

A Bit-Width Optimization Methodology for Polynomial-Based Function Evaluation

Dong-U Lee, *Member, IEEE*, and
John D. Villasenor, *Senior Member, IEEE*

Abstract—We present an automated bit-width optimization methodology for polynomial-based hardware function evaluation. Due to the analytical nature of the approach, overflow protection and precision accurate to one unit in the last place (ulp) can be guaranteed. A range analysis technique based on computing the root of the derivative of a signal is utilized to determine the minimal number of integer bits. Fractional bit requirements are established using an analytical error expression derived from the functions that occur along the data path. Global fractional bit optimization across multiple computation stages is performed using simulated annealing and circuit area estimation functions.

Index Terms—Computer arithmetic, elementary function approximation, field programmable gate arrays, finite wordlength effects, minimax approximation and algorithms.

1 INTRODUCTION

ONE of the main challenges when designing hardware-based function evaluation units is the determination of the correct number of bits for the signals in the fixed-point data path. Excessive bit-width allocation will result in wasting valuable hardware resources, whereas insufficient bit-width allocation will introduce overflows and violate precision requirements. Thus, it is desirable to find the minimal bit-widths to all signals that overcome the two shortcomings.

Previous work in the area of bit-width optimization can be divided into two classes: dynamic analysis and static analysis. Dynamic analysis relies on the use of input stimuli signals. This approach can be problematic since a large set of stimuli signals is often required to analyze a design with sufficient confidence, leading to prohibitively long simulation times. Although static analysis can produce conservative (i.e., larger) bit-width estimates than dynamic analysis, it is often the more attractive solution, especially for large designs, since only the characteristics of the input signals are needed.

The most commonly used approach for bit-width optimization for polynomial structures is a dynamic method in which a group of arithmetic operators are constrained to have the same uniform bit-width (i.e., the same number of guard bits). The design is exhaustively simulated over all possible input values and the error at the output is monitored (e.g., [1], [2], [3]). The uniform bit-width is gradually adjusted to a point where all possible inputs meet the precision requirement.

Some recent contributions, however, are based on static methods. For instance, Detrey and de Dinechin [4] produce an analytical error expression at the output signal and derive the number of guard bits required. The same number of guard bits are used for the coefficients and arithmetic operators. Similar paths are taken in [5] and [6]. In general, these approaches have been application-specific, making them difficult to apply in more generalized settings. Moreover, the guard bits of the signals have

- The authors are with the Electrical Engineering Department, University of California, Los Angeles, 420 Westwood Blvd., Los Angeles, CA 90095. E-mail: {dongu, villa}@icsl.ucla.edu.

Manuscript received 2 Nov. 2005; revised 7 Apr. 2006; accepted 26 July 2006; published online 1 Feb. 2007.

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0388-1105. Digital Object Identifier no. 10.1109/tc.2007.1031.

been constrained to be uniform. By contrast, the proposed approach is generic and flexible, making it suitable for virtually any arithmetic structure involving additions, multiplications, and memory accesses. In addition, the guard bits are allowed to be nonuniform and are minimized via a user-defined cost function.

The methods presented here target function evaluation using polynomials, which offer a rich range of design trade-offs involving memory, computational approaches, and precision. Precision is quantified using the unit in the last place (ulp). We target “faithful” rounding, results are rounded to either the nearest or next nearest fraction expressible using the available bits and are thus accurate to within 1 ulp.

The principal contributions of this work include:

- a range analysis algorithm based on computing the roots of the derivatives of the signals for computing the required integer bits,
- a precision analysis algorithm via simulated annealing using analytical error models and cost functions, and
- design space exploration on a Xilinx Virtex-4 FPGA.

2 OVERVIEW

Given a set of inputs after range reduction [7], the approximation can be performed using a single polynomial or using piecewise polynomials of user-specified degree. Single polynomial approximation involves approximating the entire range-reduced interval using a single high-degree polynomial. With piecewise polynomials, however, the interval is split into several segments and a (typically) lower degree polynomial is used within each segment. The polynomial coefficients are generated in the minimax sense via the Remez algorithm [7]. The evaluation of polynomials can be performed with a variety of scheduling strategies, where each strategy has its advantages and disadvantages. In this paper, we consider the Horner case [7], but the framework itself can be applied to any other evaluation method.

When evaluating piecewise polynomials in hardware, the input argument x is typically split into two parts: x_0 and x_1 , i.e., $B_x = B_{x_0} + B_{x_1}$, where B_x denotes the bit-width of the signal x . x_0 is the set of the leftmost bits, while x_1 are the remaining bits of x . The approximation interval is split into $2^{B_{x_0}}$ equally sized contiguous segments, x_0 serves as the index into the table holding the polynomial coefficients. Note that, for single polynomial approximation, B_{x_0} would be zero (i.e., a single segment). Since x_0 is implicitly known for a given segment, we use x_1 instead of x for the polynomial arithmetic to reduce the size of the operators. We scale x_1 to occupy the range $[0, 1)$. This requires appropriate transformations on the coefficients since they have been generated with the assumption that x will be used for the polynomial arithmetic (see [8]).

With either single polynomial or piecewise polynomials, error is introduced in two basic ways. First, there is an inherent error, ϵ_∞ , due to the imperfect match between the polynomials and the target function, i.e., the approximation error would be ϵ_∞ if the polynomials were evaluated using infinite precision. Second, there is the quantization error, ϵ_Q , introduced by finite precision effects encountered in the polynomial evaluation. Since the final output rounding can cause a maximum error of $1/2$ ulp, to ensure faithful rounding, the following condition must be met:

$$\epsilon_\infty + \epsilon_Q \leq 1/2 \text{ ulp.} \quad (1)$$

Placing extremely stringent requirements on ϵ_∞ will increase the polynomial degree and/or number of segments, but will allow greater amounts of finite-precision induced error and, thus, lead to lower bit-widths. Increasing ϵ_∞ leads to increased bit-widths.

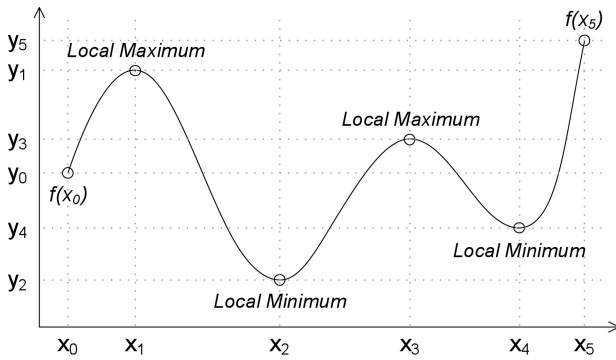


Fig. 1. Local minima and maxima of a signal. The range of this signal is $[y_2, y_5]$.

Once the coefficients have been generated, ϵ_∞ will be known. The next task is bit-width optimization, where the quantization error, ϵ_Q , due to the tables and arithmetic operations needs to be considered. The goal is to ensure bit-widths sufficient to avoid overflow and satisfy precision requirements ((1) in our case), while also avoiding unnecessarily large widths that would waste hardware resources. Each fixed-point signal z will have an integer bit-width (IB) and a fractional bit-width (FB), i.e., $B_z = IB_z + FB_z$. IB governs the range, while FB governs the precision. We separate the bit-width allocation problem into two parts: range analysis for determining the IB s and precision analysis for determining the FB s. Range analysis is performed first, followed by precision analysis. After the precision analysis, slight changes can occur to the signal ranges due to quantization effects. Therefore, corrections are performed on the IB s if needed.

3 RANGE ANALYSIS

We consider the evaluation of the function $\ln(1+x)$ over $[0, 1)$ to a target precision of 12 fractional bits (i.e., accurate to 2^{-12}). Regarding the error allocation between ϵ_∞ and ϵ_Q , we allow a maximum of 0.49 ulp for ϵ_∞ as this results in a relatively low polynomial degree and number of segments, thus facilitating explanation.

The objective of range analysis is to compute the ranges of the signals in the data path based on the ranges (and, in particular, the maximum absolute values) of the input and the coefficients. In two's complement fixed-point, the IB of a signal z is given by

$$IB_z = \lceil \log_2(Z) \rceil + 1, \quad (2)$$

where $Z = \max(|z_{min}|, |z_{max}|)$. In rare cases when Z is a power of two, IB_z needs to be incremented by an additional bit.

Range analysis has been widely studied in the literature; in particular, interval arithmetic [9] and affine arithmetic [10] are among the most commonly used methods. However, both methods suffer from overestimation problems, which grow with the depth of the data path. We propose a different range analysis technique which involves examining the local minima, local maxima, and the minimum and maximum input values at each signal. This approach is able to identify the exact range for all signals, provided that the functions being analyzed are differentiable and continuous on a closed interval, as is, of course, the case with polynomials.

Consider the function shown in Fig. 1, where the input interval is over $[x_0, x_5]$. The local minima and maxima can be found by computing the roots of the derivative. We observe that the local minima, local maxima, $f(x_0) = y_0$, and $f(x_5) = y_5$ are the potential candidates for the minimum and maximum values of the signal. The minimum value is given by $\min(y_0, y_1, y_2, y_3, y_4, y_5)$, while the

TABLE 1
Minimax Polynomial Coefficients for Approximating $\ln(1+x)$
Accurate to 12 Fractional Bits Using a Degree-4 Polynomial

Coefficient	C_4	C_3	C_2	C_1	C_0
Value	-0.05657	0.44718	-1.46995	2.82118	-1.74178
IB	-3	0	2	3	2

maximum value is given by $\max(y_0, y_1, y_2, y_3, y_4, y_5)$. For this particular example, the output range would be $[y_2, y_5]$. This technique has modest complexity since the number of potential extrema is limited by the degree of the polynomial.

3.1 Single Polynomial

We first consider approximation using a single polynomial. The approximation of $\ln(1+x)$, accurate to 12 fractional bits, requires a polynomial degree of four. The minimax coefficients and the required IB s are shown in Table 1 (rounded to five fractional digits for readability). The approximation error, ϵ_∞ , is found to be 0.24847 ulp (where the ulp is 2^{-12}). Note that IB can be negative, as in the second column of Table 1. In this example, $IB = -3$ means that the first three fractional bits of C_4 will always be zero. This fact can be exploited in hardware implementation.

The degree-4 polynomial can be evaluated with Horner's rule using an architecture analogous to the degree-2 case shown in Fig. 2, but with appropriate additional multiply-and-add units and their signal index notations. The quantization steps contained in the white squares in the figure are performed at compile time (since quantized coefficients are stored), while those in the gray squares are performed at runtime. Applying the range analysis technique described above gives the signal ranges and the IB s shown in Table 2.

3.2 Piecewise Polynomials

For the valuation of $\ln(1+x)$ accurate to 12 bits using degree-2 piecewise polynomials, four segments are needed. The minimax coefficients for each segment are listed in Table 3. Since different polynomials are used for each segment, each segment has its own ϵ_∞ . Because each coefficient is stored in a single column in the ROM, the IB s must be of the same size for a given coefficient. For instance, Table 3 indicates that C_2 varies over $[-0.02488, -0.00891]$ over the four segments, thus $IB_{C_2} = \lceil \log_2(0.02488) \rceil + 1 = -4$. Likewise, $IB_{C_1} = -1$ and $IB_{C_0} = 1$.

To compute the IB s of the internal signals and the output, range analysis is performed for each of the four segments and the maximum is identified. The range analysis results are shown in

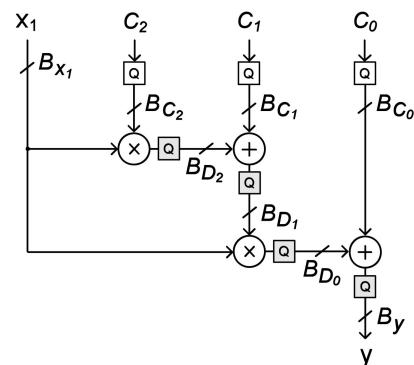


Fig. 2. Circuit for evaluating degree-2 polynomials with Horner's rule: $y = ((C_2 \times x_1) + C_1) \times x_1 + C_0$. The white "Q" squares perform quantization at compile-time, while the gray squares perform quantization at runtime.

TABLE 2
Signal Ranges and Integer Bit-Widths (IB_s) for the Single Degree-4 Polynomial Example

Signal	Range	IB
D_6	[-0.11309,-0.05657]	-2
D_5	[0.334010,0.39061]	0
D_4	[0.39061,0.66786]	1
D_3	[-1.07934,-0.80210]	2
D_2	[-1.60339,-1.07934]	2
D_1	[1.21779,1.74184]	2
D_0	[1.74184,2.43437]	3
y	[0.00006,0.69259]	1

Table 4. For example, at the output signal y , the required integer bit-width is given by $IB_y = \max(-1, 0, 1, 1) = 1$.

4 PRECISION ANALYSIS

Since ϵ_∞ is obtained during coefficient table generation, the design challenge is to determine the minimal number of fractional bits (FBs) to utilize for each signal while meeting the accuracy requirement at the output (1). The precision analysis method introduced here is based on examining the worst-case error bounds at each signal. It is an adaptation of the MiniBit technique [11], customized for polynomial evaluations, and consists of the following two main steps:

1. For the output signal, generate an analytical error expression that is a function of the FBs of the internal signals.
2. Perform simulated annealing in conjunction with a circuit area cost function to find the minimal FBs .

4.1 Single Polynomial

There are two main ways to quantize a signal to a given FB . Truncation gives a maximum error of 2^{-FB} (1 ulp), while round-to-nearest gives a maximum error of 2^{-FB-1} (1/2 ulp). Truncation requires no extra hardware resources, while round-to-nearest needs an additional adder. In order to guarantee faithful rounding, round-to-nearest needs to be performed at the output signal at all times. However, for the internal signals, the circuit designer is free to choose either quantization method. For the analysis in this section, we assume that round-to-nearest is used at all times.

Precision analysis is illustrated using a degree-1 polynomial $y = C_1 \times x + C_0$. We denote the error at a signal z as ϵ_z . Computation occurs in two steps, with quantization at each step:

$$D_0 = C_1 \times x, \quad (3)$$

$$y = D_0 + C_0. \quad (4)$$

TABLE 3
Minimax Polynomial Coefficients and ϵ_∞ for Approximating $\ln(1+x)$ Accurate to 12 Bits Using Degree-2 Piecewise Polynomials

Segment	C_2	C_1	C_0	ϵ_∞
0	-0.02488	0.24780	0.00012	0.47218 ulp
1	-0.01661	0.19881	0.22321	0.25765 ulp
2	-0.01188	0.16595	0.40550	0.15576 ulp
3	-0.00891	0.14240	0.55964	0.10126 ulp

The ulp is 2^{-12} .

The error ϵ_{D_0} at the signal D_0 is given by

$$\epsilon_{D_0} = C_1 \epsilon_x + x \epsilon_{C_1} + \epsilon_{C_1} \epsilon_x + 2^{-FB_{D_0}-1}. \quad (5)$$

Note that the rounding error $2^{-FB_{D_0}-1}$ is a conservative estimation. A tighter rounding error can be obtained by comparing the full precision of the operation against FB_{D_0} . For clarity and readability, we opt for this conservative estimation in this paper.

We assume the input x has no errors, i.e., $\epsilon_x = 0$; (5) becomes

$$\epsilon_{D_0} = x \epsilon_{C_1} + 2^{-FB_{D_0}-1}. \quad (6)$$

The error ϵ_y at the output y is given by

$$\epsilon_y = \epsilon_{D_0} + \epsilon_{C_0} + 2^{-FB_y-1} + \epsilon_\infty, \quad (7)$$

where $\epsilon_{D_0} + \epsilon_{C_0}$ is the quantization error ϵ_Q and 2^{-FB_y-1} is the 1/2 ulp final rounding error. For faithful rounding, the maximum output error, $\max(\epsilon_y)$, needs to be less than or equal to 1 ulp, i.e.,

$$2^{-FB_y} \geq \max(\epsilon_y). \quad (8)$$

Suppose the goal is to ensure that y is accurate to 16 bits, the inherent approximation error ϵ_∞ in this case is 2^{-19} (which is equivalent to $2^{-19} \times 2^{16} = 0.125$ ulp) and $\max(|x|) = 2$. As neither C_1 nor C_2 appear in (6) and (7), $\max(|C_1|)$ and $\max(|C_0|)$ are irrelevant to this particular error analysis. Using (6), (7), (8) gives

$$\begin{aligned} 2^{-16} &\geq 2 \times 2^{-FB_{C_1}-1} + 2^{-FB_{D_0}-1} \\ &\quad + 2^{-FB_{C_0}-1} + 2^{-17} + 2^{-19} \\ &\Rightarrow \frac{3}{262,144} \geq 2^{1-FB_{C_1}} + 2^{-FB_{D_0}} + 2^{-FB_{C_0}}. \end{aligned} \quad (9)$$

Since error terms are functions of the FBs and the maximum values of the signals, the optimization problem is to find the minimal FBs for the signals C_1 , C_0 , and D_0 while satisfying the inequality in (9). It is clear that error expressions of this general form can be constructed for any arithmetic circuit involving additions and multiplications (e.g., the single degree-4 polynomial example in Section 3). A suboptimal but simple solution is to use uniform fractional bit-widths (UFB), i.e., the same number of FBs for all signals. For example, in (9), this gives

TABLE 4
Signal Ranges and Integer Bit-Widths (IB_s) for the Degree-2 Piecewise Polynomial Example

Signal	Segment 0		Segment 1		Segment 2		Segment 3		Final IB
	Range	IB	Range	IB	Range	IB	Range	IB	
D_2	[-0.02486,0.00000]	-4	[-0.01660,0.00000]	-4	[-0.01187,0.00000]	-5	[-0.00891,0.00000]	-5	-4
D_1	[0.22294,0.24780]	-1	[0.18221,0.19881]	-1	[0.15409,0.16595]	-1	[0.13349,0.14240]	-1	-1
D_0	[0.00000,0.22271]	-1	[0.00000,0.18203]	-1	[0.00000,0.15393]	-1	[0.00000,0.13336]	-1	-1
y	[0.00012,0.22283]	-1	[0.22321,0.40524]	0	[0.40550,0.55944]	1	[0.55964,0.69300]	1	1

TABLE 5
Signal Bit-Widths for the Degree-2 Piecewise Polynomial Example

Signal	C_2	C_1	C_0	D_2	D_1	D_0
B	14	18	21	16	17	18
(IB,FB)	(-4,18)	(-1,19)	(1,20)	(-4,20)	(-1,18)	(-1,19)

The bit-widths in the brackets indicate the integer bit-widths (IBs) and the fractional bit-widths (FBs). Multiplier operands are shown in bold.

$$\frac{3}{262,144} \geq 2^{-UFB+1} + 2^{-UFB} + 2^{-UFB} \quad (10)$$

$$\Rightarrow UFB = 19 \text{ bits.}$$

Instead of using $UFBs$, one can do better by using nonuniform fractional bit-widths ($NFBs$) where the FBs of the signals can be different. Since each FB must be an integer, the overall optimization problem is nonconvex. Such problems can be addressed using a number of methods, including adaptive simulated annealing (ASA) [12]. ASA permits adaptation to changing sensitivities in the multidimensional parameter space, thus allowing significantly faster convergence times than traditional simulated annealing. The annealing process can be accelerated by supplying good initial variable estimates (FBs in this case). The UFB serves well for this purpose and is used for initialization.

In ASA, the user is required to supply a constraint function and a cost function. For the constraint function, error functions such as the inequality in (9) are supplied. For the cost function, we supply an area estimation model of the circuit which is a function of the signal bit-widths. The accuracy of the area estimation plays a central role to the optimization efficiency since imprecise estimations can lead to suboptimal ASA results. Since, Xilinx FPGAs are being targeted in this work, we have thoroughly investigated the way adders, multipliers, and tables are implemented on these devices. The cost functions we use precisely model the actual area usage on FPGAs, as will be shown in Section 5. Applying ASA to the degree-1 example above, we get $FB_{C_1} = 18$, $FB_{C_0} = 19$, and $FB_{D_1} = 19$. Hence, in this example, allowing nonuniform FBs reduces the bit-width of the signal C_0 by 1 bit relative to the UFB approach.

As briefly mentioned in Section 2, slight changes occur to the signal ranges found in Section 3 due to quantization effects. In very rare circumstances, these changes can, in principle, lead to increased IB requirements. This potential problem can be addressed in a rather straightforward manner. For a signal z , its quantization error ϵ_z can be found by constructing its error expression and substituting in the FBs found by ASA. ϵ_z is added to Z in (2) and IB_z is recomputed. This process is repeated for all signals. For the overwhelming majority of the signals, the IB requirements remain unchanged and the above correction step turns out to be unnecessary, but this is not problematic since its computational cost is so low. For those few signals with altered IB requirements, the correction step ensures proper handling of these quantization-induced changes.

4.2 Piecewise Polynomials

Precision analysis that can be performed for piecewise polynomial approximation is described here using the $\ln(1+x)$ degree-2 piecewise polynomial example, though, as with the polynomial methods in the previous section, the generalization to higher degrees is straightforward. Consider the degree-2 circuit in Fig. 2. Knowing that $\max(|x_1|) = 1$ and $\epsilon_{x_1} = 0$, the following general inequality can be constructed for a degree- d piecewise polynomial using Horner's rule with round-to-nearest:

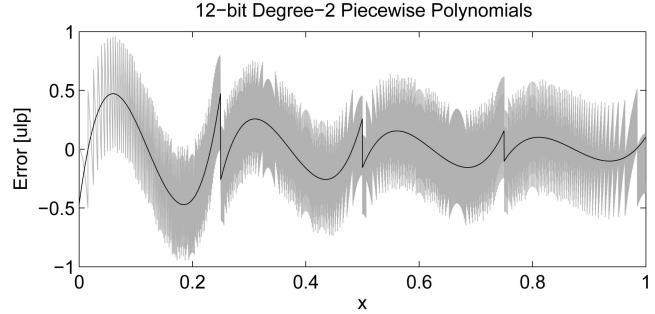


Fig. 3. Error plot in ulp for the degree-2 piecewise polynomial example with the bit-widths from Table 5 incorporated. The black curve indicates the inherent approximation error, while the gray curve indicates the total error with finite precision effects.

$$2^{-FB_y} \geq \sum_{i=0}^d 2^{-FB_{C_i}-1} + \sum_{i=0}^{d \times 2-2} 2^{-FB_{D_i}-1} + 2^{-FB_y-1} + \epsilon_\infty \quad (11)$$

$$\geq \sum_{i=0}^d 2^{-FB_{C_i}} + \sum_{i=0}^{d \times 2-2} 2^{-FB_{D_i}} + 2 \times \epsilon_\infty.$$

From (11), the ranges of the signals do not affect the error at the output (due to the fact that $\epsilon_{x_1} = 0$). Also, the highest FB constraint will be placed on the segment with the largest ϵ_∞ . For the degree-2 example, Table 3 indicates that ϵ_∞ is the largest in segment 0. Assuming again that 12 fractional bits of output accuracy are required, (11) gives

$$2^{-12} \geq \sum_{i=0}^2 2^{-FB_{C_i}} + \sum_{i=0}^2 2^{-FB_{D_i}} + 2 \times 1.15563 \times 10^{-4}. \quad (12)$$

Since segment 0 is the worst case, the FBs determined using (12) ensure that sufficient accuracies for all other segments are assured.

Table 5 gives the IBs (from Table 4) and FBs (found by ASA). The first row of the table gives the total bit-width for each signal. The second row gives the number of IBs and FBs from which the total is derived. The UFB is found to be 19 bits. ASA generally reduced the FBs of the multiplier operands (shown in bold) since multipliers have a significantly higher area penalty compared to other resources such as adders.

Fig. 3 shows the error plot in ulp for all possible input values for the degree-2 piecewise polynomial example with the bit-widths from Table 5 incorporated. The figure shows that all outputs have an absolute error within 1 ulp and, therefore, are indeed faithfully rounded.

5 EXPERIMENTAL RESULTS

The bit-width optimization flow has been fully automated within Matlab. A Xilinx Virtex-4 XC4VLX100-12 FPGA is used for experimental implementation. Designs are written in Verilog, synthesized with Synplicity Synplify Pro 7.7.1 and placed-and-routed with Xilinx ISE 7.1.03i. Designs are fully combinatorial with no pipeline registers. Although dedicated RAMs and multiply-and-add blocks are present on the device, we consider implementations based on slices (programmable logic blocks) only in order to obtain unbiased comparisons. Results for the approximation of $\ln(1+x)$ over $x = [0,1)$ are given. A maximum of 0.3 ulp is allocated for ϵ_∞ since this is found to provide a good balance between ϵ_∞ and ϵ_Q .

Table 6 explores the area and delay impact of using round-to-nearest or truncation for the internal signals. In the degree-2 circuit

TABLE 6

Area and Delay Comparisons between Using Round-to-Nearest and Truncation for the Internal Signals for Degree-2 Piecewise Polynomials

Precision [bits]	Area [slices]			Delay [ns]		
	Rounding	Truncation	Diff	Rounding	Truncation	Diff
8	123	107	16	29.561	21.999	7.562
16	245	221	24	35.555	27.955	7.600
24	628	609	19	44.944	36.721	8.223
32	3011	3006	5	47.935	40.639	7.296

shown in Fig. 2, these would correspond to signals D_2 , D_1 , and D_0 . Round-to-nearest allows smaller quantization errors, potentially reducing the internal signal bit-widths and, hence, the operator sizes. However, it requires an adder after each operation. By contrast, truncation does not require any extra hardware, but has larger quantization errors. This can potentially increase the internal bit-widths. The table indicates that truncation has a slightly smaller area than rounding. In terms of delay, truncation is significantly faster than round-to-nearest due to the absence of the rounding circuitry. Similar trends have been observed for other polynomial structures. Thus, truncation provides superior area and delay results than rounding and is used for the remaining results described in this section.

Fig. 4 is a bar graph showing area comparisons of degree-1, degree-2, and degree-3 piecewise polynomials. The upper part of each bar indicates the area used for arithmetic, while the lower part indicates the area consumed by the coefficient table. As the target precision increases, the proportion of overall area used for the coefficient table increases exponentially, as might be expected. For the degree-1 case, the table starts to occupy more area than the actual arithmetic after 14 bits. For precisions exceeding 15 bits, degree-2 starts to become more cost effective than degree-1. Furthermore, due to the diminishing area gap between degree-2 and degree-3, degree-3 is the most area efficient for precisions beyond 21 bits. Delay comparisons of degree-1, degree-2, and degree-3 piecewise polynomials are also performed. In all cases, the delay is found to increase in a linear manner with precision and, as expected, lower degrees are found to be faster. The delay gap between each degree is consistently found to be around 6 ns. Similar area and delay trends are obtained for other functions such as $\sin(x)$ over $x = [0, \pi/2)$ and \sqrt{x} over $x = [1, 4)$.

Table 7 shows area comparison results for UFB , NFB , and estimated NFB area (NFB') (used for the cost function during ASA optimization process) for degree-2 piecewise polynomials. The table shows that the absolute area savings between UFB and NFB increases with precision, while the relative savings show a

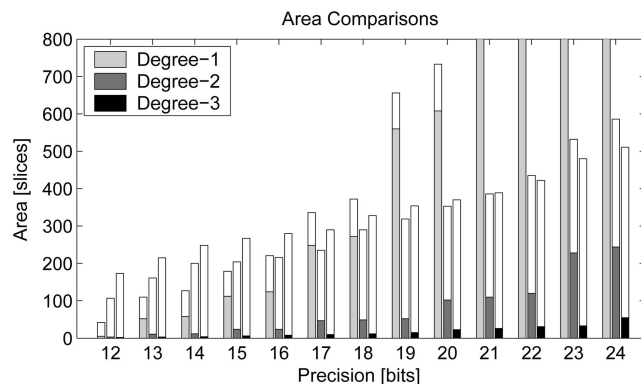


Fig. 4. Area comparisons of degree-1, degree-2, and degree-3 piecewise polynomials. The upper and lower parts of each bar indicate the area used by the arithmetic and coefficient tables, respectively.

TABLE 7

Area Comparisons for Uniform Fractional Bit-Width (UFB), Nonuniform Fractional Bit-Width (NFB), and Estimated NFB Area (NFB') for Degree-2 Piecewise Polynomials

Precision [bits]	Area [slices]				
	UFB	NFB	NFB'	$UFB-NFB$	$ NFB-NFB' $
8	116	107	102	9 (7.8%)	5 (4.7%)
16	243	221	223	22 (9.1%)	2 (0.9%)
24	642	609	586	33 (5.1%)	23 (3.8%)
32	3202	3006	2908	196 (6.1%)	98 (3.3%)

decreasing trend. Although these savings are modest in percentage terms, they are free as the NFB designs have the same error bound as the UFB designs. In addition, comparison between the actual NFB area against the estimated NFB area ($|NFB - NFB'|$), shows that our resource estimation methods are accurate to within 5 percent.

6 CONCLUSIONS

We have presented a bit-width optimization approach for designing hardware-based faithfully rounded function evaluation units via a single polynomial and piecewise polynomials. Due to the analytical nature of the proposed bit-width optimization approach, overflow protection and precision accurate to one ulp can be guaranteed. A range analysis technique based on computing the root of the derivative of a signal has been described to find the minimal number of integer bits. For precision analysis, we have described an approach where analytical error expressions are employed together with simulated annealing to find the required number of fractional bits.

ACKNOWLEDGMENTS

The authors thank Ray C.C. Cheung for his assistance. The support of the US Office of Naval Research (contract number N00014-06-1-0253) and the US National Science Foundation (grant numbers CCR-0120778 and CCF-0541453) is gratefully acknowledged.

REFERENCES

- [1] J.N. Coleman, E. Chester, C.I. Softley, and J. Kadlec, "Arithmetic on the European Logarithmic Microprocessor," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 702-715, July 2000.
- [2] B. Lee and N. Burgess, "Some Approximations on Taylor-Series Function Approximation on FPGA," *Proc. Asilomar Conf. Circuits, Systems, and Computers*, vol. 2, pp. 2198-2202, 2003.
- [3] D.M. Lewis, "Interleaved Memory Function Interpolators with Application to an Accurate LNS Arithmetic Unit," *IEEE Trans. Computers*, vol. 43, no. 8, pp. 974-982, Aug. 1994.
- [4] J. Detrey and F. de Dinechin, "Table-Based Polynomials for Fast Hardware Function Evaluation," *Proc. IEEE Int'l Conf. Application-Specific Systems, Architectures, and Processors*, pp. 328-333, 2005.
- [5] F. de Dinechin and A. Tisserand, "Multipartite Table Methods," *IEEE Trans. Computers*, vol. 54, no. 3, pp. 319-330, Mar. 2005.
- [6] R. Michard, A. Tisserand, and N. Veyrat-Charvillon, "Small FPGA Polynomial Approximations with 3-Bit Coefficients and Low-Precision Estimations of the Powers of x ," *Proc. IEEE Int'l Conf. Application-Specific Systems, Architectures, and Processors*, pp. 334-339, 2005.
- [7] J.M. Muller, *Elementary Functions: Algorithms and Implementation*. Birkhauser Verlag AG, 1997.
- [8] D. Lee, J.D. Villasenor, W. Luk, and P.H.W. Leong, "A Hardware Gaussian Noise Generator Using the Box-Muller Method and Its Error Analysis," *IEEE Trans. Computers*, vol. 55, no. 6, June 2006.
- [9] R.E. Moore, *Interval Analysis*. Prentice-Hall, 1966.
- [10] L. de Figueiredo and J. Stolfi, "Self-Validated Numerical Methods and Applications," *Brazilian Math. Colloquium Monograph*, IMPA, Brazil, 1997.
- [11] D. Lee, A. Abdul Gaffar, O. Mencer, and W. Luk, "MiniBit: Bit-Width Optimization via Affine Arithmetic," *Proc. ACM/IEEE Design Automation Conf.*, pp. 837-840, 2005.
- [12] L. Ingber, *Adaptive Simulated Annealing (ASA)* 25.15, 2004, <http://www.ingber.com/#ASA>.