I^2C to SPI Bridge

Reference Design

FPGA-RD-02157-1.2

February 2020

Disclaimers

Contents

Figures

Tables

1. Introduction

I²C and SPI are the two widely used bus protocols in today's embedded systems. The I²C bus has a minimum pin count requirement and therefore a smaller footprint on the board. The SPI bus provides a synchronized serial link with performance in MHz range. As embedded systems are required to support an increasing number of protocols and interfaces, bridge designs targeting popular protocols provide solutions to reduce development time and cost. This reference design implements an I²C slave to SPI master bridge. It serves as an interface between the standard I²C bus of a microcontroller and an SPI bus. This allows the microcontroller to communicate directly with the SPI bus through its I²C bus. This bridge extends the available SPI ports for the application processor.

The design is implemented in mixed VHDL and Verilog language. The Lattice iCEcube2™ Place and Route tool, integrated with the Synopsys Synplify Pro® and LSE (Lattice Synthesis Engine) synthesis tools is used for the implementation of the design. The design can be targeted to other iCE40™ FPGA product family devices.

2. Features

- I²C bus slave interface operating up to 400 kHz
- SPI Master operating up to 1.8 Mbit/s
- 200 byte data buffer
- Up to four slave select outputs
- Up to four programmable I/O pins
- Operating Voltage: 1.8 V to 3.3 V
- Low power mode
- Active low interrupt output

3. System Block Diagram

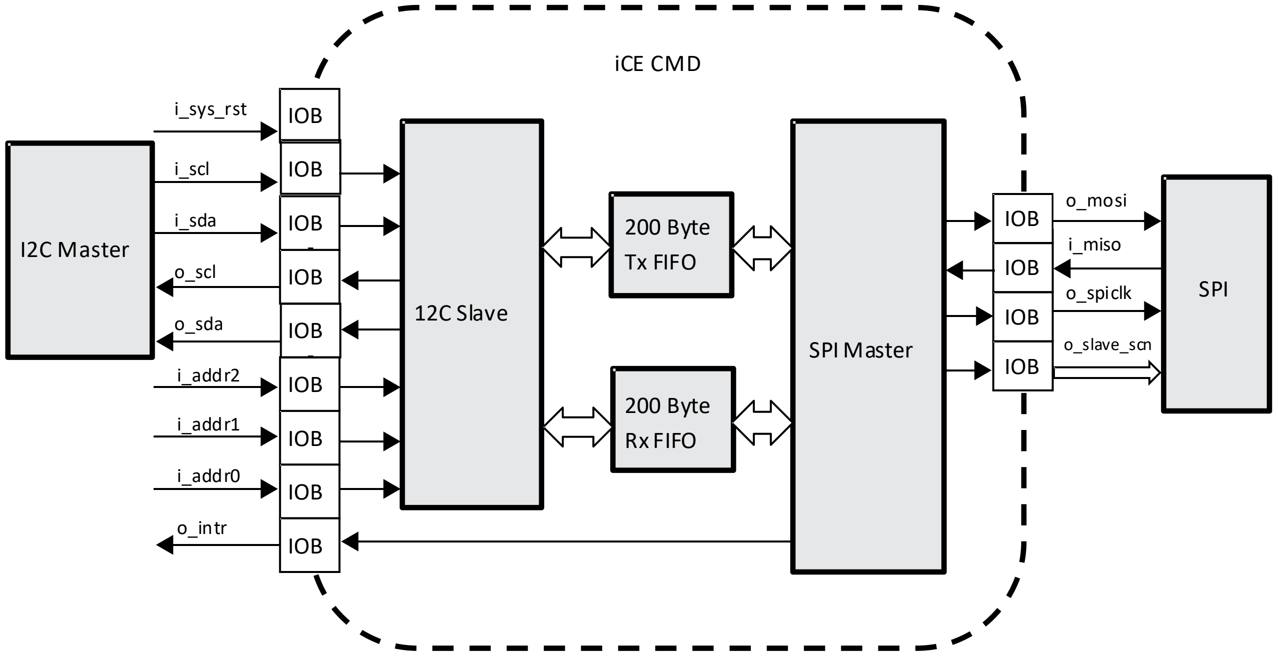


Figure 3.1. System Block Diagram

4. Signal Description

Table 4.1. Signal Descriptions

Signal	Width	Type	Description
i_sys_clk	1	Input	System Clock
i_sys_rst	1	Input	Active High Asynchronous reset
i_scl	1	Input	SCL input to I^{2}C Slave from I^{2}C Master
i_sda	1	Input	SDA input to I^{2}C Slave from I^{2}C Master
o_sda	1	Output	SDA output from I^{2}C Slave to I^{2}C Master
o_scl	1	Output	SCL output from I^{2}C Slave to I^{2}C Master
o_scl_tri_en	1	Output	Tristate Enable for I^{2}C SCL
o_sda_tri_en	1	Output	Tristate Enable for I^{2}C SDA
i_addr2	1	Input	Address bit for programming I^{2}C Slave Address
i_addr1	1	Input	Address bit for programming I^{2}C Slave Address
i_addr0	1	Input	Address bit for programming I^{2}C Slave Address
o_mosi	1	Output	Serial output from SPI Master
i_miso	1	Input	Serial Input to SPI Master
o_slave_csn	1	Output	Slave select from SPI Master
o_spiclk	1	Output	Serial Clock generated by SPI Master
o_intr	1	Output	Output Interrupt

5. Functional Description

5.1. Slave Address

The first seven bits of the first byte sent after a START condition defines the slave address of the device being accessed on the bus. The eighth bit determines the direction of the message. A ‘0’ in the least significant position of the first byte means that the master will write information to a selected slave. A ‘1’ in this position means that the master will read information from the slave. When an address is sent, each device in a system compares the first seven bits after the START condition with its address. If they match, the device considers itself addressed by the master as a slave-receiver or slave-transmitter, depending on the R/W bit.

A slave address is comprised of a fixed and a programmable part. The programmable part of the slave address enables the maximum possible number of such devices to be connected to the I^{2}C-bus. Since this has three programmable address bits (defined by the i_addr2, i_addr1, and i_addr0 pins), it is possible to have eight of these devices on the same bus.

The state of the i_addr2, i_addr1, and i_addr0 pins are latched at reset. Changes made after reset will not alter the address.

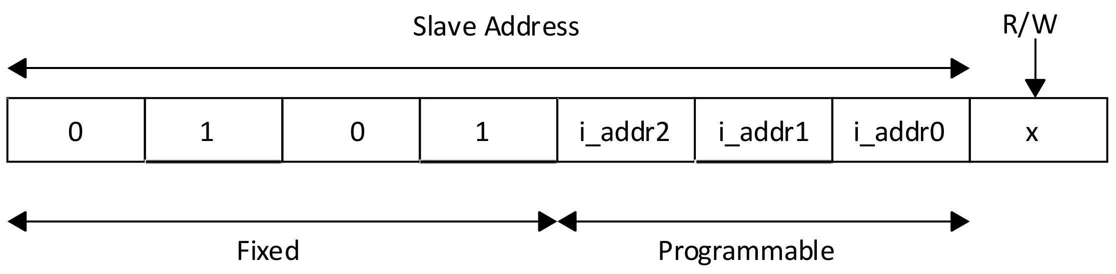


Figure 5.1. Slave Address

5.2. Write to data buffer

All communications occur through the data buffer. The data buffer is 200 bytes deep. A message begins with the slave address, followed by the Function ID. Depending upon the Function ID, zero to 200 data bytes can follow.

I^{2}C slave will place the data received into a buffer and continue loading the buffer until a STOP condition is received. After the STOP condition is detected, further communications will not be acknowledged until the function designated by the Function ID has been completed.

S	Slave Address	W	A	Function ID	A	0 to 200 bytes	A	P

Figure 5.2. Write to Data Buffer

5.3. SPI Read and Write

Data in the buffer will be sent to the SPI port if the Function ID is 01h to 0Fh. The Function ID contains the Slave Select (SS) to be used for the transmission on the SPI port. There are four Slave Selects that can be used, with each SS being selected by one of the bits in the Function ID. There is no restriction on the number or combination of Slave Selects that can be enabled for an SPI message. If more than one SSn pin is enabled at one time, the user should be aware of possible contention on the data outputs of the SPI slave devices.

0	0	0	0	SS3	SS2	SS1	SS0

Figure 5.3. Function ID

The data on the SPI port will contain the same information as the I^{2}C-bus data, but without the slave address and Function ID. For example, if the message shown in Figure 5.4 is transmitted on the I^{2}C-bus, the SPI bus will send the message shown in Figure 5.5.

S	Slave Address	W	A	Function ID	A	Data 1	A	…………	A	Data n	A	P

Figure 5.4. I^{2}C-Bus Message-Write to Buffer

S	Slave Address	R	A	Data 1	A	…………	A	Data n	NA	P

Figure 5.5. SPI Message

As the data is transmitted from the MOSI pin, SPI Master also read from the MISO pin and saved in the data buffer. Therefore, the old data in the buffer is overwritten. The data in the buffer can then be read back. If the data from the SPI bus needs to be returned to the I^{2}C-bus master, the process must be completed by reading the data buffer. I^{2}C Master can read back the data.

5.4. Read from Buffer

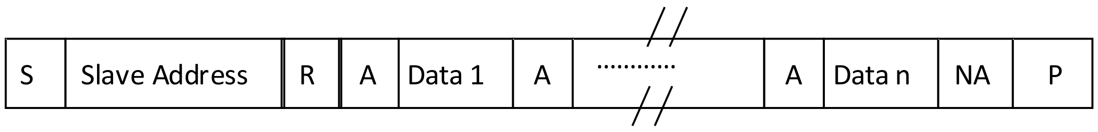


Figure 5.6. Read from Buffer

5.5. Table for Function IDs

Table 5.1. Function IDs and their Functions

Function ID	Function
01-0Fh	Write to data buffer (Function ID specifies to which SPI slave data should be written)
F0h	Configure SPI interface
F1h	Clear the interrupt
F2h	Idle mode
F4h	GPIO Write
F5h	GPIO Read
F6h	GPIO Enable
F7h	GPIO Configuration

5.6. Configure SPI interface –Function ID F0h

The SPI hardware operating mode, data direction, and frequency can be changed by sending a ‘Configure SPI Interface’ command to the I^{2}C-bus. After slave address is transmitted on the bus, the Configure SPI Interface Function ID (F0h) is sent followed by a byte which will define the SPI communications.

Table 5.2 shows Function ID F0h bit allocation. It shows the procedure for configuring SPI interface. Table 5.3 shows Function ID F0h bit description.

Table 5.2. Configure SPI Interface

Bit	7	6	5	4	3	2	1	0
Symbol	X	x	Order	x	Mode1	Mode0	F1	F0
Reset	x	x	0	x	0	0	0	0

Table 5.3. Configure SPI Interface Bit Description

Bit	Symbol	Description
7:6	-	Reserved
5	Order	0'-MSB is sent out first
		1'-LSB is sent out first
4	-	Reserved
3:2	Mode1:Mode0	Mode Selection
		01-CPOL-0,CPHA-1
		10-CPOL-1,CPHA-0
		11-CPOL-1,CPHA-1
1:0	F1:F0	SPI clock Rate
		SC18IS602/602B:
		00-1843 khz
		01-461 khz
		10-115 khz
		11-58 khz
		SC18IS603:
		00-fosc/4
		01-fosc/16
		10-fosc/64
		11-fosc/128

5.7. Clear Interrupt-Function ID F1h

An interrupt is generated after completion of every SPI transaction. This interrupt can be cleared (INT pin HIGH) by sending a 'Clear Interrupt' command.

S	Slave Address	W	A	F1h	A	P

Figure 5.7. Clear Interrupt

5.8. Idle Mode- Function ID F2h

A low-power mode may be entered by sending the 'Idle Mode' command. The Idle mode will be exited when its I^{2}C-bus address is detected.

S	Slave Address	W	A	F2h	A	P

Figure 5.8. Idle Mode

5.9. GPIO Enable –Function ID F6h

At reset, the Slave Select pins (SS0, SS1, SS2 and SS3) are configured to be used as slave select outputs. If these pins are not required for the SPI functions, they can be used as GPIO after they are enabled as GPIO. Any combination of pins may be configured to function as GPIO or Slave Selects. After the GPIO Enable function is sent, the ports defined as GPIO will be configured as quasi-bidirectional. The data byte following the F6h command byte will determine which pins can be used as GPIO. A logic 1 will enable the pin as a GPIO, while a logic 0 will disable GPIO control.

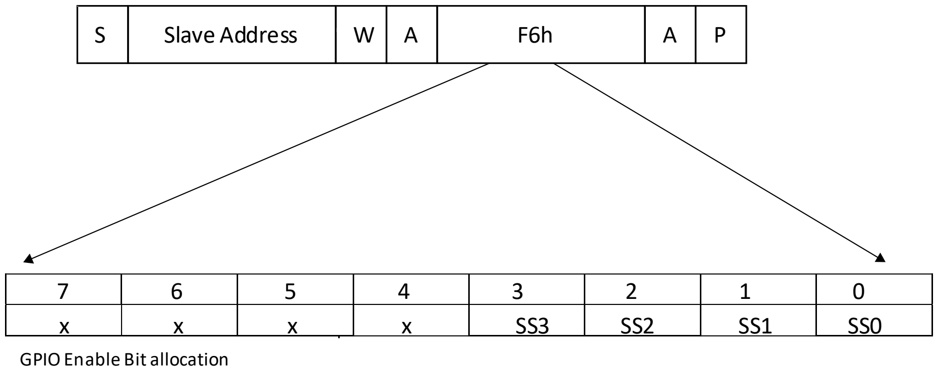


Figure 5.9. GPIO Read

5.10. GPIO Configuration –Function ID F7h

The pins defined as GPIO may be configured by software to one of four types on a pin-by-pin basis. These are: quasi-bidirectional, push-pull, open-drain, and input-only. Two bits select the output type for each port pin. The SSn pins defined as GPIO, for example SS0.0 and SS0.1, may be configured by software to one of four types. These are: Bidirectional, Input, Output- pullup, Output- pulldown. Two configuration bits in GPIO Configuration register for each pin select the type for each pin.

Table 5.4. GPIO Configuration Bit Allocation

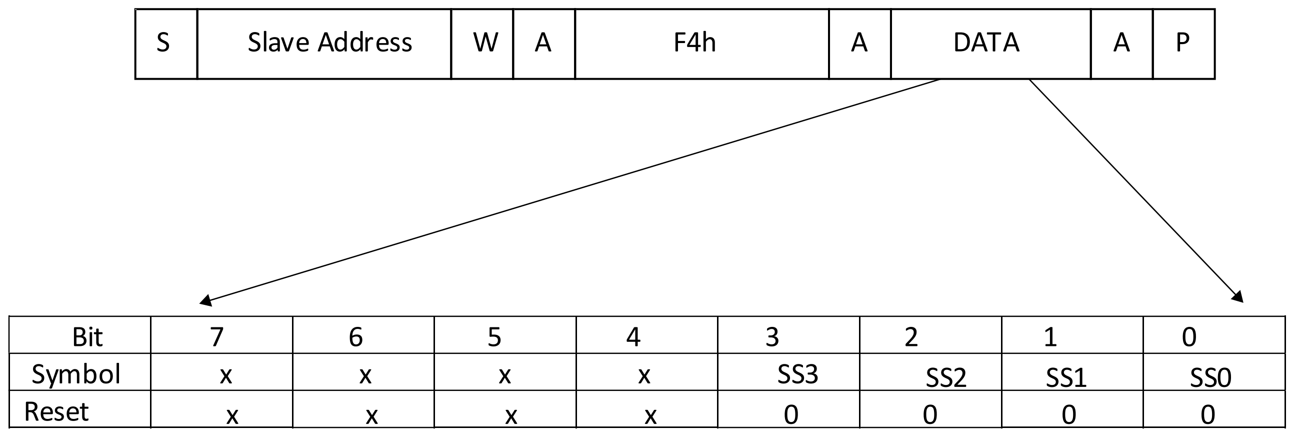
7	6	5	4	3	2	1	0
SS3.1	SS3.0	SS2.1	SS2.0	SS1.1	SS1.0	SS0.1	SS0.0

Table 5.5. GPIO Configuration Bit Description

Bit	Symbol	Description
7	SS3.1	SS3[1:0]-00:Bidirectional
6	SS3.0	SS3[1:0]-01:Input
		SS3[1:0]-10:Output-pullup
		SS3[1:0]-11:Output:pulldown
5	SS2.1	SS2[1:0]-00: Bidirectional
4	SS2.0	SS2[1:0]-01: Input
		SS2[1:0]-10: Output-pullup
		SS2[1:0]-11: Output-pulldown
3	SS1.1	SS1[1:0]-00: Bidirectional
	SS1.0	SS1[1:0]-01: Input
		SS1[1:0]-10: Output-pullup
		SS1[1:0]-11: Output-pulldown
2	SS0.1	SS0[1:0]-00:Bidirectional
	SS0.0	SS0[1:0]-01:Input
		SS0[1:0]-10:Output-pullup
1		SS0[1:0]-11:Output-pulldown

5.11. GPIO Write-Function ID F4h

The state of the pins defined as GPIO may be changed using the Port Write function. The data byte following the F4h command will determine the state of SS3, SS2, SS1, and SS0, if they are configured as GPIO. The Port Enable function will define if these pins are used as SPI Slave Selects or if they are GPIO.



S	Slave Address	W	A	F4h	A	DATA	A	P		

Bit	7	6	5	4	3	2	1	0
Symbol	x	x	x	x	SS3	SS2	SS1	SS0
Reset	x	x	x	x	0	0	0	0

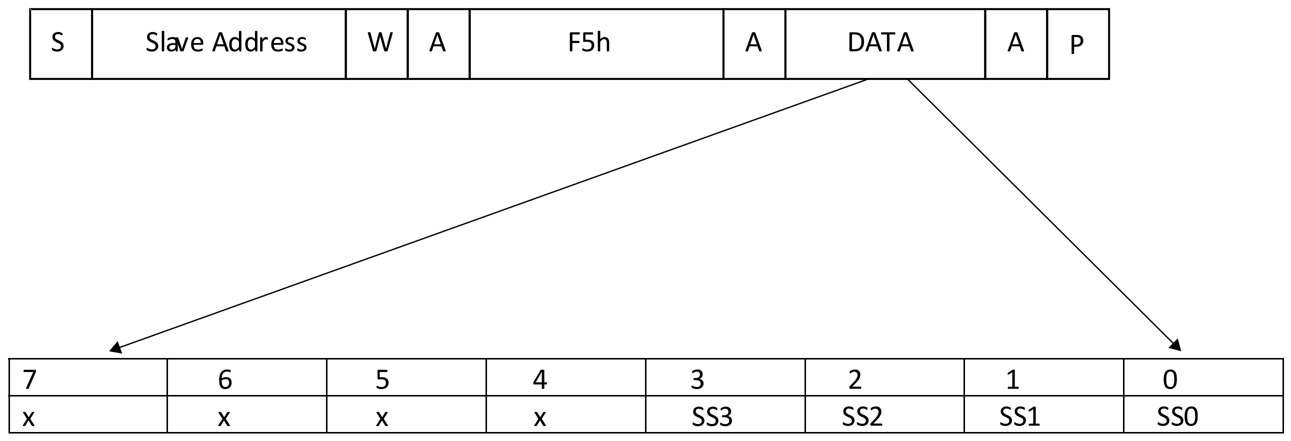
GPIO Write Bit Allocation

Figure 5.10. GPIO Write

5.12. GPIO Read-Function ID F5h

The state of the pins defined as GPIO may be read into the data buffer using the GPIO Read function. Note that this function does not return the value of the GPIO. To receive the GPIO contents, a one-byte Read Buffer command would be required. The value of the Read Buffer command will return the following byte. Data for pins not defined as GPIO are undefined. A GPIO Read is always performed to update the GPIO data in the buffer. The buffer is undefined after the GPIO data is read back from the buffer. Therefore, reading data from the GPIO always requires a two-message sequence (GPIO Read, followed by Read Buffer).

The data byte following read buffer command is the GPIO read data from read buffer.



S	Slave Address	W	A	F5h	A	DATA	A	P

7	6	5	4	3	2	1	0
x	x	x	x	SS3	SS2	SS1	SS0

GPIO Read bit allocation

Figure 5.11. GPIO Read

5.13. SPI Interface

The SPI interface can support Mode 0 through Mode 3 of the SPI specification and can operate up to 1.8 Mbit/s. The SPI interface uses at least four pins: SPICLK, MOSI, MISO, and Slave Select (SSn).

SSn are the slave select pins. In a typical configuration, an SPI master selects one SPI device as the current slave. There are actually four SSn pins (SS0, SS1, SS2 and SS3) to allow the bus to communicate with multiple SPI devices.

The Master generates the SPICLK (SPI clock) signal in order to send and receive data. The SCLK, MOSI, and MISO are typically tied together between two or more SPI devices. Data flows from the master to slave on the MOSI pin and the data flows from slave to master on the MISO pin.

5.14. Initialization Conditions

An asynchronous active high reset signal assertion is necessary to initialize the Controller to proper operating state.

6. Timing Diagram

6.1. Write Operation

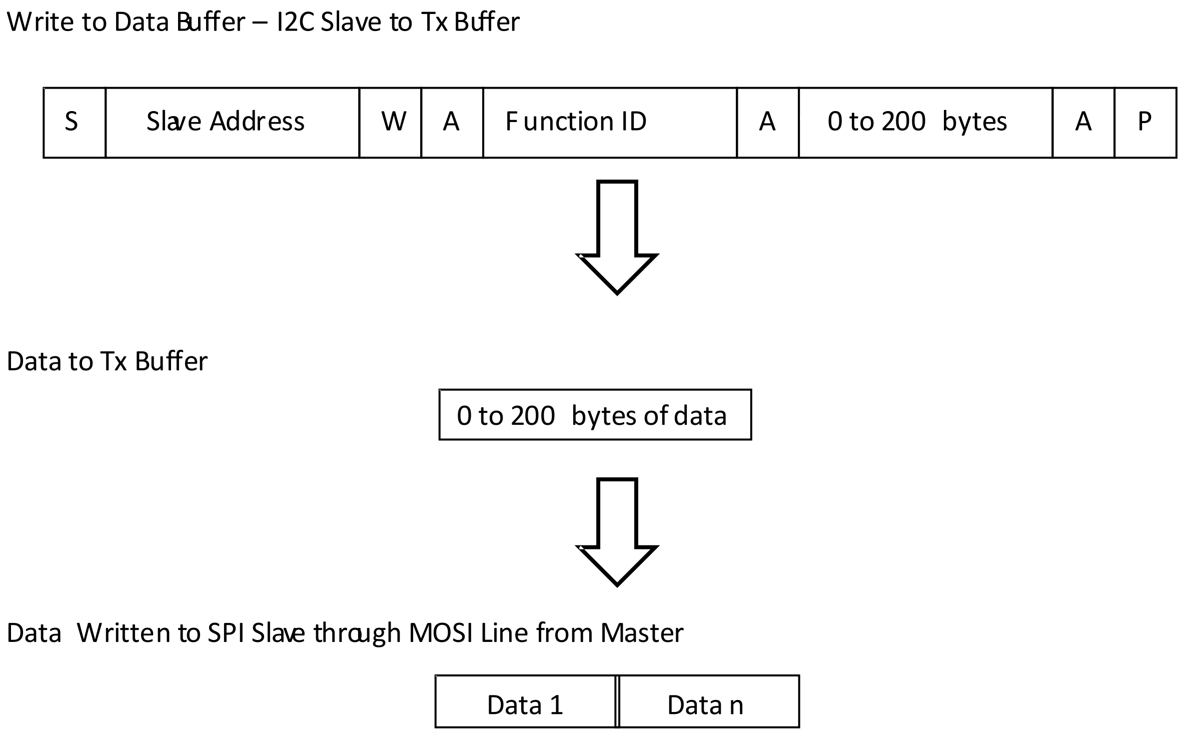

Write to Data Buffer – I2C Slave to Tx Buffer
S
Slave Address
W
A
Function ID
A
0 to 200 bytes
A
P
Data to Tx Buffer
0 to 200 bytes of data
Data Written to SPI Slave through MOSI Line from Master
Data 1
Data n

Figure 6.1. Timing Diagram - Write Operation

6.2. Read Operation

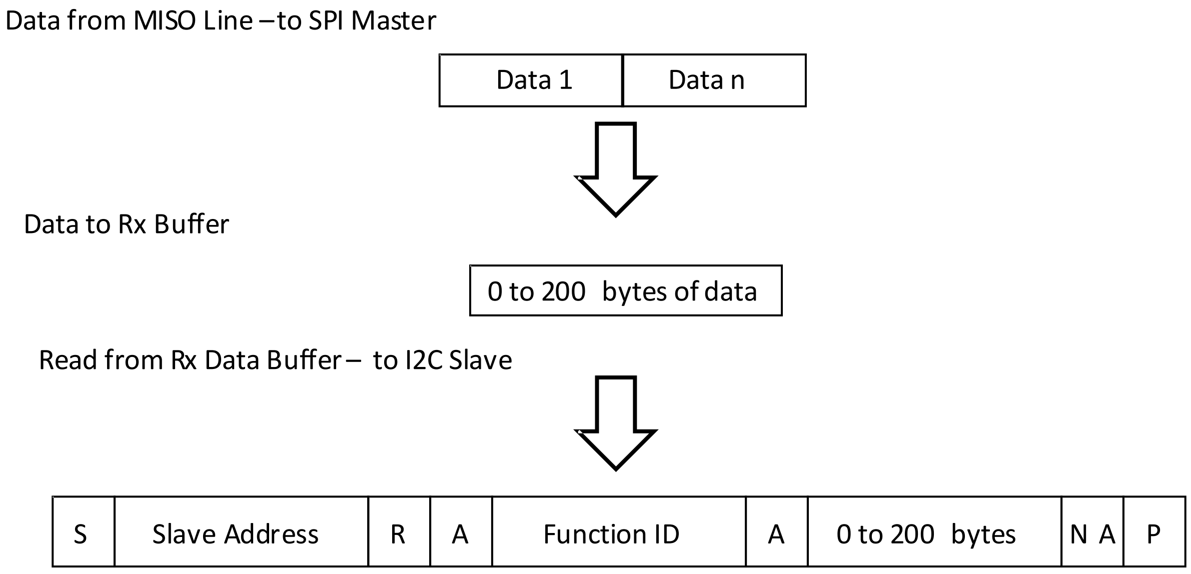
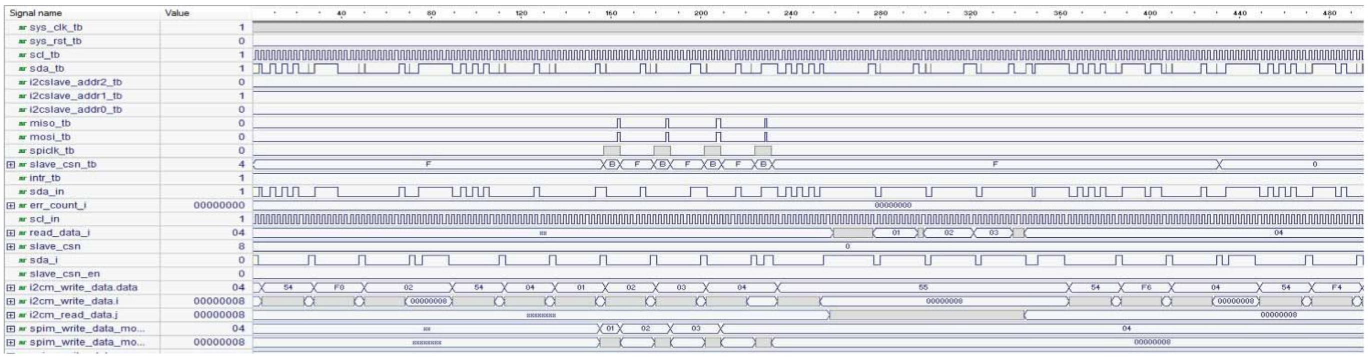


Figure 6.2. Timing Diagram - Read Operation

7. Simulation Waveforms



Signal name	Value
sys_clk_tb	1
sys_rst_tb	0
scl_tb	1
sda_tb	1
i2cslave_addr2_tb	0
i2cslave_addr1_tb	1
i2cslave_addr0_tb	0
miso_tb	0
mosi_tb	0
spiclk_tb	0
slave_csn_tb	4
intr_tb	1
sda_in	1
err_count_i	00000000
scl_in	1
read_data_i	04
slave_csn	8
sda_i	0
slave_csn_en	0
i2cm_write_data.data	04
i2cm_write_data.i	00000008
i2cm_read_data.j	00000008
spim_write_data_mo...	04
spim_write_data_mo...	00000008

Figure 7.1. Simulation Waveforms

8. Operation Sequence and Usage Example

The following example describes a typical sequence of events required to read the contents of an SPI-based EEPROM. This example assumes that the Controller is configured to respond to address 50h. A START condition is shown as ‘S’, while a STOP condition is ‘P’. The data is presented in hexadecimal format.

1. The first message is used to configure the SPI port for mode and frequency.

   ST,50,F0,02,SP SPI frequency 115 kHz using Mode 0
2. An SPI EEPROM first requires that a Write Enable command be sent before data can be written.

   ST,50,04,06,SP EEPROM write enable using SS2, assuming the Write Enable is 06h
3. Clear the interrupt. This is not required if using a polling method rather than interrupts.

   ST,50,F1,SP Clear interrupt
4. Write the 8 data bytes. The first byte (Function ID) tells which Slave Select output to use. This example uses SS2 (shown as 04h). The first byte sent to the EEPROM is normally 02h for the EEPROM write command. The next one or two bytes represent the sub address in the EEPROM. In this example, a twobyte sub address is used. Bytes 00 and 30 would cause the EEPROM to write to sub address 0030h. The next eight bytes are the eight data bytes that will be written to sub addresses 0030h through 0037h. ST,50,04,02,00,30,01,02,03,04,05,06,07,08,SP Write 8 bytes using SS2
5. When an interrupt occurs, do a Clear Interrupt or wait until the slave responds to its I^{2}C-bus address.

   ST,50,F1,SP Clear interrupt
6. Read the 8 bytes from the EEPROM. Note that we are writing a command, even though we are going to perform a read from the SPI port. The Function ID is again 04h, indicating that we are going to use SS2. The EEPROM requires that we send a 03h for a read, followed by the sub address you would like to read. We are going to read back the same data previously written, so this means that the sub address should be 0030h. We would like to read back 8 bytes so we can send eight bytes of FFh to tell the Master to send eight more bytes on MOSI. While it is sending these eight data bytes, it is also reading the MISO pin and saving the data in the buffer.

9. Implementation

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

Table 9.1. Performance and Resource Utilization

Device Family	Language	Synthesis Tool	Utilization (LUTs)	f_{MAX} (MHz)	I/Os	Architecture Resources
iCE40[1]	VHDL and Verilog	LSE	715	>50	25	(127/160) PLBs
		Syn Pro	768	>50	25	(122/160) PLBs

Note:

1. Performance and utilization characteristics are generated using iCE40LP1K-CM121 with iCEcube2 2014.12 design software and LSE support.

References

- iCE40 LP/HX Family Data Sheet (FPGA-DS-02029)

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.2, February 2020

Section	Change Summary
All	• Changed document number from RD1172 to FPGA-RD-02157. • Updated document template.
Disclaimers	Added this section.

Revision 1.1, February 2015

Section	Change Summary
Implementation	Updated Implementation section. Updated Table 9.1, Performance and Resource Utilization. • Added LSE support.
References	Updated References section.
Technical Support Assistance	Updated Technical Support Assistance information.

Revision 1.0, April 2013

Section	Change Summary
All	Initial release.