文章编号:1001-2486(2011)06-0055-06

板级高速传输总线链路层关键技术研究与实现*

周宏伟[1],陈 超[1],张丽霞[2],张 英[1],李永进[1]
(1. 国防科技大学 计算机学院,湖南 长沙 410073; 2. 湖南师范大学 数学与计算机学院,湖南 长沙 410081)

摘 要:随着高性能服务器和超大规模计算机的发展,系统设计者对板上高速互连总线的要求越来越高,如何使芯片间的数据传输延迟更小,提高计算通信比是需要解决的重要问题。论文研究了近年来发展迅速的超传输总线和PCI Express总线的链路层的特点,在此基础上提出了一种64位高速总线链路层体系结构,并对其关键技术进行了研究,设计实现了一种能够每时钟周期对16位数据进行加解扰的加解扰器,以及能够纠正链路间最大5个时钟周期延迟偏斜的线间传输延迟偏斜纠正器,功能验证结果表明所提出的设计功能正确。

关键词:板级;传输总线;链路层;加解扰;延迟偏斜纠正
中图分类号:TP302.1 **文献标识码**:A

Research and Design of Link Layer in Board-level High-speed Transportation Bus

ZHOU Hong-wei, *CHEN Chao*, *ZHANG Li-xia*, *ZHANG Ying*, *LI Yong-Jin*
(1. College of Computer, National Univ. of Defense Technology, Changsha 410073, China;
2. College of Mathematics and Computer, Hunan Normal Univ, Changsha 410081, China)

Abstract: With the development of the high performance servers and very large scale super computers, the requirement for board-level high-speed data transportation bus is higher than before. How to reduce the transfer delay between chips and improve the ratio of computation to communication is very important. In light of this, the characteristics of link layer in Hyper Transport and PCI Express buses which are very popular in recent years was studied. On the basis of this, link layer architecture for a 64-bit high-speed data transportation bus was proposed and some key technologies were researched. A 16-bit scrambler/descrambler, which can scramble or descramble a 16-bit data in one cycle, was designed. A lane-to-lane deskew logic, which can correct 5 cycles delay skews at most between two lanes, was also proposed. The verification results show that the function of our designs is correct.

Key words: board-level; data transportation bus; link layer; scrambler/descrambler; lane-to-lane deskew

计算机系统的数据通讯包括系统间通信,板间通信和板级通信,不同的通信类型具有不同的数据传输需求[1]。对系统设计者来说,总线接口最重要的功能特性是带宽、延时和设计复杂性。对于传统的并行技术,随着处理器时钟的加速和数据传输速度的提升,主要的制约因素是在现有并行策略下有限的带宽和高速设计的复杂性。对于串行技术,主要的制约因素是时钟编码和解码、包封装问题和延时问题。在并行总线中,只有所有的并行线路的时钟和主时钟相同步,接收设备才能将数据及时接收,而当主时钟频率提高到一定的程度,由于并行传输的多条线间的延迟差与时钟周期时间相接近,导致接收方越来越难通过主时钟采集到正确的信号。在板级芯片间通信领域,高速传输总线的典型代表为超传输(HyperTransport, HT)总线[2],PCI Express(PCI-E)总线[3]和QPI总线[4],它们均具有高带宽的特点。PCI-E使用了嵌入时钟的串行数据传输方式,数据的发送和接收分别需要对时钟进行编码和恢复,增加了数据的传输延迟,因此不适合用于对数据传输延时要求非常高的芯片互连。HT总线的设计目的之一就是需要适用于板上芯片间的高速互连,因此低延迟是该总线的主要特点,QPI总线也与HT类似。HT总线采用点到点互连的

* **收稿日期**:2011-06-20
基金项目:国家自然科学基金资助项目(61003075);湖南师范大学青年基金资助项目(数100636);国家"核高基"重大专项"超高性能CPU新型架构研究"资助项目(2011ZX01028-001-001)
作者简介:周宏伟(1980—),男,助理研究员,博士。

方式,提供了频率从 200MHz 到 1GHz,链路宽度从 2 位到 32 位各种不同的链路[5],目前超传输协议已经发布了 3.0 版本[6],频率更高,带宽更大。随着我国高性能服务器和巨型计算机的发展,对板上高速互连总线的设计要求越来越高。在单块主板上集成 2 到 4 个高性能 CPU,并通过高速互连总线将这些 CPU 构建成 2 ~4 路 SMP 系统,是提高系统集成密度和计算性能的主要方式[7]。目前自主 CPU 处理速度已达到 GHz 以上,如何使芯片间的数据传输延迟更小,提高计算通信比是需要解决的重要问题。在设计自主高性能系统中的高速传输总线时,采用现有的商业化知识产权(Intelligent Property, IP)核必须按照 IP 本身的协议要求进行直连接口层次及协议的设计,不但增加了协议设计的复杂性,限制了自主协议设计的灵活性,而且额外的协议转换开销也降低了整体性能。

本文将根据自主高性能系统的需求,研究板级高速传输总线链路层中的关键问题,在吸收主流高速总线的设计特点的基础上,提出以下设计与实现方案:(1)64 位高速传输总线链路层体系结构设计;(2)数据加解扰器的设计及实现;(3)线间传输延迟偏斜纠正(lane-to-lane deskew)器的设计与实现。以上研究对于设计自主协议的板级高速传输总线具有非常重要的意义。

1 板级高速传输总线链路层体系结构

为了减小传输延迟、提高传输效率,传输总线协议应该尽可能减少协议层次。传统的 PCI-E 总线具有物理层、链路层、数据传输层和软件层 4 个层次,链路层传输数据链路层协议(Data Link Layer Protocol,DLLP)报文、数据,传输层中为传输层协议(Transfer Layer Protocol,TLP)报文,用户层传输各种事务(transaction)报文。数据在多个层次间进行格式转换,每个层次增加的报文头信息降低了报文的最大有效带宽,传输效率受到影响。我们在之前的研究工作中已经基于 PCI-E 协议设计了一种多核微处理器互连接口[9],通过优化较好地平衡了不同报文类型对带宽和传输延迟的要求,但是 PCI-E 协议强调的是高带宽,其协议层次较多,物理链路延迟较大,已逐渐不能满足当前板级芯片间互连的要求。本文在之前工作的基础上,以降低传输延迟为主要目的,进一步精简互连接口的层次,优化互连协议,提出了一种板级高速传输总线协议层次结构,如图 1 所示。协议层根据芯片互连的特点优化设计了精简的互连协议报文,该层相当于传统 PCI-E 总线中的软件层和数据传输层。链路层优化了数据传输协议,链路管理和信用控制等特殊报文和数据报文统一转换为 64 位对齐的物理微包。物理层针对传输延迟进行优化。

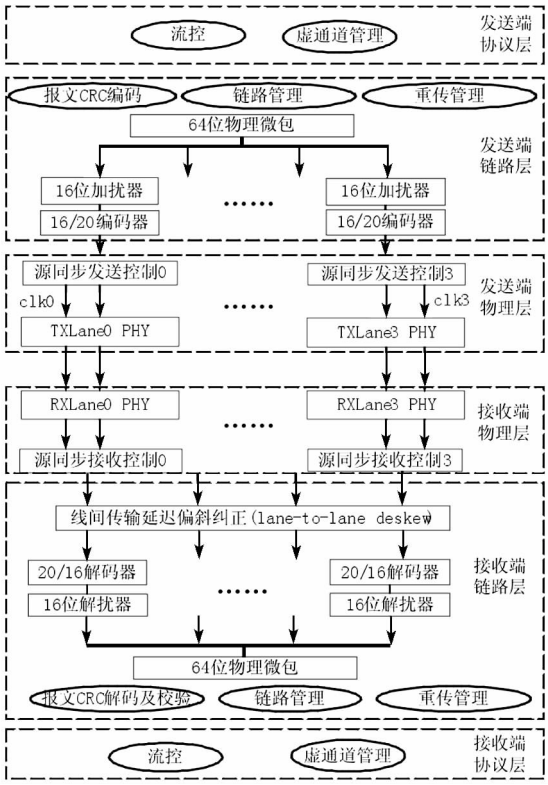

图 1 板级高速传输总线协议层次结构示意图

Fig. 1 Layer structure of board-level high-speed data transportation bus

优化后的高速传输总线协议具有物理层、链路层和协议层 3 个层次。物理层主要完成时钟和数据在物理链路上的传输,为提高传输带宽,物理层链路采用低压差分信号(Low-Voltage Differential Signaling,LVDS)传输技术,以点到点方式进行高速链接。LVDS 的特点是可以使用非常低的运行电压,转换速度快且具有很强的抗干扰能力。物理层还需要考虑时钟传输方式的选择是选择源同步还是时钟数据恢复,前者延迟小但不适合长距离传输,考虑到板级通信距离较短且对延迟要求较高,本文采用了源同步方式,通过消除 PCI-E 总线等使用串行链路时的时钟恢复时间降低传输延迟。物理链路设计中吸收了并行总线中位链路(bit lane)的特点,每 16 位数据信号组成一组位链路,使用一根时钟信号线,传输 64 位信号共需 4 个时钟信号(clk0 到 clk3)。每组时钟与数据通过源同步发送和接收控制器、发送端物理链路(TXLane PHY)和接收端物理链路(RXLane PHY)完成在物理层的传输。每对差分信号线上

的数据以超过 GHz 频率进行传输时,会遇到与传统高速串行链路相同的问题:(1)重复模式导致大量能量集中在离散频率上,进而生成大量的电磁干扰(Electro Magnetic Interference,EMI)噪音;(2)物理链路中信号线通过电容耦合方式传输信号时需要发送方保持所发送的 0 和 1 的数目尽可能相同,从而确保接收到的信号没有直流(Direct Current,DC)分量,实现 DC 平衡;(3)由于每条链路都有其各自的延迟且可能各不相同,导致通过不同物理链路达到接收方链路层数据可能相差 1 个或者多个时钟周期,造成接收方合并后的并行数据出错。为了解决以上问题,我们在链路层中设计了加解扰器、编解码器和线间传输延迟偏斜纠正(lane-to-lane deskew)逻辑。另外,链路层中还设计有报文 CRC 编解码和校验、链路管理和重传管理模块,分别完成报文的 CRC 编码、CRC 解码及校验、链路的初始化和物理微包的重传控制。协议层主要负责对协议报文的虚通道管理和流控等。

需要说明的是,如果物理链路传输距离较短,也可以不通过电容耦合的方式进行信号传输,这种情况下 16/20 编码器和 20/16 解码器可以被旁路,节省编码和解码的时间,进一步缩小数据传输延迟。本文在第 2 节和第 3 节将重点对板级高速总线链路层中关键的加解扰器和线间延迟偏斜纠正器进行设计,第 4 节是对这些设计的功能验证及验证结果分析。

2　16 位加解扰器的设计

重复模式会导致大量能量集中在离散频率上,进而生成大量的 EMI 噪音,如 10101010 这样的序列属于重复模式。扰频器可阻止发送数据流中重复模式的生成,数据流被加扰后将生成一个伪随机的比特模式,传输所辐射出的 EMI 能量分布在一个频谱范围内,减少特定频率所辐射的能量。通过线性移位反馈寄存器(LFSR)实现随机数的方法在信号传输领域被广泛使用,LFSR 的逻辑设计简单,一个 n 阶的 LFSR 能够产生连续的 2^n-1 个随机数且不重复。在发送方将原始的数据序列与随机数进行异或操作,可以打破原始数据序列中重复模式。LFSR 采用异或操作的优点是在接收方通过将接收到的数据与加扰时的随机数再次进行异或操作,能够恢复原始数据。超传输和 PCI-E 协议分别使用了 23 阶[6]和 15 阶[8]的 LFSR,本征多项式分别如式(1)和式(2)所示。自主链路接口设计为 4 条 lane 的链路传输接口,每条 lane 每个时钟节拍最大支持能够加扰 16 位的数据,因此本文基于式(2)设计 LFSR 逻辑,实现对数据的加扰和解扰。基于 16 位 LFSR 的加扰器的原理如图 2 所示。

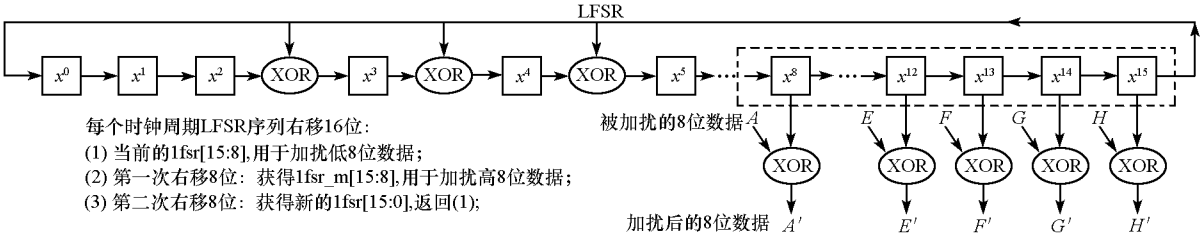

图 2　基于 16 位 LFSR 的加扰器的原理图

Fig. 2　Mechanism of the scrambler or descrambler based on 16-bit LFSR

$$G(x)=x^{23}+x^{18}+1 \quad (1)$$

$$G(x)=x^{15}+x^4+x^3+x^2+1 \quad (2)$$

在图 2 中,x^i 代表 LFSR 中 16 个寄存器中第 i 个寄存器的当前值,本文中用 lfsr[i]表示。这些寄存器在系统初始化时被复位为 1,LFSR 中的 16 位寄存器的初始值为 16'hFFFF。lfsr[15:8]可以用于对一个 8 位的输入数据{A,B,C,D,E,F,G,H}进行加扰。对于一个 16 位的输入数据,lfsr[15:8]用于对输入数据低 8 位数据进行加扰,对于输入数据的高 8 位,首先需要对当前 lfsr[15:0]右移 8 次后得到 lfsr_m[15:0],然后采用 lfsr_m[15:8]进行加扰,接着将 lfsr_m[15:0]再次右移 8 次后,产生新的 lfsr 的值 lfsr_nxt。在 LFSR 实现中,真正使用一个 8 倍[8]或者 16 倍于工作时钟的移位时钟完成一次加解扰操作中 LFSR 的 8 次或者 16 次移位非常困难,即使加解扰器的工作时钟为 500MHz,实现 8 倍数据的加解扰也需要 4GHz 的移位时钟,不可能实现。

本文根据 LFSR 的原理设计了如图 3 所示的逻辑,通过使用组合逻辑的方式可以在只使用工作时钟的情况下获得 lfsr_m 和 lfsr_nxt,用于完成对 16 位数据的加解扰以及 LFSR 的移位,不仅节省移位时钟,而且降低了设计的复杂度。lfsr_m 按照如下的方法获得:由于 lfsr_m[15:0]是 lfsr[15:0]右移 8 个移位时钟节拍之后的结果,根据图 2 中 LFSR 的原理图,从 x^8 到 x^{15} 再从 x^0 到 x^2

之间仅为简单的串连关系，因此 lfsr_m[0]、lfsr_m[1]和 lfsr_m[2]分别是 8 个移位时钟节拍之前的 x^8 到 x^{10} 中的值，分别为 lfsr[8]、lfsr[9]和 lfsr[10]。考虑到在 x^2 到 x^3 之前有一个异或运算器，因此 lfsr_m[3]的值等于上一移位时钟节拍中 x^2 中的值（为 lfsr[11]）和 x^{15} 中的值（为 lfsr[8]）互相异或的结果，因此 lfsr_m[3]的计算表达式为 lfsr[11] ∧ lfsr[8]。lfsr_m[15:4]中每一位的生成逻辑表达式通过依次类推即可获得，在图 3 中全部列出。获得 lfsr_m[15:0]后，再次右移 8 个移位时钟节拍，可以获得 lfsr_nxt[15:0]，其计算方法与计算 lfsr_m 的方法完全相同，只需要将图 3 中所有表达式等号右边的 lfsr 替换为 lfsr_m，等号左边的 lfsr_m 替换为 lfsr_nxt 即可获得产生 lfsr_nxt 的所有表达式。在实现过程中，必须保证对一个 16 位数据进行加扰和解扰操作时使用同一对 lfsr 和 lfsr_m，只有这样加扰和解扰才属于一对互逆的运算，解扰后的数据才会与加扰前的数据相同。为了实现这个目的，发送方和接收方中的 LFSR 逻辑只有在发送或接收一个有效的数据时工作，且它们的初始值相同。

```
lfsr_m[0] = lfsr[8]
lfsr_m[1] = lfsr[9]
lfsr_m[2] = lfsr[10]
lfsr_m[3] = lfsr[11] ∧ lfsr[8]
lfsr_m[4] = lfsr[12] ∧ lfsr[9] ∧ lfsr[8]
lfsr_m[5] = lfsr[13] ∧ lfsr[10] ∧ lfsr[9] ∧ lfsr[8]
lfsr_m[6] = lfsr[14] ∧ lfsr[11] ∧ lfsr[10] ∧ lfsr[9]
lfsr_m[7] = lfsr[15] ∧ lfsr[12] ∧ lfsr[11] ∧ lfsr[10]
lfsr_m[8] = lfsr[0] ∧ lfsr[13] ∧ lfsr[12] ∧ lfsr[11]
lfsr_m[9] = lfsr[1] ∧ lfsr[14] ∧ lfsr[13] ∧ lfsr[12]
lfsr_m[10] = lfsr[2] ∧ lfsr[15] ∧ lfsr[14] ∧ lfsr[13]
lfsr_m[11] = lfsr[3] ∧ lfsr[15] ∧ lfsr[14]
lfsr_m[12] = lfsr[4] ∧ lfsr[15]
lfsr_m[13] = lfsr[5]
lfsr_m[14] = lfsr[6]
lfsr_m[15] = lfsr[7]
```

图 3 lfsr_m 生成逻辑表达式

Fig. 3 Logic expression for lfsr_m

3 线间传输延迟偏斜纠正器的设计

每条链路上都有其各自的延迟，这些延迟包括芯片封装时本身管腿的延迟以及在 PCB 上的传输延迟等，在芯片封装及 PCB 设计时很有可能造成各个链路上的延迟不同，另外为了降低封装和 PCB 连线设计的复杂度，也需要链路之间能够容忍一定程度的延迟偏斜（skew）。在这延迟偏斜的影响下，接收端从不同链路接收到的数据无法正常地按照发送的节拍拼装，造成各链路数据不同步。图 4 为四条链路间产生延迟偏斜的例子，发送端和接收端之间的链路的延迟不同，链路 0 和链路 2 的延迟为 D，链路 1 的延迟为 D + 1，链路 3 的延迟为 D + 2。在延迟偏斜的影响下，接收端在 t' 时刻不能正确接收到“应该”接收到的发送端在 t 时刻发出的数据{D0, D1, D2, D3}，而接收到了{D0, X, D2, X}，导致接收数据错误。同样，在 $t'+1$，$t'+2$ 以及之后的每一个时钟节拍，接收端所接收的各链路上的数据均不能正常同步。为了解决延迟偏斜问题，本文设计了一套线间传输延迟偏斜纠正逻辑电路，其工作原理示意图如图 5 所示。

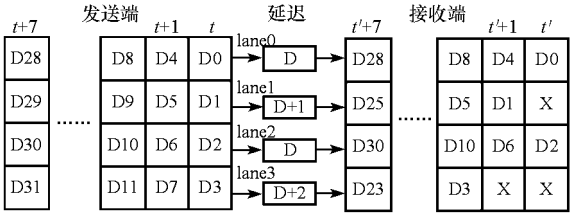

图 4 链路间产生延迟偏斜的例子

Fig. 4 Example for how does lane-to-lane skew arise

在系统初始化时，在每条链路上同时发送一组特殊的位流，称之为同步控制符（COM）。由于链路延迟的影响，这组控制符将会不等时地到达接收端，接收端采用某种方法记录下各链路接收到 COM 控制符的时机，根据接收到 COM 控制符的先后次序，对各链路延迟进行修正，实现各链路延迟的等长。一旦完成链路延迟修正，系统正常工作时可以保证各链路接收到的数据的同步到达。如图 5 所示，t 时刻发送端在每条链路上同时发送用于同步的 COM 控制符，为 COM0 ~ COM3，接收方的链路于 t' 时刻收到 COM0 和 COM2，于 $t'+1$ 时刻收到 COM1，于 $t'+2$ 时刻收到 COM3。经过传输延迟偏斜纠正器的调节，lane0 和 lane2 增加两个节拍的延迟，lane1 增加一个节拍的延迟，lane3 不改变延迟。经过调整，在 $t'+2$ 时刻，接收端的 4 条链路上将分别输出 COM0、COM1、COM2 和 COM3，四条链路达到了延迟相同，接收数据被同步。

本文提出的线间传输延迟偏斜纠正器支持对 4 通道链路延迟偏差的纠正，能够纠正线间传输延迟偏差最大为 5 个时钟周期，保证链路正常工作，其逻辑结构示意图如图 6 所示，采用级联寄存器的方式实现，每条链路均使用 6 个级联的寄存器（d0 ~ d5）构成先进先出队列保存接收到的数据。在数据接收的过程中，固定指定某条链路为

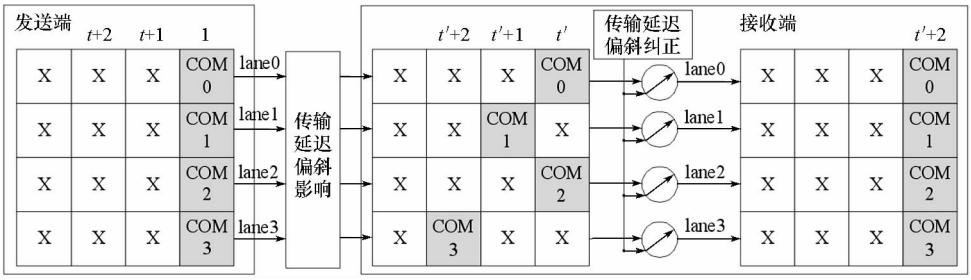

图 5　线间传输延迟偏斜纠正器的工作原理图

Fig. 5　Mechanism of the lane-to-lane deskewer

主链路,例如指定第一条链路(lane1)为主链路,当在该链路上发现接收到 COM 控制符时,检查其他链路的接收队列中是否存在 COM 控制符,如果没有,则继续接收数据。当所有的链路的接收队列均发现 COM 控制符后,则记录这些 COM 控制符在各自接收队列的寄存器的位置,校准完成。图 6 中主链路在其 d5 寄存器中发现 COM 控制符,然后检查其他 3 条链路的 COM 控制符,分别在 lane0、lane2 和 lane3 的 d3、d2 和 d0 发现 COM 控制符,将 COM 符所在的位置记录在控制逻辑中。在正常接收阶段,各条链路的数据将从之前记录的 COM 控制符所在的寄存器中获得,Lane0_Sel 到 Lane3_Sel 共 4 个选择信号分别控制从各链路的对应寄存器中获得数据,由控制逻辑提供。由于设计中采用了每条链路的接收队列为 6 个寄存器,因此任何两条链路接收到的 COM 控制符最多允许相差 5 个时钟周期。如果链路之间的延迟偏差过大,即当主链路的 COM 控制符移动到 d5 寄存器后,其他的某条链路的接收队列中仍未出现 COM 控制符,就表明这两条链路之间的延迟偏差超过了线间传输延迟偏斜纠正电路的设计容限,链路将无法正常工作。

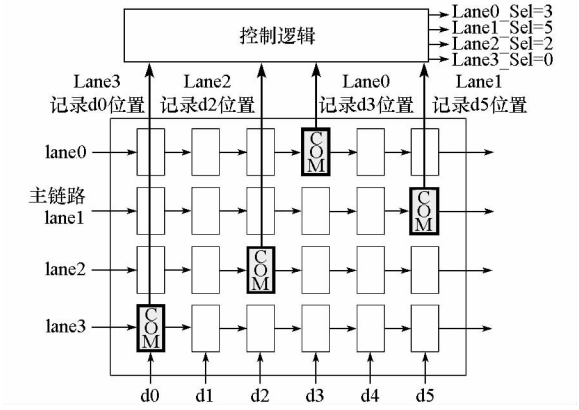

图 6　线间传输延迟偏斜纠正器的逻辑结构示意图

Fig. 6　Logic structure of lane-to-lane deskewer

4　功能验证及结果分析

为了验证本文提出的链路层体系结构及关键技术,我们设计了链路层测试结构,并对加解扰器、线间传输延迟偏斜纠正器提出了专门的功能验证机制,使用 Verilog 语言实现了关键模块,搭建了测试平台,通过 Cadence 公司的 NCVerilog9.2 进行模拟仿真。首先,我们根据本文提出的板级高速传输总线链路层体系结构对链路层进行了 RTL 级的模块设计和测试环境构建。图 7 为构建的链路层测试结构示意图。如图所示,64 位发送产生器用于产生随机的 64 位发送数据。64 位发送数据按照从低位到高位的顺序拆分为 4 个 16 位数据,送入本文设计的 4 个 16 位数据加扰器。4 条链路使用源同步技术,在发送时钟 clk_tx 的控制下源同步接口产生的 clk$i$ 和 tx_di(i=0~3)一起通过物理链路发送到接收方,接收方在接收时钟 clk_rx 的控制下接收来自物理链路的数据,送入线间传输延迟纠正器进行延迟纠正,完成 4 条链路的同步。获得同步后的接收数据 rx_di(i=0~3)分别使用对应的 16 位数据解扰器进行解扰,最后按照从低位到高位的顺序合并为 64 位的接收数据。物理链路使用延迟加载器模拟链路的延迟,通过将链路上的信号寄存 n 拍的简单方法可以实现 n 个时钟周期的延迟加载,设置每条链路上信号寄存的节拍数不同可以模拟出链路延迟不同的情况。在单独验证加扰和解扰的功能时,我们利用了加扰和解扰是一对互逆运算的特性,比较初始数据和经过加扰和解扰后的数据是否一致,确定本文设计的加扰和解扰器功能是否正确。加解扰器的功能验证机制如下:(1)首先用加扰器对初始数据进行加扰;(2)对加扰后的数据使用解扰器进行解扰,解扰器和加扰器的 LFSR 的初始值相同;(3)将解扰后的数据与初始数据进行对比:如果两个数据一致,则证明对该数据的加扰和解扰的逻辑功能正确。测试过程中,测试激励产生初始数据的方法采用了穷举法。在单独验证线间传输延迟偏斜纠正逻辑时,旁路加解扰器,发送方首先发送 64 位的控制符序列{COM0,COM1,COM2,COM3},通过 4 条链路分别发送,然后每时钟周期发送 64 位的数据序列{i,i+1,i

$+2, i+3\}$,其中 i 从 0 开始且每时钟周期递增 1。4 个延迟加载器随机产生从 0 到 5 个时钟周期的延迟且各延迟之间的差值不超过 5 个时钟周期。接收方将延迟纠正器后的数据与发送数据比较,通过验证是否相等证明传输延迟偏斜纠正逻辑是否正确。

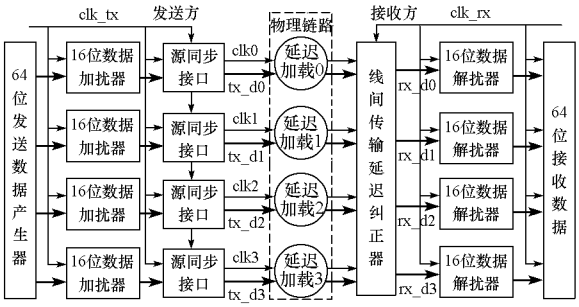

图 7 链路层测试结构示意图

Fig. 7 Structure of link layer testbench

图 8 为使用 NCVerilog 对线间传输延迟偏斜纠正器进行仿真的部分仿真波形。如图所示,在 t 时刻 4 条链路同时发送 COM 符(16′h00bc),延迟加载器分别在 4 条链路上加载 0、1、2 和 3 个时钟周期的延迟,在 $t+3$ 时刻链路 0 首先收到了 COM 符,接着其他的链路分别在 $t+4$,$t+5$ 和 $t+6$ 时刻收到 COM 符。当延迟最长的链路 3 的 COM 符在 $t+6$ 时刻到达时,所有链路均发现了 COM 符,进行 COM 对齐操作,控制逻辑记录当前 COM 符所在位置。从 $t+7$ 时刻开始,所有链路上接收到的数据与发送的数据同步。经过长时间的仿真,我们测试了大量数据的加扰和解扰,测试了各种线间延迟偏斜情况下数据的发送和接收,结果表明测试过程中接收端得到的数据与发送端发送出的原始数据完全一致,证明本文设计的 16 位数据加解扰器和线间传输延迟偏斜纠正器功能正确。

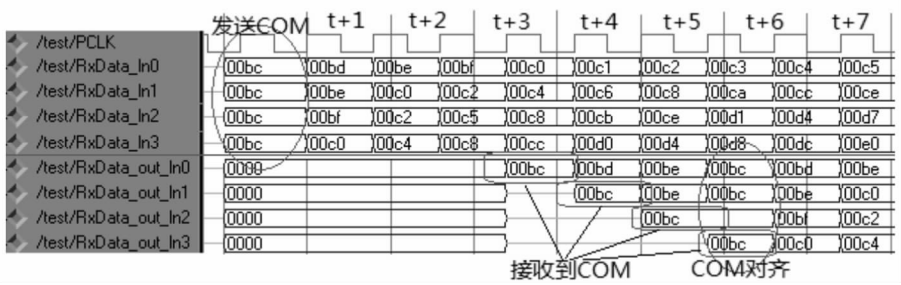

图 8 线间传输延迟偏斜纠正器的仿真波形

Fig. 8 Wave for lane-to-lane deskewer simulation

5 结 论

本文研究了板级高速传输总线链路层关键技术。首先研究了近年来发展迅速的超传输总线和 PCI-E 总线链路层特点;然后对板级高速传输总线的链路层的关键技术进行研究,提出了一种 64 位高速传输总线链路层体系结构;接着重点对其中关键的加解扰技术和线间传输延迟偏斜纠正技术进行了深入研究,提出了一种能够每时钟周期对 16 位数据进行加解扰的加解扰器以及一种能够纠正链路间最大 5 个时钟周期延迟偏斜的线间传输延迟偏斜纠正器;最后通过构建测试环境对所做的设计进行功能验证,结果表明本文所提出的设计功能正确。未来的工作包括对编解码器的优化设计、链路管理和重传管理模块的设计以及对 8 ~ 16 链路的扩展支持等。

参 考 文 献:

[1] Cavalli M. Advanced Interconnect Standards Blend Serial and Parallel Techniques for Best Performance and Scalability[S]. USA. HyperTransport Consortium. 2007:1 - 2.

[2] HTC_WP02 - 2004. HyperTransport I/O Technology Overview[S]. USA. HyperTransport Consortium. 2004.

[3] Holden B. Latency Comparison between HyperTransport and PCI Express in Communications Systems [S]. USA. HyperTransport Consortium. 2007.

[4] 320412 - 001US. An Introduction to the Intel QuickPath Interconnect[S]. Intel Inc, 2009:3 - 21.

[5] 刘涛,刘光明,郭御风. 高性能 I/O 互连协议 HyperTransport 链路接口的研究与实现[J]. 计算机工程与科学. 2006, 28(4): 50 ~ 53.

[6] The HyperTransport Consortium. HyperTransport I/O Link Specification Revision 3.00d [S]. USA. HyperTransport Consortium. 2008:41 - 44, 49 - 50.

[7] 陈超. HyperTransport 超传输关键技术研究[D]. 长沙: 国防科学技术大学, 2009.

[8] Ravi B, Don A, Tom S. PCI Express System Architecture[M]. USA: MindShare. Inc, 2003.

[9] 周宏伟,邓让钰,窦强,等. 一种多核微处理器互连接口的设计与性能分析[J]. 国防科技大学学报. 2010, 32(4): 94 - 99.