**算法1（FloatDiv: Gold3算法）**：

输入：32位单精度浮点数a，b

输出：32位单精度除法c=a/b的结果

Phase1: Prepare for TS6 Algorithm

1: A = {1’b1, a[22:0], 4’b0}

2: B = {1’b1, b[22:0], 4’b0}

3: U = {1’b0, LUT(B[22:15]), 4’b0)

Phase2: Goldschmidt Algorithm Steps

1: yi = B \* U

2: xi = A \* U

3: for i from 0 to 2

4: ri = 2 - yi

5: yi = yi \* ri

6: xi = xi \* ri

7: end for

8: T = xi

Phase3: Ccorrecting errors

1: if (T[27] == 0)

2: C = T[26:2]

3: else

4: C = T[27:1]

5: if (A > B)

6: A1 = A << 23

7: else

8: A1 = A << 24

9: D1 = B \* C - A1

10: D2 = A1 - B \* C

11: if (B < 2D1)

12: C\_new = C - 1

13: else if (B < 2D2)

14: C\_new = C + 1

15: else

16: C\_new = C

Phase4: Normalize to IEEE754 standard

1: if (T8[27] == 0)

2: Exp = a[30:23] - b[30:23] - 1 + 127

3: else

4: Exp = a[30:23] - b[30:23] + 127

5: Sig = a[31] ^ b[31]

6: c = {Sig, Exp, C\_new}

算法参数：

乘法器位宽n=28

LUT输入位宽为m-1=6

注意：中间乘法结果直接截断到n位，不做4舍5入操作。

LUT定义如下：

def LUT(X):

x = {9’h07f, X, 15’h7fff}

y = Float(1)/Float(x)

Y = y[22:0]

return Y

**算法2（FloatSqrtRoot: NR2算法）**：

输入：32位单精度浮点数b

输出：32位单精度浮点平方根c=sqrt(b)的结果

Phase1: Extend Tail, Exp, LUT

1: if b[23] == 1:

2: Exp\_c = (b[30:23] - 127) / 2

3: B = {1’b1, b[22:0], 4’b0}

4: else:

5: Exp\_c =(b[30:23] - 128) / 2

6: B = {2’b01, b[22:0], 3’b0}

7: end if

8: Z={2’b01,LUT\_Z(b[23:16]), 3’b0}

9: Z3={LUT\_Z3(b[23:16]), 3’b0}

Phase2: Newton-Rhapson Method

1: T1 = 3 \* Z

2: T2 = B \* Z

3: T3 = (T1 - T2) / 2

4: T4 = B \* T3

5: T5 = T3 \* T4

6: T6 = 3 - T5

7: T7 = T6 \* T5 / 2

Phase3: Correcting Errors

1: R = T7[27:3]

2: X = B << 23

3: if (X > (R \* R + R):

4: R = R + 1

5: else if (X < (R \* R - R)):

6: R = R - 1

7: else:

8: R = R

Phase4: Normalize

1: expc = Exp\_c + 127

2: c = {1’b0, expc, R[22:0]}

算法参数：

乘法器位宽n=28

LUT输入位宽为m-1=8

注意：中间乘法结果直接截断到n位，不做4舍5入操作。

LUT定义如下：

def LUT\_Z(X):

x = {9’h07f, X, 15’h7fff}

y = Float(1)/sqrt(Float(x))

Y = y[22:0]

return Y

def LUT\_Z3(X):

Z = LUT\_Z(X)

Y = Z \* Z

W = Y \* Z

return W