HB0924
Handbook
CoreQSPI v2.0

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

50200924. 1.0 11/20

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

This is the first publication of this document. Created for CoreQSPI v2.0.

2 Introduction

2.1 Overview

This CoreQSPI provides AHB system Interface and SPI interface to connect with the SPI memory devices. The control registers is used to configure the IP in different modes and the FIFO is used to buffer the data across clock domains.

2.2 Features

CoreQSPI supports the following features:

- Master SPI Data Rate
  - Programmable SPI clock HCLK/2, HCLK/4, or HCLK/6
  - Maximum data rate is HCLK/2
- FIFOs
  - Transmit and Receive FIFO
  - 16 byte transmit FIFO depth
  - 32 byte receive FIFO depth
- SPI Protocol
  - Master operation
  - Motorola SPI supported
  - Slave select operation in idle cycles configurable
  - Supports Extended SPI operation (1, 2, and 4-bit)
  - Supports QSPI operation (4-bit operation)
  - Supports BSPI operation (2-bit operation)
  - Support XIP (execute in place)
  - Supports three or four-byte SPI address.
- Frame Size
  - Supports 8- bit frames directly
  - Back to back frame operation supports >8-bit frames
  - Supports up to 4 GB Transfer (2**32 bytes)
- Direct Mode
  - Allows a CPU to directly control SPI interface pins.

2.3 Core Version

This handbook is for CoreQSPI version 2.0.

2.4 Supported Families

CoreQSPI supports the following families:

- PolarFire® SoC
- PolarFire®
- RTG4™
- IGLOO®2
- SmartFusion®2

2.5 Device Utilization and Performance

Device utilization and performance data is provided below for the supported device families.

Table 1 • **Device Utilization and Performance**

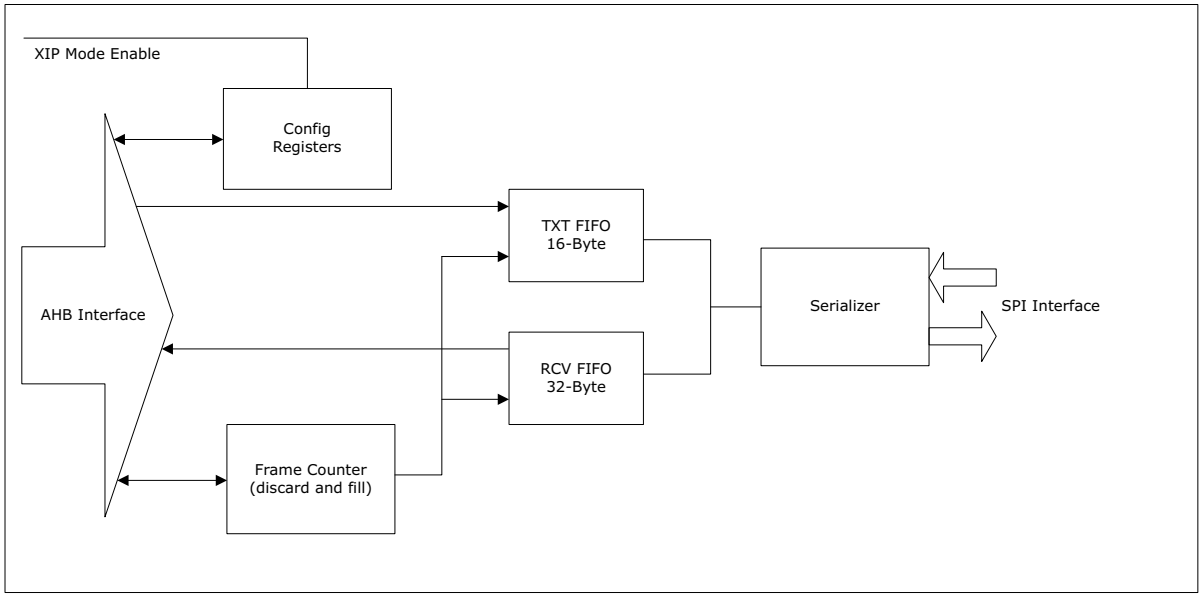
	FPGA Resources			Utilization		
Family	**Sequential**	**Combinational**	**Total**	**Device**	**Total (%)**	**Frequency (MHz)**
PolarFire SoC	531	1244	1775	MPFS250T	0.35	209
PolarFire	535	1437	1972	MPF100T	0.905	209
RTG4	548	1434	1982	RT4G150	0.65	111
IGLOO2	537	1244	1781	M2GL005	14.695	145
SmartFusion2	537	1244	1781	M2S005	14.695	145

Note: Data in this table is gathered using typical synthesis and layout settings. Throughput is computed as follows: (Bit width / Number of cycles) × Clock Rate (Performance).

3 Functional Description

The following figure shows a high-level block diagram of CoreQSPI.

Figure 1 • **High Level Block Diagram of CoreQSPI**



3.1 AHB Interface

The AHB Interface is used to control or configure the CoreQSPI in XIP Mode or SPI Mode data transfers. The XIP mode is enabled based on the init_* Input ports or can be configured through AHB interface as required. The firmware should disable the XIP mode by clearing the XIP bit in the CONTROL register, at this time it should not be executing from the QSPI device. Once this bit is written to zero the QSPI core returns to normal or SPI mode.

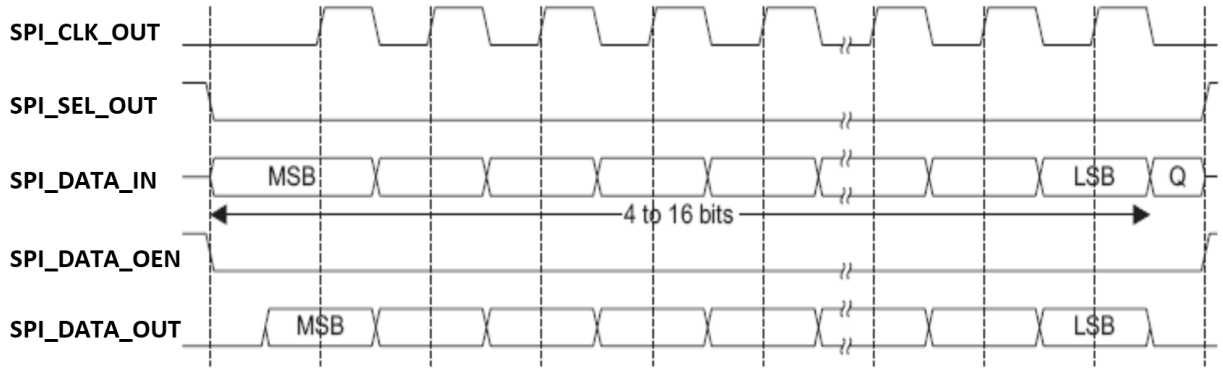
3.2 SPI Interface

The SPI Interface in the CoreQSPI supports standard Single or Normal SPI, extended dual and quad protocol Full BI /QUAD SPI operations. The extended SPI has RO and RW mode where in case of Extended RO mode the command and address bytes are driven in DQ0 alone and in Extended RW mode the command byte alone is driven in DQ0 for performance. The Full dual SPI and quad SPI protocols improve the data access time and throughput of a single I/O device by transmitting commands, addresses, and data across two or four data lines.

3.3 SPI Transfer Protocols

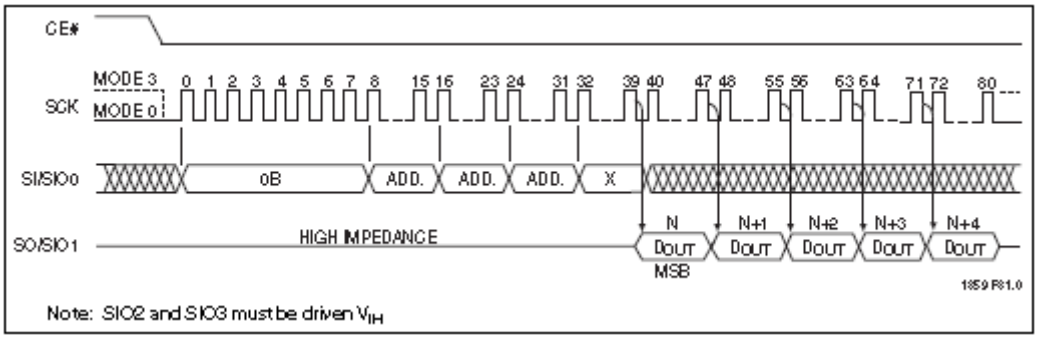
The CoreQSPI supports SPH=1 SPO=1 or SPOH=0 SPO=0 operation alone, the clock idle state being high or low with data generated on the negative SPI clock edge and sampled on the positive edge, this is the standard configuration for SPI memory devices. also the core is configured with SPS=1, meaning the slave select is held low until all frames are transmitted, each frame is 8-bits.

Figure 2 • **Basic SPI Protocol**



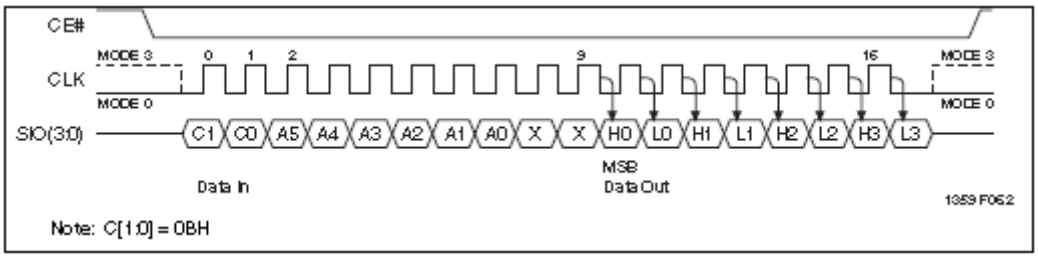
The following figure shows the SPI protocol being used to access a flash memory, you can see the command and address being transferred serially, and the data returned.

Figure 3 • **Single SPI**



This transfer takes in 80 clock cycles, when in QSPI mode 4-bits of data are transferred every clock cycle reducing to 20 clock cycles, this is shown below:

Figure 4 • **Quad SPI**



Also supported is Bi-SPI mode, in this case operation is the same as Q-SPI but only 2-bits of data are transferred every clock cycle, so the above waveform would take 40 cycles.

Note: On the waveforms the transmission length 4-16 bits should be read as multiples of 8-bits.

4 Interface

4.1 Configuration Parameters

The following table shows the parameters and supported features of CoreQSPI IP:

Table 2 • **Parameter or Generic Descriptions**

Parameter Name	Valid Range	Default	Description
SPI_MODE_EN	(0) / (1)	(1)	**SINGLE SPI** - Enable Single SPI DQ-out and DQ-in Operation Mode (Disable BI/QUAD SPI Operation mode). **EXTENDED/FULL_BI/QUAD_SPI** - Enable Extended/Full Bi/QUAD SPI Operation Mode.
XIP_MODE	YES/NO	YES	**YES** - DISABLE_XIP port - 1'b0, XIP_MODE - 1'b1 - Enable XIP Mode. **NO** - DISABLE_XIP port - 1'b1, XIP_MODE - 1'b0 - Disable XIP Mode.

All the configurable parameter or generic of the core are listed in the below table. All the parameters/generics are of integer type.

4.2 Ports

All the input and output ports of the core are listed in the table below.

Table 3 • **Input and Output Signals**

Port	Width	Direction	Description
AHB Interface			
HCLK	1	Input	AHB clock
HRESETN	1	Input	Active low asynchronous reset input.
HSEL	1	Input	AHB select
HWRITE	1	Input	AHB write
HTRANS	1	Input	AHB transfer
HSIZE[1:0]	2	Input	AHB size
HRESP	1	Output	AHB response
HADDR[23:0]	24	Input	AHB address bus
HWDATA[31:0]	32	Input	AHB write data bus
HRDATA[31:0]	32	Output	AHB read data bus
HREADY	1	Input	AHB ready input
HREADYOUT	1	Output	AHB ready output
HINTERRUPT	1	Output	SPI interrupt
SPI Interface			
SPI_CLK_OUT	1	Output	SPI: shift clock out (generated by SPI as master)

Table 3 • **Input and Output Signals**

Port	Width	Direction	Description
SPI_CLK_OEN	1	Output	SPI: clock output enable. This is active when the SPI is a master and enabled. This signal is active high.
SPI_SEL_OUT	1	Output	SPI: slave select, if necessary the SPI_CLK_OEN output can also be used as an output enable for the SSEL output.
SPI_DATA_IN[3:0]	3	Input	SPI: shift data in. Single / Normal SPI mode bit 1 is the data input. Bi-SPI mode bits 1:0 are used
SPI_DATA_OUT[3:0]	4	Output	SPI: serial data out. Single / Normal SPI mode bit 0 is the data output. BI-SPI / QUAD SPI mode bits 1:0 are used
SPI_DATA_OEN[3:0]	4	Output	SPI: output data enable active high. In normal data mode bit 0 is enabled whenever SPI_SEL_OUT is active. In Quad SPI mode all outputs are enabled whilst command bytes are clocked out and disabled for the return data. The outputs are also disabled for the last byte (two quad cycles) of the command sequence i.e. for the turnaround period. Bi-SPI mode SPI_DATA_OEN [3:2] are held permanently inactive.
SPI_RX_READY	1	Output	SPI: Indicates that data may be read from the SPI receive FIFO. Is de-asserted as the FIFO becomes empty, and asserted when either one or four bytes are available depending on RXDATAX4 Register offset 0x48
SPI_TX_READY	1	Output	SPI: Indicates that data may be written to the SPI transmit FIFO. Is de-asserted as the FIFO becomes full, and asserted when either one or four bytes can be written depending on TXDATAX4 Register offset 0x4C
XIP Interface			
DISABLE_XIP	1	Input	When asserted prevents XIP mode being invoked. **Note:** Both DISABLE_XIP XIP Interface Input and XIP_MODE Parameter should be enabled or disable for XIP mode support.
INIT_XIP	1	Input	If asserted at reset release then the controller starts up in XIP mode
INIT_CLKRATE	1	Input	Sets the clock divider value when in XIP mode
INIT_CLKIDLE	1	Input	Sets the clock idle value when in XIP mode
INIT_MODE[2:0]	3	Input	Sets how the core starts up in XIP mode 00x: Single wire SPI operation 010: Extended Bi-SPI RO operation 011: Extended Quad-SPI RO operation 100: Extended Bi-SPI RW operation 101: Extended Quad-SPI RW operation 110: Full Bi-SPI operation 111: Full Quad-SPI operation
INIT_IDLE[3:0]	4	Input	Sets the number of IDLE cycles used in XIP mode
INIT_XIPADDR	1	Input	Sets whether XIP mode uses three or four byte addressing. 0=3 Byte 1=4 Byte

5 Registers

This section describes the registers used in CoreQSPI. When in XIP mode only writes can be performed to the registers, read operations will return the SPI contents.

Table 4 • **Registers**

Address	Name	Size	Type	Reset Value	Function
0x00	CONTROL	31:0	RW	0x00000502	Control **Note:** The actual value varies based on the input INIT_MODE (assumed as 3'b000 and mentioned as reset value.)
0x04	FRAMES	31:0	RW	0x00000000	Frames
0x08	RESERVED		NA	0x00000000	Not used
0x0c	INTENABLE	6:0	RW	0x0000	Interrupt Enable
0x10	STATUS	31:0	RW	0x0000	Status and Interrupt
0x14	DIRECT	31:0	RW	0x0000	Direct Control
0x18	ADDRUP	7:0	RW	0x00	Sets forth address byte value in XIP mode.
0x40	RXDATAX1	7:0	RO	Unknown	Receive Data
0x44	TXDATAX1	7:0	WO	0x00	Transmit Data
0x48	RXDATAX4	31:0	RO	Unknown	Receive Data
0x4c	TXDATAX4	31:0	WO	0x00000000	Transmit Data
0x50	FRAMESUP	31:0	RW	0x00000001	Sets upper 16 bits off frame count.
0x54-0xFC	RESERVED	31:0	RO	0x00000000	RESERVED
Others	RESERVED	31:0	RO	0x00000000	RESERVED

5.1 Control Register (0x00)

Table 5 • **Control Register (0x00)**

Bits	Name	Reset	Description
0	ENABLE	0 XIP=1	0: Core will not respond to external signals until this bit enabled. 1: Core is active
1	RESERVED	1	1: Master
2	XIP	0 XIP=1	0: Normal operation 1: XIP mode
3	XIPADDR	0	Sets the number of bytes used for the address in XIP mode. 0 = 3 bytes 1 = 4 Bytes
9:4	RESERVED	0	
10	CLKIDLE	1 XIP= init value	Sets the clock IDLE to low (0) or high (1). Allows the core to support SPI memory operation in mode 0 or mode 3 When CLKRATE=0 this bit has no effect, the clock IDLE state will always be a '1'
12:11	SAMPLE	0 XIP=10	Sets when the core samples SDI in master mode 00: At the rising edge off SPI CLOCK, allowing half a SPI clock period round trip delay 01: On the last falling HCLK edge in the SPI clock period, at HCLK/2 allows 75% of the SPICLK period for the round trip delay plus 25% hold time 10: At the rising HCLK edge as SPICLK falls, allowing a full clock period round trip delay, hold time must be positive.
13	QSPIMODE[0]	XIP= init value	Sets whether multiple bit SPI uses 2 or 4 bits of data 0: 2-bits (BSPI) 1: 4-bits QSPI)
15:14	QSPIMODE[2:1]	XIP= init value	Sets whether multiple bit SPI operates in normal, extended or full modes: 00: Normal (single DQ0 TX and single DQ1 RX lines) 01: Extended RO (command and address bytes on DQ0 only) 10: Extended RW (command byte on DQ0 only) 11: Full. (command and address bytes are on all DQ lines)
16	FLAGSX4	0	This register bit should be set before x4 are used to read and write data. It modifies the FIFO status signals to indicate that four bytes may be read or written.
27:24	CLKRATE	1 XIP= init value	Sets the SPI clock rate (master mode) 0: Not Supported N: SPICLK = HCLK/(2*N) No can vary from 1-15 allowing rates from HCLK/2 to HCLK/30. (HCLK=250 8.3MHz-125MHz, HCLK=400 13.3MHz to 200MHz)

When using the X4 data operations an exact number of bytes MUST be written/read to/from the FIFOs and to the appropriate RX/TXDATA RX/TXDATA4 registers. The AHB/APB transaction to RX/TXDATA registers must have a byte size and the transactions to RX/TXDATA4 must have word size.

If the SPI sequence consists of 5 address bytes, followed by 35 data bytes then the five address bytes should be written using {1-word, 1-byte} or {1-byte, 1-word} or {5-bytes} and then the 35 data bytes using {8-words, 3-bytes} or {3-bytes,8-words}, or the total sequence as {9-words and 3-bytes} etc.

5.2 Frames Register (0x04)

This register is only functional in master mode. It is used to start a master transfer.

Table 6 • **Frames Register (0x04)**

Bits	Name	Reset	Description
15:0	TOTALBYTES	0000 XIP=10	Number of bytes to be sent/received bits [15:0], bits [31:16] of the frame count are set in the FRAMEUP register. This register once set will not change, an internal counter counts until it reaches the limit set by this register When this register is written the internal counter is also reset. When that counter reaches TOTALBYTES interrupts are generated and idle cycles inserted on the SPI bus (SSEL deactivated) The core will transmit data whenever the transmit FIFO is loaded, the counter is used to count bytes, terminate bursts and generate interrupts
24:16	CMDBYTES	0000 XIP=10	Sets the number of bytes For Transmit it includes SPI command, address, and data For Receive it includes SPI command and address alone For example: Page size 256 Bytes, 3 Byte addressing For Transmit TOT_BYTES [15:0] - 0x104 CMD_BYTES [24:16] - 0X104 For Receive TOT_BYTES [15:0] - 0x104 CMD_BYTES [24:16] - 0X4
25	QSPI	0 XIP=1	0: Normal Operation 1: QSPI Mode (2 or 4-bits data as set in control register)
29:26	IDLE	0	This sets the number of IDLE cycles to be inserted between the TX and RX SPI phases 0 to 15
30	FLAGBYTE	0	If set then the FIFO flags are set to byte mode This clears the FLAGX4 bit in the control register This bit always reads as zero
31	FLAGWORD	0	If set then the FIFO flags are set to word mode This sets the FLAGX4 bit in the control register This bit always reads as zero

This register must be written before each master command/data sequence is started, and when written the SPI master will enable the SPI master transfer sequence. It should not be written whilst a transfer is taking place. To change the FIFO flag setting mid transfer the CONTROL register should be used.

The number of IDLE **clock** edges between the address and RX data must be correctly set based on the SPI memory attached and the SPI command being used – this may also vary based on frequency and how the SPI memory is configured.

The FLAGBYTE/FLAGWORD bits allow the firmware to alter/force the FLAGSX4 control bit when the SPI transfer is started if the previous value is unknown or may not be at the required setting, without the overhead of writing to the control register.

5.3 FramesUP Register (0x50)

This register is only functional in master mode. It is used to set the upper 16-bits off the number off bytes to be transmitted when the required count is greater than 65535. It must be written before the FRAMES register.

Table 7 • **FramesUP Register (0x50)**

Bits	Name	Reset	Description
15:0	BYTESLOWER	0001	W: No Operation. R: Returns the lower 16-bits of the byte count from the FRAMES register
31:16	BYTESUPPER	0000	W: Writes the upper 16 bits of the number off bytes to be transmitted. R: Returns the written value

Writes to the lower 16-bits are ignored. When read the register returns the complete 32-bit byte count transfer.

5.4 Interrupt Enable Register (0x0C)

Table 8 • **Interrupt Enable Register (0x0C)**

Bits	Name	Reset	Description
0	TXDONE	0	Enables the external interrupt for the corresponding interrupt.
1	RXDONE	0	
2	RXAVAILABLE	0	
3	TXAVAILABLE	0	
4	RXFIFOEMPTY	0	
5	TXFIFOFULL	0	

The register enables interrupt generation when bits 5:0 off the status register are set.

5.5 Status Register (0x0C)

The Status register Indicates the status of the SPI core.

Table 9 • **Status Register (0x0C)**

Bits	Name	Reset	Description
0	TXDONE	0	Indicates that all the frames have been transmitted. Is cleared by writing a ‘1’ or when the FRAMES register is written.
1	RXDONE	0	Indicates that all the frames have been received and the RX FIFO is empty Is cleared by writing a ‘1’ or when the FRAMES register is written.
2	RXAVAILABLE	0	Indicates that there is at least one or four (FLAGSX4 set) received bytes available in the receive FIFO
3	TXAVAILABLE	1	Indicates that there is room for at least one or four (FLAGSX4 set) bytes in the transmit FIFO
4	RXFIFOEMPTY	1	Indicates that the RX FIFO is empty or contains less than 4 bytes when FLAGSX4 set

Table 9 • **Status Register (0x0C)**

Bits	Name	Reset	Description
5	TXFIFOFULL	0	Indicates that the TX FIFO is full or contains less than 4-empty locations when FLAGSX4 set
6	RESERVED	0	
7	READY	0	Indicates that the SPI master is IDLE and a new transmission may be started.
8	FLAGSX4	0	Read of CONTROL registers FLAGSX4. Allows Firmware to check the value at the same time as the status.

The RXAVAILABLE signal is RXFIFOEMPTY inverted and TXAVAILABLE is TXFIFOFULL inverted. The provision of AVALIABLE and EMPTY/FULL allows an interrupt be generated on the AVAILABLE or EMPTY/FULL condition.

5.6 Receive Data (0x40)

Table 10 • **Receive Data (0x40)**

Bits	Name	Type	Reset	Description
7:0	RXDATA	R	0	Received data – single byte

5.7 Transmit Data (0x44)

Table 11 • **Transmit Data (0x44)**

Bits	Name	Type	Reset	Description
7:0	RXDATA	W	NA	Data to send – single byte

5.8 X4 Receive Data (0x48)

Allows the processor to read 4-bytes from the SPI FIFO in a single transaction. Reads will return 4-bytes as APB reads are always words (PSTRB only used on writes).

Table 12 • **X4 Receive Data (0x48)**

Bits	Name	Type	Reset	Description
31:0	RXDATA4	R	0	Returns 4-bytes off receive data

5.9 X4 Transmit Data (0x4C)

Allows the processor to write 4-bytes from the SPI FIFO in a single transaction. Writes to this register must use word operations i.e. write 4-bytes at a time.

Table 13 • **X4 Transmit Data (0x4C)**

Bits	Name	Type	Reset	Description
31:0	TXDATA4	W	NA	Writes 4-bytes of data to send

When using the X4 data write operations the exact number of bytes MUST be written to the FIFO and to the appropriate TXDATA register, 0x44 for bytes and 0x4c for words. If the SPI sequence consists of 5 address bytes, followed by 35 data words then the five address bytes should be written using {1-word,1-

byte} or {1-byte,1-word} or {5-bytes} and then the 35 data words using {8-words,3-bytes} or { 3-bytes,8-words}, or the total sequence as {9-words and 3-bytes} etc

5.10 Direct Access (0x14)

This register allows direct access to the QSPI interface pins to support access to non-standard SPI devices via direct CPU control.

Table 14 • **Direct Access (0x14)**

Bits	Type	Reset Value	Field Name	Description
0	RW	0	EN_SSEL	Enable this register to drive SSEL
1	RW	0	OP_SSEL	Value driven on SSEL
2	RW	0	EN_SCLK	Enable this register to drive SCLK
3	RW	0	OP_SCLK	Value driven on SCLK
7:4	RW	0000	EN_SDO	Enable this register to drive SDATA, data and output enable
11:8	RW	0000	OP_SDO	Output value for SDATA
15:12	RW	0000	OP_SDOE	Output enables for SDATA
19:16	RO	-	IP_SDI	Input data on SDI
20	RO	0	Reserved	Reads as zero
21	RO	-	IP_SCLK	Current SCLK value
22	RO	-	IP_SSEL	Current SSEL value
23	RO	1	IDLE	Master SPI FSM is stalled/idle

The CPU can use this register to modify the QSPI interface signaling, for instance if the SSEL is signal is driven active (low) by this register then the CPU can simply send multiple single byte frames of data via the transmit FIFO and SSEL will be held active between frames. Alternatively it can bit-bang all the bits in this register to create whatever sequence is required.

5.11 Upper Address (0x18)

This register set the upper 8-bits of the address [31:24] when four byte addressing is being used in XIP mode.

Table 15 • **Upper Address (0x18)**

Bits	Name	Type	Reset	Description
7:0	RXDATA	R	0	Received data – single byte

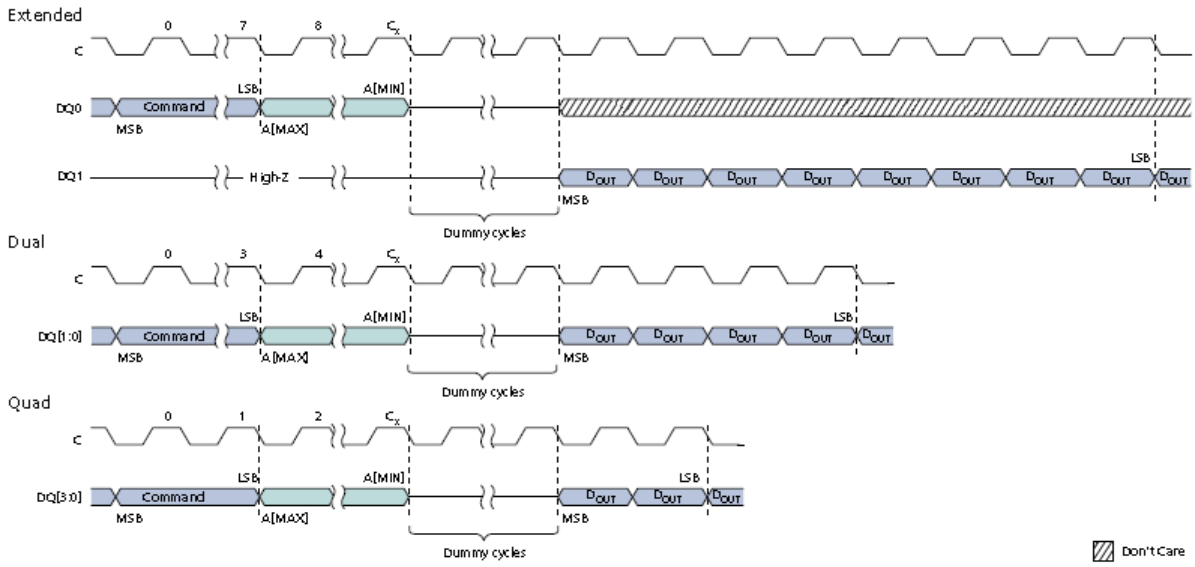
This register resets to zero which means that the initial XIP boot code must be located in the lower 16 Mbytes of the of the flash device.

6 Timing Diagrams

6.1 One Data Line Transfer

The following diagram shows an extended/normal single pad transfer, compared to dual and quad pad data transfer.

Figure 5 • **Extended Single Bit Protocol**

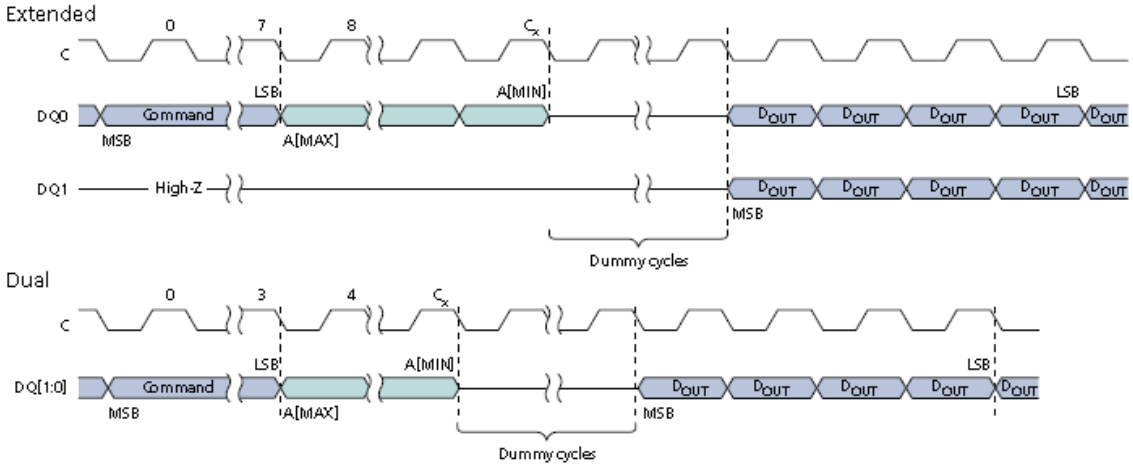


Note: 1. For extended protocol, $C_x = 7 + (A[MAX] + 1)$.
For dual protocol, $C_x = 3 + (A[MAX] + 1)/2$.
For quad protocol, $C_x = 1 + (A[MAX] + 1)/4$.

6.2 Extended RO – Dual Data Pad Transfer

The following diagram shows an extended dual pad transfer. In this case the Command and Address are sent on only DQ0 and the return data on DQ0 and DQ1.

Figure 6 • **Extended RO dual pad protocol**



Notes:
1. $C_x = 7 + (A[MAX] + 1)$.
2. Shown here is the DUAL OUTPUT FAST READ timing for the extended SPI protocol. The dual timing shown for the FAST READ command is the equivalent of the DUAL OUTPUT FAST READ timing for the dual SPI protocol.

The core **DOES NOT** directly support extended RO operation for SPI write commands where the address is transmitted serially and data is transmitted by the core in BI/QUAD mode to the SPI flash memory. There is a requirement that transmit bytes 2 to N are transmitted in the same format, i.e. serially (normal SPI) or in parallel (extended SPI RW), or in pure QUAD mode. Given that write operations are slow and in-frequent this is not a significant restriction.

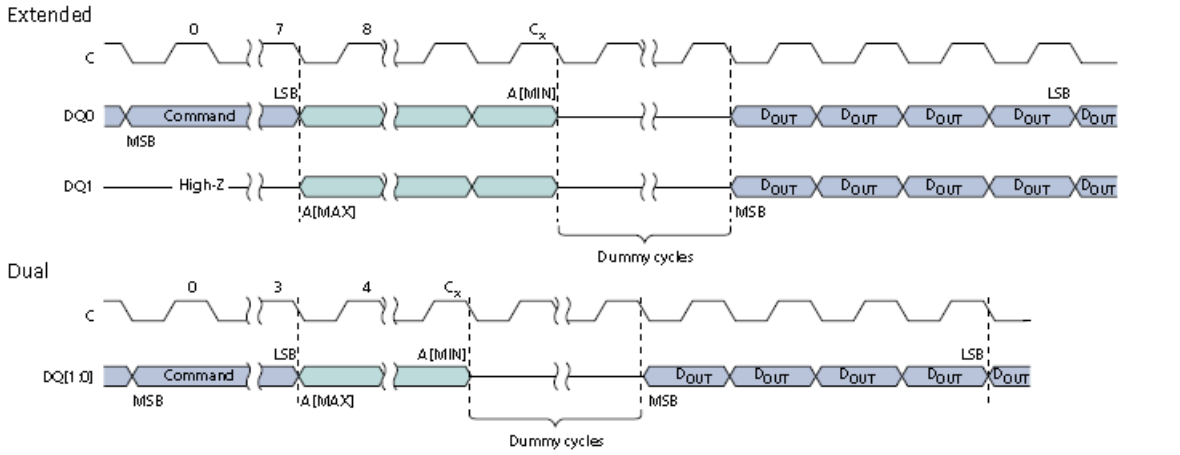
Extended RO operation for SPI write commands with parallel data **can be** performed as below:

1. Initially configure the core in extended RO operation
2. Write the command and address bytes to the transmit FIFO and wait until the IDLE bit in the DIRECT register is set. These bytes will have been transmitted serially.
3. Change the configuration to extended RW operation
4. Write the data bytes to the transmit FIFO, these bytes will be transmitted parallelly
5. On completion i.e. core is IDLE again change the configuration back to extended RO operation.

6.3 Extended RW – Dual Data Pad Transfer

The following diagram shows an extended dual rate transfer. In this case the Command is sent on only DQ0 and the address send on DQ0 and DQ1 return data on DQ0 and DQ1.

Figure 7 • **Extended RW Dual Data Pad Protocol**



Notes:
1. $C_x = 7 + (A[MAX] + 1)/2$.
2. Shown here is the DUAL INPUT/OUTPUT FAST READ timing for the extended SPI protocol. The dual timing shown for the FAST READ command is the equivalent of the DUAL INPUT/OUTPUT FAST READ timing for the dual SPI protocol.

The core supports extended RW operation for SPI write commands where the address and data are transmitted by the core in BI/QUAD to the SPI flash memory.

6.4 Extended RW – Quad Data Pad Transfer

Operation is as dual mode, but data is sent on four DQ lines.

7 Operations

7.1 Firmware or Programming Sequence

To send a command and retrieve data from a SPI flash memory the following sequence should be used.

1. Configure the CONTROL register if not already set to master operation.
2. Check the SPI master is idle, STATUS.READY is set, that is, a previous operation has completed, this step is not required; if it is known that the SPI core is idle.
3. The FRAMESUP register should be cleared or the upper 16-bits of the total byte count written. NOTE. The complete 32-bit value may be directly written to this register, this will correctly set the upper 16-bits of the byte count.
4. Write to the FRAMES registers
   - TOTALBYTES: Total number of bytes to be transferred, or the lower 16 bits.
   - CMDBYTES: Number of command bytes to be send
   - Make sure that the FIFO is configured in byte or word mode. If the data will be transferred using byte accesses than bit 30 should be set so the FIFO flags indicate byte availability, if words then bit 31 should be set so the FIFO indicates word availability.
5. Write the command bytes to the transmit FIFO using either byte of word transfers. The first 16-bytes can be directly written without checking the FIFO status. If there are more than 16-command bytes the STATUS.TXAVAILABLE bit should be checked before each additional data write to make sure that there is room in the FIFO.
6. Once the command bytes are written the read FIFO can be read whenever STATUS.RXAVAILABLE is set. If the DMA engine is being used to move data then when receive data is available the spi_rx_ready output is set to inform the DMA engine that data may be read
7. When the transfer is complete the STATUS.RXDONE interrupt will be generated as soon as all the data has been read from the FIFO.

7.2 Command and Data Configuration

The number of command and data bytes is controlled by the frames register for each SPI sequence. This supports the SPI flash memory read and writes sequences as below:

Table 16 • **Command and Data Configuration**

TOTAL BYTES [15:0]	CMD BYTES [24:16]	What Happens
1	1	The SPI core will transmit a single byte, all received data is discarded.
10	4	The SPI core will transmit four command bytes discarding the received data, and then transmit 6 dummy bytes returning the 6 received bytes.
10	10	The SPI core will transmit ten command bytes, all returned data is discarded.

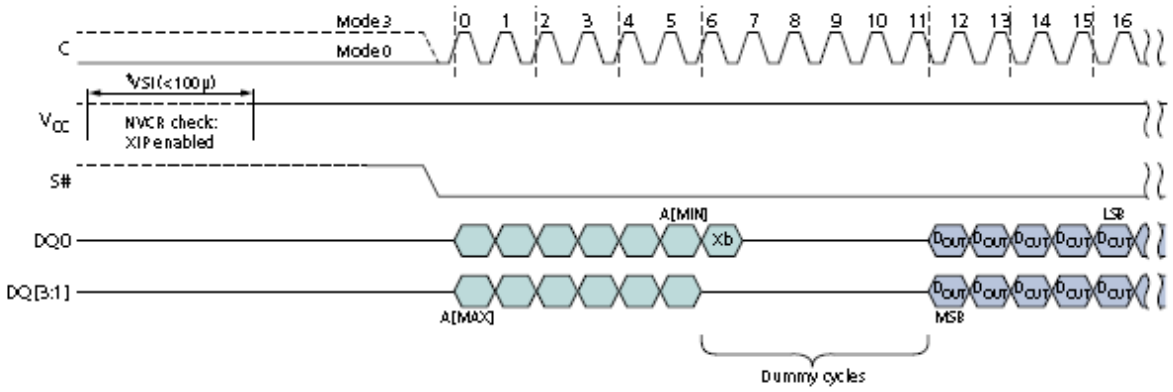
7.3 XIP Operation

7.3.1 Entry XIP Mode

Execute in place (XIP) allows a processor to directly boot from the QSPI device rather than have to move the SPI content to SRAM before execution. In this case a system configuration bit is used to set INIT_XIP input high indicating the CoreQSPI Controller in XIP mode, this is sampled on reset. Other configuration bits set the clock rate.

When in XIP mode all AHB reads simply return the 32-bit data value associated with the requested address. Each access to the QSPI device requires a 3-byte or 4-byte address transfers, a 3-byte IDLE period and 4-byte data transfer. Assuming the SPI clock is ¼ of the AHB (CPU) clock this then requires approximately 80 clock cycles per 32-bit read cycle. The QSPI device will stay in XIP mode as long as the Xb bit is zero.

Figure 8 • **XIP Transfer**



In XIP mode AHB write cycles will access the core registers allowing the values to be changed although the registers cannot be read when in XIP mode.

SIMPLY PUT: In XIP mode data is returned directly on the AHB bus in response to an AHB read, data is not read from the FIFO's. When in XIP mode the configuration registers cannot be read and all reads return SPI data. Thus, an AHB read from address 0x10 (status register) actually returns the contents of SPI address 0x14

7.3.2 Configuration of XIP Mode

If the XIP mode is enabled at reset (INIT_XIP active at reset) then the configuration registers are set to the values set on the INIT_xxxx input ports. This is typically used where the XIP device is used for the initial boot process.

The following registers need to be configured for XIP operation, refer to registers section for full details.

Table 17 • **XIP Mode Configuration**

Registers	Bits	Function	Description
CONTROL	2	xip	Enables XIP mode
CONTROL	3	xipaddr	Sets whether to use 3 or 4-byte address
CONTROL	15:13	mode	Sets whether in 1, 2 or 4-bit SPI mode
CONTROL	19	clkidle	Sets the idle value of the SPI clock
CONTROL	27:24	clkrate	Sets the SPI clock rate
FRAMES	29:26	idle	Sets the number of clocks between the address and data phases of XIP cycle. This can be considered as access time

The frames registers should be set before the XIP mode is entered by writing to the control register.

If a QSPI device requires a four byte times to start returning data (that is, eight clocks) then the FRAMES idle register should be programmed to 8.

7.3.3 Exit XIP Mode

To exit XIP mode the firmware should clear the XIP bit in the control register, at this time it should not be executing from the QSPI device. Once this bit is written to zero the QSPI core returns to normal mode and the reads access the core registers.

Once the QSPI core has exited XIP mode the SPI device needs to be taken out of XIP mode, this can be done by performing a read with an address in the first three command bytes, the fourth command byte 0x80 and the fifth and sixth[1] as 0x00, this will cause XIP mode to be exited. This sequence effectively sets the Xb shown in the XIP transfer figure above to 0x1 causing the QSPI device to exit XIP mode.

7.3.4 Throughput

The CoreQSPI core should be capable of reading a 64K byte data chunk in non XIP mode from the SPI device with 100% efficiently[2] assuming the AHB bus master is reading data from the FIFO without inserting IDLE cycles.

In QSPI-XIP mode the AHB read time will be less than TACC[3] cycles where:

TACC = 20 * PCLK/SPICLK

This read time will violate the AHB recommendation of a 16-cycle stall time i.e. HREADY de-assertion time. For accessing XIP memory this is expected and not a performance issue.

1. The sixth and final command byte will be not be output, during this period the SPI core tri-states the data bus for the turnaround cycle.

2. 100 % efficiently means that the SPI serial activity is continuous with the clock at the desired frequency with no idle cycles injected.

3.This is the number of cycles that the HREADY may be de-asserted.

8 Tool Flow

8.1 License

Obfuscated version will be license locked at the time of packaging.

8.2 SmartDesign

CoreQSPI is pre-installed in the SmartDesign IP Deployment design environment. The core can be configured using the configuration window in the SmartDesign, as shown in Figure 9.

To know how to create SmartDesign project using the IP cores, refer to *Libero SoC documents page* and use the latest SmartDesign user guide.

Figure 9 • **SmartDesign CoreQSPI XIP Mode**

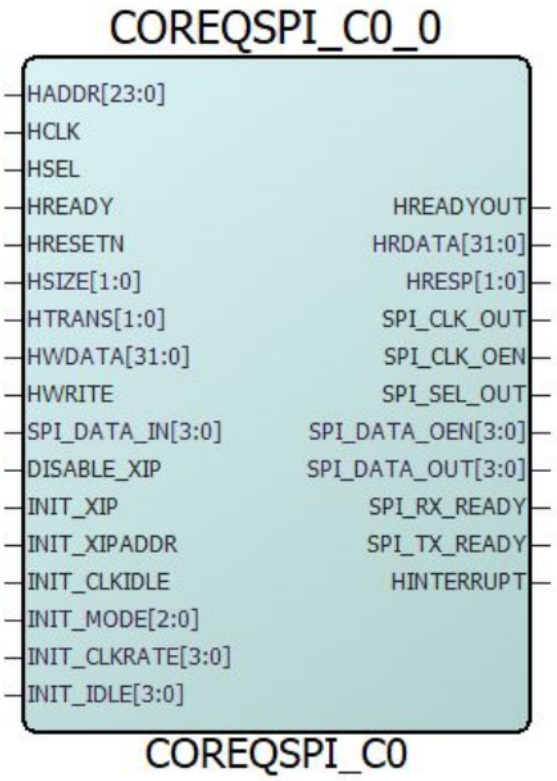
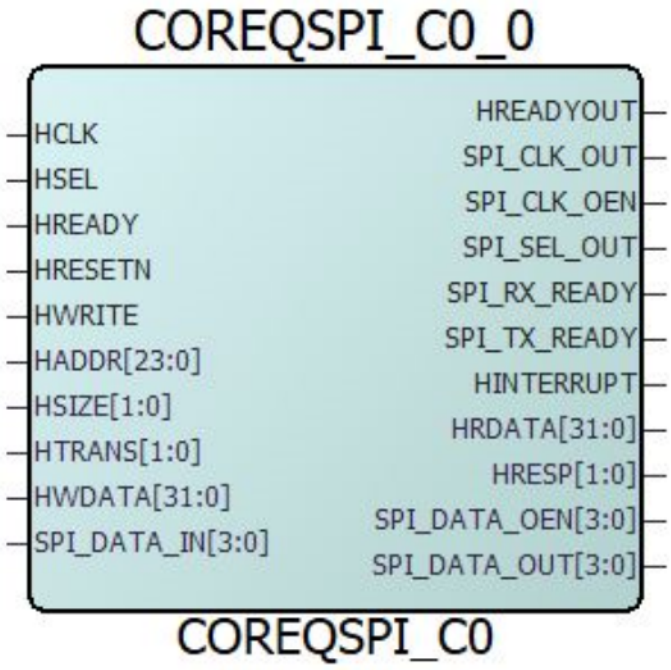


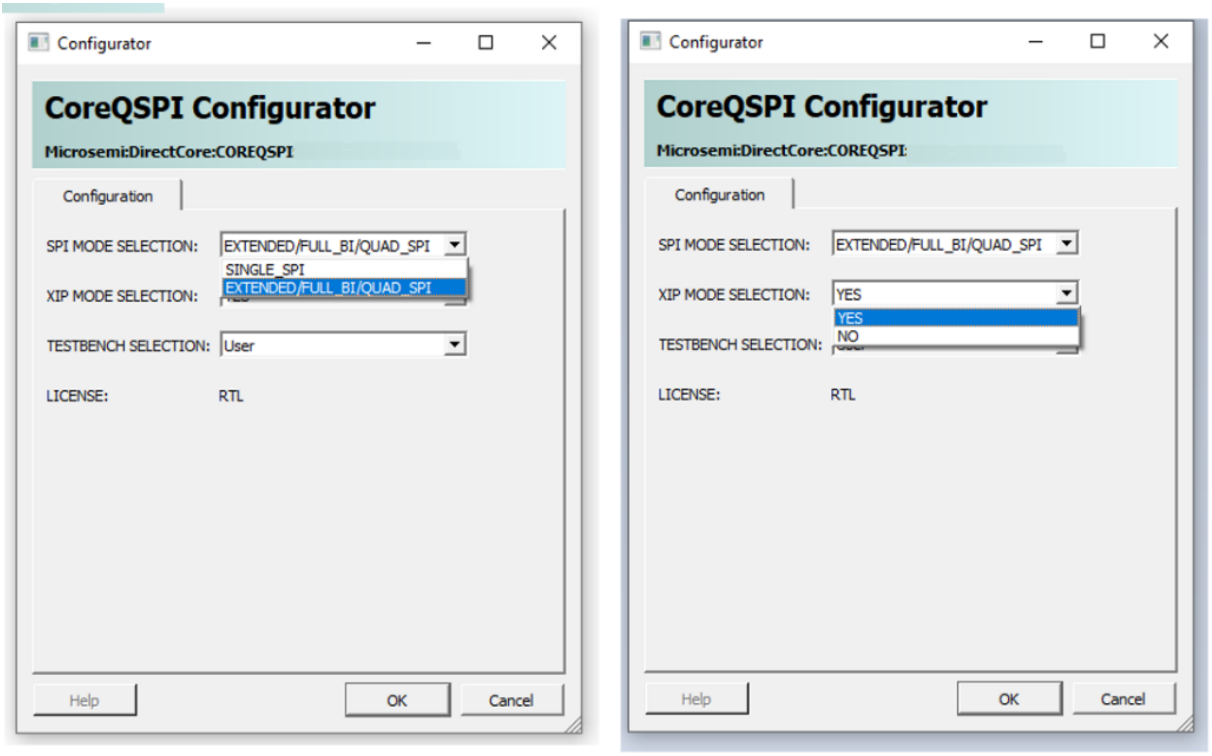
Figure 10 • **SmartDesign CoreQSPI without XIP Mode**



8.3 Configuring CoreQSPI in SmartDesign

The core is configured using the configuration GUI within SmartDesign.

Figure 11 • **Configuring CoreQSPI in SmartDesign**



For more information, see the *Knowledge Based article*.

To know how to create SmartDesign project using the IP cores, refer to *Libero SoC documents page* and use the latest SmartDesign user guide.

8.4 Simulation Flows

The User Testbench for CoreQSPI is included in all the releases.

To run simulations, select the **User Testbench** flow in the SmartDesign and click **Save and Generate** on the Generate pane.

The User Testbench is selected through the Core Testbench Configuration GUI. When SmartDesign generates the Libero SoC project, it installs the user testbench files.

To run the user testbench, set the design root to the CoreQSPI instantiation in the Libero SoC design hierarchy pane and click Simulation in the Libero SoC Design Flow window. This invokes ModelSim® and automatically runs the simulation.

8.5 Synthesis in Libero

Click **Synthesis** in the Libero software. The Synthesis window displays the Synplify® project. Set Synplify to use the Verilog 2001 standard, if Verilog is used. To run the synthesis, click **Run**.

8.6 Place-and-Route in Libero

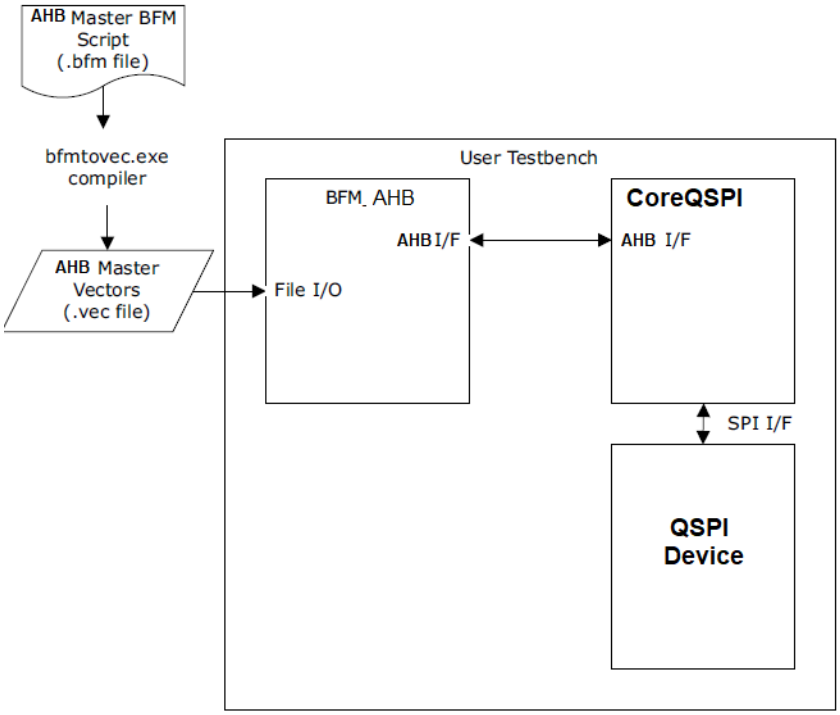
Click **Layout** in the Libero software to invoke the Designer. CoreQSPI does not require any special place-and-route settings.

9 Testbench

A user testbench is provided with the core which verifies few of the features.

The following figure shows the simulation testbench. It includes and instantiates the CoreQSPI macro and a BFM-based AHB-driven block. All BFM compilers are included for both the Linux and Windows operating system as well as the BFM vector source files. These are editable.

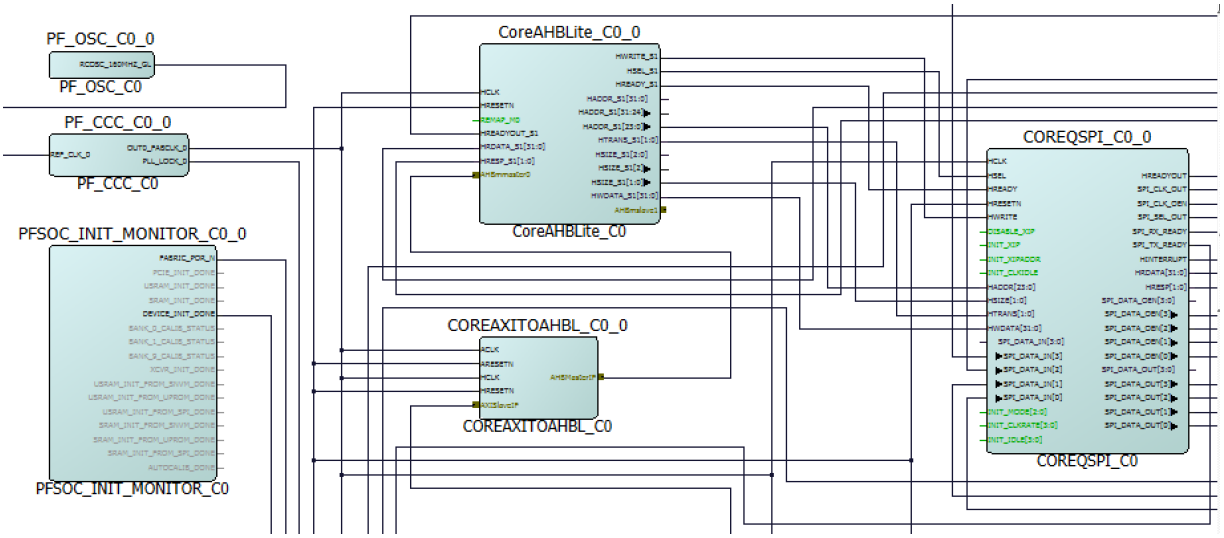
Figure 12 • **CoreQSPI User Test-bench**



10 System Integration

This section provides hints to ease the integration of the core.

Figure 13 • **CoreQSPI System Integration**



The above smart design shows the Integration of CoreQSPI. Here The PF_OSC and PF_CCC are used for clocking and PFSOC_INIT_MONITOR for Initialization done. The external AHB Master is used to configure the CoreQSPI for different mode of data transfers.