Transmission of High Bandwidth Data and Images with Integrated real-time Data Densification using a Scalable Network Model

Rekha P [1], Vishalakshi V [2], Chamaraju Y S [3]
[1] *Lecturer, Department of Electrical and Electronics Engineering, Government Polytechnic Arakere, Karnataka, India.*
[2] *Lecturer, Department of Electrical and Electronics Engineering, Government Polytechnic Channapatna, Karnataka, India.*
[3] *Lecturer, Department of Electronics and Communication Engineering, Government CPC Polytechnic Mysore, Karnataka, India.*

***Abstract*—Our civilization is increasingly reliant on the transmission and receipt of massive volumes of data across serial connections with ever-increasing bit rates. Even when contemporary serial buses with high data rates are utilized, the frame rate possible in imaging systems is frequently restricted by the serial link between a camera and a host. This article describes a scalable connection system with a bandwidth and interface standard that can be readily customized to fit a specific application. The scalable serial connection technique has been enhanced with lossless data compression, with the goal of boosting dataflow at a fixed bit rate. The compression mechanism is included into the scalable transceivers, giving a complete solution for effective data transfer over a range of interfaces. The system is entirely constructed on an FPGA, utilizing a totally hardware-based architecture. The system was implemented and tested using a Terasic DE4 board, with Quartus-II software and design and debugging tools. The impact of compressing the image and conveying the compressed data over parallel lines is comparable to that of compressing the same image within a single core with a greater compression ratio, which in this system ranges between 7.5 and 126.8.**

***Index Terms*—Bandwidth, Scalable network, Real-time data, Images, Serial buses.**

I. INTRODUCTION

Transferring high-resolution multi-spectral images between a camera or medical diagnosis device (e.g., CAT, MRI) and an offline processing system (host) can be particularly difficult, especially if transfer times must be shorter or equal to image frame time in order to avoid bottlenecks in data transfer. The bulk of current camera systems are limited by the transmission link speed to a host. One method for increasing the effective frame rate as received by a host is to apply image compression within the camera. If a fast image compression technology is available that can deliver compressed pictures in the time it takes to acquire a new frame, this strategy should enhance the effective frame rate when image data transport is the limiting constraint. Some firms have utilized a method that sends graphics to customers across numerous transceiver lines. The transceiver devices use SerDes technology, which is compatible with a variety of transceiver technologies like Ethernet, PCI Express, and ESATA/SAS. With these systems, data is sent from the source over transceiver lines without any format modifications or picture processing. For example, an x-ray detector with 2070 ×2167 pixels where each pixel is digitized with 32-bits per pixel that collects pictures at a frame rate of 750 Hz provides an uncompressed data rate of 108 Gbps. Such data speeds much exceed any single serial link that is currently available; most lines are limited to data rates ranging from 1 to 10 Gb/s. Because of transmission protocol overhead, the practical picture data rate is typically much lower than the promised bit rate, resulting in a data transfer bottleneck. In this study, we look at two techniques to addressing the ensuing transfer bottleneck. On the one hand, data is compressed using an optimized lossless compression technique, while on the other, band width is expanded by employing a flexible or scalable data connection strategy. To manage the high data rate, the system cannot be constructed using a standard sequential microprocessor-based system and software. Instead, we suggest developing

scalable multi-link system-based logic modules that are implemented on Field Programmable Gate Array logic (FPGA). A whole system is made up of an embedded system implemented on an FPGA on each side of a configurable number of transfer connections (see Fig. 1). Each system manages picture (de)compression methods, data (de)serialization, and a specific transport protocol. A typical system's design consists of a central processing unit (CPU) controlled by software and interfaced to hardware. The TCP/IP protocol is one of the most prevalent techniques for establishing a serial link using an Ethernet physical layer, which typically consists of 1-4 twisted wire pair connections. An embedded TCP/IP protocol refers to a project that is developed as an embedded system that uses TCP/IP as the standard protocol.

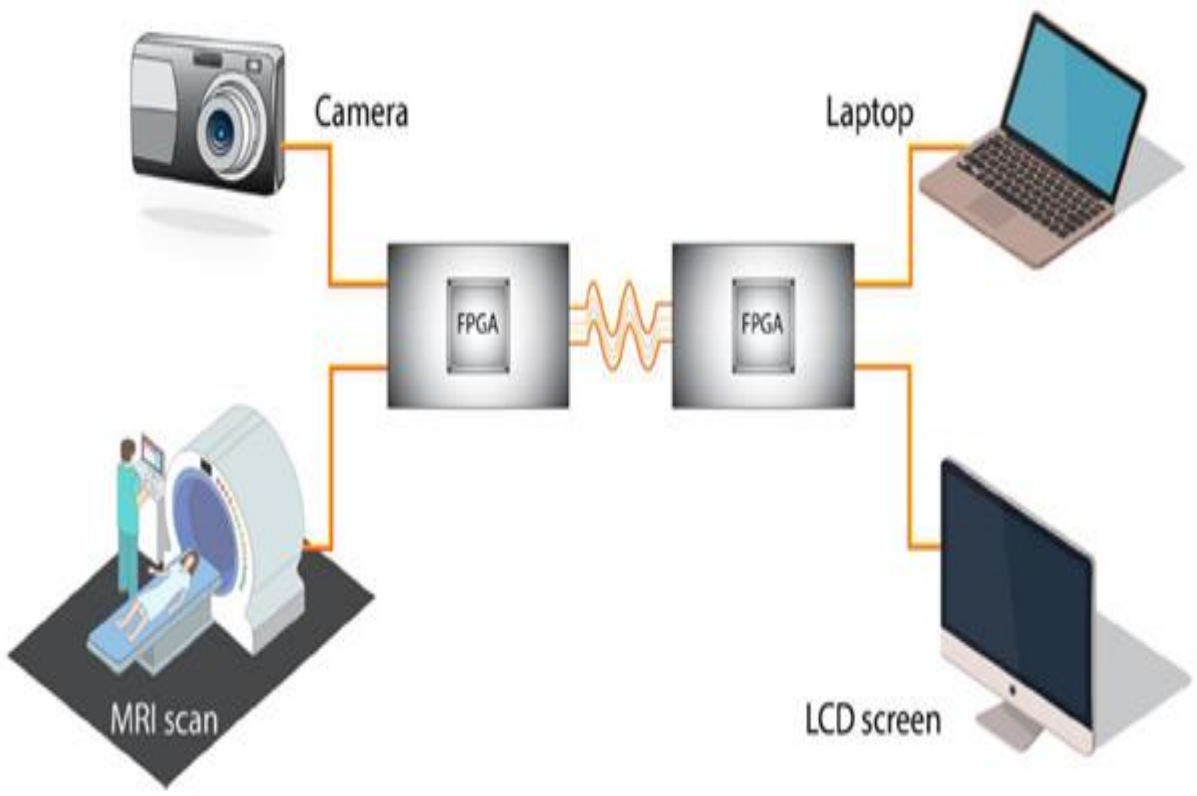
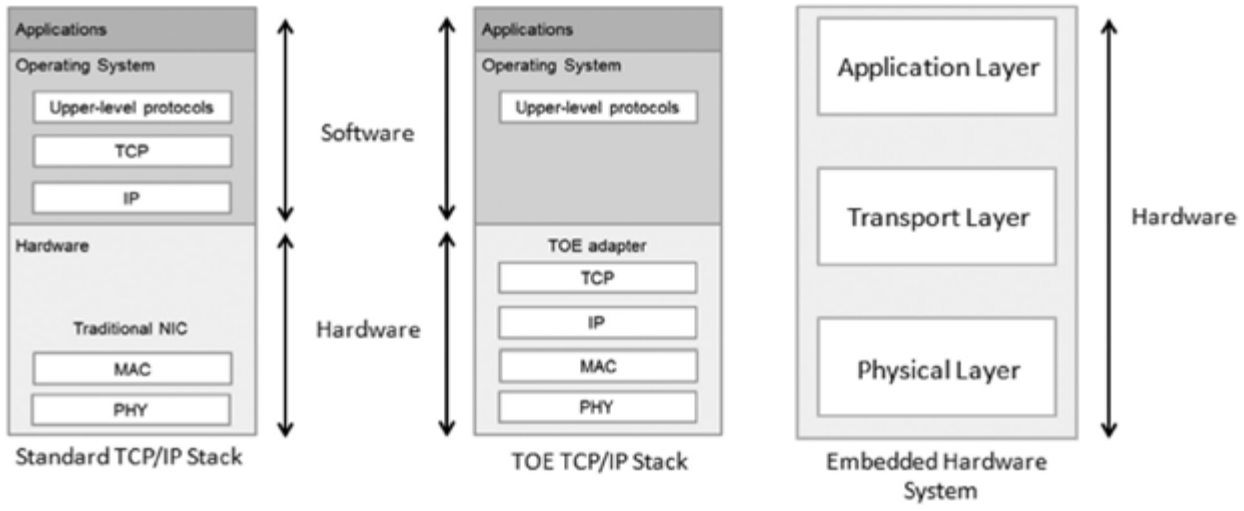


Fig. 1: An illustration of a scalable multi-link architecture that links various imaging sensors to client devices.

II. LITERATURE REVIEW

The transceiver devices use SerDes technology, which is compatible with a variety of transceiver technologies, including Ethernet, PCI Express [1-3], and ESATA/SAS standards [4-8]. With these systems, data is sent from the source over transceiver lines without any format modifications or picture processing. The embedded TCP/IP features a hierarchical protocol structure that is identical to the TCP/IP structure on a PC [9]. The primary traits are real-time, flexibility, and simplicity [10]. The embedded TCP/IP system is often created as a combination of software and hardware; the software component includes the CPU, which controls the entire system processor at a sluggish data rate, which can produce a CPU bottleneck. In recent years, an attractive solution to this problem has emerged: the TCP/IP Offload Engine (TOE), which reduces CPU overhead in order to enhance network efficiency and boost network connection throughput [11]. The increased Ethernet speed may handle speeds ranging from 10 Mbps to 10Gbps. Figure 2b depicts the TOE's architecture, which is based on traditional TCP/IP. Figure 2c depicts the architecture of the second type of embedded system, which is implemented as hardware with no software component (i.e., no CPU).



(a) (b) (c)

Fig. 2: Architecture of embedded software and hardware systems (a), (b), and an embedded hardware system (c)

III. METHODOLOGY

The Scalable Link Model with Compression (SLMC) is a new reference model that transports real-time pictures or data between cameras/detectors and clients, or a source and a destination, at high transmission rates. The SLMC is designed to serve a wide range of applications, including remote sensing camera applications that need data downsizing, high-speed data transmission, and real-time support.The SLMC splits transmitter and receiver systems into three levels, as seen in Fig. 3. Each layer has a distinct purpose, and when combined with the others, it creates a complete transceiver system. The physical layer interfaces with the real physical medium, the application layer establishes and manages the protocol for transmitting and receiving data, and the transport layer regulates the flow of data between the first two levels. One of the primary goals of the SLMC is to overcome the complexities of utilizing other reference models, such as TCP/IP and their associated protocols, as well as the restrictions of using a single transceiver link. A second goal is to boost effective transceiver speed by running many connections in parallel at the physical layer level for both the transmitter and receiver. A third goal is to minimize data volume by compressing at the transmission side and decompressing at the receiving end. A lossless compression approach is employed as part of the application layer. The SLMC provides for the control of several receiver applications at once, each of which is placed in its own domain, necessitating a few changes to the SLMC receiver side, particularly the transport and application layers.

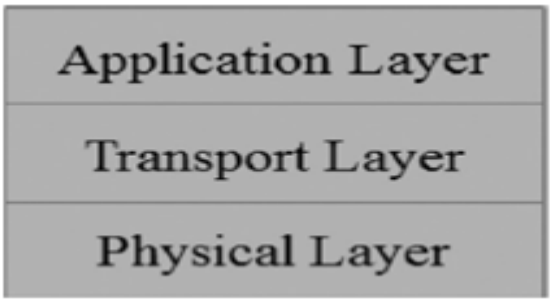


Fig. 3, SLMC architecture

IV. PROBLEM STATEMENT

The physical layer is one of the most critical components of the SLMC system. The layer's name indicates its true function, which is primarily to connect one or two applications in separate systems via physical media in order to exchange data. The physical layer contains two interface ports, one of which communicates with the rest of the embedded system within the FPGA and the other of which exports outside the FPGA to connect to transceiver ports and devices. The physical layer is made up of two sub-layers (see Fig. 4), each of which is implemented as an embedded system, with the bottom sub-layer interfacing the physical media. The

SLMC's primary structure allows it to handle a wide range of transceiver speeds, protocols, and devices, depending on the input/output (IOB) ports available for the main FPGA device needed to implement the embedded system. The SLMC physical layer is made up of many transceiver systems linked in parallel on both the transmitter and receiver sides. In general, each transceiver system may handle either 1 Gbps or 3.125 Gbps for a single transceiver speed. Different transceiver protocols offer these transceiver speeds to accommodate various physical media and transceiver devices. Data transfer via parallel nodes is critical, especially for systems that require fast data transmission rates. In SLMC systems, many transceiver protocols are employed, some of which (for example, Serial-Rapid-IO) are already based on the internal parallel structure and are adapted to work with the external parallel structure. Other protocols, like as the LVS and GIGE, which were originally designed to support a single transceiver system, have been updated to allow a parallel structure.

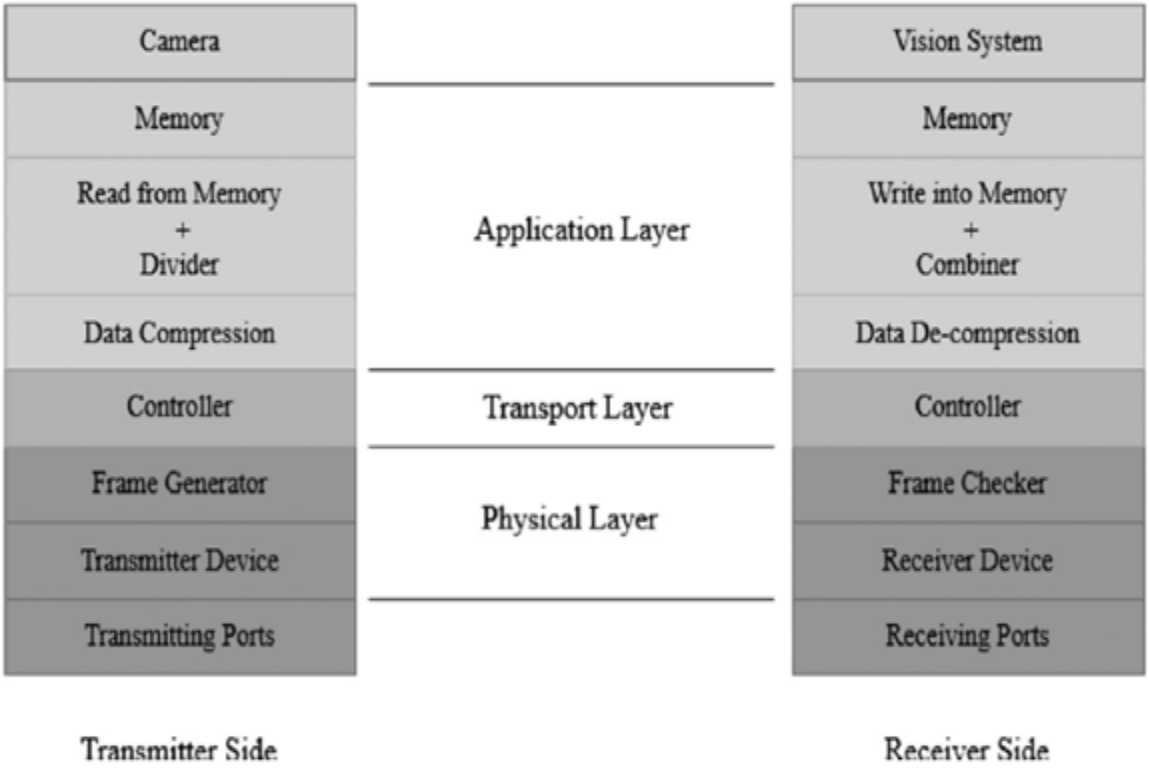


Fig. 4: An illustration of the suggested system

V. RESULTS OF TESTS AND EXPERIMENTS

The next sub-layer in the transceiver system is the 'transmitter device', which is made up of two cores, the PCS and the PMA, with a PMD in some situations. The PCS core is coupled to the frame generator and the PMA core. The PCS core encodes and forwards data to the PMA core. The 8/10 encoder is mostly employed in this system. The PCS generates low-speed parallel data, which is then transferred to the next core, the PMA, for conversion to high-speed data. The PMA core has a SER, which is used to transform low-speed parallel data into high-speed serial data, and the SER core has two input clocks, as seen in Fig. 5. The low-speed clock links to the PCS side and synchronizes parallel data, whereas the high-speed clock connects to the SER output core and triggers serial data. The low-speed clock is 125 MHz and is used to clock the whole system; the high-speed clock is 1 GHz. The low-speed clock connects to the PCS side and synchronizes parallel data, while the high-speed clock connects to the SER output core and initiates serial data. The low-speed clock is 125 MHz and is used to clock the whole system, while the high-speed clock is 1 GHz. It converts data from the physical medium or the PMD core to low-speed parallel data. This core has two input clocks, as seen in Figure 5. A

high-speed clock of 1 GHz links to the serial data side, while a low-speed clock of 125 MHz connects to the parallel data side.The low-speed parallel data generated by the PMA is sent to the PCS core, which includes a decoder. The decoding techniques used upon reception of data must be the same as those used before transmission; in this example, the 10/8 decoder is employed. The PCS creates 8-bit data with a 1-bit control signal and sends it to the frame checker. The frame checker then reads the data using the control signal. If the control value is 1, the incoming data is control data; alternatively, if it is 0, the data is original. The next step is to remove the header file added on the transmitter side and pass the data to the transport layer.

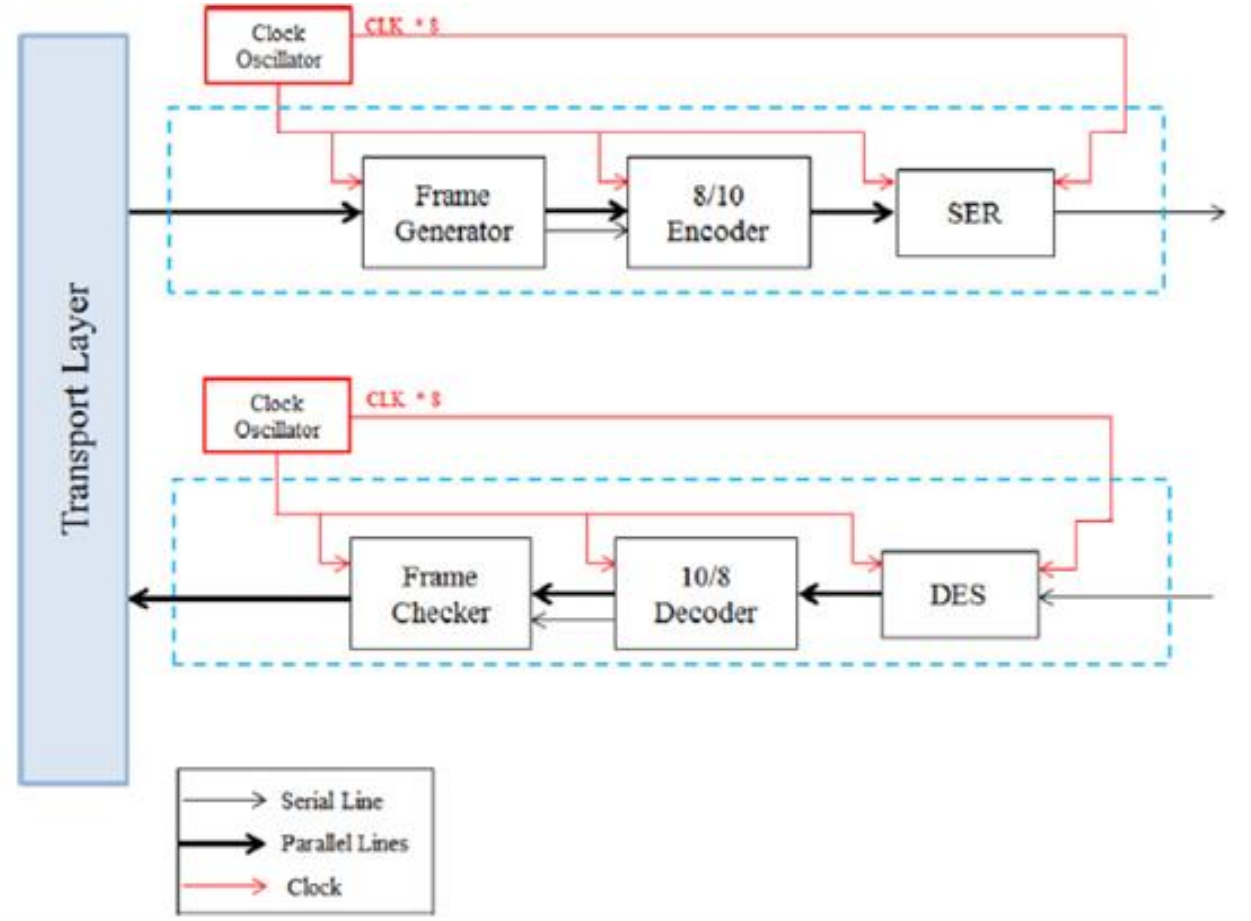


Fig. 5: Interior design of a single transceiver system using the GIGE transceiver protocol.

The system implementation is based on logic blocks (or IP cores) written in Verilog HDL using Quartus-II software and then implemented as an embedded system inside the FPGA, whereas the hardware implementation refers to the physical elements required, such as a board, an FPGA device, and transceiver ports. The SLMC systems based on different protocols were evaluated in two sections by measuring the compressed picture size, compression ratio, real execution time inside the system, compression time, transmission time, and pixel rate while using a sample of medical images. Both the compression and transmission times are determined using two embedded controllers, each of which has two triggers and is coupled in two distinct ways to obtain both values. To calculate the compression time, the first trigger is activated when the compression core receives the first pixel from the packet source, and the end-trigger is triggered when the final compressed pixel is finished being forwarded to the next core at the transport layer.To calculate transmission time, the start trigger is activated when the first pixel is read from the transport layer, and the end trigger is triggered when the final pixel is transferred to the physical media.

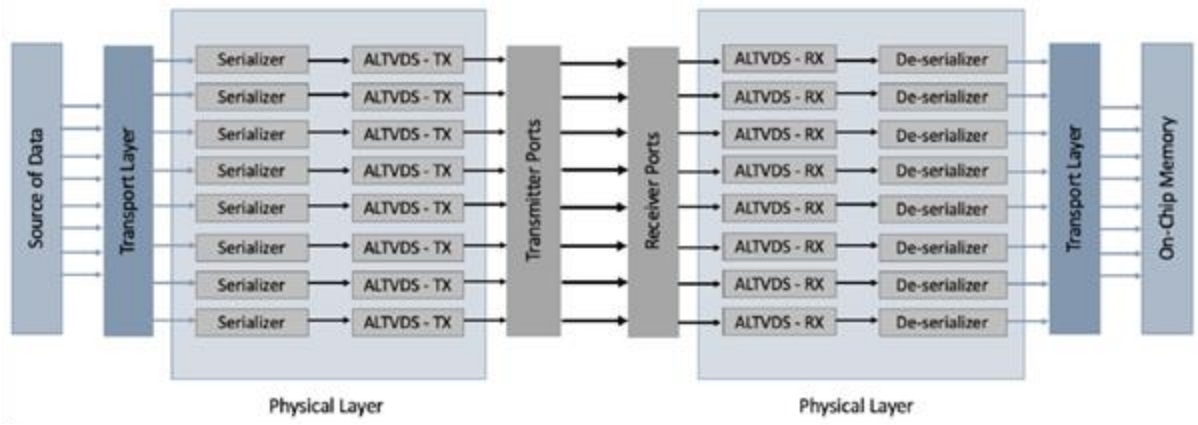


Fig. 6: An SLMC-LVS schematic diagram shows the transmitter and receiver ports

At the SLMC-LVS system depicted in Figure 6. The CR values varied from 0.9 to 15.9. Table 1 displays the transmission time, compression time, execution time, and pixel rate for the sample photos used. The compression time values ranged from 0.0621 ms to 0.148 ms, all of which were suitable for the real-time procedure, with an average value of 0.139 ms. The transmission time for the captured photos ranged from 0.004 ms to 0.069 ms, with an average of 0.065 ms. The transmission time was substantially less than the compression time because the number of cycles required by the system to compress the picture was significantly more than the number of cycles necessary to send the compressed image. The execution time for the captured photos ranged between 0.0623 ms and 0.15 ms, with 0.141 ms being the average; this number was also appropriate for the real-time process. The pixel rate (the number of pixels conveyed in one second) was calculated using the execution time. The pixel rate values varied from 0.436 Giga pixel/sec to 1.05 Giga pixel/sec, with 0.467 Giga pixel/sec being the average.

Table 1 The time it takes to compress, transmit, execute, and pixelate a sample of greyscale medical pictures using the SLMC-LVS system.

The frequency of:	Minimum value	Maximum value	Mean value
Compression-time (ms)	0.00622	0.148	0.139
Transmission-time (ms)	0.00411	0.0696	0.0655
Execution-time (ms)	0.0623	0.15	0.141
Pixel-rate (Giga pixel/*sec*)	0.436	1.05	0.467

The compression duration, compressed picture size, and CR for the applied medical photos were identical to those at the SLMC-LVS, allowing us to apply the same approach to compress data in both situations with the same input frequency.

VI. CONCLUSIONS

The SLMC was conceived and constructed as a new reference model for the multi-link system to transport pictures between two applications across physical media in real-time with high resolution and transceiver speeds. The SLMC is made up of two subsystems that transport and compress data. The transceiver subsystem has implemented transceiver protocols, such as the LVS, to match different applications, physical media, and speeds, whereas the compression system was implemented and used because the main data for the system was an image, in which case the application layer consisted of a compression method that would reduce the size of an image and thus increase the system's data rate. The SLMC physical layer was made up of eight single transceiver devices linked in parallel to provide faster data speeds. Scalability in the SLMC system may be done in two ways: through the compression system and the transceiver system. There are two techniques to reduce compression time in the compression system so that the picture may be processed in real time. First, when compressing a whole picture with a

single core, the compression time may be reduced by increasing the clock frequency; the maximum input frequency value is determined by the FPGA device. Second, the picture is divided into numerous sub-images, which are compressed in parallel utilizing multiple cores. On the one hand, using parallel lines to carry data instead of a single line increased the data rate by nearly eight times, while on the other hand, using various transceiver protocols that can work in different physical media and support various data rates triggered by different input frequency values, namely 125 MHz, 156.25 MHz, 195.3125 MHz, and 312.5 MHz. The number of transceiver systems linked in parallel is limited by the number of accessible output pins.

REFERENCES

[1] G. Sun, Z. He, in: A Real-Time Multi-Channel Signal Acquisition Card Based On PCI Express Interface, IEEE, Macau, 2009, pp. 20–24.

[2] G. Zhen, Y. Zhang, Y. Ren, in: The Design of PCI Express-Based Multi-Channel LVDS Signal Transfer Card, IEEE, Dalian, 2011, pp. V3.139–V3.141.

[3] Q. Wu, J. Xu, X. Li, K. Jia, in: The Research and Implementation of Interfacing Based on PCI Express, IEEE, Beijing, 2009, pp. 3.116–3.121.

[4] H.-. Y. Huang, L.-. W. Huang, W.-. S. Tseng, C.-. Y. Hsu, in: A 6-Gbit/s SATA Spread- -Spectrum Clock Generator Using Two-Stage Delta-Sigma Modulator, IEEE, Newport Beach, CA, 2008, pp. 333–336.

[5] W. Wu, H.b. Su, Q.z. Wu, in: Implementing a Serial ATA Controller Based On FPGA, IEEE, Changsha, 2009, pp. 467–470.

[6] R. Subramani, R. Penneru, G. Selvaraj, B. Radhakrishnan, in: Coverage Metrics for Device Level Validation of SATA and SAS Devices - An Approach, IEEE, Langkawi, 2014, pp. 163–168.

[7] W. Liu, X.-m. Zhao, B.-. J. Hu, X. Zhang, in: A High-Speed Large-Capacity Embedded Storage System Based On SATA, IET, Xi'an, 2013, pp. 1–6.

[8] A. Corporation, Implementing SATA and SAS Protocols in Altera, Altera, Devices, s.l., 2012.

[9] Ri-Kun Liao, Yue-Feng Ji, Hui Li, in: Optimized Design and Implementation of TCP/IP Software Architecture Based on Embedded System, IEEE, Dalian, 2006, pp. 590–594.

[10] Yan Hongwei, Pan Hongxia, in: The Design and Implementation of Network Data Link Layer Based on Embedded TCP/IP Protocol Stack, IEEE, Manila, 2010, pp. 227–230.

[11] S. Kim, C. Park, S. Kim, Y. Chung, in: The Offloading of Socket Information For TCP/IP Offload Engine. International Conference on Advanced Communication Technology, IEEE, 2009, pp. 826–831. 2009. ICACT.

[12] R.B. Mohammed, Novel Scalable and Real-Time Embedded Transceiver System, Uni- versity of Manchester, 2015 PhD thesis.

[13] Zhong-Zhen Wu, Han-Chiang Chen, in: Design and Implementation of TCP/IP Of- fload Engine System over Gigabit Ethernet, IEEE, Arlington, 2006, pp. 245–250.

[14] E. Chang, C. Wang, C. Liu, K. Chen, C Chen, Virtualization Technology for TCP/IP Offload Engine, IEEE Trans. Cloud Comput. (2014) 117–128.

[15] Z. Ye, H. Q. and P. Weijun, "Application of High-Speed Serial Data Transmission System in Remote Sensing Camera," in MATEC Web of Conferences, 2015.

[16] Y.A. Cengel, Heat Transfer: a Practical Approach with EES CD, McGraw-Hill Higher Education, 2002.

[17] J. Kim, Spray cooling heat transfer: the state of the art, Int. J. Heat Fluid Flow 28 (4) (2007) 753–767.

[18] W.-L. Cheng, Q.-N. Liu, R. Zhao, H.-l. Fan, Experimental investigation of parameters effect on heat transfer of spray cooling, Heat Mass Transf. 46 (8) (2010) 911–921.