CARMA Memorandum Series #[Internal]

Revised CARMA Correlator VHDL Development

Kevin P. Rauch (UMD)

April 15, 2005

ABSTRACT

This document discusses the VHDL-related coding and testing used to validate the capability and performance of the revised CARMA correlator digital hardware.

Change Record

Revision	Date	Author	Sections/Pages Affected
	Remarks		
1.0	2004-October-20	KPR	
	Initial release.		
1.1	2005-April-13	KPR	
	Updated to reflect current status.		

1. Overview

The CARMA first-light correlator will be a hybrid system with a few bands consisting of recycled COBRA hardware and the remainder incorporating upgraded digital hardware. From an observer's point of view the two types of bands differ only in the number of spectral bandwidth modes available and in the spectral resolution (and possibly efficiency) of each mode. However, significant internal signal processing differences will exist between each class of hardware. The changes strongly affect the selection of hardware components on the revised digitizer and correlator cards since a minimum level of performance will be required of the digitizer FPGAs in order for the system as a whole to function. This critical dependence on FPGA performance demands thorough up-front development testing to ensure the required targets can be met. The purpose of this document is to record the development and results of these tests.

Most of the existing CARMA VHDL codebase (targeted for COBRA hardware) can be reused on the revised hardware with little modification. The major component enhancements are as follows:

1. **Sub-ns delay correction:** Implementing a programmable sub-ns delay offset requires a digital filtering component whose coefficients are reloadable on demand from FPGA RAM.
2. **Phase offset correction:** Use numerically-controlled oscillators with adjustable phase to perform digital downconversion and programmable phase offset correction.
3. **High-demux filtering:** All digital filters must support a $DEMUX = 8$ to enable processing of wideband (500 MHz) input.
4. **Multi-bit sample support:** Multi-bit digitizer samples stress digital filtering and inter-FPGA I/O resources. Their limits impact efficiency and set the maximum possible cross-correlation sample size, regardless of lag density.
5. **Digital noise source:** Create a digital noise component suitable for filter testing and reusable in a digitizer self-test mode.

The following board-level issues require attention:

6. **Development board interfacing:** A host-level FPGA RAM reader/writer for the Stratix II DSP development board is needed for detailed verification of in-circuit design performance.
7. **FPGA layout options:** Determine the appropriate FPGA pin package and an FPGA organization on the revised cards which utilizes those pin resources effectively.
8. **Board-level simulation:** Develop a functional simulation of major board components prior to PCB fabrication to validate board performance.

Additional infrastructure-related development tasks include:

9. **CVS reorganization:** The current COBRA CVS tree is burdened with obsolete/duplicate files and a complex layout. Revised development will take place in a new, streamlined project.
10. **Filter simulator:** Upgrade and maintain the existing bit-accurate filtering and decimation simulator.

The following sections discuss each of these items in turn.

2. Fractional Sample Delay Filters

2.1. Delay Filter Design

Sub-ns delay correction is done using a fractional sample delay digital filter. A different set of filter coefficients is required for each particular fractional delay. Depending on the filter design technique, the coefficients for a particular delay value are computed on demand, or the delay is quantized to fixed precision and the appropriate row from a pre-computed table of coefficients is used. The latter minimizes CPU load but may consume a lot of RAM to store the table. Interpolation of filter coefficients can be used to reduce the size of the table and balance the two approaches. The table only needs to store coefficients for fractional delay values between [0, 0.5] samples as the coefficients for a delay x are simply those for $1 - x$ in reverse order (delayed by one sample).

Both the frequency and phase response of the delay filter depends on the delay. The worst-case response variations over all possible delays must be determined to ensure the filter performance as a whole meets the design specification. By symmetry the phase errors in delay range [0, 0.5] are the negative of those in [0.5, 1]. Thus given random delays, phase errors may largely cancel even if they exceed specification individually. The official CARMA phase error limit of 0.25 degrees per antenna can be met with a 64 tap filter (see Rauch & Hawkins 2004).

2.2. Delay Filter Implementation

Updates of the delay value will occur on the phase switch timescale—at a minimum every 16ms, and ideally every 1ms. In principle the next set of delay filter coefficients can be computed from a table stored in FPGA RAM, or written into FPGA RAM by the DSP/CPU. In the former case update rate is not a concern, but a significant fraction of on-chip RAM will be consumed, and a more complicated state machine will be needed to process the table. In the latter case sending the coefficients to each FPGA will increase data traffic to the FPGAs, which may limit the update rate. In either case, a non-trivial state machine is needed to propagate the new coefficients to the filter instantiations, as described in § 4.1.

The Altera FIR megacore requires that reloadable filter coefficients be stored in either M512 or M4K memory blocks. For a fully parallel filter, the number of memory blocks needed scales $\propto DEMUX \cdot NUM\_TAPS \cdot SAMP\_BITS$, where $DEMUX$ is the data demux factor, $NUM\_TAPS \approx 64$ is the number of filter taps, and

$SAMP\_BITS$ is the input sample bit width. The fractional delay filter is applied immediately after the digital delay line, hence $DEMUX = 8$ independent of the spectral bandwidth setting. This assumes the filters are run at 125 MHz; if 250 MHz operation is feasible, $DEMUX = 4$ can be used. A Stratix II EP2S60 contains 329 M512 and 255 M4K blocks; according to preliminary testing this is (barely) enough to support $SAMP\_BITS = 6$ if $DEMUX = 8$. Note that different megacore instantiations are required for each memory block type, which adds a bit of complexity to the VHDL.

The fractional delay filter will reside in the FPGAs containing the digital delay line and autocorrelation, both of which also use FPGA RAM. Ideally all M512/M4K should be reserved for filter coefficients. The EP2S60 also contains 2 M-RAM blocks, one of which would be consumed if the filter coefficient table is stored on-chip. This leaves one M-RAM block (half a megabit) for the delay line and lag dump memory in this case.

2.3. Development Tasks

2.3.1. Reloadable FIR Filter Coefficients

Verify operation and dynamic control of a digital filter with reloadable coefficients.

2.3.2. Filter FMAX

Can the delay filters operate at 250 MHz?

Filter FMAXs for the band shaping filters cluster around 200 MHz. Reliable operation at 250 MHz is not expected for the fractional delay component but has not been tested; this implies that a maximum of six bits per sample can be filtered. In addition, a small loss in efficiency (1-2%) in wideband mode will occur due to roll-off in filter frequency response near 500 MHz (cf. Fig. 1 of Rauch & Hawkins 2004).

2.3.3. FPGA RAM Usage

Can the digital delay line and lag dump FSM share a single M-RAM block?

Three separate digitizer logic components require dynamic access to M-RAM blocks: the digital delay line (read-write access), the fractional delay filter (read-only access), and the lag dump finite state machine (write-only access). The fractional delay filter also makes use of M512/M4K blocks to store the reloadable FIR filter coefficients; however, this use is implicit in the Altera FIR megacore instantiations and not visible to the high-level VHDL code. The digital delay line accesses the on-chip RAM continuously and hence cannot share its M-RAM block with other logic. Assuming the autocorrelations are performed in the digitzer module-attached FPGA (as for COBRA), the lag dump FSM and fractional delay filter must share the remaining M-RAM block. Both components require only occasional RAM access (between phase-switch cycles and after each integration, respectively) which makes sharing possible, subject only to data rate constraints. If necessary, an M-RAM block can be configured up to 128-bits wide to maximize throughput.

2.3.4. Delay Update Rate

What update frequency can the DSP/CPU support?
Can all M512/M4K blocks be reserved for the fractional delay filters?

To support M-RAM sharing and to simplify the coefficient-reload FSM, the on-board DSP/CPU should be responsible for calculating the delay filter coefficients and writing them to M-RAM; this avoids delay quantization noise and allows most of the shared M-RAM block to be used for lag storage and miscellaneous control registers. The Stratix II M-RAM blocks support true dual-ported access which allows independent interfacing to the on-board DSP/CPU. With this arrangement all M512/M4K blocks can be allocated to the FIR megacore instantiations, maximizing the usable digitizer sample bit width.

3. Phase Offsets and Digital Downconversion

3.1. Numerically Controlled Oscillators

The FPGAs responsible for spectral band creation (digital downconversion and decimation) can perform phase offset correction by using numerically controlled oscillators (NCOs) with a dynamically adjustable phase. The Altera NCO compiler supports adjustable frequency and phase offsets. The frequency offset could be used to vary the center frequency of the spectral band window; however, the analog response will be optimal at 750 MHz (the center of the downconverter band), so this feature will not be utilized. The phase offset will be used to compensate for block downconversion and lobe rotation corrections to flatten the phase. An NCO suitable for CARMA consumes one M4K block, 4 9x9-bit DSP multipliers, and 200 logic cells.

Eight NCOs operating in parallel are needed for $DEMUX = 8$. Total resource usage in an EP2S60 remains small. The phase offset differs for each NCO by a constant offset (relative to the first NCO) to account for the time differences between the $DEMUX$ samples which are being modulated simultaneously. The (complex) NCO outputs are multiplied by their respective input sample to produce the demultiplexed data stream representing the result of digital downconversion. The Stratix II DSP blocks can be used to perform the multiplications to save logic and maximize operating frequency. The result is input to the FIR filtering/decimation and demodulation pipeline to create the spectral window in the same manner as the narrowband modes for the COBRA-based bands.

3.2. Development Tasks

3.2.1. NCO Operation

Implement and test the digital downconverter component.

The revised CARMA design instantiates an NCO incorporating 16-bit (0.005 deg) phase input precision, 18-bit amplitude output precision, and 20-bit internal accumulator precision. Both sine and cosine outputs are generated. Note that since the modulation frequency is fixed at half of Nyquist (which shifts 250 MHz to DC), by symmetry *no* frequency harmonic generation occurs regardless of output precision: there are only four unique output phase states, for which 2-bit quantization is sufficient. The additional amplitude precision is useful only to maintain phase offset accuracy. The fixed offset frequency allows a *single* NCO to generate all $DEMUX = 8$ sin/cos outputs as the following $DEMUX - 1$ outputs are trivially related to the first, further minimizing resource usage.

Stratix II DSP block multipliers are used to perform the actual modulation; they contain hardware multipliers designed for 9-bit inputs (or multiples thereof). The CARMA design instantiates 18x9-bit multipliers (mult18x9 VHDL component) to modulate the NCO outputs by the (real) input data and produce the (complex) modulated data stream. Using hardware multipliers also conserves logic. The multiplier output is correctly rounded down to 12-bit data before further processing; neglecting detailed round-off properties, note that this is equivalent to rounding both NCO output and data input to 6-bit values prior to modulation. Thus the original 16-bit phase precision is maintained only in an average (dithered) sense.

Both RTL (functional) and post-synthesis (timing) VHDL simulations of the complete spectral bandshaping component have been performed, which includes the NCO logic to apply a phase offset to the input signal. The simulations consisted of decimating 1 ms of random input data and comparing the final output to the results of a high-level, bitwise-accurate C++ simulator (cf. § 11) in which the frequency and phase responses are monitored directly; all spectral bandwidth modes were tested for several values of the phase offset and each performed as desired.

4. High-demux Filtering Components

4.1. Parallel FIR Filters

The Altera FIR megacore has no provision for $DEMUX > 1$ input. This capability can be added as follows. Since $DEMUX$ filter outputs must be produced each clock cycle, and each output depends on all $DEMUX$ input values, a total of $DEMUX^2$ sub-filters must be instantiated inside the top-level filter component. There are $DEMUX$ unique sub-filters, each containing a fraction $1/DEMUX$ of the original filter coefficients; they correspond precisely to the corresponding polyphase decomposition of the original filter. The output of these polyphase sub-filters are combined by an adder tree to produce a single output value each clock cycle given $DEMUX$ input values. This structure is instantiated $DEMUX$ times (with successively delayed input values) to produce $DEMUX$ unique outputs every clock. The total increase in logic usage is a factor of $DEMUX$, plus a small increment for the adder trees. The number of taps in the prototype filter should be a multiple of $DEMUX$ so that the sub-filters are all the same length. Note that every particular value of $DEMUX$ generates its own set of sub-filters; this substantially expands the VHDL code since every value of $DEMUX$ requires generating and instantiating a unique set of FIR megacore components. If the coefficients of the prototype filter change, a large number of sub-filters need to be manually regenerated using Altera's Megawizard Plug-in Manager. This is tedious and must be done carefully to avoid inconsistencies.

Updating coefficients to the fractional delay filter is complicated by the need to update each of $DEMUX^2$ sub-filters with a particular subset of the total filter coefficients.

4.2. Development Tasks

4.2.1. Sub-filter Generation

To what extent (if any) can the polyphase sub-filter generation be automated?

Each unique Altera FIR megacore instantiation requires manual invocation of the "IP Toolbench" GUI (via Quartus) to set parameters defining the component, such as data widths and filter coefficient values. Although this cannot readily be automated, the GUI does allow editing parameters of previously produced components; hence amending pre-existing components as needed will not be a laborious task. Initial component variations for each spectral line mode have already been created.

4.2.2. High-demux Filters

Implement and test edge-defining filters supporting one or more values of $DEMUX > 1$.

Decimators for all spectral line modes have been coded and VHDL simulations of both RTL and post-synthesis timing models perform as intended. These modes utilize filters spanning the entire range of $DEMUX$ values from $DEMUX = 8$ (500 MHz bandwidth) down to $DEMUX = 1$ (62 MHz bandwidth and below).

5. Multi-bit Sample Support

5.1. Issues

The new digitizer modules produce 8-bit output samples and most or all of these bits will be brought onto the digitizer FPGAs. How many bits it is practical to store, transmit, and process is limited by the memory, I/O, and logic resources of the Stratix II FPGAs, respectively. Digital delay line memory usage should not exceed the size of an M-RAM block (half a megabit), of which the EP2S60 contains two. This is more than sufficient for the maximum delay needed for CARMA even maintaining full sample precision. The delay line feeds the fractional delay filters, which will be limited to at most 6-bit input samples due to M512/M4K availability (see section 2.1). The delay filter will output well over 6 bits but the output must be transmitted to the FPGA performing digital downconversion and decimation. At least 6 bits should be retained to maintain support for 4-bit cross-correlations. Sending 8-bit output requires two 32-bit inter-FPGA buses running at 125 MHz. The digital downconverter component can easily process 8-bit input samples, which will roughly double in width after modulation. The FIR filter/decimation stage will need to truncate this at some stage. The edge-defining filter consumes most of the logic; 8-bit input should be feasible at 500 MHz ($DEMUX = 8$) for $NUM\_TAPS \sim 64$. To enable 4-bit cross-correlation, each front panel LVDS connector must carry *two* 4-bit data streams; this demands demux-by-4 @ 250 MHz capability in wideband mode, which may or may not be feasible. However, some antenna signals must also travel through fan-out boards to supply all correlator cards with input; these boards were not designed to operate faster than 125 MHz. Thus although 4-bit cross-correlations could be demonstrated using a direct digitizer-to-correlator link, practical implementation may demand a redesigned, high-speed fan-out board.

FPGA layout and pin availability (cf. § 8) also determines whether two 4-bit samples can be routed to all four front-panel connectors in the first place. If the layout mirrors that of the COBRA digitizer boards, both 4-bit sample streams must be sent to all FPGAs, which implies two 32-bit data buses for data return from FPGA #1 to #0 and from #2 to #3, in addition to the two used to transport the 8-bit delay filter outputs from FPGA #0 to #1 and from #3 to #2. In all, four 32-bit buses must connect FPGA #0 to #1 and #2 to #3 to allow 4-bit cross-correlation at 500 MHz bandwidth using the original layout; to keep the pinout uniform, a total of 8 buses are needed. If a 2x2 layout is used, so that a single FPGA drives two front-panel connectors (using independent I/O buses), the total number of data buses can be reduced to seven [diagrams TBA].

5.2. Development Tasks

5.2.1. FIR Design

What are the limits of and best balance between FIR filter input sample width (signal precision) and filter length (band edge sharpness)?

High-level software simulation of the VHDL components (see § 11) showed that using 6-bit input samples reduced efficiency by 0.1% relative to 8-bit samples; for 4-bit samples the reduction was 1.5%. The EP2S60 FPGAs were found to contain enough logic to allow 12-bit filter input with filters of sufficient quality for CARMA. The implemented filters have typical in-band response variations of 0.1dB, out-of-band rejection of 35-40 dB, and edge transition widths equivalent to two channels. (The high-resolution narrow-band modes utilize longer filters exhibiting sharper edges.) Typical logic cell utilization was 60% with FMAXs near 200 MHz, comfortably above the 125 MHz target frequency.

5.2.2. FPGA Buses

Can the inter-FPGA buses operate at 125 MHz?
Can a single design support both single-ended and differential I/O?
Can performance of the various I/O signaling standards be tested with the Stratix II DSP kit??

Realizable inter-FPGA data rates depend on the I/O signaling standard and driving voltage employed (cf. http://www.altera.com/products/devices/stratix2/features/io/st2-single_ended_io.html). Typically, higher driving voltages provide greater performance potential at the cost of greater power dissipation (and perhaps EMI?). Altera claims single-ended I/O "performance targets" of 300 MHz for 3.3-V/2.5-V LVTTL or LVCMOS, 250 MHz for 1.8-V LVTTL or LVCMOS, and 200 MHz for 1.5-V LVCMOS. Even allowing a factor of two "reality check", the CARMA target of 125 MHz single-ended operation appears practical. This would allow 8-bit input and 4-bit output (cross-correlation) samples to be transported within the digitizer. As a fall-back position, 93.75 MHz operation would support 6-bit/3-bit sample transport, and as a fail-safe 62.5 MHz (4-bit/2-bit transport) would still meet CARMA requirements, at the cost of 2% efficiency relative to 125 MHz operation.

It is unclear how much effort would be required to test each of these options using the Stratix II DSP development board—or even whether it is practical (can the I/O voltage and signaling standard be reconfigured at will?) As the DSP kit's A/D converter operates at 100 MHz, achieving 93.75 MHz bus operation should be as simple as cloning Altera's I/O solution for this device. For high-speed I/O—advantageous (if not required) for the digitizer module input bus—special Stratix II features such as dynamic phase alignment (http://www.altera.com/products/devices/stratix2/features/io/st2-source_synch.html) may also prove valuable. Can these be tested using the DSP kit??

Of primary importance is to understand what demands the various I/O options impose on the physical board design. For example, can the digitizer board easily be designed to support reconfiguration of voltage and I/O standard to allow selection of the lowest power/EMI option that still operates reliably at 125 MHz *in situ*??

5.2.3. LVDS Outputs

Can the digitizer front-panel LVDS operate at 250 MHz? Can the fan-out boards operate at 250 MHz?

Answering these questions determines the feasibility of performing 4-bit cross-correlations in wideband mode; however, this capability is not a requirement, not relevant for first light, and investigating it should not delay or complicate the hardware redesign process.

6. Digital Noise Source

6.1. Design Methodology

The ideal digital noise source would use little logic, produce quality random bits (with an adjustable output width), and have a long period to support extended integration tests. In practice the component will be instantiated multiple times to simulate demultiplexed digitizer input. If these instantiations differ in the period of their pseudo-random number generators, random output with an extremely long period can be produced. Given a component generating a single random bit, it is easy to generate a vector of random bits (by changing the initial seed of each instantiation), but this does not extend the period of the sequence and could lead to correlations between bits depending on the algorithm employed. A random number generator producing multiple bits at once may require less total logic, at the likely cost of reduced statistical uniformity and perhaps also operating frequency.

6.2. Development Tasks

6.2.1. Algorithm

What base random number generation algorithm should be used?

By far the simplest hardware implementation is for algorithms based on primitive polynomials modulo 2 (e.g., Press et al. 1992). VHDL components (randbit and randsamp) have been implemented allowing selection of one of several primitive polynomials via a generic.

6.2.2. Period

Does statistical quality suffer if long period output is created simply by concatenating bits produced by short-period generators?

Not investigated. Use of multiple polynomials does enable multi-bit random sequences of extremely long period to be produced; however, possible limitations in the statistical quality of these sequences used as cross-correlation input remains unknown.

7. Host-level Interfacing

7.1. Testing Infrastructure

Interfacing the Stratix II DSP development board to the Linux host is a basic prerequisite to detailed Stratix II testing. The primary task is to develop a generic data transfer agent between the host and FPGA RAM—in particular, a Stratix II M-RAM block. VHDL component testing will be based on reading and/or writing this memory. The existing CARMA VHDL code needs to properly instantiate a block of Stratix II M-RAM instead of FLEX 10KE SRAM blocks to function on the new devices. In particular, M-RAM supports true dual-ported operation, but *not* the asynchronous, single-ported operation used in the FLEX 10KE design. The updated design can be used as the basis for new component testing by replacing the lag accumulators with data output from the component under test, which would be written to RAM using the existing lag dump state machines.

Transferring the contents of M-RAM to the host depends on the communication facilities provided by the Stratix II development board. One possible design path is to instantiate a NIOS-II processor on the FPGA and send the data using a standard protocol supported by the board, such as serial or ethernet. This requires surmounting the learning curve for the NIOS processor, but simplifies the Linux host code (no custom device drivers, etc.)

7.2. Development Tasks

7.2.1. FPGA RAM Reader

Implement a generic FPGA RAM reader/writer for the Stratix II development board.

Use of the Altera SignalTap II component allows convenient signal capture from the DSP kit Stratix II device to the host computer. Tapping the RAM data lines is therefore the simplest data acquisition solution if adequate for testing, which in terms of verifying sane FPGA component operation should be the case. Testing CPU-to-FPGA RAM I/O is a more complex task requiring development of CPU-interface infrastructure and would likely entail considerably more effort.

7.2.2. FPGA RAM Component Updates

Update the lag dump and digital delay line components to use M-RAM blocks.
Implement M-RAM sharing between the lag dump and sub-ns coefficient reload FSMs.

8. FPGA Layout

8.1. Physical FPGA Configuration

The redesign of the digitizer and correlator boards provides an opportunity to reconsider the most advantageous physical layout, interconnection, and total count of the FPGAs on each type of board. Previous analysis (Rauch & Hawkins 2004) indicates a count of four EP2S60 devices for the digitizer. The FPGA count on the correlator cards can be adjusted (subject to power dissipation constraints) to minimize the number of correlator boards and/or unique VHDL configurations required per band. Three VHDL configurations are needed for the COBRA-based bands (Rauch 2003). A desirable constraint is to use the same pinout configuration for all digitizer and correlator FPGAs. Maximizing FPGA count on the correlator cards would also minimize the need for fanout boards. Minimizing data bus count may require creative pin routing on the PCB boards to allow a single pinout for all FPGAs. If support for upward migration to denser devices is desired, the 1020-pin package is the only one available for both the EP2S60 and denser Stratix II devices.

8.2. Development Tasks

8.2.1. Layout Options

Determine the recommended FPGA count and layout for each board type, as well as the pin package and bus configuration.

9. Board-level Simulation

9.1. Simulation Priorities

The goal of a board-level VHDL simulation is to ensure inter-operation of critical board components including the FPGAs, DSP/CPU, system controller, and DRAM. For the DSP/CPU and DRAM(?) only functional models will be available. Which features need to be simulated is yet to be determined. For the FPGAs either functional or post-synthesis timing models could be used. The FPGA configurations can also be stripped down to the most relevant components (e.g., the pipeline) for testing purposes.

9.2. Development Tasks

9.2.1. Board-level Simulation

What level of detail is required of a board-level simulation?
Should the FPGAs be represented by functional or post-synthesis models?

10. Revised CVS Structure

10.1. Design Goals

The current COBRA CVS tree has accumulated a significant amount of debris over its lifetime. Development of the revised CARMA correlator will take place in a streamlined CVS project organized according to the following principles:

- *Project contents:* The project will manage all FPGA, DSP/CPU, and Linux host software (including development and test code); board schematics and specifications; and appropriate documentation. The scope of the CVS project extends to all digital hardware physically residing in a correlator crate.
- *Directory structure:* The directory hierarchy will be organized by subsystem, such as `host`, `share`, `correlator`, `correlator/fpga`, etc. Initializing a skeleton tree will promote a logically ordered structure. Hardware variants (COBRA, revised, etc.) should be accounted for from the start to avoid gratuitous duplication of files between projects.
- *File naming:* Files names should focus on function, not subsystem; conflicts between corresponding files in different subsystems are already prevented by the directory structure. For example, the correlator pipeline component pathname should be `correlator/fpga/src/pipeline.vhd`, not `correlator/fpga/src/correlator_fpga_pipeline.vhd`; the latter is redundant.
- *File sharing:* All file functionality should be made sharable between subsystems whenever possible, including VHDL components, build scripts, and so on. It is *much* more useful to generalize/reorganize reusable components than to duplicate them in individual subsystems and specialize each, unless the differences are substantial (in which case any commonality should be abstracted into a new component if possible).

10.2. Development Tasks

10.2.1. CVS Creation

Create and populate the new CARMA correlator CVS tree.

A new CVS project (carmacorl) satisfying the preceding design goals has been implemented and added to the cvs.mmarray.org:/sw/cvscarma repository. All revised correlator development described in this document resides in this project; existing COBRA correlator VHDL code relevant to CARMA has been transferred and integrated into the new framework as well. Code duplication has been eliminated and most scripts have been generalized to support both COBRA and revised hardware targets.

The CARMA_CVS, QUARTUS_ROOTDIR, and MODELSIM_PATH environment variables must be valid for the new build scripts to run. In addition, the file share/scripts/paths.tcl must be created following the instructions in share/scripts/paths_tmpl.tcl. Build scripts are to run from the root fpga directory (e.g., cobra/digitizer/fpga).

11. Software Filter Simulator

11.1. Simulator Maintenance

During development of the CARMA spectral correlator configurations a bit-accurate C++ software model of the FIR decimator and modulation/demodulation logic was created. The model is able to create test vectors used by a corresponding VHDL testbench to verify operation of the filtering component. The addition of phase flattening to the digitizer signal processing must be incorporated into the software model to maintain support for detailed verification and test vector generation. For the most part the modifications are straightforward; however, bit-accurate simulation of the Altera NCO component may require experimentation.

11.2. Development Tasks

11.2.1. C++ Simulator

Update the C++ simulation program to support the expanded digitizer signal processing.

Done. The simulation program (firsim) has been documented with doxygen and can be found in the share/fpga/test directory. A makefile to build the simulation program is included. This program also produces the test vectors used by the FIR testbenches to verify operation of the band-shaping/decimation components.

REFERENCES

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in C (2nd ed.), p. 296.

Rauch, K.P. 2003, CARMA Memo 7.

Rauch, K.P., and Hawkins, D.W. 2004, CARMA Memo 28.