NI 5644R LabVIEW FPGA Serial Peripheral Interface Example

Theory of Operation

November 20, 2012

IP Features

Serial Peripheral Interface (SPI) busses are commonly used to communicate between a controller (master) device and a target (slave) device. In general, SPI busses require four lines for communication: chip select/clock enable, serial clock, master serial data out (MOSI), and master serial data in (MISO). In some cases only a subset of these lines are used; some devices multiplex both MOSI and MISO onto a single bidirectional data line. The LabVIEW FPGA code included in the example project contains examples for both a SPI master and a SPI slave. These examples support the following SPI features without further modification:

Master:

- 4-Wire interface consisting of Chip Select, Clock, MOSI, and MISO
- 3-Wire interface consisting of Chip Select, Clock, and Bidirectional Data
- SPI modes 0-3
- Programmable word lengths up to 64-bits
- Programmable serial clock frequency
- Configurable MISO sampling timing (may increase interface clock frequency in some cases)
- Extendable data bus width with code modification

Slave:

- 4-Wire interface consisting of Chip Select, Clock, MOSI, and MISO
- SPI modes 0-3
- Programmable word lengths up to 64-bits
- Includes example register file interface

System simulation file included within project to verify expected operation

SPI protocol details and associated timing requirements are assumed to be known by the reader and are not further explained in this document beyond those that directly affect understanding of the example code.

Hardware Referenced in this Example

Description	National Instruments Part Number
NI PXIe-5644R	782376-01
CB-2162 Single-Ended Digital I/O Accessory	778592-01
Header Jumper Kit, Set of 10 Twisted Pairs, 3.5 Inches	199101-01
SHC68-C68-D4 Shielded Single-Ended Cable, 1 Meter	196275-01

Software Requirements

The NI 5644R LabVIEW FPGA Serial Peripheral Interface Example was designed and tested with the following software packages.

LabVIEW 2012 version 12.0.0
LabVIEW FPGA version 12.0.0
NI LabVIEW 2012 Support for the NI PXIe-5644R (free download from ni.com)

System Architecture

The Digital Input/Output (DIO) capability of the NI 5644R enables many applications such as data streaming, DUT control, and bus interfacing. Refer to the *NI PXIe-5644R Device Specifications* for details regarding the DIO pin-out, voltage levels, speed, etc. The NI 5644R provides 24 DIO (DIO 0 to 23) that are banked in groups of four. Each DIO bank shares a single direction control which requires that each element in a bank have the same direction (either input or output) at the same time. Therefore, care must be taken to ensure that the DIO are chosen and combined appropriately for the application. A single DIO bank of four is shown in the diagram below. The DIO is buffered between the FPGA and the VHDCI connector. In order to prevent damage to the FPGA when changing the direction of the DIO, a delay is inserted between updating the direction control of the external buffer and the direction control of the FPGA IO. The LabVIEW FPGA diagram does not have direct access to the direction control lines. All DIO lines also carry weak pull-down resistors to ground.

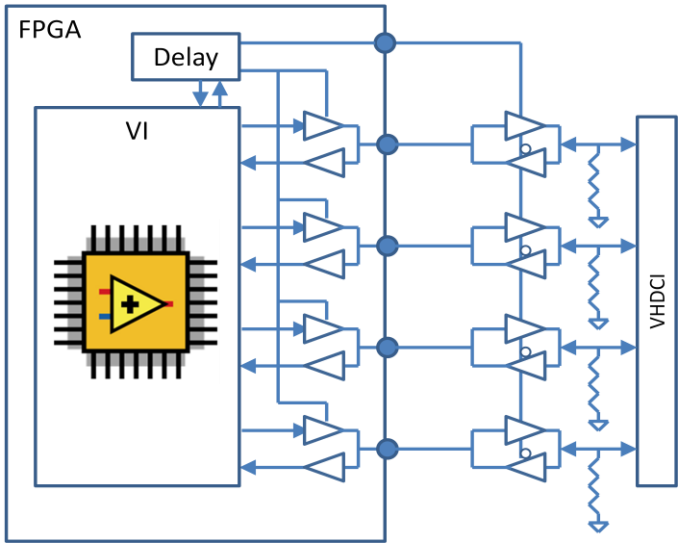

Figure 1: Architecture of Single DIO Bank

PFI lines one and two (PFI1 and PFI2) are not banked and their direction can be independently controlled. Both PFI lines are buffered similarly to the 24 DIO lines and require the same direction switching delay. Both PFI lines carry weak pull-up resistors as shown in the diagram below.

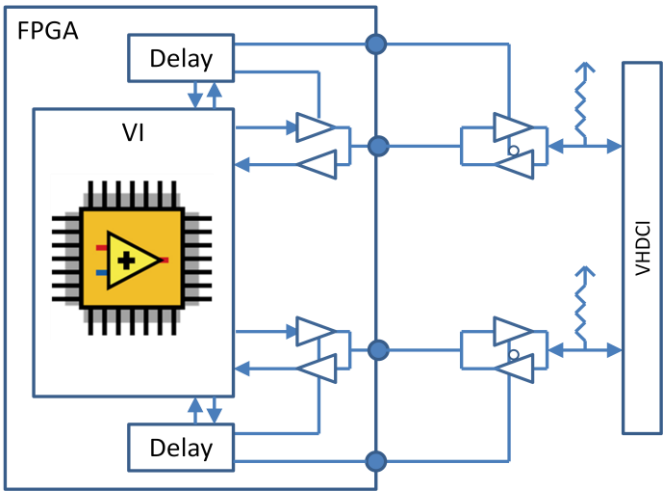

Figure 2: PFI 1 and 2 Architecture

The delay between changing the value of a DIO CLIP bank output enable and the hardware physically attaining the desired state is pictured below. At time event one, the output enable value provided to the DIO CLIP is changed from output (output enable equal to true) to input. The output enable done feedback to the diagram is changed to false on the following clock edge (time event two). The number of clock cycles between time event two and three will vary depending on the single-cycle timed loop frequency used in the diagram. At time event three, the output enable circuitry is ready to be updated again by the user's diagram; however, the physical output of the module may still require up to the maximum required direction change time to settle on the selected direction (time event four). The maximum required direction change time is specified in the *NI PXIe-5644R Device Specifications*.

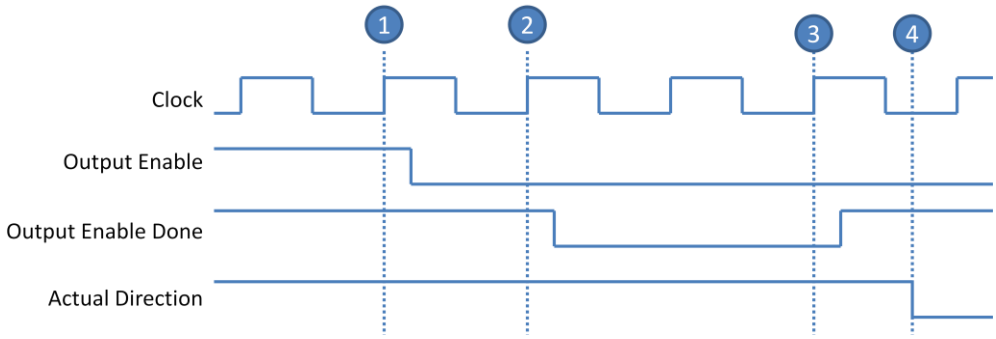

Figure 3: DIO Direction Control Timing

IP IO Selection

The base SPI Master VI utilizes four IO: Clock, Chip Select, MISO data, and MOSI data. These IO can easily be expanded to create wider interfaces or to support additional busses. Additionally, 3-wire SPI interfaces are supported by enabling the MOSI output channel to be tristated. In both cases the Clock and Chip Select lines are always driven by the master and should never be tristated; therefore, these lines should be placed in a single bank of DIO that is always used for output. If the MOSI data line is always an output of the SPI Master VI, then it can also be placed in the same bank as Clock and Chip Select. However, if it is tristatable then it should be placed in a separate bank.

As shown below, if all outputs of the SPI Master VI are unidirectional and there is no requirement to tristate the MOSI output then all outputs can be placed within a single bank of DIO and the output enable control for that bank can be tied to true. MISO data is always an input to the SPI Master VI and should never be driven. In the below example, DIO 0, 1, 2, and 3 are members of DIO Bank 3..0 which share a common direction. Since all outputs are always output, the output enable can be wired to true. DIO 4 is in DIO Bank 7..4 which is independent from the other banks and can therefore be used as input.

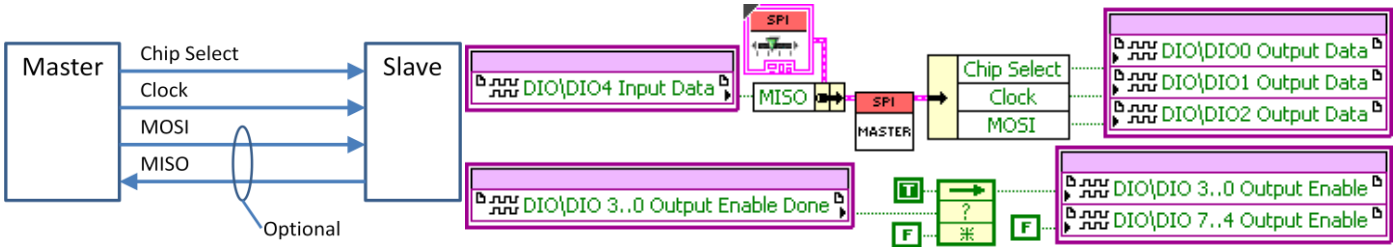

Figure 4: Unidirectional IO Planning

For bidirectional data or in cases where the master VI's output data should be tristated, the MOSI data line must be moved to an independent bank from the banks that are always input or output (note that the PFI 1 and 2 lines offer a single bit with independent direction control which could be used to save a DIO bank resource). In the below figure, a

single line of DIO (DIO 4) is used for both MOSI and MISO data. DIO 4 is in a separate bank from the Clock and Chip Select lines due to the requirement that the direction be updated during operation.

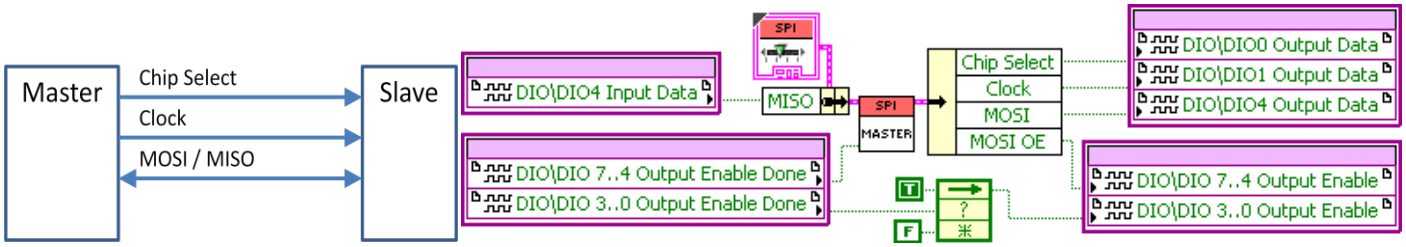

Figure 5: Bidirectional MOSI IO Planning Using Single Data Line

Optionally, two DIO lines can be used to read and write data from a slave device's bidirectional IO as shown below. This example is identical to the above example with the exception that data is read on a dedicated input (DIO 8 in this example). This approach has the advantage that the same master VI can be used for both 3-wire and 4-wire slave devices. Note that the MOSI data driven on DIO 4 below can be directly read using DIO\DIO4 Input Data; however, this samples the data at the FPGA pin rather than at the VHDCI connector.

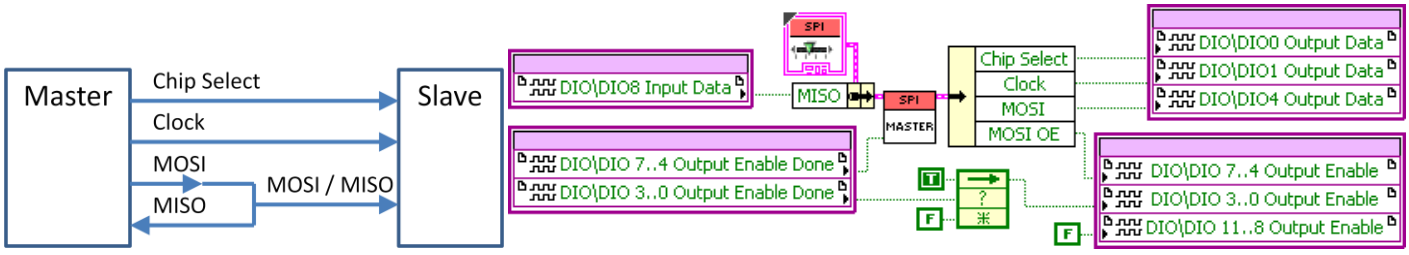

Figure 6: Bidirectional IO Planning Using Multiple Data Lines

Using the IP

Two FPGA examples are included inside the project. The "SPI Simple Example" uses controls and indicators to setup, start, and return results from the SPI Master VI. The "SPI DMA Example" uses two DMA FIFOs (one from the host to the FPGA and the other from the FPGA to the host) to control the master VI. Each of these examples includes both a master and a slave interface as described below.

SPI Master

The SPI Master VI contains all the logic necessary (finite state machines, shift registers, etc) to control a SPI bus with a Chip Select, Clock, Serial Data Out (MOSI), and Serial Data In (MISO). These signals should be directly wired to the DIO CLIP without any additional feedback nodes placed between the master VI and the CLIP as pictured below.

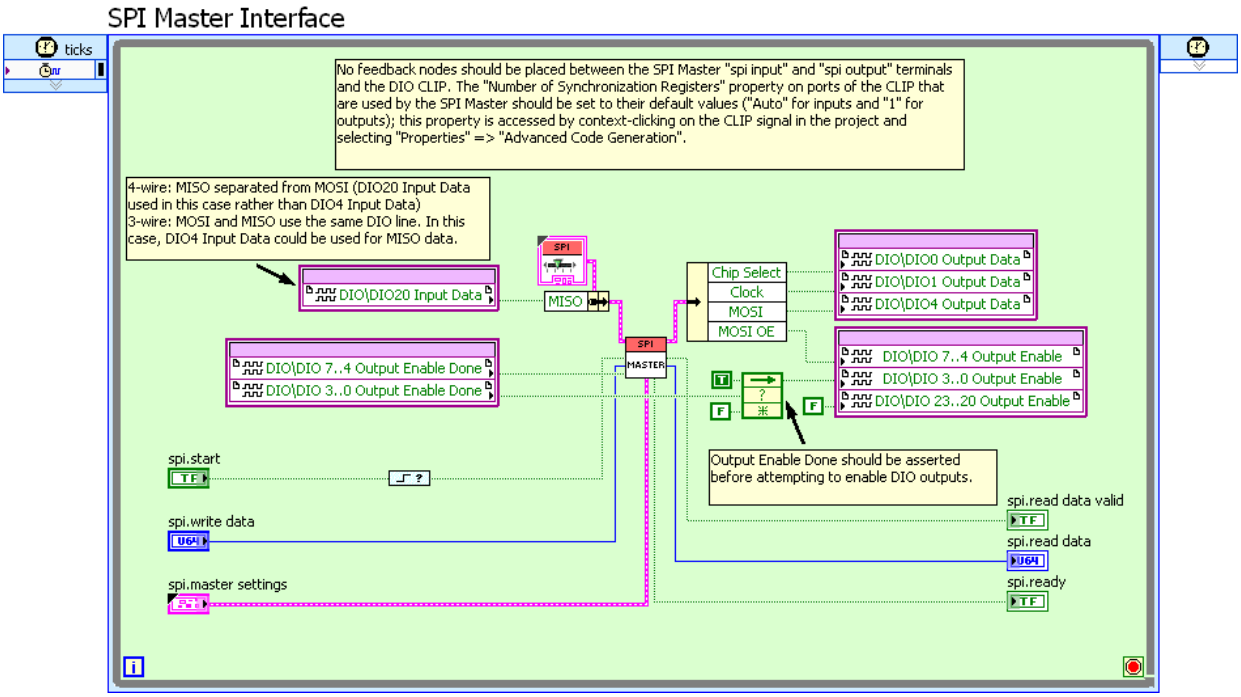

Figure 7: Example Wiring of the SPI Master VI to the DIO

SPI Master.vi Terminals

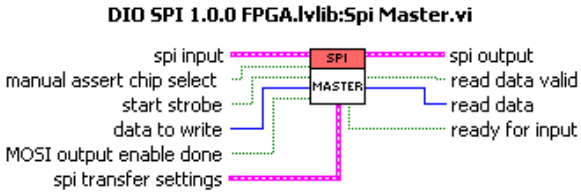

Figure 8: SPI Master VI Terminals

"Spi Input"

Serial data input to the master VI from the slave device (MISO).

"Manual Assert Chip Select"

Allows the master VI's Chip Select output to be asserted and unasserted independent from the logic inside the master. This is useful for transfers that are larger than the maximum supported transfer size of the master VI. The master VI supports up to 64-bit transfers. If more than 64-bits must be transferred, then the chip select can be asserted using this control and then multiple transactions can be made before manually deasserting the chip select to complete the transaction.

"Start Strobe"

Starts a transfer when set to true. The transfer will not complete (the ready for input returning to true) until this input is false. This input should only be asserted when ready for input is true.

"Data to Write"

This input is an unsigned 64-bit word that will be serialized out to the slave device. Data is sent most-significant bit first. The data should be aligned with the least-significant bit (last bit) of the transfer placed in the least-significant bit of this input. For example, in a 10-bit transfer to send the word 0x234 set the "Data to Write" input to 0x234.

"MOSI Output Enable Done"

The MOSI output DIO channel also makes use of an Output Enable line which has a corresponding Output Enable Done status indicator. For example, DIO 12 is a member of bank 15..12 which uses DIO\DIO 15..12 Output Enable to control the direction of the bank. The status of the bank is returned using the DIO\DIO 15..12 Output Enable Done CLIP signal. The SPI Master VI must know the status of the bank direction for proper operation. Wire the Output Enable Done signal of the bank in which the MOSI signal is a member into this input of the SPI Master VI.

"SPI Transfer Settings"

These settings control the type of SPI transfer that is made by the master VI. All settings are detailed in the IP Configuration section of this document.

"SPI Output"

This cluster output contains all SPI bus signals that the master VI drives including Clock, Chip Select, MOSI data, and the output enable for the MOSI line (MOSI OE). These signals should all be driven directly into the DIO CLIP Output Data signals corresponding to the DIO channels used to create the SPI bus.

"Read Data" and "Read Data Valid"

The Read Data output returns the data serialized into the master from the MISO line of the SPI bus. Read Data remains valid until the next transaction is started. The Read Data Valid output pulses for a single clock cycle when the SPI transaction has completed and the Read Data output is valid. The Read Data Valid signal can be used as an enable to downstream feedback nodes or can trigger a write to a memory or FIFO.

"Ready for Input"

The Ready for Input signal output from the master VI should be used to determine when the start strobe can be asserted and when the SPI transaction has started or completed. When a transaction is started, this output will deassert to false and will remain deasserted until the transaction has completed. Ready for Input will remain asserted until the next Start Strobe assertion.

SPI Slave

The SPI Slave VI contains all the logic necessary (finite state machines, shift registers, etc) to create a slave device on a SPI bus and to return data to a SPI master device. The SPI Slave VI can only be used in 4-wire mode (unidirectional MOSI and MISO data busses). The SPI Slave and User Slave Logic VIs include an example register bus interface. Other types of slave interfaces can be implemented by modifying the example code; however, only a single read/write bus is provided in the example. The SPI Slave VI must operate within a Single-Cycle Timed Loop that is running much faster (at least 2x) than the serial clock frequency. This is required because the SPI bus signals are sampled with the Single-Cycle Timed Loop clock rather than with the SPI Clock itself. The SPI bus signals should in general be wired directly into the slave VI. However, additional pipelining or logic could be inserted on these lines if needed, though this could affect the interface clock frequency requirements.

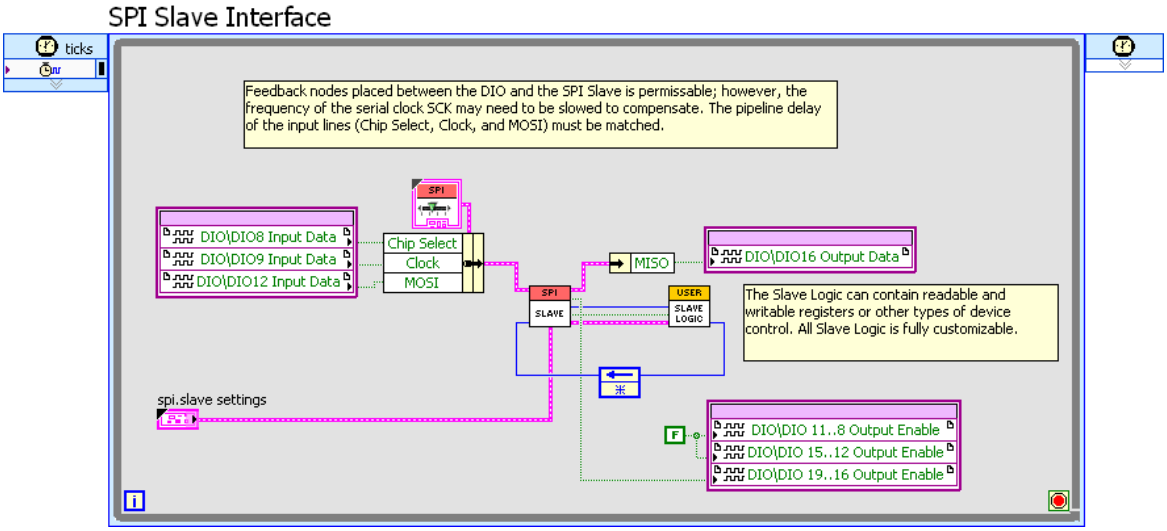

Figure 9: Example Wiring of the SPI Slave VI to the DIO

SPI Slave.vi Terminals

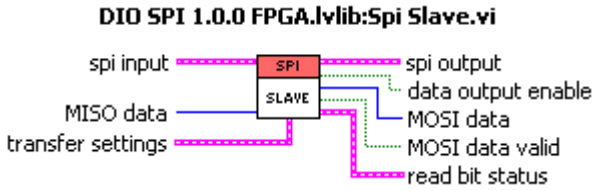

Figure 10: SPI Slave Terminals

"SPI Input"

Cluster input contains the three standard SPI control lines: Clock, Chip Select, and MOSI data.

"MISO Data"

This input is an unsigned 64-bit word that will be serialized out of the slave VI. Data is sent most-significant bit first. The data should be aligned with the least-significant bit (last bit) of the transfer placed in the least-significant bit of this input. For example, in a 10-bit transfer to send the word 0x234 set the "MISO Data" input to 0x234. Data is always serialized out of the slave starting with the assertion of chip select or the first edge of the clock depending on the setting of the clock phase (CPHA). Therefore, to implement a typical SPI interface where address and command comprise the first few bits of the transfer, the MISO data word may need to be padded with zeros.

"Transfer Settings"

These settings control the type of SPI transfer that is supported by the slave VI. All settings are detailed in the IP Configuration section of this document.

"SPI Output"

Serial data output from the slave VI to the master. This output should be wired into one of the DIO channels.

"Data Output Enable"

Controls the direction of the DIO bank in which the SPI Output is placed. The SPI Output (MISO) is tristated whenever the Chip Select is deasserted as is typical in most SPI interfaces. The status of the output enable (output enable done) is not necessary to the slave VI since the timing of the bus is governed by a SPI master device.

"MOSI Data" and "MOSI Data Valid"

MOSI data is the data serialized into the slave VI from the master device. When the SPI transaction completes (Chip Select deasserts) the MOSI Data Valid output will assert to true for a single clock cycle.

"Read Bit Status"

This output is only provided as an interface to the example User Slave Logic to detect command and address bits transferred during a SPI transaction. This allows a register file interface to be accessed for reading from a single SPI transaction. See the code for further details.

IP Configuration

The SPI Master VI is configured using a single predefined type which can either be wired to a constant or a control that can be programmable from the host. The configuration cluster is shown below.

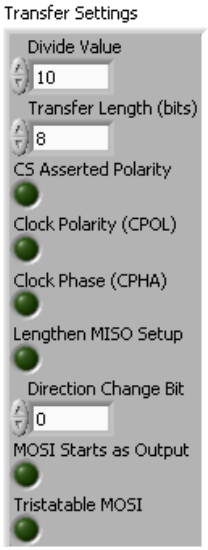

Figure 11: SPI Master VI Configuration Control

Configuration settings definitions

"Divide Value"

The SPI Master VI must be placed within a single-cycle timed loop (SCTL) in the LabVIEW FPGA diagram. The SCTL runs off of a user defined clock. The SPI interface driven to a slave from the master VI operates at a divided down clock rate from the SCTL clock frequency. The SPI interface Clock signal frequency will be equal to the frequency of the SCTL clock divided by ("Divide Value" + 1) x 2:

$$SPI\ Clock\ Frequency = \frac{\text{SCTL Clock Frequency}}{(\text{Divide Value} + 1) * 2}$$

For example, if the SPI Master VI is clocked at 100 MHz and "Divide Value" is set to 9, then the output SPI Clock will be:

$$SPI\ Clock\ Frequency = \frac{100\ \text{MHz}}{(9 + 1) * 2} = 5\ \text{MHz}$$

This also implies that the maximum SPI Clock frequency is equal to half the rate of the SCTL clock frequency. "Divide Value" is defined as an unsigned 8-bit value; therefore, values in the range of 0 to 255 are supported.

"Transfer Length (bits)"

Defines the total number of SPI Clock cycles in a transaction where a transaction is defined as beginning when the Start control of the SPI Master VI asserts to true until the SPI Master VI Ready output signal becomes true again (the ready signal becomes false when Start is set to true and remains false until all bits have been sent to the slave device). The SPI data controls are defined as unsigned 64-bit values; however, the "Transfer Length (bits)" control is defined as an unsigned 8-bit value which means that the full range of U8 values should not be used. The "Transfer Length (bits)" settings must be within the range from 1 to 64 for proper operation of the interface. A value of zero is illegal and should not be used. Values in the range from 65 to 255 can be used; however, only the lower 64-bits (corresponding to the U64 data) will be written and read – the SPI MOSI and MISO data will be driven to zero for the rest of the transaction.

"CS Asserted Polarity"

This control defines the value of the Chip Select signifying an active transaction on the SPI bus. For example, most slave devices support an active low ('0') value for chip select for the device to perceive/respond to the transaction. In this case the "CS Asserted Polarity" setting should be set to false. If the slave device requires an active high chip select then the control should be set to true. Note that not all SPI compatible devices require a chip select or some devices may use different terminology to define how a transaction is framed. The Chip Select line can generally be used for such devices.

"Clock Polarity (CPOL)" and "Clock Phase (CPHA)"

When used together, these two settings define which of the four SPI modes (0-3) are used on the bus. The modes and settings map as follows:

Mode	Clock Polarity (CPOL)	Clock Phase (CPHA)
0	False (0)	False (0)
1	False (0)	True (1)
2	True (1)	False (0)
3	True (1)	True (1)

A value of true (1) for "Clock Polarity (CPOL)" will invert the SPI Clock which means that the clock will start and end high ('1'). If "Clock Phase (CPHA)" is set to false, then both MISO and MOSI data should be driven on the bus as soon as the Chip Select asserts, the data will be sampled on the first clock edge, and then the data will be updated on the following clock edge. If "Clock Phase (CPHA)" is set to true, then MISO and MOSI data will update only after the first clock edge, will be sampled by the second clock edge, and will change on the following clock edge, etc. These combinations are pictured below:

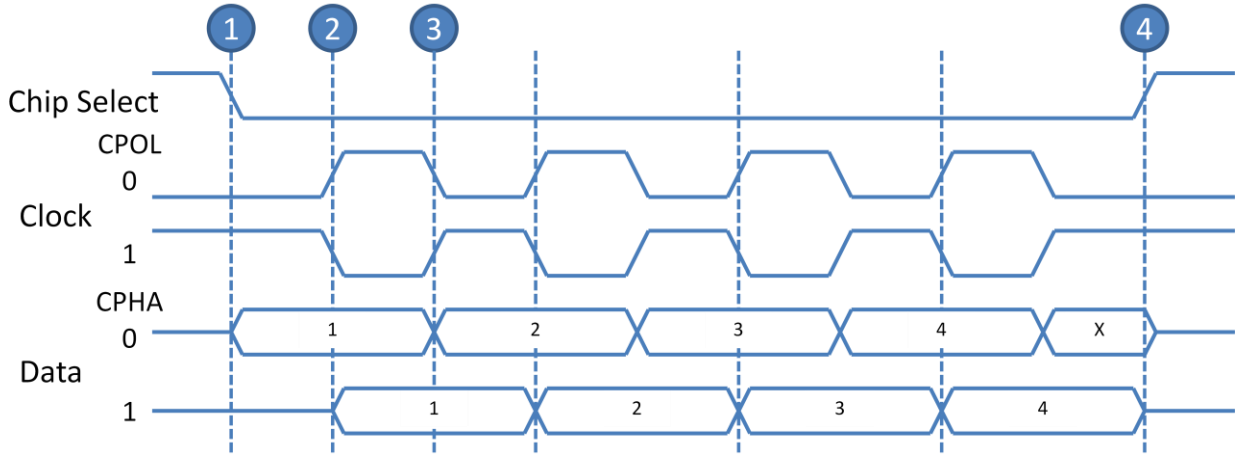

Figure 12: SPI Modes

At time event one, the chip select is asserted and data is either driven on the bus if CPHA = 0 or the bus is left in its previous state if CPHA = 1. At time event two, on the first edge of the clock (regardless of CPOL value), data is either sampled if CPHA = 0 or updated and driven if CPHA = 1. At time event three, data is sampled if CPHA = 1 or updated if CPHA = 0 – this case is the exact opposite of the time event two case. Finally when the transaction is completing at time event four by releasing the Chip Select, the data lines will be tristated. It should be noted that if CPHA = 0 then the last bit on the bus may be a do-not-care value as denoted by the "X".

"Lengthen MISO Setup"

MISO data must be sampled by the master when it is guaranteed to be stable and at the correct value. In general, SPI interfaces change the data bus on one edge of the clock and sample the data on the opposite edge of the clock based on the settings of the Clock Polarity and Clock Phase as mentioned above. Therefore, the period of the SPI Clock must be adjusted to guarantee that the setup and hold times of the MISO data are met at the master (see the System Timing portion of this document for detailed analysis of this timing). The master VI as provided will sample the MISO data at the same time that a clock edge is sent. Therefore, in most cases, due to clock-to-output time of the master, propagation delay to the VHDCI connector, and cabling delays, the hold time of the MISO data at the master will be met for both edges of the serial clock. By sampling data on the edge of the clock at which the slave device will update the data bus, the setup time margin is increased. Therefore, the SPI Clock frequency can be safely increased while keeping the setup slack positive (though reduces the timing margin). This sampling mode is enabled by setting "Lengthen MISO Setup" to true. If "Lengthen MISO Setup" is false, then MISO data is sampled normally. See image below for further explanation.

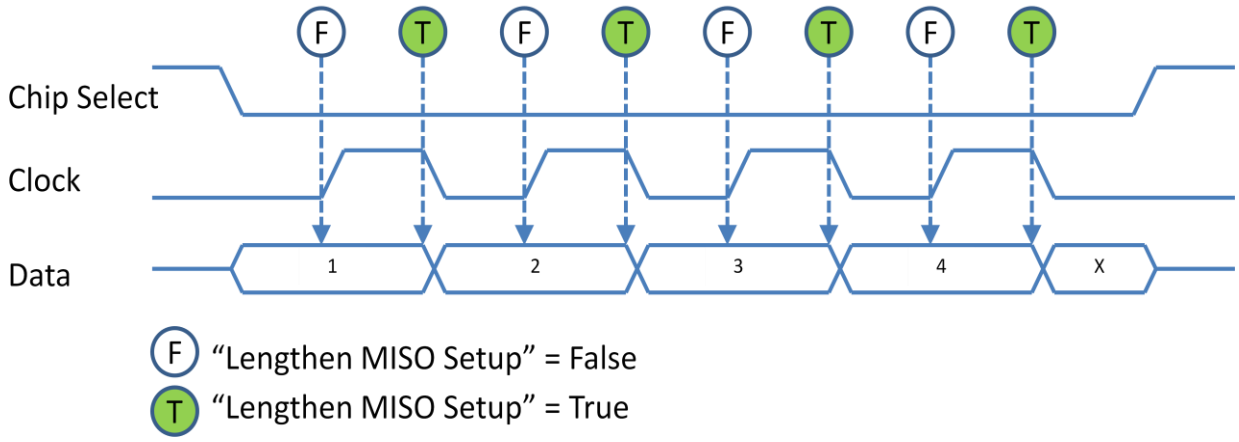

Figure 13: SPI Master VI MISO Sampling Points

Bidirectional Data Control

The final three elements of the SPI transfer settings detail when the MOSI output is enabled and disabled. See the Bidirectional Data Transfer section for additional timing information and examples. The bidirectional functionality should be used for transfers that start as input or output and then switch direction at some point during the transfer. This is common for 3-wire SPI devices.

"Tristatable MOSI"

This control enables the SPI Master VI to tristate the MOSI line when not in use and enables the "Direction Change Bit" and "MOSI Starts as Output" settings to take effect.

"Direction Change Bit"

Controls the serial clock cycle on which the MOSI data changes direction. A data word's bits are numbered from the most-significant bit down to bit zero. For example, a 20-bit number contains bits 19 down to zero. If a direction change is

required after sending 15 of the bits (direction changes when 16th bit would be sent), then the "Direction Change Bit" should be set to five which is equal to the length of the word (20) minus the length of the first half of the transfer (15). Timing diagrams of these transfers are shown in the Bidirectional Data Transfer section of this document.

"MOSI Starts as Output"

This setting controls whether the MOSI output of the SPI Master VI is driven actively for the first half of the transfer (for the bits sent prior to the "Direction Change Bit") or if it is tristated for the first half of the transfer. See the Bidirectional Data Transfer section of this document for further information.

Bidirectional Data Transfer and System Simulation

The SPI Master and Slave VIs can be simulated together directly in the LabVIEW project included with this example. Use System Testbench.vi to determine the proper transfer settings to match the required application. The following plots are examples of different transfer settings for the SPI Master and Slave VIs. It is highly recommended to simulate the data transfer using the System Testbench VI to verify that the output of the master matches requirements of the slave.

The below example transaction contains eight bits of data and an active low Chip Select. The transfer settings Divide Value is set to nine which means that the clock will toggle every ten (Divide Value + 1) clock cycles. The upper plot shows data asserted onto the bus when chip select asserts (Clock Phase = 0) and the lower plot shows data asserted only after the first edge of the clock (Clock Phase = 1). The master data output (MOSI) word is equal to 0x85. The master data input (MISO) word is equal to 0x81.

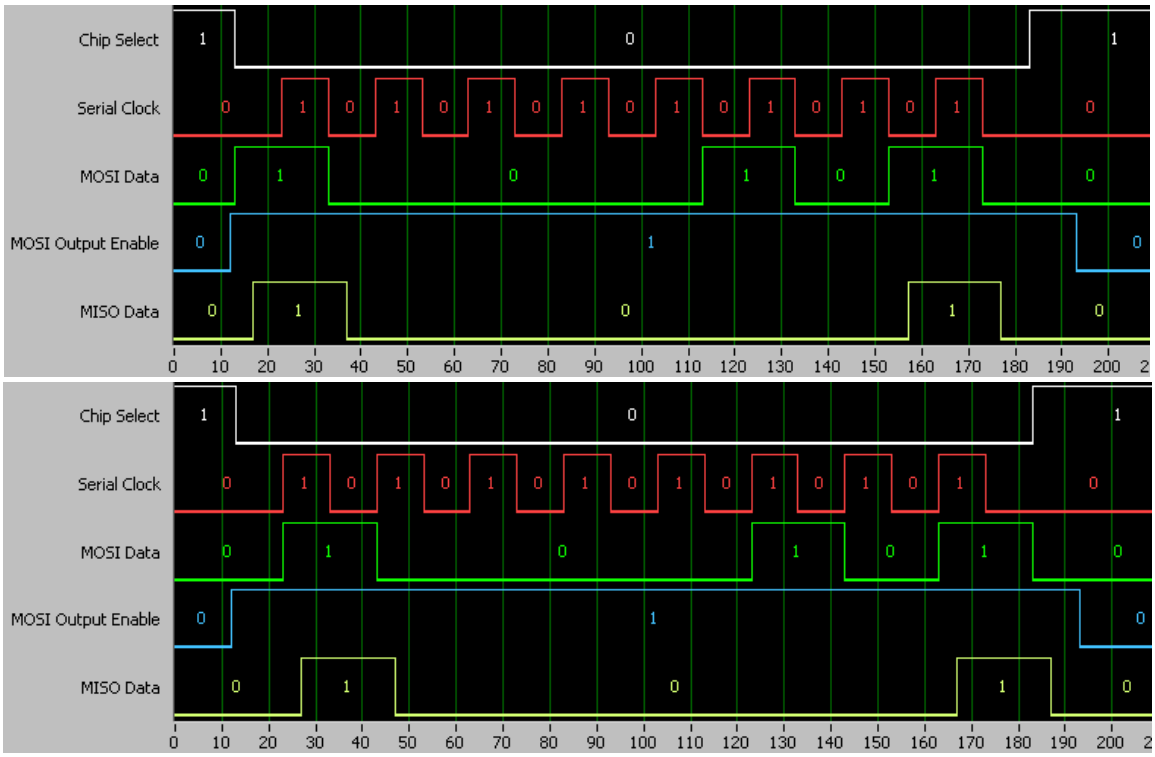

Figure 14: SPI Example Waveforms with Different Clock Phase Settings

The below examples are identical to the above excepting that CPOL = 1 and CS Asserted Polarity = 1.

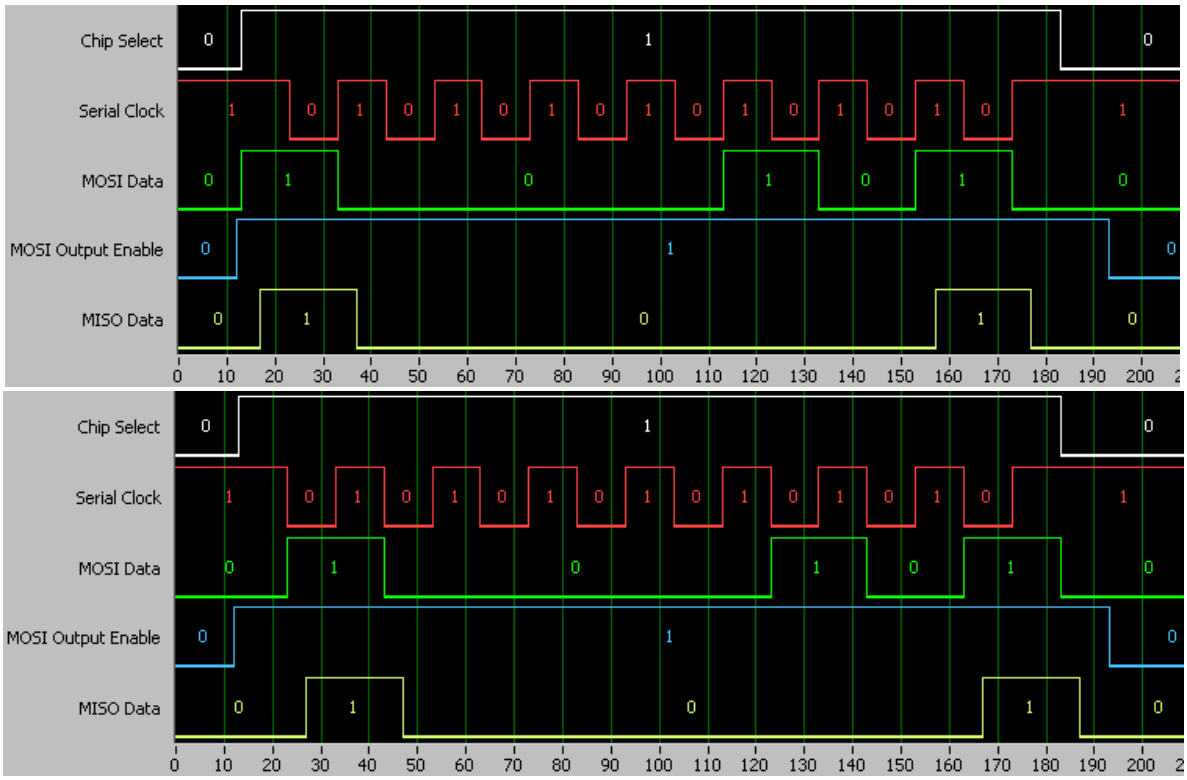

Figure 15: SPI Example Waveforms

If a 3-wire interface with bidirectional data is used, then the Tristatable MOSI control must be set to true and the appropriate starting direction and bit on which to change the direction must be chosen.

The following example shows a bidirectional transfer that starts with MOSI tristated for five serial clock cycles and then driven for three serial clock cycles (the slave in the simulation only supports 4-wire mode which is why the MISO data waveform generates a full 8-bit serial output). In SPI Mode 0, the master and slave read data on the rising edges of the serial clock and change their outputs on the falling edge. Therefore in the case where the slave device is initially driving the data line, the last falling edge of the first half of the transfer triggers the slave to change its data line to input. The master can then change its data bus to output. In order for this direction change to occur safely, the bus stalls for half a clock cycle as shown below in order to configure the NI 5644R DIO to be outputs before proceeding to toggle the clock.

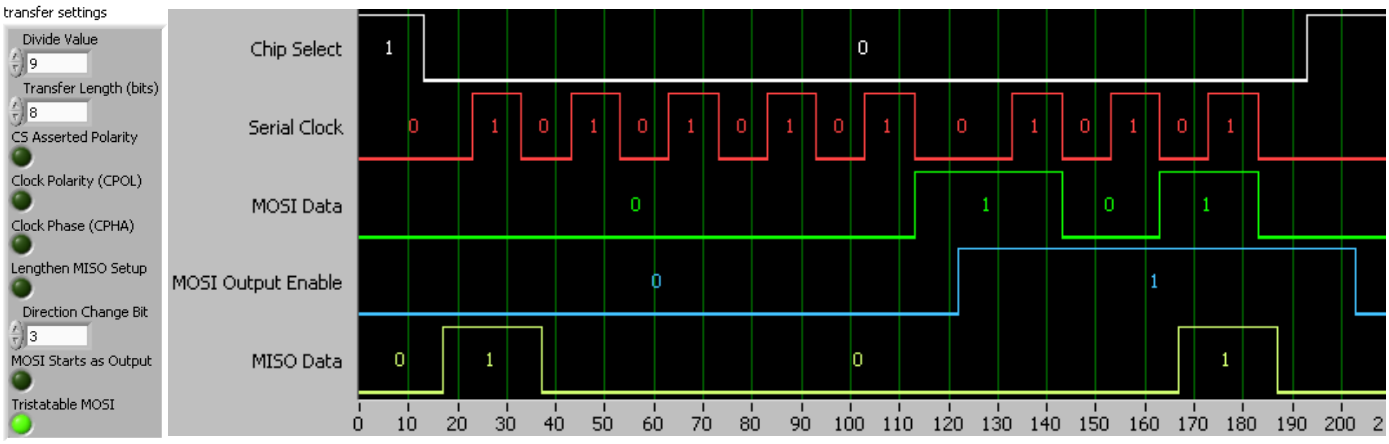

Figure 16: Bidirectional Data SPI Waveform Example

If the slave device is initially reading the data line, the last rising edge of the first half of the transfer will latch the last data bit into the slave, at this point the direction of the master output should be changed to input before sending the next edge of the clock on which the slave will enable its output. This will prevent the two devices from driving into each other which could cause damage to the slave device.

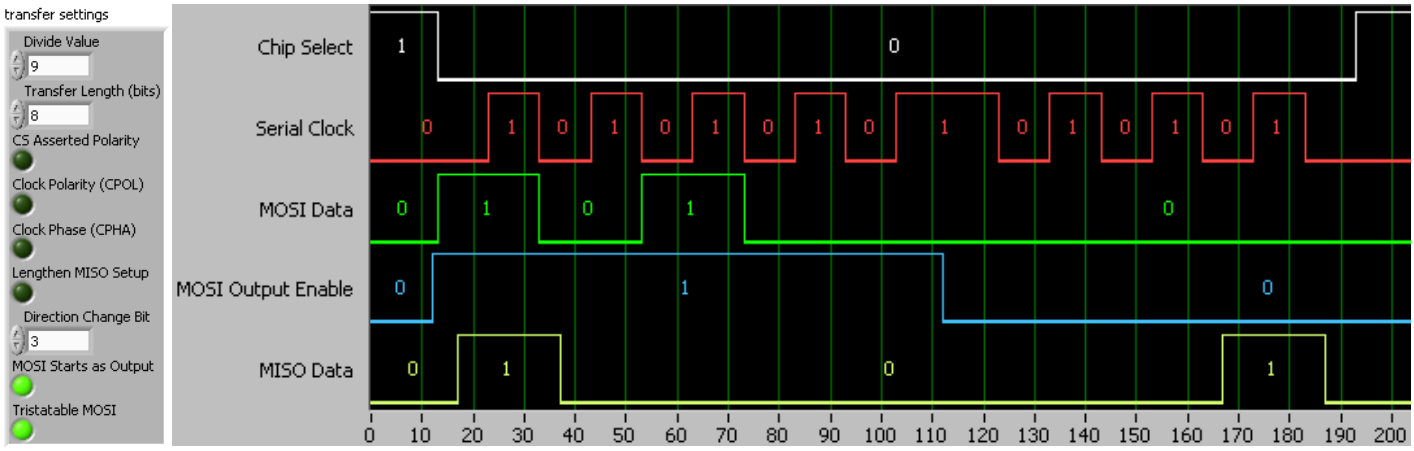

Figure 17: Bidirectional Data SPI Waveform Example

Data Bus Expansion

The SPI Master VI supports a single serial data output line and a single serial data input line as provided in this example. However, if a custom interface is needed, then the width of the data bus can easily be increased. The following diagram shows the portion of the code (inside Spi Master.vi) that handles IO data capture and generation (note that the code has been slightly reorganized and simplified to display only the relevant code):

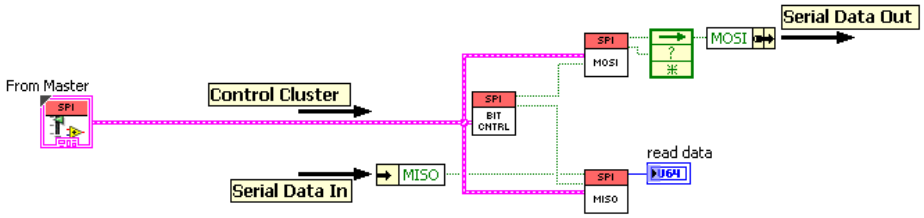

Figure 18: Single-Bit Data Handling

Additional “Mosi Logic” and “Miso Logic” VIs can be inserted as shown to widen the data path. Note that the write data controls and read data indicators will have to be processed to order the bits appropriately for your application.

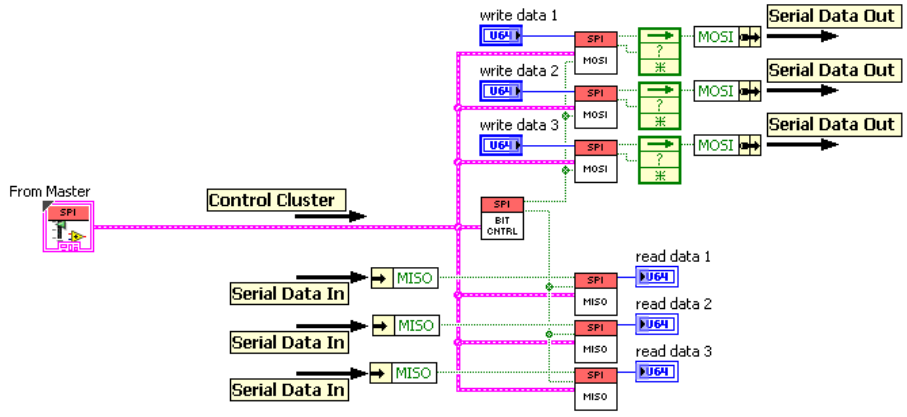

Figure 19: Multi-Bit Data Handling

System Timing

Digital interfaces require precise timing relationships between clock IO and data IO to guarantee that data will not be corrupted throughout the data path. Two types of timing relationships are typically found in digital systems: source synchronous timing and system synchronous timing. System synchronous interfaces refer to those where both a data transmitter and a data receiver use a common system clock to latch and launch data as shown below.

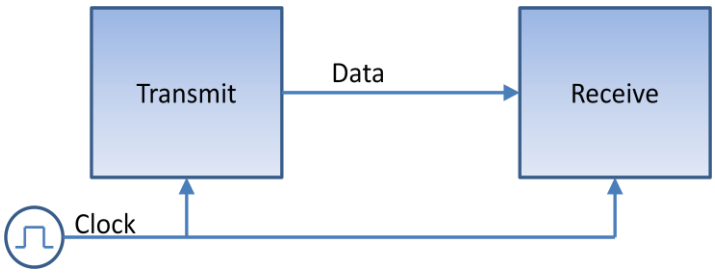

Figure 20: System Synchronous Interface

In this case, the data setup and hold time at the receiver must be validated with the knowledge of the timing of the entire system including clock-to-out time of the transmitter, clock skew between the transmitter and receiver, clock propagation delay into the receiver, etc. Due to the complex architecture of the NI 5644R, this timing cannot be guaranteed statically in all cases and this type of interface should not be used for digital control and communication using the NI 5644R DIO. A source synchronous interface is much easier to design and to verify as shown in the following figure.

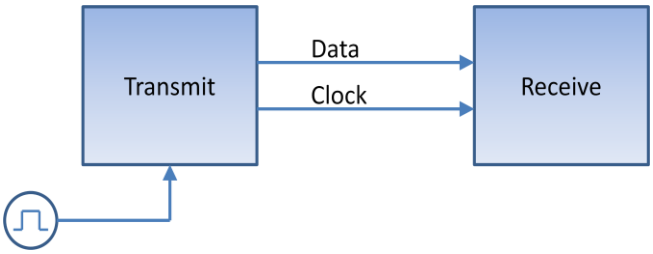

Figure 21: Source Synchronous Interface

In this case, the setup and hold time at the receiver can be verified using only the relative delay between the clock and data. For DUT control interfaces where data might be returned to the transmitter, the minimum period must accommodate the maximum clock-to-out time of the receiver as well.

The clock-to-output timing between the transmitter outputs can vary among the DIO lines (DIOx, PFIx, and Clock Out). This variance is called skew and depends on clock net skew within the FPGA, clock-to-output discrepancies between outputs of the FPGA, trace delay differences between the FPGA and the front panel, and buffer propagation delay differences among the banks of DIO. Nominally, the overall skew among all DIO should be around 1 ns. Cabling delays also affect system timing. The nominal delay through a 1 meter cable in a single direction is about 5 ns. See the Timing Example section of this document for additional information on the NI 5644R DIO timing.

The Serial Peripheral Interface example uses a source synchronous interface such that the SPI Master VI controls when data transmitted to the slave changes and when data returned to the master is captured relative to the clock transmitted by the master. A typical SPI system is shown below. The bus consists of a Chip Select line that determines when the transfer is directed to the slave and when a transfer starts and ends, a Clock which shifts logic into and out of the slave device, a Master Data Output (MOSI) line which carries data to the slave, and a Master Data Input (MISO) line which carries data to the master from the slave. Some devices do not make use of a MOSI or MISO line if only

unidirectional data transfer is supported. In some devices the MOSI/MISO functionality is multiplexed onto a single bidirectional line rather than using two dedicated direction lines to minimize pin count.

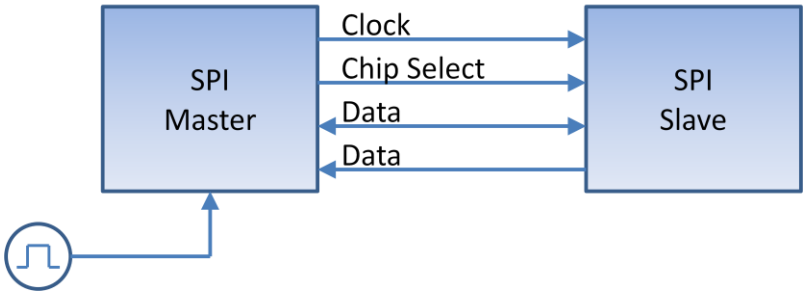

Figure 22: SPI System Topology Superset

The physical implementation of such an interface is shown below where the master device has an internal clock source that generates the clock transmitted to the slave, shifts out MOSI data, and samples MISO data. The delay element between the master and slave represents a propagation delay due to the length of the connection between the two devices which can include cabling, additional buffering, etc.

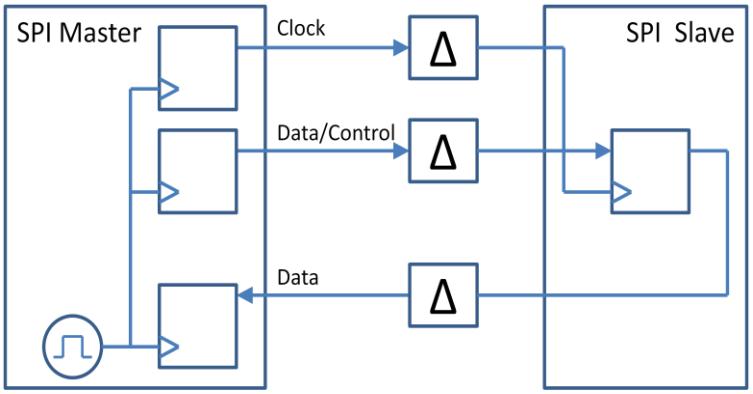

Figure 23: SPI Source Synchronous Physical Interface

Timing parameters that must be known to close static timing include (but are not limited to) the following: clock-to-out time (clock-to-Q) of the SPI clock, chip select, and MOSI signals, propagation delay from the master device to the slave device of all signals, and the propagation delay of the MISO data returning from the slave device. The setup and hold times of the input data and chip select at the slave relative to the clock should be known and the clock-to-out of the MISO data must be known as well. Some parameters such as clock skew within the master and clock duty cycle distortion are not necessary for the analysis of this system because these parameters are extremely small relative to the propagation delays which dominate the system timing.

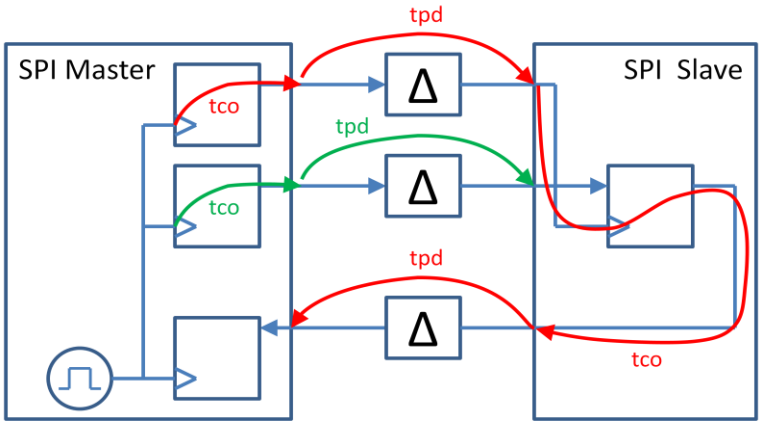

Figure 24: Timing Parameters

Timing parameter naming convention is as follows:

Clock-to-Output (Clock-to-Q) Time	tco	Clock edge to valid output delay
Propagation Delay	tpd	Includes trace delay, buffer delay, and cabling delays
Setup Time	tsetup	Required data valid time before latching clock edge
Hold Time	thold	Required data valid time after latching clock edge
Clock Period	T	Inverse of clock frequency

Master signal timing characteristics are denoted with a lowercase "m" subscript followed by the signal name. For example, the minimum clock-to-output timing of the master's clock output is denoted as $\mathrm{tco}_{\mathrm{mclk}_{\mathrm{min}}}$. Slave signal characteristics use a lower case "s". Lastly, maximum versus minimum delays make use of a subscript "min" or "max" to denote minimum and maximum respectively.

The following equations must be satisfied to close static timing of this system:

Setup time at the slave (must be applied to both Chip Select and MOSI data):

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{min}}} + \frac{\mathrm{T}_{\mathrm{min}}}{2} \geq \mathrm{tco}_{\mathrm{mdata}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{mdata}_{\mathrm{max}}} + \mathrm{tsetup}_{\mathrm{s}}$$

Hold time at the slave (must be applied to both CS and MOSI data):

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{thold}_{\mathrm{s}} \leq \mathrm{tco}_{\mathrm{mdata}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{mdata}_{\mathrm{min}}} + \frac{\mathrm{T}_{\mathrm{min}}}{2}$$

Note that the clock period is not normally used in hold equations; however, this interface is source synchronous and clock and data are changed on opposite edges of the serial clock.

Setup time at the master (applies to MISO data):

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{tco}_{\mathrm{sdata}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{sdata}_{\mathrm{max}}} + \mathrm{tsetup}_{\mathrm{m}} \leq \frac{\mathrm{T}_{\mathrm{min}}}{2} * \mathrm{N}_{\mathrm{s}}$$

$$\mathrm{N}_{\mathrm{s}} = \begin{cases} 1, & \textit{Full period sampling disabled} \\ 2, & \textit{Full period sampling enabled} \end{cases}$$

Hold time at the master (applies to MISO data):

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{min}}} + \mathrm{tco}_{\mathrm{sdata}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{sdata}_{\mathrm{min}}} + \frac{\mathrm{T}}{2} * \mathrm{N}_{\mathrm{h}} \geq \mathrm{thold}_{\mathrm{m}}$$

$$\mathrm{N}_{\mathrm{h}} = \begin{cases} 1, & \textit{Full period sampling disabled} \\ 0, & \textit{Full period sampling enabled} \end{cases}$$

The slave timing parameters used in the above equations can usually be found in the datasheet for the device being accessed. The SPI Slave VI included within the NI 5644R SPI Example has the following characteristics. Note that these timing parameters are approximate and have not been fully verified; in general, the interface will be run much slower than the frequency at which the slave VI is operating which means that the setup, hold, and clock-to-output times do not dominate system performance. The following setup, hold, and clock-to-output numbers should be used in the above timing analysis for the slave timing characteristics if the SPI Slave VI provided in the example project is used.

SPI Slave VI Setup (tsetup) Requirement:

$$\mathrm{tsetup_s} = 2 * \mathrm{T_{SCTL}}$$

SPI Slave VI Hold (thold) Requirement:

$$\mathrm{thold_s} = 2 * \mathrm{T_{SCTL}}$$

SPI Slave VI Clock-to-Output (tco):

$$\mathrm{tco_{sdata_{min}}} = 3 * \mathrm{T_{SCTL}}$$
$$\mathrm{tco_{sdata_{max}}} = 5 * \mathrm{T_{SCTL}}$$

Timing Example

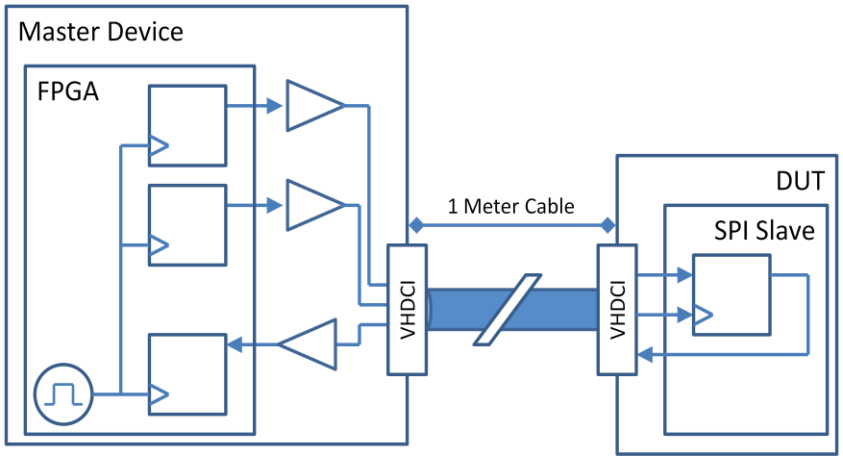

Figure 25: Typical System Schematic

Setup time at the slave:

$$\mathrm{tco_{mclk_{min}}} + \mathrm{tpd_{mclk_{min}}} + \frac{\mathrm{T_{min}}}{2} \geq \mathrm{tco_{mdata_{max}}} + \mathrm{tpd_{mdata_{max}}} + \mathrm{tsetup_s}$$

The clock-to-output time of the master ($\mathrm{tco_{mclk_{min}}}$) can be considered as the total time from the rising edge of the clock inside the master FPGA to the time that the signal shows up at the VHDCI connector. For the NI 5644R, $\mathrm{tco_{mclk_{min}}} \approx 2\,ns$ and $\mathrm{tco_{mclk_{max}}} \approx 8\,ns$. Similarly the input delay from the VHDCI connector to the logic inside the FPGA can be considered the same as the clock-to-output time. Let the delay through a one meter cable range from 4.5 ns to 7 ns for the purpose of this analysis. Let the slave device have a setup time requirement of 3 ns, hold time requirement of 2 ns, and a maximum clock-to-output time of 5 ns. Therefore, the minimum clock period of the interface is:

$$2\ \mathrm{ns} + 4.5\ \mathrm{ns} + \frac{\mathrm{T_{min}}}{2} \geq 8\ \mathrm{ns} + 7\ \mathrm{ns} + 3\ \mathrm{ns}$$

$$T_{min} \geq 23\ ns \leftrightarrow 43.5\ MHz$$

Hold time at the slave:

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{thold}_{\mathrm{s}} \leq \mathrm{tco}_{\mathrm{mdata}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{mdata}_{\mathrm{min}}} + \frac{\mathrm{T}_{\mathrm{min}}}{2}$$

Using the same numbers above:

$$8\ \mathrm{ns} + 7\ \mathrm{ns} + 2\ \mathrm{ns} \leq 2\ \mathrm{ns} + 4.5\ \mathrm{ns} + \frac{\mathrm{T}_{\mathrm{min}}}{2}$$
$$T_{min} \geq 21\ ns \leftrightarrow 47.6\ MHz$$

The hold equation shows that the interface can run faster than the setup equation. However, both equations must be satisfied for proper operation of the interface. Therefore, the maximum of the minimum period values must be used as the total minimum period of the system (e.g. 23 ns is greater than 21 ns; therefore, 23 ns must be considered the minimum period).

Setup time at the master:

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{max}}} + \mathrm{tco}_{\mathrm{sdata}_{\mathrm{max}}} + \mathrm{tpd}_{\mathrm{sdata}_{\mathrm{max}}} + \mathrm{tsetup}_{\mathrm{m}} \leq \frac{\mathrm{T}_{\mathrm{min}}}{2} * \mathrm{N}_{\mathrm{s}}$$
$$\mathrm{N}_{\mathrm{s}} = \begin{cases} 1, & \text{Full period sampling disabled} \\ 2, & \textit{Full period sampling enabled} \end{cases}$$

Using the same numbers above with full-period sampling enabled:

$$8\ \mathrm{ns} + 7\ \mathrm{ns} + 5\ \mathrm{ns} + 7\ \mathrm{ns} + 10\ \mathrm{ns} \leq \frac{\mathrm{T}_{\mathrm{min}}}{2} * 2$$
$$\mathrm{T}_{\mathrm{min}} \geq 37\ \mathrm{ns} \leftrightarrow 27\ \mathrm{MHz}$$

Hold time at the master:

$$\mathrm{tco}_{\mathrm{mclk}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{mclk}_{\mathrm{min}}} + \mathrm{tco}_{\mathrm{sdata}_{\mathrm{min}}} + \mathrm{tpd}_{\mathrm{sdata}_{\mathrm{min}}} + \frac{\mathrm{T}}{2} * \mathrm{N}_{\mathrm{h}} \geq \mathrm{thold}_{\mathrm{m}}$$
$$\mathrm{N}_{\mathrm{h}} = \begin{cases} 1, & \text{Full period sampling disabled} \\ 0, & \textit{Full period sampling enabled} \end{cases}$$

Using the same numbers above with full-period sampling enabled:

$$2\ \mathrm{ns} + 4.5\ \mathrm{ns} + 0\ \mathrm{ns} + 4.5\ \mathrm{ns} + 0\ \mathrm{ns} = 11\ \mathrm{ns} \geq \mathrm{thold}_{\mathrm{m}}$$

Therefore, as long as the insertion delay (propagation delay from the VHDCI input to the FPGA plus the hold time requirement of the FPGA) of the master device is less than 11 ns (in this example), hold time will be met. In general this should be true for most devices; therefore, the full period sampling mode of the example IP can be used for most cases.

IP FPGA Usage and Timing Information

Estimated Resource Utilization

Design Element	FPGA Slices	Registers	Look-Up Tables (LUT)
SPI Master	170	225	200
SPI Slave	50	85	80
SPI Slave User Logic (as provided in example)	130	275	200

Clocking Requirements

The SPI Master and Slave VIs both operate only within the LabVIEW FPGA Single-Cycle Timed Loop (SCTL). The loop runs off of a user-selected clock. Care should be taken to select the appropriate SCTL clocks for your application. The SPI serial clock frequency generated by this IP is calculated as follows:

$$SPI\ Clock\ Frequency = \frac{\text{SCTL Clock Frequency}}{(\text{Divide Value} + 1) * 2}$$

Divide Value is in the range [0, 255]. Therefore, the serial clock frequency is bounded in the range:

$$\left[\frac{\text{SCTL Clock Frequency}}{2}, \frac{\text{SCTL Clock Frequency}}{512}\right]$$

Therefore, a 200 MHz SCTL clock will provide frequencies within the range from 390 kHz to 100 MHz (note that only 256 actual frequencies within this range can be generated). Operating the SCTL at slower clock frequencies is recommended due to the increasingly difficult task of meeting timing within the FPGA as the SCTL clock frequency increases; this is especially true for fuller FPGAs (e.g. large designs).

Release Notes

20 November, 2012: Initial release correlating with the VST support package “NI LabVIEW 2012 Support for NI PXIe-5644R 1.0.0”.