# 跳上Avalon总线：一种简化的FPGA接口

## Hop on the Avalon bus: a simplified FPGA interface

▶ Noe Quintero 凌力尔特公司

### 引言

许多新式FPGA设计采用了一些用于控制的嵌入式处理器。一种典型解决方案需要使用诸如NIOS等嵌入式软处理器。另一种解决方案则使用包含一个内置硬处理器的SoC（片上系统）器件。图1所示为一个典型的Altera FPGA系统，该系统包含处理器和一系列通过Avalon内存映射（MM）总线连接的外设。这些处理器极大地简化了最终应用，但是要求开发人员拥有坚实的编程背景和精细复杂工具链的相关知识。这会阻碍调试工作的推进，特别是如果硬件工程师需要一种不会烦扰软件工程师即可完成外设读写的简单方法。

### 1 SPI-Avalon MM桥接器

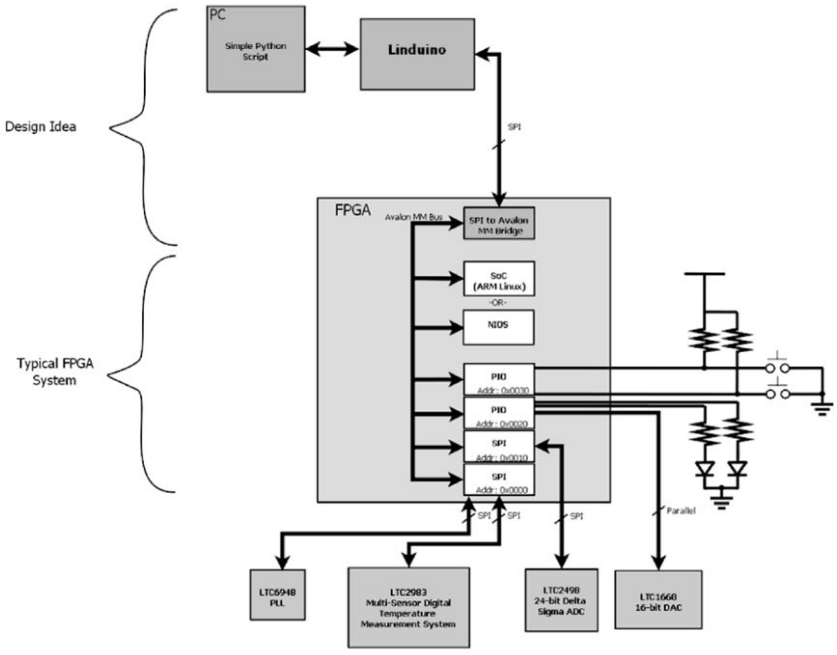


图1 通过Avalon内存映射（MM）总线连接的典型Altera FPGA系统

该设计思想运用了Altera（2015年被英特尔收购，成为其下的可编程解决方案事业部）的SPI从端至Avalon MM桥接器，以提供一种跳上Avalon总线的简单方法。采用这种方法有两项优势：它并未损害原始系统设计，而且该桥接器能够与嵌入式处理器共存。对于图1中所示的系统，SPI-Avalon MM桥接器将允许设计师直接控制LTC6948分数N PLL的频率，设定LTC1668 DAC电压，从LTC2498读取一个电压，或者从LTC2983读取温度，就像处理器一样。

Altera 提供了一款针对SPI-Avalon MM桥接器的参考设计。不幸的是，文档较为稀少，并且使用一个NIOS处理器作为SPI主控器。这实际上违背了SPI桥接器的初衷，因为NIOS处理器可直接连接至Avalon MM总线。一款实用的SPI主控器是凌力尔特的Linduino®微控制器，它是具有附加特性的Arduino克隆产品，以与LT演示板相连接。附加特性之一是电平移位SPI端口。当连接至具有低至1.2V电压的FPGA I/O块时，这种电平移位功能是特别有帮助的。Linduino固件可用于通过一个虚拟COM端口接受命令并把命令转化为SPI事务处理。

在对Altera实例设计实施了反向工程之后（图2的左侧），开发一个Python库以生

图2 荧光笔+示例代码+反向工程=Python脚本

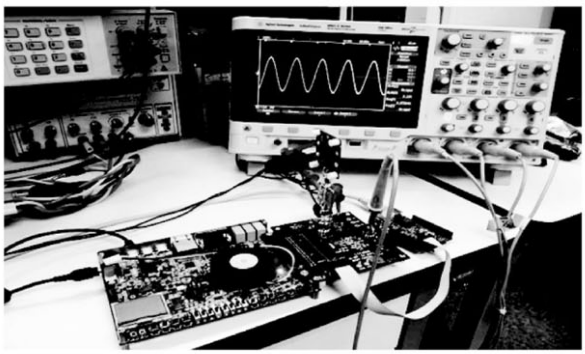

图3 DC2459在工作中

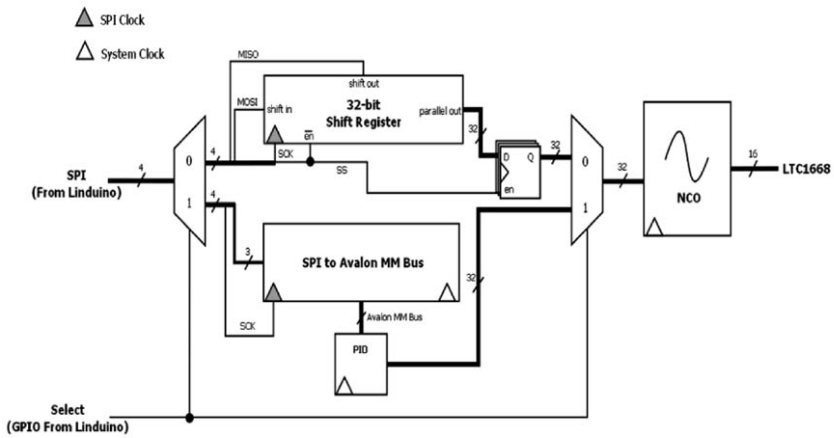


图4 DC2459A FPGA系统方框图

图5 Qsys GUI

成桥接器将要接受的数据包。这些数据包随后被转化为Linduino命令。这样，一个简单的Python脚本使得硬件工程师能够全面地控制项目，并不需要彻底改变接口协议。在LinearLabTools Python文件夹中提供了一个用于控制LTC1668 DAC的数字图形发生器频率的Python脚本实例。图3所示为演示设置。

图4给出了FPGA的系统方框图。数控振荡器（DCO）可由移位寄存器或PIO内核来控制。内置移位寄存器用于调试，因为它提供了NCO的直接控制。把GPIO线逻辑电平设定为“高”将使能SPI-Avalon MM桥接器，该桥接器接着通过Avalon MM总线控制一个32位PIO端口。然后，PIO输出控制NCO频率。

## 2 系统集成工具Qsys

当最基本的系统运行时，可以把额外的Avalon外设IP内核连接至Avalon MM总线。为了设计系统，Altera提供了一款被称为Qsys的系统集成工具。这款工具提供一个GUI以相互连接IP。Qsys随后用于把GUI系统转化为硬件描述语言（HDL）Verilog。图5所示为GUI。最后，系统将被添加至用于实施的顶层。IP的地址是完全可配置的。就给出的实例而言，PIO被设定在一个0x0的基地址单元。

一旦在FPGA中实现了设计，则LinearLabTools中提供的Python库包含两个函数以与设计接口：

transaction_write(dc2026, base, write_size, data)

transaction_read(dc2026, base, read_size)

这些函数的第一个参数是Linduino串行端口实例，第二个参数是外设在Avalon总线上的地址。这些函数分别接受和返还字节列表。编写这两个函数以在读和写IP时提供灵活性。如欲设定用于所提供实例的NCO，则所需的就是transaction_write函数。式（1）用于确定频率控制字。

频率控制字 = (期望的频率/系统时钟频率)× $2^{32}$ （1）

如要把NCO设定至 ▶▶下转第120页

◀◀上接第115页

1kHz和一个50Msps采样速率，则频率控制字数值设为85899。该数值用十六进制来表示即为 0x00014F8B，其作为一个4字节列表进行传递。于是，用于把DAC设定至1kHz的Python代码为：

transaction_write(linduino_serial_instance, 0, 0, [0x0,0x01,0x4F, 0x8B])

注：根据逻辑设计，PIO 的基地址为零。

## 3 Python Avalon总线示例

本文提供一个如图6所示的简单Python脚本，以演示FPGA设计和Python脚本的接口。它包含一个简单的文本接口以配置NCO。一个重要提示是Avalon SPI桥接器采用SPI Mode 3。这是通过反复试验而确定的正确模式；并通过分析Altera实例中的NIOS处理器SPI接口进行验证。

## 4 结论

该实例项目展示了完全无需“接触”嵌入式处理器便可控制系统的能力。这让硬件工程师不必麻烦软件工程师就能在项目方面取得进展。这种方法的好处是可以悄然地添加至FPGA，并不会影响原始设计，从而使硬件工程师可以把精力集中在硬件上。

```
"""
    The views and conclusions contained in the software and documentation are
    those of the authors and should not be interpreted as representing official
    policies, either expressed or implied, of Linear Technology Corp.

    Description:
        The purpose of this module is to use the DC2026C as a Avalon MM Bus
        interface to set the DC2463A frequency out.
"""

########################################################################
# LIBRARIES
########################################################################

import sys
sys.path.append('../../utils')
import connect_to_linduino as duino
import ltc_spi_avalon as avalon


if __name__ == "__main__":
    linduino = duino.Linduino() # Find the Linduino
    linduino.port.write("S3")   # Set to SPI Mode 3
    linduino.port.write("G")    # Set the GPIO HIGH
    try:
        print "Command Summary"
        print "    1-Send raw code"
        print "    2-Set frequency"
        print "    3-Exit program"
        user_input = input("Enter a command: ")
        while(user_input != 3):
            if(user_input == 1):
                code = int(input( "Enter raw 32 bit code: "))

                # Send the data to the Avalon MM bus addr 0
                avalon.transaction_write(linduino, 0, 4,
                                         [code & 0xFF ,(code>>8) & 0xFF ,
                                          (code>>16) & 0xFF, (code>>24) & 0xFF])
            elif (user_input == 2):
                freq = float(input("Enter desired frequency(Hz): "))
                float_code = freq/50000000*(2**32-1)
                code = int(float_code)

                # Send the data to the Avalon MM bus addr 0
                avalon.transaction_write(linduino, 0, 4,
                                         [code & 0xFF ,(code>>8) & 0xFF ,
                                          (code>>16) & 0xFF, (code>>24) & 0xFF])
            else:
                print "**** Invalid Command ****"
            print "Command Summary"
            print "    1-Send raw code"
            print "    2-Set frequency"
            print "    3-Exit program"
            user_input = input("Enter a command: ")
    finally:
        linduino.close() # Close the port
```

```
Looking for COM ports ...
Available ports: [(23, 'COM24')]

Looking for Linduino ...
    Found Linduino!!!!
Command Summary
    1-Send raw code
    2-Set frequency
    3-Exit program
Enter a command:
```

图6 Python Avalon总线示例