HIGHLY EFFICIENT MULTI-GIGABIT COMMAND-STATUS PACKET TUNNELING TECHNIQUE IN INTER-FPGA PACKET STREAMING ARCHITECTURE

LOH MUI SOON

UNIVERSITI SAINS MALAYSIA

2016

HIGHLY EFFICIENT MULTI-GIGABIT COMMAND-STATUS PACKET TUNNELING TECHNIQUE IN INTER-FPGA PACKET STREAMING ARCHITECTURE

By

LOH MUI SOON

A Dissertation submitted for partial fulfilment of the requirement for the degree of Master of Science (Electronic Systems Design Engineering)

August 2016

Acknowledgment

First and foremost, I would like to take this opportunity to express my gratitude and thanks to Dr Patrick Goh Kuan Lye from the School of Electrical and Electronics Engineering, University Science of Malaysia, for his relentless guidance and courage throughout the undertaking of this research project. Many constructive feedbacks and advices are given, besides sacrificing his valuable time travelling to my company, which helped to make this research run smoothly.

I also would like to thank Keysight Technologies for providing the computer infrastructures including high performance computing workstation and engineering design software licenses for all simulations demonstrated in this research and the Ministry of Higher Education of Malaysia for financial sponsor in MYBRAIN15 program.

Last but not least, I also would like to express my gratitude to all my beloved spouse, parents and fellow course mates who have supported or encouraged me throughout the undertaking of this two-year master studies program in MSc in Embedded System Design Engineering in University Science of Malaysia.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Abbreviations and Nomenclature

Abbreviation	Meaning
ACK	Acknowledge
ADC	Analog-to-Digital Converter
ASIC	Application Specific Integrated Circuit
ASM	Algorithmic State Machine
CDR	Clock and Data Recovery
CMD	Command
CMD-Status	Command and status
CSPT	CMD-Status Packets Tunneling
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
EOP	End-of-packet
FIFO	First-In-First-Out
FPGA	Field-Programmable Gate Array
Gbps	Gigabit-per-second
GTP	Gigabit-Transceiver P-series
HDL	Hardware description language
I2C	Inter-Integrated Circuit
ITK-MAC	Interlaken-to-MAC
LSB	Least Significant Bit
LVDS	Low-voltage differential signaling
MAC	Media Access Controller
MGT	Multi-Gigabit Transceivers
MISO	Master-In-Slave-Out
MOSI	Master-Out-Slave-In
MSB	Most Significant Bit
PCB	Printed Circuit Board
PCS	Physical Coding Sublayer
PISO	Parallel Input to Serial Output
PLB	Processor local bus
PLL	Phase-Lock-Loop
PMA	Physical Medium Attachment
RTL	Register-Transfer-Level
SerDes	Serializer-Deserializer
SIPO	Serial-In-to-Parallel-Output
SNK	Sink
SOP	Start-of-packet
SRC	Source
SPI	Serial Peripheral Interface
SRC	Source
TAT	Turn-around time
TDM	Time-Division-Multiplexing

TEKNIK PENEROWONGAN PAKET PERINTAH DAN STATUS GIGABIT BERGANDA YANG BERKECEKAPAN TINGGI DALAM SENI BINA PENGALIRAN PAKET ANTARA FPGA

Abstrak

Protokol siri berkelajuan tinggi yang dibina dengan penghantar-terima gigabit berganda dalam FPGA adalah tulang belakang untuk industri-industri komunikasi data. Protokol ini kini merupakan satu keperluan asas untuk aplikasi hari ini dan juga untuk keperluan sistem generasi akan datang. Walau bagaimanapun, pemindahan paket perintah dan status dengan cekap dalam kawalan pautan antara FPGA melalui penghantar-terima gigabit berganda tanpa pembaziran jalur lebar data telah menjadi satu cabaran dalam reka bentuk sistem moden. Walaupun teknik pemultipleksian pembahagian masa boleh digunakan untuk menangani isu ini, slot-masa khusus yang tidak digunakan untuk kawalan pautan paket memberikan kesan negatif kepada kecekapan jalur lebar dan seterusnya prestasi sistem. Satu lagi teknik penyelesaian yang biasa digunakan adalah dengan permindahan paket perintah dan status dalam pautan kawalan berasingan; biasanya dilaksanakan dengan protokol siri yang berkelajuan rendah hingga sederhana seperti I2C dan SPI. Walaupun penyelesaian ini boleh dilaksanakan dengan pemacu peranti yang sedia ada, pembaziran kependaman akan berlaku, contohnya seperti dalam pemindahan 512-bait data penjodoh untuk mengkonfigurasikan penapis-FIR yang mempunyai 128-tap. Dalam disertasi ini, satu seni bina berkependaman rendah and cepat untuk penerowongan paket perintah and status dalam protokol siri Interlaken berkelajuan 25 Gbps untuk penyaluran data antara FPGA yang berkelajuan gigabit berganda telah dicadangkan. Hasil simulasi menunjukkan bahawa seni bina yang dicadangkan beroperasi dengan kecekapan yang tinggi, hanya mengunakan 13.21% dan 10.34% daripada kitar jam yang diperlukan berbanding implementasi konvensional SPI dan I2C. Seni bina ini juga mengekalkan keserasian dengan pautan kawalan yang dilaksanakan secara berasingan menggunakan SPI atau I2C untuk mempermudahkan reka bentuk keseluruhan sistem, mengurangkan risiko pembangunan produk dan kos sistem.

HIGHLY EFFICIENT MULTI-GIGABIT COMMAND-STATUS PACKET TUNNELING TECHNIQUE IN INTER-FPGA PACKET STREAMING ARCHITECTURE

Abstract

High speed serial protocols build upon multi-gigabit transceivers in the FPGA are the backbone of data communication industries. These protocols are now a fundamental requirement for today's applications as well as addressing the needs of next generation systems. However, transferring command and status packets explicitly in inter-FPGA control links efficiently via multi-gigabit transceivers without wasting data bandwidth become a challenge when architecting a modern design. Though time-division-multiplexing techniques could be deployed to address this issue, dedicated but unused time-slots for control link packets adversely affect the bandwidth efficiency and thus the system performance. Another common solution focuses on transferring the command and status packets in a separate control link; typically implemented in low to medium bandwidth serial protocols such as I2C and SPI. Though this solution is simple to implement in hardware with readily available device drivers, an unnecessary high latency overhead is introduced such as when transferring a 512-byte filter data coefficients to configure a 128-tap FIR filter. In this dissertation, a highly efficient, low latency command-status packet tunneling architecture built on top of the 25 Gbps Interlaken serial protocol for multi-gigabit inter-FPGA data streaming is proposed. Simulation results show that the proposed architecture works successfully with a high efficiency, utilizes only 13.21% and 10.34% of the clock cycles required in the conventional SPI and I2C implementations respectively. The proposed architecture also maintains a backward compatibility with control links implemented separately using SPI or I2C serial protocols to simplify the overall system design, reduce product development risks and system costs.

CHAPTER 1

INTRODUCTION

1.1 Overview

Today, with the establishment of advanced fabrication process technology at much smaller process nodes, multi-gigabit high-speed transceiver becomes very common in many modern application specific integrated circuits (ASIC) and high-end field-programmable gate array (FPGA) devices. With increasing demand of very high data transfer rate, high speed serial protocol today has been established to operate in the multi-gigabit-per-second (Gbps) range to overcome many limitations faced in using parallel bus architectures. High speed transceiver is the backbone of wireless, wireline and data center industries. They are responsible for moving massive amount of data from one point to another known as point-to-point technology. Transmitting and receiving huge amount of data through a single package is now a fundamental requirement for today's applications as well as addressing the needs of next generation systems. The multi-gigabit transceiver available in high-end FPGAs such as Altera Aria-10 or Xilinx's Kintex-7 FPGAs is uniquely position to address this need as a single package solution providing up to eight terabits-per-second (8-Tbps) of total aggregate bandwidth.

The design of high speed data packet transfer in modern systems such as industrial data acquisition system (DAQ), software defined networking (SDN) [1] or network function virtualization (NFV) architecture often involved multi-gigabit transceiver links between target and initiator chips in the form of FPGA or ASIC. The

importance and popularity of FPGA has increased nowadays due to the programmability that FPGA devices could offer. Using FPGA device, system designer can design and simulate their system using register-transfer level (RTL) design entry and implement the synthesized RTL netlist into FPGA device using state-of-the-art electronic design automation (EDA) tool provided by the FPGA vendors. Any design bug found later after being implemented could be fixed by re-programming the FPGA device, this in turn provide a short turn-around time (TAT) for bug fixing compared to ASIC which require a wafer mask spin which could take weeks to months. The short TAT for a product design using FPGA also provides other benefits, such as fast time-to-market, flexibility to deal with last minute changes and low product development risks as a system implemented within a FPGA device could be thoroughly validated for correctness. Also product design using FPGA incurs low non-recurring engineering (NRE) cost as wafer mask spin is not required since FPGA devices are normally bought off-the-shelf.

Leveraging on the inter-FPGA multi-gigabit transceiver (MGT) [2] link, high-speed data streaming at a transfer rate in the multiple of Gbps is practical [3] and often become part of the requirement for modern product design. However, transferring command and status packets used efficiently in modern system designs involving multi-gigabit transceiver links, without causing data transfer bandwidth wastage, is become a challenge when architecting a new system design. The state-of-the-art technique for transferring command-status packets within the multi-gigabit high-speed channel is based upon time-division-multiplexing (TDM) [4],[5],[6] where certain percentage of the multi-gigabit channel bandwidth is allocated for command and status (CMD-Status) packets as inter-chip control link. This specially allocated

time-slot for CMD-Status packets is typically fixed apart from the total available data bandwidth for multi-gigabit data streaming across two different chips or FPGAs.

Another common industrial solution normally focuses on separating command and status packets in a separate control link from high speed channels used for multi-gigabit data streaming. The control links for inter-chip communication are typically implemented using low to medium bandwidth serial protocol based upon state-of-the-art Inter-Integrated Circuit (I2C or pronounced I-Squared-C) [7], [8] and Serial Peripheral Interface (SPI) bus [8].

A highly efficient command and status packets tunneling architecture for inter-FPGA packet streaming to address the problem of idling time, bandwidth wastage is desirable. Designing this novel command and status packets tunneling architecture is highly desirable and this will be the main focus of this research.

1.2 Problem Statements

The conventional state-of-the-art technique for transferring command-status packets for inter-FPGA communication using the multi-gigabit transceivers is based upon the time-division-multiplexing (TDM) technique, where a fixed percentage of the multi-gigabit channel bandwidth is allocated for CMD-Status packets in inter-FPGA control link. This specially allocated time-slot for CMD-Status packets is typically fixed apart from the total available channel bandwidth and results in a limitation to the total number of CMD-Status packets to be transferred at any one time. This technique also degrades the total payload efficiency of the multi-gigabit channel where the unused bandwidth specially allocated for CMD-Status slot become a total

wastage. Furthermore, the turnaround latency involved to tunnel CMD-Status across the multi-gigabit channels using TDM technique is long, un-deterministic and is highly dependent upon the queue of time-slot allocated for CMD-Status packets; which is time-multiplexed with the high-speed streaming data packets.

Typical industrial solution focuses on separating CMD-Status packets used as control link from the high-speed multi-gigabit-per-second data streaming channel and implemented separately in the state-of-the-art I2C or SPI serial interfaces. This solution, though is simple to implement using both hardware and readily available standard device drivers for SPI and I2C in the firmware development, it increases complexity for synchronization in user applications development. Increased in complexity will become obvious when a continuous configuration data stream from the control link is required to fine tune a Finite Impulse Response (FIR) filter used as equalizer, sweeping across a wide range of frequencies for an analog-to-digital converter (ADC) feeding digital data to a high-end FPGA at data rate of 600MB/s.

Separating CMD-Status packets used in control link implemented using I2C or SPI serial interfaces also increases idling time for the target and initiator functions implemented in two individual high-end FPGAs. These FPGAs waiting for control command or status from each other in order to make important decision while performing high-speed data streaming. Furthermore, in a modern system design, high-end FPGAs are capable of transferring data in the rate between 3.125-Gbps to 150-Gbps. Idling time incurred in waiting for slow or medium bandwidth I2C or SPI serial interfaces used for CMD-Status communication to make critical decision before

next operation could continue will become a final system bottleneck to be experienced by the end user.

A highly efficient CMD-Status packets tunneling architecture for inter-FPGA control link leveraging on multi-gigabit serial protocol for packet streaming to address the problems of bandwidth wastage and inter-FPGA idling time is highly desirable and this will be the main focus of this research.

1.3 Objectives

The objectives of this research are:

i. To design a command-status (CMD-Status) packet tunneling architecture to tunnel CMD-Status packets across streaming data for a system involving inter-FPGA high-speed data transfer utilizing multi-gigabit transceiver links.

ii. To significantly improve overall turn-around time in terms of read and write latency measured in clock cycles, for CMD-Status packets used in control link with a highly efficient burst read-write method.

1.4 Project Scopes

There are many ways to design a highly efficient command-status (CMD-Status) tunneling architecture depending on the serial protocol used to pair with the multi-gigabit transceivers within a high end FPGA. This research project focuses on the design of the proposed CMD-Status tunneling architecture build on top of Altera

25G Interlaken IP Core [9] with a 4-lane configuration, each lane with speed of 6.25 Gbps resulting in a 25 Gbps inter-FPGA high-speed link.

The design on the CSPT architecture will be done in Verilog hardware description language (HDL). Functional simulation will be performed to validate the design using Altera Quartus II [10] FPGA design and simulation tool with Altera 25G Interlaken IP Core behavioral model.

The efficiency in terms of read-write latency measured in the number of clock cycles for CMD-Status packet tunneling will be measured and benchmarked against conventional SPI and I2C implementations. Backward compatibility in the design of this proposed CMD-status tunneling architecture with conventional SPI-based CMD-Status control link will be maintained.

1.5 Research Contribution

This proposal for developing a CMD-Status packet tunneling architecture for inter-FPGA control link leveraging on multi-gigabit serial protocol for high-speed data streaming could address the problems of bandwidth wastage in the multi-gigabit channel and minimizing inter-FPGA idling time. It will also simplify the overall system architecture design by eliminating the SPI and I2C serial interfaces. This will eventually lead to minimizing system cost and reduce product development risks involving control link built on top of the multi-gigabit inter-FPGA communication.

1.6 Thesis Organization

The remainder of this dissertation is organized as follows:

Chapter 2 reviews the suitable high-speed serial protocols for point-to-point data transfer and various low to medium bandwidth serial interfaces including I2C, SPI and time-division-multiplexing method used in control links.

Chapter 3 describes the overall design methodology of this research starting with high-level view of FPGA-to-FPGA data streaming with CMD-Status packet tunneling architecture, RTL-based design flow, design methodology for the SPI-based implementations and the CSPT architecture. It then goes into details of design for all critical blocks used in SPI implementations and the CSPT architecture. This chapter ends with a chapter summary outlining the leveraging on reusable blocks from various stages, simplifying the design and simulation efforts by half.

Chapter 4 begins with a high level simulation setup, followed by flowcharts to summarize the operations of various implementations. Then, simulation results showing read and write latency for SPI-enhanced implementation and CMD-Status tunneling operation of the CSPT architecture will be presented. This chapter ends with an analysis and discussion on the performance comparisons across four different implementations.

Chapter 5 summarizes and concludes the results from the performance comparisons across four different implementations described in Chapter 3 and outlines future recommendations for improvement related to this research.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter starts with some in depth study on high-speed serial interconnects widely adopted in design based upon multi-gigabit transceiver (MGT) technologies from Altera or Xilinx; the two leading Field-Programmable Gate Array (FPGA) device vendors. Reasoning on using serial interconnects versus wide parallel bus is explained. Example applications using MGT in inter-FPGA system design are illustrated with important considerations for state-of-the-art implementations. It is followed by exploring suitable high-speed serial protocols for inter-FPGA point-to-point data transfer. The three main serial protocols, SerialLite III from Altera, Aurora v3.0 from Xilinx and lastly the vendor-independent Interlaken protocol are explored with particular interests on Interlaken due to the highly scalability, high payload efficiency and back pressure support this protocol could offer. Time-division-multiplexing (TDM) architecture is being explained with main focus on feasibility study and the efficiency for command-status tunneling. Thereafter, three low to medium bandwidth serial interfaces: I2C and SPI are discussed briefly, with focuses on their performance and suitability to be used to send command and retrieve status in control link between two FPGAs. Before chapter summary, a section to discuss about recent researches on inter-chip communication or control links is presented. This chapter ends with a chapter summary comparing all three main multi-gigabit serial protocols and comparison of I2C and SPI for control link applications and reason to choose them for this research.

2.2 High-Speed Serial Interconnects in FPGA

The traditional interconnects between FPGA devices are normally realized in the form of parallel bus. To increase data rate for the parallel bus, it requires higher clock rate and wider bus. However, using higher clock rates or wider parallel bus compromises the parallel bus performance as it increases the skew (different signals arriving at different times) and reduces the timing budget [11]. Furthermore, a wider parallel bus is more difficult to layout and route in printed circuit board (PCB). The ever-increasing data transfer throughput requirements have resulted in multi-load parallel, single ended IO bussing schemes converting into point-to-point serial low-voltage differential buses [12].

High-speed serial interconnects benefit from low-voltage differential signaling (LVDS) up to gigabit-per-second or the pseudo current-mode logic (PCML) IO standard for multi-gigabit data rate. These interconnects use embedded clocking technologies to overcome clock and data skews problems usually faced by wide parallel buses. This makes it possible to easily achieve signaling rate in the range of a few gigabit-per-second (Gbps) to over 10s' of Gbps of bandwidth. In addition, high-speed serial interconnects also lead to reduction in terms of the total number of IO pins required for a chip and this leads to simpler PCB wiring and contribute to overall system cost savings [13] for a product designed using these interconnects.

Increasingly common in ASIC and FPGA designs are the ultra-fast serial communication channels known as multi-gigabit transceivers (MGT), used for high-speed data streaming over a backplane between FPGA-to-FPGA, or over longer

distances. The popular MGT is the GTP transceiver [14] from Xilinx's Virtex 5 FPGA family which is compatible with the GTX transceivers embedded in Xilinx's Spartan-6 FPGA devices as illustrated in Table 2.1. There are six types of MGTs in Xilinx's FPGA: GTR, GTP, GTX, GTH, GTY and GTZ. The GTH and GTY transceivers provide low jitter required for optical interconnects and auto-adaptive equalization features, the rest are just the names with different speeds in different device families.

Table 2.1. Xilinx multi-gigabit transceiver offerings [19].

Xilinx's FPGA Devices	**MGT Type**	**Max Performance[1]**	**Max Transceiver**	**Peak Bandwidth[2]**
Virtex UltraScale+	GTY	32.75	128	8,384 Gb/s
Kintex UltraScale+	GTH/GTY	16.3/32.75	44/32	3,268 Gb/s
Virtex UltraScale	GTH/GTY	16.3/30.5	60/60	5,616 Gb/s
Kintex UltraScale	GTH	16.3	64	2,086 Gb/s
Virtex-7	GTX/GTH/ GTZ	12.5/13.1/ 28.05	56/96/16	2,784 Gb/s
Kintex-7	GTX	12.5	32	800 Gb/s
Artix-7	GTP	6.6	16	211 Gb/s
Zynq UltraScale+	GTR/GTH/ GTY	6.0/16.3/ 32.75	4/44/28	3,268 Gb/s
Zynq-7000	GTX	12.5	16	400 Gb/s

Note: [1]: Gbps [2]: Combined transmit and receive

Figure 2.1 shows the transmitter and receiver units embedded in each GTP transceiver inside the Xilinx's FPGA. GTPs are represented as configurable hard-macros or more commonly referred to as a tile. Each tile includes a pair of transceivers, which share the reset logic and also a phase lock loop (PLL) as basic components to this block. The main components of this GTP transceiver are the physical medium attachment (PMA) sublayer, which is serializing the data for the transmitter and de-serializing the data for the receiver unit, and a physical coding sublayer (PCS), which

processes the data before serialization and after de-serialization. The shared PLL will be used to lock to a reference clock (CLKIN) and generates the high-speed serial clock for the transmitter. Also, the same PLL is used to generate the parallel clock (XCLK) for the parallel input to serial output (PISO) and a seed clock for the clock and data recovery (CDR) circuit.

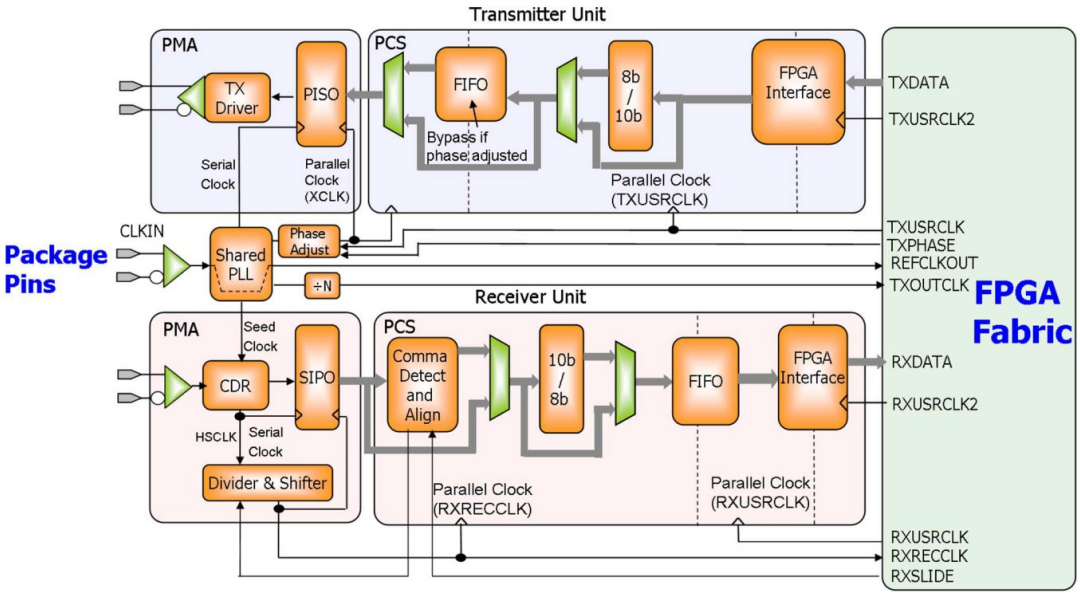

Figure 2.1. Simplified block diagram of a GTP transceiver. Half a tile is shown [15].

Figure 2.2 below shows an example implementation for a high-speed multi-gigabit serial link based on a GTP transceiver running at 2.5 Gbps utilizing the GTP SerDes with 8b10b coding because it is the most widely adopted coding standard for serial data in applications such as Gb-Ethernet and FibreChannel [16]. Similarly, the 2.5 Gbps data-rate is commonly used in optical transmissions standard such as OC-48/STM-16x/2.5 G SONET [16].

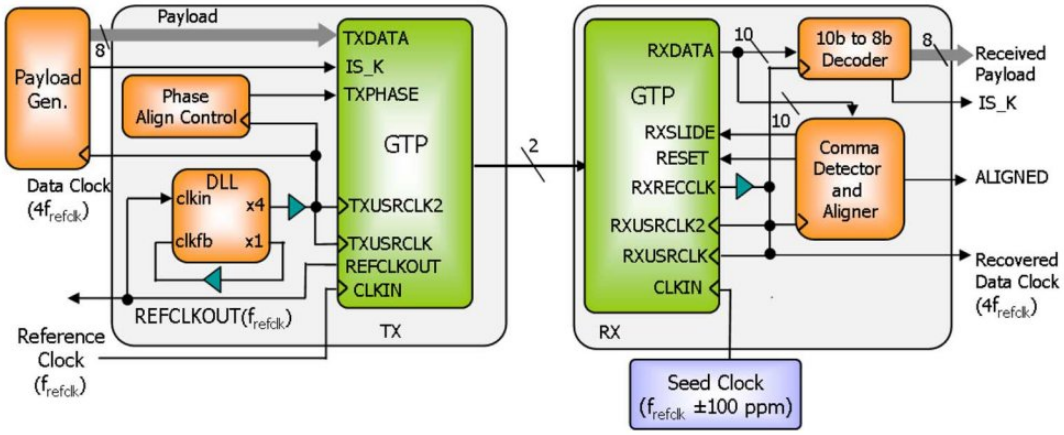

Figure 2.2. A high-speed serial link based on a GTP transceiver [15].

Figure 2.3 shows a MGT from Xilinx's Virtex 5 FPGA family known as GTP used for FPGA-to-FPGA high-speed data transfer.

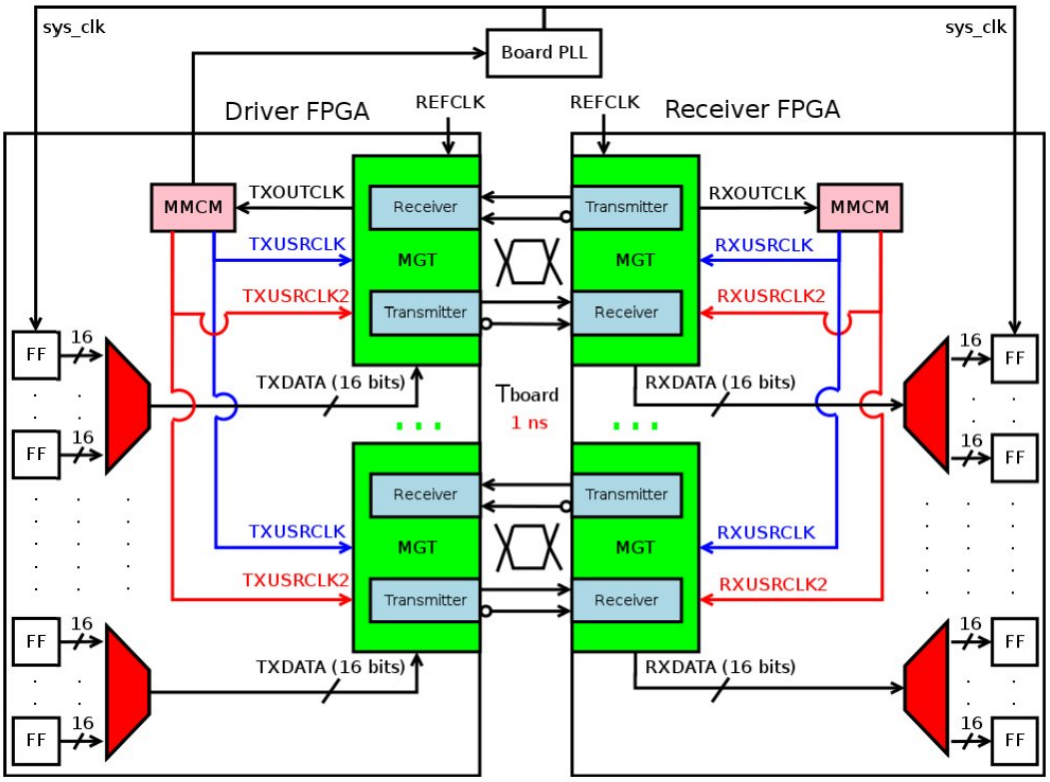

Figure 2.3. MGT for high-speed Inter-FPGA communication [4].

Synchronization of the system clock and the fast clock is achieved based on the self-synchronous [17] method where the data stream contains both the data and the clock.

Both TXOUTCLK and RXOUTCLK are fed by the same reference clock so that they are constantly remain synchronous with the same frequency.

The multi-gigabit high-speed serial interconnects commonly used for inter-FPGA point-to-point data transfer are SerialLite, Aurora and Interlaken [24] links, these multi-gigabit serial protocols are lightweight in which these interconnects use minimal FPGA area thus permits use of smaller and cheaper FPGAs. These multi-gigabit serial protocols also do not require complex user software drivers at hardware abstraction layer where applications can connect to the FIFO-like user interface to simplify modern system design.

2.3 SerialLite High-Speed Serial Protocol

The Altera SeriaLite III streaming IP core is for high speed serial communication protocol for chip-to-chip, board-to-board and backplane applications for data streaming. This IP core consists of a physical coding sublayer (PCS), a physical media attachment (PMA) and a media access control (MAC) block. The interfacing with FPGA fabric is through the Avalon-ST interface. The SerialLite III protocol as shown in Figure 2.4 offers high-bandwidth, low overhead frames, low I/O count, and supports scalability in both number of lanes and lane speed.

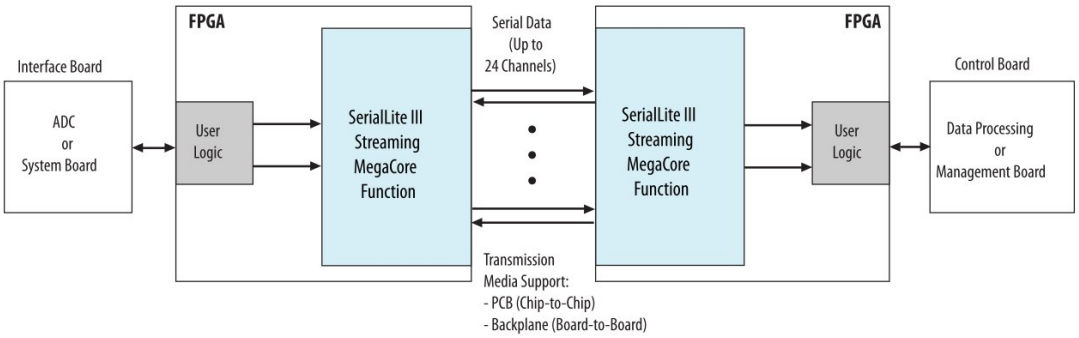

Figure 2.4. Typical system application using Altera SerialLite III [34].

The protocol features supported by the SerialLite III for the transfer of high bandwidth streaming data over unidirectional or bidirectional high-speed serial link are simplex and duplex operations. This serial protocol supports single or multiple lanes, 64B/67B physical layer encoding, payload and idle scrambling, error detection and low latency for point-to-point data transfer.

The SerialLite III supports continuous and burst mode according to application needs. In the continuous mode, user data is transmitted over the link and received at the receiving link at the same rate without gaps in the stream on the user interface, as if a data pipe is transparently forward all data presented on the user interface to the far end of the link.

Continuous mode is used commonly for applications which require a simple interface to transmit a single, high bandwidth data stream. Example of application is the sensor data links for radar and wireless infrastructure, where data converters can connect to either end of the links with minimal interface logic, and both ends of the link operates from a same transceiver reference clock.

When operating in burst mode, the SerialLite III streaming link accepts bursts of data across the user interface and transmits each burst across the link as a discrete data burst. This operating mode is suitable for applications where the data stream is divided into bursts of data such as the uncompressed digital video, where the data stream is divided into lines of display raster. The key advantage for burst mode is it supports multiplexing of multiple data streams across the link and more flexibility to the clocking.

Table 2.2. SerialLite III performance and resource utilization comparison [34].

Direction	Clocking Mode	Number of Lanes	ECC	Arria 10 - Parameters			Stratix V GX - Parameters		
				Per-Lane Data Rate (MBps)	ALMs	M20K	Per-Lane Data Rate (MBps)	ALMs	M20K
Source	Standard	24	Disabled	17400	2596	39	10312.50	5830	39
	Standard	24	Enabled	17400	8176	72	10312.50	11136	72
	Advanced	24	Disabled	17400	2990	39	10312.50	2990	39
	Advanced	24	Enabled	17400	6706	72	10312.50	11146	72
Sink	Standard	24	Disabled	17400	3881	49	10312.50	5461	49
	Standard	24	Enabled	17400	3881	50	10312.50	5489	50
	Advanced	24	Disabled	17400	3167	0	10312.50	4265	0
	Advanced	24	Enabled	17400	3167	0	10312.50	4265	0
Duplex	Standard	24	Disabled	17400	6258	88	10312.50	8845	88
	Standard	24	Enabled	17400	11184	122	10312.50	14198	122
	Advanced	24	Disabled	17400	5706	39	10312.50	7548	39
	Advanced	24	Enabled	17400	8821	72	10312.50	12619	72

In standard clock mode, the SerialLite III Streaming IP core operates in a pure streaming mode, copying the sink input data and forward to the source output directly. User clocks to drive the user interface for both the source and sink are generated by Altera Quartus Prime software automatically. In advanced clocking mode, Quartus Prime allows user to specify clock to interface with the source core. This clocking mode will be very useful when part-per-million (PPM) differences between the user clock generated by the IO PLL or fractional Phase-Lock-Loop (fPLL) and the user clock are beyond tolerable range.

Table 2.2 above compares the typical resources and expected performance for different SerialLite III Streaming IP core variations for Arria 10 and Strtix V GX FPGA devices operating in standard and advanced clocking modes. The resource

usage in terms of adaptive logic modules (ALMs) on advanced clocking mode is about 20% less as compared to standard clocking mode. This gives high-level user flexibility to design their system using advanced clocking mode in order to minimize resource usage.

2.4 Aurora High-Speed Serial Protocol

The Aurora protocol [29] developed by Xilinx is an open link-layer protocol based on 8B/10B coding for point-to-point communications between FPGAs. This protocol also defines control words for framing separation or idle periods and implements the control of the MGT inside the FPGA. It also supports initialization of the link, error handling, flow control, etc, which are transparent to the higher-level users.

Figure 2.5 illustrates the Aurora protocol, Xilinx's IP core introduced on Virtex®-II Pro and Virtex-4 FX FPGAs, serial communications protocol for multi-gigabit links used to transfer data between devices using one or many MGTs running at any supported line rate to provide a low cost, general purpose, data channel with throughput from 622 Mbps to over 100 Gbps. The connections between devices can be full-duplex (data in both directions) or simplex.

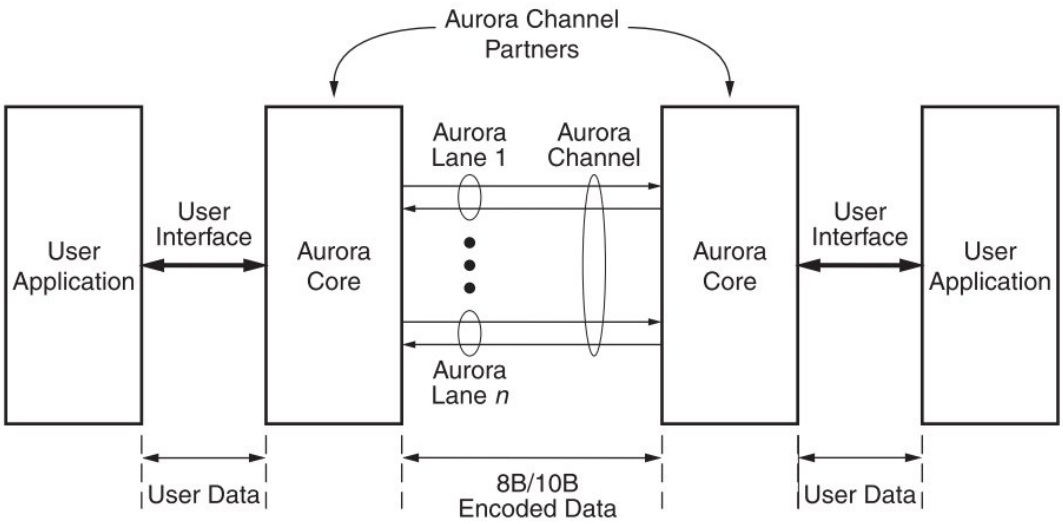

Figure 2.5. Aurora channel overview [29].

The Aurora core applications include chip-to-chip links, board-to-board and backplane links, one-way connections and ASIC connections. The usage of Aurora core for chip-to-chip links reduces the number of traces and layers required on a PCB and hence minimizing the cost. In addition, the Aurora core is scalable both in line data rate and channel width, to allow inexpensive legacy hardware to be used in new, high-performance systems. While for one-way connections, when a back channel is not available, the Aurora core applies a simplex protocol that provides several ways to perform unidirectional channel initialization, reducing costs due to unused full-duplex resources. Recent research has successfully made Aurora core to operate in Altera Aria II FPGA with modification with details works in [42].

2.5 Interlaken High-Speed Serial Protocol

Figure 2.6 shows two dominant high-speed chip-to-chip interface protocols for networking applications: System Packet Interface Level 4 Phase 2 (SPI4.2) [21] and XGMII (10 Gigabit Media Independent Interface) Attachment Unit Interface (XAUI)

[31]. While SPI4.2 offers important features such as channelization, programmable burst sizes, and per-channel backpressure. However, the excessive width of the interface limits its scalability, and the source-synchronous nature of the protocol reduces its effective reach. Conversely, XAUI is a narrow 4-lane interface, offers long reach, and suits various implementations including FR4 on PCB, backplanes, and cable. Yet as a packet-based interface it lacks channelization and flow control, restricting it from several applications.

The weakness for both SPI4.2 [19] and XAUI [31] protocols is only fixed configurations are available, limiting the interface capacity for huge data transfer applications. To resolve the limitations on both SPI4.2 and XAUI, a new chip-to-chip interface protocol, namely Interlaken is defined to allow the design of a narrow, high-speed and channelized packet interface.

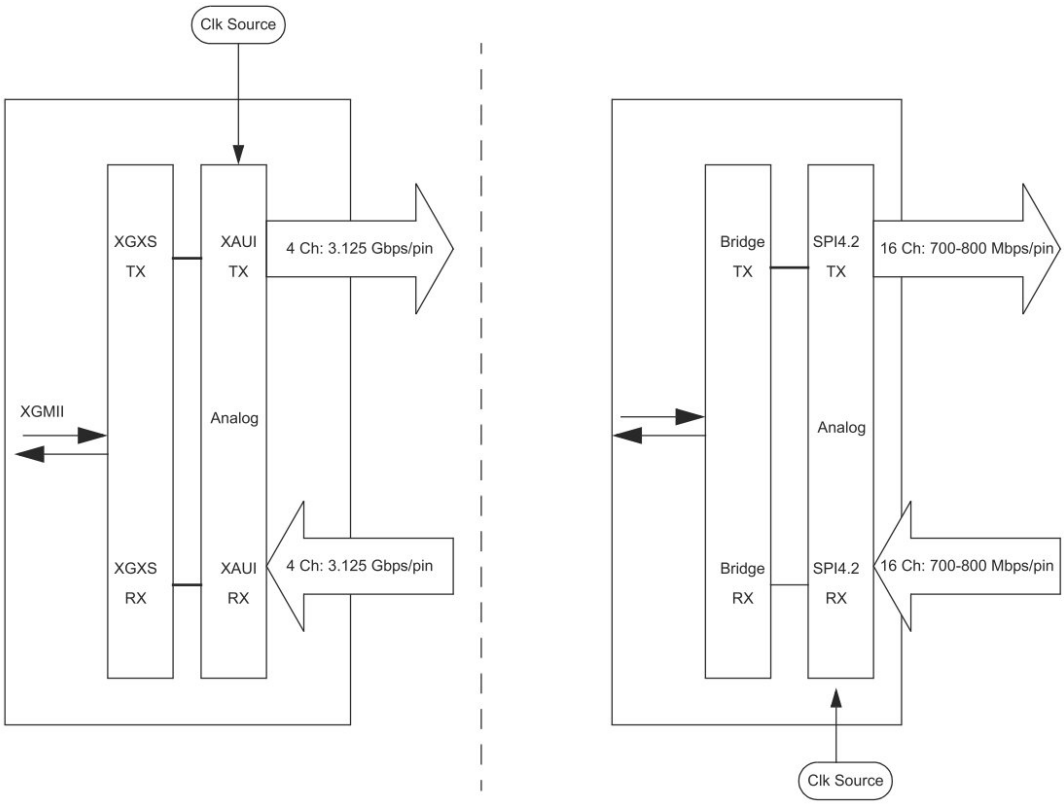

Figure 2.6. The conventional XAUI versus SPI4.2 interfaces [24].

Interlaken are commonly used in applications such as Framer/MAC (Media Access Controller) to Network Processor Unit (NPU) or Layer-2 or Layer-3 (L2/L3) switch interface as shown in Figure 2.7.

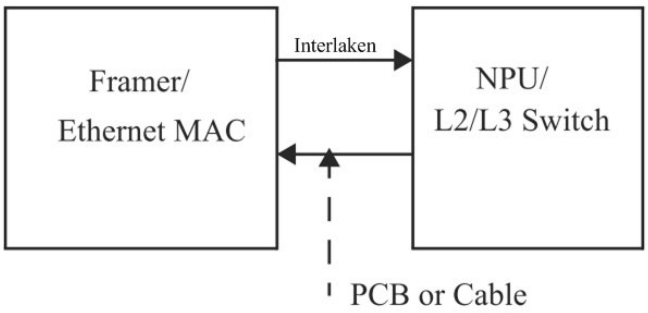

Figure 2.7. Framer/MAC to NPU/L2 or L3 switch [24].

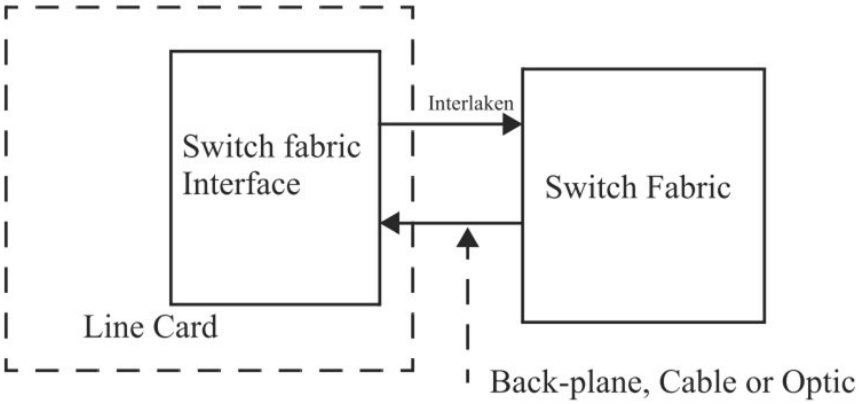

Figure 2.8. Line Card to Switch Fabric interface [24].

Figure 2.8 shows Line Card to Switch Fabric interface. In addition, this protocol also well-suited for multi-gigabit high-speed point-to-point inter-FPGA data transfer.

Two fundamentals structures that define the Interlaken Protocol are Meta Frame and the data transmission format leveraging from the SPI4.2. Interlaken protocol support bursts, error detection and retransmission on error. Data transmission across

the Interlaken interface took place on configurable number of SerDes lanes, from one to inherently no maximum.

Interlaken serial interface uses modified 64B/66B known as 64B/67B encoding with 4.5% overhead and uses the IEEE 802.3ae 10 Gigabit Ethernet specification [31]. This 64B/67B encoding is used to delineate word boundaries, to provide randomness to the Electromagnetic Interference (EMI) generated by the electrical transitions, and to allow for clock recovery besides maintaining DC balance [17]. The modified 64B/66B encoding scheme fixes DC imbalance or more specifically known as unbounded baseline wander resulted from the accumulated excess of 1's or 0's transmitted from an individual SerDes lane.

Figure 2.9 shows Altera Interlaken PHY IP Core [27], it is a high speed serial communication protocol for chip-to-chip packet transfers which supports 1 to 24 lanes running at 6.5536 Gbps or greater in Altera Arria V GZ and Stratix V devices. The key advantages of Interlaken are scalability and its low IO pin count compared to earlier protocols such as SPI 4.2 [19]. Other important features include flow control, low overhead framing, and extensive data integrity and error checking.

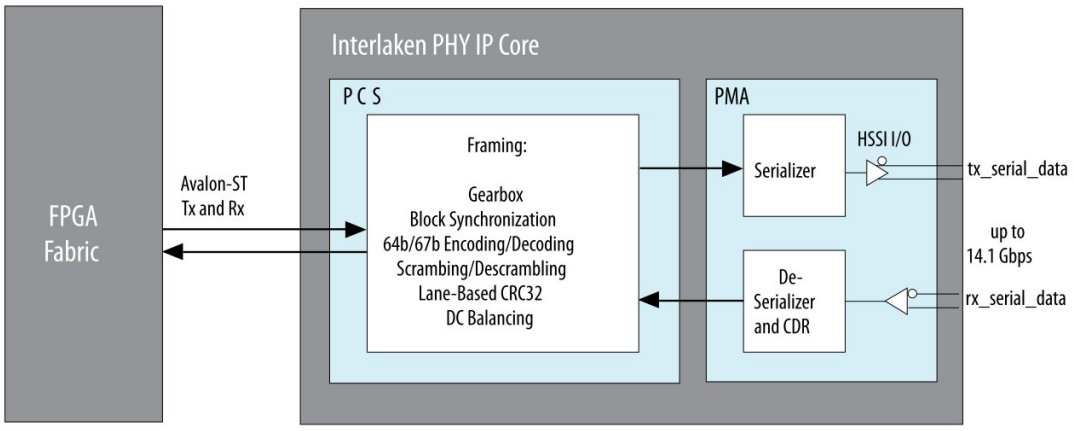

Figure 2.9. Altera Interlaken PHY IP Core [26].

The Interlaken transmits and receives high speed differential serial data using the pseudo current-mode logic (PCML) I/O standard. The protocol accepts packets on 256 logical channels and is expandable to accommodate up to 65,536 logical channels. Packets are split into small bursts and can be interleaved when necessary. Per channel flow control and integrity checking are available in the burst semantics.

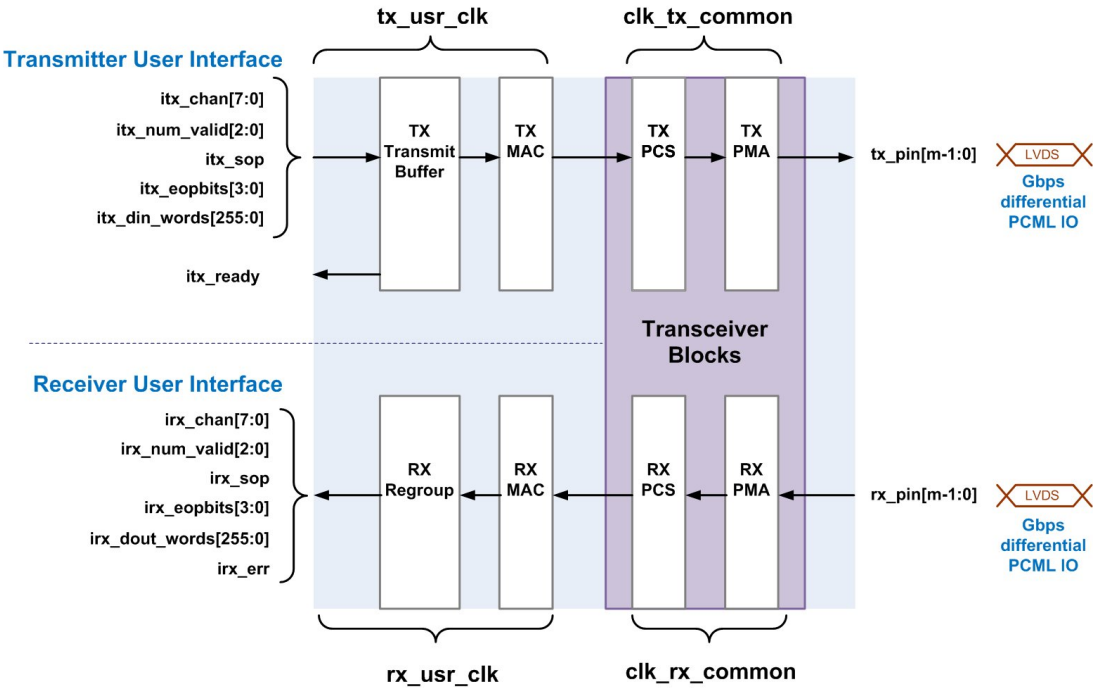

Figure 2.10. Altera 50G Interlaken block diagram [26].

Figure 2.10 shows the Altera 50G Interlaken MegaCore function which consists of two paths: an Interlaken transmit (TX) path and an Interlaken receive (RX) path.

2.5.1 Altera 50G Interlaken IP Core Packet Mode Operation

Figure 2.11 illustrates a packet mode data transfer of 83 bytes on the transmit interface into the Altera 50G Interlaken IP core. To start a data transfer, 'itx_sop' is asserted high when 'itx_din_words' is ready as shown within time t_0 and t_1. At the following rising edge of the clock at time t_1, the IP core detected 'itx_sop' at logic

high, indicating that the value on ‘itx_din_words’ in the current cycle is the start of an incoming data packet. ‘itx_chan’ carries value 8’h2 indicates to the IP core that the data originates from channel number two.

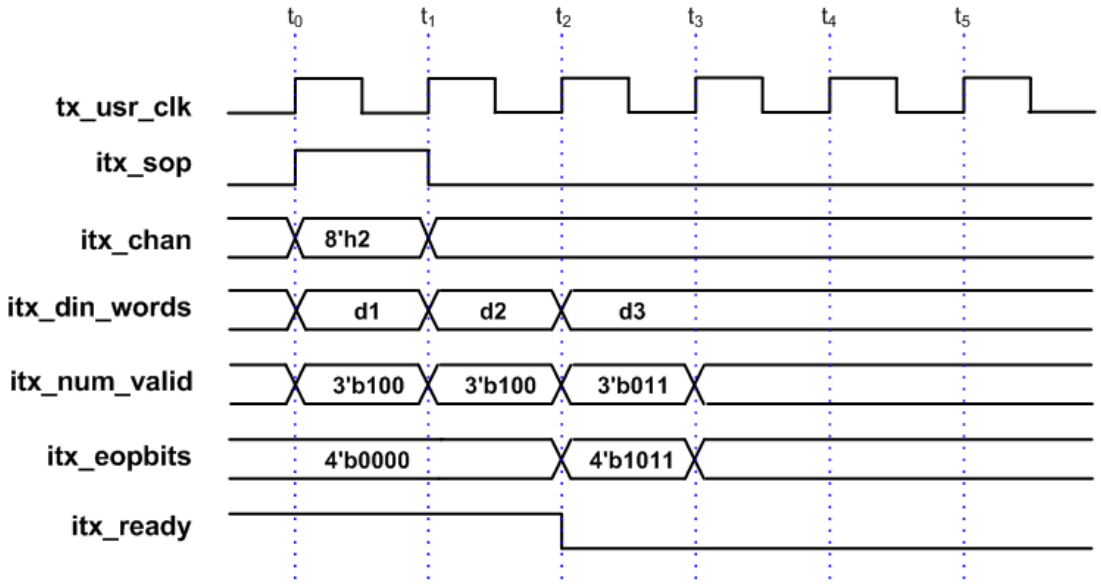

Figure 2.11. Interlaken packet transfer on transmit interface in packet mode

During the start-of-packet (SOP) cycle between time t_0 and t_1 with data value d1 and the cycle that follows the SOP cycle with data value d2 within time t_1 and t_2, ‘itx_num_valid[2:0]’ with values 3'b100 shows these two cycles each carries 32 bytes (4-lane x 8-byte) of data in d1 and d2 respectively. In the following clock cycle with time t_2 and t_3, data d3 is presented with ‘itx_num_valid[2:0]’ with value of 3'b011 indicates the current data symbol contains three 64-bit words (or 3-lane x 8-byte) of valid data. Within time t_2 and t_3, ‘itx_eopbits[3]’ set high indicates the current cycle is an end-of-packet (EOP) cycle and the other three bits ‘itx_eopbits[2:0]’ at value of 3'b011 indicates that only last three bytes of the final valid data word are valid data bytes. In the EOP cycle within time t_2 and t_3, the IP core receives two full words (2 x 8 = 16 bytes) and three bytes of valid data (indicated by ‘itx_eopbits[2:0]’ at value of 3'b011 showing only last three bytes of the final valid data word from d3 are valid

data bytes) for a total of 19 valid bytes. The total packet length received at the receiver user interface is 32-byte of d1 within time t_0 and t_1, added with 32-byte of d2 within time t_0 and t_1, added with 19 valid bytes of d3 within time t_2 and t_3 totaling 83-byte (32 + 32 + 19 bytes).

Back pressure is supported in Interlaken protocol, signal 'itx_ready' set high is used to indicate to the transmitter user interface that the TX Transmit Buffer as shown in Figure 2.10 is almost full and can only receive another four 256-bit word in the following four clock cycles.

Example in Figure 2.12 shows the Altera 50G Interlaken IP Core accepts the first four data symbols: d1, d2, d3 and d4 totaling 128 bytes (32-byte x 4) of a data stream from time t_0 to t_3 and time within t_5 and t_6. The clock cycles from time t_1 to t_3 in which the application transfers the data values d2 and d3 to the IP Core are grace-period cycles following the de-assertion of 'itx_ready' setting this signal to a logic low. In general, the Altera 50G Interlaken IP Core supports up to four cycles of grace period before TX Transmit Buffer is full and stop receiving any incoming data stream.

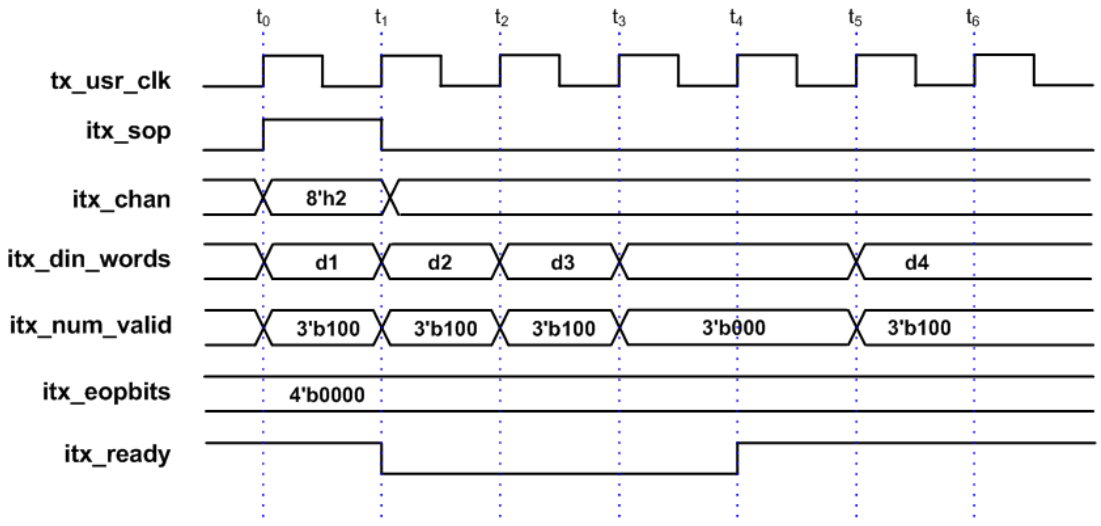

Figure 2.12. Interlaken packet transfer on transmit interface with back pressure.

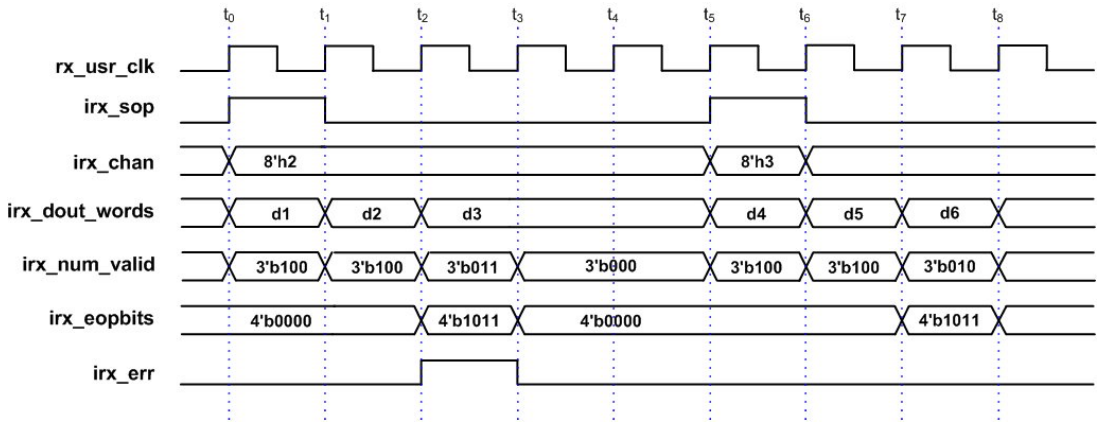

Figure 2.13. Altera 50G Interlaken IP Core receiver side example with irx_err error.

Example in Figure 2.13 illustrates the attempt of a 83-byte data packet transfer received on the receiver user interface to channel-2 detected as packet corruption after receiving an active 'itx_error' signal during a packet transfer with CRC or other errors within time t_2 and t_3. The errored packet transfer is followed by two idle cycles from time t_3 to t_5. Following the errored packet, the IP core retransfers an uncorrupted packet to channel-3 from time t_5 to t_8.

2.6 Time Division Multiplexing Technique

Figure 2.14 below shows input serial-to-parallel converters (ISERDES) and output parallel-to-serial converters (OSERDES) blocks connected to Low-Voltage Differential Signaling (LVDS) differential IO buffer to support data rate more than 1 Gbps [39] for time division multiplexing (TDM) in inter-FPGA communication. A 4-bit wide SERDES is used in this TDM architecture, in case the number of signals passing through one OSERDES to ISERDES is not in a multiple of 4, some OSERDES inputs will be unutilized and left unconnected.