

一种高性能异步 FIFO 的设计与实现

李 冬¹, 赵志凯²

(1 中国矿业大学 信息与电气工程学院, 江苏 徐州 221008; 2 中国矿业大学 计算机科学与技术学院, 江苏 徐州 221008)

摘 要: 提供了一种全新的高性能异步 FIFO 设计方案. 首先定义了 FIFO 的通信协议和总体结构设计, 然后围绕如何提高 FIFO 性能依次论述了存储阵列设计、读写控制逻辑和空/满判断逻辑的设计方法. 通过与 FPGA 本身的 FIFO 模块比较, 该方案可以提高 FIFO 性能 30% 以上.

关键词: 异步 FIFO; 亚稳态; 多时钟

中图分类号: TP302

文献标识码: A

文章编号: 1000-7180(2010)08-0145-04

Design and Implementation of High Performance Asynchronous FIFO

LI Dong¹, ZHAO Zhi-kai²

(1 School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou 221008, China;

2 School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221008, China)

Abstract: This paper presents a high performance FIFO design that interfaces subsystems on a chip working at different speeds. The communication protocol and the architecture of FIFO are proposed at first. Aim at improving its operation speed, the design of storage array, read/write control logic and full/empty status logic are described specifically. Compared with the existing FIFO block, the design's performance improvement ratio is more than 30 percent.

Key words: asynchronous FIFO; meta-stability; multiple clock

1 引言

随着计算机及电子器件的飞速发展, 高速处理器中的逻辑越来越复杂, 芯片规模不断扩大, 在一个芯片中常常含有多个不同的高速时钟. 当数据从一个时钟域传输到另一个时钟域时, 确保数据在高速状态下的可靠传输就成为一个重要问题. 异步 FIFO (First In First Out) 单元成为解决这类跨时钟域的数据可靠传输问题的不可缺少的元件.

异步 FIFO 是一个缓冲存储部件, 用于在两个不同时钟域之间进行数据交换. 数据从一个时钟域生成并写入 FIFO, 然后从另外的时钟域读出并被使用. 为了提高 FIFO 的性能, 需要精心设计 FIFO 的体系结构和电路实现. 为此, 文中对已有的工作进行了仔细研究. 不同的设计有不同的读写控制策略和空/满逻辑判断方法. 有些方法会导致性能降低, 比

如存储阵列会严重影响系统性能^[1]、在设计中使用减法器来进行空/满逻辑判断也会降低系统性能^[2]. 文中基于 FIFO 存储电路的功能和特点, 讨论了 FIFO 设计的关键技术, 重点探讨了高性能异步 FIFO 的设计和实现方法.

2 总体结构

2.1 外部端口

FIFO 是一种用来作为缓冲的存储器, 它能对数据进行快速顺序的存储和发送, 主要用来解决不同速率器件间的速率匹配问题. 图 1 是采用 FIFO 连接的数据通信系统.

图 1 中, 数据发送系统和数据接收系统采用不同的工作时钟. 数据发送系统产生数据, 数据接收系统使用数据. FIFO 模块通过不同的端口分别与数据发送系统和数据接收系统连接, 其中与数据发送系统连接的端口统称为“写端口”, 与数据接收系统

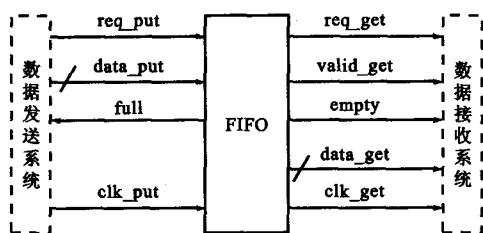


图1 采用FIFO连接的数据通信系统

连接的端口统称为“读端口”。写端口包括四组信号，分别是：

- (1) req_put 来自数据发送系统的发送请求信号；
- (2) data_put 数据发送总线；
- (3) full 该值为1时表示FIFO存储空间满；
- (4) clk_put 数据发送系统工作时钟。

读端口包括五组信号：

- (1) req_get 来自数据接收系统的数据请求信号；
- (2) valid_get 标志接收的数据是否有效；
- (3) empty 该值为1时表示FIFO存储空间空；
- (4) data_get 数据接收总线；
- (5) clk_get 数据接收系统工作时钟。

2.2 通信协议

FIFO的写端口和读端口分别遵循不同的通信协议。写端口的数据通信协议如下：

当数据发送系统有数据需要传输时，在 clk_put 时钟上升沿来临之前，数据需要放在 data_put 总线上，并给出 req_put 信号，如果在时钟上升沿来临时，full 信号为低电平，则数据被写入 FIFO。如果时钟上升沿来临时，full 为高电平，则数据发送系统需要一直等待 full 信号变为低电平，数据才能正常发送。如图2所示，在时钟沿 A，数据被正确写入 FIFO；在时钟沿 B，数据不能被写入 FIFO，需要等待 FIFO 腾空。

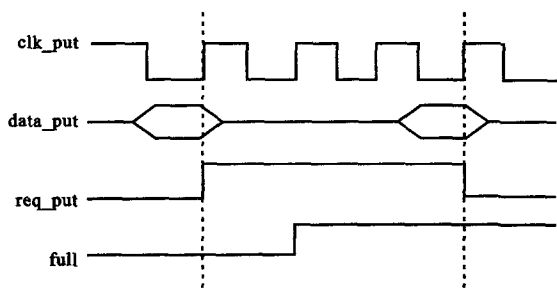


图2 FIFO写端口的时序图

读端口的数据通信协议如下：

当数据发送系统有数据需要传输时，在 clk_get 时钟的上升沿来临之前，数据接收系统首先发送 req_get 信号，如果 FIFO 中存在有效数据，那么数据将被放在 data_get 数据总线上，同时 valid_get 和 empty 信号也被置为相应的值。在 clk_get 的下一个时钟沿来临时，数据接收系统可以根据后两个信号来决定数据总线上的 data_get 数据是否有效。valid_get 表示当前读取的数据是否有效，empty 表示当前数据读取之后，FIFO 是否为空，在系统初始化时，empty 的初值置为 0。

2.3 结构设计

FIFO的总体结构如图3所示。与RAM相比，FIFO对存储阵列按顺序读写，不需要额外的地址端口。

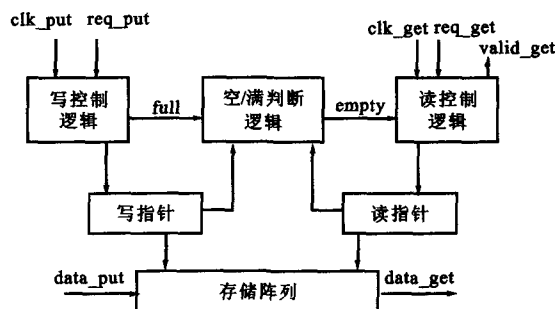


图3 FIFO的总体结构

FIFO的存储阵列类似于一片双端口的RAM，可以同时执行读操作和写操作，读写操作受控于不同的时钟。空/满状态判断逻辑根据读、写指针的变化实时检测当前FIFO的空/满状态。读控制和写控制逻辑分别根据相应的读写信号，控制存储阵列执行读操作和写操作。

3 存储阵列与读写控制逻辑

改善存储阵列与读、写控制逻辑是提高FIFO性能的有效途径。

3.1 存储阵列设计

当前广泛使用的片上存储单元有SRAM、DRAM和Flash等多种形式。由于对FIFO进行读、写操作的性能与存储阵列的性能密切相关，因此我们选择SRAM来构建存储阵列。

SRAM是静态随机存取存储器，它的每一个单元由6个尺寸不同的晶体管构成^[1]，它存储数据是依靠晶体管之间的正反馈结构。在执行读写操作时，由于它既不需要像DRAM那样的预充电—求值过程，也不需要Flash那样的编程—擦除过程，因而效率较高。

通过对晶体管尺寸的优化,读写性能可以得到较大改善.为了加快读操作的时间,SRAM 采用了灵敏放大单元,以便加快读操作的时间并提高数据的可靠性.

多个 SRAM 单元排列在一起构成了存储阵列.为了便于同时优化读、写性能,阵列的排列需要尽量接近方形.

3.2 读写控制逻辑

FIFO 的读写控制逻辑控制读写指针和存储阵列进行工作. FIFO 对存储阵列按地址依次读写.系统初始化之后, FIFO 的读写指针均指向 0 地址,每执行一次读操作或者写操作,相应的读写指针顺序加 1.当指针移到存储阵列的最后一个位置后,下一个操作会使指针返回地址 0.读写指针如此周而复始地变化, FIFO 空间得到了最有效的运用.

写指针总是指向下一个要写入的单元,每执行完一次写操作后,写指针指向下一个将要写入的存储单元.在写操作之前,需要判断写指针指向的单元是否可以写入,这需要空/满判断逻辑的信号支持.在 clk_put 时钟的上升沿,如果 req_put 信号有效,并且 full 无效,那么数据 data_put 在这一时钟沿被写入存储阵列,同时写指针加 1 指向下一个将要被写入的单元.如果 full 信号有效,那么 req_put 信号需要一直保持,直到 full 信号无效数据才能被写入.

与此类似,读指针总是指向当前被读出的单元.每执行完一次读操作后,读指针指向下一个将要读出的存储单元.在读操作之前,需要判断读指针指向的单元是否可以读出,这需要空/满判断逻辑的信号支持.在 clk_get 时钟的上升沿,如果 req_get 信号有效,并且 empty 无效,那么数据在这一时钟沿被放在 data_get 数据总线上,同时读指针加 1 指向下一个将要被写入的单元, valid_get 信号置为有效.如果 empty 信号有效,那么 valid_get 信号被置为无效, req_get 信号需要一直保持,直到 empty 信号无效数据才能被读出.

对于数据量比较大的异步通信系统来说,上述读写控制逻辑可以实现每个时钟周期写入或者读出一个数据项,大大提高了数据读写的效率.同时,时序控制逻辑简单,便于实现.我们采用 verilog 语言实现了上述读写控制逻辑,并通过逻辑综合在 FPGA 中进行了实现.

4 空/满判断逻辑分析

4.1 空/满判断逻辑

在 FIFO 进行读写操作之前,需要正确判断 FI-

FO 的存储状态,以免出现误读或者误写.正确判断 FIFO 的空/满状态是 FIFO 设计的关键点之一.空/满状态判断的原则是写满而不溢出,读空而不多读^[3],以免造成数据丢失或者误用.

如何设计 FIFO^[4] 的空/满状态判断逻辑关系 FIFO 的读写性能.当读写指针在同一时刻指向同一个内存位置时, FIFO 处于空或者满状态,否则为非空非满状态, FIFO 可以随意执行读写操作.然而,当读写指针重合时,具体是空还是满要通过逻辑进行分析.传统的判断方法是另外设置一个状态位^[5],作为最高位,其余作为地址位.当系统初始化时,状态位和地址位均置 0.当读写指针的地址位和状态位全部吻合时, FIFO 处于空状态;当读写指针的地址位相同而状态位不同时,说明写指针比读指针多循环一次, FIFO 处于满状态.这种状态判断逻辑工作频率低且占面积大,不适用于高度集成化和快速数据处理的场合^[3].

为了快速判断空/满状态,我们采用为每个存储单元附加一个锁存器来标志单元的存储状态.当某个单元被写入数据时,锁存器被置 1 表示单元处于满状态.当某单元的数据被读出时,锁存器的状态被置 0,表示单元处于空状态.假设第 i 个单元的满状态用 f_i 表示,空状态用 e_i 表示.对于含有 N 个存储单元的 FIFO 来说,可以分别采用式(1)和式(2)来进行空/满状态判断.公式中的 AND 和 OR 分别表示“逻辑与”操作和“逻辑或”操作.

$$\text{full} = \text{AND}_{i=1}^N(f_i) \quad (1)$$

$$\text{empty} = \text{OR}_{i=1}^N(e_i) \quad (2)$$

以上判断逻辑如果用组合逻辑实现,虽然运行效率较高,但是逻辑规模大,需要占用较大的芯片面积.为了克服这个缺点,采用动态逻辑来实现状态判断,从而在减小芯片面积的同时获得更高的运行效率.图 4 是采用动态逻辑来判断含有 4 个存储单元的 FIFO 的空/满状态判断的电路图.从图中可以看出,空/满判断逻辑只需要较少的芯片资源.对于含有更多存储单元的 FIFO 来说,判断逻辑可以进行相似的扩充.

4.2 亚稳态的避免

异步电路设计需要妥善处理亚稳态的问题.当异步信号在不同的时钟域中传输时,由于工作时钟不同,如果某信号在变化的过程中恰好被采样,那么采样信号的状态变得不可确定,这种现象称为亚稳态^[1].

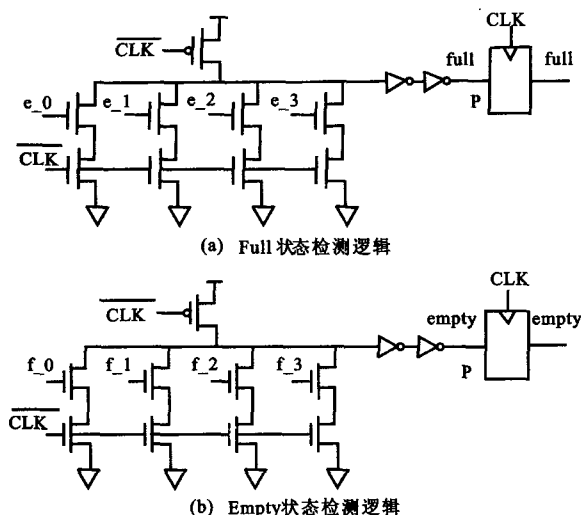


图4 采用动态逻辑的空/满状态检测电路

在异步 FIFO 的设计中,发送端口和接收端口采用不同的工作时钟.而空/满判断逻辑与两个时钟均密切相关,因而可能产生亚稳态的问题.一旦产生亚稳态,说明信号电平不稳定.如果这种状况持续的时间大于一个时钟周期,不稳定的数据状态就会造成下一级的触发器错误接收或者发送电平信号,从而影响整个 FIFO 的数据传输.在异步 FIFO 数据传输过程中,由于发送端和接收端的时钟不同,建立时间和保持时间的冲突不可避免,因而一定会产生亚稳态的问题.

解决亚稳态的问题需要从分析时序电路的特性入手.对于时序器件来说,信号相对于时钟满足建立时间和保持时间是保证信号被可靠采样的前提.根据亚稳态产生的条件,可以对不同时钟域输入的信号采用双所存器连续锁存两次.如果第一个锁存器锁存到一个不稳定的电平信号,在经过一个时钟后可能趋于稳定,从而得到一个稳态的信号值.虽然亚稳态不可避免,但采用这种双所存结构之后,电路出现亚稳态的概率可以降到非常低的程度.

5 实验验证

采用 FPGA 对文中设计的 FIFO 的控制逻辑和状态检测逻辑进行了验证.数字逻辑部分采用了

Verilog 硬件描述语言进行实现.验证实验在 Falcon-E25TG 开发平台上进行,使用的 FPGA 为 Xilinx 的 SPARTAN 3E.设计和仿真环境采用 Xilinx 的 ISE 9.1i. FIFO 的存储体部分例化了 FPGA 提供的块 RAM. FIFO 的数据宽度为 8 位,深度为 4096.通过实验,文中设计的 FIFO 单元的读写性能可以达到 400MHz 左右,比 ISE 工具自动生成的 FIFO 单元的工作频率提高了约 100MHz, FIFO 性能提高幅度达到 30% 以上.与 ISE 自动生成的 FIFO 相比,本文中 FIFO 所占用的 FPGA 资源有所增加,约增加 15% 左右.在高性能设计中,在增加少许资源的情况下获得显著的性能提升是值得的.

6 结束语

文中讨论了高速异步 FIFO 电路的设计方法,通过精心设计 FIFO 体系结构、读写电路和状态检测电路,可以实现 FIFO 工作性能的大幅提升.文中的设计用于 FPGA 设计,可以有效解决多时钟域之间的高速数据通信问题.

参考文献:

- [1] Rabaey J M, Chandrakasan A, Nikolic B. 数字集成电路——电路、系统与设计[M]. 2 版. 周润德,译. 北京:清华大学出版社, 2004.
- [2] 刘洪波, 龙娟, 郝晓莉,等. 异步状态判断的研究与设计[J]. 微电子学与计算机, 2007, 24(3): 81-84.
- [3] 徐海铭, 程月东. 高速异步 FIFO 设计[J]. 电子与封装, 2009(6): 14-16.
- [4] Chelcea T, Nowick S M. A low-latency FIFO for mixed-clock systems[C]// Proceedings of the IEEE Computer Society Annual Workshop on VLSI (WVLSI'00). Orlando, 2000: 119.
- [5] 于海, 樊晓桢. 基于 FPGA 异步 FIFO 的研究与实现[J]. 微电子学与计算机, 2007, 24(3): 210-213.

作者简介:

李冬女, (1989-), 研究方向为电子电路.

赵志凯 男, (1983-), 博士研究生. 研究方向为机器学习、模式识别.