Development of an embedded software platform based on Zynq FPGA SoC for Astrophysics applications

David S. Miranda Guillén
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria
Las Palmas de Gran Canaria, Spain
dmiranda@iuma.ulpgc.es

Pedro Pérez Carballo
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria
Las Palmas de Gran Canaria, Spain
carballo@iuma.ulpgc.es

***Abstract*—This paper describes a software platform used for scientific data capture in the field of astrophysics. This platform captures, processes, analyses, and stores 32 analogue signals at high speed. It is configured and controlled remotely via a client-server architecture application.**

Keywords - FPGA; Petalinux; Baremetal; AMP; OpenAMP; Zynq; ZedBoard; ADC; Quijote; NFS

I. Introduction

The Instituto de Astrofísica de Canarias (IAC) and his partners are working in the European project QUIJOTE-CMB (Q-U-I JOint TEnerife CMB) [1], an experiment aiming the characterization of the polarization of the Cosmic Microwave Background, and other galactic or extragalactic physical processes that emit in microwaves in the frequency range 10-42GHz, and at large angular scales (1 degree resolution).

IAC propose the creation and design of a software platform that allows an electronic system based on FPGA SoC to capture scientific data. The system (Fig. 1) consists of a power supply, a 32-channel AD7768 conversion subsystem [2] in differential mode and a subsystem consisting of two ZedBoards, each of which has a dual-core ARM Cortex™-A9 processor [3]. Both boards work together in a master-slave configuration and communicate via an FMC interface for the ADC configuration and sampled data transfer. For configuration, status display and other system operations, a client-server application is developed to perform these operations over a network using the TCP/IP protocol.

II. System architecture

The tasks of the system are shown in Fig.2. As can be seen, the system starts from a global configuration state and enters in a loop for data capture, processing, storing, and sending data loop. All these processes run concurrently in both the hardware and software domains. To maintain data consistency, a basic criterion is the possibility to modify the system configuration when the output buffer is empty, or the data will be discarded. In the FPGA SoC system architecture (Fig. 3), the main blocks and interfaces of the system can be identified.

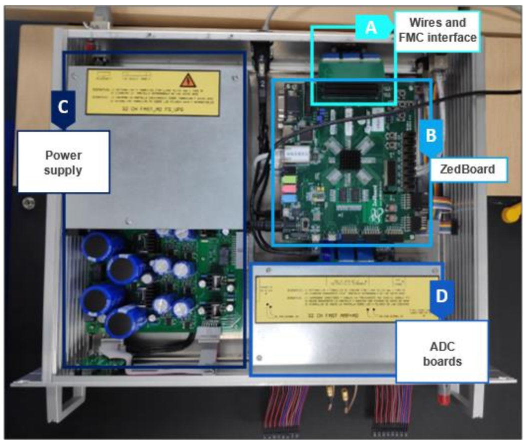


Figure 1. Top view of the equipment

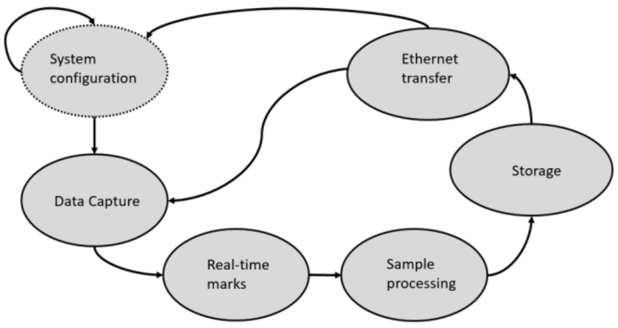


Figure 2. Main system processes

The processes shown in Fig. 2 have been mapped onto the reference architecture, using the heterogeneous architecture of the Zynq device [4]. The configuration processes have been mapped into a hardware/software system on CPU0 and CPU1 as dedicated blocks. The real-time data processing and marking processes are mapped in the hardware domain (Programmable Logic) and the storage processes in the software domain controlled by CPU0. Finally, data transfer processes are mapped in the software domain on CPU1. At a second level of task granularity, user interface management processes are included.

TABLE I. TASK MAPPING IN HETEROGENEOUS PROCESSING RESOURCES

	Task mapping		
	Software domain		Hardware domain
Task	CPU0 Linux	CPU1 Baremeta l	PL
System configuration	✓		✓
Data Capture		✓	✓
Processing		✓	✓
Real-Time Marks			✓
Data Buffer	✓	✓	✓
Data Storage	✓		
Ethernet transfer	✓		
User Interface	✓		

III. BLOCKS OF ARCHITECTURE

As shown in Fig. 3, the SoC FPGA Zynq is organised into two main parts: PS (Processing System) and PL (Programmable Logic). The PL includes a set of blocks that control the configuration of the power supply, the matching system, and the data conversion, as well as auxiliary signals that control certain devices external to the equipment.

On the other hand, the PS uses two ARM Cortex A9 cores in an asymmetric multiprocessing configuration (AMP) to support the heterogeneous operation of the system. Communication between the two subsystems is achieved through shared memory and dedicated interrupt mechanisms.

IV. SOFTWARE PLATFORM

For AMP implementation, a Linux-baremetal configuration has been used, where CPU0 (Core 0) includes a PetaLinux Operating System adapted to the Xilinx Zynq architecture, specifically configured for the ZedBoard, and CPU1 (Core 1) runs a specific baremetal application for data capture. The synchronisation of the processes running both processors uses an unsupervised OpenAMP system [5]. This is an open-source system for the synchronisation of processes in a heterogeneous system that employs a message passing mechanism using shared resources between both CPUs (Fig. 4). In this case, the processors run independently each under its own software stack, with no central software coordinating the operation.

For Linux system, Xilinx has a complete distribution, Petalinux [6], which provides a development environment based on Yocto [7], with a set of tools that allows the creation of customised embedded systems. This distribution is responsible for supporting the applications developed for the configuration and management of the hardware platform.

V. DATA FLOW

To configure the system, the user accesses through a CLI (Command Line Interface) in a client-server architecture that facilitates the introduction and modification of key parameters for the operation of the equipment (Fig. 5).

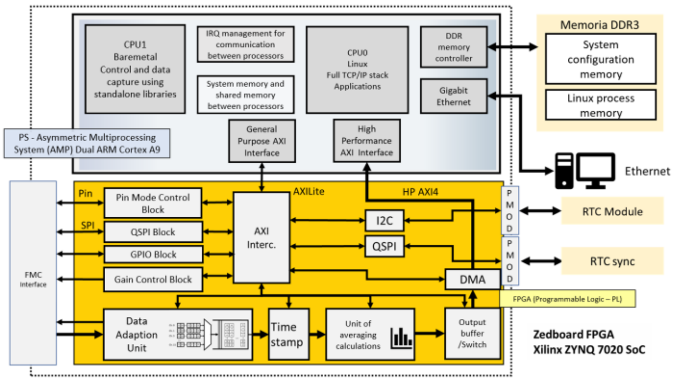

CPU1
Baremetal
Control and data capture using standalone libraries
IRQ management for communication between processors
System memory and shared memory between processors
CPU0
Linux
Full TCP/IP stack
Applications
DDR memory controller
Gigabit Ethernet
Memoria DDR3
System configuration memory
Linux process memory
PS - Asymmetric Multiprocessing System (AMP) Dual ARM Cortex A9
General Purpose AXI Interface
High Performance AXI Interface
Ethernet
Pin
Pin Mode Control Block
SPI
QSPI Block
GPIO Block
Gain Control Block
AXI Interc.
AXILite
HP AXI4
I2C
QSPI
DMA
PMOD
PMOD
RTC Module
RTC sync
FMC Interface
FPGA (Programmable Logic – PL)
Data Adaption Unit
Time stamp
Unit of averaging calculations
Output buffer /Switch
Zedboard FPGA
Xilinx ZYNQ 7020 SoC

Figure 3. FPGA SoC system architecture overview diagram

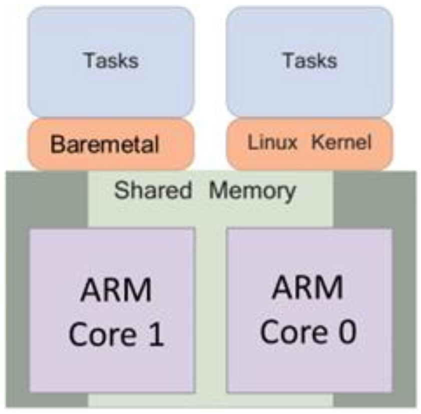
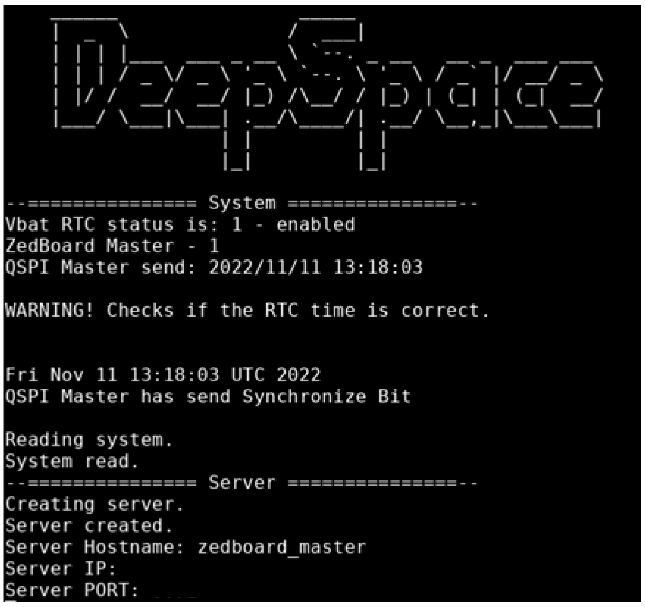


Figure 4. Asymmetric Multi-Processing System



```
--=============== System ===============--
Vbat RTC status is: 1 - enabled
ZedBoard Master - 1
QSPI Master send: 2022/11/11 13:18:03

WARNING! Checks if the RTC time is correct.


Fri Nov 11 13:18:03 UTC 2022
QSPI Master has send Synchronize Bit

Reading system.
System read.
--=============== Server ===============--
Creating server.
Server created.
Server Hostname: zedboard_master
Server IP:
Server PORT:
```

Figure 5. Server Booting process

CPU0 reads the data from the OCM (On-Chip Memory) shared memory and writes the necessary files to the Linux file system for remote storage. For this purpose, it manages the creation of the corresponding files using the drivers available on the system. Fig. 6 shows the data flow of the captured samples through the processing system.

CPU0 is also responsible for running the server that facilitates the configuration and control of the platform from a remote client. The user can define the SPI configuration parameters, the gain, the number of samples, etc. The data obtained by the server from the client is sent to the hardware by means of the corresponding designed drivers.

For security reasons, the parameters cannot be changed by direct access to the equipment, but the configuration parameters are sent from the client to the server running on the CPU0 processor. The server checks the allowed value ranges and the state of the system. This provides controlled access to the system blocks. Communication between the PS and the PL takes place via shared memory, where data is transmitted via a DMA (Direct Memory Access), controlled by CPU1. Therefore, CPU1 only writes to the shared memory, while CPU0 only reads from the shared memory. This communication is restricted to a specific area of system memory that has a high data transfer (128 bits per access).

The transmission of the data processed by the platform is sent via the Ethernet interface with a transmission speed of 1 Gb/s. It should be noted that this data must be taken on the Ethernet frame for an interface in raw mode.

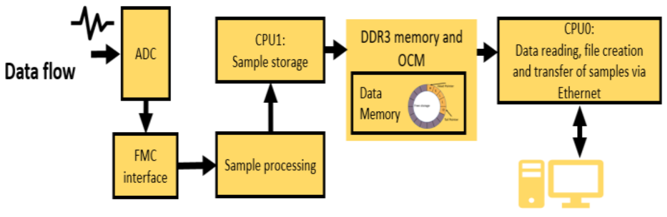


Figure 6. Data flow of captured samples

About the storage of the captured data, it was decided to send it directly to a storage server via the NFS protocol. The decision is justified because the system generates a significant amount of captured data, especially when sampling at 256 KSPS for long periods of time.

VI. Main Application

The application that controls this client-server architecture, write in C/C++, can be seen in Fig.7. It starts up at the beginning of the system, where it calls the functions responsible for each layer for the initialisation of the server, the initial configuration of the system and the ADC, and performs the first reading for the creation of the configuration file that contains the system status. The server starts up with the IP assigned to the equipment via DHCP and remains in listening mode on the assigned ports. In addition, it starts the power supplies for the rest of the system (ADC's) in the planned sequence and synchronises the clocks of the master and slave cards. This application is divided into four layers:

1. Main layer: includes the main function of the program and is responsible for calling the functions that initialise the server, the initial configuration of the system and the configuration of the ADC. In addition, it performs a first read of the system to create a status file.
2. Communication layer: handles TCP/IP connections and communications between client and server.
3. Command management layer: receives commands and verifies that they are correct or can be processed, generating error messages if they are not.
4. Logic layer: handles all the application logic, i.e., it processes the data received from the commands, validates them, and communicates with the lower layer to carry out the configuration of the commands received. This layer is supported by different Linux drivers, developed for this application and low-level functions, and designed to implement the necessary functions to control the logic of the AD and the hardware system implemented on the PL of the Zynq device.

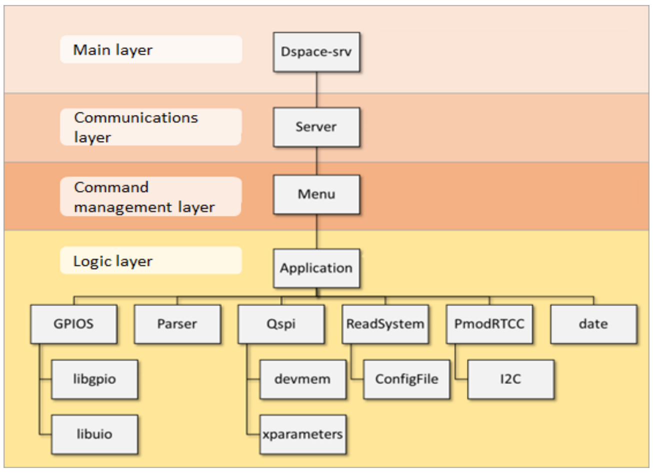


Figure 7. Server application architecture

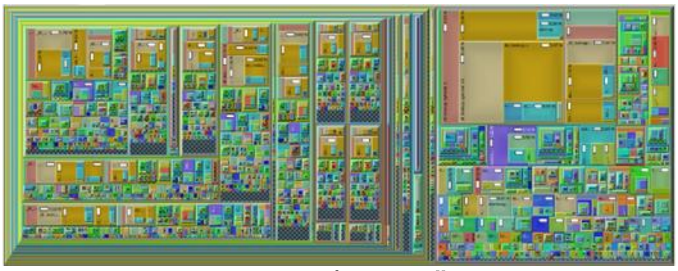


Figure 8. Server functions call map

VII. Profiling

The last step in system validation is the profiling stage. In this stage it is necessary to use profiling tools to detect those errors that have occurred during the programming stage. This step has been carried out with different tools such as Valgrind and KCachegrind (Fig. 8) for the study of memory usage and the dd command for the analysis of network bandwidth.
The data obtained with the dd tool shows that the maximum transfer speed that can be achieved on the system is approximately 50 MB/s. This is far from the manufacturer's specified speed of 125 MB/s (1 Gb/s), so this is a bottleneck for transferring a large data stream [8] at very high capture rates (Fig. 9).

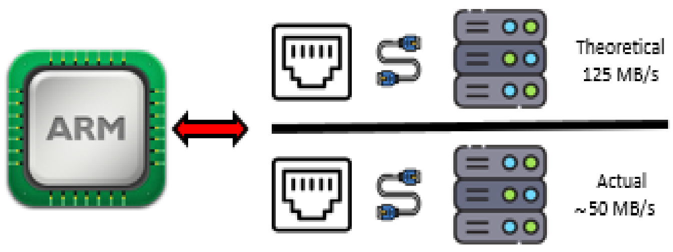


Figure 9. Theoretical transfer speed vs. actual transfer speed

VIII. Conclusion

The implementation of a Linux-baremetal AMP system for the integrated management of the system has been successfully realised. The baremetal part controls the data capture and communicates with the Linux part for data processing. The latter in turn, manages the system resources in an optimal way and handles them responsibly depending on the state of the system. At high capture speeds, bottlenecks can occur in the Ethernet transfer, therefore, solutions such as applying jumbo frames in the NFS protocol and investigating those kernel variables related to the TCP/IP protocol that can generate a detriment in the performance of the Ethernet connection are under study.

References

[1] IAC. (n.d.). *QUIJOTE | Instituto de Astrofísica de Canarias • IAC*. Retrieved March 7, 2022, from https://www.iac.es/es/proyectos/quijote

[2] Analog Devices Inc. (2021). *AD7768/AD7768-4*. https://www.analog.com/media/en/technical-documentation/data-sheets/ad7768-chips.pdf

[3] Avnet Electronics Marketing. (2012). *ZedBoard Zynq^{TM} Evaluation and Development Hardware User's Guide*. https://digilent.com/reference/_media/zedboard:zedboard_ug.pdf

[4] Xilinx Inc. (2014). *Zynq-7000 All Programmable Software Developers Guide*. *821*, 93. http://www.xilinx.com/support/documentation/user_guides/ug821-zynq-7000-swdev.pdf

[5] Xilinx. (n.d.). *OpenAMP - Xilinx Wiki - Confluence*. Retrieved June 11, 2022, from https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841718/OpenAMP

[6] Xilinx Inc. (2020). *PetaLinux Tools Documentation Reference Guide*. https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf

[7] Linux Foundation. (2022). *Software – Yocto Project*. https://docs.yoctoproject.org/index.html

[8] Moorthy, P., & Kapre, N. (n.d.). A Case for Embedded FPGA-based SoCs in Energy-Efficient Acceleration of Graph Problems.