RAA 2019 Vol. **19** No. **7**, 102(10pp) doi: 10.1088/1674–4527/19/7/102

http://www.raa-journal.org *http://iopscience.iop.org/raa*

***R**esearch in*
***A**stronomy and*
***A**strophysics*

The design and implementation of a ROACH2+GPU based correlator on the Tianlai dish array

Chen-Hui Niu[1,2,3], Qun-Xiong Wang[2,4], David MacMahon[3], Feng-Quan Wu[2], Xue-Lei Chen[2,5,6], Ji-Xia Li[2,5], Hai-Jun Tian[4], Guillaume Shippee[3], Dan Werthimer[3] and Xiao-Ping Zheng[1]

[1] College of Physical Science and Technology, Central China Normal University, Wuhan 430079, China; *zhxp@mail.ccnu.edu.cn*

[2] Key Laboratory for Computational Astrophysics, National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China; *xuelei@cosmology.bao.ac.cn, wufq@bao.ac.cn*

[3] University of California Berkeley, Campbell Hall 339, Berkeley CA 94720, USA

[4] College of Science, China Three Gorges University, Yichang 443002, China

[5] School of Astronomy and Space Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

[6] Center of High Energy Physics, Peking University, Beijing 100871, China

Received 2019 March 29; accepted 2019 April 9

Abstract A digital correlator is a crucial element in a modern radio telescope. In this paper, we describe a scalable design for the correlator system of the Tianlai pathfinder array, which is an experiment dedicated to testing key technologies for conducting a 21 cm intensity mapping survey. The correlator implements the FX design, which firstly performs a fast Fourier transform (FFT) including polyphase filter bank (PFB) computation using a Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) Reconfigurable Open Architecture Computing Hardware-2 (ROACH2) board, then computes cross-correlations by employing Graphics Processing Units (GPUs). The design has been tested both in laboratory and in actual observation.

Key words: instrumentation: interferometers

1 INTRODUCTION

In a radio astronomy telescope array, the correlator is implemented to obtain the short time average of signal correlations, which is called the (interferometric) visibility. By performing the cross-correlation, phase information about the radio wave is preserved, and the sky radio intensity map can be reconstructed in the synthesis imaging process by using the visibilities as input data (Thompson et al. 2001). The correlator thus serves a central function in radio telescopes.

The Tianlai project is a 21 cm intensity mapping experiment with the aim of measuring the baryon acoustic oscillations (BAO) features in the matter power spectrum (Chen 2012). Currently, two pathfinder arrays have been built in a radio quiet site in Hongliuxia, Balikun county, Xinjiang, China. The cylinder array includes three cylindrical reflectors, which are 15 meters wide and 40 meters long oriented in the north-south direction. It has a total of 96 dual polarization receivers, which generate 192 input channels. The dish array includes 16 dishes with 6-meter aperture. Each dish is equipped with a dual polarization receiver, and a total of 32 input channels is produced. The radio signal collected in the antenna feed is amplified by a low noise amplifier, which is then converted to an optical signal and transmitted via an optical fiber cable to the station house located 8 km away. Afterwards, the optical signal is converted back to a radio signal, which is sent to the receiver. The analog receiver utilizes a heterodyne design with the intermediate frequency band of 135−235 MHz. The digital backend samples this signal at a rate of 250 Mega samples per second (MSPS). The correlator handles the full polarizations, and produces both cross-correlations and auto-correlations.

A prototype system with 32 input channels based on this channel has been built and installed on the pathfinder Tianlai dish array. The design of this system (e.g., the employment of a network switch instead of communication within one chassis) also allows it to be expanded to a scale which can handle the 192 receiver channels. The current Tianlai cylinder array uses a 192-channel digital correlator built by the Institute of Automation, Chinese Academy of Sciences, which relies on Field-Programmable Gate Array (FPGA) boards and digital signal processors (DSP) that they designed themselves. Here we introduce the 32-channel system as well as the 192-channel system design in some details for future reference.

2 SYSTEM DESIGN

The visibilities of the interferometry array are usually computed in either of the following two ways: 1) the XF type in which the time-ordered voltage signals from the different receiver channels are paired and convolved with each other to produce the cross-correlations. A Fourier transform is then performed to obtain the visibilities at different frequencies; 2) the FX type in which the voltage signal of each receiver channel is first Fourier transformed to produce a spectrum, and then each pair is cross-correlated for the same frequency. In either way, the result is integrated for some pre-specified duration to yield the final output. As multiplying in frequency domain is equivalent to convolving in the time domain, the results from these two types of correlators are identical. With modern digital technology and larger telescope arrays, the FX type is more convenient to implement (Price et al. 2016), thus it has became more popular, and we also choose the FX type in this design.

One design of the Tianlai correlator system we considered is based on an FPGA board for the data sampling and performing the fast Fourier transform (FFT). An instantaneous frequency spectrum is then derived from the time series data. The data from different channels are then transposed, i.e., re-arranged so that the data with same frequencies from different receivers are put together. The transposed data are sent to an array of Graphics Processing Units (GPUs) via a 10 Gbit s^{-1} network switch, which computes the cross-correlations. Computationally, the cross-correlation is a multiply and accumulation (CMAC) process. The FPGA board we installed is the Reconfigurable Open Architecture Computing Hardware-2 (ROACH2) board[1], which has been widely utilized in radio astronomy projects (Hickish et al. 2016) (e.g., the Precision Array for Probing the Epoch of Re-ionization (PAPER) (Parsons et al. 2010)). Our design is built upon the PAPER correlator model (Parsons et al. 2008), which creates a flexible and scalable hybrid correlator system.

Our correlator consists of the F-engine, the network switch and the X-engine. The F-engine is dedicated to performing the Fourier transform, while the X-engine is dedicated to computing the cross-correlations. The network switch is used to transpose the data. We use the CASPER ROACH2 board for the F-engine and the NVIDIA$^{\rm TM}$ GPU board for the X-engine. The required number of ROACH2 boards is dependent on the number of receiver inputs. Suppose the F-engine consists of M ROACH2 nodes and the X-engine has N GPU nodes. Every ROACH2 node handles m-way analog radio inputs, and after performing the FFT, it outputs m spectra. Each spectrum has F frequency points. To optimize data traffic, each GPU node processes F/N frequency points, and each frequency point includes $M \times m$ conjugate multiply computations. Therefore, the ROACH node should divide the F point spectrum into N bands, and then send the specified band to the corresponding GPU node for the CMAC computation.

A block diagram of the correlator structure is shown in Figure 1. The F-engine is built with CASPER ROACH2 (R2 2012 version). Each ROACH2 board is connected with two daughter analog-to-digital converter (ADC) boards through the Z-DOK+ connectors. The ADC board is the CASPER ADC 16×250-8 Q2 2012 version, which uses four HMCAD1511 chips made by Hittite Microwave$^{\rm TM}$ with a total of 16 inputs[2] and samples 16 analog signal inputs with 8 bits at 250 Msps. The ADC input ports are Sub-Miniature-A connectors, which require analog signal of -8.5 dBm Gaussian noise. We find that good linearity can be achieved for an input ranging from –12 dBm to 6 dBm.

The ROACH nodes are controlled by a ROACH2-server. In our case, it is built on a Dell$^{\rm TM}$ PowerEdge T110 with Intel$^{\rm TM}$ Xeon(R) CPU E31220, which can itself be controlled remotely through the Internet. The ROACH2 boards can be booted either by loading the kernel from the ROACH2 server (net boot) or from the onboard memory chip (solo boot). A Berkeley Operating System for ReProgrammable Hardware (BORPH) operating system, which is a full-featured Linux operating system supporting FPGA applications (So Hayden Kwok-Hay 2007), is run on the ROACH2 boards, and the ROACH2 boards

[1] https://casper.berkeley.edu/wiki/ROACH2

[2] https://casper.ssl.berkeley.edu/wiki/ADC16x250-8

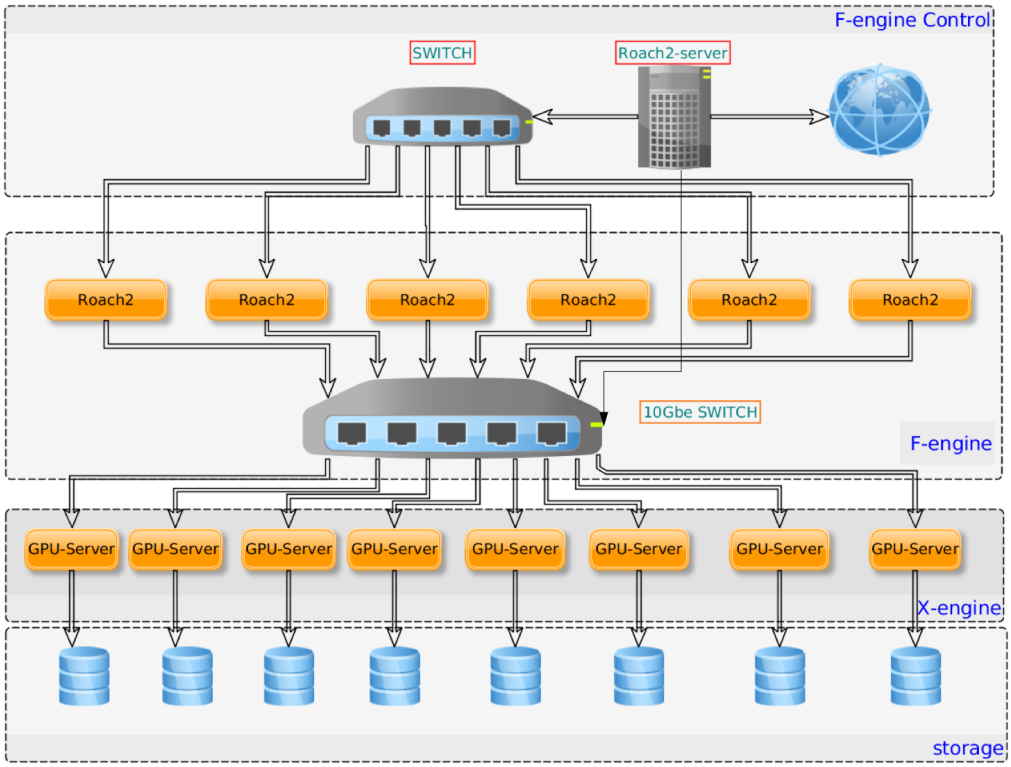

Fig. 1 Block diagram of the Tianlai dish array correlator.

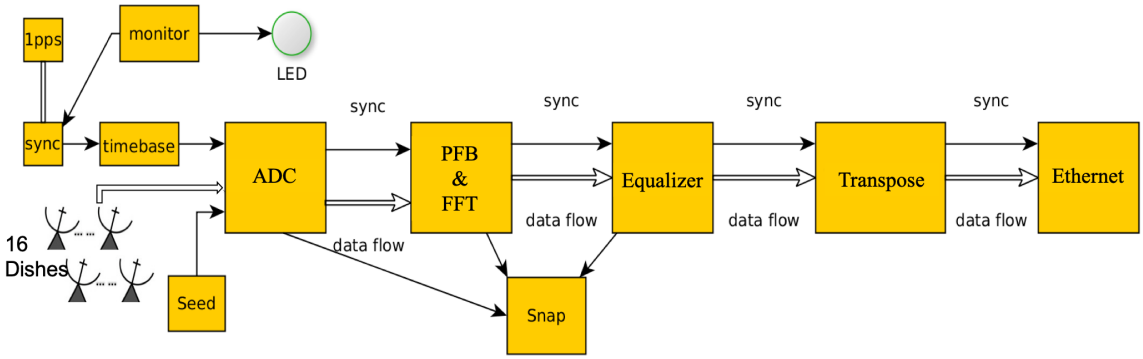

Fig. 2 The F-engine data flow block diagram. The data flow starts from ADC, goes though the PFB, Equalizer, Transpose, then is sent via an Ethernet interface to the X-engine.

can be controlled via an interface called the Karoo Array Telescope Control Protocol (KATCP) (Foley et al. 2016). Compiled binary executable programs (bof file) can be uploaded to the ROACH2 boards as firmware.

The output data from the F-engine are initially grouped by receiver channels, i.e., a block of spectra has the same receiver channel but different frequencies. However, only the correlations for the data with the same frequency are non-zero and need computation, and for this computation the spectra from different receiver channel pairs are needed by the computing unit. So, the data should be re-arranged according to their frequencies. This is done by a pre-specified program in the F-engine. The data are packaged and then sent to different ports on the X-engine.

In the X-engine, the correlations are obtained and then sent to a hard drive server for storage. Our X-engine is built with NVIDIA GTX 690 GPUs[3]. Each GPU node has two GTX 690 units, which are equipped with two GPU cores. The CASPER High Availability Shared Pipeline Engine (Hashpipe)[4] is used to manage the GPU threads and data transfer within each GPU node.

[3] https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-690

[4] https://github.com/david-macmahon/hashpipe

For the dish array, one ROACH2 board can handle the 32 inputs, and the cross-correlation can be done with 1 GPU node. Indeed, in this case there is no need to rely on the network switch, and data from the F-engine can be sent to the X-engine port directly. However, as an option for the cylinder correlator, we also considered the design of a system with 192 receiver input channels. In this case, the F-engines consist of six ROACH2 boards, and each board handles 32 inputs. For the X-engine, we estimate the total amount of computation for the 192-input system is 35.6 TFLOPS, while the computing performance of each node is 4.8 TFLOPS. We thus estimate at least 8 GPU nodes (16 GPUs) are required for computation.

2.1 F-engine

The F-engine includes four main function blocks: 1) the FFT block that channelizes the digital data from ADC; 2) the Equalizer block that truncates the data to reduce the size; 3) the Transpose block that transposes the data; 4) the Ethernet block that sends the data to the corresponding GPU nodes based on the frequency.

As the functions and requirements for our correlator are very similar to those of the PAPER experiment correlator, which also incorporated the ROACH2 system, we based our design on its design[5]. The whole design can be simulated with the Matlab Simulink$^{\mathrm{TM}}$ CASPER tool set, which has a graphical user interface (GUI) for the FPGA programming.

An F-engine data flow block diagram is shown in Figure 2. The whole system is triggered and synchronized with the 1 pulse per second (1PPS) block. A monitor block is also added to watch the status of operation. A Seed block is included for testing by generating digital pseudo-random Gaussian noise for the ADC. Additionally, a Snap block is added to debug the ADC, polyphase filter bank (PFB) and Equalizer.

Each input data stream from the ADC is channelized in the frequency domain with a PFB to minimize energy leakage. A Hamming window is multiplied within each data stream. Two length options for the FFT are implemented in our design, which are 512 and 1024 respectively.

The FFT of a real number input will generate complex numbers, but with Hermitian symmetry on the Fourier components, i.e., the negative frequency part is just the complex conjugate of the corresponding positive frequency component, only half of the complex numbers are independent. To avoid wasting memory and data transportation, the input data block of two input channels is combined as the real and imaginary components respectively to form a complex input, and then FFTed together (Proakis & Manolakis 1996). Taking two time series $x_1[n]$, $x_2[n]$ as the real part and imaginary part, the combined complex number is given as

$$y[n] = x_1[n] + j \cdot x_2[n]. \tag{1}$$

Assume the Fourier pair as

$$y[n] \leftrightarrow S[\nu]. \tag{2}$$

The Fourier transform of $x_1[n]$, $x_2[n]$ can be recovered with the Fourier transform of $y[n]$, i.e.,

$$\begin{aligned} \mathscr{F}\{x_1[n]\} &= \frac{S_r[\nu] + S_r[-\nu]}{2} + j \cdot \frac{S_i[\nu] - S_i[-\nu]}{2}, \\ \mathscr{F}\{x_2[n]\} &= \frac{S_i[\nu] + S_i[-\nu]}{2} - j \cdot \frac{S_r[\nu] - S_r[-\nu]}{2}, \end{aligned} \tag{3}$$

where $S_r[\nu]$ and $S_i[\nu]$ are the real and imaginary parts of the Fourier transform of $y[n]$ respectively. This recovery is implemented in the block. The output data numbers of the PFB block are 36 bits long, with the first 18 bits for the real part and the last 18 bits the imaginary part.

In the pipeline, an Equalizer block follows the FFT. It serves two purposes: (1) To compensate for the varying spectral response by multiplying the data with a frequency dependent adjustment factor, so as to obtain a nearly flat spectral response, and achieve good dynamical range in the digital sampling. (2) To reduce the amount of data to be exchanged via the network switch, the original 18 bit data are truncated to 4 bits by the Equalizer. To preserve the maximum amount of information, the Equalizer should be designed such that the data should most frequently fall in a suitable range. For radio astronomy, the data are usually noise-dominated, with occasional outliers mostly coming from radio frequency interferences (RFIs). For such a signal, the root mean square (RMS) can be estimated from the autocorrelation power levels,

$$\mathrm{RMS} = \sqrt{\frac{P}{2N}}, \tag{4}$$

where P is the integrated autocorrelation value and N is the number of samples in the integration. The Equalizer is dedicated to desaturating the output while preserving most information. The RMS of the 4-bit output falls between 2 and 3, out of the full 4-bit range of –7 to +7[6]. Considering

[5] https://casper.berkeley.edu/wiki/PAPER_Correlator_Manifest

[6] https://casper.ssl.berkeley.edu/wiki/PAPER_Correlator_EQ

the RFI, the Equalizer is designed as follows (see Fig. 3). The 36 bit complex data are divided into an 18 bit real part and an 18 bit imaginary part. They are then multiplied with the scale factor, which is an unsigned fix_18_7 number, i.e., an 18 bit long fixed point number with 7 bits after the binary point, and the product is a fix_36_24 number. The most significant bits are usually 0, except for the strong RFI. This number is then rounded even from 21 bit, then the 22nd to 25th bits are selected while the other bits are discarded. The output is a fix_4_3 number, in the range of $1.001_{\rm b}$ to $0.111_{\rm b}$. The b means in binary format. The two 4 bit real numbers are then re-packed to an 8-bit complex number and sent to the next block in the pipeline.

After the FFT step, the data are directed to the X-engine for cross correlation computation. This computation is distributed over multiple units, and each unit would need the same frequency signal from all receiver units. The data need to be transposed (the so called "corner turn"), so as to arrange the data in the desired order, i.e., converting the data shape from [input channel, frequency] to [frequency, input channel], as shown in Figure 4.

This is achieved in the F-engine by writing the data blocks consecutively to a dual-port RAM, and then reading out the data not in the original order, but at pre-specified addresses, so that they can be regrouped in the desired order. The regrouped data are then packaged with a destination address in their headers and redirected to the X-engine via the network interface controllers (NICs) onboard. As we have four 10GbE NICs for each ROACH2 board, the spectrum is split into four sub-bands, denoted by tid=0, 1, 2, 3.

The frequency ordered data are divided into four blocks, so that each block contains 128 (for 512 frequency bins) or 256 (for 1024 frequency bins) frequency points, which are consecutively written into the four dual port RAMs. Each FPGA processes 32 input channels from the ADCs. The 32 inputs are sampled in parallel which are written as 16 complex numbers, with each complex number corresponding to the spectrum of two real inputs. After going through the Equalizer, the length of the data is 4 bit for each real number, and 8 bit in total for the complex number. Within such a sub-block, the data already have the same frequency but different input channels, which are suitable for use by the X-engine. The reading program will read 64 bits of data at a time, corresponding to 16 inputs of 4 bits, so it could acquire 32 inputs of the same frequency by reading in two numbers from the RAM.

Synchronous ports are used in the functional block design, to ensure the synchronization of different ROACH2 nodes to the same clock cycle. After all parameters such as the IP and MAC addresses are set, a synchronization signal called the "Arm signal" is sent to each node. Then, all the ROACH2 nodes are initialized and waiting for a 1PPS signal to trigger the system. The Arm signals at the different nodes are not required to be synchronized, but the 1PPS signals are synchronized by choosing the same length of cable for each signal. After this operation, all the ROACH2 nodes will be synchronized. A related diagram is shown in Figure 5.

2.2 X-engine

When the F-engine is initialized, a trigger signal is also sent to initiate the X-engine. The X-engine receives the data from the F-engine in packets, which are delivered into the different computing nodes, where the CMAC computations are performed. For one GPU node, there are two NVIDIA GTX690 cards, each containing two GPU cores, two Intel$^{\rm TM}$ Xeon CPUs and four 10 Gbps NICs. These four GPU cores work separately, each processing the CMAC of 256 frequency points for the 32 input correlator, or 32 frequency points if there are 192 inputs and 1024 frequency channels. The CMAC process uses the `xGPU` package (Clark et al. 2013), which is written in CUDA-C and is optimized on GPU memory resources by specific thread tasks.

The data movement in the X-engine is managed by Hashpipe[7], which specializes in communication between threads in both GPUs and CPUs. A sketch of the data flow in the X-engine pipeline is shown in Figure 6. We open four Hashpipe threads for each GPU node (note these are NOT the GPU threads) and also set up three ring buffers in the memory for each GPU node. Each ring buffer is divided into three memory segments. The data from the network switch are received by the net thread, which reassembles the data sequence as required by the `xGPU` code, and stores the data in the so-called GPU ring buffer. The GPU thread then processes the data in the buffer and computes the correlations. After the GPU has finished the computation, it outputs the result to the so-called CPU ring buffer. The CPU thread then collects the result in a suitable format, which is then put in the so called disk ring buffer. Finally, the disk thread saves the data on the disk ring buffer to files on the storage system. In the pipeline, before one thread starts to work, it will fill the ring buffer and alert the next

[7] `https://github.com/david-macmahon/hashpipe`

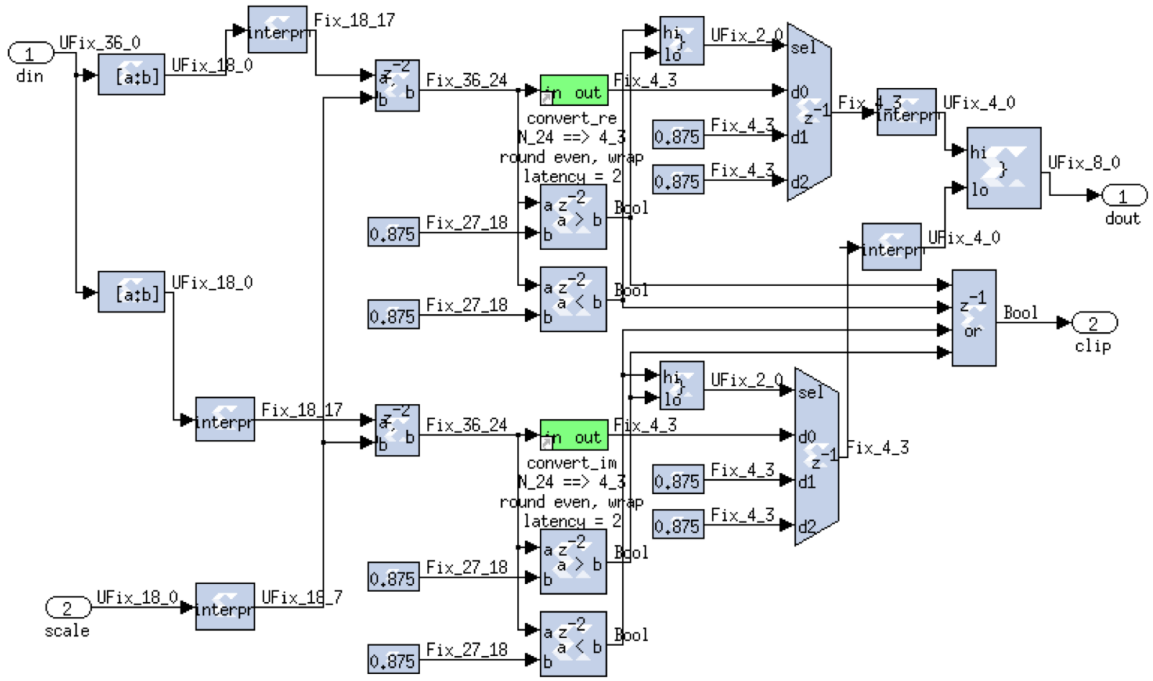

Fig. 3 Equalizer block. The scale factor could be varied both with different frequency and input channel.

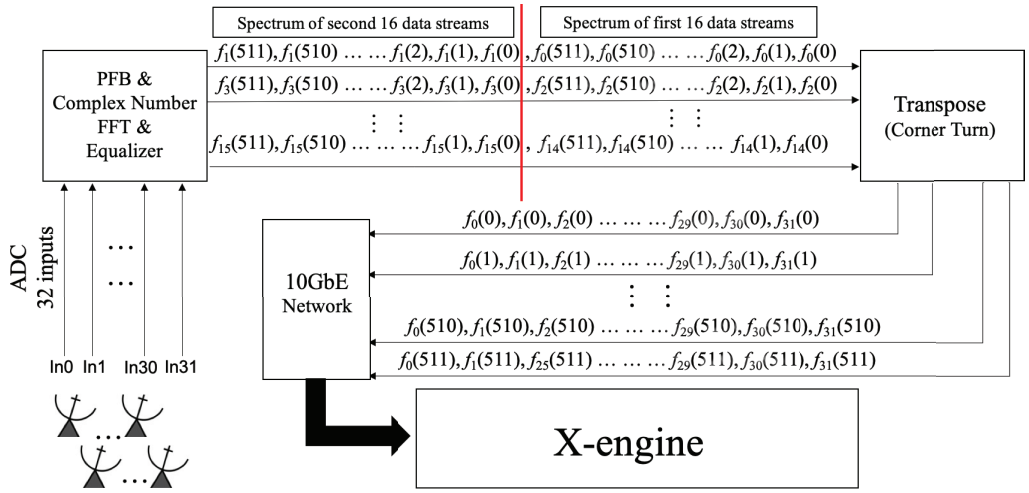

Fig. 4 Data Transpose from the original grouping to the transposed one. $f_i(\nu)$ is the Fourier transform of time stream, subscript i is the antenna input index that ranges from 0 to 31, ν is the frequency channel index that ranges from 0 to 511 for the model with 512 frequency bins in this illustration.

thread to prepare. For each ring buffer, the upstream and downstream threads would write and read a different segment at the same time. The size and number of sub-buffers are selected to avoid data outflow. Hashpipe also provides a status buffer which can extract key-value pairs in each thread. This key-value is updated every running cycle. The status can be viewed using a GUI monitor that has been written in both Ruby and Python. The GPU CMAC is done by `xGPU` which is written by Clark et al. (2013). We also test the GPU performance of `xGPU` code with different numbers of antenna stations. A single GTX690 core will achieve peak performance of 1.2 TFLOPS when the number of antenna stations increases to 96, and stays stable with more inputs. We reach 42% of the official peak performance which is 2.8 TFLOPS for a single core[8].

[8] *https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-690/performance*

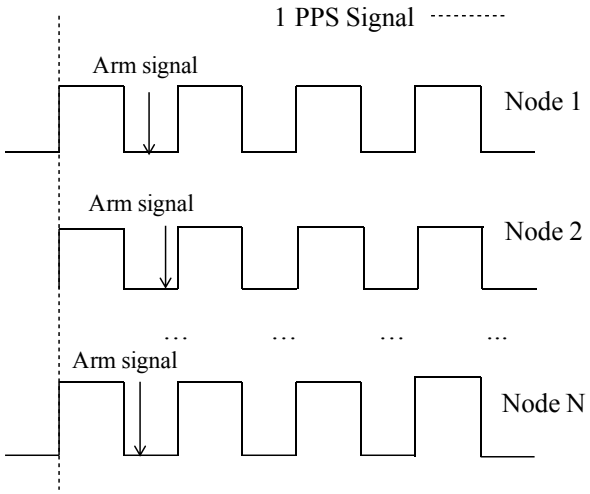

Fig. 5 Synchronization of the ROACH2 nodes. The *dashed lines* indicate the start trigger using the 1PPS signal.

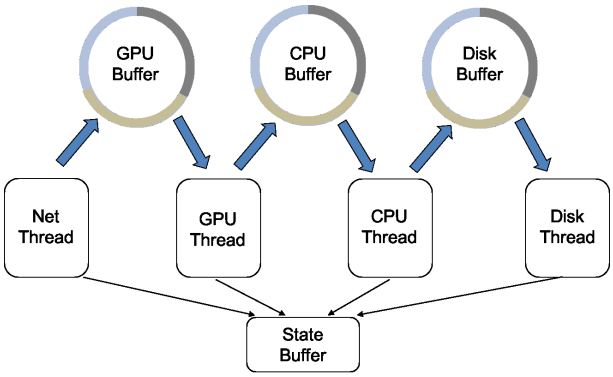

Fig. 6 Hashpipe thread manager diagram.

2.3 Network and Data Packet

The dish pathfinder array with a total of 32 inputs does not require the 10GbE network switch, as its data flow is still within the limit of direct communication through the 10GbE network. However, we designed the system with flexibility and scalability, so that it could be extended to larger arrays. Indeed, we seek a design which can at least handle the 192 inputs of the cylinder array. Our design that includes eight ROACH2s can handle a maximum of 256 inputs. The design is also partially tested, although in the end it is not implemented in full.

The data communication on the 10GbE is handled via the User Datagram Protocol (UDP). The data packet format is shown in Figure 7, which includes the header, content and cyclic redundancy checksum (CRC) for error correction. In the header, after the UDP header, an application header is added to signify the packet properties. The application header is 8-bytes long, which is divided into three parts: 6 bytes for packet counter MCNT, 1 byte for Fid and 1 byte for Xid. The counter MCNT denotes the time sequence of the data, and Hashpipe can detect packet loss by checking if there is any jump in the MCNT sequence. The Fid is the identification of the ROACH2 node from which the packet is produced. The Xid denotes the GPU core where the package is sent, from which one could also know the frequency band in the package.

When considering the correlations, the natural order is to analyze one receiver channel, then run cross-correlation with all channels (including itself as auto-correlation). The cross-correlations are computed in time segments, in which the data are in the format of [time, (fixed) channel, (run) input channel]. This is also shown in Figure 7, where In0 means data from input 0. The data from the F-engine output are in the form of complex numbers with a 4-bit real part and 4-bit imaginary part. Each packet contains $1/32$ of the whole frequency band, i.e., for the 1024 frequency bin model, there are $1024/32 = 32$ frequency channels in each packet. In order to optimize the 10 GbE data transfer, we stack eight time channels from the next time stamp at the

Header			MCNT (6Bytes)	Fid (1Byte)	Xid (1Byte)
Pyload	t_0	ch0	In0, In1, In2, In3, In4, ..., In30, In31		
		⋮	⋮		
		ch31	In0, In1, In2, In3, In4, ..., In30, In31		
	t1	ch0	In0, In1, In2, In3, In4, ..., In30, In31		
		⋮	⋮		
		ch31	In0, In1, In2, In3, In4, ..., In30, In31		
	⋮	⋮	⋮		
CRC			8 Bytes		

Fig. 7 Packet Data Format in our system. For the 512 FFT length model, each time is from ch0 to ch15.

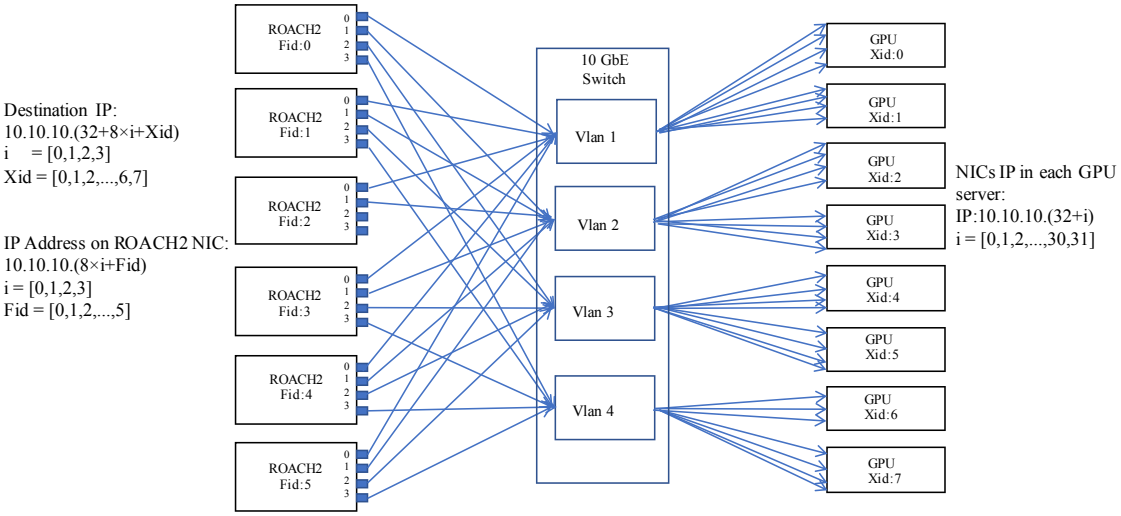

Fig. 8 Network and IP assignment strategy in our system. The 10GbE Switch is divided into four Vlans.

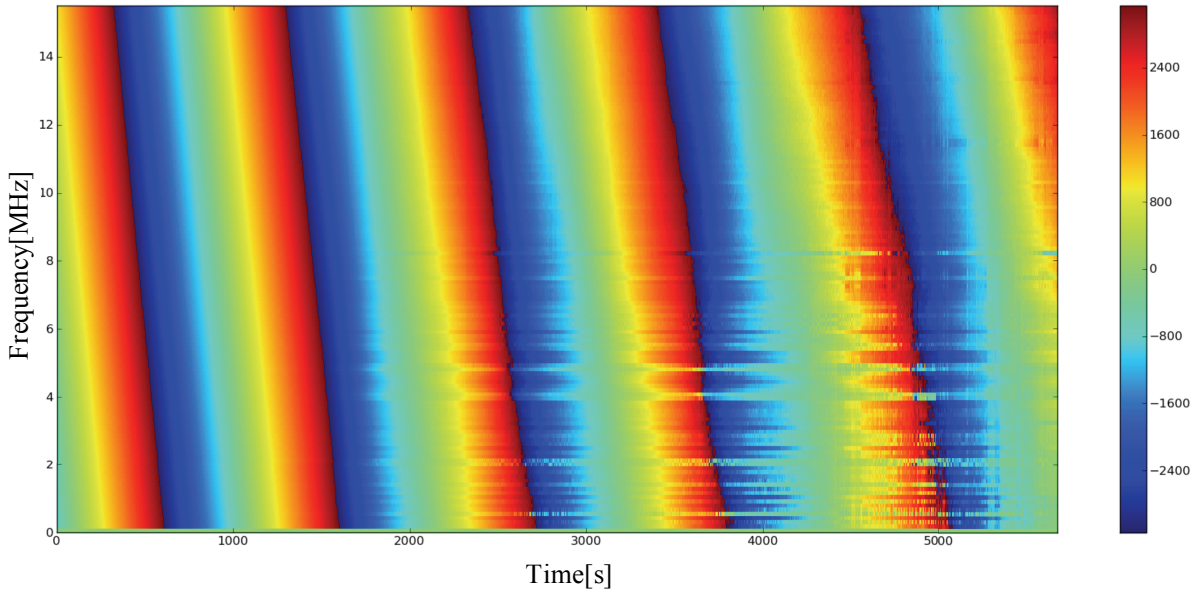

Fig. 9 Phase fringe between two telescope inputs by observing the Sun.

same frequency band in each packet. So, the total size for one output packet from the F-engine is

$$N_{\rm chan} \times N_{\rm input} \times 8 + {\rm Header} + {\rm CRC} = 8208({\rm Bytes}). \quad (5)$$

The data are transferred between the F-engine and X-engine via a 10GbE switch. In our system, each GPU node has four 10GbE ports. As a result, for the 192-inputs we need a total of 4×6 ROACH2 nodes + 4×8 GPU nodes $= 56$ ports on the 10GbE switch. We have a Mellanox$^{\rm TM}$ SX1024 Switch which has 48 ports of 10GbE and 12 ports of 40GbE. Furthermore, the 12 ports of 40 GbE can be split into $12 \times 4 = 48$ 10GbE ports. This would be sufficient for our purpose if we build the correlator for the 192 inputs using this system.

In Figure 8, we show the IP assignment strategy in our network system. The 10 GbE ports on each ROACH2 node are denoted by indexes $i \in [0, 1, 2, 3]$. After the transpose, the output from the ith port of the ROACH2 board contains $\frac{BW}{4} \times i$ to $\frac{BW}{4} \times (i + 1)$ frequency points between eight packets going to different GPU cores. Considering port 0 for example, we will have 0-256 frequency points. As each GPU core processes 32 frequency points, the data output from Port 0 must be sent to eight GPU cores. Given that each GPU node has four GPU cores, only two nodes are required to receive the data, namely, node0 and node1. Thus we can divide the ports of the 10GbE switch into four subnets using a Virtual Local Area Network (VLAN), which will direct the data from port 0 on each ROACH2 board to GPU node0 and node1. The source of the packet received in the GPU core could be located from its Fid in the header, as discussed earlier.

3 TESTS AND EXPERIMENTS

We have conducted a number of simple tests both in laboratory and on site to check the performance of the correlator.

In a frequency chirp test of the F-engine, we feed in a monotone analog signal, then after going through the F-engine we grab the packet using Wireshark$^{\rm TM}$, a software package to view the UDP packet. As expected, the F-engine produced the correct result. We also tested the correlation of two identical signals with a delay. For the correlation of a signal in the form $A{\rm e}^{i(2\pi f t+\phi)}$ and one with time delay τ, the cross-correlation should be

$$V = I_1^* \cdot I_2 = I{\rm e}^{i2\pi f\tau}. \quad (6)$$

When the delay τ is a constant, the phase will be linear with frequency and have a slope of $k = 2\pi\tau$. We use a power splitter to split the signal, and added an $L = 7.5\,{\rm m}$ radio frequency cable in one of the signal paths to simulate a time delay, The result is consistent with a linear slope, with $\tau = L/v$ where the signal speed is found to be $v = 0.7c$, as expected for the signal speed in the RF cable used.

We also did some tests about the linearity of our correlator. By adjusting the input power of the Gaussian white noise signal and the corresponding output, we found good linearity in the range of –12 dBm to 6 dBm input power.

Finally, the correlator was installed on the dish array and we observed bright radio sources such as the Sun, Cygnus A and others. Interference fringes can be clearly seen for the bright sources. Figure 9 shows the visibility of the Sun through two inputs of the dish array during a period of 1.5 hours. The phase of visibility varied with frequency and time caused by Equation (6). Note that the phase disturbance during the last 40 minutes is caused by external RFIs.

4 SUMMARY

A correlator system based on the ROACH2-GPU framework is developed for the Tianlai Dish array with 32 inputs, and the design allows scalable expansion to a larger array, e.g., the 192 input correlator. This correlator design is flexible and scalable. Two FFT lengths, 1024 and 512, are implemented. In the correlator system, different ROACH2 boards and X-engine are running the same F-engine gateware and X-engine software with different parameters. The X-engine consists of the `xGPU` computation core and the Hashpipe data flow management system. The hardware at hand is adequate for the 32 input correlator, but for the 192 input correlator, it can only handle $1/8$ bandwidth right now. The only thing that needs modification in the software is to change some parameters such as frequency band configuration for the F-engine and X-engine. Furthermore, the ROACH2-SWITCH-GPU framework can also be used for different purposes at the same time. The switch has a broadcast mechanism, where we could install a different backend, e.g., a backend to search for fast radio bursts (FRBs) to the system by extending the switch system and adding some FRB nodes. We also did some experiments with our instrument, and the 32-input correlator system worked normally. Given the additional hardware, it can also be extended to the 192-input system, or even to larger systems in the future.

Acknowledgements We thank the support from the CASPER community for making the hardware and software they developed available, and offering helps when

we encountered problems. Chenhui Niu acknowledges the China Scholarship Council for providing support during his visit to the CASPER group in UC Berkeley. The Tianlai project has been supported by the Repair and Procurement Program of the Chinese Academy of Sciences (CAS), the National Natural Science Foundation of China (Grant Nos. 11473044, 11633004, 11773011 and 11761141012), MoST Grants (2016YFE0100300 and 2012AA121701), and the CAS Frontier Science Key Project (QYZDJ-SSW-SLH017).

References

Chen, X. 2012, in International Journal of Modern Physics Conference Series, 12, 256

Clark, M. A., LaPlante, P. C., & Greenhill, L. J. 2013, International Journal of High Performance Computing Applications, 27, 178

Foley, A. R., Alberts, T., Armstrong, R. P., et al. 2016, MNRAS, 460, 1664

Hickish, J., Abdurashidova, Z., Ali, Z., et al. 2016, Journal of Astronomical Instrumentation, 5, 1641001

Parsons, A., Backer, D., Siemion, A., et al. 2008, PASP, 120, 1207

Parsons, A. R., Backer, D. C., Foster, G. S., et al. 2010, AJ, 139, 1468

Price, D. C., Kocz, J., Bailes, M., & Greenhill, L. J. 2016, Journal of Astronomical Instrumentation, 5, 1602002

Proakis, J. G., & Manolakis, D. G. 1996, Digital Signal Processing: Principles, Algorithms and Applications (Upper Saddle River, NJ: Prentice Hall)

So, Hayden Kwok-Hay 2007, Borph: An Operating System for Fpga-based Reconfigurable Computers, PhD Thesis, University of California at Berkeley

Thompson, A. R., Moran, J. M., & Swenson, Jr., G. W. 2001, Interferometry and Synthesis in Radio Astronomy (2nd ed. New York: Wiley, c2001.xxiii)