APB_SPI_SLAVE 数据手册

1. 模块结构

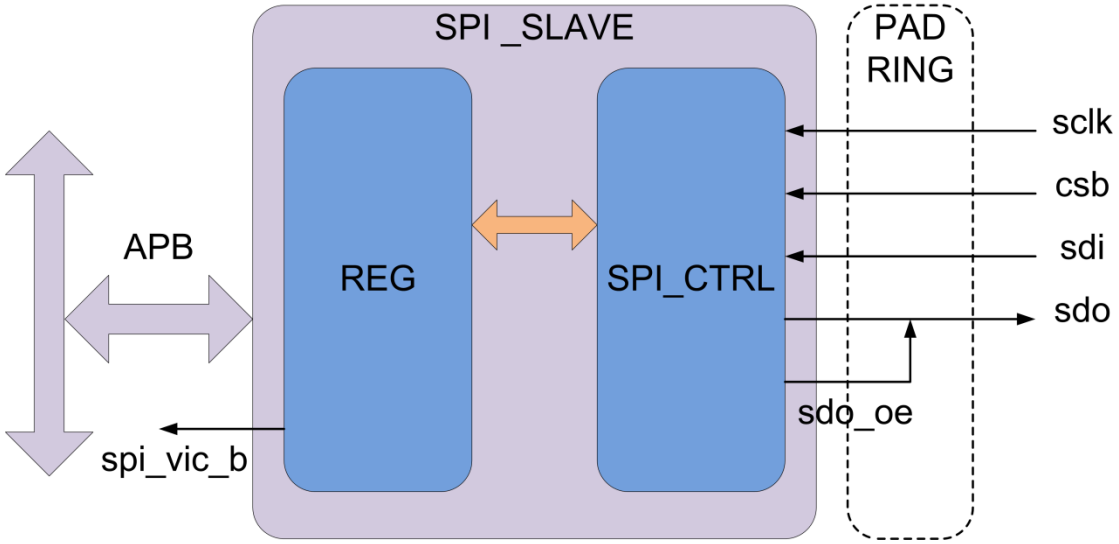

图 1 SPI 结构图

2. 寄存器地址

模块支持弹性数量的寄存器，每个寄存器（1Byte）拥有 SPI 地址和 APB 地址。每次 APB 读写行为同时操作 4 个寄存器，即 1word。每次 SPI 同时操作 1 个寄存器，即 1Byte。在范例的实现中，设计了 16 个寄存器，其地址在表 1 列出。不支持 APB 非对齐读写，即 APB_PADDR 的低 2bit 必须为 00。

表 1 寄存器地址表（HEX）

APB Addr	SPI Addr			
00000000	03	02	01	00
00000004	07	06	05	04
00000008	0B	0A	09	08
0000000C	0F	0E	0D	0C

3. SPI 总线

3.1. SPI 信号描述

表 2 SPI 接口信号

Signal	Source	Description
sclk	master	SPI clock. CPOL=0, CPHA=0.
csb	master	Chip select. Avtive Low.
sdi	master	MOSI
sdo	slave	MISO
sdo_oe	slave	sdo 信号对应数字 PAD 输出使能

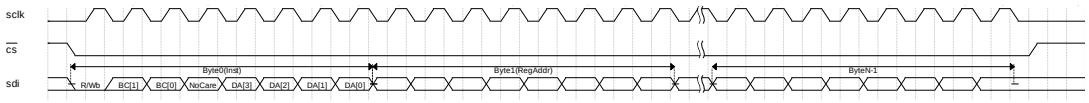

图 2 SPI 写时序

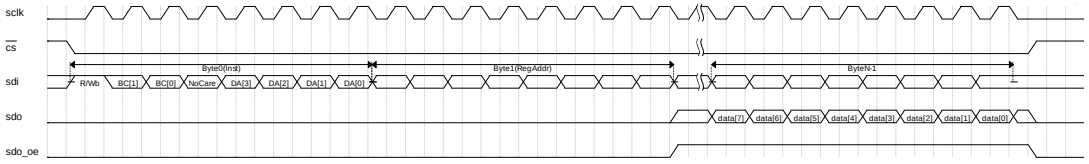

图 3 SPI 读时序

表 3 SPI 指令字节结构

Byte0(Inst)			
[7]	R/Wb	Read/Write_b	1: Read Reg; 0: Write Reg
[6]	BC[1]	Byte Count[1:0]	N= Byte Count +3
[5]	BC[0]		
[4]	No Care	Don't Care	
[3]	DA[3]	Device Address[3:0]	SPI slave device address
[2]	DA[2]		
[1]	DA[1]		
[0]	DA[0]		

在一个有效的 CS 脉冲内，支持多字节（N=3:6）传输。每字节中 MSB first。其中第 0 字节为指令 Inst，第 1 字节为寄存器地址 Reg Addr。第 2:N-1 字节为数据。字节数由指令的[6:5]位即 Byte Count[1:0]决定，N= Byte Count+3。第 2 字节操作的寄存器为地址 Device Address 的寄存器；如有多寄存器操作，即 Byte Count!=00，则第 3 字节操作的是地址 Device Address-1 的寄存器，同理，往后的寄存器地址依次递减。如为写操作，则 sdi 的第 2:N-1 字节为需要写入地址依次为 Reg Addr: Reg Addr- Byte Count 的数据。如为读操作，则 sdi 为 don't care，sdo 的第 2:N-1 字节为读出的地址依次为 Reg Addr: Reg Addr- Byte Count 的数据。

4. 中断系统

中断信号 spi_vic_int，1 表示有中断，0 表示无中断。SPI_SLAVE 模块有 1 种中断，即 SPI 写中断。当 SPI 发起一项写操作时，中断被拉起。中断消除的条件是通过 APB 读任意寄存器。

5. 典型范例

本节展示了与所提供的的 HDL 代码和 testbench 对应的模块描述。二次开发者可基于此案例进行面向实际应用场景的开发。5.2 节展示了与代码对应的仿真波形，按时间顺序。

5.1. 基本设置

5.1.1. 寄存器初始值设置

表 4 范例代码寄存器初始设置

APB Addr(HEX)	SPI Addr(HEX)	data(BIN)	data(HEX)
0000_0000	00	00000000	00

	01	00010001	11
	02	00100010	22
	03	00110011	33
0000_0004	04	01000100	44
	05	01010101	55
	06	01100110	66
	07	01110111	77
0000_0008	08	10001000	88
	09	10011001	99
	0A	10101010	AA
	0B	10111011	BB
0000_000C	0C	11001100	CC
	0D	11011101	DD
	0E	11101110	EE
	0F	11111111	FF

5.1.2. 其他设置

SPI device Address 设置为 4'h5。

SPI 时钟频率 10kHz。

系统时钟频率 5MHz。

5.2. 仿真波形

5.2.1. APB 读写

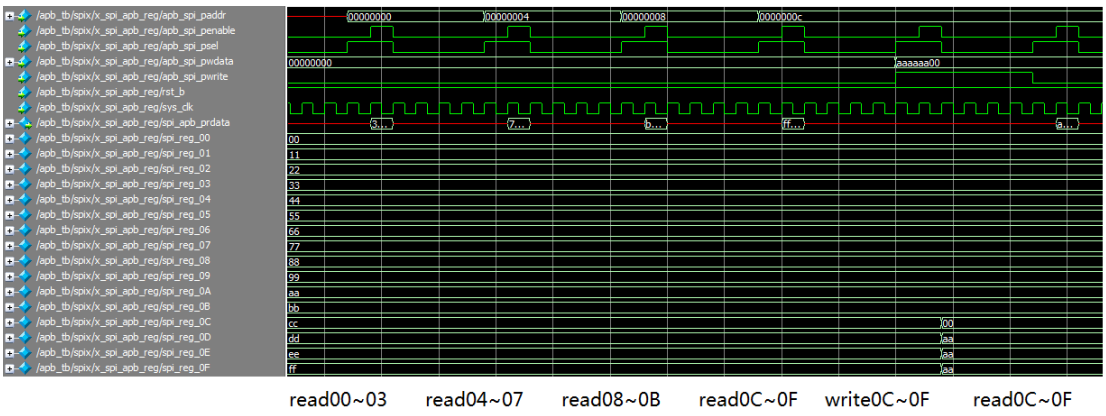

图 4 APB 读写

依次地，用 APB 读地址 32'h00000000 输出 32'h33221100，

读地址 32'h00000004 输出 32'h77665544，

读地址 32'h00000008 输出 32'hBBAA9988，

读地址 32'h0000000C 输出 32'hFFEEDDCC。

然后，用 APB 给地址 32'h0000000C 的寄存器写 32'hAAAAAAAA，可以看到

寄存器 0C 的值从 8'hCC 变成了 8'h00，

寄存器 0D 的值从 8'hDD 变成了 8'hAA，

寄存器 0E 的值从 8'hEE 变成了 8'hAA，

寄存器 0F 的值从 8'hFF 变成了 8'hAA.

再接着，用 APB 读地址 32’h0000000C 的寄存器，输出 32’hAAAAAA00。

5.2.2. SPI 读写

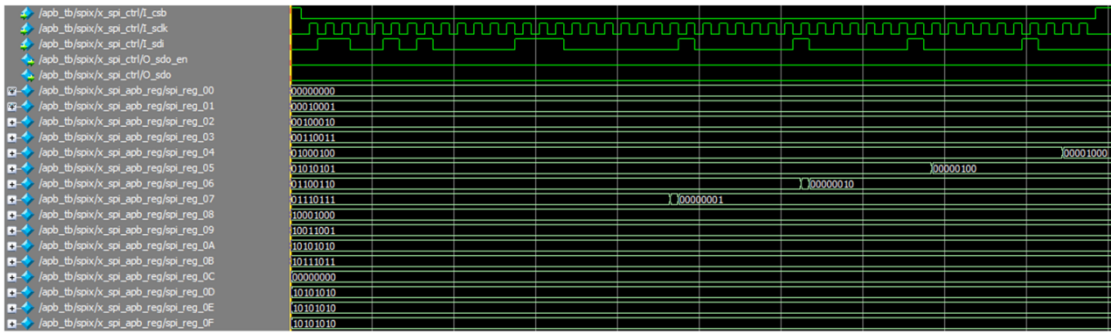

图 5 SPI 写

SPI 总线发送数据串：01100101_00000111_00000001_00000010_00000100_00001000。即对地址 07 及其向低位的总计 4 个寄存器写 01、02、04、08，依次地。在波形中可以看到，寄存器的值依次地发生了变化。

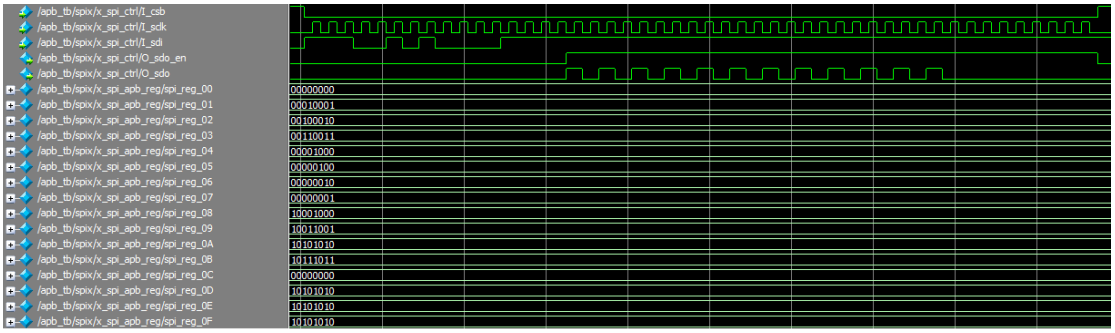

图 6 SPI 读

SPI 总线发送数据串：11100101_00001111_11111111_11111111_11111111_11111111。即读地址 0F 及其向低位的总计 4 个寄存器。sdo_en 在指令字节和地址字节无效。在输出数据的 4 字节有效。sdo 输出的值与寄存器相符合，依次为 10101010_10101010_10101010_00000000。

5.2.3. 再次 APB 读

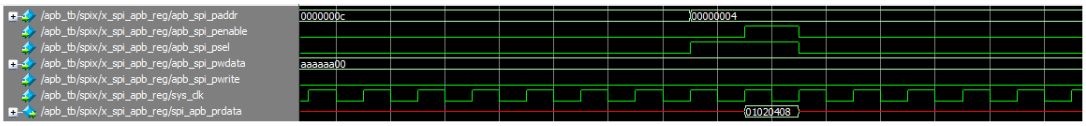

图 7 再次 APB 读

通过 APB 读上文中通过 SPI 改变了的寄存器，可以读出更新后的数据。

5.2.4. 中断系统

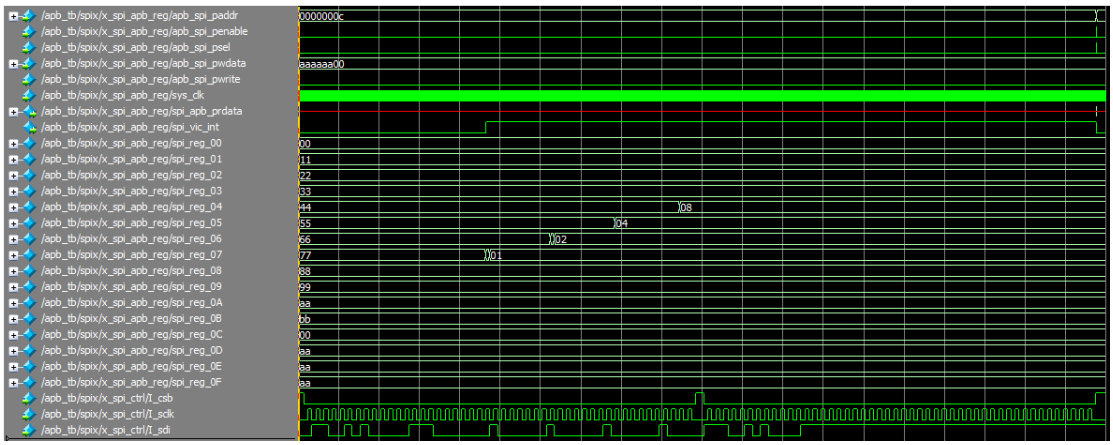

图 8 中断时序

在 5.2.2 的 SPI 写后，中断信号被拉起，直到 5.2.3 中 APB 读寄存器为止。