Eurasian Journal of Science & Engineering
ISSN 2414-5629 (Print), ISSN 2414-5602 (Online)
EAJSE

Real-time Transceiver System Based on Rapid-I/O Protocol

Rand B. Mohammed[1] & Roelof van Silfhout[2]

[1] Mechatronics Engineering Department, Ishik University, Erbil, Iraq
[2] School of Electrical and Electronic Engineering, University of Manchester, United Kingdom
Correspondence: Rand B. Mohammed, Ishik University, Erbil, Iraq.
Email: rand.basil@ishik.edu.iq

Received: October 5, 2018 Accepted: November 22, 2018 Online Published: December 1, 2018

doi: 10.23918/eajse.v4i2p39

Abstract: With the all transceiver systems the transmission speed is the most critical factor. The main objective of this paper is to design a new reference model for the multi-link system that is used to carry images in a real-time system, with a high resolution at high transceiver speeds, between two applications over physical media. The new system, called Transceiver System based on Rapid-I/O protocol (TRIO), for both transmitter and receiver, based on eight parallel devices was placed at the bottom of the system architecture, at both transmitter and receiver. The transceiver system is designed as scalable system in which the system data rate increased when multiple lines connected in parallel are used to carry data between the transmitter and the receiver instead of using a single line. By transmitting the same packet of data, once over a single transceiver system and then over eight transceiver systems, ideally an eight-fold improvement in bit-rate is expected. The TRIO is implemented on a field programmable gate array (FPGA) and a Terasic DE4 board will be used as the main platform for implementing and testing the embedded system, while Quartus-II software and tools are used to design and debug the embedded hardware system.

Keywords: Rapid-IO Protocol, High Transceiver Speed, Parallel Transceiver Systems, Embedded System, FPGA

1. Introduction

To enable different devices such as computers, entertainment electronics and telecommunications connecting together and to communicate with each other smoothly and efficiently, a set of rules, called communication protocols, must be followed. A computer network is made up of hardware components that are responsible for the task of transmitting information from one computer to another. In order to organise communication, computers are programmed in a primitive machine language which only uses ones and zeros. A computer network is used and controlled through computer programming, by creating a complex software system called a network operating system. The network computing system needs to handle communication functions and tasks with different levels of complexity. These tasks and functions are bundled together in layers, which in turn are structured one on the top of the other and are available to the user or an application through a suitable interface. Computer networks have developed over the years; the first computer network, for instance, focused primarily on hardware and software protocols as a secondary consideration. This strategy, however, has changed radically, and today's software protocols are highly organised (Meinel, 2013).

Reference model is a term used to describe how a system carries data between two points. The main standard reference models are ISO/OSI and TCP/IP, with the ISO/OSI model mostly being a theoretical reference model and the TCP/IP model being a practical development for practical application (Meinel, 2013; Severance, 2015; Bonaventure, 2011). The TCP/IP, for example, can be used by implement it as an embedded system (Hong, 2011), to carry data over the Ethernet (Zhong-Zhen, 2006) but cannot operate in real time because it works as a client and a server, with a series of requests and responses when carrying data; it is also highly costly with complex undertaking when using different protocols to carry data with higher speed (Yong, 2011). The embedded system is implemented inside the FPGA either as partially software and partially hardware, or as full hardware.

The architecture of the system that is implemented as partially software and partially hardware is shown in Figure1. The TCP/IP is one of the most commonly used models; this embedded system consists of an embedded CPU that controls the hardware system inside the FPGA device and is itself controlled by the host PC (Ri-Kun, 2006; Yan, 2010; Hashimoto, 2010). The processor's data-rate is determined by the CPU speed. A data congestion inside the CPU is caused either when processing the incoming data inside the FPGA, and the processing takes time higher than the required time to generate a new image which causes loss of some of the input images, or when carrying data over the network, which is known as a 'network bottleneck' (Kim, 2009). The architecture of the second embedded system, which is implemented as hardware without a software part and a CPU is shown in Figure 2.

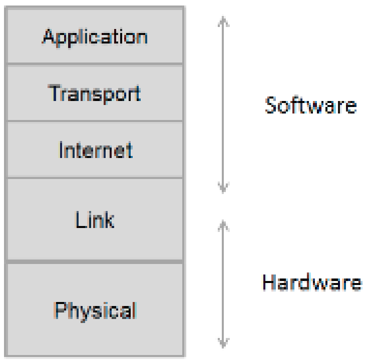


Figure 1: The Architecture of the embedded software and hardware systems

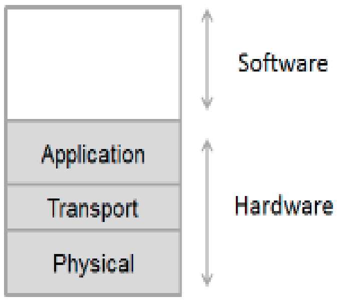


Figure 2: The Architecture of the embedded hardware system

2. Methodology

The TRIO (Transceiver system based on Rapid-I/O) is a complete reference model used to connect two applications placed at two different sides, a transmitter and a receiver, through a physical medium, by providing a full transceiver system that connects between the two points. The TRIO used to carry real-time images between two systems, a source and a destination, at high transceiver speeds. The TRIO divides the transmitter and receiver systems into three layers, i.e. application, transport and physical, as shown in Figure 3. Each layer has its own job to do, and in combination with the others it provides a full transceiver system. The name of each layer is given to reflect part of its job, since the application layer contacts the actual application at both system sides, the physical layer contacts the actual physical media, and the transport layer controls the flow of data between the other layers.

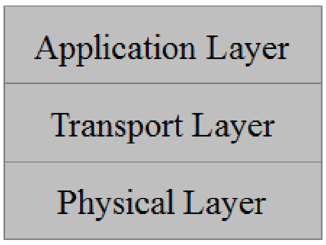


Figure 3: The TRIO architecture

One of the main purposes behind designing the TRIO is to overcome the complexity involved in using other reference models, such as TCP/IP, and their required protocols as well as the limitations of using a single transceiver system. A second purpose is to increase transceiver speed by using a less complex and cheaper parallel Ethernet network. With the TRIO, a parallel network connection is used instead of a single network connection, by connecting a multiple single transceiver system in parallel at the physical layer, for both the transmitter and the receiver, which ensures a higher system speed based on the number of devices connected in parallel (Mohammed, 2017).

2.1 Application Layer

The application layer is the top layer of the TRIO which provides the system with the ability to access an application device in order to exchange data. The TRIO has a full architecture for the transmitter and receiver sides, and different processors are applied to the data.
On the transmitter side, as shown in Figure 4, the application layer consists of several sub-layers, some of which are physical devices, such as the source of data (camera) with an external memory device, and others are implemented as an embedded system inside the FPGA device. The camera generates real-time data which are stored temporarily inside the external memory and which then interface with the embedded system inside the device. The embedded system part of the application layer consists of three sub-layers (see Figure 4) which carry the data before the embedded system interfaces with the memory in order to read the data and process it accordingly. The first sub-layer provides a connection between the external part of the application layer and the rest of the system, the second sub-layer controls reading data from the memory controller, the third sub-layer divides long packets read from the device into multiple sub-packet.

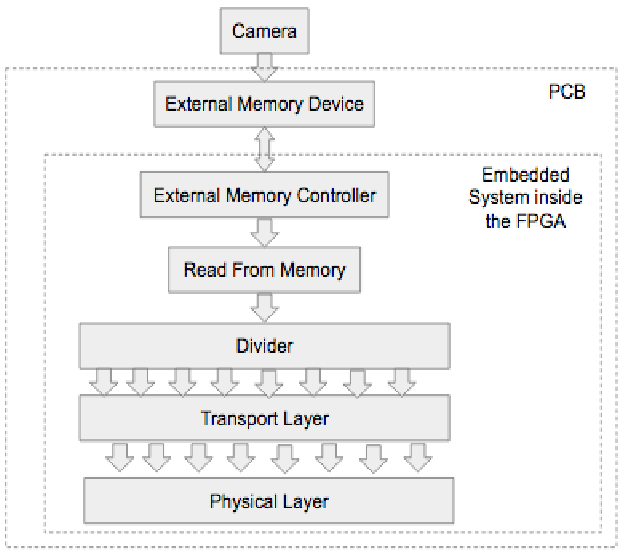


Figure 4: A schematic diagram for the TRIO application layer on the transmitter side

However, on the receiver side, as shown in Figure 5, the application layer is placed at the final stage of the system and consists of several sub-layers, some of which are physical devices such as the destination device with the external memory device, while the others are implemented as an embedded hardware system. The external memory device interfaces with the embedded system inside the device and is used to store the data received in order to forward them to a destination application such as the Vision System. The embedded system part of the receiver application layer consists of several sub-layers. The first sublayer combines sub-packets of data in order to generate a full packet of the original width. The second sublayer is responsible for controlling writing data into the final sub-layer (the memory controller), which provides a connection with the external memory device and is used to monitor data flowing between the embedded system and the memory device.

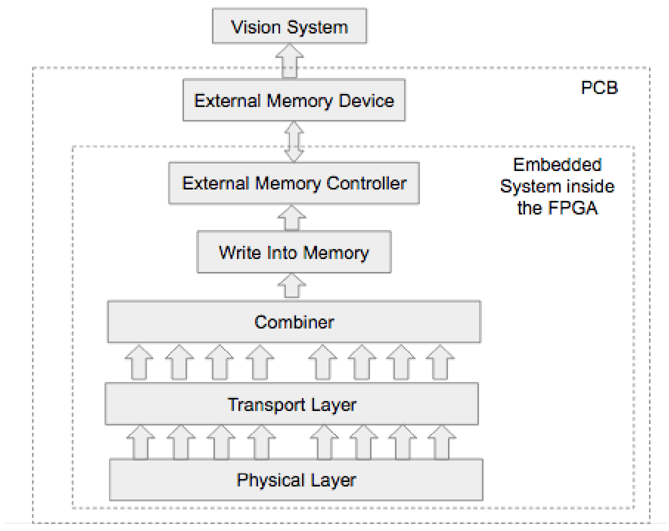


Figure 5: A schematic diagram for the TRIO application layer on the receiver side

2.2 Transport Layer

The transport layer is implemented at both the transmitter and the receiver sides, between the application layer and the physical layer. The main objective of this layer is to synchronise data flow. The transport layer has two interface ports, one connected to the application layer and the other one connected to the physical layer. On the transmitter side, the transport layer carries data from the application layer to the physical layer. However, on the receiver side, this layer carries data between the physical layer and the application layer by changing the format of data taken from the physical layer into a suitable format for sending to the application layer. The operations applied to data at the receiver side are exactly the opposite of those applied to the same layer on the transmitter side.

Each interface port consists of multiple cores that are used to connect one sub-model from the application layer with a single transceiver system at the physical layer that supports 3.125 Gbps as a maximum transceiver speed. These cores are called 'data controllers' (see Figures 6 and 7) and are implemented as an embedded system and connected in parallel at each side. All the data controller cores are triggered with the same clock in order to synchronise data movement.

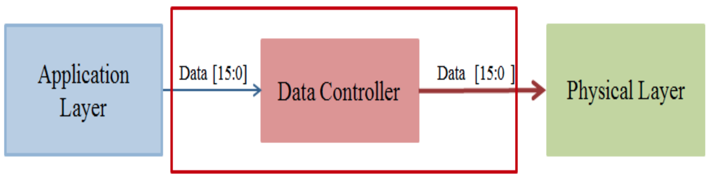


Figure 6: The transport layer architecture for a 3.125 Gbps single device on the transmitter side

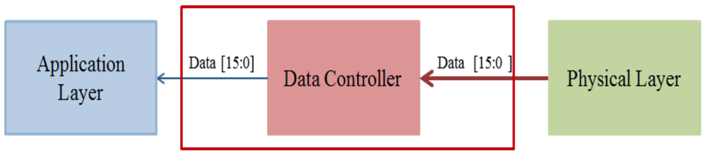


Figure 7: The transport layer architecture for a 3.125 Gbps single device on the receiver side

2.3 Physical Layer

The physical layer is one of the most important parts of the TRIO system. Its name defines its real purpose, which is based mainly on connecting one or two applications in different systems through physical media, in order to exchange data. The physical layer has two interface ports, one of which is connected with the rest of the embedded system inside the FPGA while the other one exports outside the FPGA in order to connect to the transceiver ports and devices. The physical layer consists of two sub-layers, both of which are implemented as an embedded system, and the bottom sub-layer interfaces the physical medium.

Data transmission over parallel nodes is important, especially for those systems requiring high data transmission rates. Many transmission protocols are designed based on parallel methods such as the Rapid-I/O. In TRIO systems, the Serial-Rapid-I/O (Xiao-yun, 2010; Fengfeng, 2010; Xiangyang, 2013) is already based on the internal parallel structure and are modified to be used with the external parallel structure.

The Rapid-I/O as a Transceiver Protocol:

The Rapid- I/O standard is a new standard that provides high-speed and low-latency interconnection technology, and it is commonly used in many industrial applications. It is designed as a high-speed packet switch with full duplex architecture employed to connect chip-to-chip and board-to-board (Adams, 2001; Sukhtankar, 2004). It is a switched fabric specifically designed for developers and researchers by the leader in the field of embedded computing. It can be used with different types of application, such as wireless infrastructure, military, scientific, storage, edge networking and industrial equipment. Moreover, the Rapid-I/O is the best choice for high performance embedded systems, due to its reliability, flexibility, high performance and cost effectiveness (Changrui, 2010). Rapid-I/O technology is based on the hierarchical layered protocol, which is currently viewed as the most effective way to build an embedded system, as it allows for long-term flexibility in sub-system upgrades. The ISO/OSI model is the most well-known example of a hierarchical layered protocol. Moreover, the Rapid- I/O standard consists of three main layers: the logical layer, the transport layer and the physical layer (Shippen, 2007).

For the TRIO, the Rapid-I/O is an embedded transceiver protocol. For transmitting data, it converts low-speed parallel data into high-speed serial data, and for receiving data it converts high-speed serial data into low-speed parallel data. The TRIO's physical layer consists of multiple single transceiver systems connected in parallel, and each transceiver system is divided into two sublayers. For the transmitter side, we have a frame generator (FG) with a transmitter device, while the receiver side has a frame checker (FC) and a receiver device. Both the frame generator and the frame checker cores are created as a customise IP-blocks for this system while the transmitter and the receiver devices used are provided by Altera in Quartus-II library. The transmission and receipt devices consist of the PCS (Physical Coding Sublayer) and the PMA (Physical Medium Attachment) cores. A frame generator and frame checker are implemented as embedded systems inside the device in order to interface the transport layer at one side and the PCS core at the other side, through the data and control buses. The PCS core consists of a 16/20 encoder on the transmitter side and a 20/16 decoder on the receiver side, while the PMA works on the transmitter side as a serialiser (SER) and on the receiver side as a de-serialiser (DES).

2.3.1 The Frame Generator (FG)

This is the first sublayer in the transceiver system on the transmitter side. It connects between the transmitter device and one part of the transport layer. The FG has two interface ports, one with the transport layer over a 16-bit data-bus, while the other is with the PCS core on the transmitter side with two buses, the control bus and the data bus. The data bus, with a 16-bit signal, carries data to the PCS, while the 2-bit control bus is used to inform the PCS of the type of the incoming data. If they are equal to 0, the information being carried over the data-bus is either a length or a payload-data; however, if the value of the control signal is 1, the information carried over the data bus is control data. Besides connecting two layers, the other main purpose of the FG is to generate the data in a format acceptable to the next core, i.e. the PCS core, by adding a header file to the original data packet (see Figure 8). The process inside this core involves reading data from one part of the transport layer equal to 16-bit widths and various lengths (between 46 bytes and 1024 bytes), then adding a header file to each data packet. The header is used to align the PCS core with the coming data and it is assigned to 0xBCBC for at least three clock cycles. The next two bytes are used to specify the length of the packets in bytes, and the value of the length varies according to the length of the coming packet, which is assigned by the previous core. For the transceiver systems, the length is fixed to 0x400 bytes (1024 bytes).

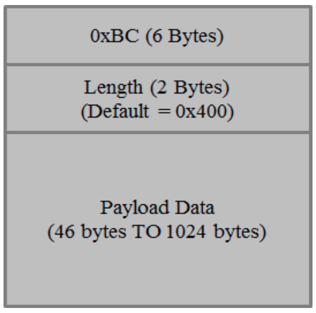
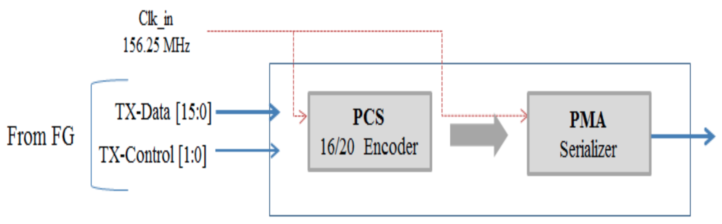


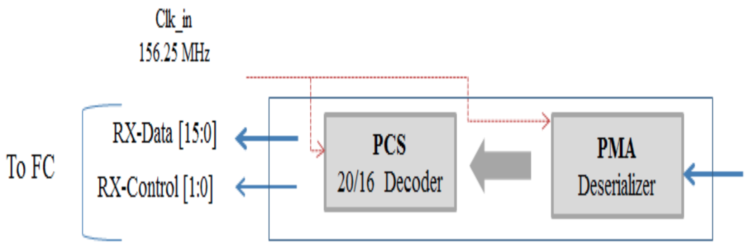
Figure 8: The Rapid-IO header file

2.3.2 The Physical Coding Sublayer (PCS)

The PCS core is the first sublayer in the transceiver system and is implemented inside the system as a part of the physical layer. It works as an encoder on the transmitter side and as a decoder on the receiver side. The PCS core has two interface ports, as shown in Fig. 9 a and b, one with the FG/FC via the data and control bus, while the other is with the PMA core. The data and control buses are used to define the type of incoming data, which is required to encode and decode them accordingly. The 16/20 encoder and the 20/16 decoder are used with this protocol, and each PCS is connected directly to one FG on the transmitter side and one FC on the receiver side. The full TRIO consists of eight PCS cores connected in parallel on the transmitter side, and eight PCS connected in parallel on the receiver side. The data carried between the PCS and the PMA are low-speed parallel data 20 bit in width. The clock that is used to clock the whole embedded system with the PCS, the system side at the PMA is a low-speed 156.25MHz clock, and all the systems' parts are connected to the same clock in order to synchronise the process (Corporation, 2014).



(a)



(b)

Figure 9: The Rapid-I/O transceiver system at: (a) the transmitter side (b) the receiver side

2.3.3 The Physical Medium Attachment (PMA)

For the Rapid-I/O transceiver protocol, the PMA is the final stage in the physical layer, and it connects the embedded system with the physical medium on both the transmitter and the receiver sides. The PMA core is based on the SerDes (Serialise Deserialise) process. For transmissions, the PMA works as an SER by converting low-speed parallel data into high-speed serial data, in order to send them out to the physical medium. On the receiver side, the PMA works as a DES by converting high-speed serial data received from the physical medium into low speed parallel data in order to forward them to the PCS core. The PMA core has two input clocks, one of which is a low-speed clock on the system side with a value of 156.25MHz, while the other one resides on the medium side with a value of 3.125GHz. Based on these two clocks, the system will serialise and de-serialise the data. Two modes are available to use with the Rapid-I/O protocol, a single-data rate (SDR) and a double-data rate (DDR). The maximum speed supported with SDR is 3.125 Gbps, while with the DDR it is 6.25 Gbps.

For a full TRIO architecture, the physical layer consists of eight single transceiver systems connected in parallel. These transceiver systems can work as transmitter systems or as receiver systems, or as both. The maximum theoretical data rate supported by runs up to 25 Gbps if the TRIO acts either as a transmitter or as a receiver system connected to each other, while if the TRIO acts as a full transceiver system which is connected to another full transceiver system, the maximum theoretical data rate can be up to 50 Gbps. The input/output of the TRIO is exported to outside of the FPGA and connected to other systems on the opposite side as a device-to-device network topology. The ports used to carry the signal to and from the system are PCML I/O ports, which are available with many transceiver devices such as the PCI-Express and the HSMC interface boards (Corporation, 2005, 2009, 2014).

3. Implementation

System implementation is based on software and hardware. Software in this regard relates to systems cores that are created using Quartus-II software and the Qsys builder to design the IP cores, and then implemented inside the FPGA as an embedded hardware system, while the hardware implementation means the necessary physical elements, such as a board, an FPGA device and transceiver ports. The Qsys builder is a newer version of the SOPC builder that provides high design performance, reliability and faster verification. It is used to generate a fully embedded system based on IP blocks inside the Qsys and connecting them internally based on the NoC, which can significantly save effort and time for the FPGA as well as provide high reliability in relation to interconnections along with higher operating frequencies (Corporation, 2011).

The system was tested based on two different data sources; the first source was data read from fixed memory (internal or external), while the second one was data read from a real-time data generator. The memory was used as a test case for off-line systems when data are stored in the temporary location in order to apply further processes later, while the real-time generator was used as a test case for a system applying real-time images/data and requiring an immediate process.

Transceiver System for the Rapid-I/O Transceiver Protocol:

The physical layer includes eight subsystems connected in parallel. Each sub-system consists of a Frame Generator (FG) with a transmitter device on the transmitting side and a Frame Checker (FC) with a receiver device on the receiving side. The FG and the FC cores both were implemented as a custom IP cores by writing Verilog-HDL codes for their internal processes and then converting them to IP blocks.

The FG, as shown in Figure 10, was implemented as an IP block with two interface ports: one was the Avalon-ST sink connected internally inside the Qsys builder with an individual part of the transport layer, and the other port was and Avalon-ST source that was exported to outside the Qsys builder on the top level and connected with the transmitter device through data and control signals. The internal processes for this core is read data from the transport layer, added a header file to each packet of data, and then forwarded the data with their relevant control signals to the transmitter device.

The FC, shown in Figure 11, was implemented as an IP block with two interface ports: the first one was the Avalon-ST source that connected the FC inside the Qsys builder with an individual part of the transport layer, while the other one was the Avalon-ST sink that consisted of data and control buses which were exported from the Qsys builder to the system's top level and connected with the receiver device. The internal processes inside this core received information over the data bus with a control signal from the receiver device. The control signal defined the type of information carried over the data bus: if the control bus was equal to 0, the received information was original data; otherwise, it was control-data, following which the process removed all header files and control messages from the incoming data and generated the original packet of data in order to forward them on to the next core on the transport layer. Both Avalon-ST source for the FG core and Avalon-ST sink for the FC core had a data bus 16 bits in width and a control bus 2 bits in width. Each bit from the control bus related to 8 bits from the data bus.

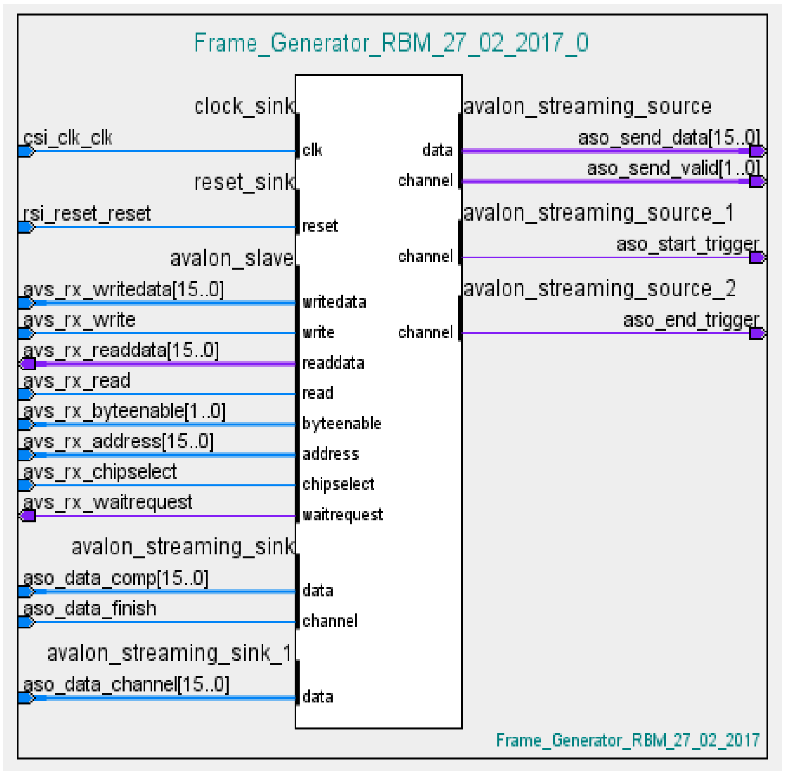


Figure 10: The Frame Generator IP core for the Rapid-I/O transceiver protocol

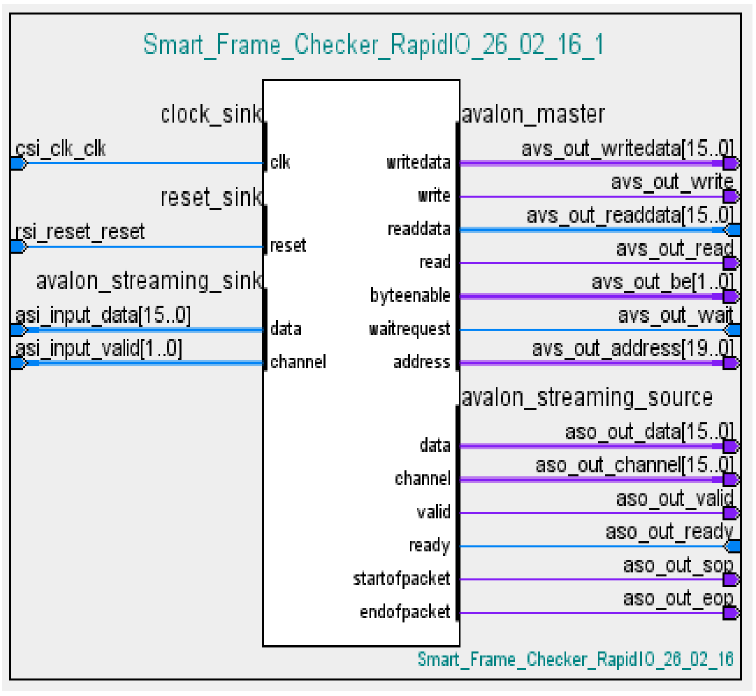


Figure 11: The Frame Checker IP core for the Rapid-I/O transceiver protocol

4. Test of the Proposed System

The system has been tested by programming the FPGA device with the embedded system and then configuring and debugging the FPGA device with the signal-tap analyser using a JTAG cable. Inside all systems an embedded counter was implemented and used to calculate the number of cycles for each system between reading the first byte from the source and writing the last byte to the destination memory. The final value of the counter was displayed by the signal-tap analyser. The counter was triggered by the same clock used to trigger the system, for the transceiver system based on the Rapid-I/O protocol the clock-value was either 156.25MHZ, 195.3125MHz or 312.5MHz.

Terasic's DE4 board, as shown in Figure 12, consists of many components that are used to support several applications, and it was used herein as a main testing board for the transceiver system, namely the FPGA device (used as a main component on the board) with data sources including the external memory, the on-chip memory, and the built-in generator (Terasic, 2003-2015).

The transceiver system was tested by exporting the transmitter and receiver signals to bank 1 on the HSMC board (High Speed Mezzanine Card). HSMC's bank 1 consisted of eight ports assigned with a PCML voltage, with each port supporting up to 6.5 Gbps, meaning that bank 1 is suitable for different transceiver protocols with different speeds (Corporation, 2009). Both systems were tested in two ways. For the full transceiver system, this was achieved by connecting two transceiver systems together, exporting each one to a separate HSMC board, either on the same board as Figure 12 shows, or a separate testing board such as two DE4-boards, using an SMA Cable with an XTS board.

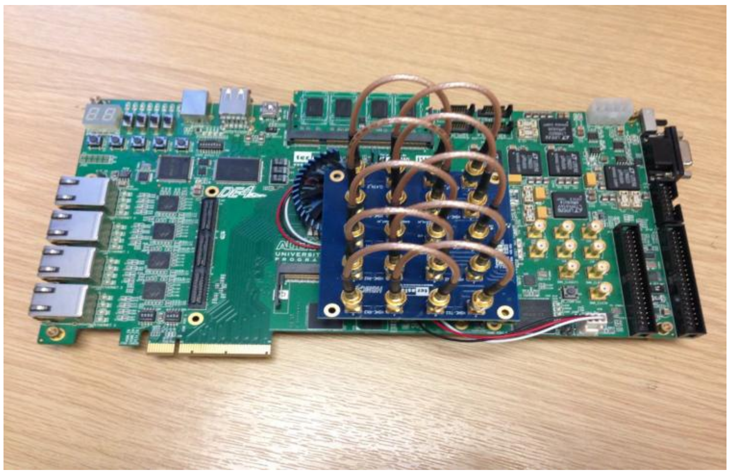


Figure 12: The HSMC's TXs and RXs connecting together through SMA cables and an XTS board on the DE4 board (Mohammed, 2017)

5. Experimental Results

The transceiver system was tested by carrying data between two sides over the physical medium, once over the parallel lanes and once over a single lane with two test cases based on different types of source data: a fixed memory and a real-time generator.

Three factors were used to compare the single and parallel systems in both test cases. First, we compared values read from the source with the value written to the destination, using a signal-tap analyser by comparing the signals of P1, that shown in Figures (13-14), with those of P2 and all the signals of P3 with all those of P4, in order to check whether the read data were received and written in the correct order and format. Secondly, we measured the processing time for the system using counter-2, i.e. the number of cycles required to complete the full process, starting from reading the first byte from the source to writing the last byte into the destination. This value was found by using the embedded counter inserted into all the systems, and its value was displayed with the signal-tap analyser under P5 in order to count the processing time for each system.

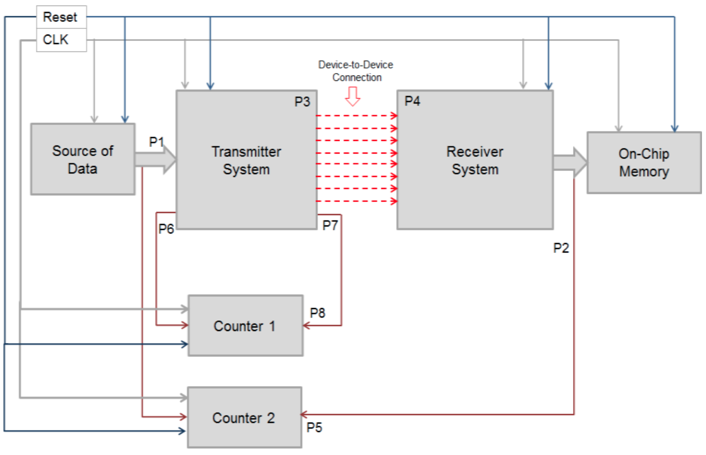


Figure 13: A schematic diagram of a basic test for a parallel transceiver system

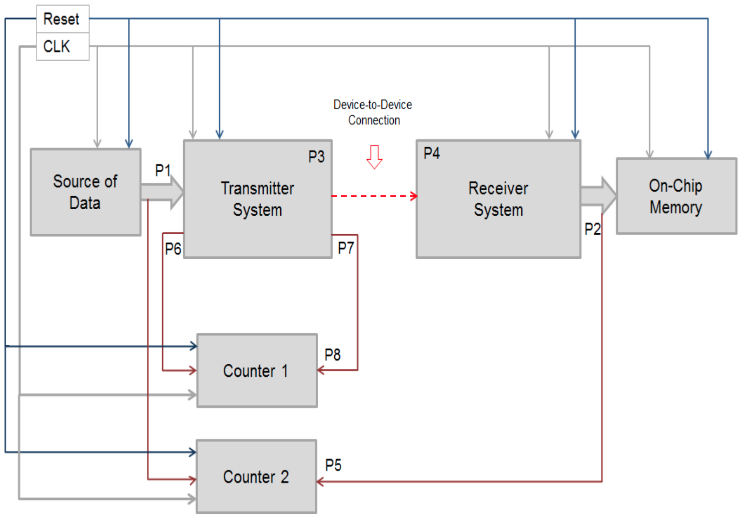


Figure 14: A schematic diagram of a basic test for a single transceiver system

The practical data rate was determined in the actual system by calculating the total required time starting with reading the first byte from the memory at the transmitter side and ending with writing the last byte from the packet into the memory at the receiver side, which could determine the delay in the system between reading and writing the data, then converting this time into a data rate using equation (1). The time was calculated on the basis of the value provided by counter-2 and the value of input frequency resulting from equation (2).

$$Practical\ data-rate = \frac{Size\ of\ the\ Input\ Packet\ (bit)}{Processing\ Time\ (sec)} \qquad (1)$$

$$Processing\ Time\ = \frac{Counter\ Value\ (decimal)}{Input\ Frequency\ Value\ (Hz)} \qquad (2)$$

However, the pixel-rate means the amount of pixels that can be carried at the transmitter side during one second in time. With transceiver systems, the image is carried with its original size, so beside the size of the input image the pixel data-rate will depend on the execution time as shown in equation (3). The size of the input image is fixed at 256x256-pixels, while the execution time is varied according to the time required to read and transmit the full packet. The time required to read data is fixed for both the single and the parallel transceiver systems, while the transmission-time is varied according to the number of the lanes used to carry data and the value of the input frequency, which depends on the transceiver protocol. The time was calculated on the basis of the value provided by counter-1 and the value of input frequency.

$$Pixel\ rate\ (pixel/\mathrm{sec}) = \frac{size\ of\ the\ inpput\ packet\ (pixel)}{Execution\ Time\ (sec)} \qquad (3)$$

The maximum size of a packet carried through each system cycle was 256x1024 bits. The Rapid-I/O devices were tested with different clock values: 156.25 MHz, 195.3125 MHz and 312.5 MHz. With this system and the available FPGA device (i.e. the Stratix-IV GX), the maximum value that could be run without losing some of the signals was 312.5MHz. In the first test case, when the data were provided by a fixed memory, the system took 4600h clock cycles to carry 256 Kbits through a single lane, the time taken varies according to the value of the input frequency used to trigger the system. In this instance, the system tool was 0.114ms at 156.25MHz, 0.0917ms at 195.3125MHz and 0.0573ms at 312.5MHz. However, the system requires A00h to carry the same packet of data through eight parallel lanes, and the time taken varies according to the input frequency used to trigger the system, which is equal to 0.0163ms at 156.25MHz, 0.013ms at 195.3125MHz and 0.008192ms at 312.5MHz. Theoretically, however, the time required to carry 256 Kbits through a single lane is 0.104ms, 0.083ms and 0.0524ms at 156.25MHz, 195.3125MHz and 312.5MHz, respectively. Through eight parallel lanes, the time required is 0.013ms, 0.0104ms and 0.0065ms at 156.25MHz, 195.3125MHz and 312.5MHz, respectively. The difference between the theoretical and the measured values was due to the delay inside the system, because the theoretical values were determined when the system showed no delay whatsoever, which was the ideal case.

The maximum measured data rate provided by a single transceiver system was 2.29Gbps, 2.699Gbps and 4.574Gbps at 156.25MHz, 195.3125MHz and 312.5MHz, respectively, while the maximum practical data rate provided by a single transceiver system was 16.08Gbps, 20.16Gbps and 32Gbps at 156.25MHz, 195.3125MHz and 312.5MHz respectively. The data rate to carry data over parallel lanes is 6.99 times faster than carrying them over a single lane, which is the main benefit of using parallel lanes instead of a single lane. When using the Rapid-I/O protocol, three different frequencies can be used which allow increasing the data rate and reducing the system execution time, which adds another benefit when using this protocol type. With a transceiver system, the data are supposed to be received by the receiver application in the same order and format.

The pixel-rate depends on two factors: the execution time, which varies according to the value of the input frequency and the size of the input image which is equal to 65536 pixels (256x256-pixels). With Rapid-I/O as a transceiver protocol, three frequency values are used to trigger the system, since there are three pixel-rates: 0.138 Gpixel/sec, 0.1735 Gpixel/sec and 0.278 Gpixel/sec at 156.25MHz, 195.3125MHz and 312.5MHz, respectively.

6. Conclusion

In conclusion, the TRIO was designed and implemented as a new reference model for the multi-link system that was used to carry images in a real-time system with a high resolution at high transceiver speeds between two applications over physical media. The theoretical study and the experimental results yielded conclusion regarding the approaches implemented by the system with the available technologies based on supporting devices and different transceiver speeds. The transceiver systems were designed as scalable systems in which the system data rate increased when multiple lines connected in parallel are used to carry data between the transmitter and the receiver instead of using a single line. The transceiver system was tested by transmitting the same packet of data, once over a single transceiver system and then over eight transceiver systems, ideally an eight-fold improvement in bit-rate is expected. In practice, we found that the result was 32 Gbps. When carrying the same packet of data through a single line, the transceiver data rate was 4.574Gbps. The speed in the parallel system was 6.99 times as high. The pixel rates were 0.138Gpixel/s, 0.1735Gpixel/s and 0.278Gpixel/s at the Rapid-I/O protocol when the input frequencies were 156.25MHz, 195.3125MHz and 312.5MHz respectively.

References

Adams, J., Katsinis, C., Rosen, W., & Hecht, D. (2001). *Simulation Experiments of a High-Performance RapidIO-based Processing Architecture.* Cambridge: MA.

Bonaventure, O. (2018). *Computer Networking: Principles, Protocols and Practice*. Retrieved from http://cnp3book.info.ucl.ac.be/2nd/cnp3bis.pdf

Changrui, W., Fan, C. & Huizhi, C. (2010). A High-Performance Heterogeneous Embedded Signal Processing System based on Serial RapidIO Interconnection. *3rd International Conference on Computer Science and Information Technology* (pp. 611 - 614). Chengdu, China. Publisher: IEEE Xplore.

Corporation, A. (2005). *High Speed Differential I/O Interfaces in Stratix Devices*. Altera.

Corporation, A. (2009). *High Speed Mezzanine Card (HSMC)*. Altera.

Corporation, A. (2011). *Applying the Benefits of Network on a Chip* Architecture *to FPGA System Design*. Altera.

Corporation, A. (2014). *ALTGX Transceiver Setup Guide for Stratix IV Devices*. Altera.

Fengfeng, W., Song, J., Wujian, L. & Yuan, W. (2010). A Serial Physical Layer Design in RapidIO. *IEEE International Conference of Electron Devices and Solid-Sate Circuits EDSSC* (pp. 1 - 4). Hong Kong, China. Publisher: IEEE Xplore.

Hahimoto, K., & Moshnyaga, V. (2010). A New Approach for TCP/IP Offload Engine Implementation in Embedded Systems. *Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers* (pp. 1249 - 1253). Pacific Grove, CA, USA. Publisher: IEEE Xplore.

Hong, R. (2011). Research and Application of TCP/IP Protocol in Embedded System. *IEEE 3rd International Conference on Communication Software and Networks* (pp. 584 - 587). Xi'an, China. Publisher: IEEE Xplore.

Jongmok, J., Donggil, K., & Dongik, L. (2013). IEEE 1588- based Clock Synchronization for Embedded Networked System with SRIO. *International Conference on ICT Convergence ICTC* (pp. 843 - 845). Jeju, South Korea. Publisher: IEEE Xplore.

Kim, S., Park, C., Kim, S., & Chung, Y. (2009). The offloading of socket information for TCP/IP Offload Engine. *International Conference on Advanced Communication Technology ICACT* (pp. 826 - 831). Publisher: IEEE Xplore.

Meinel, C., & Sack, H. (2013). *Internetworking*. Berlin: Springer.

Mohammed, R. B. (2017). *Novel Scalable and Real-time Embedded Transceiver system*. PhD thesis, University of Manchester.

Ri-Kun, L., Yue-Feng, J., & Hui, L. (2006). Optimized Design and Implementation of TCP/IP Software Architecture Based on Embedded System. *International Conference on Machine Learning and Cybernetics* (pp. 590 - 594). Dalian, China. Publisher: IEEE Xplore.

Severance, C. (2015). *Introduction to Networking*. USA. Retrieved from http://do1.dr-chuck.net/net-intro/EN_us/net-intro.pdf

Shippen, G. (2007). *System Interconnect Fabrics: Ethernet versus RapidIO Technology*. Austin, Texas. Retrieved from http://cache.freescale.com/files/32bit/doc/app_note/AN3088.pdf

Sukhtankar, S., Hecht, D., & Rosen, W. (2004). A Novel Switch Architecture for High- Performance Computing and Signal Processing Networks. *3rd IEEE International Symposium on Network Computing and Applications* (pp. 215 - 222). Cambridge, MA, USA. Publisher: IEEE Xplore.

Terasic (2003-2015). *DE4 User Manual*. Publisher: Altera.

Xiangyang, L. (2013). Implementation and Transmission Error Handling of Multi-channel LVDS. *Fourth International Conference on Emerging Intelligent Data and Web Technologies* (pp. 466 – 470). Xi'an, China. Publisher: IEEE Xplore.

Xiao-yun, H., Hai-bing, S., In-zhang, W., & Wei, W. (2010). Multi-Processor Parallel System Based on High-Speed Serial Transceiver. *Second International Workshop on Education Technology and Computer Science* (pp. 178 - 181). Wuhan, China. Publisher: IEEE Xplore.

Yan, H., & Pan, H. (2010). The Design and Implementation of Network Data Link Layer Based on Embedded TCP/IP Protocol Stack. *International Conference on Networking and Information Technology* (pp. 227 - 230). Manila, Philippines. Publisher: IEEE Xplore.

Yong, J., & Qing-Sheng, H. (2011). *40Gbps multi-connection TCP/IP Offload Engine*. International Conference on Wireless Communications and Signal Processing (pp. 1 - 5). Nanjing, China. Publisher: IEEE Xplore.

Zhong-Zhen, W., & Han-Chiang, C. (2006). Design and Implementation of TCP/IP Offload Engine System over Gigabit Ethernet. *Proceedings of 15th International Conference on Computer Communications and Networks* (pp. 245 - 250). Arlington, VA, USA. Publisher: IEEE Xplore.