A CPU MODULE FOR A SPACECRAFT CONTROLLER WITH HIGH THROUGHPUT SPACEWIRE INTERFACES

Session: Onboard Equipment & Software

Short Paper

Toru Sasaki, Minoru Nakamura, Tadashi Yoshimoto, Minoru Yoshida,

and Shoji Yoshikawa

Advanced Technology R&D Center, Mitsubishi Electric Corporation, 8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, 661-8661, Japan

E-mail: Sasaki.Toru@eb.MitsubishiElectric.co.jp,

Nakamura.Minoru@ea.MitsubishiElectric.co.jp,

Yoshimoto.Tadashi@ap.MitsubishiElectric.co.jp,

Yoshida.Minoru@ea.MitsubishiElectric.co.jp,

Yoshikawa.Shoji@ap.MitsubishiElectric.co.jp

ABSTRACT

We have developed a CPU module for a spacecraft controller with high throughput SpaceWire interfaces. This CPU module is a main module of the spacecraft controller which executes AOC(Attitude and Orbit Control) and DH(Data Handling) processing. The amount of data from payload subsystems has been increasing in recent spacecrafts, such as earth observing spacecrafts. With that in mind, we selected SpaceWire for its high data transfer rate. To accelerate data transfer throughput without requiring excessive CPU resources, we have developed a SpaceWire controller and a DMA controller. We have made a functional model of the CPU module and evaluated its performance for data transfer. This paper discusses the architecture of the CPU module and the results of its evaluation.

1 INTRODUCTION

To develop a spacecraft controller in a short time and at low cost, we have recognized the importance of establishing a design standard. SpaceWire is becoming a standard for networks of onboard satellites all over the world. In Japan, too, a spacecraft controller equipped with SpaceWire has already been developed [1]. The flexibility and high-throughput of SpaceWire makes it very attractive for the network in our spacecraft controller. The first step in our program for adopting SpaceWire was developing the CPU module for the spacecraft controller, which is designed for the small and medium size satellites. This CPU module uses DMA to accelerate the data transfer through the SpaceWire. Here, we introduce this spacecraft controller and its CPU module, in particular.

2 SPACECRAFT CONTROLLER

Fig. 1 shows a block diagram and picture of the spacecraft controller. The CPU module's main functions are AOC and DH. Sensors, such as thermistors, star sensors and Fiber Optic Gyros (FOGs), are connected to the Sensor I/F module, which uses an FPGA and ADC to gather digital and analog data from the sensors. The I/O module interfaces with actuators such as reaction wheels. The Data Recorder module has an FPGA and SDRAMs for storing data from pay-load subsystems and housekeeping data. A soft-core CPU is embedded in each FPGA for controlling of synchronization and interfaces between modules.

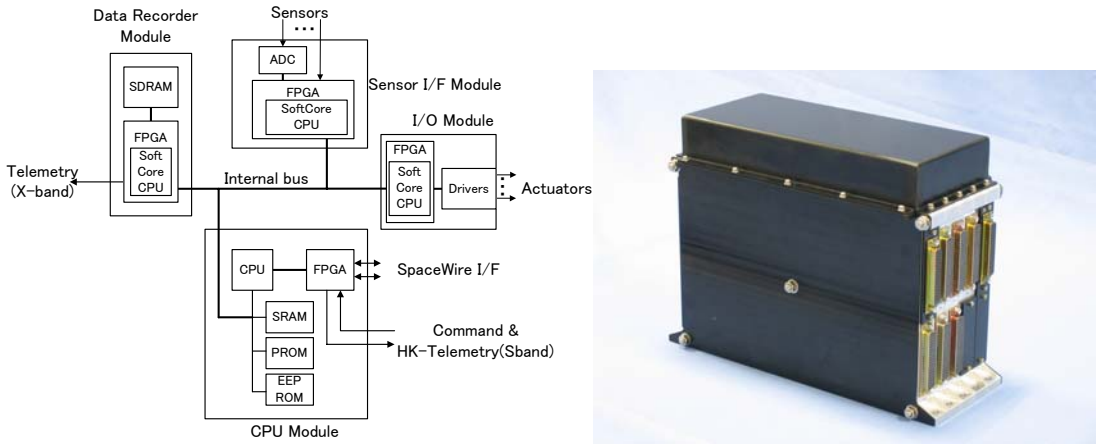

FIG.1. The block diagram and the picture of the spacecraft controller
The size is 310(W)×116(D)×236(H)mm and the weight is 5 kg

3 CPU MODULE

Table 1 gives the specification of the CPU module and Fig. 2 shows the picture of the CPU module. The CPU module is composed of HR5000 CPU, which is developed by JAXA, memories and an Anti-Fuse FPGA. The CPU and the FPGA are connected with PCI bus. Fig. 3 shows a function block diagram of the FPGA. The SpaceWire Controller manages 2-channel SpaceWire interfaces which are supposed to be connected to mission payloads. The DMA Controller has 4 channels for sending and 4 channels for receiving to handle the data transfer of SpaceWire and other peripherals. The FPGA has also a PCI Bus Bridge Controller, a WatchDogTimer, a Timer, a GPIO an Interrupt Controller and a Serial Interface Controller. The PCI Bus Bridge Controller controls the PCI bus between HR5000 and the FPGA.

We describe the specification of the DMA Controller, which has two special functions to enhance the throughput of the SpaceWire. Firstly, this DMA Controller has two sets of control registers, count registers, and destination or source address registers in each channel. This design enables software to configure one set of registers while the DMA controller is transferring data under the other configured set of registers(double register mode). Therefore we can reduce the delay for DMA restart. Secondly, the DMA Controller detects EOP(End of Packet) of SpaceWire packet. The burst-length can be set 1, 2, 4 or 8 in word (4 bytes), but the length of the SpaceWire packet may not be a multiple of the burst-length. This situation might cause the DMA Controller to wait for all the data being filled in the burst length. To

prevent the DMA Controller from waiting, the DMA Controller can detect EOP and can send the residues separately.

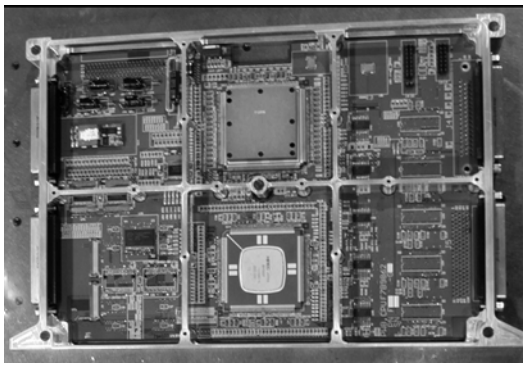

FIG.2. The picture of the CPU module. The two largest ICs in the CPU module are an Anti-Fuse FPGA(upper) and HR5000 CPU(lower).

Table 1. Specifications of the CPU module

CPU	HR5000 (core 100MHz bus 50MHz)
FPGA	RTAX2000(Actel)
SpaceWire Interface	2ch
System Memory	Asynchronous SRAM(2MB) with ECC
Power Consumption	4.2W

To transfer data by using the DMA Controller, software sets a DMA count register and a source or destination address register and selects the burst-length. Then the DMA Controller begins DMA transfer once the software sets a start-bit of a DMA control register. If we use the double register mode, we can set another registers and keep the channel ready for the next DMA transfer while executing the former DMA transfer. DMA transfer is executed until the detection of EOP or the expiration of a DMA count register. Transfer completion is notified by an interrupt or a flag in the DMA status register.

We also developed software which works on the CPU module and manages the transfer of the data by using SpaceWire Controller and DMA Controller. The software uses an original operating system(OS). The device drivers in the OS handle the controllers in the FPGA.

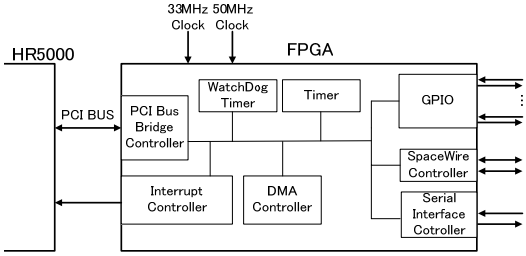

FIG.3. FPGA function block diagram

4 EVALUATION

Our evaluation of the CPU module was especially focused on the effect of the double register mode. We compared the data transfer rate using the double register mode with the rate using a single register(single register mode) and examined the effects of varying the length of SpaceWire packets.

For this evaluation, a SpaceWire interface was connected to a loopback wiring. And we made the CPU module to send data from the system memory and to receive the same data through the loopback. We measured the time for the completion of transferring data. The link speed of SpaceWire was set 10Mbps. The result is shown

in Fig. 4. We found that the transfer rate by the double register mode was enhanced up to about 1.5 times faster. At the length of 384 bytes, the data transfer by the double register mode was 7.01Mbps. The logical data transfer rate of the SpaceWire with 10Mbps link speed is estimated as about 8Mbps. Hence our controller is said to reach about 88% of the maximum performance of SpaceWire with 10Mbps link speed.

5 DISCUSSION

We can see a drop around the short-length packet in Fig. 4. It indicates that the double register mode does not increase the efficiency so much when transferring short-length packets. We suppose that the problem is in software's overhead in re-setting the DMA registers. If the overhead is quite large, the former DMA transfer will complete before software starts up the next DMA transfer. This situation reduces the performance of the double register mode because the double register mode needs a little more complicated process in the completion and start of a DMA than the single. We expect that we can improve the overhead by optimizing the software and get better efficiency of the double register mode around shorter length packets.

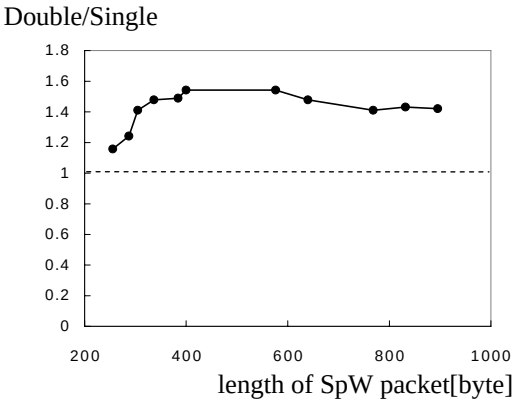

FIG. 4. The ratio of the double to single register mode in transfer rate as a function of length of SpW packet.

6 CONCLUSION

We have developed a CPU module with SpaceWire interfaces for a spacecraft controller. The data through the SpaceWire is managed by a SpaceWire Controller and a DMA controller in an FPGA. The DMA controller has two sets of registers for configuration and detects the arrival of EOP to promote the throughput of the data transfer. The measurement of data transfer showed that using the double register mode enhanced the throughput by 50% at its best, compared to the single register mode. And we found that we need the improvement of the software which handling the double register mode to increase the efficiency of transferring short-length packets. Moreover we will implement the RMAP protocol or the SpW-RT(Spacewire Real Time) protocol in the CPU module. We plan to report the status of these work on the next Conference.

7 REFERENCES

1. Tadayuki Takahashi, Takeshi Takashima, Seisuke Fukuda, Satoshi Kuboyama, Masaharu Nomachi, Yasumasa Kasaba, Takayuki Tohma, Hiroki Hihara, Shuichi Moriyama, Toru Tamura, "Space Cube 2 – An Onboard Computer Based on SpaceCube Architecture", International SpaceWire Conference 2007, Dandee, UK Sep. 17th-19th, 2007.