The CARMA Correlator

D. W. Hawkins (dwh@ovro.caltech.edu)

Revision: 1.10

May 27, 2006

Contents

1 Introduction ................................................. 3

2 System-level design considerations ......................... 5

3 1GHz Digitizer .............................................. 7
    3.1 Digitizer specifications .................................. 7
    3.2 Atmel AT84AD001B (dual 1GHz digitizer) ...................... 8
    3.3 RF input signal conditioning ................................ 11
    3.4 Quantization state monitoring .............................. 11
    3.5 Threshold control ......................................... 11
    3.6 Synchronization ............................................. 11
    3.7 Thermal monitoring ......................................... 12
    3.8 Notes ...................................................... 13

4 1GHz Clock Generation ...................................... 14
    4.1 COBRA board clock delay control .......................... 14
    4.2 CARMA board clock delay control ........................... 14
    4.3 AD9956 AgileRF synthesizer ................................ 17
    4.4 Notes ...................................................... 20

5 Stratix II FPGA ............................................ 22
    5.1 FPGA Pin assignments ....................................... 24
    5.2 FPGA Power estimates ...................................... 29
    5.3 Stratix II DSP Kit ........................................... 30
    5.4 Notes ...................................................... 32

6 PowerQUICC II Pro PowerPC CPU ............................. 33
    6.1 Processor requirements ..................................... 33
    6.2 Signal processing ........................................... 34
    6.3 PowerQUICC II Pro features ................................. 35
    6.4 Real-time requirements ..................................... 36
    6.5 Processing time estimates ................................... 36
    6.6 Notes ...................................................... 37

7 Clocks ...................................................... 38
    7.1 Notes ...................................................... 40
8 Power

8.1 Power Estimates ........................................... 41
8.2 Power Sequencing and Management .......................... 41
1 Introduction

This document describes the CARMA Correlator System. The Combined Array for Research in Millimeter-Wave Astronomy (CARMA) was formed by the merger of two university-based millimeter arrays; the Owens Valley Radio Observatory (OVRO) millimeter array, and the Berkeley-Illinois-Maryland Association (BIMA) millimeter array. The CARMA Correlator System processes signals from 15 antennas with a total bandwidth coverage of up to 4GHz. The correlator processes data in 500MHz bands, with 8 independently tunable bands used in the 4GHz system. The CARMA Correlator will consist of two hardware designs; the first-light correlator, and the revised hardware described in this document. The first-light correlator consists of three 500MHz correlator bands constructed by recycling the COBRA Correlator System used at OVRO, while the design of the remaining five bands is described in this document. Details of CARMA can be found at www.mmarray.org, while details on the COBRA system can be found at www.ovro.caltech.edu/~dwh/correlator. Further references to the CARMA Correlator in this document refer to the revised hardware.

The CARMA Correlator consists of two main hardware components; digitizer boards and correlator boards. The digitizer board converts analog signals to digital and performs antenna-based processing, while the correlator boards cross-correlate the signals from pairs of antennas. The functionality of a correlator board is a subset of that of a digitizer board, and this document describes a common design to implement both board types.

The CARMA Digitizer Board consists of the following key components;

- Atmel AT84AD001B dual 8-bit 1GHz digitizer
- Analog devices AD9956 1GHz clock generators
- Altera Stratix II FPGAs
- Freescale 8349E PowerQUICC II Pro PowerPC CPU
- Hot-swap and power sequencing control
- Heat-sinking and thermal monitoring

Each channel of the 1GHz digitizer demultiplexes the 1GHz clock rate 8-bit data by two, and outputs 500MHz clock rate 16-bit data as differential LVDS. An Altera Stratix II FPGA is then used to receive this 500MHz clock rate data, and further demultiplexes the data by 4, producing a 125MHz clock rate 64-bit data stream internal to the FPGA. Each channel of the digitizer is routed to a separate FPGA to ease data pipelining. The digitizer board FPGAs are used to remove the effects of geometric delay, to (optionally) filter the data to bandwidths narrower than the sampled 500MHz analog bandwidth, and to calculate the autocorrelation of the digitized data. Data is transmitted to the correlator boards via front panel LVDS connectors; each digitizer has four front panel connectors, with each connector containing 34 signals which are typically configured to transport either one 32-bit data bus and one clock, or two 16-bit data buses and two clocks. Correlator boards have four front panel LVDS connectors so are able to receive digitized data from 4 or 8 antennas, from which a subset of the 6 or 28 baselines can be calculated. The CARMA Digitizer Board will contain 4 FPGAs, and hence a CARMA Correlator Board will too. These four CARMA Correlator Board FPGAs will be configured to process 16-baselines per board (4 per FPGA). A 500MHz band in the CARMA Correlator System will process data from 15-antennas, calculating 105-baselines of cross-correlations, using 8 digitizer boards and 8 (or perhaps 7) correlator boards.

The CARMA Correlator System boards will be designed as 6U compact PCI (cPCI) boards as used for the COBRA board designs. A CARMA Digitizer board front panel will consist of RF connectors, LVDS connectors, a reset switch, and LEDs, while a correlator board will be similar, but will not require the RF connectors. Photos of the COBRA hardware can be seen at www.ovro.caltech.edu/~dwh/correlator. Figure 1 shows the 6U board layout proposed for the CARMA digitizer board.
Figure 1: CARMA Digitizer. The figure shows the component placement or blocks for the main components on the board, and the main bus widths.
2 System-level design considerations

The correlator system processes a huge volume of data. The averaging used to increase signal-to-noise has the benefit of reducing the data rate to a rate compatible with standard 100Mbit/1Gbit ethernet network data rates. The heirarchy of data processing for a 500MHz CARMA band is:

- Each antenna is sampled at 1GHz to 8-bits; i.e., 8Gbps or 1GB/s per antenna, 120Gbps or 15GB/s for 15 antennas.

- The sampled data for each antenna is received into a digitizer board FPGA as 16-bits at 500MHz.

- The digitizer boards process data and transmit it to correlator boards. For the 500MHz bandwidth mode, the processed 1GHz sampled data is quantized to 4-bits, and demultiplexed by 4 to give a 16-bit data stream at 250MHz clock rate. Each digitizer front panel connector is used to transport two antennas to the correlator boards. Other bandwidth modes produce lower bandwidth outputs, so use LVDS rates slower than 250MHz.

- A digitizer board calculates the auto-correlation of the two antennas it digitizes. The largest auto-correlation is 513-lags, with a 513-channel real-valued spectrum.

- A correlator board calculate the cross-correlation for 16-baselines. The largest cross-correlation is 1024-lags, with a 513-channel complex-valued spectrum. Cross-correlation data consists of both an in-phase and quadrature spectrum, or an upper-sideband and lower-sideband spectrum.

- The 15-antennas in the CARMA array use phase-switching to implement sideband separation. The phase-switch uses a 1024Hz, 1024pps (pulses-per-second), timebase with each antenna Walsh sequence consisting of a 16-state 180-degree Walsh sequence nested inside a 16-state 90-degree Walsh sequence, i.e., a phase-switch sequence completes, and sideband separation is possible, every 256 phase-switch clocks or every 250ms.

- On a board, data can be transferred from the FPGAs to the on-board CPU every Walsh state, i.e., every 1/1024Hz = 977μs, or at the completion of the 16-state 180-degree sequence, i.e., every 16/1024Hz = 15.625μs. The maximum data volume in either case is; 16-baselines×1024-lags×4-bytes/lag = 64kB. This data needs to be transferred off the FPGAs before the next phase-switch sequence completes, and should take no more than about 10-percent of the phase-switch sequence time (since the transfer will be performed to main memory, and the CPU will also need to access main memory to average the new data with previous data), i.e., the bandwidth required for data dumps every phase-switch is 64kB×1024×10 = 640MB/s, and for every 16-states is 40MB/s.

- The CARMA control system expects data every 500ms. The data transfer to the host (cPCI crate CPU) consists of 15 auto-correlations and 210 cross-correlations (two phases or sidebands per cross-correlation). Actually, depending on the hardware configuration the number of cross-correlations calculated in a band can exceed the numbers just stated. Each correlator board can calculate up to 16 cross-correlations, and there will be 8 correlator boards, so the maximum number of correlations calculated is 256 (with some calculated redundantly). Assuming 1024-samples per correlation and 4-bytes per sample, the volume of data transferred every 500ms for 15 auto- and 210 cross-correlations is 900kB, and for 15 auto- and 256 cross-correlations is 1084kB. This data should be transferred to the host in under 100ms, so the bandwidth required over the PCI bus is about 10MB/s. The COBRA boards achieve a 32-bit/33MHz PCI bus bandwidth of 40MB/s, so this bandwidth should be realizable on the CARMA boards, so transfers should take on the order of 25ms.
• The cPCI crate CPUs are typically Force Computers (now Motorola) CPC735-AR2, or Trenton Technologies CPLE, 1GHz x86 CPUs with 100Mbit/s ethernet interfaces. The maximum sustained transfer rate over 100Mbit/s ethernet is about 8MB/s, so the transfer of the 1MB of correlation data collected from all the correlator boards in a band will take about 125ms.

• The control system CPU receives data from the correlator system cPCI crate CPUs using 1Gbit/s ethernet. The control system CPU uses a 1Gbit/s ethernet switch, which in turn is connected to 8 correlator bands using 100Mbit/s ethernet. The combined data from 8 bands, fed to a link with 10 times the bandwidth should not contribute to the system-level latency. However, the 1Gbit/s port on the control system CPU should be reserved for exclusive use by cPCI crate connections.

• Data in the correlator system is collected relative to GPS time, i.e., the 500ms data transfers from the cPCI crates to the control system occur relative to GPS time. Data collected from a 0.0s to 0.5s time segment will undergo processing on the board CPU (e.g., collect the last phase-switch dump, convert to floating-point accumulate with the other averages, and perform an FFT), lets assume that takes 25ms, then will be transferred to the cPCI host (another 25ms, but perhaps slower), and then transferred over ethernet to the control system CPU (125ms), so the control system will receive the data about 200ms after each 500ms boundary.

• The system-level latency can be reduced by reducing the PCI transfer latency and by reducing the ethernet latency. The CARMA boards will use a 64-bit/33MHz PCI interface (providing double the bandwidth of a 32-bit/33MHz interface). The cPCI crate will have spare slots, so the addition of a 1Gbit/s ethernet board is also an option. However, since the control system CPU is receiving data from 8 bands, the latency is unlikely to be reduced by a factor of 10, as the latency will then be dominated by the 1Gbit/s link bandwidth. For example, if it is assumed that the sustained rate over a 1Gbit/s link is simply 10 times that observed for a 100Mbit/s ethernet link, then the bandwidth of the link is 80MB/s. The CARMA boards will be used to implement 5 correlator bands, so the transfer of those 5 bands requires about 60ms. Transfer of 8 bands of data, would require 100ms, so the mixed COBRA hardware/CARMA hardware system will have a latency due to ethernet somewhere in the 60 to 100ms region.

If the control system CPU contains multiple 1Gbit/s links, then modern switches can be configured to bond multiple channels act as higher bandwidth links, e.g. 2 channels could be bonded to create a 2Gbit/s link between the control system CPU and the switch, doubling the link bandwidth (this scheme requires that the cPCI crate CPUs also use 1Gbit/s ethernet so that the data can arrive at the switch faster than the control system CPU can consume it).
3 1GHz Digitizer

The following sections describe the 1GHz digitizer requirements and the Atmel 1GHz digitizer component.

3.1 Digitizer specifications

Aperture jitter is the sample-to-sample variation in the effective point in time at which a sample is acquired by an analog-to-digital converter (ADC) or digitizer. Aperture jitter is expressed as an RMS quantity. Aperture jitter results in voltage errors that are increasingly worse at high-frequencies. These errors result in loss of signal-to-noise, and hence loss in the effective number of bits (ENOB) of an ADC for higher frequencies. If the aperture jitter is to result in a voltage uncertainty of less than half a least-significant bit, then for an N-bit ADC, and a maximum input frequency of \( f_{\text{max}} \) an aperture jitter maximum is

\[
t_a < \frac{1}{2\pi f_{\text{max}} 2^N}.
\]

For example, for the 2-bit COBRA digitizers sampling a 500MHz to 1GHz signal, the aperture jitter needed to be less than \( 1/(2\pi \times 1\text{GHz} \times 2^2) = 40\text{ps}_{\text{RMS}} \). But for an 8-bit digitizer sampling the same band, the jitter requirement drops by a factor of 64 to 0.6ps\(_{\text{RMS}} \). This jitter is the combined jitter of the clock generator circuit and the digitizer jitter.

Aperture jitter is related to the signal-to-noise ratio by

\[
\text{SNR} = -20 \log_{10}(2\pi t_a f)
\]

for a full-scale sinusoid of frequency \( f \). The SNR converts to ENOB via

\[
\text{ENOB} = \frac{\text{SNR} - 1.76}{6.02}
\]

For example, for an input band of 500MHz to 1GHz, with 1ps of clock jitter, the SNR is 44dB and the ENOB is 7.0-bits, while for 5ps of clock jitter, the SNR is 30dB and the ENOB is 4.7-bits.

The CARMA digitizer FPGAs will perform digital downconversion, FIR filtering, digital upconversion, and then quantization to the desired number of output bits. The required number of input bits from the digitizer into the FPGA for each of the possible output bandwidths is 6-bits to 8-bits. The jitter analysis above indicates that a reasonable design target for the total clock jitter should be less than 1ps (providing an ENOB of better than 7-bits). The Atmel AT84AD001B has a typical aperture uncertainty (jitter) of 0.4ps RMS, so the 1GHz clock generator circuit jitter needs to be less than 0.6ps RMS.
3.2 Atmel AT84AD001B (dual 1GHz digitizer)

The Atmel AT84AD001B is a dual 1GHz digitizer, with the following features (http://www.atmel.com/products/Broadband/):

- Dual ADC with 8-bit Resolution
- 1 Gsps Sampling Rate per Channel, 2 Gsps in Interlaced Mode
- Single or 1:2 Demultiplexed Output
- LVDS Output Format (100-Ohm)
- 500 mVpp Analog Input (Differential Only)
- Differential or Single-ended 50-Ohm PECL/LVDS Compatible Clock Inputs
- Power Supply: 3.3V (Analog), 3.3V (Digital), 2.25V (Output)
- LQFP144 Package
- 3-wire Serial Interface
  - 19-bit transactions (3-bit address, 16-bit data), write-only control.
  - 1:2 or 1:1 Output Demultiplexer Ratio Selection
  - Full or Partial Standby Mode
  - Analog Gain (±1.5dB) Digital Control
  - Input Clock Selection
  - Analog Input Switch Selection
  - Binary or Gray Logical Outputs
  - Synchronous Data Ready Reset
  - Data Ready Delay Adjustable on Both Channels
  - Interlacing Functions:
    * Offset and Gain (Channel to Channel) Calibration
    * Digital Fine SDA (Fine Sampling Delay Adjust) on One Channel
  - Internal Static or Dynamic Built-In Test (BIT)
- Low Power Consumption: 0.7W Per Channel
- 1.5 GHz Full Power Input Bandwidth (-3 dB)
- Channel to Channel Input Offset Error: 0.5 LSB Max (After Calibration)
- Gain Matching (Channel to Channel): 0.5 LSB Max (After Calibration)
- Low Bit Error Rate \(10^{-13}\) at 1 Gsps
- Typical aperture uncertainty of 0.4ps (RMS)

The CARMA Digitizer Board will treat the two digitizers as independent samplers, and will provide two clocks from separate 1GHz clock synthesizers (to enable precise clock alignment). The interlaced option can select one of the input clocks, producing a 2GHz sampler, which can be used for lab testing.
Figure 2 shows a block diagram of the AT84AD001B digitizer, while Table 2 lists the pin functions. The control signals to the digitizer use 2.25V logic levels, the input-high voltage, $V_{IH}$, is typically 1.85V, but can be as high as 2.65V, so the system controller operating with 2.5V I/O levels can drive the control signals. The calibration output also uses 2.25V logic, and can be read by the system controller. The digitizer LVDS outputs produce an output-low voltage, $V_{OL}$, of 1.10V (typical) and an output-high voltage, $V_{OH}$, of 1.35V (typical), giving a 250mV differential voltage swing LVDS signal centered on an offset voltage of 1.225V. The offset voltage and swing are compatible with the 2.5V and 3.3V LVDS I/O signalling modes of the Stratix II LVDS I/O pins.

The data ready reset control signal will require some thought. The signal needs to be pulsed to reset the digitizer channels, and that reset needs to occur relative to the 1pps reference with sub-nanosecond accuracy. The system controller can probably generate the signal, but it will likely be driven by the system controller as a single-ended 2.5V signal to an LVDS buffer, and that signal routed to the digitizer (unless one of the smaller FPGA pin banks can be configured for LVDS operation without losing too many I/O pins).
Table 1: Atmel AT84AD001B dual-channel 1GHz digitizer pin functions.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCCA</td>
<td>Analog supply</td>
<td>3.3V, 180mA(max)</td>
</tr>
<tr>
<td>VCCD</td>
<td>Digital supply</td>
<td>3.3V, 350mA(max)</td>
</tr>
<tr>
<td>VCCO</td>
<td>Output supply</td>
<td>2.25V, 215mA(max)</td>
</tr>
<tr>
<td>GNDA, GNDD, GNDO</td>
<td>Ground</td>
<td></td>
</tr>
<tr>
<td>VINI, VINIB</td>
<td>In-phase input</td>
<td>Analog 500mVpp</td>
</tr>
<tr>
<td>VINQ, VINQB</td>
<td>Quadrature input</td>
<td>Analog 500mVpp</td>
</tr>
<tr>
<td>CLKI, CLKIN</td>
<td>In-phase clock</td>
<td>PECL/ECL/LVDS, 600mVpp</td>
</tr>
<tr>
<td>CLKQ, CLKQN</td>
<td>Quadrature clock</td>
<td>PECL/ECL/LVDS, 600mVpp</td>
</tr>
<tr>
<td>DDRB, DDRBN</td>
<td>Data ready reset</td>
<td>PECL/ECL/LVDS, 600mVpp</td>
</tr>
<tr>
<td>MODE</td>
<td>Operating mode (and reset)</td>
<td>2.25V logic input</td>
</tr>
<tr>
<td>LDN</td>
<td>3-wire serial select</td>
<td>2.25V logic input</td>
</tr>
<tr>
<td>CLK</td>
<td>3-wire serial clock</td>
<td>2.25V logic input</td>
</tr>
<tr>
<td>DATA</td>
<td>3-wire serial write-only data</td>
<td>2.25V logic input</td>
</tr>
<tr>
<td>DOAI[7:0], DOAIN[7:0]</td>
<td>In-phase output data port A</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>DOBI[7:0], DOBIN[7:0]</td>
<td>In-phase output data port B</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>DOIRI, DOIRIN</td>
<td>In-phase output data in-range</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>CLKOI, CLKOIN</td>
<td>In-phase output data ready</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>DOAQ[7:0], DOAQN[7:0]</td>
<td>In-phase output data port A</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>DOBQ[7:0], DOBQN[7:0]</td>
<td>In-phase output data port B</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>DOIRQ, DOIRQN</td>
<td>Quadrature output data in-range</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>CLKOQ, CLKOQN</td>
<td>Quadrature output data ready</td>
<td>2.25V LVDS</td>
</tr>
<tr>
<td>CAL</td>
<td>Calibration status</td>
<td>2.25V logic output</td>
</tr>
<tr>
<td>DIODE</td>
<td>Temperature diode</td>
<td>voltage output</td>
</tr>
</tbody>
</table>
3.3 RF input signal conditioning

The COBRA digitizer boards required an input RF amplifier to improve the impedance match to the digitizer. Its likely something similar will be needed on the CARMA boards (if only to act as an ‘input fuse’, an amplifier is easier to replace than a digitizer!).

I believe the Atmel digitizer has a power sequencing requirement; I/Os can not be driven until the digitizer is powered. The digitizer uses multiple power supplies, so there is a chance that those supplies also have a sequencing requirement. I/Os to the digitizer include the RF input and the control signals. The RF input will likely require an RF switch after the RF amplifier (experience with the COBRA boards shows that shutting off the RF amplifier power supply does not shut off the amplifier output). The I/O controls from the system control FPGA should be tristated by a power-good signal that originates from the digitizer. The 1GHz clocks will also need some form of on/off control. Perhaps the VCO output can be buffered by a tri-stateable driver.

3.4 Quantization state monitoring

The COBRA boards used a 2-bit digitizer which required precise control of the digitizer threshold voltages to achieve optimal signal-to-noise. The CARMA board 8-bit digitizers will also require some form of quantization state monitoring, but will likely have less flexibility with respect to threshold control.

3.5 Threshold control

The COBRA Digitizer boards route 2-bits from the digitizer into the FPGAs. For optimal signal-to-noise in the cross-correlation estimates, the fraction of counts in the four output bit patterns from the digitizer must be maintained at a certain fraction. The desired fractional count was achieved in COBRA by measuring the quantization states, and then adjusting the threshold voltages on the digitizer until the required fractions were met.

The ability to control the digitizer transfer function was fairly unique on the COBRA board SPT7610 digitizer. Similar control on a digitizer that did not have a method for controlling thresholds would be to have an adjustable gain section in front of the digitizer (or as part of the digitizer).

In the CARMA digitizer, since all 8-bits will be routed to the FPGA, as long as the input voltage swing is kept within a factor of four of the maximum (i.e., from 8-bits down to 6-bits), threshold control could be performed digitally. However, if the signal level drops below 6-bits, then signal-to-noise will be lost.

3.6 Synchronization

The digitizers on a board, and the digitizers between boards will need to be synchronously reset. The objective is to use the 1pps time reference and the 10MHz reference to generate a single 10MHz period pulse (100ns long) aligned with the 1pps edge. Given that the 1pps signal is synchronized to the 10MHz signal, the synchronizing pulse should be seen at the same time on all boards. The board-to-board delays will result in a skew, but this can be corrected for by adjusting clock phases.

The timing of the digitizer reset pulse will likely require sub-nanosecond alignment relative to the 10MHz reference. This may preclude using a general purpose I/O on the system controller FPGA for generating this signal (since the clock-to-output uncertainty could be too great). One of the FPGA specialized DDR control channels might work, since those channels have some form of timing adjustment in them.
3.7 Thermal monitoring

The COBRA digitizer boards exhibit a digitizer delay coefficient of about 40ps/degree-C. It’s likely that the Atmel digitizers will experience a similar effect (this feature is not documented in digitizer data sheets).

The AT84AD001B contains an on-die diode-connected transistor for use in temperature monitoring. p44 of the data sheet shows the diode connection, and shows that it requires an external 1mA current source. p45 shows the diode voltage versus junction temperature, i.e., for the temperature range 20-degrees-C to 125-degrees-C the voltage varies by about 155mV, or about 1.5mV per degree-C. The diode monitoring device should provide 0.5-degree-C accuracy (or precision anyway).

Do the MAXIM temperature sensor devices contain the current source, ADC etc. What does the evaluation board use to measure the on-chip diode?
3.8 Notes

- Determine the I/O sequencing requirements. I recall reading that the I/Os cannot be powered if the chip core voltage is not powered. This would mean that the analog RF requires a switch, and the clock requires a tri-state (of just turn off the clock power plane). The maximum ratings show that VCCA and VCCD must track to within 0.8V. The data sheet doesn’t seem to have any other comments regarding sequencing, so check out the application notes for the evaluation board.

- The 3-wire serial control interface looks non-standard, and uses a 2.25V supply, so should be controlled from the system controller FPGA using 2.5V logic. pp40-41 of the data sheet shows the timing. A transaction requires 20 clocks, but additional clocks after that point don’t matter. So, it would be possible to run the PowerQUICC SPI interface through the FPGA, and just use the FPGA as a logic level translator.

- Power sequencing.

- The built-in-test for static mode gives you the option of using the 3-wire serial interface to program the 8-bit output data from the digitizer. This would enable a manual sweep over the full digitizer output 8-bit range. Nice.

- The built-in-test dynamic mode outputs a ramp (0 to 256?). But this mode only operates at up to 750MHz. Is there an easy way to create a divided clock? Use a divide-by-1 or divide-by-2 divider in the feedback loop, or a 2GHz VCO and a divider with divide by 2 and divide by 4 option? The digitizer could then be fed a 1GHz or 500MHz clock.

- How about a fanout chip with internal divider, the feedback path could always run with the same clock, but the output could be divided by 2; problem, division outside the loop. If the divider were tri-statable, then that would satisfy the I/O sequencing requirement.

- Perform basic tests using the evaluation kit.

- Interface to the Stratix II DSP kit.

- Look at the internal clock and calibration control features, can I use them for threshold control or clock alignment?

- I had notes some about a nice compact way to generate 8-bit histogram logic (i.e., a histogram with 256-columns). The 2-bit digitizer needed only four columns, i.e., four parallel counters.

- Look at the Atmel digitizer gain control; ±1.5dB is a total gain of 3dB, i.e., a factor of 2, so 1-bit of gain control.

- Look at the Howard Johnson/Bob Pease tutorial for the National Semi 1GHz ADC interfaced with a Xilinx FPGA. The Xilinx XCell journal has an article, and National has an online seminar or tutorial. Look at their app notes. The CARMA Digitizer can’t use the National part since the clocks to the dual digitizer are not independent.
4 1GHz Clock Generation

The 1GHz clock generated to the digitizers in the correlator system must be coherent across digitizer and correlator boards in the same band, i.e., the board clocks must be generated from a common reference. The most convenient way to implement a coherent clocking scheme is to distribute a low-jitter 10MHz reference and use phase-locked loops (PLLs) to generate the higher clock frequencies used on the digitizer and correlator boards.

When the correlator system is powered up, the clock phases across multiple boards will be locked, however, the clock edges will not be aligned (due to variable propagation delays through parts, and clock divisions in various circuits). To align clocks across multiple boards, some kind of 1GHz clock delay (phase) adjustment control is required.

4.1 COBRA board clock delay control

To align the digitizer and correlator board clocks in the COBRA system, the COBRA boards contain clock delay circuits and phase detector circuits (phase detectors that are not part of a PLL). These two components are used to align clocks using software. The clocks are first coarsely aligned on individual digitizer boards, and between digitizer and correlator boards. Coarse alignment is performed using the phase detectors to provide a coarse measure of the phase between clocks, and the delay circuits are used to adjust the clock phase. Coarse alignment aligns the clocks to within a few nanoseconds. The clock alignment is then ‘fine-tuned’ by looking at the noise source and adjusting clock phases until the cross-power noise spectra have flat (±50ps) phase.

The clock delay circuits on the COBRA boards consist of coarse delay control with an Analog Devices AD9501 delay line, and fine delay control with an OnSemi 100E195 ECL delay line. The COBRA boards use a 15.625MHz reference and multiply that by 64 to produce the 1GHz digitizer clock. The reference frequency choice was determined by the 1GHz PLL which could only use integer multiplier values. Unfortunately 15.625MHz is not a standard reference frequency, so the oscillators are custom-ordered parts, and they tend to have higher jitter than a standard reference such as 10MHz.

The COBRA board AD9501 delay line is used in the 15.625MHz reference path. The AD9501 has a nominal delay step size of 130ps and can be programmed in 256 steps (a total delay of 33ns). The 1GHz digitizer clock is generated using a PLL, and the 1GHz clock is passed through the 100E195 delay line which has 10ps of delay precision. This part has a delay range of 128 steps, however, only the first 16 steps can be used without causing clock glitches. This is acceptable since fine steps up to 130ps can be programmed using the ECL delay line, and coarse steps greater than 130ps can be programmed using the AD9501. The delays in the AD9501 and 100E195 parts were not particularly consistent between parts. To avoid having to calibrate the delay lines on the boards, the clock alignment algorithm uses nominal delay values, and iterates until the slope of the cross-power spectra phase is within ±50ps.

4.2 CARMA board clock delay control

The CARMA digitizer boards will generate their 1GHz clocks using an Analog Devices AD9956 DDS-based AgileRF synthesizer. This is a highly integrated device containing PLL logic and DDS (direct digital synthesis) logic. Figure 3 shows a block diagram of how the AgileRF part will be configured to generate a 1GHz clock. The phase detector output (a charge-pump) generates current pulses which are filtered to produce a voltage to an external 1GHz VCO. The output of the VCO is the 1GHz digitizer clock, and is fed back into the AgileRF synthesizer. The feedback signal is divided-by-4 to create a 250MHz signal to clock the DDS (which can operate at a maximum of 400MHz). Phase-lock in the circuit is achieved by programming the DDS such that it outputs the same frequency as the reference. With the DDS in the feedback path, it is possible to insert phase...
Figure 3: Analog Devices AD9956 DDS-Based AgileRF Synthesizer. The AD9956 can be used as a PLL to generate a 1GHz reference for the digitizer. The DDS in the feedback path of the PLL provides phase control of the 1GHz signal.
	offsets, i.e., delay the 1GHz clock. Since this ‘delay’ is created digitally, it is precise, and no part-to-part differences will exist. This feature will be used to align the digitizer clocks on a board, and between boards.

Analog Devices AD9956 AgileRF Synthesizer features (www.analog.com/agilerf):

• 400MSPS internal DDS clock speed
• 48-bit frequency tuning word
• 14-bit programmable phase offset
• Integrated 14-bit DAC
• Serial I/O control
• 200MHz phase frequency detector (PFD) inputs
• 655MHz PFD input dividers
• 2.7GHz Programmable RF divider
• 8 phase/frequency profiles
• Linear frequency sweeping capability in DDS
• Multi-chip synchronization

The DDS generates a sinusoidal waveform by generating a 48-bit lookup (0 to $2\pi$) into a sine table. Different frequencies are generated by programming a step size, or frequency tuning word (FTW). The base frequency generated at the DDS output is

$$f_{\text{dds}} = \frac{\text{FTW} \times f_{\text{clk}}}{2^{48}} \quad 0 \leq \text{FTW} \leq 2^{47} \quad (4)$$

For example, if the reference input in Figure 3 is 10MHz, then the DDS needs to be configured to generate a 10MHz output. With a 250MHz DDS clock frequency, the FTW is

$$\text{FTW} = \frac{f_{\text{dds}} \times 2^{48}}{f_{\text{clk}}} = \frac{10\text{MHz} \times 2^{48}}{250\text{MHz}} = \frac{2^{48}}{25} = 11258999068426.24 \approx 11258999068426.24. \quad (5)$$
When the AgileRF synthesizer is locked $f_{ref} = f_{dds}$ and $f_{clk} = f_{vco}/R$ (where $R$ is the RF divider setting), so (4) can be rewritten in terms of the VCO output frequency

$$f_{vco} = \frac{R f_{ref} 2^{48}}{\text{FTW}}$$

where for the CARMA 1GHz generator the RF divider $R = 4$. Since only the integer portion of the FTW calculation for a 10MHz reference can be loaded into the DDS register, there is a frequency error at the 1GHz VCO output of

$$f_{vco} - 1GHz = \frac{4 \times 10MHz \times 2^{48}}{\text{FTW}} - 1GHz = 21.3 \mu Hz$$

So the 1GHz PLL frequency is $1GHz + 21.3 \times 10^{-6}Hz$, with a period of $1ns - 2.13 \times 10^{-23}s$.

Is this frequency or period error acceptable? First, consider the 1GHz PLLs; if two 1GHz PLLs are locked, then will their relative phases change over time? No, they will both be configured identically and will lock identically (they may have a relative phase, but it will not change over time). The same argument holds for the relative phase of the FPGA PLLs. Now consider the phase of the FPGA PLLs relative to the 1GHz PLLs; if the FPGA PLLs are configured for 125MHz operation, then they will phase-lock to exactly 125MHz, with exactly an 8ns period. This means that total time for eight periods of the 1GHz clock will be shorter than one period of the FPGA 125MHz clock by $8 \times 2.13 \times 10^{-23}s = 1.70 \times 10^{-23}s$. Over time this period error will accumulate to be a significant fraction of the FPGA PLL period; it will take $8 \times 10^{-9}/2.13 \times 10^{-23} = 375 \times 10^{12}$ 1GHz PLL periods for the digitizer clock to slip by 8ns, i.e., about 375,000s or 4.3 days. The solution to the slight difference in clock phases is to ensure that the digitizer output clock is used to clock the FPGA deserializer logic and to clock the deserialized data into a FIFO, and that the FPGA PLL clock is used to clock data out of the FIFO. FIFOs were going to be needed anyway for clocking between the digitizer and the FPGAs, since the digitizer clock phase can be arbitrarily adjusted relative to the 10MHz reference using the AD9956 DDS in the feedback loop (this type of phase adjustment is used to align all the 1GHz digitizer clocks by adjusting individual clock phases until the noise-source phase is flat, and it can also be used for sub-sample delay tracking).

What is the effect of a difference in clock frequencies for FIFO writer and reader clocks? The operation of a FIFO starts with the writer clocking data in, followed by a FIFO not-empty flag being asserted. The reader uses the not-empty flag to determine when it can read. The not-ready flag is generated with respect to the input clock domain, and then synchronized to the output clock domain, so by the time the reader sees not-empty asserted, there may be more than one data item in the FIFO. When both input and output clocks are the same, the FIFO level remains constant. In the more general use of a FIFO when clocks are different, either the reader will read faster than the writer writes and the FIFO will empty, or the writer will write faster than the reader reads, and the FIFO will full. There are FIFO empty and full flags that user-defined FIFO-control state machines use to detect these conditions.

In the digitizer-to-FPGA application of FIFOs, given that the digitizer clock is slightly higher than 1GHz, the FIFO writer is faster than the FIFO reader (the FPGA). So, assuming that the digitizer clock and FPGA clock can be reset to a known phase state (lets assume they are aligned), then once the FIFO indicates to the FPGA logic that it is not-empty, the FPGA will start streaming data from the digitizer. Lets assume that after not-empty assertion, there are two data values in the FIFO. After 4.3 days the digitizer clock will have accumulated enough phase that an extra write will occur before an FPGA read, and the FIFO will then contain three data values. The FPGA FIFO depth is a compile time parameter, and the FIFOs are generally implemented using RAM blocks, so their depth is often larger than is required. Lets assume its 16 deep. So, starting with a FIFO depth of 2 just after reset, it would take 14 times 4.3 days, or 60 days of continuous operation of the correlator before a FIFO full condition occurred. Since the correlator clocks will be reset on a daily basis (eg. every time a track is started), the logic will never experience a FIFO full condition. If the
1GHz FTW was rounded such that the clock was slower than 1GHz, then a FIFO empty condition would occur after 4.3 days. That could also be dealt with by holding off the FIFO not-empty signal until the FIFO was half-full. Using a clock frequency slightly higher than 1GHz is simpler to deal with.

The slight error in the AD9956 1GHz output can be eliminated by using a reference that does not cause an FTW rounding error. For example, the COBRA correlator bands each use a 15.625MHz reference on the timing board in each crate. The use of that frequency as a reference causes no FTW rounding. The discussion in this section has focused on using a 10MHz reference, as that is the frequency the CARMA master clock generates, and is also the clock that the 1pps signal is synchronized to. If the CARMA digitizer/correlator board clocking scheme uses the 1pps signal to generate a globally synchronized PLL reset scheme, then either the board PLL reference needs to be the master clock 10MHz reference, or the 10MHz reference needs to be used to lock a low-jitter synthesizer to 15.625MHz, and that 15.625MHz used as the board PLL reference.

The phase of the DDS output can be offset using the 14-bit DDS offset register. For a DDS output frequency of 10MHz, dividing the 100ns period into 14-bit steps represents a step size of $\frac{100\text{ns}}{2^{14}} = 0.61\text{ps}$, i.e., at a reference frequency of 10MHz, the AgileRF can implement a 100ns delay span in 6.1ps steps. If the reference frequency is 15.625MHz, then the frequency tuning word is a whole integer (so there is no frequency error), the delay span is 64ns, and the delay step size is 3.9ps. These numbers exceed the precision on the COBRA boards, and in addition are ‘exact’ since the delays are generated digitally.

### 4.3 AD9956 AgileRF synthesizer

Figure 4 shows a block diagram of the AD9956 showing the pin functions of the device. Also shown in the figure is the support logic to generate a 1GHz clock to the digitizer. The digitizer test modes have a maximum operating frequency of 750MHz, and the clock to the digitizer needs to be disabled during power sequencing, so the 1GHz VCO and 250MHz CML signals are routed to an RF switch (Hitite HMC349MS8G/349MS8GE) which enables the selection of either clock signal, or the disabling of its output. Table 2 also lists the AD9956 pin functions.

Figure 4 shows the 1GHz clock to the digitizer as a single-ended signal. The Atmel documentation indicates that there is no loss in performance (increase in jitter) if the clock is driven single-ended (evaluation board doc5317.pdf p48, and doc5459a.pdf). However, there if the path from the AD9956 to the digitizer is long, then a balun may be desirable on the output of the RF switch, as then the clock routing to the digitizer would be differential.
Figure 4: Analog Devices AD9956 AgileRF synthesizer block diagram showing the device pin functions, and surrounding support components. The synthesizer is shown driving a 1GHz VCO. The digitizer clock source is either disabled, or selected as the 1GHz VCO output or the CML output (250MHz) via an RF switch. The 250MHz option is used for the digitizer test modes.
Table 2: Analog Devices AD9956 AgileRF synthesizer pin functions.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVDD</td>
<td>Analog core supply</td>
<td>1.8V, 85mA (max)</td>
</tr>
<tr>
<td>DVDD</td>
<td>Digital core supply</td>
<td>1.8V, 45mA (max)</td>
</tr>
<tr>
<td>DVDDIO</td>
<td>Digital I/O supply</td>
<td>3.3V, 20mA (max)</td>
</tr>
<tr>
<td>CP_VDD</td>
<td>Charge-pump supply</td>
<td>3.3V, 15mA (max)</td>
</tr>
<tr>
<td>AGND, DGND</td>
<td>Ground</td>
<td></td>
</tr>
<tr>
<td>PLLREF, PLLREF#</td>
<td>PLL reference input</td>
<td>Analog 200-600mVpp</td>
</tr>
<tr>
<td>PLLOSC, PLLOSC#</td>
<td>PLL feedback input</td>
<td>Analog 200-600mVpp</td>
</tr>
<tr>
<td>IOUT, IOUT#</td>
<td>DAC analog output</td>
<td>10-15mA full-scale</td>
</tr>
<tr>
<td>CP_OUT</td>
<td>Charge-pump output</td>
<td>Current selectable to 4mA</td>
</tr>
<tr>
<td>REFCLK, REFCLK#</td>
<td>2.7GHz RF divider input</td>
<td>Analog 350-1000mVpp</td>
</tr>
<tr>
<td>DRV, DRV#</td>
<td>622MHz CML driver output</td>
<td>750mVpp</td>
</tr>
<tr>
<td>CP_RSET</td>
<td>Charge-pump current set</td>
<td></td>
</tr>
<tr>
<td>DRV_RSET</td>
<td>CML driver current set</td>
<td></td>
</tr>
<tr>
<td>DAC_RSET</td>
<td>DAC current set</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>Device reset</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>IO_RESET</td>
<td>Serial interface reset</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>CS#</td>
<td>Chip select</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial clock</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>SDI/O</td>
<td>Serial data in or out</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>SDO</td>
<td>Serial data out</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>SYNC_OUT</td>
<td>Synchronization clock output</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>PLL_LOCK/SYNC_IN</td>
<td>PLL lock detect, or</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td></td>
<td>synchronization clock output</td>
<td></td>
</tr>
<tr>
<td>I/O_UPDATE</td>
<td>Update control</td>
<td>LVCMOS logic input</td>
</tr>
<tr>
<td>FS[2:0]</td>
<td>Profile select</td>
<td>LVCMOS logic input</td>
</tr>
</tbody>
</table>
4.4 Notes

- Take a look at the AD9956 and FPGA PLL signals on a scope with the PLLs locked to the same 10MHz reference. Try restarting each PLL independently. Leave things running for a day and see that the phases have changed.

- The fact that the digitizer clock frequency is not exactly 1GHz causes a slight error in the secondary lobe rotation calculation if the nominal value of 1GHz is used. The actual value can be used to get rid of this small error. The fact that all digitizer PLLs lock to the same frequency means that there are no digitizer-to-digitizer phase errors, which would be the case if two digitizers had their 1GHz clocks generated with different methods. This can be shown using the AD9956 by using an FTW on one that is rounded up, and an FTW on another that is rounded down.

- Add details on the Sirenza and Minicircuits 1GHz VCO part numbers.

- DAC output filter design

- Charge-pump output filter design

- Is a balun really required in the REFCLK feedback path? What would the jitter effect be using a single-ended 1GHz signal from the splitter?

- What’s the termination required in the CML DRV# signal?

- I looked at CML drivers and multiplexers from Micrel. Even the 7GHz parts had a jitter spec of 10ps RMS, so exceed the jitter required for the digitizer clock.

- What is the VCO output power? The REFCLK has a -5dBm to 4dBm range, so losing 6dBm through the splitter could be tolerated without needing an amplifier. I think the digitizer clock input power is -9dBm to 6dBm, so it also has the range to lose some power through a splitter. Eg. if the VCO output power is 6dBm, then the REFCLK and digitizer clock would see 0dBm.

- The AgileRF data sheet has information on synchronization between multiple devices. The clock being discussed is SYNC_CLK, but its not clear yet whether this is a clock we care about (assuming we have pins, it’ll be routed to the system controller, but perhaps it should go to the digitizer FPGA receiving the digitized data). Its quite possible that with the DDS in the feedback loop that this clock is inherently synchronized between multiple devices, and the datasheet is referring to the more general case when the DDS might not be in the feedback loop.

- Test the two evaluation boards to confirm that I can control the relative phases of two 1GHz clocks appropriately, and at that time I can look at the other clock synchronization issues.

- Use the evaluation board to confirm the pins required for the control interface, and the pins that are optional.

- AgileRF $17.24 (qty 1000-5000, Aug19th, 2004) ea PLUS external VCO / VXCO PLUS filter

- The control interface is like SPI, but it has a variable word-width; an 8-bit instruction (address plus read or write), followed by a multi-byte command. Since the commands are in integral bytes, an SPI controller in 8-bit/16-bit/32-bit mode can probably be used to control the device. What options does the PowerQUICC SPI interface have? It would probably be best for the system controller FPGA to control this interface, as it could do so with higher timing precision if required.
• If used, the phase-profile select pins need to be synchronized to the phase switch heart-beat, so should originate from the system controller. I think these signals are latched by the I/Q UPDATE control relative to the SYNC CLK, so check the timing required for those signals.

• The 1GHz digitizers have a divide-by-two in their outputs. This divide-by-two is unsynchronized since its outside the PLL loop - does this matter? Well, it means that the cross-correlation of two digitizers can vary by 1ns depending on when their ping-pong buffers start outputting. Correlating against the noise source fixes this, but its just not 'perfect'. If the digitized divided by two output was made part of the PLL loop, then all digitizers will be in the same state. However, the digitizer divide-by-2 may introduce too much jitter. Anyway, its probably worth putting a 2:1 multiplexer in the feedback path to select between the two clocks - the mux can be pulled off to see if it contributes to jitter. This gets worse, since the FPGA deserializers introduce another factor of 4. This is all stuff that I'll work on in a section on clocks.

• What is the 1GHz clock jitter? This will be a function of the AgileRF device and the 1GHz VCO it drives. I may need to look into whether a VCO or a VXCO is appropriate.

• Jitter testing: What effect does the charge-pump gain setting resistor have, what kind of filters are needed to reduce spurs, etc etc.

• Testing: I could put in a phase slope as a clock delay and then take it out using an FIR filter with an equivalent phase slope.
5  **Stratix II FPGA**

Figure 5 summarizes the processing requirements of the CARMA Digitizer board:

- Interface to 16-bit 500MHz data from the 1GHz digitizer.
- 180-degree phase switch demodulation (1ms rate).
- Data demultiplexing (to 125MHz)
- Digital delay line at demultiplexed input rate (for 1ns delay steps)
- Digital downconversion with frequency and phase control (numerically controlled oscillator or DDS). Phase control for lobe-rotation (phase offset) correction.
- Asymmetric FIR filtering (flat spectral magnitude with a phase slope) for sub-nanosecond delay correction.
- Quantization state monitoring (histogram of up to 8-bit sampling).
- Requantization of data to 2-bits, 3-bits, etc.
- LVDS transmission (digitizer) and reception (correlator) of front panel data.

The Stratix II FPGA data sheet indicates that the FPGA can accept data at up to 1Gsps. The data is likely to be received by the FPGA in double-data-rate (DDR) mode, where both the rising and falling clock edge of a 500MHz clock is used to capture data onto the FPGA. To interface to a 500MHz data stream from the FPGA then, a 500MHz single-data-rate would be possible, or to be more conservative, a 250MHz DDR interface. The 250MHz DDR clock used by the FPGA will need to be derived from the digitizer clock, or from the digitizer clock generator circuit. The details of the data and clock interface will be determined with the help of the digitizer, clock, and FPGA evaluation boards.

Figure 1 shows a block diagram of the CARMA digitizer board. The four digitizer FPGAs will require the following buses: 32-bit LVDS front-panel transceivers, 16-bit LVDS digitizer data receivers, 128-bit digitized data inter-FPGA buses, and a 32-bit processor interface. The FPGA digitizer data buses will operate at up to 125MHz or possibly 250MHz, while the processor interface bus will operate at around 100MHz. Multi-drop buses are difficult to operate at much above 50MHz (due to transmission line reflections and load capacitances). Figure 1 shows that the 100MHz processor bus is pipelined through each FPGA. The FPGA bus traces will all be short and point-to-point. PCB layout of the traces should be fairly simple.
Figure 5: Digitizer FPGA functions. Digitized data is received by the FPGA and demultiplexed into a 64-bit wide internal bus (not all samples will be needed in all bandwidth modes). The digitized data is passed through a digital delay line (FPGA memory) to delay data in 1ns steps. The output of the delay line is downconverted to a complex-valued baseband signal where it is FIR filtered and decimated. The output of the FIR filter is remodulated to half the bandwidth of the signal, converted to a real-valued signal, and quantized to the appropriate output bit width. Data is then routed to the front panel LVDS cables and to the digitizer board auto- and cross-correlation logic.
5.1 FPGA Pin assignments

The basic requirements for a CARMA digitizer FPGA are:

- Digitizer input: 16-bits LVDS input at 500MHz clock rate.
- Front panel LVDS: 32-bits LVDS output/input at 125MHz clock rate.
- 128-bit data path between FPGAs.
- CPU interface: 32-bit multiplexed address/data bus pipelined through the FPGA (64-pins + control signals required)

The device package selected for the CARMA FPGAs is the 1020-pin package. There are four devices available in this package; EP2S60, EP2S90, EP2S130, and EP2S180. The choice of package was based on the required number of I/Os, the device density options, and that this package has the maximum number of PLL channels. The EP2S60 is the smallest device, and its functionality should be a subset of the larger devices, the remainder of this section analyzes the EP2S60. The pins on the Stratix II devices are divided into a number of banks. Pins within a bank can be used to implement different single ended or differential I/O standards. There are some restrictions as to mixing standards in banks. The Stratix II databook contains figures of the device I/O banks.

Figure 6 shows the pin functions on the EP2S60 device. The LVDS I/O banks are on the left and right sides of the device, while the non-LVDS banks are on the top and bottom of the device. Figures 7 and 8 show the LVDS and single-ended bus assignments required by the CARMA Digitizer Board as shown in Figure 1. Note that clock and control signal assignments are missing from the figures, so the assignments are incomplete. The point of the figures is to show that even with a 1020-pin device most of the available I/O pins will be used.

TODO: revise, and add databook references to, the following comments.

Four pin banks of the EP2S60 support LVDS; the banks are located on the left and right sides of the device (banks 1, 2, 5, and 6). Each LVDS pair is either a transmitter or receiver. The transmitter pair has a serializer component, while each receiver pair has a deserializer component. Each bank contains 21 serializers and 21 deserializers. Each bank of 21-channels can be driven by a Fast PLL (FPLL), banks 1 and 2 on the left of the FPGA can be driven by a left-center FPLL, and the same can be done with the right-side banks 5 and 6, and the right-center FPLL.

The FPGA interface to each 1GHz digitizer requires 16-bit data plus clock at 500MHz clock rate. The FPGA deserializer component can be configured to receive this data stream and then demultiplex each bit to 4-bits, i.e., the 16-bits of input deserializers would produce 64-bits at 125MHz internal to the FPGA. The 16-bits of digitizer data need to route to pins within the same bank, and the clock from the digitizer would need to route to a clock pin that can drive an FPLL within that bank (top or center). A single FPGA pin bank (one of banks 1, 2, 5, or 6) is adequate to interface to one digitizer channel. The dual-1GHz digitizer can be interfaced to the side of one FPGA, or more likely will be split such that one digitizer output goes to one FPGA and the other goes to another FPGA (minimizing the inter-FPGA bus requirements). The data receive logic has a number of data alignment and clock phase control options that will be investigated on the Stratix II Development Kit. The FPLL has a number of programming modes that also need to be tested.

The COBRA boards have four front panel connectors that carry 8 16-bit digitized data streams at 125MHz using LVDS format. The LVDS interface uses transceivers, i.e., boards can both transmit and receive data. To implement an identical interface on the revised boards, 16+1 channels of transmitters and receivers within a bank can be used for each 16-bit plus clock output. Two banks would be required per digitizer FPGA to drive the 32-bit LVDS cable. The transmitter and receiver
Figure 6: Stratix II EP2S60 1020-pin package pin functions.
Figure 7: Stratix II EP2S60 digitizer FPGA LVDS I/O assignments. The left-hand LVDS transmitters and receivers will be used to implement the 32-bit 250MHz clock rate bidirectional front panel LVDS interface (128 pins required). The right-hand LVDS receives will be used to interface to the 16-bit 500MHz clock rate digitizer data bus (32-pins required). The pin assignments shown allow for Figure 1 FPGA1 to receive digitized data on LVDS receivers near the bottom of the FPGA, while FPGA2 can receive data near the top of the FPGA. If I/O becomes limited, then one bank can be used for LVDS, and the other used for single-ended I/O signals.
Figure 8: Stratix II EP2S60 digitizer FPGA I/O bus assignments. The pins are grouped into blocks of 32-bits. The first four blocks from the left will be used for inter-FPGA data pipelining, while the last block will be used for the processor interface.
pins on the FPGA would be daisy-chained such that each pair could be used as a transmitter or receiver.

The pin banks 3, 4, 7, and 8 support single-ended I/O. These banks do not support LVDS, but they do support several types of differential I/O. The choice between using single-ended or differential would need to consider radiation and more importantly power consumption. If differential signaling was used then to match the bus bandwidth of single-ended you have to operate half the number of connections at twice the clock frequency. The routing of differential signals is also more critical, and routes must occur between pairs of the correct polarity. And finally, the potential show-stopper, not every differential pair can be configured as output or input, they are either transmitter or receiver (however, data is typically sent and received from the FPGAs, so this just reduces the available choices). The pin banks 3, 4, 7, and 8 contain 84 I/O pins (or 21-differential transmitters, and 21-differential receivers). Since there are two banks on the top and bottom of the FPGAs, this gives 168-pins on a side. These pins would be used for the 128-bits inter-FPGA routing and the CPU interface.

The data rate between FPGAs can be doubled using double-data-rate (DDR) techniques. Whether DDR is supported on all pins in banks 3, 4, 7, and 8 needs to be investigated (there may be special clocking requirements that need to be included on the board layout, eg. clock routing).

The EP2S60 also contains smaller pins banks; 9, 10, 11, and 12. These banks are used for clock inputs and outputs, and contain only 6 to 8 pins. The pins can be configured as single-ended I/O.
5.2 FPGA Power estimates

From Kevin: 5/26/06

Correlator FPGA simulation (including correlation logic, all the I/O buses and a PLL). Board total power:

VCCINT ~ 40 A @ 1.2 V  
VCCIO ~ 2.5 A @ 2.5 V  
VCCPD ~ 200 mA @ 3.3 V

This was for worst-case I/O where every bit of every bus toggled every clock cycle (125 MHz), and four 6x resolution baselines on the chip. About half the VCCIO current went to the front panel LVDS I/O banks. Leakage current was ~1.3 A @ 60 C. The PLL and clock routing together used less than 1 W.

Using 1.5 V I/O:

VCCINT ~ 40 A @ 1.2 V  
VCCIO ~ 0.7 A @ 1.5 V (inter-FPGA)  
VCCIO ~ 1.3 A @ 2.5 V (LVDS)  
VCCPD ~ 200 mA @ 3.3 V

The MAXIM MAX5065 multiphase synchronous buck converters and the Intel/AMD VRM-compatible converters can deliver 60A at 1.2V, so VCCINT generation for four FPGAs should not be an issue. The VCCIO supply is easily handled by a single-phase synchronous buck converter. The VCCPD can just use a linear regulator off the cPCI 5V.

Design option: use per-FPGA linear regulators or integrated dc-switchers. The lower-current might allow the use of parts that can be loaded on the back of the PCB. The use of linear regulators will depend on power dissipation, while the use of switchers will depend on the support passives, eg. inductors.
5.3 Stratix II DSP Kit

The Stratix II DSP kit is an Altera development board containing an EP2S60, SDRAM, SRAM, Flash, ADCs, DACs, and a MAX CPLD (for configuration loading). The kit has numerous I/Os (headers on the top of the PCB, and two connectors on the bottom).

The following are several tests the kit might be used for:

- PLL configuration, external waveform observation (to confirm relative timing of reference and outputs), and control tests; PLL enable, reset, clock enable, etc.

- Deserializer configuration (this in not possible in LVDS mode, as LVDS requires 2.5V VCCIO, and the board uses 3.3V. Is there a 3.3V mode?)

- 100MHz clock-rate correlator (using the onboard ADCs) (test FIR filter, digital downconversion, delay correction)

- Power consumption (versus tool predictions)

- Interfacing to the PowerQUICC (perhaps at a reduced clock rate)

- System controller state machine testing, eg. the fast passive parallel (FPP) configuration controller could be tested using one kit to program the FPGA on another.

- System controller to FPGA interface; either by creating multiple internal buses in the FPGA, or connecting two kits together. A ModelSim simulation of the interface would also be adequate.

- The diode temperature sensor pins are available on two single-pin headers, TM1 and TM2. Get a MAXIM device and test reading the temperature.

The reference kit is also a good example of FPGA circuit design and component selection (the following subsystems are of interest to the CARMA design)

- 16-layer board with 10 signal layers, 6 power/ground planes. Powered from a well-regulated 16V supply (p11 May 2005 datasheet). VCCINT 1.2V, VCCIO 3.3V, and VCC5.

- Power supply components:
  - LTC1778 1.2V (synchronous buck) VCC1,2
  - LTC1778 5V (synchronous buck) VCC5
  - LTC1778 3.3V (synchronous buck) VCC3,3
  - LM2678-ADJ 8V (buck) (feeds AVCC5 regulator)
  - LT1085-ADJ 5V (linear) AVCC5
  - LTC1778: LTC Design Solutions 43 'High performance switch mode power solutions for Altera low voltage FPGAs’, June 2004, indices the part can be used to generate 1.2V at up to 12A.
  - The LTC1778 circuit uses a Schottky diode across the bottom MOSFET apparently to increase efficiency, look into that.
  - the DS43 document warns about light-load conditions increasing the ripple voltage.
  - Take some measurements of the ripple on the board. Is there any way to measure current?
The Mictor on the board connects to some of the differential signals. Unfortunately all pins connect to LVDS transmitter pairs, so testing of the deserializer component via the Mictor is not possible. There are 13 differential transmitters, so it would be possible to transmit the data over to a COBRA board (though only for a spectral line mode - which is fine since the board only has 100MHz ADCs). See p39-40 of the datasheet. The connector has 27 high-speed I/O, 25 signal 2 clocks (in and out). The JTAG signals are also available on the Mictor (for NIOS debuggers).

Use the Mictor connector as debug interface to the logic analyzer. Figure out which connections need to be clocks. I'll want several Mictor connectors on the CARMA board.

100MHz oscillator-clock (on A16)

p14 datasheet has a block diagram of the clocks, look at the clock inputs (what type of pin do they go to?) and outputs (from enhanced PLLs or I/O pins?). Is there a clock pin leading to a Fast PLL and an Enhanced PLL? Test both types of PLL.

Take a look at the FPGA PLL power circuits, just a ferrite bead between it and the 3.3V digital power plane?

Take a look at the ADC/DAC power and signal coupling

The connectors on the bottom (J31, J33) of the board are for TI-EVM/FPDP.

The connectors near the ADC/DAC (J5, J6) are called ADI, and are intended for interfacing with Analog Devices boards.

Pinout notes:

- TI-EVM connector J31: 17-pins in top (single-ended) I/O banks
- TI-EVM connector J33: 62-pins in top (single-ended) I/O banks
- ADI connector J5, J6: 34-pins on LHS of the device, 8 LVDS receiver pairs and 9 LVDS transmitter pairs.
- Proto headers J23, 24, 25: 41-pins, 19 differential receiver channels from the RHS banks.
- Proto headers J26, 27, 28: 41-pins, 19 differential transmitter channels from the RHS banks.

The ADI or proto connectors could possibly be used to test LVDS and deserializer interfacing with the 1GHz digitizer, but **PROBLEM: LVDS requires 2.5V VCCIO**. The VCCIO power plane can not be changed to 2.5V, as the 3.3V plane is connected to lots of other devices, and the FPGA VCCIO pins are tied directly to the plate (see p26 of the schematic). So, there is no way to test LVDS interface with the DSP kit.

Any of the connectors could be used for PQ interfacing, and FPGA-to-FPGA bus implementation. The 100-mil headers on the top-side of the PCB are probably easiest to access (since the TI-EVM would need a special connector adapter).
5.4 Notes

- LVDS voltage compatibility: are the FPGA and digitizer levels compatible (I think they are both 2.5V LVDS interfaces)? Will the COBRA 3.3V LVDS interfaces compatible with the CARMA 2.5V LVDS interfaces?

- Deserializer setup. Does the input clock have to be slower than the data rate by the same factor as the deserialization? Eg. if the deserializer captures 500MHz clock rate data, and deserializes by 4 to 125MHz, does the input reference clock have to be 125MHz? The handbook diagrams seem to imply it. The PLL input divider could be used to divide by 4. Setup a test design to check the settings.

- DPA. Will the digitizer interface need it, and if so, what can be used for the training pattern, is it required?

- CPU interface (pin requirements)

- Front panel connectors:
  - replace the four connectors with eight smaller (stackable?) ones?
  - route the unused LVDS banks on FPGA0 and FPGA3 to additional (up to four more 16-bit channels) front panel connectors (if the existing are replaced)? The phase alignment circuits on the receiver should be able to account for skew between connector channels.
  - Stratix II High-Speed Development Kit would have all the best connectors on it.
  - Individual connectors are nice since given 8 connectors you could route 4+4 of each digitizer or 5+3 if fanout required it. Currently the destination of a digitized pair is fixed by the fact that both signals have to go to the same correlator board.

- What number of output connectors would be needed to eliminate the fanout boards?

- What about polarization - connector requirements?

- The LVDS transmitter and receiver channels can operate at 1Gbps. This would imply that sending 4-bits at 250MHz over the cables should be possible. If the fanout boards are eliminated, and a new high-speed connector and cable is used, then higher SNR wideband modes are possible.
6 PowerQUICC II Pro PowerPC CPU

The digitizer and correlator boards produce a continuous stream of data. The FPGAs correlate (multiply-and-accumulate) the digitized data, and dump data to the board processor on millisecond timescales. The board processor converts data to single-precision floating point and performs additional processing, eg. FFT, corrections, and accumulations (including phase switch demodulation). The sideband separated data is then transferred to the host CPU (a Linux host) every 100ms or 500ms. The data flow from the FPGAs to the board processor, accumulating data at the board processor, and transfer of results to the Linux host can easily become a bottleneck in the system due to processor bus contentions. The following sections provide the basis for the CARMA Digitizer Board processor selection and describe the features available on the Freescale 8349E.

6.1 Processor requirements

The processor requirements for the CARMA Digitizer Board are:

- 300MHz to 600MHz processor frequency
- 3W to 6W power dissipation
- Single-precision floating-point unit (FPU), IEEE-754 format
- Three independent buses;
  - Memory interface (eg. SDRAM or DDR SDRAM)
  - PCI interface (eg. 32-bit or 64-bit, 33MHz or 66MHz)
  - Peripheral interface
- The PCI interface must support PCI agent mode (i.e., the processor can be configured as a PCI peripheral board, not just a host or system board)
- Multi-channel DMA controller

The estimate for the maximum number of lags in the CARMA revision is 1024 (the number is rounded up to the nearest power-of-two). Based on 16-baselines per correlator board, the bus bandwidth requirements can be estimated.

- Peripheral Bus Bandwidth
  - Data specification:
    16-baselines, 2-phase bins (worst-case). 1024-lags, 4-bytes per lag is 128kB of data every 15.625ms (assuming the current 1024pps phase switch, 16-state hardware demodulation, and support for both 90-degree and 180-degree phase switching).
    The data needs to be transferred via DMA to main memory, where the host CPU can convert it to floating-point format, FFT it, process, and average the data.
    FPGA-to-Memory transfers should take no more than 10-percent of the 15.625ms correlation dump period (due to the fact that this data also needs to be processed). The peripheral bus bandwidth required is;
    \[ 128kB / 1.5625ms = 80MB/s \]

- PCI Bus Bandwidth
  - The data transfers to the host over PCI occur every 500ms. The transfer should complete within 100ms. There will be up to 20 boards in a crate, so each board must complete its transfer within 5ms. The PCI bus bandwidth required is;
- 128kB/5ms = 25MB/s

- **Memory Controller Bandwidth**
  - The memory controller bus bandwidth needs to be between 100MB/s to 1GB/s. This requirement is fairly easy to meet on an embedded processor, e.g., a 64-bit, 66MHz bus has a (theoretical) maximum bandwidth of 528MB/s.

### 6.2 Signal processing

The correlation data signal processing tasks performed by the COBRA board processors are:

- Read integer-valued lags from the FPGAs
- Convert the lags to floating-point format
- Normalization
- FFT
- Spectrum phase correction
- Accumulation into appropriate phase bins (sideband separation)
- Transfer data to the host every 100ms or 500ms

The CARMA Correlator System will implement delay and phase corrections on the digitizer board, so the phase correction performed by the COBRA correlator boards will be eliminated, and the rate at which FFTs are performed will be reduced. During development, it is likely that the new boards would be programmed to operate identically to the COBRA boards, so that the delay and phase control on the new boards can be compared relative to the older method.

The CARMA Correlator Board operations will be:

- Read integer-valued lags from the FPGAs
- Convert the lags to floating-point format
- Normalization
- Accumulation into appropriate phase bins (sideband separation)
- FFT
- Transfer data to the host every 100ms or 500ms

where the FFT can occur after accumulation, since phase corrections are not applied. Data is accumulated in float-point format, as the dynamic range of the data can exceed that stored in a 32-bit integer.

The CARMA Digitizer Board will perform the same operations as the Correlator board, but only on 2 autocorrelations (not the 16 cross-correlations the correlator works with). In addition to the correlation processing, the digitizer operations are:

- Digitizer control
- 1GHz clock control
- Whole nanosecond delay line control
- Sub-nanosection FIR filter tap download
- Lobe rotation phase control
- Quantization-state monitoring and control
6.3 PowerQUICC II Pro features

The Freescale (previously Motorola) MPC8349E PowerQUICC II Pro PowerPC CPU meets the requirements for the CARMA Digitizer Board processor. The CPU is a highly-integrated system-on-chip, i.e., includes a CPU core, multiple buses, and integrated peripherals. Figure 9 shows a block diagram of the MPC8349E.

The major features of the MPC8349E relevant to the COBRA Digitizer Board design are:

- PowerPC processor core
  - operates up to 667MHz
  - IEEE-754 FPU
  - 32kB instruction and 32kB data caches
  - software-compatible with all PowerPC processors

- Three independent processor buses
  - 64-bit 333MHz DDR SDRAM memory bus
  - 32-bit 133MHz local (peripheral) bus
  - 32/64-bit 33/66MHz PCI interface

- PCI messaging unit for PCI bus communications

- Multichannel DMA controller
Several other processors were investigated for the CARMA Digitizer Board, e.g., Freescale Cold-Fire MCF547X or MCF548X, AMCC PowerPC 440EP, Freescale PowerPC 8540, Freescale PowerPC 8548E. Most of these devices failed in the requirement of being able to operate as a PCI target device, i.e., they either could not generate a PCI interrupt, or they did not include a messaging unit that would enable efficient host-to-target communications. Other devices based on ARM or x86 cores failed either due to lack of a floating-point unit, or their lack of integration and debug features.

6.4 Real-time requirements

The CARMA Digitizer Boards will run Linux. Linux is not a real-time operating system, however, experience with the COBRA system showed that the Linux 2.4 kernel could handle 100ms data rates. The Linux 2.6 kernel has a number of improvements and should be able to handle the 15.625ms events on a digitizer or correlator board. The Linux processing does not have to perform operations with microseconds accuracy, it just has to complete its operations within the 15.625ms window before the next data set is ready, or needed (in the case of digitizer tap coefficients). If benchmarking shows that Linux is unable to hit 15.625ms windows, then buffering in the system controller FPGA of say 16 sets of data (e.g., correlator data coming to the processor, or tap coefficients going to the FPGAs) can be stored. The Linux host would then need to update these buffers every 250ms.

6.5 Processing time estimates

The e300 core reference manual indicates the instruction timing of the FPU. The floating-point multiplications and additions take a single clock, while divides can take multiple clocks. In the COBRA software divisions generally occur as scaling factors, so operations are optimized by calculating the inverse of the scale factor outside of a lag-loop, and the multiplying by the inverse inside the lag-loop. The calculations that will dominate the signal-processing are then executed in a single-clock.

The estimates in this section can be benchmarked on the MPC8349E-MDS (modular development system) board. The board is a PCI form-factor board containing an 8349E processor running at 667MHz, with 512MB of SDRAM, and runs the Linux 2.6 kernel. Freescale supplies a Linux kernel and board-support-package (BSP), and the configuration for the board appears in the standard PowerPC Linux distributions.

Estimates for 16-baselines, 1024-lags, 667MHz processor:

- Lag dump processing:
  - Integer-to-floating-point conversion; 1-clock per lag, 16384-clocks total, 25μs.
  - Normalization (floating-point multiply); 1-clock per lag, 16384-clocks total, 25μs.
  - Accumulation (floating-point add); 1-clock per lag, 16384-clocks total, 25μs.
  - Total time; 75μs. For a lag dump every 15.625ms, this is a small processing load. Lag dump processing could potentially occur every phase-switch (977μs), but at that rate, the Linux kernel would likely be an issue (without buffering the phase-state dumps).

- FFT processing:
  - An FFT operation count is proportional to $N \log_2 N$, and with $N = 1024$ corresponds to 10240-clocks per FFT. The FFT of the 16-correlations, 2-sidebands of data then requires 327680-clocks, or 491μs.
  - The FFT could also be performed every 15.625ms phase dump (and would take half as long, since only one phase bin is dumped).

- Transfer to host (performed via DMA), minimal overhead.
If the benchmark estimates are confirmed on the MPC8349E, then the clock rate of the processor on the CARMA boards can be reduced proportionally, eg. run the core at 333MHz and reduce the processor power consumption.

The data sets processed by the correlator system are larger than the processor cache, i.e., the data cache is 32kB, whereas the 16-correlations of data on a correlator board require 128kB (two sidebands). The performance measurements will determine the impact of the cache and bandwidth for transfers to and from main memory.

6.6 Notes

- Benchmark the processor using the MPC8349E-MDS (modular development system) board.

- Look at the Linux board support package for the MPC8349E-MDS. Can the CARMA Digitizer Board implement identical functionality, so that it can boot the same version of Linux? This would entail using the same memory (with I2C serial presence detect EEPROM), and Flash, etc.

- Estimate the buffering required from the system controller FPGA. Eg. 64kB lags every 15.625ms, 16 buffers worth, is 1MB. Do I have enough SRAM internal to the system controller FPGA? It would probably be worth adding SRAM onto the local bus (two banks?) or system controller (what type ZBT-SRAM?).
7 Clocks

Figure 10 shows the clocking within the CARMA Correlator System. The clocks in the system are all phase-locked to a common 10MHz reference, originating from the master clock unit (not shown). The phase-locked clocks will have arbitrary phases relative to each other (due to the 10MHz distribution, component delays, and the different types of PLLs used). The digitizer and correlator board FPGAs use FIFOs whenever a data bus crosses a clock domain to ensure that the data is correctly received without setup or hold violations. This allows the clock mis-alignment to be measured, and corrected. In the case of propagation delay, once the mis-alignment is measured, the appropriate delay setting can be stored, and when power is cycled, the delay corrections restored.

The two digitizer channels shown in Figure 10 represent either two channels on the same board, or two channels on different boards. A correlator board can receive up to four LVDS cables from digitizer boards. Each LVDS cable will transport two digitized antenna signals. The clock carried with the data on the LVDS cable will be used to register the data at its destination, the correlator receive FIFO. The correlator FPGA then uses its PLL locked to its copy of the 10MHz reference to generate the clock for its side of the FIFO. Once data from the four front-panels is received into the four FPGAs, the clock domain is assumed to be coherent, i.e., the four PLLs in the four correlator FPGA have locked with identical phase relationships relative to the 10MHz reference. This allows the correlator board inter-FPGA buses to simply clock data out from one FPGA, and clock data in to the next FPGA. The digitizer board FPGAs will have the same property, i.e., if the two digitizer channels in Figure 10 were on the same board, then data could be clocked between the two FPGA processing logic blocks without using a FIFO.
Figure 10: CARMA Digitizer clock domains. The figure shows how FIFO logic is used on the digitizer and correlator boards whenever a clock domain is crossed.
7.1 Notes

- 10MHz and 1GHz are integer related, but 10MHz/125MHz/62.5MHz, etc, are not. The assumption of the four digitizer FPGA PLLs or four correlator FPGA PLLs locking to the same phase needs to be confirmed. This is likely the case where the 1pps pulse needs to be used to synchronously reset or enable the PLL logic on each FPGA. Test this with two PLLs on one FPGA, and two PLLs on two Stratix kits.

- Figure out how to program the FPGA PLLs. Capture some traces. Look at the output divider functions (they are described like they are synchronized to the PLL, i.e., the divisions can be controlled with absolute phase).

- 10MHz reference and 1pps pulse synchronous with the 10MHz reference.

- 10MHz PLLed to give 1GHz, divided by to to give 500MHz. Ambiguity with the digitizer divide-by-2; can it be reset synchronously with say a 1pps pulse?

- 500MHz to 125MHz demultiplex-by-4 in the FPGA. How does it get synchronized? Can I reset the deserializer? Can the digitizer SYNC_OUT plus logic be used to reset the deserializer (in a useful way)?

- Does the deserializer actually need to be fed a 125MHz clock? Can I fed the PLL the 500MHz clock and use its input divided to divide by 4?

- LVDS output buses, where does the clock come from? What frequency is it, i.e., 125MHz DDR, 125MHz SDR, what about the spectral line modes?

- LVDS input buses; receive data using the LVDS clock into a FIFO. Clock out using the FPGA clock.

- If an FPGA receives 10MHz, can it multiply by 6.25 and 12.5 to create the 62.5MHz and 125MHz clocks. Note that it would not be valid to create a 125MHz clock and divide it by two, since there would be an ambiguity (unless the division was within a PLL loop).

- What clock will the FPGA use for the processor interface? It would be nice if the clock originates from the processor bus clock.
8  Power

8.1  Power Estimates

Power Budget:

- 65W total per board
- 15W maximum per FPGA (on say the correlators if they are loaded with the densist FPGAs).
- 15W/1.2V = 12.5A per FPGA (which is not much different than Kevin’s simulation estimate)
- 60W of FPGA power coming from a 90-percent efficient dc-dc converter will dissipate 6W at the converter.
- Digitizer: Atmel 1.4W dual.
- CPU+DDR+SYS? 5W-10W?

- Power sources: CPU+SYS should get power from the PCI bus. The rest should be powered from the custom power supply (or supplies).
- If dc-dc converters are used to generate all on-board voltages, then only one custom supply voltage will be required.
- Power tolerances? Since the FPGAs can draw huge currents, there is the potential for voltage drops to exceed the supply tolerance. The advantage of using an on-board dc-dc converter is that 48V (a standard telecom voltage) at a lower current can be routed through the cPCI connectors and converted down to 1.2V with a corresponding increase in current.

8.2  Power Sequencing and Management

- Power Sequencing:
  - PowerQUICC core and I/O voltages ??
  - Stratix II voltages; core ??, 3.3V I/O, LVDS, CML, ...
  - Digitizer voltages; ??
- Power Management:
  - It’d be nice for the board to boot just the CPU+mem+sys and then be able to selectively power up the FPGAs, and the digitizers. This would allow minimal power consumption. Probably need BusSwitches between the CPU and some devices so that power is not applied to the device I/O pins when the device is off (I think the digitizer IC is the only concern here).
  - All devices should have separate power segments, with segments being fed from current limiting supplies.
- PCI hot swap.
- TI and Linear Tech have lots of good app notes on how to sequence and manage power.
- Digitizer voltages; low noise analog rails should use linear regulators, everything else could probably use more efficient DC-to-DC converters.
- Copy ideas from the eval boards.
• Do I need the COBRA custom power supplies anymore (certainly not the -5V and -2V ECL supplies). It would be ideal if I could use just the cPCI supplies. The COBRA digitizer boards required 60W, cPCI per board spec is 35W, with luck we can stay beneath that.

• Monitoring
  – Power supply voltages and currents
  – Do the digitizers and FPGAs have on-chip temperature diode?
  – Add SPI/I2C temperature sensors under the board.