

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.

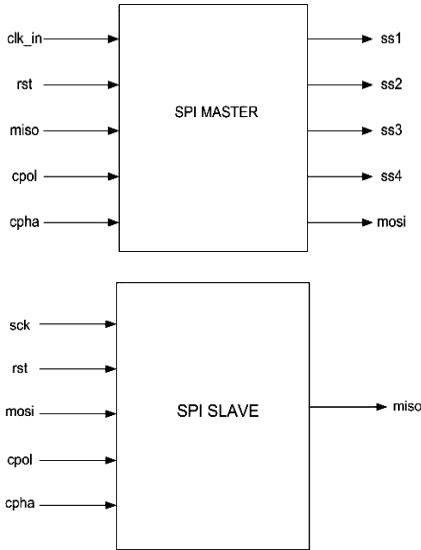


Fig. 1. SPI Master and SPI Slave Block Diagram

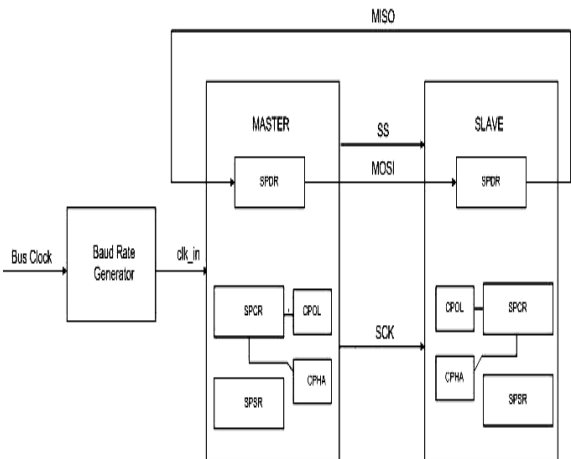


Fig. 2. SPI Block Architecture

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].

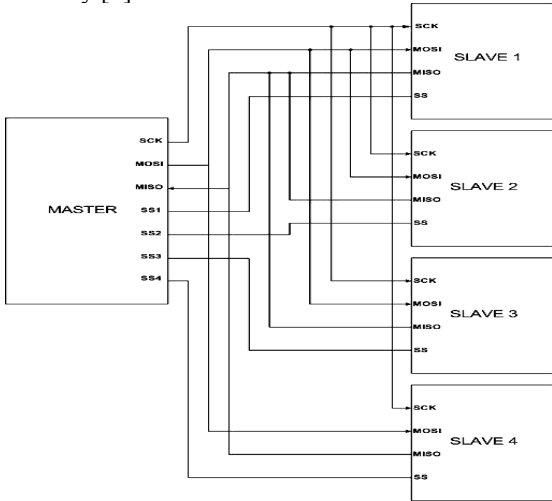


Fig. 3. Single master multi slave block diagram

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)

