A Flexible FPGA-to-FPGA Communication System

An Wu*, Xi Jin*, Xueliang Du**, ShuaiZhi Guo*

*Department of Physics, University of Science and Technology of China, Hefei, Anhui Province, China

**Department of System verification, Chinese Academy of Science Institute of Automation, Beijing, Beijing Province,China

wuan@mail.ustc.edu.cn, jinxi@ustc.edu.cn, xueliang.du@ia.ac.cn

***Abstract*—In high-performance computing systems, each computing node communicates via a high-speed serial bus to ensure sufficient data transfer bandwidth. However, each computing node of different bus protocols is very difficult to communicate directly, which is not conducive to the extensibility of HPC (High performance computing) clusters. In this paper, we propose UPI, a inter-node communication interface based on FPGA, which can transmit different bus protocols (PCIe protocol and Ethernet protocol) simultaneously. More importantly, many different bus-supported computing nodes can be connected to the same HPC system. We implemented our UPI system on "Gemini" prototype verification board with two Xilinx Virtex-6 FPGAs. The results show that the transmission speed of the UPI can reach 11.04Gpbs (PCIe Gen2 X4) and 4.32Gpbs (Gigabit Ethernet) when DMA payload sizes is greater than 260KB and 80KB, respectively.**

***Keyword*—FPGA-based SoCs, PCIE, Gigabit Ethernet, HPC**

I. INTRODUCTION

With the rapid development of high-performance FPGA-based devices, high-performance computing system performance bottleneck has shifted from the ability of single node to the architecture of the HPC clusters. In the system-level or board-level interconnect system, high-speed serial bus technology with its enormous advantage is rapidly replacing traditional parallel bus technology and becoming the main technology of high-speed FPGA-based SoC design. As the applications of the high-speed serial bus technology gradually expand to all research area, more and more articles focus on the technology, especially the multi-node interconnect technology and the high-speed I/O interface technology.

Effective inter-node communication is receiving significant attention due to its increasingly important applications in high-performance computing, thus inter-node interconnection technology is drawing lots of attractions from more and more researchers. The interconnection methods are based on three main protocols: PCIe, Ethernet, and Serial RapidIO (SRIO). However, they have their own using domain. They can't communicate each other directly, and for the same reason, each computing node can't be compatible on the means of communication.

In order to achieve a good compatibility between these protocols, many problems cannot be neglected. For example, different bus communication requires bridging conversion, the process is complicated and performance loss is hard to avoid. The problem appears in these bus controllers' hardware design. In FPGA-based SoCs, both of Ethernet controller and PCIe controller have an external physical layer (PHY) chip. The controllers' PHY chip of mentioned three types are different, mainly due to its PHY's interface signals and communication speeds are not identical. Ethernet PHY chip mainly realizes 64b/66b encoding (Gigabit Ethernet), and PCIe PHY chip completes the 8b/10b encoding (PCIe 2.0 and below). However, they still have a lot in common, e.g., scrambling and parallel-to-serial conversion all need to be implemented in PHY chip.

The simplest way to solve the compatibility issue is to add all these bus protocols to a computing node, but it requires more PHY chips, differential pairs and hardware resources. A better approach is to communicate with these computing nodes through the same interface without changing the interface types. Generally, the physical layer of PCIe, Ethernet and SRIO cannot be shared. We try to merge the physical layer of three bus protocols, finally, we built a unified physical layer interface which is actually the same functions for their upper layer protocols.

The benefits of a unified physical layer design are as follows:

- A unified physical layer can provide a compatible interconnection bus with three protocols.
- Either PCIe, Ethernet or SRIO devices can be plugged into the same interface for communication.
- The problem of performance loss in bridging process can be settled.

In this paper, we proposes UPI (Unified PHY Interface) system, a flexible interconnection system of inter-node communications based on FPGA devices. Our design shares

Manuscript received February 24, 2016. This work is a follow up of the invited journal of the accepted conference paper for the 18th International Conference on Advanced Communication Technology. This research was supported by the "Strategic Priority Research Program" of the Chinese Academy of Sciences, Grant No.XDA06010402-4.

An Wu is with the Department of Physics, University of Science and Technology of China, Hefei, Anhui Province, China (corresponding author to provide phone: +86-159-5510-2092; e-mail: wuan@ mail.ustc.edu.cn).

Xi Jin is with the Department of Physics, University of Science and Technology of China, China (e-mil:jinxi@ustc.edu.cn).

XueLiang Du is with the Department of System verification, Chinese Academy of Science Institute of Automation, China (e-mil: xueliang.du@ia.ac.cn).

Shuaizhi Guo is with the Department of Physics, University of Science and Technology of China, China (e-mil:dybjxmg@mail.ustc.edu.cn).

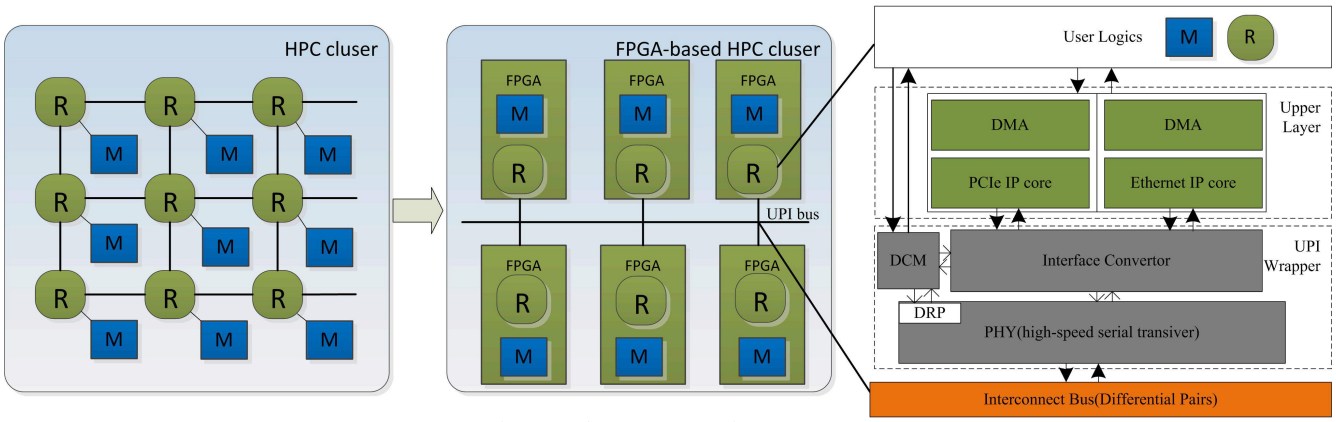


Fig.1. Hardware structure of UPI system

the same portion of PCIe 2.0 and Gigabit Ethernet's physical layers, and merges their different parts. The UPI can transmit PCIe and Ethernet packets with one physical layer chip. The computing node of HPC (High performance computing) clusters using different protocols can connect with each other, meanwhile, the bridging delay and loss in performance can be eliminated, and through the interface design we can implement a more flexible and efficient FPGA-based computing clusters.

The remainder of this paper is organized as follows: background and related work are discussed in section II. We describe the design of UPI in section III. The experiments and results are discussed in Section IV. Finally, we present a conclusion in Section V.

II. Related Work

A. Background on PCIe and Gigabit Ethernet

PCIe is a high-speed serial bus includes transaction layer, data link layer and physical layer. The Transaction layer contains TLP (Transaction Layer Packets) control mechanism. The Data Link layer primary responsibility is to provide a reliable mechanism for exchanging TLPs between the two components on a link [1]. At the physical layer (PHY), the PCIe bus provides a serial high throughput interconnect medium between two devices. PCIe PHY contains two sub-layer: Physical Coding Sub-layer (PCS) and Physical Media Attachment (PMA)[2]. There have been three versions of the PCIe bus. For a single lane, data transfer rate for versions 1.x, 2.x and 3.x are 2,4 and 8Gbps [3].

There are two layers in hardware of Gigabit Ethernet, including physical layer and data link layer (DLL). The main function of DLL is to complete the frame transmission and frame reception. Gigabit Ethernet PHY has three main functions: First, It's provide transferring path of data to data terminal equipment; Second, to be a proper entities for data transmission, not only to ensure that data transfers properly on it, but also to provide sufficient bandwidth and reduce channel congestion; third, complete management of PHY. Upper layer and PHY interconnect with each other via a MII/GMII/RGMII/SGMII interface, through the manage interface in MII, the upper layer can control and monitor PHY [4].

B. Xilinx GTX

Xilinx GTX is a programmable high-speed serial transceiver capable of speeds from 500Mbps to 12.5Gbps.

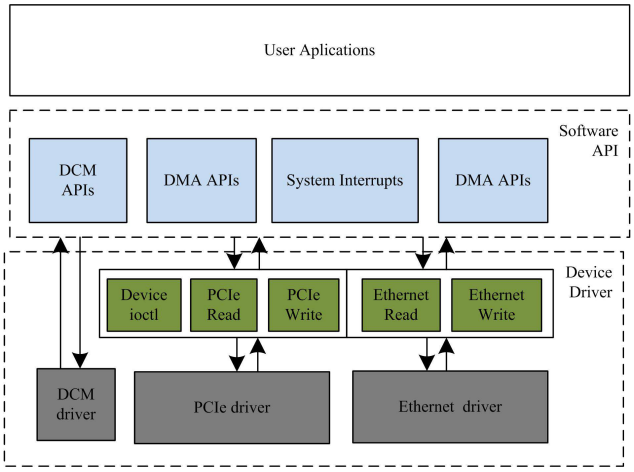


Fig.2. Software structure of UPI system

GTX module in Xilinx FPGA can be realized different serial interconnect protocols, such as SATA, PCIe, EMAC and SRIO. Dynamic Reconfiguration Port (DRP) is an interface module which allows the dynamic change of parameters of the GTX. Through the DRP interface, we can realize the dynamic changes of each interconnect protocol, making it possible for system to adapt to the protocol change. QPLL (Quad PLL) and CPLL (Channel PLL) are two kinds of PLL circuit with different clock rates embedded in GTX module [5].

C. Existing Work

Ethernet is a widely used protocol in HPC computing sysems, which key is its inter-node routing policy. Most system designers make use of the Ethernet protocol to constitute multi-node communication network, the transmission rules follows the Ethernet protocol. However, as more and more high-performance embedded devices appear, researchers are turning their attentions to the direct communication of GPU, FPGA, DSP and other processing units. As a high-speed communication interface, the performance and bandwidth of PCIe meets our design requirements.

Many technologies based on PCIe protocol are proposed, such as InfiniBand and Hypertransport. The InfiniBand Architecture (IBA) is an industry-standard fabric designed to provide high bandwidth/low-latency computing, scalability to ten-thousand nodes and multiple CPU cores per server platform, and efficient utilization of compute processing resources. InfiniBand adapters and switches deliver 56Gb/s bandwidth today and are expected to deliver 100Gb/s by

2016 [6].

For chip-to-chip communications, AMD HyperTransport (HT) are used to connect between CPUs, as well as between CPU and memory. It provides the integral interconnect backbone structure that links all of the core functional units (processor, memory and I/O elements) in a board-level system. As an optimized board-level architecture, HyperTransport provides the lowest possible latency, harmonizes interfaces and supports scalable performance [7].

BlueLink is custom interconnect toolkit for commodity FPGA clusters. Traditional standard protocols such as Ethernet and Interlaken are a boon for FPGA-to-other-system interconnect, they are inefficient and unnecessary for FPGA-to-FPGA interconnect. BlueLink can use all the transceivers available on an FPGA board, over any physical medium. Comparing to 10G Ethernet, 10G BlueLink uses 65% of the logic and registers of 10G Ethernet and uses 15% of the memory of 10G Ethernet. To consider throughput, BlueLink's latency is about equivalent to Ethernet in the fully-loaded case.

III. System Design

In this section we illustrate UPI hardware architecture and software API, as well as explaining the flexibility and compatibility of UPI architecture.

A. Overall System Architecture

Classic HPC cluster structure is shown in Figure 1. The letter M indicates the the computing machine, and letter R indicates the routing policy. We implemented this structure on FPGA-based devices as shown in the middle of Figure 1. In our UPI system, the computing machine and routing policy are integrated into FPGA's user logics. Each FPGA-based computing node is communicate with each other through our high-speed UPI bus. The UPI interface is responsible for connecting each FPGA-based node between user logics and UPI bus.

The UPI interface consists of a hardware component and a software component as shown in the right of Figure 1 and Figure 2. The hardware component mainly consists of two different types of bus controllers, the unified FIFO interface, Interface Convertor and the DRP control module. Interface Convertor is used to convert two different physical layer interface signals into a standard GTX interface signals. The DRP control module is used to dynamically reconfigure the parameters of GTX.

Software component is divided into two parts, including standalone board driver and testing code. Our standalone board driver consists of PCIe and Ethernet device driver, DRP control module driver and DMA driver. Test code consists of PCIe TLP packets reading and writing tests, Ethernet TCP/UDP packets reading and writing tests, equipment switching test. To make our architecture more flexible, the software API provides the simplest and effective way to call the underlying driver functions, as well as shielding the details of low-level operations. By switching the functions of high-speed serial transceiver, user logics can easily achieve the mutual communication between the two protocols.

B. DRP Control Module

DRP control module (DCM) needs to complete link speed selection task and link training control task. Thus, two important parameters for DCM to consider are link width and link data rate. After system boot up, PCIe requires link training process to negotiate the link width and link speed between two sides of PCIe controller. Ethernet also has a similar link training process. The key function of DCM is a link training state machine, which stores the link state of two protocols. When the link is switched on, DCM stores current link state and jump to the next state without need to retrain link. The switching time of the two controller is CPLL reset to CPLL locked. The DCM state machine is shown in Figure 3.

At the beginning of link initialization process, both sides are in Silent mode. Host side starts to seek Device side by sending training sequence in Seek mode. If device side sends its training sequence back, the state will jump to the Discovery mode. In this state, each side sends current link width and link rate for handshaking.

UPI has three configuration modes for link width:

1) *If both sides are four lanes, DCM will jump to ISM_4X_MODE state;*

2) *Theoretically, two lanes have six kinds of interconnect methods. Considering that each channel sends the same packets data, DCM will jump to ISM_2X_MODE state;*

3) *In this mode, DCM will jump to ISM_1X_MODE_LANE3, ISM_1X_MODE_LANE2, ISM_1-X_MODE_LANE1 or ISM_1X_MODE_LANE0 state, the details is as follows.*

The configuration modes can be dynamically changed by ISM_2X_RECOVERY, ISM_1X_RECOVERY or Discovery state. We build a 32-bit width data interface to transmit packet and connect it to Interface Convertor module. Each channel has different negotiation methods corresponding to UPI channel features in mode three.

Fig.3. DRP control module link training state machine

The link speed can be changed by modifying QPLL or CPLL multiplication factor and crossover factor. GTX supports 16, 32, 64 and other data width, higher data width can be achieved by stitching several GTXs. In order to complete the link speed and width adjustment, DRP control

module generates different interface signals in different states. For example, when the bus controller is PCIe Gen2 X4, the CPLL reference clock can be adjusted to 125MHz, CPLL output rate 2.5GHz and data link width can be adjusted to 32 bit data with 4bit k symbol indicator.

C. Interface Convertor

PCIe PHY interface is called PIPE interface, and Ethernet PHY interface is called RGMII In order to achieve a mutually compatible interfaces, we need to make PCIe and Ethernet physical layer interface convert to the same GTX interface. This conversion process is completed in Interface Converter. Table 1 shows the interface signals associated with the GTX, including indicators sending and receiving 32-bit words with 4-bit k symbol and link status control signals.

TABLE I
INTERFACE CONVERTOR RELATED SIGNAL

	Signal	I/O	Description
GTX	mac_phy_rxdata[31:0]	I	receive data
	mac_phy_rxdatak[3:0]	I	K character indication
	mac_phy_txdata[31:0]	O	transmit data
	mac_phy_txdatak[3:0]	O	K character indication
	phy_mac_rxvaild[3:0]	O	receive data is valid
	phy_mac_rxelecidle[3:0]	O	receive electrical idle
	phy_mac_phystate[3:0]	O	PHY functions
	mac_phy_txdetectrx_loopback[3:0]	I	enable receiver detec-tion sequence
	mac_phy_txelecidle[3:0]	I	transmit electrical idle
	mac_phy_txcompliance[3:0]	I	Compliance sequence
	macphy txpolarity[3:0]	I	Invert the received data when asserted
	mac_phy_powerdown[2:0]	I	PHY power down
DRP	bus_state_write[5:0]	O	current controller state write to DRP
	bus_state_read[5:0]	I	current controller state read to DRP
	bus_switch_en	O	bus switch enable
	bus_link_speed[3:0]	O	bus link speed
	bus_link_width[15:0]	I	bus link width

The Interface Convertor mainly includes the following features:

- A set of dual-port RAM, the amount of which is equal to the converted upper interfaces.
- Combining and scattering data to meet UPI interface data width.
- Generating PHY's control signals. If the control signals provided by GTX interface, it can be directly connected to GTX. Signals that GTX does not provide will be generated in Interface Converter according to the control signals' relations or be set to a constant value. For example, the PCIe PIPE interface signal mac_phy_blockaligncontrol is only used in PCIe 3.0, which isn't used in our design, so we give it a constant value in Interface Converter.
- Transmitting link information to the DCM module, completing real-time parameter changes for GTX. The link status signals will change into the DCM signals.

Interface Convertor signals are illustrated in Table 1. Some signals, such as clock and reset, are omitted in this table.

D. Data structures

There are three kinds of data structure in our UPI system: Primary data with DMA descriptors, PCIe and Ethernet packets data with their own protocols, GTX data with Control symbol. The Primary data has been stored into DDR with some DMA descriptors. When PCIe or Ethernet controller obtains a writing command from CPU, a segment of primary data will be sent to the controller. PCIe controller transfers primary data to Transaction Layer Packets(TLP). Data Link Layer Packets(DLLP) are in charge of link maintenance. TLP packets will be assembled with a sequence number and a LCRC code in Data Link Layer. Each TLP can accommodate 4096 Bytes data payload. In Ethernet controller, related packets are TCP, UDP, etc., which have 256 Byte's data payload.

When GTX receives packets data form controllers, each kind of packets is marked with a identifier and two kinds of control symbols (CON symbol and END symbol). We distinguish PCIe and Ethernet packets with binary code "01" and "10", respectively. Different packets have different CON symbol. For example, STP symbol is added to the head of TLP packets corresponding to the identifier. The details of CON symbols are listed in table 2. In our design, the packets with identifier is only generated and digested in UPI layer.

TABLE II
CONTROL SYMBOLS IN DCM

Standard	identifier	Control symbol
TLP	01	K27.7
DLLP	01	K28.2
TCP	10	K28.0
UDP	10	K28.4

E. Hardware Interface and Software API

Our UPI system interface includes the following functions:

- FIFO-based DMA interface.
- DCM interface.
- System interrupt interface.

We use FIFO as the DMA interface for three reasons. First, FIFO can be used between two clock domains. Our PCIe system clock is 125MHz while Ethernet is 100MHz. Two clock domains can be isolated by FIFO interface. Second, FIFO is a standard interface. It can shield the low-level details of UPI hardware, so the transmission process in UPI is transparent for user logic. Third, the width and depth of FIFO can be configured by circumstances of the design and it's convenient for developers to comprehend packet structures. FIFO-based DMA interface is used to transmit data from system's memory to system's I/O controller.

The DCM interface uses a simple DCM bus to read and write DCM registers. We provide DCM interface to make our system easier to operate. For some customized HPC clusters, the DCM interface provides a function to change the bus protocol to meet developers' requirements. Through the DCM Interface, developers even can change the type of communication protocol, which makes the system more flexible of protocol switching.

System interrupt interface is a standard vectored interrupt interface. The main functions provided by the system interrupt interface are software interrupt generating, interrupt

source searching, interrupt number generating and interrupt priority setting. Different interrupt numbers mean that different interrupt exceptions and different CPU response process.

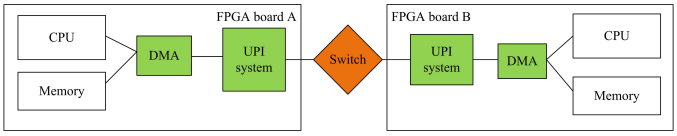


Fig.4. Structure of FPGA-based verification system

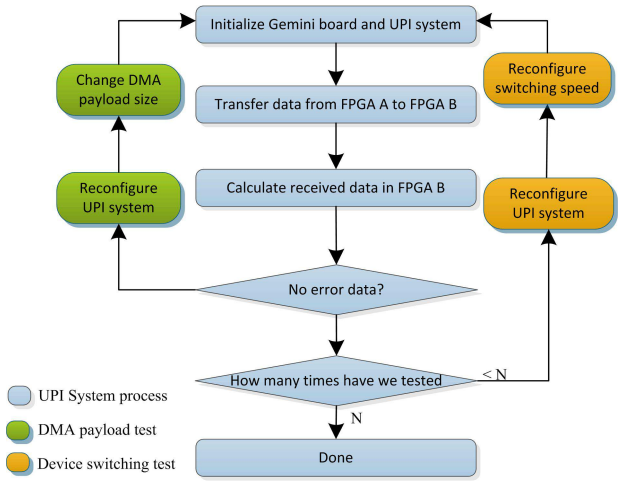


Fig.5. Flow chart of UPI testing program

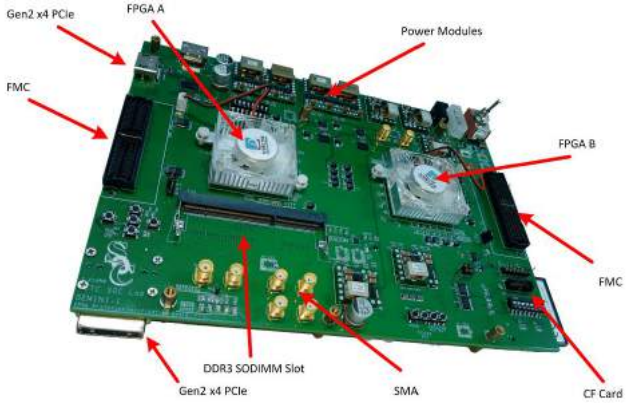
Therefore, a interrupt-supported CPU is required to handle PCIe and Ethernet interrupts. We set these two kinds of interrupts to the same priority, and provide the appropriate interrupt number and interrupt type functions, make CPU to poll interrupt easily after the interrupt exceptions is generated.

Corresponding to the hardware modules, software API is also provides three functions: 1: DMA data read and write. 2: DCM read and write. 3: software interrupts. These APIs are listed as follows, some basic functions such as system reset are omitted in this list:

```
unsigned int drp_write(unsigned char addr, unsigned int data);
unsigned int drp_read(unsigned char addr);

int PCIe_dma(int len, int *ddr_data_1, int *ddr_data_2);
int GMAC_dma(int len, int *ddr_data_1, int *ddr_data_2);

void PCIe_Interrupt(unsigned int ictl_prio,int ictl_number);
void GMAC_Interrupt(unsigned int ictl_prio,int ictl_number);
```



IV. EVALUATION

In this section we implemented UPI system on "Gemini" prototype verification board. Then we evaluate UPI's performance and compare it with other I/O technologies. Finally, we show UPI's flexibility by presenting three practical applications that employ UPI as their communication interface.

A. System Verification Platform

Our system verification platform called "Gemini" is shown in Figure 6. The main hardware components provided by "Gemini" board are two PCIe slots, two Xilinx xc6vlx365tff1156-1 FPGAs, a SODIMM (Small Outline Dual In-line Memory Module) DDR3 and a CF Card's slot. The UPI with a DMA-oriented Synopsys PCIe controller and a Synopsys Ethernet controller is integrated into each FPGA. we also implemented a SoC (system on chip) with a Microblaze CPU. The structure of FPGA-based verification system is shown in Figure 4. The Microblaze CPU communicate with UPI by ARM AXI (Advanced Extensible Interface) bus.

Figure 5 shows the whole process of UPI verification test. Primary data has been stored into DDR with some DMA descriptors. When UPI obtaining a writing command from CPU, a data segment will be sent to UPI system. UPI systemtransfers primary data to physical layer. A typical program for UPI test is as follows:

1) Download bit and elf file to the board.

2) Initialize the CF card, DDR controllers, UPI system.

3) Configure DMA descriptors, move data from CF card into DDR, configure GTX to PCIe gen2.0 X4 mode.

4) Transfer data from FPGA A to FPGA B.

5) Store data in another DDR address in FPGA B.

6) Calculate complete time.

7) Compare the two data segments, calculate error amount if they are different.

8) Switch GTX to Ethernet mode. Repeat steps 3-5.

9) Reconfigure switching speed, and repeat steps 3-6.

B. System test

The UPI simulation process uses standard functional test to simulate UPI system as shwn in Figure 7. The simulation model used in our test is described as follows: 1: UPI_top is the top-level module of UPI system. The gmac, pcie and gpio are some sub-modules in UPI_top module. 2: Our UPI test system have two differential input clock. The host provide 100MHz reference clock and transmit the clock to Gemini board. We use an extra BUF module to receive reference clock before clock is divided in PLL module. After clock is divided in PLL, it connected with GTX input clock, auxiliary clock and AXI interface clock. 3: The test system has only one global reset clock, which comes form Gemini board reset pin. The GTX reset process is faster than controllers, so we build a 10us delay circuit to make the GTX and controllers reset at the same time.

System test uses Vivado 14.3 ISIM simulation and VCS

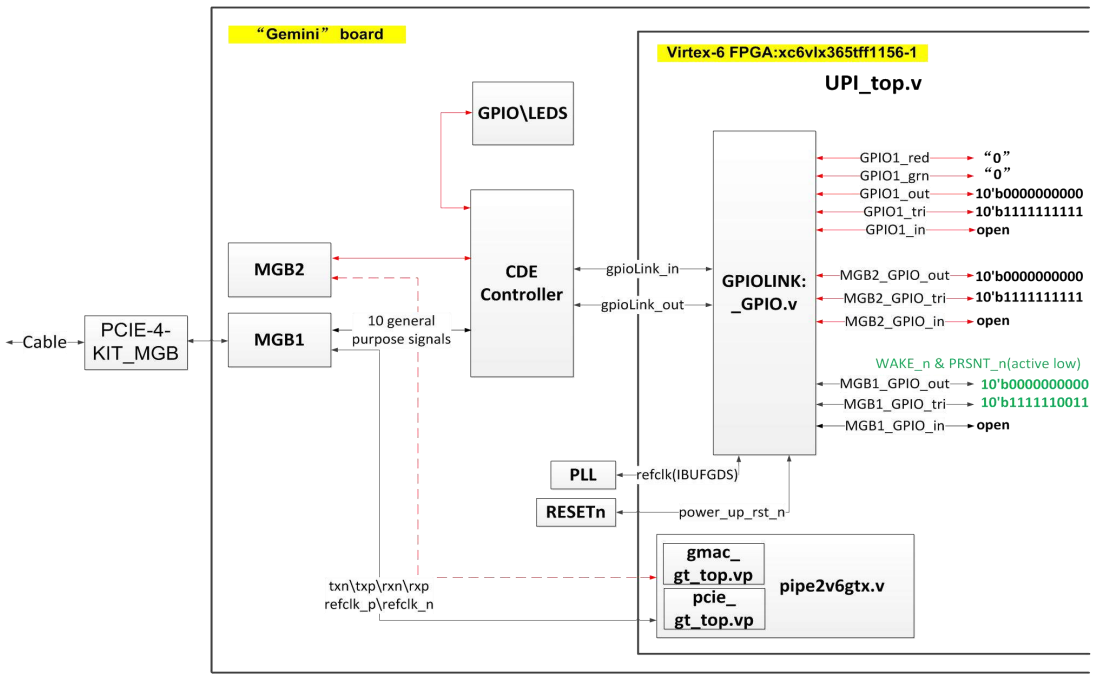


Fig.7. Structure of UPI system test

2014.09-3 simulation tools to verify our design. After functional simulation pass, system test uses Synplify Premier H-2013.03 for RTL implementation and Vivado Chipscope for testing signals observation. The simulation result is shown in Figure 7.

The signal uli_lane_oe_3 available after the other three channels' signals because of the UPI system treat lane four as a special channel. Lane four is used to send control signal and some special communication signals.

Fig.8. UPI system simulation test

Figure 9 shows the FIFO state when UPI system starts to work.

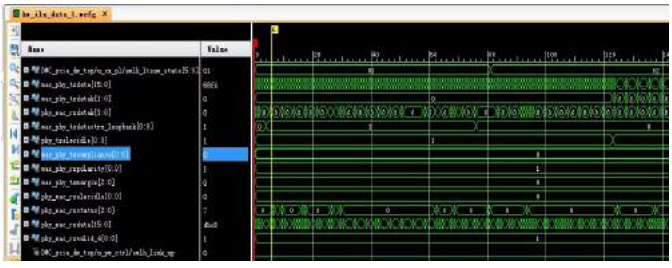


Fig.10. DCM and Interface Convertor states of UPI system

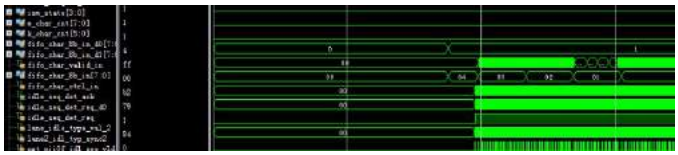


Fig.9. FIFO state of UPI system

Using chipscope to observe the signals come form DCM and Interface convertor, we can clearly see that the current link is in the negotiation state before time of 124us as shown in Figure 10.

After time of 124us, DCM state machine will jump into the next state for exchange link information and link width.

The final placement and layout of the UPI system are shown in Figure 11. Under the control of the PCIe and Ethernet function can be switched by DCM. It uses fewer resources to complete switching function and improves the flexibility of UPI system. This process needs cost some extra area. However, it would increase the system running time significantly for low data throughput or real time applications.

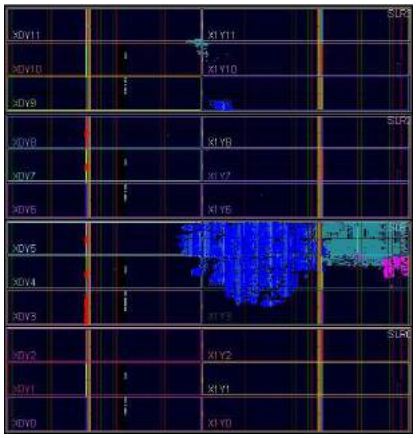


Fig.11. UPI system layout diagram

The design was constrained to the right hand corner of the device where the PCIe and Ethernet blocks resides, as shown in Figure 2. The place with red marked is our UPI wrapper. The maximum clock rate of UPI system is limited only by the design, build tools, and FPGA placing and routing methods.

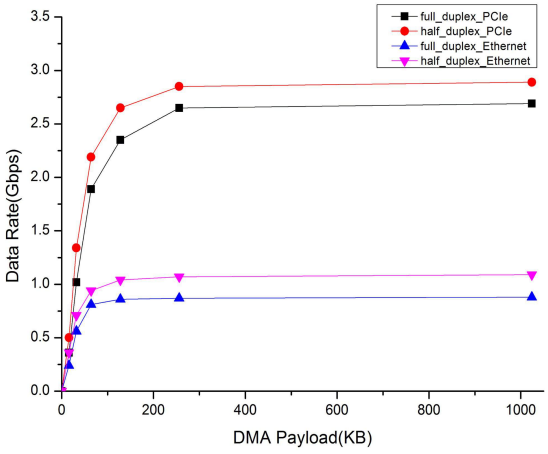


Fig.12. Performance of UPI system

C. *Experimental Results*

Maximum data rates of PCIe and Ethernet are 20Gbps and 4Gbps, respectively. Effective data rate(MB/s) =serial bus clock frequency * 1 Byte(bit/8)* number of ports * encoding format * half-duplex/ full-duplex.

For our design, using X4 lanes, 8/10encoding, full-duplex mode and 2.5GHz(PCIe)/1.25G(GMAC) serial bus clock, effective data rate is 16Gbps/8Gpbs, i.e., 4Gpbs/2Gpbs per lane. System performance can't reach the maximum data rate because of some limits. The highest data rate in test is about 74% of the maximum data rate, calculated from the clock frequency of PLL.

We repeat our test more than 200 times. With the DMA payload sizes keep growing, the performance of UPI also keeps increasing. In DMA payload test, the data rate of both controllers increases as the amount of data increase, as shown in Figure 12. When the DMA payload sizes is larger than 80KB, data rate of Ethernet hold steady at 1.1Gbps while data rate of PCIe still increasing. The data rate of PCIe becomes saturated at 2.8Gbps when payload sizes is larger than 260KB. Ethernet data rate entering saturation more quickly than PCIe because of the maximum packet sizes of Ethernet. Although the data rate of Ethernet is slower than PCIe, the Ethernet physical layer uses 64b/66b encoding, makes Ethernet more effective than PCIe in terms of transmission efficiency.

TABLE III
RESULTS OF DEVICE SWITCHING TEST

	bit error	
Switching time(us)	**full_duplex_PCIe**	**full_duplex_Ethernet**
2.58	100%	100%
1.85	92.6%	93.72%
1.43	46.71%	72.7%
1.04	18.5%	29.35%
0.61	0.037%	0.045%

In Device switching test, we recorded the relationship between bit error and switching time as shown in table 3. UPI can normally work at 2.58us. But as the switching speed accelerated to 1.85us, the bit error increase significantly. When the switching time is less than 500ns, no data entered FPGA B. There are three reasons why the system appears this phenomenon: 1) The switching time is close to the GTX's required reset time, when switching time is less than 500ns, GTX's CPLL is always in the unlocked state, and GTX is unable to complete the transfer task. 2) Some transferring data are stored in FIFO, the data will be flushed after GTX's CPLL reset. In this situation, all data already transmitted are no error. 3) High speed switching causes analog circuits crosstalk especially differential pairs.

When the switching time is more than 2.58us, and the transmission data is greater than 260KB, all transmitted date can be received with no bit errors, except for the write data errors generated by DDR itself.

The resource consumption of UPI system is listed in table 4. We also compare the performance of UPI system with other interconnect I/O technologies as shown in table 4. The results showed that despite the fact that our systems take more resources, UPI system achieves a better flexibility and compatibility. When performance and resources are able to meet the demand of bandwidth between nodes, we made computing nodes of two different protocols compatible.

TABLE IV
RESOURCE CONSUMPTION OVERVIEW

Resource	Our Design in FPGA A	Our Design in FPGA B	UPI system	Resource Available
LUTs	94528(41.5%)	101064(44.4%)	11853(5.2%)	227520
I/O	5(0.4%)	5(0.4%)	N/A	1156
Flip-Flops	65372(14.4%)	84432(18.6%)	8224(1.8%)	455040
BRAM	38(9.1%)	45(10.8%)	6(1.4%)	416

TABLE V
SYSTEM PERFORMANCE OF DIFFERENT INTERCONNECT I/O TECHNOLOGIES

System	Link Rate	Configuration	Link Rate	PCI support	Ethernet support	LUTs
Bluelink[8]	10G, 40G	1x, 4x	10G	No	Yes	2009
Infiniband[9]	40G	LLC QDR 4x	10G	Yes	No	64105
1000 based-X Ethernet MAC	1G	1x	1.25G	Yes	No	11853
PCIe soft IP(stratix IV)	5G	1x Gen2	5G	No	Yes	1805
UPI	11.04G/4.32G	1x, 2x, 3x, 4x	2.76G/1.08G	Yes	Yes	5500

D. Practical Application of UPI

UPI has been used in three practical applications in various institutes. These applications include: MaPU (Mathematics Process Unit) commodity FPGA clusters; HDR (High Dynamic Range) vedio clouding system; MOND (Modified Newtonian Dynamics) hardware accelerator for astronomical data. All these applications require real-time communication with multiple computing nodes. UPI can be used in the application of a good compatibility with the traditional single protocol HPC clusters, as well as adding new UPI-based computing node into computing system. UPI provides a better system compatibility and interface flexibity as shown in table 5.

V. Conclusions and Future Work

A flexible and compatible interconnect interface has been proposed for FPGA-based multi-node communication. We completed a multi-node communication interface with the benefits of high flexibility and good compatibility. We implemented our design on "Gemini" prototype verification board with two Xilinx Virtex-6 FPGAs. The experimental results show that both two bus protocols can be received and transmitted without error when DMA payload size is greater than 260KB (PCIe) and 80KB (Ethernet) and switching time is greater than 2.58us. Through the interface we can easily connect two different HPC clusters. The performance loss caused by traditional bridge equipment is eliminated.

Acknowledgment

This research is supported by the "Strategic Priority Research Program" of the Chinese Academy of Sciences, Grant No.XDA06010402-4.

References

[1] Budruk, Ravi, Don Anderson, and Tom Shanley., "PCI express system architecture," *Addison-Wesley Professional*, 2004.

[2] Intel Corporation's, *PHY Interface for the PCI Express(TM) Architecture*, Specification Version 0.5. pp. 1-15, Aug. 16, 2002.

[3] Gong, Jian, et al., "An efficient and flexible host-fpga pcie communication library," *Field Programmable Logic and Applications (FPL)*, 2014 24th International Conference on. IEEE, 2014.

[4] Koch D, Beckhoff C.,"Hierarchical reconfiguration of FPGAs," *Field Programmable Logic and Applications (FPL)*, 2014 24th International Conference on. IEEE, 2014: 1-8.

[5] Xilinx Inc., *7 Series FPGAs GTX/GTH Transceivers User Guide*, April 22,2013.

[6] Islam, Nusrat S., et al.,"High performance RDMA-based design of HDFS over InfiniBand.," *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE Computer Society Press, 2012.

[7] Shainer, Gilad, et al.,"Maximizing application performance in a multi-core, NUMA-aware compute cluster by multi-level tuning.," *Supercom-puting. Springer Berlin Heidelberg*, 2013.

[8] Theodore Markettos, "A Interconnect for commodity FPGA clusters: standardized or customized?," *Field Programmable Logic and Ap-plications (FPL)*, 2014 24th International Conference on. IEEE, 2014.

[9] TPolybus Systems Corporation, "InfiniBand cores. ," http://www.polybus.com/iblink layer website/ibcores brochure alt.pdf

An Wu received his B.S. degree in 2011 from School of Anhui University, Anhui province, China, and he is currently a Ph.D. student in Department of Physics in University of Science and Technology of China, Anhui, China, under the supervision of Prof. Xi Jin. His current research work is mainly on SoC design technology, VLSI design and FPGA-based Hardware Accelerator design.

Xi Jin received the B.S. degree from University of Science and Technology of China, Anhui, China, and he is currently an associate professor in Department of Physics in University of Science and Technology of China, Anhui, China. His research interests include SOC design technology, VLSI design, computer-aided design methodologies for SoC system integration and FPGA-based Hardware structure design.

Xueliang Du received the Ph.D degree from University of Science and Technology of China, Anhui, China, and he is currently an associate professor in Institute of Automation Chinese Academy of Sciences, Beijing, China. His research interests include High-Performance SoC Design, DSP Design and FPGA-based prototyping.

Shuaizhi Guo received his B.S. degree from University of Science and Technology of China, and he is currently a M.S. student in Department of Physics in University of Science and Technology of China, Anhui, China, under the supervision of Prof. Xi Jin. His current research work is mainly on FPGA-based Hardware Accelerator design.