OpenCores.Org

spiMaster IP Core Specification

Author: Steve Fielding

sfielding@base2designs.com

Rev. 1.1

November 5, 2008

Revision History

Rev.	Date	Author	Description
1.0	4/08/08	Sfielding	Created
1.1	5/13/08	Sfielding	Added missing registers

Contents

1 Introduction

spiMaster is a SPI (Serial Peripheral Interface) IP core, operating as a SPI master. It can support basic SPI bus accesses, and SD/MMC memory cards

- Full SD/MMC memory card support, including card initialization, block read, and block write.
- Basic SPI bus access.
- 512 byte receive and transmit Fifos.
- 8-bit slave Wishbone interface.
- Separate clocks for Wishbone interface and SPI core logic.
- SPI clock frequency configurable via bus interface.
- Data transfer at speeds close to SD/MMC card maximum rate.

2

Architecture

WISHBONE
COMPATIBLE

3

Operation

These are the steps required to initialize SD/MMC memory card, perform a block write, followed by a block read.

Initialize

Set SPI_TRANS_TYPE_REG = SPI_INIT_SD

Set SPI_TRANS_CTRL_REG = SPI_TRANS_START

Wait for SPI_TRANS_STS_REG != TRANS_BUSY

Check for SPI_TRANS_ERROR_REG [1:0] == INIT_NO_ERROR

Block Write

Write 512 bytes to SPI_TX_FIFO_DATA_REG

Set the SD block address registers:

SD_ADDR_7_0_REG

SD_ADDR_15_8_REG

SD_ADDR_23_16_REG

SD_ADDR_31_24_REG

Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK

Set SPI_TRANS_CTRL_REG = SPI_TRANS_START

Wait for SPI_TRANS_STS_REG != TRANS_BUSY

Check for SPI_TRANS_ERROR_REG[5:4] == WRITE_NO_ERROR

Block Read

Set the SD block address registers:

SD_ADDR_7_0_REG

SD_ADDR_15_8_REG

SD_ADDR_23_16_REG

SD_ADDR_31_24_REG

Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK

Set SPI_TRANS_CTRL_REG = SPI_TRANS_START

Wait for SPI_TRANS_STS_REG != TRANS_BUSY

Check for SPI_TRANS_ERROR_REG[3:2] == READ_NO_ERROR

Read 512 bytes from SPI_RX_FIFO_DATA_REG

4

Registers

Register Address	Name
0x0	SPI_MASTER_VERSION_REG
0x1	SPI_MASTER_CONTROL_REG
0x2	TRANS_TYPE_REG
0x3	TRANS_CTRL_REG
0x4	TRANS_STS_REG
0x5	TRANS_ERROR_REG
0x6	DIRECT_ACCESS_DATA_REG
0x7	SD_ADDR_7_0_REG
0x8	SD_ADDR_15_8_REG
0x9	SD_ADDR_23_16_REG
0xa	SD_ADDR_31_24_REG
0xb	SPI_CLK_DEL_REG
0x10	RX_FIFO_DATA_REG
0x12	RX_FIFO_DATA_COUNT_MSB
0x13	RX_FIFO_DATA_COUNT_LSB
0x14	RX_FIFO_CONTROL_REG
0x20	TX_FIFO_DATA_REG
0x24	TX_FIFO_CONTROL_REG

SPI_MASTER_VERSION_REG

Bit Position	Name	Description
[7:4]	VERSION_NUM_MAJOR	Major revision number
[3:0]	VERSION_NUM_MINOR	Minor revision number

SPI_MASTER_CONTROL_REG

Bit Position	Name	Description	Default	R/W
0	RST	1 = Reset core logic, and registers. Self clearing	0	W

TRANS_TYPE_REG

Bit Position	Name	Description	Default	R/W
[1:0]	TRANS_TYPE	Sets the transaction type, where; 0 = DIRECT_ACCESS 1 = INIT_SD 2 = RW_READ_SD_BLOCK 3 = RW_WRITE_SD_BLOCK	0	R/W

TRANS_CTRL_REG

Bit Position	Name	Description	Default	R/W
0	TRANS_START	1 = Start transaction. Self clearing	0	W

TRANS_STS_REG

Bit Position	Name	Description	Default	R/W
0	TRANS_BUSY	1 = Transaction busy		R

TRANS_ERROR_REG

Bit Position	Name	Description	Default	R/W
[5:4]	SD_WRITE_ERROR	0 = WRITE_NO_ERROR 1 = WRITE_CMD_ERROR 2 = WRITE_DATA_ERROR 3 = WRITE_BUSY_ERROR		R
[3:2]	SD_READ_ERROR	0 = READ_NO_ERROR 1 = READ_CMD_ERROR 2 = READ_TOKEN_ERROR		R

Bit Position	Name	Description	Default	R/W
[1:0]	SD_INIT_ERROR	0 = INIT_NO_ERROR 1 = INIT_CMD0_ERROR 2 = INIT_CMD1_ERROR		R

DIRECT_ACCESS_DATA_REG

Bit Position	Name	Description	Default	R/W
[7:0]	TX_DATA	Set TX_DATA prior to starting a DIRECT_ACCESS transaction. Note that the SPI bus has no concept of a read or write transaction. Thus every DIRECT_ACCESS transaction transmits data from the SPI master, and receives data from the SPI slave.	00	W
[7:0]	RX_DATA	Read RX_DATA after completing a DIRECT_ACCESS transaction		R

SD_ADDR_7_0_REG

Bit Position	Name	Description	Default	R/W
[7:0]	SD_ADDR_7_0	SD_ADDR[7:0]. Normally set to zero, because memory accesses should occur on a 512 byte boundary. Set the SD/MMC memory address before starting a block read or block write	00	R/W

SD_ADDR_15_8_REG

Bit Position	Name	Description	Default	R/W
[7:0]	SD_ADDR_15_8	SD_ADDR[15:8]. Normally set SD_ADDR[8] to zero, because memory accesses should occur on a 512 byte boundary	00	R/W

SD_ADDR_23_16_REG

Bit Position	Name	Description	Default	R/W
[7:0]	SD_ADDR_23_16	SD_ADDR[23:16]	00	R/W

SD_ADDR_31_24_REG

Bit Position	Name	Description	Default	R/W
[7:0]	SD_ADDR_31_24	SD_ADDR[31:24]	00	R/W

SPI_CLK_DEL_REG

Bit Position	Name	Description	Default	R/W
[7:0]	SPI_CLK_DEL	SPI_CLK_DEL controls the frequency of the SPI_CLK after SD initialization is completed. To set the clock frequency during SD initialization you will need to modify the constant SLOW_SPI_CLK in spiMaster_defines.v SPI_CLK_DEL = (spiSysClk / (SPI_CLK * 2)) – 1	00	R/W

RX_FIFO_DATA_REG

Bit Position	Name	Description	R/W
[7:0]	RX_FIFO_DATA	SD/MMC block read data. Note, fifo size matches the SD/MMC block size of 512 bytes.	R

RX_FIFO_DATA_COUNT_MSB

Bit Position	Name	Description	R/W
[7:0]	FIFO_DATA_COUNT_MSB	MSByte of FIFO_DATA_COUNT. Indicates the number of data entries within the fifo.	R

RX_FIFO_DATA_COUNT_LSB

Bit Position	Name	Description	R/W
[7:0]	FIFO_DATA_COUNT_LSB	LSByte of FIFO_DATA_COUNT. Indicates the number of data entries within the fifo.	R

RX_FIFO_CONTROL_REG

Bit Position	Name	Description	Default	R/W
0	FIFO_FORCE_EMPTY	1 = force fifo empty. Deletes all the data samples within the fifo. Self clearing.	0	W

TX_FIFO_DATA_REG

Bit Position	Name	Description	R/W
[7:0]	TX_FIFO_DATA	SD/MMC block write data. Fifo size matches the SD/MMC block size of 512 bytes.	W

TX_FIFO_CONTROL_REG

Bit Position	Name	Description	Default	R/W
0	FIFO_FORCE_EMPTY	1 = force fifo empty. Deletes all the data samples within the fifo. Self clearing.	0	W

5

Clocks

Name	Source	Rates (MHz)			Remarks	Description
		Max	Min	Res		
spiSysClk	Input Pad	-	-	-	Duty cycle 50/50.	SPI system clock.
clk_i	Input Pad	SpiSysClk * 5	spiSysClk		Duty cycle 50/50.	Wishbone bus clock.

Table 1: List of clocks

6

IO Ports

Port	Width	Direction	Description
spiSysClk	1	input	spi logic clock.
clk_i	1	input	WISHBONE clock input. Can be asynchronous to usbClk. spiSysClk <= clk_i <= spiSysClk * 5
rst_i	1	input	WISHBONE reset. Synchronous to clk_i. Resets all logic.
address_i	8	input	WISHBONE address input
data_i	8	input	WISHBONE data input
data_o	8	output	WISHBONE data output
writeEn	1	input	WISHBONE write enable
strobe_i	1	input	WISHBONE strobe input
ack_o	1	output	WISHBONE acknowledge output
spiClkOut	1	output	SPI clock. Clock speed configurable
spiDataIn	1	input	SPI serial data from slave
spiDataOut	1	input	SPI serial data to slave
spiCS_n	1	input	SPI device chip select

Table 2: List of IO ports

7

Wishbone Datasheet

<table>
<tr><th colspan="3"><em>WISHBONE DATASHEET</em>
<em>for SpiMaster IP Core</em></th></tr>
<tr><td>Description</td><td colspan="2">Specification</td></tr>
<tr><td>General Description:</td><td colspan="2">8-bit slave input and output port</td></tr>
<tr><td>Supported cycles:</td><td colspan="2">SLAVE READ/WRITE</td></tr>
<tr><td>Data port Size:</td><td colspan="2">8-bit</td></tr>
<tr><td>Data port granularity:</td><td colspan="2">8-bit</td></tr>
<tr><td>Data port, max operand size:</td><td colspan="2">8-bit</td></tr>
<tr><td>Data transfer ordering:</td><td colspan="2">N/A</td></tr>
<tr><td>Data transfer sequencing:</td><td colspan="2">Undefined</td></tr>
<tr><td rowspan="9">Supported signal list and cross reference to equivalet WISHBONE signals:</td><td>Signal Name</td><td>WISHBONE Equiv.</td></tr>
<tr><td>address_i</td><td>ADR_I</td></tr>
<tr><td>data_i[7:0]</td><td>DAT_I()</td></tr>
<tr><td>data_o[7:0]</td><td>DAT_O()</td></tr>
<tr><td>we_i</td><td>WE_I</td></tr>
<tr><td>strobe_i</td><td>STB_I</td></tr>
<tr><td>ack_o</td><td>ACK_O</td></tr>
<tr><td>clk_i</td><td>CLK_I</td></tr>
<tr><td>rst_i</td><td>RST_I</td></tr>
</table>

Table 3: WISHBONE data sheet

8

Resource Utilization

Design Entity	Logic Cells	Memory bytes
spiMaster (top level)	906	1024

Table 4 Resource utilization for Altera CycloneEP2C20