文章编号：2095-4980(2013)06-0897-05

一种多通道动态均衡先进先出缓存机制

谢廷婷[1]，彭鼎祥[2]，郑积仕[1]

(1.福建工程学院 计算机与信息科学系，福建 福州 350108；2.星网锐捷网络有限公司，福建 福州 350002)

摘　要：为了在多通道数据传输中实现各通道数据的均衡与高效传输，提出一种基于现场可编程门阵列(FPGA)的多通道动态先进先出缓存设计方法。该方法通过建立小型存储单元，动态维护各通道的数据传输状态，并采用数据库索引技术，为各通道数据高速、动态地分配缓存空间。本文以 FPGA 为平台，实现了该方法。仿真实验结果表明，该方法能通过较少的缓存设置，实现各个通道的均衡传输，节约存储资源，并提高各通道的数据传输效率。

关键词：数据缓存；先进先出；存储单元；性能分析；现场可编程门阵列

中图分类号：TN92　　**文献标识码：**A　　**doi：**10.11805/TKYDA201306.0897

A multi-channel dynamic equilibrium FIFO caching mechanism

XIE Ting-ting[1]，PENG Ding-xiang[2]，ZHENG Ji-shi[1]

(1.Computer and Information Science Department，Fujian University of Technology，Fuzhou Fujian 350108，China；
2.Ruijie Networks Co.，Ltd.，Fuzhou Fujian 350002，China)

Abstract：A multi-channel dynamic First Input First Output(FIFO) caching design method based on Field Programmable Gate Array(FPGA) is proposed in order to achieve a balanced and efficient transmission of the channel data in the process of multi-channel data transmission. The method, which is realized on FPGA platform, manages to dynamically maintain the data transmission status of each channel by establishing a small storage unit, and dynamically allocate the cache space for each channel data quickly by using the database indexing technology. Experimental results show that the method can achieve balanced transmission of each channel, save the storage resources and improve the efficiency of data transmission for each channel with fewer caching settings.

Key words：data caching；First Input First Output；Cell；performance analysis；Field Programmable Gate Array

多通道数据缓存操作在数据通信传输系统中经常出现，例如在同步数字体系(Synchronous Digital Hierarchy，SDH)通信网络中，有多种接口类型，它们对传输带宽的需求不同；又例如在端到端服务质量(End to End Quality of Service)解决方案中，从源端到目的端的一条流就可以看成是具有特定带宽需求的一个数据通道。因此，一个数据处理器往往需要处理多个具有不同带宽需求的数据通道，这种情况可以称为带宽非均衡多通道数据传输。那么在数据处理器中，为了实现数据的安全传输，需要为各个数据通道提供缓存，并需要设计一种缓存机制在保证数据安全的前提下，实现各个通道的均衡传输，从而提高数据处理器的总传输带宽。

1　现有缓存机制及其缺点

在解决上述带宽非均衡的多通道数据缓存问题上[1-2]，现有方法主要有：

1) 固定深度先进先出(FIFO)缓存

即为每个数据通道分配固定深度的 FIFO 缓存。其缺点是当某个通道无数据传输时，将浪费存储资源，对于一定数量级的多通道数据来说，这个方法是不能接受的。

2) 深度可预设 FIFO 缓存

收稿日期：2012-10-26；**修回日期：**2012-12-10
基金项目：福建工程学院校科研发展基金青年项目(GY-Z13011)

将存储资源划分为一组相同大小的处理单元，称为存储单元，简称 Cell，并采用 3 个列表来进行维护：通道指针列表、缓存空间列表和链表指针列表。通道指针列表记录各个通道的 FIFO 缓存的首个 Cell 的 Cell 指针；缓存空间列表记录所有的 Cell 被哪个通道的 FIFO 缓存所占用；链表指针列表保存各个 Cell 所链接的下一个 Cell 指针。目前绝大多数芯片都采用这种方法实现 FIFO 缓存，其优点是深度可预先配置，缺点是逻辑设计和预设操作复杂，而且在设置之后，各个 FIFO 缓存的深度是固定的。

对于这个方法，在预先配置完成后，某个通道的缓存深度是固定的，这样并不能提供很高的灵活性，因此如果某个通道没有传输数据，其分配的存储单元将被浪费。

2 多通道动态 FIFO 缓存机制

2.1 方法总体说明

为了避免上述现有方案的缺点，本文提出了多通道动态 FIFO 缓存机制[3]。通过可编程设计技术，在 FPGA 芯片内部以及多路数据缓存和传输过程中能够实现各个端口之间的均衡数据传输[4]。结构框图见图 1，主要包含数据存储器、存储单元索引模块、通道索引模块、写入控制模块和读出控制模块。其中数据存储器就是一般的 FIFO 逻辑，不再赘述，而其他各个模块详细描述如下。

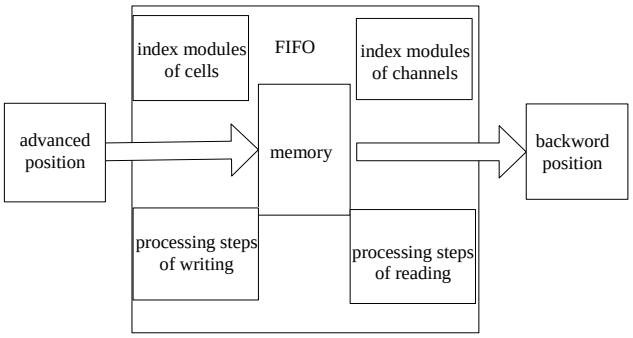

Fig.1 A multi-channel dynamic FIFO
图 1 多通道动态 FIFO

2.2 存储单元索引模块

存储单元索引模块如图 2 所示，分为 3 个部分，并完成以下 3 个任务：

1) 维护通道指针列表。通道指针列表用来指示通道 FIFO 缓存中首个 Cell 的地址指针，例如通道 0 记录保存的是 Cell 102 的地址指针，当通道 0 需要输出数据时，就从 Cell 102 读出数据。

2) 维护未用 Cell 列表。未用 Cell 列表用来保存未被分配给通道的 Cell，未用 Cell 列表可以最多保存 1 024 个 Cell，即本设计总共有 1 024 个 Cell 可提供给各个通道使用。如图 2 中，记录 0 保存着 Cell 99 的地址指针，当某个通道需要分配新的 Cell 来保存时，则将记录 0 的 Cell 99 从未用 Cell 列表中删除，并且将 Cell 99 分配给该通道用于数据缓存，同时将 Cell 99 的地址指针添加到通道索引模块的相关指针列表中。

3) 维护未用 Cell 统计。统计未用的 Cell 数，从而计算数据存储器中剩余的存储空间，以得出该数据存储器是否有足够的存储空间[5]。

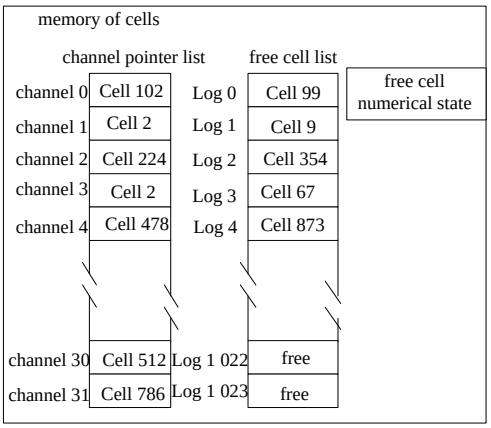

Fig.2 Index modules of cells
图 2 存储单元索引模块

2.3 通道索引模块

通道索引模块的数据组织形式如图 3 所示，该模块用来存放各个通道已经占用的 Cell 的指针。在写入控制模块进行写入操作的过程中，需要给某个通道分配新的 Cell 时，通道索引模块从存储单元索引模块的未用 Cell 列表读出 1 个 Cell 指针提供给写入控制模块，同时将 Cell 指针写入该通道的指针列表中，表示该通道占用该 Cell；在读出控制模块进行读出操作的过程中，1 个 Cell 数据被读完时，通道索引模块将该 Cell 指针从该通道的指针列表中读出并删除，同时将该 Cell 的指针写入到存储单元索引

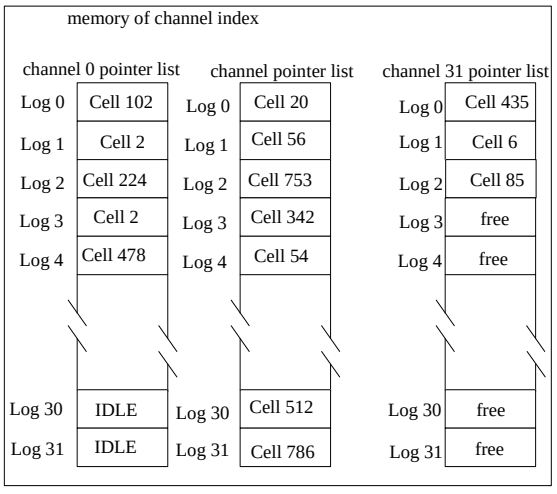

Fig.3 Index modules of channels
图 3 通道索引模块

模块的未用 Cell 列表中。

这里如果每个通道最多占用 32 个 Cell(图 3 中通道 1 就占用了 32 个 Cell),那么 32 个通道就需要 32×16×32=16 K 字节的存储空间。此外，通道索引模块还能统计各个通道占用的存储单元数，从而得到各个通道目前缓存的数据量大小。通过通道索引模块，可以得出：1) 如果该通道的数据流量大，数据就容易在 FIFO 缓存中累积[6]，则相应的 FIFO 缓存所占用的 Cell 数就多；2) 如果该通道的数据流量小，数据就不容易在 FIFO 缓存中累积，则相应的 FIFO 缓存所占用的 Cell 数就少；3) 如果该通道没有流量，则没有数据累积，就不分配 Cell；4) 一个通道的 FIFO 缓存最多只能占用 32 个 Cell，存在上限。

2.4 写入控制模块

写入控制模块主要完成以下 3 个工作：1) 通过调度模块判断哪个通道需要进行数据传输；2) 往数据存储器中写入数据，并维护好当前的 Cell 指针；3) 在某些时候进入丢帧周期，丢帧机制模块丢弃无法传输的帧数据。

1) 调度模块

调度模块采用固定优先级(Strict-Priority，SP)调度算法实现多通道选 1 的操作，也就是通过对符合数据传输条件的通道进行调度，从而选择其中的一个通道作为数据传输的通道。这里符合数据传输条件的通道指的是：a) 该通道的上游模块有发送数据请求；b) 该通道在数据存储器中占有的存储单元小于通道缓存门限。调度模块将调度结果，即通道号发送给写入操作模块。

2) 丢帧机制模块

丢帧机制模块用来做出决策，决定某个通道是否丢弃上游模块的数据，即指如果某个通道符合丢帧条件，则整帧丢弃这个通道进入的帧数据，即尾部丢弃。丢帧的条件为：a) 数据存储器中某个通道占用的存储单元大于通道缓存门限；b) 各个通道占用的总存储器资源大于总缓存门限；c) 当前操作帧是处于帧间位置。丢帧机制模块将决策后的通道号发送给写入操作模块。

3) 写入操作模块

写入操作模块的操作流程如图 4 所示，具体操作步骤如下：a) 在 IDLE 状态，如果有通道符合数据传输条件或者符合丢帧条件，则进入 PRI 状态；b) 在 PRI 状态，根据固定优先级调度得到将要进行数据写入的通道号，如果这时通道符合数据传输条件，则进入 GETPOINT 状态;如果这时通道符合丢帧条件,则进入 DROP 状态;c) 在 GETPOINT 状态，根据通道号得到当前的操作地址指针，并进入 WRITE 状态；d) 在 WRITE 状态，读取上游模块该通道的数据并写入到数据存储器，如果这时地址跨存储单元边界(如果一个存储单元为 32 字节，则边界是低 5 位为 0)，则进入 CELL 状态；如果上游模块无数据请求，则进入 END 状态；e) 在 CELL 状态，将目前的存储单元指针写入到通道索引模块的相应通道中，同时如果该通道所占用的存储单元数小于某个限度，则从存储单元索引模块中读取 1 个新的存储单元指针，并赋予当前写操作指针；如果该通道所占用的存储单元数超过一定的限度，则为该通道打上满标记，告知下一个数据帧不再进入数据存储器。f) 在 DROP 状态，丢弃从上游模块读取的帧数据，直到以下 2 种情况出现就返回 IDLE 状态。这 2 种情况是指：上游数据模块该通道无数据传输请求；上游数据模块该通道有数据请求，并检测到一个帧结束标识，同时检测到不满足丢帧条件。

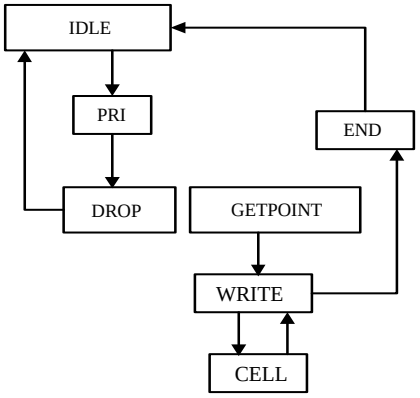

Fig.4 Processing steps of writing
图 4 写入操作流程

从写入控制模块的流程描述来看，数据帧输入某个通道后，Cell 是动态分配给该通道，这个过程无需进行预先设置。

2.5 读出控制模块

读出控制模块操作流程如图 5 所示，具体步骤如下：1) 在 IDLE 状态，如果接收下游有数据请求，则进入 PORT 状态；2) 在 PORT 状态，获取下游模块提供的通道号，进而获取当前的存储单元指针，进入 READ 状态；3) 在 READ 状态，读取整个存储单元的数据发送给下游模块，并将该通道的当前指针指向下一个 CELL 指针，结束数据读取，进入 IDLE 状态。

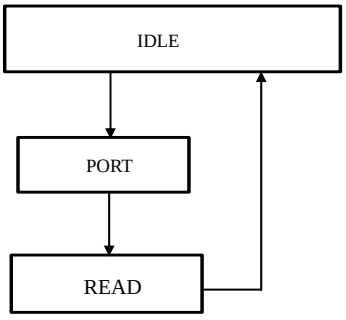

Fig.5 Processing steps of reading
图 5 读出控制操作流程

3 FPGA 实现及实验结果

3.1 FPGA 实现

本设计采用 Xilinx 公司的 FPGA 芯片[7−8]—Spartan6 芯片，具体芯片型号为 XC6SLX16FTG256-1。

在 Spartan6 芯片中，实现了 4 个复用 SDH 接口，可以配置为：E1/CE1/同步接口，即，整个设计最多可以提供 128 个 CE1 通道。如图 6 所示，本文所论述的设计，在收发通路上，各实例化为 4 个模块，形成收发的 128 个通道，这 128 个收发通道连接 128 通道的 HDLC 收发控制器，提供 128 路的动态缓存。

对于接收路径来说，32 个通道动态缓存 FIFO 模块接收来自 HDLC 控制器模块的数据，形成至多 128 个通道的数据队列，提供给 DMA 发送给 CPU。

对于发送路径来说，32 个通道动态缓存 FIFO 模块接收来自 DMA 的数据，形成至多 128 个通道的数据队列，提供给发送 HDLC 控制器模块，进而发送给接口。

可见，在整个设计中，动态缓存 FIFO 可以实现对片内缓存资源的管理，形成灵活的动态配置[9]。如果配置为 4 个 CE1 接口，就会管理 128 个缓存通道；如果配置为 4 个 E1 接口，就可以配置为 4 个缓存通道；依此类推。

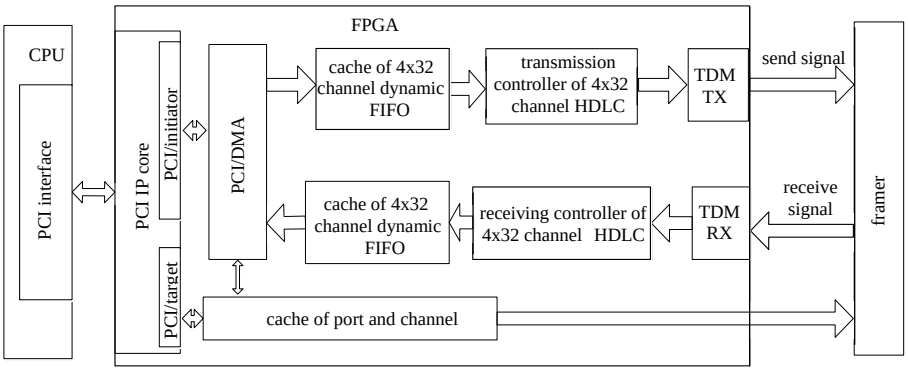

Fig.6 An instance based on a 32-channel dynamic FIFO

图 6 32 通道动态缓存 FIFO 机制实例

3.2 实验结果

1) 资源利用率对比

由表 1 可见，本文设计的 FIFO 机制，各方面资源都得到了节省，特别是用于数据存储的片内 Block Ram 资源，得到大大节省，从而使得 128 通道的设计能在 XC6SLX16FTG256-1 较小规模的 FPGA 芯片中实现。

表 1 不同 FIFO 机制资源对比

Table1 Resource comparison of different FIFO

comparison item	4x32 channels dynamic FIFO		other FIFO	
	resource amount	percentage of total resources/(%)	resource amount	percentage of total resources/(%)
slices	560	24.58	810	35.56
FFS	4 120	22.60	7 040	38.63
distributed RAM	1.2 Kb	0.80	2 Kb	1.47
block ram	256 Kb	44.44	2 048 Kb	355.56

2) 测试验证结果

本设计希望在小规模 FPGA 芯片[8−9]上实现多通道的数据传输，特别是实现 4 个 CE1 的 128 个通道线速传输，而多通道动态 FIFO 机制为这种设计提供了可能。在图 6 的设计实测中，得到如下数据：

a) 对于通道最多的情况：4 个 CE1 的 128 通道数据传输

① 每个通道支持 64 Kbps 线速收发(64 字节短报文)；

② 总共支持 4 个 CE1 通道的 8 Mbps 带宽；

③ 接收路径上未出现 Overrun 错误；

④ 发送路径上未出现 Underrun 错误。

b) 对于速率最快的情况：4 个同步口 32 Mbps(每个同步口支持 8 Mbps 传输率)

⑤ 64 字节短报文，32 Mbps 线速收发；

⑥ 接收路径上未出现 Overrun 错误；

⑦ 发送路径上未出现 Underrun 错误。

4　小结

4.1 缓存机制性能分析

本文提出的 FIFO 缓存机制，优点在于：

1) 存储资源的动态利用：对于一定的存储资源，利用本文的设计实现，可以为每个数据通道动态分配存储资源。如果某个通道的数据量大，则分配的存储单元 Cell 就越多，那么占用的存储资源就越多；相反，如果某个通道的数据量小，则分配的存储单元 Cell 就越少，那么占用的存储资源就越少；如果某个通道没有数据传输，则不占用存储资源。这样如果各个通道的数据传输带宽不均衡，则可以充分利用有限的存储资源为高速数据传输通道提供缓存，从而增加高速数据传输通道的缓存深度，进而保证数据的安全传输。2) 自动化程度高：本文提供的缓存机制，需要配置的参量只有存储单元 Cell 的大小和某通道最大占用的 Cell 数。存储资源的动态分配完全是逻辑设计自动完成。3) 支持极多通道的缓存设计：本文提供的缓存机制，可以支持极多通道的缓存设计。如果通道数量达到 10 万数量级以上，例如 65 536 个通道，那么固定深度 FIFO 的缓存机制和深度可预测的缓存机制都是不可行的。因为对于深度 FIFO 的缓存机制，耗费的存储资源不可接收；对于深度可预测的缓存机制，其复杂程度在设计上无法实现。

4.2 结论

在带宽非均衡多通道数据传输中，本文设计的基于 FPGA 的多通道 FIFO 缓存机制可以实现存储资源在多个通道之间的动态分配，使得高速通道可以得到充分的带宽。目前设计模型中 65 536 个通道缓存机制的传输带宽可以达到 10 Gbps，很大程度上提高了数据处理器的总传输带宽。而且该方法也为实现极多通道的缓存设计提供了一个很好的方案，具有广阔的应用前景。

参考文献：

[1] HUI Yi,ZHOU Zhijie. A cross layer resource allocation algorithm based on service priority[J]. Journal of System Simulation, 2009,21(17):5391–5395.)

[2] LIU Honglan,HE Weidong,GAO Qingshi. Distributed deployment and QoS performance analysis in a grid service scheduling algorithm[J]. Computer Science, 2011,38(6):31–34.)

[3] WANG Peng,GUO Qi,XU Dongming. A design and implementation of SDRAM for multi-channel data transmission[J]. Journal of China Integrated Circuit, 2009,18(1):49–53.)

[4] 夏金军,庄奕琪,包军林,等. 一种基于 FPGA 的高速数据缓存的设计[J]. 微计算机信息, 2008,24(11–2):226–228. (XIA Jinjun,ZHUANG Yiqi,BAO Junlin,et al. One design of data cache based on FPGA[J]. Microcomputer Information, 2008,24(11–2):226–228.)

[5] LI Zeng,DUAN Hancong,NIE Xiaowen,et al. Optimization of end to end route in hierarchical P2P administrative net[J]. Journal of Chinese Computer Systems, 2012,33(1):54–57.)

[6] 冯元伟,冯宗明. 基于 FPGA 和单片机的多通道同步触发信号发生器[J]. 太赫兹科学与电子信息学报, 2012,12(1):110–113. (FENG Yuanwei,FENG Zongming. The synchronization trigger signal generator of multi-channel based on FPGA and chip microcomputer[J]. Journal of Terahertz Science and Electronic Information Technology, 2012,12(1):110–113.)

[7] 王志鹏. 可编程逻辑器件原理与程序设计[M]. 北京:国防工业出版社, 2005. (WANG Zhipeng. The principle and program design in PLD[M]. Beijing:National Defense Industry Press, 2005.)

[8] LONG Zuli,WANG Ziyun. Test technology and realization on ATE[J]. Computer Engineering and Applications, 2011,47(6):65–67.

[9] 王龙,陈晓光. 基于 FPGA 的音频监控系统的设计与实现[J]. 太赫兹科学与电子信息学报, 2010,8(5):582–589. (WANG Long,CHEN Xiaoguang. Designing and implementing audio monitoring system with FPGA platform[J]. Journal of Terahertz Science and Electronic Information Technology, 2010,8(5):582–589.)

作者简介：

谢廷婷(1978–)，女，福州市人，硕士，讲师，研究方向为网络与数据通信.email: 11476763@qq.com.

彭鼎祥(1976–)，男，福州市人，博士，高级工程师，研究方向为数据通信.

郑积仕(1976–)，男，福州市人，博士，讲师，研究方向为智能控制.