

iCE65 as SPI Master

Overview

This design example illustrates the implementation of an SPI Master using iCE65 FPGAs. It generates all the necessary interfacing signals to connect an SPI Master to an SPI Slave as well as Microprocessor using standard address/data bus system and hence enables host processor to connect to the SPI bus and transact with SPI Slaves. The Master is capable of handling up to four SPI slaves.

SPI Description

SPI™, or Serial Peripheral Interface is a serial, synchronous 4-wire communication protocol that is standard across microprocessors, microcontrollers, and peripherals that enables communication between microprocessors, peripherals and inter-processors. The communication makes use of Master - Slave system using 4 lines (3 common and 1 exclusive) as follows:

- MOSI (Master Out, Slave in)
- MISO (Master In, Slave Out)
- SCLK (Serial Clock)
- SS (Slave select)

The main features of the SPI interface, unlike many other serial interfaces, is that it allows for a full duplex serial communication. This is possible due to the presence of exclusive lines for data in both directions. The availability of another exclusive line used to select a particular Slave, reduces the overhead of address decoding in a multi-slave environment. These two features combine to add great speeds on the SPI bus (with clock speeds up to 70 MHz).

A typical SPI Bus with a Master and multiple Slaves is illustrated in Fig 1 above. Fig 2 illustrates a typical timing diagram of the SPI system. It also illustrates the different variants of the SPI that is resultant of an inverted clock polarity and shifted sense of clock phase from the SPI bus point of view.

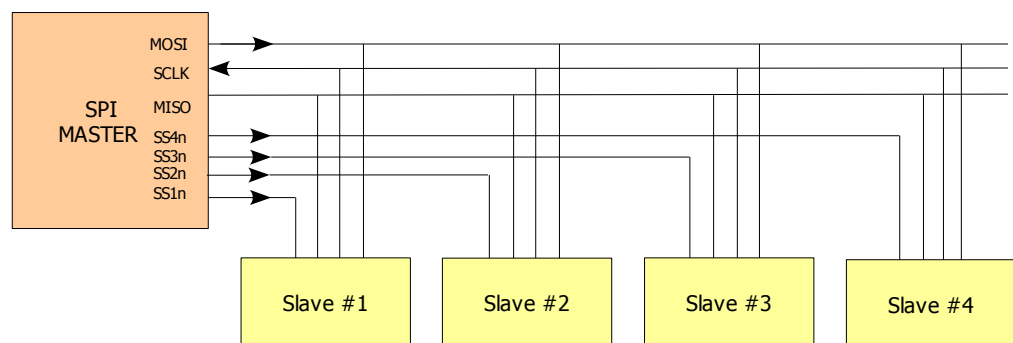


Fig 1: SPI bus system

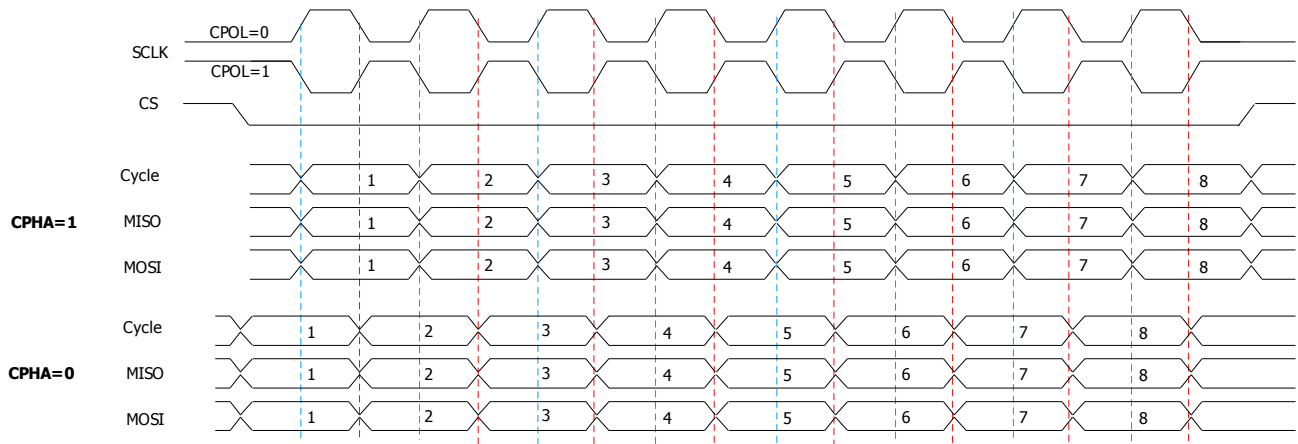


Fig 2: SPI variants and typical timing

Implementation

Features of SPI Master Design Example described in this application note: 1. Generates active low chip-select for up-to 4 Slaves. 2. Operates in CPOL=0 and CPHA=0 mode. 3. Each Data Transaction length is 16-bits 4. Read and Write FIFO of 16x16 size.

The Fig 3 below shows the overall interface of SPI master to both host processor and SPI slave device. The controller FSM block monitors host side transaction requests to the SPI master through CPU interface logic. Depending on the host processor request, appropriate control signals are sent to the controller FSM along with the address and data. The controller FSM is the heart of the SPI master which makes sure that correct transactions take place in sequence and that the SPI devices timings are not violated in the process.

sclk is the SPI bus clock, derived from **clk_sys** (system clock). Master's data on **mosi** is valid during the first rising edge of **sclk**. Similarly the slave posts its data when **sclk** is low so that the Master can sample **miso** during the subsequent rising edge of **sclk** (CPOL=0 and CPHA=0 mode). The host processor selects the SPI master by holding the **cs_n** signal LOW. When **cs_n** is low, the address bus bits determine to which SPI register the host is pointing. The host processor reads a particular register when **oe_n** is LOW and writes to a particular register when **we_n** is LOW.

SPI Master generates an interrupt at the end of each data transaction. Also, reading status register at any moment gives SPI Master busy/idle status, Read and Write FIFO empty/full conditions. Data, Status and Configuration Register Read latency is 3 cycles. Fig 5, 6, and 7 illustrates timing diagram of SPI Master Read/Write Cycles.

Table 1 lists the SPI Master registers and the address mapping.

Table 3 lists the configuration register bit mapping.

- INTREN Enables Interrupt Generation and INTRCLR clears interrupt.
- CLKDIV{3,2,1,0} bits configures **sclk** frequency. Table 2 lists relationship between **clk_sys** frequency and **sclk**.
- START bit denotes SPI read/write transaction start. Must be set to HIGH after configuration to start data transaction.
- FIFORST clears the FIFO when HIGH.
- LEN{3,2,1,0} represents block length of data transaction. Example: "0000" represents block length 1, "1111" represents block length 16.
- ADDR{1,0} denotes slave select address. Example: when "00", **s_select(0)** goes low and when "11", **s_select(3)** goes low.

Table 4 lists the status register bit mapping

- WFFULL, WFHFULL and WFEMPTY bits represent Write FIFO full, half full and empty
- RDFULL, RDHFULL and RDEEMPTY bits represent Read FIFO full, half full and empty
- BUSY bit HIGH indicates SPI Master is busy

Write-Only Mode: FIFORST bit of configuration register aids clearing the Read FIFO contents if a write data transaction samples unusable data on MISO line.

Read-Only Mode: SPI Master can be configured in Read-only Mode by keeping Write FIFO empty, during which MOSI line is pulled HIGH and SPI Master continues to sample data on MISO line.

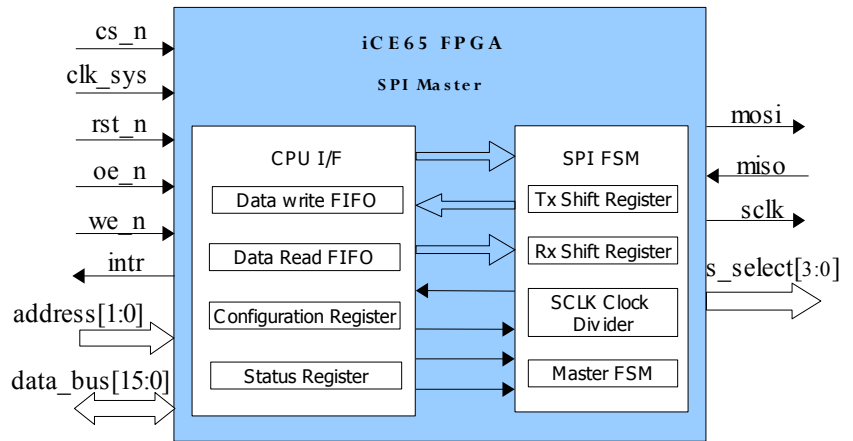


Fig 3: Block Diagram of SPI Master with Processor Interface

Address	Mode	Register Description
00	R/W	Configuration Register
01	W	Write Data FIFO(16x16)
10	R	Status Register
11	R	Read Data FIFO(16x16)

Table 1: SPI Master Registers

CLKDIV	SCLK Frequency
0000	clk_sys/4
0001	clk_sys/6
0010	clk_sys/8
...	
1111	clk_sys/34

Table 2: SPI SCLK Frequency

C14-15	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
X	INTRCLR	INTREN	CLKDIV3	CLKDIV2	CLKDIV1	CLKDIV0	START	FIFORST	LEN3	LEN2	LEN1	LEN0	ADDR1	ADDR0

Table 3: Configuration Register

S7-15	S6	S5	S4	S3	S2	S1	S0
X	WFFULL	WFHFULL	WFEMPTY	RDFULL	RDHFULL	RDEEMPTY	BUSY

Table 4: Status Register

Table 5 below shows the post P&R resource utilization summary of this design when implemented on an iCE65 FPGA. Table 6 lists the Pin Description and CPU/SPI Slave Interface details of SPI Master Design

Device	Logic Cells	IO Cells
iCE65L04-UCB284	400	31

Table 5: Resource Utilization

Processor Interface Signals						
Pin Name	Direction	Description	I/O Standard	I/O Bank	Pin Location	Comments
clk_sys	IN	System Clock	SB_LVCMOS(2.5)	Bank 2	V11	Global Clock input
cs_n	IN	Active low chip select	SB_LVCMOS(2.5)	Bank 2	AB2	
rst_n	IN	Active low reset	SB_LVCMOS(2.5)	Bank 2	V6	
oe_n	IN	Active low Read Enable	SB_LVCMOS(2.5)	Bank 2	T7	
we_n	IN	Active Low Write Enable	SB_LVCMOS(2.5)	Bank 2	R8	
data_bus15	INOUT	Data Bus[15]	SB_LVCMOS(2.5)	Bank 2	V7	
data_bus14	INOUT	Data Bus[14]	SB_LVCMOS(2.5)	Bank 2	T8	
data_bus13	INOUT	Data Bus[13]	SB_LVCMOS(2.5)	Bank 2	R9	
data_bus12	INOUT	Data Bus[12]	SB_LVCMOS(2.5)	Bank 2	V8	
data_bus11	INOUT	Data Bus[11]	SB_LVCMOS(2.5)	Bank 2	R10	
data_bus10	INOUT	Data Bus[10]	SB_LVCMOS(2.5)	Bank 2	V9	
data_bus9	INOUT	Data Bus[9]	SB_LVCMOS(2.5)	Bank 2	T10	
data_bus8	INOUT	Data Bus[8]	SB_LVCMOS(2.5)	Bank 2	Y4	
data_bus7	INOUT	Data Bus[7]	SB_LVCMOS(2.5)	Bank 2	Y5	
data_bus6	INOUT	Data Bus[6]	SB_LVCMOS(2.5)	Bank 2	AB6	
data_bus5	INOUT	Data Bus[5]	SB_LVCMOS(2.5)	Bank 2	AB7	
data_bus4	INOUT	Data Bus[4]	SB_LVCMOS(2.5)	Bank 2	AB8	
data_bus3	INOUT	Data Bus[3]	SB_LVCMOS(2.5)	Bank 2	AB9	
data_bus2	INOUT	Data Bus[2]	SB_LVCMOS(2.5)	Bank 2	Y6	
data_bus1	INOUT	Data Bus[1]	SB_LVCMOS(2.5)	Bank 2	Y7	
data_bus0	INOUT	Data Bus[0]	SB_LVCMOS(2.5)	Bank 2	Y9	
address1	IN	Address Bus[1]	SB_LVCMOS(2.5)	Bank 2	Y10	
address0	IN	Address Bus[0]	SB_LVCMOS(2.5)	Bank 2	AB10	
intr	OUT	Interrupt, Active High	SB_LVCMOS(2.5)	Bank 2	AB15	
SPI Slave Interface Signals						
s_select3	IN	Slave Select Line[3]	SB_LVCMOS(2.5)	Bank 2	AB11	
s_select2	IN	Slave Select Line[2]	SB_LVCMOS(2.5)	Bank 2	AB12	
s_select1	IN	Slave Select Line[1]	SB_LVCMOS(2.5)	Bank 2	AB13	
s_select0	IN	Slave Select Line[0]	SB_LVCMOS(2.5)	Bank 2	AB14	
MOSI	OUT	Master Out Slave In	SB_LVCMOS(2.5)	Bank 2	V13	Recommended Pull up
MISO	IN	Master In Slave Out	SB_LVCMOS(2.5)	Bank 2	T11	Recommended Pull up
SCLK	OUT	SPI Clock	SB_LVCMOS(2.5)	Bank 2	R11	Recommended Pull up

Table 6: Pin Description and sample mapping to iCE65 I/O pins(for LVCMOS 2.5 logic)

Conclusion

This design example demonstrates the implementation of a SPI Master using iCE FPGAs. The low pin count properties of the 4 wire SPI serial interface, and its high throughput characteristics due to full duplex communication are well complemented by iCE FPGA's very low power capabilities. This makes iCE FPGAs an obvious choice for SPI Master interface in all low power applications.

Disclaimer

SiliconBlue is providing this document, design example, or information "as is". SiliconBlue does not ensure that this implementation or information is void of any claims of infringement. As a reader / implementer, you are responsible for obtaining any rights you may require for your implementation. SiliconBlue also does not warranty the fitness of this design example to be readily implemented in any specific context. It is your sole responsibility to validate this design example and its correctness while implementing it for your requirement.

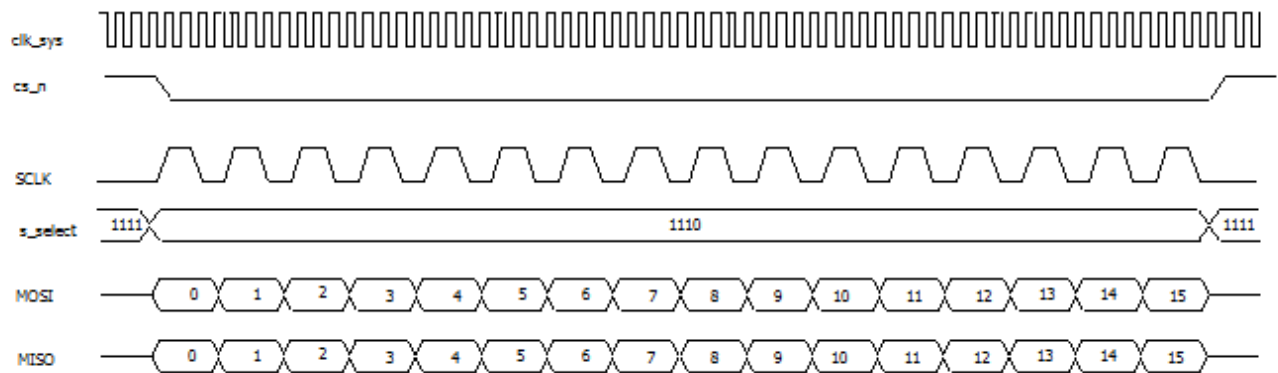


Fig 4. Timing Diagram: Read and Write on MOSI and MISO

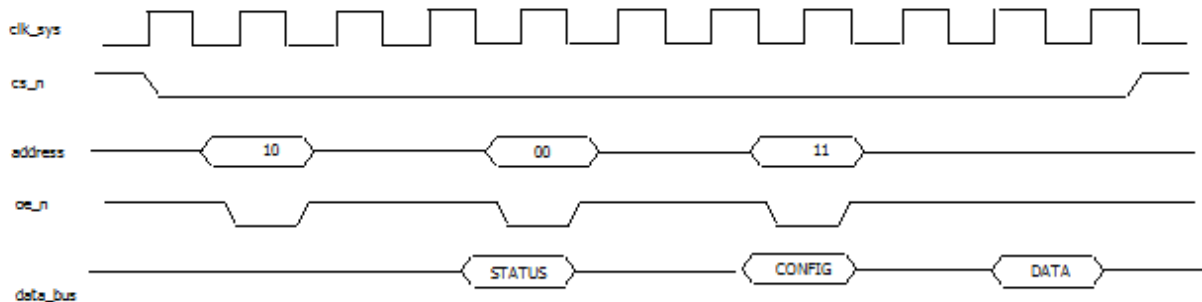


Fig 5. Timing Diagram: Read status, configuration and data

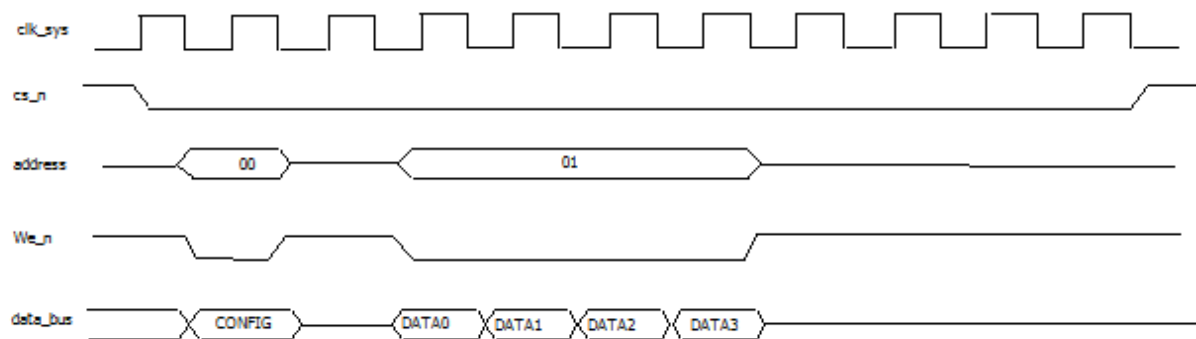


Fig 6. Timing Diagram : Write configuration and data

