input round to nearest result with 12-bits of precision is shown in figure 15 The hardware components include the PPA shown in figure 14, a 2^13x6 bit compensation table, a (3,2) compressor, and a final 10-bit carry propagate adder (CPA). The compensation table lookup and PPA computation can be overlapped in time. Using standard building blocks the proposed technique takes approximately 10 more gate delays than a single table lookup but reduces the table size by 50at the cost of 408 additional logic gates. Table 3.3.1 lists the area and delay required for each of the components of the computation. The delay for the approximation with compensation can be further reduced by slightly increasing the area using sophisticated design techniques. For example the six most significant bits from the PPA computation can be added using a compound adder to produce $A + B$ and $A + B + 1$. Then the result of the (3,2) compressor bits q_7 through q_{10} can select the proper result from the compound adder while the lower order four bits of the result propagate through the (3,2) compressors and CPA.

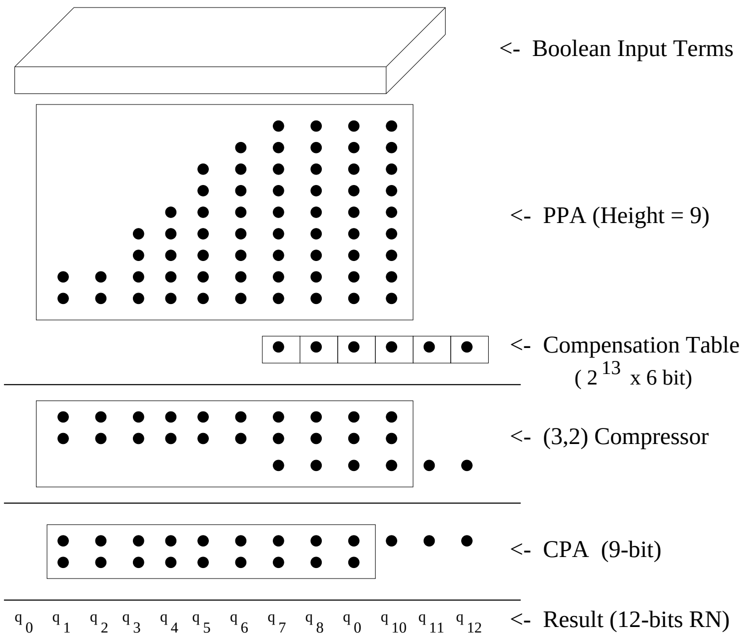

Figure 15: Final hardware configuration for reciprocal unit with PPA of height 9.

Table 2: Reciprocal Unit Area and Latency for PPA with the maximum height of 9.

Technique	Functional Block	Area (gates)	ROM (bits)	Gate Delays
Approx	Input Logic	56		1
	PPA's	245		8
	Compensation Table		49,152	9 (overlapped)
	3:2 Compressor (10 bits)	50		2
	CPA (9 bits)	62		8
	Total	413	49,152	19
Single Tbl	Total		98,304	9

3.3.2 Reciprocal computation with a PPA of height 18

The design technique to maximize the precision of a fixed height PPA structure discussed in section 3.2 was applied to the reciprocal function for a PPA with maximum height of 18. The worst case difference between the round to nearest result and the computed result is 28 units. Therefore, a table with 5 bits that has a range of $2^5 = 32$ units is adequate to use for the compensation table.

Figure 16 shows the structure of the reciprocal unit needed to compute the reciprocal function. As compared to the PPA of height 9 the PPA hardware requirement increased by 1.9 times while the compensation table size decreased by 17performance decreased by four gate delays from that of the PPA with height 9.

Table 3: Reciprocal unit area and Latency for PPA with the maximum height of 18.

Technique	Functional Block	Area (gates)	ROM (bits)	Gate Delays
Approx	Input Logic	207		1
	PPA's	435		12
	Compensation Table		40,960	9 (overlapped)
	3:2 Compressor (10 bits)	55		2
	CPA (9 bits)	80		8
	Total	777	40,960	23
Single Tbl	Total		98,304	9

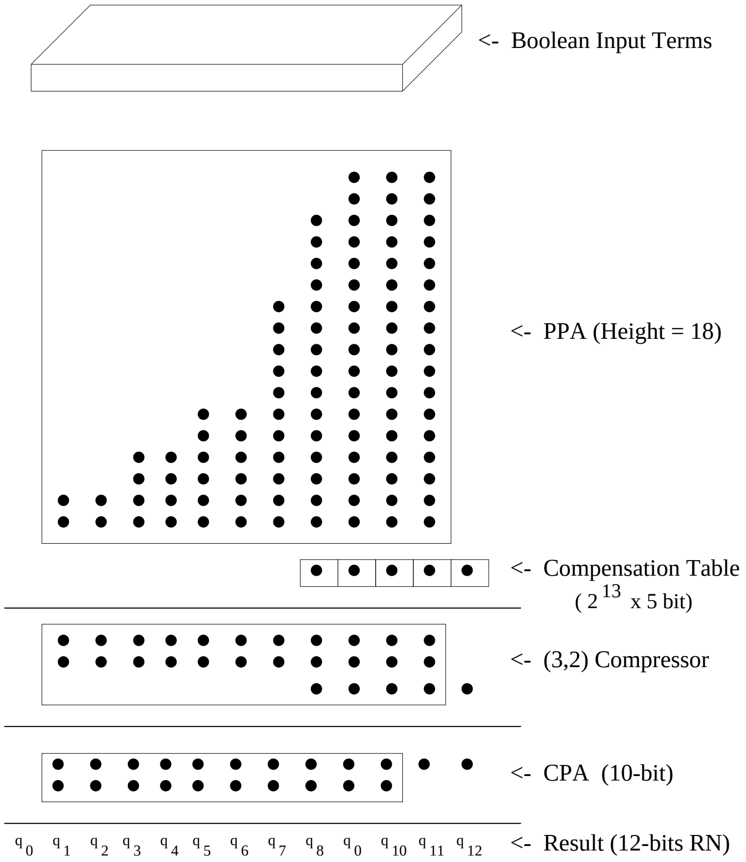

Figure 16: Final hardware configuration for reciprocal unit with PPA of height 18.

3.3.3 Reciprocal computation with a PPA of height 36

The design technique to maximize the precision of a fixed height PPA structure discussed in section 3.2 was also applied to the reciprocal function for a PPA with maximum height of 36. The worst case difference between the round to nearest result and the computed result for this design is 8 units. Therefore, a table with 3 bits with a range of $2^3 = 8$ units is adequate to use for the compensation table.

Figure 17 shows the structure of the unit needed to compute the reciprocal function. As compared to the PPA of height 9 the PPA hardware requirement increased by 3.9 times while the compensation table size decreased by 50latency increased by nine gate delays as compared to the unit with a PPA of height 9.

Table 4: Reciprocal Unit Area and Latency for PPA with the maximum height of 36.

Technique	Functional Block	Area (gates)	ROM (bits)	Gate Delays
Approx	Input Logic	250		2
	PPA's	1190		16
	Compensation Table		24,576	9 (overlapped)
	3:2 Compressor (10 bits)	65		2
	CPA (9 bits)	102		8
	Total	1607	24,576	28
Single Tbl	Total		98,304	9

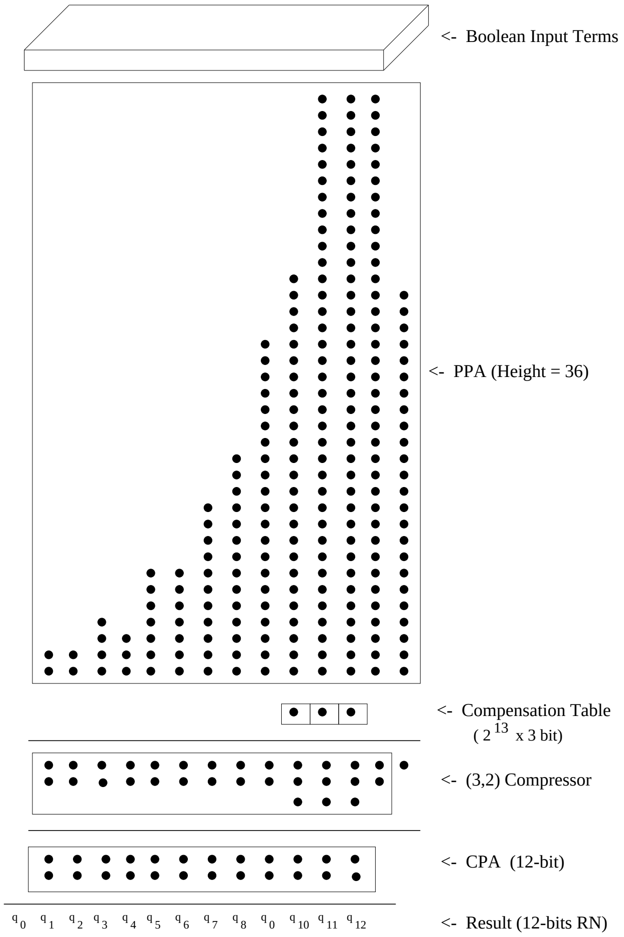

Figure 17: Final hardware configuration for reciprocal unit with PPA of height 36.

3.3.4 Area Time trade off

To compare the area required to implement the three design points and compare these results to that of a single lookup table the relationship between the area of a standard gate and ROM memory must be considered. We assume that 2 bits of memory is approximately equivalent to one logic gate. One bit of ROM memory requires a transistor for the memory element, a word line transistor, a bit line transistor, plus a percentage of the decode and output logic.

Figure 18 shows that the PPA computation can reduce the required area by 50-75gate delays. In addition, the gates used to implement the computation portion of the result are reconfigurable so that they may be used to calculate other higher order functions such as square root, division, e^x, and various trigonometric functions.

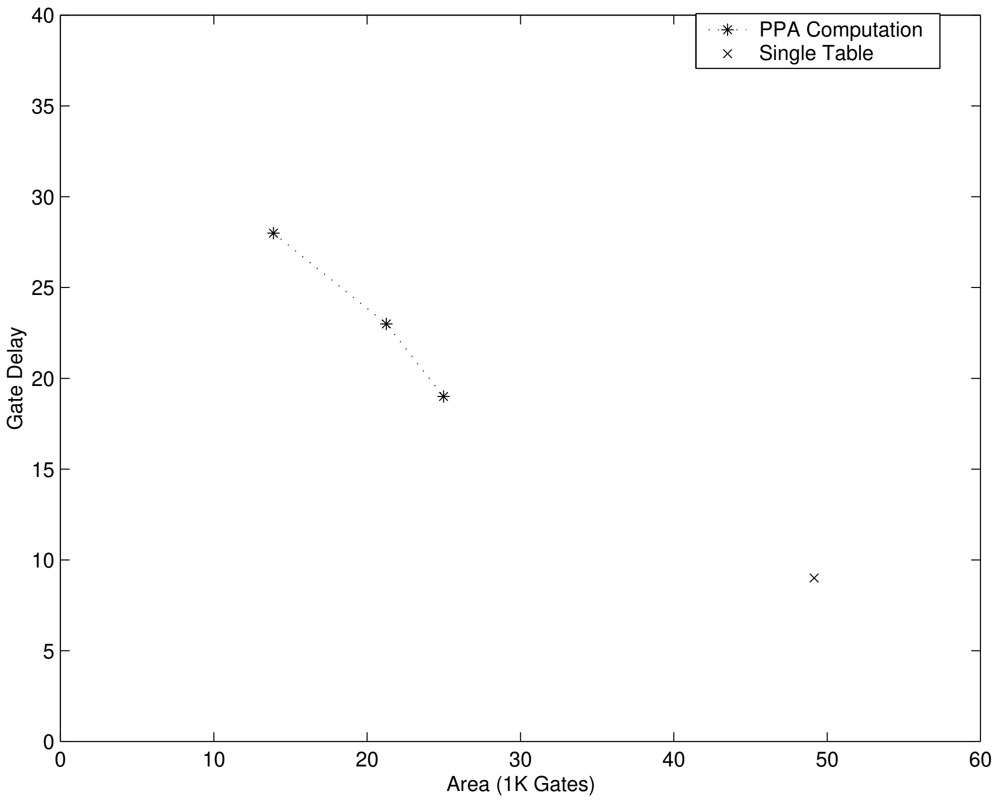

Figure 18: The Area-Time tradeoff for a reciprocal Unit with 12-bit output precision. An increase in computation increases the reciprocal latency but decreases the compensation table size and therefore decreases the area needed implement the unit.

4 PPA Computation Technique applied to the Bipartite Reciprocal table

The approximation theory technique developed in the previous sections may be appropriately applied to faithful bipartite ROM reciprocal tables [DM95]. The bipartite ROM reciprocal hardware structure in figure 19 indicates that two ROM tables are required to produce a result in borrow save format.

Assume that the input is $ibits$ and the reciprocal result is $i-2=jbits$. The larger table stores the $j+2$ bit reciprocal for approximately $2^{\frac{2}{3}i}$ inputs. The 'N' table stores an adjustment amount of $\frac{1}{3}j$ bits for approximately $2^{\frac{2}{3}i}$ of the inputs. The 'P' table is about three times the size of the 'N' table. Furthermore, the 'P' table stores the reciprocal of the input the same value that we compute using the approximation technique with table compensation. Therefore, we propose replacing the 'P' table with a PPA and compensation table to reduce the hardware needed in the bipartite ROM tables. Here the PPA computation can be overlapped with the compensation table and the 'N' table lookups. The bipartite ROM reciprocal table technique requires a carry propagate add if a non-redundant answer is required. We expect that the proposed technique can reduce the area needed for the 'P' table by $\frac{1}{3}$ to $\frac{1}{2}$. The PPA, compensation table, and 'N' table results could then be combined in a single (4,2) compressor stage followed by a carry propagate add. The latency to obtain the final non-redundant result is increased by 2 to 4 gate delays over the standard bipartite ROM reciprocal table technique.

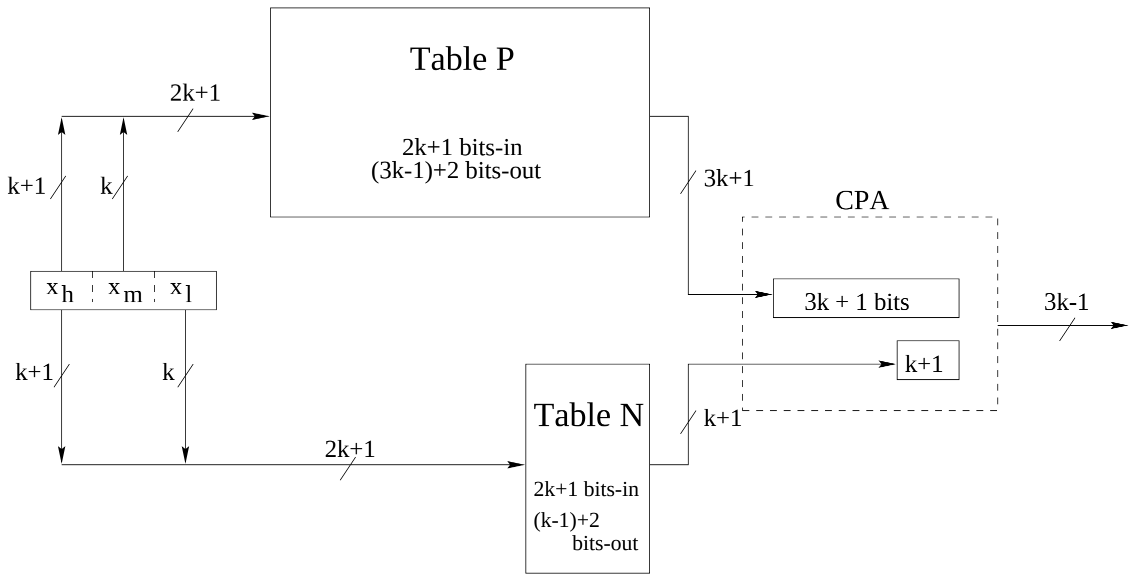

Figure 19: A $j+2=3k+1$ bits-in $j=3k-1$ bits-out Faithful Reciprocal table. [DM95]

5 Conclusions

In this paper we developed a technique to produce a fixed precision result for multiplicative functions. This technique uses a partial product array to compute an approximation of the function and then applies a compensation table to produce a fixed precision result.

In addition, we proposed several techniques to optimize the worst case precision that is computed in a fixed height PPA. These techniques include logical manipulations, boolean term approximations, and boolean term elimination.

To understand the area-time tradeoff between the PPA computation and table lookup, we derived three optimal design points and applied the approximation technique to these design points. The three design points included computations with PPA heights of 9, 18, and 36 and compensation tables with entries of 6, 5, and 3 bits respectively.

The proposed technique reduces the hardware area required by 50% to 75% over a single lookup table. The hardware area required to compute the approximation is much less than that of a lookup table. However the latency increases by about 2 to 3 times that of a single table access. The increase in latency is primarily due to the final carry propagate add required to combine the computed approximation with the compensation table result.

In the previous section, we discussed the application of the proposed technique to bipartite ROM reciprocal tables. The bipartite ROM reciprocal table technique requires a table lookup of the reciprocal function and a carry propagate add with an adjustment factor found by another table lookup. The PPA computation of the reciprocal function and compensation table lookup can done in parallel with the table lookup of the adjustment factor without an increase in latency. To produce a non-redundant result the bipartite table output requires a carry propagate add. The final result can be determine by one (4,2) compressor stage that combines the PPA, compensation table, and the bipartite 'N' table followed by one CPA. The proposed technique would require only a few additional gate delays as compared to the bipartite ROM reciprocal table technique.

Finally, the PPA hardware can be reconfigured to compute an approximation for other higher order functions that can be expressed in the form of a multiply. Therefore, from a reconfigurability perspective the proposed technique increases the hardware utility.

References

[DM95] D. DasSarama and D. W. Matula. Faithful Bipartite ROM Reciprocal Tables. In *IEEE Symposium on Computer Arithmetic '95*, pages 17–28, July 1995.

[Man90] David Mandelbaum. A systematic method for division with high average bit skipping. In *IEEE Transactions on Computers*, pages 127–130, January 1990.

[Sch93] Eric Mark Schwarz. *High-Radix Algorithms for High-Order Arithmetic Operations.* PhD thesis, Stanford University, 1993.

[Ste72] Renato Stefanelli. A suggestion for a high-speed parallel binary divider. In *IEEE Transactions on Computers*, pages 42–55, January 1972.