EXIMIUS
Excellence Everyday

SHIP IP User Guide

SPI IP

Revision: 0.8.3

15 October 2021

Table of Contents

Table of Figures

Table of Tables

Revision History

Revision	Date	Author	Summary of Changes
0.8	8/29/2021	Ethiraj	Initial draft
0.8.1	9/15/2021	Ethiraj	CSR Updates
0.8.2	9/29/2021	Ethiraj	Updated Sequence of operations with dummy read details
0.8.3	10/15/2021	Ethiraj	Updated the block diagram in the sequence examples

Table 1 Revision History

1. Overview

SPI Master and SPI Slave IP provide a mapping function between the AVMM parallel programming and the SPI serial function. SPI Slave IP maps the AVMM Master I/F to SPI-S I/F and is used in the Follower Chiplet. The SPI Master IP maps SPI Master I/F to AVMM Slave I/F and is used in the Leader Chiplet.

SPI IPs will be used in Chiplet Control Subsystem and Security Subsystem.

1.1 Usage in Chiplet Control Subsystem

SPI Master and Slave IP will be used for Chiplet Control. SPI Slave IP is used to configure the AIB Channel registers using the AVMM I/F in the IP. The SPI I/F in the Slave IP is driven by the SPI Master IP from another chiplet. In the example below, the SPI Master transfers AIB configuration instructions from NIOS to the SPI slave via the SPI I/F. SPI Slave IP then configures the AIB registers using the AVMM I/F connected to Channel 0.

The SPI IP cores in the AIB Chiplet system are shown in Figure 1.

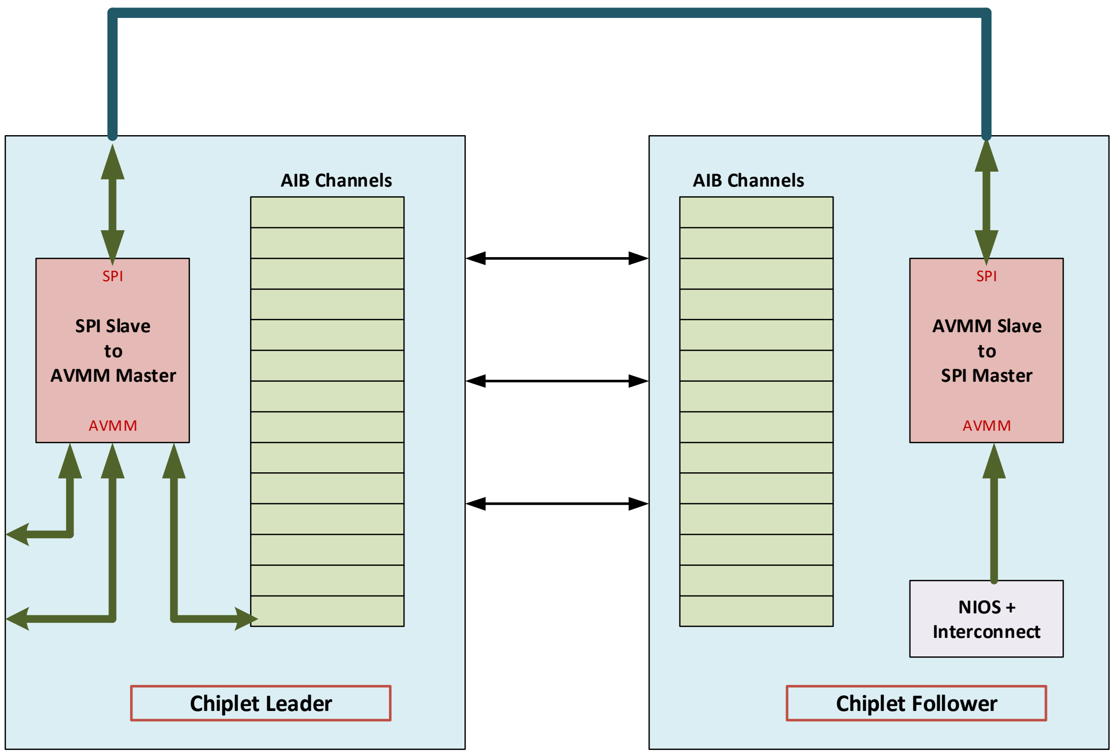


Figure 1 SPI IP Cores in AIB Chiplet System

1.2 Usage in Security Subsystem

SPI Master IP will be used in the Security Subsystem. SPI slave used is a 3rd party IP. SPI master and slave exchange messages during the security boot up. SPI Master exchanges messages with the Security subsystem SPI slave.

Figure 2 below shows the IP usage in the Security Subsystem.

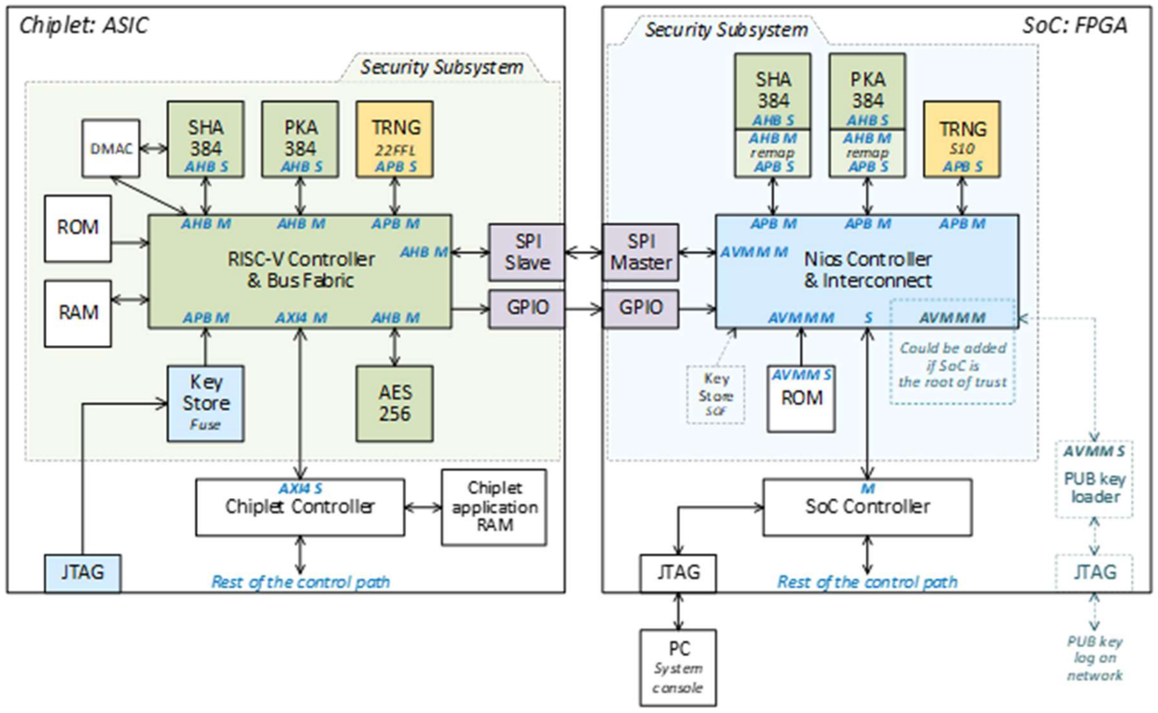


Figure 2 SPI Master IP in Security Subsystem

2. Features

2.1 SPI Interface Format

The interface supports the following:

- Active low Slave select.
- Mode0: CPOL = 0 and CPHA = 0
- MSB in the data byte is transferred first.

The SPI interface protocol format shown in Figure 3

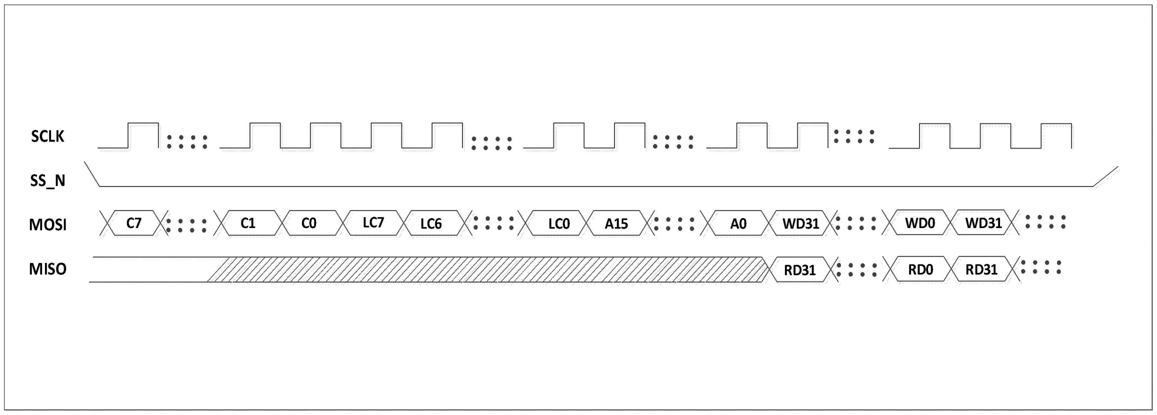


Figure 3 SPI Interface Protocol Format

The protocol fields are shown in Table 2 below.

Field	Number of bits
Command	8
Burst Count	8
Address	16
Data	Multiple word data

Table 2 SPI Protocol Fields

Four commands are supported:

Command	Hex Value	Description
Single Read	8’h0	Burst count LC =0
Single Write	8’h1	Burst count LC=0
Burst Read	8’h20	Burst count = burst wd
Burst Write	8’h21	Burst count = burst wd

Table 3 SPI Command Details

2.1.1 Common SPI Interface Format

SPI Master will use a common SPI Interface format to transmit and receive data in both Security and Non-Security mode.

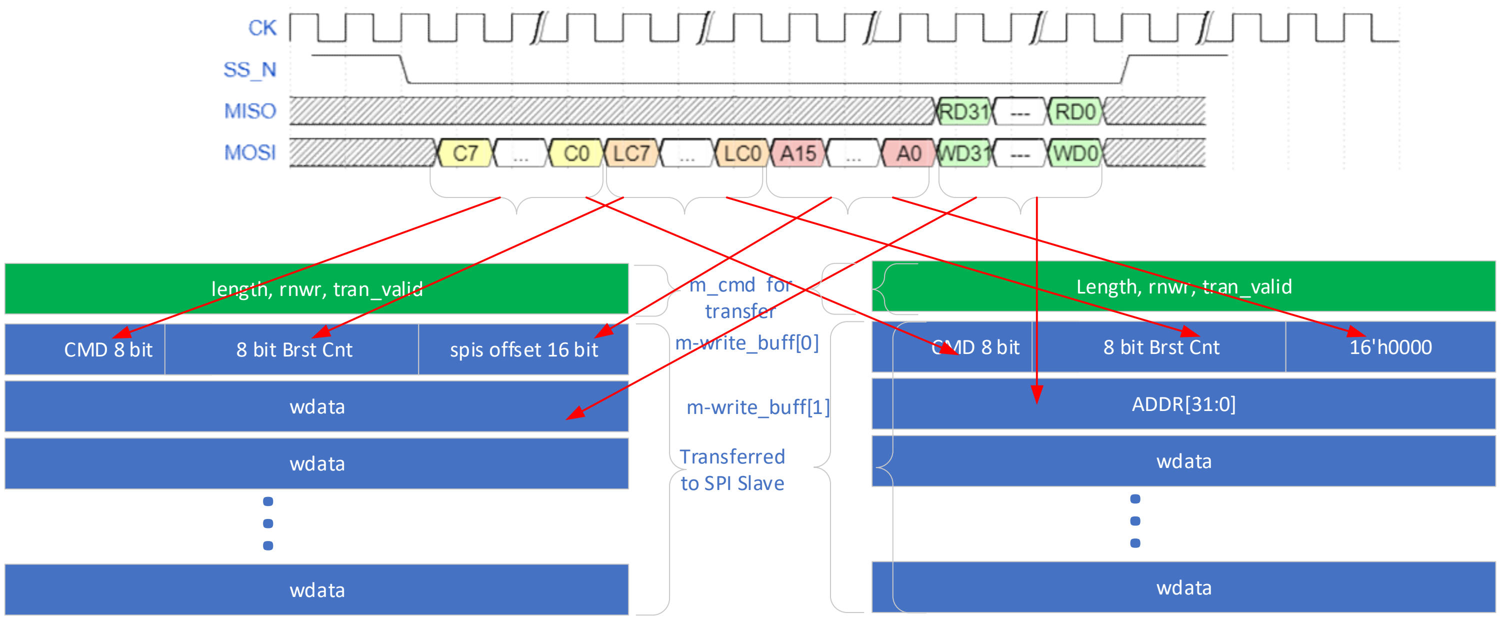


Figure 4 Common SPI Interface

2.2 *Sequence of operations*

The basic chiplet system can consists of two chiplets, a leader chiplet and a follower chiplet. The SPI slave IP is used to configure the AIB channel registers and other CSR on the chiplet. The SPI interface in the SPI slave is controlled by the C4 Bumps driven by a SPI master from another chiplet follower. For the following discussion, the chiplet follower (FPGA) initiates the AIB programming and is referred to as the initiator (of the transaction). As an example, the sequence to write and read the AIB Channel registers is described below:

A write or read of the AIB Channel registers in the follower chiplet by the master chiplet via the SPI interface is multi-step process. In the chiplet system configuration (Figure 1), the NIOS Processor in the Master

Chiplet is the initiator and interfaces with the SPI Master IP via AVMM interface. In the follower Chiplet, the SPI Salve IP interfaces to the AIB Channel registers via AVMM interface.

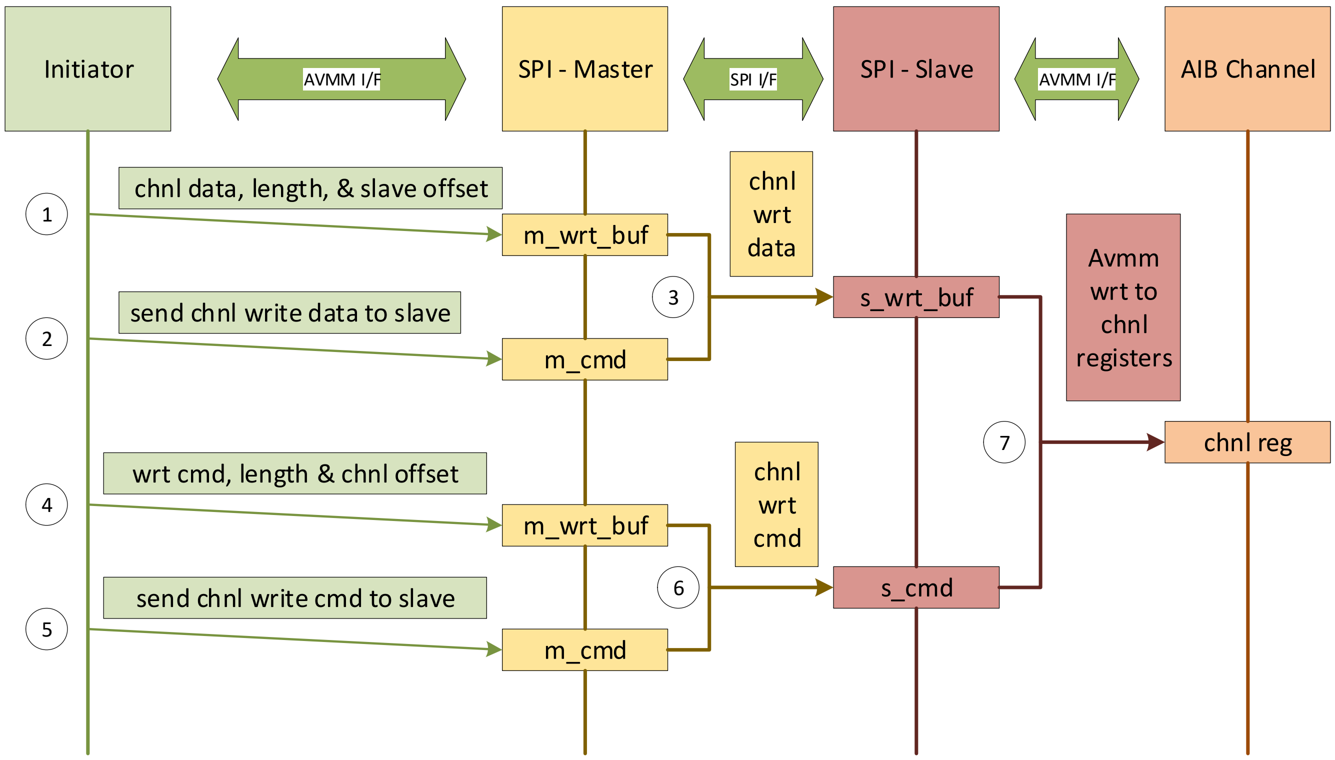


Figure 5 Flow diagram for write

For Writes

- Initiator programs the write data (including the cmd, length and offset for slave) in the SPI Master's write buffer (m_wbuf) by AVMM writes.
- Initiator then transfers the write data by writing to m_cmd (set s_trans_valid to 1) to begin transfer to SPI Slave IP. This will transfer and store the follower channel data in slave's write buffer (s_wbuf).
- Initiator polls m_cmd to for completion of the SPI transfer to slave. Transfer is complete when s_trans_valid is 0.
- When the write data has been transferred to SPI Slave, Initiator stores the slave write command in m_wbuf. The command has the AIB Channel register offset, write command, length.
- Initiator transfers to slave the write command in m_wbuf by writing to m_cmd (set s_trans_valid to 1). This is to store the contents of m_wbuf in the slave's s_cmd register.
- Initiator polls m_cmd to for completion of the SPI transfer to slave.
- SPI slave transfers the write data in s_wbuf as per s_cmd to AIB channel registers via AVMM transactions. The write is initiated when the s_cmd register's avmm_trans_valid is set to 1.
- Initiator must poll slave's s_cmd to determine the completion of the write command to AIB channel. The write is complete when bit avmm_trans_valid is 0. When programming the master's m_cmd register for polling of s_cmd, the burst length should include 1 additional dword (dummy read). The dummy read word is stored in master's read buffer location 0x1000. The valid polling (s_cmd) data is available in location 0x1004.
- A new command cannot be issued till the current transactions are completed.

For Reads

- Initiator stores the read command (with AIB Channel offset, read, length) in SPI Master’s m_wbuf by AVMM writes.
- Initiator then transfers to slave the command in m_wbuf by writing to m_cmd (set s_trans_valid to 1). This will transfer and store the contents of m_wbuf in Slave’s s_cmd register.
- Initiator polls m_cmd to for completion of the SPI transfer to slave. Transfer is complete when s_trans_valid is 0.
- SPI Slave initiates AIB Channel registers read via AVMM transactions and stores the returned read data in s_rbuf. The read is initiated when the s_cmd register’s avmm_trans_valid is set to 1.
- Initiator must poll slave’s s_cmd to determine the completion of the read command to AIB channel. The read is complete when bit avmm_trans_valid is 0. When programming the master’s m_cmd register for polling of s_cmd, the burst length should include 1 additional dword (dummy read). The dummy read word is stored in master’s read buffer location 0x1000. The valid polling (s_cmd) data is available in location 0x1004.
- When the read of the AIB channel by the slave is complete, the Initiator triggers a SPI-M read of the Slave s_rbuf to transfer the AIB Channel read data to SPI Master m_rbuf. When programming the master’s m_cmd register for the transfer, the burst length should include 2 additional dword (dummy reads). The dummy read words are stored in masters read buffer locations 0x1000 and 0x10004. The valid read data is available from location 0x1008 onwards.
- Initiator polls m_cmd to for completion of the SPI transfer of the read data from slave. Transfer is complete when s_trans_valid is 0.
- When the above SPI transaction is completed, the Initiator reads the data from the SPI Master m_rbuf via AVMM reads. The AVMM reads should account for the two additional dummy reads at the beginning of the read cycle.

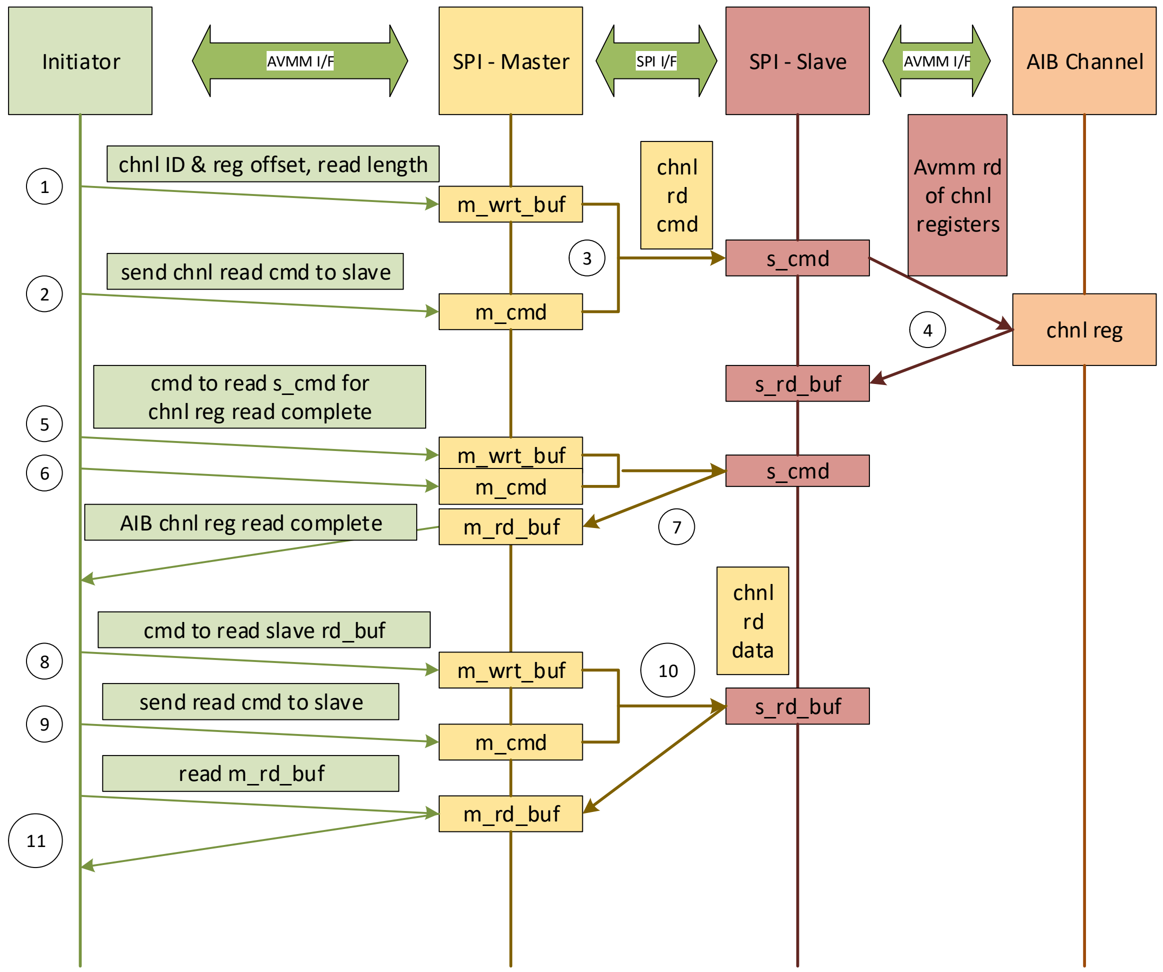


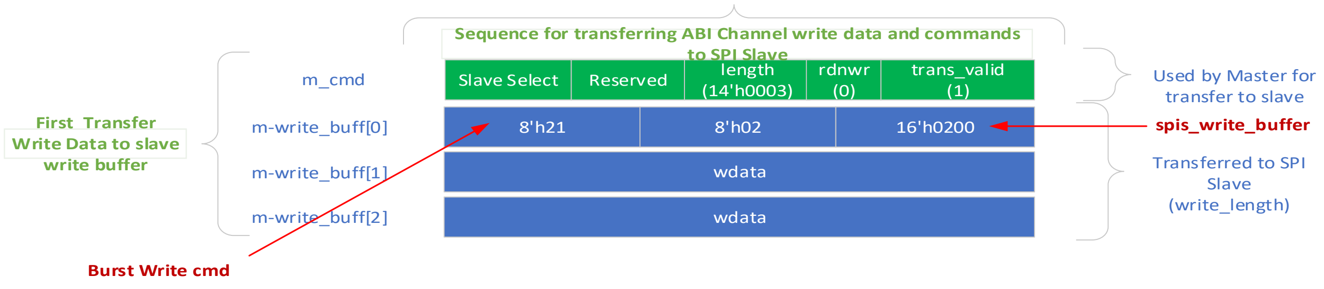
Figure 6 Flow diagram for read

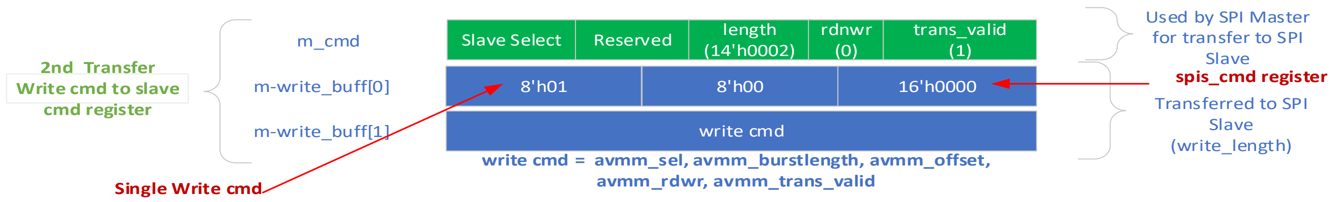
2.2.1 Write from Initiator

The steps below are an example for writing data to an AIB Channel register:

1. Initiator programs the master's write buffer (m_wbuf) starting at 0x0200 with write command, length, slave write buffer address and the data to be written.
2. Initiator programs master's command register (m_cmd) to transfer the contents of m_wbuf to slave. Slave stores AIB channel data in its write buffer (s_wbuf).
3. Initiator polls m_cmd register for completion of the above transfer. When m_cmd bit [0] is de-asserted, the transfer is complete. This is required before the next transfer command can be issued.
4. Initiator programs m_wbuf with the address and contents of slave command register (s_cmd) and transfers the data to the slave. Slave stores this in s_cmd.
5. Polling of m_cmd is done to confirm transfer completion.
6. Slave will start to write the contents of its s_wbuf to the AIB Channel.
7. Initiator will have to poll the slave's s_cmd register to check the completion of AIB Channel writes by issuing a read command (accounting for one dummy dword) to slave to poll its s_cmd register. The write is complete when s_cmd bit [0] is de-asserted. The polling data from s_cmd is stored in master's read buffer (m_rbuf).
8. When the polling is complete, the Initiator will read m_rbuf to confirm completion of write to AIB Channel registers.

Since the data transfer on the SPI interface is full duplex, there is "MISO" data from the slave for every MOSI data transfer from the master. This MISO data is saved in master's read buffer m_rbuf. When the transfer from master is a write to slave, the master's read buffer (m_rbuf) is reset at the end of the transfer by the logic by default. This default behavior can be disabled by setting wbuf_sftrst_ctrl (master's register wbuf_fifo_control[1]) and rbuf_sftrst_ctrl (master's register rbuf_fifo_control[1]) to 1'b1. When the wbuf_sftrst_ctrl and rbuf_sftrst_ctrl is set to 1, the buffer resets are controlled by soft reset bits in wbuf_fifo_control[0] and rbuf_fifo_control[0] and should be toggled 0 ->1 ->0.

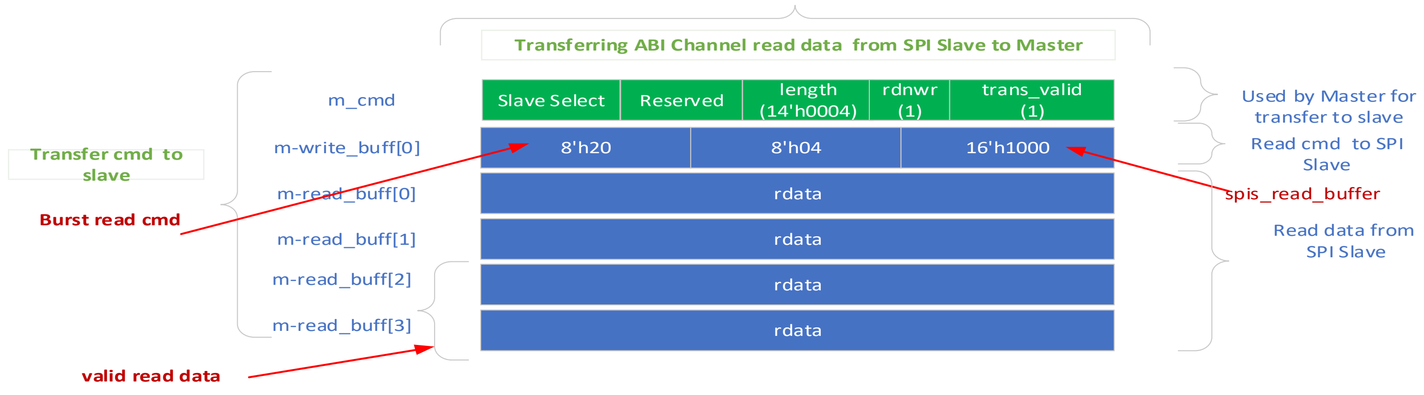

Sequence for transferring ABI Channel write data and commands to SPI Slave
m_cmd
Slave Select
Reserved
length (14'h0003)
rdnwr (0)
trans_valid (1)
Used by Master for transfer to slave
First Transfer Write Data to slave write buffer
m-write_buff[0]
8'h21
8'h02
16'h0200
spis_write_buffer
m-write_buff[1]
wdata
Transferred to SPI Slave (write_length)
m-write_buff[2]
wdata
Burst Write cmd


m_cmd
Slave Select
Reserved
length (14'h0002)
rdnwr (0)
trans_valid (1)
Used by SPI Master for transfer to SPI Slave
2nd Transfer Write cmd to slave cmd register
m-write_buff[0]
8'h01
8'h00
16'h0000
spis_cmd register
Transferred to SPI Slave (write_length)
m-write_buff[1]
write cmd
write cmd = avmm_sel, avmm_burstlength, avmm_offset, avmm_rdwr, avmm_trans_valid
Single Write cmd

2.2.2 Read from initiator

The steps below are an example for reading data from an AIB Channel register:

1. Initiator programs the master's write buffer (m_wbuf) starting at 0x0200 with read command, length, and the contents of the slave's command (s_cmd) register.
2. Initiator programs master's command register (m_cmd) to transfer the contents of m_wbuf to slave. Slave will store this data in s_cmd
3. Initiator polls m_cmd register for completion of the above transfer. When m_cmd bit [0] is de-asserted, the transfer is complete.
4. When this transfer is completed, the slave begins the read of the AIB channel registers. The Channel read data is stored in slave's read buffer(s_rbuf).
5. Initiator will have to poll the slave's s_cmd register to check the completion of AIB Channel reads by issuing a read command (accounting for one dummy dword) to slave to read its s_cmd register. The read is complete when s_cmd bit [0] is de-asserted. The polling data from slave's s_cmd is stored in master's read buffer (s_rbuf).
6. When AIB Channel read is completed, Initiator will have to transfers the data read to the master and will issue a command (accounting for two dummy dwords) to the slave to read it s_rbuf and transfer the data.
7. The data from slave is stored in master's read buffer m_rbuf.
8. Initiator polls m_cmd register for completion of read data transfer. When m_cmd bit [0] is de-asserted, the transfer is complete.
9. Initiator reads master's m_rbuf for the AIB Channel data, accounting for the two dummy reads.



2.2.3 Usage in Security System

During the security boot up processing, the SPI master sends data (writes) to the SPI slave in the security chiplet or retrieves data (reads) from the security chiplet. During the authentication process, the chiplet initiates the communication by asserting the GPIO pin (ready_int interrupt in the SPI Master). The ready_int interrupt is not used by the SPI Master. When the interrupt is received -

1. NIOS sets up the command register (m_cmd) in the SPI Master to perform a read of the SPI slave
    a. Sets up the length, read command and s_trans_valid to 1
2. SPI Master stores the data from SPI Slave in the m_rbuf (starting at offset 0x1000)
3. NIOS (AVMM Master) then reads the contents of the m_rbuf
4. NIOS (AVMM Master) writes the data (signature data) to the m_wbuf (starting at offset 0x200)
5. NIOS (AVMM Master) sets up the write command in m_cmd to transfer the signature data to the slave.

Ensure that the buffer auto reset is disabled by setting rbuf_sftrst_ctrl (master's register rbuf_fifo_control [2]) to 1'b1. See SPI Master Register for details.

2.2.4 Read and Write Buffers

The read and write buffers in the SPI Master and Slave are implemented using fifo. The size of the write buffer and read buffer is parameterized. The default size for read and write buffers is 64DX32W. A maximum depth of 512 is supported. A new transaction always starts at write buffer address 0x200 and read buffer address 0x1000. Random access is not supported.

Since the data transfer on the SPI interface is full duplex, there is "MISO" data from the slave for every MOSI data transfer from the master. This MISO data is saved in SPI master's read buffer m_rbuf. Read buffer m_rbuf is soft reset by design at the beginning of a SPI transfer. This is the default behavior. This default behavior can be disabled by setting rbuf_sftrst_ctrl in the SPI master register. Please refer to the section on registers for more details

3. SPI Slave Registers

Configuration and status registers shown in Table 4 SPI Slave Configuration Registers below

The size of the write buffer and read buffer is parameterized. A new transaction always starts at write buffer address 0x200 and read buffer address 0x1000. The default size for read and write buffers is 64DX32W. Random access is not supported.

Register Name	Offset	bits	R/W	Name	Description

s_cmd	0x0	31:24	RW	avmm_burstlength	AVMM Burst Length in dword
		23:21	RO	Reserved	
		20:19	RW	avmm_sel	Top AVMM address for selecting one of the three AVMM interface: 2’b00: Select AVMM0 2’b01: select AVMM1 2’b10: Select AVMM2 2’b11: Reserved
		18:2	RW	avmm_offset	17-bit AVMM start address
		1	RW	avmm_rdnwr	0: Write transaction 1: Read transaction
		0	RW	avmm_trans_valid	When set to ‘1’, AVMM configuration to AIB channel will start. When this bit is ‘0’, the transition has finished.
s_status	0xC	31:0	RO	reserved	
s_diag_0	0x10	31:19	RO	reserved	Values recorded at the end of the transfer
		18:16	RO	spi interface s/m state	
		15:8	RO	spi transfer burstcount	
		7:0	RO	spi transfer length	
s_diag_1	0x14	31:20	RO	reserved	
		19:16	RO	avmm interface s/m state	
		15:8	RO	Avmm transfer burst count	
		7:0	RO	Avmm transfer length	
buffer_fifo_status	0x40	0	RO	wbuf_wr_overflow	Write buffer overflow status. Clear status with write buffer write soft reset 0x44[bit 0]
		1	RO	wbuf rd_underflow	Write buffer underflow status. Clear status with write buffer read soft reset 0x44[bit 1]
		2	RO	rbuf_wr_overflow	Read buffer overflow status. Clear status with read buffer write soft reset 0x44[bit 2]
		3	RO	rbuf rd_underflow	Read buffer underflow status. Clear status with read buffer read soft reset 0x44[bit 3]
buffer_fifo_control	0x44	0	Wo/SC	wbuf_wr_soft_reset	Clears the write buffer overflow sticky status when set to 1. Self clears. Read returns zero.
		1	Wo/SC	wbuf_rd_soft_reset	1. Resets read side of the write buffer when set to 1. 2. Clears the write buffer underflow sticky status when set to 1.

					Self clears. Read returns zero.
		2	Wo/SC	rbuf_wr_soft_reset	Clears the read buffer overflow sticky status when set to 1. Self clears. Read returns zero.
		3	Wo/SC	rbuf_rd_soft_reset	1. Resets read side of the read buffer when sft_rst_ctrl (bit 4 0X44) is set to 1. Self clears. 2. Clears the read buffer underflow sticky status when set to 1. Self clears. Not gated by sft_rst_ctrl (bit 4 0x44) Read returns zero.
		4	RW	sft_rst_ctrl	When set to: 0: Enables write and read buffer read side soft reset by logic. Soft reset from bits 1 and 3 of this register is disabled. 1: Disables write and read buffer read side soft reset by logic and enables soft reset from bits 1 and 3 of this register
s_write_buffer	0x0200-0x09ff	31:0	RW		Used for generic data transfer from master to slave. The content of the data is understood by SPI slave decoding and firmware.
s_read_buffer	0x1000-0x17ff	31:0	RO		Used for generic data transfer from slave to master. The content of the data is understood by SPI slave and firmware.

Table 4 SPI Slave Configuration Registers

4. SPI Master Registers

Configuration and status registers shown in Table 5 SPI Master Configuration Registers below.

The size of the write buffer and read buffer is parameterized. A new transaction always starts at write buffer address 0x200 and read buffer address 0x1000. The default size for read and write buffers is 64DX32W. Random access is not supported

Register Name	Offset	bits	R/W	Name	Description
m_cmd	0x0	31:30	RW	Slave Select	00: Slave 0 is selected 01: Slave 1 is selected 10: Slave 2 is selected 11: Slave 3 is selected
		29:23	RO	Reserved	

		22:16	RO	Reserved	
		15:2	RW	spim_burst_length	Specifies the burst length in dwords when SS_n go low to SS_n go high **Note:** When not used in the security system, the burst length, when transferring the read data from the slave, should include two dummy dwords. **Refer to Sequence of operations for reads**
		1	RW	spim_rdnwr	0: Write transaction 1: Read transaction
		0	RW	s_trans_valid	1: SPI M transfer to SPI slave in progress 0: Idle
m_status	0xC	31:0	RO	reserved	
m_diag_0	0x10	31	RO	reserved	Values recorded at the end of the transfer
		30:28	RO	spi interface s/m state	
		27:14	RO	spi transfer burstcount	
		13:13	RO	spi transfer length	
m_diag_1	0x14	31:3	RO	reserved	
		2:0	RO	avmm interface s/m state	
wbuf_rdback	0x20	31:0	RO		Holds NIOS write data to write buffer location 0x0200 only for read back
wbuf_fifo_status	0x40	0	RO	wr_overflow	Write buffer overflow status. Clear status with write buffer write soft reset 0x44[bit 0]. ***Note: The is a delay of 2-3 cycles before the cleared status is reflected in this register***
		1	RO	rd_underflow	Write buffer underflow status. Clear status with write buffer read soft reset 0x44[bit 1]. ***Note: The is a delay of 2-3 cycles before the cleared status is reflected in this register***
wbuf_fifo_control	0x44	0	RW	wbuf_wr_soft_reset	Clears the write buffer overflow sticky status. Write 1 and then 0.
		1	RW	wbuf_rd_soft_reset	3. Resets read side of the Resets read side of the write buffer when set to 1 and then to 0. 4. Clears the write buffer underflow sticky status

					when set to 1 and then to 0.
rbuf_fifo_status	0x48	0	RO	wr_overflow	Read buffer overflow status. Clear status with read buffer write soft reset 0x4C[bit 0]. ***Note: The is a delay of 2-3 cycles before the cleared status is reflected in this register***
		1	RO	rd_underflow	Read buffer underflow status. Clear status with read buffer read soft reset 0x4C[bit 1]. ***Note: The is a delay of 2-3 cycles before the cleared status is reflected in this register***
rbuf_fifo_control	0x4C	0	RW	rbuf_wr_soft_reset	Clears the read buffer overflow sticky status. Write 1 and then 0
		1	RW	rbuf_rd_soft_reset	5. Resets read side of the read buffer when rbuf_sftrst_ctrl (bit 2 0X4C) is set to 1. 6. Clears the read buffer underflow sticky status. Write 1 and then 0 to clear. Not gated by rbuf_sftrst_ctrl (bit 2 0x4C) Write 1 and then 0.
		2	RW	rbuf_sftrst_ctrl	When set to: 0: Enables read buffer fifo read soft reset by logic at the beginning of a SPI transfer. Soft reset from bit 1 of this register is disabled. 1: Disables read buffer fifo read soft reset by logic and enables soft reset from bit 1 of this register.
m_write_buffer	0x0200-0x09ff	31:0	RW		Used for generic data transfer from master to slave. The content of the data is understood by SPI slave decoding and firmware.
m_read_buffer	0x1000-0x17ff	31:0	RO		Used for generic data transfer from slave to master. The content of the data is understood by SPI slave and firmware.

Table 5 SPI Master Configuration Registers

5. Clocks and Resets

5.1.1 SPI Slave Clock and Resets

Port	Direction	Width	Description
SPI Slave Interface			
sclk	in	1	SPI Serial clock from SPI Master. This is free running clock.
rst_n	in	1	Async system reset. Active low.
AVMM Master Interface 1 and Interface 2			
s_avmm_clk	in	1	avmm clock. This is free running clock.
s_avmm_rst_n	in	1	avmm reset. Active low.

5.1.2 SPI Master Clock and Resets

Port	Direction	Width	Description
SPI Master Interface			
sclk_in	in	1	Serial SPI clock. This is free running clock.
sclk	out	1	SPI I/F clock. This is free running clock to slaves.
rst_n	in	1	Async system reset. Active low.
AVMM Slave Interface			
m_avmm_clk	in	1	avmm_clock. This is free running clock.
m_avmm_rst_n	in	1	avmm_reset. Active low.

6. Interfaces

6.1.1 SPI Slave

Port	Direction	Width	Description
SPI Slave Interface			
ss_n	in	1	Slave select
mosi	in	1	Master out slave in
miso	out	1	Master in slave out
AVMM Master Interface 0			
s_avmm0_addr	out	17	AIB channel access addressing. Bits[16:11] – Channel ID Bits[10:0] – Channel register offset in byte
s_avmm0_byte_en	out	4	Byte enable – supports only dword.
s_avmm0_write	out	1	avmm write enable
s_avmm0_read	out	1	avmm read enable
s_avmm0_wdata	out	32	avmm write data
s_avmm0_rdatavld	in	1	avmm read valid when asserted
s_avmm0_rdata	in	32	avmm read data, valid when avmm_rdatavld is 1
s_avmm0_waitreq	in	1	avmm slave ready signal. When this signal is '0', the slave is ready
AVMM Master Interface 1			
s_avmm1_addr	out	17	AIB channel access addressing. Bits[16:11] – Channel ID Bits[10:0] – Channel register offset in byte
s_avmm1_byte_en	out	4	Byte enable – supports only dword.
s_avmm1_write	out	1	avmm write enable

s_avmm1_read	out	1	avmm read enable
s_avmm1_wdata	out	32	avmm write data
s_avmm1_rdatavld	in	1	avmm read valid when asserted
s_avmm1_rdata	in	32	avmm read data, valid when avmm_rdatavld is 1
s_avmm1_waitreq	in	1	avmm slave ready signal. When this signal is ‘0’, the slave is ready
AVMM Master Interface 2			
s_avmm2_addr	out	17	AIB channel access addressing. Bits[16:11] – Channel ID Bits[10:0] – Channel register offset in byte
s_avmm2_byte_en	out	4	Byte enable – supports only dword.
s_avmm2_write	out	1	avmm write enable
s_avmm2_read	out	1	avmm read enable
s_avmm2_wdata	out	32	avmm write data
s_avmm2_rdatavld	in	1	avmm read valid when asserted
s_avmm2_rdata	in	32	avmm read data, valid when avmm_rdatavld is 1
s_avmm2_waitreq	in	1	avmm slave ready signal. When this signal is ‘0’, the slave is ready

6.1.2 SPI Master

Port	Direction	Width	Description
SPI Master Interface			
ss_n_0	out	1	Slave select 0. Active low.
ss_n_1	out	1	Slave select 1. Active low.
ss_n_2	out	1	Slave select 2. Active low.
ss_n_3	out	1	Slave select 3. Active low.
mosi	out	1	Master out slave in
miso	in	1	Master in slave out
ready_int	in	1	Optional Interrupts/Ready GPIO signal from slave. High means ready. (Not used)
AVMM Slave Interface			
m_avmm_addr	in	17	AIB channel access addressing. Bits[16:11] – Channel ID Bits[10:0] – Channel register offset in byte
m_avmm_byte_en	in	4	Byte enable
m_avmm_write	in	1	avmm write enable
m_avmm_read	in	1	avmm read enable
m_avmm_wdata	in	32	avmm write data
m_avmm_rdatavld	out	1	avmm read valid when asserted
m_avmm_rdata	out	32	avmm read data, valid when avmm_rdatavld is 1
m_avmm_waitreq	out	1	avmm slave ready signal. When this signal is ‘0’, the slave is ready

7. References

SIP IP Core High-Level Specification (HAS) Version 0.6 dated July 06, 2021