Introduction

The Serial Peripheral Interface (SPI) is used primarily for synchronous serial communication between a host processor and its peripherals. The SPI bus is often selected because of its low pin count and full-duplex mode that can achieve data throughput in the tens of Mbps range. The SPI bus uses a 4-wire interface with two unidirectional data lines to communicate between the master and the selected slave. It supports one master with multiple slaves on one bus and allows protocol flexibility for the bit transferred.

This reference design implements a SPI slave device interface that provides full-duplex, synchronous, serial communication with the SPI master. The data size of the SPI bus can be configured to either 16 or 8 bits. The SPI Slave Controller reference 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 design is implemented in VHDL. The Lattice iCECube2™ Place and Route tool integrated with 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

- Supports all four modes of CPOL and CPHA operation (00/01/10/11)
- Supports variable data widths (8 and 16 bits)
- Provision for easy integration of any processor interface
- IP-XACT version 1.2 compliant
Functional Description

**Figure 2. Functional Block Diagram**

**SPI Slave Receiver**
MOSI sampling – SPI slave receives the data on the MOSI line based on CPOL and CPHA modes as follows:

Sample at positive edge of SCLK for:
- i_cpol = ’0’ and i_cpha = ’0’
- i_cpol = ’1’ and i_cpha = ’1’

Sample at negative edge of SCLK for:
- i_cpol = ’1’ and i_cpha = ’0’
- i_cpol = ’0’ and i_cpha = ’1’

After the data is received, rx_ready goes high and valid received data is available on the bus. If new data has arrived and the last data received has not yet been read, then the rx_error signal goes high to indicate a receive error.

**SPI Slave Transmitter**
Sending data on the MISO line – The SPI slave transmits the data on the MISO line from a shift register based on CPOL and CPHA as follows:

- CPOL = 0 and CPHA = 0: Data is placed before the rising edge of sclk
- CPOL = 1 and CPHA = 0: Data is placed before the falling edge of sclk
- CPOL = 0 and CPHA = 1: Data is placed at the rising edge of sclk
- CPOL = 1 and CPHA = 1: Data is placed at the falling edge of sclk

At the end of data transmission, the tx_ready signal goes high. When the transmitter is busy transmitting a data stream, the tx_error signal goes high if the transmit buffer data is over-written by the processor interface.
Clock Requirements
For proper operation of the SPI slave, the system frequency should be twice that of the SCL frequency at a minimum.

Signal Descriptions

Table 1. Signal Descriptions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Width</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_csn</td>
<td>1</td>
<td>Input</td>
<td>Active low chip select</td>
</tr>
<tr>
<td>i_data</td>
<td>16</td>
<td>Input</td>
<td>Input data from the processor interface</td>
</tr>
<tr>
<td>i_wr</td>
<td>1</td>
<td>Input</td>
<td>Active low write enable</td>
</tr>
<tr>
<td>i_rd</td>
<td>1</td>
<td>Input</td>
<td>Active high read enable</td>
</tr>
<tr>
<td>o_data</td>
<td>16</td>
<td>Output</td>
<td>Output data to processor interface</td>
</tr>
<tr>
<td>o_tx_ready</td>
<td>1</td>
<td>Output</td>
<td>Transmitter ready status – High indicates the transmitter is ready to send additional data</td>
</tr>
<tr>
<td>o_rx_ready</td>
<td>1</td>
<td>Output</td>
<td>Receiver ready – High indicates the receiver is ready to receive additional data</td>
</tr>
<tr>
<td>o_tx_error</td>
<td>1</td>
<td>Output</td>
<td>Indicates error in transmission of data</td>
</tr>
<tr>
<td>o_rx_error</td>
<td>1</td>
<td>Output</td>
<td>Indicates error in reception of data</td>
</tr>
<tr>
<td>i_cpol</td>
<td>1</td>
<td>Input</td>
<td>Polarity of the clock</td>
</tr>
<tr>
<td>i_cpha</td>
<td>1</td>
<td>Input</td>
<td>Phase of the clock</td>
</tr>
<tr>
<td>i_lsb_first</td>
<td>1</td>
<td>Input</td>
<td>LSB sent first when ‘1’. MSB goes first when ‘0’.</td>
</tr>
<tr>
<td>o_miso</td>
<td>1</td>
<td>Output</td>
<td>Slave output to master</td>
</tr>
<tr>
<td>i_mosi</td>
<td>1</td>
<td>Input</td>
<td>Slave input from master</td>
</tr>
<tr>
<td>i_ssn</td>
<td>1</td>
<td>Input</td>
<td>Slave select from master</td>
</tr>
<tr>
<td>i_sclk</td>
<td>1</td>
<td>Input</td>
<td>Serial clock from master</td>
</tr>
<tr>
<td>i_sys_rst</td>
<td>1</td>
<td>Input</td>
<td>Asynchronous active low reset</td>
</tr>
<tr>
<td>i_sys_clk</td>
<td>1</td>
<td>Input</td>
<td>System clock</td>
</tr>
<tr>
<td>o_tx_ack</td>
<td>1</td>
<td>Output</td>
<td>Transmission acknowledged from slave</td>
</tr>
<tr>
<td>o_tx_no_ack</td>
<td>1</td>
<td>Output</td>
<td>Transmission not acknowledged from slave</td>
</tr>
</tbody>
</table>

Initialization Conditions
An asynchronous active low reset signal assertion is necessary to initialize the SPI slave to the proper operating state. Receive and transmit buffers are initialized to zero during the reset condition.

Configurable Parameters
DATA_SIZE – This parameter configures the data width of the SPI transaction. It can take either of two values: 16 (the default value) or 8.

Operation Sequence
1. Write a data slave, set up CPOL, CPHA and LSB/MSB first mode.
2. Enable the SPI slave by making i_ssn = 0.
3. Generate the SCL depending upon CPOL and CPHA.
4. Receive the data from the i_data line to the MOSI line depending upon CPOL, CPHA and LSB/MSB.
5. Similarly, drive the MISO line depending upon CPOL, CPHA and LSB/MSB.
6. Repeat steps 1, 2, 3, 4 and 5 for four different sets of CPOL, CPHA and LSB/MSB first modes and input the data pattern to the slave.
**Timing Diagram**

Signal definitions:
- $i_{sclk}$ – Serial clock line (Generated by master)
- $i_{cpol}$ – Clock polarity
- $i_{cpha}$ – Clock phase
- $i_{ssn}$ – Slave select
- $o_{miso}$ – Output from the SPI slave
- $i_{mosi}$ – Input to the SPI slave

*Figure 3. Timing Diagram*

![Timing Diagram](image1)

**Simulation Waveforms**

*Figure 4. Simulation Waveforms*
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

<table>
<thead>
<tr>
<th>Family</th>
<th>Language</th>
<th>Utilization (LUTs)</th>
<th>f_MAX (MHz)</th>
<th>I/Os</th>
<th>Architecture Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>iCE40</td>
<td>VHDL</td>
<td>217</td>
<td>254</td>
<td>48</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Performance and utilization 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

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2012</td>
<td>01.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>