Design Specification for the Post-Correlator Integrator in the EVN MkIV correlator

EVNDOC #112

Version 1.00

Harro Verkouter

December 18, 2001

1 What is the Post Correlator Integrator

The Post Correlator Integrator (PCInt for short) is a device that will enable enhancement of the scientific output of the EVN MkIV correlator [1] and/or perform various calculations on the data e.g. correction, normalization, FFT.

As the name of the device suggests, these calculations will be carried out post-correlation. This means that the geometrical delay model has been applied to the datastreams and that actual correlation has already taken place.

Without the PCInt, the correlated data would be flushed to disk, ready to be inspected and if found good, exported to the end user: the PI. With the PCInt in place, an intermediary step after flushing to disk and prior to exportation could be inserted. The output of the PCInt would then be used as input to the inspection/export stage. As such, the output of the PCInt is similar to the output of the correlator itself.

As an aside, even if the PCInt would do no processing at all, it would still enhance the output of the correlator just by enabling subsecond integration time for the whole correlator (256k complex lags/integration). This is something we cannot sustain with the current setup.

[1] The PCInt could be used by both the EVN and the DZB correlators. It is our goal to keep development as parallel as possible between the two systems. However, some parts of this document may be more DZB specific whereas other portions may be more EVN specific.

2 Functional Requirements

This section will briefly discuss the various envisaged functions the PCInt would have to perform, as well as some of the requirements that we would like the system to fulfill.

2.1 Functions

This section lists the functions that we already know we want the PCInt to perform. It is not unimaginable that in the future more functions will be added.

- FFT. Transform the correlated lag data into the frequency domain. Most of the astronomical data processing software works on data in the frequency domain.
- Multiple Field Center processing. By applying a (time dependent) correction to the already correlated data, it is possible to produce a new datastream that behaves as if it was correlated with a different phase center.
- Filter. It must be possible to filter out data that matches (or does not match) various criteria.
- Integrate. It must be possible to integrate the data.
- Normalization. The data coming from the correlator must first be properly normalized.
- Corrections. It must be able to apply various corrections, e.g. quantization corrections.
- RFI Mitigation. This item is DZB specific. For the WSRT, the PCInt would be used to detect/filter RFI spikes in the data.

2.2 Requirements

It was assumed that the PCInt always at least did some form of datareduction; integration or FFT'ing the data. Recent discussions, however, revealed that there are applications in which it is useful to dump the raw correlator data at its maximum rate. This led to a notion of possibly pushing the requirements for the whole system. It was felt that 160MByte/s was the maximum useful datarate. This means that the requirements have shifted (upward) with respect to the previous design. At this point I will refrain from specifying more detailed datarate requirements. In the end, all that counts for the design of the system is the total anticipated datarate it has to cope with. In the orginal specification a number of 40MByte/s is targeted at. In the new design we will target at 160MByte/s sustained datarate.

3 Installation requirements

Whichever solution for the PCInt is chosen, it is clear that it should fit in the existing physical, electrical and software environment. This section will briefly describe the constraints we know of.

3.1 Physical environment

- Our correlator consists of four VME racks. Per rack there are some unoccupied VME-slots. The equipment should fit in these slots. This implies that the boards have a 6U VME form factor.
- The PCInt will read data from the correlator boards via a high speed serial link, a so called COMM-port. The lenght of this cable is limited; if the length exceeds $\approx 25cm$ the throughput of the cable will/can be affected.
- Any equipment installed (either in the VME racks or somewhere else) must provide easy access to cabling and or pushbuttons (for reset?).
- Heat production must be controlled. It may prove necessary to install more fans or increase the heat dissipative capacity of the existing system.

3.2 Electrical environment

- All equipment should comply to european voltage levels: 220V AC/50Hz.
- The boards to be placed in the rack should be compliant to the VME64 standard.
- The PCInt must interface to the standardized Texas Instruments C40-high speed serial link.
- Single PCInt units will interface to a network infrastructure (ethernet) using normal UTP cabling, 10/100BaseTX. Sometime in the future the datachannel may be enhanced to use 1000BaseSX or 1000BaseFX, whichever is more appropriate.

3.3 Software environment

- For data transport and control message transport over the network we will be using standard TCP/IP socket communication.
- We will strive to integrate the PCInt control as transparently as possible within the structure of the Jodrell Correlator Control Software, JCCS.
- Input data formats need to be defined. We have the input from correlator board to PCInt and we could also regard the raw data which is stored on disk as input to the offline processing software and hence we could discuss

it as well under output formats since it is the output of the online data-capture software. The format of the data as it comes from the correlator boards is available from Albert Bos.

- Output dataformats should be specified. There are two points in the system at which output has to be defined: how the raw data coming from the individual PCInt boards is stored on disk and second, how the (optionally) processed data is stored on disk. The fact that after dumping the data to disk processing is optional hints at that maybe there should be no difference between these formats at all. Also, it must be kept in mind that we will want to read the (optionally processed) data into AIPS++. This has implications for our program j2ms2, which is responsible for doing just that.

- Realtime constraints. Basically, in the whole PCInt there is only one action which is time-critical. It is the reading of data off the COMM port. This task has to complete before the new integration arrives at the COMM port. Sending the data out over e.g. the ethernet may take longer than a single integration. It is necessary to buffer the data in that case.

4 Implementation details

In this section we will propose an implementation capable of satisfying all previously mentioned requirements.

The diagram shows a schematic view of the system. All components shown on the figure will be discussed. You will find the multiplicity of the components listed. Sometimes these multiplicities are optional, not known yet, subject to how much money is available or subject to other constraints but at least envisaged to be > 1. In these cases the multiplicity is noted by a variable in *italic* font. Possible multiplicity of components has a lot of impact on the design of software. Therefore it is advisable to identify all locations where you would want multiplicities > 1 as early as possible in your project.

The idea behind the design is quite simple, actually. In the current system, the correlated data is read from the correlator boards by the Realtime system via the VME bus. After that, the data is sent off via ethernet to the current DDD machine where the data is reformatted and written to disk. With the PCInt in place, the datapath will be slightly different. The correlator boards put out the correlated data over the high speed serial link. Four correlator boards write their data to one SBC. The SBC, in turn, puts out the data over the ethernet onto a DDD machine where the data just gets flushed to disk. In this design it is assumed that all processing takes place offline in order to be able to keep up with the maximum datarate. After the data is stored on disk, offline software is run to perform the desired actions on the data.

4.1 Glossary of components found in the system

This section will describe (shortly) all components shown in *figure 1.* This list will contain the components that we will find in the system **after** the PCInt has been implemented; as such some of the components will not be found in the current system whereas others are. The items not found in the current system will be marked with a ★.

CCC The Correlator Control Computer. A UNIX host running software responsible for controlling the whole of the correlator system.

Correlator rack A VME rack populated with a number of VME-boards. Each of the four correlator racks contains the following boards:

- **Correlator board** Custom built board where all the correlations are actually calculated. A correlator board features a high speed serial link on its frontpanel, connected to one of the on-board Texas Instruments C40 Digital Signal Processors (DSP).
- **Realtime System** A UNIX Single Board Computer (SBC) acting as rack controller. Interfaces to all the installed hardware via the VME-bus and interfaces to the CCC via ethernet.

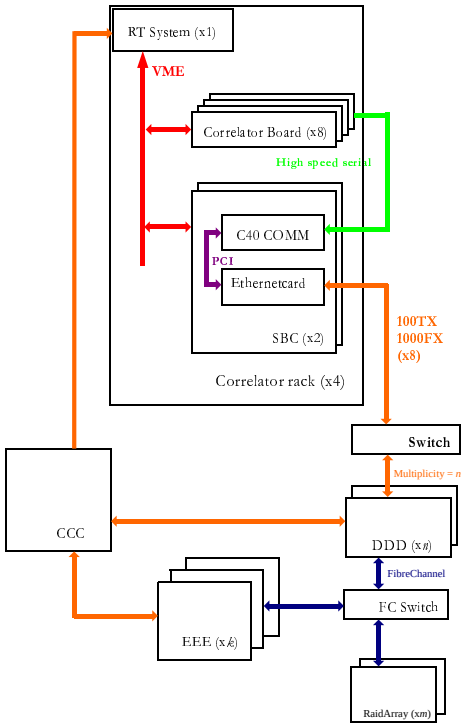

Figure 1: SCHEMATIC OVERVIEW WITH ALL COMPONENTS INSTALLED

- ★ **SBC** An Intel Pentium© based SBC with two PCI Mezzanine Card (PMC) slots, an onboard ethernet controller, memory controller and VME controller. The PMC slots will be fitted with
  - ★ **C40 COMM** A PMC module that features four high speed serial links.
  - ★ **Gbit eth** A Gigabit PMC ethernet card.
- ★ **DDD** The Data to Disk Distributor. Even though there is a DDD machine in the current system this one is marked with a ★. The PCInt poses new requirements on the DDD machine and on top of that it is anticipated that we need more than one machine to cope with the datarate.
- ★ **EEE** Evaluation and Export Engine(s). These machines are the ones where the software runs that does the actual computations on the captured data. In order to keep processing time within acceptable limits it is assumed that

multiple machines will be running in parallel to work through the data.

★ **Raid Array** SCSI Disks with RAID controllers feature flexible fast transparent accessible voluminous storage. Just what's needed. Maybe, cheaper IDE disk arrays will be able to do the trick as well.

★ **Switch** Switch(es) inserted in the datapath between the SBC outputs and the DDD machine(s) to ensure maximum throughput and flexible signal routing.

★ **FC Switch** FibreChannel switch inserted between the DDDs, EEEs and the RAID Arrays.

As is evident from this section, a lot of the necessary hardware is not available in the current system. This design of the PCInt is scalable; it is possible to implement it in (at least) two stages. Below these two stages are described.

4.1.1 Stage 1

The SBC comes with an embedded 10/100Mbit/s ethernet controller. We propose to use the 100Mbit link to enable subsecond integration times for the full correlator. It should be 'trivial' to sustain 0.25s integration with the full correlator (trivial in the sense of datarate, not necessarily implementation). 0.25s integration time means a total output datarate of the correlator on the order of 10MByte/s. Per SBC that boils down to 1.25MByte/s which is easily sustained by the 100Mbit link.

Furthermore, the DDD host which is responsible for flushing the data to disk can be just a single machine. Modern IDE disks can sustain 20MB/s. As far as ethernet hardware required the following is envisaged: the eight 100MBit/s links go into a single switch with also a 1Gbit/s link, to which the DDD host is attached. In this stage we have a single DDD host receiving all the data.

4.1.2 Stage 2

In the next stage we will upgrade the PCInt to accomodate the full datarate. The SBCs will be fitted with Gigabit ethernet cards, a gigabit switch needs to be acquired and mass storage (RAID array) must be attached. The existing 100Mbit links remain in use as the route via which the SBCs boot and via which control information can be sent/received.

It may prove to be necessary to have multiple DDDs in place in order to sustain the desired 160MByte/s datarate. Since the amount of data produced is likely to grow non-linearly, it is advisable to have multiple EEEs in place. It may be possible to have just a pool of machines which can be used for DDD tasks as well as for EEE tasks. There may be no compelling argument to have a set of machines dedicated to DDD tasks and a set of machines dedicated to EEE tasks. Each processing/datacapture node needs to be fitted with a fibrechannel adapter and S/W to access the storage concurrently with the other nodes in the

system (for point-to-point connections this arbitrating software is not necessary, however, we're not sharing diskspace either in that case).

5 Effort involved

This section will cover all tasks that need to be done in order to arrive at a production capable system.

It makes sense to split the work into different parts: a system independent part and an application specific part. This is to ensure that the DZB correlator can use this system as well as the EVN correlator.

The system independant part comprises of the work that needs to be done in order to get the correlated data stored somewhere on disk in some pre-defined format. After that, application specific processing takes over.

Furthermore, it makes sense to investigate which tasks are needed for the two different stages, mentioned in the previous section.

The target is to have a system ready by the end of stage 1 which has been designed and implemented in such a way that the transition from stage 1 to stage 2 is really an upgrade and should not introduce more development. This can be achieved because the conceptual difference between stage 1 and stage 2 is zero: they are the same. The time it takes to do the transition from stage 1 to 2 wholly depends on how well stage 1 has been implemented.

5.1 The system independent part for Stage 1

The effort described in this section ensures that the data will eventually end up somewhere on disk, ready for further processing.

For the test setup we require a working VME rack + HPUX RT System + correlator boards + input board + SBC. All but the SBC is already up and running.

Furthermore, we need a standalone Linux machine (as a bootp-server and test DDD host) in order to boot the SBC and control it. Currently, a standalone Linux machine is up and running.

Action **1** ⇒ We are left with ordering at least one SBC and a PMC high speed serial link module.

5.1.1 The SBC

- The SBC must be booted with Linux. This must be set up/figured out. We may need to build a kernel ourselves first.
- The SBC must be able to communicate with the RT System in the rack via the VME bus. This requires a technical part (dealing with the VME bus) and a logical part. The technical part must be developed. The logical

part can be taken from the existing code since a communication protocal has already been developed in order to control the PCInt DSP boards.

- A driver for the high speed serial links must be written. I think it is safe to say that there is very little chance that there is a Linux driver for this specific PMC card. The driver will enable us to read the data off the module into the SBCs main memory.
- A task must be written that runs on the SBC which is responsible for the following modules:
  - accepting commands from the RT System
  - read data from the serial links
  - sort the data (would be very nice if that could be done on the SBC, rather than offline) this requires that there must be a mechanism via which the SBC receives the sorting information from the RT.
  - tag the datablocks with an identiefier that uniquely identifies the block (such that later on, offline task can find back where the data is/came from)
  - sends the data out over an ethernet interface Optimally, the SBC is passive in this. It listens if somebody desires to make contact with it. If somebody does, they will receive the data. This enables a mechanism where the DDD host(s) decide who reads which SBC(s) and where the SBC(s) push the data across to the DDD host(s).
  - read out the correlatorboards via the VME bus and/or output the data via the VMEbus to the RT host in the rack. The technique to do either should be rather trivial, it needs to be discussed how to control the SBC task to make it read/write via the VME bus.

5.1.2 The DDD host

In this section it is assumed that there will be some form of interaction between the Online Software and the DDD host (see later on for details on this).

Given the fact that it must be possible to run all of the corrections offline as well as doing them on the fly (if datarate low enough) and that we may want to switch off on-the-fly normalization etc., e.g. just the raw correlator counts are wanted, it makes sense to design a system where the capture software (the equivalent of the current EVNFRA_TRANSFER routine) and the processing software share a lot of functionality. A more elaborate discussion follows under section 5.2.

At minimum, the capture software should feature the following functionality

- A basic command interface should be up and running, e.g. to shut down the system or maybe even to start it up, preferrably also remotely. By that it is meant that the operator does not have to log on to the machine

and start processes manually but that operations can be started from a central place, e.g. CCC.

- It is desirable to have some 'autodetection' protocol in the system. The DDD hosts can then dynamically keep track of which SBCs are online and which aren't. Simple UDP broadcasts spring to mind.
- The minimum information required by the DDD host is that it is informed when a new job is about to begin such that it can set up its connections and work out how to balance the load and where to put the data. The information required contains at least the following items:
  - experimentid
  - jobid
  - subjobid
  - bocfrate
  - integrationtime
  - boardmask

  These variables describe which correlator boards take part in the experiment (boardmask), also it notifies the DDD hosts which experiment/job/subjobid it is that is starting (needed for directory where to store the data) and the bocfrate+integrationtime+boardmask indicates how much data is to be expected to come off the system, which enables an algorithm to work out a load-balancing scheme (if needed).
- The DDD hosts should be informed when a job is finished such that the job can be cleaned up neatly, close files etc.
- Some (remote)monitoring scheme must be present such as to enable monitoring the sytem from a central place
- The same holds for a debugging scheme. It must be able to access the system remotely and find out what it's doing.

Note that the last two are not really necessary to make a running system but for the production system at least the monitoring should be in place.

5.1.3 The online software

From the previous section it follows that it would be nice to have some form of commucation between the Correlator Control Computer (CCC) and the datacapture hosts. As it must be possible to even control the test setup PCInt, it makes sense that the package developed for that purpose be included in the JCCS in due time. Since the control module most presumably will be a standalone module, it is to be expected that incorporation into JCCS will lead to no great problems. Furthermore it is to be expected that the online software needs to be modified in the correlator configuration area. More details on (possible) implications for the online software can be found in the tasklist, task 6e.

5.2 The application specific part for Stage 1

It is possible to define two distinct datarate domains for the DDD software. The 'low datarate' domain and the 'high datarate' domain. The difference being that we define the 'low datarate' domain to be the span of datarates for which it is possible to do the most basic corrections on the fly. Think of normalization, quantization corrections, van Vleck corrections. In the 'high datarate' domain there will be not enough CPU power to perform these calculations on the fly. The big difference between the two domains will be the format of the output data. In the low datarate domain, the system would write floating point data, whereas in the high datarate domain the system would write integers (just the counters). As a result, the datafiles must contain some description of what kind of data they contain and which processing steps have been run on them.

It is important not to forget the constraints mentioned earlier in this document regarding dataformats. The output of processed data should for convenience be almost equal to non processed data in order to minimize the number of different dataformats.

Tasks to implement and considerations to take into account encompass

- it must be easy to use
- flexible/easy to configure/control operation of all available processing mechanisms
- it must be easy to add new algorithms or other operations (loadable modules?). Maybe even make the capture software capable of using these modules
- be able to run parallel. Presumably we do not need real clustering software like e.g. Beowulf. Our needs do not require loadbalancing, fail-safety or anything fancy that is usually found in these systems.
- enable support for output to different formats?

Once these tasks have been succesfully implemented, a workable Stage 1 machine should be up and running.

5.3 System independent part for Stage 2

In order to complete the full project hardwarewise, the following should be done.

Action **2** ⇒ Fit the SBCs with PMC gigabit ethernet controllers. It will shorten development time dramatically if the card comes with a Linux driver.

Action **3** ⇒ Buy a gigabit switch with on the order of 16 ports. At least eight inputs and two or more outputs (i.e. two or more DDD hosts) are necessary.

Action **4** ⇒ Buy one or more RAID systems, the fibrechannel switch with ∼eight ports. The estimate is that four ports is not enough to accomodate more than one RAID system, more than one DDD hosts and more than one EEE host. Furthermore one fibrechannel controller and license for the S/W that enables concurrent access to the RAID systems between the machines per machine has to be aqcuired.

From the software point of view there should be little changes to be made, as argued before.

5.4 Application specific part for Stage 2

Some of the development mentioned in section 5.2 might be put off to here. At this stage it becomes really important to be able to do something to the data. Provided Stage 1 was completed according to the description given before, this should not pose a problem. It would mean enhancing functionality or adding new functionality.

6 Facts and figures

This section will discuss expected datarates, throughputs and will aim to provide justification for choosing the components described in figure 1.

It is not necessary to discuss possible astronomical requirements in too great detail. As mentioned before, for the design of the PCInt and the choice of components, it is enough to know what the maximum expected output datarate of the system will be. How the correlator is actually configured does not matter too much (e.g. think of recirculation enabled yes or no, cross polarization products yes or no).

6.1 Output of the Correlator boards

Let's define (for ease of use) the basic unit of correlation to be the full readout of one correlator board (i.e. 32 correlator chips). It is not necessary that the correlator board is fully used all the time but for the calculations we want to get a figure for the maximum capabilities of the system.

For these considerations it is sufficient to assume that all actions on the data (these can be left undefined at the moment) are synchronized by a central interrupt: the Begin-Of-Correlator-Frame interrupt, or BOCF interrupt for short. This BOCF interrupt occurs with a user-configurable frequency. In the calculations this BOCF rate will be identified as f_{bocf}, in units of Hz.

For one correlator board, a readout of all registers of all correlator chips yields 20,000 32-bit words of data. This includes 16,000 32-bit words of correlated data and 4,000 32-bit words of monitor data. This leads to a datavolume of

$$\begin{aligned} DV &= 80,000\,\text{bytes} \\ \Rightarrow DV &= 78.125\,\text{kByte per readout} \end{aligned} \tag{1}$$

The outputdatarate for one correlator board now can be linked to the actual BOCF frequency. At each BOCF interrupt, the correlator chips are readout. Recirculation might be configured. This means that the data is sent through the correlator multiple times. Let us call this the recirculation factor n_{recirc}, indicating exactly how many times. What happens is that the correlator board first does n_{recirc} readouts before one sample is complete. As a result, one sample is produced every $\frac{f_{bocf}}{n_{recirc}}$. It might be that the user desired to integrate for some amount of time. Integration is done on whole samples and it is a user-configurable parameter, n_{intg}. n_{intg} will indicate how many samples the user wishes to integrate. Consequently, at the output of one correlator board we will

measure a datarate equal to

$$\begin{aligned} DR &= \frac{(DV \cdot n_{recirc}) \cdot (\frac{f_{bocf}}{n_{recirc}})}{n_{intg}} \\ &= \frac{(DV \cdot f_{bocf})}{n_{intg}} \\ &= 78.125 \cdot \frac{f_{bocf}}{n_{intg}} \,\text{kByte/s} \end{aligned} \tag{2}$$

As is clearly visible from equation 2, indeed the recirculation factor has no effect on the output datarate.

It is now possible to estimate worst case (i.e. maximum) datarate figures for both the EVN and DZB correlators. From a datarate point of view the main difference between the two is the maximum bocf rate; $f_{bocf} = 64\,\text{Hz}$ for the EVN correlator and $f_{bocf} = 100\,\text{Hz}$ for the DZB. Assuming no integration takes place, the following maximum expected datarate per correlator board result

$$\begin{aligned} DR_{EVN} &= (78.125 \cdot 64)\,\text{kByte/s} \\ &= 5,000\,\text{kByte/s} \\ &= 4.88\,\text{MByte/s} \end{aligned} \tag{3}$$

$$\begin{aligned} DR_{DZB} &= (78.125 \cdot 100)\,\text{kByte/s} \\ &= 7812.5\,\text{kByte/s} \\ &= 7.63\,\text{MByte/s} \end{aligned} \tag{4}$$

6.2 The SBC

6.2.1 Into the SBC

The high speed serial link between the correlator board and the SBC has a listed bandwidth of 20 MByte/s. Each PMC-COMM module has four of such links. Each SBC will receive data of four correlator boards. The manual of the PMC-COMM module states that the bandwith decreases to $\sim$ 15 MByte/s if all four channels are used in parallel. Therefore the assumption is that 15 MByte/s will be the maximum throughput of the serial links.

It is unlikely that the system can sustain this maximum throughput. 50 % of the maximum throughput is more likely to be a value that can be sustained. This leads to an estimated sustained throughput per serial link of

$$BW_{comm} = 7.5\,\text{MByte/s} \tag{5}$$

This leads to the following conclusions re. performance of the COMM port

- For the EVN no problems are expected with the COMM ports since the expected maximum datarate is $\sim$ 60% of the estimated sustained throughput

- For the DZB maximum throughput may prove to be difficult since the maximum expected datarate is practically equal to the estimated sustained datarate. It remains to be seen if the COMM ports can actually sustain 7.5 MByte/s.

As it is also possible to read the data into the SBC from the correlator boards via the VME bus, we can estimate (based on the performance of the VME bus) what datarate can be achieved using this route. Tests by Albert Bos have shown the VME bus to be capable of transferring data at a rate of ~ 7 MByte/s. From this we can draw the following conclusions:

$$BW_{VME} = 7.0\,\text{MByte/s} \tag{6}$$

$$BW_{avail,corrboard} = 0.875\,\text{MByte/s, 7MByte/s for 8 boards} \tag{7}$$

$$\Rightarrow Nreadout_{max} = \frac{0.875}{DV}$$

$$\Rightarrow Nreadout_{max} = \sim 11\text{, using 1}$$

$$\Rightarrow T_{int,min,vme} = \frac{1}{Nreadout_{max}}$$

$$\Rightarrow T_{int,min,vme} = \sim 0.1\ s \tag{8}$$

6.2.2 Inside the SBC

Once the data have arrived on the PMC-COMM module, the data have to be transferred to the SBC's main memory. From there it must be transferred to the output device. All devices on the SBC are connected to the system bus, a 32 bit/32 MHz PCI bus. The theoretical maximum throughput of such a PCI bus is 132 MByte/s. Experiments by others ([De Vos/Kant], [MkV recording]) have shown that is possible to sustain at least 60 MByte/s over this bus.

As noted before, the data have to be sent over the PCI bus twice. This leads to the following total datarate that has to be sustained by the PCI bus on the SBC (remember that one SBC has to deal with the data of four correlator boards)

$$\begin{aligned} DR_{PCI,EVN} &= 2 \cdot (4 \cdot DR_{EVN}) \\ &= 2 \cdot (4 \cdot 4.88)\,\text{MByte/s, using eq. 3} \\ &= 39.04\,\text{MByte/s} \end{aligned} \tag{9}$$

$$\begin{aligned} DR_{PCI,DZB} &= 2 \cdot (4 \cdot DR_{DZB}) \\ &= 2 \cdot (4 \cdot 7.63)\,\text{MByte/s, using eq. 4} \\ &= 61.04\,\text{MByte/s} \end{aligned} \tag{10}$$

Comparing equations 9 and 10 to the measured sustained throughput of 60 MByte/s, the following can be concluded

- For the EVN it seems that sustaining the maximum datarate is feasible since the maximum datarate is $\sim 66\%$ of the measured sustainable rate.

- For the DZB, again, it seems that sustaining the maximum datarate remains questionable. Again, the expected maximum datarate is practically equal to the measured sustainable throughput.

6.2.3 Out of the SBC

For the output of one SBC, two envisaged media are envisaged. The first will be a normal 100Mbit/s ethernet link. The second will be a Gigabit ethernet adapter. Other media exist but ethernet is flexible, (relatively)cheap and will meet our requirements.
Condider the maximum output datarate of one SBC. As stated before, one SBC will be responsible for handling data of four correlator boards, hence we find

$$\begin{aligned} DR_{output,EVN} &= 4 \cdot DR_{EVN} \\ &= 4 \cdot 4.88\,\text{MByte/s, refer eq. 3} \\ &= 19.52\,\text{MByte/s} \end{aligned} \tag{11}$$

$$\begin{aligned} DR_{output,DZB} &= 4 \cdot DR_{DZB} \\ &= 4 \cdot 7.63\,\text{MByte/s, refer eq. 4} \\ &= 30.52\,\text{MByte/s} \end{aligned} \tag{12}$$

Compare this to the estimated throughput of the ethernet connections

100Mbit For a 100Mbit link, using TCP/IP protocol, the theoretical throughput is $\sim 9.5\,\text{MByte/s}$. However, a more reasonable figure for sustained throughput would be 50% of the maximum throughput, i.e. $\sim 5\,\text{MByte/s}$.

Gigabit or, 1000Mbit. Basically, the estimates are just a factor 10 higher, amounting up to $\sim 50\,\text{MByte/s}$. This is a very low estimate since [De Vos/Kant] have shown that 60 MByte/s can be sustained over Gigabit ethernet.

It is obvious that a 100Mbit link cannot be used for the maximum datarate. In both the EVN and DZB cases integration has to take place. It is possible to estimate the lowest integration time for both systems. Since integration time, bocf rate and output datarate are linked together by eq. 2 and the fact that we want to limit this value to $1.25\,\text{MByte/s}$ $(= \frac{5\,\text{MByte/s}}{4\,\text{correlator boards}})$ it follows

$$\begin{aligned} 1.25 &\geqslant 7.63 \cdot 10^{-2} \cdot \frac{f_{bocf}}{n_{intg}} \\ \Rightarrow \frac{f_{bocf}}{n_{intg}} &\leqslant 16.4 \\ \Rightarrow \frac{n_{intg}}{f_{bocf}} &\geqslant \frac{1}{16} \end{aligned} \tag{13}$$

Equation 13 is in units of time. It means that whatever the bocf rate is, the integration time must be $\geqslant \frac{1}{16}^{th}$ of a second in order for the 100Mbit link to be able to sustain the datarate.

In the case where the data is read into the SBC via the VME-bus, again the 100Mbit link seems to be the limiting factor. In this case, only one SBC will be used, hence only one 100Mbit link will be available for transport to the DDD host(s). As the estimated throughput of the 100Mbit link is $\sim 5\,\mathrm{MByte/s}$ and the fact that this speed is lower than the VME-bus speed, if follows that we cannot sustain $0.1\,s$ integration with this system. We find

$$\begin{aligned} 5.0 &\geqslant 8 \cdot 7.63 \cdot 10^{-2} \cdot \frac{f_{bocf}}{n_{intg}}, \text{ using 7} \\ \Rightarrow \frac{f_{bocf}}{n_{intg}} &\leqslant 8.2 \\ \Rightarrow \frac{n_{intg}}{f_{bocf}} &\geqslant \frac{1}{8} \end{aligned} \tag{14}$$

In short, an integration time of $0.125\,s$ should be feasible.

For the Gigabit ethernet adapter the actual estimate is not all that important since either of the estimates is well beyond the maximum of the two anticipated maximum datarates. The 'worst case scenario' is the DZB situation, where the SBC has to deal with 30 MByte/s. This required bandwidth is $\sim 60\%$ of the lowest estimate for the sustained throughput hence no problems are expected here for both systems.

6.3 Output of the full correlator

As the system contains eight SBCs it is possible to work out the requirements for the offline system. Evidently it has to be able to deal with the combined output of all the SBCs. The total maximum output datarate of the two systems is

$$\begin{aligned} DR_{EVN} &= 8 \cdot DR_{SBC,EVN} \\ &= (8 \cdot 19.52)\,\mathrm{MByte/s}, \text{ using eq. 11} \\ &= 156.16\,\mathrm{MByte/s} \end{aligned} \tag{15}$$

$$\begin{aligned} DR_{DZB} &= 8 \cdot DR_{SBC,DZB} \\ &= (8 \cdot 30.52)\,\mathrm{MByte/s}, \text{ using eq. 12} \\ &= 244.16\,\mathrm{MByte/s} \end{aligned} \tag{16}$$

These numbers indicate the potential of both these MkIV correlators. However, it must be noted that this is lag-data. As such, the net scientific output is half the numbers given in eqs. 15 and 16.

Moving these amounts of data around in a network should not pose real problems, provided that the network infrastructure is Gigabit ethernet based. This includes Gigabit switche(s) and multiple datacapture hosts. Multiple switches may or may not be necessary, depending on the actual design of the switch. Multiple capture hosts will be necessary to sustain the maximum datarate. More on that topic in the next section.

6.4 Capture and Storage

Following the data downstream, two hurdles have yet to be taken. Capturing the data that has been sent across the ethernet and storing it on disk. As the total datarate of either system is very high, the datacapture hosts and mass storage device(s) attached to them must be able to deliver this. The requirements pose a firm constraint on the design of the infrastructure.

6.4.1 The datacapture hosts

First the datacapture hosts are encountered. Since all of the network will be Gigabit ethernet based, each datacapture host will have to be fitted with a Gigabit ethernet card. For the moment let's consider standard PCI based workstations (cheap, readily available and easy to develop upon). Estimates in previous sections (see secs. 6.2.2 and 6.2.3 for PCI bus performance and Gigabit ethernet throughput) show that one datacapture host should be able to capture data for two SBCs. This leads to the conclusion that, in order to sustain the full datarate, four datacapture hosts need to be in place.

Again it must be noted, that the situation for the EVN is more likely to be sustainable. The requirements for the DZB system are (again) quite close to either the measured sustainable throughput or the theoretically estimated throughput.

6.4.2 The Storage subsystem

As for the storage subsystem not many candidates are available, the most logical choice seems to be a SCSI based RAID array. It must be noted that these RAID arrays are not cheap. However, they are the sole solutions that will surely meet our requirements. It seems appropriate to use RAID arrays in a so called Storage Area Network (SAN). The difference between plain RAID systems and SAN is easily explained as follows. Raid arrays can be attached to a single host machine via a 'normal' RAID controller. Now the RAID array is only accessible from this specific host. In order to circumvent that, vendors attach the RAID controller to a Network Interface Card (NIC) and put the whole assembly in a single box. Nowadays, this NIC most often is a FibreChannel controller. As a result of this solution, the storage has become an entity in the network. The storage logically resides at the same level as a hub, router or workstation. The connection from a host machine to the storage is then routed via the FibreChannel network rather than a direct connection via the PCI bus/RAID controller.

Some of the unique features of these SAN-RAID arrays include

- Scalability: at any time, the storage capacity can be enlarged simply by putting in more disks (provided the device is not full yet).
- Speed/Safety: these systems are specifically designed to either get the most performance or the most safety out of the system. Evidently, the focus in the case of the PCInt is on speed.

- Flexibility: since these SAN-RAID arrays are separate units, i.e. not attached to any particular computer, the storage can be easily shared transparently between multiple machines.

Measurements have shown (see [Rorke Galaxy45]) that write transfers up to 135 MByte/s can be sustained by these arrays. Again, it might be safer to assume a 60 % of that, resulting in 80 MByte/s. This would indicate that in order to sustain the full datarate for the EVN, two parallel RAID arrays would be necessary. For the DZB situation, three parallel arrays would be necessary. However, this might prove to be impossible to configure since there are four datacapture hosts. It would mean that the data has to be split. Easier would be to implement four parallel arrays. If that can be done is fully dependent on the financial situation.

Concluding, it may be stated that the total datarate sustainable by the system can be expressed as follows

$$DR_{total,sustainable} = n_{array} \cdot 80\,\text{MByte/s} \tag{17}$$

where n_{array} is the number of parallel RAID arrays present in the system.

7 Targets, Tasks and Deliverables

This section aims to provide a timeline, a list of work-packages, milestones and decision points. This is to have a clear overview of the work involved and events by which it is possible to track progress in the development. Also, some of the vagueness of previous sections is removed. The discussions in previous sections mostly listed an argumentation with optionally some alternatives, suggestions or enhancements that could be thought of. This section aims to provide a clear road to a defined system.

Since it is difficult to present parallel tasks in a textdocument like this, the presentation might be somewhat different than when looking a project plan. The goal is to order identified tasks logically together. Where possible parallel development is an option, it will be mentioned explictly.

7.1 Preliminaries for the test setup

Identified tasks for constructing a test setup incorporate

1 Getting the correlator testrack back online. The rack has been switched off for a year. In the meantime the network situation re. the HP Realtime system booting from daw02 has changed. Includes

- **a** The development PC (with TexasInstruments' CodeComposerStudio software installed) needs to be found and installed near the test rack.
- **b** It must be verified that the PC can communicate with the correlator board DSPs via the JTAG emulator.

2 There must be a Linux PC installed near the test rack. This machine will be used as control and datacapture host. As such the following needs to be fulfilled

- **a** Before one or more SBCs become available, set up this PC to have two ethernet cards. One will be used for setting up a private network with all the SBCs and the control PC in it. The other will preferrably be used to connect the control PC to the local NFRA network (needs to be discussed with systeembeheer). The idea is that network traffic from and to the SBCs does not interfere with the NFRA network.
- **b** The SBCs need to boot from a remote computer. This control PC will act as the bootp-server. This must be set up.

3 It is necessary to select and buy at least one SBC so development in that area can start.

4 The PMC-COMM module can be ordered. However it is not as urgent as the SBC. Development of the SBC software can start long before a PMC-COMM module becomes available (see task 6f).

7.2 Making the test setup work

In order to make the test setup work, a lot of software needs to be written. Development of some parts of the software can start before a SBC or PMC-COMM module is available.

5 Since it is likely that the capture software and the processing software will become complex projects, a solid make-system will have to be set up. This make-system should include support for

- **a** easy adding of third party libraries and easy control of where these support libraries are located.
- **b** easy adding of libraries, modules and applications. This encompasses that new code will be picked up automatically by the system.
- **c** it must run under Linux as well as under Solaris (Sun).

6 A lot of work can be done without hardware really being available. Actually, these tasks preferrably should be done without using any hardware! Things that spring to mind are design issues like

- **a** infrastructure/datapaths. Think about and write up how the envisaged system should be controlled. It must be defined how to move information around the system. Discuss which parts in the system need which kind of information to operate properly. How to set up control paths to all elements. Details of communication messages need to be thought of. Discuss protocols for communication. Also, discuss a protocol for autodetecting which hosts in the network are available. It enables the software to dynamically find out which SBCs, DDDs and/or EEEs are currently online. Compare to e.g. the ARP protocol for ethernet networks.
- **b** interface. How to interface to the user. Identify which parts need to be interactively controlled and which parts dont. Think of the processing software. How to indicate which processing is desired and how to optionally control the parameters of the processing.
- **c** SBC task. Think of a design how the SBC reacts to commands and how it makes things happen. This propably involves Inter-Process Communication (IPC) as there may be different processes running on the SBC, e.g. the process that monitors the VME bus for commands, the process that monitors the ethernet for incoming connections.
- **d** dataformats. Define how the data is represented at each stage in the process. Includes description of diskformat after dumping and/or processing.
- **e** operational environment. It must be discussed and defined how operation of the system is envisaged. Questions that have to answered include how does one indicate that the PCInt should be used? Do we want to be able to switch between using/not using the PCInt at all? Does the PCInt disrupt current operational software? There are a number of programs in

the operational environment that rely on output of certain programs. Furthermore, the administration of jobs that have been correlated might be inflicted. Backup procedures might have to be reviewed and investigated. Error checking: how does JCCS react if not all of the hardware appears to be present. At what level of the JCCS should the PCInt be visible and to what extent should it be controllable/monitorable? The advent of the PCInt also has implications for the data_handler/CDI in JCCS. These processes migh time out since they will not be used if the PCInt is used. In short: investigate the impact of the PCInt and identify where in the JCCS awareness of the PCInt is needed.

Apart from designing, implementing software could already start since the SBC will be running Linux. Given that the high level software will be coded in C++ and provided that the proper abstractions are made in the design, a lot of high level software can be developed and tested, even without the actual hardware in place. The object oriented nature of C++ allows for easy implementing fake hardware. Softwaredevelopment can be started on

f A framework for the SBC software. Abstractions can be made for control channel, data-input path, data-output path and optionally some processing (sorting). . This enables good and easy testing of the logical level of the software.

g A framework for the processing software can be designed. Interfaces between the individual processing modules can be defined and some default/test modules can be developed to test the functionality and the control of the system.

7 When a SBC comes available, work must start on the following issues

a The SBC must be made to boot off a bootp-server. Once the SBC boots a Linux kernel, development could start on the SBC-specific software.

b Get the VME driver to work, i.e. figure out how to program the VME controller such that communication between the HPUX-Realtime system in the correlator rack and the SBC is possible.

c Test and verify the control path RT⇔SBC.

d Implement, test and verify the control path DDD⇔SBC.

e Implement, test and verify datapath DDD⇔SBC.

f Implement, test and verify datapath CorrelatorBoard⇔SBC. It must be noted that this cannot be done unless a PMC-COMM module is installed and a driver is available (see task 8). Also, the data must be sorted by the SBC. This is an intermediate step between receiving the data and sending it out again.

g Heat production and powersupply issues must be investigated. The correlator racks of the production correlator must be upgraded; a new powersupply must be installed.

Of course, not only the SBC software needs to be written, the existing control path from CCC→RT→PCIntDevice must be validated.

h The currently used control software (dzbtp, dzbsp) might have to be changed to reflect the change in PCInt hardware.

i Functionality may have to be added to support operation/control of the PCInt

8 If a PMC-COMM module is available and installed on the SBC work can start on coding a driver for the module. This driver makes the hardware available to the Operating System of the SBC, Linux. Once operation of the device is available communication between the CorrelatorBoard and the SBC can be tested and implemented.

By now, if all previous tasks have been completed, development can continue on further. More specifically, the following tasks will focus on operating a test version of the system.

9 Develop a module that really controls the SBCs from a central place. This means implementing a scheme that should have been thought out in task 6a.

10 The SBC control module must be functional and must respond to the various commands (via ethernet as well as via the VME bus).

11 Develop a module that controls the DDDs from a central place.

12 The DDD task must be functional, in that it must respond to configuration commands, it must be able to start and stop capturing data.

13 Some elementary processing modules (maybe even non-functional ones) should be available and should eventually be applied on-the-fly, in order to test the logic of the system.

14 A basic off-line utility should be produced, also to test offline processing of the data.

It would be nice to have multiple SBCs and/or DDDs available by this time. Not necesseraly all hardware but just to find out if the control modules, the datapaths etc. all deal correctly with multiplicities > 1. Most notably, the multiplicity of the SBCs is more important at this stage since it was envisaged that at the end of Stage 1, a moderate datarate (~ 10 MByte/s) could be sustained. This requires all the SBCs to be in place but there is no need yet for multiple DDDs since a well equipped workstation should easily be able to deal with this datarate.

7.3 Migrating from test setup to Stage 1

Obviously, some more work must be done before the test setup can be declared to have reached the end of Stage 1. As noted before it requires (hardwarewise)

15 Buying all SBCs and PMC-COMM modules.

16 Buying and installing powersupplies in the correlator racks.

17 Installing the PMC-COMMs on the SBC and install the SBCs in the correlator racks.

18 Buy a multiple 100 Mbit/s $\Rightarrow$ 1Gbit/s switch and cabling.

19 Buy a well equipped workstation and install it in the basement as DDD. It should feature (besides the usual equipment)

- Gbit ethernet card
- 10/100Mbit card to attach the machine to the local NFRA or basement network
- A lot of diskspace
- Maybe multiprocessor

Also software effort is involved in migrating from the test setup to Stage 1. Identifiable task are

20 Do incorporate the PCInt control module in JCCS.

21 EVNFRA_SETUP, which is the routine via which CCC controls the Correlator setup, must be adapted to be able to configure the PCInt.

22 JCCS must be PCInt aware and comply to the requirements mentioned in or following from task 6e.

23 The processing software **must** be able to at least do basic processing like van Vleck correction, quantization correction and normalization.

24 The offline software j2ms2, which is responsible for translating correlated data to AIPS++ MeasurementSets must be changed such that it can interpret the output of the DDD software. This will enable easy inspection of the data as well as the possibility of shipping out the data to the PI.

Provided that these tasks are fulfilled, the PCInt has now reached an operational stage. It should be feasible to make regular use of the PCInt from this stage on.

7.4 Upgrading from Stage 1 to Stage 2

Most of the issues that need to be dealt with in order to faciliate this step were already discussed in section 5.3. For sake of completeness a list of tasks is presented here

25 Select, buy and install PMC-Gigabit ethernet adapters. Take care that they come with Linux support.

26 Buy a Gigabit switch, adapters and cabling.

27 Buy a number of rack-mountable workstations.

28 Buy mass storage devices, fibrechannel controllers for the workstations and software that enables concurrent access to the storage from multiple hosts.

After setting up and installing the hardware, only little configuration changes should be necessary in order to change to using this setup. The fact remains that if Stage 1 was completed satisfactory, upgrading from Stage 1 to Stage 2 really involves nothing more than installing bigger and faster hardware.

Contents

References

[De Vos/Kant] Smaller Distances Interconnect, Figure shown at Lofar Meeting, May 8 2001, Dwingeloo.

[MkV recording] MkV recording of 512Mbit/s using a disk based system

[Rorke Galaxy45] Document describing properties of the Galaxy 45 RAID array, available in PDF from http://www.rorke.com