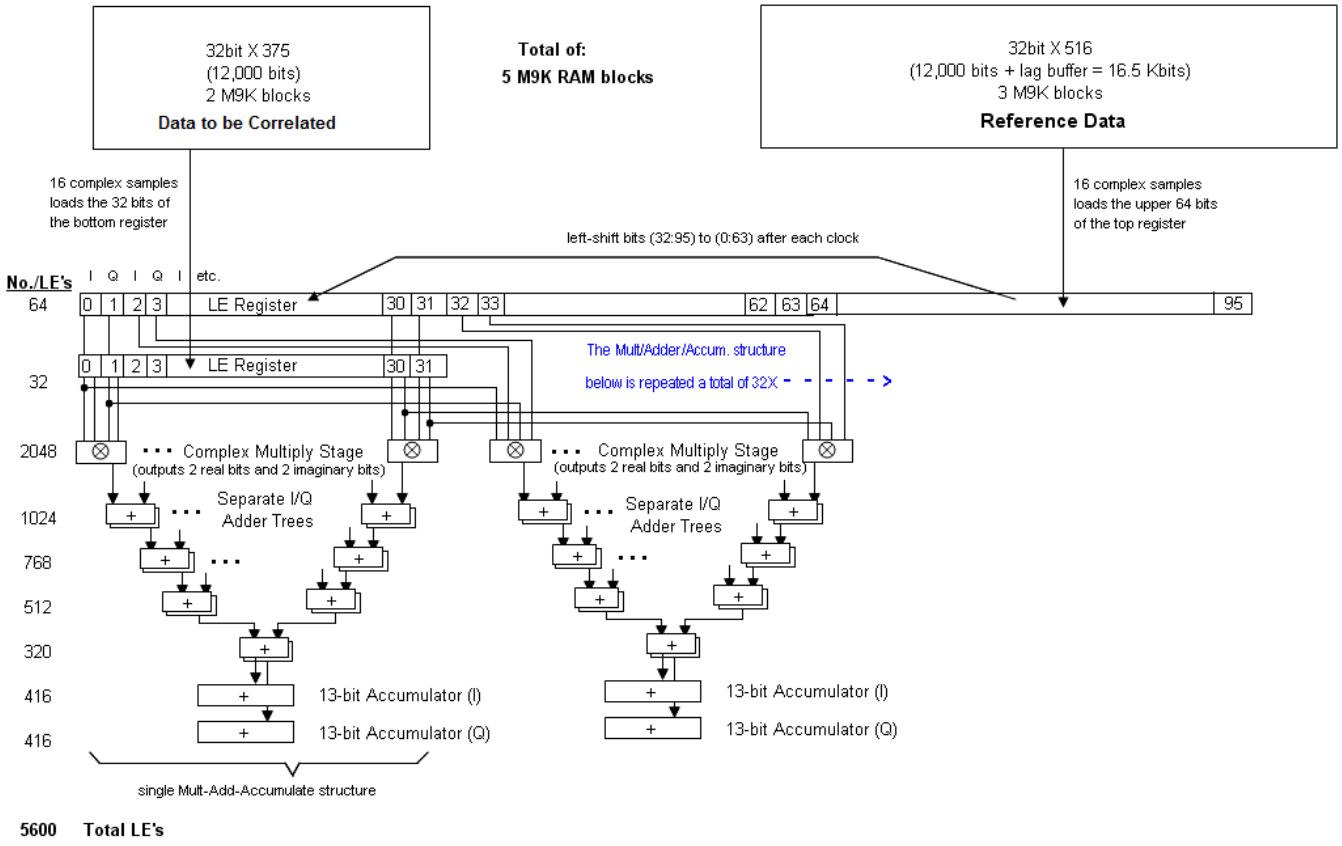
Partial Correlation Engine

The goal of this project was to provide a structure capable of 32 complex correlations per second between an array of Reference Data and an array of Correlation Data. The arrays are arranged as 6000 complex samples per segment by 800 segments. In this example, each sample has been reduced to a single complex bit – the sign bit – of the original data. It was further required that this computational machinery, along with other logic, fit in an Altera Cyclone III EP3C80. That achievement required a careful analysis, logic partitioning, and frugality with the available FPGA resources.

The RAM blocks at the top the diagram are used as cache memory containing the two sets of data to be correlated. They are updated from a DDR2 memory store after the correlation of each 6000-sample segment for all 2250 lags. The RAM block in the upper right of the diagram holds the reference signal against which each data collection is correlated. The left M9K RAM holds samples (sign bits) of the data to be correlated, including the 6000 complex samples that represent the zero-lag point in the computation, along with samples both before and after the zero-lag point, for a total of 8250 complex samples.

That number of samples will almost, but not quite, fit into 2 M9K block RAMs. However, it's not necessary to have all 8250 points resident in the block RAM at the same time; only 6000 points at one time are needed for rapid-reloading of the computational structure. So, after a few of the lags are computed, and the corresponding samples are no longer required for the current segment correlation, the correlation engine's state machine could schedule the writing of the last few samples into the M9K RAM.

Alternatively, this M9K buffer could simply be "padded" with a few LE's to create a slightly larger local RAM. Another alternative is to reduce the requirements to calculate only 2192 lags. In that case, all the required data could be loaded into the two M9K RAMs in one operation. Ultimately, it was decided to build the buffer needed for the lag data out of 3 M9K RAMs. Although somewhat wasteful if the RAMs were required by other logic, they were not.



The single Multiply-Add-Accumulate structure in the lower left of the diagram above produces, in one clock, a single partial (16 complex multiples, summed) output sample.

32 identical Multiply-Add-Accumulate structures, suggested by the right side of the diagram, produce, in one clock, 32 of these partial complex output samples (lags). After each clock, a new 32-bit piece of each input stream is loaded into each register as noted in the diagram above. At the same time, the contents of the top register are left-shifted by 32 positions. (Loading both input registers after each clock is the only way to avoid storing/reloading each of the 32 partial outputs each clock). Doing this 6000/16 = 375 times results in 32 completed output samples, each of which is passed to CORR2. Note that the upper half of the top register and all of the lower register may not be necessary; it may be possible to use the RAM outputs directly. The above process is repeated 71 times to produce 2250 lags (2250/32 ~= 71). Computing the completed value for each of the 2250 output samples requires:

(71 groups of 32 lags X (375 clocks per 32 lags) = 26625 clocks = 138.45 us

The structure can therefore perform one correlation of (2250 lags by 800 segments) in 138.45 us. X 800 = 110.76 ms, or about 9 complex correlations/sec. Four of these structures were required in order to meet the system performance goal of 32 complex correlations per second.