Fast-sampler Correlator High-Level Software Implications

Kevin P. Rauch (UMD)

Feb 10, 2012

ABSTRACT

This document discusses the impact of the new fast-sampler based correlator hardware on the existing high-level software infrastructure, organized by component (monitor, data, control).

1. Hardware Overview

The existing CARMA correlator consists of eight independent (single-polarization) observing bands, each of which can be configured to operate with a total bandwidth ranging from 2 MHz to 500 MHz and providing complete baseline coverage for up to 15 inputs. The hardware for each observing band resides in a single cPCI crate containing 8 digitizer boards, 7 correlator boards, and 1 host CPU board (plus a timing card distributing clock synchronization signals). All boards run a Linux OS and are fully networked. In the default setup all networking for the digitizer and correlator boards (hereafter, CARMA boards) is routed through the crate backplane to the host CPU, which is connected to the site network via a 100 Mbps front-panel ethernet connection; however, the CARMA boards all have their own front-panel ethernet jacks as well, and backplane networking is mainly a convenience to simplify the network topology. Pairs of crates can also be combined to produce an observing band with complete baseline coverage for up to 30 inputs, providing support for 23-input single-polarization and 15-station full-Stokes observations.

The host CPU board contains an Intel x86 CPU and is effectively an IBM PC compatible computer in a cPCI form factor. The CARMA boards contain a Freescale PPC CPU as well as several Altera FPGAs, sensor devices, and other low-level hardware. Physical interaction between the host board and CARMA boards consists of PCI bus transactions, and the host is responsible for probing for and booting the latter. At the user-space level, however, communication between host and boards occurs via ordinary network socket connections, and is effectively hardware-independent; in principle, the boards could just as well be in a different crate (and in a remote location), so long as they are reachable via TCP/IP connections.

Compared to the CARMA correlator, the new fast-sampler hardware exhibits a decentralized, unhosted physical arrangement that breaks the simple identification of boards with bands. In this case each chassis (form factor TBD) houses several bandformer cards containing one data FPGA and other low-level devices (including connection to an ADC, physical format TBD), all managed by a single chassis CPU card. The chassis CPU card will contain an x86 CPU and run Linux; in comparison with the CARMA correlator, however, it is the analog of the PPC board control CPU, not the x86 host CPU, as it communicates directly with the FPGAs (albeit via PCIe as opposed to a custom local bus), with sensor hardware (via I^2C), the ADCs, etc.; that these devices are now distributed across several physical PCBs in multiple chassis slots is conceptually irrelevant. Unlike the CARMA boards, in the current design concept each bandformer control CPU now manages four ADCs (instead of two) and eight FPGAs (instead of four). This is functionally similar to combining two CARMA digitizers boards into one, with one crucial difference: each FPGA now processes four observing bands (astrobands) simultaneously for the associated ADC. Hence a more accurate analogy is to imagine combining 16 CARMA digitizer cards together (two from each crate, each pair connected to the same four antennas), all controlled by a single CPU. A single bandformer CPU (and FPGA) now deals with multiple astrobands; it is this fact which will impact the existing software most noticeably. **Code can no longer assume a specific piece of correlator hardware is tied to only one astroband at a time.**

When bandformer boards are used as a correlation backend, the (now) correlation control CPUs will encounter a different mix of observing bands. In this case each correlation FPGA still calculates baselines from only a single astroband; however, the FPGAs in one chassis may not all be associated with the same astroband (in the current concept, each chassis will process parts of one or two bands).

A new, separate correlator computer (most likely a rack mount system) will be needed to manage correlator operation. Functions like phase flattening, for example, involve iterative analysis of data harvested from multiple chassis in real time. It will also act as the communication agent to the RTS. It may be desirable to place this computer and the chassis CPUs on a separate subnet to reduce traffic on carma.pvt—in the case of CARMA boards host IP communication is confined to the crate backplane, whereas the chassis CPUs will use normal ethernet cabling.

The following sections explore the impact of these changes on the key software components of monitoring, visibility data, and control. Each highlights some key questions/conclusions, followed by a detailed rationale discussion.

2. Monitor Data

- Significant updates may be required.
- Is RTS code robust to monitor packets containing multiple astrobands?

At the lowest level, the `CorrelatorBoardMonitorReader` process runs on the PPC and is responsible for reading the board hardware sensors and encapsulating the results into C++ monitor data packets. These packets are captured by the local board server (`CorrelatorBandServer` process), where their contents are merged with additional monitor data points. The relevant container classes include `MonitorCarmaCommon`, `MonitorCarmaCorrelator`, and `MonitorCarmaDigitizer`. After aggregating output from the boards and supplying default output for any missing ones, the host (x86) band server streams the monitor data to the RTS for processing.

Historically the concepts of crate number and band number have been interchangeable. The advent of full-Stokes observing modes led to the introduction of astroband number to distinguish between astronomical observing band and hardware crate number. The fast-sampler hardware further confuses the issue as the bandformer chassis CPU will generate some monitor points tied to hardware (e.g., temperatures), which should fit easily into the current hierarchies, while other points (e.g., digital gain values) will refer to specific astronomical observing bands. The updated monitor classes will need to accommodate this in some way. The number of FPGAs managed by the bandformer CPU is greater than for the CARMA board PPCs, which changes the array dimensions for FPGA-based monitor points in the container classes. The existing COBRA and CARMA boards already differ in FPGA count, so presumably adding a third concrete sub-class encapsulating bandformer CPU monitor output will be straightforward. As to upgrading, it seems conceptually easiest to vectorize currently scalar values to transport multi-band monitor data in a single monitor packet (at least for the chassis CPU output), but if this format is awkward for the RTS then the correlator server might need to re-sort output into multiple RTS-friendly packets. In either case, *any code explicitly or implicitly assuming a single monitor packet is associated with only a single astroband will need to be re-examined*.

The meaning of slot number also changes. With only a single chassis CPU the slot number is not relevant to it as it is to the CARMA board server; instead it will apply on an FPGA basis. In the current concept two FPGA cards occupy a single physical slot. Since cards can be swapped out individually, the monitor stream should track both physical location (are upper/lower cards in one rail assigned unique slot numbers??) and board serial number.

3. Correlation Data

- Moderate updates required; no architectural changes.
- Is collation (by astroband) and/or buffering of incoming band objects required of the new correlator server?

The `CarmaDataReader` class receives the raw FPGA data in memory-mapped format and converts it into fully populated band data objects. The new hardware will similarly memory-map the FPGA RAM, so no significant modifications will be needed to capture the data—hardware access is hidden by the data driver. The main question is whether data needs to be segregated into astroband-specific band objects. The `CorrelatorBand` object header has a band number member, as do each of the individual baselines it contains. If the header variable is ignored (or used to denote the chassis number), there is no impetus to publish multiple band objects at this level—the downstream band server (running on the new correlator computer) can collate incoming data into band-specific objects, if required by the RTS pipeline. The low-level reader logic will need generalization to support multiple bands per FPGA.

Currently the CARMA host servers buffer board data until all boards have been received or time expires, whichever comes first. However, no dummy data is inserted for missing baselines, so depending on the flexibility of the RTS pipeline, the host buffering may be extraneous, and removing it might simplify correlator data management in the new mixed-astroband environment.

4. Correlator Control

- Significant modifications required.
- Careful thought needed to maintain forward/backward compatibility.
- Embrace or avoid a major shift in the correlator control API?

The `CarmaBoardControl` component receives control commands by proxy from the host band server (the `CorrelatorBandServer` process), with which the RTS communicates directly. Currently and by necessity RTS communication is crate-oriented, but with a close correspondence to astroband; at the top level, RTS control is based on astrobands. On the other hand, some commands such as `sourceName()` are logically independent of astroband. As mentioned previously, in the new system the chassis server will be a modified CARMA board server that potentially requires control commands from *all* astrobands to operate. This will require some conceptual modifications; e.g., the `astroBandNumber()` method will no longer be meaningful as currently defined. FPGA reconfiguration is distinctly different as a single bandformer FPGA will process inputs to four independent astrobands (though correlator FPGAs will continue to process part of a single astroband). In particular it is very likely that only a few combinations of modes will be available per FPGA—a significant change from the fully independent astrobands of the CARMA correlator, where the bandwidth/bitmode of each can be specified individually.

At the top level the correlator setup tool will need to enforce these restrictions to minimize submission of invalid scripts. If the RTS is to detect invalid requests (optional, but would improve error reporting), then `configastroband()` must be modified to work in blocks of four (or all eight) bands. Alternatively, the correlator server could buffer configuration commands into blocks of four before proxying them to the chassis servers, at the cost of exposure to dependency risk: unless/until all four configuration commands are received, nothing happens in this setup. However, the control path has proven to be quite stable, so the option needn't be discounted out of hand. Maintaining backward compatibility with the CARMA correlator factors into this decision.

The bandformer hardware will implement a digital second LO. New correlator control commands are required to manage this functionality and RTS control will need modification, including support for correlator-based Doppler tracking.

The new correlator server will be a modified version of the current CARMA correlator host server, now managing RTS access to all astrobands. There is no point trying to run a separate process for each band since every server would still need to connect to every chassis, which is highly redundant. Also the chassis server (a modified CARMA board server) must necessarily manage multiple astrobands, so core server support for multi-band operation is a requirement in any case. For the most part this can be hidden from the RTS, which could continue to operate with the concept of fully independent astrobands (except for the constraints on reconfiguration). A simple, backward-compatible way for the correlator server to receive commands for multiple astrobands is to have it listen on multiple ports; currently each CARMA band server already listens on a unique port as specified in the ini-file. An alternative is to generalize (vectorize) the control API to allow parameters for multiple astrobands to be specified in one call; this option is flexible and efficient, but more invasive to implement.

Although the hard configuration constraints are for blocks of four astrobands, in practice reconfiguration must be done eight at a time due to the continuous transfer of data between octets of bandformer FPGAs—configuring any one will bring down the entire ring communication network, which will then require full reinitialization. Hence at the hardware level all astrobands are coupled in some way; this suggests additional benefits to adding multi-astroband support to the control API itself, as opposed to attempting to hide this change deep in the system.