FSL 总线 IP 核及其在 MicoBlaze 系统中的应用

作者：时间：2008-06-02 来源：单片机及嵌入式系统应用

引 言

本文引用地址：http://www.eepw.com.cn/article/83481.htm

随着半导体制造工艺的发展，以 *FPGA*(现场可编程门阵列)为代表的新一代可编程逻辑器件(PLD)的逻辑资源密度不断增加，使得可编程技术很自然地就与系统芯片集成技术(SoC)的结合日益紧密，并逐步成为可配置平台技术(configurable platform)的主流。

目前，各主要 PLD 厂商基于 *FPGA* 的可配置平台虽然大都采用"微处理器十可编程逻辑"的架构，但在开发基于 *FPGA* 的嵌入式系统时，却采用了各自不同的方式来整合处理器系统与片上的其他逻辑资源(大多数以用户 *IP 核*形式出现)。MicroBlaze 软核处理器是 Xilinx 公司为其 FPGA 器件开发的，其特有的 *FSL*(Fast Simplex Link，快速单向链路)总线，可以实现用户 *IP 核*与软核处理器的高速连接，为设计者提供了一条解决这类问题的途径。

1 MicroBlaze 软核处理器

1.1 概 述

MicroBlaze 是基于 Xilinx 公司 FPGA 的微处理器软 *IP 核*。它采用 *RISC* 架构和哈佛结构的 32 位指令和数据总线，内部有 32 个 32 位宽度的通用寄存器；在 150 MHz 的时钟频率下，最高可达到 125 DMIPS 的处理性能，其逻辑结构如图 1 所示(图中省略了指令侧的同类接口)。使用 Xilinx 公司提供的 EDK(嵌入式系统开发套件)，可以在参数化的图形界面下方便地完成嵌入式软处理器系统的设计。其突出的优点，一是设计灵活性；二是可以整合用户自定义 IP 核，使得算法可以在硬件中并行地执行而不是在软件中串行执行，从而极大地加速软件的执行速度，即所谓的硬件加速。

1.2 MicroBlaze 软核总线接口

MicroBlaze 软处理器核具有丰富的接口资源。目前，最新版本的 MicroBlaze 软核支持的接口标准有：

◆带字节允许的 *OPB*(On-chip Peripheral Bus，片上外设总线)V2.0 接口；

◆高速的 *LMB*(Local Memory Bus，本地存储器总线)接口；

◆*FSL* 主从设备接口；

◆XCL(Xilinx Cache Link，Xilinx 缓存链路)接口；

◆与 MDM(微处理器调试模块)连接的调试接口。

OPB 是对 IBM Core Connect 片上总线标准的部分实现，适用于将 IP 核作为外设连接到 MicroBlaze 系统中。*LMB* 用于实现对片上的 blockRAM 的高速访问。 *FSL* 是 MicroBlaze 软核特有的一个基于 FIFO 的单向链路，可以实现用户自定义 IP 核与 MicroBlaze 内部通用寄存器的直接相连；而 XCL 则是 MicroBlaze 软核新增加的，用于实现对片外存储器的高速访问。MicroBlaze 软核还有专门的调试接口，通过参数设置，开发人员可以只使用特定应用所需要的处理器特性。

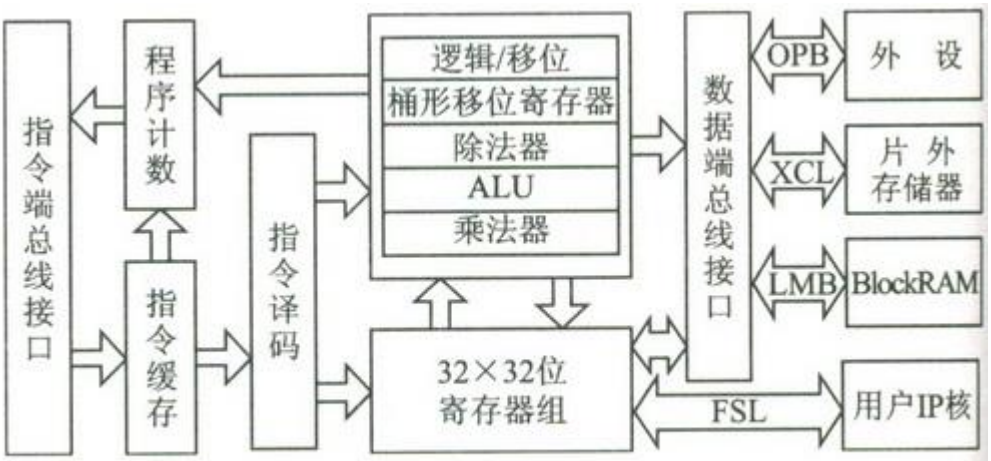

图 1 MicroBlaze 软核

1.3 MicroBIaze 系统的硬件加速

将用户 IP 核整合到基于 MicroBlaze 的嵌入式软核处理器系统中，通常有两种方法：一种方法是将 IP 核连接到 *OPB* 总线；第二种方法就是将用户 IP 连接到 MicroBlaze 专用的 FSL 总线上。OPB 与 FSL 比较如表 1 所列。

表 1　OPB 与 FSL 比较

比较内容	OPB	FSL
结构特点	共享式 支持多主多从模式 分离的指令与数据通道	非共享式 点对点模式 直接与通用寄存器相连
实现方式	分布式的与或逻辑 独立的仲裁器	基于 FIFO
数据宽度①	8,16,32,64,128	8,16,32
数据流向	多对多	单向(主→从)
最大工作频率/MHz	239②	600
最小占用资源(slices)	46	21
可扩展性	1～16 主设备 从设备数仅受资源限制	8 对 FSL 主从接口③
MicroBlaze 指令支持	无	get、put 等 FSL 读写指令

注：①数据分别来自 opb_v20 和 FSL_v20 数据手册；

②该数据是在总线配置为 1 主 2 从情况下得到的；

③该数据对 MicroBlaze 软核而言。

表 2　FSL 接口的 I/O 信号

信号名称	I/O	功能说明
FSL_M_Clk	I	FSL 总线主设备接口时钟
FSL_M_Data	I	FSL 总线主设备接口的输入数据
FSL_M_Control	I	在每个时钟沿与数据一起发送的单位控制信号
FSL_M_Write	I	控制 FSL 主设备接口的写允许信号
FSL_M_Full	O	表示 FIFO 已满的 FSL 主设备接口输出信号
FSL_S_Clk	I	FSL 总线从设备接口时钟
FSL_S_Data	O	FSL 总线从设备接口的输出数据
FSL_S_Control	O	在每个时钟沿与数据一起接收的单位控制信号
FSL_S_Read	I	控制 FSL 从设备接口的读确认信号
FSL_S_Exists	O	表示 FIFO 已有数据的 FSL 从设备接口输出信号

从表 1 可见，尽管 OPB 和 FSL 都是 MicroBlaze 软核与 FPGA 其他片上逻辑资源连接的主要途径，但其特点决定了分工是不同的：OPB 总线适用于将低速和低性能要

求的设备连接到 MicroBlaze 系统中；而 FSL 总线则适用于将时间要求高的用户自定义 IP 核整合到基于 MicroBlaze 的软核系统中，以实现硬件加速。

2 FSL 总线

2.1 FSL 总线接口

FSL 总线是一个基于 FIFO 的单向点对点通信总线，主要用于 FPGA 的两个模块问进行快速的通信。FSL 总线 IP 核结构如图 2 所示，FSL 接口的 I/O 信号如表 2 所列。

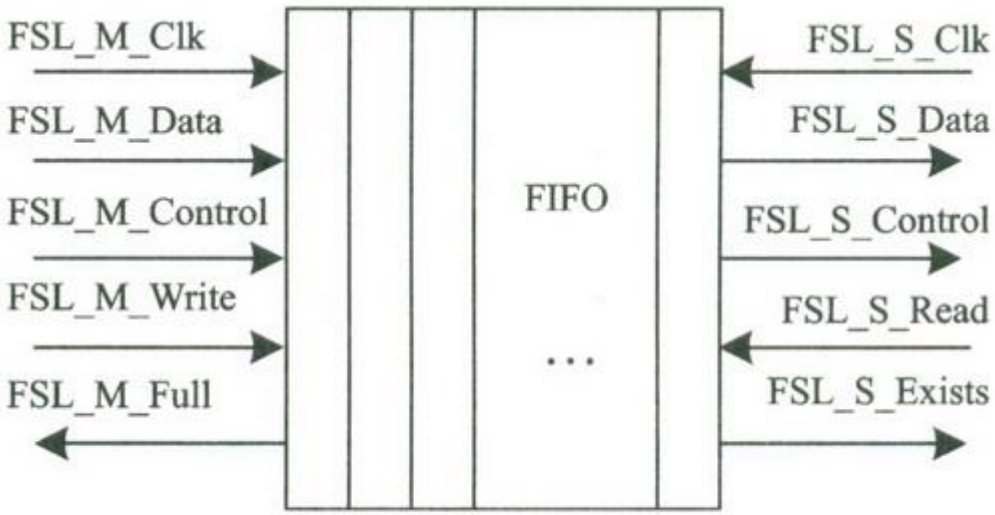

图 2　FSL 总线 IP 核

该接口的主要特点：

◆单向的点对点通信；
◆非共享的无仲裁通信机制；
◆支持控制位与数据分离的通信；
◆基于 FIFO 的通信模式；
◆可配置的数据宽度；
◆高速的通信性能(独立运行达到 600 MHz)。

2.2 *FSL* 总线的写操作时序

对 *FSL* 总线的写操作是由 *FSL*_M_Write 信号控制 的。图 3 是 FSL 总线的写操作时序。FSL 主设备在第一个时钟上升沿检查到 FSL_M_Full 信号未置高，就允许主设备将 FSL_M_Write 置高，并将 FSL_M_Data 和 FSL_M_Control 推上总线，在下一个时钟周期这些数据就被总线读取并送入 FIFO 了。图中的 Write2 和 Write3 是一组"背靠背"的连续写操作。在 Write3 时，FIFO 满使得 FSL_M_Full 信号被置高，迫

使主设备取消自己的 FSL_M_Write 信号，直到一次读操作将 FSL_M_Full 置低后，才可以发起另一次写操作。因此，图中暗示着在 Write4 处也发生了一次从设备的读操作，否则 FSL_M_Full 将再次置高。

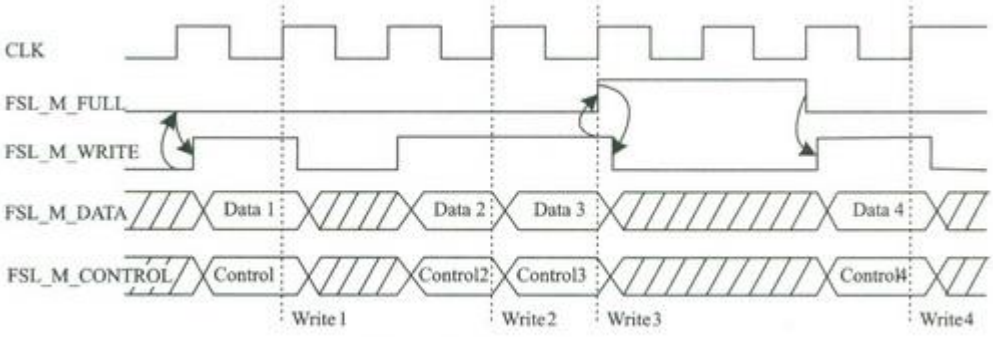

图 3　FSL 总线写操作时序

2.3 FSL 总线读操作时序

对 FSL 总线的读操作是由 FSL_S_Read 信号控制的，图 4 是 FSL 从设备的 3 次读操作时序。当 FSL 总线上存在有效数据(FSL_S_Exists='1')，FSL_M_Data 上的数据和 FSL_M_Control 上的控制位就立即可以被 FSL 从设备读取。一旦从设备完成读操作，FSL_S_Read 信号必须置高一个时钟周期，以确认从设备成功完成了一次读操作。在读操作发生后的时钟上升沿(图中 Read2 处)，FSL_M_Data 和 FSL_M_Control 会被更新为新数据，同时 FSL_S_Exists 和 FSL_M_Full 信号也会被更新。同样，这里暗示着在 Readl 和 Read2 之间发生了两次主设备的写操作。

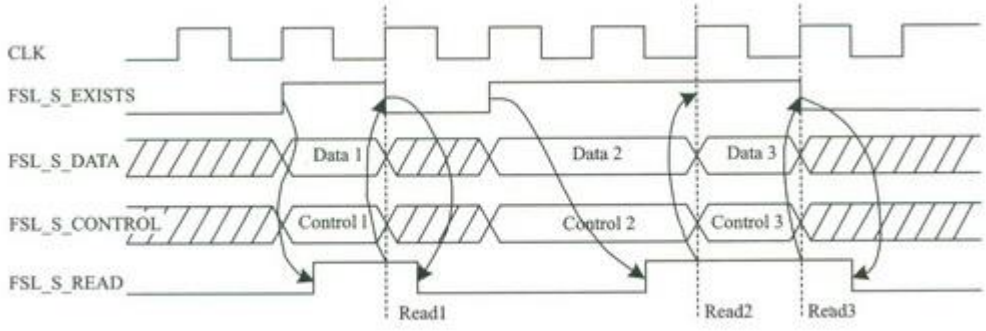

图 4　FSL 总线读操作时序

3 FSL 总线用法

3.1 使用 FSL 总线 *IP 核*的设备间通信

目前 Xilinx 提供的 FSL 总线 *IP 核*的版本是 FSL_V20。两个设备要使用 FSL 进行数据传输，就必须分别作为主设备或从设备连接到 FSL 核上。如果需要进行双向的传输，只要两个设备分别作为主从设备，使用两个 FSL 核连接即可。

无论是作为主设备或是从设备，都需要通过在设备的微处理器外设描述文件(MPD)中进行相应的定义，以实现所需类型的 FSL 接口。下面这段代码就是一个分别定义了 FSL 主设备接口 FSL_OUT 和从设备接口 FSL_IN 的 MPD 文件：

```
BEGIN my_fsl_peripheral
OPTI0N IPTYPE="PERIPHERAL"
OPTI0N IMP_NETLIST=TRUE
BUS_INTERFACE BUS="FSL"_IN，BUS_STD=FSL，BUS TYPE="SLAVE"
BUS_INTERFACE BUS="FSL"_OUT，BUS_STD=FSL，BUS_TYPE=MASTER
##Ports
PORT CLK=””，DIR=IN，SIGIS=CLK
PORT RESET=””。DIR=IN
PORT FSL_S_READ=FSL_S_Read，DIR=out，BUS=FSL_IN
PORT FSL_S_DATA=FSL_S_Data，DIR=in，VEC=[o:31]，BUS=FSL_IN
PORT FSL_S_CONTROL=FSL_S_Control，DIR=in， BUS="FSL"_IN
PORT FSL_s_EXISTS=FSL_S_Exists，DIR=in，BUS=FSL_IN
PORT FSL_M_WRITE=FSL_M_Write，DIR=out，BUS=FSL_OUT
PORT FSL_M_DATA=FSL_M_Data，DIR=out，VEC=[o:31]，
BUS=FSL_OUT
PORT FSL_M_CONTROL=FSL_M_Control，DIR=out， BUS="FSL"_OUT
PORT FSL_M_FULL=FSL_M_Full，DIR=in，BUS=FSL_OUT
```

3.2 通过 FSL 与 MicroBlaze 通信

MicroBlaze 软核的 FSL 总线接口支持最多 8 对 FSL 连接，具体实现多少接口由系统硬件描述文件(MHS)中的参数 C_FSL_LINKS 决定。默认情况下该参数为 0，表示不实现 FSL 接口。当需要使用 FSL 总线把 MicroBlaze 和 *FPGA* 中的一个或多个逻辑模块连接起来时，必须设置该参数的值为相应的模块数。该参数的取值范围是 0～8。

在 MicroBlaze 指令集中还有针对 FSL 总线操作的指令，它们分别是：

◆get，put——阻塞式数据读写 FSL，控制信号被置为 0；
◆nget，nput——非阻塞式数据读写 FSL，控制信号被置为 0；
◆cget，cput——阻塞式控制位读写 FSL，控制信号被置为 1；
◆ncget，ncput——非阻塞式控制位读写 FSL， 控制信号被置为 1。

4 *FSL* 总线应用实例

在下面的实例中，尝试通过 *FSL* 总线技术，将实现特定函数功能的用户自定义 *IP 核*整合到 MicroBlaze 软核系统中，以实现硬件加速的目的。这里以一个矢量汉字(vector font)还原功能的硬件模块的整合为例，说明 *FSL* 总线的应用过程。所使用的开发平台是 Memec Insight 公司生产的 Virtex—II 系列的 MicroBlaze 开发板，板上采用的 *FPGA* 器件为 Virtex—II 1000，系统时钟为 100 MHz，开发工具为 Xilinx 公司的 EDK 6.3 及 ISE 6.3。

4.1 FSL 总线应用方案

如图 5 所示，vectOr_font 核通过 FSL_Code-与 FSL_Lattice 两条 FSL 总线与 MicroBlaze 软核直接相连。

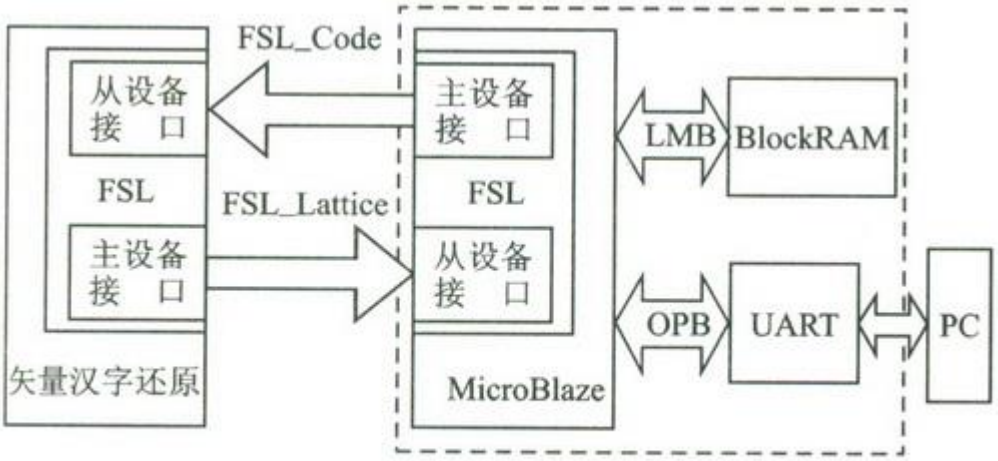

图 5 FSL 总线应用

对于 FSL_Code 总线，MicroBlaze 核是主设备，而 vector_font 核是从设备。这样 MicroBlaze 可以通过 FSL_Code 总线向 vectOr_font 核发送汉字的区位码(或者其他格式的汉字编码，由使用的矢量字库和还原算法决定)以及汉字的属性信息(如字体、大小等)。

对于 FSL_Lattice 总线则正好相反。vector font 核作为主设备可以通过它向 MicroBlaze 核发送经过还原处理后的汉字点阵数据以及汉字点阵尺寸信息(用于将点阵数据在显存中组织成正确的显示格式)。

4.2 数据传输指令与控制位指令的应用

FSL 提供的独立于数据传输的控制位可以用来对正在传输中的数据进行标记。为

了区分区位码数据与汉字属性数据，以及点阵数据与汉字点阵尺寸数据。 MicroBlaze 分别通过 FSL 的数据传输指令和控制位传输指令来发送汉字的区位码和汉字属性信息，接收点阵数据和汉字点阵尺寸信息。对应的实现代码如下：

//使用非阻塞的数据写函数向 FSL 总线写入汉字区位码
Microblaze_nbwrite_datatsl(code,O)
//使用非阻塞的控制位写函数向 FSL 总线写入汉字属性信息
maicroblaze_cnbwrite_cnlfsl(attibute，O)
//使用非阻塞的数据读函数从 FSL 总线读取汉字点阵数据
microblaze_nbread_datafsl(1attice[i]，O)
//使用非阻塞的控制位读函数从 FSL 总线读取汉字点阵尺寸信息
microblaze_cnbread_cnlfsl(size，O)

代码中用到的与 FSL 有关函数的定义，都在 include 目录下的 mb_inteRFace. h 文件中。其中，各函数的第二个参数代表进行读写操作的 FSL 总线接口的编号，对应 Mi—croBlaze 软核的 8 对 FSL 接口。该参数的取值范围从 0 到 7。本例中,MicroBlaze 只使用了一对 FSL 接口，故而值为 0。

4.3 实现步骤

首先，在 Base System Wizard 中设计图 5 虚线框中所示的一个简单的 MicroBlaze 嵌入式处理器系统。然后，在 XPS 集成开发环境下完成用户自定义 *IP 核*(本例中即 vectoz__font 核)的添加、Microblaze 核 FSL 接口的添加(设置参数 C_FSL_LINKS=1)，同时添加两个 FSL 总线 *IP 核*，分别用于实现 FSL_Code 和 FSL_Lattice 总线。另外，将两个 FSL 总线 IP 核的参数 C_USE_CONTROL 置为 1，以打开 FSL 总线的控制位传输功能。所有这些改动，最后都会被更新到 MES 文件中。这样，硬件平台生成工具 platgen 就可以根据它生成所需要的 *FPGA* 配置文件了。

硬件的实现完成后，进行相应软件参数的设置，如将系统标准输入输出设备指向 UART 模块等。然后，用库生成工具 libgen，根据 MSS(系统软件描述文件)文件，将所需外设函数库的头文件添加进工程中。

通过调用这些函数，可以操作和控制这些外设。通过 Tool 项里的 build 命令，调用 mb—gcc：编译工具，将编写的应用程序编译成 ELF 文件，再用 updatebitstrcam 命令将程序代码对应的 RAM 初始化数据添加到前面生成的 *FPGA* 配置文件中，生成最终的 bit 配置文件。最后，使用 download 命令将 bit 文件下载到目标板中。

以上就是整个 FSL 应用实例设计的实现过程。本例只是为了说明 FSL 总线的使用。实际应用中，还可以根据具体情况通过 FSL，将更多的用户自定义 IP 核(如 DCT、FFT 等)添加到．MicroBlaze 软核系统中去。

结 语

在嵌入式系统的开发中，人们一直希望能够有一个满足自己需要的"定制"的嵌入式处理器，而不是手头大量存在的通用微处理器。但是，直接将用户自定义 IP 核添加到处理器核中，不仅受到处理器原架构的束缚，还有可能降低处理器的性能(处理器工作频率)；而通过与内部寄存器直接相连的 FSL 接口，用户自定义 IP 可以在不破坏处理器原有结构的情况下，紧密地与 MicroB- laze 软核结合在一起。这样，即使关键路径覆盖了用户 IP 核，由于它在处理器内核之外，也不会导致处理器时钟频率的降低。

通过对 FSL 总线的分析以及上述实例的验证，证明了在基于 MicroBlaze 的 SoC 系统设计中，一方面可以针对具体应用进行"量体裁衣"式的设计；另一方面，利用其专用的 FSL 总线接口技术，实现嵌入式软处理器系统与用户自定义逻辑的整合，从而在不提高系统主频的前提下，通过部分函数功能的硬件实现来提升系统的性能。