SPI Master Controller

October 2012 Reference Design RD1141

Introduction

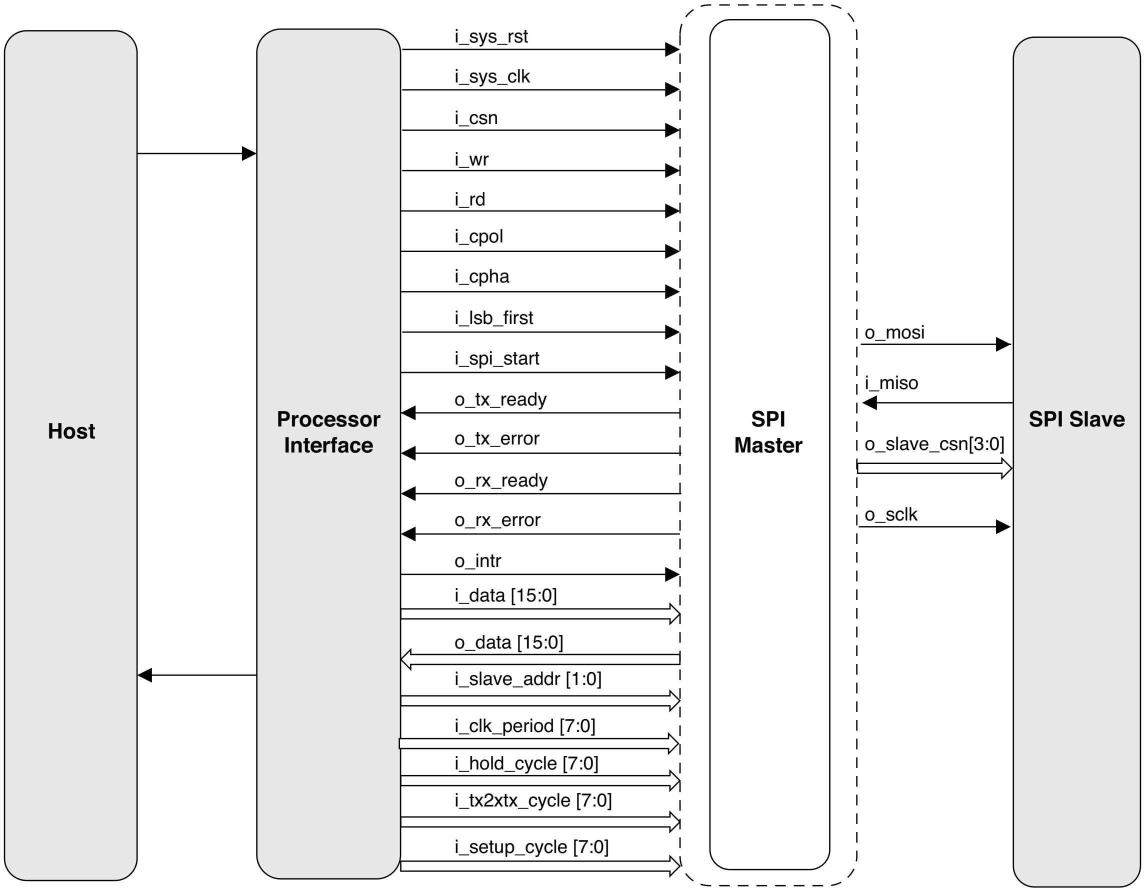
The Serial Peripheral Interface (SPI) bus provides an industry standard interface between processors and other devices. This reference design documents a SPI Master Controller designed to provide an interface between a generic processor with parallel bus interface and external SPI devices. The SPI Master Controller can communicate with multiple off-chip SPI ports.

The data size of the SPI bus can be configured to either 16 or 32 bits. The design can also to be configured to use an internal FIFO or not. The SPI Master Controller design supports all modes of CPOL and CPHA (00, 01, 10 and 11).

This design uses three pins (clock, data in and data out) plus one select for each slave device. A SPI is a good choice for communicating with low-speed devices that are accessed intermittently and transfer data streams rather than reading and writing to specific addresses. A SPI is an especially good choice if we can take advantage of its full-duplex capability for sending and receiving data at the same time.

This reference design is implemented in VHDL. The Lattice iCECube2™ Place and Route tool integrated with the Synplify Pro synthesis tool is used for the implementation of the design. The design uses an iCE40™ ultra low density FPGA and can be targeted to other iCE40 family members.

Figure 1. Block Diagram

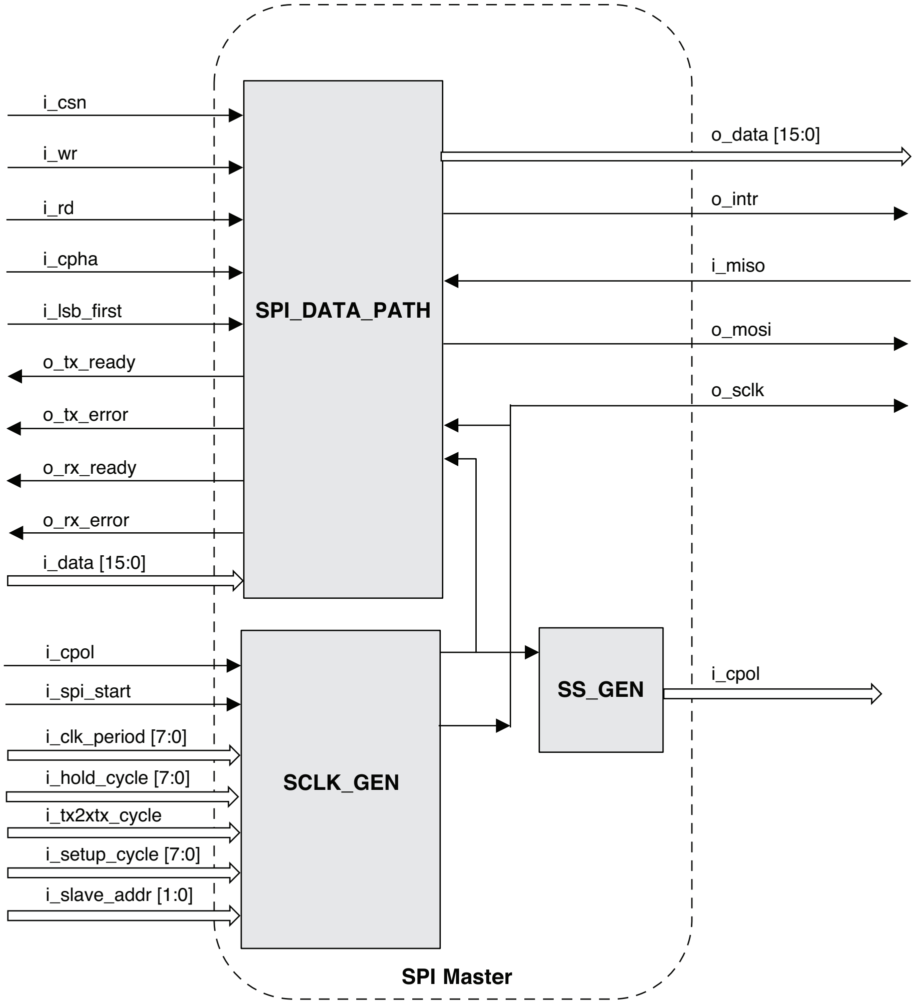


Features

- Four SPI slave select lines based on address
- Provision for easy integration of any processor interface
- Compile time configurable features
  - CPOL and CPHA modes – 00, 01, 10, 11
  - Configurable SCLK period
  - Configurable setup, hold and time interval between two SPI transactions
- Parameterized data width
- User-configurable read and write data FIFOs
- P-XACT version 1.2 compliant

Functional Description

Figure 2. Functional Block Diagram



The SPI Master Controller provides an interface between the generic parallel processor with a generic processor interface bus and a SPI device.

On the internal side, the SPI has a parallel bus that connects the SPI master with a processor and other on-chip components. There are input pins for write enable and read enable, as well as pins for selection of CPOL and CPHA, least-significant bit or most-significant bit first, transmitter and receiver ready signals, data and address bus, bus-to-input setup and hold cycles.

On the external side, the SPI master has a standard SPI bus interface:

- SLCK (Serial Clock) – Generated by the master to synchronize the data transfers
- MISO (Master In, Slave Out) – Transfers data going to the master from the slave
- MOSI (Master Out, Slave in) – Transfers data going from the master to the slave
- SS_N (Slave Select) – Asserted by the master to begin data transfer. There are four slave select outputs for selecting four different SPI slaves.

SPI master data samples from the MISO line depend upon the current bit count and CPOL/CPHA mode. A shift register on the receive data path converts serial-to-parallel conversion. Similar parallel-to-serial conversion takes place in the transmit data path.

Signal Descriptions

Table 1. Signal Descriptions

Signal	Width	Type	Description
i_sys_rst	1	Input	Asynchronous active low reset
i_sys_clk	1	Input	System clock
i_csn	1	Input	Active low chip select
i_data	16	Input	Input data from the processor interface
i_wr	1	Input	Active low write enable
i_rd	1	Input	Active high read enable
o_data	16	Output	Output data to the processor interface
o_tx_ready	1	Output	Transmitter ready – Indicates additional data can be written
o_rx_ready	1	Output	Receiver ready – Indicates additional data can be read
o_tx_error	1	Output	Indicates error in transmission of data
o_rx_error	1	Output	Indicates error in received data
i_cpol	1	Input	Polarity of the clock
i_cpha	1	Input	Phase of the clock
i_lsb_first	1	Input	LSB sent first when ‘1’. MSB goes first when ‘0’.
i_slave_addr	2	Input	Slave address to select a slave device
i_spi_start	1	Input	Start SPI master transactions
i_setup_cycles	1	Input	SPIM setup time in terms of i_sys_clk
i_hold_cycles	1	Input	SPIM hold time in terms of i_sys_clk
i_tx2tx_cycles	1	Input	SPIM interval between data transactions in terms of i_sys_clk
i_clk_period	1	Input	SCLK clock period in terms of i_sys_clk
o_mosi	1	Output	Serial data output from master
i_miso	1	Input	Serial data input to the master
o_slave_csn	4	Output	Slave select from master
o_sclk	1	Output	Serial clock is generated by the master
o_intr	1	Output	Interrupt output from SPI master

Design Module Description

To begin communication, the master first configures the clock using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are commonly in the range of 1-70 MHz.

The master then pulls the slave select of the desired slave low based on the slave address and starts issuing the clock pulses.

SCLK Generation Module

The SCLK generation module generates the SPI Master clock, SCLK, based on input control signals (clock period, setup, hold time and SPI transaction to transaction interval) all expressed in terms of system clock cycle counts. This module is comprised of a simple FSM which enables clock generation, holding SCLK at low or high based on CPOL = 0 or CPOL = 1. It is also responsible for generating the chip select for the SPI slave, adhering to setup and hold times as well as the wait time between two transactions.

SPI Data Path

SPI data path samples the MISO line and drives the MOSI line based on the CPOL/CPHA modes as follows:

- At CPOL=0, the base value of the clock is zero
  - For CPHA=0, data is read on the clock's rising edge and the data is changed on the falling edge
  - For CPHA=1, data is read on the clock's falling edge and the data is changed on the rising edge
- At CPOL=1, the base value of the clock is one (inversion of CPOL=0)
  - For CPHA=0, data is read on the clock's falling edge and the data is changed on the rising edge
  - For CPHA=1, data is read on the clock's rising edge and the data is changed on the falling edge

The SPI Master module supports four modes: {CPOL, CPHA} = {00, 01, 10, 11}. This module is responsible for sampling the MISO line and shifting the data based on the current bit count of the SPI transaction as well as the CPOL and CPHA modes. This module is also responsible for pushing bits of data on the MOSI line from a shift register based on bit counts and CPOL/CPHA modes.

During each SPI clock cycle, a full duplex data transmission occurs:

- The master sends a bit on the MOSI line; the slave reads it from that same line
- The slave sends a bit on the MISO line; the master reads it from that same line

While all four of these operations happen each cycle, they may not be used or required.

Write FIFO

This 16x16 FIFO is compile time configurable and stores data written by the processor. Before the SPI transactions take place, the data to be written to the slave device is written to the FIFO by the processor. The SPI Master Controller then reads data from this FIFO and sends it to the slave.

Read FIFO

This 16x16 FIFO is compile time configurable and stores data written by the slave device. The processor can read data from this FIFO after a SPI transaction.

Initialization Conditions

No user-specific initialization conditions, except that the i_sys_rst must be held low initially to bring up the design in the correct operating state.

Configurable Parameters

- FIFO_REQ – This Boolean parameter configures the FIFO usage.
  - If TRUE, two 16x16 FIFOs are inferred
  - If FALSE, no FIFOs are inferred

- DATA_SIZE – This parameter configures the data width of the SPI transaction. It can take two values:
  - 16 (default value)
  - 8 (in this case the higher eight bits of the data in the FIFO are filled with zeros)

Operation Sequence

If the parameter FIFO_REQ is set to ‘0’ then:

- A total of 16 transactions take place
- Four sets of data are used in the four transactions; each checks boundary conditions
- Four transactions each with CPOL/CPHA modes: 00, 01, 10 and 11
- Varying clock period, setup, hold delay and transaction intervals

Test procedure:

1. Write data to the SPI Master and set up CPOL and CPHA.
2. Set up the following parameters: clk_period, setup, hold, tx2tx cycles, LSB/MSB first modes and slave_addr.
3. Initiate SPI transactions by writing ‘1’ to SPI start of the SPI Master.
4. Wait for the transaction to complete and then read the data.
5. Repeat steps 2, 3, and 4 for four different clk_period, setup, hold, tx2tx cycles and slave_addr configurations.
6. Repeat steps 1-5 for four different sets of CPOL, CPHA and LSB/MSB first modes as well as the input data pattern to the SPI Master.
7. After these sequences, the simulation ends.

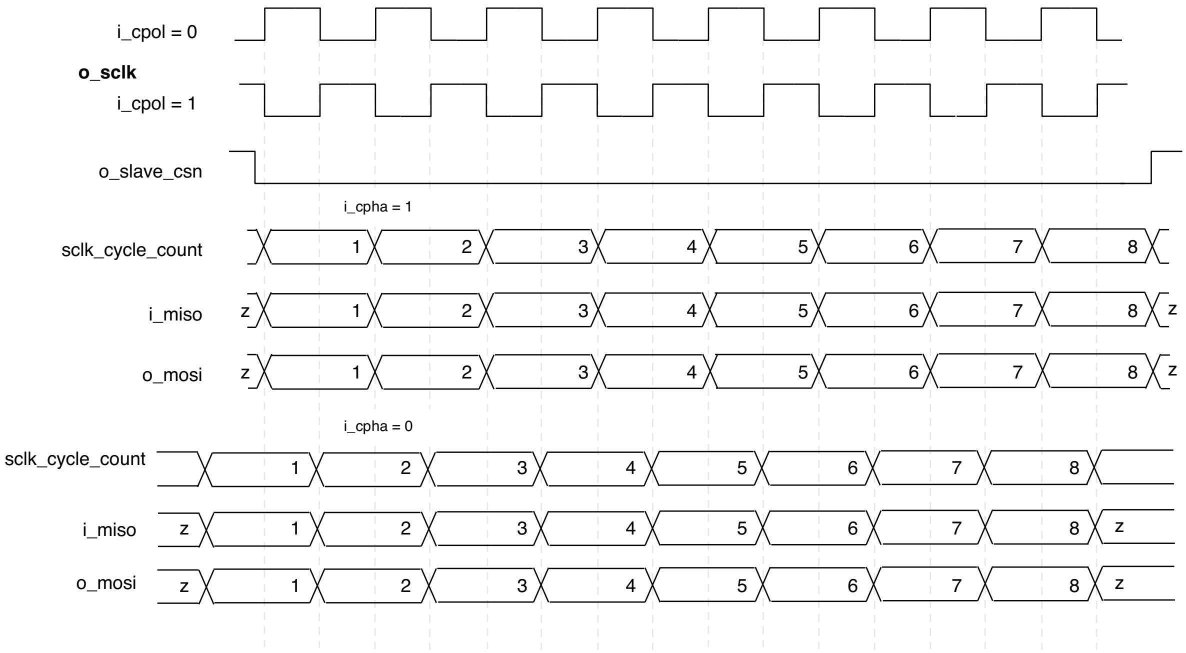
If the parameter FIFO_REQ is set to ‘1’ then 16 words are written to the FIFO before transactions are started.

Test procedure:

1. Set up CPOL and CPHA.
2. Set up the following parameters: clk_period, setup, hold, tx2tx cycles, LSB/MSB first modes and slave_addr.
3. Initiate SPI transactions by writing a ‘1’ to the SPI start of SPI Master.
4. Wait for the transaction to complete and then read the data.
5. Repeat steps 2, 3, and 4 for four different clk_period, setup, hold, tx2tx cycles and slave_addr configurations.
6. Repeat steps 1-5 for four different sets of CPOL, CPHA and LSB/MSB first modes as well as the input data pattern to the SPI Master
7. After these sequences, the simulation ends.

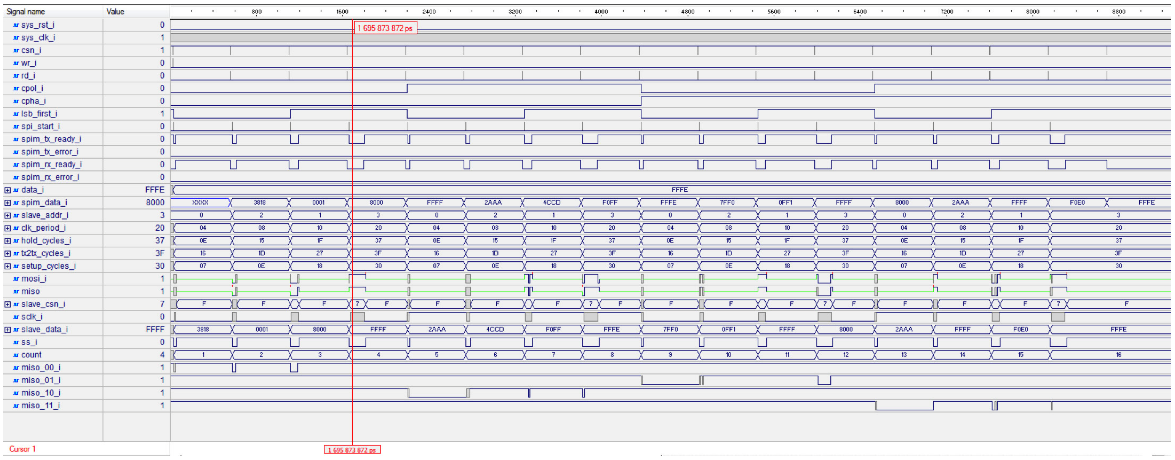
Timing Diagram

Figure 3. Timing Diagram



Simulation Waveforms

Figure 4. Simulation Waveforms



Signal name	Value
sys_rst_i	0
sys_clk_i	1
csn_i	1
wr_i	0
rd_i	0
cpol_i	0
cpha_i	0
lsb_first_i	1
spi_start_i	0
spim_tx_ready_i	0
spim_tx_error_i	0
spim_rx_ready_i	0
spim_rx_error_i	0
⊞ data_i	FFFE
⊞ spim_data_i	8000
⊞ slave_addr_i	3
⊞ clk_period_i	20
⊞ hold_cycles_i	37
⊞ tx2tx_cycles_i	3F
⊞ setup_cycles_i	30
mosi_i	1
miso	1
⊞ slave_csn_i	7
sclk_i	0
⊞ slave_data_i	FFFF
ss_i	0
count	4
miso_00_i	1
miso_01_i	1
miso_10_i	1
miso_11_i	1

Cursor 1

Implementation

This design is implemented in VHDL. When using this design in a different device, density, speed or grade, performance and utilization may vary.

Table 2. Performance and Resource Utilization

Family	Language	Utilization (LUTs)	f_{MAX} (MHz)	I/Os	Architectural Resources
iCE40[1]	VHDL	360	125	86	N/A

[1] Performance utilisation characteristics are generated using iCE-40LP1K-CM121 with iCEcube2 design software.

References

- iCE40 Family Handbook

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
October 2012	01.0	Initial release.