

OPL3 Math

October 12, 2013

This document tries to incorporate known facts about the inner OPL3 workings to help software developers design and/or improve emulators.

Contents

1	Conventions	1
2	Sine wave	2
2.1	Volume units	2
2.2	Logarithmic-exponential transformation	2

1 Conventions

As the OPL3 uses 1's complement to implement subtractions and negations, we will define special subtraction and unary minus operators for this purpose:

$$\begin{aligned}x \ominus y &:= x - y - 1 \\ \ominus x &:= -x - 1\end{aligned}$$

As $x \ominus y$ can be written as $x + (0 \ominus y)$, these operations can easy implemented using either the NOT operation if the bit length of y is a multiple of 8, or by using the XOR operation with $2^n - 1$ if y is of bit length n .

Furthermore, we will define a special notation for hexadecimal numbers. They will be printed in typewriter font, using a \$ as suffix, e. g. $x \ominus 1\mathbf{F}_{\$}$.

We will also define $\neg x$ to be the NOT operation on all bits of x , $x \otimes y$ to be x XOR y on all bits on x and y , and $x \wedge y$ to be the x AND y , so that it looks more like math.

To denote the bit length n of a variable x , we will write $x_{|n|}$, and to extract a single bit m of a variable y , we will write $y_{|(m)|} \in \{0; 1\}$. We will also define here for convenience that $x_{|n|}$ is equal to $x \wedge (2^n - 1) = x \bmod 2^n$, so that x can be an expression.

This all incorporates to this:

$$(x_{|n|} \ominus y_{|n|})_{|n|} = (x + \neg y)_{|n|} = (x + (y \otimes (2^n - 1)))_{|n|}$$

Please note that—if not specified otherwise—all arithmetics in this document will be in \mathbb{N} and fractional results of expressions will be rounded towards zero. Unfortunately, this makes arithmetics sensitive to order of execution; thus let's define the order of execution to be from left to right and from top to bottom if ambiguous, e.g., $a \cdot \frac{b \cdot c}{d}$ will be calculated (from left to right) as $a \cdot ((b \cdot c)/d)$.

2 Sine wave

2.1 Volume units

According to the “OPLx decapsulated” document by Matthew GAMBRELL and Olli NIEMITALO¹, the OPL3 contains a ROM with the first quarter of a sine wave. The sine wave is saved as a logarithmic/exponential table combination with 256 entries each, so that the full round-trip of 2π “in the real world” is 1024 in OPL3 units.

The idea behind this is to write $\sin(x)$ as $2^{\log_2(\sin(x))}$. Now, to multiply the amplitude with some value y , we can write

$$\begin{aligned}\sin(x) \cdot y &= 2^{\log_2(\sin(x)) \cdot y} \\ &= 2^{\log_2(\sin(x)) + \log_2(y)}\end{aligned}\tag{1}$$

. Of course, this looks cumbersome, but it allows us to substitute an expensive multiplication with a simple addition.

Furthermore, we are talking about an *audio* chip here. The Bel (named after Alexander Graham BELL), is a logarithmic unit. It may sound more familiar to you when we scale it down by 10; then, we got the *Decibel*. The nature of this unit is somehow like NEWTON's law of gravity, though in some reverse manner. If you compare two values a and b (e.g., how loud two sounds are), you could say that a is two times as loud as b by saying $a = 10 \text{ dB} \cdot b$, though you say nothing about how loud a and b are exactly. The Bel is just a scaling factor and can be defined as $x \text{ B} = 2^x$, or $x \text{ dB} = 2^{x/10}$.

Now let's have a look at formula (1) again. If we express y in z Bels, we get a less cumbersome formula:

$$\sin(x) \cdot y = 2^{\log_2(\sin(x)) + z}\tag{2}$$

Now we have to deal with the $2^{\log_2(\sin(x))}$ part.

2.2 Logarithmic-exponential transformation

The two formulas now given are the only place in this document where the arithmetics are done within \mathbb{R} , and $[\cdot]$ here means “round to the closest value within \mathbb{N} ”:

$$\begin{aligned}\Phi^*(x) &:= \left[\left(2^{x/256} - 1 \right) \cdot 2^{10} \right] \\ \varphi^*(x) &:= \left[-256 \cdot \log_2 \left(\sin \left((x + 0.5) \cdot \frac{\pi}{2 \cdot 256} \right) \right) \right] \\ x &\in \{0 \leq y \leq 255 \mid y \in \mathbb{N}\}\end{aligned}$$

¹https://docs.google.com/document/d/18IGx18NQY_Q1PJVZ-bHywao9bhsDoAqoIn1rIm42nwo/edit

Here, Φ^* is the exponential function, and φ^* is the logarithmic function (if you haven't guessed that yet). The graphs in figure 2 on the following page and figure 1 are plots of these functions.

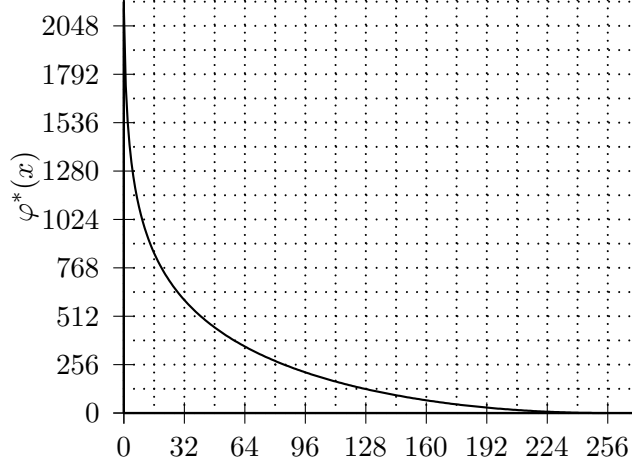


Figure 1: Logarithmic sine table

If you look at it, you will see that $\Phi^*(x) \in [0; 1018]$ and $\varphi^*(x) \in [0; 2137]$, so we cannot simply write $2^{10} + \Phi^*(255 - \varphi^*(x))$, as this will only be defined for $x \leq 83$ (2^{10} is the hidden bit). To get a full sine wave, x has to go as high as $2^{10} - 1$. For this, we have to refine these functions.

First, the logarithmic function, which we will define for arbitrary $x \geq 0$:

$$\varphi(x) := \begin{cases} \varphi^*(x) & \text{if } 0 \leq x_{|10|} < 100_{\S} \\ \varphi^*(255 - x_{|8|}) & \text{if } 100_{\S} \leq x_{|10|} < 200_{\S} \\ \varphi^*(x_{|8|}) + 2^{15} & \text{if } 200_{\S} \leq x_{|10|} < 300_{\S} \\ \varphi^*(255 - x_{|8|}) + 2^{15} & \text{if } 300_{\S} \leq x_{|10|} \end{cases}$$

Please note the offset of 2^{15} in the latter two cases—this is used as a sign bit, which comes in handy now that we refine the exponential function. Because $\varphi^*(x)$ is always $\leq 2137 < 2^{15}$, this can be added by OR-ing the 15th bit.

$$\Phi(x) := \begin{cases} 0 + (2^{10} + \varphi(255 - x_{|8|})) \cdot 2/2^{(x/256)} & \text{if } x_{|(15)|} = 0 \\ 0 \ominus (2^{10} + \varphi(255 - x_{|8|})) \cdot 2/2^{(x_{|14|}/256)} & \text{if } x_{|(15)|} = 1 \end{cases}$$

To get a nice full sine wave, we can finally write $\Phi(\varphi(x))$. The amplitude of the resulting sine wave is $\Phi(\varphi(255)) = \Phi(0) = 2^{10}$.

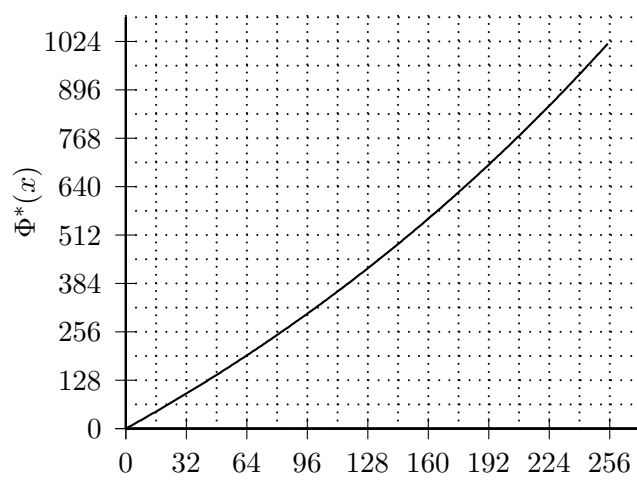


Figure 2: Exponential sine table