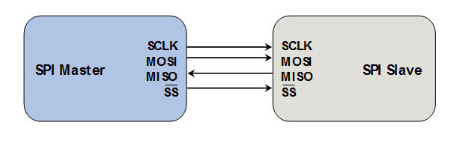
1. **SPI原理介绍**

SPI（Serial Peripheral Interface，串行外围设备接口），是Motorola公司提出的一种同步串行接口技术，是一种高速、全双工、同步通信总线，在芯片中只占用四根管脚用来控制及数据传输。

* 1. **SPI标准四线总线**

SPI总线传输只需要4根线就能完成，如下图所示。这四根线的作用分别如下：



SCK(Serial Clock)：SCK是串行时钟线，作用是Master向Slave传输时钟信号，控制数据交换的时机和速率；

MOSI(Master Out Slave in)：在SPI Master上也被称为Tx-channel，作用是SPI主机给SPI从机发送数据；

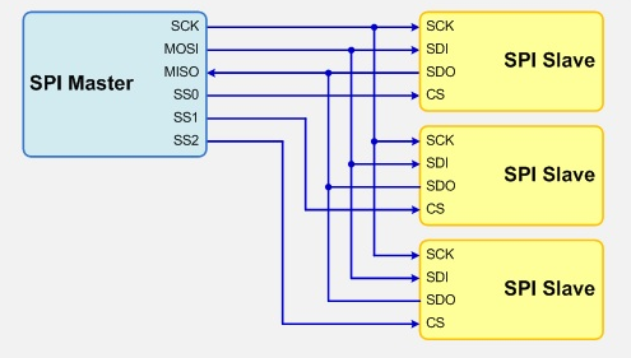
CS/SS(Chip Select/Slave Select)：片选信号线，作用是SPI Master选择与哪一个SPI Slave通信，低电平表示从机被选中(低电平有效)；

MISO(Master In Slave Out)：在SPI Master上也被称为Rx-channel，作用是SPI主机接收SPI从机传输过来的数据。

* 1. **主从模式**

SPI总线采用主从模式（Master-Slave）的控制方式，支持单Master多Slave。

单Master多Slave的典型结构如下图所示。



SPI规定了两个SPI设备之间通信必须由主设备Master来控制从设备Slave。也就是说，如果FPGA是主机的情况下，不管是FPGA给芯片发送数据还是从芯片中接收数据，写Verilog逻辑的时候片选信号CS与串行时钟信号SCK必须由FPGA来产生。一个Master可以设置多个片选(Chip Select)来控制多个Slave。

SPI协议还规定Slave设备的clock由Master通过SCK管脚提供给Slave，Slave本身不能产生或控制clock，没有clock则Slave不能正常工作。

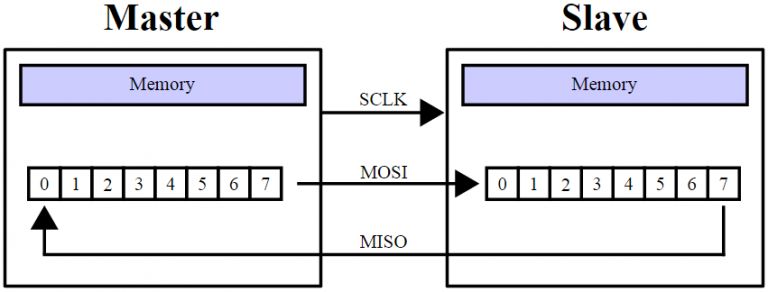
* 1. **同步传输**

SPI总线在传输数据的同时也传输了时钟信号，所以SPI协议是一种同步（Synchronous）传输协议。Master会根据将要交换的数据产生相应的时钟脉冲，组成时钟信号，时钟信号通过时钟极性(CPOL)和时钟相位(CPHA)控制两个SPI设备何时交换数据以及何时对接收数据进行采样，保证数据在两个设备之间是同步传输的。

* 1. **全双工**

SPI总线协议是一种全双工的串行通信协议，数据传输时高位在前，低位在后。

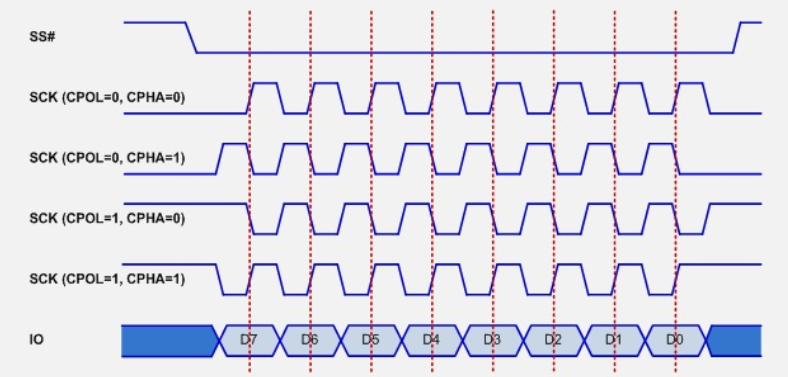
SPI协议规定一个SPI设备不能在数据通信过程中仅仅充当一个发送者（Transmitter）或者接受者（Receiver）。在片选信号CS为0的情况下，每个clock周期内，SPI设备都会发送并接收1 bit数据，相当于有1 bit数据被交换了。数据传输高位在前，低位在后（MSB first）。SPI主从结构内部数据传输示意图如下图所示。



* 1. **SPI总线传输的4种模式**

SPI总线传输共有4种模式，由时钟极性(CPOL，Clock Polarity)和时钟相位(CPHA，Clock Phase)来定义，其中CPOL参数规定了SCK时钟信号空闲状态的电平，CPHA规定了数据是在SCK时钟的上升沿被采样还是下降沿被采样。

这四种模式的时序图如下图所示。



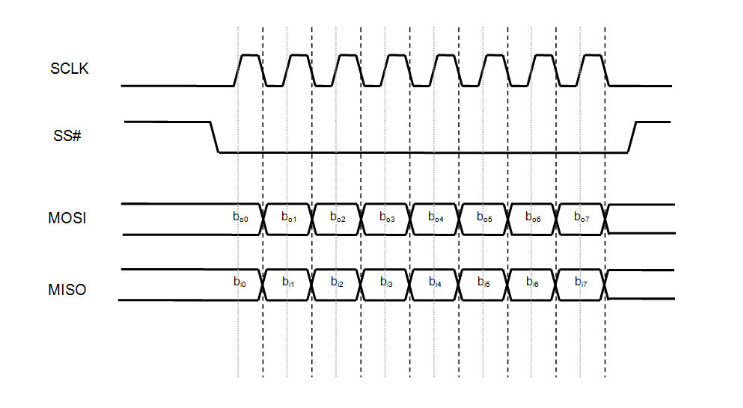
模式0：CPOL= 0，CPHA=0。SCK串行时钟线空闲是为低电平，数据在SCK时钟的上升沿被采样，数据在SCK时钟的下降沿切换。

　　模式1：CPOL= 0，CPHA=1。SCK串行时钟线空闲是为低电平，数据在SCK时钟的下降沿被采样，数据在SCK时钟的上升沿切换。

　　模式2：CPOL= 1，CPHA=0。SCK串行时钟线空闲是为高电平，数据在SCK时钟的下降沿被采样，数据在SCK时钟的上升沿切换。

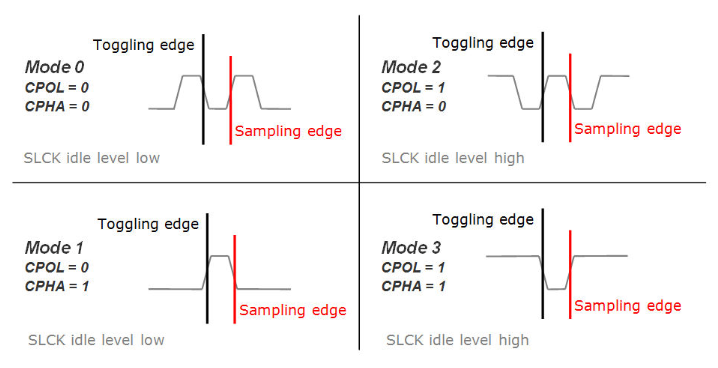
　　模式3：CPOL= 1，CPHA=1。SCK串行时钟线空闲是为高电平，数据在SCK时钟的上升沿被采样，数据在SCK时钟的下降沿切换。

其中比较常用的模式是模式0和模式3。

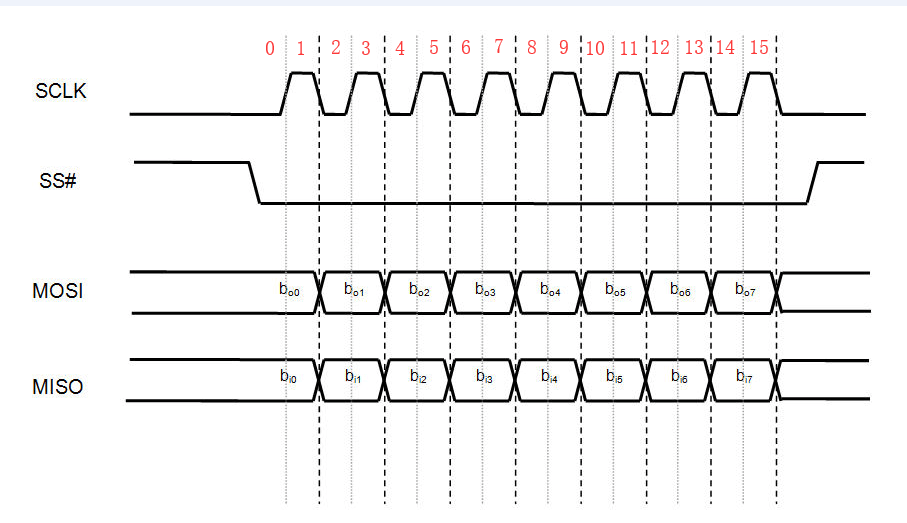
为了更清晰的描述SPI总线的时序，下面展现了模式0下的SPI时序图。

如上图所示在模式0下，在空闲状态下，SCK串行时钟线为低电平。当SS被主机拉低以后，数据传输开始，数据线MOSI和MISO的数据切换(Toggling)发生在时钟的下降沿(上图的黑色虚线)，而数据线MOSI和MISO的数据的采样(Sampling)发生在数据的正中间(上图中的灰色实线)。

下图清晰的描述了其他3种模式下数据线MOSI和MISO的数据切换(Toggling)位置和数据采样位置的关系图。



* 1. **SPI读操作流程**

这里以实现模式0的时序为例说明，其他模式的实现与此类似。

当主机通过SPI总线往从机发送一个字节(8-bit)的数据时，首先主机把CS片选信号设置为0，表示准备开始发送数据。整个发送数据过程其实可以分为16个状态：

　 状态0：SCK为0，MOSI为要发送的数据的最高位，即I\_data\_in[7]；

　　状态1：SCK为1，MOSI保持不变；

　　状态2：SCK为0，MOSI为要发送的数据的次高位，即I\_data\_in[6]；

　　状态3：SCK为1，MOSI保持不变；

　　状态4：SCK为0，MOSI为要发送的数据的下一位，即I\_data\_in[5]；

　　状态5：SCK为1，MOSI保持不变；

　　状态6：SCK为0，MOSI为要发送的数据的下一位，即I\_data\_in[4]；

　　状态7：SCK为1，MOSI保持不变；

　　状态8：SCK为0，MOSI为要发送的数据的下一位，即I\_data\_in[3]；

　　状态9：SCK为1，MOSI保持不变；

　　状态10：SCK为0，MOSI为要发送的数据的下一位，即I\_data\_in[2]；

　　状态11：SCK为1，MOSI保持不变；

　　状态12：SCK为0，MOSI为要发送的数据的下一位，即I\_data\_in[1]；

　　状态13：SCK为1，MOSI保持不变；

　　状态14：SCK为0，MOSI为要发送的数据的最低位，即I\_data\_in[0]；

　　状态15：SCK为1，MOSI保持不变。

　　一个字节数据发送完毕以后，产生一个发送完成标志位O\_tx\_done，并把CS信号拉高完成一次发送。通过观察上面的状态可以发现状态编号为奇数的状态要做的操作实际上是一模一样的，所以写代码的时候为了精简代码，把状态号为奇数的状态全部整合到了一起。

* 1. **SPI写操作流程**

当主机通过SPI总线接收从机的一个字节(8-bit)的数据时，首先主机把CS片选信号设置为0，表示准备开始接收数据。整个接收数据过程其实也可以分为16个状态，但是与发送过程不同的是，为了保证接收到的数据准确，必须在数据的正中间采样，也就是说模式0时序图中灰色实线的地方才是代码中锁存数据的地方，所以接收过程的每个状态执行的操作相应为：

　　状态0：SCK为0，不锁存MISO上的数据；

　　状态1：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[7]；

　　状态2：SCK为0，不锁存MISO上的数据；

　　状态3：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[6]；

　　状态4：SCK为0，不锁存MISO上的数据；

　　状态5：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[5]；

　　状态6：SCK为0，不锁存MISO上的数据；

　　状态7：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[4]；

　　状态8：SCK为0，不锁存MISO上的数据；

　　状态9：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[3]；

　　状态10：SCK为0，不锁存MISO上的数据；

　　状态11：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[2]

　　状态12：SCK为0，不锁存MISO上的数据；

　　状态13：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[1]

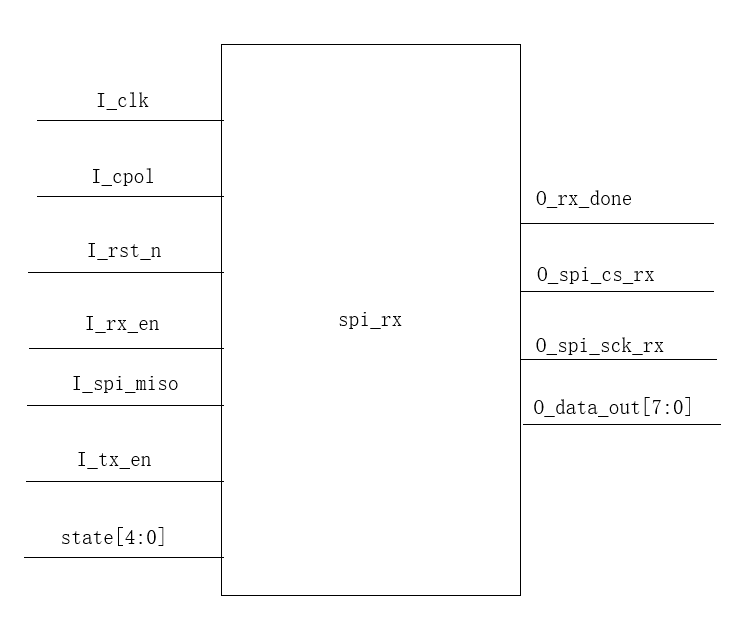
　　状态14：SCK为0，不锁存MISO上的数据；

　　状态15：SCK为1，锁存MISO上的数据，即把MISO上的数据赋值给O\_data\_out[0]

　　一个字节数据接收完毕以后，产生一个接收完成标志位O\_rx\_done并把CS信号拉高完成一次数据的接收。通过观察上面的状态可以发现状态编号为偶数的状态要做的操作实际上是一模一样的，所以写代码的时候为了精简代码，把状态号为偶数的状态全部整合到了一起。

1. **模块划分及接口定义**
   1. **SPI接收数据模块**

SPI接受数据模块spi\_rx用于主机从从机接收数据，在系统时钟的控制下，根据时钟极性决定SPI时钟的空闲状态，同时当接受使能有效时，拉低片选，按位接收MISO线上来自从机的数据，每当接收完成一个字节数据O\_data\_out时，产生接收完成标志。当全部接收数据完成后，则将片选拉高，结束接收数据过程。

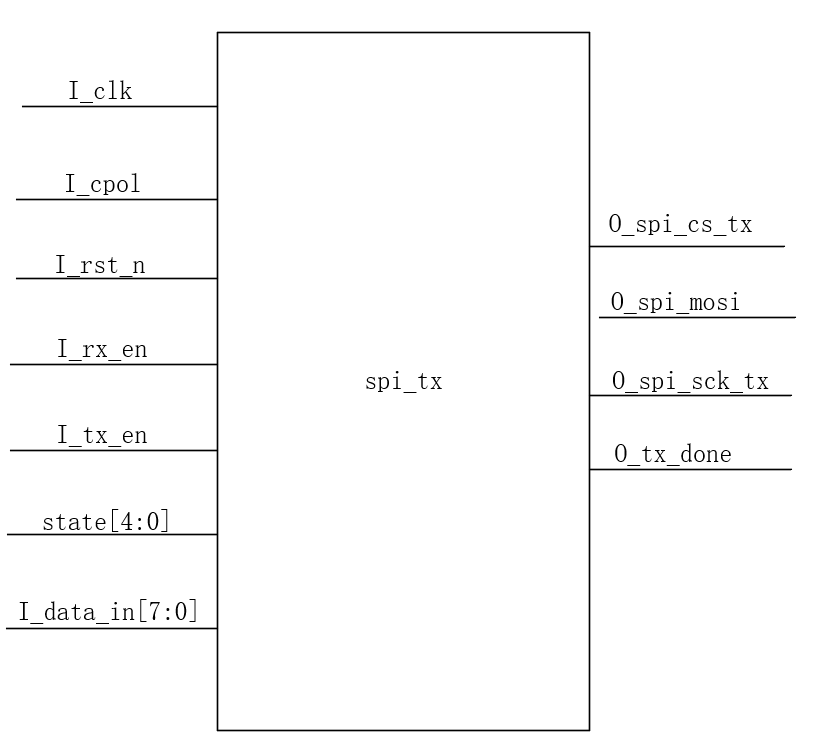


|  |  |  |  |
| --- | --- | --- | --- |
| 端口名 | 位宽 | 方向 | 描述 |
| I\_clk | 1 | 输入 | 系统时钟 |
| I\_cpol | 1 | 输入 | 时钟极性 |
| I\_rst\_n | 1 | 输入 | 复位信号，低电平有效 |
| I\_rx\_en | 1 | 输入 | 接收使能，高电平有效 |
| I\_spi\_miso | 1 | 输入 | 接收数据 |
| I\_tx\_en | 1 | 输入 | 发送使能，高电平有效 |
| State | 5 | 输入 | 模式状态 |
| O\_rx\_done | 1 | 输出 | 接收完成标志，，高电平有效 |
| O\_spi\_cs\_rx | 1 | 输出 | 接收片选，低电平有效 |
| O\_spi\_sck\_rx | 1 | 输出 | 接收部分SPI时钟 |
| O\_data\_out | 8 | 输出 | 接收数据 |

表2.1 SPI接收数据模块端口说明

* 1. **SPI发送数据模块**

SPI发送数据模块spi\_tx用于主机向从机发送数据，在系统时钟的控制下，根据时钟极性决定SPI时钟的空闲状态，同时当发送使能有效时，拉低片选，按位发送MOSI线上的数据，每当发送完成一个字节数据I\_data\_in时，产生发送完成标志。当全部发送数据完成后，则将片选拉高，结束发送数据过程。

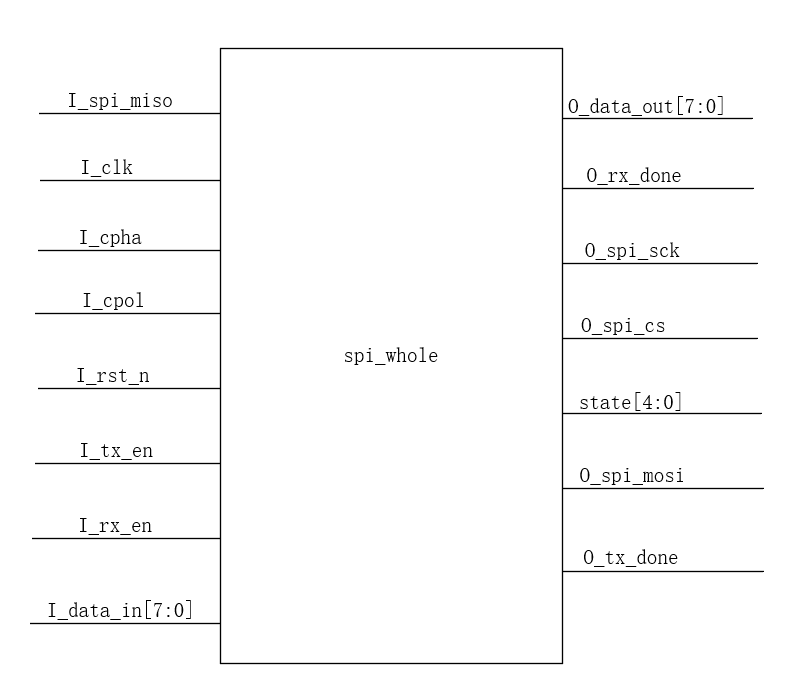


|  |  |  |  |
| --- | --- | --- | --- |
| 端口名 | 位宽 | 方向 | 描述 |
| I\_clk | 1 | 输入 | 系统时钟 |
| I\_cpol | 1 | 输入 | 时钟极性 |
| I\_rst\_n | 1 | 输入 | 复位信号，低电平有效 |
| I\_rx\_en | 1 | 输入 | 接收使能，高电平有效 |
| I\_tx\_en | 1 | 输入 | 发送使能，高电平有效 |
| State | 5 | 输入 | 模式状态 |
| I\_data\_in | 8 | 输出 | 发送数据 |
| O\_spi\_cs\_tx | 1 | 输出 | 发送片选，低电平有效 |
| O\_spi\_mosi | 1 | 输出 | 发送数据 |
| O\_spi\_sck\_tx | 1 | 输出 | 发送部分SPI时钟 |
| O\_tx\_done | 1 | 输出 | 发送完成标志，高电平有效 |

表2.2 SPI发送数据模块端口说明

* 1. **SPI顶层模块spi\_whole**

SPI顶层模块spi\_whole用于同时调用发送模块spi\_tx和接收模块spi\_rx，当发送使能和接收使能同时有效时，拉低与其通信的从机的片选线，在系统时钟控制下，由时钟极性和时钟相位共同决定SPI通信模式，在SPI时钟的同步下，根据SPI协议规定，同时完成主机向从机发送数据和主机接收来自从机的数据。当通信完成，将片选拉高，结束通信过程。



|  |  |  |  |
| --- | --- | --- | --- |
| 端口名 | 位宽 | 方向 | 描述 |
| I\_spi\_miso | 1 | 输入 | 接收数据 |
| I\_clk | 1 | 输入 | 系统时钟 |
| I\_cpha | 1 | 输入 | 时钟相位 |
| I\_cpol | 1 | 输入 | 时钟极性 |
| I\_rst\_n | 1 | 输入 | 复位信号，低电平有效 |
| I\_tx\_en | 1 | 输入 | 发送使能，高电平有效 |
| I\_rx\_en | 1 | 输入 | 接收使能，高电平有效 |
| I\_data\_in | 8 | 输入 | 发送数据 |
| O\_data\_out | 8 | 输出 | 接收数据 |
| O\_rx\_done | 1 | 输出 | 接收完成标志，高电平有效 |
| O\_spi\_sck | 1 | 输出 | SPI时钟 |
| O\_spi\_cs | 1 | 输出 | 片选信号，低电平有效 |
| State | 5 | 输出 | 模式状态 |
| O\_spi\_mosi | 1 | 输出 | 发送数据 |
| O\_tx\_done | 1 | 输出 | 发送完成标志，高电平有效 |

表2.3 SPI顶层模块端口说明

1. **仿真与结果分析**

图3.1（a）和图3.1（b）均为四线标准SPI全双工模式0下的仿真波形图。为了方便分析，以模式0下第一个字节的发送和接收为例说明。其他情况下的仿真波形与此类似。

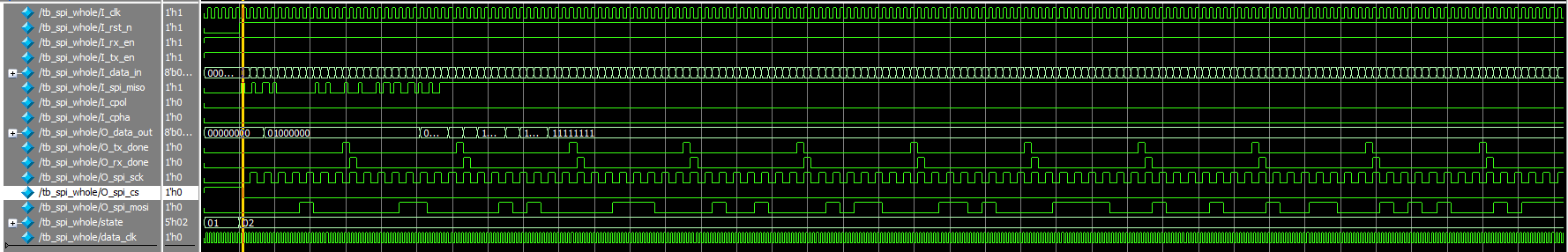


图3.1(a) 模式0仿真波形图

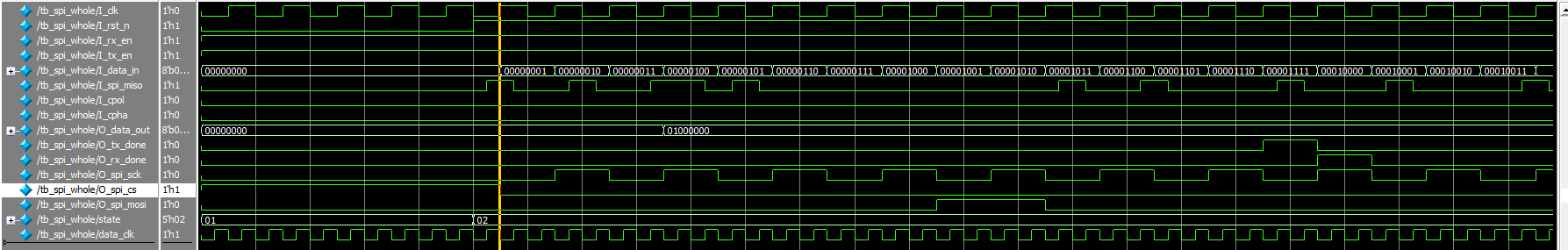


图3.1(b) 模式0仿真波形图

上图3.1（b）为模式0下仿真波形图，如图所示，时钟极性I\_cpol为低电平，时钟相位I\_cpha为低电平，表示进行的是SPI模式0下的数据传输。在模式0下SPI时钟O\_spi\_sck空闲态为低电平，位数据发送过程在每个SPI时钟的下降沿采样，在SPI时钟的上升沿切换。位数据接收过程在每个SPI时钟的上升沿采样（锁存），在SPI时钟的下降沿切换。同时，该图清晰展示了第一个字节数据发送和接收的波形。

先分析发送数据波形。如图中所示，发送使能I\_tx\_en为高电平有效。在复位信号I\_rst\_n有效拉低后，经历了5个系统时钟I\_CLK周期的复位后I\_rst\_n拉高结束复位过程。在随后出现的下一个I\_clk系统时钟周期的上升沿片选信号O\_spi\_cs被拉低，表示开始通信，此时将I\_data\_in[7]发送到主出从进线MOSI线完成D[7]位数据的发送。在随后的第一个SPI时钟O\_spi\_sck上升沿I\_data\_in[6]做发送准备（数据切换），在下一个SPI时钟下降沿I\_data\_in[6]被发送出去。如此重复，直到完成一个字节数据的发送时，可以看到发送完成标志O\_tx\_en被短暂拉高出现一个窄脉冲，表明一个字节的发送完成。

再分析接收数据波形。如图中所示，接收使能I\_rx\_en为高电平有效。在复位信号I\_rst\_n有效拉低后，经历了5个系统时钟I\_CLK周期的复位后I\_rst\_n拉高结束复位过程。在随后出现的下一个I\_clk系统时钟周期的上升沿片选信号O\_spi\_cs被拉低，表示开始通信，在第一个SPI时钟O\_spi\_sck上升沿锁存MISO线上的电平信号到O\_data\_out[7]，在SPI时钟下降沿MISO线上的电平信号做下一次接收准备（数据切换），在第二个SPI时钟上升沿锁存MISO线上的电平信号到O\_data\_out[6]，如此重复直到完成一个字节数据的接收时，可以看到发送接收标志O\_rx\_en被短暂拉高出现一个窄脉冲，表明一个字节的接收完成。

下面给出了模式1、模式2、模式3下的仿真波形图，可以清楚地看到本设计可以完整完成SPI通信四种模式下的数据传输。

模式1：

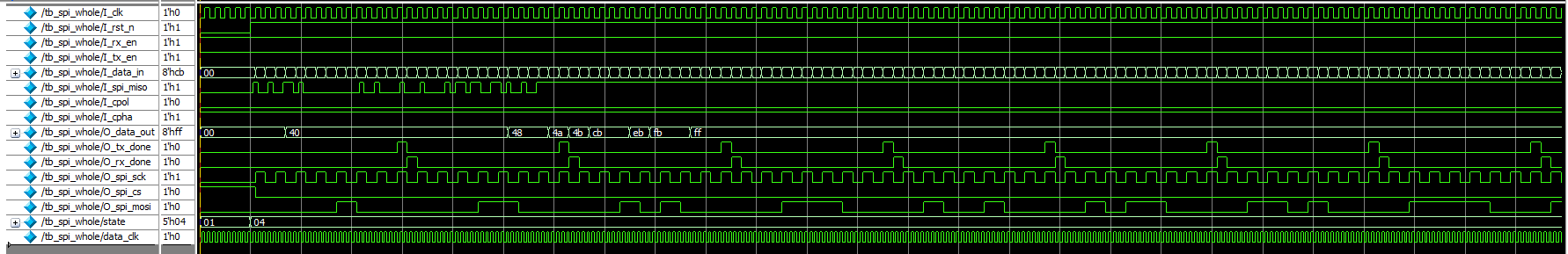


图3.2 模式1仿真波形图

模式2：

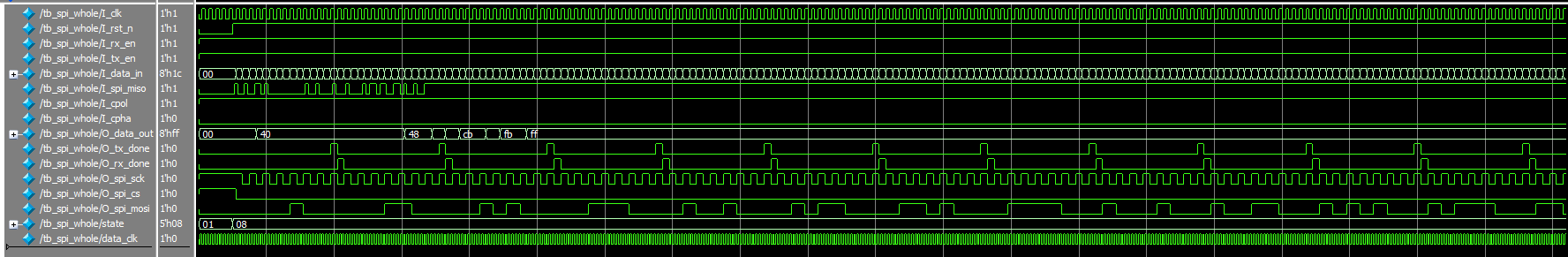


图3.3 模式2仿真波形图

模式3：:

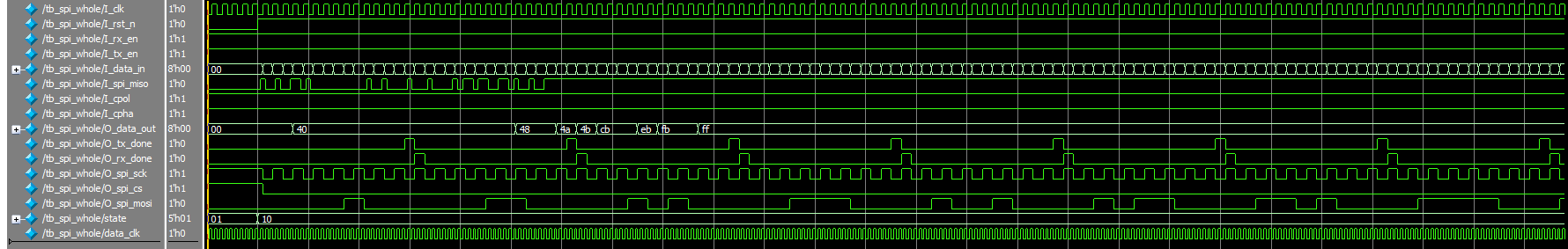


图3.4 模式3仿真波形图

1. **FPGA验证**

为了保证验证结果的严谨性，在验证思路上，本设计采用第三方生产的转接板，该转接板自带标准SPI从机接口，能够配合完成我们所设计的SPI主机接口的验证。

验证思路如下图验证框架所示。以FPGA作为主机，上位机作为从机，将设计的SPI主机接口比特流文件烧写到FPGA中，在主机FPGA的控制下，与转接板自带的SPI从机接口配合，完成发送数据与接受数据操作的验证。发送数据操作与接收数据操作的验证分别在下文4.1与4.2中具体叙述。

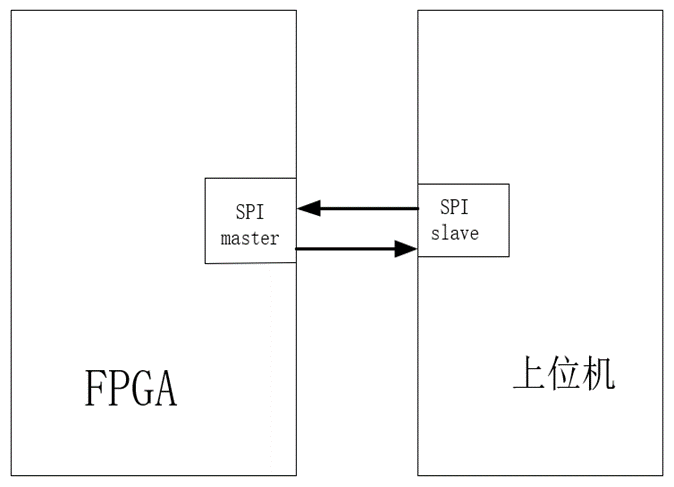


图4.1 验证框架

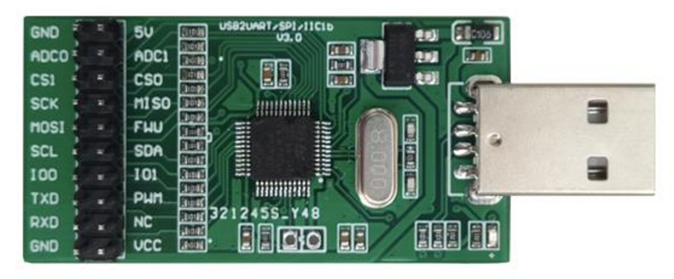


图4.2 验证中使用的转接板

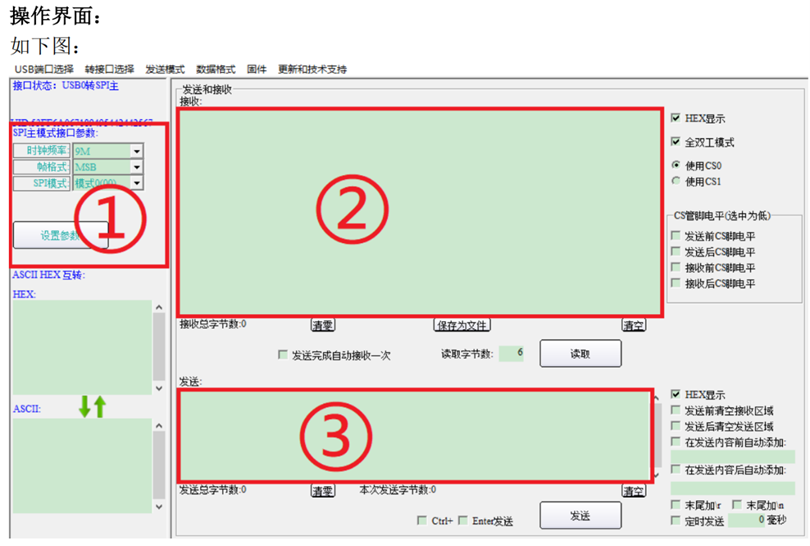


图4.3 上位机的操作界面

* 1. **FPGA验证——发送数据操作**

验证思路：作为主机的FPGA通过我们所设计的SPI主机接口往转接板自带的SPI从机接口发送数据，SPI从机接口将接收到的数据通过与转接板相接的USB发送给上位机，上位机再将收到的数据通过串口调试助手界面实时显示出来，并可以作为比对，即可验证发送数据操作能正确执行。

首先，完成设计代码的综合、实现与比特流生成。

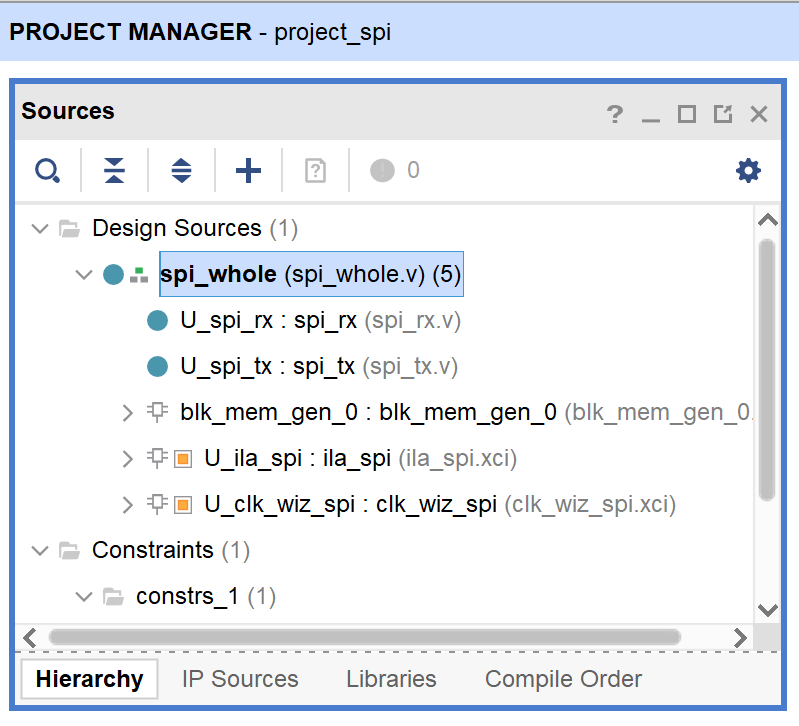
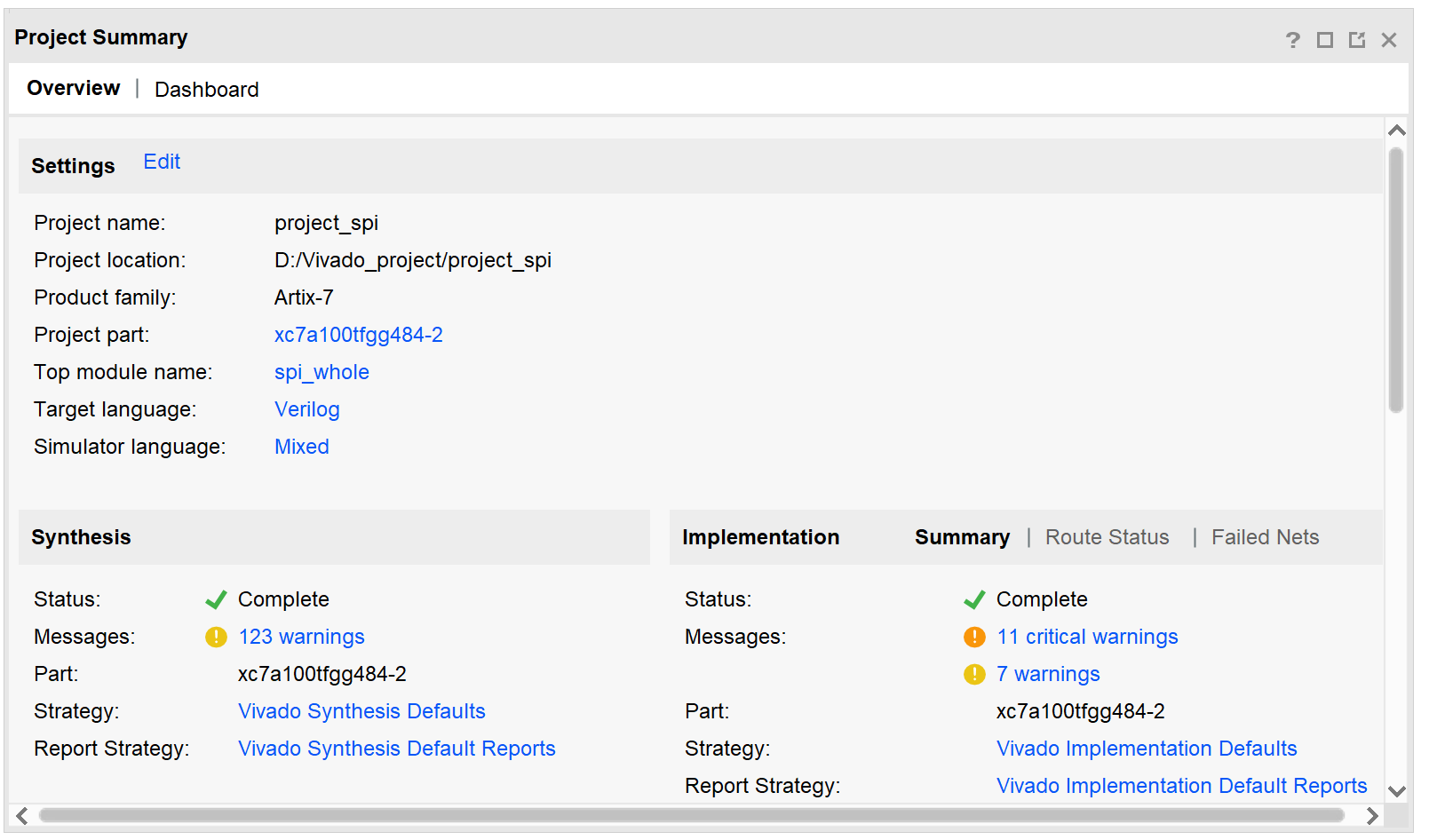


图4.4 工程文件



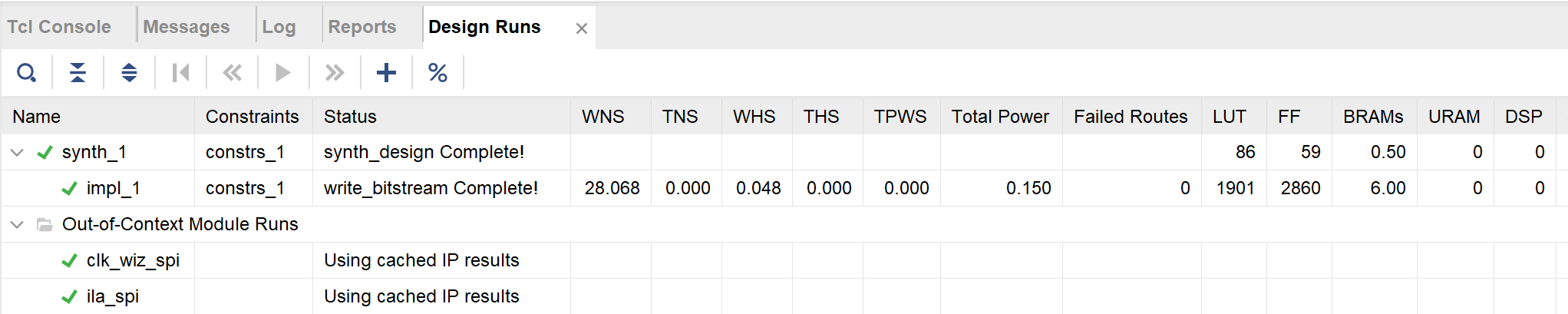


图4.5 综合与实现成功

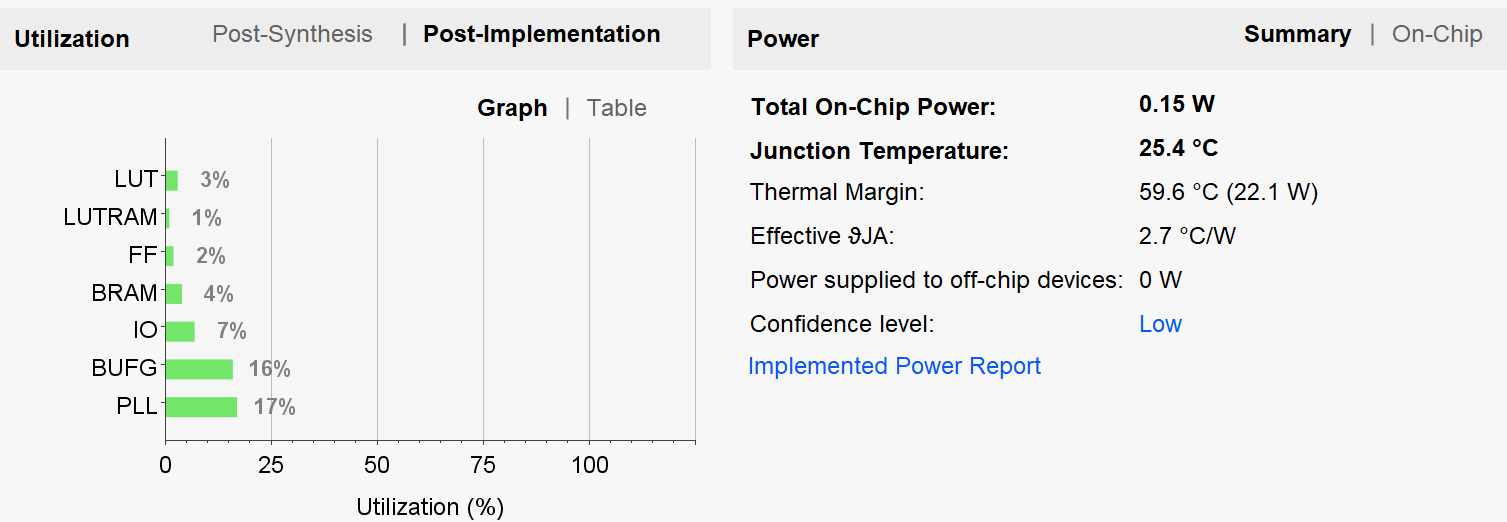


图4.6 资源报告

将作为主机的FPGA控制代码与SPI主机接口比特流文件一起烧写到FPGA开发板中。再将FPGA开发板通过杜邦线与转接板的SPI从机接口相连。

1. **单段字节发送操作验证**

FPGA主机控制代码如下，向接口内发送00~FF数据，如果串口调试助手能正确接收并显示，则表明单段字节发送验证通过。

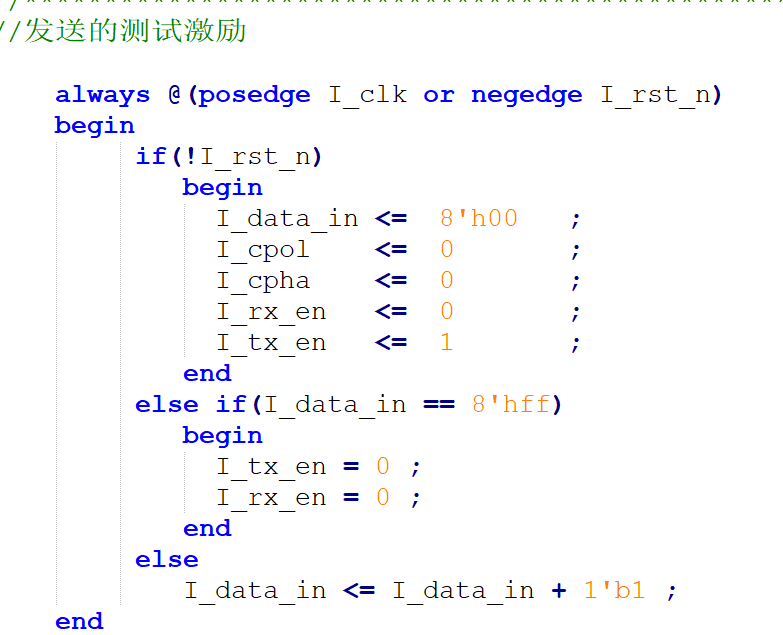


图4.6 单字节发送激励

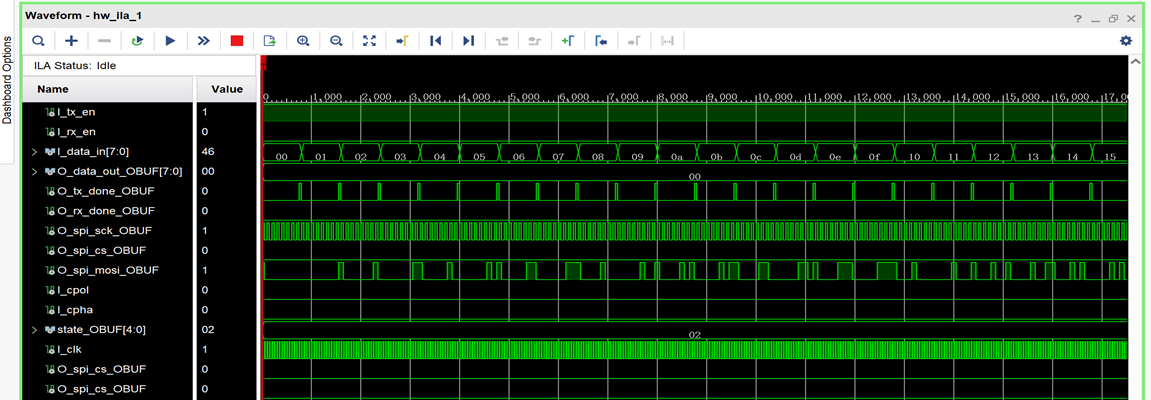


图4.7 数据发送过程中通过ILA捕捉到的片上运行情况

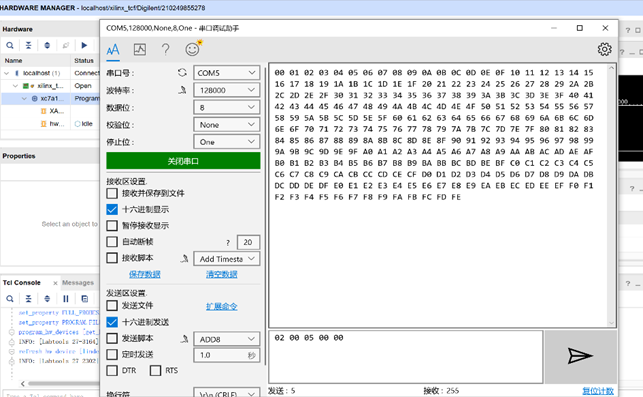


图4.8 通过串口调试助手正确接收到的单段数据

可以看到串口调试助手上接收到的数据正是我们往接口内发送的数据，即表明该SPI接口在发送单段数据操作下验证通过。

1. **长字节循环发送操作验证**

FPGA主机控制代码如下，向接口内循环发送00~FF数据，如果串口调试助手能正确接收斌显示，则表明长字节循环发送操作验证通过。

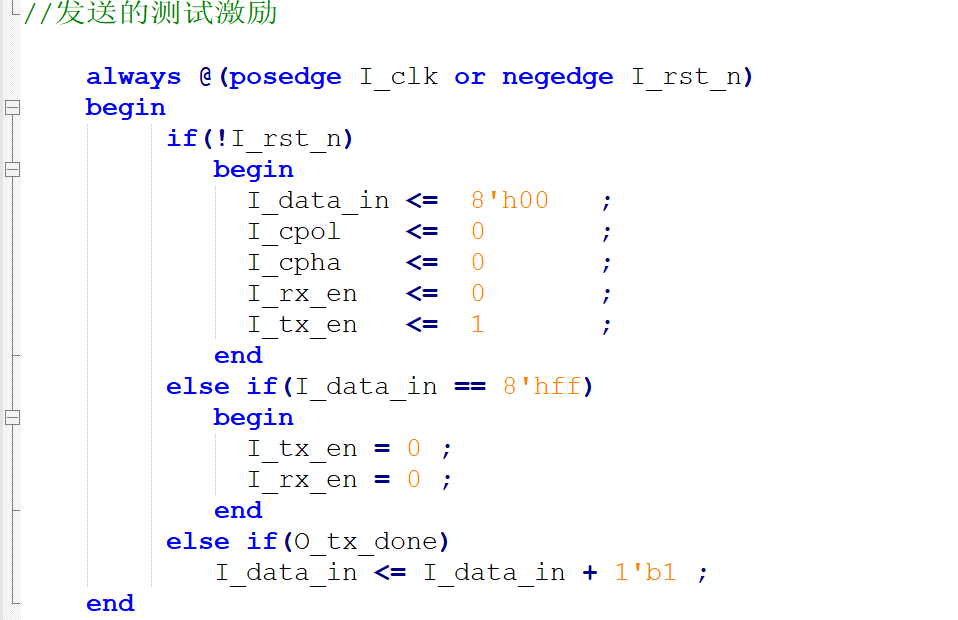


图4.9 长字节循环发送数据激励

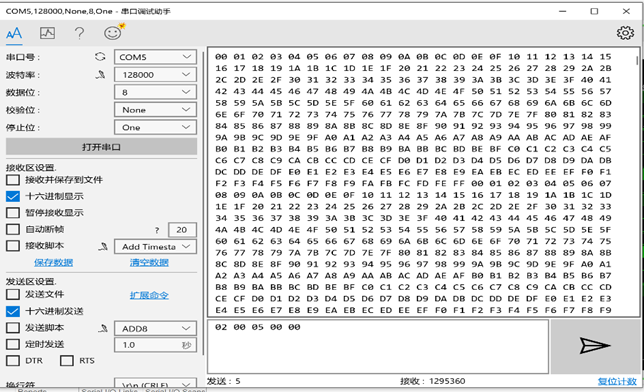


图4.10 通过串口调试助手正确接收到的长字节数据

同样，可以看到串口调试助手上接收到的数据正是我们往接口内发送的数据，即表明该SPI接口在发送长字节数据操作下验证通过。

* 1. **FPGA验证——接受数据操作**

验证思路：

第一步，上位机通过串口调试助手先将数据预存到转接板自带的存储器中。此存储器能被SPI从机接口读。

第二步，FPGA通过我们所设计的SPI主机接口在SPI SCK时钟信号的配合下，能将预存的数据通过SPI从机接口读出，并存储在FPGA的RAM中。

第三步，通过ILA可以探测到存储到RAM中的数据，并将其与我们先前写入转接板的数据进行对比，即可完成我们所设计的SPI接口读数据操作的验证通过。

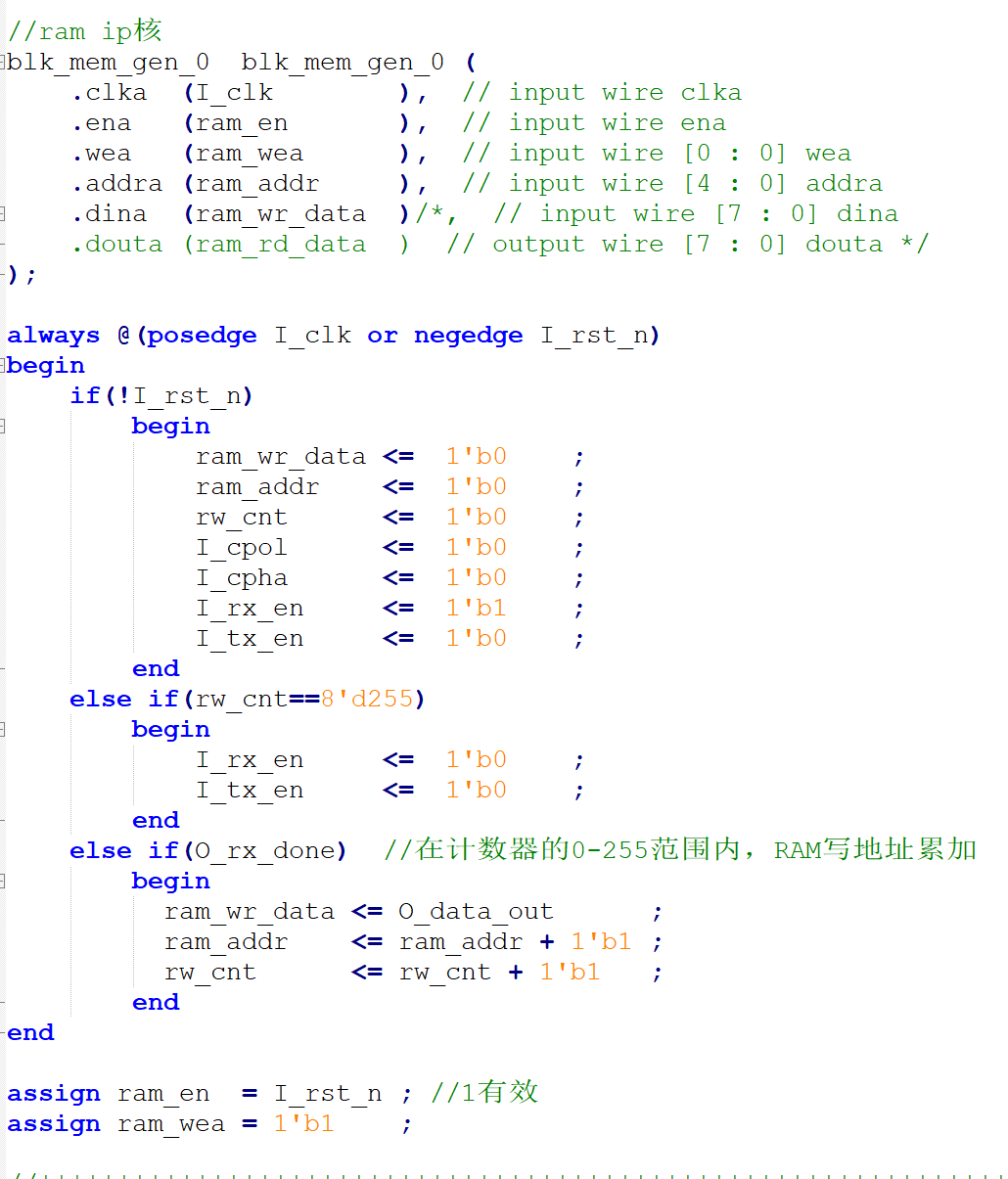


图4.11 FPGA接收数据并将其存入RAM的设计代码

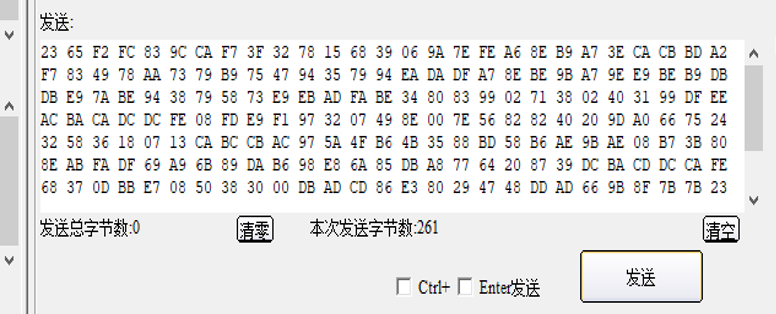
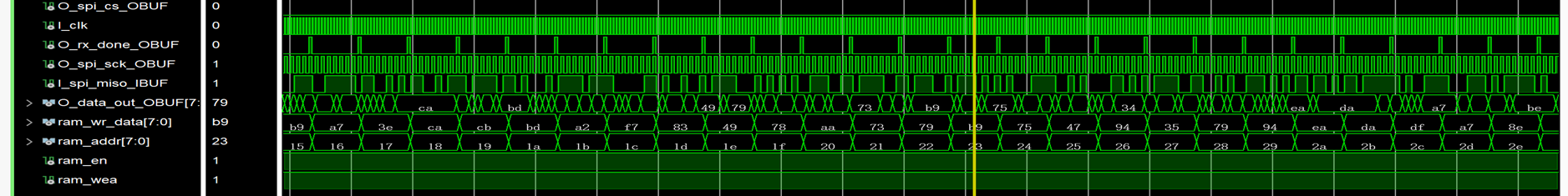
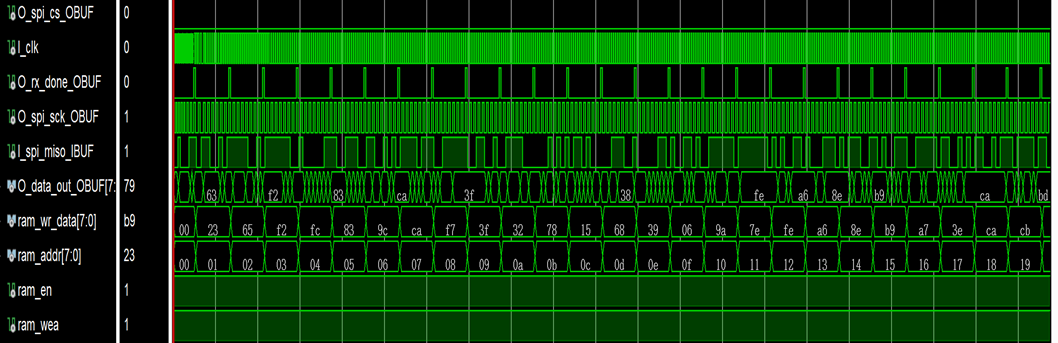


图4.12 连续多字节数据预存



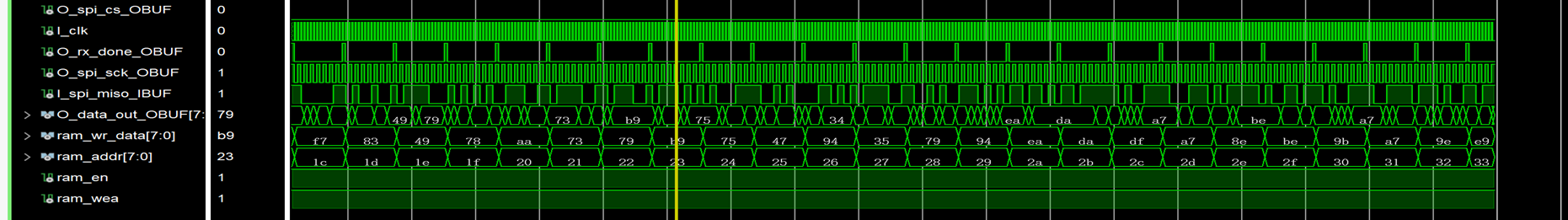


图4.13 ILA捕捉到的存入RAM中的数据信息

通过ILA可以捕捉到存入RAM中的数据信息，可以看到存入RAM的数据正是我们前面预存到转接板等待被读的数据，表明该SPI接口接受数据操作验证通过。

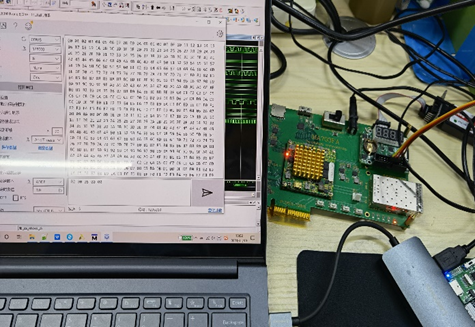


图4.14 FPGA验证线路连接显示