Pre-Scaling to Avoid Overflow and Maintain Maximum Precision in Fixed-Point Multiplication

D.R. Brown III

November 5, 2012

1 Problem Setup and Notation

In almost all real-time DSP algorithms, we need to perform multiplication and addition. Those operations are straightforward when done in floating point, but are trickier when done in fixed point. This note discusses the special considerations that must be taken when performing fixed-point multiplication.

Consider the real numbers x and y. These numbers are quantized to a fixed-point representation as x_q and y_q where

$$x_q = x + \epsilon_x \tag{1}$$

$$y_q = y + \epsilon_y \tag{2}$$

where ϵ_x and ϵ_y are the quantization errors in x_q and y_q, respectively. The quantized variables x_q and y_q are assumed to have N_x and N_y total bits as well as M_x and M_y fractional bits, respectively.

We wish to compute the product

$$z_q = x_q * y_q \tag{3}$$

as accurately as possible given the constraint that z_q is a fixed-point variable with N_z total bits. In the next section, we determine the maximum number of fractional bits M_z that we can have for z_q while avoiding overflow/saturation and we also determine the optimal amount of pre-scaling on the multiplicands x_q and y_q to maintain maximum precision.

2 Analysis

As discussed in lecture, to avoid overflow, we require the largest positive value of z_q to be at least as large as the largest possible positive product of x_q and y_q, i.e.,

$$\frac{2^{N_z-1}-1}{2^{M_z}} \geq \left(\frac{-2^{N_x-1}}{2^{M_x}}\right)\left(\frac{-2^{N_y-1}}{2^{M_y}}\right). \tag{4}$$

In order to maintain full precision, we also require

$$M_z \geq M_x + M_y. \tag{5}$$

If both conditions can be satisfied, then you should set $M_z = M_x + M_y$ and you are done. There is no need to determine optimum pre-scaling factors.

In many cases, both of these conditions can't be satisfied since setting $M_z = M_x + M_y$ causes the first condition to simplify to $N_z \geq N_x + N_y - 1$. For example, if x_q, y_q, and z_q are all short variables ($N_x = N_y = N_z = 16$), then both conditions can't be satisfied. When both conditions can't be satisfied, it is much better to throw away precision, i.e. violate the second condition in (5), than have overflow. This means we need to determine the best value of M_z (which will not satisfy the second condition) and then determine the best pre-scaling factors for x_q and y_q to maintain as much precision as possible in the product. The following sections describe how to do this.

2.1 Determining the Largest M_z That Avoids Overflow

At this point, we know we need to throw away precision, but we don't want to throw away too much precision. Our goal is to find the largest value of M_z such that the product $x_q * y_q$ will not overflow in the container z_q.

To do this, we compute the largest possible positive product of x_q and y_q, i.e.,

$$\frac{2^{N_z-1}-1}{2^{M_z}} \geq \left(\frac{-2^{N_x-1}}{2^{M_x}}\right)\left(\frac{-2^{N_y-1}}{2^{M_y}}\right). \tag{6}$$

We are given M_x, M_y, N_x, N_y, and N_z, and we need to solve for M_z. We can rearrange this last equation to write

$$\frac{2^{N_z-1}-1}{\left(\frac{-2^{N_x-1}}{2^{M_x}}\right)\left(\frac{-2^{N_y-1}}{2^{M_y}}\right)} \geq 2^{M_z} \tag{7}$$

and then assuming that $2^{N_z-1} \gg 1$, we can simplify this last expression to

$$2^{N_z-N_x-N_y+M_x+M_y+1} \geq 2^{M_z} \tag{8}$$

which implies an upper bound on M_z as

$$\boxed{M_z \leq N_z - N_x - N_y + M_x + M_y + 1.} \tag{9}$$

As an example, suppose $M_x = 11$, $M_y = 13$, and $N_x = N_y = N_z = 16$. Then we should select $M_z = 9$ to maintain maximum precision while avoiding overflow. This means that the LSB of the product has a value of $2^{-M_z} = 2^{-9}$ which is significantly less precise than the full-precision product which would have an LSB with a value of $2^{-(M_x+M_y)} = 2^{-24}$. Nevertheless, this is the best we can do while avoiding overflow.

Note that M_z might be a negative number. This is ok; a negative number of fractional bits just means the implicit decimal point is moved to the right.

For example, suppose $M_x = 3$, $M_y = 5$, and $N_x = N_y = N_z = 16$. Then we should select $M_z = -8$. This just means that the LSB of product has a value of $2^{-M_z} = 256$.

2.2 Determining Optimum Pre-Scaling Factors P_x and P_y

Once we've determined a value for M_z according to the procedure in the previous section, if it violates the condition in (5), we know that we will need to throw away some precision before computing the product. A bad way to do this is

```
zq = (xq * yq) >> P;
```

because overflow occurs before we throw away precision. Instead, we throw away precision *before* the product as

```
zq = (xq >> Px)*(yq >> Py);
```

where P_x and P_y are the pre-scaling factors that we need to determine.

First off, how many *total bits* of precision do we need to throw away? The full-precision product has $M_x + M_y$ fractional bits, but we've determined that we can only have M_z fractional bits to avoid overflow. So we must throw away a total of $P = M_x + M_y - M_z$ fractional bits. Hence, we require $P_x + P_y = P$.

Now, we will determine how to best allocate those bits between P_x and P_y. We can write the fixed-point product as

$$\begin{aligned} z_q &= x_q * y_q && (10)\\ &= (x + \epsilon_x)(y + \epsilon_y) && (11)\\ &= \underbrace{xy}_{\text{true answer}} + \underbrace{x\epsilon_y + y\epsilon_x + \epsilon_x\epsilon_y}_{\text{error}} && (12) \end{aligned}$$

Recall that x and y are the unquantized multiplicands. Since we quantized these values with N_x and N_y total bits and M_x and M_y fractional bits, respectively, we know these numbers are on the order of

$$x \sim 2^{N_x - 1 - M_x} \quad (13)$$

$$y \sim 2^{N_y - 1 - M_y}. \quad (14)$$

We also know the quantization errors are on the order of an LSB value *after pre-scaling*. Since the number of fractional bits in x_q and y_q after pre-scaling is $M_x - P_x$ and $M_y - P_y$, respectively, we can say

$$\epsilon_x \sim 2^{-(M_x - P_x)} \quad (15)$$

$$\epsilon_y \sim 2^{-(M_y - P_y)}. \quad (16)$$

Hence, the total error $\epsilon = x\epsilon_y + y\epsilon_x + \epsilon_x\epsilon_y$ in the product is on the order of

$$\epsilon \sim 2^{N_x - 1 - M_x} 2^{-(M_y - P_y)} + 2^{N_y - 1 - M_y} 2^{-(M_x - P_x)} + 2^{-(M_x - P_x)} 2^{-(M_y - P_y)}. \quad (17)$$

Note that the third term in the sum on the right hand side is much smaller than the first two, since the product of two quantization errors should be much smaller than the product of a quantization error with an unquantized value. We will ignore that third term in the subsequent analysis since it is insignificant. Simplifying the remaining two terms, we can say the total quantization error of the product is on the order of

$$\epsilon \sim \left(2^{-(M_x+M_y+1)}\right)\left(2^{N_x+P_y}+2^{N_y+P_x}\right). \tag{18}$$

We want to find P_x and P_y to minimize this quantity subject to the constraint that $P_x+P_y=P$. Note that the first term in (18) is a positive constant, i.e. not a function of P_x or P_y. This means it doesn't affect the minimization. Using this fact, and substituting $P_y=P-P_x$ into (18), we can write the minimization problem as

$$\boxed{P_x = \arg\min_{k=0,\ldots,P}\left(2^{N_x+P-k}+2^{N_y+k}\right)} \tag{19}$$

where arg means "give me the value of k that achieves the minimum". Note that P_y is then straightforwardly computed as $P_y=P-P_x$. Also note that the optimum values for P_x and P_y don't depend on the number of fractional bits, except indirectly through P since $P=M_x+M_y-M_z$.

The easiest way to find this minimum is to just plot it in MATLAB for given values of N_x, N_y, and P using the `stem` command. As an example, suppose $M_x=11$, $M_y=13$, and $N_x=N_y=N_z=16$. We know from the previous section that we should set $M_z=9$ to avoid overflow. This means that we must throw away $P=M_x+M_y-M_z=15$ fractional bits to avoid overflow. If we plot (19) for $k=0,\ldots,15$ with the `stem` command in MATLAB, we get the result shown in Figure 1. These results imply that $P_x=7$ or $P_x=8$ minimizes the error in the fixed-point product. Hence, either

```
zq = (xq >> 7)*(yq >> 8);
```

or

```
zq = (xq >> 8)*(yq >> 7);
```

would give the best results for this example.

3 Conclusion

Fixed-point processing requires special care to maintain as much precision as possible while avoiding overflow. Throwing away precision is undesirable, but it isn't as catastrophic as overflow. Our goal is to maintain as much precision as possible for as long as possible while avoiding overflow. This note explains how to do this when performing fixed-point multiplication.

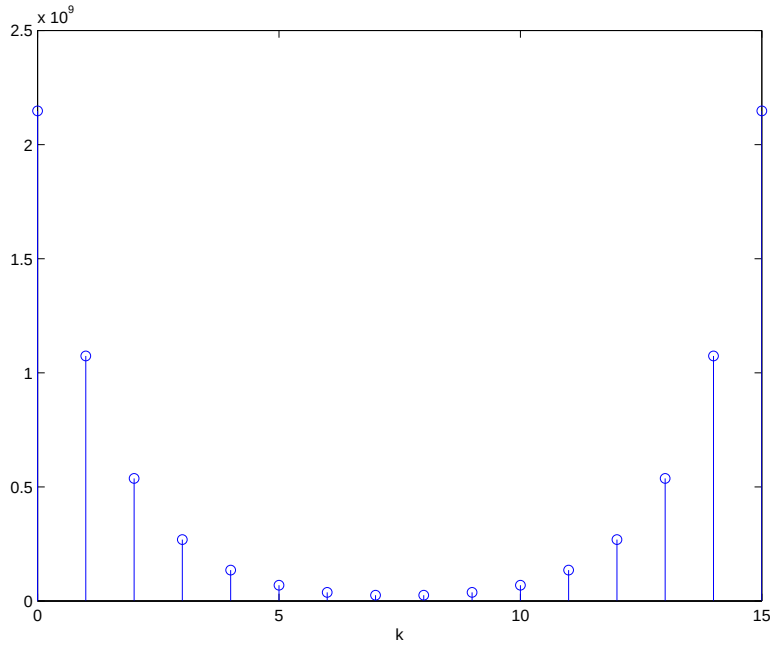

Figure 1: Total quantization error in product as a function of k for the example with $M_x = 11$, $M_y = 13$, $N_x = N_y = N_z = 16$ and $M_z = 9$.