Binary Logarithm Calculator

Mainly used methods:

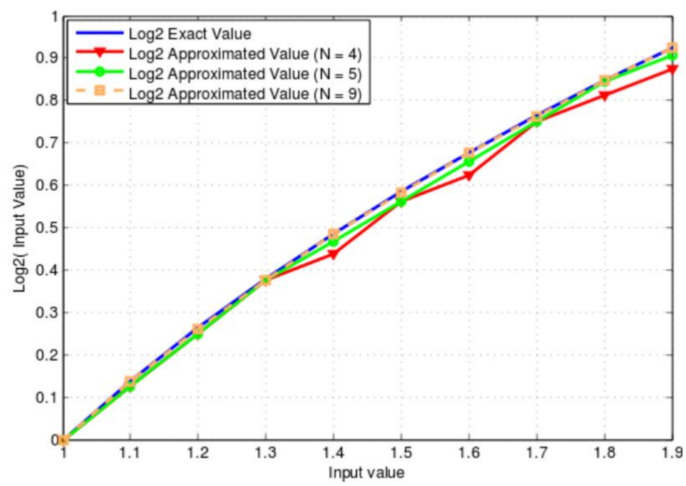
1. Look up table (LUT)
2. CORDIC => atanh => log2n = atanh(n – 1 / n + 1)
3. Iterative method

Chosen method: Iterative

- No range reduction
- Around the same speed as CORDIC w/ fixed point conversions.

$$X = m2^e$$

$$log_2(X) = \overbrace{\underbrace{e}_{IntegerPart} + \underbrace{log_2(m)}_{FractionPart}}^{log_2(X)}$$



$$\log_2 x = n + \log_2 y \quad \text{where } y = 2^{-n}x \text{ and } y \in [1, 2)$$

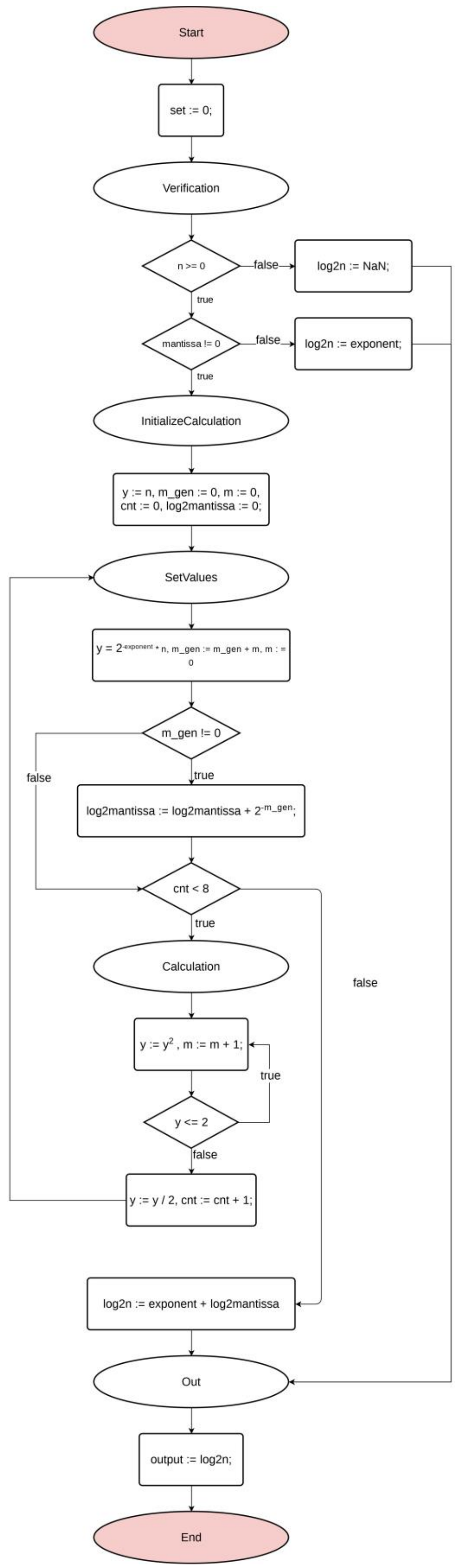
For normalized floating-point numbers, the integer part is given by the floating-point exponent,[57] and for integers it can be determined by performing a count leading zeros operation.[58]

The fractional part of the result is $\log_2 y$ and can be computed iteratively, using only elementary multiplication and division.[56] The algorithm for computing the fractional part can be described in pseudocode as follows:

1. Start with a real number y in the half-open interval $[1, 2)$. If $y = 1$, then the algorithm is done, and the fractional part is zero.
2. Otherwise, square y repeatedly until the result z lies in the interval $[2, 4)$. Let m be the number of squarings needed. That is, $z = y^{2^m}$ with m chosen such that z is in $[2, 4)$.
3. Taking the logarithm of both sides and doing some algebra:
$$\begin{aligned}\log_2 z &= 2^m \log_2 y \\ \log_2 y &= \frac{\log_2 z}{2^m} \\ &= \frac{1 + \log_2(z/2)}{2^m} \\ &= 2^{-m} + 2^{-m}\log_2(z/2).\end{aligned}$$
4. Once again $z/2$ is a real number in the interval $[1, 2)$. Return to step 1 and compute the binary logarithm of $z/2$ using the same method.

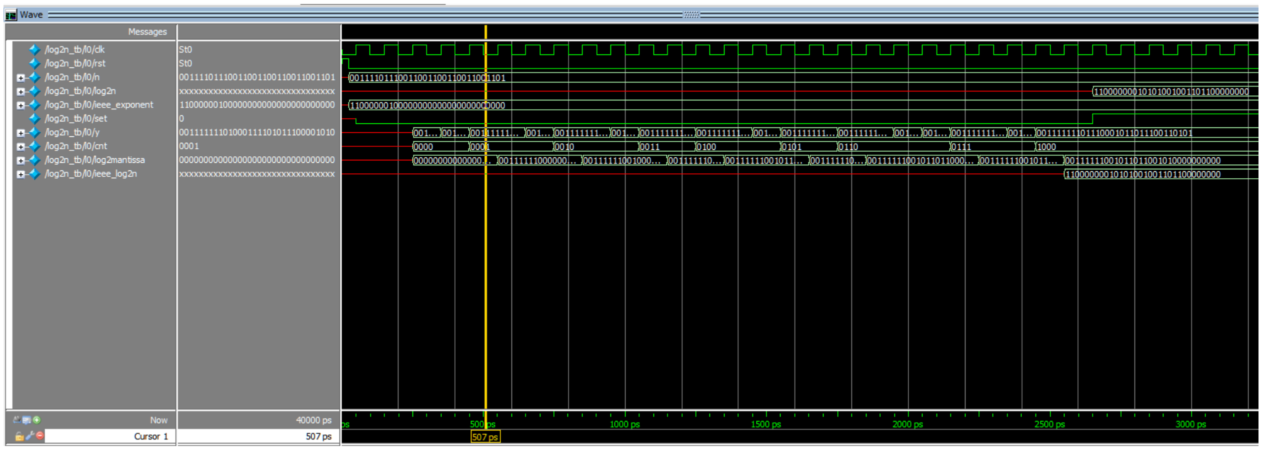
The result of this is expressed by the following recursive formulas, in which m_i is the number of squarings required in the i-th iteration of the algorithm:

$$\begin{aligned}\log_2 x &= n + 2^{-m_1}\left(1 + 2^{-m_2}\left(1 + 2^{-m_3}\left(1 + \cdots\right)\right)\right) \\ &= n + 2^{-m_1} + 2^{-m_1-m_2} + 2^{-m_1-m_2-m_3} + \cdots\end{aligned}$$

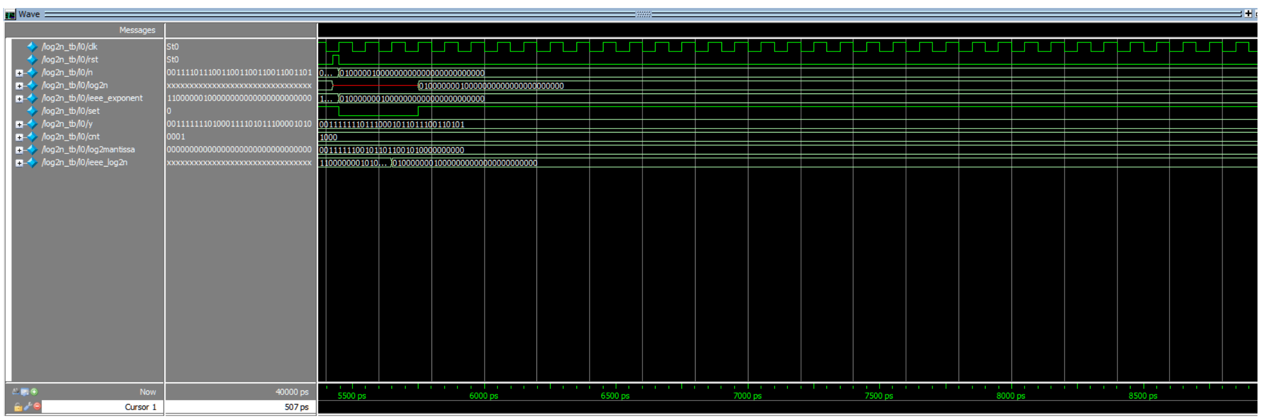


Start
set := 0;
Verification
n >= 0
false
log2n := NaN;
true
mantissa != 0
false
log2n := exponent;
true
InitializeCalculation
y := n, m_gen := 0, m := 0,
cnt := 0, log2mantissa := 0;
SetValues
y = 2-exponent * n, m_gen := m_gen + m, m : =
0
m_gen != 0
false
true
log2mantissa := log2mantissa + 2-m_gen;
cnt < 8
true
false
Calculation
y := y2 , m := m + 1;
true
y <= 2
false
y := y / 2, cnt := cnt + 1;
log2n := exponent + log2mantissa
Out
output := log2n;
End

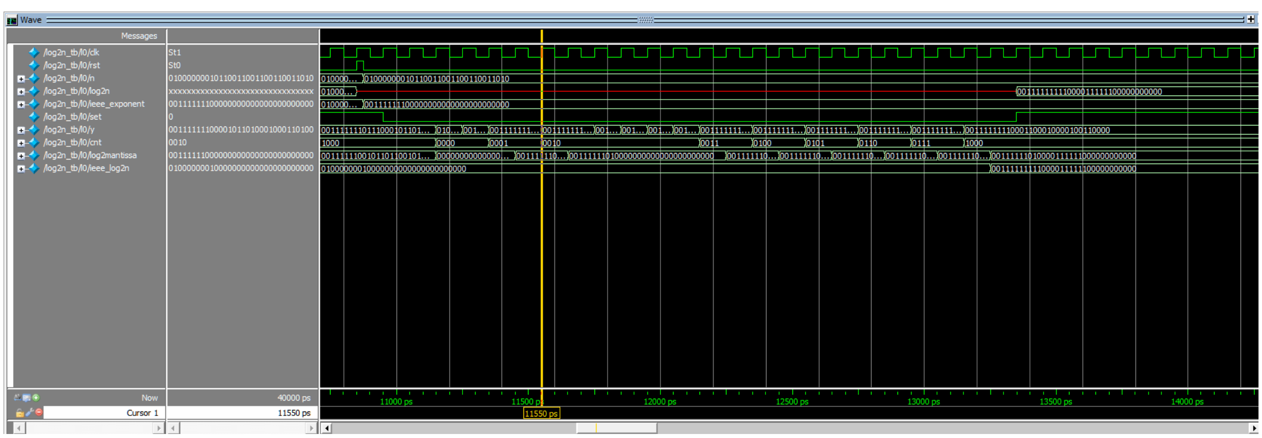
input:0.100000
output:-3.321960



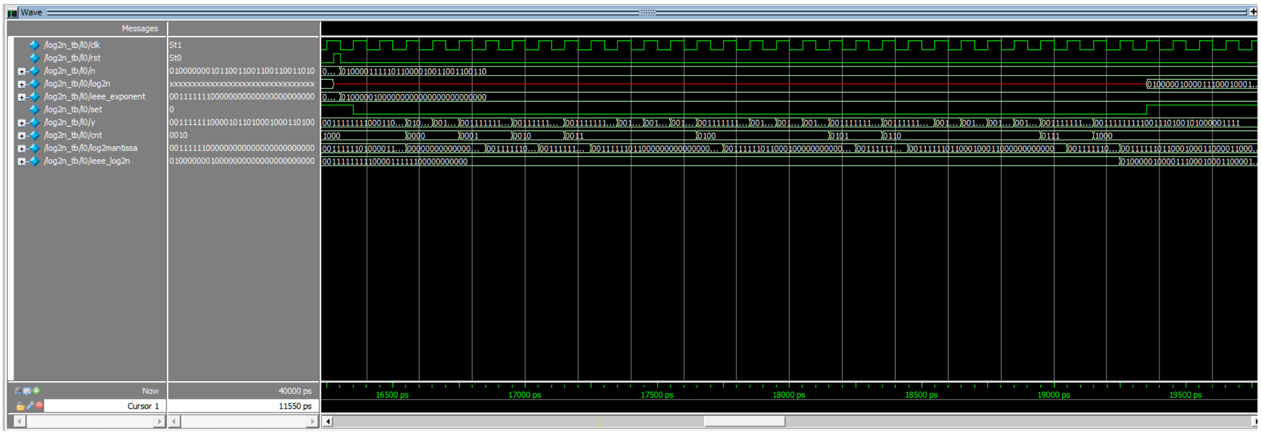
input:8.000000
output:3.000000



input:3.400000
output:1.765381



```
# input:472.299988
# output:8.883556
```



```
# input:0.000000
# output:1.#QNAN0
#
# input:0.000000
# output:1.#QNAN0
#
# input:-0.100000
# output:1.#QNAN0
#
```