**ATA Memo No. 40**

# Processing Architectures For Complex Gain Tracking

Larry R. D'Addario
2001 October 25

## 1. Introduction

In the baseline design of the IF Processor [1], each beam is provided with separate and independent phase tracking for each antenna. The phase is adjusted for every data sample at a programmable rate, and the rate may be updated at least 1000 times per second by a local microprocessor. This is accomplished by a DDS-style phase generator driving sine and cosine lookup tables to generate a complex gain value, which is then multiplied by the current complex data sample. In this design, the magnitude of the complex gain can be varied by re-loading the lookup table so that all entries are scaled by the desired magnitude. Provisions are made for doing this from the local microprocessor, but it requires that the magnitude be updated far more slowly than the phase. The arrangement is more than sufficient for accurate tracking of a target source, even if the source is a low earth orbit satellite and even in the case of an expanded (3 km) array. The gain magnitudes can be different among the antennas, thereby allowing some beam shaping, but it was expected that they can remain fixed for a given observation.

It has recently been suggested [2] that more general tracking capability is desirable, where the full complex gain (phase and magnitude) can be varied rapidly. It would then be possible, using known algorithms, to generate an array beam with its peak tracking a target source and simultaneously with a null tracking an interfering source. It may also be possible to accomplish this under the constraint that only the phases are varied (provided that the number of antennas is large), but feasible and robust algorithms for this are not presently known.

In this memo, I examine alternative designs that would provide for rapid phase and magnitude tracking of the complex gains.

## 2. Update Rates

As reported in [3], the maximum phase rate for sidereal sources in the planned ATA (0.7 km extent) is 0.951 Hz, and for a satellite in a 350 km circular orbit it is 287 Hz. For a 3 km array, these become 4.075 and 1233 Hz, respectively. Using the largest of these values, and attempting to keep the complex gain always within 1% of its ideal value, we find that the phase must be updated every 1.29 microseconds (775 kHz rate). This is far longer than the sample period, ∼10 nsec (100 MHz). However, in the baseline design each successive sample is from a different subband, so that the phase must be updated for every sample.

If we are not simply tracking one target, but also are tracking (in nulls of the beam) one or more interferers, the necessary update rate may be different. To estimate this rate, consider that one way to create a null is first to compute the gains for a uniformly-weighted beam toward the target and for another beam toward the null. Let the gain of the first beam in the direction of the second be $g_x$; if the complex gains for the second beam are all weighted by $g_x$ and subtracted by those for the first beam, the resulting complex gains will produce a beam with the desired peak and null (although it may not be the optimum such beam). This method can be extended to produce a beam with several nulls. The final complex gain for each antenna is thus a linear combination of the complex gains that would be needed for each of the constituent beams. The time derivative of each gain, and thus its required update rate for a given accuracy, is the same linear combination of those for the constituents; if the constituent beam for the fastest moving source has high weight, then it will tend to dominate the derivative. Normally we expect that the target's beam has the largest weight; but usually it is a sidereal source, moving slowly. Nevertheless, for a worst-case estimate, let us take a 350 km satellite as the fastest object and assume that its beam has the largest weight. We then find that the required update rate for 1% accuracy is nearly the same as for tracking such an object as the target, namely about 775 kHz for an expanded array or 180 kHz for the initial array.

The 1% accuracy critereon is somewhat arbitrary. It corresponds to a net relative accuracy in the phased array voltage gain of $.01/\sqrt{N}$ for $N$ antennas, providing that no additional loss of precision occurs in the

subsequent signal processing. This is also the residual gain in the direction of an intended null, relative to the target's gain; for $N = 350$, it is $5.35 \times 10^{-4}$ (voltage) or –64.4 dB.

**3. Design for Time-Domain Architecture**

First consider a design which, unlike the baseline, treats each channel as a single band rather than analyzing it into subbands. This alternative includes separate (but somewhat coarse) delay tracking for each beam.

With a single band it is not necessary to update the complex gain every sample, so the gain tracking logic can operate at a much slower clock rate. A reasonable configuration for the logic of a complex gain function generator is shown in Figure 1. Here there are separate DDS-like linear interpolators for the phase and magnitude. At each update, the phase is presented to sine and cosine lookup tables to convert to (real,imaginary) format; and these results are multiplied by the separately-generated magnitude. The final result is

$$g(t) = [1 + \alpha(t)][\sin \phi(t) + j \cos \phi(t)]$$

where $\phi(t)$ is the phase and $\alpha(t)$ is the variable part of the magnitude. This form simplifies the multipliers if $\alpha(t) \ll 1$, as further explained in the next section.

The change in phase $\Delta\phi$ and change in magnitude $\Delta\alpha$ at each update must be loaded into the appropriate registers periodically; details of this are not shown in the figure. Even though the required update rate of 775 kHz is far below the data sampling rate, it is fast enough that the mechanism of Figure 1 cannot readily be implemented in a microprocessor. We assume that it is done in hardware. But we assume that updating of $\Delta\phi$ and $\Delta\alpha$ is slow enough (a few kHz or less) that those values can be computed and loaded by a microprocessor.

An alternative arrangement with real and imaginary interpolators would avoid the lookup tables and multipliers, which would be a significant saving. But it would likely require that the reloading of the parameters be much more frequent, requiring that more of their computaton be implemented in hardware rather than in a microprocessor. This is because the trajectory of the gain in the complex plane is very nearly a circle. Overall, it is believed that the configuration in Figure 1 will be simpler.

It should be possible to implement the gain generator in logic that can be clocked at more than 100 MHz, which is 129 times the required update rate. Much of the logic can then be shared among the 8 generators required for the 4 dual-polarization beams of one antenna channel. In addition, less pipelining and more combinatorial logic can be used, reducing the gate count. One method of sharing the logic among channels is shown in Figure 2. There are 8 sets of the parameter registers $\phi, \Delta\phi, \alpha, \Delta\alpha$ and 8 output registers, one for each beam and polarization; the gain generator is clocked at 8 times the single-beam update rate, cycling through the sets of registers. This results in the updating of the beams' complex gains being staggered in time, which must be taken into account by higher-level processors that compute and load the parameters. Timing in the gain generator must be synchronized at the system level so that the actual update time for each beam is known; this is the purpose of the SYNC signal shown in Figure 2. It is convenient if the SLOW_CLOCK that drives the gain generator is a sub-multiple of the data sample clock; if the latter is 100 MHz, then the ratio can be 16 and the gain generator can run at 6.25 MHz. (A more detailed design may include other timing considerations that may lead to a higher internal speed for the gain generator logic.)

**4. Numerical Considerations**

In this section, I consider the word sizes and computational accuracies needed within the tracking generator in order to maintain 1% accuracy in the complex gains. The word lengths shown in Figure 1 are the results of these considerations.

In the phase generator, 10b are required in the phase representation in order to achieve this accuracy. The lookup tables must then produce at least 7b numbers for the sine and cosine; 8b is actually more convenient, allowing some margin. Only one table representing one quadrant of sine is actually needed, so the table's size is 256 words of 8b each. The 10b phase is used to calculate two 8-bit addresses (one for sine and one for cosine) in this table and to set the sign of the result. The $\phi$ and $\Delta\phi$ registers need to be larger in order to avoid roundoff errors during the extrapolation. Assuming that at least 500 gain updates are desired per re-loading of these registers from higher level processors, another 9 bits are needed within the extrapolator to avoid roundoff, so that the registers should be at least 19b long; 20b registers are more convenient.

Similarly, the magnitude generator needs to produce results with at least 7 significant bits. The format depends on the range of magnitudes that needs to be represented. For the purpose of creating nulls as discussed in the Introduction, it seems safe to assume that variations in gain will not exceed a few percent of the nominal value. In that case, letting the gain be $g = g_0 + \alpha$, it is efficient to let the interpolator generate only $\alpha$, whose length need be only a few bits. The situation is especially simple if we can assume that $g_0 \equiv 1$. Again an additional 9 bits are needed in the registers to avoid roundoff errors, leading to 12b registers for $\alpha$ and $\Delta\alpha$.

The two multipliers can then be implemented as 4b $\times$ 8b, with the results rounded to 8b, followed by an 8b adder. The full representations of all numbers are shown in Figure 1, using the notation $n.m$; this indicates a fixed-point number of length $n$ bits with its LSB $m$ bits to the right of the binary point. Note that, with signed operands, multiplying an $n.m$ number by an $a.b$ number gives a $(n + a - 1).(m + b)$ result if no precision is lost.

**5. Design for Frequency-Domain Architecture**

In the baseline design [1], a frequency-domain architecture was proposed in which coarse delay tracking is followed by analysis into 16 subbands, in common for all beams. Then each beam is separately tracked in complex gain, with a phase slope across the subbands. The data is processed in blocks of 16 samples, one from each subband.

Attempting to extend this so that the gain tracking includes both magnitude and phase leads to an arrangement like Figure 3. This is similar to the time-domain architecture of Figure 2, but there are some important differences. The gain generator now computes the complex gain for each of the 16 subbands of one beam and stores them in a length-16 memory before going on to the next beam. The memory is then read at the full data clock speed (FAST_CLOCK in Figure 3) and applied to the data samples from the 16 subbands for one block; the same data is then re-read for the next block, continuing until it is next updated by the gain generator.

The gain generator needs to be slightly more complicated so as to implement the phase slope. This is done by making the phase generator portion as a two-stage DDS, and supplying an additional parameter $\tau$ by which the phase is incremented at each update. A block diagram for this is given in [1]. An additional register is provided for each beam's bank to hold $\tau$, so the size of the register file (and the re-loading rate for the upstream computers) is increased only slightly over the time domain architecture.

While a scheme like this might be feasible, there are some difficulties with it. First, the gain generator is now time shared among 128 results but the required update rate for any one result is no smaller. Achieving an update rate of 775 kHz then requires that SLOW_CLOCK run at 99.2 MHz, which is about the same as FAST_CLOCK. Whereas the 775 kHz rate applies to an expanded array (3 km), it can be reduced by a factor of 4 for the initial array (0.7 km). At least this factor of 4 is needed to make the implementation feasible. Further reduction can be achieved by relaxing the 1% accuracy critereon. Second, the 8 separate dual-port memories may be difficult and/or expensive to implement in the Xilinx FPGAs that we intend to use. These memories must operate at the full data rate, and sometimes an update (write) will occur at the same time and for the same address as a read. Each one is only 256 bits, which is too small for efficient implementation with the FPGA's block RAMs. A much more detailed design is needed to determine the feasibility of this architecture.

In fact, this arrangement may not be adequate for some purposes. It does not provide independent gain tracking by subband, but rather just a phase slope (fine delay) across the subbands of any one beam. If we desire to create a beam with one or more nulls far from the delay-tracking center, it is known [2] that the nulls will have narrow bandwidth. But if there are several narrow band interferers at different frequencies within the channel such that they fall in different subbands, then each might be placed in a null provided that the subbands can be tracked separately. To support this, an extension of the design of Figure 3 would require 128 separate register banks rather than 8, with a total of 512 parameters to be re-loaded by the host processor; the re-load rate is estimated at (1 kHz)$\times$78b$\times$128 = 9.98 Mb/s for each of 4 IF channels of each antenna.

## REFERENCES


[1] L. D'Addario, "ATA IF Processor: Description of the preliminary baseline design," 2001-Aug-02. http://astron.berkeley.edu/∼ldaddari/ata/baselineDesignAll.pdf

[2] G. Bower, presentation at ATA Engineering Meeting, August 2001.

[3] L. D'Addario, "ATA IF Processor: Requirements," rev 2.1, 2001-Jul-14, Appendix A. http://astron.berkeley.edu/∼ldaddari/ata/ifpRequirements.pdf

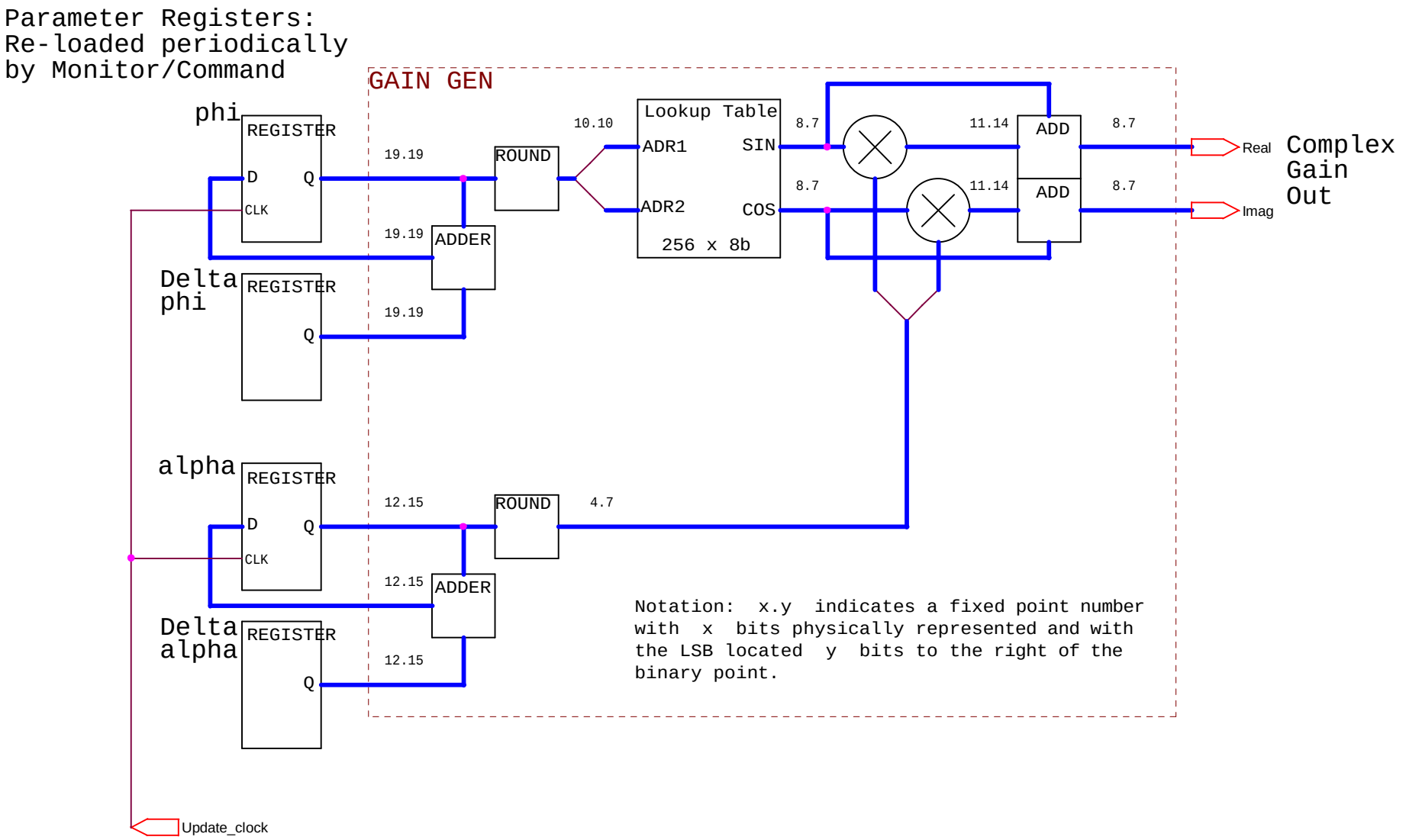


Figure 1: Complex Gain Generator

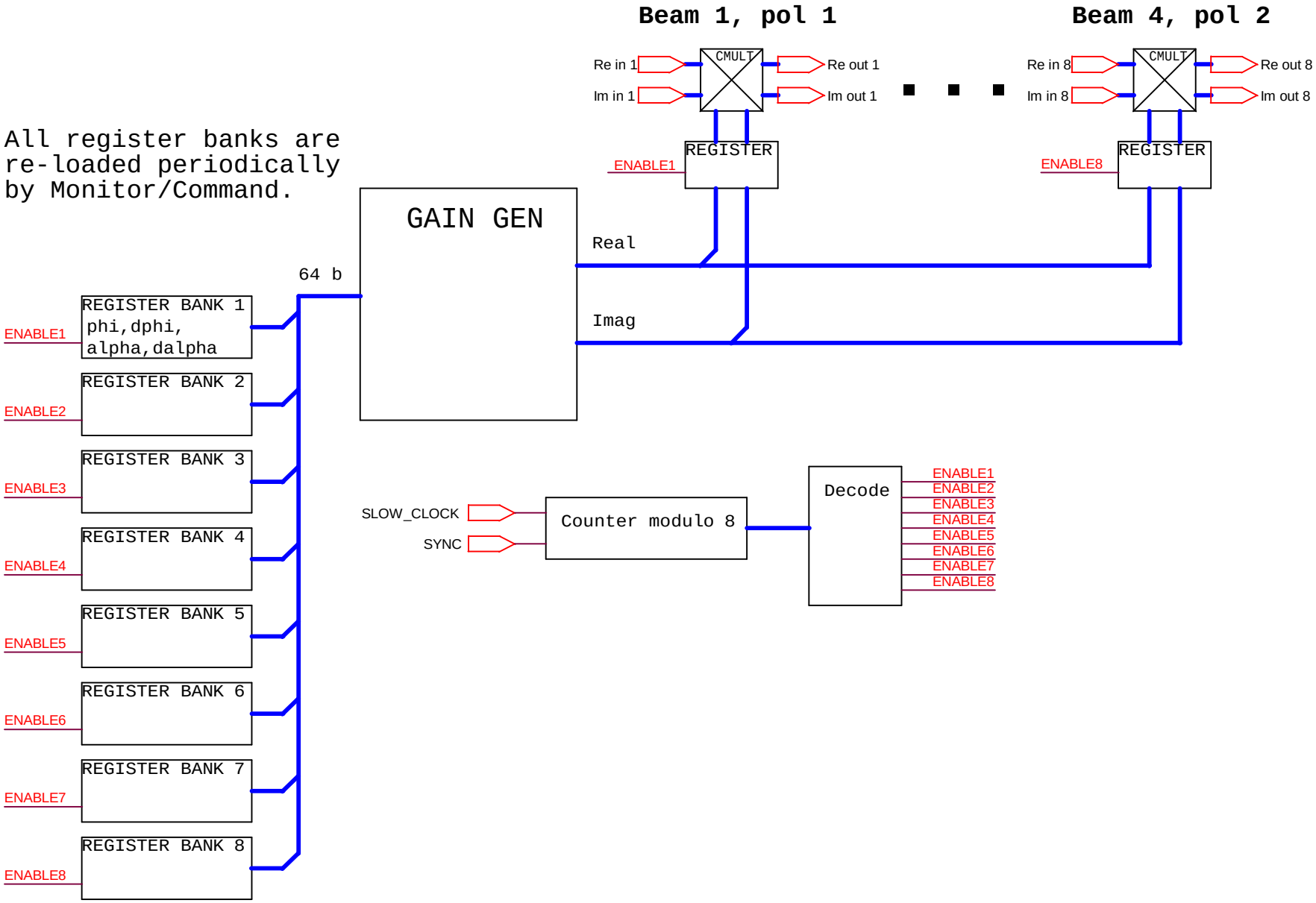


Figure 2: Time-sharing a complex gain generator among four dual-polarization beams.

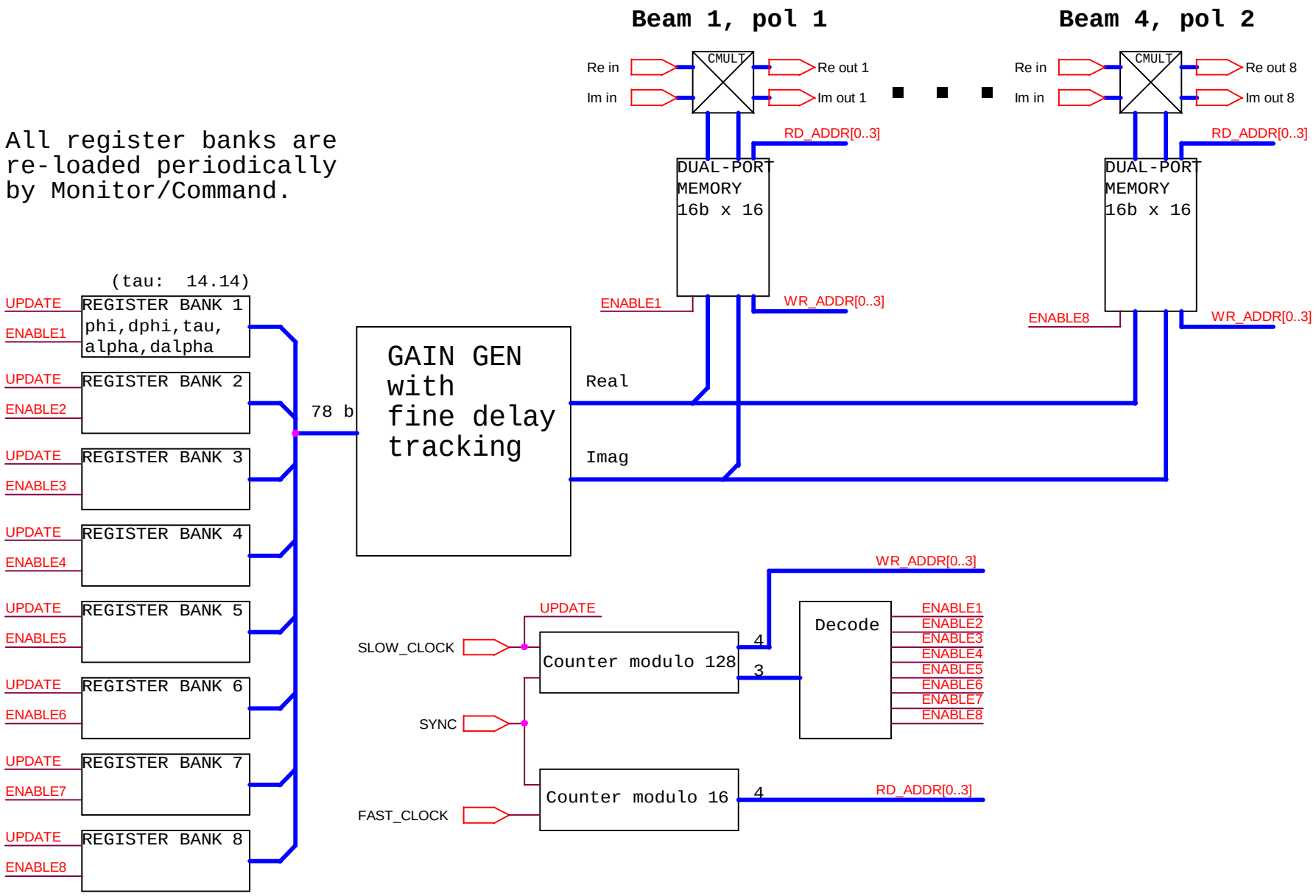


Figure 3: Design for frequency-domain architecture.