CARMA Correlator Revision

D. W. Hawkins (dwh@ovro.caltech.edu)
Revision: 1.1
March 2, 2006

Contents

1 Introduction 3

2 Task overview 4
   2.1 Processor .......................................... 4
   2.2 Low-level software ..................................... 6
   2.3 High-level software ..................................... 7
   2.4 Correlator FPGA ...................................... 8
   2.5 System controller FPGA ................................ 9
   2.6 Clock reference ....................................... 9
   2.7 1GHz PLL .......................................... 9
   2.8 1GHz Digitizer ........................................ 10
   2.9 Power supplies ....................................... 10
   2.10 Board design ........................................ 10

3 Task estimates 11
1 Introduction

This document summarizes the tasks required to implement the CARMA Correlator Revision, i.e., the full 4GHz 15-antenna correlator system. The CARMA design revision will satisfy concerns raised about the spectral resolution capabilities of the correlator bands based on the COBRA hardware, and is required due to the obsolescence of key components in the COBRA design. The COBRA hardware components that are now end-of-life are; the 1GHz digitizers, and the Texas Instruments TMS320LC31 DSP.

The Altera FPGA family used in the COBRA designs, the FLEX10K series, has also been superceded by several generations of FPGA devices. The Stratix II series is the device that will be used on the CARMA revision. The FPGA hardware requirements are: interface to the 1GHz digitizers, perform digital downconversion, filtering, and correlation operations, and transport data at LVDS logic levels over the front panel cabling. The Stratix II family meets these requirements, and the device density is such that they provide a significant increase in spectral resolution. In addition, the device density is such that a correlator board can process 16-baselines per board, reducing the fanout requirements of the digitizer boards, eliminating the need for the LVDS fanout boards used in the COBRA-based CARMA bands.
2 Task overview

The following sections provide an overview of tasks.

2.1 Processor

The processor requirements for the CARMA revision are:

- 300MHz to 500MHz processor frequency
- 2W to 3W power dissipation
- Floating-point unit (FPU)
- 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)

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, 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, eg. a 64-bit, 66MHz, bus has a (theoretical) maximum bandwidth of 528MB/s.
The following is the sequential list of processor tasks. Each task is dependent on one or more of the previous tasks, however, some development on all tasks has been performed (given that the development boards are shipped with hardware and software that covers each task).

1. **Processor selection**
   
   (a) Freescale MCF5485 Coldfire.  
       Failed. Inflexible clocking modes, and the device cannot act as a PCI agent/peripheral.  
   
   (b) AMCC 440EP PowerPC.  
       In progress. Issues identified; PCI bus DMA command codes, and PCI agent mode capabilities. Awaiting resolution.  
   
   (c) Freescale PowerQUICC II Pro PowerPC.  
       Awaiting hardware.

2. **Benchmark bus performance**
   
   Both PowerPC devices should have no problems meeting bus requirements. Testing is in-progress.

3. **Operating system selection**
   
   The objective is to use Linux, as it provides a well-known environment, and the development boards selected all run Linux, so drivers already exist for all the standard processor devices. The correlator has real-time requirements that must be met for correct operation. The Linux 2.6 kernel may meet these requirements without modification, if not, real-time extensions will be investigated.

4. **Root file system**
   
   The processor will boot, load its Linux kernel (from Flash, or via the host), and then load a filesystem into a RAM disk. The contents of this filesystem can be determined using a development board (based on the filesystem provided with the board). The size of the filesystem is required to determine the amount of RAM required on a board.

5. **Bootloader**
   
   The bootloader sets up the processor and then boots Linux. The development boards all ship with the U-Boot bootloader, and open-source bootloader, written and supported by Denx Engineering. The Denx engineers frequently respond to my questions on the PowerPC developers mailing list and would be able to assist me, or be subcontracted for any bootloader work. The objective is to copy the processor infrastructure from a development board to minimize, or eliminate, any bootloader modifications.

6. **Host-to-target communications**
   
   Implement a host-to-target communications protocol and benchmark it.  
   The objective is to develop a virtual network interface over PCI. A brief web search showed up several implementations, and Beowulf Clustering users may have something of use.

7. **Hardware interface**
   
   The objective is to find an appropriate development board, copy the majority of its design, and then add the CARMA specific hardware, eg. the FPGAs on the peripheral bus. The main work here is determining the bus wiring, developing a bus simulator, VHDL interface logic, and then checking processor-to-FPGA timing. The bus will likely need buffering, due to the multiple FPGA loads, and some hardware simulation will be required (the PADS PowerPCB HyperLynx package was used during the COBRA development, and we now have that software).
2.2 Low-level software

The software tasks can be considered in terms of low-level and high-level tasks. The low-level tasks are closely coupled to the hardware, and most were mentioned in the previous section.

Additional low-level tasks that are required in the final design are: digitizer board PLL control, digitizer control, and board monitor points. Many of the control interfaces will likely be via registers, and so the drivers are very simple. The board monitor points, e.g. power supply voltage and current sensing, and temperature sensors, will most likely be located on an SPI or I2C bus. The Linux lm_sensors project has a large number of drivers written for these types of devices. This project can be consulted to see what bus infrastructure exists in the Linux kernel for SPI and I2C, and then what devices attached to those buses is supported. These kinds of sensors can easily be tested using a parallel port on an x86 PC.


2.3 High-level software

The PowerPC user-space instruction set is identical across processor versions, so high-level software development can occur on any PowerPC Linux system. The real-time aspect of that software will not be realized, but at least the compile and test cycle does not need to include an embedded target.

The COBRA correlator software consists of a real-time OS on the DSPs (μC/OS-II), and Linux on the x86 host in the cPCI crates. The host control software is based on the ACE C++ library. This library provides a set of classes and frameworks that implement common multi-threaded and multi-process programming design patterns. The ACE-based host control code could easily migrate to the embedded PowerPC processor. The ACE library would need to be built for PowerPC Linux and its library size requirements determined. Tools within the ACE distribution can be used to reduce its memory footprint, and the library is currently used for embedded projects.

The CARMA high-level software infrastructure uses CORBA. If the high-level software developer (Tom) wants to use CORBA on the PowerPC processor, its impact on memory and PCI bus bandwidth will need to be quantified. The source of the CORBA library will also have to be determined. For example, the TAO library is built on top of ACE, and is an implementation of CORBA. CORBA requires a network infrastructure to operate, and a lot of behind-the-scenes communications takes place (eg. remote object location).

The following real-time tasks are required on the PowerPC processor:

1. Digitizer FPGA control (FIR filter taps, lobe rotation, etc)
2. Phase switch demodulation (loading registers)
3. Correlation result transfer (FPGA-to-memory)
4. Integer-to-floating-point conversion
5. Normalization
6. FFT
7. Delay correction
8. Phase switch demodulation (0-degree and 90-degree bins)
9. Averaging

The COBRA DSP documentation contains extensive benchmark results that should be repeated for the PowerPC. FFTW, the ‘fastest FFT in the West’ should be investigated for use performing the FFT.

The following tasks are currently performed by the x86 control host:

1. Control interface; implements a control server providing a network interface to control multiple boards.
2. Data interface; implements a data server that collects data from multiple boards, sorts the data, performs delay correction and secondary lobe rotation correction, and averaging.
3. Monitor interface; implements a monitor server that collects monitor data from multiple boards.

The functionality of this design should be preserved. For example, independent of the observatory control system, an extensive set of lab debugging tools exists. Any design change in the high-level software needs to account for lab development too.
2.4 Correlator FPGA

The CARMA revision digitizer boards will use the Altera Stratix II FPGAs. This selection is based on their ability to interface directly to the 1GHz digitizers selected. The digitizers ping-pong their output data into two buses that operate at 500MHz clock rate, or 250MHz double-data-rate (DDR).

Digitizer board tasks

1. Digitizer interface
2. Digital downconversion and phase control (secondary lobe rotation)
3. Delay line (whole nanoseconds)
4. FIR filtering (bandwidth selection and sub-nanosecond delay correction)
5. 180-degree phase switch demodulation
6. Front-panel LVDS distribution
7. Autocorrelation
8. Processor interface

Correlator board tasks

1. Front panel LVDS reception
2. FPGA-to-FPGA data transfers
3. Cross-correlation
4. Phase-switch demodulation
5. Processor interface

The functionality of a correlator board is a subset of that of a digitizer board. Hence the digitizer board will be developed first, built and tested, and then the correlator boards built.

Altera’s Stratix II DSP Development Kit contains an FPGA, two 125MHz digitizers, and multiple I/O interfaces. This board will be used to test each component and to confirm the maximum operating frequency of the design reported by the Altera place-and-route tool versus actual hardware. The DSP kit also contains a network interface that could be used to transfer data from the board. The DSP kit is supplied with an example design that implements an Altera NIOS II processor, a 32-bit processor. This processor can then boot Linux, or the eCOS RTOS. The NIOS II processor uses a small fraction of the chip, so correlation logic could be added to the reference design for a real-world test of the correlation logic.

The 1GHz digitizer interface requires the demultiplexing of 500MHz clock rate data from the digitizer into a wider, slower, data stream within the FPGA. The DSP kit can be used to implement and test this logic; initially at 100MHz using the on-board digitizers, and then using the external digitizer board at whatever frequency is possible.
2.5 System controller FPGA

The system controller FPGA tasks are;

1. Correlation control signals; the time reference and phase reference signals will be inputs, the correlate, dump, and other control signals to the FPGAs will be outputs.

2. Interface (glue) logic for processor control of the 1GHz PLL, monitoring devices, etc.

The system controller on the CARMA revision boards is considerably simpler than the COBRA boards, due to the fact that the PowerPC processor has absorbed most of the functionality, eg. memory controller, and PCI interface.

2.6 Clock reference

The COBRA boards use a 15.625MHz reference \( f_{\text{ref}} \) to generate the 1GHz \( (64f_{\text{ref}}) \) digitizer clocks, and 125MHz \( (8f_{\text{ref}}) \) and 62.5MHz \( (2f_{\text{ref}}) \) FPGA clocks. The PLLs used in the design had limited divisor options, and the FPGAs do not include PLLs.

The Stratix II FPGAs contain an abundance of PLL logic, and the 1GHz digitizer reference clock is also very flexible. The option of using a scheme similar to the BIMA antenna 10MHz with missing 1pps needs to be invesigated. This could be used to unambigously lock the PLLs, easing the power-on clock alignment requirements.

The objective of the revision clocking scheme is to be able to power-up the system with a consistent starting phase on each clock. If the power-up state is consistent, then calibrated clock adjustments can be stored, and the system can power-up with clocks aligned, or so closely aligned that a noise source phase flattening routine can be run to correct for any residuals.

2.7 1GHz PLL

1GHz PLL tasks;

1. Test the PLL and VCOs

2. Jitter; from one board? Between two boards?

3. Control interface; SPI bus, external latching?

4. Linux-side control

5. Digitizer interface.

6. Power supplies; voltages, sequencing, noise requirements
2.8 1GHz Digitizer

1GHz digitizer tasks;

1. Evaluate the digitizers (Atmel and National parts)
   (a) Bandwidth?
   (b) The digitizers are dual-digitizers; what is the noise coupling like, is it acceptable?

2. Determine the interface to the FPGAs

3. Determine the control interface. If the digitizers need to be controlled in any way, does one digitizer fare better than the other. For example, the Atmel part has two independent clocks, whereas the National part, only one.

4. RF interface; amplifiers, baluns, etc.

5. Power supplies; voltages, sequencing, noise requirements (PSRR).

2.9 Power supplies

Power supply tasks;

- How many, how much current?
- Sequencing.
- Objectives;
  - Hot-swap processor interface
  - At power-up, only the processor, and memory are powered.
  - The processor control software can then turn on power to the rest of the board, eg. digitizer clock, digitizers, FPGAs. The system controller FPGA can contain logic to ensure supply sequence requirements are met.

An example of how the power requirements effect the board design is the digitizer IC. The digitizer has a requirement that RF and I/O signals not be applied while the part is off. This requires that an RF amplifier or switch precede the device until its on, and that the FPGA control signals are tri-stated until the device is on.

2.10 Board design

Board design tasks;

- Engineering specification
- Schematic capture
- PCB place-and-route
- PCB manufacture
- PCB loading
- Hardware testing
3 Task estimates

This section contains estimates of task effort for each developer; Hardware (Hawkins), Correlator FPGA (Rauch), Software (Costa).

Table 1: Hardware task estimates

<table>
<thead>
<tr>
<th>Task</th>
<th>Effort (weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor</strong></td>
<td></td>
</tr>
<tr>
<td>Evaluation/Benchmarking</td>
<td>3</td>
</tr>
<tr>
<td>Drivers</td>
<td>3+</td>
</tr>
<tr>
<td><strong>Clock design</strong></td>
<td></td>
</tr>
<tr>
<td>Clock analysis</td>
<td>1</td>
</tr>
<tr>
<td><strong>1GHz PLL test</strong></td>
<td></td>
</tr>
<tr>
<td>Processor interface</td>
<td>1</td>
</tr>
<tr>
<td>Digitizer interface</td>
<td>1</td>
</tr>
<tr>
<td>System controller</td>
<td>1</td>
</tr>
<tr>
<td><strong>FPGA</strong></td>
<td></td>
</tr>
<tr>
<td>Clock analysis</td>
<td>1</td>
</tr>
<tr>
<td>Process interface</td>
<td>1</td>
</tr>
<tr>
<td>Digitizer interface</td>
<td>1</td>
</tr>
<tr>
<td>Power</td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td>1</td>
</tr>
<tr>
<td>1GHz Digitizer</td>
<td></td>
</tr>
<tr>
<td>1GHz PLL</td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td></td>
</tr>
<tr>
<td><strong>Digitizer board design</strong></td>
<td></td>
</tr>
<tr>
<td>Engineering specification</td>
<td>2</td>
</tr>
<tr>
<td>Schematic capture</td>
<td>2</td>
</tr>
<tr>
<td>PCB place-and-route</td>
<td>2</td>
</tr>
<tr>
<td>PCB manufacture</td>
<td>2</td>
</tr>
<tr>
<td>PCB loading</td>
<td>2</td>
</tr>
<tr>
<td>Hardware testing</td>
<td>4</td>
</tr>
<tr>
<td><strong>Correlator board design</strong></td>
<td></td>
</tr>
<tr>
<td>PCB place-and-route</td>
<td>2</td>
</tr>
<tr>
<td>PCB manufacture</td>
<td>2</td>
</tr>
<tr>
<td>PCB loading</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 2: Correlator FPGA task estimates

<table>
<thead>
<tr>
<th>Task</th>
<th>Effort (weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Altera Stratix II DSP kit testing</strong></td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>2</td>
</tr>
<tr>
<td>100MHz correlator</td>
<td>2</td>
</tr>
<tr>
<td>FPGA I/O deserializer logic</td>
<td>1</td>
</tr>
<tr>
<td>1GHz ADC test</td>
<td>1</td>
</tr>
<tr>
<td><strong>FPGA-to-Processor interface</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Correlation logic</strong></td>
<td></td>
</tr>
<tr>
<td>Dual-ported RAM use</td>
<td>1</td>
</tr>
<tr>
<td>M-RAM control registers</td>
<td>1</td>
</tr>
<tr>
<td>FIR logic</td>
<td>1</td>
</tr>
<tr>
<td>Delay/phase control</td>
<td>1</td>
</tr>
<tr>
<td>Misc. logic</td>
<td>1</td>
</tr>
<tr>
<td><strong>Board-level simulation</strong></td>
<td></td>
</tr>
<tr>
<td>System controller (FPGA control signals only)</td>
<td>1</td>
</tr>
<tr>
<td>Processor peripheral bus</td>
<td>1</td>
</tr>
<tr>
<td>Simulation</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3: High-level software task estimates

<table>
<thead>
<tr>
<th>Task</th>
<th>Effort (weeks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux development system setup</td>
<td>2</td>
</tr>
<tr>
<td>Software selection (ACE vs CORBA, FFTW, etc)</td>
<td>2</td>
</tr>
<tr>
<td>Board-level real-time task coding</td>
<td>4</td>
</tr>
<tr>
<td>Board-level real-time task testing</td>
<td>2</td>
</tr>
<tr>
<td>Network infrastructure setup</td>
<td>2</td>
</tr>
<tr>
<td>Board-to-host development and test</td>
<td>4</td>
</tr>
<tr>
<td>System test</td>
<td>4</td>
</tr>
</tbody>
</table>