



Implementation of a Multichannel Serial Data Streaming Algorithm Using the Xilinx Serial RapidIO Solution

Charles A. Doxley
Glenn Research Center, Cleveland, Ohio

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., “quick-release” reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.
- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question to help@sti.nasa.gov
- Fax your question to the NASA STI Information Desk at 757-864-6500
- Telephone the NASA STI Information Desk at 757-864-9658
- Write to:
NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199



Implementation of a Multichannel Serial Data Streaming Algorithm Using the Xilinx Serial RapidIO Solution

Charles A. Doxley
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

Acknowledgments

I gratefully acknowledge the support of the Space Communications and Navigation (SCaN) Compatibility Test Sets Project and the entire management team, the SCaN Network office, and the NASA Glenn Research Center. I would also like to acknowledge the entire Data Link and Modem Module development teams for their significant contributions to learning, debug and verify the Serial RapidIO solution.

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

This report is available in electronic form at <http://www.sti.nasa.gov/> and <http://ntrs.nasa.gov/>

Implementation of a Multichannel Serial Data Streaming Algorithm Using the Xilinx Serial RapidIO Solution

Charles A. Doxley
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Abstract

In the current world of applications that use reconfigurable technology implemented on field programmable gate arrays (FPGAs), there is a need for flexible architectures that can grow as the systems evolve. A project has limited resources and a fixed set of requirements that development efforts are tasked to meet. Designers must develop robust solutions that practically meet the current customer demands and also have the ability to grow for future performance.

This paper describes the development of a high speed serial data streaming algorithm that allows for transmission of multiple data channels over a single serial link. The technique has the ability to change to meet new applications developed for future design considerations. This approach uses the Xilinx Serial RapidIO LOGICORE Solution to implement a flexible infrastructure to meet the current project requirements with the ability to adapt future system designs.

Introduction

For the Compatibility Test Sets Project, the overall architecture of the system required communication between a data link design and modem design across a high performance communications interconnect technology. The project decided to use multiple field programmable gate array (FPGA) technologies to meet all of the design criteria. The CTS project set a goal to deliver the FPGA cards in a packaged environment that achieved heat reduction and noise reduction simultaneously; therefore, the team chose a chassis with a VSX serial backplane that increases airflow efficiency while isolating the power supply from the rest of the system components. With the decision to have multiple FPGA cards and independently developed hardware systems, board to board communication had to be delivered within the chassis using some form of serial data interface. After a detailed analysis, the Serial RapidIO (SRIO) protocol, which is an open standard protocol (Ref. 1), was chosen since it supports all the system interconnect needs and utilizes the serial backplane of the VSX chassis as a communication interface.

During the early stages of project planning, the engineering team concluded that a robust, scalable design would be needed to meet the requirements of a continuously evolving architecture. The team outlined a system on multiple hardware platforms that had capabilities to meet current design considerations and enabled scaling to meet future design considerations. In a progressive programmable architecture, the demand for higher bandwidth applications continues to increase while the hardware is fixed and limited. Board to board FPGA communication is important to provide high bandwidth and reliable data transfer between processing elements, and is critical to overall FPGA-based system performance (Ref. 2). To keep up with these demands, there are many efforts directed towards parallel structures and architectures to keep up with the ever growing demand for more bandwidth. Since we have fixed hardware, having multiple streams of data driven at a single clock rate allows a designer to process larger amounts of data in a more efficient manner. This creates the necessary flexibility to implement a reprogrammable architecture to meet future communication considerations for high bandwidth applications.

RapidIO technology has clearly become the dominant embedded interface technology in military and robust, high-performance embedded applications (Ref. 3). The RapidIO is a high performance, packet switched system level interconnect architecture. The interconnect architecture is an open standard developed by the RapidIO Trade Association which addresses the needs of numerous embedded infrastructure applications. The interconnect is intended primarily as an intra-system interface, allowing chip to chip, board to board and chip to board communications with performance levels ranging from 1 to 60 Gb/s (Ref. 4). Xilinx developed a SRIO core that delivers serial data at up to 10 Gb/s. Some of the key features of the SRIO are (Ref. 5):

- Packet switched—point to point interconnect to connect processors, coprocessors, memory, and memory mapped I/O
- Low overhead and low latency; optimized as an “inside-the-box,” device-level interface
- Small silicon footprint; can be implemented in ASICs and even FPGAs without consuming all of the gates
- Reliable delivery of Packets and recovery in hardware
- Layered architecture to fit of switch fabrics
- Support of Various I/O technologies: Standard LVDS and Serial Gigabit

Construction of the Algorithm

This paper presents a high speed serial data streaming algorithm that is designed for bi-directional communication to transmit multiple data streams across a serial backplane. The algorithm was developed to achieve efficient board to board communications from one FPGA card to another utilizing a VSX backplane inside a chassis. The algorithm was created to provide independent system interconnect of multiple channels over a single serial link. The design was created to free up other FPGA peripheral hardware resources for utilization which helps to effectively maximize the usage of the hardware platform that has a finite set of resources. The technique developed also meets the needs of a reconfigurable technology that can adapt to diverse and practical future applications.

The method introduces a unique adaptation of the functionality of the Xilinx RapidIO Endpoint solution to produce a flow control algorithm that can create multiple data channels capable of running at independent rates. Xilinx developed a RapidIO Endpoint solution that can be used to deliver a layered approach for implementing the RapidIO protocol which allows users to integrate portions of the design for their application.

Implementation of the Algorithm

The high speed serial data streaming algorithm was developed utilizing the Xilinx Implementation of the LogiCORE IP Serial RapidIO Design solution. The User Interface contains four ports where packets are issued that are intended for a remote device or where packet issued by a remote device are consumed. These four interfaces are Initiator Request, Initiator Response, Target Request, and Target Response (Ref. 6). The algorithm uses the initiator and target response ports and various features of each of the interface ports to create a bi-directional flow control algorithm that partitions the serial data connection into multiple independent data streams. The algorithm creates a transformation of traditional operation of the response port, which is typically used as a conduit for transactions requiring a response, and uses it as a low bandwidth, high priority operation to control data transactions. The algorithm creates messages that are transmitted on the `tresp_status` field across the target response port into the FPGA fabric. These messages are on a bus that are only 4 bits wide, compared to the 64 bit wide bus used to transport data from endpoint to endpoint. The priority filed on the status messages are given the highest priority so the built in flow control algorithm of the Xilinx LogiCore makes these packets of critical priority. The formation of packets is performed using the `tresp_vld_n`, `tresp_sof_n`, `tresp_eof_n` and the `tresp_rdy_n` hand shake signals (Ref. 6) to verify when a valid packet needs to be transmitted across the fabric.

Additional features of the algorithm showcase the customization of the `iresp_status` and `iresp_ftype` fields. Transforming these fields that indicate packet status for a transaction, into a decoder that partitions various data sources into active `ireq_data` transactions targeting the FPGA fabric. Additional fields such as `ireq_addr`, `ireq_dest_id`, `treq_addr`, as well as the normal user interface signals are used to packetize the data and for mapping of the data streams to their proper destination. The chart in Table 1 describes how the algorithm utilizes the available handshake signals to create a robust real time flow control process that operates within the serial rapidIO architecture.

The technique uses status messages that are derived from FIFO fill levels to generate status states that are transmitted across the SRIO interface, Figure 1 shows how the data flows from the target request, target response, initiator request and initiator response interfaces into the field programmable gate array fabric in a bi-directional fashion. The status messages are controlled by a state machine which is driven by the various watermark levels of the fifo. The state machine generates the status flow messages that control the data flow across the serial connection. The current implementation uses FIFO with a depth of 64 32-bit words. The watermark levels are at 25 and 50 percent of the fifo fill levels.

TABLE 1.—CHART SHOWS THE HANDSHAKE SIGNALS FOR A TWO CHANNEL IMPLEMENTATION OF THE DATA STREAMING ALGORITHM

Binary command	Hexadecimal	CH2	CH1	Data link
[0 0 0 0]	0	Stop CH2	Stop CH1	No steam
[0 0 0 1]	1	Stop CH2	Start CH1	Stream CH1 only
[0 0 1 0]	2	Start CH2	Stop CH1	Steam CH2 only
[0 0 1 1]	3	Start CH2	Start CH1	Toggle streams

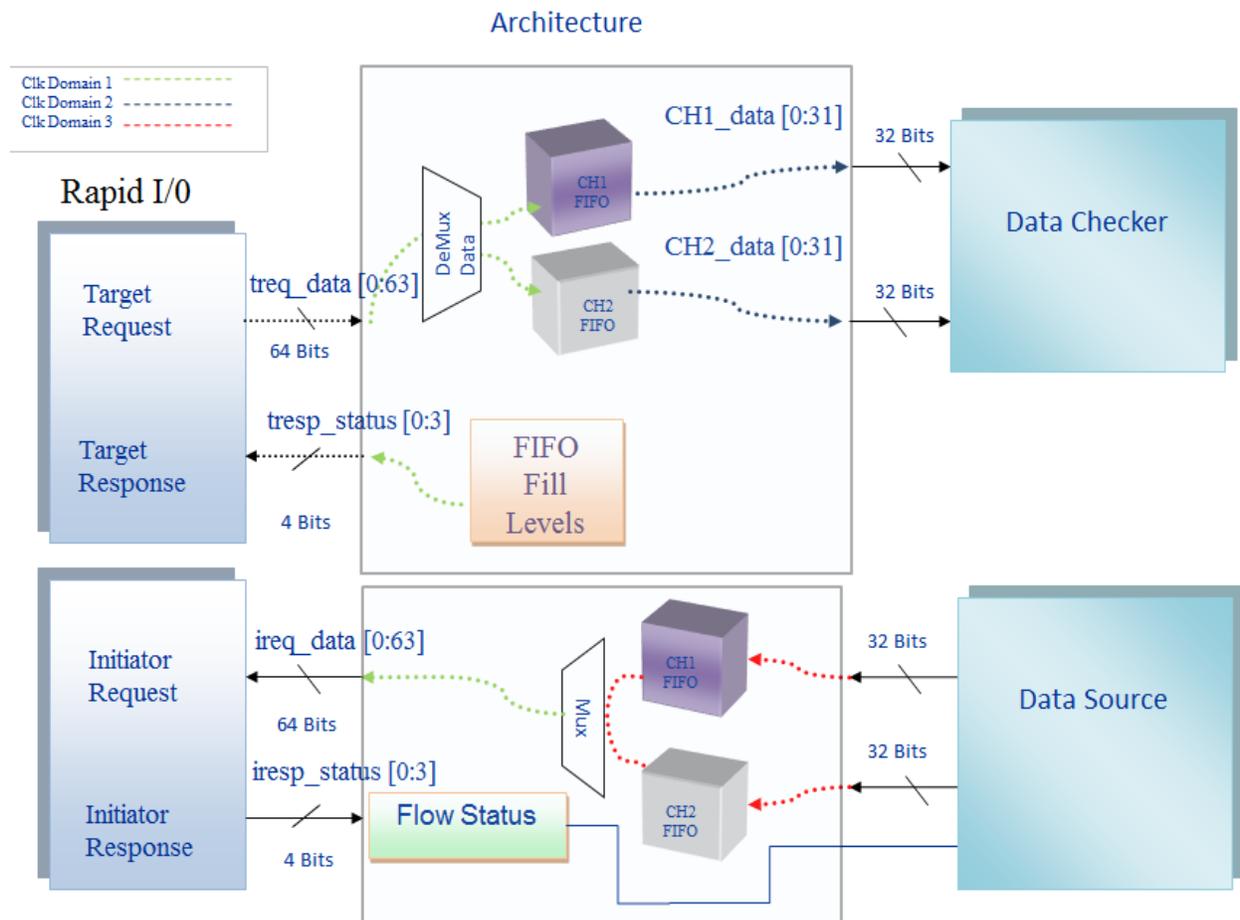


Figure 1.—Block diagram of the data flow and flow control messages for the implementation of the algorithm in hardware.

Hardware Implementation

The development of the algorithm was performed using Xilinx System Generator tool for digital signal processing. System Generator is a software tool that allows for the creation and verification of designs for Xilinx FPGAs. The Serial RapidIO core was created using Xilinx core generator software tool.

To implement the algorithm, multiple netlists designs were created to test each scenario. The algorithm was implemented using Virtex 5 FPGAs and Virtex 6 FPGAs, using the Xilinx 315 and 110 chips, respectively. A Tekmicrosystems high speed digitizer board was the platform that housed the FPGAs and was the hardware boards placed inside the VSX chassis. There was a series of tests conducted ranging from operation using a single data channel running at maximum rate by shutting off all other data channels, to dual channels operating at a fixed rate that were identical data rates, and eventually multiple channels running all at independent rates. All designs rates were verified using Xilinx Chipscope debug and verification tool. All tests were performed using a fixed packet length on each of the channels. Each Channel had the same fixed packet length for the duration of testing for each set of tests.

Results

Table 2 shows the results of multiple test cases for a dual channel implementation of the algorithm with variable packet lengths over a single transceiver line.

The results of the first phase of testing demonstrated that the algorithm successfully transferred data at the designed rates. It also proved that the Xilinx RapidIO cores deliver data at nearly the maximum rates advertised. A utilization of approximately 93 percent of the 10 GB bandwidth for four channel operation. The most efficient results occurred using 16 word packets, and showed similar near 88 percent efficiency using 8 word packets as compared to approximately 89 percent efficiency 32 words or maximal length packets. A decreased efficiency was demonstrated in testing when the variable length packets exceeded a length of 16 words per packet transmitted across the RapidIO fabric. In the single channel operation the best utilization occurred using 32 word packets. The efficiency reached near 92 percent of the 2.5 GB bandwidth for the single channel. Table 3 shows the algorithms efficiency reached near 89 percent of the 10 GB bandwidth for the four operational channels.

TABLE 2.—BLOCK DIAGRAM SHOWS THE UTILIZATION OF THE SINGLE LANE IMPLEMENTATION OF THE TECHNIQUE USING VARIOUS PACKET LENGTHS

pkt len	dwords/min	Gb/s	%Utilization
1	1782087773	1.90089363	0.76035745
2	1824715359	1.94636305	0.77854522
4	1904409023	2.03136963	0.81254785
8	1977397828	2.10922435	0.84368974
16	2139002813	2.27478783	0.90991325
32	2139002813	2.28160301	0.91264120

TABLE 3.—BLOCK DIAGRAM SHOWS THE UTILIZATION OF THE FOUR LANE IMPLEMENTATION OF THE TECHNIQUE USING VARIOUS PACKET LENGTHS

pkt len	dwords/min	Gb/s	%Utilization
1	6967365626	7.43218740	0.74321874
2	7218171058	7.69972510	0.76997251
4	7852584513	8.37646290	0.83764629
8	8227355773	8.77623670	0.87762367
16	8642376897	9.21853536	0.92189456
32	8332533998	8.88843180	0.88884318

The algorithm also demonstrated the capability to have multiple channel operation which shows that the current algorithm's architecture can meet the requirements for many of the waveforms and modes of operation that are outlined in the Space Networks User Guide. Showing that multiple channel operations can achieve data rates higher than the current rates outlined in the CTS project requirements also demonstrates the algorithm's flexibility to adapt to reconfigurable technologies and future system architecture redesigns or upgrades. The algorithm successfully changes the flow of data to meet the programmable rate changes demonstrated in testing. The design modification of the response interface still has ports with other available data buses that are not utilized in the algorithm. These available buses present the possibility of further architecture modifications that could meet even more complex waveform adaptations and unique architecture applications. The algorithm presented can successfully meet all the criteria for the modem module development team while freeing up additional resources for other design teams to utilize for their development efforts.

Further design work has begun to implement larger FIFO buffers and also utilizing new watermark levels to improve the overall efficiency of the algorithm.

References

1. Rapid IO Trade Association Rapid IO Specification 2.1 <http://www.rapidio.org/rapidio-specifications/>
2. Mak, Terrence, Sedcole, Pete, Cheung, Peter, and Luk, Wayne, "On-FPGA communication architectures and design factors," in Proc. IEEE International Conference on Field Programmable Logic and Applications (FPL), 2006.
3. RapidIO: The Interconnect Architecture for High Performance Embedded. Whitepaper. http://www.rapidio.org/files/techwhitepaper_rev3.pdf
4. Ying, Ji. Kong, Chao and Zhi, Hui, "Design of RapidIO User-level Communication Interface Based on Socket in Real-Time Applications," in Computational Intelligence and Software Engineering (CiSE), 2010.
5. Cox, Tom. "Serial RapidIO Embraces Role as System-Level Fabric," COTS Journal Volume 10 Number 1 January 2008.
6. Xilinx, Inc. LogiCORE IP Serial RapidIO User Guide http://www.xilinx.com/support/documentation/ip_documentation/srio_ug503.pdf

