ASTRONOMY AND SPACE SCIENCE
www.csiro.au

ADE Frequency Configuration and Zoom Modes

Keith Bannister

ASKAP Commissioning and Early Science Memo 016
April 20, 2017

CSIRO Astronomy and Space Science
Cnr. Vimiera and Pembroke Roads
PO Box 76, Epping, NSW 1710, AUSTRALIA Telephone : +61 2 9372 4100
Fax : +61 2 9372 4310

Copyright and disclaimer

Important disclaimer

Contents

Summary

The ASKAP Design Enhancements (ADE) system is a direct-sampled system which has very flexible frequency configuration. We describe the system from the point of view of understanding how the frequencies and zoom modes work, and discuss various options on how the appropriate control and metadata can be generated and made available to various components. We describe some use cases on how the system will be used from the perspective of an astronomer, and provide examples. The aim of this document is to describe the capabilities of the system, and help inform future design choices.

Contents

1 Direct-sampled systems

Traditional radio astronomy receivers translate the input signal from the sky frequency down to a lower frequency, with a process known as heterodyning: namely, mixing (or multiplying) the sky signal by a local oscillator (LO) with a specifically chosen frequency. The product signal contains the signal at the sum and the difference of the sky frequency and the LO frequency. The sum or difference product can be filtered to give a signal that can be sampled. In most cases, the sampling rate is much less than the sky frequency.

Direct-sampled systems have no LO or mixers. Instead, the samplers sample the incoming signal directly. They work because the samplers have an analog bandwidth that is larger than the sky frequency. This is the case for ASKAP which operates in the frequency range 0.7–1.8 GHz.

Another key concept is that of Nyquist sampling. The Nyquist sampling theorem states that, to recover all the information in a given signal with power in the frequency range $[0, B]$, it must be sampled with a sampling rate of $f_s = 2B$. Perhaps less obvious, is that a signal with power in the frequency range $[B, 2B]$ can also be sampled with the same sampling rate: $f_s = 2B$ (!). In fact, as long as all the signal power falls in one of the kth Nyquist zones, defined as $[(k-1)B, kB]$ with $k \in 1, 2, 3...$, a signal can be sampled with a sampling rate of $f_s = 2B$ without aliasing.

2 ADE: General principle of operation

ADE operates using the direct sampling principle. The samplers sample at a rate comparable to the sky frequency (either 1536 or 1280 MHz). For a desired sky frequency, the incoming signals are analog filtered to be in one of the 2nd or 3rd Nyquist zones and sampled. Not all of the resulting bandwidth can be processed downstream. So, the sampled signal is channelised, and only a subset of the channels (nominally 300 of them), is selected for downstream processing.

Downstream processing can be configured to split the coarse channels into a configurable number of fine channels, as desired. The choice of resolution in the fine channels is known as 'zoom mode'. Only 54 channels from the output of the fine filterbank are sent to the correlator, and the remaining channels are discarded. In the case of the standard 18 kHz channels (no zooms, $N = 1$, See Table 3.1), the 10 discarded channels are duplicated in other coarse channels, and can be discarded without loosing total bandwidth. For finer resolutions ($N > 2$), the total bandwidth is reduced commensu-

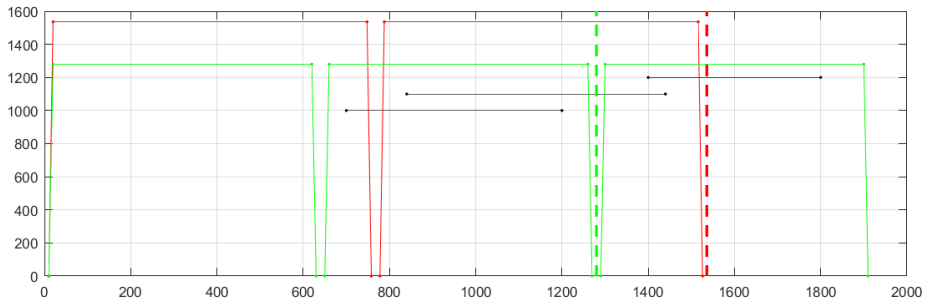

Figure 2.1: ADE Bands (from John Tuthill). TODO: Add labels and annotations.

rately, and additional configuration is required to determine which set of 54 channels will be processed by the correlator.

3 Data flow

In this section we describe how the system works, with reference to Figure 3.1.

Starting with a photon trundling it's way across the universe...

The PAF element senses the electric field at the focal plane. PAF elements are sometimes referred to as 'ports'.

The PAF domino contains the LNA, which amplifies the signal, a set of 4 selectable filters, which select the observing band, and an RF over fibre transmitter (RFoF). The RF signal is sent directly from the domino over up to 3 km of fibre to the control building.

The digital receiver (DRX) chassis converts the RFoF back to an electrical signal, filters the signal with selectable select filters, and samples it. The sampling rate is set to either 1536 MHz or 1280 MHz, depending on the desired observing band. Each sampling rate requires a separate firmware build. Once the signal is in digital form, a polyphase filterbank channelises the data into either 768 or 640 (respectively) 1 MHz channels. All channels are sent to a configurable 16x24 channel multiplexer, which can send any combination of 16 channels down any of 24 different fibres. Each fibre carries 16x1 MHz channels for 16 ports. The fibres themselves are bundled into 2 groups (ribbons) of 12 each. A DRX chassis processes 16 ports and outputs a total of 384x1 MHz channels. 12 chassis are required to process the 192 ports (188 PAF ports + 4 spares) in an antenna.

Two optical cross connects take a 12 way ribbon from each of the 12 DRX chassis in an antenna and do a full transpose. Each output ribbon now carries 16 MHz

of bandwidth for 192 ports.

A beam former chassis takes 3x12 way ribbons from a cross connect. The bandwidth from 3 ribbons are split by 2, to enable the 6 FPGAs on the beam former to operate independently on 8x1 MHz each, in two parallel chains operating on 4x1 MHz channels. Each FPGA does a beam forming operation, using weights supplied externally, to weight 192 ports into 72 beams in 1 MHz channels. Configurable coarse delays are applied, before a fine filterbank on the 1 MHz beam formed data, which produces 64N channels, where $N \in 1, 2, 4, 8, 16, 32$, depends on the desired zoom mode. $N = 1$ is colloquially known as 'no zooms'. The fine filterbank outputs only 54 contiguous channels irrespective of zoom mode. The choice of which 54 channels is set by software configuration. Fringe rotation is applied to the fine filterbank channels. Each output fibre carries 4x54=270 channels for 72 beams for a single antenna. Each antenna requires 8 chassis to make beams for 384x1 MHz channels (we number them $n \in 1, 2, ..8$ in this document). Currently only 7 chassis per antenna are fitted.

Another optical cross connect takes a 12-way ribbon from the same beam former chassis in each of 36 antennas. Three 144x144 cross connects takes the input of 12 antennas each. Three ribbons, one from each cross connect, are connected to a correlator chassis. Each ribbon now contains 270 channels for 72 beams for 12 antennas. There are 8 sets of 3 cross connects. Cross connect n is connected to beam former n on each antenna, and supplies data to correlator block n, $n \in 1, 2, ...8$.

A correlator chassis takes 3x12-way ribbons from the cross connect, which contains 4 MHz of bandwidth for 72 beams and all 36 antennas. It does cross correlation and accumulation. The accumulated result is transmitted using UDP packets over Ethernet to a switch (not shown). A correlator block, which comprises 12 chassis, processes 48 MHz of bandwidth from beam former and cross-connect. 8 blocks are required to process 384 MHz. Currently, only 6.25 blocks are slated for installation (to hit the 300 MHz requirement).

The correlator IOC does some reordering of the data (namely collecting together all data from a single chassis) and sends nicely-formatted packets (containing sky frequency) to the ingest pipeline.

The ingest pipeline does some more munging and writes the data to a CASA measurement set.

All beams / ports stay together.

Phew!

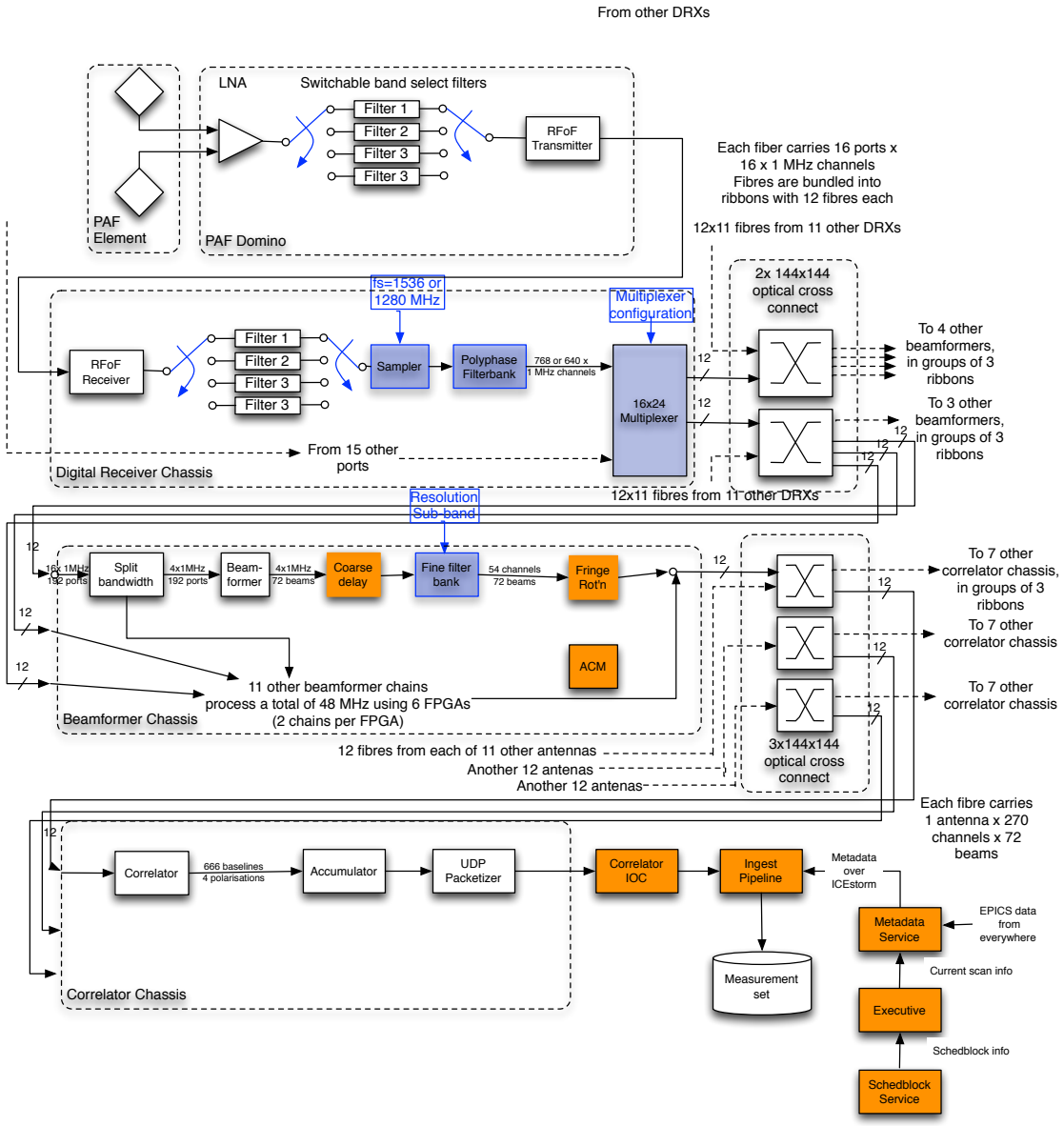

Figure 3.1: Block diagram of the ADE system. Components and data that require frequency configuration are shown in blue. Components that require knowledge of the frequency configuration are shown in orange.

3.1 Zoom Modes

ADE supports zoom modes, which are equivalent to frequency resolutions, as shown in Table 3.1. In principle, each zoom mode (N) can be set on a per-beamformer-FPGA basis. I.e. 8x1MHz channels (not necessarily contiguous) must share the same frequency resolution. The output of the beam former has a fixed *number of channels*, which means that as the resolution is reduced, so is the spanned bandwidth. Internally, the beam former creates $64N$ fine channels from a 1 MHz coarse channel. In addition to a zoom resolution (N), a set of 54-channels, known as a 'sub-band' ($0 \leq b < 2^{N-1}$) must be chosen from those $64N$ channels. For $N = 1$, there is only 1 sub-band (i.e. the $b = 0$ sub-band). Example: In order to span 1 MHz of bandwidth in the $N = 4$ mode (2.315 kHz resolution), the same 1 MHz channel must be identically beam-formed 8 separate times, each time selecting a different sub-band of $b \in 0, 1, 2, ..., 7$.

Table 3.1: ASKAP zoom modes. Resolution is rounded to the nearest Hz. Actual $N = 1$ resolution is 1000/54 kHz.

N	resolution (kHz)
1	18.516
2	9.259
3	4.630
4	2.315
5	1.157
6	0.579

3.2 What is the bandwidth / channel limit?

The total number of channels / processed bandwidth that ADE can support is a somewhat tricky question. The DRX can output 384 coarse channels, but there will not be enough beamformers or correlators to process the full band. The plan is to fit 7 beamformers, which can process 336 coarse channels (equivalent to 336 MHz for $N = 1$). The actual requirement is to process 300 MHz (at $N = 1$), which doesn't even nicely fit the native beam former FPGA step size of 8 channels. One compromise is to process 304 MHz, which is at least a multiple of 8 MHz, and requires 6.33 beamformers.

In fact, the constraint is actually the network bandwidth from the correlator to the Pawsey supercomputer. The network bandwidth is a function of the total number of channels (not the processed bandwidth). There is a little wiggle room in that number as the network to Pawsey, has some ability to be upgrade (by lighting up more fibres). Also, the quantity of metadata being sent over the link is not yet fully resolved, so it is possible that more or less of the planned bandwidth will be occupied by metadata.

For the sake of the rest of this document, we assume that the correlator is limited to output $304 \times 54 = 16416$ channels.

3.3 Fringe rotation

Need to describe how the frequency-dependent aspects work (TBD).

3.4 ACMs and weights

Need to describe how the frequency-dependent aspects work (TBD). Important: For zoom modes, the weights need to be repeated for each coarse frequency chain in the beam former which has the same sky frequency, but different sub-band.

3.5 Ingest pipeline

The ingest pipeline is responsible for taking correlator data in the form of UDP packets from the Correlator IOC, and merging with metadata (e.g. source name, sky frequency, antenna track/slew flags, uvw co-ordinates), and writing a CASA measurement set. The metadata is supplied by the 'metadata service' which assembles a structure of information from data around the system once per correlator cycle, and transmits it, via a publish-subscribe service[1], to the ingest pipeline.

Currently the frequency 'configuration' of the ingest pipeline is described entirely from a single number (central sky frequency of the first card in MHz) that is supplied in the metadata. The ingest pipeline only finds out the sky frequency when metadata arrives. The ingest pipeline is entirely unaware of the frequency configuration *in advance*.

The ingest pipeline is an MPI process that assigns 1 MPI rank per correlator chassis. Each MPI rank is, therefore, responsible for 216 channels (i.e. 4 MHz $N = 1$ 'no zooms' modes). Currently each rank allocates an array to store all channels. The current parellisation in the ingest pipeline requires that the 4 coarse channels processed by an MPI rank are contiguous, and all fine channels in those coarse channels map (via the algorithm in Section 9), to channels in the same 4 coarse channels.

4 Frequency configuration

In this section we describe a logical flow of how the frequency setup could be done for ADE. The current system, or even a future system, might not work this way, but it's useful to introduce some nomenclature at this point, so that following discussion of how things work currently, and how they might work in future, is a little easier to talk

[1] ICE - Internet Communication Service, is a standard for allowing pieces of software on different computers to talk nicely to each other (also known as Remote Procedure Call). ICEstorm is the publish/subscribe mechanism.

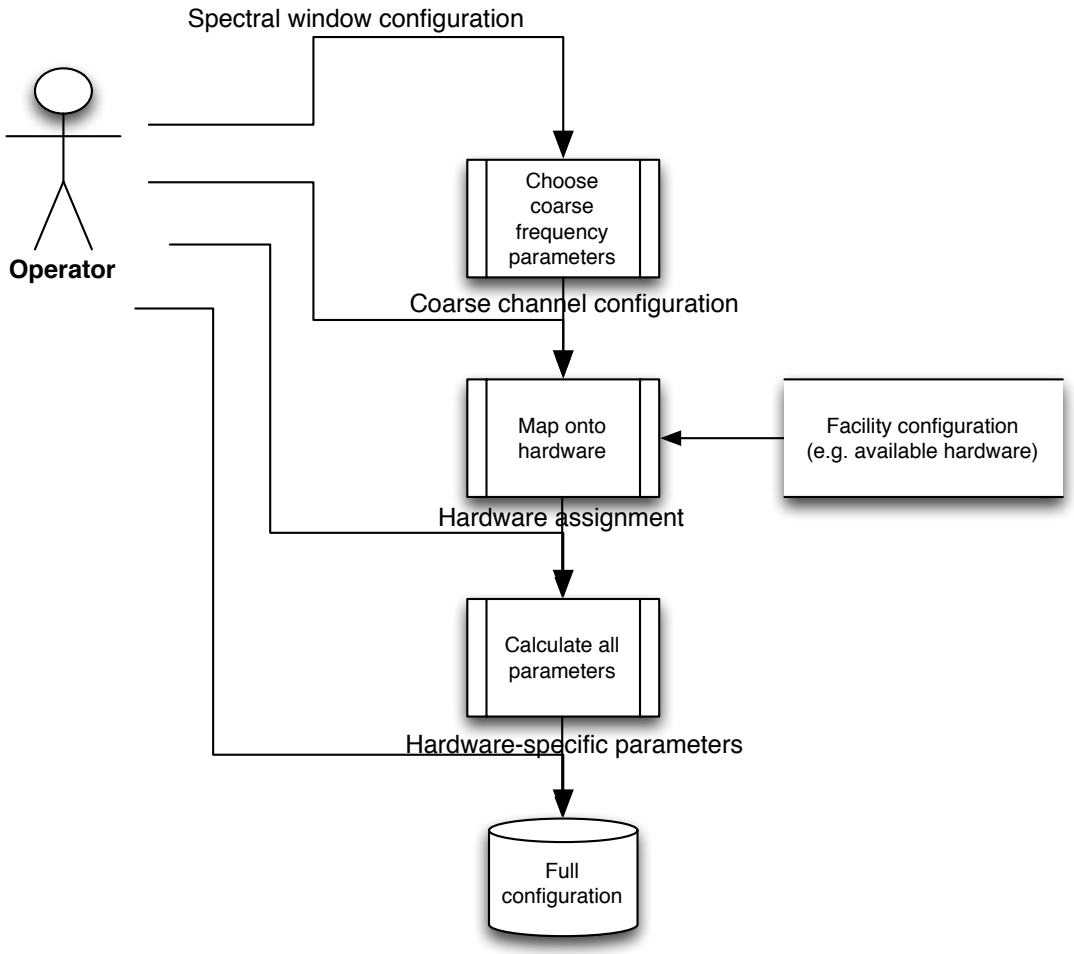

Figure 4.1: Flow diagram showing how frequencies and zoom modes should be setup.

about.

Setting up frequencies and zoom modes involves four steps, as shown in Figure 4.1.

4.1 Step 1: Start with high-level parameters

You might start with astronomer-friendly parameters: ie. Centre frequency, resolution and band width of each of ($s \geq 1$) spectral windows.

4.2 Step 2: Work out coarse channels frequency configuration

Once you have those, you can work out the *coarse frequency parameters*, i.e. the setup in terms of DRX coarse frequency channels:

1. Operating band for the whole system. See Table 4.1.
2. For each coarse frequency channel: choose a centre frequency, frequency resolution (N), and sub-band (b)

4.2.1 Constraints

Coarse frequency channels cannot be chosen arbitrarily at this point. For example, this step is constrained by the fact that beam former FPGAs operate on 8x1 MHz channels, so the same fine filterbank resolution must be used for each of the channels to be assigned to a given beam former FPGA. Thus, fine filterbank resolutions must be assigned to channels in groups of 8 coarse channels. Also, it's useful to check that all channel centre frequencies fall in the operating band.

4.3 Step 3: Map to hardware

The desired coarse frequency parameters can be mapped in a myriad of different ways on the available hardware. The next step is to map the coarse frequency parameters onto the hardware, i.e. the following *hardware mapping*

1. Decide *which* beam former FPGAs will process *which* coarse channels. Assign the fine filterbank resolution to each FPGA on each beam former, and a sub-band to each channel processed by that FPGA. Requires data from the FCM as to which parts of the system are available (e.g. correlator blocks).

Constraints:

- Beam former FPGAs operate on 8x1 MHz channels, so the same fine filterbank resolution must be used for each of the channels to be assigned to a given beam former FPGA.
- The total output number of channels is ≤ 16416 (TBC).
- The ingest pipeline requires that the 216 channels output by a correlator chassis be contiguous in frequency.

4.4 Step 4: Compute remaining parameters for use by the system

Once the coarse frequency parameters and hardware mapping have been chosen, the following *computed parameters*, which are required by other parts of the system, can be calculated (if one is careful) (the thing before the colon is the thing that *needs* the information):

1. PAF: Switch configuration
2. DRX: Switch configuration and sampling rate. See Table 4.1.

3. DRX: Memory address table for multiplexer configuration
4. Beamformer: Sub band settings (1 per channel), zoom mode (N), (1 per FPGA).
5. Ingest pipeline: Contiguous spectral window parameters for CASA measurement set header[2]. I.e. For each spectral window: number of channels, channel 1 sky frequency, bandwidth (and span??).
6. Ingest pipeline: Mapping from UDP packet headers to spectral window and channel number.
7. Ingest pipeline: UDP port numbers to listen to particular streams for
8. Correlator IOC: IP address & UDP port numbers to send visibilities to (married up to the ingest pipeline).
9. Fringe rotation control: (either ingest or correlator IOC): Centre frequency of each band being rotated (TBD)
10. CRAFT: Frequencies being processed by each beam former FPGA

Table 4.1: Operating bands and DRX and PAF filter settings. The inversion is defined at the output of the sampler.

Band	DRX sample rate (MHz)	Nyquist band (MHz)	Inverted?	Sky frequencies (MHz)	Available bandwidth (MHz)
1	1280	640-1280 (2nd)	Y	700-1200	500
2	1536	768-1536 (2nd)	Y	840-1440	600
3	1280	1280-1920 (3rd)	N	1400-1800	400

5 Existing frequency setup

Here we describe the way frequencies are handled in the ADE hardware at the time of writing (27 April 2016). It's worth pointing out that ADE is *not finished* so it's not surprising it's being done this way.

5.1 Hardware setup sequence

The Composite IOC runs setChanelsFrequency(band, bandList) with the same arguments on DRX, Beamformer and Correlator classes. The bandList is a set of 384 structures containing frequency (MHz), the desired zoom mode (one of an enumeration representing Table 3.1), which sub-band of fine frequency channels to send to the

[2]The CASA measurement set format supports non-contiguous spectral windows. In practice, no software, probably including CASA, and certainly not ASKAPsoft, support such a wacky case

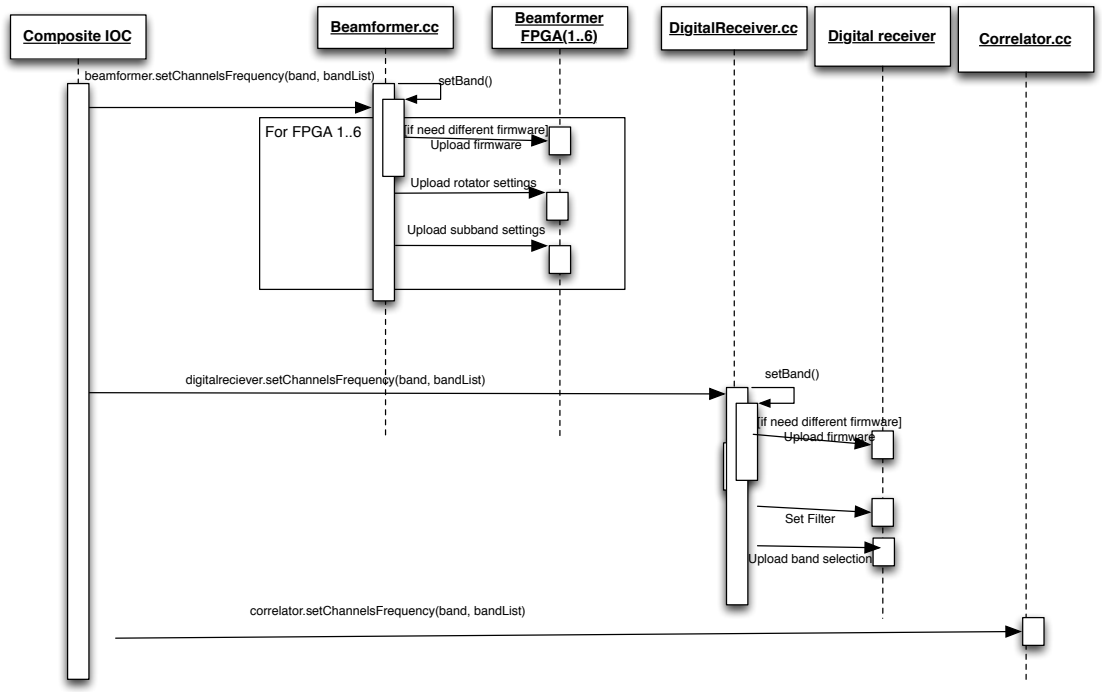

Figure 5.1: Sequence diagram showing showing how a beam former, digital receiver and correlator common software and FPGAs are configured.The composite IOC communicates with individual Beamformer, Correlator and DRX IOCs (not shown), but the logical flow is identical.

correlator.

5.1.1 DRX Setup

setChannelsFrequency(band, bandList) does a setBand(), which configures the FPGA with the correct firmware (if required) and then uploads the channel mapping to the DRX multiplexer.

5.1.2 Correlator Setup

setChannelsFrequency(band, bandList) doesn't actually touch the hardware. The incoming array is simply saved, so that the outgoing data can be tagged with the appropriate frequencies.

5.1.3 Beamformer setup

In principle the zoom can be different for each channel. In practice, the code throws an exception if the zooms are not identical for all bands.

On the beam former, setChanelsFrequency calls setBand(Band, zoom) with the zoom setup of the first channel in bandList. setBand sets the receiver filter (????!!!). It queries the current FPGAs and checks if it supports the desired zoom. If not, it chooses the bit file from a fixed set of file names strings (!) and loads it onto the FPGA.

After setBand(), setChannelsFrequency calculates rotator and sub band settings. The rotation is no-longer required, although it remains in the code due to a legacy requirement. [3].

The sub-band setting selects a contiguous set of 54 channels (a 'sub-band') from the output of the fine filterbank, splits the coarse channel into 64N channels. In the case of no zooms ($N = 1$), the fine filterbank natively produces 64 channels. The 10 discarded channels (5 either side) are in in the skirts of the DRX coarse filter response, and, more importantly, are in the passband of the adjacent channel. They can therefore be safely discarded. In the case of zooms ($N > 1$), the 54 channels must be carefully chosen so that a set of fine channels can be stitched together without gaps. The FPGAs choose a sub-band by being given the number the first of 54 channels to output. The low-level software sets first output fine channel for coarse channel $c \in 0, 1, ...383$ to

[3] As a point of history: the DRX coarse channelisation used to be a naive, 32/27 oversampled polyphase filter bank. The naive oversampling results in a 'rotation' of the fine filter banks: i.e. in adjacent coarse channels, the DC channel of the fine filterbanks were separated by 5/32 more than the sampling rate (i.e. the separation was 1.18 MHz, rather than 1 MHz). To align the channels, the beam former needed to rotate the numbering of the the output of the fine filterbank by 5 channels for every coarse frequency channel. The coarse filter banks recently improved by changing the commutation in the DRX filterbank, so now, the DC channels of all fine filter banks align without requiring a rotator. As the rotator remains in the beam former firmware at the time of writing, the common software uses a fixed rotation of 32N channels

$subband\_channel1 = 5 * N + fineStep(c) * 54$. The $5N$ is just to skip the channels in the skirts. $bandList[c].fineStep$ is the the c'th fineStep value in the bandList value supplied by the IOC. So, to span a full coarse channel $bandList[c].fineStep$ should span the range $0, 1, ...N - 1$ while $bandList[c].freq$ remains constant.

The same 384 rotator and sub band settings are uploaded to *to all FPGAs* in the system, suggesting the beam former FPGAs (which only process a 8 MHz of the band each) themselves know which values to apply when they get the data. Presumably the data are tagged with some label (probably sky frequency), so the FPGAs can lookup the correct values when the the data for a mere 8 MHz arrives in their happy little heads.

5.2 Limitations of the existing design

The ingest pipeline is also parallelised assuming a fixed frequency mapping, which is defined by a function 9. Additional mappings could be generated if required. There is probably some flexibility in frequency mapping in the ingest pipeline, but the streams would need to know in advance what frequencies they're processing, and no communication is allowed between streams.

A bunch of frequency assignment logic is hidden from TOS in the common software layer. i.e. in Beamformer::setCenterFrequency() and DigitalReceiver::setCenterFrequency()

Previous discussions (See ASKAPSDP-1261 and ASKAPTOS-2890) on the ingest pipeline have regarded the correlator channel width as a fixed number, stored in the Facility Configuration Manage (FCM). This is inappropriate for at least two reasons: (1) The zoom configuration can be more structured than a single number (e.g. you could have a number of zoom windows, each with its own centre frequency and resolution), and (2) the zoom configuration is (in principle at least) dynamic, and should be settable on a *per-scan* basis.

6 Use cases

I'm too gutless, incompetent and bored to draw UML diagrams, so here it goes in text:

6.1 UC1: Single zoom over a contiguous band, fixed for a schedblock

1. Step 1: User specifies centre frequency f and zoom mode N applicable to all scans in the parset.
2. The User creates a schedblock with the specified parset and schedules it.
3. Step 2: The system calculates bandwidth for specified zoom, and chooses a

continuous set of coarse frequency channels with the specified bandwidth. The zoom mode for each channel is set to N. The sub-band for coarse channel indexed c is set to $b = cmodN$.

4. Step 3: The system allocates channels $c \in [0, 304)$ to beamformers $n \in [1, 8]$ with $n = c/48 + 1$ (integer arithmetic).
5. Step 4: The system calculates all remaining parameters
6. The system writes a parset for the ingest pipeline specifying a single, contiguous spectral window. The system also generates a mapping between the UDP headers and the spectral window channel number.
7. The system configures the hardware, including loading beam former and DRX FPGA firmware, setting DRX and beam former weights and sub-bands, and issuing a synchronous reset.
8. The system starts the ingest pipeline and the metadata service.
9. The system starts the first scan. The metadata service sends metadata for the first scan, including unique frequency configuration name, to the ingest pipeline. All antennas are marked as flagged.
10. The system continues through all specified scans. Hardware configuration is never changed.

6.2 UC2: Multiple zoom bands, fixed for a schedblock

Likely use case for GASKAP.

1. Step 1: User specifies centre frequency f_i (MHz) and zoom mode N_i, and bandwidth B_i (MHz), for $i \in 1, 2..S_{max}$ different spectral windows for the whole parset
2. The User creates a schedblock with the specified parset and schedules it.
3. Step 2: When the parser is executed: For each spectral window i: The system calculates a coarse frequency configuration (TODO).
4. Step 3: The system calculates a harware mapping (TODO).
5. Step 4: The system calculates all remaining parameters
6. The system writes a parset for the ingest pipeline specifying S_{max}, contiguous spectral windows. The system also generates a mapping between the UDP headers and the spectral window i, and channel numbers in that spectral window.
7. The system configures the hardware, including loading beam former and DRX FPGA firmware, setting DRX multiplexer and beam former weights and sub-bands, and issuing a synchronous reset.

8. The system starts the ingest pipeline and the metadata service.
9. The system starts the first scan.
10. The system continues through all specified scans. Hardware configuration is never changed.

6.3 UC3: Frequency configuration changed per scan

A likely use-case for FLASH follow-up of absorption lines. The most likely case is that the zoom configuration (number of windows, and resolution) would remain the same, but the zoom window central frequency would change. This use case describes an entire reconfiguration on a per-scan basis (i.e. number of windows, resolution and central frequency could change). It may be possible to define a separate, less challenging use case, where only the frequency changes, but we'll leave that for a future date.

A super-set of UC2. As per UC2 except:

1. Step 1: User specifies centre frequency f_i (MHz) and zoom mode N_i, and bandwidth B_i (MHz), for $i \in 1, 2..S_{max}$ different spectral windows for each scan in the parset.
2. When the schedblock runs, the system parses *all* scans and finds the unique set of frequency configurations.
3. Step 2: The system generates a full configuration for each frequency configuration in the parser, and tags each one with a unique name.
4. For each unique frequency configuration: The system writes a parset for the ingest pipeline specifying a single, contiguous spectral window. The system also generates a mapping between the UDP headers and the spectral window channel number. The frequency configuration is tagged with the unique name.
5. The system starts the first scan. The metadata service sends metadata for the first scan, including unique frequency configuration name, to the ingest pipeline. All antennas are marked as flagged.
6. If it is the first scan, or if the frequency configuration has changed from the previous scan, the system configures the hardware, including loading beam former and DRX FPGA firmware, setting DRX and beam former weights and sub-bands, and issuing a synchronous reset.
7. When the hardware is configured the antenna flags set according to whether they are on source or not.
8. The system continues through all specified scans. Hardware configuration is only required if the frequency configuration name changes between scans.

6.4 UC4: Frequency configuration specified by expert user

A super-set of UC3. As per UC3 except:

1. Step 1: User specifies creates 1 or more frequency standard configuration file offline, using the same inputs for UC3, and the same code that runs online for UC3.
2. System serialises full frequency configuration to disk in some human-readable format (YAML?)
3. User edits configuration files for a crazy project.
4. User creates parset that specifies a configuration file for each scan.
5. User creates and schedules schedblock.
6. When the schedblock runs, the system parses *all* scans and finds the unique set of frequency configurations.
7. Step 2: The system loads a full configuration for each frequency configuration in the parset.
8. Proceeds as per UC3, but with the configurations as loaded, rather than generated automatically.

6.5 Example UC2: Two zoom bands fixed for a schedblock

Here we consider an example frequency setup which would be typical for observing Galactic HI. The initial requirement is:

1. Window 1 (HI zoom) 8 MHz band: 1414 - 1422 MHz, 1.157 kHz resolution
2. Window 2 (continuum) 177 MHz band: 1400 - 1477 MHz, 18.5 kHz resolution

6.5.1 Step 1 - high level parameters

By writing the above specification we have essentially completed step 1.

6.5.2 Step 2 - Coarse frequency configuration

For Step 2 (coarse frequency configuration), all sky frequencies fit between 1400-1800 MHz, so we can use band 3. We note that this is a non-inverting band, so for the remaining assignments we will set the first channel as the lowest channel in frequency.

For Window 1, we require 1.157 kHz channels which requires $N = 5$ which requires 16 sub-bands to cover 1 MHz. In order to span the ~ 8 MHz Window 1 bandwidth we need 6912 fine channels, i.e. a contiguous set of 128 sub-bands of 54 channels each. Thus we assign the first 128 coarse channels to Window 1. To form a contiguous set, we need to select sub bands $b \in 0, 1, 2, ..., 15$. Thus, the first 16 channels share a common centre frequency of 1414 MHz, $N = 5$ and sub-bands $b = 0, 1, 2, ..., 15$. The

second set of 16 channels have centre frequency of 1415 MHz, $N = 5$ and sub-bands $b = 0, 1, 2, ..., 15$. The remaining 6 sets of 16 channels are assigned in the same way, each having a centre frequency incrementing by 1 MHz, with identical $N = 5$ and range of sub-bands.

Now for Window 2. We have already assigned 128 coarse channels, so (assuming an $N = 1$, 304 MHz system) we have only 176 coarse channels remaining. For the 18.5 kHz resolution we choose $N = 1$, which requires only 1 sub-band per MHz. Thus the remaining 176 channels have a centre frequency starting at 1400 MHz and increasing by 1 per channel, $b = 0$ and $N = 1$. Now all coarse channels have been assigned.

The main constraint to check at this point is that the fine filterbank resolutions have been set in groups of 8. This has been satisfied: Window 1 has $8 \times 16 = 128$ channels, and Window 2 has $8 \times 22 = 176$ channels.

6.5.3 Step 3 - Map to hardware

We have 7 beamformers numbered $n = 0, 1, ...6$, each containing 6 FPGAs that process 8 coarse channels each (i.e. 48 channels per beam former). Window 1 requires 128 channels which requires 16 FPGAs or the equivalent of 2.6 chassis. Obviously in this case we need the ability to program FPGAs inside a beam former separately. All FPGAs in beamformers $n = 0$ and $n = 1$ are assigned to Window 1 channels in sequence, as are the first 4 FPGAs of beam former $n = 2$. The remaining FPGAs/beamformers are assigned to Window 2, starting with withe the first 2 FPGAs in beam former $n = 2$.

The constrains at this step are also satisfied. All FPGAs operate on 8 sequential channels each, and use the same zoom mode. The total number of channels is $6912 + 9504 = 16416$, which equals the maximum allowed. Each beam former FPGA processes 8 contiguous channels, so the requirement that each correlator card (which process data from half an FPGA) process 216 contiguous channels, is also satisfied.

6.5.4 Step 4 - Compute remaining parameters

It's pointless going on here, as this is detail that is straightforwardly computed given the above mapping. It just requires a bit of care.

7 Implementation Recommendations

A few thoughts:

The observer knows, in advance, which frequency configurations will be required for a given schedblock. I see no reason why the system shouldn't enjoy the same luxury. This would make the ingest pipeline's job (for example) substantially easier.

The ingest pipeline currently infers the current (real-time) frequency configuration from a single number sent via the metadata service: a frequency in MHz. The frequency configuration is too complex to be parametrised by a single number. The full frequency configuration should be made available in some other, natural format (e.g. data structures obtained via an ICE service, a parset or a YAML file), and a key into that configuration (e.g. index or name) should be sent in real time, so that all other components (e.g. ingest pipeline, GUIs, hardware) can know what the real-time configuration is.

We need to have the frequency configuration generated online at some level: changing frequency for a simple, contiguous configuration, should be as easy as changing one number in a parset.

The code to generate (and check!) a full frequency configuration should be made so that it can be run in the online system, as well as offline (as long it's supplied with a suitable FCM).

The code should be able to generate, and load a human-readable frequency configuration file, that can be hand-edited, or generated by an offline script, as required by an expert user.

The current low level hardware API (setChannelsFrequency) is probably fine and should serve us well in the long term. Some of the low-level routines (e.g. Beamformer.cc) need to be modified slightly to allow for more flexibility. E.g. Beamformer.cc currently requires all FPGAs have the same zoom mode, whereas (in principle) a different zoom can be specified for each of the 6 FPGAs in a beam former.

8 Acknowledgements

The author would like to thank John Tuthill, Dave McConnell, Maxim Voronkov, Aidan Hotan, Malte Marquarding and Craig Haskins for helpful discussions and clarifications.

9 Appendix A - ingest pipeline frequency mapping algorithm

```
uint32_t VisConverter<VisDatagramADE>::mapChannel(uint32_t channelId)
{
  ASKAPDEBUGASSERT(channelId < 216);
  const uint32_t fineOffset = channelId % 9;
  const uint32_t group = channelId / 9;
  ASKAPDEBUGASSERT(group < 24);
  const uint32_t chip = group / 4;
  const uint32_t coarseChannel = group % 4;
```

```
    return fineOffset + chip * 9 + coarseChannel * 54;
}
```

10 Versions

May 2 2016 - Initial version sent for comment - Revised with comments from D McConnell, M. Voronkov, J. Allison

CONTACT US

t 1300 363 400
+61 3 9545 2176
e enquiries@csiro.au
w www.csiro.au

YOUR CSIRO

Australia is founding its future on science and innovation. Its national science agency, CSIRO, is a powerhouse of ideas, technologies and skills for building prosperity, growth, health and sustainability. It serves governments, industries, business and communities across the nation.

FOR FURTHER INFORMATION

CSIRO Astronomy and Space Science
Keith Bannister
t +61 2 9372 4295
e keith.bannister@csiro.au
w Astronomy and Space Science