Chiplet SPI IP User Guide

Revision 1.0

4 January 2022

Table of Contents

REVISION HISTORY IV

1. CHIPLET SPI INTELLECTUAL PROPERTY (IP) INTRODUCTION 1

2. CHIPLET SPI IP FUNCTIONAL DESCRIPTION 2

2.1 Input/Output Ports 3
2.1.1 IO Ports for SPI Leader 3
2.1.2 IO Ports for Follower 4

2.2 SPI Mode Configuration 5

2.3 Leader IP and Follower IP Functional Diagram 6

3. SPI PROGRAMMING 7

3.1 SPI Operation Overview 7
3.1.1 Reads and Writes of the Leader Registers 7
3.1.2 Configuring the SPI System 7
3.1.3 Writes to a Target 7
3.1.4 Reads from a Target 7

3.2 Leader Register Descriptions 8

3.3 Write to Follower Register 8

3.4 Read from Follower Register 10

3.5 Large Message Transfer Programming 11

3.6 Follower Message Formats 11

3.7 Follower Commands 12

3.8 Follower Configuration Register Descriptions 12

3.9 Follower Command Detailed Operation 13

3.10 Auto Write Sequence 15

3.11 Auto Read Sequence 16

3.12 Write to Target Sequence 18

3.13 Read from Target Sequence 19

3.14 Follower AIB Application Configuration 20

4. ERROR HANDLING AND DEBUGGING 22

5. EXAMPLE DESIGN 23

6. REFERENCES 24

List of Figures

Figure 1. Chiplet SPI IP Connection Example 1
Figure 2. Leader and Follower IP in an Example AIB Chiplet System 2
Figure 3. Example Chiplet SPI System with AIB 3
Figure 4. Chiplet SPI IP Supports SPI Mode0 6
Figure 5. Leader and Follower Interaction Diagram 6
Figure 6. Leader to Follower Register Setup Programming Sequence 9
Figure 7. Leader Read Programming Sequence 11
Figure 8. Initiator Auto Write to AVMM Target 15
Figure 9. Auto Read for Follower Programming Sequence 16
Figure 10. Initiator Auto Read from AVMM Target 17
Figure 11. Initiator Write to AVMM Target 18
Figure 12. Initiator Read from AVMM Target 19
Figure 13. Follower AVMM Interface 21
Figure 14. Example Design Testbench in Github 23

List of Tables

Table 1. SPI Leader IO Ports 4
Table 2. SPI Follower IO Ports 5
Table 3. SPI Leader Configuration Register Descriptions 8
Table 4. Follower Commands 12
Table 5. Follower Configuration Register Descriptions 13
Table 6. Follower Commands Detail 14
Table 7. AVMM Address Map for Accessing a 24 Channel AVMM Stack 22

Revision History

Revision	Date	Author	Summary of Changes
0.1	12/13/21		Initial draft
1.0	12/31/21		Update of figures and descriptions

1. Chiplet SPI Intellectual Property (IP) Introduction

The Serial Peripheral Interface (SPI) is a widely used interface between microcontrollers and peripherals. It uses separate clock and data lines, and uses a select line to choose the chiplet device that is active. SPI is a synchronous, full duplex leader-follower-based interface. The data exchange between the SPI Leader (formerly master) and the SPI Follower (formerly slave) is synchronized on the rising or falling clock edge. Both leader and follower can transmit data at the same time.

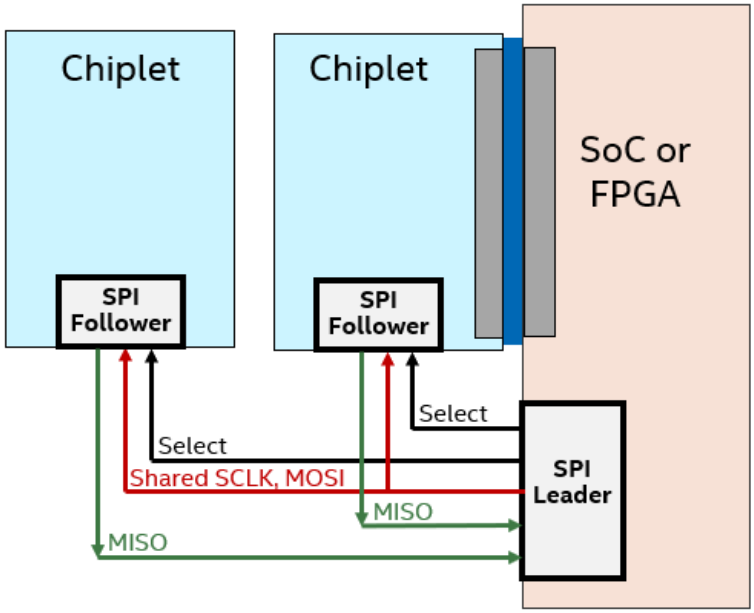

Figure 1. Chiplet SPI IP Connection Example

In an AIB chiplet subsystem, the Chiplet SPI IP enables inter-chip communication between a Leader and one or more Followers. The Chiplet SPI IP cores are a very versatile and important part of the control plane of the Chiplet/SoC system. Figure 1 shows how a SPI Leader and SPI Followers are used in a system. The SPI Leader can be implemented in an Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA). The SPI Leader can be used to program multiple SPI Followers.The SPI Followers can program different ASIC devices. The open source testbench provides an example design, shown in Figure 2, showing how a user would implement the SPI IP in the AIB chiplet subsystem.

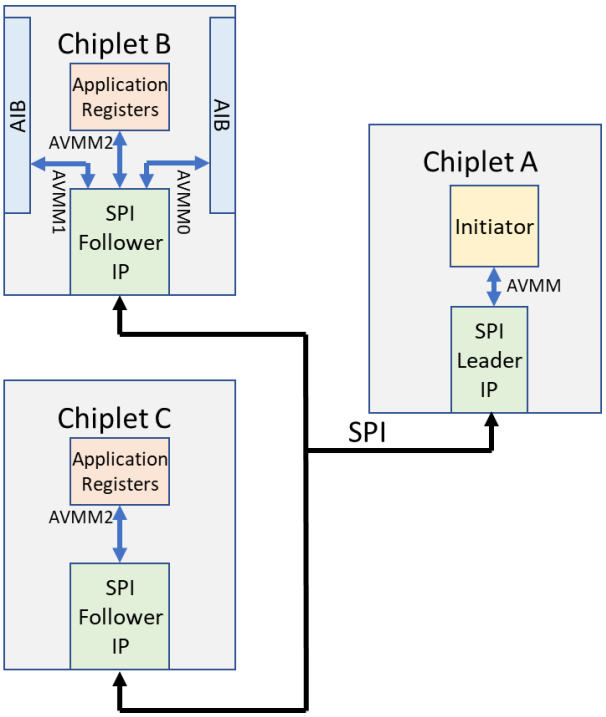

Figure 2. Leader and Follower IP in an Example AIB Chiplet System

2. Chiplet SPI IP Functional Description

This document describes both the Chiplet SPI Follower IP (“Follower”) and Chiplet SPI Leader IP (“Leader”). The Follower maps serial SPI commands, either to Follower internal register writes and reads, or to target write/read ports. The write/read ports are shown as Avalon Memory Mapped interfaces AVMM0, AVMM1 and AVMM2 in Figure 2 above.. The Leader IP does the complement: it maps a parallel write/read port (also the AVMM port) to serial SPI commands. In Figure 3, the left side Chiplet B uses the Follower IP and the right side Chiplet A shows the Leader IP.

A basic chiplet system consists of one chiplet with the Leader IP and one or more chiplets with the Follower IP. On the Follower chiplet(s), the Follower IP is used to configure and control the Advanced Interface Bus (AIB) interfaces and the application registers on the chiplet. In this example, one of the ports on the Follower IP is connected with the AVMM0 port to the AIB interface. The Follower IP has a total of three AVMM ports. Figure 3 shows two additional AVMM ports on the Follower IP that are available to connect to other chiplet targets, such as an additional AIB interface or to application registers.

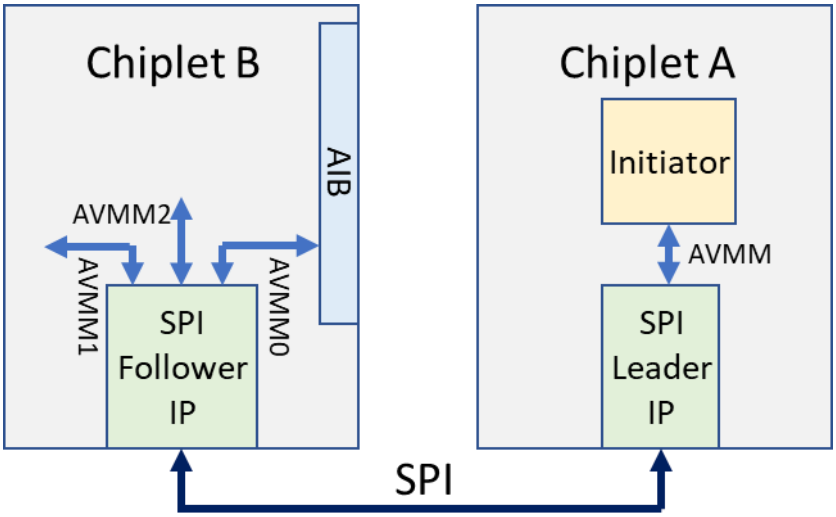

Figure 3. Example Chiplet SPI System with AIB

The Follower can be driven by a Leader from another chiplet such as an ASIC, System on Chip (SoC) or FPGA. The Leader IP itself is controlled by logic inside the same chiplet through an AVMM interface; this logic is referred to in Figure 2 as the Initiator. In an Intel FPGA example, the Initiator may be a NIOS soft IP microcontroller.

2.1 Input/Output Ports

2.1.1 IO Ports for SPI Leader

Leader IO Ports	In/Out	Width	Description
SPI Leader Interface			
sclk	out	1	SPI serial clock from SPI leader. The SPI IP operates with a continuous sclk.
ss_n[3:0]	in	2	Follower select. Active low means the follower access is enabled. High means the follower is not enabled. The SPI Leader can support up to four SPI Followers.
mosi	out	1	Leader out, follower in
miso[3:0]	in	4	Leader in, follower out. The SPI Leader accepts separate miso lines for up to four SPI Followers.
AVMM Follower interface			
avmm_clk	in	1	avmm clock(Free running clock. IP design should not assume the relationship of sclk and avmm_clk. They are asynchronous)
avmm_rst_n	in	1	avmm reset
avmm_addr	in	17	AIB channel access addressing. Bit [16:11] channel ID, [10:0] for offset in byte
avmm_byte_en	in	4	Byte enable
avmm_write	in	1	avmm write enable
avmm_read	in	1	avmm read enable
avmm_wdata	in	32	avmm write data

Leader IO Ports	In/Out	Width	Description
avmm_rdatavld	out	1	avmm read valid when asserted to ‘1’
avmm_rdata	out	32	avmm read data, valid when avmm_rdatavld is ‘1’
avmm_waitreq	out	1	avmm follower ready signal. When this is ‘0’, the follower is ready.
System Input			
spi_clk_in	input	1	Clock input used for SPI master clock out.
rst	input	1	Asyncronous reset
Parameter			
WR_BUFFER_SIZE			Buffer depth in DWORD (32 bit) (current default setting is 512)
RD_BUFFER_SIZE			Buffer depth in DWORD (current default setting is 512)

Table 1. SPI Leader IO Ports

2.1.2 IO Ports for Follower

Follower IO Port	In/Out	Width	Description
SPI Follower Interface			
sclk	in	1	SPI serial clock from SPI Leader. The SPI IP operates with a continuous sclk.
rst	in	1	Async system reset
ss_n	in	1	SPI Follower select. Active high means the SPI Follower access is in progress. Low means the SPI Follower is idle.
mosi	in	1	SPI Leader out, SPI Follower in
miso	out	1	SPI Leader in, SPI Follower out. The SPI Follower IP does not support tristating the miso. Each SPI Follower needs a separate miso connection to the SPI Leader. This protocol eases system implementation and eliminates the possibility of SPI Follower driver conflict.
AVMM common clock and reset			
avmm_clk	in	1	avmm clock (Free running clock. IP design should not assume the relationship of sclk and avmm_clk. They are asynchronous)
avmm_rst	in	1	avmm reset
AVMM leader interface 0			
avmm0_addr	out	17	AIB channel access addressing. Bit [16:11] channel ID, [10:0] for offset in byte
avmm0_byte_en	out	4	Byte enable
avmm0_write	out	1	avmm write enable
avmm0_read	out	1	avmm read enable
avmm0_wdata	out	32	avmm write data
avmm0_rdatavld	in	1	avmm read valid when asserted to ‘1’

Follower IO Port	In/Out	Width	Description
avmm0_rdata	in	32	avmm read data, valid when avmm_rdatavld is ‘1’
avmm0_waitreq	in	1	avmm follower ready signal. When this is ‘0’, the SPI Follower is ready.
AVMM leader interface 1			
avmm1_addr	out	17	AIB channel access addressing. Bit [16:11] channel ID, [10:0] for offset in byte
avmm1_byte_en	out	4	Byte enable
avmm1_write	out	1	avmm write enable
avmm1_read	out	1	avmm read enable
avmm1_wdata	out	32	avmm write data
avmm1_rdatavld	in	1	avmm read valid when asserted to ‘1’
avmm1_rdata	in	32	avmm read data, valid when avmm_rdatavld is ‘1’
avmm1_waitreq	in	1	avmm follower ready signal. When this is ‘0’, the SPI Follower is ready.
AVMM leader interface 2			
avmm2_addr	out	17	AIB channel access addressing. Bit [16:11] channel ID, [10:0] for offset in byte
avmm2_byte_en	out	4	Byte enable
avmm2_write	out	1	avmm write enable
avmm2_read	out	1	avmm read enable
avmm2_wdata	out	32	avmm write data
avmm2_rdatavld	in	1	avmm read valid when asserted to ‘1’
avmm2_rdata	in	32	avmm read data, valid when avmm_rdatavld is ‘1’
avmm2_waitreq	in	1	avmm follower ready signal. When this is ‘0’, the SPI Follower is ready.
Parameter			
WR_BUFFER_SIZE			Buffer depth in DWORD (current default setting is 512)
RD_BUFFER_SIZE			Buffer depth in DWORD (current default setting is 512)

Table 2. SPI Follower IO Ports

2.2 SPI Mode Configuration

Of the four modes possible in a SPI implementation, the Chiplet SPI IP only supports Mode0. In this mode, both Leader and Follower operate with a continuous sclk.

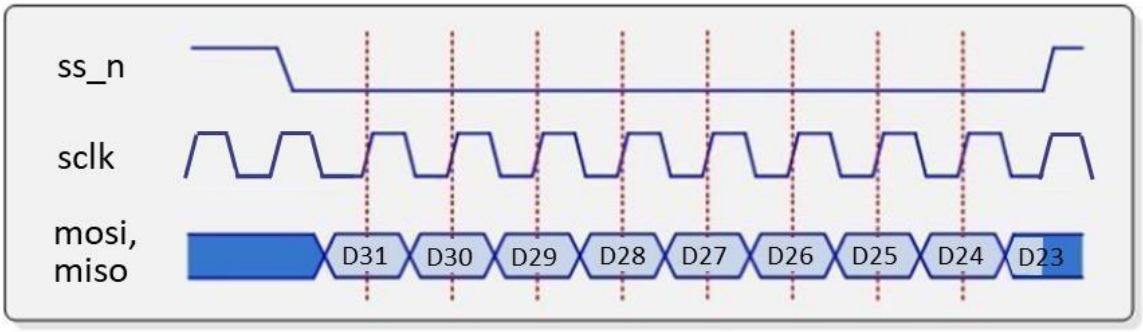

Figure 4. Chiplet SPI IP Supports SPI Mode0

The Chiplet SPI IP supports DWORD (32 bit) transfers. The total number of write or read data transfers is a multiple of 32 bits. For each DWORD, the MSB bit will be transferred first: bit 31, then bit 30, and finally bit 0.

2.3 Leader IP and Follower IP Functional Diagram

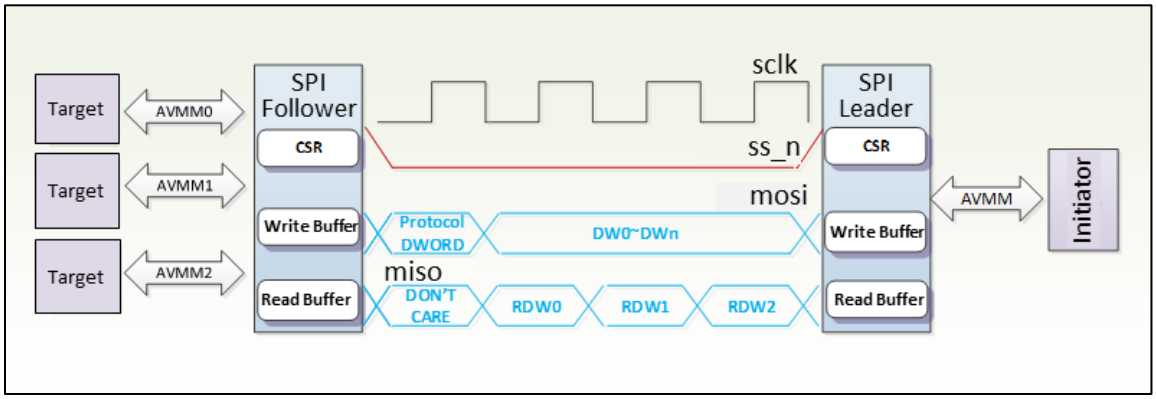

Figure 5. Leader and Follower Interaction Diagram

The Leader IP and Follower IP have similar functional components and interfaces. The Leader IP has an AVMM Follower interface that the user should connect to an Initiator's AVMM Leader interface. Internally, the Leader IP uses a command and status register (CSR) submodule along with Write Buffer and Read Buffer submodules. At the Leader, the content to and from the SPI bus is determined by the Initiator and the Follower.

The Follower has CSR registers that may be programmed over the SPI bus by the Leader. On the Follower, a Write Buffer may be accessed via the SPI bus by the Leader. The Write Buffer may be sent to any of the Follower's AVMM0, AVMM1 or AVMM2 interfaces. Also on the Follower, the Read Buffer may be accessed by any of the Follower's AVMM0, AVMM1 or AVMM2 interfaces. The Read Buffer may be sent via the SPI bus to the Leader.

The Follower IP decodes the command from the Leader on the mosi signal. The miso signal from the Follower sends data back to the Leader if directed by the Leader's command.

3. SPI Programming

3.1 SPI Operation Overview

A SPI system as shown in Figure 3 is controlled by the Initiator, which is hardware or a microcontroller in one of the chiplets. The Initiator directs a sequence of operations to execute writes and reads with targets attached to the Follower.

3.1.1 Reads and Writes of the Leader Registers

The Initiator may directly read and write Leader registers through the Leader's AVMM interface. The Leader register addresses and fields are described in Table 3.

3.1.2 Configuring the SPI System

1. The Initiator configures the Leader by writing the Leader's local registers using the Leader's AVMM interface.
2. The Initiator selects which Follower to communicate to by writing the Command Register in the the Leader.
3. The Initiator configures the Follower by composing Register Write commands into the Leader Write Buffer and then triggering the Leader to send the Write Buffer to the Follower. The Follower interprets the received SPI data as a Register Write command and stores the value in a local register within the Follower IP.

3.1.3 Writes to a Target

1. The Initiator selects which target attached to the Follower to communicate with (e.g. AVMM0, AVMM1 or AVMM2) by composing a Register Write command into the Leader Write Buffer and then triggering the Leader to send the Write Buffer to the Follower. The Initiator's Register Write command is addressed to the Follower Command Register0 which selects the target.
2. The Initiator composes an Auto Write command in the Leader Write Buffer, along with data to be sent to the target. The Initiator triggers the Leader to send the Write Buffer to the Follower. The Follower interprets the received SPI data as a Auto Write command, and sends the write data to the target. "Auto Write" means the write data is sent to the target with this single command from the Leader.

3.1.4 Reads from a Target

1. The Initiator selects which target attached to the Follower to communicate with (e.g. AVMM0, AVMM1 or AVMM2) by composing a Register Write command into the Leader Write Buffer and then triggering the Leader to send the Write Buffer to the Follower. The Initiator's Register Write command is addressed to the Follower Command Register0 which selects the target. If the target is already programmed into the Follower Command Register0, this step may be skipped.
2. The Initiator composes an Auto Read command in the Leader Write Buffer. The Initiator triggers the Leader to send the Write Buffer to the Follower. The Follower interprets the received SPI data as a Auto Read command, and reads from the target and returns the read data over SPI to the Leader. "Auto Read" means the read data is read from the target and returned to the Leader with this single command from the Leader.

3. The Initiator reads from the Leader's read buffer using the Leader's AVMM interface.

3.2 Leader Register Descriptions

Register Name	Offset	Bit	R/W	Field Name	Use Description
Command Register	0x0	31:30	RW	Follower Select	When '00' SPI Follower 0 is selected. When '01' SPI Follower 1 is selected. When '10' SPI Follower 2 is selected. When '11' SPI Follower 3 is selected.
		29:23	RO		Reserved
		22:16	RO		Reserved
		15:2	RW	Burst Length	Specifies the burst length-1 in DWORDs when ss_n goes low to ss_n goes high
		1	RW	rdnwr	1 = read transaction 0 = write transaction This bit is only used by Firmware to remember what kind of transaction. SPI Leader always does bidirectional transaction and does not use this bit.
		0	RW	trans_valid	1 = SPI bus is transferring data (read) or Command SPI Leader to transfer data (write) 0 = SPI bus is idle
Status Register	0xc	31:0	RO		Reserved
Diag Register 0	0x10	31:0	RO		Reserved
Diag Register 1	0x14	31:0	RO		Reserved
Write Buffer	0x200-0xfff	31:0	WO		Used for generic data transfer from SPI Leader to SPI Follower. The content of the data is understood by SPI Follower decoding and firmware. Random access the content of write buffer is not supported. For a new transaction always start at the address 0x200.
Read Buffer	0x1000-0x17ff		RO		Used for generic data transfer from the SPI Follower to SPI Leader. The content of the data is understood by SPI Follower and firmware. Random access to the content of the write buffer is not supported. For a new transaction always start at address 0x1000.

Table 3. SPI Leader Configuration Register Descriptions

3.3 Write to Follower Register

The Initiator performs the following steps to program a Leader to Follower write operation:

1. Wait until the trans_valid field of the Command Register bit 0 = 0 to make sure the SPI system is idle.
2. Perform AVMM writes to the Write Buffer for the total data to be transferred to the SPI bus.

3. Set which Follower to write by writing the Command Register Follower Select bits 31:30.
4. Compose the Leader Command Register value. Set Command Register bit 1 rdnwr=0 for a write operation. Set Command Register's Burst Length bits 15:2 to the burst length-1 for the total burst DWORDs on the SPI bus. Set the trans_valid bit to 1. This triggers the Leader to send the command and write data across the SPI bus.
5. Keep checking bit 0 of the Command Register. When it drops to zero, then the write procedure is finished. The SPI bus is idle.

The Leader does not care about the content for the write operation. Only the Follower uses the content.

Even if the transaction is a write, the Leader will sample miso when ss_n goes active and will store the data into the read buffer. The Initiator may choose to ignore or read from the read buffer.

The following is an example of an Leader write to the Follower to perform register setup in the Follower.

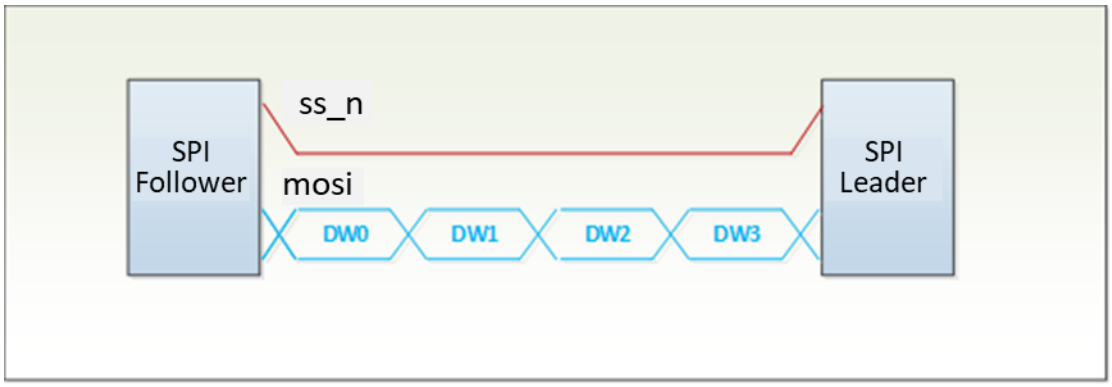

Figure 6. Leader to Follower Register Setup Programming Sequence

The Initiator writes commands for the Follower into the Leader's write buffer. Commands for the Follower are described in Table 4.

```
//Program the SPI leader write buffer. Always start at 17'h200
avmm_if_mspi.cfg_write(17'h200, 4'hf, 32'h1010_0000);
avmm_if_mspi.cfg_write(17'h204, 4'hf, 32'h0080_0200);
avmm_if_mspi.cfg_write(17'h208, 4'hf, 32'h0017_0800);
avmm_if_mspi.cfg_write(17'h20c, 4'hf, 32'hdead_beef);

//Program SPI leader command register including setting bit 0
//to 1'b1 to start the transaction
avmm_if_mspi.cfg_write(17'h000, 4'hf, 32'h0000_000d);

//Keep polling SPI leader command register bit 0 until it goes to 0
rdata_reg[0] = 1'b1;
while (rdata_reg[0] !== 1'b0) begin
   avmm_if_mspi.cfg_read (17'h000, 4'hf, rdata_reg);
   $display("%0t: cmd polling:  rdata_reg =  %x", $time, rdata_reg);
end
```

Notes:

1. The Leader controls the length of the ss_n signal = LO. In this case, the burst length is 4 DWORD.
2. The transfer is bi-directional for all transactions: miso is always sampled and mosi is always set. The Leader does not need to know whether is it doing a read or write operation. Firmware can take advantage of that for maximum bandwidth usage. Command Register field rdnwr, bit 1, is used for firmware bookkeeping only.

3.4 Read from Follower Register

The following are the steps for the Initiator to program a read by the Leader from the Follower.

1. Wait for Command Register trans_valid (bit 0) = 0 to make sure the SPI interfaces are idle.
2. Select which Follower to read from by setting Command Register Follower Select bits 31:30.
3. Perform AVMM writes into the Leader's write buffer to set up a read request write transaction.
4. Compose the Leader Command Register value. Set the Command Register rdnwr bit 1 for write operation and set Write Burst Length bits 15:2 to the burst length-1 for the total burst word DWORDs on SPI bus. Set Command Register trans_valid to 1. The Leader will send the command across the SPI bus and sample the read data from the Follower.
5. Wait until Command Register trans_valid = 0. When it is zero, the read request procedure is finished.
6. Read the contents of the Leader's read buffer using the Leader's AVMM interface.

Note: Since the Leader will send out data from the write buffer on every transaction, the Initiator may set the Leader's write buffer to all zeros before initiating a read transaction to avoid potential confusion.

Figure 7 is an example of an Leader read from the Follower for register setup. This example follows the previous Leader write example and attempts to read back three registers (RDW0, RDW1, and RDW2) from the Follower.

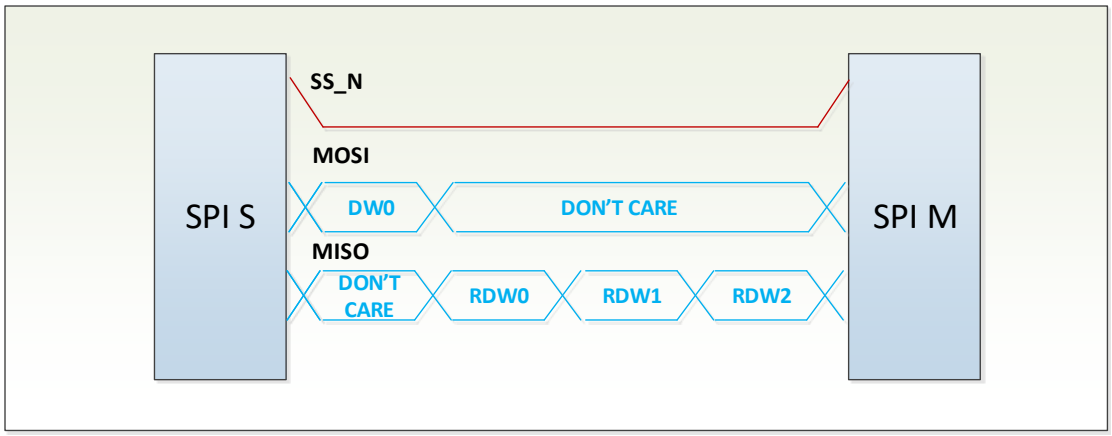

Figure 7. Leader Read Programming Sequence

First, the Leader must setup all the necessary entries to send over the SPI bus to the Follower. Since this is a read procedure, the Leader needs to send only an instruction word to the Follower to indicate a read command, to locate the base address, and to define how many DWORDs to read over SPI.

```
// Programm SPI Leader's write buffer for a Follower read command
avmm_if_mspi.cfg_write(17'h200, 4'hf, 32'h0010_0000);
// Program Leader's command register to trigger the Follower read
// Follower supplies data on miso as long as ss_n is low. Here we
// program Leader to hold ss_n low for 4 DWORDs
avmm_if_mspi.cfg_write(17'h000, 4'hf, 32'h0000_000d);
//Poll SPI Leader's command register bit 0 until it goes to 1'b0
//using a function written for this purpose
leader_polling();
//Read back follower registers from read buffer
//First word is always don't care
avmm_if_mspi.cfg_read (17'h1000, 4'hf, rdata_reg);
avmm_if_mspi.cfg_read (17'h1004, 4'hf, rdata_reg);
// Do something with rdata_reg
avmm_if_mspi.cfg_read (17'h1008, 4'hf, rdata_reg);
// Do something with rdata_reg
avmm_if_mspi.cfg_read (17'h100c, 4'hf, rdata_reg);
// Do something with rdata_reg
```

3.5 Large Message Transfer Programming

The above procedure assumes the write buffer/read buffer can hold the entire message. If the message to be sent by the Leader is bigger than the Follower's write buffer or the read buffer, the Initiator must divide the message into multiple segments.

It is preferable to implement the write buffer and read buffer using simple dual port RAM. Random access of the buffer is not required. The buffer size is parameterized in the Leader and Follower IP.

3.6 Follower Message Formats

Messages to the Follower are composed of 32 bit DWORDs as follows, received on the mosi signal:

DW0: Command CMD[31:28], Burst Length Burstlen[27:19], Address ADDR[18:0] which has the same format as Command Register0[20:2] (see below)

DW1 - DWn: Write data or don't care, depending on the CMD

The Burstlen[27:19] and ADDR[18:0] fields of the message to the Follower are used according to the Command type as described in Table 6. Responses from the Follower are composed of 32 bit DWORDs as follows, sent on the miso signal:

DW0: Dummy DW (see Command Register1 hdr_sel setting in Table 5)

DW1 - DWn: Read data or don't care, depending on the CMD and Command Register1 auto_rd_lat setting in Table 5

3.7 Follower Commands

DW0 in a message to the Follower the CMD[31:28] 4-bit command field. The supported commands are shown in Table 4.

Command	4 bit CMD[31:28] Value	Description
Register Read	0	Follower SPI register write
Register Write	1	Follower SPI register read
Buffer Read*	2	Follower returns data from Follower Read Buffer
Buffer Write*	3	Follower writes data into Follower Write Buffer
Auto Read	6	Read from AVMM target using AIB channel addressing
Auto Write	7	Write to AVMM target using AIB channel addressing
Reserved	4, 5, 8 – 0xF	Reserved, do not use

* The Buffer Read and Buffer Write commands are included primarily for debugging purposes. Reads and writes between the Initiator and the targets may be accomplished without using these commands.

Table 4. Follower Commands

3.8 Follower Configuration Register Descriptions

Registers	offset	bit	R/W	Field Name	Description
Command Register0	0x0	31:30	RO		Reserved
		29:21	RW	avmm_burst_len	AVMM Burst Length in dword -1
		20:19	RW	avmm_sel	Top AVMM address for selecting one of the three AVMM interfaces: 2'b00: Select AVMM0 2'b01: Select AVMM1 2'b10: Select AVMM2 2'b11: Reserved
		18:2	RW	start_addr	17-bit AVMM start address
		1	RW	rdnwr	1 = read transaction 0 = write transaction
		0	RW	trans_valid	When set to '1', AVMM configuration to

Registers	offset	bit	R/W	Field Name	Description
					AIB channel will start. When read value of this bit is '0', transaction finished.
Command Register1	0x4	31:25	R/W		Reserved
		24:23	R/W	auto_rd_lat	Defines how many DWORDs in the auto read back value. 2'b00: two DWORD latency 2'b01: three DWORD latency 2'b10: four DWORD latency 2'b11: five DWORD latency
		22	R/W	hdr_sel	When set to '1', dummy header DWORD for the SPI Follower read is a user-defined value. When set to '0', the cmd_reg0 content will be selected
		21:16	R/W	auto_chan_num	How many channels need to be programmed or read, minus one. Default value is 23 (24 channels).
		15:0	R/W	auto_offset_addr	Incremental address offset of channel during auto mode. Default value is 2048.
Header Register	0x8	31:0	R/W	hdr_reg	User defined dummy header value at the beginning of the read DWORD.
Status Register	0xc		RO		Reserved
Status Register	0xc	31:0	RO		Reserved
Diag Register 0	0x10	31:0	RO		Reserved
Diag Register 1	0x14	31:0	RO		Reserved
Write Buffer	0x200-0xfff	31:0	RW		For any single write, data will be stored in 0x200. Random access of the write buffer is not supported.
Read Buffer	0x1000-0x17ff	31:0	RO		For any single read, data will be stored in 0x1000. Random access of the read buffer is not supported.

Table 5. Follower Configuration Register Descriptions

3.9 Follower Command Detailed Operation

Table 6 describes the details and operation for each Follower command.

CMD	Signal	DW0	DW1	DW2~DWn	Description
Reg Read	mosi	CMD = 4'h0 Burstlen = don't care ADDR = Follower SPI register base addr	don't care	don't care	Single or burst read. The SPI Follower will continue to send register data on miso as long as ssn_is held low. Read latency is one DWORD.
	miso	Dummy DW	register data read	register data read	
Reg Write	mosi	CMD = 4'h1 Burstlen = don't care ADDR = Follower SPI register base addr	register data write	register data write	Single or burst write. The SPI Follower will continue to accept register data on mosi as long as ssn_is held low.
	miso	Dummy DW	read Buffer data	read Buffer data	
Buffer Read	mosi	CMD = 4'h2 Burstlen = don't care ADDR = don't care	don't care	don't care	No address needs to be specified. Always start with beginning of read buffer.
	miso	Dummy DW	read Buffer data	read Buffer data	
Buffer Write	mosi	CMD = 4'h3 Burstlen = don't care ADDR = don't care	write buffer data	write buffer data	No address needs to be specified. Always start with beginning of write buffer.
	miso	Dummy DW	read Buffer data	read buffer data	
Auto Read	mosi	CMD = 4'h6 Burstlen = AVMM burst ADDR = AVMM base	don't care	don't care	Address field includes 2-bit top AVMM interface selection, 6-bit channel ID and 11-bit base AVMM address. This operation will use channel number configuration to replicate multiple channels.
	miso	Dummy DW	default rd0	buffer data depends on latency config	
Auto Write	mosi	CMD = 4'h7 Burstlen = AVMM burst ADDR = AVMM base	write buffer data	write buffer data	Address field includes 2-bit top AVMM interface selection, 6-bit channel ID and 11-bit base AVMM address. This operation will use channel number configuration to replicate multiple channels.
	miso	Dummy DW	read Buffer data	read Buffer data	

Table 6. Follower Commands Detail

3.10 Auto Write Sequence

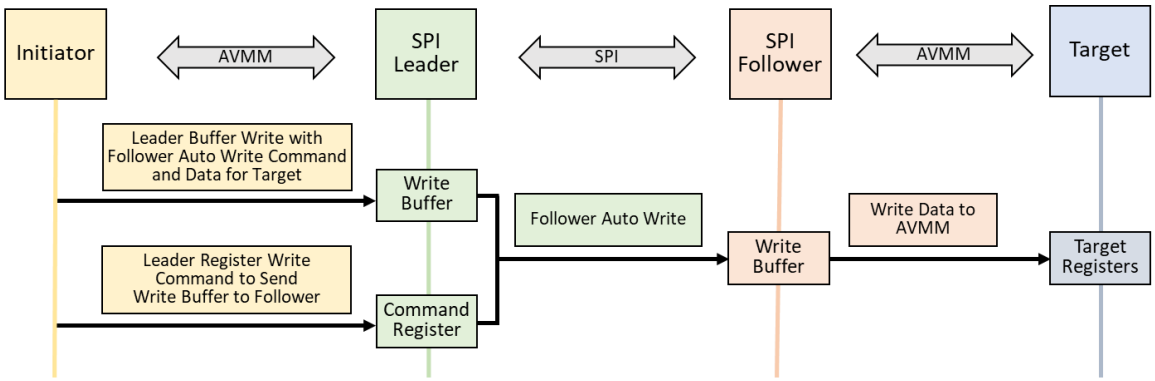

Figure 8. Initiator Auto Write to AVMM Target

The Initiator performs a typical Auto Write sequence as follows:

1. Set up the Follower Command Register1 according to your AVMM target. If the target is an application register block, the fields auto_chan_num and auto_offset_addr are typically 0. If the target is an AIB PHY, set auto_chan_num to the number of channels to be written, minus one. If the target is an AIB PHY, set auto_offset_addr to 0x800.
2. Compose the Auto Write command into the Leader Write Buffer. DW0 contains CMD=4'h7 for Auto Write, Burstlen= number of DWORDs to write minus 1, and ADDR= which AVMM target, which channel ID if appropriate, and the AVMM start address to write to the target. The Initiator writes DW1 though DWn into the Leader Write Buffer as the data to be written to the target. In the code example below, the burst length is three, which means there are four (4) DWORDs for Follower to write into the AVMM target. In the code example below, the target is an AIB PHY. The base address is 0x31C, which denotes the start register address in the AIB PHY.
3. Compose the Leader Command Register setting. Set Leader Command Register bit 1 rdnwr=0 for a write operation. Set Leader Command Register's Burst Length bits 15:2 to 4 for a total of 5 DWORDs to be sent across the SPI bus (the Follower Auto Write command DWORD and the four DWORDs to be written to the target). Set the trans_valid bit to 1. This triggers the Leader to send the Leader Write Buffer's contents over the SPI interface.
4. After the Follower receives the DW0 with the Auto Write command, it will pull the four (4) following DWORDs into its Write Buffer, and then write the four DWORDs to the target. The Follower will next check the Follower Command Register1 field auto_chan_num. If auto_chan_num value plus 1 is more than one, the Follower will repeat writing the four DWORDs for the other channels in the same target.
5. To complete the Auto Write, the Initiator first polls the Leader Command Register until the SPI bus is idle. Then, the Initiator polls the Follower Command Register0 until the Follower's writes to the target are complete.

The following example is taken from the basic_spi_test.inc of the Github release:

```
// Compose the Auto Write command into DW0 of the Leader Write Buffer
//  Burstlen=3 (Four DWORDSs to write to target),
//   ADDR = 0x31C (avmm target 0, addr 0x31C)
```

```
avmm_if_mspi.cfg_write(17'h200, 4'hf, {4'h7, 9'h3, 19'h31c});
// Write the DWORDs to be sent to the target into the Leader Write Buffer
avmm_if_mspi.cfg_write(17'h204, 4'hf, 32'haaaa_bbbb);
avmm_if_mspi.cfg_write(17'h208, 4'hf, 32'hcccc_dddd);
avmm_if_mspi.cfg_write(17'h20c, 4'hf, 32'heeee_ffff);
avmm_if_mspi.cfg_write(17'h210, 4'hf, 32'h5555_6666);
// Compose the Leader Command Register value
// Follower Select=0, Burst Length=4 (Five DWORDs to send over SPI),
//   rdnwr=0 (write), trans_valid=1 (send)
avmm_if_mspi.cfg_write(17'h000, 4'hf, 32'h0000_0011);
// Check if write command/data has been sent to follower
// by polling Leader Command Register bit 0
master_polling();
// Check that the Follower has finished AVMM writes by
//  polling Follower Command Register0 bit 0
follower_polling();
```

3.11 Auto Read Sequence

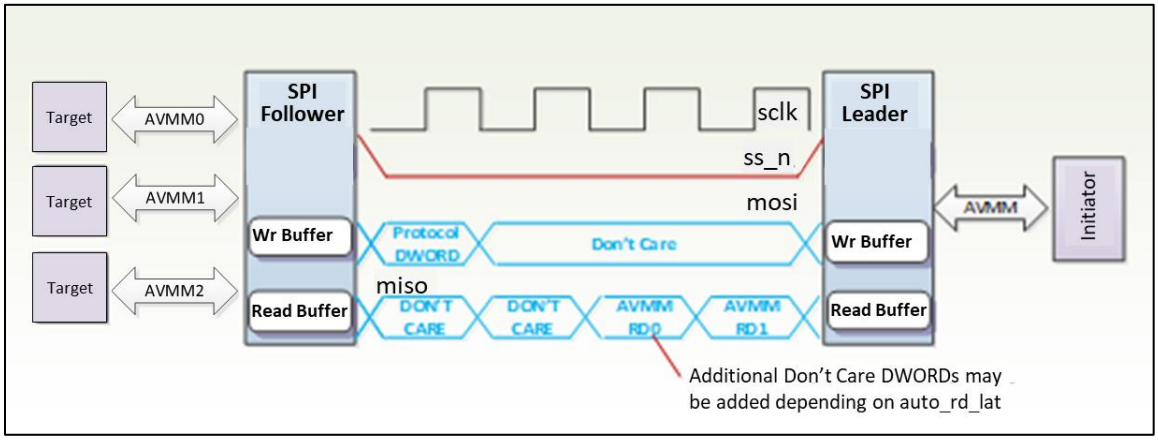

Figure 9. Auto Read for Follower Programming Sequence

The Auto Read command causes the Follower to read from the AVMM target and then to sends the data back to the Leader's Read Buffer all in one SPI transaction.

Because the AVMM clock and SPI sclk are asynchronous, the delay in SPI sclks from the Auto Read command to the read data available is varied. The Follower holds the read data for a specific number of DWORDs to assure that the read data is ready to be sent to the Leader. The Follower Command Register1 auto_rd_lat field is used to set the number of don't care DWORDs until valid read data is on the SPI bus. An auto_rd_lat value of 0 means the third DWORD is the first valid read data.

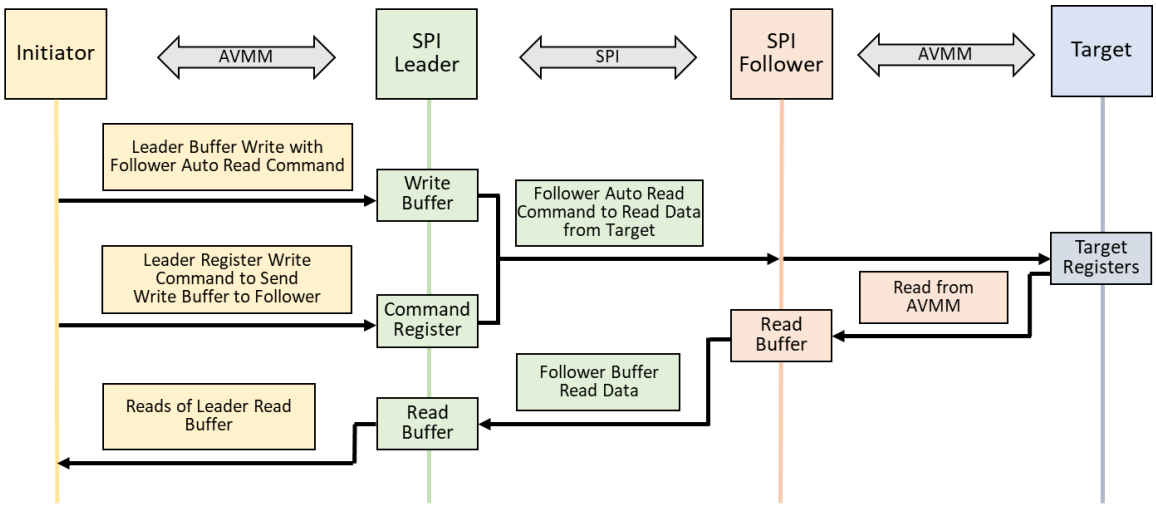

Figure 10. Initiator Auto Read from AVMM Target

The Initiator performs a typical Auto Read sequence as follows:

1. Set up the Follower Command Register1 according to your AVMM target. If the target is an application register block, the fields auto_chan_num and auto_offset_addr are typically 0. If the target is an AIB PHY, set auto_chan_num to the number of channels to be written, minus one. If the target is an AIB PHY, set auto_offset_addr to 0x800. The following example shows a 24-channel four-burst Auto Read. The Initiator programs the Follower Command Register1 field auto_chan_num to 23 to indicate that 24 channels are to be written.
2. Compose the Auto Read command into the Leader Write Buffer. DW0 contains CMD=4'h6 for Auto Read, Burstlen= 3 (4-1) which means there are four AVMM reads per channel. The Follower Auto Read Command field ADDR is set to the which AVMM target, which channel ID if appropriate, and the AVMM base address. In this example, the AVMM target is 0, the channel ID is 0 and the AVMM base address is 0x31C.
3. Compose the Leader Command Register setting. Set Leader Command Register bit 1 rdnwr=1 for a read operation. The Leader Command Register field Burst Length specifies how many DWORDs are in the transaction (how long the ss_n signal goes low). The operation is 24x4 + 2 (Follower Command Register 1 field auto_rd_lat=0 sets a latency of 2 DWORDs until valid read data) DWORDs long, which is a Leader Command Register field Burst Length value of 97, or 0x61. Set the trans_valid bit to 1. This triggers the Leader to send the Leader Write Buffer's contents over the SPI interface.
4. Wait until the Leader is idle, Command Register field trans_valid (bit 0)=0.
5. Perform AVMM reads of the Leader's Read Buffer.

The following example is taken from the basic_spi_test.inc of the Github release:

```
// Compose the Follower Auto Read command: CMD=6, Burstlen=3
//  (4 DWORDs to read from each of the target AIB PHY's channels),
//  ADDR=0x31C
avmm_if_mspi.cfg_write(17'h200, 4'hf, {4'h6, 9'h3, 19'h31c});
// Write to Leader Command Register: Burst Length =
//  (4 * 24 channel)+ 2 dummy DWORDs - 1 = 0x61, rdnwr=0 (write),
//  trans_valid=1 (send)
```

```
avmm_if_mspi.cfg_write(17'h000, 4'hf, {16'h0, 14'h61, 2'h1});
master_polling ();              // Wait until idle
// Initiator can now read all 4x24 data from the Leader Read Buffer
for (int i=0; i<13; i++) begin
       avmm_if_mspi.cfg_read((17'h1000 + i*4), 4'hf, rdata_reg);
       // do something with rdata_reg
end
```

3.12 Write to Target Sequence

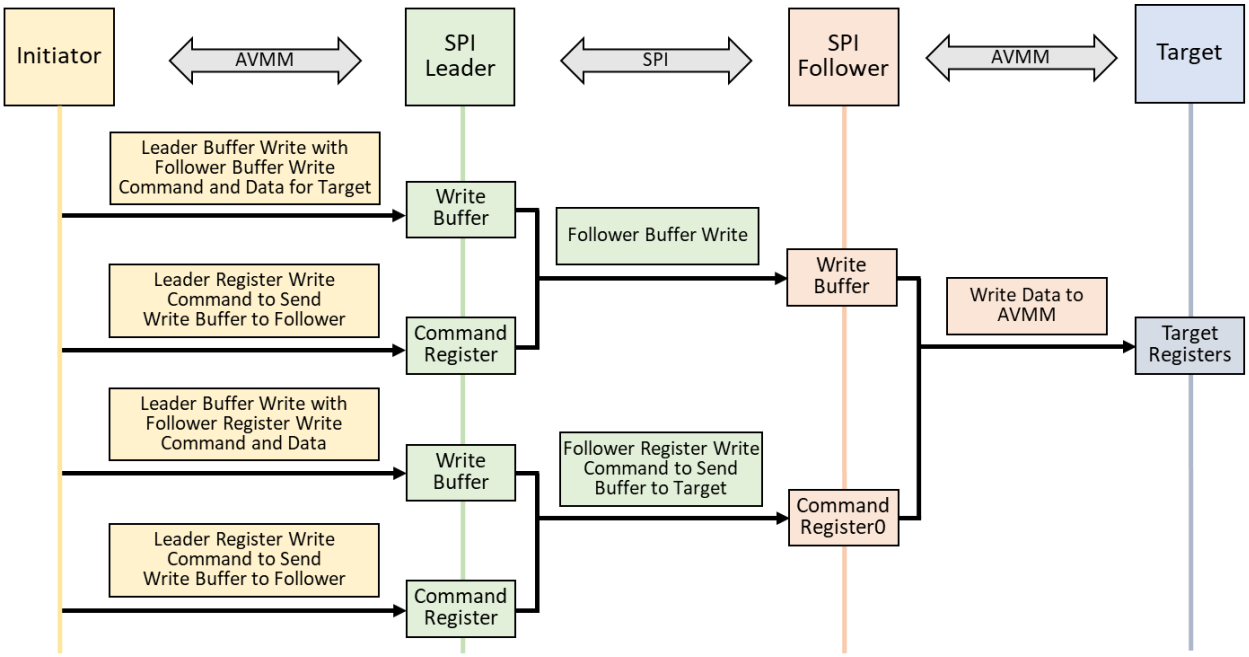

Figure 11. Initiator Write to AVMM Target

The Initiator sets up the Follower's write buffer with data meant for the target:

1. Wait until the trans_valid field of the Leader Command Register bit 0 trans_valid=0 to make sure the SPI system is idle.
2. Perform an AVMM write to the start of the Leader Write Buffer at address 0x200 with the Follower Buffer Write command (32'h30000000).
3. Perform AVMM writes to the Leader Write Buffer starting with address 0x204 with the data to be sent to the target.
4. Set which Follower to write by writing the Leader Command Register Follower Select bits 31:30.
5. Set Command Register bit 1 rdnwr=0 for a write operation. Set Command Register's Burst Length bits 15:2 to the value burst length-1 for the total burst DWORDs on the SPI bus. Set the trans_valid bit to 1. This triggers the Leader to send the write buffer's contents (command DWORD and write buffer data) across the SPI bus into the Follower Write Buffer.

The Initiator sets up the Follower's AVMM address and target selection by programming the Follower's Command Register0.

6. Wait until the Leader Command Register bit 0 trans_valid=0.

7. Perform an AVMM write to the start of the Leader Write Buffer at address 0x200 with the Follower Register Write command to Follower Command Register0 at address 0x000 (32'h10000000).
8. Perform an AVMM write to the 2nd DWORD of the Leader Write Buffer at address 0x204 with the value to be programmed into the Follower Command Register0:
    - avmm_burst_len=number of DWORDs - 1 to sent to the target from the Follower's Write Buffer
    - avmm_sel=which of AVMM0, 1 or 2 to send the Follower's Write Buffer Data
    - start_addr=target's AVMM start address
    - rdnwr=0 (write)
    - trans_valid=1
9. Set Leader Command Register bit 1 rdnwr=0 for a write operation. Set Leader Command Register's Burst Length bits 15:2 to 1 for a total of 2 DWORDs to be sent across the SPI bus (the Follower Register Write command DWORD and the Follower Command Register0 value). Set the trans_valid bit to 1. This triggers the Leader to send the write buffer's contents. The Follower will in turn send its own Write Buffer to the target upon receiving the Command Register0 value.
10. Wait until the Leader Command Register bit 0 trans_valid=0.

3.13 Read from Target Sequence

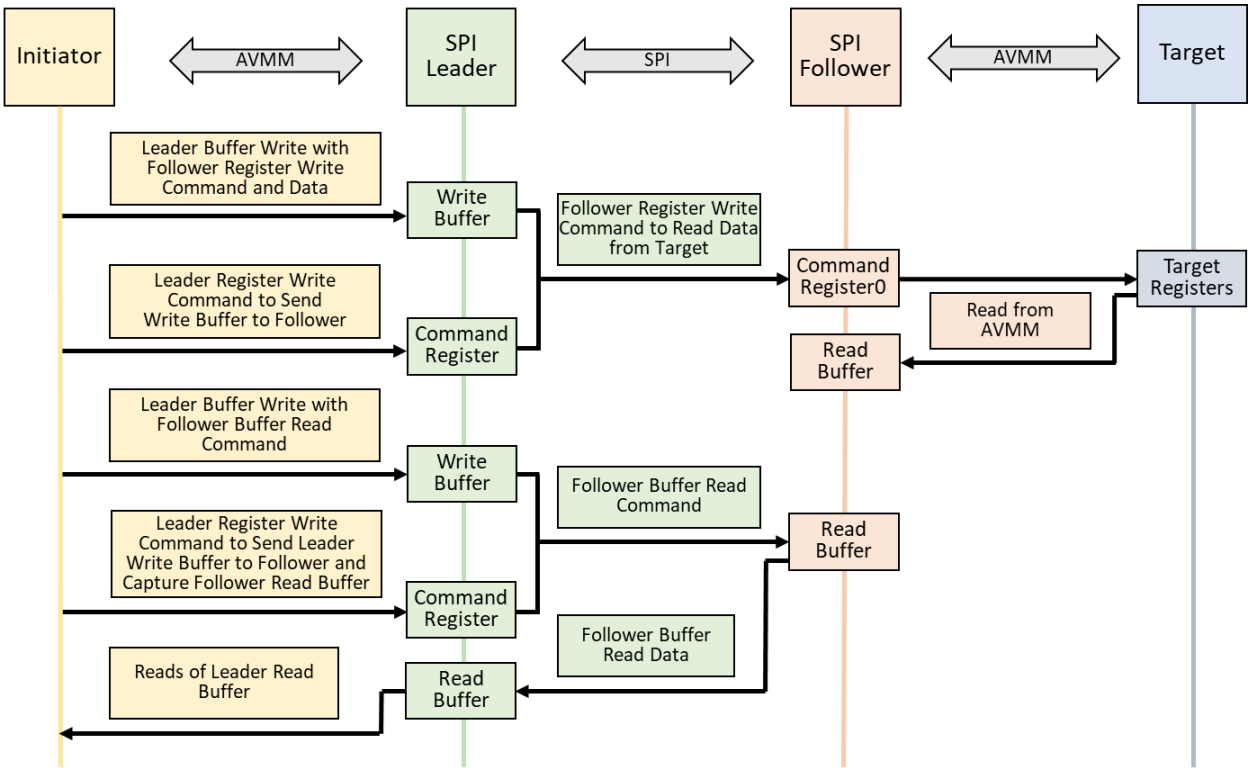

Figure 12. Initiator Read from AVMM Target

The Initiator sets up the Follower's AVMM address and target selection by programming the Follower's Command Register0.

1. Wait until the Leader Command Register bit 0 trans_valid=0.
2. Perform an AVMM write to the start of the Leader Write Buffer at address 0x200 with the Follower Register Write command to Follower Command Register0 at address 0x000 (32'h10000000).
3. Perform an AVMM write to the 2nd DWORD of the Leader Write Buffer at address 0x204 with the value to be programmed into the Follower Command Register0:
    - avmm_burst_len=number of DWORDs - 1 to read from the target into the Follower's Read Buffer
    - avmm_sel=which of AVMM0, 1 or 2 to read from into the Follower's Read Data
    - start_addr=target's AVMM start address
    - rdnwr=1 (read)
    - trans_valid=1
4. Set Leader Command Register bit 1 rdnwr=0 for a write operation. Set Leader Command Register's Burst Length bits 15:2 to 1 for a total of 2 DWORDs to be sent across the SPI bus (the Follower Register Write command DWORD and the Follower Command Register0 value). Set the trans_valid bit to 1. This triggers the Leader to send the write buffer's contents. The Follower will in turn read from the target into its Read Buffer upon receiving the Command Register0 value.

The Initiator reads from the Follower's read buffer with data from the target:

1. Wait until the Leader Command Register bit 0 trans_valid=0.
2. Perform an AVMM write to the start of the Leader Write Buffer at address 0x200 with the Follower Buffer Read command (32'h20000000).
3. Set which Follower to write by writing the Leader Command Register Follower Select bits 31:30.
4. Set Leader Command Register bit 1 rdnwr=1 for a read operation. Set Leader Command Register's Burst Length bits 15:2 to the value (total burst DWORDs - 1) on the SPI bus. This should be the Follower's avmm_burst_len + 1 for the dummy word. Set the trans_valid bit to 1. This triggers the Leader to capture the the Follower's Read Buffer contents (dummy DWORD and Follower Read Buffer data) from the SPI bus into the Leader Read Buffer.
5. Wait until the Leader is idle, Command Register field trans_valid (bit 0)=0.
6. Perform AVMM reads of the Leader's Read Buffer.

3.14 Follower AIB Application Configuration

A chiplet using the Follower IP to program an AIB interface combines the SPI interface on one side and the AVMM Leader on the other side. In an AIB subsystem the Follower IP is used to program a sequence of AIB channel register accesses, as shown in Figure 13.

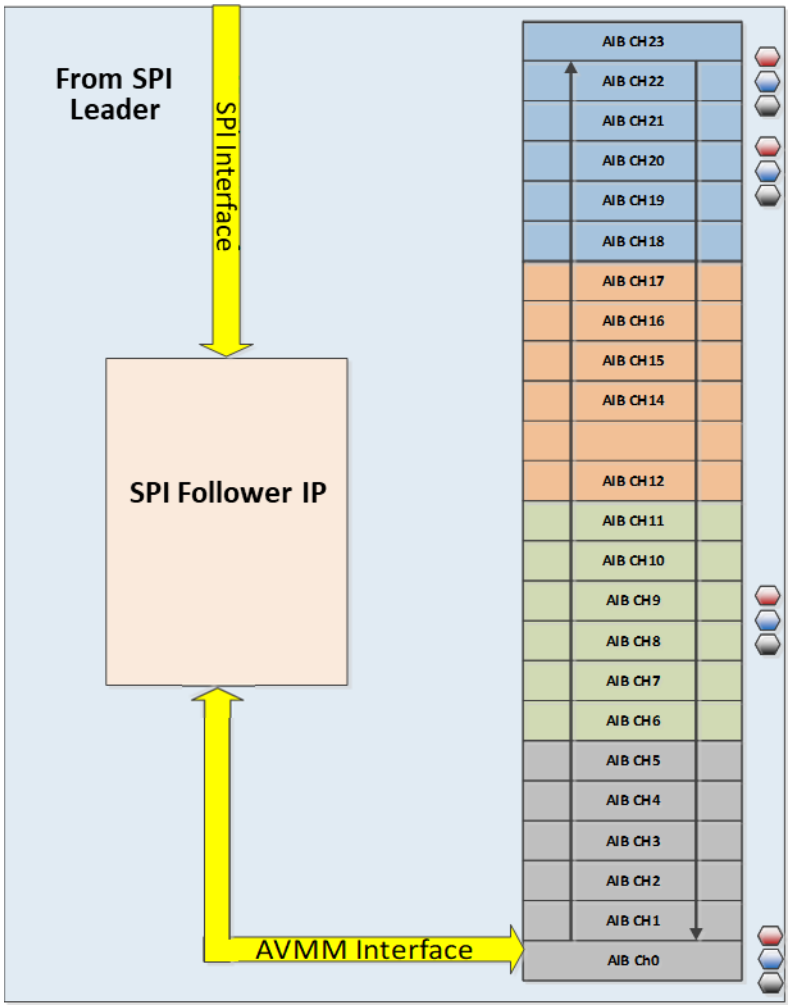

Figure 13. Follower AVMM Interface

The address map of the 24 channel AIB stack is shown in Table 7. The Follower IP is programmed by the Leader to use the correct address offset for each channel as listed in the table.

IP Component	AVMM Base Address	Limit Address	Base Address Channel ID
AIB ch 0	0x0000	0x03FF	6'b0000_00
AIB ch 1	0x0800	0x0BFF	6'b0000_01
AIB ch 2	0x1000	0x13FF	6'b0000_10
AIB ch 3	0x1800	0x1BFF	6'b0000_11
AIB ch 4	0x2000	0x23FF	6'b0001_00
AIB ch 5	0x2800	0x2BFF	6'b0001_01
AIB ch 6	0x3000	0x33FF	6'b0001_10
AIB ch 7	0x3800	0x3BFF	6'b0001_11
AIB ch 8	0x4000	0x43FF	6'b0010_00
AIB ch 9	0x4800	0x4BFF	6'b0010_01
AIB ch10	0x5000	0x53FF	6'b0010_10
AIB ch 11	0x5800	0x5BFF	6'b0010_11
AIB ch 12	0x6000	0x63FF	6'b0011_00
AIB ch 13	0x6800	0x6BFF	6'b0011_01
AIB ch 14	0x7000	0x73FF	6'b0011_10
AIB ch 15	0x7800	0x7BFF	6'b0011_11
AIB ch 16	0x8000	0x83FF	6'b0100_00
AIB ch 17	0x8800	0x8BFF	6'b0100_01
AIB ch 18	0x9000	0x93FF	6'b0100_10
AIB ch 19	0x9800	0x9BFF	6'b0100_11
AIB ch 20	0xA000	0xA3FF	6'b0101_00
AIB ch 21	0xA800	0xABFF	6'b0101_01
AIB ch 22	0xB000	0xB3FF	6'b0101_10
AIB ch 23	0xB800	0xBBFF	6'b0101_11

Table 7. AVMM Address Map for Accessing a 24 Channel AVMM Stack

4. Error Handling and Debugging

Since the SPI protocol itself does not have error handling, both the Leader and the Follower should allow some flexibility for any unexpected violation, like early termination of the ss_n signal. No lock should ever happen, and graceful recovery should always be allowed. That means, at any stage of the state machine, de-assertion of the ss_n signal should let both the Leader and the Follower go to idle state.

5. Example Design

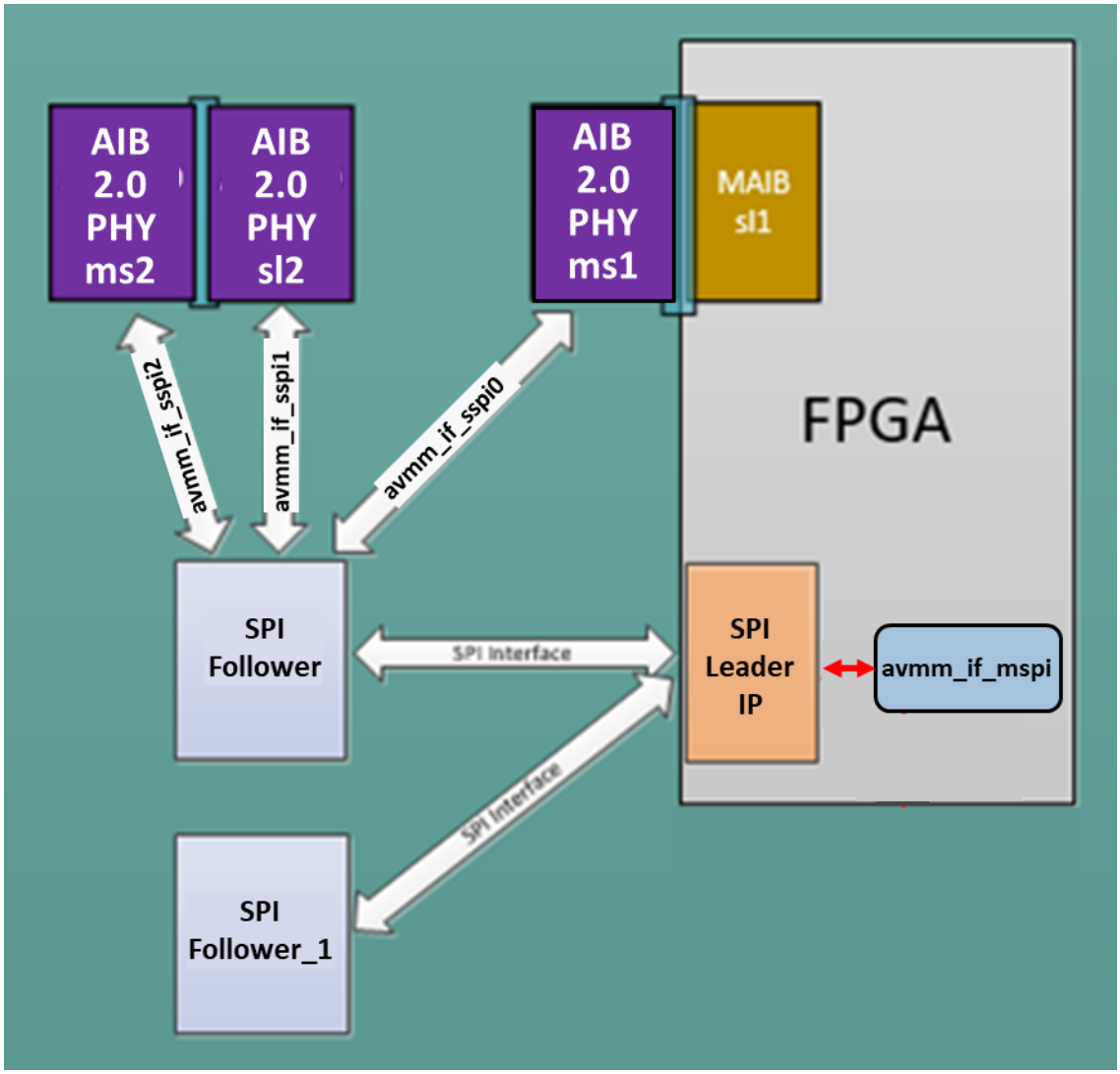

Figure 14. Example Design Testbench in Github

This example shows how to program a 24 channel AIB PHY and then start traffic in both the left AIB 2.0 PHY pair and the right AIB 2.0 PHY/MAIB pair. You can find the above example design testbench top_tb.sv in https://github.com/chipsalliance/aib-protocols/tree/main/spi-aib/dv/tb. Check README.txt in aib-protocol/spi-aib/dv/sims for how to run the test.

Two test cases are provided. The basic_spi_test.inc covers all the basic SPI commands described in this user guide. Another test vector, fifo2x_test.inc, mimics the real AIB system bringup. The Leader will program the Follower through the SPI bus and connects to three AIB 2.0 PHY (shown in Figure 12). The left pair (labeled ms2 and sl2) are operating in Gen2 mode, and right pair (labeled ms1 and MAIB sl1) are operating in Gen1 mode.

6. References

Chiplet SPI IP Repository, CHIPS Alliance Github location:
https://github.com/chipsalliance/aib-protocols/tree/main/spi-aib

AIB PHY IP Repository, CHIPS Alliance Github location:
https://github.com/chipsalliance/aib-phy-hardware

Advanced Interface Bus (AIB) Specification, Revision 2.0.2
https://github.com/chipsalliance/AIB-specification