

Design and Implementation of a High Speed Serial Peripheral Interface

Anand N, George Joseph, Suwin Sam Oommen, and R Dhanabal
School of Electronics Engineering, VIT University, Vellore, India.

Abstract—Serial Peripheral Interface is a synchronous protocol that allows serial communication between a master and a slave device. The purpose of this paper is to provide a full description of a high speed SPI Master/Slave implementation. The designs are based on Motorola's SPI Block Guide V03.06. This paper discusses design approaches that can offer prospective ways of controlling SPI-bus, incorporating the flexibility of handling two slaves at a time. Starting from the initial specifications till the final physical design, the design phases are systematically elaborated. The whole design is implemented in Verilog 2001, and mapped onto Xilinx's Virtex 5 FPGA devices.

Keywords— Serial peripheral interface (SPI), master & slave, serial data transfer, serial communication

I. Introduction

Serial peripheral interface (SPI) is one of the most commonly used serial protocols for both inter-chip and intra-chip low/medium speed data-stream transfers. It is used to communicate between a microcontroller and other devices like external EEPROMs, DACs, ADCs, etc. [1].

In the world of communication protocols, SPI is often considered as "little" communication protocol. It is important not to forget the purpose of each protocol. Ethernet, USB, and SATA are meant for "outside the box communications" and data exchanges between whole systems while SPI is aptly suited for communication between integrated circuits for low/medium data transfer speed with on-board peripherals [2], [3].

In SPI the data exchange takes place between the master and the slave device [8]. The master device provides a clock signal to attain synchronization. The instance of when the data can change and when the data is valid for reading is controlled by the clock signal. The clock line is controlled by the master device. Data transfer happens only when there is a clock manipulation.

During the case of a device transmitting a data, the incoming data must be read before an attempt to transmit again. An exchange of data always takes place between the devices. In SPI protocol, a device cannot be just a transmitter only device or a receiver only device [9]. The master device

controls the clock line SCK and the data exchange between the devices are controlled by SCK clock line.

II. SPI Protocol Overview

The working of the SPI module is essentially based on the contents of an eight bit serial shift register present in both the Master and the Slave. The transmissions take place based on the clock signal which is generated by the master. The Master, when it wants to send a byte of data to the Slave, places the byte in its shift register and similarly, the Slave can place the content in its shift register. As eight clock pulses are generated, the bits contained in the Master's shift register are transferred by means of the MOSI line to the Slave's shift register and the slave transfers its shift register content by means of the MISO signal line back to the master. So the contents of the two shift registers get exchanged. SPI uses the following signals for transmissions across its interface [4]:

SS: This stands for Slave Select. When it goes high, the corresponding slave device will be selected. The slave select line is used by master device to select which slave to initiate communication with the master [5].

SCK: This stands for serial clock. This signal synchronises the transmissions taking place across the bus.

MOSI: It is serial single bit data line, which the SPI master generates based on internally shifted value of the master data register.

MISO: It is serial single bit data line, by which the SPI slave communicates with the master. It sends out the serially shifted out bits from the slave data register.

The functionality of each of the sub-block is briefly outlined as follows:

Data Register and eight bit shift register: It forms the main part of the SPI system. When a SPI transfer takes place, a bit of data gets shifted out of the SPI master's data register and subsequently the serial data which comes from the slave's data register is serially shifted into the master's data register. Consequently, by the time one SPI transmission completes, i.e. after sixteen clock cycles, the contents of the master and slave will have been exchanged [5].

Control Register: This is the register which provides the user preliminary control over the SPI operation. This register, when appropriately configured, can be used to control the data transfer of the SPI. This includes enabling the SPI, configuring the SPI in master or slave mode, setting the sampling of data, configuring clock polarity, etc. [6], [7], [10]. Based on clock polarity (cpol) and clock phase (cpha) values, four different modes of operation are supported in the SPI [4].

Baud rate register: Baud rate generation consist of series of divider stages. Eight bits in the SPI baud rate register determine the value with which the bus clock is divided. This provides the end user ample choices for baud rate generation with divisors ranging from 2 to 128. The baud rate generator is active only if the SPI is operating in the master mode.

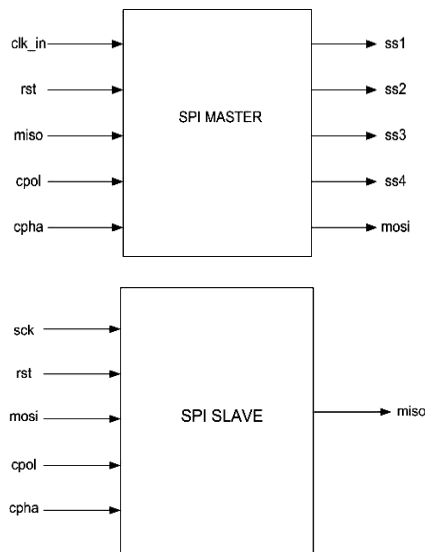


Fig. 1. SPI Master and SPI Slave Block Diagram

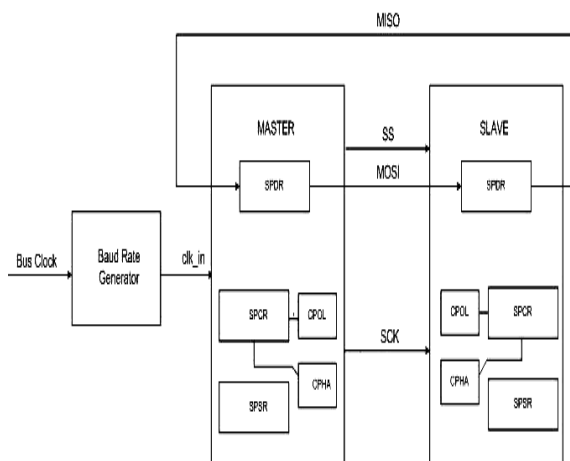


Fig. 2. SPI Block Architecture

III. Key Features of SPI Master and SPI Slave

SPI has low or medium (n MHz to 10n MHz) data transfer rate depending on implementation. SPI offers multiple transfer rates based on the SPI master baud rate, which can be programmed by the user. The SPI module supports multi slave operation. The master and slave can be transmitter or receiver based on its mode of operation. It is capable of receiving and transmitting on both rising and falling edges of the clock independently [1].

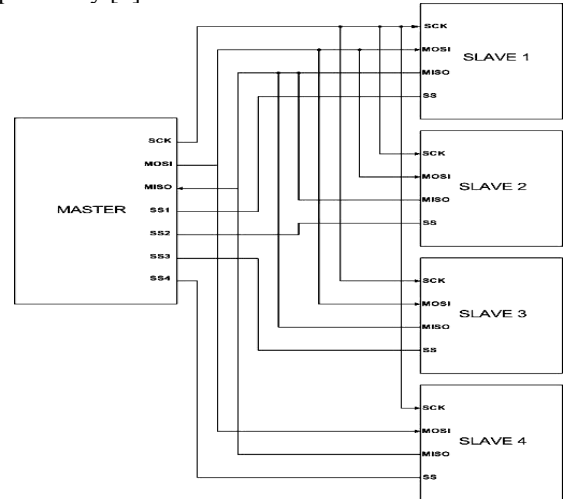


Fig. 3. Single master multi slave block diagram

IV. Implementation and Results

The control register can be programmed by the user to initiate the data transfer between the master and the slaves. By appropriately programming the control register, the SPI module can be made to operate in master mode or slave mode. The status register gives information on the current position of the data transfer operation. From the SPI status register, information on whether the data transfer has completed or not can be inferred.

The design of the SPI module is done in Verilog HDL and verified using ModelSim-Altera 6.5b, followed by logic synthesis in Xilinx ISE 13.2. Fig 4 depicts the simulation result showing the SPI interface signals from the master and slave. The design has been mapped to Xilinx's Virtex 5 xc5vlx30-3ff324 FPGA device and is shown in Fig 5. Using TSMC 180 nm technology library, the physical design is carried out using Cadence SoC Encounter. The final physical design view is as shown in Fig 6. The design when synthesized, gives information on the speed of the individual master and the slave modules. The results are shown below:

Timing Summary (master):

Speed Grade: -3

Minimum period: 1.101ns (Maximum Frequency: 908.348MHz)

Minimum input arrival time before clock: 1.722ns
Maximum output required time after clock: 2.779ns

Timing Summary (slave):

Speed Grade: -3
Minimum period: 0.631ns (Maximum Frequency: 1584.284MHz)
Minimum input arrival time before clock: 1.663ns
Maximum output required time after clock: 2.775ns



Fig. 4. Simulation results

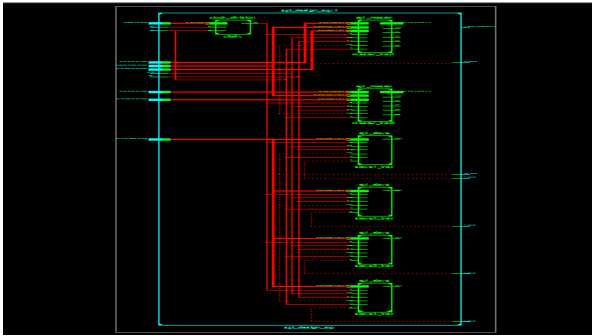


Fig. 5. Implemented SPI module

TABLE I
Timing Summary of SPI Master and Slave Modules

Module	Minimum period of operation(ns)	Minimum input arrival time before clock(ns)	Maximum output required time after clock(ns)
SPI Master	1.101	1.722	2.779
SPI Slave	0.631	1.663	2.775

TABLE II
Device Utilization Summary of the SPI Module

	Used	Available	Utilization
Number of Slices	31	6144	0.51%
Number of Slice Flip Flops	47	12288	0.38%
Number of 4 input LUTs	50	12288	0.41%
Number of bonded IOBs	26	240	10.83%
Number of GCLKs	2	32	6.25%

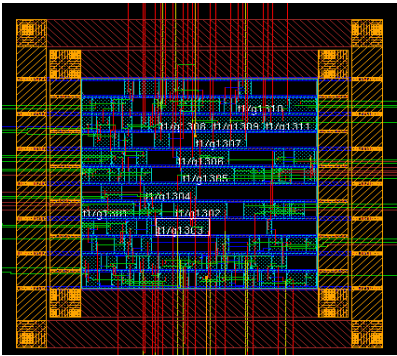


Fig. 6. Physical Design View

TABLE III
Comparison of Delays

Device targeted: Xilinx Virtex 5 xc5v1x30-3ff324	SPI master	SPI slave
	Clock to set up delay(ns)	
Existing design[1]	3.520	3.303
Our work	1.722	1.663

The timing summary from Xilinx shows that the implemented SPI module has better results in terms of speed of operation when compared with the existing design.

v. Conclusion

In this paper we proposed a high speed serial peripheral interface. The simulations have shown considerable improvement in speed when compared to the existing architecture. The proposed design also has multi slave handling capability with no performance degradation.

References

[1] A.K. Oudjida, M.L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui & Y.N. Alhoumays, "Design and Test of General-Purpose SPI Master/Slave IPs on OPB Bus," 2010 IEEE.

[2] Frederic Leens, "An introduction to SPI and I2C protocol," IEEE Instrumentation and Measurement magazine, February 2009.

[3] A.K. Oudjida et al, "FPGA Implementation of I2C and SPI Protocols: A Comparative Study," Proceedings of the 16th edition of the IEEE International Conference on Electronics Circuits and Systems ICECS, pp. 507 -510, December 13-16, 2009.

[4] Motorola Inc., "SPI Block Guide V03.06," March 2003.

[5] Altera, "SPI Controller, Hard Processor System (HPS) Technical Reference Manual (TRM) V1.2," November 2012.

[6] FreescaleSemiconductorInc., "MC68HC912D60 manual V 4.0," September 2010.

[7] Jianlong Zhang, Jiwei Wang, Chunyu Wu, Wenjing Zhang, "The design and realization of a comprehensive SPI interface controller," Second International Conference on Mechanic Automation and Control Engineering (MACE), 2011 IEEE.

[8] Lattice Semiconductor Inc., "SPI Slave Controller V 01.0," October 2012.

[9] Lattice Semiconductor Inc., "SPI Wishbone Controller V 01.5," January 2011.

[10] Greco M, Bussa M.P, Ferrero L, Maggiora M, Verna A, "VHDL Implementation of a SPI controller for PANDA digital signal processing," Nuclear Science Symposium and Medical Imaging Conference,2011 IEEE