

The CORE-MATH asin is correctly rounded

Paul Zimmermann

May 4, 2026

Abstract—We prove that the CORE-MATH binary64 arc-sine function is correctly rounded.

Index Terms—IEEE 754, binary64 format, correct rounding.

We consider the CORE-MATH `asin.c` file from commit `3ba8f60`. The function `asin(x)` is only defined for $|x| \leq 1$, and it increases from $-\pi/2$ to $\pi/2$ on $[-1, 1]$ (see Figure 1). Since $\text{asin}(-x) = -\text{asin}(x)$, and all approximations used in

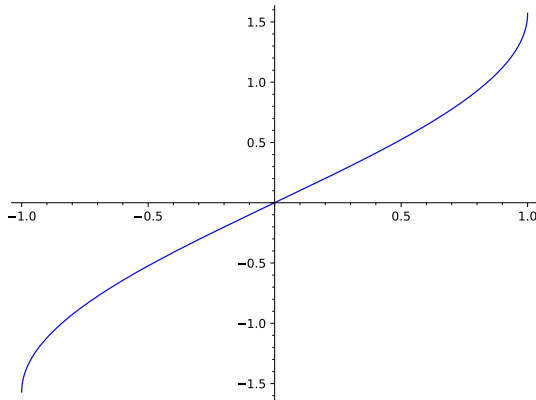


Fig. 1. Graph of $\text{asin}(x)$ on $[-1, 1]$.

the implementation are odd, it suffices to assume $x \geq 0$. We consider all four rounding modes from the C23 standard: to nearest-even, toward zero, toward $+\infty$, and toward $-\infty$. We denote by $\circ(\cdot)$ the rounding mode.

I. BRANCH $0 \leq x < x_0$

Let $x_0 = 0x1.7137449123ef6p-26$. For $0 \leq x < x_0$, CORE-MATH returns $\text{fma}(2^{-55}, x, x)$. We prove this is the correct rounding of $\text{asin}(x)$.

Lemma 1: For $0 \leq x < x_0$, x a binary64 number, and any rounding mode, $\text{fma}(2^{-55}, x, x)$ is the correct rounding of $\text{asin}(x)$.

Proof: The lemma is true for $x = \pm 0$, since it yields ± 0 . For $0 < x < 0.5$, we have:

$$x < \text{asin}(x) < x + 0.2x^3.$$

First consider $0 < x < 2^{-26}$. Then $0.2x^3 < 2^{-54}x$, and since $x < 2^{53}\text{ulp}(x)$, it follows $x < \text{asin}(x) < x + \frac{1}{2}\text{ulp}(x)$, thus $\text{asin}(x)$ rounds either to x or to $\text{nextup}(x)$, the latter case for rounding upward. The formula $\text{fma}(2^{-55}, x, x)$ returns the same value.

Now consider $2^{-26} \leq x < x_0$. We then have $\text{ulp}(x) = 2^{-78}$. Since $\text{asin}(x) - x$ is increasing for x positive, we have $\text{asin}(x) \leq x + \tau$, where τ is the value of $\text{asin}(x) - x$ for the

binary64 number just below x_0 . We check numerically that $\tau < 2^{-79}$, thus $x < \text{asin}(x) < x + \frac{1}{2}\text{ulp}(x)$, and the result follows as above. ■

Note: the bound $x < x_0$ is optimal, since for $x = x_0$, $\text{asin}(x)$ rounds to $\text{nextup}(x)$ to nearest, whereas $\text{fma}(2^{-55}, x, x)$ rounds to x .

We might replace $m = 2^{-55}$ in $\text{fma}(m, x, x)$ by any positive constant less or equal to 2^{-54} (for $x < x_0$ just below a power of two, if $m > 2^{-54}$, $\text{fma}(m, x, x)$ rounds to $\text{nextup}(x)$ to nearest). However, $\text{fma}(2^{-54}, x_0, x_0)$ rounds to x_0 to nearest, whereas $\text{asin}(x_0)$ rounds to $\text{nextup}(x_0)$. Thus even increasing m to the maximal value, the bound x_0 cannot be increased.

II. BRANCH $x_0 \leq x \leq 0.5$

For $x_0 \leq x \leq 0.5$, first an integer j is computed, $0 \leq j \leq 32$, and a polynomial approximation depending on j is used:

$$q(x) = x \cdot (c_0 + c_1 + c_2t + c^3t^2 + \dots + c_7t^6),$$

where $t = x^2 - j \cdot 2^{-7}$ is a reduced argument. This is detailed in Algorithm `FastPathSmall`, which uses an auxiliary routine (`muldd`).

Algorithm 1 (FastPathSmall)

Input: x a binary64 number, $x_0 \leq x \leq 0.5$

Output: the correct rounding $\circ(\text{asin}(x))$ or FAIL

- 1: $j = \lfloor \circ(x^2) \cdot 2^7 \rfloor$
 - 2: $t = \text{fma}(x, x, -2^{-7}j)$
 - 3: let c_0, c_1, \dots, c_7 the polynomial coefficients for j
 - 4: $t_2 = \circ(t \cdot t)$
 - 5: $g_2 = \circ(c_2 + \circ(tc_3))$, $h_4 = \circ(c_4 + \circ(tc_5))$
 - 6: $h_6 = \circ(c_6 + \circ(tc_7))$, $g_4 = \circ(h_4 + \circ(t_2h_6))$
 - 7: $e_2 = \circ(g_2 + \circ(t_2g_4))$, $d = \circ(te_2)$
 - 8: $f_h = c_0$, $f_\ell = \circ(c_1 + d)$
 - 9: $h, \ell = \text{muldd}(x, 0, f_h, f_\ell)$
 - 10: $\epsilon = \circ(\circ(xt), 0x1.10p-52)$
 - 11: $\ell_b = h + \circ(\ell - \epsilon)$, $u_b = h + \circ(\ell + \epsilon)$
 - 12: if $\circ(\ell_b) = \circ(u_b)$ return $\circ(\ell_b)$ else return FAIL
-

A. Branch $x_0 \leq x \leq 2^{-4}$

The case $j \geq 1$ is more difficult to analyze, since in the error bound ϵ at line 10, the reduced argument t might vanish. We thus analyze separately the case $j = 0$, which corresponds to $x_0 \leq x \leq 2^{-4}$ (the integer j is rounded to the nearest even integer in the code), in which case we have $t = \circ(x^2)$.

We wrote a SageMath program which, starting from the interval $[x_0, 2^{-4}]$, computes using Sollya an upper bound s

Algorithm 2 (muldd)

Input: x_h, x_ℓ, c_h, c_ℓ binary64 numbers**Output:** h, ℓ approximating $(x_h + x_\ell)(c_h + c_\ell)$

- 1: $h \leftarrow \circ(x_h c_h)$
 - 2: $\ell_1 \leftarrow \text{fma}(x_h, c_h, -h)$
 - 3: $\ell_2 \leftarrow \circ(x_h c_\ell)$
 - 4: $\ell_3 \leftarrow \circ(x_\ell c_h)$
 - 5: $\ell \leftarrow \circ(\ell_2 + \ell_3) + \ell_1$
-

such that $q(x) - \text{asin}(x) < sx^3$, then checks using Gappa that $\ell_b - q(x) + sx^3 \leq 0$. If this holds, then adding the two inequalities yields $\ell_b < \text{asin}(x)$, thus $\circ(\ell_b) \leq \circ(\text{asin}(x))$. If this does not hold, then we cut the current interval in two and recurse. The Sollya bound on the reduced intervals might be smaller, which might allow one to conclude. Using this program, we were able to prove $\ell_b < \text{asin}(x)$ for all four rounding modes in $[x_0, 2^{-4}]$. This SageMath program took up to about 70 minutes for rounding upwards, for which the value of ℓ_b is the largest one.

We wrote a similar program that computes a lower bound s' such that $q(x) - \text{asin}(x) > s'x^3$ with Sollya, then checks using Gappa that $u_b - q(x) + s'x^3 \geq 0$. Adding both inequalities yields $\text{asin}(x) < u_b$, thus $\circ(\text{asin}(x)) \leq \circ(u_b)$. This program took up to about 90 minutes for rounding down or toward zero, for which the value of u_b is the smallest one.

In summary, we have proven $\circ(\ell_b) \leq \circ(\text{asin}(x)) \leq \circ(u_b)$, thus if $\circ(\ell_b) = \circ(u_b)$, it is $\circ(\text{asin}(x))$.

B. Branch $2^{-4} < x \leq 0.5$

We used exhaustive search in this branch, as described in §III. This exhaustive search revealed some failures with a previous error bound, for example with $\epsilon = \circ(\circ(xt), 0x1.0\text{fp-52})$, the input $x = 0x1.\text{fa3c79a3c19abp-3}$ is not correctly rounded for rounding toward zero and no FMA contraction.

III. BRANCH $0.5 < x \leq 1$

For $x = 1$ the code returns $\circ(p_h + p_\ell)$, where $p_h + p_\ell$ is a double-double approximation of $\pi/2$. This is clearly the correct rounding. We thus consider now $0.5 < x < 1$.

For $0.5 < x < 1$, CORE-MATH uses the formula

$$\text{asin}(x) = \frac{\pi}{2} - 2 \text{asin} \sqrt{\frac{1-x}{2}},$$

where $0 \leq \sqrt{(1-x)/2} \leq 0.5$, and $0 \leq 2 \text{asin} \sqrt{(1-x)/2} < 1.0472$, thus there is a cancellation of at most one bit in the above formula.

We used exhaustive search in this branch, using the algorithm described in Section II-C of [4]. The original error bound for this branch was $|zt|\epsilon + 2^{-100}$, with z, t reduced arguments and $\epsilon = 0x1.962\text{p-52}$. We found it failed for $x = 0x1.3f47056fc030ap-1$ with rounding toward zero and no FMA contraction. The exhaustive search shows it works with an error bound $|zt|\epsilon$ with $\epsilon = 0x1.99\text{p-52}$.

IV. ACCURATE PATH

The accurate path (routine `as_asin_refine`) uses a similar algorithm as the accurate path for `acos(x)` [3].

In this section we prove the correctness of the accurate path for $x_0 \leq x \leq 2^{-4}$, since for $2^{-4} < x < 1$ we performed an exhaustive search, which checks the accurate path is correct when it is called. Moreover we know from §I that `asin(x)` is correctly rounded for $0 \leq x < x_0$.

The accurate path takes as input the binary64 number x and the approximation ϕ of `asin(x)` computed by `FastPathSmall` ($\phi = \ell_b$). From ϕ , an integer $j \in [0, 32]$ is deduced by $j \approx \lceil 64 \cdot |\phi|/\pi \rceil$. For our domain of interest $|x| < 2^{-4}$, we have $|\phi| < 0.0626$ and $0 \leq j \leq 1$. Since ϕ has the same sign as x :

$$\text{asin}(x) = \pm j \frac{\pi}{64} + \delta, \quad (1)$$

where \pm is a plus sign for $x > 0$, a minus sign for $x < 0$, and $|\delta|$ is bounded by $\approx \pi/128$. Indeed, due to approximations and rounding errors, $|\delta|$ might slightly exceed $\pi/128$. For a given j , sign of x , and rounding mode, $|\delta|$ is maximal at the boundaries of the range of x with given sign yielding this value of j . We found that at all these boundaries, $|\delta| < 0.0246$. Eq. (1) translates to:

$$\delta = \text{asin}(x) \mp j \frac{\pi}{64},$$

which with $\delta' = \text{sign}(x)\delta$ translates to:

$$\sin(\delta') = \cos(j \frac{\pi}{64})|x| - \sin(j \frac{\pi}{64})\sqrt{1-x^2}. \quad (2)$$

Double-double approximations of $\cos(j \frac{\pi}{64})$ and $\sin(j \frac{\pi}{64})$ are tabulated, which together with a double-double approximation of $\sqrt{1-x^2}$, yield a double-double approximation of $\sin(\delta)$. Using a polynomial approximation of `asin(t)`, we deduce a double-double approximation of δ . Then it suffices to plug this approximation into Eq. (1). Twenty-nine hard-to-round cases are not rounded correctly by this algorithm, and need to be dealt with separately.

For simplicity, we omit roundings in lines 19 and 25 of Algorithm `AccuratePath`. The only differences with the accurate path for `acos(x)` [3] are in the computation of j , the sign of v , and the fact that `fastsum` is called instead of `sum` at line 31. Indeed, $f_h + f_\ell$ approximates δ after line 27. For $j = 0$, the `fastsum` condition $|f_h| \geq |p_\ell|$ is trivially satisfied, since $j' = 0$ thus $p_\ell = 0$. For $j = 1$, according to Eq. (1), if `asin(x)` is close to $\pi/64$, then δ can be smaller than $|p_\ell| = 0x1.8469898cc518\text{p-53}$. There are only about 50 values of x near $\sin(\pi/64)$ such that δ is smaller than $|p_\ell|$; we checked that the accurate path is never called for these values. Thus the `fastsum` condition is fulfilled.

We split the analysis of Algorithm `AccuratePath` in three parts: in §IV-A we analyze the approximation of $\sqrt{1-x^2}$, in §IV-B we analyze the approximation of $\sin(\delta)$, from which in §IV-C we deduce an approximation of δ and thus of `asin(x)` using Eq. (1).

Algorithm 3 (AccuratePath)

Input: x, ϕ binary64 numbers, with $\phi \approx \text{asin}(x)$ **Output:** $p_h + p_\ell$ approximating $\text{asin}(x)$

```
1:  $s_2 = \text{o}(x^2)$ ,  $d_2 = \text{fma}(x, x, -s_2)$ 
2:  $c_{2h}, c_{2\ell} = \text{fasttwosum}(1, -s_2)$ 
3:  $c_{2\ell} = \text{o}(c_{2\ell} - d_2)$ 
4:  $c_{2h}, c_{2\ell} = \text{fasttwosum}(c_{2h}, c_{2\ell})$ 
5:  $c_h = \text{o}(\sqrt{c_{2h}})$ 
6:  $c_\ell = \text{o}(\text{o}(c_{2\ell} - \text{fma}(c_h, c_h, -c_{2h})) \cdot \text{o}(0.5/c_h))$ 
7:  $I = 0 \times 1.45\text{f}306\text{dc}9\text{c}883\text{p}+4$   $\triangleright I \approx 64/\pi$ 
8:  $j = \lfloor \text{o}(|\phi| \cdot I) \rfloor$ 
9:  $C_h + C_\ell \approx \cos(j\pi/64)$   $\triangleright$  tabulated
10:  $S_h + S_\ell \approx \sin(j\pi/64)$   $\triangleright$  tabulated
11:  $d_{sh} = \text{o}(|x| - S_h)$ ,  $d_{s\ell} = -S_\ell$ 
12:  $d_{ch} = \text{o}(c_h - C_h)$ ,  $d_{c\ell} = \text{o}(c_\ell - C_\ell)$ 
13:  $M = 0 \times 1.8\text{p}-4$ 
14:  $S_c = \text{o}(\text{fma}(S_h, d_{ch}, M) - M)$ 
15:  $d_{Sc} = \text{fma}(S_h, d_{ch}, -S_c)$ 
16:  $C_s = \text{o}(\text{fma}(C_h, d_{sh}, M) - M)$ 
17:  $d_{Cs} = \text{fma}(C_h, d_{sh}, -C_s)$ 
18:  $v = \text{o}(C_s - S_c)$ 
19:  $d_v = (C_h d_{s\ell} + C_\ell d_{sh}) - (S_h d_{c\ell} + S_\ell d_{ch}) - (d_{Sc} - d_{Cs})$ 
20:  $v, d_v = \text{fasttwosum}(v, d_v)$ 
21:  $j' = \text{sign}(x)j$ 
22:  $v_2, d_{v2} = \text{muldd}(v, d_v, v, d_v)$ 
23:  $v = \text{sign}(x)v$ ,  $d_v = \text{sign}(x)d_v$ 
24: let  $c_0 t + c_1 t^3 + \dots + c_7 t^{15}$  approximating  $\text{asin}(x)$  where
     $c_i = c_{ih} + c_{i\ell}$  for  $i \leq 4$   $\triangleright$  tabulated
25:  $r = v_2(c_5 + v_2(c_6 + v_2 c_7))$ 
26:  $f_h, f_\ell = \text{polydd}(v_2, d_{v2}, 5, r)$ 
27:  $f_h, f_\ell = \text{muldd}(v, d_v, f_h, f_\ell)$ 
28:  $p_h = \text{o}(0 \times 1.921\text{f}b54442\text{dp}-5 \cdot j')$ 
29:  $p_\ell = \text{o}(0 \times 1.8469898\text{cc}518\text{p}-53 \cdot j')$ 
30:  $p_s = \text{o}(-0 \times 1.\text{fc}8\text{f}8\text{cbb}5\text{bf}6\text{cp}-102 \cdot j')$ 
31:  $p_\ell, p_s = \text{fastsum}(f_h, f_\ell, p_\ell, p_s)$ 
32:  $p_h, p_\ell = \text{fasttwosum}(p_h, p_\ell)$ 
33:  $p_\ell = \text{o}(p_\ell + p_s)$ 
```

Algorithm 4 (polydd)

Input: $x = x_h + x_\ell$ double-double, n integer, a double r **Output:** $h + \ell$ approximating $c_0 + c_1 x^2 + \dots + (c_{n-1} + r)x^{2n-2}$

```
1:  $h, t \leftarrow \text{fasttwosum}(c_{n-1}, h, r)$ 
2:  $\ell \leftarrow \text{o}(t + c_{n-1}, \ell)$ 
3: for  $i$  from  $n - 2$  downto 0
4:    $h, \ell \leftarrow \text{muldd}(x_h, x_\ell, h, \ell)$ 
5:    $h, e \leftarrow \text{fasttwosum}(c_{i,h}, h)$ 
6:    $\ell \leftarrow \text{o}(\text{o}(\ell + c_{i,\ell}) + e)$ 
```

A. Approximation of $\sqrt{1 - x^2}$

The double-double approximation $c_h + c_\ell$ of $\sqrt{1 - x^2}$ is computed at lines 1-6. Since this is exactly the same computation as for $\text{acos}(x)$ [3, §IV-A], we simply reuse the final

Algorithm 5 (fastsum)

Input: $x_h + x_\ell$ and $y_h + y_\ell$ two double-double pairs**Output:** $s_h + s_\ell$ approximating $(x_h + x_\ell) + (y_h + y_\ell)$

```
1:  $s_h, \ell \leftarrow \text{fasttwosum}(x_h, y_h)$ 
2:  $s_\ell \leftarrow \text{o}(\text{o}(x_\ell + y_\ell) + \ell)$ 
```

Algorithm 6 (fasttwosum)

Input: a, b binary64 numbers**Output:** h, ℓ such that $h + \ell$ approximates $a + b$

```
1:  $h \leftarrow \text{o}(a + b)$ 
2:  $t \leftarrow \text{o}(h - a)$ 
3:  $\ell \leftarrow \text{o}(b - t)$ 
```

result from that analysis:

$$\left| c_h + c_\ell - \sqrt{1 - x^2} \right| < 2^{-101.924}. \quad (3)$$

B. Approximation of $\sin(\delta')$ We split the analysis in two cases: $j = 0$ and $j = 1$.

Case $j = 0$. This case may happen when $x_0 \leq |x| \leq x_1 := 0 \times 1.92155\text{f}7\text{a}3667\text{dp}-6$. We then have $C_h = 1$, and $C_\ell = S_h = S_\ell = 0$. It follows $d_{sh} = |x|$, $d_{s\ell} = 0$, Gappa proves $0.999 < c_h < 1.001$ thus $d_{ch} = c_h - C_h$ is exact, $d_{c\ell} = c_\ell$, $S_c = d_{Sc} = 0$, $C_s = \text{o}(\text{o}(|x| + M) - M)$ is the upper part of $|x|$ (rounded to a multiple of 2^{-56}), d_{Cs} is the lower part of $|x|$ such that $|x| = C_s + d_{Cs}$, $v = C_s$, $d_v = d_{Cs}$, thus $v + d_v = |x|$. We thus have exactly $\sin(\delta') = v + d_v$ in Eq. (2).

Case $j = 1$. This case may happen when $0 \times 1.92155\text{f}7\text{a}3667\text{cp}-6 \leq |x| \leq 2^{-4}$ (there is a small overlap with $j = 0$ due to the different rounding modes). A double-double approximation $v + d_v$ of $\sin(\delta')$ is computed at lines 7-20. The analysis depends on the precomputed constants C_h, C_ℓ, S_h, S_ℓ at lines 9 and 10.

Let $V = (C_h + C_\ell)|x| - (S_h + S_\ell)(c_h + c_\ell)$. Gappa 1.7.0 finds the bound (obtained for rounding toward zero):

$$|v + d_v - V| \leq 2^{-105.944}.$$

The approximations $C_h + C_\ell$ of $\cos(j\pi/64)$ and $S_h + S_\ell$ of $\sin(j\pi/64)$ have an absolute error bounded by $2^{-114.119}$. It follows:

$$\begin{aligned} |V - \sin(\delta')| &\leq |C_h + C_\ell - \cos(j\pi/64)| \cdot |x| \\ &\quad + |S_h + S_\ell| \cdot |c_h + c_\ell - \sqrt{1 - x^2}| \\ &\quad + |S_h + S_\ell - \sin(j\pi/64)| \cdot \sqrt{1 - x^2}. \end{aligned}$$

Since $|x| \leq 2^{-4}$, the first term above is bounded by $2^{-118.119}$. Since $|S_h + S_\ell| \leq 1$, and $|c_h + c_\ell - \sqrt{1 - x^2}| < 2^{-101.924}$ by Eq. (3), their product is bounded by $2^{-101.924}$. The last term is bounded by $2^{-114.119}$, thus:

$$|V - \sin(\delta')| \leq 2^{-118.119} + 2^{-101.924} + 2^{-114.119} < 2^{-101.923}.$$

Combined with the above bound on $|v + d_v - V|$, this yields:

$$|v + d_v - \sin(\delta')| < 2^{-105.944} + 2^{-101.923} < 2^{-101.836}. \quad (4)$$

Since for $j = 0$ we have $v + d_v = \sin(\delta')$, the above inequality holds in both cases ($j = 0$ or $j = 1$).

C. Approximation of δ

Lines 21-27 compute a double-double approximation $f_h + f_\ell$ of δ , from the approximation $v + d_v$ of $\sin(\delta')$. First, since $|\delta| < 0.0246$ and from Eq. (4), we have $|v + d_v| < 0.0247$, thus after line 20, we have $|v| < 0.025$ and $|d_v| \leq 2^{-58}$ after the `fasttwosum` call at line 20. Line 23 restores the correct sign, since $\delta = \text{sign}(x)\delta'$ thus $\text{asin}(\delta) = \text{sign}(x)\text{asin}(\delta')$.

We analyze separately the cases $j = 0$ and $j = 1$, since for $j = 0$, we need a bound on the relative error, whereas for $j = 1$, a bound on the absolute error is enough, as in the `acos(x)` case.

Case $j = 1$. This case is similar to that of `acos(x)` [3]. Let us denote $q(t) = c_0t + \dots + c_7t^{15}$ the polynomial approximation of $\text{asin}(t)$ used in Algorithm `AccuratePath`. After the `muldd` call at line 27, Gappa proves for $t = v + d_v$ after line 23:

$$|f_h + f_\ell - q(t)| < 2^{-106.759}.$$

We now have:

$$|f_h + f_\ell - \delta| \leq |f_h + f_\ell - q(t)| + |q(t) - \text{asin}(t)| + |\text{asin}(t) - \delta|.$$

The first term in the right-hand side is bounded by $2^{-106.759}$ from just above. Sollya bounds the second term over $[-0.025, 0.025]$ by $|q(t) - \text{asin}(t)| < 2^{-108.871}$. Since $|t - \sin(\delta)| < 2^{-101.836}$ by Eq. (4), the third term is bounded by $2^{-101.836}$ multiplied by a bound for the derivative of $\text{asin}(t)$ over $[-0.025, 0.025]$. This derivative is maximal at $t = \pm 0.025$, where it is less than 1.0004, this yields a bound of $2^{-101.836} \cdot 1.0004 < 2^{-101.835}$ for the third term. Combining, we get:

$$|f_h + f_\ell - \delta| \leq 2^{-106.759} + 2^{-108.871} + 2^{-101.835} < 2^{-101.777}.$$

At lines 28-30, we multiply the integer j' , by a triple-word approximation of $\pi/64$, with error less than 2^{-155} . Since $j' = \pm 1$, the multiplication by j' is exact. The error from the approximation of $\pi/64$ is bounded by 2^{-155} , thus after line 30:

$$|p_h + p_\ell + p_s - j' \frac{\pi}{64}| < 2^{-155}.$$

Gappa proves that the rounding error of calling Algorithm `fastsum` is bounded by 2^{-107} , thus after line 31:

$$\begin{aligned} & |p_h + p_\ell + p_s - \text{asin}(x)| \\ & < 2^{-101.777} + 2^{-155} + 2^{-107} < 2^{-101.738}. \end{aligned}$$

On line 32, Gappa proves that the precondition $|p_\ell| \leq |p_h|$ of `fasttwosum` is satisfied, and the rounding error due to `fasttwosum` is bounded by 2^{-109} . Finally, on line 33, Gappa proves the rounding error is bounded by 2^{-107} . We thus have for the final values p_h and p_ℓ :

$$|p_h + p_\ell - \text{asin}(x)| \leq 2^{-101.738} + 2^{-109} + 2^{-107} < 2^{-101.691}. \quad (5)$$

Case $j = 0$. In this case we have to analyze the relative error, because the term $j\pi/64$ vanishes in Eq. (1). Remember that

$|x| \leq x_1 = 0 \times 1.92155f7a3667dp-6$. Sollya proves for $|x| \leq x_1$:

$$|q(x) - \text{asin}(x)| < 2^{-106.995} |\text{asin}(x)|. \quad (6)$$

In Algorithm `AccuratePath`, since $j' = 0$, we have $p_h = p_\ell = p_s = 0$ at lines 28-30, thus at line 31 $p_\ell = f_h$ and $p_s = f_\ell$, at line 32 $p_h = f_h$ and $p_\ell = 0$, and at line 33 $p_\ell = f_\ell$. Thus the final values of p_h, p_ℓ are the values of f_h, f_ℓ after line 27.

Remember that $v + d_v = |x|$ after line 19, thus $v = |x|$ and $d_v = 0$ after line 20, and $v = x$ and $d_v = 0$ after line 23. It follows $x_0 \leq |v| \leq x_1$. Since after line 20, we have $|d_v| \leq \text{ulp}(v) \leq 2^{-52}|v|$, using $|d_v| \leq 2^{-52}|v|$, Gappa proves:

$$|f_h + f_\ell - q(x)| < 2^{-100.356} x.$$

Together with Eq. (6), using $|x| \leq |\text{asin}(x)|$ for $|x| \leq x_1$, and $p_h = f_h, p_\ell = f_\ell$, this yields:

$$|p_h + p_\ell - \text{asin}(x)| < 2^{-100.341} |\text{asin}(x)|. \quad (7)$$

Theorem 1: For any binary64 number x , $x_0 \leq |x| \leq 2^{-4}$, such that $\text{asin}(x)$ has less than 42 identical bits after the round bit, if $p_h + p_\ell$ is the double-word approximation of $\text{asin}(x)$ computed by Algorithm `AccuratePath`, then $\circ(p_h + p_\ell)$ is the correct rounding of $\text{asin}(x)$.

Proof: Let $y = \text{asin}(x)$. If y has less than 42 identical bits after the round bit, then y is at distance at least $2^{-43}\text{ulp}(y)$ from a rounding boundary z : $|y - z| \geq 2^{-43}\text{ulp}(y)$. Since $\text{ulp}(y) > 2^{-53}|y|$, this yields $|y - z| \geq 2^{-96}|y|$.

First assume $j = 1$, thus $x'_1 \leq |x| \leq 2^{-4}$ with $x'_1 = 0 \times 1.92155f7a3667cp-6$. Since for $|x| \geq x'_1$, $\text{asin}(x) \geq 2^{-5.349}$, we get $|y - z| \geq 2^{-101.349}$. Thus Eq. (5) shows that $|p_h + p_\ell - \text{asin}(x)| < |\text{asin}(x) - z|$, i.e., $p_h + p_\ell$ is closer from $\text{asin}(x)$ than any rounding boundary. It follows $\circ(p_h + p_\ell) = \circ(\text{asin}(x))$.

Now assume $j = 0$, thus $|x| \leq x_1 = 0 \times 1.92155f7a3667dp-6$. In this case, $|y - z| \geq 2^{-96}|y|$ together with Eq. (7) shows that $|p_h + p_\ell - \text{asin}(x)| < |\text{asin}(x) - z|$ too. ■

The CORE-MATH test suite `asin.wc` contains all hard-to-round inputs with at least 42 identical bits after the round bit for $2^{-6} \leq x < 2^{-4}$. Since all these hard-to-round inputs are correctly rounded, we conclude from Theorem 1 that the accurate path is correct in the $j = 1$ case. For $x_0 \leq x \leq x_1$, which corresponds to $j = 0$, it contains all hard-to-round inputs with at least 43 identical bits after the round bit. From the proof of Theorem 1, it suffices to check hard-to-round inputs with at least 46 identical bits after the round bit, since if less than 46 identical bits, we would have $|y - z| \geq 2^{-100}|y|$, and Eq. (7) shows $|p_h + p_\ell - \text{asin}(x)| < |\text{asin}(x) - z|$. Since these hard-to-round inputs are correctly rounded, we conclude that the accurate path is correct in the $j = 0$ case too. Thus it is correct for any input x , $x_0 \leq |x| \leq 2^{-4}$. This completes the correctness proof of the arc-sine function.

REFERENCES

- [1] BOLDO, S., GRAILLAT, S., AND MULLER, J. On the robustness of the 2sum and fast2sum algorithms. *ACM Trans. Math. Softw.* 44, 1 (2017), 4:1–4:14.
- [2] MULLER, J.-M., BRUNIE, N., DE DINECHIN, F., JEANNEROD, C.-P., JOLDES, M., LEFÈVRE, V., MELQUIOND, G., REVOL, N., AND TORRES, S. *Handbook of Floating-point Arithmetic (2nd edition)*. Birkhäuser Basel, July 2018.
- [3] ZIMMERMANN, P. The CORE-MATH acos is correctly rounded, 2026. <https://core-math.gitlabpages.inria.fr/acos.pdf>.
- [4] ZIMMERMANN, P. The GNU libc atanh is correctly rounded. In *ARITH 2026 - 33rd IEEE International Symposium on Computer Arithmetic* (Fulda, Germany, June 2026).