DEVELOPMENT OF A SPACEWIRE/RMAP-BASED DATA ACQUISITION FRAMEWORK FOR SCIENTIFIC DETECTOR APPLICATIONS

Session: Missions & Applications

Short Paper

Takayuki Yuasa, Kazuhiro Nakazawa, Kazuo Makishima,
The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo, Japan

Hirokazu Odaka, Motohide Kokubun, Takeshi Takashima, Tadayuki Takahashi
Department of High Energy Astrophysics, Institute of Space and Astronautical Science (ISAS), Japan Aerospace Exploration Agency (JAXA), 3-1-1 Yoshinodai, Sagamihara, Kanagawa 229-8510, Japan

Masaharu Nomachi,
Laboratory of Nuclear Studies, Graduate School of Science, Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043

Iwao Fujishiro, and Fumio Hodoshima
Shimafuji Electric Incorporated, 8-1-15 Nishikamata, Ota, Tokyo, Japan 144-0051

E-mail: yuasa@amalthea.phys.s.u-tokyo.ac.jp, nakazawa@phys.s.u-tokyo.ac.jp, maxima@phys.s.u-tokyo.ac.jp, odaka@astro.isas.jaxa.jp, kokubun@astro.isas.jaxa.jp, ttakeshi@stp.isas.jaxa.jp, takahasi@astro.isas.jaxa.jp, nomachi@fn.lns.sci.osaka-u.ac.jp,fujishiro@shimafuji.co.jp, hodo@shimafuji.co.jp

ABSTRACT

We have been developing a multipurpose modularized data acquisition framework based on SpaceWire and Remote Memory Access Protocol (RMAP). The framework consists of a SpaceWire/RMAP library that runs on a small size computer SpaceCube, and a template hardware description language code for an FPGA which is used on several kinds of circuit boards with SpaceWire protocol stack. Users need to implement a VHDL module which receives data from their detectors and sends commands to them. Data transfer is done by accessing user defined registers or memory blocks in the FPGA from the SpaceCube user program. In this paper, we present an overview of this framework, and describe an example of its implementation in our gamma ray imager experiment is also presented.

1 A DATA ACQUISITION FRAMEWORK ON SPACEWIRE/RMAP

Remote Memory Access Protocol (RMAP) provides standard read/write methods over SpaceWire connection. We have been developing a data acquisition (DAQ) framework for scientific detectors based on SpaceWire and RMAP. As shown in figure 1, this DAQ framework consists of a small size computer called SpaceCube, a circuit board, and software libraries for them. Below are detailed explanations of SpaceCube, circuit boards, and the prepared framework.

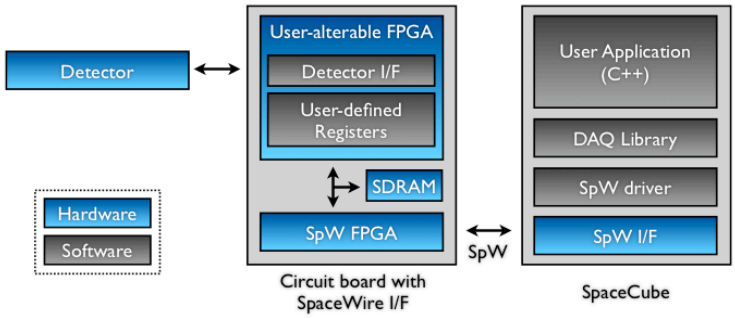

Figure 1 The block diagram of the DAQ framework. Users have to code a detector I/F module and user-defined registers in VHDL, and construct a user application in C++.

1.1 SpaceCube

SpaceCube is a small size (52x52x55 mm) computer with SpaceWire I/Fs developed by Shimafuji Electric, Inc. and JAXA. Figure 2 shows its appearance, and its major characteristics are listed in table 1. SpaceCube is operated by The Real-time OS Nucleus (TRON) which is widely employed as an operating system for embedded systems. In this DAQ framework, a user application which runs on SpaceCube works as a data/command handler and recorder. It fetches data and sends commands from/to a user-alterable FPGA via RMAP. SpaceWire protocol stack (an IP core and an I/F driver) is provided by NEC Software, Ltd. Our RMAP library is written in C++ and provides users with simple operations, such as *read(address)* and *write(address, data),* to access their boards.

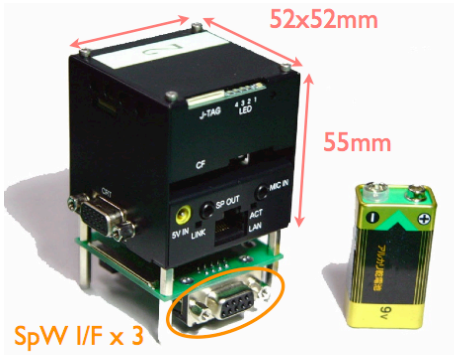

Figure 2 An appearance of SpaceCube. Right is a 006P type battery for size comparison.

Table 1 Specification of SpaceCube.

CPU	VR5701(MIPS core) 200, 250, 300 MHz
RAM	DDR SDRAM 64 MB
Flash Memory	16MB
I/F	SpaceWire x 3, CF, XGA, USB, LAN, Stereo Audio, RS232C
Power	+5 V
Size	52x52x55 mm

1.2 Circuit Boards with SpaceWire I/F and the Internal Structure of User-alterable FPGA

We have been developing a several kinds of circuit boards with SpaceWire I/F, for such purpose as analogue-digital conversion, digital-analogue conversion, and digital input/output. Each circuit board is equipped with a SpaceWire/RMAP protocol stack FPGA (hereafter, SpaceWire FPGA), a user-alterable FPGA, SDRAM, and circuit for independent functionality. Figure 3 is an appearance of an example of digital input/output board. Between the two FPGAs, a simple data bus (hereafter we call External Bus) is defined. An RMAP engine, implemented in SpaceWire FPGA, interprets an RMAP command packet, and then access the user-alterable FPGA's address space via External Bus if the requested address is of that FPGA. The SDRAM, which is connected to SpaceWire FPGA, is also accessible from SpaceCube and the user-alterable FPGA via SpaceWire and External Bus, respectively.

As shown in figure 4, a user-alterable FPGA consists of function modules connected to an on-chip bus. The on-chip bus provides a standard way for data transfer among modules, and enables SpaceWire FPGA to access each modules' address space via External Bus and on-chip bus (figure 4). In VHDL, we have written several template functional blocks for

the user-alterable FPGA, such as an on-chip bus arbiter, a bus interface module, a data-buffering module, and an External Bus adapter module. Users can construct their own read-out chip by connecting those modules.

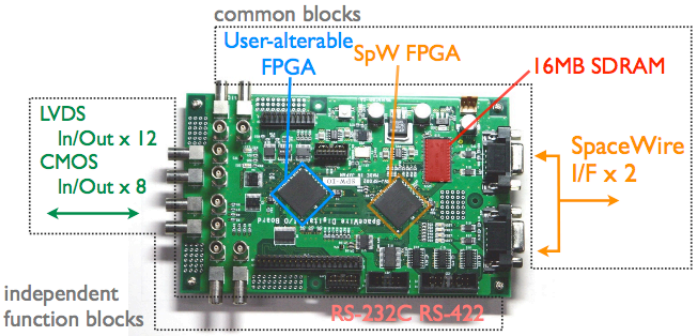

Figure 3 An appearance of a digital input/output board. Two FPGAs (Xilinx Spartan-3, XC3S1000 for SpaceWire FPGA and XC3S400 for the user-alterable FPGA) are used to separate commonly needed SpaceWire/RMAP functionality and user dependent blocks.

The whole framework structure is modularized in both software and hardware. Especially, in the user-alterable FPGA, all functional blocks, including a detector I/F module which should be coded by each user, only communicate with on-chip bus I/F modules as shown in figure 4. Therefore the portability of user-coded modules is reasonably high. This allows users to use the same modules both in their bread board models and in the flight model, and hence to reduce costs for the entire projects.

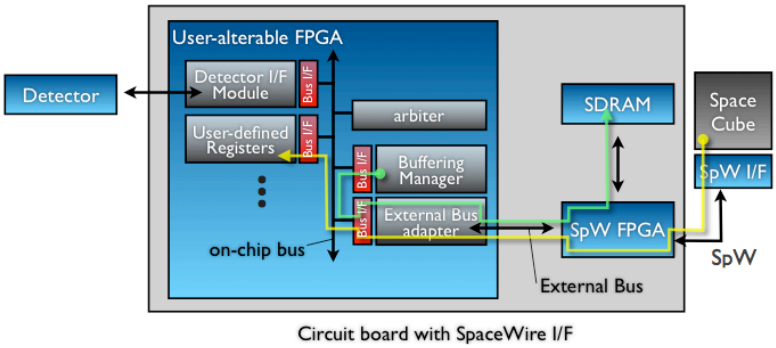

Figure 4 The internal structure of a user-alterable FPGA. For instances, yellow and green arrows represent routes of access from SpaceCube to user-defined registers in the user-alterable FPGA, and from Buffering Manager in the FPGA to the SDRAM, respectively.

2 An Example of Implementation of the Framework

To verify the performance of the developed framework, we experimentally employed it as a read-out system for a gamma ray imaging detector, which has been developed by our group [1]. The detector consists of pixelated inorganic scintillator array (4x4x10mm/pixel, 10x10=100pixels), viewed by a 256 channel multi-anode photo multiplier tube (PMT) and a 12 bit ADC unit. The AD converted signal is received by a digital input/output board, then read out by SpaceCube. A block diagram and a total picture of the system are shown in figure 5 and 6, respectively.

When a gamma ray photon is absorbed in one of the scintillation crystals, produces charge pulse in some or all of the 256 anode channel of the PMT. The analogue signals from the 256 channel PMT anodes are converted to serial digital signals in the 12 bit ADC unit. In the user-alterable FPGA, we have developed a particular detector I/F module for the ADC unit (De-serializer in figure 5). It de-serializes the data transferred from the ADC unit, and stores them into the SDRAM across the on-chip bus , an SDRAM buffering manager, and External Bus. The size of an event is

(ADC 12 bit + Header 4 bit) * 256 ch + Time (32 bit) = 4128 bit = 516 bytes.

The stored data are transferred from the SDRAM to SpaceCube via SpaceWire, and then sent to recorder PC as TCP/IP packet. Spectral and image analyses are done on an offline environment with the ROOT system. From the 256 pulse-height data, we can reconstruct the gamma-ray energy, and determine the interaction position as their centers of gravity.

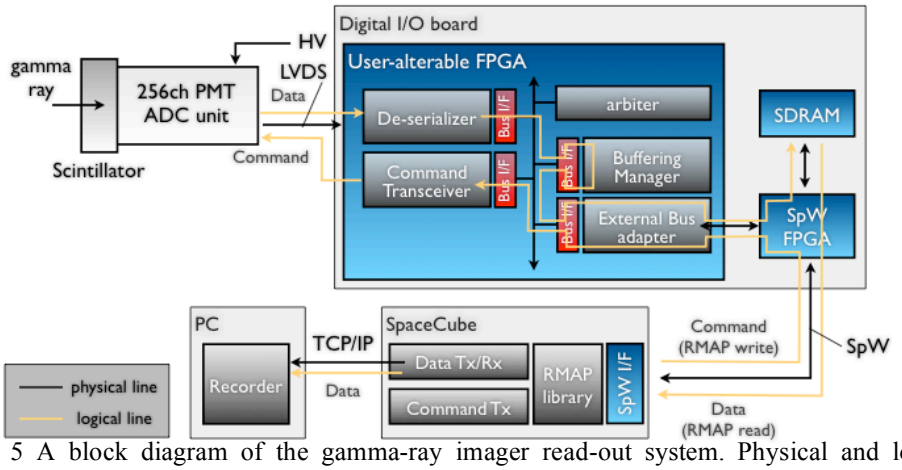

Figure 5 A block diagram of the gamma-ray imager read-out system. Physical and logical connections between blocks are shown in black and orange arrows, respectively.

We irradiated gamma rays for radioisotope ^{137}Cs to the whole area of the imager. The DAQ framework worked well and gamma ray event data were successfully transferred via SpaceWire/RMAP. Configurable parameters of the ADC unit (trigger mode, trigger threshold, peaking time, etc) were also properly controlled according to change commands from SpaceCube. Figure 7 shows a reconstructed image of 500,000 gamma ray events (~250 MBytes). In the image, each bright point corresponds to single scintillator pixel. All of 100 pixels, each 4x4 mm^2, have been successfully resolved.

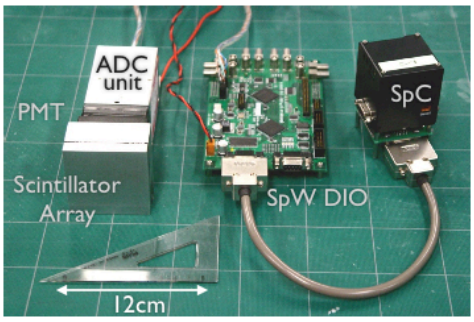

Figure 6 A photograph of our gamma-ray imager (left), digital input/output board (center),, and SpaceCube (right).

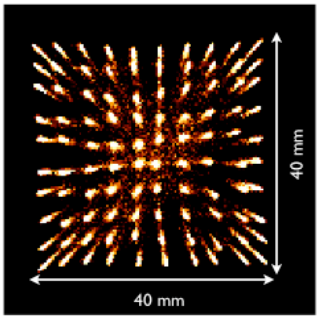

Figure 7 A reconstructed image of 5x10^5 gamma ray events from ^{137}Cs.

3 The Framework in Other Experiments

This DAQ framework based on SpaceWire and SpaceCube is planned to be used in such science missions as,

PoGO : Balloon-borne gamma-ray polarimeter experiment by SLAC, Hiroshima U., & JAXA
HEFT : Balloon-borne hard X-ray imaging experiment by JAXA, Cal Tech et al.
SDS-I SWIM : SpaceWire I/F verification Module on a Japanese small satellite by JAXA& UT
NeXT : Next Japanese cosmic X-ray Satellite by JAXA et al.
TEM : Transmission Electron Microscope with X-ray micro calorimeter by JAXA et al.

---

[1]. M. Kokubun and S. Hirakuri et al., “Development of an Active Gamma-ray Imaging Spectrometer with Pixelated Scintillators”, IEEE NSS-MIC 2006, October 29 - Nov. 4, 2006, at San Diego