Postprint: Construction of a Digital Terminal Experimental Platform Based on ROACH2

Authors: Zhang Meng, Zhang Hailong, Wang Jie, Li Jian, Tohti Nur

Date: 2019-12-04T00:00:00+00:00

Abstract

To address the real-time processing requirements of astronomical signals, a radio astronomy digital backend experimental platform based on ROACH2 was established. Raw experimental data were acquired through simulations utilizing development environments such as MATLAB and Xilinx; signal control and preprocessing were implemented using the CASPER hardware platform, with data packets transmitted to a computing server via high-speed Ethernet for post-processing. The constructed experimental platform achieves the complete workflow of simulation, compilation, and operation, providing a sound experimental environment for research on real-time astronomical signal processing.

Full Text

Construction of a Digital Backend Experimental Platform Based on ROACH2

Authors: ZHANG Meng[1],[2], ZHANG HaiLong[1],[2],[3]*, WANG Jie[1], LI Jian[1], Tohtonur[1]

Affiliations:

[1] Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi, Xinjiang 830011, China

[2] University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

To meet the real-time processing requirements for astronomical signals, we have constructed a radio astronomy digital backend experimental platform based on ROACH2. Utilizing development environments such as MATLAB and Xilinx for simulation, we obtained raw experimental data. The CASPER hardware

platform was employed to implement signal control and preprocessing, with data packets transmitted via high-speed Ethernet to computing servers for post-processing. The established experimental platform has realized the complete workflow of simulation, compilation, and execution, providing a robust experimental environment for research on real-time astronomical signal processing.

Keywords: Digital Backend; ROACH2; Experimental Platform; CASPER

The adoption of multi-beam and PAF (Phased Array Feed) receiving technologies has led to exponential growth in the volume of astronomical observation data acquired by radio telescopes, necessitating real-time processing to overcome bottlenecks in transmission and storage speeds. The application of high-speed broadband digital backends for real-time signal processing can satisfy future demands for massive data real-time processing. Current digital backend systems primarily consist of programmable hardware platforms for astronomical signal control and preprocessing, along with high-performance computing systems for control and complex processing.

The CASPER (Collaboration for Astronomy Signal Processing and Electronics Research) hardware platform has been widely adopted by many large radio telescopes for their digital backend systems due to its free, open-source, and reusable nature. The GBT (Green Bank Telescope) pulsar backend system GUPPI (Green Bank Ultimate Pulsar Processing Instrument), the PUPPI (The Puerto Rico Ultimate Pulsar Processing Instrument) used at Arecibo, and the CASPSR (CASPER Parkes Swinburne Recorder) at Parkes all employ CASPER' s early hardware systems IBOB (Interconnect Break-out Board) and BEE2 (Berkeley Emulation Engine). The VEGAS system at GBT, the PSRIX backend at Effelsberg, and the BPSR (Berkeley Parkes Swinburne Recorder) at Parkes utilize the first-generation ROACH system, employing multiple ROACH boards for channelization and consolidating signals from IBOB and BEE2 platforms onto a single board.

In China, both the FAST telescope in Guizhou and the Tianma telescope in Shanghai have used ROACH2 to construct their backend systems for frequency channelization and high-speed streaming data processing.

To further investigate existing digital backend systems and achieve real-time processing of high-speed astronomical signals, it is essential to build a stable experimental platform that provides a sound experimental environment for research. Leveraging the versatility and open-source nature of CASPER' s digital backend hardware and software platforms, this paper implements the construction of an experimental platform based on CASPER hardware and software, and conducts systematic testing of the established environment.

1 Hardware Platform

Most devices developed by the CASPER group are based on Xilinx FPGA chips. A typical CASPER signal processing platform usually connects to one or two ADCs for signal acquisition and preliminary preprocessing, then transmits data via network links to other FPGA boards or CPUs/GPUs for complex signal processing. Currently available CASPER-supported FPGA boards include IBOB, the ROACH series, SKARAB, and the SNAP series. For this experimental platform construction, we selected the stable and widely used CASPER hardware platform ROACH2, with the physical laboratory unit shown in [Figure 1: see original paper].

The ROACH (Reconfigurable Open Architecture Computing Hardware) series boards represent a reconfigurable open architecture computing hardware framework, serving as standalone programmable platforms for radio astronomy signal processing. ROACH2 is the latest version in the ROACH series, utilizing Xilinx Virtex-6 series FPGAs. It retains the advantages of ROACH while improving overall performance in processing capability, I/O throughput, and memory bandwidth. It employs the same PowerPC 440EPx as ROACH but adds a JTAG (Joint Test Action Group) interface for on-chip testing.

As illustrated in [Figure 2: see original paper], the ROACH2 hardware platform is structurally divided into five independent subsystems: the FPGA subsystem, processor subsystem, board management subsystem, test and debug subsystem, and signal storage subsystem. The core component is the Virtex-6 series XC6VSX475T FPGA for signal processing. The PowerPC 440EPx independent processor runs a streamlined Linux system to provide control functions. Four 36×2M QDR2 (Quad Data Rate 2) SRAMs and a single 72-bit DDR3 RDIMM (Registered Dual In-line Memory Module) slot are connected to the FPGA for data computation and storage. Two ZDOK interfaces connect to ADCs and other devices, while four SFP+ (Quad SFP+ Mezzanine board) ports support up to 8×10GbE data links for high-speed output to downstream systems.

2 Software Environment

The CASPER toolkit provides a design environment for CASPER hardware platforms. The main components are collectively known as MSSGE (MATLAB/Simulink/System Generator/EDK). The MSSGE toolkit is a software development platform for CASPER hardware platforms that integrates these components into a unified design and development environment. This development environment was originally designed by the Berkeley Wireless Research Center as the BEE_XPS toolkit for BEE hardware platforms and was later extended to be universally applicable to all CASPER hardware platforms. The environment offers a graphical design interface using Xilinx System Generator tools, enabling an integrated design, compilation, and implementation process.

Within the MSSGE toolkit components, MATLAB provides a scriptable backend for Simulink, with all scripts implementable through MATLAB. Simulink

serves both as a schematic drawing tool for CASPER ROACH2 system models and as a high-level simulation environment for signal processing system design. System Generator is used for FPGA programming implementation and represents Xilinx' s system-level design tool, which can be understood as a conversion utility that automatically maps abstract modules designed in Simulink into reliable hardware implementations. During compilation, System Generator converts Simulink schematics into HDL code (VHDL or Verilog), significantly simplifying the development process for digital signal processing algorithms. The EDK (Embedded Development Kit) component is used for embedded development on FPGAs. CASPER hardware compilation is based on Xilinx EDK, which can compile the hardware description language code generated in the previous step into bitstream files and convert them into operating system-executable .bof files. EDK is an extended functional toolset for ISE.

This experimental platform software can be deployed on Ubuntu x64, RHEL, or CentOS x64 systems. Practical testing has revealed version compatibility differences between MATLAB service packs and Xilinx XSG/System Generator, requiring careful selection of corresponding MATLAB and Xilinx versions along with the appropriate MSSGE libraries.

This paper demonstrates the process using CentOS 7 with MATLAB 2012b and Xilinx 14.7 as an example. The environment setup flow is shown in [Figure 3: see original paper]. On CentOS 7, install Xilinx 14.7 first, followed by MATLAB 2012b. The toolkit requires two libraries: the CASPER library for DSP blocks and the BEE XPS library for hardware support blocks. These two libraries are now bundled in a single mlib_devel directory that users can download from GitHub. It is important to note that during development environment setup, consistency must be maintained between software versions, installation paths, and library file storage paths. Xilinx 14.7 removed certain modules that ROACH2 depends on. These modules, called pcores (peripheral cores), exist in EDK and must be copied to the XPS_ROACH2_base/pcores folder for updating. After downloading and installing, edit the default path in startsg to reflect the actual file storage path. Once the changes are completed, running startsg will launch MATLAB. After the Xilinx and CASPER libraries are successfully imported, design work can commence.

Following completion of the software and hardware system construction, it is necessary to validate the platform' s functionality through practical examples. This paper tests the constructed platform using examples provided by CASPER.

The example testing implements the complete processing flow from platform software design to hardware execution, as shown in [Figure 4: see original paper]. Design and simulation are performed using Simulink in MATLAB, files are compiled using casper_xps, and after compilation generates executable files, the .bof executable files are uploaded to the corresponding folder on the ROACH2 platform and remotely controlled to execute on ROACH2 via Python scripts.

Design begins with graphical modeling in the Simulink compilation interface.

Create a new simulation diagram, add the Xilinx System Generator and XSG core configuration blocks, then select required modules from the library basic elements according to the described design architecture, set parameters, and construct the complete top-level design structure. This paper uses the establishment of a broadband spectrometer as an example, with the overall graphical program shown in [Figure 5: see original paper]. Three main libraries are used when building architecture diagrams in Simulink: the CASPER XPS library (yellow blocks encapsulating hardware interfaces, etc.), the CASPER DSP library (green blocks implementing DSP functions, etc.), and the Xilinx library (blue blocks providing low-level functions such as multiplexing, delay, and addition).

After completing the design in Simulink, simulation can be performed directly within the environment to ensure no errors exist before compilation. Click the execute button in the top toolbar to run the simulation. If errors occur, a diagnostic window will pop up for individual handling. Some designs can also verify correctness by checking whether the oscilloscope display matches the settings.

Once the design functionality is verified, it must be compiled into code recognizable by the FPGA. Enter the command `casper_xps` in MATLAB to launch the compiler module, which will display the compilation interface shown in [Figure 6: see original paper]. Keep all options at their default settings, ensure the listed design is the one intended for compilation, then click `gcs` to return to the system path of the most recently selected module, and finally click "Run XPS" to begin compilation.

After a considerable waiting period, if compilation completes successfully, a dialog box indicating successful compilation will appear, as shown in [Figure 7: see original paper]. Following successful compilation, multiple files will appear in the right-hand folder directory panel. Within the bit_files folder, binary files with the .bof suffix can be found. The *.bof file is the executable file that can run on ROACH2.

The processes of design, simulation, execution, and compilation are all implemented on CASPER' s software development platform. To execute the next steps on the ROACH2 hardware platform, the compiled .bof file must be uploaded to the corresponding folder on the ROACH2 platform. After multiple attempts, NFS (Network File System) is recommended for .bof file upload—remotely mounting the relevant directory from another server to the corresponding directory on the ROACH2 platform. This allows .bof file upload simply by copying the file to the server. The compiled executable file can then be remotely controlled to run on ROACH2, completing configuration and communication while achieving display output.

This paper uses the establishment of a 2048-channel spectrometer as an example. A BPSG4 signal generator is used to produce an 800 MHz clock signal, with the clock source connected to clk_i on the ADC and the ROACH2 clock frequency

set to 200 MHz. The input signal is digitized by the ADC to produce four parallel time sequences. Python is used to control the spectrometer' s execution on ROACH2, converting it to frequency domain signals for final spectral output.

During the experiment, a broadband signal is first generated using a noise source, then a bandpass filter with 98-122 MHz bandwidth is used to select frequency components within the specified range. The signal spectrum after 60 integrations is shown in [Figure 8: see original paper].

The constructed experimental platform can achieve data transmission between hardware devices via 10GbE SFP+ ports, read preprocessed data through the PPC' s 1GbE port, and transmit ADC-digitized signals to the FPGA through ZDOK ports. Modules defined in HDL can be embedded and instantiated within Simulink designs to create new Xilinx and CASPER modules. Using modules from Simulink and CASPER libraries, new hardware designs can be constructed to implement relevant astronomical signal processing functions. Testing demonstrates that the experimental platform built in this paper meets the algorithm testing requirements for radio astronomy digital backends.

Due to continuous updates to operating systems and application software with mutual incompatibility, a series of version mismatch issues were encountered during the experiment. As hardware and software platforms evolve, design modules are constantly being revised, replaced, deprecated, or added, affecting calls to different version libraries. During construction, error messages were encountered regarding outdated module versions, backward EDK versions, and non-existent parameters. These issues were resolved by attempting to replace software packages with different versions. CASPER tutorial documentation is no longer updated on its original website. The libraries and .bof files involved in this experimental platform construction can be downloaded from the Xinjiang Astronomical Observatory Data Center at: http://data.xao.ac.cn/static/RAOCH2_platform.tar.gz.

In accordance with the real-time processing requirements for massive radio astronomical signals, this paper investigates the construction of digital backend system environments. Leveraging the versatile CAPSER hardware development platform and its corresponding software environment, we have constructed a digital backend system experimental platform based on ROACH2. Using examples provided by CAPSER, we tested, debugged, and analyzed the constructed platform, achieving the complete process of example design, simulation, compilation, and execution. Experimental results demonstrate that the platform built in this paper operates stably and reliably. The constructed digital backend environment has currently been applied in related research fields including pulsar signal simulation processing, radio frequency interference mitigation algorithm testing, and high-speed digital signal transmission.

References [1] Yang Wenjun, Yang Jun, Jiang Wu, Xia Bo, Li Jian, Cui Lang, Zhang Hua, Li Peng, Gao Zhifu. Establishment of the DBBC2 Digital Backend

System at Nanshan Station of Xinjiang Astronomical Observatory[J]. Astronomical Research & Technology, 2018, 15(01): 32-39. [2] WERTHIMER D. The CASPER collaboration for high performance open source digital radio astronomy instrumentation. In: Proceedings of General Assembly and Scientific Symposium. Istanbul: IEEE, 2011. [3] RANSOM S M, DEMOREST P, FORD J, et al. GUPPI: Green bank ultimate pulsar processing instrument[C]//American Astronomical Society Meeting Abstracts# 214. 2009, 214. [4] SARKISSIAN J M, CARRETTI E, STRATEN W V. The Parkes Pulsar Backends[C]. AIP Conference Proceedings, 2011, 1357, 351-352 . [5] PARSONS A R, BACKER D C, SIEMION A, et al. A Scalable Correlator Architecture Based on Modular FPGA Hardware, Reuseable Gateware, and Data Packetization[J]. Publications of the Astronomical Society of the Pacific, 2008, 120(873): 1207-1221. [6] CHANG C, WAWRZYNEK J, BRODERSEN R W, et al. BEE2: a high-end reconfigurable computing system[J]. IEEE Design & Test of Computers, 2005, 22(2). [7] LAZARUS P, KARUPPUSAMY R, GRAIKOU E, et al. Prospects for high-precision pulsar timing with the new Effelsberg PSRIX backend[J]. Monthly Notices of the Royal Astronomical Society, 2016, 458(1): 868-880. [8] KEITH M J, JAMESON A, VAN STRATEN W, et al. The High Time Resolution Universe Pulsar Survey I: System configuration and initial discoveries[J]. Monthly Notices of the Royal Astronomical Society, 2010, 409(2): 619-627. [9] NAN R D, LI H X. Progress of FAST in science, technique and instrument (in Chinese). Sci Sin-Phys Mech Astron, 2014, 44: 1063-1074. [10] YU Y Z, PENG B, LIU K, et al. FAST ultra-wideband observation of abnormal emission-shift events of PSR B0919+06[J]. Science China(Physics, Mechanics & Astronomy), 2019, 62(05): 36-41. [11] Zhao Panzi, Li Binhua, Mao Longye, Tao Yong. FPGA Implementation of Lucky Imaging Algorithm[J/OL]. Astronomical Research & Technology: 1-7. [12] Ji Zhicheng, Gao Chunneng, Wu Dinghui. FPGA Digital Signal Processing Design Tutorial: System Generator Introduction and Advanced[M]. Xidian University Press, 2008.

Note: Figure translations are in progress. See original paper for figures.