

Extending CASPER support to third-party RFSoc boards: a case study with TQ47DR

Jian Li^{1,2,3} , Maozheng Chen^{1,2,3*}, Xuefeng Duan^{1,3} , Yanling Liu^{1,3}, Xin Pei^{1,2,3} ,
Hailong Zhang^{1,2} , Peng Li¹, Qingsong Zhang^{1,2}, Kai Wang^{1,3}

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

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

³Xinjiang Key Laboratory of Microwave Technology, Urumqi 830011, China

*Correspondence: chen@xao.ac.cn

Received: September 8, 2025; Accepted: October 27, 2025; Published Online: October 28, 2025; <https://doi.org/10.3724/ati2025076>; <https://cstr.cn/32083.14.ati2025076>

© 2026 Editorial Office of Astronomical Techniques and Instruments, Yunnan Observatories, Chinese Academy of Sciences. This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Citation: Li, J., Chen, M. Z., Duan, X. F., et al. 2026. Extending CASPER support to third-party RFSoc boards: a case study with TQ47DR. *Astronomical Techniques and Instruments*, 3(4): 301–311. <https://doi.org/10.3724/ati2025076>.

Abstract: The open-source Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) toolflow has become a popular choice for building reconfigurable digital backends in radio astronomy. We extend this toolflow to the third-party TQ47DR Radio-Frequency System-on-Chip (RFSoc) platform, a cost-effective and widely available board. Our implementation includes a custom PetaLinux system, a lightweight control server, a deterministic clock-management driver, and platform-specific yellow-block adaptations that expose the on-chip data converters and 100-Gigabit Ethernet interfaces. System validation demonstrates high-quality converter performance, stable packet streaming, and real-time spectrometry, confirming that such third-party hardware can be integrated seamlessly into this open-source programming ecosystem for next-generation instruments.

Keywords: RFSoc; CASPER toolflow; Digital backend; Radio astronomy instrumentation; Third-party development boards

1. INTRODUCTION

As radio astronomy continues to demand wider bandwidths, higher sensitivity, and real-time processing capability, digital backends must balance scalability with performance. Such systems are expected to support multi-channel high-speed acquisition, wideband data capture, and high-throughput transmission, while also maintaining flexibility and cost-effectiveness.

In 2018, Xilinx introduced the Radio-Frequency System-on-Chip (RFSoc)¹, originally designed for 5G and radar applications, which has become increasingly attractive for radio astronomy backends. RFSoc integrates Field Programmable Gate Array (FPGA) logic, Advanced RISC Machine (ARM) processors, multi-gigasample Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs) (12–14-bit, variant-dependent), and high-speed Ethernet into a single chip, providing up to 16 ADC and 16 DAC channels with ADC rates up to 5.9 Giga-Samples Per Second (GSPS) and DAC rates up to 10 GSPS, while retaining a low-power, compact form factor². Researchers worldwide have since begun exploring

RFSoc-based backends for astronomical applications. Liu et al.^[1] systematically characterized the performance of Xilinx RFSoc data converters, showing that their Spurious-Free Dynamic Range (SFDR), Signal-to-Noise-and-Distortion Ratio (SNDR), Effective Number Of Bits (ENOB), and channel isolation closely match vendor specifications in most scenarios, and are sufficient for radio-astronomy receivers. Toledo-Moreo et al.^[2] reported an acquisition system based on an RFSoc board for a ground-based astronomical experiment, showing the feasibility of direct wideband digitization and real-time processing. Zhang et al.^[3] proposed a heterogeneous backend scheme for the QiTai radio telescope (QTT), based on RFSoc and Graphics Processing Units (GPUs), implementing direct Radio Frequency (RF) sampling, pre-processing, and high-speed data transmission, and verified the design with H₂CO

1. <https://www.amd.com/en/products/adaptive-socs-and-fpgas/soc/zyng-ultrascale-plus-rfsoc.html>

2. <https://docs.amd.com/v/u/en-US/ds889-zyng-usp-rfsoc-overview>

absorption line observations at the Nanshan 26-m telescope. Du et al.^[4] developed a sub-band division algorithm on RFSoc hardware to process ultra-wideband pulsar signals, demonstrating its applicability to real-time pulsar observations. In addition, Burnett^[5] and Pei et al.^[6] explored the integration of RFSoc platforms with array receivers for scalable backend architectures. Men et al.^[7] developed an RFSoc-based F-engine for the ARGOS demonstrator array³. Finally, early demonstrations by Pei et al.^[8] confirmed the potential of RFSoc as a compact and efficient alternative to traditional FPGA-based systems.

Over the past two decades, the Center for Astronomy Signal Processing and Electronics Research (CASPER)⁴ has pioneered an open-source, modular toolflow that combines Simulink-based design with FPGA firmware generation. The CASPER ecosystem provides a comprehensive toolflow for radio signal processing, including yellow blocks for hardware interfaces, green blocks for digital signal processing Internet Protocol (IP), and a Python-based software stack. Beyond its open-source libraries, CASPER has also developed a series of FPGA hardware platforms, ranging from the Reconfigurable Open Architecture Computing Hardware1/2 (ROACH1/2), the Square Kilometre Array Reconfigurable Application Board (SKARAB), and the Smart Network ADC Processor 1/2 (SNAP1/2)^[9] to more recent RFSoc-based boards. These technologies have been widely used in radio astronomy instruments such as Precision Array for Probing the Epoch of Re-ionization (PAPER)^[10], Green Bank Ultimate Pulsar Processing Instrument (GUPPI)⁵, Versatile GBT Astronomical Spectrometer (VEGAS)^[11], Hydrogen Epoch of Reionization Array (HERA), CASPER Parkes Swinburne Recorder (CASPSR)^[12], ROACH2 Digital Backend (R2DBE)^[13], and MeerKAT radio telescope (MeerKAT)⁶. In China, CASPER-based backends have also been adopted at several major facilities, including the Five-hundred-meter Aperture Spherical Telescope (FAST) 19-Beam backend^[14]. The Digital Backend System (DIBAS)^[15] deployed at the Tianma 65-m telescope and the 40-m radio telescope at Yunnan Observatories, Chinese Academy of Sciences, are both built on the CASPER ROACH2 platform, and the 26-m Nanshan Radio Telescope (NSRT) employs the CASPER platform for fast radio burst^[16] and pulsar observations^[17]. Zhang et al.^[18] investigated a baseband spectral line digital backend based on the SNAP platform.

Meanwhile, CASPER continuously tracks advances in radio technology, adopting new techniques to develop reusable radio signal-processing platforms and driving upgrades of radio-astronomy backends. However, for RFSoc specifically, support in the toolflow remains focused on a few official evaluation boards (e.g., ZCU111, ZCU208, RFSoc 4x2)⁷, which restricts broader adoption of cost-effective third-party platforms and runs counter to CASPER's modular, reconfigurable philosophy.

To address this limitation, we have integrated the third-generation RFSoc chip, model TQ47DR, a cost-effective,

high-performance RFSoc platform, into the CASPER ecosystem. Compared with CASPER's official RFSoc boards, TQ47DR provides dual 100 Gigabit Ethernet (GbE), one pulse per second (PPS) and multiple trigger I/Os, and Multi-Tile Synchronization (MTS) that is straightforward to implement. It is readily available from vendors and reduces hardware cost by 60%–80% while matching the ADC/DAC channel counts and providing expanded I/O (for example, an additional 100 GbE port). Table 1 summarizes these differences.

Our contributions include a customized PetaLinux kernel aligned with CASPER requirements, porting of the Karoo Array Telescope Control Protocol (KATCP)⁸ server for runtime host-firmware communication, a dedicated clock-management driver for precise synchronization, and CASPER-compliant yellow blocks that expose high-speed Radio Frequency Down Converter (RFDC) and 100 GbE interfaces. Together, these efforts enable TQ47DR to participate fully in CASPER's design-simulation-compilation-deployment workflow and provide a practical pathway toward scalable digital backends in radio astronomy.

2. CASPER TOOLFLOW AND SOFTWARE STACK

2.1. Modular Design with Simulink and Vivado

The CASPER toolflow adopts a block-based graphical design paradigm. Engineers use Simulink to construct FPGA designs by dragging, configuring, and connecting modules from dedicated libraries. These modules are categorized into two main types: Digital Signal Processing (DSP) blocks and hardware interface blocks. The DSP blocks, typically visualized in green or brown, are composed of low-level system generator primitives and can be dynamically parameterized via scripts. The hardware interface blocks, or “yellow blocks” include platform-specific modules such as ADC/DAC interfaces, Ethernet cores, clocking logic, and memory I/O. These are tightly coupled to the hardware and include associated Hardware Description Language (HDL) drivers, constraint files, and IP cores.

3. <https://argos-telescope.eu/>

4. <https://casper.astro.berkeley.edu>

5. <https://safe.nrao.edu/wiki/bin/view/CICADA/NGNPP>

6. <https://indico.tlabs.ac.za/event/48/contributions/620/317/341/Kapp-HPSP-MeerKAT-2014.pdf>

7. <https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/rfsoc/readme.html>

8. https://indico.ict.inaf.it/event/74/contributions/1046/attachment/s/657/721/KATCP_MeerKAT_Overview.pptx.pdf

Table 1. Comparison of RFSoc development boards (ZCU111, ZCU208, and TQ47DR)

Feature/Board	ZCU111	ZCU208	TQ47DR (This work)
RFSoc chip	XCZU28DR (Gen 1)	XCZU48DR (Gen 3)	XCZU47DR (Gen 3)
ADC channels	8 @ 4.096 GSPS (12-bit)	8 @ 5 GSPS (14-bit)	8 @ 5 GSPS (14-bit)
DAC channels	8 @ 6.554 GSPS (14-bit)	8 @ 9.85 GSPS (14-bit)	8 @ 9.85 GSPS (14-bit)
DDR4 memory	4 GB PS+4 GB PL	4 GB PS+8 GB PL	4 GB PS
Ethernet	4 × 25 GbE SFP28	4 × 25 GbE SFP28	Dual 100 GbE QSFP28
Clocking	LMK04208+LMX2594+ext ref	LMK04828+LMX2594+ext ref	LMK04832+ext ref
Trigger/Sync I/O	Not available	Not available	1 PPS+multiple TRIG I/O
Multi-chip sync support	Very difficult	Difficult	Easy to implement

2.2. Compilation Pipeline and Firmware Artifacts

The CASPER toolflow compiles Simulink DSP designs into deployable firmware and matching runtime assets in a single pass. A single build yields .bit (Vivado PL bitstream), .fpg (CASPER package with the register map/Block Random Access Memory (BRAM) inits used by casperfpga/KATCP), and .dtbo (device tree overlay that binds RFDC). At runtime, PetaLinux boots, the .dtbo is applied, and KATCP loads .fpg, exposing registers and BRAM for scripted control and telemetry. This alignment reduces HDL overhead and enables reuse across third party RFSoc boards within a standard design → simulation → compile → deploy workflow. A detailed description of this toolflow and the resulting artifacts is provided by Burnett^[5]. With these artifacts in place, the casperfpga client and the on board KATCP server constitute the software control plane channel used at runtime to load the design and access registers (see section 2.3).

2.3. Runtime Communication via Casperfpga and KATCP

The casperfpga⁹ Python library serves as the primary software interface for interacting with CASPER-based firmware running on reconfigurable hardware platforms. It allows users to load bitstreams, read and write registers, configure DSP blocks, and manage Ethernet interfaces. This library removes the complexity of low-level hardware control, allowing seamless interaction from high-level Python scripts or graphical user interfaces.

In addition to the firmware and hardware modules, a key component enabling communication between the host and FPGA is the KATCP server. This lightweight, text-based protocol runs on the embedded PetaLinux system of the RFSoc platform and provides the main interface for software control.

The KATCP server acts as a middleware layer between the casperfpga client and the FPGA fabric, offering a unified interface to read and write registers, monitor hardware status, and configure processing pipelines. It exposes user-defined registers, memory-mapped control blocks, and yellow block metadata to remote clients via Transmission Control Protocol (TCP).

This interface is essential for integrating custom logic developed with Simulink, using high-level software control. Once the bitstream is loaded and the device tree is con-

figured, the KATCP server running on the RFSoc’s ARM core allows real-time control, data acquisition, and system monitoring directly from backend software or graphical environments.

3. HARDWARE AND SOFTWARE INTEGRATION: ENABLING CASPER ON TQ47DR

3.1. System Overview of the TQ47DR Board

The TQ47DR RFSoc development board serves as the core hardware platform in this study. It is based on the Xilinx Zynq UltraScale+RFSoc XCZU47DR-1156I chip, which integrates programmable logic (PL), a quad-core ARM Cortex-A53 processing system (PS), 8-channel 5 GSPS 14-bit ADCs and 8-channel 9.85 GSPS 14-bit DACs, enabling full-duplex analog I/O at extremely high sample rates. It provides direct RF sampling capability up to several GHz, as shown in Fig. 1. In addition to the high-speed data converters, the board includes:

- 4 GB DDR4 memory (4 GB for PS);
- LMK04832 with Voltage-Controlled Crystal Oscillator (VCXO);
- External 10 MHz reference clock and 1 PPS synchronization;
- Dual Quad Small Form factor Pluggable 28 (QSFP28) ports for 100 GbE connectivity;
- Integrated GbE (PS-side) for control and management;
- Joint Test Action Group (JTAG), Universal Asynchronous Receiver/Transmitter (UART), and General Purpose Input Output (GPIO) interfaces for development and debugging.

The RFSoc fabric on TQ47DR provides the standard mix of programmable logic (Look-Up Table (LUT)/Flip-Flop (FF)), on-chip memories (BRAM and Ultra RAM (URAM)), DSP48 slices, and high-speed serial transceivers that map to the Ethernet Physical Layer (PHY). In practice, these resources are broadly comparable to those of Xilinx’s ZCU111 (Gen 1) and ZCU208

9. <https://github.com/casper-astro/casperfpga>

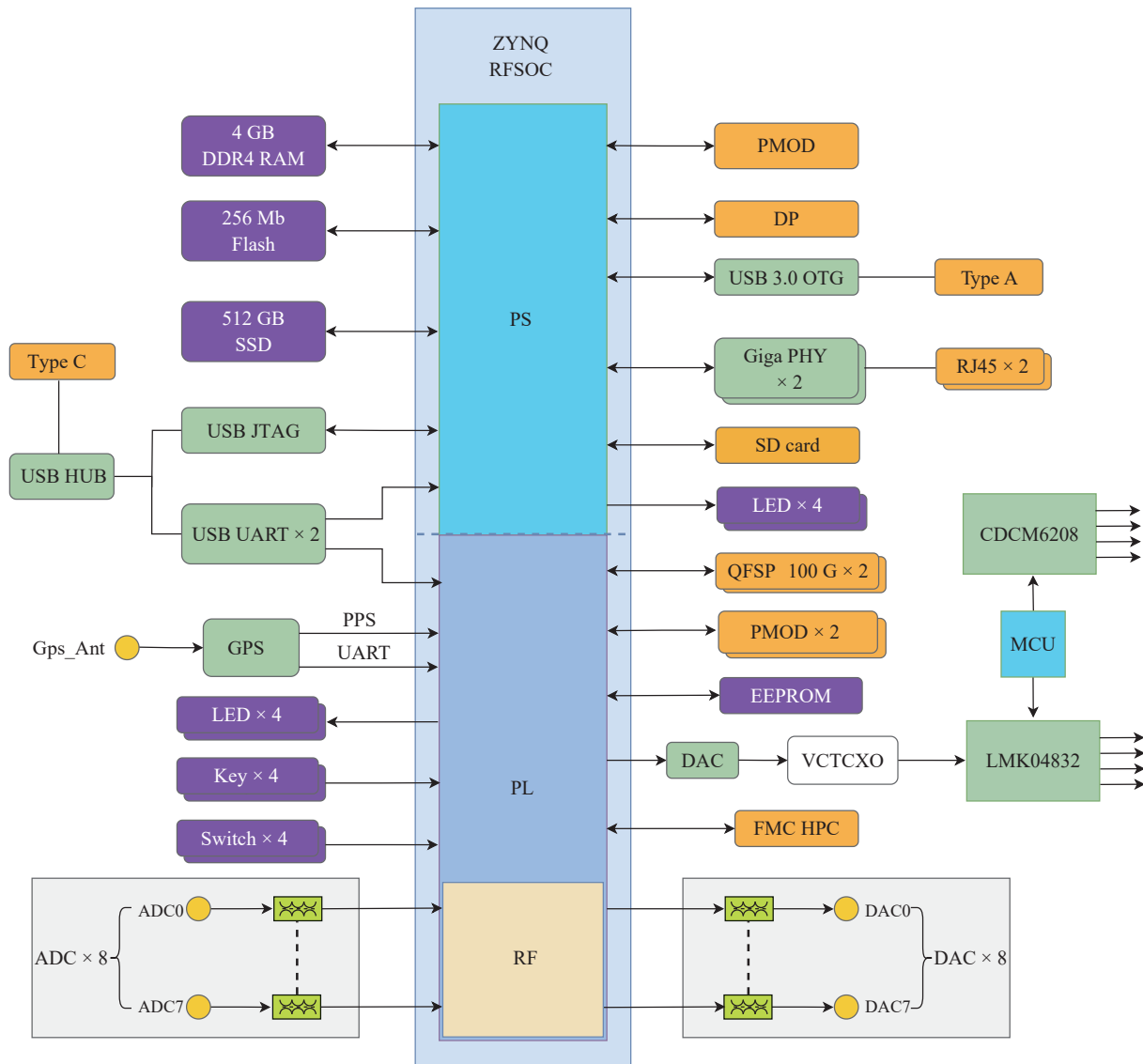


Fig. 1. System-level block diagram of the TQ47DR RFSoc board. The ZU47DR integrates PS and PL resources, connected to high-speed ADC/DACs, DDR4 memory, dual 100 GbE ports, and precision clocking (LMK04832, VCXO, EXT_REF), 1 PPS, TRIGS. Auxiliary interfaces include JTAG/UART, USB, GPS, user I/O, and expansion via FPGA Mezzanine Card (FMC) HPC.

(Gen 3) evaluation boards. As shown in Table 1, TQ47DR adds dedicated synchronization I/O—1 PPS and multiple trigger inputs—together with dual 100 GbE interfaces, which better meet the requirements of a general-purpose digital sampling platform for radio astronomy.

3.2. Custom PetaLinux Deployment without Vendor BSP

A Vivado Multiprocessor System-on-Chip (MPSoC) design was first created for the TQ47DR board (no vendor Board Support Package (BSP) is available), configuring only the essentials—DDR controller, UART, Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I²C), Secure Digital (SD), Ethernet Media Access Control (MAC), and Advanced eXtensible Interface (AXI) interconnect bridges to the PL—and explicitly setting the M_AXI_HPM0_FPD master port to a 32-bit data width to match the CASPER toolflow's 32-bit register access width used by KATCP/casperfpga. After instantiation and

verification of these peripherals, the hardware handoff is exported as an *Xilinx System Archive (XSA)* (successor to Hardware Description File (HDF)) packaging IP configuration, address map, clocks, and constraints; this *XSA* is then supplied to PetaLinux to generate a board-specific embedded Linux image (kernel, device tree, and required drivers).

Based on the exported *XSA* file, the PetaLinux build system is then used to compile and package the software stack, producing the key boot artifacts BOOT.BIN, image.ub, and boot.scr. These files respectively provide the first-stage bootloader and hardware configuration, the Linux kernel and root filesystem, and the boot script required to initialize the system. Together, they allow the RFSoc platform to boot reliably into a customized embedded Linux environment that supports CASPER toolflow integration. It is worth noting that for certain peripherals, such as the SD card controller and Ethernet interfaces, the

device tree may require manual modification to ensure correct initialization and runtime access. This step is essential for bridging the software-hardware interface on RFSoc platforms because it ensures proper boot flow, peripheral access, and integration with runtime control protocols.

3.3. SD Card Boot Configuration

To enable system startup on the TQ47DR platform, we adopted an SD card boot strategy consistent with Xilinx PetaLinux guidelines¹⁰. The SD card is partitioned into two regions: a *boot* partition and a *rootfs* partition.

The boot partition contains the three essential files described above, namely BOOT.BIN, image.ub, and boot.scr. These files provide the first-stage bootloader and hardware configuration, the embedded Linux kernel with device tree, and the boot script for system initialization. Together, they ensure correct hardware bring-up and integration with CASPER runtime control protocols.

For the root filesystem, we adopted the minimal Ubuntu Base 22.04 release¹¹, which provides a lightweight yet extensible environment. This distribution was chosen for its compatibility with ARM64 architectures and the availability of standard Linux packages required for CASPER software deployment. The *rootfs* partition was formatted as ext4, after which the extracted Ubuntu Base root filesystem was copied to the partition.

During system boot, the First Stage Boot Loader (FSBL) loads BOOT.BIN, which contains components such as Universal Bootloader (U-Boot). The FSBL is responsible for initializing PS and PL peripherals, and for loading U-Boot. Subsequently, U-Boot executes the boot.scr script to load the Linux kernel (image.ub) into memory. Finally, the kernel starts up, mounts the root filesystem (*rootfs* partition) of the SD card, and completes the root filesystem switch. This provides a stable, Ubuntu-based runtime environment that supports both low-level hardware access and high-level KATCP control libraries.

3.4. KATCP Runtime Service on Rootfs

After preparing the SD card boot, we deploy the KATCP server on the RFSoc PetaLinux system to provide a runtime control channel for CASPER clients. Prior to building and installing the KATCP service, the low-level driver libraries required by the RFSoc data converters and register access must be provisioned. In particular, we install the RFSoc RFDC userspace driver and the lightweight platform abstraction layer libmetal, both sourced from the Xilinx *embeddedsw* repository¹².

Prerequisites. Using the PetaLinux toolchain (aarch64), we build and stage the headers and libraries for rfdc and libmetal into the root filesystem, ensuring they are available under standard include and library paths (e.g., /usr/include/, /usr/lib/). This step guarantees that the KATCP server can link against the RFSoc control primitives and access memory-mapped registers exposed by the firmware.

Building and deployment. As part of the porting pro-

cess, we cloned the KATCP source¹³, configured it against the installed dependencies, and compiled it using either the PetaLinux SDK cross-toolchain on the host or a native toolchain on the TQ47DR board, yielding the KATCP server binary used for runtime control in our system. The resulting server binary is staged under /bin. For further details on cross-compilation, please refer to the UG1144 PetaLinux Tools Reference Guide. A minimal example workflow is provided below:

```
git clone https://gitlab.ras.byu.edu/alpaca/casper/katcp.git
cd katcp && ./configure --host = aarch64-linux-gnu\
--prefix = /bin
make -j && DESTDIR = ./build
make install
```

Service integration. For unattended operation, the KATCP server can be installed as a systemd unit so that it starts automatically after networking and driver initialization. A minimal unit file is:

```
[Unit]
Description = TCPBorphServer allows programming
and communication with the FPGA
Wants = network.target
After = syslog.target network-online.target
[Service]
Type = simple
ExecStart = /bin/tcpborphserver3 -b /lib/firmware
Restart = on-failure
RestartSec = 10
KillMode = process
[Install]
WantedBy = multi-user.target
```

At runtime, the server provides a TCP socket on port 7147, mediating access to user-defined registers, memory-mapped control blocks, and CASPER yellow-block metadata.

3.5. Precision Clock Management and Synchronization

For a precise and reconfigurable clock, the TQ47DR provides flexible clocking capabilities, integrating an LMK04832 clock chip along with an onboard VCXO, external reference inputs, and synchronization interfaces. The clock chip is managed by a microprocessor, which communicates with the PS through a dedicated UART interface for configuration and control. To support this architecture, we developed a custom clock management driver that programs the device by parsing register configuration files

10. <https://docs.amd.com/r/en-US/ug1144-petalinux-tools-reference-guide/Booting-a-PetaLinux-Image-on-Hardware-with-SD-Card>

11. <https://cdimage.ubuntu.com/ubuntu-base/releases/22.04/release/>

12. <https://github.com/Xilinx/embeddedsw>

13. <https://gitlab.ras.byu.edu/alpaca/casper/katcp.git>

exported from Texas Instruments Clocks and Synthesizers (TICS) Pro Software¹⁴. These configurations are transmitted via the PS UART1 to the coprocessor, enabling run-

time clock setup and synchronization management. The overall architecture of the clock management and synchronization subsystem is shown in Fig. 2.

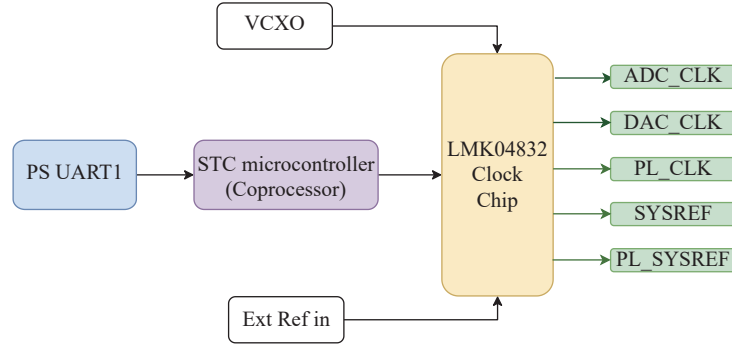


Fig. 2. Clock management subsystem of the TQ47DR. The LMK04832 clock chip, controlled by an STC microcontroller via PS UART1, distributes timing signals including ADC_CLK, DAC_CLK, PL_CLK, SYSREF, and PL_SYSREF. The subsystem integrates an onboard VCXO and supports external reference and synchronization inputs, ensuring deterministic timing for RFSoc data converters and FPGA fabric.

3.6. Designing CASPER-compatible Blocks for Third-party Platforms

For TQ47DR to work within the CASPER toolflow, we extended the Simulink CASPER library by adding new yellow blocks and introducing timing constraints tailored to the board. These include TQ47DR.m (configuration and simulation) and tq47dr.py (board description and Vivado project generation). Concurrently, we updated rfdc_mask.m and related files to match the RFDC configuration (including ADC and DAC). In addition, we patched the 100 GbE yellow-block stack to correct Address Resolution Protocol (ARP) table handling (addressing inconsistencies between destination IP and MAC resolution). Further implementation details and minimal examples are available in our TQ47DR development repository¹⁵ and example programs¹⁶.

In this work, we also developed a set of signal-processing yellow blocks that directly invoke Very High Speed Integrated Circuit Hardware Description Language (VHDL) modules and IP cores, to implement functions such as polyphase filter banks and Fast Fourier Transform (FFT). Each block was first validated in isolation using loopback and diagnostic firmware to ensure signal integrity, clock-domain crossing correctness, and AXI interface timing. Once verified, the blocks were integrated into the broader Simulink design, enabling end-to-end compilation within the CASPER toolflow.

This integration methodology demonstrates how third-party RFSoc boards can be fully incorporated into the CASPER ecosystem. The process allows reuse of the open-source design flow while accommodating hardware diversity.

4. SYSTEM VALIDATION AND PERFORMANCE EVALUATION

4.1. ADC Performance Evaluation

To evaluate our setup, a low-phase-noise RF source

(Rohde & Schwarz SMA100B, 8 kHz–20 GHz) feeds the RFSoc ADC through a low-loss, phase-stable coaxial cable (MiCable B10 series). Fixed attenuation is inserted where needed to set the input level and protect the front end. The RFSoc board is housed in a metal enclosure that provides effective Electromagnetic interference (EMI) shielding. All measurements are performed at room temperature, with a 50 Ω input termination, unless stated otherwise.

We use a minimal FPGA design in which ADC samples are continuously written to on-board BRAM in time order. A host utility, built with the casperfpga library, arms the buffer, retrieves contiguous blocks via the runtime register interface, and stores the raw streams to disk for offline analysis. The physical connection and dataflow are summarized in Fig. 3.

Following established practice for RFSoc data-converter characterization, we evaluate three representative sampling rates: 2, 4 and 5 GSps. For each rate, a single tone is swept across the band and ~ 10 frequencies are recorded to ensure uniform spectral coverage. Each record is windowed with a Hamming window, a 16384-point FFT is computed, and power spectra are obtained by correlation. We report three standard metrics.

First, SFDR reflects the ADC's ability to suppress harmonics and intermodulation products. It can be shown as,

$$SFDR = 20 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{spur,max}}} \right) \text{ dBc}, \quad (1)$$

where P_{signal} is the power of the fundamental tone and

14. <https://www.ti.com/tool/TICSPRO-SW>

15. https://github.com/xao-lijian/mlib_devel-m2019a_tq47dr2025

16. https://github.com/xao-lijian/tutorials_tq47dr

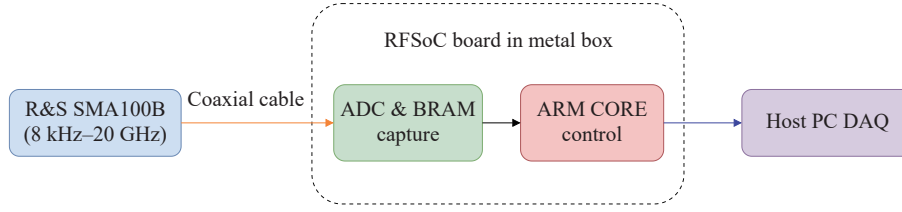


Fig. 3. ADC test signal path and dataflow. A low-phase-noise RF source (R&S SMA100B) feeds the RFSoc ADC through a low-loss, phase-stable coaxial cable. Samples are buffered in on-board BRAM and read out by the host PC for analysis. The RFSoc board is mounted in a metal enclosure for EMI suppression.

$P_{\text{spur,max}}$ is the strongest spurious component; SFDR reflects the ADC’s ability to suppress harmonics and intermodulation products.

Unlike SFDR, SNDR integrates all noise-plus distortion. It can be shown as

$$\text{SNDR} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{noise}} + P_{\text{distortion}}} \right) \text{ dB}, \quad (2)$$

where P_{noise} denotes in-band noise power and $P_{\text{distortion}}$ aggregates harmonic/intermodulation contributions across the band.

Equally important is ENOB. It provides an intuitive resolution measure derived from SNDR. It can be shown as

$$\text{ENOB} = \frac{\text{SNDR} - 1.76}{6.02}, \quad (3)$$

and typically decreases towards the Nyquist limit as non-ideal effects grow.

Fig. 4 aggregates SFDR and SNDR over all swept frequencies at 2, 4 and 5 GSps. At low input frequencies, SFDR consistently exceeds 70 dBc, while SNDR is lower because it integrates quantization noise and distortion across the band. Both metrics degrade as the tone approaches the Nyquist limit. The corresponding ENOB, shown in Fig. 5, remains within 10–12 bits across the tested range. It approaches 12 bits at low frequencies and gradually falls to approximately 10 bits near Nyquist. Measurements at 2, 4 and 5 GSps follow the same trend, indicating that ENOB is not strongly rate-dependent within this range.

4.2. Multi-Tile Synchronization (MTS) Test

We evaluated the MTS capability of the RFSoc

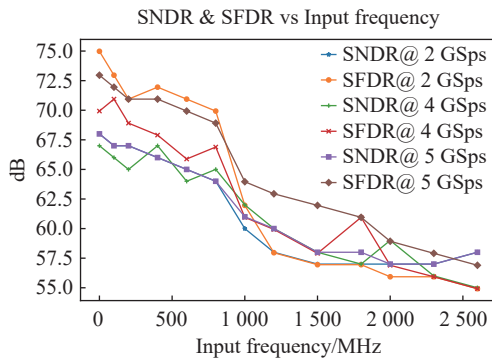


Fig. 4. Measured SFDR and SNDR of the RFSoc ADC at different sampling rates.

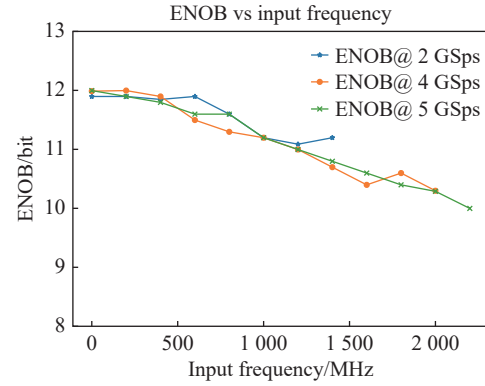


Fig. 5. Measured ENOB of the RFSoc ADC at different sampling rates.

under controlled laboratory conditions, by feeding a broadband signal from a signal generator, covering the 0–2.5 GHz range, into an eight-way power divider, with the outputs connected to the eight ADC channels of the RFSoc. With MTS enabled, we characterized synchronization performance by repeatedly powering up the system and recording data across different frequency points.

In the specific testing procedure, to quantify synchronization performance, channel 0 was used as a reference. The relative delay and phase of each remaining channel were estimated using a cross-correlation algorithm, and the corresponding time offsets were derived from the measured phase differences as a function of frequency.

We estimate the inter-channel delay from the measured phase difference as

$$\Delta t = \frac{\Delta \phi}{2\pi f}, \quad (4)$$

where $\Delta \phi$ is the unwrapped phase difference (in radians) between the channel under test and the reference channel, and f is the tone frequency at the point of measurement (in Hz).

To quantify initialization stability across multiple power cycles, we define the delay fluctuation for each channel as

$$\Delta t_{\text{jitter}} = \max_{i=1,\dots,N} (t_i) - \min_{i=1,\dots,N} (t_i), \quad (5)$$

where t_i is the measured delay of the i -th power-up and N is the number of measurements. This metric captures the channel-to-channel timing uncertainty introduced by startup variability.

Table 2 summarizes the results over six power cycles, obtained using a 900 MHz tone at 0 dBm input power. Most channels showed sub-5 ps fluctuation, with the worst-case variation measured at 8.52 ps (channel 4).

These results confirm that the MTS mechanism provides consistent inter-channel synchronization across repeated initializations, well within the tolerance required for wide-band radio astronomy signal processing.

Table 2. Delay variation across multiple power cycles for each channel

Channel	1 st	2 nd	3 rd	4 th	5 th	6 th	$\Delta t_{\text{jitter}}/\text{ps}$
Channel 0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Channel 1	-9.16	-8.40	-7.57	-8.10	-8.12	-6.65	2.51
Channel 2	-0.14	-1.22	1.64	3.26	-1.73	2.96	4.99
Channel 3	-17.10	-17.76	-17.19	-16.90	-18.91	-17.90	2.01
Channel 4	6.85	6.49	-1.17	-1.67	5.72	-1.14	8.52
Channel 5	-9.01	-9.90	-13.81	-14.08	-9.70	-12.70	6.38
Channel 6	-14.83	-14.25	-13.36	-13.76	-14.02	-15.12	1.76
Channel 7	-14.56	-13.91	-10.39	-10.61	-13.72	-10.14	4.42

Example: For channel 4, the maximum delay is 6.85 ps and the minimum is -1.67 ps. The fluctuation is therefore $\Delta t_{\text{jitter}} = 6.85 - (-1.67) = 8.52$ ps.

4.3. 100 GbE Throughput Test

We first validated the physical layer using the Xilinx Integrated Bit Error Ratio Tester (IBERT) software¹⁷. A 24 h stress run showed error-free transmission, with the eye diagram showing an open-Unit Interval (UI) margin of 78.9%, indicating excellent signal integrity.

To evaluate 100 GbE throughput and packet loss, we also developed a CASPER-based test program¹⁸. The transmitter exposes two control registers in `pkt_sim`: `payload_len` sets the number of data beats per frame (each beat is 512 bit = 64 B), and `period` sets the transmit gap (inter-frame interval) in FPGA PL clock cycles (f_{clk}). The generator creates payloads up to 8512 B and prefixes each frame with a 64-bit counter in the first 8 bytes, facilitating drop detection on the receiver side¹⁹. The payload per frame is

$$L_{\text{bits}} = \text{payload_len} \times 64 \times 8, \quad (6)$$

and the target payload throughput is

$$R_{\text{b/s}} = \frac{f_{\text{clk}}}{\text{period}} \times L_{\text{bits}}, \quad R_{\text{Gb/s}} = \frac{R_{\text{bps}}}{10^9}. \quad (7)$$

For the measurements reported here, we used the following parameters: for $f_{\text{clk}} = 256$ MHz, `payload_len` = 130, and `period` = 172,

$$R_{\text{Gb/s}} = \frac{256 \times 10^6}{172} \times (130 \times 64 \times 8) \div 10^9 \approx 99.07. \quad (8)$$

The program splits the data stream evenly into two flows, each sent to a different destination IP, so that two servers receive in parallel, lowering per-host ingress load and reducing drops under heavy Network Interface Card (NIC) load. The dual-server RFSoc-based 100 GbE test architecture can be found in the footnote-linked document, which details the overall test setup and implementation²⁰.

A Mellanox SN2700 high-performance Ethernet switch acts as the intermediate node between the

TQ47DR RFSoc and two reception servers, enabling low-latency, non-blocking forwarding of split 100 GbE streams for stable RFSoc-server data interaction. Two servers with Intel Xeon Gold 6240C CPU, Mellanox ConnectX-5 100 GbE NIC, and 128 GB DDR4 RAM were used for reception, running Ubuntu 18.04 with NVIDIA Voltaire Messaging Accelerator (VMA) accelerated User Datagram Protocol (UDP) listeners.

Receiver performance was verified with VMA-accelerated UDP listeners that log one-second bitrate and packet rate. As shown in Fig. 6, each server sustains ~ 49.5 Gb s⁻¹ (approximately 7.4×10^5 frame s⁻¹ with jumbo frames), yielding ≈ 99 Gb s⁻¹ aggregate. Loss counters remain constant over the interval, indicating no additional receiver-side drops. The loss counter operates by checking the continuity of the 64-bit frame counter inserted by the FPGA transmitter, thereby detecting and tallying any missing packets.

As a practical note, under normal conditions, packets sent by the FPGA are reliable. Most losses occur on the server side, specifically at the NIC or within the receiving application. In our experiments we used two servers for reception and monitoring and applied targeted optimizations to the receiver. With four or eight receivers, even without such optimizations, the system can achieve zero packet loss.

17. <https://docs.amd.com/r/en-US/ug908-vivado-programming-debugging/In-System-IBERT>

18. https://github.com/xao-lijian/net_speed_dual_ip

19. https://github.com/xao-lijian/net_speed_dual_ip/blob/main/doc/s/100g_test_program.pdf

20. https://github.com/xao-lijian/net_speed_dual_ip/blob/main/docs/rfsoc_data_split.pdf

```

root@om-gpu0:/home/uwb# LD_PRELOAD=libvma.so ./udp_s1 -b 10.17.16.11 -r 67108864 -C 8
VMA INFO: VMA VERSION: 9.6.4-1 Release built on Jul 14 2022 14:45:59
VMA INFO: Cmd Line: ./udp_s1 -b 10.17.16.11 -r 67108864 -C 8
VMA INFO: OFED Version: MLNX_OFED_LINUX-5.7-1.0.2.0:
VMA INFO: Log Level INFO [VMA_TRACELEVEL]
VMA INFO: Listening on 10.17.16.11:60000 ...
[server_1_ls] rate = 49.190 Gb/s, pps = 739036, lost(before) = 600003
[server_1_ls] rate = 50.061 Gb/s, pps = 752116, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003
[server_1_ls] rate = 49.386 Gb/s, pps = 741984, lost(before) = 600003
[server_1_ls] rate = 49.684 Gb/s, pps = 746451, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 600003
[server_1_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 600003

root@om-gpu1:/home/uwb# LD_PRELOAD=libvma.so ./udp_s2 -b 10.17.16.13 -r 67108864 -C 8
VMA INFO: VMA VERSION: 9.6.4-1 Release built on Jul 14 2022 14:45:59
VMA INFO: Cmd Line: ./udp_s2 -b 10.17.16.13 -r 67108864 -C 8
VMA INFO: OFED Version: MLNX_OFED_LINUX-5.7-1.0.2.0:
VMA INFO: Log Level INFO [VMA_TRACELEVEL]
VMA INFO: Listening on 10.17.16.13:60000 ...
[server_2_ls] rate = 47.684 Gb/s, pps = 716407, lost(before) = 380852
[server_2_ls] rate = 49.546 Gb/s, pps = 744382, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744217, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852
[server_2_ls] rate = 49.535 Gb/s, pps = 744218, lost(before) = 380852

```

Fig. 6. Console output from the two VMA-based UDP receivers during the 100 GbE test. Each host reports the one-second average bitrate and packet rate; together they sustain an aggregate of $\approx 99 \text{ Gb s}^{-1}$.

4.4. Spectrometer Demonstration

To demonstrate the complete signal-processing chain, we implemented a spectrometer on the TQ47DR platform following the CASPER tutorial example. The system was configured with a 2048 MSps sampling rate and a 2048-point FFT, employing CASPER’s RFDC, polyphase filter bank (PFB), FFT, accumulation (ACC), ACC control, and BRAM modules (see SUPPLEMENTARY MATERIAL). Broadband noise, filtered through a 0.1–1 GHz band-pass filter, was injected into ADC0. After simulation and compilation, the FPGA firmware was built. Using the casperfpga library, we uploaded and programmed the firmware. Key parameters, such as FFT_SHIFT and ACC_LEN, were also configured.

Spectral data were read from BRAM and displayed dynamically, as shown in Fig. 7, confirming correct operation of the spectrometer design. The measured spectrum covered a bandwidth of 0.1–1 GHz, showing excellent agreement with the expected filter response, validating both the stability of the system and the firmware implementation.

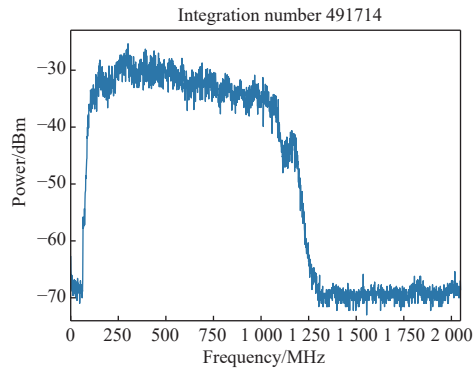


Fig. 7. Spectral output of the implemented spectrometer on the TQ47DR platform, read from BRAM and displayed in real time.

5. CONCLUSION AND OUTLOOK

Our work demonstrates end-to-end integration of the third-party TQ47DR RFSoc platform into the CASPER toolflow, bridging a practical gap between open-source design methods and widely available hardware. Through a

custom PetaLinux build, porting of the KATCP protocol, a clock-management driver, and CASPER-compliant yellow blocks, we achieve seamless design-simulation-compilation-deployment on TQ47DR. System-level validation demonstrates excellent ADC performance, deterministic multi-tile synchronization, and 100 GbE streaming at a $\approx 99 \text{ Gb s}^{-1}$ payload throughput (i.e., near line rate), together with a functional real-time spectrometer, supporting the platform’s suitability for wideband radio astronomy.

The use of TQ47DR reduces hardware costs by 60%–80% relative to official CASPER boards, while matching ADC/DAC channel counts and expanding I/O (for example, with an extra 100 GbE port). The approach has already been demonstrated in a practical setting—supporting initial pulsar observations with the NSRT and ongoing digital-backend development for QTT—and informs the related efforts of a small radio-telescope array at Guizhou Normal University and continuum observations with a 40 m telescope. These results underscore the practical and economic value of broadening CASPER hardware applicability with third-party RFSoc platforms.

Our future work will generalize platform support with reusable yellow-block templates and automated device-tree generation, and pursue long-duration deployments. In the longer term, dynamic reconfiguration and embedded machine learning within RFSoc fabrics may enable adaptive, power-efficient pipelines for next-generation radio astronomy.

ACKNOWLEDGEMENTS

We thank the staff responsible for the development of the CASPER libraries. We acknowledge the provision of various hardware support by the Radio Technology Laboratory of Xinjiang Astronomical Observatory, Chinese Academy of Sciences. The work was supported by the Light in China Western Region program (2022-XBQNXZ-012, 2022-XBQNXZ-015), the National Natural Science Foundation of China (12173087), the Foundation of Guizhou Provincial Education Department (KY(2023)059), the Natural Science Foundation of Xinjiang Uygur Autonomous Region (2024D01B88), the National Natural Science Foundation of China (12203094),

the Tianshan Talent Training Program (2024TSY-CCX0071), and Liupanshui Science and Technology Development Project (52020-2024-PT-01).

AI DISCLOSURE STATEMENT

AI-assisted technology is not used in the preparation of this work.

AUTHOR CONTRIBUTIONS

Jian Li led the practical implementation of the study, including adapting the CASPER library to TQ47DR, building the kernel and root filesystem, developing the FPGA firmware and timing constraints, and drafting the manuscript. Maozheng Chen conceived and supervised the project, and critically reviewed and revised the manuscript. Xuefeng Duan assisted with laboratory validation and data processing. Yanling Liu carried out ADC characterization experiments and prepared the figures. XinPei and Hailong Zhang provided guidance on the writing of this paper. Qingsong Zhang implemented 100 GbE link tests and IBERT validation. Peng Li assisted with the testing and handled the purchase of materials. Kai Wang handled data curation and reproducibility scripts and helped to edit the manuscript. All authors read and approved the final manuscript.

COMPLIANCE NOTE

All experiments were conducted on hardware lawfully procured through reputable local vendors. This manuscript presents generally available integration methods (software configuration, device-tree overlays, and CASPER-compatible blocks) and does not include non-public or export-controlled information.

DECLARATION OF INTERESTS

Hailong Zhang is an editorial board member for *Astronomical Techniques and Instruments*. He was not involved in the editorial review or the decision to publish this article. The authors declare no competing interests.

REFERENCES

- [1] Liu, C., Jones, M. E., Taylor, A. C. 2020. Characterizing the performance of high-speed data converters for RFSOC-based radio astronomy receivers. *Monthly Notices of the Royal Astronomical Society*, **501**(4): 5096–5104.
- [2] Toledo-Moreo, R., Pérez-Lemus, C., Pérez-Lizán, D., et al. 2025. An acquisition system based on an RFSOC board: Demonstration for a ground-based astronomical experiment. *IEEE Instrumentation & Measurement Magazine*, **28**(4): 60–68.
- [3] Zhang, Q. S., Chen, M. Z., Li, J. 2024. General heterogeneous real-time baseband backend system scheme applied to QTT based on RFSOC+GPU. *Frontiers in*

- Astronomy and Space Sciences*, **11**: 1420066.
- [4] Du, X., Zhang, H. L., Guo, S. C., et al. 2024. A sub-band division algorithm for ultra-wide bandwidth pulsar signals based on RFSOC. *Frontiers in Astronomy and Space Sciences*, **11**: 1391070.
- [5] Burnett, M. C. 2023. On algorithmic design methodologies, heterogeneous RFSOC/GPU beamformers, and cryogenic antenna efficiency evaluation for phased array receivers in radio astronomy. Doctoral thesis, Brigham Young University.
- [6] Pei, X., Wang, N., Werthimer, D., et al. 2022. Design of RFSOC-based Digital Phased Array Feed (PAF) and hybrid architecture beamforming system. *Research in Astronomy and Astrophysics*, **22**(4): 171–186.
- [7] Men, Y., Barr, E., Bansod, A., et al. 2025. An RFSOC-based F-engine for ARGOS. *arXiv: 2507.19386*.
- [8] Pei, X., Li, J., Wang, N., et al. 2021. Design of a multifunction high-speed digital baseband data acquisition system. *Research in Astronomy and Astrophysics*, **21**(10): 63–74.
- [9] Hickish, J., Abdurashidova, Z., Ali, Z., et al. 2016. A decade of developing radio-astronomy instrumentation using CASPER open-source technology. *Journal of Astronomical Instrumentation*, **5**(4): 1641001.
- [10] Parsons, A. R., Backer, D. C., Foster, G. S., et al. 2010. The precision array for probing the epoch of re-ionization: Eight station results. *The Astronomical Journal*, **139**(4): 1468–1480.
- [11] Prestage, R. M., Bloss, M., Brandt, J., et al. 2015. The versatile GBT astronomical spectrometer (VEGAS): Current status and future plans. In Proceedings of 2015 USNC-URSI Radio Science Meeting (Joint with AP-S Symposium).
- [12] Sarkissian, J. M., Carretti, E., van Straten, W. 2011. The Parkes Pulsar Backends. In Proceedings of American Institute of Physics Conference Series.
- [13] Vertatschitsch, L., Primiani, R., Young, A., et al. 2015. R2DBE: A wideband digital backend for the event horizon telescope. *Publications of the Astronomical Society of the Pacific*, **127**(958): 1226.
- [14] Jiang, P., Yue, Y. L., Gan, H. Q., et al. 2019. Commissioning progress of the FAST. *Science China Physics, Mechanics Astronomy*, **62**(5): 7–28.
- [15] Prestage, R. M., Bloss, M., Brandt, J., et al. 2014. Experiences with the design and construction of astronomical instrumentation using CASPER: The digital backend system. In Proceedings of American Astronomical Society Meeting Abstracts.
- [16] Liu, Y. L., Chen, M. Z., Li, J., et al. 2024. Design and application of an S-band fast radio bursts search pipeline for the Nanshan 26 m Radio Telescope. *Research in Astronomy and Astrophysics*, **24**(7): 075008.
- [17] Ergesh, T., Li, J., Duan, X. F., et al. 2022. Development of pulsar digital backend based on RFSOC. *Research in Astronomy and Astrophysics*, **22**(2): 18–30.
- [18] Zhang, Q. S., Chen, M. Z., Li, J. 2026. Development and application of a real-time baseband spectral line digital backend. *Astronomical Techniques and Instruments*, **3**(2): 170–183.

SUPPLEMENTARY MATERIAL

Fig. S1 shows the complete design diagram of the spectrometer developed based on CASPER; its detailed introduction is presented in Section 4.4.

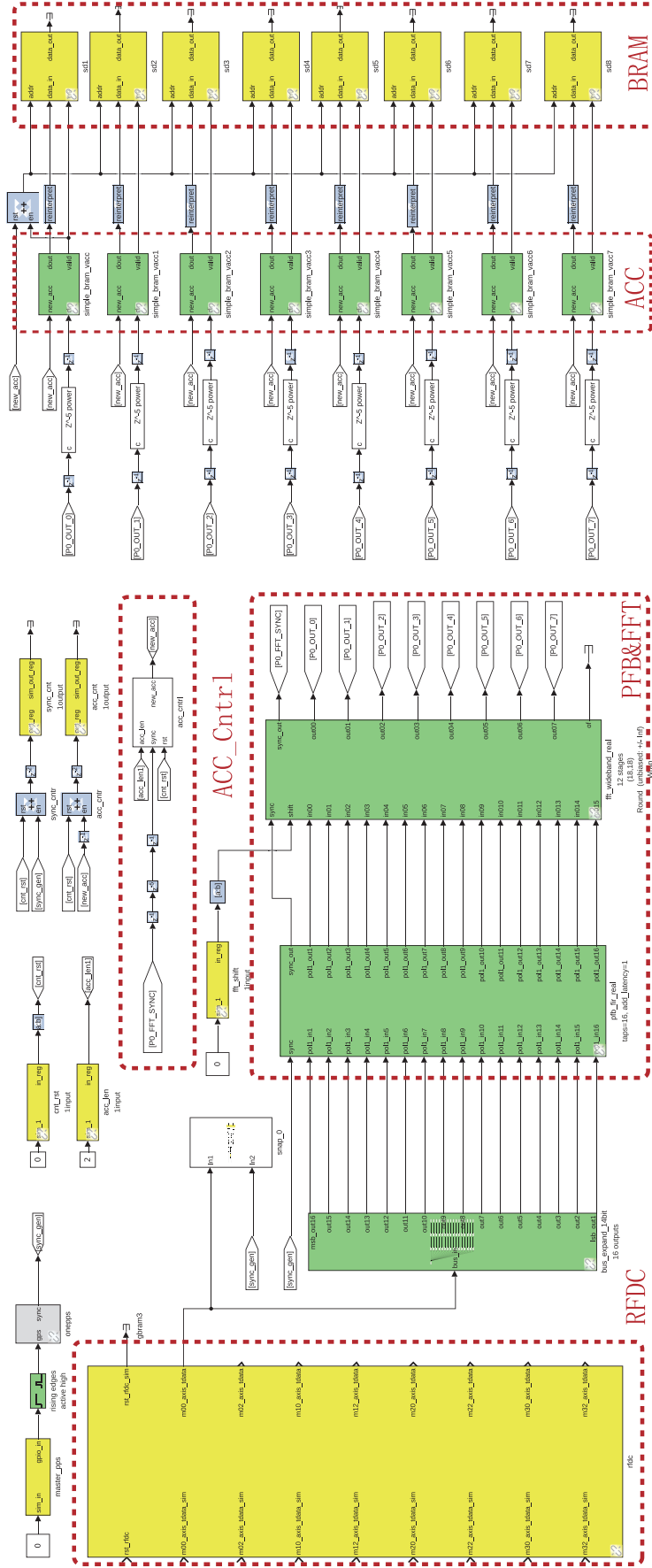


Fig. S1. Spectrometer design implemented on the TQ47DR platform using CASPER libraries, including RFDC, PFB, FFT, ACC, and BRAM modules.