Dolphin Technology

Digital Design Courses

UART/SPI Controller

Design Requirements

Revision 0.0.1

19 February 2025

Design Requirements

Digital Design Courses

UART/SPI Controller

Revision 0.0.1

19 February 2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

REVISION HISTORY

Revision	Date	Description of Changes
0.0.1	19 February 2025	Initial

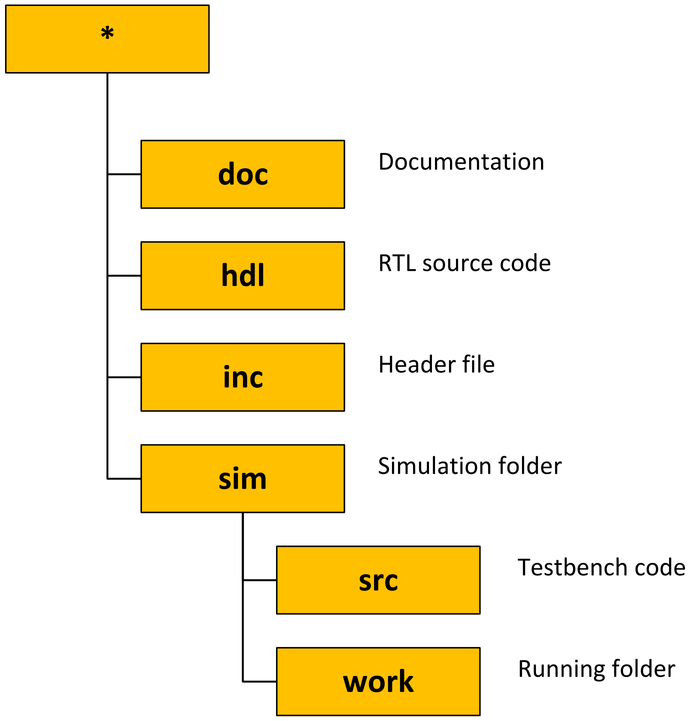
1. General requirements

Each group can have a maximum of **3 members.**

Submission deadline: **11:59 PM, May 10, 2025**

Packaging Requirements:

Directory Structure:



Contents:

- **Specification file** describing the design
- **RTL source code**
- **Testbench source code**

2. APB_UART

Simplified Universal Asynchronous Receiver/Transmitter (UART)

Description:

apb_uart is a simplified UART APB controller that performs two main tasks:

- Converts serial data received from a peripheral device into parallel data sent to the host (CPU).
- Converts parallel data received from the host into serial data sent to a peripheral device.

The host can read the UART operation status at any time.

2.1. Requirements

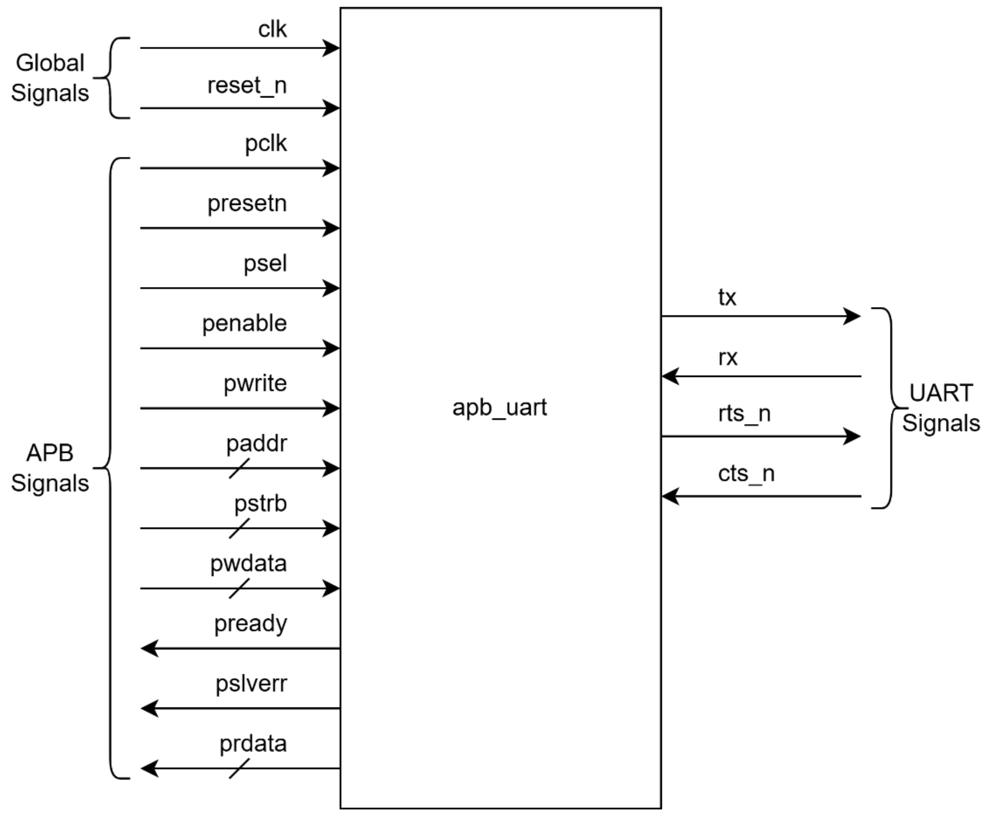
Supports baud rate values: 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200, 230400 (configurable via parameters).

Supports clear-to-send (CTS) and request-to-send (RTS) handshaking signals (automatic flow control).

Programmable UART frame (configured via APB registers), including:

- Number of data bits: 5 to 8
- Number of parity bits: 0 or 1
- Parity type (if used): odd or even
- Number of stop bits: 1 or 2

Operation control via the APB protocol.



-

Figure 1. apb_uart block diagram

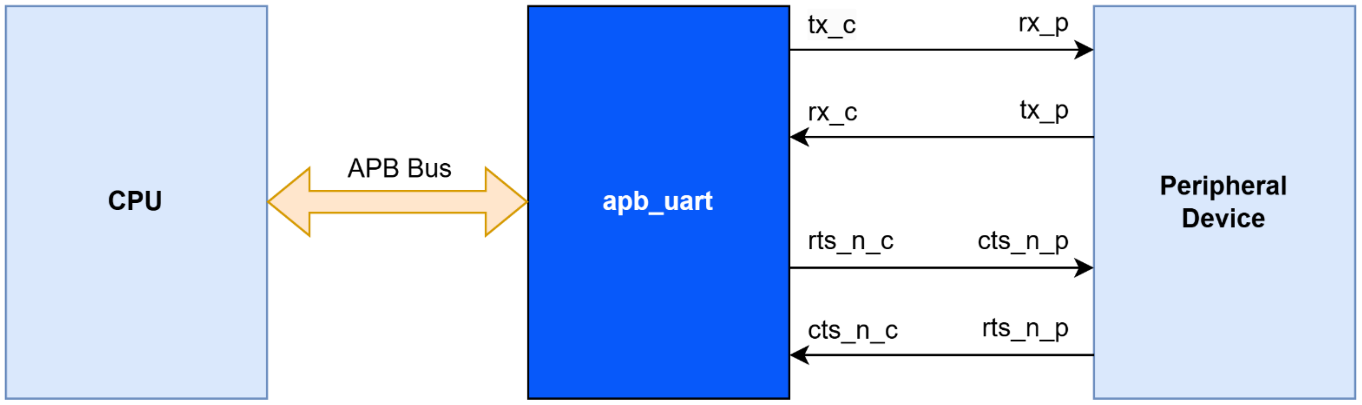


Figure 2. System using apb_uart

Table 1. apb_uart parameter description

Parameter	Value	Description
BAUDRATE	`CFG_BAUDRATE Default: 115200	Baud rate value

Table 2. apb_uart signals description

Signal Name	Width	I/O	Description
Global Signals			
clk	1	Input	System clock signal
reset_n	1	Input	System asynchronous reset, active LOW
APB Completer Signals			
pclk	1	Input	APB clock signal
presetn	1	Input	APB reset signal, active LOW, connected directly to the system reset
psel	1	Input	Select, indicates that the APB completer is selected and that a data transfer is required
penable	1	Input	Enable, indicates the second and subsequent cycles of an APB transfer
pwrite	1	Input	Direction, indicates an APB write access when HIGH and an APB read access when LOW
pstrb	4	Input	Write strobe, indicates which byte lanes to update during write transfer
paddr	12	Input	APB address bus
pwdata	32	Input	APB write data
pready	1	Output	Ready, is used to extend an APB transfer by the completer
pslverr	1	Output	Transfer error
prdata	32	Output	APB read data
UART Signals			
rx	1	Input	Serial data receive
cts_n	1	Input	Clear-to-send handshaking signal
tx	1	Output	Serial data transmit
rts_n	1	Output	Request-to-send handshaking signal

2.2. Functional Description

apb_uart operates based on its register block, which provides information about the data to be processed, UART frame configuration, control signals, and status signals. For more details, refer to **Table 3**.

The host can control and monitor the module's operation by accessing its register block via the APB protocol. **apb_uart** uses the values programmed into its registers by the host to execute the appropriate data transmission/reception process with a peripheral device through the UART protocol.

Each UART transaction transmits or receives one character, where the bit width is equal to the configured number of data bits. **Figure 2** illustrates how **apb_uart**, the peripheral device, and the host are connected. The control flows of the module are shown in **Figures 3 and 4**.

Note:

- System clock signal and APB clock signal would use the same frequency.
- In case the number of data bits is smaller than **tx_data**/**rx_data** bitwidth, most significant bits of **tx_data**/**rx_data** would be masked. For example, when the number of data bits is 5, only **tx_data**[4:0]/ **rx_data**[4:0] are valid.

2.3. Registers Block

Table 3. apb_uart's register block description

Register Name	Register Description	Address	Register SW Access	Field Name	Field Reset Value	Position	Field Description
tx_data_reg	TX data register	0x0	RW	tx_data	0	[7:0]	Parallel data from the host which is converted to serial data sent to peripheral device
				rfu	0	[31:8]	Reserved for future use
rx_data_reg	RX data register	0x4	RO	rx_data	0	[7:0]	Parallel data archived after serial-to-parallel conversion on data received on peripheral device
				rfu	0	[31:8]	Reserved for future use
cfg_reg	UART frame configuration register	0x8	RW	data_bit_num	0	[1:0]	The number of data bits 2'b00: 5 bits 2'b01: 6 bits 2'b10: 7 bits 2'b11: 8 bits
				stop_bit_num	0	[2]	The number of stop bits 1'b1: 1 bits 1'b0: disable
				parity_en	0	[3]	Parity enable 1'b1: enable 1'b0: disable
				parity_type	0	[4]	Parity type select 1'b1: Even 1'b0: Odd
				rfu	0	[31:5]	Reserved for future use
ctrl_reg	Operation control Register	0xC	RW	start_tx	0	[0]	Set to start converting parallel data received from the host to serial data sent to peripheral device
				rfu	0	[31:1]	Reserved for future use
stt_reg	Operation status register	0x10	RO	tx_done	1'b1	[0]	Set to indicate that previous parallel-to-serial conversion on data received from the host is completed
				rx_done	0	[1]	Set to indicate that previous serial-to-parallel conversion on data received on peripheral device is completed
				parity_error	0	[2]	A parity error occurs when the parity of the received

							character does not match the calculated parity
				rfu	0	[31:3]	Reserved for future use

2.4. Control Flow

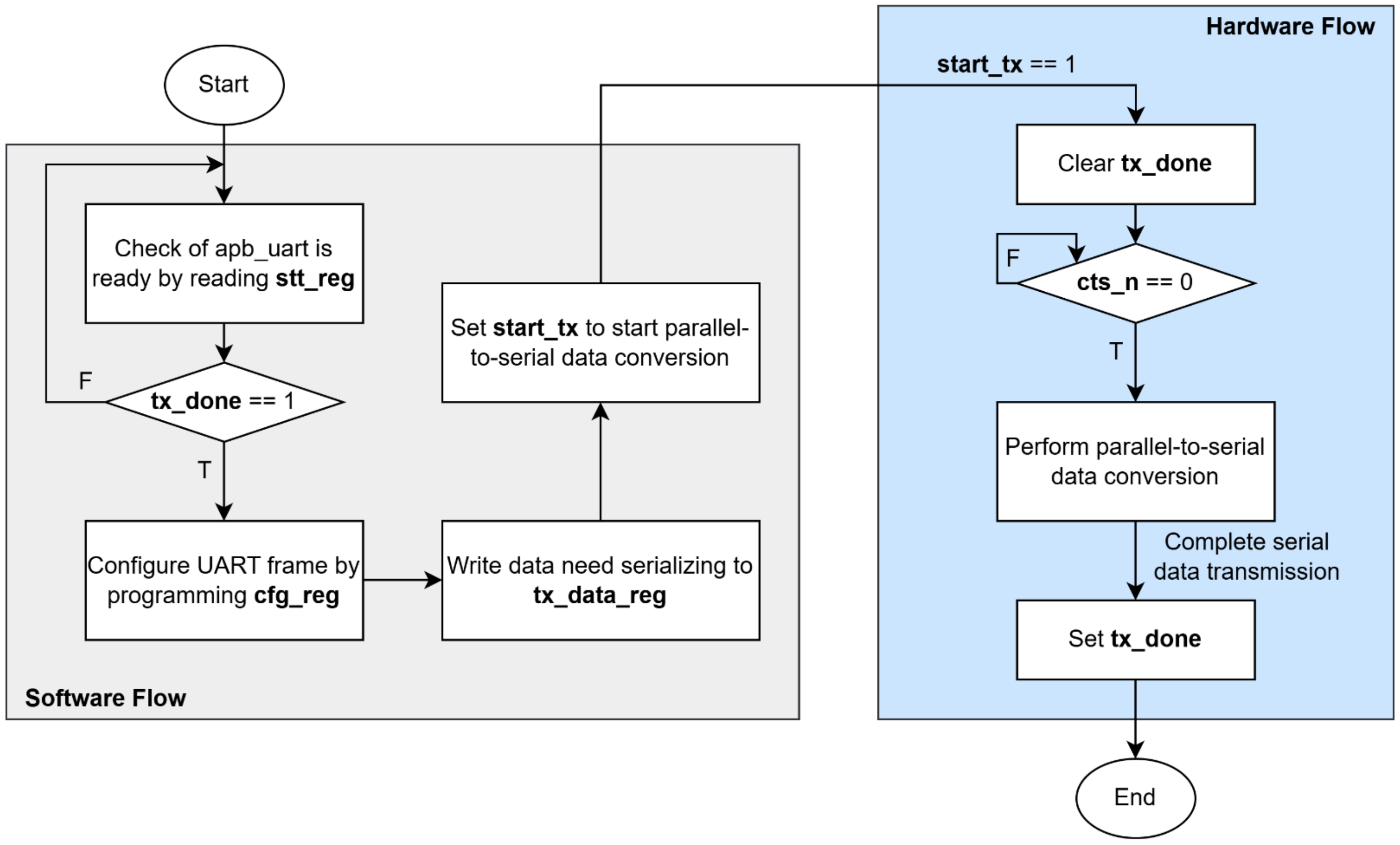

Start
Check of apb_uart is ready by reading **stt_reg**
tx_done == 1
F
T
Configure UART frame by programming **cfg_reg**
Write data need serializing to **tx_data_reg**
Set **start_tx** to start parallel-to-serial data conversion
Software Flow
Hardware Flow
start_tx == 1
Clear **tx_done**
F
cts_n == 0
T
Perform parallel-to-serial data conversion
Complete serial data transmission
Set **tx_done**
End

Figure 3. Parallel-to-serial data conversion control flow

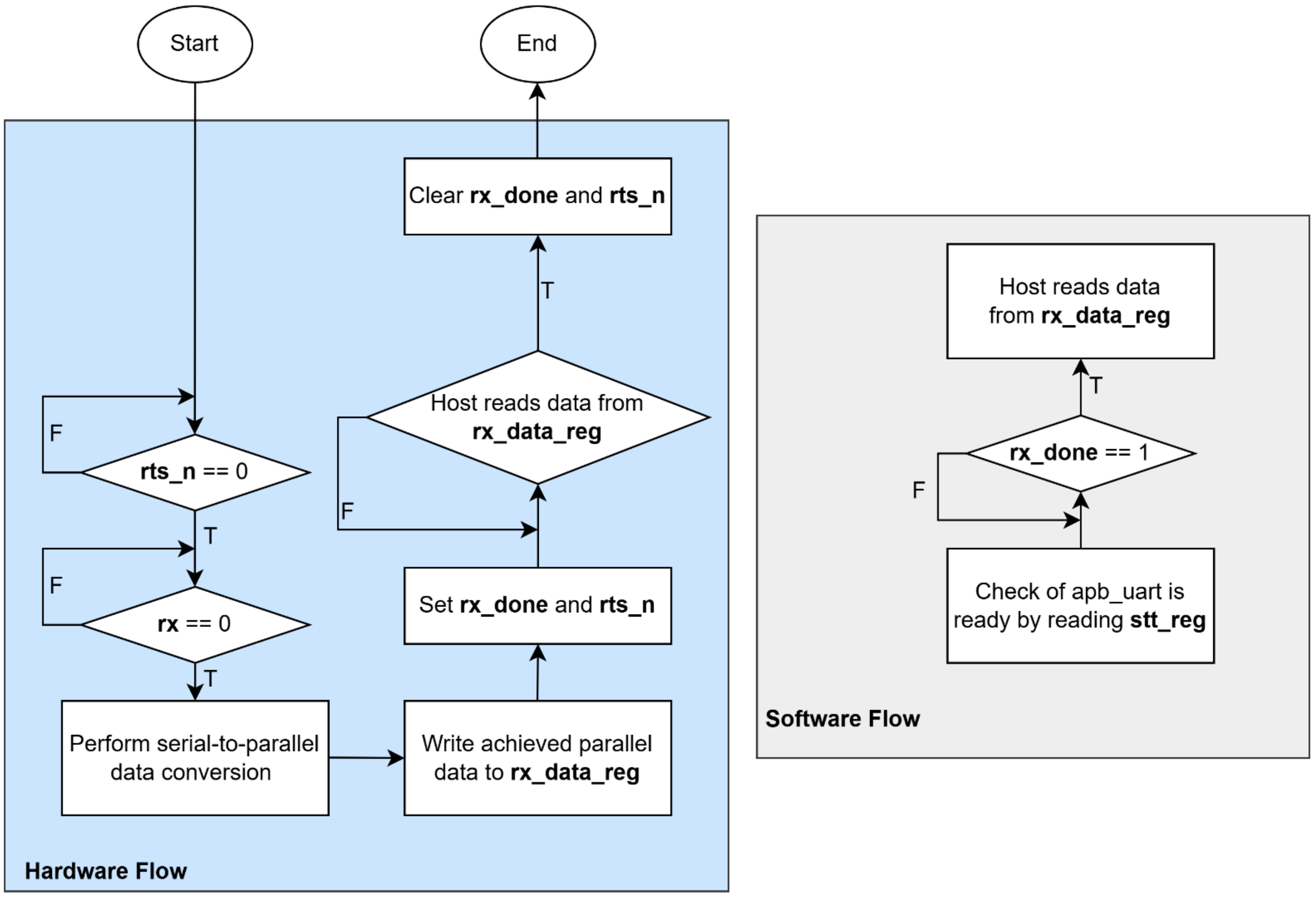

Start
End
Clear **rx_done** and **rts_n**
T
Host reads data from **rx_data_reg**
F
F
rts_n == 0
T
F
rx == 0
T
Set **rx_done** and **rts_n**
Perform serial-to-parallel data conversion
Write achieved parallel data to **rx_data_reg**
Hardware Flow
Host reads data from **rx_data_reg**
T
rx_done == 1
F
Check of apb_uart is ready by reading **stt_reg**
Software Flow

Figure 4. Serial-to-parallel data conversion control flow

3. APB_SPI

Serial Peripheral Interface (SPI)Controller.

Description:

apb_spi is a simplified SPI Master controller performing two main tasks below:

- Converts serial data received from a peripheral device to parallel data sent to the host (CPU)
- Converts parallel data received from the host to serial data sent to a peripheral device

The host can read the SPI operation status at any time.

3.1. Requirements

- Supports number of SPI SLAVES: 1, 2, 4 (parameter configuration)
- Programmable SPI configuration (registers configuration using APB protocol), including:
    - Polarity of the clock (CPOL)
    - Timing of the data bits relative to the clock pulses (CPHA)
    - Direction Select
    - Only Master mode

Operation control using APB protocol.

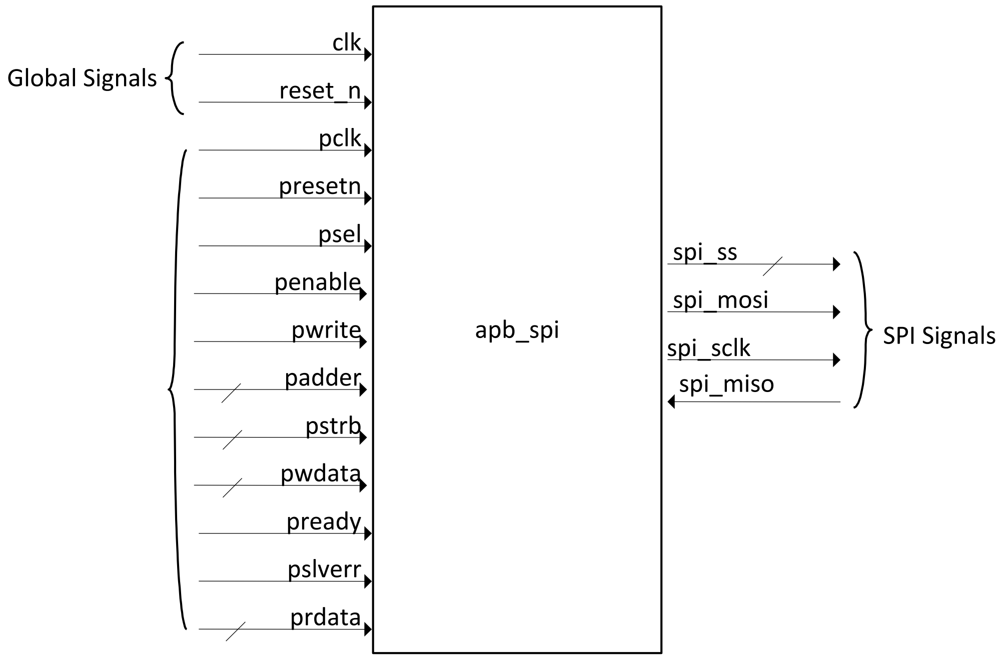


Figure 5. apb_spi block diagram

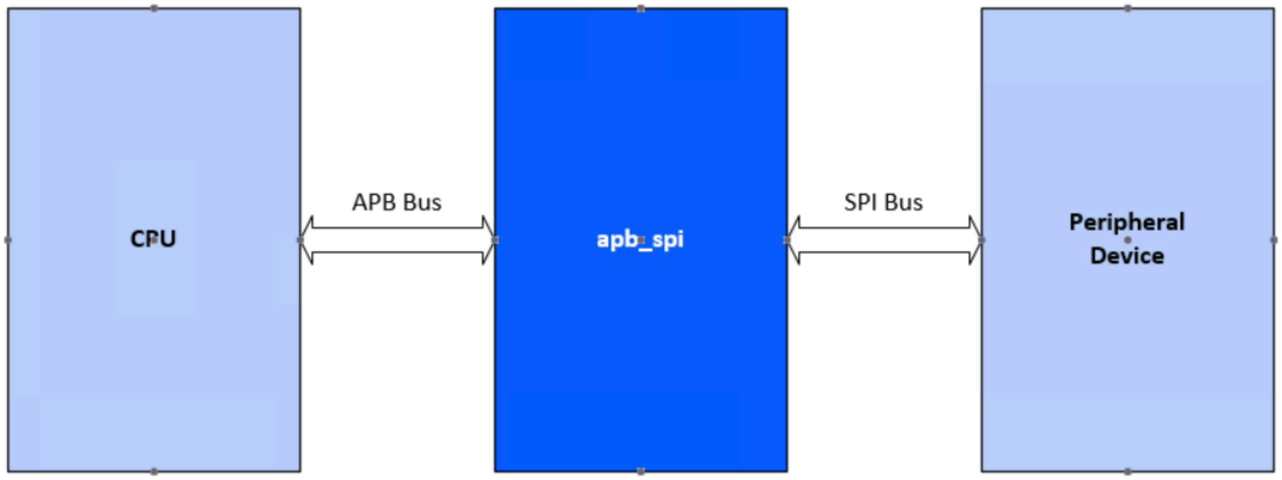


Figure 6. System using apb_spi

Table 4. apb_spi parameter description

Parameter	Value	Description
SPI_SLAVE_NUM	` CFG_SPI_SLAVE_NUM Default: 2	Number of SPI Slaves

Table 5. apb_spi signals description

Signal Name	Width	I/O	Description
Global Signals			
clk	1	Input	System clock signal
reset_n	1	Input	System asynchronous reset, active LOW
APB Completer Signals			
pclk	1	Input	APB clock signal
presetn	1	Input	APB reset signal, active LOW, connected directly to the system reset
psel	1	Input	Select, indicates that the APB completer is selected and that a data transfer is required
penable	1	Input	Enable, indicates the second and subsequent cycles of an APB transfer
pwrite	1	Input	Direction, indicates an APB write access when HIGH and an APB read access when LOW
pstrb	4	Input	Write strobe, indicates which byte lanes to update during write transfer
paddr	12	Input	APB address bus
pwdata	32	Input	APB write data
pready	1	Output	Ready, is used to extend an APB transfer by the completer
pslverr	1	Output	Transfer error
prdata	32	Output	APB read data

SPI Signals			
spi_ss	SPI_SLAVE_NUM	Output	SPI Slave Select (In Master Mode)
spi_mosi	1	Output	SPI Master Out, Slave In (In Master Mode)
spi_miso	1	Input	SPI Master In, Slave Out (In Master Mode)
spi_sclk	1	Output	SPI Serial Clock (In Master Mode)

3.2. Functional Description

apb_spi operates based on its register block, which provides information about the data to be processed, SPI frame configuration, control signals, and status signals. For more details, refer to **Table 6**.

The host can control and monitor the module's operation by accessing its register block via the APB protocol. **apb_spi** uses the values programmed into its registers by the host to execute the appropriate data transmission/reception process with a peripheral device through the SPI protocol.

Figure 6 illustrates how **apb_spi**, the peripheral device, and the host are connected. The control flows of the module are shown in **Figure 7**.

Note:

- System clock signal and APB clock signal would use the same frequency.

3.3. Registers Block

Table 6. apb_spi's register block description

Register Name	Register Description	Address	Register SW Access	Field Name	Field Reset Value	Position	Field Description
tx_data_reg	TX data register	0x0	RW	tx_data	0	[7:0]	Transmit Data
				rfu	0	[31:8]	Reserved for future use
rx_data_reg	RX data register	0x4	RO	rx_data	0	[7:0]	Receive Data
				rfu	0	[31:8]	Reserved for future use
cfg_reg	SPI configuration Register	0x8	RW	ctrl_cpol	0	[0]	Polarity of the clock 0: Idled clock value is 0 1: Idled clock value is 1
				ctrl_cpha	0	[1]	Timing of the data bits relative to the clock pulses 0: Tx on trailing edge, Rx on leading edge 1: Tx on leading edge, Rx on trailing edge
				ctrl_order	0	[2]	Order of data transfer Sets the order of data transfer. When this bit is logic 1, the LSB is transmitted first. When this bit is logic 0, the MSB is transmitted first
				ctrl_slave_en	0	[6:3]	Slave Select (One-hot)
				ctrl_rd	0	[7]	Read Data from Slave 0: Ignore Data from Slave 1: Get Data from Slave
				ctrl_scks	0	[9:8]	Select the serial data clock frequency of the spi_sclk signal 2'b00: fclk/4 2'b10: fclk/16 2'b01: fclk/8 2'b11: fclk/32
				rfu	0	[31:10]	Reserved for future use
ctrl_reg	Operation control Register	0xC	RW	start_op	0	[0]	Set **start** for apb_spi to transmit/receive data
				rfu	0	[31:1]	Reserved for future use
stt_reg	Operation status register	0x10	RO	busy	1'b0	[0]	Indicates that the Master is busy. If this bit is at logic 1, the CoreSPI Master is currently transmit/receive data.
				rfu	0	[31:1]	Reserved for future use

3.4. Control Flow

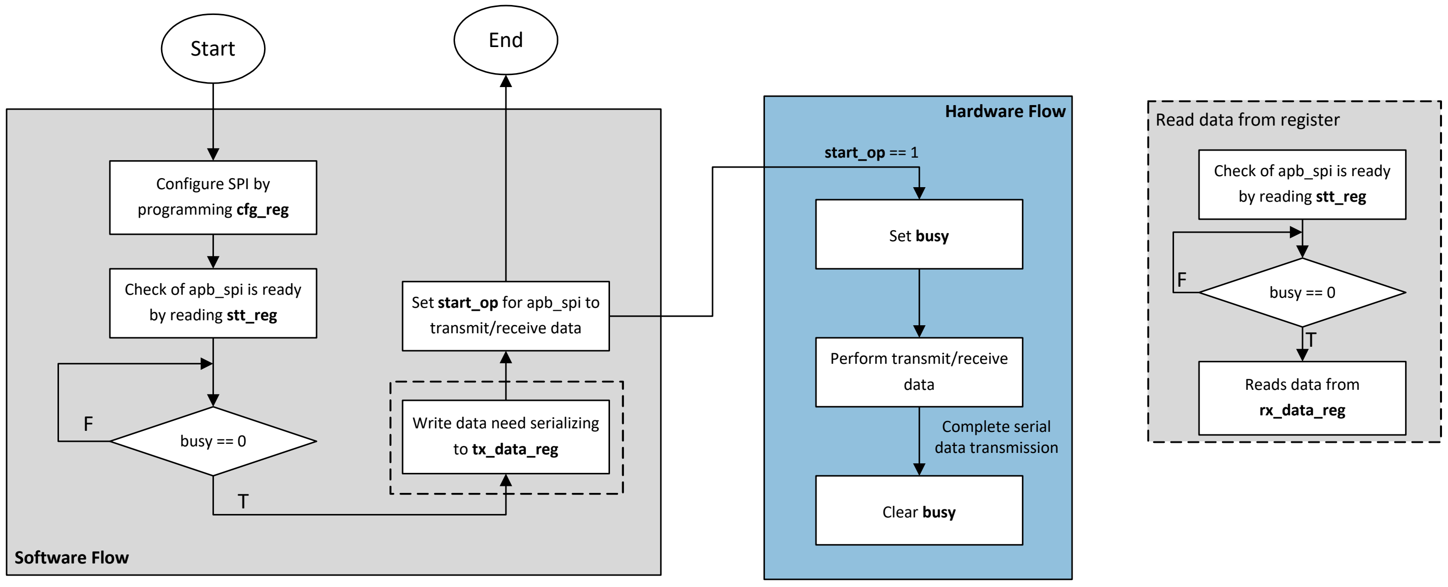

Start
End
Software Flow
Configure SPI by programming cfg_reg
Check of apb_spi is ready by reading stt_reg
busy == 0
F
T
Write data need serializing to tx_data_reg
Set start_op for apb_spi to transmit/receive data
Hardware Flow
start_op == 1
Set busy
Perform transmit/receive data
Complete serial data transmission
Clear busy
Read data from register
Check of apb_spi is ready by reading stt_reg
busy == 0
F
T
Reads data from rx_data_reg

Figure 7. APB SPI flow control