SLink: An Efficient Point-to-Point Protocol for Chip-to-Chip Interconnection

Peinan Li[1, 2, a], Ping Xue[1, b], Chen Lin[2, c], Yinping Jiang[2, c], And Hongyu Meng[2, c]

[1] Department of Automation, Harbin University of Science & Technology, Harbin, 150080, China

[2] Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China

[a]email: lipeinan2014@ia.ac.cn, [b]email: xueping@hrbust.edu.cn,

[c]email: {lin.chen1, yinping.jiang, menghongyu2014}@ia.ac.cn

Keywords: point-to-point protocol; low latency; high bandwidth; FPGA prototype

Abstract. Multi-chip architectures have been a focal point not only in high-performance computing but also in the new generation of embedded systems. For point-to-point topology, high latency and low bandwidth are always found in existing chip-to-chip protocols. This prevents the efficiency of information communication in mutli-chip system. In this paper, we propose a novel low-latency, low-cost and high-scalability protocol, named SLink. We provide an implementation, including the programming model, the simulation and FPGA prototype verification environment. Compared to PCIe 2.0, the results of experiments demonstrate that the latency decreases by 61.0 percent; the maximum bandwidth increases by 55.6 percent and the area decreases by 97.5 percent.

Introduction

NVIDIA, together with IBM, propose a novel supercomputer architecture which adopts NVLink connecting CPU and Multi-GPU effectively [1]. AMD, cooperating with ARM and other companies, founded the HSA Foundation, aiming at enabling GPU as a first class co-processor to the CPU via architecture definition, and expanding the parallel programming models [2, 3]. Many institutes are also researching new Multi-core architecture [4, 5]. As the bridge of linking all independent SoC, chip-to-chip interconnect plays a critical role in speeding up the progress of optimizing HPC architecture.

At present, there are two major topology of chip-to-chip interconnect: tree based on routing protocol, such as PCIe, RapidIO; and fully mesh based on point-to-point protocol, such as HyperLink and HyperTransport. In a tree topology, the data communication is limited by the bisection bandwidth [1]. That is, the communication data between left and right part must be transported through the root node, resulting in bottlenecks. Though most routing protocols are also applicable for point-to-point interconnection, complicated architectures bring low efficiency and high latency. Therefore, point-to-point protocol will dominate in chip-to-chip interconnection.

There are two interconnection technology, including parallel technology, such as HyperTransport, Parallel RapidIO; and serial technology, such as PCIe, Serial RapidIO (SRIO), and HyperLink. Along with the rapidly growing clock frequency, clock skew limits the development of parallel technology. Thus, serial technology will be prevalent in the future.

PCIe, developed by Intel, supports for many functions, and keeps backwards compatibility, so its actual transmission speed is up to 64% of the theoretical transmission speed [6]. Thus, PCIe is suitable for non-real time application. RapidIO, developed by Motorola and Mercury, has efficient transmission method for small data sizes [7], while high latency for big data sizes in real time applications. SRIO, while transmitting high-definition pictures, provides a bandwidth of 2.32Gbps per lane, 74 percent of theoretical maximum transmission speed [8]. HyperLink, developed by TI [9], is used only in their own products. Though HyperTransport, developed by AMD, has low pin count, it is still a parallel transmission protocol [10]. When the number of core or the clock frequency increases, there will exist great difficulties in accomplishing the design.

To meet the requirements of real-time and high bandwidth of chip-to-chip interconnect, we proposed a high-speed serial point-to-point interconnect protocol, named SLink, meaning Speedy Link. SLink is a layered protocol, consisting of the Transaction Layer, the Data Link Layer, and the Physical Layer based on SerDes. Furthermore, SLink supports concise packets, convenient programming method, and configurable CRC detection and retransmission mechanism.

This paper will depict this protocol and an implementation method in detail. To evaluate the performance of SLink, we build a verification environment according to actual chip-to-chip scenarios. Compared to PCIe 2.0, the results of experiments illustrate that the latency decreases by 61.0%; the available bandwidth increases by 55.6%, and the area required for the controller decreases by 98%.

The rest of this paper is organized as follows. Section I introduces the details of the protocol. Section II presents an implementation method of SLink. Section III reports some experiments compared to PCIe 2.0. Section IV concludes this paper.

Specification of SLink

SLink is a high efficient and scalable serial interconnect designed for future computing and communication in chip-to-chip scenarios. Chips with SLink communicate via a logical connection called physical link. The link is a based on LVDS. At the physical level, a link is composed of one or more lanes. Each lane consists of two differential signaling pairs, with one pair for receiving (RX) data and the other for transmitting (TX). In this paper, lane counts are written with an "x" prefix. Compared single transmission, this method can reduce cross-talk, improve signal integrity and timing predictability [11], and provide high transmission efficiency.

SLink is a layered protocol, consisting of the Transaction Layer, the Data Link Layer, and the Physical Layer, as illustrated in Fig. 1. The bus interfaces include DMA control interface, which is used for reading and writing data from memory, and the configuration interface for configuring transmission mode or observing transmission status by writing or reading specific function registers.

The Transaction Layer.

The Transaction Layer provides four significant services: configuring transmission mode, reading and writing data from memory, generating request packets and analyzing packets received from the Data Link Layer.

The Write request and Data packet is composed of a header control word, operating data and optional CRC, as is illustrated in Fig. 2. The header control word contains the destination address and the operation mode. The read request packet consists of a header control word only, with different request type compared to write request packet.

The Data Link Layer.

The Data Link Layer is responsible for the integrity of the operating data, including assembling and analyzing K-codes, dispatching packets to different lanes, assembling packets from enabled lane according to transmission mode, and performing CRC detection and retransmission mechanism. K-codes in SLink, described in Table 1, are compatible with a subset of PCIe [12]. This Layer is also responsible for generating CRC response packet, composed of a header control word only as well, with different packet type compared to request packet.

The Physical Layer.

The Physical Layer, between the Data Link Layer and physical link, is designed based on SerDes. The Physical Layer in the sender receives parallel data from the upper layer, utilizes 8b10b encoding, and sends them to SerDes module to serialize the valid data. The receiver receives data from physical link, deserializes the signals, utilizes 8b10b decoding, and then sends data the Data Link Layer.

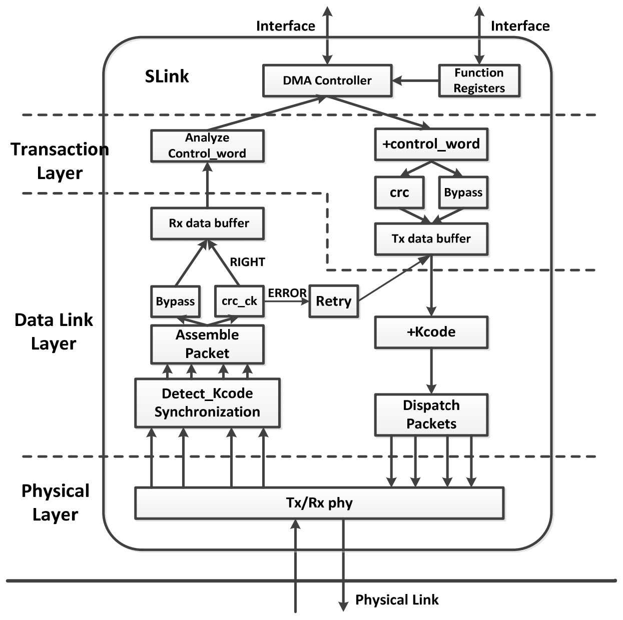

Fig. 1 SLink Layered Architecture

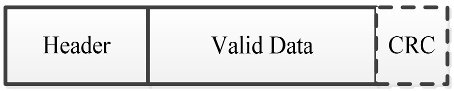

Fig. 2 Write request and Data packet

Table 1 Specification of K-codes

K-code	Value	Abbreviation	Description
K28.3	0x7C	IDL	Idle status
K28.5	0xBC	COM	Initial value
K23.7	0xF7	PAD	Padding chars
K27.7	0xFB	STP	Start of packets
K29.7	0xFD	END	End of packets

63:32						
Remote Address						
31:24	23:8	7:5	4	3:2	1	0
Reserved	**Data Length**	**Lanes Number**	**CRC Enable**	**Packet Type**	**Reserved**	**Request Type**

Fig. 3 Format of packet Header

Implementation

In this paper, we proposed an implementation solution of SLink. Refer to Fig. 2. AXI 3.0 is used as the external interface bus, which supports burst transaction to improve transaction efficiency. Both external professor memory with AXI interface and the FIFOs employ 64-bit data width, which satisfies parallel 16-bit to four lanes. The Physical Layer adopts PIPE and SerDes module of PCIe 2.0, supporting x4 with each lane running at 2.5 Gbps or 5Gbps.

Specific function registers.

This implementation solution contains four 32-bit configuration registers and one 32-bit read-only status register. The status register is responsible to signify start, end and other progress statuses of each operation. And the other registers are responsible for the configuration of operation mode.

As is illustrated in Fig. 3, the header control word consists of the information of the operation mode. Each field is defined as follows.

Remote Address: destination memory address of write operation or source memory address of read operation.

Data Length: The number of data in this packet, up to 512KB per operation.

Lane Number: The number of lanes required in this operation. This field supports x1, x2, and x4 mode. S0 (SLink in SoC0) must initialize this parameter, while S1 (SLink in SoC1) will synchronize with S0 to ensure a stable transmission.

Packet Type: Request packet or CRC response packet.

Operation Type: Write operation or read operation.

Reserved: For later functions expansion.

CRC Selection.

According to the method in reference [13], we adopt CRC-16 to improve reliability of transmission. CRC-16 can detect the following errors:

1. All single-bit and 2 single-bit errors. All odd number of bit errors.
2. All bit errors of "burst length" 16 or less.

4. 99.9969 percent of all errors of burst length 17.

5. 99.9985 percent of all errors of burst length greater than 17.

If a bit is wrong, it is likely that the next bit is in error too. Burst-error begins and ends with an error; and there may be some mistakes in the intervening bits.

Based on CRC check, the retransmission mechanism of SLink is enabled by specific function registers, and accomplished by hardware automatically. Different from software implementation, this method has the advantage of higher encoding/decoding speed [14] and convenient programming interface for application developers.

FIFO Selection.

Within this implementation, the size of the transmit FIFO and the receive FIFO are same to the maximum packet size SLink supports, which is 512KB. This selection has many advantages:

1. Solve the issues caused by asynchronous clocks.
2. Retransmission data can be reached from FIFO directly, and no software judgement is needed.
3. No more retransmission while encountering bus contention.
4. No redundant FIFO flag, such as transmit FIFO full, transmit FIFO empty, receive FIFO empty, etc.

Programming model.

Strictly according to the field definition of every register, this paper shows the programing model in Fig. 4, which contains read and write operations with optional CRC check and retransmission mechanism.

Notes that the codes between 1 and 10 lines explain all variables in codes. Codes between 11 and 13 lines judge whether this operation will change transmission speed, which determines reset for physical link. Codes between 14 and 19 lines configure registers to initiate operations. Codes between 20 and 22 lines query the state of transmission, and then finish the operation successfully.

```
Algorithm 1 SLink Configuration
input: Variables: $pointer$, $is\_new\_configuration$, $option$, $N$, $xmode$, $crc\_en$, $gtps$, $local\_addr$,
    $remote\_addr$
output: : $op\_finished$
 1: %$pointer$ : the base address of SLink in system memory map.%
 2: %$is\_new\_configuration$ : whether this operation is new for bandwidth and number of each
    lane%
 3: %$option$ : request type, 0 for write, and 1 for read%
 4: %$N$ : data length of this transmission%
 5: %$xmode$ : lane number, 0 for x1, 1 for x2， 2 for x4%
 6: %$crc\_en$ : CRC enable flag， 1 is active%
 7: %$gtps$ : bandwidth per lane, 0 for 2.5Gbps, 1 for 5Gbps.%
 8: %$local\_addr$ : local address.%
 9: %$remote\_addr$ : remote address.%
10: %$op\_finished$ : end of transmission.%

11: if $is\_new\_configuration$ then
12:     $pointer[0/4] = 0$
13: end if

14: $SOFT\_RESETN = 1$
15: $START = 1 << 4$
16: $pointer[4/4] = (option << 0) + (xmode << 5) + (crc\_en << 4) + (N << 8)$
17: $pointer[8/4] = remote\_addr$
18: $pointer[12/4] = local\_addr$
19: $pointer[0/4] = gtps + SOFT\_RESETN + START$

20: while $pointer[16/4]\&1$ do
21:     $op\_finished = 0$
22: end while
23: $op\_finished = 1$
```

Fig. 4 Programing model

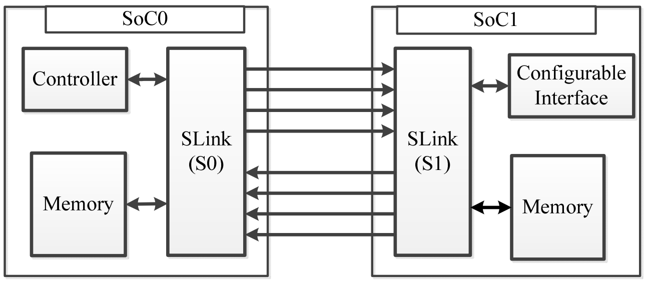

Fig. 5 Composition of Verification and Evaluation Environment

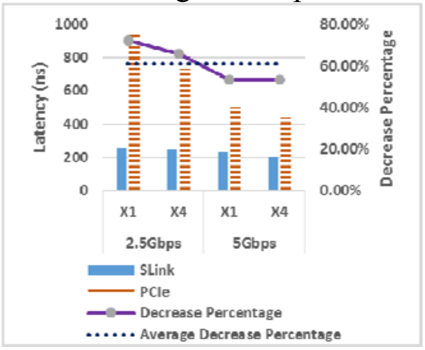

Fig. 6 Comparison of Latency between PCIe and SLink (without CRC check)

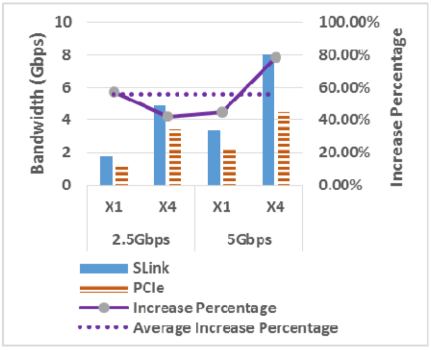

Fig. 7 Comparison of Bandwidth between PCIe and SLink (without CRC check)

Evaluation

Environment.

To verify the rightness and evaluate the performance of the implementation, we built a test environment similar to actual chip-to-chip interconnect. As illustrated in Fig. 5, SoC0 and SoC1 have their own addressing space and memory.

Taking the advantage of easy debugging, fast iteration and convenient timing utility, software simulation is an excellent way to verify and evaluate the design. In this experiment, we used VCS 2014.03.

There are totally 24 kinds of transaction modes, including different configuration of request type, CRC check option, transmission per lane and lane number. Being able to configure both S0 and S1 at the same time, we verified all kinds of modes, recorded signal variation and evaluated performance.

Latency.

The latency in transmission progress is the time between S0 reads the first number from source address and S1 writes it to destination. The smaller the latency is, the better real-time performance will be.

In the configuration of x1 and x4 mode, 2.5Gbps and 5Gbps per lane, with different data lengths, we recorded the latency without CRC check, and with CRC check.

For comparative analysis, we also built an environment to evaluate the performance of PCIe 2.0.When both SLink and PCIe 2.0 transmit 512 Bytes, the latency results measured are shown in Fig. 6, SLink reduces the latency by 61% on average compared to PCIe 2.0, and there is a 72% reduction in x1 with the lane running at 2.5Gbps.

Bandwidth.

The total transmission time ("T") in simulation is the time between S0 reads the first number from source address and S1 writes the "N" numbers to the destination address.

When both SLink and PCIe 2.0 transmit 512 Bytes, the bandwidth results measured are shown in Fig. 7. The results show that, SLink increases the bandwidth by 55.6% on average, and there is a 78.5% increment in x4 with the lane running at 5Gbps.

FPGA Verification.

To verify and evaluate SLink in real hardware, we adopt two FPGAs for downloading SoC0 and SoC1 individually. This experiment environment is designed based on HAPS-70 prototyping verification platform with the architecture shown in Fig. 5. The controller of SoC0 is MicroBlaze. The configurable interface of SoC1 is a configurable state machine, receiving control messages from SoC1 through GPIO pins. These control messages include:

Transmission mode: Write specific function registers, and feedback a success signal after configuration.

Initialize memory: Assure the original data in memory is different from operating data.

Read operating data: To verify the write operation, it needs to send one operating data to SoC0 after write operation.

Limited by the tools, the clock frequency of the Transaction Layer can only reach 100MHz, which satisfy full speed transmission while physical link running at 5Gbps. When SLink is configured with higher physical link transmission speed, the transmission will be blocked, which will be optimized in latter work.

The start flag and the end flag of the transmission in each SoC are recorded by counters, running at 200MHz in order to catch the pulse in 100MHz more precisely. The latency is the difference value of the two start counters. The total transmission time is the difference value of the start counter in S0 and the end counter in S1.

The computing results are shown in Fig. 8.

Because the calculation methods of FPGA verification and software simulation are different, there is slight deviation between the experiment results. However, the conclusion is same:

a) When the transmission is configured without CRC check, the latency basically remains unchanged in the same mode, while it varies inversely when the speed of the physical link (transmission speed per lane multiplied by lane number) is different.

b) When the transmission is configured without CRC check, the bandwidth is proportional to the transmission speed of physical link.

c) When the physical link transmission speed is configured with one lane running at 5Gbps, the bandwidth of SLink reach 3.9Gbps, 78 percent of theoretical maximum bandwidth. Without the consideration of 8b10b encoding, the efficiency is almost 97.5 percent.

Synthesis result comparison.

Firstly, DC 2013.12 is used to compare the reached clock frequency and area between SLink and PCIe 2.0. In worst case, with TSMC 40ns LP-CMOS being the target library, clock uncertainty equaling 0.4ns, timing being the optimization targets, the results are shown in Table 2. The maximum clock frequency of SLink controller can reach 600MHz, which is almost 2.4 times of PCIe 2.0. The area of SLink controller is 1431um2, almost 1/50 of PCIe 2.0.

Secondly, with Synplify Premier 2014.03 synthesizing components of Synopsys, Vivado 2014.4 synthesizing components relative to Xilinx (e.g. MicroBlaze), the synthesizing strategy being Vivado Synthesis Defaults, the implementing strategy being Performance Explore, the results of resource comparison are shown in Table 3. The LUT resource required by SLink controller is almost 1/52 of PCIe 2.0, basically equal to the results of DC.

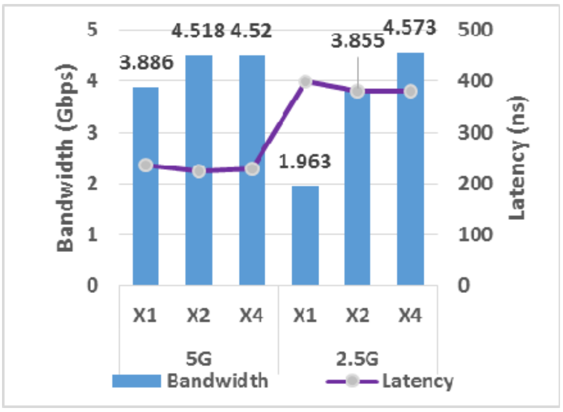

Fig. 7 Verification results of FPGA without CRC check

Table 2 Comparison of Synthesis results between SLink and PCIe 2.0

Synthesis Object	SLink Controller	PCIe Controller
Clock Frequency	600MHz	250MHz
Area	16431um2	599160um^2

Table 3 Comparison of FPGA resources between SLink and PCIe 2.0

Resource	SLink	PCIe	Percentage
Slice_LUT	1624	84998	1/52
Registers	1277	45965	1/36
Slice	618	25727	2/83
LUTasLogic	1624	83876	1/52
F-F Pairs	1868	89228	1/48

Conclusion

A novel point-to-point protocol has been proposed for high performance chip-to-chip communication. We proposed an implementation method, and evaluated the performance through software simulation and FPGA prototyping. The results show that in point-to-point interconnection, SLink is better than PCIe, both in efficiency and area.

In the current implementation, when CRC check and retransmission mechanism is enabled, SLink must write data to memory after the whole packet check, resulting in low bandwidth. To improve the transmission efficiency, the implementation of CRC check mechanism can be optimized. In addition, more concise encoding is an alternative. To utilize processor more efficiently, the interrupts of the transmission status should be supported. These improvements will be accomplished in latter work.

Acknowledgement

In this paper, the research was sponsored by the “Strategic Priority Research Program” of the Chinese Academy of Sciences, Grant No. XDA06011000.

References

[1] NVIDIA NVLink High-Speed Interconnect Application Performance Brief, http://www.nvidia.com/object/nvlink.html, 2014, 09.

[2] Rogers P, Fellow A C. Heterogeneous system architecture overview[C]//Hot Chips. 2013, 25.

[3] HSA Platform System Architecture Specification 1.1, http://www.hsafoundation.com/standards/, 2016,01,21.

[4] Power J, Hestness J, Orr M S, et al. gem5-gpu: A heterogeneous cpu-gpu simulator[J]. IEEE Computer Architecture Letters, 2015, 14(1): 34-36.

[5] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]//Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014: 675-678.

[6] Marcus G, Gao W, Kugel A, et al. The MPRACE framework: An open source stack for communication with custom FPGA-based accelerators[C]//Programmable Logic (SPL), 2011 VII Southern Conference on. IEEE, 2011: 155-160.

[7] LIANG Xiao- hu, WANG Le, ZHANG Ya- di. Analysis and Comparison of High Serial Bus Rapid IO and PCI Express Protocol. Aeronautical Computing Technique, 2010,03: 62-75 (in Chinese).

[8] Zhang F, Wu Q, Ren G. A high-speed serial transport platform based on SRIO for high-resolution image[C]//Image and Signal Processing (CISP), 2010 3rd International Congress on. IEEE, 2010, 5: 2441-2444.

[9] KeyStone Architecture HyperLink User Guide, http://www.ti.com/lit/ug/sprugw8c/sprugw8c.pdf, 2013,7.

[10] HyperTranport^{TM} I/O Link Specification, http://media.wix.com/ugd/071cb6_53b2dc066f2d4408b5c9368dc447e2f5.pdf, 2010,5,6.

[11] Massoud Y, Kawa J, MacMillen D, et al. Modeling and analysis of differential signaling for minimizing inductive cross-talk[C]//Proceedings of the 38th annual Design Automation Conference. ACM, 2001: 804-809.

[12] PCI Express Base Specification Revision 3.1a, https://pcisig.com/specifications/pciexpress/, 2015,12,7.

[13] Koopman P, Chakravarty T. Cyclic redundancy code (CRC) polynomial selection for embedded networks[C]//Dependable Systems and Networks, 2004 International Conference on. IEEE, 2004: 145-154.

[14] MING-DER SHIEH M H W A S, Chen C, Lo H F. A systematic approach for parallel CRC computations[J]. Journal of information science and engineering, 2001, 17: 445-461.