SPI Core for Xilinx S3E/A/AN Starter Kit DAC, AMP, & ADC controllers

William Gibb

SPI Controller Core for the Linear Technology Pre-Amplifier, ADC and DAC onboard the Xilinx Spartan 3E/3A/3AN Starter Kits. Revision 0.7

Opencores.org

William Gibb

williamgibb@opencores.org

12/7/2009

Revision History

Rev	Date	Author	Description
0.1	June 13th, 2002	Simon Srot	First Draft
0.2	July 12th, 2002	Simon Srot	Document is lectured
0.3	December 28th, 2002	Simon Srot	Support for 64 bit character len added.
0.4	March 26th, 2003	Simon Srot	Automatic slave select signal generation added.
0.5	April 15th, 2003	Simon Srot	Support for 128 bit character len added.
0.6	March 15th, 2004	Simon Srot	Bit fields in CTRL changed.
0.7	December 7th, 2009	William Gibb	Forked core for specific use with S3E/A/AN Dev boards.
0.7	December 7th, 2009	William Gibb	Bit fields in CTRL changed.
0.7	December 7th, 2009	William Gibb	Updated Documentation

Contents

Figures

Tables

Acknowledgement

I'd like to thank Simon Strot (simons@opencores.org) for providing the SPI Master core, which this core and documentation is directly based off of. This is a fork of the Rev 0.6 version of the SPI Master Core project, hosted on Opencores.org.

Introduction

This document provides specifications for the SPI (Serial Peripheral Interface) Master core, modified for use with the ADC and DAC circuitry on the Spartan 3E/3A/3AN starter kits. Synchronous serial interfaces are widely used to provide economical board-level interfaces between different devices such as microcontrollers, DACs, ADCs and other. Although there is no single standard for a synchronous serial bus, there are industry-wide accepted guidelines based on two most popular implementations:

- SPI (a trademark of Motorola Semiconductor)
- Microwire/Plus (a trademark of National Semiconductor)

Many IC manufacturers produce components that are compatible with SPI and Microwire/Plus.

The modified SPI Master core is compatible with the following Linear Technology devices:

- LTC6912-1 Dual-Amp
- LTC1407A-1 Dual A/D
- LT2624 Quad Amp

At the hosts side, the WISHBONE compliant interface has been stripped, to allow for raw access to the SPI core.

Features:

- **Full duplex synchronous serial data transfer**
  - **Independent RX/TX channel control.**
- **MSB or LSB first data transfer. Fixed LSB for Rx.**
- **Rx and Tx on both rising or falling edge of serial clock independently.**
  - **Control words provided for use with Linear Tech devices.**
- **Fully static synchronous design with one clock domain**
- **Technology independent Verilog.**
  - **Implementation notes are board specific.**
- **Fully synthesizable.**
  - **145 Slices in Spartan 3 fabric**

IO Ports

Host Interface

Port	Width	Direction	Description
clk	1	Input	System Clock
rst	1	Input	Positive Edge, Asynchronous Reset
ampDAC	1	Input	Pre amplifier / DAC select. Asserted to select DAC for TX.
data_in	24	Input	Input data bus.
load_ctrl	1	Input	CE for the control register.
load_div	1	Input	CE for the divider register.
go	1	Output	Output indicating the core is active in a transfer
chanA	14	Output	Channel A from the Adc.
chanB	14	Output	Channel B from the Adc.
adcValid	1	Output	Signal asserting the Adc channel data is valid.

Table 1: Host Interface Signals

All outputs to the host interface, including the chanA/chanB signals, are registered. The data on chanA/B is valid after adcValid is asserted, until the next receipt (RX) of data begins.

SPI External Connections

Port	Width	Direction	Description
ss_pad_o	2	Output	Slave select output signals. Preamp select is bit 1, DAC select is bit 0.
sclk_pad_o	1	Output	Serial clock output.
mosi_pad_o	1	Output	Master out slave in data signal output
conv	1	Output	Conv signal for ADC
miso_pad_i	1	Input	Master in slave out data signal input

Table 2: SPI External Connections

Registers

The modified SPI Core contains two registers directly accessible from the host, and a third register which can be made accessible via the control register.

Control Register (CTRL)

- Bit Width: `SPI_CTRL_BIT_NB. Default is 14.
- Access: Write while !Tip.
- Name: ctrl
- Reset value: `SPI_CTRL_BIT_NB'b0

The control register is 14 bits wide, and controls the operation of the SPI Core. Each bit of the register is assigned a particular function.

Bit #	13	12	11	10	9	8	7	6 to 0
Name	TXC	SAMPLE	LSB	TX_NEGEDGE	RX_NEGEDGE	GO	WRITE	LEN

Table 3: Control Register

TXC

This signal enables writing data to the TX register. This signal acts as a clock enable (signal:capture) to the data register in the TX unit. It is automatically cleared after 1 clock cycle. Data to be transmitted should be placed on the

SAMPLE

Writing 1 to this bit enables the RX when GO is asserted. This enables reads from the ADC to be independent from writes to the Pre-Amplifier and DAC.

LSB

If this bit is set, the LSB is sent first on the line (bit data[0]). If this bit is cleared, the MSB is transmitted first (which bit in data register that is depends on the CHAR_LEN field in the CTRL register). This bit does not affect the RX, since that is fixed.

TX_NEGEDGE

If this bit is set, the mosi_pad_o signal is changed on the falling edge of a sclk_pad_o clock signal, or otherwise the mosi_pad_o signal is changed on the rising edge of sclk_pad_o.

RX_NEGEDGE

If this bit is set, the miso_pad_i signal is latched on the falling edge of a sclk_pad_o clock signal, or otherwise the miso_pad_i signal is latched on the rising edge of sclk_pad_o.

GO

Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after the transfer finished. Writing 0 to this bit has no effect.

NOTE: All registers, including the CTRL register, should be set before writing 1 to the GO_BSY bit in the CTRL register. The configuration in the CTRL register must be changed with the GO_BSY bit cleared, i.e. two Writes to the CTRL register must be executed when changing the configuration and performing the next transfer, firstly with the GO_BSY bit cleared and secondly with GO_BSY bit set to start the transfer. When a transfer is in progress, writing to any register of the SPI Master core has no effect.

WRITE

Writing 1 to this bit enables the TX when GO is asserted. This enables writes to the Pre-Amplifier and DAC to be independent from reads from the ADC.

LEN

This field specifies how many bits are transmitted in one transfer. Up to 64 bits can be transmitted. Bear in mind that the default size of the TX register is 24 bits. This has no effect on the number of bits received.

CHAR_LEN = 0x01 … 1 bit

CHAR_LEN = 0x02 … 2 bits

…

CHAR_LEN = 0x7f … 127 bits

CHAR_LEN = 0x00 … 128 bits

Control Words

A set of example control words are provided, in Verilog. These match parameters used when interfacing with the ADC/DAC chips of the S3E/A/AN starter kits.

```
      parameter CTRL_PREP      =     14'h0E18; //TXC=0, SAMPLE = 0,| LSB=1, TXN=1, RXN=1, GO=0,|
WR=0, LEN=24bits
     parameter CTRL_TXC        =     14'h2E18; //TXC=1, SAMPLE = 0,| LSB=1, TXN=1, RXN=1, GO=0,|
WR=0, LEN=24bits
      parameter CTRL_GOSAMPLE =     14'h1F18; //TXC=0, SAMPLE = 1,| LSB=1, TXN=1, RXN=1, GO=1,|
WR=0, LEN=24bits
      parameter CTRL_GOWRITE  =     14'h0F98; //TXC=0, SAMPLE = 0,| LSB=1, TXN=1, RXN=1, GO=1,|
WR=1, LEN=24bits
      parameter CTRL_GOALL    =     14'h1F98; //TXC=0, SAMPLE = 1,| LSB=1, TXN=1, RXN=1, GO=1,|
WR=1, LEN=24bits
```

Data TX Register

- Bit Width: 24.
- Access: Write while !Tip && capture bit of CTRL is asserted.
- Name: data
- Reset value: 24'b0

This register holds data that is to be shifted out to the preamplifier or DAC. It is written to through d_datain, and is enabled by writing TXC=1 on the control register.

Divider

- Bit Width: `SPI_DIVIDER_LENGTH. Default is 5.
- Access: Write while !Tip, by asserting load_div as a CE.
- Name: divider
- Reset value: `SPI_DIVIDER_LENGTH'b0

The value in this field is the frequency divider of the system clock clk to generate the serial clock on the output sclk_pad_o. The desired frequency is obtained according to the following equation:

F_sclk=F_clk/((Divider+1)*2)

Operation

Due to the nature of the modifications done to the original SPI Master Core, this SPI Core can no longer be considered a general purpose SPI compliant core. For the purposes of communicating with the Linear Technology ICs onboard with the onboard SPI bus, it will work fine; however the core does not have the capability to talk to other SPI devices on the same spi_mosi. A different core would need to be used to communicate with those devices as well.

Host Interface

The core is controlled through the host interface and the Control Register. There are a few steps required to use the ADC/DAC convertors.

1. If the ADC is going to be used, the programmable preamplifier must have a gain assigned to each channel to turn it on. See Pre-amplifier data sheet for more information about configuring it.
2. If the DAC is going to be used, the data must be written to the dac along with the appropriate DAC command and dac selection. See the DAC datasheet to for more information about configuring it.
3. If the ADC is going to be used, there is no special action needed.

The core can easily be driven by a finite state machine. An example sequence of operations is as follows:

1. Load_Div = 1, data_in=divider value
2. Load_Ctrl=1, data_in=CTRL_TXC, ampDAC=0;
3. Data_in=amplifier_gain;
4. Load_Ctrl=1, data_in=CTRL_GOWRITE, ampDAC=0;
    a. Wait for go signal to go low
5. ampDAC=1; now you can selectively read or write to the DAC/ADC ICs.

When a sample is captured by the SPI core from the ADC, the data is available on two 14 bit ports, chanA and chanB. There is a signal, adcValid, which goes high for 1 clock cycle after the transmission has been completed; this signal indicates the output is ready to be read. These outputs are registered, and will not change until the next sample capture has begun.

SPI Interface

The serial interface consists of slave select lines, serial clock lines, as well as input and output data lines. All transfers are either full or half duplex transfers of a fixed number of bix (RX) or programmable number of bits(TX). In addition to the slave select lines for the chips in which have simple chip selects, the core also provides the timing for the conv signal used as a chip select / sample

trigger for the ADC. See the section Core Configuration to find out more about the timing of the conv signal.

Compared to the SPI/Microwire protocol, this core has some additional functionality. It can drive data to the output data line in respect to the falling (SPI/Microwire compliant) or rising edge of the serial clock, and it can latch data on an input data line on the rising (SPI/Microwire compliant) or falling edge of a serial clock line. It also can transmit (receive) the MSB first (SPI/Microwire compliant) or the LSB first. The RX and TX registers do not share the same flip-flops, which means that what is received from the input data line in one transfer will not affect what is transmitted on the output data line in the next transfer if no write access to the TX register is executed between the transfers. If no additional data is written to the TX register, additional writes will continue to send the same data.

Architecture

The SPI Core contains four modules/files. The hierarchy of signals and modules can be seen in Figure 1.

- Spi_top.v : This instances all of the other modules and contains the control register and associated logic.
- Spi_clgen.v : This contains the clock divider/generator circuitry for the SPI Clock.
- Spi_shift_out.v : This contains the tx circuitry for shifting data out.
- Spi_shift_in.v : This contains the rx circuitry for shifting data in.

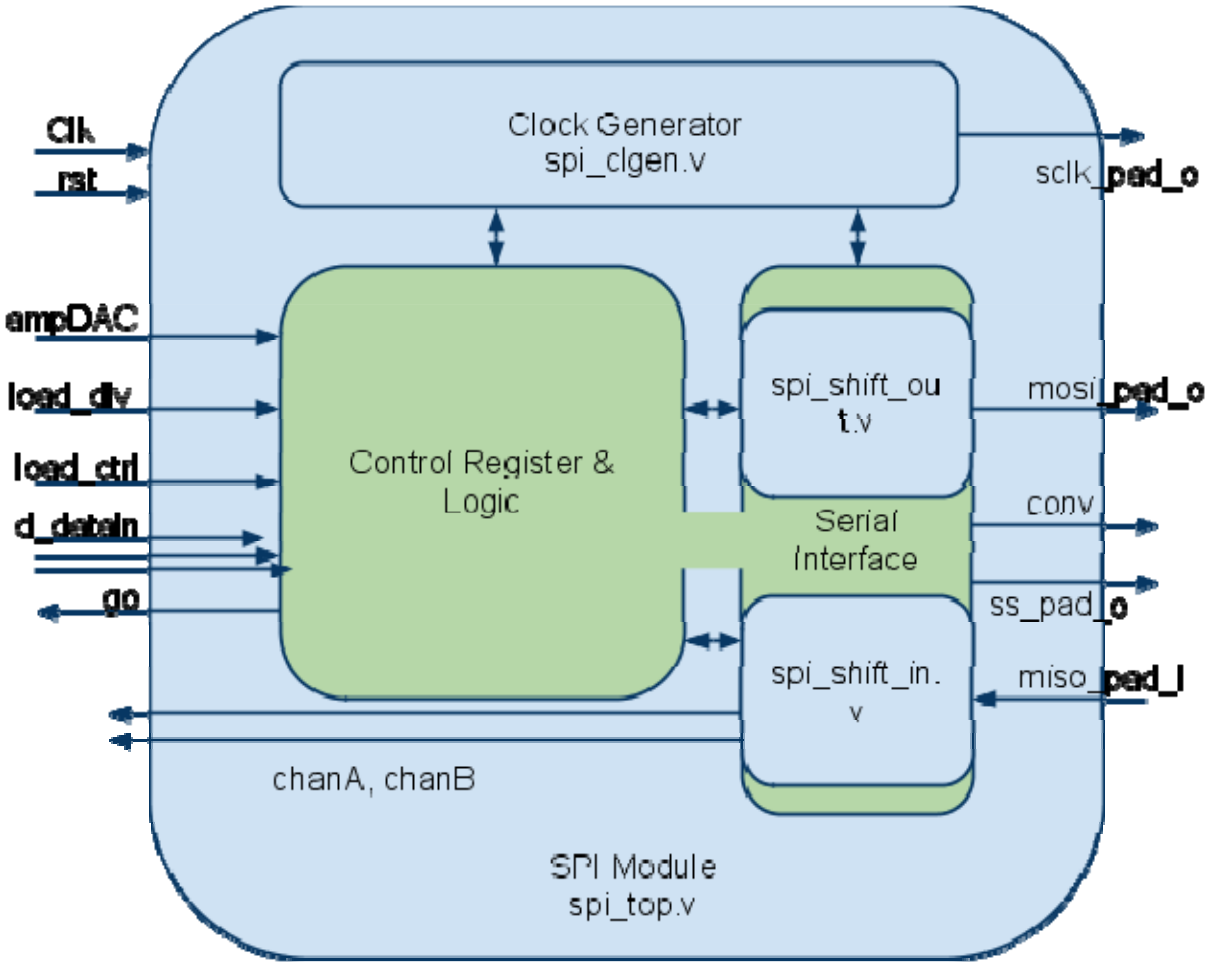

Figure 1 SPI Module block diagram

Design Notes on the Operation of the spi_shift_out.v module

Inputs:

- go - enables tip to go high ie start transmitting data

- capture - this latches the datain when enabled. behavior: clockenable
- {pos,neg}_edge - control the flow of data out of the device while tip is asserted
- tx_negedge - indicates whether to send the data on the posedge or negedge of the clk

outputs:

- tip - indicates that a transfer is in progress. triggered by go.
- last - indicates transmitting last bit. valid the whole time of the last spi clk cycle.
- it going high indcates the transfer is complete.
- sout - serial output data

internal notes

- data is transmitted on the clk edge indicated by tx_negedge.
- assign tx_clk = (tx_negedge ? neg_edge : pos_edge) && !last;
- when it is asserted, it moves neg_edge and pos_edge to the clock
- not !last prevents it from changing during transmission of the last bit

Design Notes on the Operation of spi_clgen.v

- Uses enable from the spi_top module to turn itself on.
- Use a mux in spi_top to control which module, spi_rx or spi_tx, gets to feed to the last input.

Design Notes on the Operation of the spi_shift_in,v module

Inputs:

- go - enables tip to go high ie start receivinging data
- capture - this latches the datain when enabled. behavior: clockenable
- {pos,neg}_edge - control the flow of data into the device while tip is asserted
- rx_negedge - indicates whether to capture the data on the posedge or negedge of the clk

outputs:

- tip - indicates that a transfer is in progress. triggered by go.
- last - indicates transmitting last bit. valid the whole time of the last spi clk cycle. it going high indcates the receipt of data is complete.
- p_out - serial output data. no associated data valid signal from this module. Valid signal generated by top level module.

internal notes

- data is received on the clk edge indicated by rx_negedge.
- assign rx_clk = (rx_negedge ? neg_edge : pos_edge) && (!last || s_clk);
- when it is asserted, it moves neg_edge and pos_edge to the clock
- not !last prevents it from changing while recieving the last bit

Core Configuration

The timing of the conv signal is determined based off of how many system clock cycles after go_rx is asserted. These are currently parameterized in spi_top.v. Set the value of CONVCOUNT to be the number of cycles after go_rx is asserted that conv goes high. Set the value of MAXCOUNT to be the number of cycles after go_rx is asserted that conf is forced low. You may need to change these values if your changing the spi clock frequency. Be sure to the check the datasheet of the ADC for more information about the timing requirements of the CONV signal.

```
parameter MAXCOUNT = 24;

parameter CONVCOUNT = 12;
```

To meet specific system requirements and size constraints on behalf of the core functionality, the SPI Master core can be configured by setting the appropriate define directives in the spi_defines.v source file. The directives are as follows:

SPI_DIVIDER_BIT_NB

This parameter defines the maximum number of bits needed for the divider. Set this parameter accordingly to the maximum system frequency and lowest serial clock frequency:

SPI_DIVIDER_BIT_NB = log(base2) [(f_systemMax/(2*f_sclk_min)-1].

Default value is 8.

Many of the other parameters may be modified, but are of little use given then purpose of this modified core is to provide a raw, low level interface to the ADC/DAC/AMP chips on the starter kits.

Testing and Simulation

Models of the digital interfaces of the LT ADC, DAC and AMP have written, and are provided along with the core, with a test bench included. These can be used to test the SPI Core.

Resources

Original SPI Master Core project http://opencores.org/project,spi

Spartan 3AN Starter Kit http://www.xilinx.com/products/devkits/HW-SPAR3AN-SK-UNI-G.htm

Spartan 3E Starter Kit http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-US-G.htm