DESIGN AND IMPLEMENTATION OF SYNTHESIZABLE SPACEWIRE CORES*

Session: SpaceWire Components

Short Paper

P. Aguilar-Jiménez, V. López, S. Sánchez, M. Prieto, D. Meziat

Space Research Group. Dpto. Automática. Universidad de Alcalá

E-mail: paguilarj@srg.aut.uah.es, mpm@srg.aut.uah.es, vlopezalvarez@gmail.com, chan@srg.aut.uah.es, meziat@aut.uah.es

ABSTRACT

The Space Research Group of the University of Alcalá is developing a library of IP cores to provide main space systems and subsystem SoC implementation. One of these is an ECSS-E-50-12A [1] based SpaceWire core covering the standard specification in two different approaches: a basic codec and a router core. Our goal is to achieve a technology independent synthesizable core optimized for timing and area.

In this paper, a brief description of the developed core is presented. After improving the synthesis results adding internal FPGAs, the resulting programming file is tested on different devices to check the post layout simulation results.

1 INTRODUCTION

Into the research areas of the Space Research Group (SRG) one of them is dedicated to de development of synthesizable IP cores of main spacecraft systems and subsystems. Some of these cores are based in the SpaceWire standard [1], giving us a deep knowledge of the standard and the technologies involved in the development of IP cores for space applications.

The SpaceWire Codec and Router here presented are intended to be on board of future missions and project at which SRG is currently involved.

2 SPACEWIRE CODEC

The SpaceWire Codec [1][2] has been developed accordingly to the ECSS-E50-12A Standard and it is planned to provide a maximum data rate close to 400 Mb/s [1]. Its design constraints are synthesizable, device independent and optimized for area and time resources what makes analysis a key role in the process leading to the working core. Its design is based in the well known block diagram shown in [2]:

* This work has been supported by the CICYT (grant ESP2005-07290-C02-02)

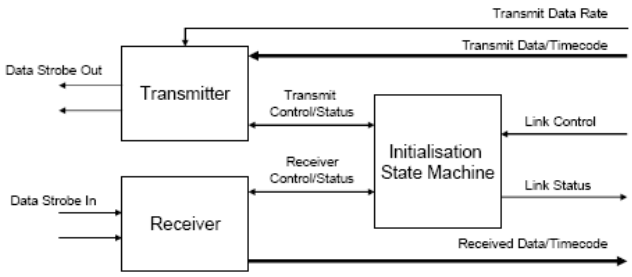

Figure 1. SpaceWire Codec block diagram.

Our approach is based in state machines for all three blocks. At the transmitter and receiver these state machines performs the serial to parallel conversion, flagging the different reception events to the main state machine and attending the transmit requirements issued by the reception buffer management and the host interface. Transmitter and receiver synchronization is based in the scheme described in [1] and in the size of the receive buffer, 56 N-chars.

One of the first concerns is that of dealing with high clock frequencies to provide as high data rates as possible. That is an affordable issue considering that the data signal of the standard's DS encoding holds one bit per each state of the recovered clock signal at the receiver [2], as shown in the next figure:

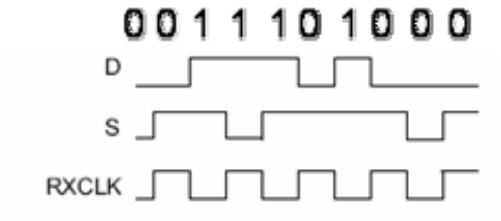

Figure 2. Receiver clock recovery.

Our proposal is base in the use of Double Data Rate for both transmitter and receiver. Using DDR registers give us the ability to manage clock frequencies half the desired data rate, with a 200 MHz top clock limit in the design. The cost of this advantage is an increased complexity of the design, working with two data flows, one for the even bits (b0, b2, b4, b6, b8) and another for odd ones (b1, b3, b5, b7, b9).

Strobe sequence generation at the transmitter is another challenging issue when optimizing design for area and timing. Our proposal is based in processing data stream as two independent data flows. Working with even and odd sequences, as stated before, strobe sequence is generated in two (odd and even) sequences too, using a quite simple algorithm based in the XOR logical properties and the receiver's XORing clock recovery scheme [1]. Taking into consideration that initial state for data and strobe signals is low ('0') and that first data bit transmitted after reset is a zero parity bit causing a high state transition on the strobe signal [1], it is clear that even bits are transmitted with clock's high state and odd bits with clock's low state. In that case, strobe even and odd sequences generation is as simple as

$$S_{par} = D_{par} \oplus 1 = \overline{D_{par}}$$
$$S_{impar} = D_{impar} \oplus 0 = D_{impar}$$

Both sequences are combined by a DDR register leading to the transmitted strobe signals. Additional delays due to the combinational logic are equalized using pipelining registers at the cost of one or two latency cycles at first bit transmission.

3 SPACEWIRE ROUTER

The SpaceWire Router implemented [1][3] is planned as a switching matrix to interconnect up to 224 SpaceWire codecs [1] rising up a SpaceWire network made of nodes (codec pairs) and at least one SpaceWire router, although complex topologies can easily grow up adding several routing switches. In its basic approach the router performs logical addressing and wormhole routing, maintaining a four nodes routing matrix as a constant VHDL array structure.

It is designed as a separate entity from the codec, having both a matching interface to easily connect. Router interface have a generics based interface to be configured for a different number of nodes, being three the minimum permitted. An internal resource was configured too accordingly to the number of nodes connected to.

In case of a desired output port is busy the router will stop reading packets from the transmitting node, avoiding its reception buffer gets empty and maintaining the flow control in the link. This simplified router behaves accordingly to the ECSS-E50-12A Standard. Although the router can support up to 224 addresses in logical addressing and 32 addresses in path addressing, the required number of pad for such configurations makes them a study case. Our router supports up to eight nodes in a star topology network without connecting to other routers.

Special care to crosstalk between adjacent paths belonging to different channels inside the FPGA was taken to ensure all time constraints are met in the design.

4 SYSTEM DEVELOPMENT AND TESTING

Simulations, both functional and post layout, and synthesis have been made using Mentor Graphics Corporation tools. For simulation Modelsim 6.0 was the application selected while for synthesis it was Leonardo Spectrum 2005. Post layout simulations are performed including VITAL libraries provided by Xilinx and Actel. Glitch handling differs at both, while in Xilinx VitalPathDelay arguments Mode, Xon and MsgOn are set to VitalTransport, true and false, in Actel are set to OnEvent, false and true; to avoid annoying To avoid those VitalGlitch warning messages simulations are run with the "+no_glitch_msg" optional argument at the vsim command.

The map and layout tools are device dependant. The initial codec prototype was tested in a Spartan IIE using Xilinx ISE tools to generate and load the bitstream file into the FPGA. That test showed the codec was right designed and it consumed a total gate count of 4952. Compared to other cores [4] the result is considered good taking into consideration it is an initial prototype. The next steps were synthesizing the codec for Actel proasic3 A3P250 using Designer, and provide it with connectors to test the equipment against a Star Dundee SpaceWire PCI2 board. Following this test a Spacewire-USB Brick was used to connect both cards, and finally the Star Dundee SpaceWire-USB Brick was replaced by our router implementation.

Although the design is intended as device independent, some details like system clock signals generation and distribution are treated as device dependant using FPGA resources

such as PLL and DLL. That gives a better performance in such critical areas for optimization and can be designed as external modules to the main design.

5 CONCLUSIONS

The implementation of synthesizable SpaceWire components based in the ECSS-E50-12A standard should take into account technology dependant issues such as design constraints, time and area optimization and accurate management of system clock frequencies. These issues will be common to any device selected, representing a physical limit to the standard.

When optimizing for particular details, like clock management or memory resources, device selected resources will play a key role, being the design tailored for a specific FPGA when looking for an improvement in the core performance.

It is necessary the support of qualified test equipment to prove the accuracy of our prototype. StarDundee commercial Spacewire equipment provides a good platform to perform the final stage of the test procedure.

6 FUTURE WORKS

With the 2010 horizon for the INTAμSAT-1 mission, additional works are being done to improve connectivity to other subsystems and payload components. One of these works is focused in the development of host interfaces for the Spacewire codec, having special attention to RMAP [5]. Data transfers and bus interfaces are also improvement goals for the codec, where DMA and AMBA are well known.

Concerning the router, a prototype card is the next step to further test its functionality between our own codec and the Spacewire PCI2 card. Improvements should be done to support both addressing schemes, giving chance to router interconnection and different and deeper network topologies. Redundant links between router and the associated management of the routing tables is also an improvement goal.

7 REFERENCES

[1] S.M. Parkes et al, “SpaceWire – Links, Nodes, Routers and Networks”, European Cooperation for Space Standardization, Standard No. ECSS-E50-12A, Issue1, January 2003.

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

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

[4] Gaisler Research, “SpaceWire Codec with RMAP. GRSPW / GRSPW-FT CompanionCore Data Sheet”, Gaisler Research AB, September 2006, Version 1.0.0.

[5] S.M. Parkes and C. McClements, “SpaceWire Remote Memory Access Protocol”, submitted to DASIA 2005.