Backend simulations and prototyping report

Designing a Next-Generation Radio Facility for Multi-Messenger Astronomy

Deliverable No:	**D6.1**
Deliverable/Document Title:	**Backend simulations and prototyping report**
Version:	**1.0**
Release date:	**2024-06-28**
Dissemination level:	**Public**
Status:	**Submitted**
Author:	**Weiwei Chen, Ewan Barr, Niclas Esser, Yunpeng Men**

Funded by the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement ***No 101094354***. The content of this material reflects the opinions of its authors and does not in any way represent the opinions of the European Union

Document history

Version	Date	Content and changes	Edited by
1.0	2024-06-28	Initial Release	Ewan Barr

Peer review history

Partner	Reviewer	Date	Version
FORTH-ICS	Stefanos Papadakis	25.06.24	1.0

Release history

Signed off by	Release Date	Version	Signature
John Antoniadis	13.08.24	1.0	[illegible]

Disclaimer

Applicable Documents

- D3.2 System Requirements Specification
- D5.3 Front-end subsystem design

Table of Contents

Abbreviations and Acronyms

Acronym	Description
ADC	Analogue-to-Digital Converter
CMAC	UltraScale Integrated 100G Ethernet Subsystem
DAC	Digital-to-Analogue Converter
DDR	Double Data Rate
FPGA	Field-Programmable Gate Array
GbE	Gigabit Ethernet
GPU	Graphics Processing Unit
Gsps	Gigasamples per second
HLS	High-level synthesis
IP	Intellectual Property
OFIR	Optimal Finite Impulse Response
OS-PFB	Oversampling Poly-phase Filterbank
PPS	Pulse per second
QSFP	Quad Small Form-factor Pluggable
RFDC	Radio Frequency Data Converter
RFSoC	Radio Frequency System on a Chip
RTL	Register Transfer Level
Rx	Receiver
TRL	Technology Readiness Level
UDP	User Datagram Protocol
VLBI	Very Long Baseline Interferometry

1. Overview

This document contains the results of simulations and prototyping for the ARGOS digital signal processing backend as conducted by Work Package 6 (WP6) of the ARGOS-CDS project. WP6 is responsible for the design of the digitisation and channelisation system for the ARGOS antennas and with the data transport, correlation, and beamforming that is required to enable the telescope to perform as an interferometer.

In this document, we present simulation results for the following:

- Data transport for digitised/channelised voltages streams between the ARGOS antennas and the downstream processing hardware.
- Beamforming optimisations based on different array geometries.

We further present the results of the following prototyping activities:

- Development of a demonstrator digitiser-channeliser on an RFSoC evaluation board.
- Development of a demonstrator beamformer-correlator on tensor-core capable GPUs.

Intended audience

This document is intended for use by system designers and engineers working on the realisation of the ARGOS telescope.

Assumptions

At the time of writing, the system requirements for the final ARGOS telescope are in flux, and no technology readiness levels (TRLs) or freeze-in dates have been defined for the ARGOS hardware. Hence, we here make several assumptions to aid in performing simulations:

- We assume a full-size ARGOS array will consist of 1120, dual-polarisation elements with 2 GHz of bandwidth each.
- For simulations, we principally assume that we are solving the problem with hardware that can be procured at the time of writing. Where relevant, we discuss the implications of hypothetical future upgrades to such hardware.

2. Data Transport Network Simulations

Problem Statement

Assuming that ARGOS consists of 1120 antennas with dual-polarisation receivers digitising at a sampling rate of 4 Gsps, the input data rate from the telescopes is 72 Tb/s. Due to the requirement for ARGOS to be compatible with existing VLBI infrastructures, it is necessary to oversample in the ARGOS channelisers (details discussed later). With an oversampling factor of 32/27, the telescope's data rate increases to 85 Tb/s. This rate is 42 times higher than the current state-of-the-art MeerKAT radio telescope in South Africa, posing a significant challenge in designing a data transport network capable of distributing the telescope data over the processing hardware for correlation, beamforming, and other downstream processing. A network architecture for the data transport network must be designed and shown via simulation to be viable for the ARGOS array.

Assumptions

It is assumed that ARGOS will use digital receivers at the antenna that digitise, channelise, and packetise the received voltages before sending them into an Ethernet network. Justification of the use of on-antenna digitiser/channelisers is beyond the scope of this document. However, such a design that combines digitisation and channelisation into one hardware component has a significant impact on the design of a data transport network for ARGOS. In this case, the input to the network is now channelised rather than unchannelised voltages streams (which is the case for MeerKAT for example). When using unchannelised inputs, the data must first be routed to channelisers, and the resulting channelised data then routed to the downstream processing endpoints. In the case of ARGOS, only the routing between the channelisers and the downstream processing systems need to be considered, simplifying the network design and reducing the required network throughput.
We further assume that each ARGOS antenna has a single 100-GbE fibre connection that carries all digitised voltages. This is sufficient to accommodate the 76 Gb/s of data produced at each antenna.

Topology selection

We initially considered several network topologies when investigating the data transport network for ARGOS. However, the standard F-X correlator architecture imposes constraints on network design, as it must be possible to route the data from each antenna to all processing endpoints. This leaves only variants of fully connected networks for consideration. Here, we consider a variant of the folded Clos network (a kind of multistage circuit-switching network, name after Charles Clos), where we allow the antenna data to be directly routed into the spine switches. This is reasonable as there is no requirement to send data between antennas, only between antennas and processing endpoints and between processing endpoints (but at much lower volume than the volume from the antennas).
The aforementioned processing endpoints are likely to be GPU or FPGA servers that perform correlation and beamforming on discrete portions of the frequency band for all antennas. In the discussion below, they are interchangeably referred to as nodes, processing nodes or processing endpoints.

Network diagram

In Figure 2.1 we present a schematic of the nominal modified folded Clos network considered for ARGOS

Table 2.1: The items and their properties inside the network topology diagram

Symbol	Description	Interface (individual)	Number
ant0 … ant1119	Antenna	100 GbE	1120
spine0 … spine4	Spine switch	800 GbE, 64 port	5
leaf0 … leaf4	Leaf switch	800 GbE, 64 port	5
node cluster 0 … node cluster 4	Processing node cluster	200 GbE, 104 port	5

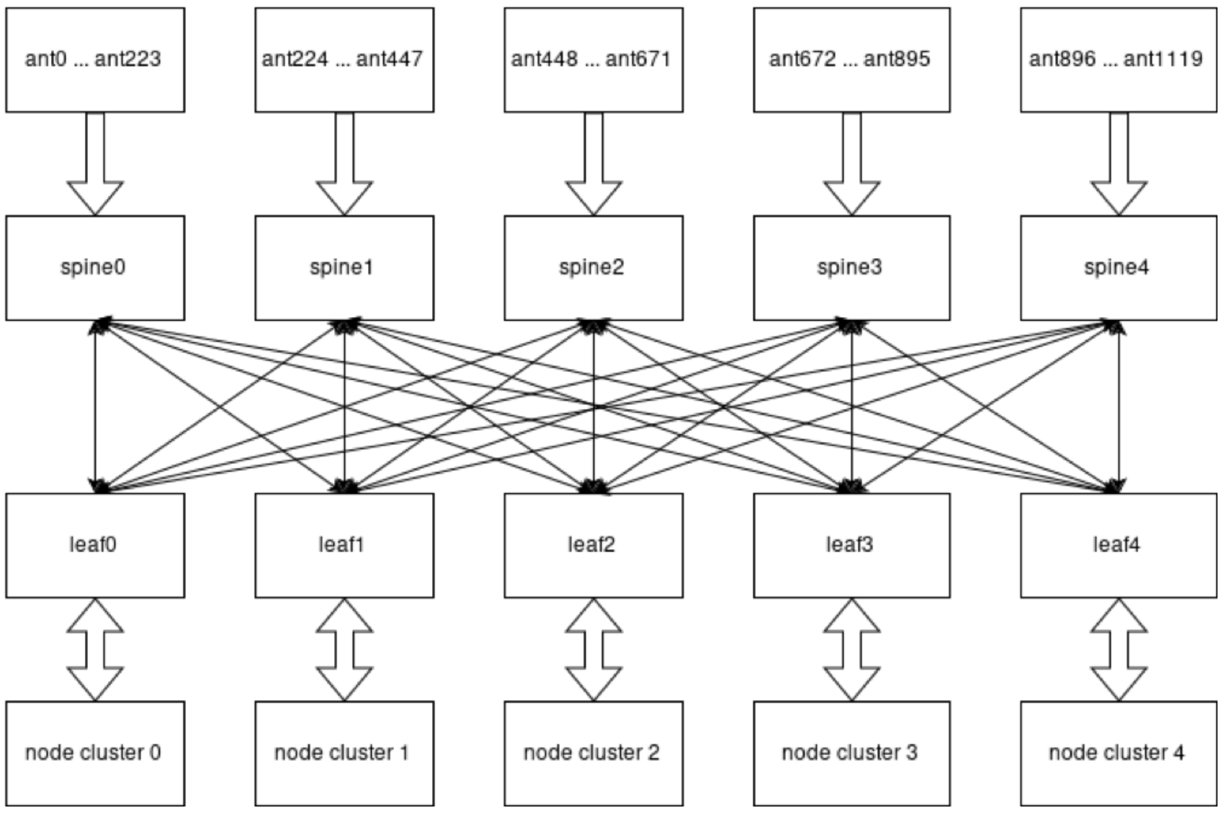

Figure 2.1: Example of a modified folded Clos network topology for ARGOS data transport.

In this example (Figure 2.1), we use the same switch variant for both spines and leaves. The current state-of-the-art in commercial Ethernet switching is 800 GbE. Here, we assume the use of 800-GbE switches with 64 physical ports per switch (an example of a switch with such properties would be the Nvidia SN5600: 920-9N42F-00RI-7C0). Each physical port can be split into 8 logical 100-GbE ports or 4 logical 200-GbE ports while the total number of logical ports can not exceed 256 (this has been a restriction in the majority of top-of-the-line switches since 40 GbE became available and it is reasonable to assume it will be the same in future). For reasons of redundancy, fault tolerance and future extensibility, we leave 4 physical ports in each spine switch and 6 physical ports in the leaf switch unused. Each of the 1120 antennas, transmits data over a 100-GbE connection. To ingress and egress data from all antennas, a minimum of 5 spine switches are needed. Each spine switch uses 28 physical ports for uplinks, with each port connecting to 8 different antennas using 8-way breakout fibres. Each spine switch has 32 ports available for downlinks. They are distributed to all leaf switches in a round robin.
Each leaf switch uses the remaining 26 physical ports for connecting to a node cluster. According to the data rate calculation (see next section) that based on prototyping activities on the ARGOS beamformer, we assume an upper cap on the bandwidth to each processing node in the node cluster of approximately 166 Gb/s. As such, each of the 26 remaining physical 800-GbE ports is broken-out to 4x 200-GbE logical ports with 4-way breakout fibres. In this mode, each processing cluster can host up to 104 processing nodes.

The number of processing nodes required to satisfy the ARGOS requirements is currently unknown. In the case that ARGOS must produce a very large number of beams (or must share its beamformer hardware with other compute intensive downstream processing) the network must be expanded to accommodate a larger number of nodes. Below we describe the scaling properties of this network topology.

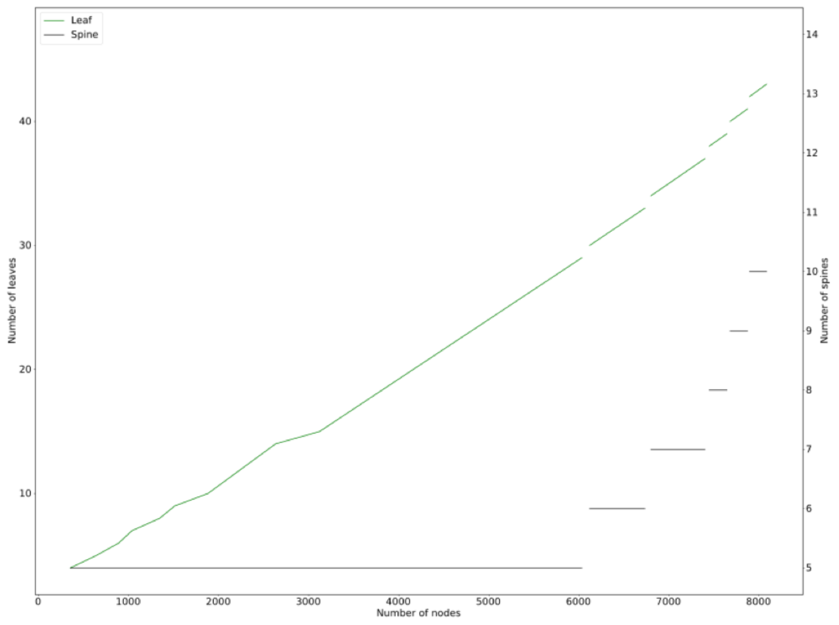

Figure 2.2: Numbers of spine and leaf switches as a function of number of nodes.

Scalability and flexibility

In order to reduce project risk and to be flexible to changes in the processing node design and science requirements for ARGOS, it is necessary to provide a flexible and scalable network architecture. A Clos network design such as the one presented in Figure 2.1 can be scaled by modifying the number of leaves and/or spines depending on the specific constraints that must be met. Examples of such changes are listed in Table 2.2. For the nominal example shown in Figure 2.1, there are 5 spine switches and 5 leaf switches, which support 520 processing nodes (see the highlighted green column in Table 2.2). With the number of spine switches fixed, there would be around 200 additional new nodes available after every new leaf switch is added (shown in the third row in Table 2.2). When using 5 spine switches it is possible to support a total of 6784 processing nodes with 32 leaf switches. For more processing nodes, further spine switches are needed.

Table 2.2: Number of processing nodes supported by different spine/leaf combinations. The data rates here show how the antenna data is distributed as it flows from the spines to the processing nodes. The first column is highlighted in red as it results in a data rate per node that would exceed the throughout a 200-GbE network connection. The 5 spine, 5 leaf combination that we consider to be the nominal configuration for ARGOS is highlighted in green.

Spine switch	5	5	5	5	5	5	5	5	5	5
Leaf switch	4	5	6	7	8	9	10	11	12	13
Processing Nodes	288	520	752	984	1216	1448	1680	1912	2144	2376
Data rate from spine to leaf switch (Gb/s)	21280	17024	14187	12160	10640	9458	8412	7738	7093	6458
Data rate per node (Gb/s)	296	164	113	87	70	59	52	45	40	36

To show the trend of the changes at larger scales, the numbers of the processing nodes, spine switches and leaf switches are parameterized (starting with 5 spine switches and 4 leaf switches) and shown in Figure 2.2. It is clear from Figure 2.2 that the number of the leaf switches and processing nodes does not increase evenly when a new spine switch is added. This is due to the new available ports from adding a new spine switch becoming less effective for supporting uplinks from new leaf switches.

It should be noted here that we have maintained the requirement of a 200-GbE connection to each node in Figure 2.2. In theory the bandwidth to the nodes could be decreased to 100 GbE after we reach 7 leaves, although it should be noted that this would require 8-way breakouts and would be capped by the logical port count limit on the switch.

Load balancing

The architecture simulated here contains implicit assumptions about how data will be routed in the network. Here we make these assumptions explicit:

- It is assumed that all data in the network (with the exception of some required protocol traffic) will be UDP multicast.
- Processing nodes will subscribe to data streams as a function of frequency. This allows the data out of each spine to be evenly distributed between all leaves. In turn this imposes a constraint that the same frequency channels should not be subscribed to on different leaves, as this will result in packet replication in the spines.
- There will be sufficient entropy in the combination of source IP, source port, destination IP and destination port to allow for even load balancing on link-aggregated connections between spines and leaves. This is relatively easy to achieve as there are significantly more nodes on each leaf compared to link-aggregated ports.
- Should multiple nodes require subscription to the same data this would be done by nodes on the same leaf, thus delegating packet replication to the corresponding leaf and leaving the leaf-to-spine traffic unaffected.
- It is the responsibility of the backend control software to specify the multicast subscription mapping over all nodes so as to not overload individual links in the network.

Comparison to other solutions

Similar Clos network architectures for radio telescope data are currently in place at the MeerKAT and Effelsberg radio telescopes. While these operate at lower data rates than ARGOS, their data rates as a function of the technology they use (e.g. 2 Tb/s at MeerKAT using 40 GbE networking) are comparable to ARGOS. We note that there are specific lessons about such networks that have been learned from MeerKAT, in particular that switches chosen must have reasonable transmit buffering to safely handle packet collisions. Such collisions may also be avoided by careful tuning and synchronisation of the packet transmission from the digital Rx system but this may require costly FPGA development. Purchasing switches with deep transmit buffers eliminates the problem of packet collisions without requiring special behaviours from the digital Rx.

The mid and low frequency arrays of the Square Kilometre Array use a different networking solution that shares a similar topology but uses different technology. There, instead of using standard Ethernet switches, they use P4 switches. These are hardware programmable switches that allow for extremely fine-grained network control. The intended use-case for such switches is primarily hyper-scale data centres with highly dynamic network requirements. Such switches are more expensive per unit than standard Ethernet switches and incur a higher operating cost due to the need to maintain the P4 software used to define the network behaviour. However, they can also achieve higher power efficiencies than standard Ethernet switches due to their ability to restrict unwanted protocol traffic in the network. More information about the power savings possible with P4 switches is required to understand if P4 may be of benefit to ARGOS.

Conclusion

Simulation of the scaling behaviour of a modified Clos network shows that the architecture is well suited to handling the data rates expected for the ARGOS telescope. The architecture is flexible enough to be capable of handling situations requiring the ARGOS backend to be composed of a very large number of relatively low bandwidth processing endpoints. Similar architectures have been successfully deployed at the MeerKAT and the Effelsberg radio telescopes.

3. Beamforming Strategies

Problem statement

The provisioning of wide-field, time-domain survey capability is one of the most important science requirements for ARGOS. This necessitates the real-time formation of a large number of tied-array beams using as many antennas is possible. There are multiple strategies for forming such beams and the relative merits of these are strongly dependent on the array geometry. Here we simulate how the cost of beamforming with different strategies varies as a function of array geometry.

Array geometry

The geometric configuration of the antennas is the main determinant of the optimal beamforming strategy. Here we present several array layouts and discuss the constraints they impose on the beamforming strategy used for the telescope. The computing cost and baseline samples of different array types are listed in Table 3.1.

Note that we do not consider visibility beamforming (i.e. beamforming by summing visibilities as opposed to voltages) to be a viable strategy for an array of the expected size of ARGOS. For such a strategy to be viable (i.e. superior to other approaches), the number of integrated visibilities in the correlator must be > (Na + 1) / 2, where Na is the number of antennas. For an ARGOS size array this implies integration times of > 286 microseconds which is insufficient to meet the time resolution requirements for time-domain science (see REQ-ARGOS-BE-13 and REQ-ARGOS-BE-32). Detailed investigation of the scientific impact of the choice of array geometry is out of scope for this document.

Regular arrays: Arrays of antennas placed in a regular grid are known as regular arrays (see Figure 3.1). Examples of such arrays include the Murchison Wide-field Array (MWA), the Hydrogen Intensity and Real-Time Analysis eXperiment (HIRAX) and the Canadian Hydrogen Intensity Mapping Experiment (CHIME). One of the most enticing features of a regular array is that it enables efficient Fast Fourier transform (FFT)-based beamforming. In this method we may directly FFT the gridded voltages at the positions of the receiving elements to Nyquist sample the spatial power distribution on the sky. Due to the wide applicability of FFTs, high performance FFT implementations have been developed for a large selection of hardware platforms. Using hardware acceleration, this technique can generate a large number of beams effectively at relatively low computing cost. Furthermore, the regular gridding creates a large number of redundant baselines, which significantly reduce the number of operations needed to image the field compared to an irregular array. For example, the unique baseline fraction of a regular array can be as low as 0.3%. The most obvious downside of such array is that it produces sparse and clustered baseline samples, thus the UV plane (the spatial-frequency sampling of the sky) is potentially sampled neither sufficiently nor evenly for the needs of a given science case. This particularly affects the dynamic range of the data which is crucial in imaging and studying small scale structure.

Direct FFT beamforming has several drawbacks:

- Beams cannot be arbitrarily placed on the sky.
- The number of beams produced is dependent on the number of antennas.
- The position of beams is depending on observing frequency.

The latter drawback can be minimized by the post-beamforming combination of different beams at different frequencies to create beams that have a similar pointing centre at all frequencies. The drawback here is that the sensitivity as a function of frequency is no longer mathematically smooth.

The big-O complexity of FFT beamforming by this method is Na $\log_2$ Na, where Na is the number of antennas.

Irregular array: These arrays do not place antennas on a predefined grid but at positions determined by some optimisation target function (usually the completeness of the UV sampling required for some scientific experiment). Example are MeerKAT, the Square Kilometre Array, the Next Generation Very Large Array (ngVLA), the Deep Synoptic Array (DSA-2000) and the Atacama Large Millimeter/submillimeter Array (ALMA). Such arrays are often composed of multiple antenna distributions. For example, the MeerKAT telescope is composed of a Gaussian distributed core and a set of spiral arms (shown in Figure 3.2). These configurations have the benefit that they can support the UV-sampling needs of multiple science cases (e.g. a dense core for time-domain survey work and long spiral baselines for high-dynamic-range imaging). Unlike with a regular array there is no trivial FFT-based solution for beamforming an irregular array. In this case, the beamformer has to sum the voltages of each antenna with the appropriate complex weighting for each beam position in a method referred to as "brute force" beamforming. Additionally, there are usually no redundant baselines in such an array, which increases the computational cost of imaging.

Unlike FFT beamforming, brute-force beamforming is agnostic about the number or placement of the antennas in the array and by default aligns beam pointing positions at all frequencies. However, the big-O complexity now scales with the number of beams, Nb, and number of antennas, Na, as Na × Nb. This is more efficient than FFT beamforming when creating Nb << Na. However, when creating large numbers of beams where Nb >~ Na, FFT beamforming is ~ Nb / ($\log_2$ Na) times more efficient.

Hybrid array: An array that contains both regular and irregular placements of elements can provide some of the advantages of both array types. An example of such array is shown in Figure 3.3. It contains 1120 dishes, of which 75% are used to form a regular core grid, the rest are Gaussian distributed. It consists of 626640 baselines, of which 275044 are unique, making the unique baseline fraction as 0.4389.

The corresponding UV sample distribution is shown in Figure 3.4. As can be seen, the UV samples are distributed rather evenly thanks to the mixture of regular and Gaussian placing of the antennas.
A possible use case of such an array is to use only the grid core for wide-field time-domain survey experiments, using FFT beamforming. The full array would then be used for forming images or forming a small number of high spatial resolution beams at specific positions (e.g. for pulsar timing), using brute-force beamforming. Additionally, the core can be further divided in multiple sub-section with different phase centres to obtain even larger field of view.

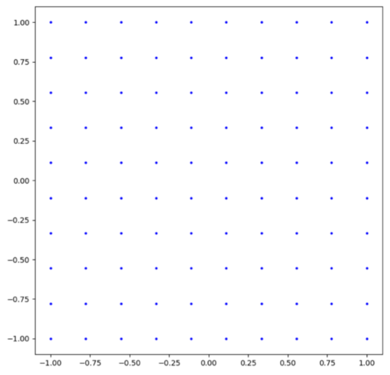

Figure 3.1: Regular array

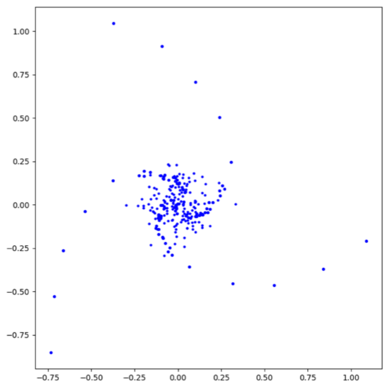

Figure 3.2: Gaussian-spiral array

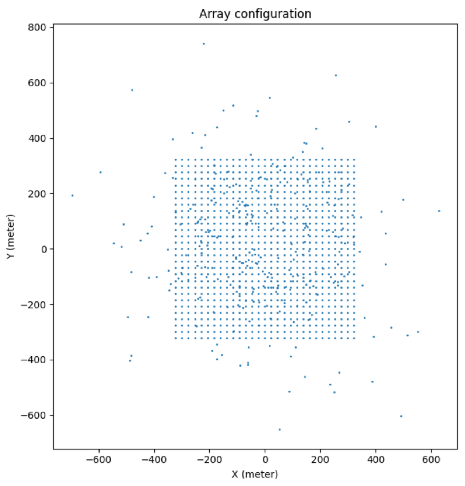

Figure 3.3: Hybrid configuration

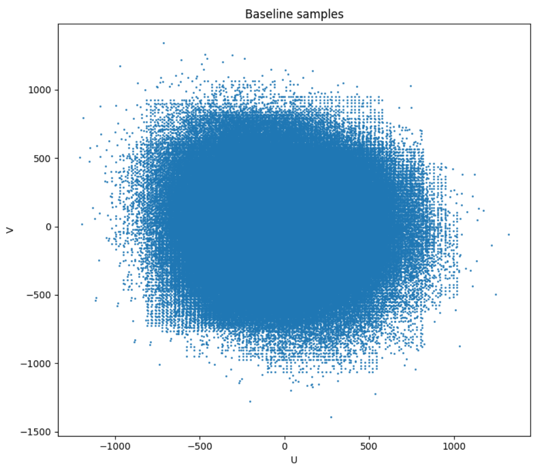

Figure 3.4: UV samples for Hybrid configuration

Table 3.1: A comparison of computing cost and baseline samples of different array type.

Array type	**Computing cost**	**UV samples**	**Application**
Regular grid	low	sparse, uneven	wide-field transient search
Gaussian-spiral	high	smooth, even	high dynamic range observations
Hybrid	low	smooth, even	multiple

Data rates

The ARGOS beamformer and correlator system must be capable of handling an incoming data rate of:

$$Rd = BW \times Na \times Npol \times Ndim \times Ro \times Nbit$$

Where Rd is total data rate for a full array in bits/second, BW is the bandwidth of the digitizer in Hz, Na is the number of antennas in the ARGOS array, Npol is the number of the polarization used, Ndim is the number of dimensions in each sample (1 for real-valued samples and 2 for complex samples), Ro is the oversampling ratio used by the channelisers and Nbit is the number of bits used to represent each numerical value. For the current ARGOS design:

- BW = 2 GHz
- Na = 1120
- Npol = 2
- Ndim = 2,
- Ro = 32/27 ~ 1.1851
- Nbit = 8

With this set of parameters, the total data rate Rd is 84.954×10^{12} bit/s or approximately 85 Tb/s. Based on the performance of the beamformer prototype presented later in this document, we may make an estimate of the number of processing nodes required to handle the full ARGOS data rate.

The number of processing nodes needed for a given number of channels can be calculated as follows:

$$N_{node} = \mathrm{ceil}(N_{ch} / \mathrm{floor}(N_{ch} \times R_{node} / R_d))$$

Where Nnode is the number of the processing nodes, Nch is the number of the channels produced by the digital Rx channeliser system and Rnode is the maximal data rate a processing node can ingest. Prototyping results suggest that beamformer nodes running current generation hardware should be capable of ingesting approximately 166 Gb/s of data. Assuming that the channelisers produce 1024 frequency channels over the full 2-GHz band, this would allow each processing node to ingest two frequency channels for all antennas. Using the equation above we find that 512 processing nodes would be needed to satisfy the full data rate of the channelisers.

Conclusions

Two main beamforming approaches are available to ARGOS: FFT-based and brute-force. The applicability of the FFT-based method is dependent on the antenna placement being regular while the brute-force method is universally applicable to all array geometries. The FFT-based method can be significantly more computationally efficient than the brute-force method allowing it to produce large numbers of beams with relative ease, albeit with the downsides that the number and positioning of beams cannot be arbitrary and the beams have different pointing positions as a function of frequency. Despite these constraints, this method can be useful for performing wide-field survey work. For anything but a fully regular array, the brute-force method is the only viable beamforming approach that can use all antennas.

Assuming current hardware and a GPU beamformer, beamforming an 1120-antenna ARGOS array would require 512 processing nodes. Here a processing node does not imply a physical server, but instead a single network endpoint with processing capability (as such several processing nodes could be hosted in a physical server). The amount of compute resources required by each processing node will depend on the beamforming approach and thus the array geometry.

4. Prototyping overview

As part of Task 6.2, the ARGOS-CDS grant agreement states that WP6 will perform prototyping activities on real hardware with the goal of determining robust in-project estimates of the performance of the various COTS components that may comprise the backend. In the following section we present the results of prototyping beamforming and correlation on GPUs and of building a digital receiver system on RFSoC.

Objectives

WP 6 has identified the following critical prototyping activities necessary to evaluate the performance and suitability of COTS components similar to those that may comprise the ARGOS backend:

- Prototyping of digital receiver hardware
- Prototyping of Tensor Core-enabled beamforming

Prototyping of digital receiver hardware focuses on the evaluation of the Zynq Ultrascale+ RFSoC XCZU48DR-2FFVG1517E Field Programmable Gate Array (FPGA) chip. This is a member of the new family of Radio Frequency Systems on a Chip (RFSoC) processors that incorporate analogue to digital conversion onto the same silicon on which the FPGA is implemented. The objective of this prototyping activity is to assess the performance, cost and energy efficiency of the RFSoC processors for the ARGOS array.

Prototyping of the compute accelerators for beamforming focuses on the evaluation of both Tensor-core capable GPUs and PCIe-mounted FPGAs for beamforming and correlation. The objective of this prototyping activity is to compare between GPU and FPGA solutions for ARGOS, considering performance, cost and energy efficiency.

Assumptions and constraints

We continue to use the assumptions laid out in the Simulation section with the addition of the following:

- We assume that all ARGOS array configurations are equally valid and no consideration is taken as to the science impact of a given array configuration.
- We assume that no ARGOS data products pre-beamformer/correlator require more than 8-bits to represent a single sample (16 bits in the case of complex numbers).

Previous work

In Esser & Men (2024), hardware accelerators for beamforming (e.g. GPU and FPGA) were evaluated taking compute density, cost, power consumption, flexibility, interfaces and development cost into account. The study is based on prototyping activities of a beamformer used for Phased Array Feed (PAF) receivers. According to the document a GPU-based beamformer is the preferred, due to its flexibility, re-use for other computing task (e.g. correlator, post-processing tasks), faster prototyping and usage of Tensor Core processors.
The instrumental differences of a PAF compared to single feed telescope array (e.g. ARGOS) are significant. However, the process of beamforming is almost identical for a PAF as it is for an ARGOS-like array.

Note that we do not provide prototyping results for the correlator in this document. The intention is to use the high-performance Tensor Core Correlator (see Romein, 2021). The performance of this software is already well known from the literature and we have been in extensive communication with the author to ensure the software's viability for ARGOS. Work is ongoing to integrate this software into the prototype beamformer pipeline, but at the time of writing no concrete results are available. Reporting on the combined performance of the beamformer and correlator will be the subject of future work.

5. Digital Rx Prototyping

Description

The digital Rx system samples the voltages at the antenna and processes the data using an oversampled polyphase filter bank (OS-PFB). To handle the 1-3 GHz ARGOS band, each channelizer will process two frequency bands (1-2 GHz and 2-3 GHz) and two polarizations (X and Y). Our design employs a sampling rate of 2 Gsps for each input. The resulting digitised data rate is calculated as 2 (bands) x 2 (polarizations) x 2 GHz x 8 bits = 64 Gb/s. The primary objective of this prototyping effort is to assess the feasibility and performance of the RFSoC 4x2 board for developing the digitizer and channelizer for the ARGOS project, including validating the functionality, performance, and integration of various FPGA IPs. The prototyping effort also identifies and addresses potential issues early in the development cycle, reducing risks and uncertainties in the larger ARGOS project.

Scope

The following are considered in-scope for this prototyping activity:

1. Design and implementation
    1. Development and implementation of a channeliser capable of processing two frequency bands (1-2 GHz and 2-3 GHz) and two polarizations (X and Y).
    2. Utilization of Xilinx and custom HLS IPs for ADCs, oversampled FIR filters, FFT, quantization, buffering, and network modules.
2. Testing and validation
    1. Performing C simulations, RTL simulations, and hardware testing for all developed IP modules.
    2. Validation of the timing closure and resource usage on the RFSoC 4x2 board.
    3. Testing the 1PPS trigger functionality to ensure precise time-stamping.
    4. Verification of the DDR4 memory's capability to handle the required data rates for buffering.
3. Integration
    1. Integration of all IP modules to work together within the channeliser framework.
    2. Ensuring compatibility with subsequent beamforming and correlation stages in ARGOS.
4. Performance Metrics
    1. Measurement and reporting of the performance metrics of the channelizer, including data throughput, latency, and accuracy.

The following are considered out-of-scope for this prototyping activity:

1. Production-ready design
    1. The prototype will not be optimized for mass production or long-term deployment.
    2. Reliability and robustness for deployment in various operational environments are not covered.
2. Full ARGOS system integration
    1. While the channelizer's compatibility with ARGOS beamforming and correlation stages will be ensured, the complete integration (including physical integration) with the entire ARGOS system is out of scope.

System requirements

This prototyping activity is intended to de-risk several requirements on the ARGOS backend. Here we paraphrase the corresponding requirement and provide the requirement ID:

1. The system must handle two frequency bands (1-2 GHz and 2-3 GHz) and support processing of two polarizations (X and Y) to meet the RF frontend design. (REQ-ARGOS-BE-01)
2. An OS-PFB is required to enable compatibility with existing VLBI networks. The OS-PFB provides a flat frequency response across the whole band and is trivially invertible to allow for the production of wide-band time-domain data after beamforming. (REQ-ARGOS-BE-05)

3. The system must have 1PPS synchronization to obtain precise timetsamps. Such timestamps are required for the astronomical applications such as high-precision pulsar timing. (REQ-ARGOS-BE-14)

We explicitly do not test here the frequency stability of the digitiser (REQ-ARGOS-BE-15) or the delay buffering and boresight steering capabilities required of the digital Rx (REQ-ARGOS-BE-04).

Hardware selection

The RFSoC 4x2 board is a state-of-the-art FPGA development platform designed by Real Digital. It integrates Xilinx's RFSoC technology, combining the FPGA fabric with high-speed ADCs and DACs on a single chip. Key features and specifications of the RFSoC 4x2 board include: (1) Four ADC inputs capable of sampling at up to 5 GHz with 14-bit precision; (2) A QSFP28 interface supporting high-speed Ethernet connections, offering flexible data transmission rates (4x25 Gbps, 2x50 Gbps, or 1x100 Gbps); (3) Two DDR4 memory modules (4GB each, 64-bit, 2400 MHz). The RFSoC 4x2 board is well-suited for the channelizer application, accommodating up to four inputs sampled at 2 GHz and supporting an output data rate up to 100 Gb/s.

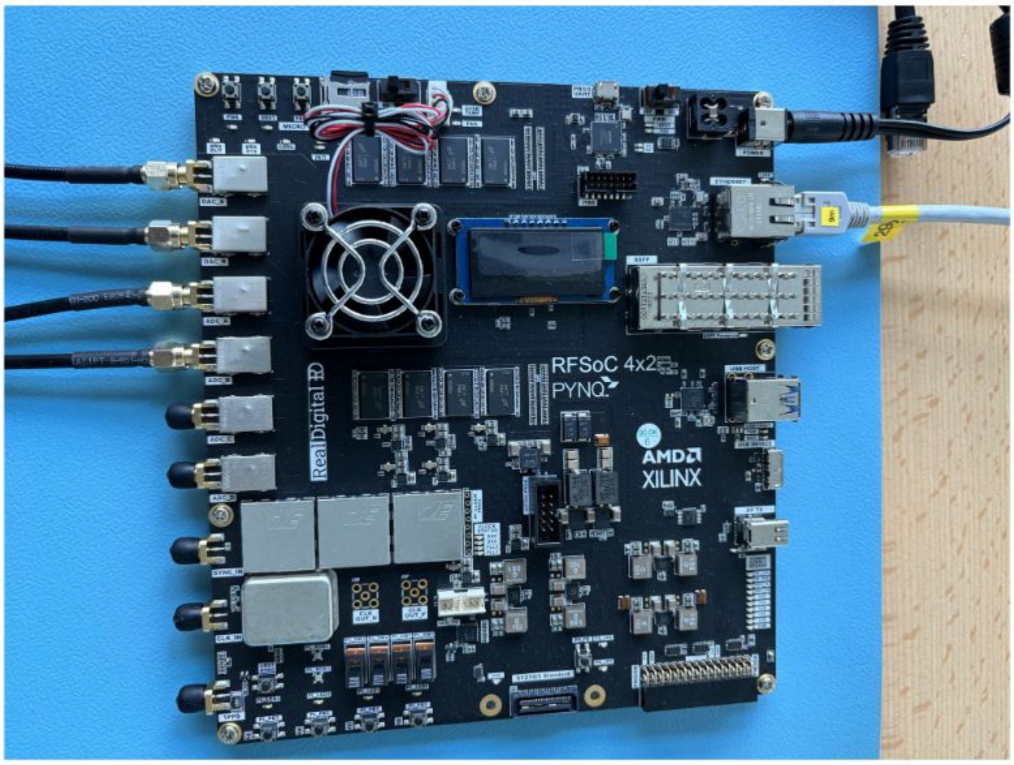

Figure 5.1: Photograph of a RFSoC4x2 board on a bench at the MPIfR

Development environment

Development of the digital Rx prototype was conducted at the Max-Planck Institute for Radio Astronomy in Bonn. The following tools were used in development: (1) Vitis HLS was used for developing the HLS-based IPs; (2) Vivado was used for integrating the different types of IPs (e.g. HLS IPs and Vivado IPs), synthesis of the design and generation of the bitstream file that runs on the FPGA; (3) Python and the PYNQ frameworks were used the development software to run on the ARM CPU that performs the setting up the peripheral equipment and the control of the data transfer between the programmable logic and processing system on the RFSoC.

System architecture

Block diagram

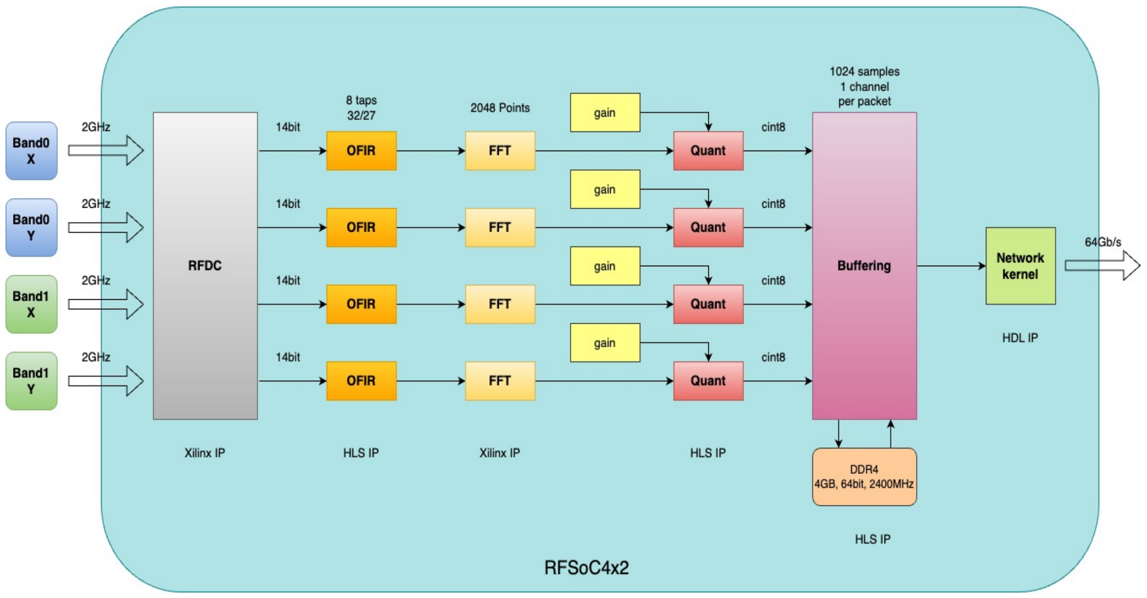

Figure 5.2: Block diagram of RFSoC-based digital Rx design

Detailed design

The system design contains the custom HLS IPs and Xilinx IPs, including:

RFDC: This is the Xilinx IP for the ADCs and DACs on the RFSoC chip, capable of setting up the sampling frequency, mixer configurations, and more. The IP has already been tested on the board.

OFIR: This is the oversampled FIR filter of the OS-PFB. The current design employs 8 taps and an oversample ratio of 32/27. The module is implemented using HLS, with fixed coefficients in the firmware. The C simulation has been successfully completed. The RTL simulation and hardware testing are pending.

FFT: This is the Xilinx SSR FFT IP, which handles complex inputs with multiple samples per clock. Our current design uses a 2048-point FFT.

Quant: This is the quantization module, which converts the FFT output to an 8-bit range. This module will be implemented using HLS and is pending.

Buffering: This module transforms the FFT output from TxF format to FxT format. It requires a large buffer (4*1024*1024*2B=8 MB), which exceeds the FPGA's RAM resources, so the onboard DDR4 (64-bit, 2400 MHz) will be used. Although the bandwidth is theoretically sufficient, verification is pending. The module will be implemented using HLS.

Network kernel: This module comprises a UDP IP and CMAC IP. The module has been implemented and tested.

Data flow

The connection between modules is using the AXI4-stream interface, and the data flow can be described as follows:

1. The four inputs will be sampled on the RFSoC chip at a clock frequency of 2 GHz, producing four data streams. Each output stream is 128 bits wide, including 8 samples from one input.
2. Each output data stream from the ADC module then goes through an oversampled FIR filter, maintaining the same format as the input.
3. Each output data stream from the OFIR module then goes through the Xilinx SSR FFT module, producing an output data stream with 8 channels per clock cycle. The output is in complex ap_fixed<25,12> format.
4. The data stream then passes through a quantization module to cast the data range to -128 to 127 (complex int8).
5. The data stream is buffered in the DDR4 memory and outputs 8 samples of one channel per clock cycle. The output data stream contains 1024 samples for each successive channel in one frame.
6. Finally, the data stream is packetized in the SPEAD format and transferred through the QSFP28 interface.

Algorithm development

The details of the algorithm behind the of OS-PFB can be found in Smith et al. 2020. A visualisation of the difference between critically sampled and oversampled frequency channels is shown in Figure 5.3.

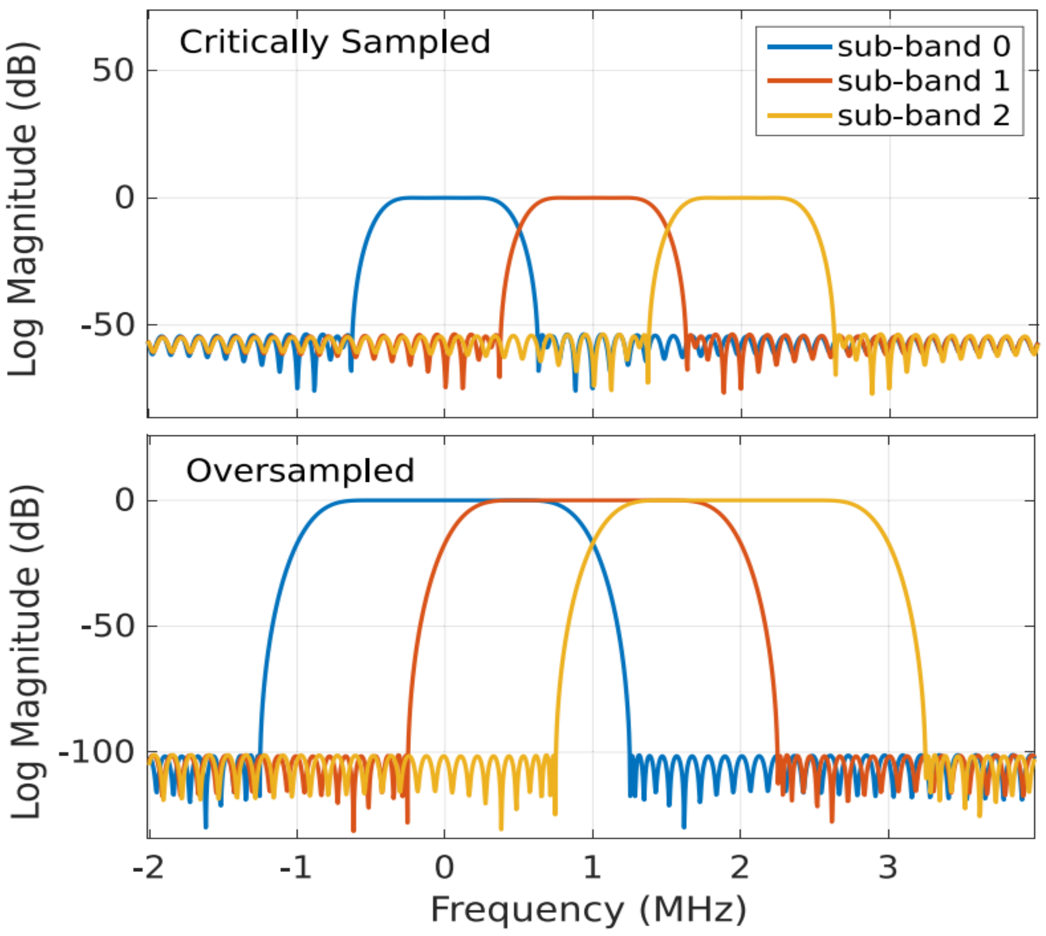

Figure 5.3: Frequency response of data after critical sampling vs after oversampling.

FPGA programming

The main programming effort includes the development of the oversampled FIR filter using HLS, and integrating it with Xilinx IPs (e.g. RFDC and FFT) in the Vivado block design. Then, the RTL simulation and synthesis are done in Vivado,

Software development

The primary software development involves Python scripts utilizing the PYNQ library, which provides a wrapper for drivers to communicate with all peripheral devices. Below is an example of RFDC testing.

```
import matplotlib.pyplot as plt
import numpy as np
import xrfclk
import xrfdc
from pynq import Overlay
from pynq import allocate
from time import sleep
import asyncio

xrfclk.set_ref_clks(lmk_freq=245.76, lmx_freq=409.6)
overlay = Overlay("/home/xilinx/design_1.bit", ignore_version=True)
overlay.download()

py_buffer = allocate(shape=(4096,), dtype=np.int16)
print("py_buffer physical address {}".format(hex(py_buffer.physical_address)))
overlay.dma.axi_dma_2.recvchannel.transfer(py_buffer)

# refresh snapshot
overlay.snapshot.snapshot_2.write(0x10, 0x0)
overlay.snapshot.snapshot_2.write(0x10, 0x1)
sleep(1)
overlay.snapshot.snapshot_2.write(0x10, 0x0)

py_buffer.sync_from_device()
```

Test plan

ADC Testing

1. Input known signal waveforms to ADCs.
2. Capture and analyze digitized output from ADCs.
3. Verify the output matches the input signals.

OFIR Filter Testing

1. C simulation and RTL simulation
    1. Input noise data to the kernel.
    2. Compare the output with reference data calculated from pure python code.

OFIR Filter + FFT + Quant Testing

1. C simulation and RTL simulation
    1. Input noise data to the kernel.
    2. Compare the output with reference data calculated from pure Python code.
2. Hardware test (not done yet)
    1. Input a single-tone signal into the ADCs.
    2. Capture the output stream and transfer to ARM CPU memory.
    3. Verify the output spectrum.

Buffering test (pending)

1. C simulation and RTL simulation
    1. Input noise data to the kernel.
    2. Compare the output with reference data calculated from pure Python code.
2. Hardware test
    1. Input a single-tone signal to the ADCs.
    2. Capture the output stream and transfer to ARM CPU memory.
    3. Verify the output spectrum and test the throughput.

Packetization and Network Transmission Testing

1. Hardware test
    1. Generate counter number by a module and input to the network kernel.
    2. Capture the output packet in the network.
    3. Verify the output network packet.

1PPS synchronization Testing (pending)

1. Hardware test
    1. Input the 1PPS signal to ADC and 1PPS port.
    2. Capture the data of ADC and 1PPS on FPGA.
    3. Verify if the data are synchronized.

System Integration Testing (pending)

1. Hardware test
    1. Input a single-tone signal to the ADCs.
    2. Capture the output packet in the network.
    3. Verify the output network packet.

Metrics Used for Validation

1. Data throughput (Gbps) and latency (ms) of the system.
2. Timing accuracy of 1PPS trigger.
3. Resource Utilization
4. Power consumption

Test setup

For the test setup, we use the following devices to support the experiments on RFSoC 4x2 board:

- The signal generator (Rohde & Schwarz SML02) is used to generate a test signal with a frequency range from 9 kHz to 2.2 GHz;
- The digital oscilloscope (Tektronix DPO 4034) is used for visualization and analysis of signal waveforms from DACs of RFSoC 4x2 and the signal generator;
- The 1 GbE network is used for communication between the on-chip ARM CPU and laptop;
- The JTAG is used for hardware debugging of the FPGA design;
- The QSFP28 transceiver and optical fibre are used for testing the 100 GbE network.

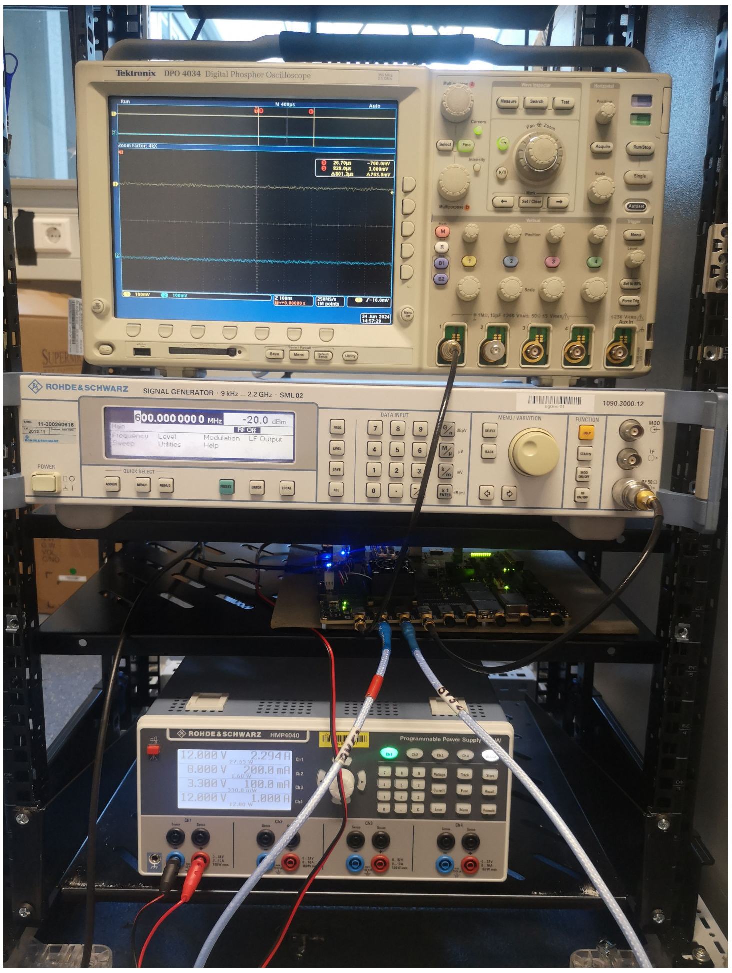
Figure 5.4: Testing setup showing the signal generator, oscilloscope and power supply used.

Results

1. We have tested the ADC on the RFSoC. The plot below shows the spectrum of the data captured from the ADC output. A 600 MHz single-tone signal was input, and the sampling rate was 2.048 GHz.

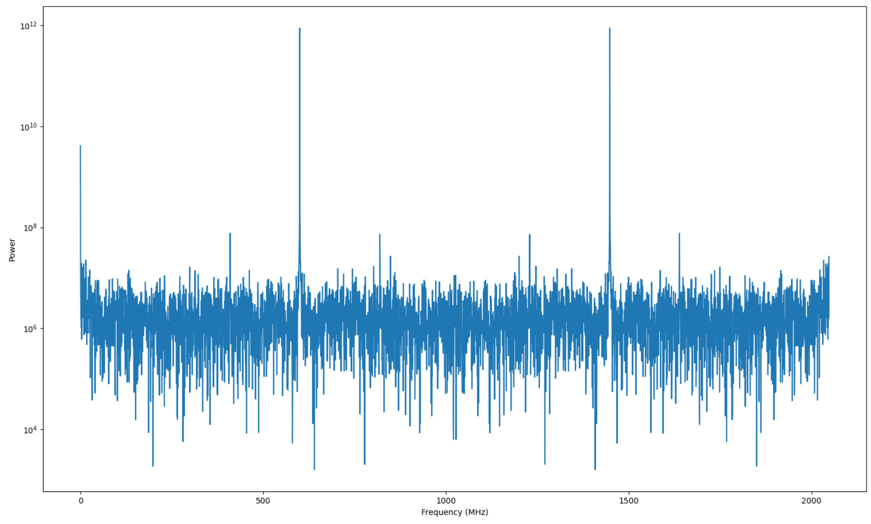

Figure 5.5: Spectrum of the captured data of RFSoC ADC.

2. The custom oversampled FIR filter has been tested in HLS C simulation with a custom FFT IP. The plot below compares the frequency response of an 8-tap PFB with a Hamming window between the simulation and the theoretical calculation.

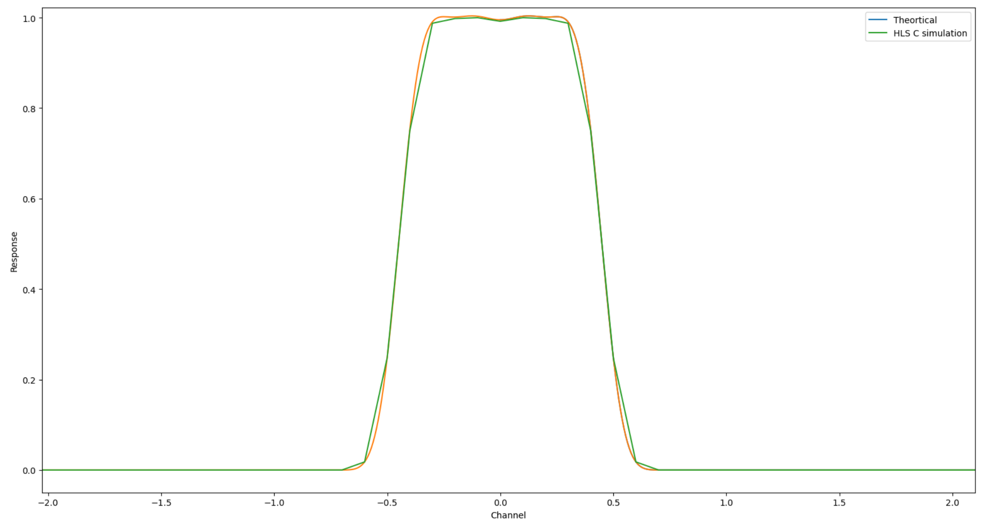

Figure 5.6: Comparison of the PFB channel response between the HLS C simulation and theoretical calculation.

Resource utilization

The resolution utilization of the design is derived from the synthesis of the IPs in Vivado. This provides only an estimate of the utilization, which may differ after the final implementation.

Table 5.1: Resource Utilization. In the table, LUT means Lookup Table, FF means Flip Flops, BRAM means Block RAM and DSP means digital signal processor.

Module	**LUT**	**FF**	**BRAM**	**DSP**
RFDC	1.5%	0.6%	0	0
OFIR	4x 5%	4x 0.7%	4x 3%	4x 1.5%
FFT	4x 5%	4x 3%	4x 0.3%	4x 1.5%
Quant	-	-	-	-
Buffering	-	-	-	-
Network	3%	2.8%	5.3%	0
Total	<50%	<20%	<20%	<15%

Challenges and solutions

Technical challenges

1. **Buffering:** Buffering the output stream and transposing the data from TxF format to FxT format is challenging because as requires a large buffer (at least 8 MB) to stream the data in real-time. The buffer size exceeds the available memory on the FPGA.
2. **Timing Closure:** The total LUT resource usage is nearing 50% will pose challenges for achieving timing closure.

Problem-solving approaches

1. **Buffering**: The DDR4 memory on the RFSoC4x2 board will be utilized to address this challenge, and thorough testing of its functionality is necessary.
2. **Timing closure:** The required clock frequency is less than 300 MHz, which should help in achieving timing closure. Rigorous testing is essential to confirm this.

Conclusions

Summary

The prototyping effort focused on developing a channelizer using the RFSoC 4x2 board, integrating various FPGA IPs and custom modules. Here are the key findings and outcomes:

1. The ADC has been successfully tested and shown to operate at 2 GHz sampling rate with 14-bit precision.
2. A custom oversampled FIR filter has been developed using HLS, and both C simulation and RTL simulation have been successfully passed.
3. A wrapper for the Xilinx Super Sample Rate (SSR) FFT module has been implemented with an AXI4-stream interface and tested in RTL simulation.
4. A custom HLS SSR FFT module has been developed for C simulation testing of the oversampled FIR filter.
5. Resource usages for these modules have been obtained and are detailed in the resource utilization section.

Lessons learned

Image building

1. Building a custom Linux image for the RFSoC FPGA can be challenging. A more practical approach is to use the official Linux image and develop custom overlays on top of it.
2. For remote connection to the RFSoC 4x2 board via DHCP, running a SOCKS proxy is necessary to ensure proper operation of JupyterLab.

Zynq development

1. DMA (Direct Memory Access) can facilitate high-throughput data transfer between the Programmable Logic (PL) and Processing System (PS) on the chip.

RFDC

1. Due to PLL configuration restrictions of the RFDC module, a 2 GHz clock frequency may not be supported using the internal reference clock, whereas 2.048 GHz is supported.
2. The PYNQ library provides a wrapper for drivers that enables control and configuration of the ADCs and DACs.

HLS development

1. Clock-level triggering may not be supported in HLS because the trigger signal is sampled at one stage of the internal state machine, rather than every clock cycle.

FFT

1. There are three FFT IPs provided by Xilinx: Vitis HLS SSR FFT library, Vivado FFT IP, and Vitis Model Composer SSR FFT module. The latter is the preferred choice as it supports SSR FFT and has lower resource usage.
2. To ensure the central frequency of the output bandpass is at the center, the oversampled FIR filter requires shift and overlap adjustments.

3. For a 2048-point FFT with ap_fixed<14,1> input, the maximum value of each stage increases by a factor of 2. Given this scenario, scaling is not necessary because the output value of each stage should remain within 27 bits, which is a restriction of the DSP on the FPGA.

Future work

Areas for future improvement or further development based on the prototyping effort include:

1. Perform hardware testing for the HLS OFIR module combined with the Xilinx SSR FFT.
2. Implement the quantization and buffering modules using HLS.
3. Validate DDR4 throughput capabilities to ensure it supports the buffering module requirements.
4. Test the functionality of the 1PPS trigger for precise timestamping.
5. Validate the network kernel functionality on the RFSoC 4x2 board.
6. Conduct integration testing with the network kernel to ensure seamless operation within the channelizer system.

Additional documentation

Below we include links to additional documentation and data sheets relevant for this prototyping activity.:

https://www.realdigital.org/hardware/rfsoc-4x2
https://pynq.readthedocs.io/en/latest/
https://docs.amd.com/r/en-US/ug1085-zynq-ultrascale-trm
https://docs.amd.com/r/en-US/pg269-rf-data-converter
https://docs.amd.com/r/2022.1-English/ug1483-model-composer-sys-gen-user-guide/Vector-FFT

6. Beamformer Prototyping

Description

Prototyping activities for the ARGOS beamformer have focused on the development of Tensor-core enabled beamforming software that can make maximally efficient use of GPU hardware. The goals of this prototyping activity were to:

1. Design and test a beamformer pipeline for ARGOS
2. Demonstrate that there are beamforming algorithms that can make use of Tensor cores.
3. Benchmark the performance of the pipeline components.
4. Estimate the computing resources required to beamform a full-size ARGOS array.

Software development

The prototype beamformer for ARGOS was developed in C++ / CUDA and builds upon the PSRDADA_CPP library (https://gitlab.mpcdf.mpg.de/mpifr-bdg/psrdada_cpp).

Pipeline design

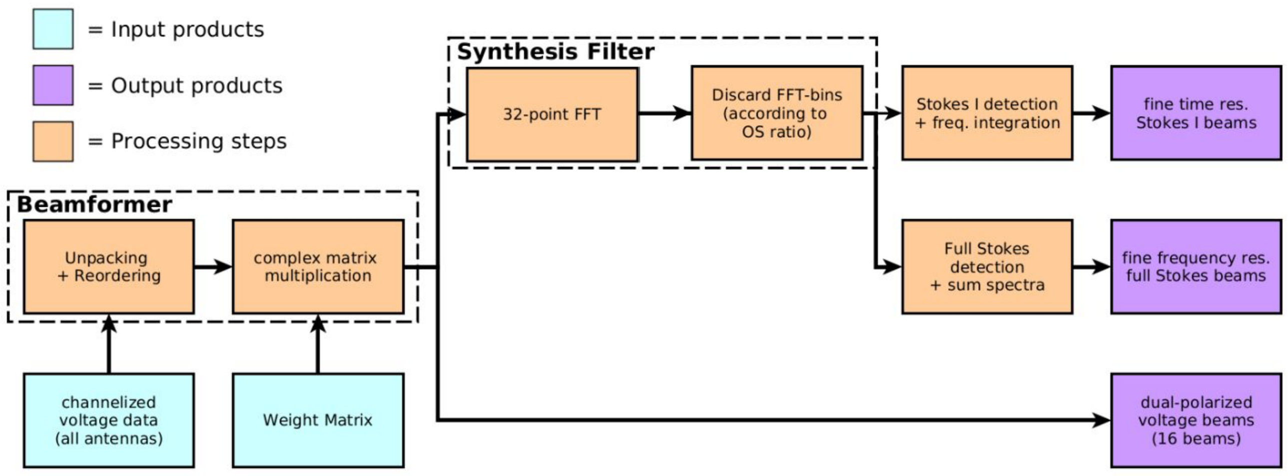
= Input products
= Output products
= Processing steps
Synthesis Filter
32-point FFT
Discard FFT-bins (according to OS ratio)
Stokes I detection + freq. integration
fine time res. Stokes I beams
Beamformer
Unpacking + Reordering
complex matrix multiplication
Full Stokes detection + sum spectra
fine frequency res. full Stokes beams
channelized voltage data (all antennas)
Weight Matrix
dual-polarized voltage beams (16 beams)

Figure 6.1: Prototype beamforming pipeline.

Figure 6.1 shows the beamformer pipeline design used for the prototype beamformer. The pipeline is principally composed of 4 stages:

Stage 1: Data arriving from network or file go thorough **unpacking and reordering** to be converted into the internal representation required by downstream components of the pipeline. This stage is performed partly in network capture code and partly by applying a multidimensional asynchronous memory copy from the host to the GPU. The asynchronicity allows for overlapping of the memory copies with the subsequent processing stages.

Stage 2: The actual **beamforming** is a complex general matrix multiplication (CGEMM). To perform the CGEMM, Tensor Cores are utilized. After this stage dual-polarized voltage beams are computed. The outputs of the CGEMM are dual-polarisation voltages beams. As subset of these beams are required for specific science cases (VLBI and pulsar timing) and are retained at full resolution. The remaining beams continue to stage 3.

Stage 3: A **synthesis filter** is applied to the output of the CGEMM to undo the oversampling of the channelizer, converting the data into its critically sampled equivalents. Currently, the synthesis filter is implemented by applying a 32-point forward FFT and discarding the edge bins with respect to the oversampling ratio (e.g. for a 32/27 oversampling we discard 5 edge bins).

Stage 4: The outputs of the synthesis filter are **detected and integrated** in two substages to produce a full Stokes output and a Stokes I only output (see below for details).

Output products

One design requirement of the beamformer is to provide different output products to satisfy different science use cases simultaneously. The three beamformed outputs produced by the pipeline are described below.

Dual-polarisation voltage beams are used for science cases that rely on having voltage data (e.g. VLBI and pulsar timing). As no integration takes place, the voltage beams produce the highest data volume per beam. The data rate of a single beam is equal to the data rate of a single antenna. To keep the output data rate of the beamformer low, only 16 voltage beams are produced.

The **full-Stokes beams** provide high spectral resolution by further channelizing the data. This data product is used for spectroscopic observations. Assuming a coarse channelization of 1024 channels with an oversampling ratio of 32/27, the full Stokes spectrum of the prototype beamformer has a frequency resolution of 72 kHz. This is achieved by not performing any frequency integration on the data in after the synthesis filter, resulting in 1024 x 27 = 27648 channels across the 2 GHz band. In future, the resolution will be made more flexible by supporting different fine channelization modes with different FFT lengths in the synthesis filter (e.g. 128, 512 and 1024) up to a frequency resolution of 2.2 kHz. The spectra produced in this mode are integrated over a user-defined interval.

The **Stokes I high temporal resolution beams** are primarily intended to be used for fast transient searching. In contrast to the full-Stokes beams, the high time resolution beams are integrated in frequency. The FFT length in the synthesis filter determines the time resolution of this data product. For instance, when using a 32-point forward transform in the synthesis filter (the minimum for a 32/27 oversampling ratio) the minimum corresponding time resolution is 14 µs. These data may also be integrated in time to produce coarser resolution beams.

Note that by increasing the number of FFT points used in the synthesis filter the time resolution of the Stokes I beams becomes coarser, while the frequency resolution of full Stokes beams becomes finer. A trade-off between the two data products has to be made on an individual observation basis.

Benchmarks

The following benchmark assumes the following batch properties processed by a single node (GPU):

- Number of antennas: 1120
- Number of polarizations: 2
- Number of channels: 2
- Number of samples: 262.144
- Number of integrated spectra: 32
- Number of FFT points in synthesis filter: 32

The benchmarks were carried out on an NVIDIA L40 GPU hosted in a SUPERMICRO A+ Server AS-4125GS-TNRT with dual 32-core AMD EPYC Genoa 9334 CPUs and 768 GB of RAM.

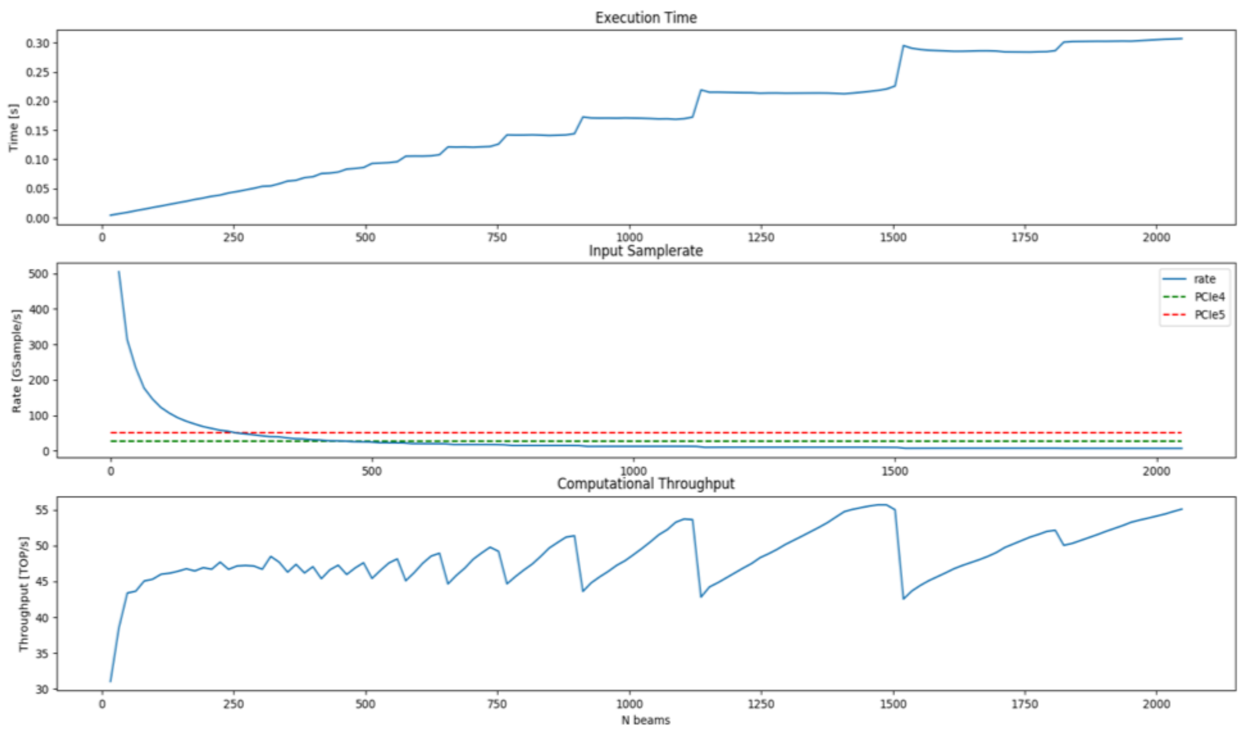

Figure 6.2: Beamformer benchmark on a Nvidia L40 GPU, 50 iterations per data point. (1) shows the averaged time to execute the GPU-kernel, (2) shows the maximum ingest sample rate the beamformer can process, (3) shows the computational throughput (note that this is not the absolute throughput in TOPS but is scaled by some unknown constant).

The benchmarks show the performance of the beamformer as a function of the number of computed beams. For numbers of beams fewer than 272 and 512, the beamformer operates faster than the theoretical PCIe5 and PCIe4 throughputs, respectively (i.e. beyond these numbers we are performance capped by the PCIe bus). The beamformer operates most efficiently for certain numbers of beams (e.g. 1120, 1536), which is visible where the computational throughput peaks or before the execution time steps up. This is due to the performance of the underlying CGEMM operation changing with input matrix dimensions. From the presented benchmark we can estimate the number of GPUs that are required to produce N beams for the full ARGOS array. This is shown in Figure 6.3.

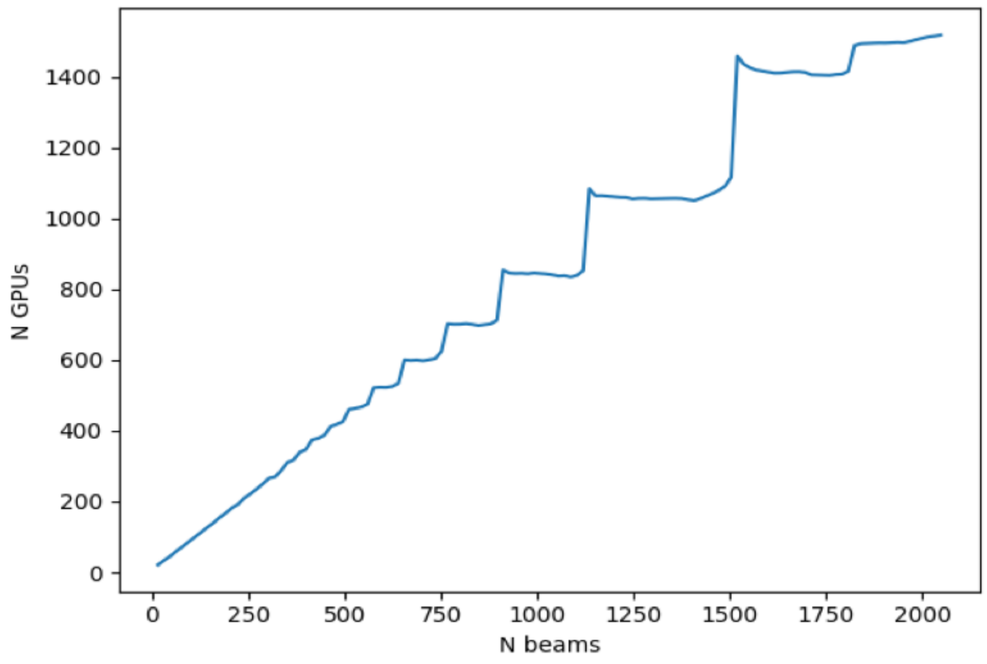

Figure 6.3: Number of GPUs vs number of beams

Conclusion

Summary

This prototyping activity focused on the development, testing and benchmarking of a Tensor Core capable C++ / CUDA beamformer for ARGOS. This development was a success, providing a means to benchmark the beamformer performance and estimate the compute resources required to form different numbers of beams for the full ARGOS array. We see that with approximately 1000 GPUs it is possible to form upwards of 1500 beams for wide-field survey work.

Lessons learned

- CUDA Tensor Cores are not currently natively aware of complex numbers and as such CGEMMs must be written as a combination of GEMMs.
- Manual JIT (just-in-time) compilation or JIT compilation using the Nvidia Runtime Compiler (NVRTC) can offer significant performance improvements when certain parameters are known ahead of time.
- The spectroscopic outputs of the beamformer pipeline are likely not required for ARGOS as the use case they satisfy is also satisfied by imaging. Removing this branch of the pipeline would result in a small increase in performance.
- For a sufficiently large number of beams, the beamformer performance approaches the GPUs theoretical performance limit, indicating that the code can make maximal use of the hardware.

Future work

In order to turn the prototype pipeline into a working pipeline for the ARGOS pathfinder array, several additional pieces of software will be required. Some of these, like the code to produce beam weights are already developed and in operation at other facilities but others are not commonly used elsewhere and require careful development. In particular, effort is required to de-risk the filterbank inverter that will be required to convert the channelised dual-polarisation voltage beams back into pure wide-band time-domain signals as is required for their use in VLBI experiments. This work has already begun and results are expected in a few months.

References

1. J. P. Smith et al., "A High-Throughput Oversampled Polyphase Filter Bank Using Vivado HLS and PYNQ on a RFSoC," in IEEE Open Journal of Circuits and Systems, vol. 2, pp. 241-252, 2021, doi: 10.1109/OJCAS.2020.3041208.
2. Romein, J. W., “The Tensor-Core Correlator” in A&A, 656, A52, 2021, dio: 10.1051/0004-6361/202141896
3. N. Esser, Y. Men, 2024 “Device Evaluation for the CryoPAF CBF”, Unpublished internal institute document, https://argos-cloud.ia.forth.gr/index.php/s/nBmHAX76oLSPDMP