SHIP IP User Guide

SPI IP

Revision: 0.8

29 August 2021

Table of Contents

Table of Figures

Table of Tables

Revision History

Revision	Date	Author	Summary of Changes
0.8	8/29/2021	Ethiraj	

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.

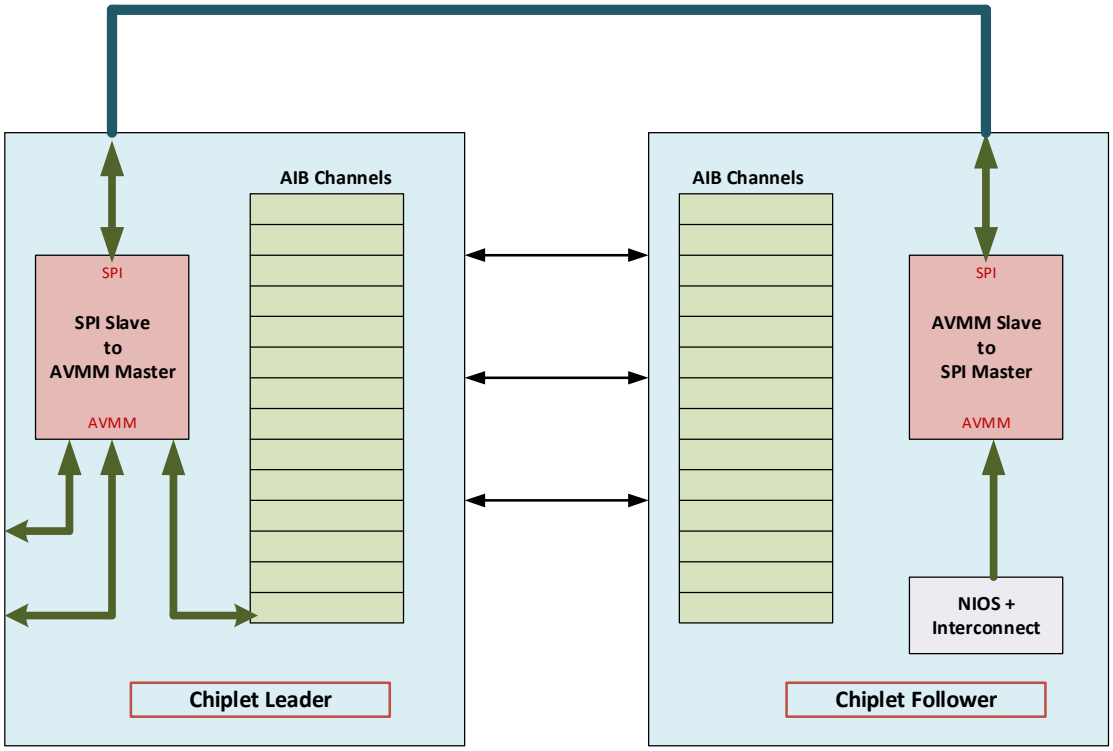


Figure 1 SPI IP Cores in AIB Chiplet System

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.

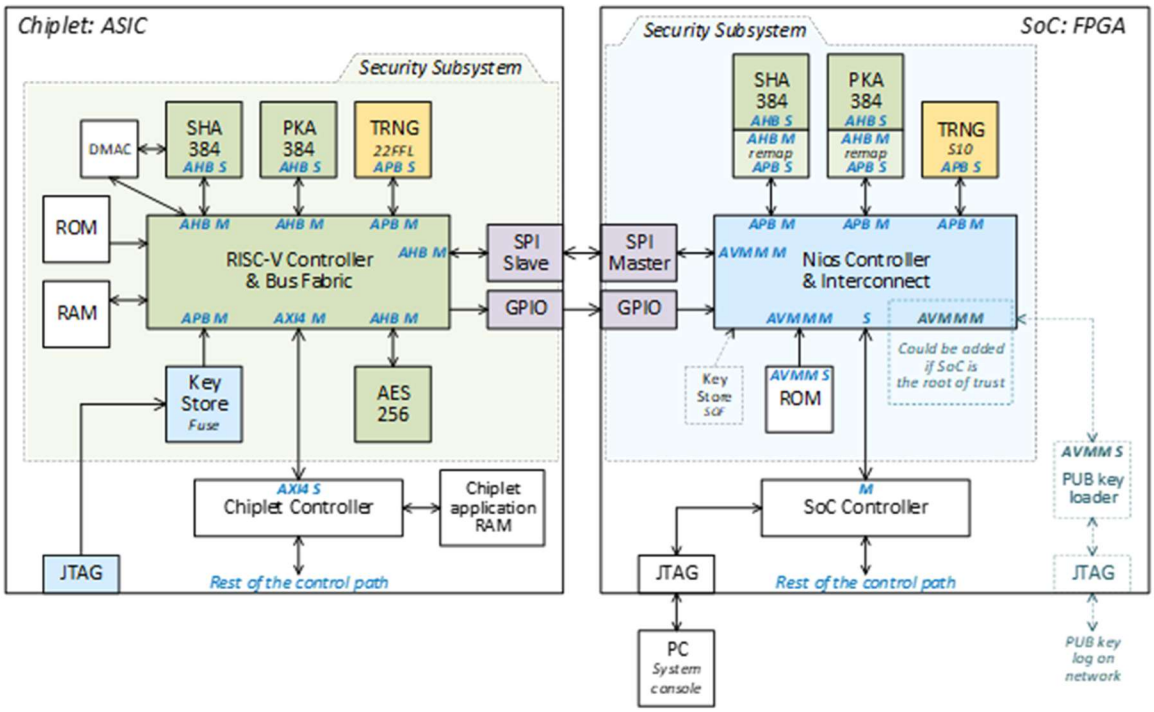


Figure 2 SPI Master IP in Security Subsystem

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

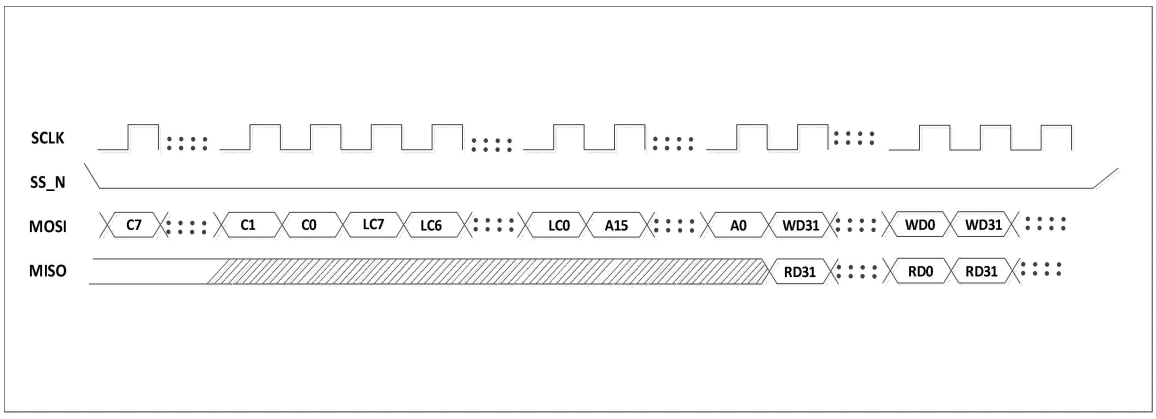


Figure 3 SPI Interface Protocol Format

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.

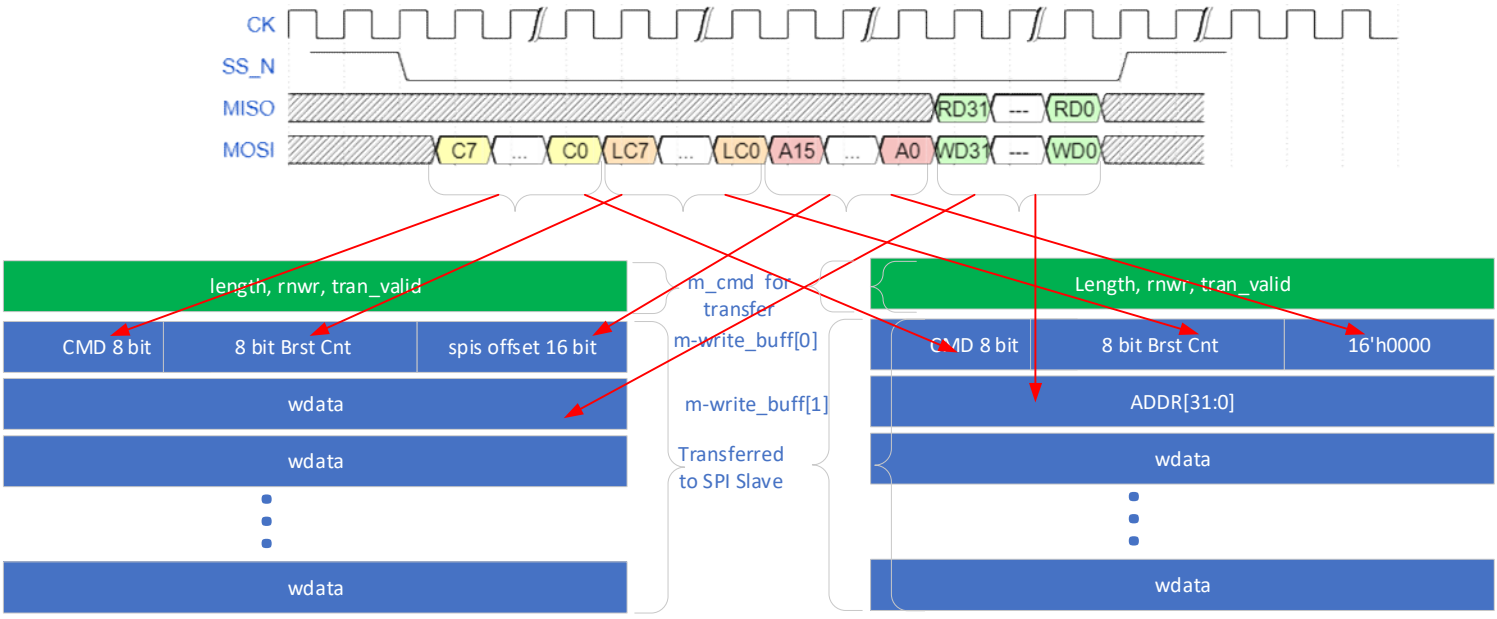


Figure 4 Common SPI Interface

2.2 AVMM Interface Format

Waveforms for AVMM configuration write and read are given below:

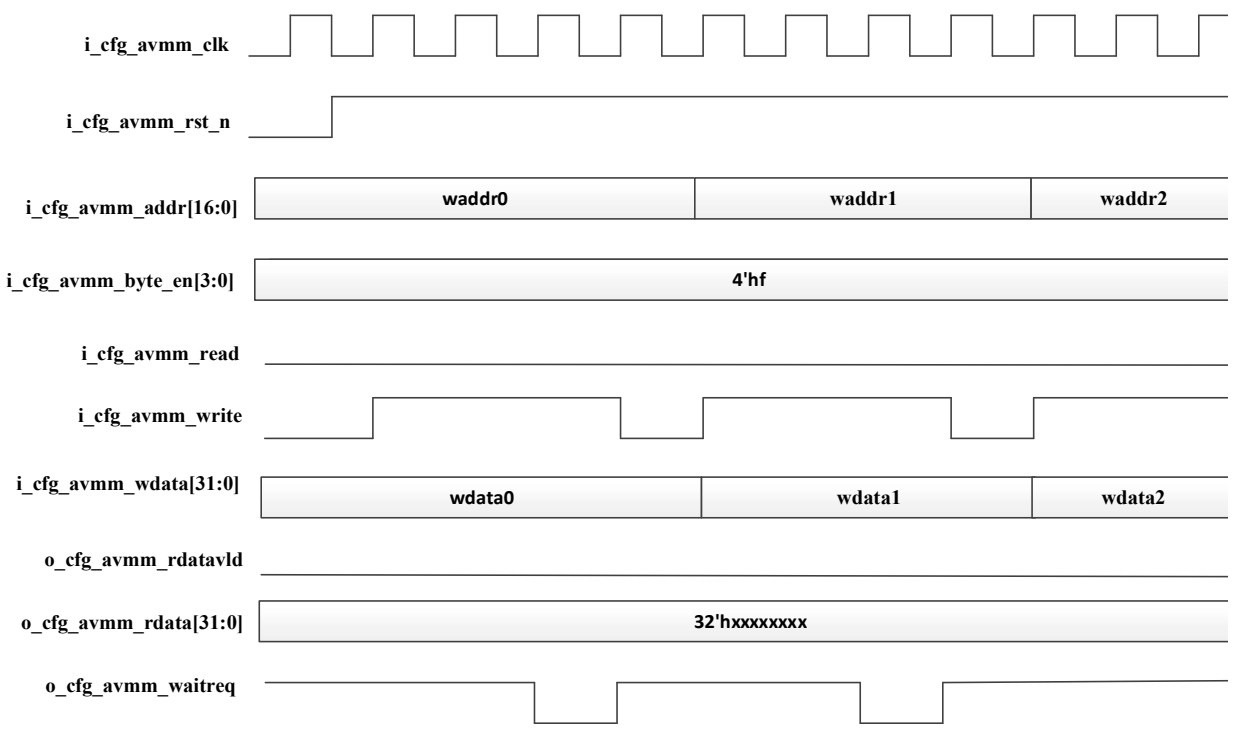


Figure 5 AVMM Configuration Write

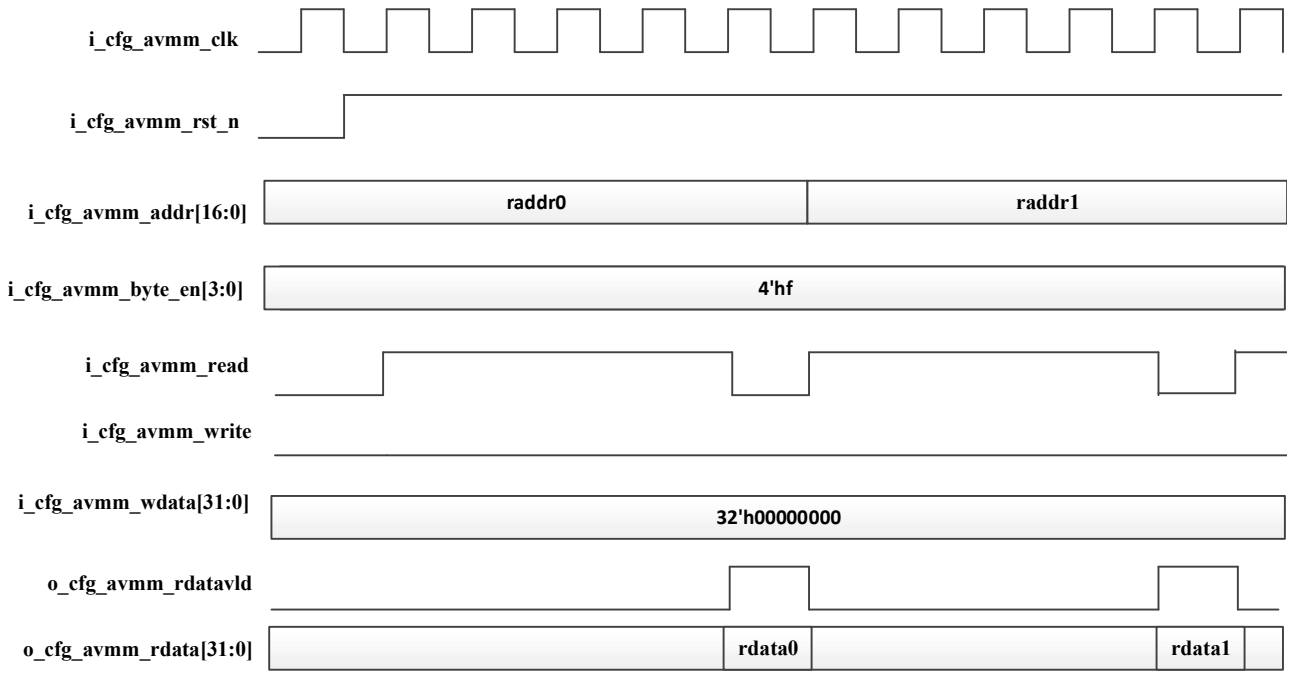


Figure 6 AVMM Configuration Read

2.3 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.

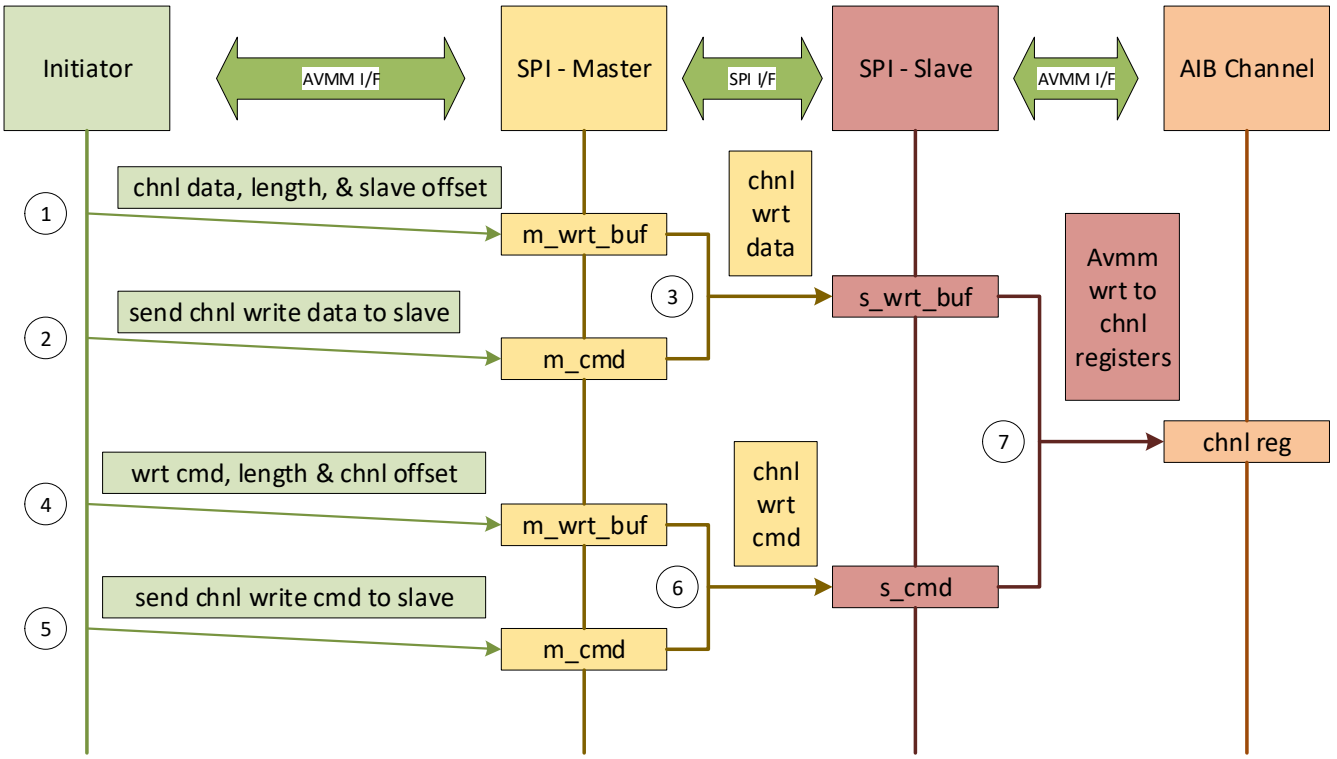


Figure 7 Flow diagram for write

For Reads

- Initiator first transfers the read command with AIB Channel offset, read, length to SPI Master "m_write_buffer" by AVMM writes.
- Initiator then transfers the command in m_write_buffer by writing to m_cmd. This will transfer and store the contents of m_write_buffer in the slave's s_cmd register.
- SPI Slave transfers the command to read the AIB Channel registers via AVMM transactions and stores the returned read data from the AIB Channel in s_read_buffer.
- When the read of the AIB channel by the slave is complete, the Initiator triggers a SPI-M read of the SPI slaves s_read_buffer to transfer the AIB Channel read data to SPI Master's m_read_buffer
- When the above SPI transaction is completed, the Initiator reads the data from the SPI Master m_read_buffer via AVMM reads

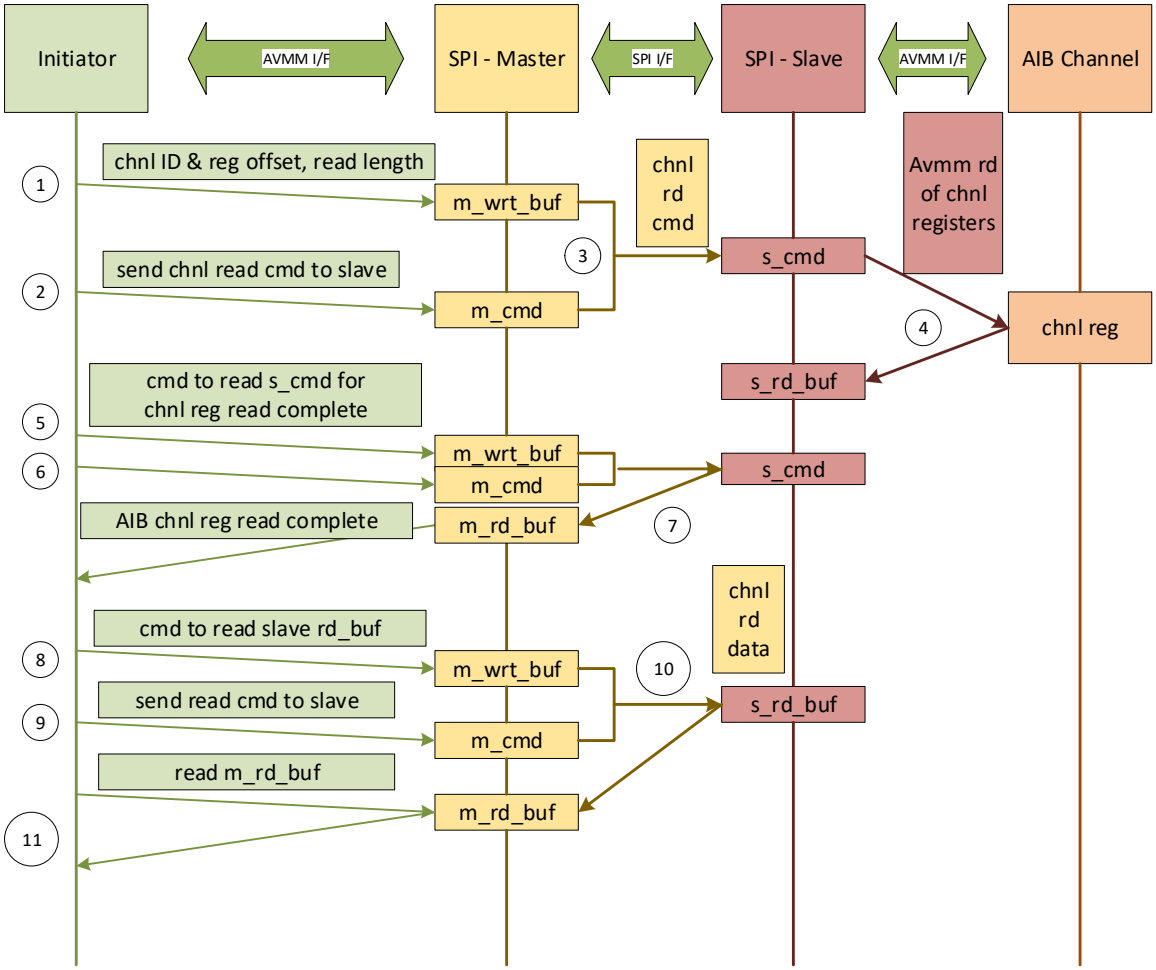


Figure 8 Flow diagram for read

2.3.1 Write from Initiator

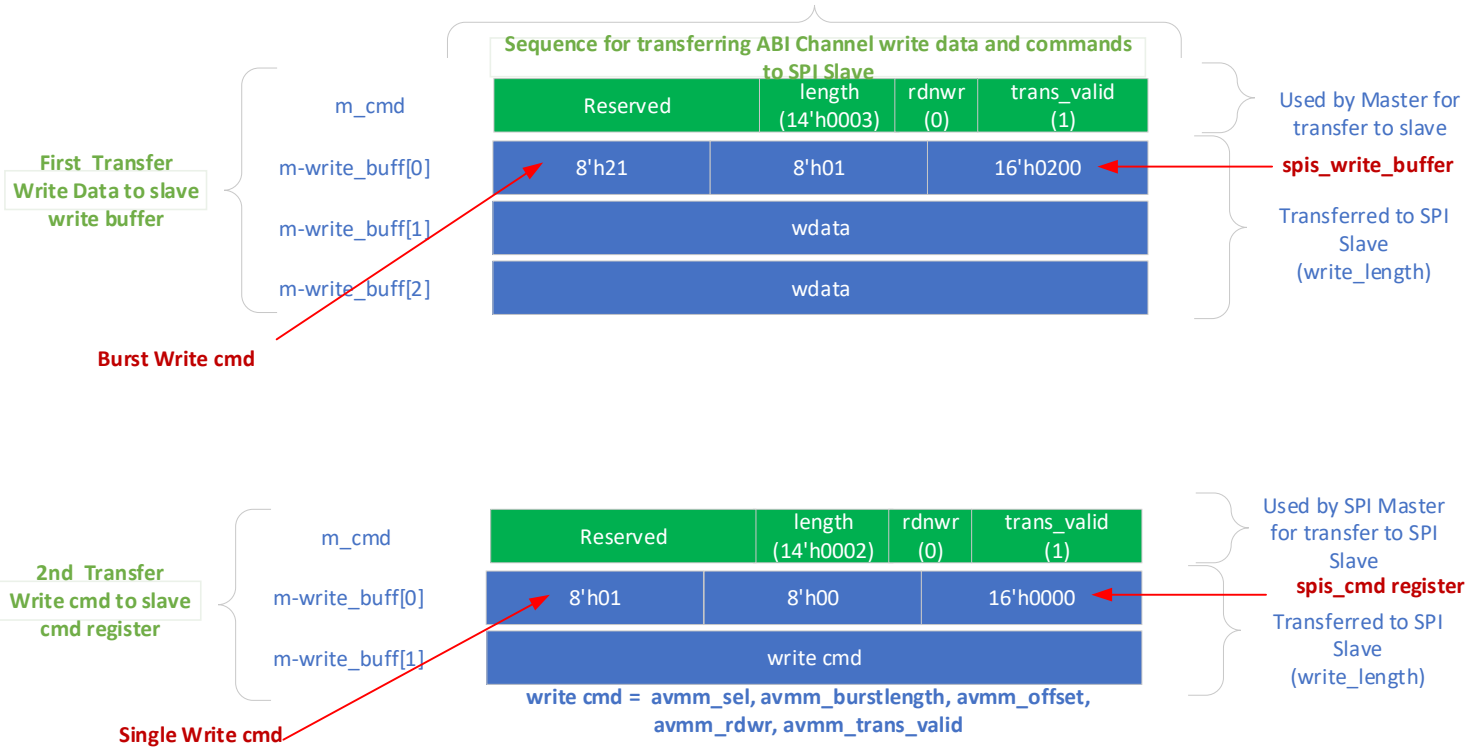
This sequence below shows how the initiator writes data 0x000000ff to address 0x9800(as an example) of AIB channel register

1. Initiator programs 0x000000ff in m_write_buffer (starting at 0x200).
2. Initiator then writes the following to m_cmd_register (offset 0x0)
    a. spi_slave_offset for s_write_buffer = 0x200
    b. rdnwr = 1
    c. Length = 0
    d. s_trans_valid = 1
3. This will trigger a SPI-M write to SPI-S to transfer contents of m_write_buffer to s_write_buffer.
4. Initiator polls on “s_trans_valid” in m_cmd. When s_trans_valid is 0, the SPI-M has completed the transfer to SPI Slave. The next command from the initiator can now be programmed.
5. Initiator programs the following in the m_write_buffer (starting at 0x200:
    a. Avmm interface selection = 1
    b. Burst length = 0
    c. AIB Channel ID = 010011
    d. AIB register offset = 0
    e. rdnwr = 0
    f. Avmm_trans_valid = 1
6. Initiator writes the m_cmd_register (offset 0x0) with
    a. spi_slave_offset for s_cmd =0x0

b. rdnwr = 0
c. length =0
d. s_trans_valid = 1

7. SPI-M will start a write to SPI-S to transfer the contents of m_write_buffer to s_cmd
8. Poll on “s_trans_valid” in m_cmd. When s_trans_valid is 0, the SPI-M has completed the transfer of the “command” to SPI Slave. SPI-S has the command and data to start the AVMM writes to the Channel.
9. SPI Slave IP initiates an AVMM write transfer to the AIB channel register addressed by the offset in s_cmd with the write the data in s_write_buffer
10. The initiator has been checking if the write command to the AVMM channel has been completed by polling the SPI-S’s “avmm_trans_valid” bit in the s_cmd register. To do this, the initiator programs the m_cmd register with:
    a. spi_slave_offset = 0x0
    b. rdnwr = 1
    c. s_trans_valid =1

    This triggers a SPI-M read of the s_cmd register and the read data is stored in m_read_buffer (offset 0x1000). When s_trans_valid = 0, the initiator reads the contents of the m_read_buffer to check if “avmm_trans_valid” = 1’b0. If avmm_trans_valid = 1’b0, then the write command to the ABI channel has been completed.



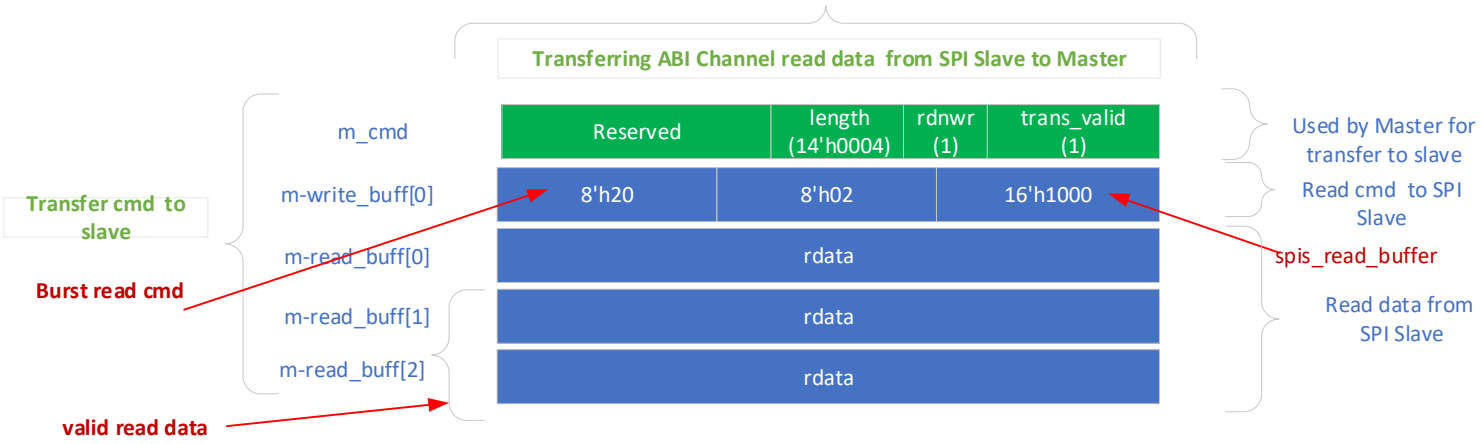
2.3.2 Read from initiator

This sequence below shows how the initiator reads data from address 0x9800(as an example) of AIB channel register

1. Initiator programs the channel read command in m_write_buffer (0x200)
    a. AVMM interface selection = 0
    b. Burst length = 0

c. AIB Channel ID = 010011
d. AIB register offset = 0
e. Read command = 1
f. Avmm_trans_valid = 1

2. Initiator then writes the following to m_cmd_register (offset 0x0)
   a. spi_slave_offset for s_cmd = 0x0
   b. rdnwr = 1
   c. Length = 0
   d. s_trans_valid = 1
3. This will transfer the contents of m_writer_buffer to s_cmd
4. Initiator polls on “s_trans_valid” in m_cmd. When s_trans_valid is 0, the SPI-M has completed the transfer of channel read command to SPI Slave. SPI-S has the command to start the AVMM read to the Channel
5. Slave IP initiates a read to the avmm_slave with the address and length in s_cmd (offset 0x0)
6. The read data from the Channel is stored in s_read_buffer at starting at offset 0x1000. Bit avmm_trans_valid in the s_cmd register is set to 0 when the AVMM slave read is complete.
7. The initiator now must check if the slave read from AVMM channel is complete by polling the SPI-S’s “avmm_trans_valid” bit in the s_cmd register. The initiator programs the m_cmd register with:
   a. spi_slave_offset =(0x0)
   b. rdnwr = 1
   c. s_trans_valid =1
8. This triggers a SPI-M read of the s_cmd register and the read data is stored in m_read_buffer (offset 0x1000). This is repeated till “avmm_trans_valid” is 0
9. The initiator must program the SPI Master to read the s_read_buffer in the SPI slave to transfer read date to m_read_buffer and programs the m_cmd register:
   a. spi_slave_offset for s_read_buffer = 0x1000
   b. rdnwr = 1
   c. Length = 0
   d. s_trans_valid = 1
10. Initiator polls on “s_trans_valid” in m_cmd. When s_trans_valid is 0, the SPI-M has completed the transfer of channel read data to the m_read_buffer
11. Initiator reads the data from m_read_buffer.



2.3.3 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_read_buffer (starting at offset 0x400)
3. NIOS (AVMM Master) then reads the contents of the m_read_buffer
4. NIOS (AVMM Master) writes the data (signature data) to the m_write_buffer (starting at offset 0x200)
5. NIOS (AVMM Master) sets up the write command in m_cmd to transfer the signature data to the slave.

the shift register are being transmitted.

3. SPI Slave Registers

Configuration register details and the buffer details are shown in Table 4 SPI Slave Configuration Registers below.

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	0x4	31:0	RO	reserved	
s_diag_0	0x8	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	0xC	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	
wbuf_fifo_status	0x40	0	RO	wr_overflow	Wrbuf overflow status – Sticky bit
		1	RO	rd_underflow	Wrbuf underflow status – Sticky bit
wbuf_fifo_control	0x44	0	RW	wbuf_soft_reset	Write 1 to reset. Write 0 to remove reset
rbuf_fifo_status	0x48	0	RO	wr_overflow	Rdbuf overflow status – Sticky bit

		1	RO	rd_underflow	Rdbuf underflow status – Sticky bit
rbuf_fifo_control	0x4C	0	RW	rbuf_soft_reset	Write 1 to reset. Write 0 to remove reset
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

SPI master configuration registers are shown in the Table 5 SPI Master Configuration Registers below.

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
		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/4	31:0	RO	reserved	
m_diag_0	0x10/8	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/c	31:3	RO	reserved	
		2:0	RO	avmm interface s/m state	
wbuf_rdback	0x20	31:0	RO		Holds NIOS write data to wbuf location 0x0200 only for read back
wbuf_fifo_status	0x40	0	RO	wr_overflow	Wrbuf overflow status – Sticky bit
		1	RO	rd_underflow	Wrbuf underflow status – Sticky bit
wbuf_fifo_control	0x44	0	RW	wbuf_soft_reset	Write 1 to reset. Write 0 to remove reset

rbuf_fifo_status	0x48	0	RO	wr_overflow	Rdbuf overflow status – Sticky bit
		1	RO	rd_underflow	Rdbuf underflow status – Sticky bit
rbuf_fifo_control	0x4C	0	RW	rbuf_soft_reset	Write 1 to reset. Write 0 to remove reset
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

The design assumes a frequency relationship between the spi clock and avmm clock in both the SPI Master and SPI Slave IPs. The minimum frequency for the AVMM clock should be 1/2 the frequency of SPI clock.

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