SPACEWIRE RMAP IP CORE

Session: SpaceWire Components

Long Paper

Steve Parkes, Chris McClements, Martin Dunstan

University of Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

Wahida Gasti

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: sparkes@computing.dundee.ac.uk, cmcclements@computing.dundee.ac.uk, mdunstan@computing.dundee.ac.uk, wahida.gasti@esa.int

ABSTRACT

The SpaceWire Remote Memory Access Protocol (RMAP) provides a standard mechanism for reading from and writing to memory in a remote SpaceWire node. This simple but powerful capability is already being designed into components like the SpW-10X router and missions like Bepi Colombo and MMS.

The development of a generic IP core implementing the RMAP protocol will enable users to readily implement the RMAP protocols in FPGAs or ASICs, customising it to their specific mission needs. The RMAP IP core is being developed by University of Dundee for ESA. This paper describes the main features of the RMAP IP core, its architecture, interfaces and initial performance.

1 INTRODUCTION

1.1 SPACEWIRE

SpaceWire is a communications network for use onboard spacecraft. It is designed to connect high data-rate sensors, large solid-state memories, processing units and the downlink telemetry subsystem providing an integrated onboard, data-handling network. SpaceWire links are serial, high-speed (2 Mbits/sec to 200 Mbits/sec or higher), bi-directional, full-duplex, point-to-point data links which connect together SpaceWire equipment. Application information is sent along a SpaceWire link in discrete packets. Control and time information can also be sent along SpaceWire links. SpaceWire is defined in the European Cooperation for Space Standardization ECSS-E50-12A standard [1]. It is being used on many space missions.

1.2 RMAP

The remote memory access protocol (RMAP) [2] provides a means for one SpaceWire node to write to and read from memory inside another SpaceWire node. The aim of RMAP is to standardize the way in which SpaceWire units are configured and to provide a low-level mechanism for the transfer of data between two SpaceWire nodes. For example RMAP may be used to configure a camera or a mass memory device. The camera device may then write image data to allocated areas of memory in the mass memory, or the mass memory may read image data from the camera.

RMAP provides three commands: read, write and read-modify-write:

- The read command reads one or more bytes of data from a specified area of memory in a destination node. The data read is returned in an RMAP reply packet.

- The write command writes one or more bytes of data to a specified area of memory in a destination node. An acknowledgement may be returned to the initiator of the write command if requested in the command.

- The read-modify-write command reads a register (or memory) returning its value and then writes a new value, specified in the command, to the register. A mask can be included, in the command, so that only certain bits of the register are written. This may be used to provide a variety of semaphore and handshaking operations.

The RMAP standard is currently going through the European Cooperation for Space Standardization (ECSS) review and approval process and should be issued formally in the first quarter 2009. In the meantime the draft specification has been used for several devices and missions.

1.3 IP CORES

SpaceWire has been adopted for use on many space missions partly because of the ready availability of intellectual property (IP) cores, components, software drivers, and development support and test equipment. A SpaceWire CODEC designed by the University of Dundee and implemented in VHDL is available as an IP core from ESA for European space projects [3]. This CODEC has been designed into several SpaceWire chips including the SpW-10X SpaceWire router chip designed by University of Dundee and Austrian Aerospace and implemented in an Atmel radiation tolerant ASIC [4]. A wide range of development support and test equipment is available from STAR-Dundee Ltd [5] and other organizations.

With the development of the RMAP standard there is a need for an IP core that implements the RMAP protocol. This core needs to be configurable to suit many different applications.

2 RMAP IP CORE CONTEXT

The RMAP IP core fits between User logic and the SpaceWire interface as illustrated in Figure 1. There are two types of RMAP IP core:

- The one that sends out RMAP commands and receives any replies, which is referred to as the Initiator RMAP Interface
- The one that receives RMAP commands executes them and sends out any required replies, which is referred to as the Target RMAP Interface.

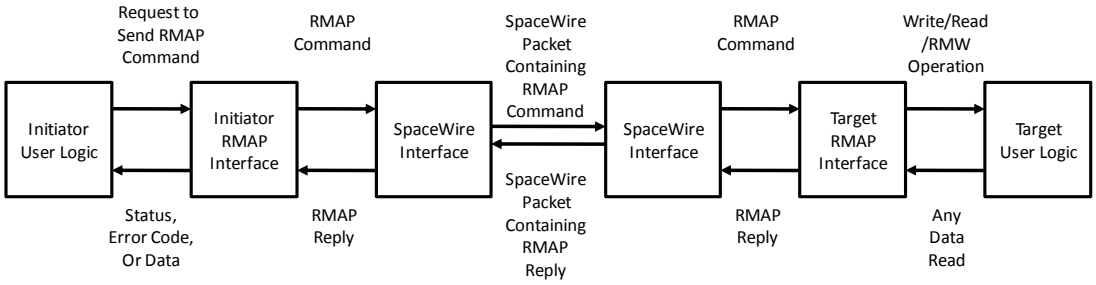

Figure 1 RMAP Initiator and Target

The initiator user logic prepares the command along with any associated data for a write or read/modify/write (RMW) command. In the case of a read command the initiator user logic also prepares information detailing where the data from the read reply is to go. Once all this information is ready the initiator user logic instructs the initiator RMAP interface to send the command. The initiator RMAP interface reads the relevant command and data information from the user logic, forms the RMAP header, generates header and data CRCs and then sends the command. It will also reject the command if the information provided does not correspond to a valid RMAP command. The RMAP command is sent in a SpaceWire packet over the SpaceWire interface.

The SpaceWire packet is received by the SpaceWire interface at the target. If the packet is an RMAP command packet it will be passed to the target RMAP interface which checks that it is valid and then executes the command, writing to or reading from target user logic. In the case of a write command the RMAP data field is written into memory or registers within the target user logic. If an acknowledgment to the write command has been requested then this will be formed by the target RMAP interface and sent out of the SpaceWire interface. In the case of a read command data is read from target user logic memory or registers by the RMAP interface and returned in a reply packet via the SpaceWire interface.

Reply packets travel across the SpaceWire fabric back to the initiator of the original command. They are received by the SpaceWire interface and (if they are RMAP replies) are passed to the initiator RMAP interface. Replies to write commands may signal to the initiator user logic that the command has been executed. Replies to read commands will write the data read from the target user logic to the required designation in the initiator user logic.

A node can contain both an initiator RMAP interface and/or a target RMAP interface i.e. be able to both send and receive commands. An initiator/target is illustrated in Figure 2.

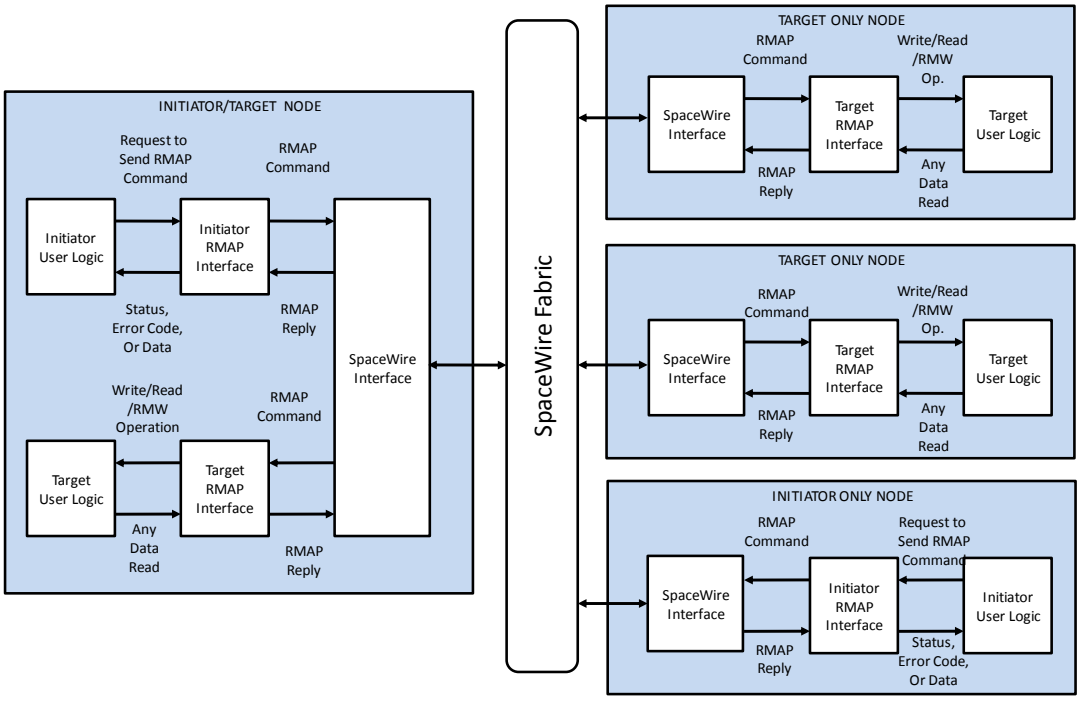

Figure 2 System using RMAP Initiator/Target

3 RMAP IP CORE ARCHITECTURE

The architecture of the RMAP IP core is illustrated in Figure 3.

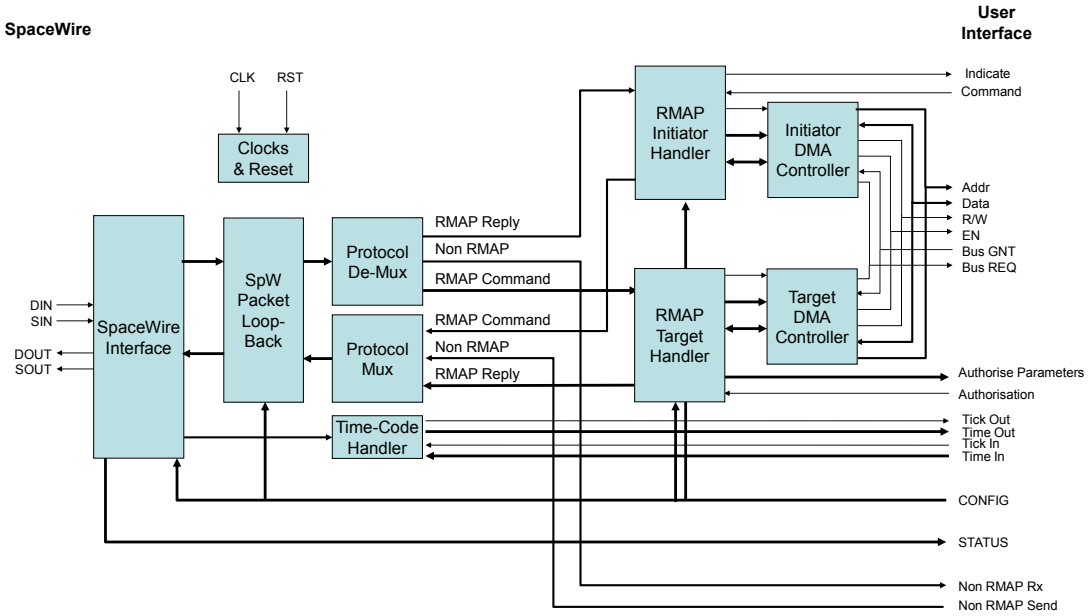

Figure 3 RMAP IP Core Architecture

The SpaceWire Interface is responsible for receiving SpaceWire packets and time-codes and passing them to the SpaceWire Loop-Back and Time-Code Handler respectively. It also transmits SpaceWire packets when requested to do so by the

SpaceWire Loop-Back. The SpaceWire Interface is configured by CONFIG inputs and SpaceWire link status information is made available on the STATUS outputs.

The SpaceWire Loop-Back block provides a means of looping back SpaceWire data characters, EOPs and EEPs. When Loop-Back is disabled, received SpaceWire packets are passed to the Protocol De-Multiplexer and SpaceWire packets from the Protocol Multiplexer are passed to the SpaceWire interface for transmission. When Loop-Back is enabled, received SpaceWire packets are passed immediately to the SpaceWire transmitter, and SpaceWire packets from the Protocol Multiplexer are passed immediately to the Protocol De-Multiplexer. Time-codes are not affected by the SpaceWire Loop-Back block.

The Protocol Multiplexer is responsible for deciding which SpaceWire packet is to be passed to the SpaceWire Interface via the SpaceWire Loop-Back for transmission. There are three sources of SpaceWire packet for transmission: RMAP Initiator Handler which sends RMAP commands, RMAP Target Handler which sends RMAP replies and the non-RMAP interface which may provide non-RMAP packets for transmission.

The Protocol De-Multiplexer is responsible for deciding where received SpaceWire packets are to go: the RMAP Initiator Handler receives RMAP replies, the RMAP Target Handler receives RMAP commands and the non-RMAP interface is passed any other packets.

The RMAP Initiator Handler is responsible for generating and sending RMAP commands based on information provided in user memory. When appropriate it will wait for any reply to the command and inform the initiator user application that the RMAP operation has completed or failed. Any data received with a reply is placed in memory specified by the user application when the RMAP command was generated. Multiple outstanding transactions are permitted.

The Initiator DMA controller provides an interface to user memory for reading RMAP commands from memory that are to be sent and for writing any data in RMAP replies to memory. It is responsible for gaining access to the user data bus and performing necessary memory read and/or write operations.

The RMAP Target Handler is responsible for checking and responding to valid RMAP commands. It will set up the Target DMA controller to perform reads and writes to user memory and registers and will form the reply to the RMAP command for sending by the SpaceWire interface. The RMAP Handler is configured by the CONFIG inputs and status information from the RMAP Handler is available on the STATUS outputs.

The Target DMA controller provides an interface to user memory and registers for writing data sent in an RMAP command to memory or reading from memory specified in an RMAP command. It is responsible for gaining access to the user data bus and performing memory read and/or write operations as determined by the RMAP command. The Target DMA controller competes with the Initiator DMA controller for access to the user data bus.

The Time-Code Handler is responsible for checking time-codes and maintaining the value of the time-code counter. It will assert the TICK_OUT signal when a valid time code is received and put the value of each valid time-code on the TIME-OUT output. The time-code handler can also generate time-codes using the TICK_IN and TIME-IN inputs.

The Configuration and Status registers hold configuration and status information for the RMAP IP core. On power up certain configuration registers are loaded with default values specified by the CONFIG interface. Thereafter the configuration values may be changed by writing to the configuration registers either by a SpaceWire-RMAP command or by the user logic writing to the appropriate registers. Status information from the RMAP IP core is held in status registers which can be read by SpaceWire-RMAP command or by the user logic. Certain status information is also available on dedicated signals, STATUS, from the RMAP IP core.

The Clock and Reset block is responsible for providing the user reset signal, RESET, to the relevant parts of the SpW/RMAP IP core ensuring a clean condition after the reset signal has been asserted. It is also responsible for generating any necessary clock signals from the single clock input signal, CLK.

4 RMAP IP CORE PERFORMANCE

The RMAP Target IP core has been extensively tested using an in-house test bench that performs extensive testing, covering many possible configurations and error conditions.

The RMAP Target IP core has been implemented in both Xilinx and Actel FPGAs. On the Xilinx Spartan 3 device the complete design related SpaceWire interface operates comfortably with link speeds of 200 Mbits/s. An initial implementation on an Actel AX1000 FPGA operates at 100 Mbits/s transmit speed and over 150 Mbits/s receive. Little effort has been spent on optimising either design for performance.

5 CURRENT AND FUTURE WORK

At present work is focussed on completing the RMAP Initiator design and testing this in Xilinx and Actel FPGAs. Two alpha customers are currently working with the RMAP Target IP core providing feedback on the design.

Some effort will be spent improving the performance of the Actel AX1000 implementation.

The RMAP IP core will be available from ESA for use on ESA projects only and from STAR-Dundee Ltd for use on any other project.

6 ACKNOWLEDGEMENTS

The authors acknowledge the support of ESA for the present work which is funded by ESA under ESA Contract No. 220774-07-NL/LvH.

7 REFERENCES

[1] ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

[2] ECSS “SpaceWire Protocols”, ECSS-E-ST-50-11C, Draft 1.3, July 2008

[3] C. McClements, S.M. Parkes, and A. Leon, “The SpaceWire CODEC,” International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands, November 2003.

[4] C. McClements, S. Parkes, G.Kempf, “SpW-10X SpaceWire Router User Manual”, Issue 3.4, July 2008 available from http://www.atmel.com/dyn/products/product_card.asp?part_id=4339

[5] www.star-dundee.com