Gemini FPGA Hardware Platform for the SKA Low Correlator and Beamformer

E. Kooistra* (1), G. A. Hampson(2), A. W. Gunst(1), J. D. Bunton(2), G. W. Schoonderbeek(1), A. Brown(2)

(1) ASTRON, Oude Hoogeveensedijk 4, Dwingeloo, The Netherlands
(2) CSIRO Astronomy and Space Science, Corner Vimiera and Pembroke Roads, Marsfield NSW 2122, Australia

Abstract

In this paper the hardware designed for the SKA Low (Square Kilometre Array) correlator and beamformer (CBF) is discussed. SKA-Low is a low frequency aperture array (LFAA) to be located in remote Western Australia. The array is collecting radio signals in the frequency range from 50 to 350 MHz. The large number of dual polarization antennas (131072) are distributed over a total of 512 stations with a maximum spacing of 65 km. Each station forms a single dual polarization beam with a bandwidth of 300 MHz. The bandwidth is split into 384 coarse channels of 781250 Hz in width by LFAA station beamformers. The CBF hardware receives these station beam signal and splits them into finer channels using a polyphase filterbank. Thereafter the fine channels from all stations are aligned in time, correlated and simultaneously beamformed. There are two independent beamformers, one for pulsar search (PSS), that creates 500 array beams, and another for pulsar timing (PST), that creates 16 array beams. A variety of additional functionality and modes is required in the correlator and beamformers to meet the requirements. The core of the CBF hardware is a board with a Field Programmable Gate Array (FPGA), called Gemini, that can implement all functions of the CBF. The amount of input data and processing for CBF is that large, that it requires several hundreds of these FPGA boards. The current estimate is that the SKA Low CBF will need 288 FPGAs. The same board can be used for all functions, because it is the interconnect between boards and the FPGA firmware that determine the function. This versatile hardware design could therefore potentially be used for other radio telescopes or data processing systems as well.

1. CBF Architecture

The SKA Low Correlator and Beamformer (CBF) processes station beams under the control of the telescope manager to produce visibilities for imaging and array beams for pulsar search and pulsar timing [1]. As such the CBF is largely a compute engine for SKA Low. The stations and the CBF use narrow band processing algorithms. This has multiple advantages like: phase shifts can be used to compensate for geometrical delays between stations, complex gains can be used for beamforming, RFI can be excised in a smaller number of channels and the frequency bands can be processed independently in parallel. The CBF application cannot fit into a single FPGA and also not in a few FPGAs on a single board, because of the input load and number of links from the stations, and because of the amount of processing. Therefore, the input from the stations needs to be received by multiple FPGAs and then redistributed to enable each FPGA to do an independent part of the processing for all inputs. Data can be separated in independent parts either in frequency, in space or in time. In the CBF the separation in time is not used. The separation in frequency can be done based on the coarse channels or the fine channels that are output by a filterbank. The separation in space is possible after a beamformer operation that creates multiple beam directions, the data for each beam direction can be processed further independently. The correlator and beamformers for PST and PSS operate on fine channels, however all three use different fine channel resolution. Therefore, there are three separate fine channel polyphase filterbanks in the CBF. The filterbanks operate per station input and are therefore part of the station based processing. The correlator and beamformers combine all station inputs and are therefore part of the array based processing in the CBF.

Figure 1 shows the signal processing flow concept for CBF. The cross represents the point-to-point fiber links between FPGAs. These are used for both data distribution in between FPGAs and for collecting data together for the CBF output. The memory allows for reordering the data and for replacing lost or flagged data, to prepare the data for the processing or for output. After a cross connect each FPGA first aligns all inputs and then processes these inputs in the array based processing.

![Figure 1. Signal processing flow concept for CBF](image-url)
to the array based processing that implements the correlator and beamformers. Similarly, all FPGAs in the array based processing also contain an identical firmware image.

Figure 2. Possible CBF partitioning on hardware using 'local' (1 and 3) and 'remote' (2) cross connects

The cross connect marked by 1 in Figure 2 could first do a redistribution of coarse channels from groups of stations. The cross connect marked by 2 then further distributes the fine channels in a second hop to the array based processing. The cross connect marked by 3 could be used to collect data results over groups of channels. Note that the ‘local’ cross connects 1 and 3 are used in both directions, because they exchange data between FPGAs within one partition, whereas the ‘remote’ cross connect 2 between FPGAs in different partitions of the system is used in only one direction. The dotted arrows in Figure 2 show that the other direction can be used to collect data for output. With the dotted arrows Figure 2 becomes more symmetrical and this leads to another possible partitioning on hardware that is shown in Figure 3.

Figure 3. Current CBF partitioning on hardware using three cross connects

In the partitioning of Figure 3 all FPGA nodes contain the same firmware image and all can do the same functions, but each for a fraction of the total processing. The estimated processing load for the CBF requires the use of 288 FPGAs. This implies that with 512 stations each FPGA will receive station beams from 2 stations, so 256 FPGAs receive input and 32 do not. In the first cross connect marked by 1 in Figure 3 the coarse channels from groups of 16 stations so 8 FPGAs are redistributed. After the fine channel filterbank the cross connects marked by 2 and 3 are used to further redistribute the fine channels, such that eventually each FPGA receives fine channel data from all 512 stations, but for a 1/288 fraction of the 300 MHz band. After the processing the cross connects are used again to collect the results for output. Hence the same cross connect hardware can be used multiple times to improve link utilization. In this scheme the cross connects form a processing cube of $8 \times 6 \times 6 = 288$ FPGAs whereby cross connect 1 is 8x8 and occurs 36 times, cross connect 2 is 6x6 and occurs 48 times and cross connect 3 also is 6x6 and occurs 48 times. In total the CBF architecture then has $36 + 48 + 48 = 132$ cross connects and each FPGA uses $8 + 6 + 6 = 20$ links. Together these cross connects implement a full mesh interconnect between all 288 FPGAs in three hops.

2. Gemini Hardware

Existing boards like Redback-3 [2] made by CSIRO and UniBoard2[3] made by ASTRON historically always had four or more FPGAs per board. In the past this made sense, as the FPGAs could share clock and power resources and were heavily interconnected using parallel communications for data sharing. However, modern FPGAs have about 100 transceivers per FPGA, that are capable of running >10Gbps and even >25Gbps, and each FPGA is required to connect to 10’s of other FPGAs that cannot all fit onto one board. Furthermore, using only one (large) FPGA per board reduces the cost, power, heat and risk of failure per board. Therefore, the Gemini board has only one FPGA and the interconnect between FPGAs is dependent on optical cross connects. These cross connects are relatively cheap, so the structure of a system with multiple Gemini LRUs can easily be configured dependent on the application, by appropriately wiring the fibers in optical backplanes. In this way the Gemini LRU is generic and easily combined into larger architectures using application specific optical backplanes.

Figure 4. Layout of the Gemini LRU

Figure 4 shows the layout of the Gemini Line Replaceable Unit (LRU). The Gemini LRU board will have one Xilinx Ultrascale+ FPGA. The left side of the FPGA provides four 100GbE QSFP interfaces using 16 28G transceivers. The right side of the FPGA provides 36 transceivers at 25G via 3 Mid Board Optics (MBO) modules. One more transceiver is used for the 1/10GbE control interface. Hence in total 53 28G transceivers are used. There is one DDR4 memory module located at the top side of the FPGA. For other applications than the CBF, the Gemini LRU also supports the JESD204B standard for multi gigabit serial transceiver interface to ADCs via the MBO links.

The use of one FPGA per board is not new. The CASPER community [4] with their ROACH boards and latest SKARAB board have a long history of using only one FPGA per board and relying on multi gigabit Ethernet.
switches for the streaming data I/O, also between FPGAs. The Gemini board also uses multi gigabit Ethernet I/O, but for the interconnect between FPGAs it relies on point-to-point fiber interconnect. The point-to-point links will use Ethernet, but not switched externally, only inside the FPGA firmware. The advantage of using fixed cross connects is that it avoids the power consumption, space and cost of using switches. The disadvantage can be that the connections are fixed, but for the CBF application that is fine, because the CBF is an I/O intense application. The SKARAB board is typically packaged in a single box with dedicated AC-DC power supply. The Gemini board could also be packaged in a single box, but it is intended to be used with multiple Gemini LRUs in a subrack. The Gemini Subrack will contain 12 Gemini LRUs. The estimated power consumption of the Gemini LRU is approximately 220 W. The Gemini LRU operates on -48V DC and uses liquid cooling. The DC power and the liquid cooling are supplied via a backplane.

The CBF will receive the station beams and transmit the visibilities and array beams via the 40/100GbE interface. The load of one station beam is 11.4 Gbps. Each FPGA receives data from up to two stations so the input load per FPGA is then 22.8 Gbps and with 512 stations the total input load for the CBF is 5.8 Tbps. The CBF output load is somewhat less than the input load and can also be output via the 100GbE interface. With four 100GbE the Gemini LRU can input and output maximum 400 Gbps. The internal cross connections between the FPGAs in the CBF will be done via the MBO 25G interfaces. With three MBO the Gemini LRU can redistribute a data load of maximum 900 Gbps. In total the CBF application will have 288 Gemini LRU located in 24 Gemini Subracks and the cross connects are organised in 24 optical backplane structures. The redistribution of data is preferably done in one hop to avoid that data is only moved in an FPGA without any processing. Whether this is possible depends on the number of parallel processing FPGA nodes and the number of 25G ports per FPGA. For the CBF three hops will be used.

A single DDR4 memory module that can fit up to 128 GByte is provided on the Gemini LRU. This DDR4 memory allows the full bandwidth, CBF input data to be buffered. Inside the FPGA chip package, the Ultrascale+ FPGA will have several GByte of High Bandwidth Memory (HBM) that can cover for the high data rate buffering. This allows it to implement multiple buffering operations but with limited memory depth for each. An alternative to HBM memory is external Hybrid Memory Cube (HMC). As HBM is currently in development HMC was used on the Gemini Proof of Concept (POC) board that was produced in January 2017 and shown in Figure 5. The HMC interface to the FPGA use bundles of 16 transceivers, so it provides a high data rate memory. The Gemini LRU will use HBM instead of HMC. The Xilinx Ultrascale+ XCVU9P FPGA type on Gemini POC can also fit on the new Gemini LRU.

The Gemini LRU does have clock and pulse per second (PPS) inputs, but the intention is to run the boards on local clocks. The synchronization between boards is then obtained using the IEEE1588 precision timing protocol (PTP) via the 1/10GbE control port. The Gemini LRU supports tuning of the local clock using White Rabbit functionality to achieve ns accuracy. Accurate timing between FPGA nodes is necessary to avoid input buffer overflow when data from different inputs needs to be aligned. The actual timestamping of the station beam data has already been done in the LFAA stations. Throughout all subsequent data transport, buffering and processing, this time information is passed along with the data in packet headers or implicitly by the data order.

A custom function in firmware on the FPGA provides a monitoring and control (M&C) interface via 1/10GbE. The M&C interface provides access to the memory mapped registers in the FPGA design via a memory mapped (MM) bus. The M&C task can be divided into initialization, occasional or slow control and fast regular control. In astronomical applications the need for fast regular control typically relates to the control measures needed to compensate for the geometrical delay variation between stations due to the rotation of the Earth. For the CBF the calibration of the array beams will also require fast control, because SKA Low is a new instrument that aims to achieve beam calibration goals far above existing instruments. The calibration data rates are exacerbated by the large number of beams formed. Once the geometrical delay is compensated and the beams are calibrated, the control and monitoring that remains can be qualified as either initialization or slow control. The initialization control concerns for example loading an FPGA image and tuning for example loading an FPGA image and setting Ethernet MAC addresses. The slow control concerns for example the monitoring of temperatures and status of communication links. The amount of control that is needed relates linearly to the amount of data that is processed, because more data typically implies more weights. In existing astronomical instruments, it appears that the M&C data rates that are needed are at least four orders of magnitude less than the input data rate.
Therefore, the 1/10GbE interface on the Gemini LRU provides sufficient capacity for M&C, also during bursts.

4. Firmware Specification

Figure 6 shows a general block diagram of a digital signal processing (DSP) application on an FPGA. Within the FPGA design there are two standard interfaces, one is the memory mapped (MM) interface for M&C and the other are the streaming (ST) interfaces that transport the data from input via data move functions and data process functions to the output. The data process functions change the data. The data move functions reorder or select the data. The input/output functions provide the physical access to external memory and the transceivers, and also include data packetizing. In addition, the FPGA design will also contain test functions to be able to isolate and diagnose faults when they occur during operations. All I/O, data move, data process and test functions can have MM registers that can be accessed via M&C.

![Figure 6. FPGA firmware design using standard interfaces for streaming (ST) data and memory mapped (MM) monitoring and control](image)

The data I/O, data move and data processing for an FPGA design can be specified using the array notation. The array notation follows the array indices as used in the programming languages C and Python, so the last index in the array varies the fastest. In a document the array notation can also use subscripts instead of indices to more clearly identify parallel data streams. For example, the signal $S_{node}[t][ch]$ describes a total input signal for a range of nodes in parallel. A node can be an entire FPGA. All nodes receive the same type of signal, but from another source. Per node the input signal consists of channel samples in series that are grouped in blocks of channels [ch] per time instant. The index [t] counts the time instants. A cross connect transposes the order of node inputs and channels in space, and can be described by a swap between the subscript [node] and index [ch]. If the number of channels is larger than the number of nodes, then the index [ch] can be separated in [ch_n][chn], whereby index [ch_n] has the same range as subscript [node] and together [ch_n][chn] cover the same range as [ch]. After the cross connect the transposed input to the nodes can now be described by $S_{ch,n, node}[t][chn]$. In this the first subscript [ch_n] reflects the number of nodes. The second subscript [node] now reflects that each node receives a fraction of the channels from all nodes, as indicated by index [chn]. A corner turn transposes the samples in time, and can be described by first separating the time index into two indices [t][ti]. In this index [ti]

defines the range of the corner turn time interval and index [t] is the continuous time that now counts these intervals. The corner turn between channels and time intervals can now be described by swapping the [ti] index and the [ch] index to get $S_{node}[t][ch][ti]$. By remapping indices, separating indices and by swapping parallel subscripts and/or serial indices, it is possible to also clearly specify other data move functions like selecting, multiplexing and packetizing data. Data processing functions can also be described by changing indices. For example, if the input of a critically sampled filterbank is described by $a[t][tc]$, so as a series of time samples in blocks of [tc], then the output can be described by $A[t][ch]$, whereby each block of [tc] time samples has been transformed into [ch] channels. For DSP functions it is important that the algorithm and quantization are clearly defined, but when it comes to the implementation it is also crucial that the order of the data input and the data output are clearly defined, and that is possible using the array notation.

5. Conclusion

The SKA Low Correlator and Beamformer needs to perform several functions. In order to implement those functions and the cross connections in between them, a versatile FPGA based hardware platform has been designed. A fiber based backplane is used to tailor the hardware platform to the application. Together with a technology independent firmware framework, this forms an excellent basis to build the SKA Low Correlator and Beamformer and has the potential to be used also for other radio astronomical applications.

6. References


