Analysis and optimization for hardware implementation of sine/cosine with faithful rounding and monotonicity through piecewise quadratic polynomial

Keli Long¹, Hu Chen², and Xujun Li^{1, a)}

Abstract Piecewise quadratic polynomial approximation is a well-established hardware function evaluation technique. In this paper, we demonstrate the analysis and optimization of hardware implementation of sine and cosine through piecewise quadratic polynomial for single-precision floating-point operations. First, detailed approximation error analysis was used to rapidly obtain the non-uniform segmentation and guarantee faithful rounding. Second, theoretical analysis demonstrates that the monotonicity of the algorithm using segmentation is consistent with the original sine and cosine function. Moreover, the bit-width is optimized for the sine and cosine floating-point arithmetic units based on static analysis and dynamic monitoring to reduce the hardware cost. Experimental results show that the implementation for sine and cosine functions meets the requirements of faithful rounding and monotonicity with reasonable hardware cost.

Keywords: monotonicity, faithful rounding, bit-width optimization, polynomial approximation

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

Piecewise quadratic polynomial approximation for hardware implementation of sine and cosine has drawn a significant concern [1, 2, 3, 4, 5, 6, 7, 8]. Such scheme adopts the second-order polynomial to approximate the sine and cosine functions in each segmented sub-interval after segmenting the range of the interest. Meanwhile, to achieve the faithful rounding of the single-precision floating-point of the IEEE 754 standard, the sub-interval segmentation should be based on the analysis of approximation error. Usually, an effective method for approximation error analysis is to obtain the uniform size sub-interval that achieves the accuracy requirements through the analysis of the relationship between the approximation error and the size of the sub-interval in worst case [8, 9, 10]. However, such method requires substantial subintervals and an large coefficient table for faithful single-precision rounding [11, 12, 13]. To reduce the number of subintervals, non-uniform segmentation methods in [9, 11, 12, 13, 14, 15, 16] are utilized at the cost of the analysis complexity. Moreover, piecewise approximation

and faithful rounding will probably introduce uncertainty on monotonicity. To ensure the monotonicity, work [17] adjusts the coefficient table empirically by considering the output in all ranges; work [18] adjusts the appropriate slopes and Table of Initial Values (TIV); and work [19] presents an algorithm that transforms the flat segments into monotonic segments. Unfortunately, these methods will result in a large coefficient table due to the lack of detailed analysis of the location and size of the non-monotonic region.

Additionally, to reduce hardware cost, the bit width on the critical path is required to be optimized. In general, the bit-width optimization is implemented based on the result of error analysis [1, 9, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]. Whereas, it will be time-consuming to dynamic monitor the error result via exhaustively considering all combination with different inputs and outputs to optimize the bit-width [8, 30].

In this paper, we first obtain the coefficients of a piecewise second-order polynomial using a Taylor series. Then, an approximation error analysis for the quadratic piecewise polynomial of sine function is presented obtain sub-intervals through non-uniform segmentation, achieving the faithful rounding of single-precision floating-point operations. In addition, the monotonicity for segmentation is further theoretically analyzed. Finally, we complete the bit-width optimization for the sine and cosine functions in a two-stage manner that initially analyze the quantization error statically and then dynamically monitor the error result within a small input range. The proposed method can be also used for the cosine function.

2. Framework

The inputs and outputs of the target structure are IEEE-754 single-precision floating-point format numbers. The target structure implements the function defined by the following equation:

$$\begin{cases} \text{Output} = \sin(\text{Input} \times 2\pi) \\ \text{Output} = \cos(\text{Input} \times 2\pi) \end{cases} \quad (1)$$

Let $f(x) = \sin(2\pi x)$ or $\cos(2\pi x)$ for $x \in [a, b)$. This interval $[a, b)$ is segmented into m ($m \geq 2$) contiguous subintervals, each of which is represented as $[a_i, a_{i+1})$, where $i \in [0, m-1]$, $i \in \mathbb{Z}$, $a_0 = a$, $a_m = b$. According to Taylor's theorem, the quadratic polynomial approximation of the sine function in the range of $[a_i, a_{i+1})$ is given as follows:

¹ School of Physics and Optoelectronics, Xiangtan University, Yuhu District, Xiangtan 411105, China

² Hunan Greatleo Microelectronics Ltd., Changsha 410000, China

^{a)} xjli@xtu.edu.cn

$$R = C_0 + (C_1 + C_2 \times D) \times D \quad (2)$$

where

$$\begin{cases} C_0 = \sin(2\pi x_i) \\ C_1 = 2\pi \cos(2\pi x_i) \\ C_2 = -2\pi^2 \sin(2\pi x_i) \\ D = x - x_i \end{cases}, x, x_i \in [a_i, a_{i+1}) \quad (3)$$

Fig. 1 presents the hardware architecture to calculate the sine and cosine functions. First, the preprocessing stage includes five steps. In Step 1, the input is compressed to x_f within the range of $[0, 1]$ through the periodicity of the sine or cosine function. In Step 2, x_f is mapped to x_u within the range of $[0, 0.25]$, and the sign-bit of the output is attained by the symmetry of the sine and cosine functions. The function Sel_I is used to indicate whether $\sin(\text{Input} \times 2\pi)$ or $\cos(\text{Input} \times 2\pi)$ is calculated. In Step 3, x_u is mapped to x within the range of $[0, 0.125]$ and the Sel function can be obtained, that is, $\text{Sel} = \text{Sel_I}$ and $x = x_u$ if $x_u \leq 0.125$, according to Eq. (4) $\text{Sel} = \overline{\text{Sel_I}}$ and $x = 0.25 - x_u$ otherwise. In Step 4, x_i and Addr are obtained from x and Sel. In Step 5, D is obtained by subtracting x_i from x .

$$\begin{cases} \sin(2\pi x_u) = \cos(2\pi(0.25 - x_u)) \\ \cos(2\pi x_u) = \sin(2\pi(0.25 - x_u)) \end{cases} \quad (4)$$

Second, coefficients C_0, C_1, C_2 are read from the look-up table that is stored in the ROM. Third, the result, R is computed using Eq. (2). Finally, the output is obtained after normalization.

As shown in Fig. 1, multiplication is a floating-point operation that involves the multiplication of the mantissa and addition of exponents; addition is also a floating-point operation that involves the matching of exponents and addition.

Fig. 1 Architecture of the design

3. Non-uniform segmentation

Assume that $ULP(p)$ represents 1 unit in the last place (ULP) for the real number p in the floating-point form. For simplification, ULP refers to $ULP(\text{Output})$ in this paper. There are three sources of error between the output and the real part of the initial function: 1) the approximation error ε_a due to polynomial approximation; 2) the quantization error ε_q due to finite precision for $C_0, C_1, C_2, M_0, M_1, M_2$, and 3) the rounding error of output does not exceed 0.5 ULP. Because the total error $\varepsilon \leq \varepsilon_a + \varepsilon_q + 0.5 \text{ ULP}$, for faithful rounding ($\varepsilon \leq 1 \text{ ULP}$), it requires that $\varepsilon_a + \varepsilon_q \leq 0.5$

ULP. According to Taylor's theorem, the error term for the second-order polynomial approximation of the sine function is given by the following:

$$r_2(x) = -\frac{8\pi^3 \cos(2\pi\xi)(x - x_i)^3}{6} \quad (5)$$

where ξ is between x_i and x .

Due to $\sin(2\pi x) \geq 4\sqrt{2}x$ for $x \in [0, 2^{-3})$, $ULP(\sin(2\pi x)) \geq ULP(4\sqrt{2}x)$. Although $ULP(4\sqrt{2}x)$ is occasionally slightly greater than $4\sqrt{2}ULP(x)$, to simplify the analysis, we assume that $ULP(4\sqrt{2}x) \approx 4\sqrt{2}ULP(x)$. Therefore, the approximation error ε_a is given as follows:

$$\varepsilon_a = \left| \frac{r_2(x)}{ULP(\sin(2\pi x))} \right| \leq \frac{8(|x - x_i|_{\max})^3}{ULP(x)} \quad (6)$$

When $\varepsilon_a < 0.5 \text{ ULP}$, according to Eq. (6), non-uniform segmentation is obtained as shown in Table I. Additionally, for this segmentation, $x_i = (a_i + a_{i+1})/2$ for interval $x \in [2^{-14}, 2^{-3})$ and $x_i = a_i$ for interval $x \in [0, 2^{-14})$. For example, for $x \in [2^{-4}, 2^{-3})$, $ULP(x) = 2^{-4-23} = 2^{-27}$, if $|x - x_i| \leq 2^{-10-1}$, then $\varepsilon_a \leq 0.125 \text{ ULP}$ based on Eq. (6); Similarly, if $|x - x_i|_{\max} = 2^{-10}$ for $x \in [2^{-4}, 2^{-3})$, existing $\varepsilon_a \geq 0.5 \text{ ULP}$, and if $|x - x_i|_{\max} = 2^{-12}$, the number of sub-intervals will be larger; thus, $|x - x_i|_{\max} = 2^{-11}$ is obtained for $x \in [2^{-4}, 2^{-3})$. Furthermore, we can obtain $\varepsilon_a \leq 0.125 \text{ ULP}$ for all intervals according to Table I and Eq. (6).

Table I Non-uniform segmentation of the region $[0, 0.125]$

Intervals	Sub-interval length
$[2^{-4}, 2^{-3})$	2^{-10}
$[2^{-7}, 2^{-4})$	2^{-11}
$[2^{-10}, 2^{-7})$	2^{-12}
$[2^{-13}, 2^{-10})$	2^{-13}
$[2^{-14}, 2^{-13})$	2^{-14}
$[0, 2^{-14})$	2^{-15}

Because $\varepsilon_a \leq 0.125 \text{ ULP}$, requiring $\varepsilon_a + \varepsilon_q \leq 0.5 \text{ ULP}$ leads to requiring $\varepsilon_q \leq 0.375 \text{ ULP}$. Let $\varepsilon_{C0}, \varepsilon_{C1}, \varepsilon_{C2}, \varepsilon_{M0}, \varepsilon_{M1}, \varepsilon_{M2}$ denote the errors of the output caused by the finite precision of C_0, C_1, C_2, M_0, M_1 and M_2 , respectively. Then, the quantization error ε_q is given by the following:

$$\varepsilon_q \leq \varepsilon_{C0} + \varepsilon_{C1} + \varepsilon_{C2} + \varepsilon_{M0} + \varepsilon_{M1} + \varepsilon_{M2} \quad (7)$$

Error allocation is accomplished by adjusting the bit width of the coefficients and intermediate results, as indicated in Section 5.

4. Monotonicity

The input is compressed to x , which is in the range of $[0, 2^{-3})$ and is used to evaluate the polynomial. If $x \in [0, 2^{-15})$, $x_i = 0$, and we obtain $C_0 = C_2 = 0, C_1 = 2\pi$ from Eq. (2); then, $\text{Output} = 2\pi x$. In addition, for $x = 2^{-3}$, assuming $C_1 = C_2 = 0$ and $C_0 = \sin(\pi/4)$; then, $\text{Output} = C_0$. Thus, the monotonicity of the polynomial approximation needs to be analyzed in the range of $x \in [2^{-1}, 2^{-3})$.

As shown in Fig. 2, the piecewise polynomial approximation for the sine function is non-monotonic in the range of $[a_i, a_{i+2})$ via software simulation. x is discrete, which conforms to the IEEE-754 standard single-precision floating-point form. We assume that, in the interval $[2^j, 2^{j+1})$, the

Fig. 2 Graph of piecewise polynomial approximation for sine function in the range of $x \in [a_i, a_{i+2}]$

distance between the two closest x values is $\beta = 2^{j-23}$ and the length of the non-monotonic region is $\alpha = \alpha_1 + \alpha_2$. If $\beta \geq \alpha$, the non-monotonic region will be skipped, and the polynomial approximation will be monotonic in interval $[2^j, 2^{j+1})$. Regarding the monotonicity at the junction of intervals $[2^{j-1}, 2^j)$ and $[2^j, 2^{j+1})$, we consider both intervals simultaneously, and the smaller $\beta = 2^{j-1-2}$ of the two intervals is regarded as β of interval $[2^{j-1}, 2^{j+1})$.

According to Eqs. (2) and (3), the polynomial used to approximate the sine function in the range of $x \in [a_i, a_{i+1})$ is given by the following:

$$f(x) = \sin(2\pi x_i) + 2\pi(x - x_i)\cos(2\pi x_i) - 2\pi^2(x - x_i)^2\sin(2\pi x_i) \quad (8)$$

We assume that there are $x_1 = a_{i+1}$, $x_2 = a_{i+1} - \alpha_2$, $b_i = (a_{i+1} - a_i)/2$, $b_{i+1} = (a_{i+2} - a_{i+1})/2$, $x_{i+1} = a_{i+1} + b_{i+1}$, $x_i = a_{i+1} - b_i$, $b_{i+1} \geq b_i \gg \alpha_2 > 0$, $A_i = 2\pi a_i$, $A_{i+1} = 2\pi a_{i+1}$, $B_i = 2\pi b_i$, $B_{i+1} = 2\pi b_{i+1}$. Then, we can know that $B_i < 2^{-8}$, $B_{i+1} < 2^{-8}$ and $B_i \leq B_{i+1} \leq A_{i+1} < \pi/4$ from Section 3. According to Eq. (8), when $f(x_1) = f(x_2)$, ignoring the quadratic term of α_2 , $2\pi\alpha_2$ is given by the following:

$$2\pi\alpha_2 = Q(B_i) = G_1(B_i) - G_2(B_i) \quad (9)$$

where

$$\begin{cases} G_1(B_i) = \frac{G_0(B_i)}{H(B_i)}, G_2(B_i) = \frac{G_0(-B_{i+1})}{H(B_i)} \\ H(B_i) = \cos(A_{i+1} - B_i) - B_i \sin(A_{i+1} - B_i) \\ G_0(t) = \sin(A_{i+1} - t) \left(1 - \frac{t^2}{2}\right) + t \cos(A_{i+1} - t) \end{cases}$$

Furthermore, we can get:

$$Q'(t) = \frac{t \cos(A_{i+1} - t)}{H^2(t)} P(t, B_{i+1}), t \in [0, 2^{-8}) \quad (10)$$

where

$$\begin{cases} P(t, B_{i+1}) = G_0(-B_{i+1}) - P_0(t) \\ P_0(t) = \sin(A_{i+1} - t) + \frac{t \cos(A_{i+1} - t)}{2} \end{cases} \quad (11)$$

We can know that $B_{i+1} = B_i$ or $B_{i+1} = 2B_i$ from Section 3. And we can get $dG'_0(-t)/dt \leq 0$ for $t \in [0, 2^{-8})$, $A_{i+1} \in [0, \pi/4)$, then $G_0(-B_{i+1}) \geq G_0(-2t)$ for $B_{i+1} \in [t, 2t]$, thus

$P(t, B_{i+1}) \geq P(t, 2t)$ from Eq. (11). Meanwhile, we can get $P'(t, 2t) > 0$ and $P(0, 0) = 0$ for $t \in [0, 2^{-8})$, $t \leq A_{i+1}$, then $P(t, 2t) \geq 0$, so $P(t, B_{i+1}) \geq 0$ for $P(t, B_{i+1}) \geq P(t, 2t)$. From Eq. (10), $Q'(t) \geq 0$ for $t \leq A_{i+1}$, thus $Q(B_{i+1}) \geq Q(B_i)$. Therefore, according to Eq. (9), we can get:

$$2\pi\alpha_2 \leq Q(B_{i+1}) = \frac{N(B_{i+1})}{L(B_{i+1})} \quad (12)$$

where

$$\begin{cases} N(B_{i+1}) = 2 \left(B_{i+1} \cos B_{i+1} - \left(1 - \frac{B_{i+1}^2}{2}\right) \sin B_{i+1} \right) \\ L(B_{i+1}) = \frac{H(B_{i+1})}{\cos A_{i+1}} \end{cases}$$

We can get $L(0) = 1$, and $L'(t) \geq 0$ for $t \in [0, 2^{-8})$ and $t \leq A_{i+1}$, thus $L(B_{i+1}) \geq 1$. Thus, according to Eq. (12), we know that:

$$2\pi\alpha_2 \leq N(B_{i+1}) \quad (13)$$

Assuming $T(t) = t^3/3 - N(t)$, then $T(0) = 0$ and $T'(t) = t^2(1 - \cos t) \geq 0$ for $t \in [0, 2^{-8})$, thus $T(B_{i+1}) \geq 0$, which lead to is $N(B_{i+1}) \leq B_{i+1}^3/3$. From Eq. (13), we can get $2\pi\alpha_2 \leq B_{i+1}^3/3$. Therefore, α_2 is given by:

$$\alpha_2 \leq \frac{B_{i+1}^3}{6\pi} = \frac{4\pi^2 b_{i+1}^3}{3} \quad (14)$$

Similarly, we also can get $\alpha_1 \leq B_{i+1}/3$. Thus,

$$\alpha = \alpha_1 + \alpha_2 \leq \frac{8\pi^2 b_{i+1}^3}{3} \quad (15)$$

According to the non-uniform segmentation shown in Table I and Eq. (15), $\beta \geq \alpha$ in the interval $[2^{j-1}, 2^{j+1})$, $j \in [-14, -4]$, $j \in \mathbb{Z}$; $\beta \geq \alpha$ when $x \in [2^{-1}, 2^{-3})$. For example, if $j = -14$, $x \in [2^{-1}, 2^{-13})$, then $\beta \geq 2^{-3}$ and $\alpha < 2^{-40}$. Therefore, the non-uniform piecewise polynomial approximation for the sine function is monotonic.

5. Bit-width optimization

To reduce the hardware resource, we can decrease the signal bit-width of the main data path using a static analysis, and further compress the bit width via a dynamic monitoring.

5.1 Static analysis

To reduce costs, we consider the following: 1) allocate more errors to coefficients and intermediate results that undergo more complex operations, such as C_2 ; 2) allocate smaller errors to those that have a greater impact factor on the output, the impact factor is the value of the parameters 1, D , or D^2 in Eq. (16) corresponding to the item; and 3) allocate smaller errors to the coefficients and intermediate results that are closer to the end of the data path to reduce precision loss; namely, $\varepsilon_{c2} \geq \varepsilon_{m2} \geq \varepsilon_{m0}$ and $\varepsilon_{c1} \geq \varepsilon_{m1} \geq \varepsilon_{m0}$. Thus, we consider the following error allocation: $\varepsilon_{c0}, \varepsilon_{m0} \leq 2^{-6}\text{ULP}$, $\varepsilon_{c1}, \varepsilon_{c2} \leq 2^{-3}\text{ULP}$, $\varepsilon_{m1} \leq 2^{-5}\text{ULP}$, and $\varepsilon_{m2} \leq 2^{-4}\text{ULP}$. According to Eq. (2) and Fig. 1, we obtain the following:

$$\begin{cases} \varepsilon_{C0} = 1 \times \text{ULP}(C_0), \varepsilon_{M0} = 1 \times \text{ULP}(M_0) \\ \varepsilon_{C1} = D \times \text{ULP}(C_1), \varepsilon_{M1} = D \times \text{ULP}(M_1) \\ \varepsilon_{C2} = D^2 \times \text{ULP}(C_2), \varepsilon_{M2} = D \times \text{ULP}(M_2) \end{cases} \quad (16)$$

According to Eqs. (3) and (16) as well as Table I, the bit width of the coefficients and intermediate results could be obtained as shown in Table II. For example, according to Table I, $D \leq 2^{-10-1}$, then $\varepsilon_{C2} \leq 2^{-22}ULP(C_2)$ from Eq. (16). Because $\varepsilon_{C2} \leq 2^{-22}ULP(C_2)$, requiring $\varepsilon_{C2} \leq 2^{-3}ULP$ leads to requiring $ULP(C_2) = 2^{19}ULP$. Due to $|C_2| = 2\pi^2 \sin(2\pi x_i)$ from Eq. (3) and $\sin(2\pi x_i) \leq 2 \sin(2\pi x)$ for $x, x_i \in [2^{-15}, 2^{-3})$ from Section 3, then $|C_2| \leq 4\pi^2 \sin(2\pi x)$, thus $|C_2| \leq 2^6 Output$ for $Output = \sin(2\pi x)$. Thus, the bit-width of C_2 should be $23 - 19 + 6 + 9 = 19$, where 23 represents a 23-bit mantissa for the output in the single-precision floating-point format, and 9 means the 1 sign-bit and 8 exponent bits.

Table II Bit-width for coefficients and intermediate results obtained by static analysis

Signals name	C_0	C_1	C_2	M_0	M_1	M_2
Bit width	39	39	19	42	41	20

Table III Bit-width of coefficients and intermediate results obtained by dynamic monitoring

Signals name	C_0	C_1	C_2	M_0	M_1	M_2
Bit width	37	35	18	37	35	20

5.2 Dynamic monitoring

To reduce the signal bit-width for the polynomial evaluation, dynamic monitoring is used, which comprises the following three main steps. The results are shown in Table III.

Step 1. Reduce the bit width of the signal in the circuit presented in Fig. 1.

Step 2. The outputs generated with all inputs in the range of $[2^{-15}, 2^{-3})$ are tested for precision and monotonicity. If the test passes, signal bit-width reduction is successful. Otherwise, the reduction fails, and the state of the circuit should be restored to its original state after Step 1.

Step 3. Repeat Steps 1 and 2 for the next signal to complete the reduction of all signals.

Because inputs are mapped to the range of $[0, 2^{-3}]$ and polynomial approximation is performed, we only need to test inputs in the range of $[0, 2^{-3}]$. In addition, when inputs are in the range of $[0, 2^{-15})$, $Output = 2\pi x$. Thus, we do not need to perform a dynamic error analysis within the range of $[0, 2^{-15})$, and only need to consider the error of the multiplier.

6. Result

6.1 Precision assessment

Fig. 3(a) and (b) indicate that errors are less than 1 ULP for the outputs of the sine and cosine operations, respectively, and the inputs are in the range of $[2^{-16}, 2^{-3}]$. All inputs are compressed to the range of $[0, 2^{-3}]$. When the inputs are in the range of $[0, 2^{-15})$, the output for the sine operation is equal to $Input \times 2\pi$, for the cosine operation is equal to 1. It is easy to determine that the error in this range is less than 1 ULP based on Section 3. Thus, the output error is less than 1 ULP, and the arithmetic unit achieves the accuracy requirement for sine and cosine operations.

Fig. 3 Error of outputs from (a) sine and (b) cosine operations, and the inputs are in the range of $[2^{-16}, 2^{-3}]$.

6.2 Monotonicity assessment

Fig. 4(a) and (b) present the difference for two adjacent outputs for the sine and cosine operations, respectively, and the inputs are in the range of $[2^{-16}, 2^{-2}]$. The two adjacent outputs are obtained by the sine or cosine operations from any two adjacent inputs; moreover, the difference is obtained by subtracting one output from the other, and the output as the subtracted number is calculated from the larger input. Thus, within the input range $[2^{-1}, 2^{-2}]$, the outputs monotonically increase and decrease for the sine and cosine operations respectively. The inputs are compressed to the range of $[0, 0.25]$ by the periodicity and symmetry of the sine and cosine functions. Thus, the outputs of our circuit for sine and cosine operations have the same monotonicity as the original function.

Fig. 4 Difference for all two adjacent outputs that are from the (a) sine and (b) cosine operations, and the inputs are in the range of $[2^{-1}, 2^{-2}]$.

Fig. 5(a) presents the outputs of the sine operation, and the inputs are in the range of $[2^{-6} - 2^{-26}, 2^{-6} + 2^{-25}]$. The outputs are monotonous. Fig. 5 (b) presents the outputs of the instruction SINPUF32 in TMS320F2837xD; the inputs are also in this range. The outputs are not monotonous.

6.3 Resource

A Xilinx Virtex-4 XC4VLX100-12 FPGA was used for the experimental implementation. The designs are written in Verilog, synthesized with Synplicity Synplify Pro 7.7.1, and placed-and-routed with Xilinx ISE 7.1.03i. The hardware

Fig. 5 (a) Outputs of sine operation, (b) outputs of the instruction SINPUF32 in TMS320F2837xD; input signals are in the range of $[2^{-6} - 2^{-26}, 2^{-6} + 2^{-25}]$.

resource cost of our designed circuit and that of other related work [22] is shown in Table IV. Although dedicated RAMs and multiply-and-add blocks are present on the device, we consider implementations based on slices (programmable logic blocks) only to obtain unbiased comparisons. As shown in Table IV, the circuit in the referenced work [22] uses a similar quadratic polynomial approximation to complete the 32-bit operation, and is approximately equal to that used in this study considering the area. Furthermore, the circuit from the previous study is superior in terms of delay, but cannot preserve the monotonicity of the outputs.

Table IV Synthesis results for logic-only implementations

ITEM COMPARED	Designed in this study	Degree-2 Piecewise Polynomials [22]
Area(Slices)	3077	3011
Delay(ns)	63.842	47.935

7. Conclusion

In conclusion, we presented a detailed analysis and bit-width optimization for a piecewise quadratic polynomial hardware sine and cosine evaluation. All sources of error are carefully analyzed, the total error is limited, and faithful rounding for single-precision sine and cosine operations is achieved. In particular, monotonicity is preserved by limiting the size of the non-monotonic region. Furthermore, the bit width optimization is completed by detailed static analysis and dynamic monitoring to reduce the hardware resource cost. Ultimately, the experimental results show that the single-precision floating-point arithmetic unit for the sine and cosine functions achieves faithful rounding and monotonicity within the rational hardware resources. In addition, we believe that the detailed analysis and optimization presented in this study is significant for the optimization of piecewise polynomial hardware operations for other functions.

References

- [1] D. De Caro, *et al.*: “High-performance special function unit for programmable 3-D graphics processors,” *IEEE Trans. Circuits Syst. I, Reg. Papers* **56** (2009) 1968 (DOI: [10.1109/TCSI.2008.2010150](https://doi.org/10.1109/TCSI.2008.2010150)).
- [2] W. Jose, *et al.*: “Efficient implementation of a single-precision floating-point arithmetic unit on FPGA,” 2014 24th International Conference on Field Programmable Logic and Applications (FPL) (2014) 1 (DOI: [10.1109/FPL.2014.6927391](https://doi.org/10.1109/FPL.2014.6927391)).
- [3] M. Sadeghian, *et al.*: “Optimized linear, quadratic and cubic interpolators for elementary function hardware implementations,” *Electronics* **5** (2016) 17 (DOI: [10.3390/electronics5020017](https://doi.org/10.3390/electronics5020017)).
- [4] J.-A. Pineiro, *et al.*: “High-speed function approximation using a minimax quadratic interpolator,” *IEEE Trans. Comput.* **54** (2005) 304 (DOI: [10.1109/TC.2005.52](https://doi.org/10.1109/TC.2005.52)).
- [5] E. Walters III: “Linear and quadratic interpolators using truncated-matrix multipliers and squarers,” *Computers* **4** (2015) 293 (DOI: [10.3390/computers4040293](https://doi.org/10.3390/computers4040293)).
- [6] E.G. Walters and M.J. Schulte: “Efficient function approximation using truncated multipliers and squarers,” 17th IEEE Symposium on Computer Arithmetic (ARITH’05) (2005) 232 (DOI: [10.1109/ARITH.2005.18](https://doi.org/10.1109/ARITH.2005.18)).
- [7] J. Cao, *et al.*: “High-performance architectures for elementary function generation,” *Proceedings 15th IEEE Symposium on Computer Arithmetic* (2001) 136 (DOI: [10.1109/ARITH.2001.930113](https://doi.org/10.1109/ARITH.2001.930113)).
- [8] A. Alimohammad, *et al.*: “A unified architecture for the accurate and high-throughput implementation of six key elementary functions,” *IEEE Trans. Comput.* **59** (2010) 449 (DOI: [10.1109/TC.2009.169](https://doi.org/10.1109/TC.2009.169)).
- [9] D. De Caro, *et al.*: “Minimizing coefficients wordlength for piecewise-polynomial hardware function evaluation with exact or faithful rounding,” *IEEE Trans. Circuits Syst. I, Reg. Papers* **64** (2017) 1187 (DOI: [10.1109/TCSI.2016.2629850](https://doi.org/10.1109/TCSI.2016.2629850)).
- [10] S. Nagayama, *et al.*: “Numeric function generators using piecewise arithmetic expressions,” 41st IEEE International Symposium on Multiple-Valued Logic (ISMVL) (2011) 7 (DOI: [10.1109/ISMVL.2011.32](https://doi.org/10.1109/ISMVL.2011.32)).
- [11] T. Sasao, *et al.*: “Numerical function generators using LUT cascades,” *IEEE Trans. Comput.* **56** (2007) 826 (DOI: [10.1109/TC.2007.1033](https://doi.org/10.1109/TC.2007.1033)).
- [12] D.-U. Lee, *et al.*: “Hierarchical segmentation schemes for function evaluation,” *Proc. 2003 IEEE International Conference on Field-Programmable Technology (FPT)* (2003) 92 (DOI: [10.1109/FPT.2003.1275736](https://doi.org/10.1109/FPT.2003.1275736)).
- [13] D.-U. Lee, *et al.*: “Hardware implementation trade-offs of polynomial approximations and interpolations,” *IEEE Trans. Comput.* **57** (2008) 686 (DOI: [10.1109/TC.2007.70847](https://doi.org/10.1109/TC.2007.70847)).
- [14] V. Paliouras, *et al.*: “A floating-point processor for fast and accurate sine/cosine evaluation,” *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **47** (2000) 441 (DOI: [10.1109/82.842112](https://doi.org/10.1109/82.842112)).
- [15] H.-J. Ko, *et al.*: “A new non-uniform segmentation and addressing remapping strategy for hardware-oriented function evaluators based on polynomial approximation,” *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (2010) 4153 (DOI: [10.1109/ISCAS.2010.5537607](https://doi.org/10.1109/ISCAS.2010.5537607)).
- [16] D.-U. Lee, *et al.*: “Hierarchical segmentation for hardware function evaluation,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **17** (2009) 103 (DOI: [10.1109/TVLSI.2008.2003165](https://doi.org/10.1109/TVLSI.2008.2003165)).
- [17] C. Iordache and D.W. Matula: “Analysis of reciprocal and square root reciprocal instructions in the AMD K6-2 implementation of 3DNow!,” *Electron. Notes Theor. Comput. Sci.* **24** (2000) 34 (DOI: [10.1016/S1571-0661\(05\)80621-8](https://doi.org/10.1016/S1571-0661(05)80621-8)).
- [18] F. de Dinechin and A. Tisserand: “Multipartite table methods,” *IEEE Trans. Comput.* **54** (2005) 319 (DOI: [10.1109/TC.2005.54](https://doi.org/10.1109/TC.2005.54)).
- [19] D.B. Thomas: “A general-purpose method for faithfully rounded floating-point function approximation in FPGAs,” 2015 IEEE 22nd Symposium on Computer Arithmetic (2015) 42 (DOI: [10.1109/ARITH.2015.27](https://doi.org/10.1109/ARITH.2015.27)).
- [20] D. Wang, *et al.*: “A high-throughput fixed-point complex divider for FPGAs,” *IEICE Electron. Express* **10** (2013) 20120879 (DOI: [10.1587/leex.10.20120879](https://doi.org/10.1587/leex.10.20120879)).
- [21] J. Detrey and F. de Dinechin: “Table-based polynomials for fast hardware function evaluation,” 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP’05) (2005) 328 (DOI: [10.1109/ASAP.2005.61](https://doi.org/10.1109/ASAP.2005.61)).
- [22] D.-U. Lee and J. D. Villasenor: “A bit-width optimization methodology for polynomial-based function evaluation,” *IEEE Trans. Comput.* **56** (2007) 567 (DOI: [10.1109/TC.2007.1013](https://doi.org/10.1109/TC.2007.1013)).
- [23] F. de Dinechin, *et al.*: “Automatic generation of polynomial-based hardware architectures for function evaluation,” ASAP 2010 - 21st IEEE International Conference on Application-specific Systems, Architectures and Processors (2010) 216 (DOI: [10.1109/ASAP.2010.5540952](https://doi.org/10.1109/ASAP.2010.5540952)).
- [24] F. de Dinechin, *et al.*: “Fixed-point trigonometric functions on FPGAs,” *ACM SIGARCH Comput. Archit. News* **41** (2013) 83 (DOI: [10.1145/2641361.2641375](https://doi.org/10.1145/2641361.2641375)).
- [25] C. Chen: “High-order Taylor series approximation for efficient computation of elementary functions,” *IET Comput. Digit. Tech.* **9** (2015) 328 (DOI: [10.1049/iet-cdt.2014.0158](https://doi.org/10.1049/iet-cdt.2014.0158)).
- [26] J.Y.L. Low and C.C. Jong: “A memory-efficient tables-and-additions method for accurate computation of elementary functions,” *IEEE Trans. Comput.* **62** (2013) 858 (DOI: [10.1109/TC.2012.43](https://doi.org/10.1109/TC.2012.43)).
- [27] F. Merchant, *et al.*: “Efficient realization of table look-up based double precision floating point arithmetic,” 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID) (2016) 415 (DOI: [10.1109/VLSID.2016.113](https://doi.org/10.1109/VLSID.2016.113)).
- [28] S. Nandi, *et al.*: “Fixed point implementation of trigonometric

- function using Taylor's series and error characterization," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2016) 442 (DOI: [10.1109/ICACCI.2016.7732085](https://doi.org/10.1109/ICACCI.2016.7732085)).
- [29] J. Detrey and F. de Dinechin: "Floating-point trigonometric functions for FPGAs," 2007 International Conference on Field Programmable Logic and Applications (2007) 29 (DOI: [10.1109/FPL.2007.4380621](https://doi.org/10.1109/FPL.2007.4380621)).
- [30] B. Lee and N. Burgess: "Some results on taylor-series function approximation on FPGA," The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers (2003) 2198 (DOI: [10.1109/ACSSC.2003.12923700](https://doi.org/10.1109/ACSSC.2003.12923700)).