Correlator built on Roach2-GPU framework for Tianlai Array*

C. H. Niu[1,2,3], Q. X. Wang[2,4], D. Macmahon[3], J. X. Li[2], F. Q. Wu[2†],G. Shippee[3], H. J. Tian[4], X. L. Chen[2], D. Werthimer[3]
[1] *Central China Normal University ,Luoyu Road, Wuhan, China*
[2] *National Astronomy Observatory , Chinese Academy of Sciences*
Datun(A) Road, No.30,Beijing, China
[3] *University of California Berkeley, Campbell Hall 339, Berkeley CA 94720*
[4] *China Three Gorges University,Yichang China 443002*

Correlators play an important role in radio astronomy interferometery. Considering Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuit technologies require longer development periods, we decided to build a correlator based on the ROACH2-GPU framework, as it is more scalable, more flexible, and quicker to deploy. Our correlator includes an F-engine used for Analog to Digital Conversion (ADC) and Polyphase Filter Bank (PFB) computation, built on a ROACH2 board[1](Reconfigurable Open Architecture Computing Hardware); and an X-engine focused mainly on conjugate multiply-accumulate operation (CMAC), built on GPUs (Graphical Processing Units). The correlator is designed for the Tianlai Interferometer Array.

Keywords: Correlator, Tianlai array, ROACH2, GPU

1. Introduction

In radio astronomy, correlators are implemented with the purpose of increasing visibility through the process of correlation. With this visibility information, one can reconstruct the sky map(Thompson et al., 1986). The process of correlation, which involves two or more signals, can be performed in two ways. In XF type correlation, the signal is first convolved and then a Fourier Transform is taken. In FX type correlation, the Fourier Transform is taken first and then the signals are cross multiplied. As multiplying in frequency domain is equivalent to convolving in the time domain, the results from these two types of correlators are identical. With larger telescope arrays, FX correlation is much quicker than XF correlation(Price et al., 2016), thus FX correlation is more proper to use in a large array system.

Tianlai is a radio interferometer array located in Hongliuxia, Xinjiang, in the north-west of China(Chen, 2012). It is designed for BAO observation through observing the 21cm line. The array includes three cylinder reflectors and 16 dish telescopes. Each cylinder is 40m long with 15m wide, and each dish telescope has a 6-meter aperture. The Tianlai Array is shown in Figure 1. Inputs for the Tialai pathfinder experiment are: the cylinder array, with $96 \times 2 = 192$ inputs; and the dish array, with $16 \times 2 = 32$ inputs. The number of inpus will increase in the future, which will cause computation to become even more costly for correlator. By the demand of Tianlai, we plan to build a correlator based on the ROACH2-GPU framework. Processes like ADC, FFT called F-engine are implemented on ROACH2 Board, Cross Multiply Accumulation which is simple in computational complexity but with high computational intensity accomplished in GPU, we called that part X-engine. Data transfer between ROACH2 and GPU server via 10Gbit/s switch.

This correlator has already been partially installed and tested in the Hongliuxia observatory. In this paper we introduce the sketch of the correlator in Section 2; the F-engine in Section 2.1; the X-engine in Section 2.2; the network system in Section 3; some checks and experiments for the correlator in Section 4; and a summary of the work with future prospects in Section 5.

*This paper is found by China Scholarship Council

†Corresponding author:F. Q. Wu. Email: wufq@bao.ac.cn

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

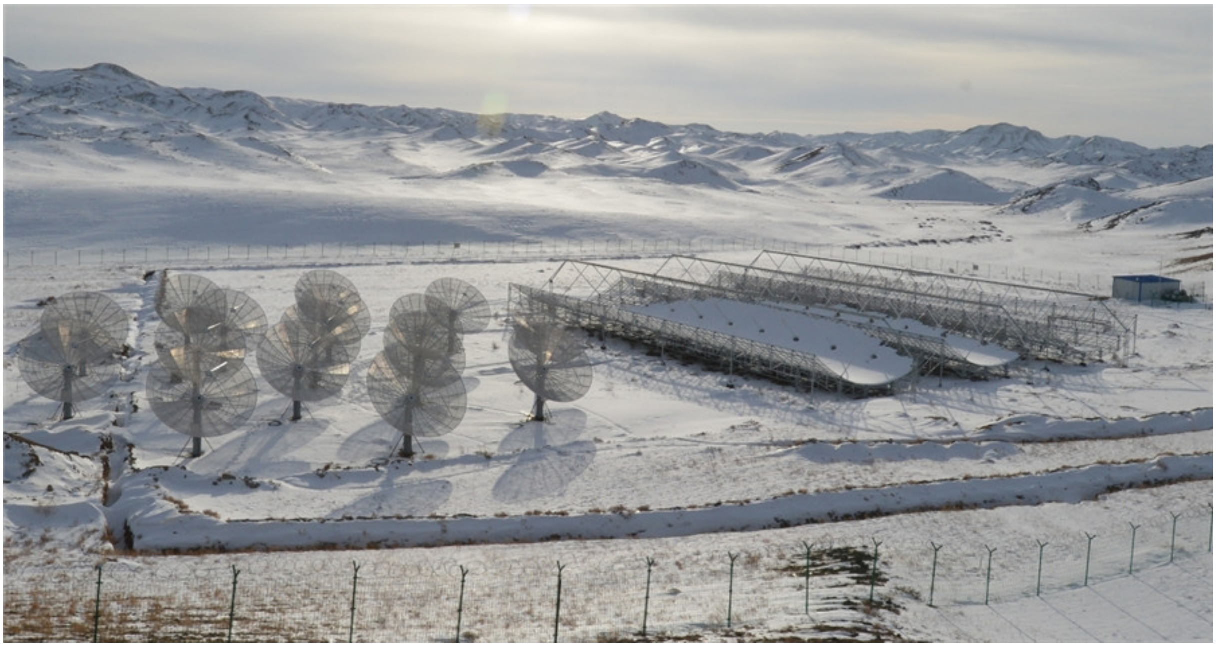


Fig. 1. Tianlai interferometer Array. On the left is the 16 dish telescope. On the right are the three cylinder telescopes.

2. Whole System

The ROACH2-GPU framework for correlators is popular in many radio interferometer arrays, including PAPER (Precision Array for Probing the Epoch of Re-ionization)(Parsons et al., 2010), LEDA (Large Aperture Experiment to Detect the Dark Ages), and others. Our structure builds from the PAPER correlator model (Parsons et al., 2008), and creates a flexible and scalable hybrid correlator system for the Tianlai project.

For the whole correlator system, we assume the F-engine has M ROACH2 nodes, and the X-engine has N GPU nodes. Every ROACH2 node handles m-way analog radio inputs, and after performing a FFT, outputs m spectrums. Additionally, 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. We will present the F-engine model in detail in 2.1. In the GPU node, the F/N frequency points will be distributed to N cores to do conjugate multipliclation accumulation. We will introduce the X-engine in detail at 2.2.

The structure for the Tianlai correlator is shown in Figure 2. The ROACH node can be controlled by a ROACH2-server through the Internet. Via a KATCP (Karoo Array Telescope Control Protocol) protocol, one can load the specified FPGA model to the ROACH2 board. We designed several FPGA models on the CASPER (Collaboration for Astronomy Signal Processing and Electronics Research) platform, such as an ADC raw data sample model which outputs the digital data by sampling the analog signal directly, 32-input and 64-input model which are for the test of small scalar experiment based on PAPER model. We also built two different FFT length (1024 and 512) correlator models for the different frequency resolution requirements of Tianlai. The six ROACH2 boards which are referred to as F-engines, sample 192-input analog signals from the receiver with 8 bit resolution. The sampled data goes through an FFT, Equalizer, Transpose, and 10Gb Ethernet functional block, ultimatley reaching the 10Gb Ethernet switch. Each ROACH2 board has 4 10Gb Ethernet ports connected to a switch, and we assign static MAC addresses to each port on the switch. In addition, we divide the network into several sub-network VLANs. In the X-engine, we implement a distributed computational structure to do the CMAC. We have 8 GPU nodes to deal with different frequency bands, and in each GPU node, we installed two GTX690 GPUs, each equipped with two GPU cores. Thus each GPU node can compute the conjugate multiply accumulation for 128 frequency points from 192 inputs. Hashpipe is used to manage thread and data transfer in each

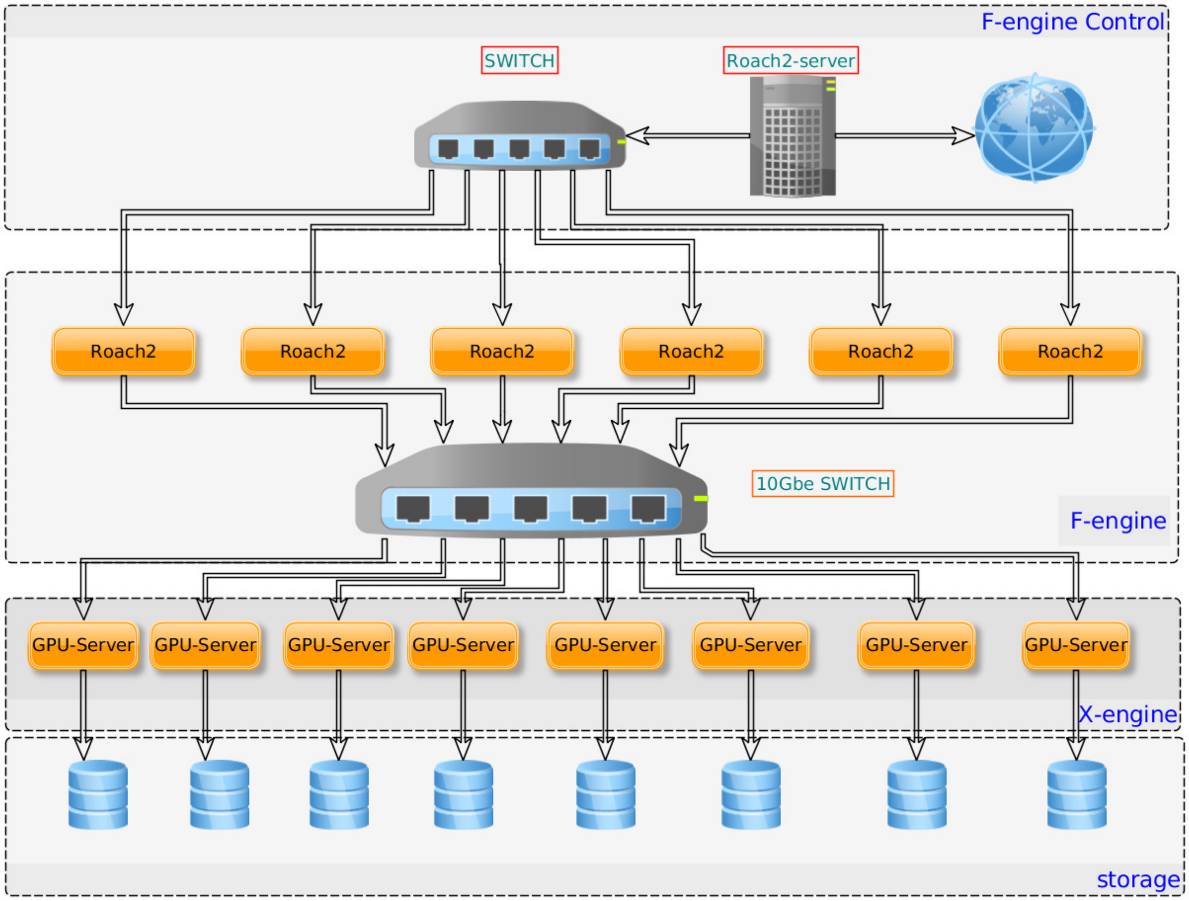


Fig. 2. Structure of Tianlai Correlator.

GPU node.

The required number of ROACH2 boards is dependent on the number of antenna inputs. We have 2 ADC daughter boards installed in every ROACH2 board, while each ADC has 16 inputs. If we only consider the cylinder telescopes, we have 192 inputs, so 6 ROACH2 boards are needed. The number of GPU nodes is determined by the computational intensity. After several tests, we found the peak computational performance for each GPU core is 1.2 TFLOPS, giving us a total of 4.8 TFLOPS peak computational performance for each GPU node with 4 cores. The total computational intensity for Tianlai pathfinder experiment is thus 35.6 TFLOPS. Therefore we presume 8 GPU nodes will meet the demand for the present system.

2.1. *F-engine*

The F-engine model is built on the CASPER platform. This platform allows the user to focus on the design and function of the model by introducing a graphical user interface which abstracts away from FPGA code. Most ROACH2 hardware already has its corresponding GUI yellow block implemented by CASPER developers[2]. In our correlator, the F-engine is expected to have the ability of ADC, FFT, frequency dependent data bit choosing, and the sending of different frequency bands of data from all antenna inputs to the corresponding GPU nodes. We can realize these functions by dragging the corresponding blocks from the CAPSER tools and considering the data interface, delay and data flow issue.

[2] https://casper.berkeley.edu/

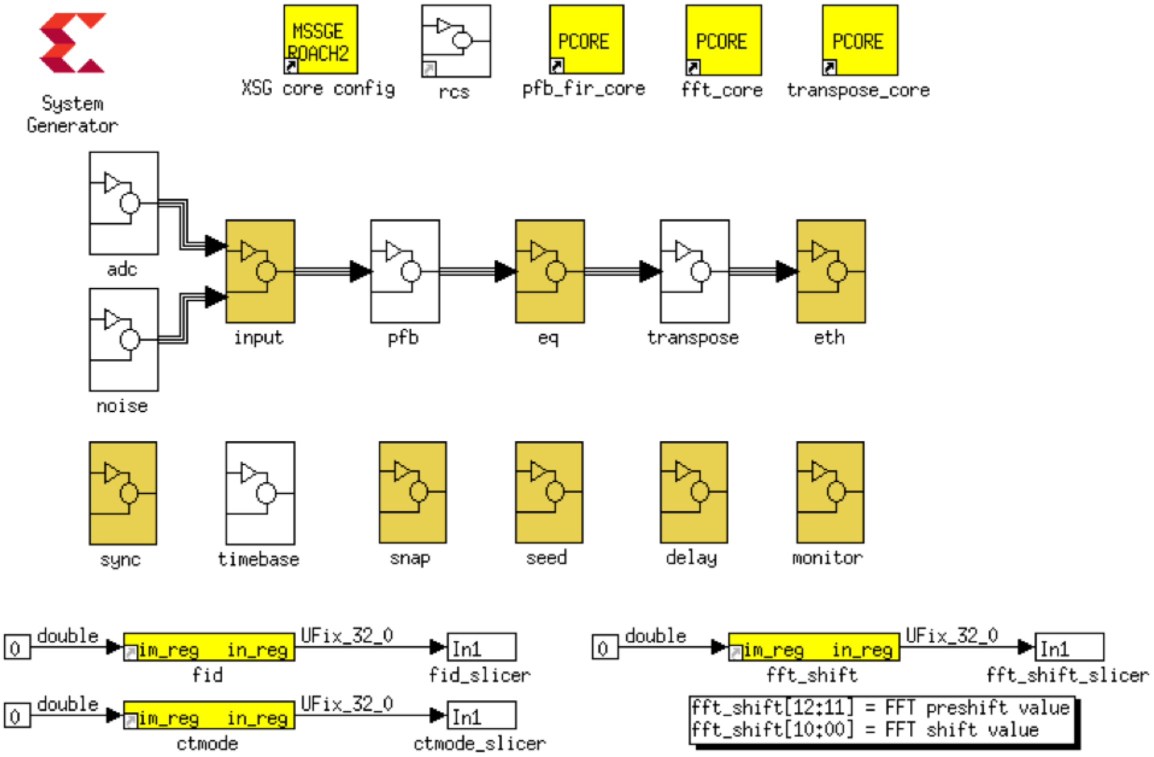


Fig. 3. F-engine model modified from PAPER correlator

2.1.1. *Function Blocks*

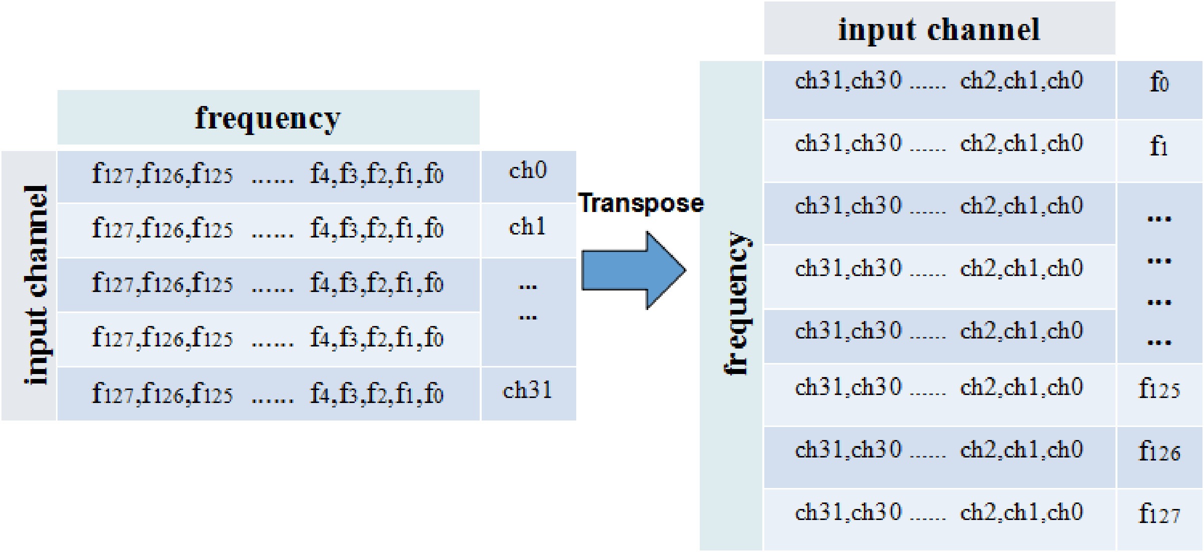
The demand of the Tianlai correlator is similar to that of the PAPER correlator. As a result, our F-engine model is based on the PAPER F-engine model that is designed by David Macmahon (David M. , 2014). The F-engine sketch is showed in Figure 3. As a radio-frequency interface, 2 16-input ADC daughter boards are mounted on a ROACH2 board. Each input is sampled with 8-bit resolution and then transferred to a PFB block, where the FFT is processed using a polyphase filter minimize energy leakage.

To reduce the data exchange amount in the switch, we divide the 18-bit data into 4 bits chunks before sending it to the Ethernet. Because different receivers have different frequency responses, we use an Equalizer block to take the data bits at the correct position corresponding to the correct frequency channels. Each frequency channel of each input will have its own adjustment factor. After the Equalizer block, data flow for each input is divided into a 4-bit real and a 4-bit imaginary component.

2.1.2. *Transpose*

For the digital data after the ADC and FFT, various frequency data of a particular signal channel and time is arranged in a successive order in the data flow. However, frequency channels must be distributed among a number of servers simultaneously, so suffering and making a transposition are mandatory before sending X engine with the frequency band data from all signal channels for a particalar time. This need convert data shape from $[inputs, frequency]$ to $[frequency, inputs]$, like figure 4 shows. This operation could be achieved in a dual-port RAM through writing, reading from different address,alternative destination IP and changing header. Figure 5 shows the overview of block design, if one output reading and writing address of dual-port RAM, we could see how exactly the reading and writing address change under flag signal from '*tid*','*fid*' etc.

The ethernet block is the last block of the F-engine model. The only difference between the models used on the ROACH2 nodes is the IP and MAC configurations for the Ethernet block. One ROACH2 node



input channel	frequency	
	f_{127},f_{126},f_{125} ...... f_4,f_3,f_2,f_1,f_0	ch0
	f_{127},f_{126},f_{125} ...... f_4,f_3,f_2,f_1,f_0	ch1
	f_{127},f_{126},f_{125} ...... f_4,f_3,f_2,f_1,f_0	...
	f_{127},f_{126},f_{125} ...... f_4,f_3,f_2,f_1,f_0	...
	f_{127},f_{126},f_{125} ...... f_4,f_3,f_2,f_1,f_0	ch31

frequency	input channel	
	ch31,ch30 ...... ch2,ch1,ch0	f_0
	ch31,ch30 ...... ch2,ch1,ch0	f_1
	ch31,ch30 ...... ch2,ch1,ch0	...
	ch31,ch30 ...... ch2,ch1,ch0	...
	ch31,ch30 ...... ch2,ch1,ch0	...
	ch31,ch30 ...... ch2,ch1,ch0	f_{125}
	ch31,ch30 ...... ch2,ch1,ch0	f_{126}
	ch31,ch30 ...... ch2,ch1,ch0	f_{127}

Fig. 4. Dataflow conversion of the Transpose. Left: Original data flow. Right: Transposed data flow.

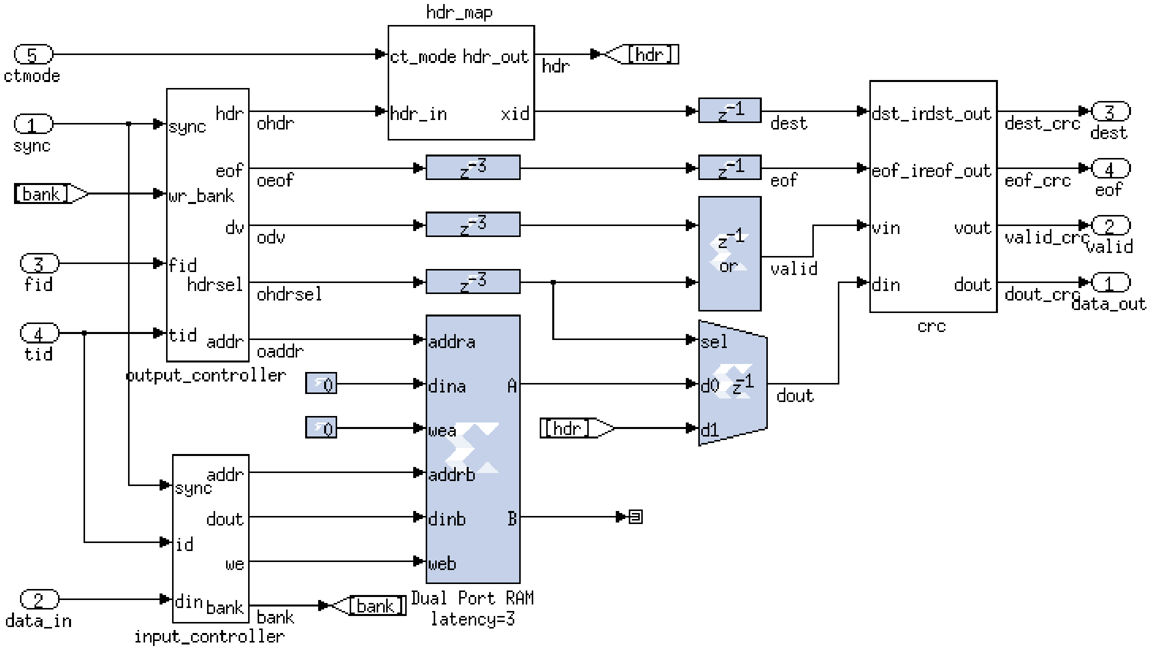


Fig. 5. Structure of the second class sub-module in Transpose. One can modify the dest IP and packet size in this model.

has four 10Gbps ports. Without using the loopback mode used in the PAPER model, all of the Ethernet ports only send and do not receive packets, and each of them will have a different IP information decided by '*tid*'. Detailed strategy for IP assignment is described in sec 3.2

When the F-engine is implemented, one can simulate the design on Matlab with CASPER tools. After de-bugging the model from the simulation, a 'bof' file is generated which can be uploaded to the FPGA as firmware.

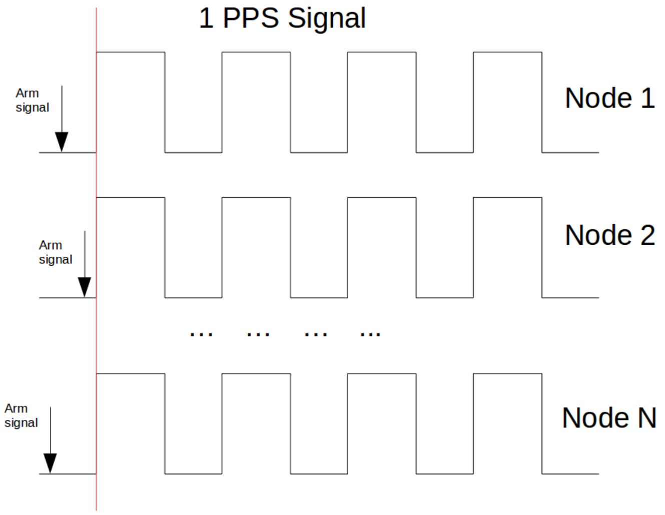


Fig. 6. Synchronizing N ROACH2 nodes. The red lines indicate the start trigger using 1PPS signal

2.1.3. *Control System*

Each ROACH2 node is installed with a Power PC that runs a linux system. The ROACH2 server has a service called Network File System (NFS), which allows for the sharing of files, such as 'bof' files, with the Power PC on the ROACH2 node. Any time the ROACH2 boots up, it will load the linux kernel from the ROACH2 server or from the on-board memory chip. These booting methods are referred to as 'Net boot' or 'Solo boot', respectively. When the whole system is running under Net boot, all ROACH2 boards load the 'bof' files from the NFS on ROACH2 server.

There are sync ports in all functional blocks of a single ROACH2 node. The sync port ensures every component inside the model is ready to run. However, the clocks are not synchronous between different ROACH2 nodes. In this F-engine model, a sync model is used to synchronize all ROACH2 nodes to the same clock cycle by sending a 1PPS signal to each node. After all the parameters, such as the IP and MAC addressess are set, a sync signal called the 'Arm signal' is sent to each node. Then, all the ROACH2 nodes wait for a 1PPS signal to trigger the system. Arm signals at the different nodes are not required to be synchronized, but the 1PPS signals are made simultaneous by choosing the same length of cable for each signal. After this operation, all the ROACH2 nodes will be synchronized. A diagram is showed in Figure 6.

2.2. ***X-engine***

The X-engine in this system is mainly responsible for packet receiving, data distribution, Conjugate Multiply Accumulation and data storage. For one GPU node, there are 2 GTX690 cards, each containing 2 GPU cores, 2 CPUs and four 10 Gbits NICs. These four GPU cores work separately, each processing the CMAC of 32 frequency points. The CMAC process uses the 'xGPU' package provided by Michael Clark etc. (Clark et al., 2013). The 'xGPU' code is written in CUDA-C and is optimized on GPU memory resources by specific thread tasks. It was originally designed to solve large array signal correlation processes.

2.2.1. *Thread Manager:Hashpipe*

Data management in each GPU node is done via Hashpipe[3], which was developed by David MacMahon, Jeff Cobb et al. Hashpipe specializes in communication between threads. Moreover, it can also map CPU memory to GPU memory. In thread collaboration, data is stored in a ring buffer between different threads. When the ring buffer is filled by the former thread, a warning message will be sent to the next thread.

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

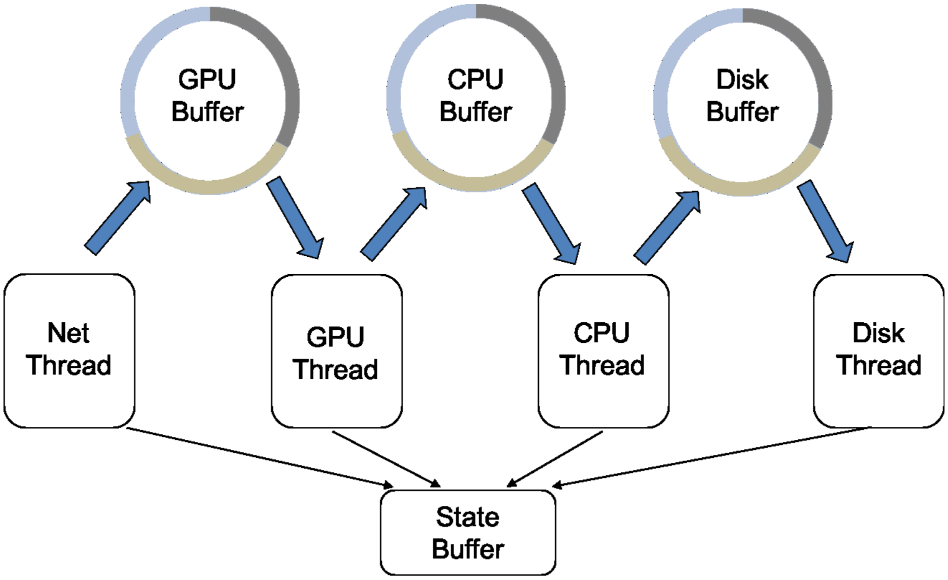


Fig. 7. Hashpipe thread manager structure.

The Hashpipe sketch is shown in Figure 7. We open 4 threads and 3 ring buffers on each GPU node, and each ring buffer is divided into 3 memory zones. The net thread is responsible for receiving data from the ROACH2 nodes; reassembling the data sequence as required by the xGPU code; and storing the data in the GPU ring buffer. When the ring buffer is full, the GPU thread starts to process the data in the GPU ring buffer. After the GPU is finished with the CMAC operation, it outputs the data to the CPU ring buffer and cleans GPU data buffer. The CPU thread converts the data into the form that will ultimately be stored on disk, and the disk thread writes the data to storage. Hashpipe also provides a status buffer which can extract key-value pairs in each thread. This key-value will update every running cycle. The status can be viewed using GUI monitor that has been written in both Ruby and Python.

3. Local Network Framework

3.1. *Packet Format*

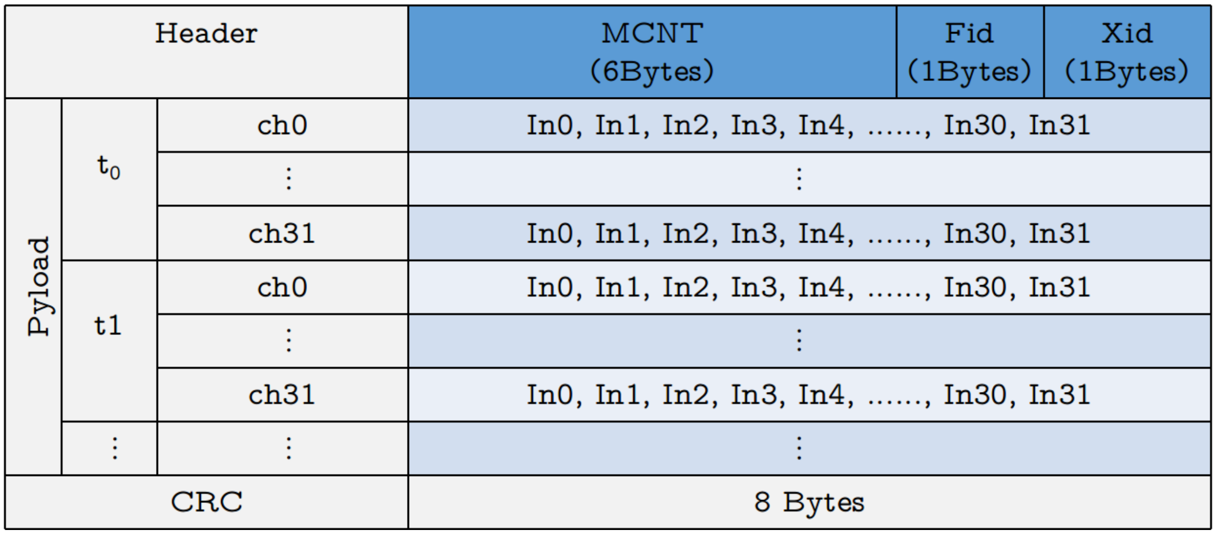
The packet format for the system can be found in Figure 8. The header is 8 Bytes which has been divided into 6 Bytes for 'Mcnt', 1 byte for 'Fid', and 1 byte for 'Xid'. 'Mcnt' is used to count the number of packets. For the antenna data received at the same time, 'Mcnt' should be the same for all different ROACH2 nodes. Hashpipe could evaluate the packet loss from the 'Mcnt' sequence. 'Fid' stands for F-engine ID and gives information about which ROACH2 node the packet is from. Every GPU node has four GPU cores, each will only process $1/N_{GPU}$ total bandwidth, where N_{GPU} stands for total number of GPU cores in the system. 'Xid' denotes the GPU core where the package is sent, as each core only process data from a frequency subband, so, from this 'Xid', one could also know the freuqency band in the package.

The payload's data shape is formatted as $[time, channel, input]$, In Figure 8, 'In0' means data from input 0. The data from the F-engine output is a complex number with a 4-bit real part and 4-bit imaginary part. Each packet contains $1/32$ of whole frquency band, i.e. there are $1024/32 = 32$ frequency channels in each packet. In order to optimize the 10 Gbe data transfer, we add 8 time channels from the next time stamp at same frequency band to each packet. So the total size for one output packet from F-engine is:

$$N_{chan} \times N_{input} \times 8 + Header + CRC = 8208(Bytes). \quad (1)$$

3.2. *IP Assignment Strategy*

The data flow between the F-engine and the X-engine is connected by 10 GbE Switch. In our system, every ROACH2 node has four 10GbE ports, and each GPU node also has four 10 GbE ports. As a result, we



Header			MCNT (6Bytes)	Fid (1Bytes)	Xid (1Bytes)
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. 8. Packet Data Format in our system. For 512 FFT length model, each time is from *ch*0 to *ch*15

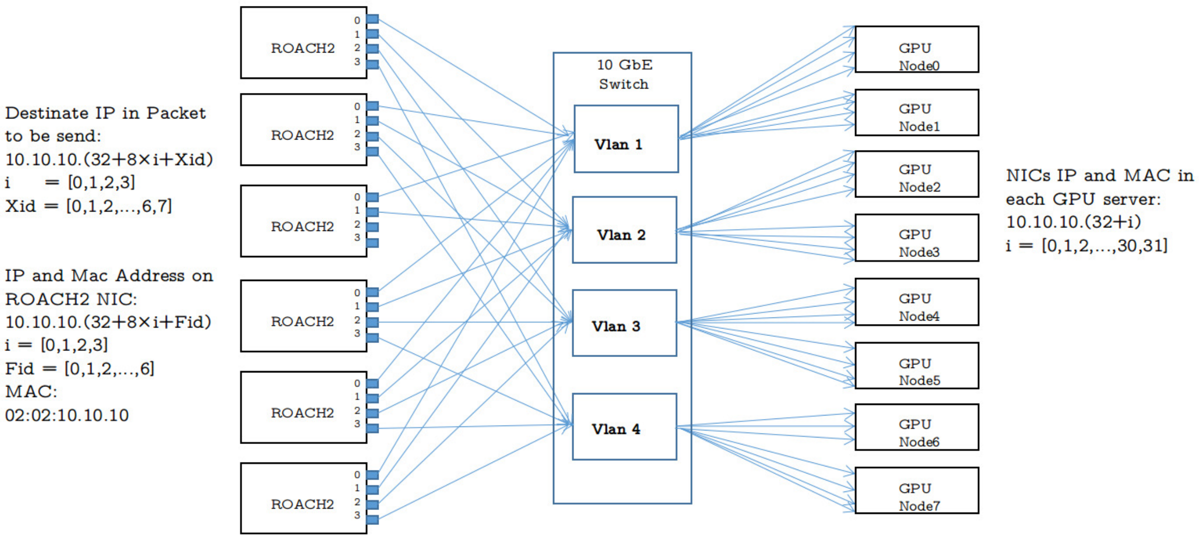


Fig. 9. Network and IP strategy in our system. We divide the 10 Gbe Switch into four Vlans to avoid packets loss

need a total of 4 × 6 Roach2 nodes + 4 × 8 GPU nodes = 56 ports on 10GbE switch. To satisfy this, we use the Mellanox SX1024 Switch which has 48 ports of 10GbE and 12 ports of 40GbE. Furthermore, 12 ports of 40 GbE can be split into 12 × 4 = 48 ten GbE ports.

In Fig.9, we show the IP strategy in our network system. 10 GbE ports on each ROACH2 node are denoted by index $i \in [0, 1, 2, 3]$. After the transpose function model, the output from the ith port of ROACH2 board will contain $\frac{BW}{4} \times i$ to $\frac{BW}{4} \times (i+1)$ frequency points between 8 packets going to different GPU cores. Taking 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 8 GPU cores. Given that each GPU node has 4 GPU cores, only two nodes are required to receive the data, namely, node0 and node1. Thus we can devide the ports of 10GbE switch into 4 subnets using Vlan, which will guide the data from port 0 of each ROACH2 board to GPU node0 and node1. The packet received in the GPU core could be recognized the source of the data from the 'Fid' in Header, as introduced in section 3.1.

4. Experiment

We have performed several tests on this system. In our tests, we had 6 ROACH2 nodes but only 1 GPU node. Thus we could only handle 1/8th of the whole frequency band from all inputs.

4.1. *ADC and F-engine experiment*

To test the ADC part and also for the convenience of cross checking the later part, we design a ADC sampling model on the ROACH2 which packaged raw ADC samples directly and then send them to the GPU server without doing the FFT, Equalizer, Transpose etc. In the experiment, we input a 20MHz single tone analog signal into the ADC, then, receive data from the GPU node, finally, we make a spectrum analysis of the output data. The result was a sine wave with a frequency of 20.0001 MHz, which is pretty consistent with the input signal.

We also did a frequency chirp test in the F-engine. We input a single tone analog signal, after the F-engine, grab the packet using Wireshark instead of Hashpipe in the GPU node. Finally, as expected, we found the data point in right place.

4.2. *Whole system test*

In the correlation of two inputs, phase of visibility is dependent upon the signal time delay τ between two antennas. If the input signal behaves like $Ae^{i(2\pi ft+\phi)}$, two antennas with delay τ between them will have different inputs: $Ae^{i(2\pi ft+\phi)}$ and $Ae^{i(2\pi f(t+\tau)+\phi)}$. After correlation, visibility is:

$$V = I_1^* \cdot I_2 = Ae^{-i(2\pi ft+\phi)} \cdot Ae^{i(2\pi f(t+\tau)+\phi)} = Ie^{i2\pi f\tau} \tag{2}$$

The phase of visibility is $\Phi = (2\pi\tau)f$, and varies from $[-\pi, \pi]$. When the delay τ is a constant, phase will be linear with frequency, and will have a slope of $k = 2\pi\tau$. We did two experiments to simulate the effects of a time delay. In the first experiment, we generated a time delay in the FPGA design, and then added Gaussian White Noise (GWN) to the two inputs. The time delay we added in the FPGA design is 2.5×10^{-8}s, giving a theoretical slope of $k = 1.571 \times 10^{-7}$rad/Hz. The k_c we found from visibility data of correlator was $k_c = 1.561 \times 10^{-7}$ rad/Hz, which only has a tiny difference of 0.6% from the theoretical expectation. Another experiment we did was using a $7.5m$ radio cable added in one input signal path to simulate a time delay due to the distance of signal travel. After fitting of the slope, we found the speed the radio signal travel through the wire is 0.7c, where c is the speed of light, which is quite reasonable with the performance of RF cable.

4.3. *Observation Test*

We have also used this correlator to observe the Sun, Cygnus A, and other radio sources. It was able to detect the interference fringe of sources. Additionally, in a collaborative observation with another correlator built by Institute of Automation, Chinese Academy of Sciences, we found the visibilities of these two correlators were nearly identical. Figure 10 shows the observation from sun.

5. Summary

The correlator system we designed based on the ROACH2-GPU framework is more flexible. We designed 1024 and 512 FFT length models for different frequency resolutions, and an ADC sampling model to store the raw data sampled by ADC. In the correlator, the F-engine gateware and X-engine sofware are almost the same in the different ROACH2 nodes or X-engine nodes. X-engine essentially consists of 'xGPU' computation core and 'Hashpipe' data flow manage system. In the future, we can enlarge the antenna inputs simply by adding ROACH2 nodes, switch and GPU codes. The only thing need to be modified in the software is to change some parameters, for example, IP, Fid, Xid, and frequency band configuration for Fengine and Xengine. Furthermore, ROACH2-SWITCH-GPU framework can also be used for different purposes at the same time. Switch has a broadcast mechanism, we could extend other backend like the FRB backend to the system, simply by extending the switch system and adding some FRB nodes. We also

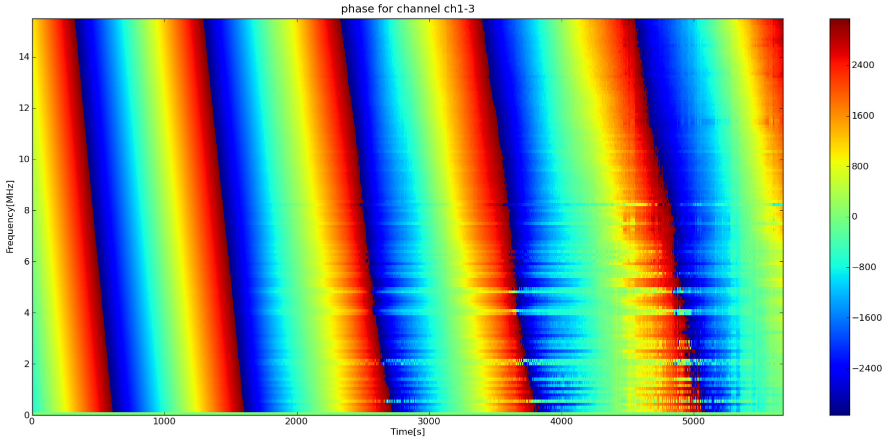


Fig. 10. Phase fringe between two telescope inputs by observing of sun.

did some experiments with our instrument, and the correlator system is working normally. Actually this system is not only useful for Tianlai-like interferometer system, but also easy to be adding more functions in other telescopes with different demands.

References

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

Thompson, A. R., Moran, J. M., & Swenson, G. W. 1986, New York, Wiley-Interscience, 1986, 554 p.,

David Macmahon, 2014, “PAPER Correlator Manifest,” https://casper.berkeley.edu/wiki/PAPER_Correlator_Manifest.

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

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

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

Clark, M. A., LaPlante, P. C., & Greenhill, L. J. 2013, International Journal of High Performance Computing Applications, Vol. 27, p. 178-192, 27, 178