|  |  |
| --- | --- |
| 学 号： | 0122009361520 |



**第六届**

**全国大学生集成电路创新创业大赛**

**校内选拔赛**

**题** **目** 题目一 基于开源软核处理器的SoC片上系统的构建

**学** **院** 信息工程学院

**专** **业** 通信工程

**班** **级** 通信zy2001

**姓** **名** 杨子睿

**联系方式** 13367910668

2022 年 2 月 26 日

目 录

[1 设计简介 1](#_Toc128495268)

[1.1 总体介绍 1](#_Toc128495269)

[1.2 系统流程框图 1](#_Toc128495270)

[2 硬件设计方案 2](#_Toc128495271)

[2.1 系统整体硬件架构 2](#_Toc128495272)

[2.2 部署处理器逻辑内核及基本模块 3](#_Toc128495273)

[2.3 总线矩阵 3](#_Toc128495274)

[2.4 片上外设的创建 5](#_Toc128495275)

[2.4.1 AHB片上存储器外设的实现 5](#_Toc128495276)

[2.4.2 GPIO外设的实现 5](#_Toc128495277)

[2.4.3 UART外设的实现 6](#_Toc128495278)

[3 软件设计方案 7](#_Toc128495279)

[3.1 总线的地址映射 7](#_Toc128495280)

[3.2 GPIO的软件封装 7](#_Toc128495281)

[3.3 UART的软件封装 8](#_Toc128495282)

[3.4 软件模拟SPI协议的实现 8](#_Toc128495283)

[4 系统布线和资源消耗 10](#_Toc128495284)

[4.1 系统的RTL视图 10](#_Toc128495285)

[4.2 系统的资源消耗 10](#_Toc128495286)

[5 仿真与上板测试 11](#_Toc128495287)

[5.1 总线仿真 11](#_Toc128495288)

[5.2 逻辑分析仪对SPI通信的信号进行捕获 12](#_Toc128495289)

[6 参考文献 13](#_Toc128495290)

[7 附录 13](#_Toc128495291)

# 1 设计简介

## 1.1 总体介绍

本系统是基于ARM Cortex-M3 Design Start的SoC片上系统，利用ARM Cortex-M3处理器内核及ARM DesignStart M3 (Eval)的cmsdk模块独立设计的SoC架构。由FPGA板卡的硬件资源搭建而成。

搭载本系统的硬件平台为ZYNQ XA7Z020，其具备逻辑资源丰富、价格低的优点。该系统的一级AHB总线上预留的主机接口，二级AHB和APB总线上预留的外设接口，可供后续的开发丰富系统功能和软件应用。

## 1.2 系统流程框图

本系统主要部分包括，由一级的AHB总线桥接的Cortex-M3处理器内核，FPGA的板上资源RAM接口以及二级总线接口，并在二级总线上挂载了串口和GPIO基础外设。使用GPIO外设成功通过软件模拟了SPI协议的主机通信。

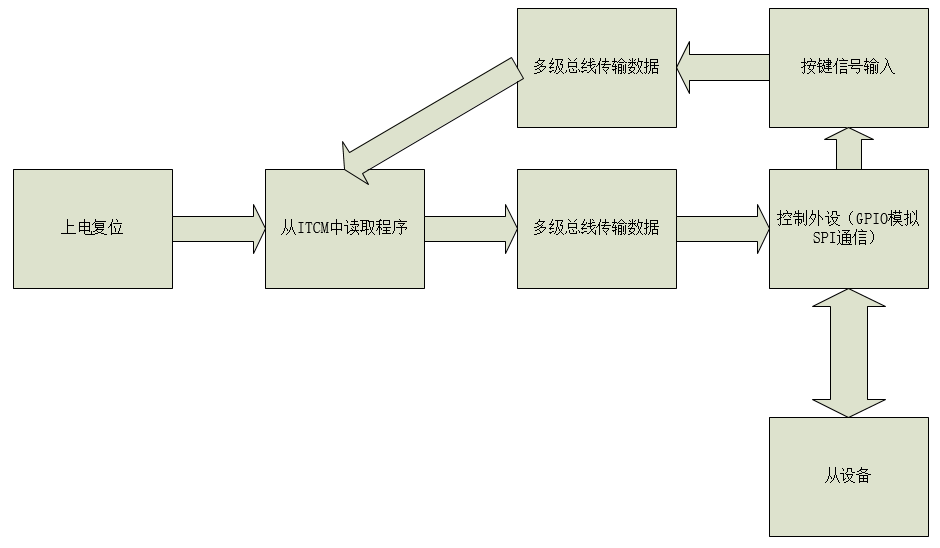


图1 系统流程框图

# 2 硬件设计方案

## 2.1 系统整体硬件架构

本系统基于FPGA片上硬件资源设计，硬件主要模块包括：

1. ARM Cortex-M3处理器；
2. 一、二级AHB和二级APB系统总线；
3. 总线间接口和总线外设间接口；
4. AHB和APB外设：
5. 由FPGA片上BRAM实现的程序和数据存储器；
6. 片上功能外设，目前仅包括UART和GPIO，后续可扩展；
7. 系统时钟；
8. SWD调试接口。



图2 系统总体硬件架构

## 2.2 部署处理器逻辑内核及基本模块

ARM官网获取的ARM DesignStart M3(Eval)包中含有处理器内核cortexm3ds\_logic.v，以及其中cmsdk包中众多的软核基本模块组件，如AHB、APB总线模块，总线接口模块，串口模块等等。通过将处理器内核和其他基本模块相互连接构成具有正常功能的软核，在顶层文件中对这些模块进行例化并连接对应的信号。处理器内核的RTL视图如下图3所示。

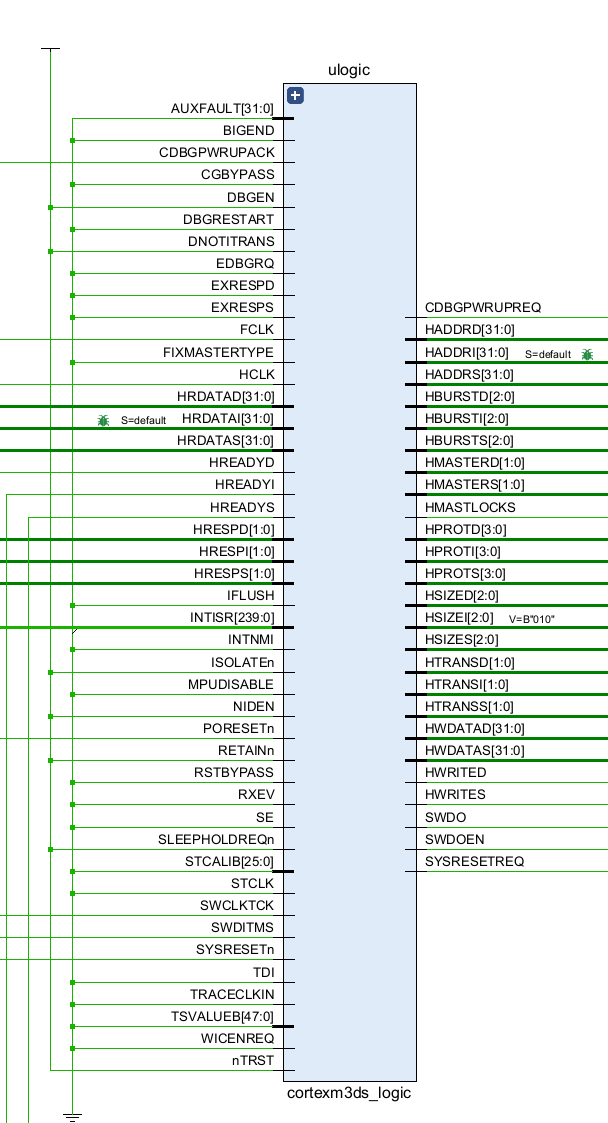


图3 处理器内核RTL视图

## 2.3 总线矩阵

本系统使用ARM公司的工具——Cortex -M System Design Kit（CMSDK）生成AHB总线矩阵（AHB Bus Matrix），利用总线矩阵模块作为系统总线与ITCM、DTCM及低速外设的互联模块，同时配置好各个部件的地址映射。

一级AHB互联总线矩阵有5个slave端和5个master端，5个slave端分别为和ARM处理器内核连接的控制总线I-Code Bus、数据总线D-Code Bus和系统总线System Bus以及两个预留主机接口，可用于扩展DMA或者硬件加速器。5个master模块分别为程序存储器ITCM、数据存储器DTCM、AHB到APB的总线桥、AHB到AHB的同步总线桥以及一个预留的一个从机接口。具体结构如图3所示。Slave端与master在总线矩阵内部进行互联与仲裁，从而使M3内核能够通过预设的地址进行访问和控制系统的各个部分。



图4 一级AHB总线的互联结构

二级AHB总线矩阵通过AHB同步桥和一级AHB总线矩阵相连，有2个master端，1个slave端，其下可以直接挂载AHB外设，如目前挂载的AHB GPIO。

二级APB总线矩阵实质由一个总线地址译码器和一个总线从设备多路选择器构成，对来自一级AHB总线的地址信号进行译码为总线主从设备进行了端口地址映射，并选中相应的从及设备。APB总线从设备多路复用器的RTL级视图如下图5。

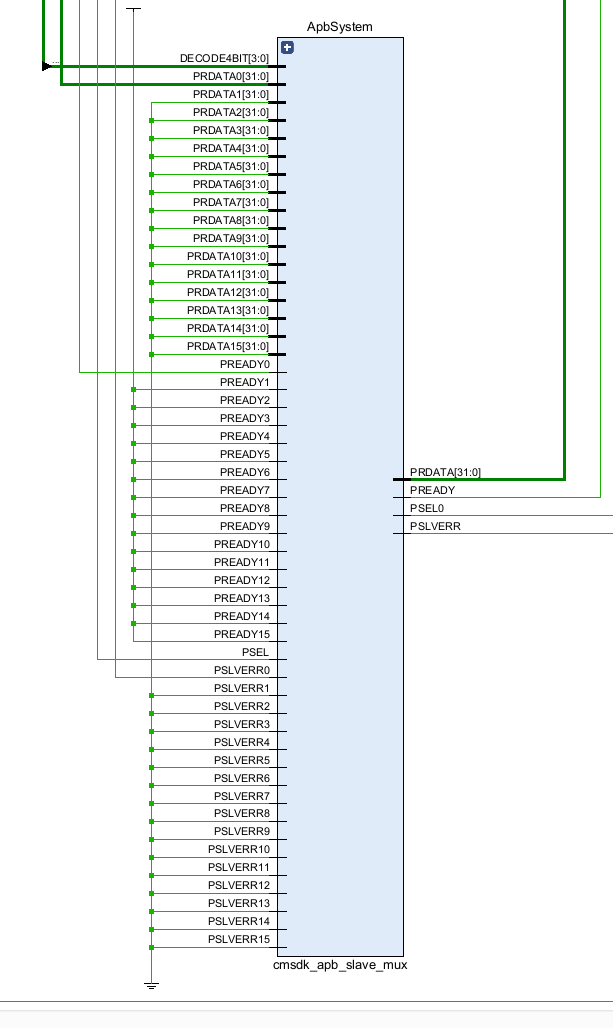


图5 APB总线从设备多路复用器的RTL级视图

## 2.4 片上外设的创建

### 2.4.1 AHB片上存储器外设的实现

为了使Cortex-M3处理器可以运行软件代码，需要将所编写的软件代码转换成机器码，这些机器码可以直接运行在CortexM3处理器上。本设计中，通过使用FPGA内部的片上块处理器BRAM作为程序存储器ITCM实现存储功能，同时也使用BRAM作为数据存储器DTCM实现存储功能。使用官方提供的cmsdk包中的基本模块进行构建，如BRAM模块cmsdk\_fpga\_sram.v，AHB和BRAM接口模块cmsdk\_ahb\_to\_sram.v等。

在KEIL中的软件代码编写完成后，将软件代码以.hex文件的形式导出，在verilog硬件代码的程序存储器ITCM中通过readmemh函数读取.hex文件中的十六进制数据从而完成软件和硬件之间的交互。

### 2.4.2 GPIO外设的实现

AHB\_GPIO同样使用cmsdk包中的模块实现，其中的文件及其功能描述如下图所示：

|  |  |  |
| --- | --- | --- |
| 模块 | 文件名 | 功能描述 |
| AHB\_GPIO | cmsdk\_ahb\_gpio.v | 将GPIO连接到AHB总线 |
| cmsdk\_ahb\_to\_iop.v | GPIO底层逻辑 |
| cmsdk\_iop\_gpio.v |

其对应的RTL视图如下图7所示：

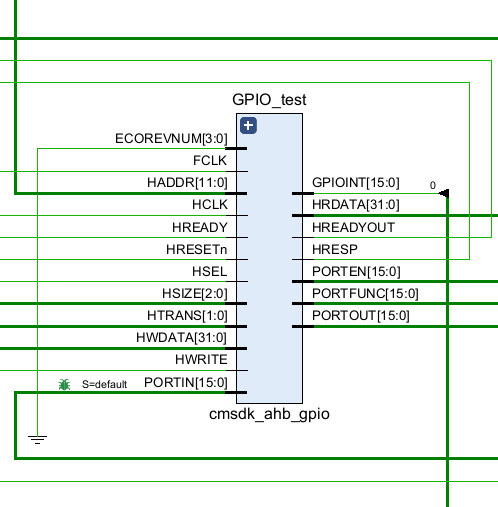


图7 GPIO模块对应的RTL视图

该GPIO是双向GPIO且具有功能丰富的寄存器资源可用于各种配置，如配置为推挽输出，浮空输入，管脚复用等等，物理地址为0x50000000，可通过对GPIO的可编程寄存器进行操作来设置GPIO的功能。本题采用按键外部中断触发使用GPIO软件模拟的SPI通信，并用GPIO输出控制LED的闪烁来表示程序的正常运行。

其中外部中断的硬件代码如下图8所示。该处理器内核共有240个用户可配置的中断，本系统将可配置中断的低四位由低到高分别配置为串口接收中断、串口发送中断、串口空闲中断和GPIO外部中断。

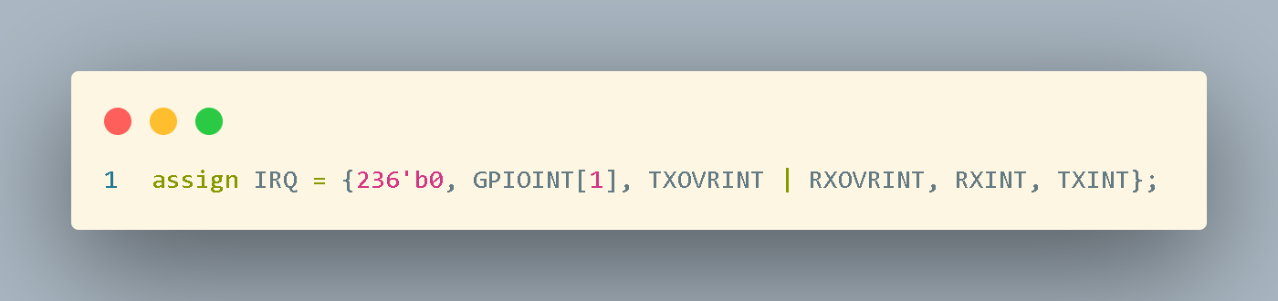


图8 硬件中断IRQ部分代码

### 2.4.3 UART外设的实现

本系统中串口的物理地址为0x40001000，可通过操作寄存器设置波特率，设置工作方式，查询工作状态等。该模块的RTL视图如下图9所示。

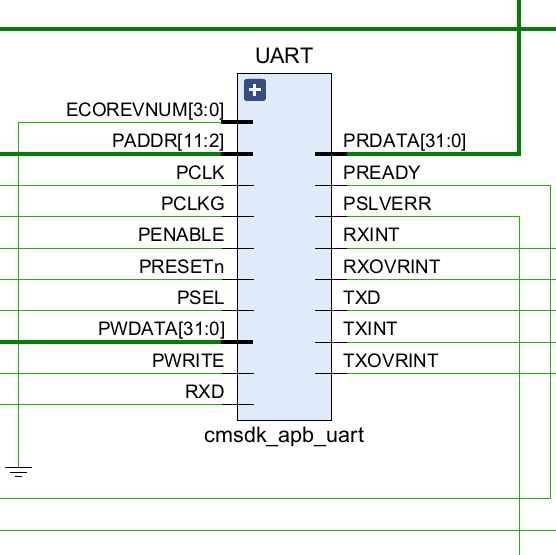


图9 UART对应的RTL视图

# 3 软件设计方案

## 3.1 总线的地址映射

根据自行设计的总线矩阵，进行硬件地址的软件映射，KEIL中的软件代码如下图10所示。

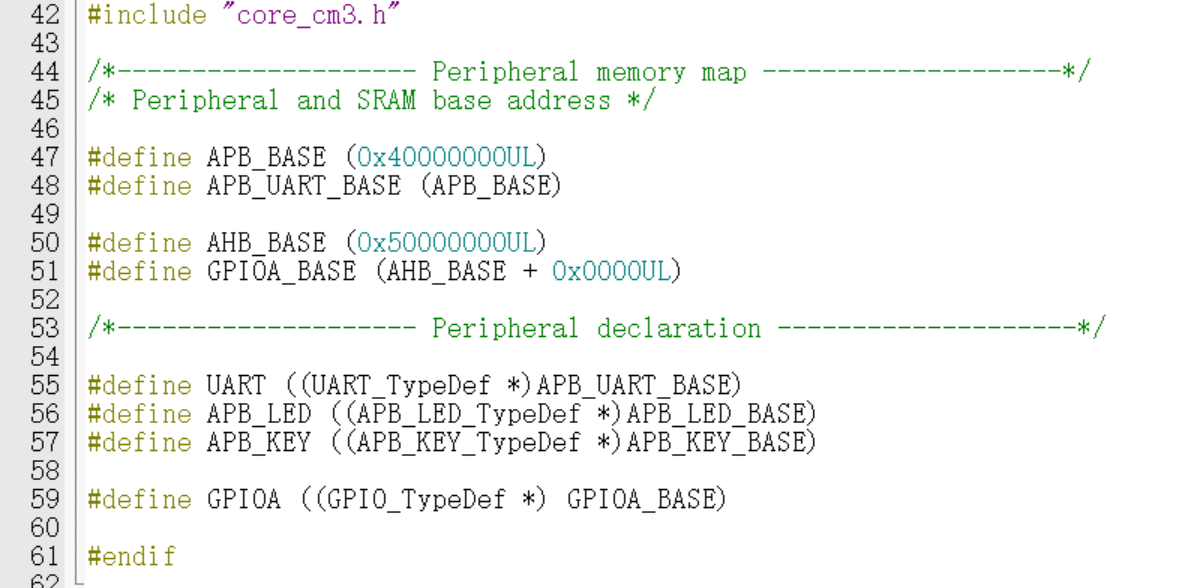


图10 总线矩阵及外设的地址映射

## 3.2 GPIO的软件封装

根据官方CMSDK工具生成的GPIO硬件模块，通过官方提供的Arm Cortex-M3 System Design Kit使用手册，可以在KEIL中编写相应结构体定义和外设库函数的相应软件代码。具体代码如下图11和图12所示。

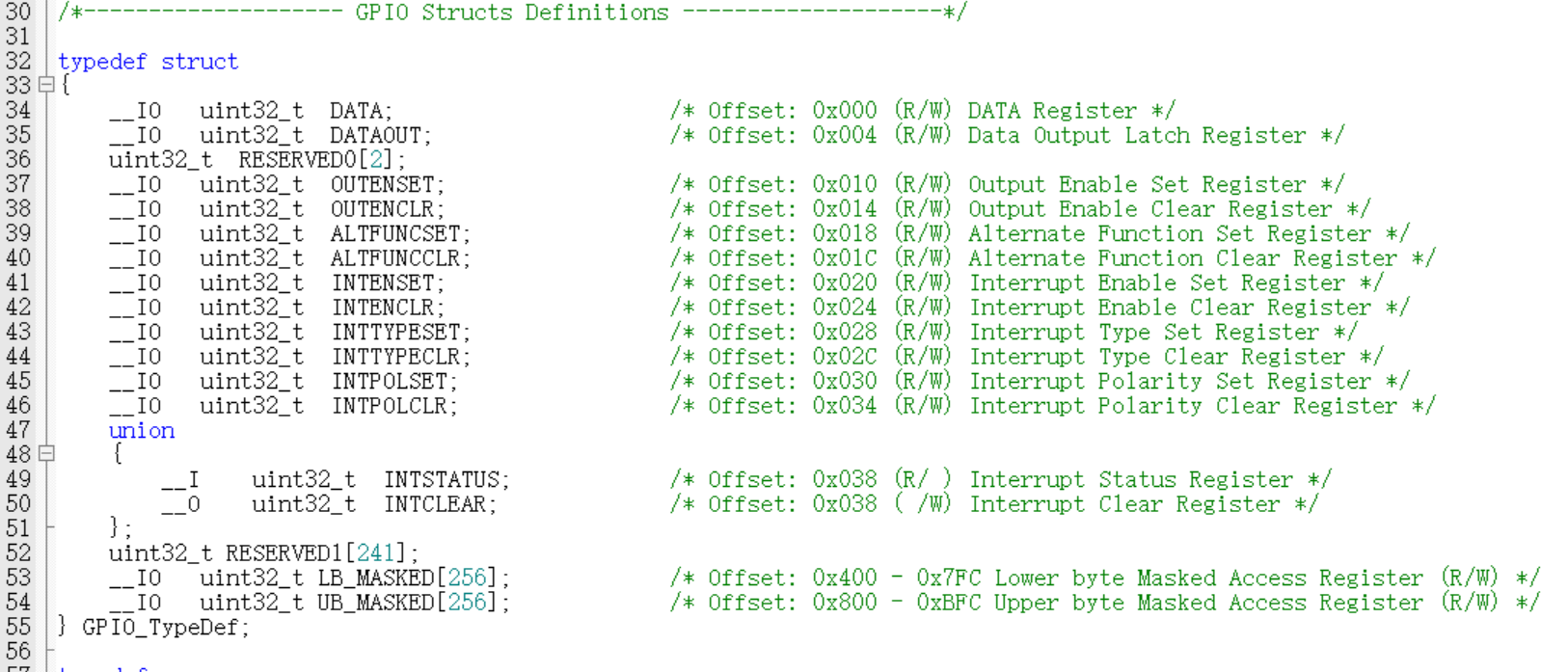


图11 部分GPIO外设的结构体定义代码

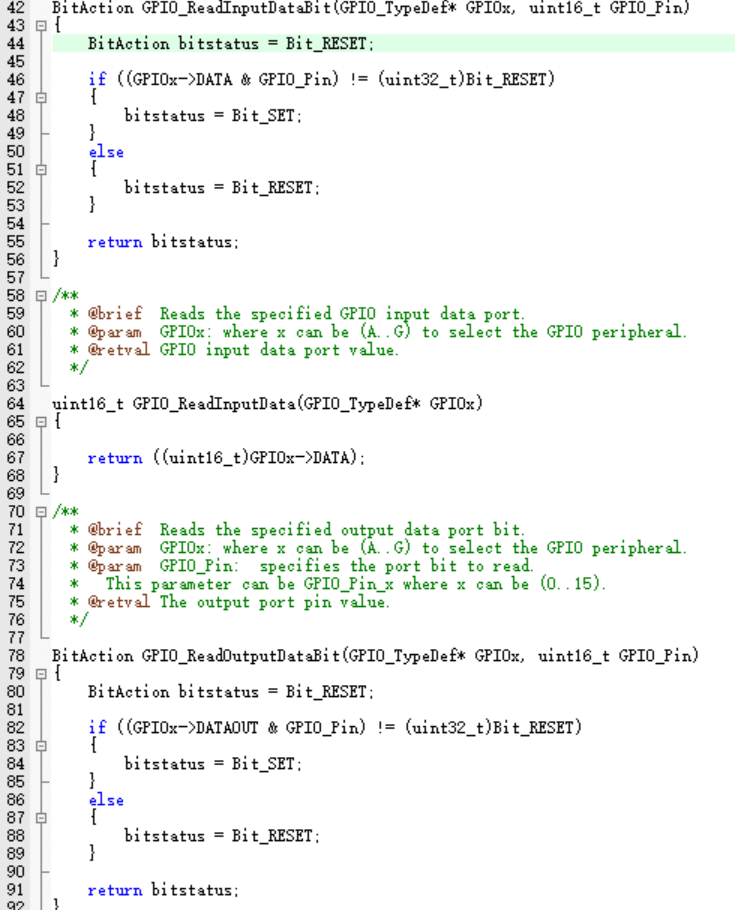


图12 部分GPIO外设的库函数封装代码

## 3.3 UART的软件封装

UART的软件封装和GPIO类似因此在此省略。

## 3.4 软件模拟SPI协议的实现

SPI，是一种高速的，全双工，同步的通信总线，并且在芯片的管脚上只占用 四根线，节约了芯片的管脚，同时为 PCB 的布局上节省空间，提供方便。 SPI 以主从方式工作，需要至少 4 根线，事实上 3 根也可以（单向传输时）。 也是所有基于 SPI 的设备共有的，它们是 SDI（数据输入），SDO（数据输出）， SCK（时钟），CS（片选）。 SPI 总线处于空闲状态时，片选信号 CS 要维持高电平，CLK 信号为低电平；当 SPI 总想开始工作时，首先是片选 CS 拉低，然后是 CLK 时钟按照规定的时钟频率 5 输出，传输的数据通过 DI 数据进行发送；当数据发送完成后，片选 CS 要拉高代 表结束，同时 CLK 信号输出为低电平。一个时钟只发送一个 bit，先发送高位，再 发送低位。

SPI 有 4 种模式：

（1）第一个沿接收数据 CPHA(clock phase)=0；

（2）第二个沿接收数据 CPHA(clock phase)=1；

（3）空闲时钟为高 CPOL(clock polarity)=1；

（4）空闲时钟为低 CPOL=0。 一般使用 0 模式：时钟空闲为低，第一个时钟沿接收数据。其时序图如下图 13。

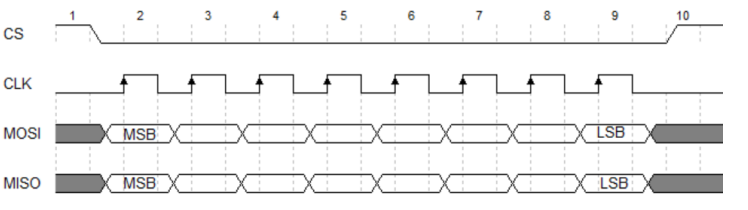


图13 SPI时序图

因此软件部分的SPI代码主要实现部分现如下图14所示。

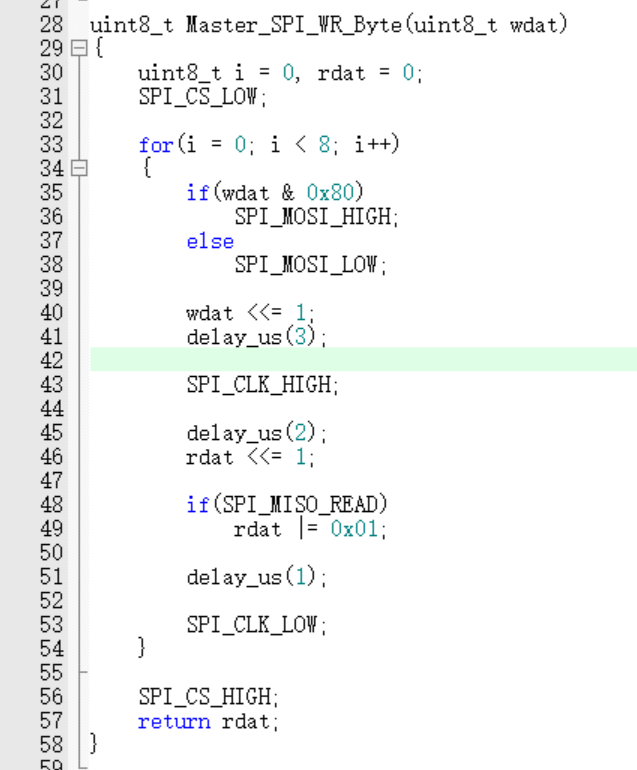


图14 SPI的软件主要实现部分

# 4 系统布线和资源消耗

## 4.1 系统的RTL视图

经过QUARTUS综合和布局布线之后系统的RTL视图如下图15所示。

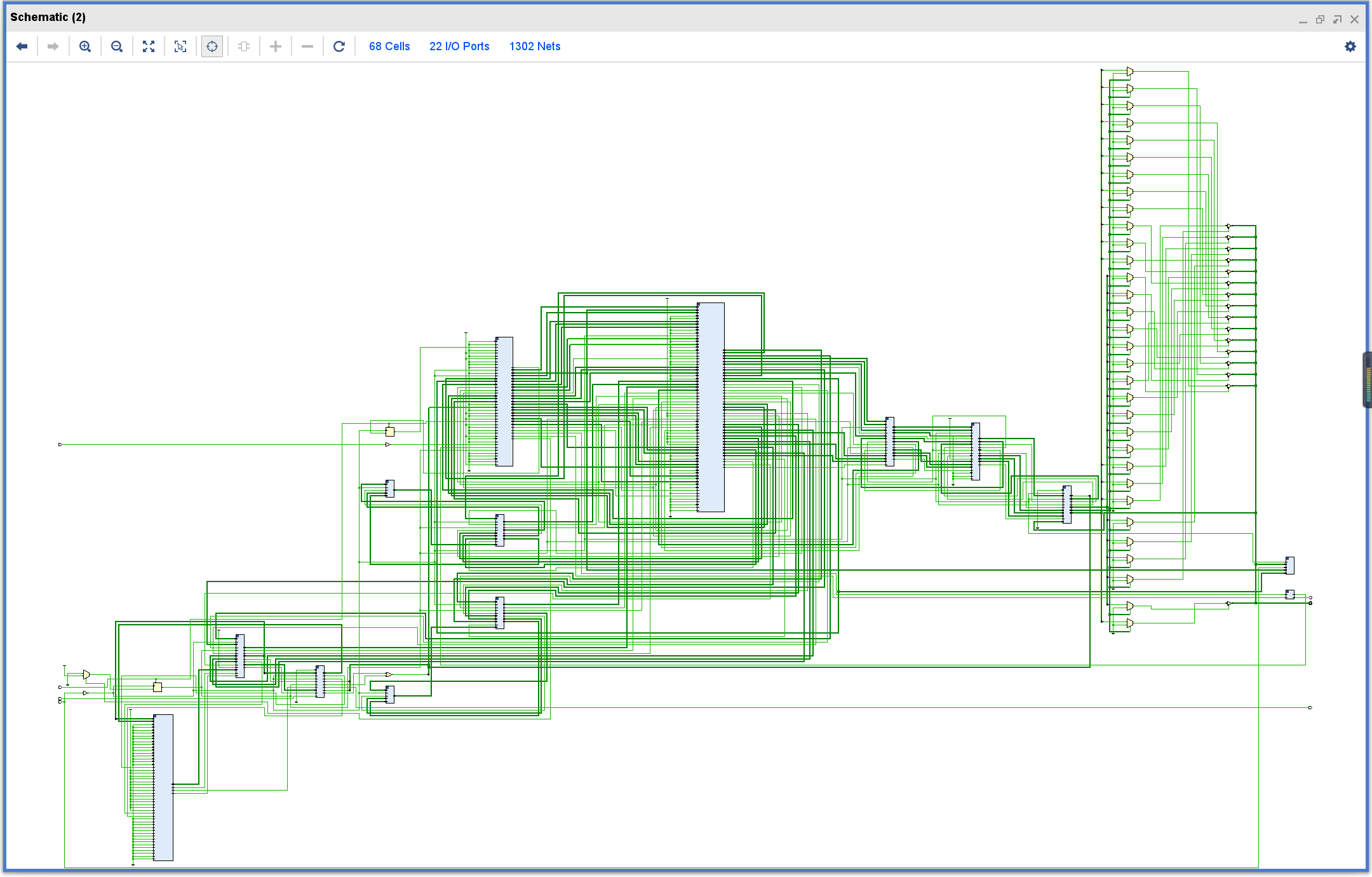


图15 系统总体的RTL视图

## 4.2 系统的资源消耗

本系统的FPGA的资源消耗如下图16所示：

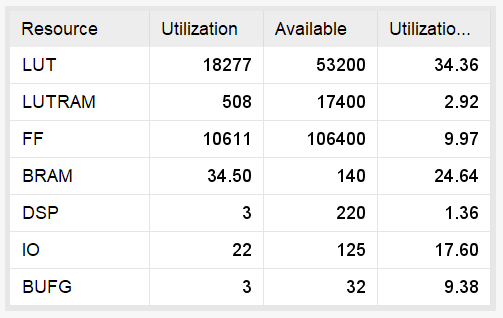


图16 系统的资源消耗

该开发板共有LUT资源53200，BRAM资源34.5，共使用LUT资源18277，BRAM资源24.64，资源利用较为合理。

# 5 仿真与上板测试

## 5.1 总线仿真

总线I-Code仿真如下图17所示，通过对比I-Code总线上和反汇编代码（如下图18所示）中的地址以及数据来检验硬件部分是否能够正确执行软件部分的代码。

经过数据对比，可以得出总线能够正常传输数据，硬件调试成功。

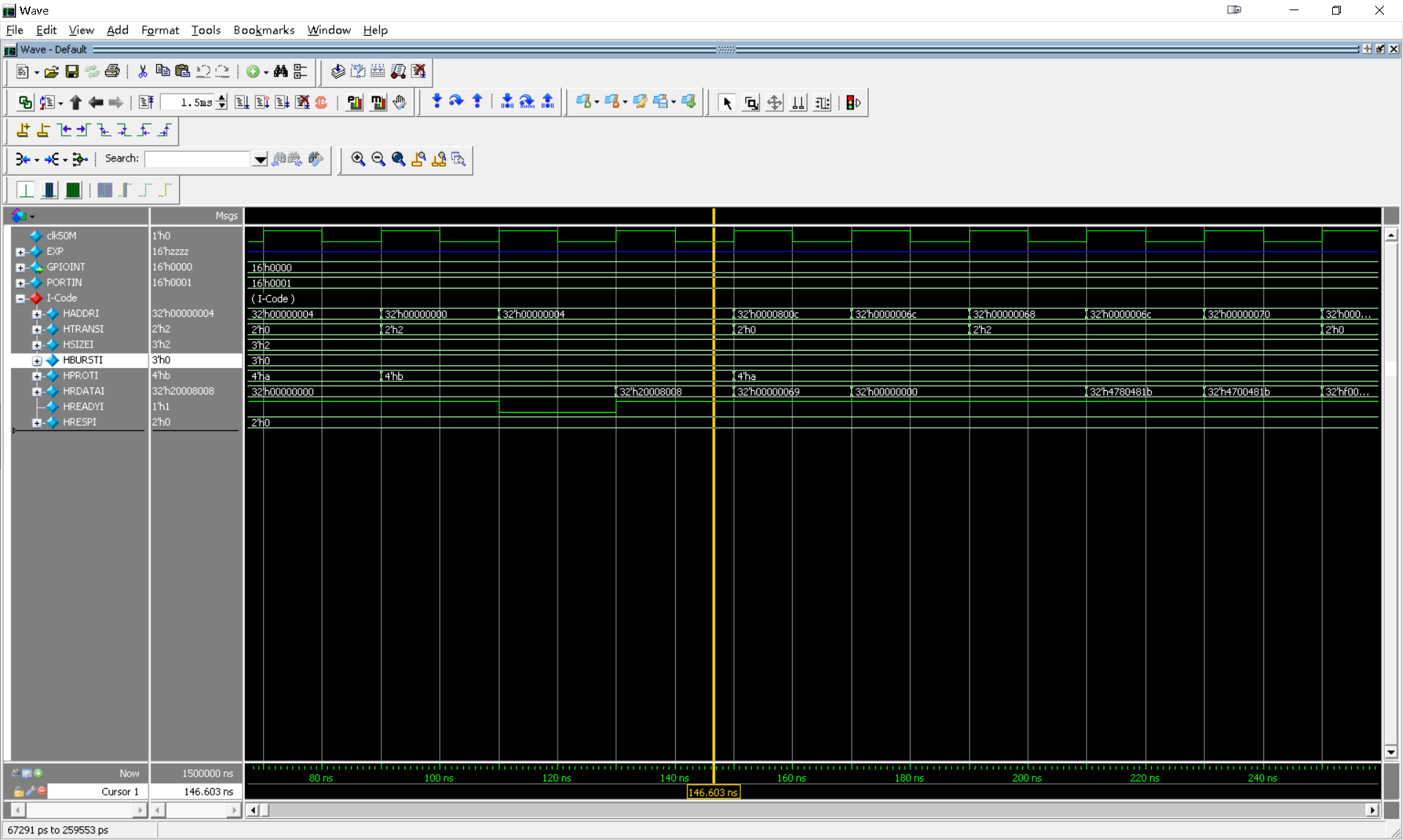


图17 I-Code总线的仿真

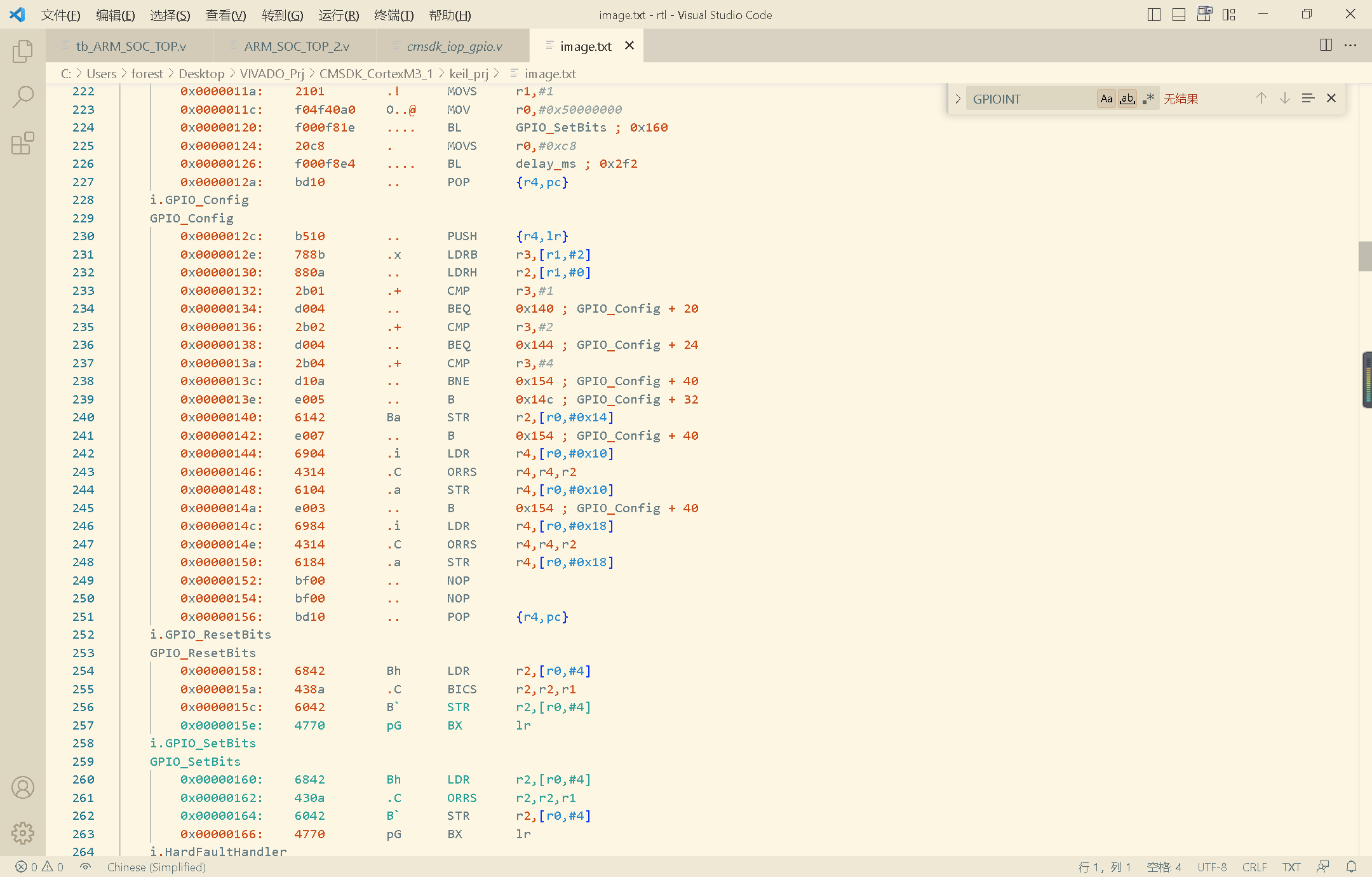


图18 经过KEIL反汇编后的反汇编代码

## 5.2 逻辑分析仪对SPI通信的信号进行捕获

在顶层代码中增添VIVADO的逻辑分析仪IP核ila，对其中的信号进行逻辑分析。捕获信号如下图19所示。

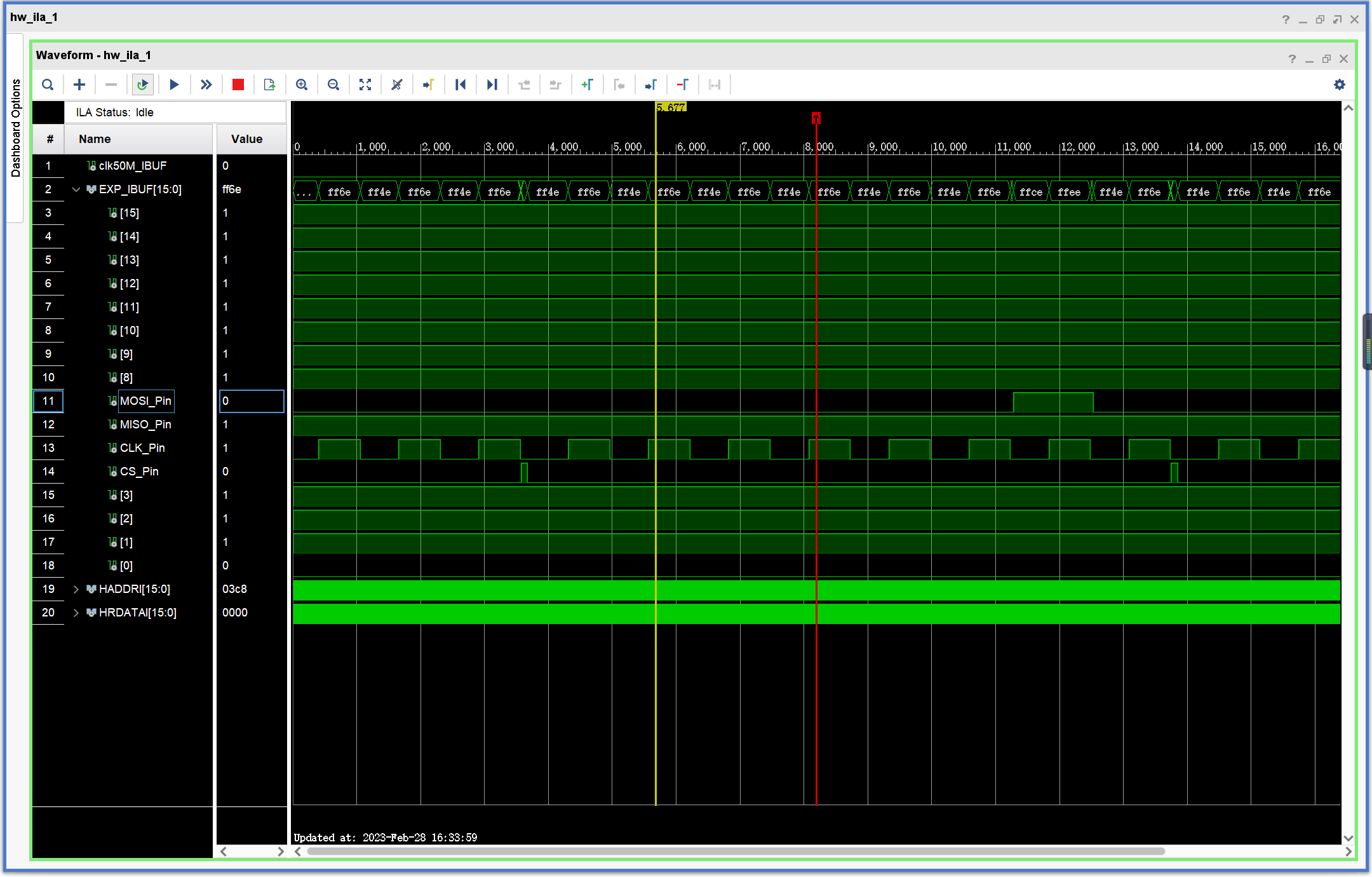


图19 利用VIVADO的逻辑分析仪对SPI信号进行捕获

在软件代码的while循环中反复通过SPI发送SOC内核名称，使用VIVADO的逻辑分析仪ila，抓取SPI主机的四个关键信号，上图为一个传输周期的信号，对比图14可验证信号传输的正确性，成功完成了GPIO模拟SPI主机协议并发送数据的功能。下图20为软件部分while循环中的代码。

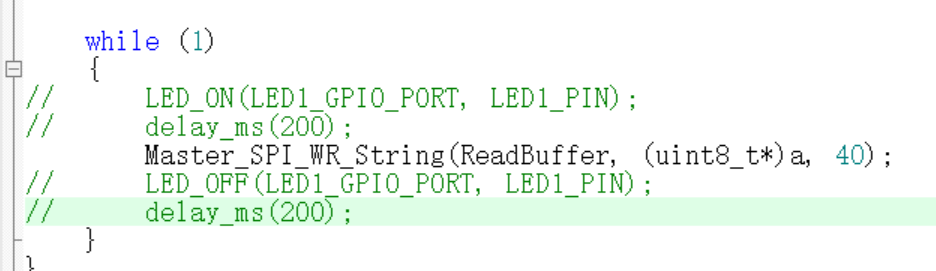


图20 软件部分while循环的代码

# 6 参考文献

1. ARM Cortex-M0 全可编程 SoC 原理及实现:面向处理器﹑协议、外设、编程和操 作系统/何宾编著.—北京:清华大学出版社,2017.
2. 谭盾. 基于ARM Cortex-M0核的MCU设计及应用[D].电子科技大学,2020.DOI:10.27005/d.cnki.gdzku.2020.002435.

# 7 附录

这是我的Github仓库网址：[JimmyForest/The-school-preliminary-competition-of-College-Students--innovation-and-Entrepreneurship-Competition (github.com)](https://github.com/JimmyForest/The-school-preliminary-competition-of-College-Students--innovation-and-Entrepreneurship-Competition)。