Specification OPB-SPI Slave

Daniel Koethe

November 15, 2007

Contents

1 Introduction

This document describe a SPI Slave core designed for the Xilinx EDK. [1]

1.1 Features

- OPB-Clock and SPI-Clock are complete independent
- SPI can run faster than OPB if guaranteed that no TX-FIFO Underrunn or RX-FIFO Overrunn occure.
- variable transfer length 2..32

1.2 Limitations

- designed only for Xilinx Spartan-3/Virtex-4 at the moment
- only Slave Operation

2 Core configuration

Description	Parameter Name	Allowable Values	Default Value
System Parameter			
Base address for OPB SPI	C_BASEADDR	0x00	0x00000000
High address for OPB SPI	C_HIGHADDR	BASEADDR+0x3F	BASEADDR+0x3f
OPB address bus width	C_OPB_AWIDTH	32	32
OPB data bus width	C_OPB_DWIDTH	32	32
Target FPGA Family	C_FAMILY	spartan3,virtex4	virtex4
User Parameter			
Shift register width	C_SR_WIDTH	8-32	8
Shift MSB First	C_MSB_FIRST	true, false	true
SPI Clock Polarity	C_CPOL	0,1	0
SPI Clock Phase	C_CPHA	0,1	0
FIFO Size Width(TX/RX)[1]	C_FIFO_DEPTH	4-7	4
DMA_EN	C_DMA_EN	true, false	false

Table 2.1: Generics

[1]FIFO depth is 2^{Value} =(16,32,64,128)

3 IO-Ports

Port	**width**	**direction**	**Description**
SPI_SCLK	1	input	Serial clock input
SPI_MOSI	1	input	Master Out Slave in
SPI_MISO	1	output	Master in Slave out
SPI_SS	1	input	Slave select
opb_irq	1	output	IRQ Output

Table 3.1: external ports

4 Registers

4.1 Adressmap

Name	Adress	Acess	Description
SPI_CR	0x00	R/W	SPI Control Register
SPI_SR	0x04	R/W	SPI Status Register
SPI_TD	0x08	W	SPI Transmit Data Register
SPI_RD	0x0C	R	SPI Receive Data Register
TX_THRESH	0x10	R/W	TX-Threshold Prog Full/Emty
RX_THRESH	0x14	R/W	RX-Threshold Prog Full/Emty
TX_DMA_CTL	0x18	R/W	TX DMA Control
TX_DMA_ADDR	0x1C	R/W	TX DMA Base Adress Offset
TX_DMA_NUM	0x20	R/W	TX DMA Number of Transfers
RX_DMA_CTL	0x24	R/W	RX DMA Control
RX_DMA_ADDR	0x28	R/W	RX DMA Base Adress Offset
RX_DMA_NUM	0x2C	R/W	RX DMA Number of Transfers
DGIE	0x40	R/W	Device global IRQ Enable Register
IPISR	0x44	R/W	IRQ Status Register
IPIER	0x48	R/W	IRQ Enable Register

Table 4.1: Address-Map

4.2 SPI_CR

Bit	Name	Acess	Reset Value	Description
31	DGE	R/W	0	Device Global Enable 0: Disable 1: Enable
30	TX_EN	R/W	0	Transmit Enable 0: Disable 1: Enable
29	RX_EN	R/W	0	Receive Enable 0: Disable 1: Enable
29	RESET	R/W	0	Reset Device(self cleared) 0: Normal Operation 1: Reset SPI-Core(SR/FIFO)
29..0	reserved			

Table 4.2: SPI_CR Register

4.3 SPI_SR

Bit	Name	Acess	Reset Value	Description
31	TX Prog Full	R	0	Prog Full Flag 1: FIFO Prog Full
30	TX Full	R	0	Full Flag 1: FIFO Full
29	TX Overflow	R	0	Overflow Flag 1: FIFO Overflow (Cleared only at Reset)
28	TX Prog Empty	R	0	Prog Empty Flag 1: FIFO Prog Empty
27	TX Empty	R	0	Full Flag 1: FIFO Empty
26	TX Underflow	R	0	Underflow Flag 1: FIFO Underflow (Cleared only at Reset)
25	RX Prog Full	R	0	Prog Full Flag 1: FIFO Prog Full
24	RX Full	R	0	Full Flag 1: FIFO Full
23	RX Overflow	R	0	Overflow Flag 1: FIFO Overflow (Cleared only at Reset)
22	RX Prog Empty	R	0	Prog Empty Flag 1: FIFO Prog Empty
21	RX Empty	R	0	Full Flag 1: FIFO Empty
20	RX Underflow	R	0	Underflow Flag 1: FIFO Underflow (Cleared only at Reset)
19	Chip Select	R	0	Chip Select Flag 0: CS_N Low 1: CS_N High
18	TX DMA Done	R	0	Transmit DMA done 0: TX DMA in progress 1: TX DMA all Transfers done
17	RX DMA Done	R	0	Receive DMA done 0: RX DMA in progress 1: RX DMA all Transfers done
16..0	reserved			

Table 4.3: SPI_SR Register

4.4 TX_THRESH

Bit	Name	Acess	Reset Value	Description
31..16	TX_THRESH_PROG_FULL	R/W	0	Transmit Prog Full Threshold $[1..2^{C\_FIFO\_DEPTH} - 2]$
15..0	TX_THRESH_PROG_EMPTY	R/W	0	Transmit Prog Empty Threshold $[1..2^{C\_FIFO\_DEPTH} - 2]$

Table 4.4: TX_THRESH Register

4.5 RX_THRESH

Bit	Name	Acess	Reset Value	Description
31..16	RX_THRESH_PROG_FULL	R/W	0	Receive Prog Full Threshold $[1..2^{C\_FIFO\_DEPTH} - 2]$
15..0	RX_THRESH_PROG_EMPTY	R/W	0	Receive Prog Empty Threshold $[1..2^{C\_FIFO\_DEPTH} - 2]$

Table 4.5: RX_THRESH Register

4.6 DGIE

Bit	Name	Acess	Reset Value	Description
31	DGIE	R/W	0	Global IRQ Ebable 0: Disable 1: Enable
29..0	reserved			

Table 4.6: DGIE Register

4.7 IPISR

Bit	Name	Acess	Reset Value	Description
31	TX_Prog_Empty	R/ToW[1]	0	IRQ Prog Empty Flag
29	TX_Empty	R/ToW	0	IRQ Full Flag
28	RX_Prog_Full	R/ToW	0	IRQ Prog Full Flag
27	RX_Full	R/ToW	0	IRQ Full Flag
26	SS_FALL	R/ToW	0	IRQ SS FALL Flag
25	SS_RISE	R/ToW	0	IRQ SS RISE Flag
24..0	reserved			

Table 4.7: IPISR Register

4.8 IPISE

Bit	Name	Acess	Reset Value	Description
31	TX_Prog_Empty	R/W	0	IRQ Prog Empty Enable
29	TX_Empty	R/W	0	IRQ Full Enable
28	RX_Prog_Full	R/W	0	IRQ Prog Full Enable
27	RX_Full	R/W	0	IRQ Full Enable
26	SS_FALL	R/W	0	IRQ SS FALL Enable
25	SS_RISE	R/W	0	IRQ SS RISE Enable
24	TX_DMA_DONE	R/W	0	IRQ TX Transfer done Enable
23	TX_DMA_DONE	R/W	0	IRQ RX Transfer done Enable
22..0	reserved			

Table 4.8: IPISE Register

1: IRQ enabled

[1]Read and ToggleOnWrite (writing 1 clears the bit)

5 System Integration

To integrate this IP-Core in your System, unzip the opb_spi_slave.zip to your project-directory. Then Rescan the user repository with *Project → Rescan User Repositories*. This will take some seconds. After this you find the core in the *IP Catalog → Project Repository*.

5.1 MPD-File

```
BEGIN opb_spi_slave
 PARAMETER INSTANCE = opb_spi_slave_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x7d600000
 PARAMETER C_HIGHADDR = 0x7d60ffff
 BUS_INTERFACE MSOPB = mb_opb
 PORT sclk = opb_spi_slave_0_sclk
 PORT ss_n = opb_spi_slave_0_ss_n
 PORT mosi = opb_spi_slave_0_mosi
 PORT miso = opb_spi_slave_0_miso
 PORT opb_irq = opb_spi_slave_0_opb_irq
END
```

5.2 UCF-File

```
# assign I/O Pins
NET opb_spi_slave_0_sclk_pin LOC= AA24; # must CC capable IO in virtex-4
NET opb_spi_slave_0_ss_n_pin LOC= V20;
NET opb_spi_slave_0_mosi_pin LOC= AC25;
NET opb_spi_slave_0_miso_pin LOC= AC24;
NET opb_spi_slave_0_miso_pin SLEW = FAST;

#### Module OPB_SPI_Slave constraints
Net opb_spi_slave_0_sclk_pin TNM_NET = spi_clk;
TIMESPEC TS_spi_clk = PERIOD spi_clk 40 ns;

NET "opb_spi_slave_0_mosi_pin" TNM = "spi_in";
#NET "opb_spi_slave_0_cs_n_pin" TNM = "spi_in";
TIMEGRP "spi_in" OFFSET = IN 5 ns VALID 10 ns BEFORE "opb_spi_slave_0_sclk_pin" HIGH ;

NET "opb_spi_slave_0_miso_pin" TNM = "spi_out";
TIMEGRP "spi_out" OFFSET = OUT 14 ns AFTER "opb_spi_slave_0_sclk_pin" LOW ;
```

5.3 Register Header

```
#include "xparameters.h"

#define XSS_CR (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x00))
#define XSS_SR    (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x01))
#define XSS_TD    (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x02))
#define XSS_RD    (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x03))
#define XSS_TX_THRESH   (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x04))
#define XSS_RX_THRESH   (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x05))
#define XSS_TX_DMA_CTL  (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x06))
#define XSS_TX_DMA_ADR  (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x07))
#define XSS_TX_DMA_NUM  (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x08))
#define XSS_RX_DMA_CTL  (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x09))
#define XSS_RX_DMA_ADR  (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x0A))
#define XSS_RX_DMA_NUM  (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x0B))

#define XSS_DGIE (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x10))
#define XSS_IPISR (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x11))
#define XSS_IPIER (XPAR_OPB_SPI_SLAVE_0_BASEADDR + (4* 0x12))

//XSS_SPI_CR
#define XSS_CR_SPE_MASK (0x01)
#define XSS_CR_TX_EN_MASK (0x02)
#define XSS_CR_RX_EN_MASK (0x04)
#define XSS_CR_RESET_MASK (0x08)

//XSS_SPI
// Transmit
#define XSS_SR_TX_PROG_FULL_MASK 0x0001
#define XSS_SR_TX_FULL_MASK 0x0002
#define XSS_SR_TX_OVERFLOW_MASK 0x0004
#define XSS_SR_TX_PROG_EMPTY_MASK 0x0008
#define XSS_SR_TX_EMPTY_MASK 0x0010
#define XSS_SR_TX_UNDERFLOW_MASK 0x0020
// Receive
#define XSS_SR_RX_PROG_FULL_MASK 0x0040
#define XSS_SR_RX_FULL_MASK 0x0080
#define XSS_SR_RX_OVERFLOW_MASK 0x0100
#define XSS_SR_RX_PROG_EMPTY_MASK 0x0200
#define XSS_SR_RX_EMPTY_MASK 0x0400
#define XSS_SR_RX_UNDERFLOW_MASK 0x0800
// Chip Select
#define XSS_SR_CHIP_SELECT_MASK 0x1000
// DMA
#define XSS_SR_TX_DMA_done 0x2000
#define XSS_SR_RX_DMA_done 0x4000


// Device Global Interrupt Enable
```

```
#define XSS_DGIE_Bit_Enable 0x0001

// Interrupt /Enable Status Register
#define XSS_ISR_Bit_TX_Prog_Empty  0x0001
#define XSS_ISR_Bit_TX_Empty       0x0002
#define XSS_ISR_Bit_TX_Underflow   0x0004
#define XSS_ISR_Bit_RX_Prog_Full   0x0008
#define XSS_ISR_Bit_RX_Full        0x0010
#define XSS_ISR_Bit_RX_Overflow    0x0020
#define XSS_ISR_Bit_SS_Fall        0x0040
#define XSS_ISR_Bit_SS_Rise        0x0080
#define XSS_ISR_Bit_TX_DMA_done 0x0100
#define XSS_ISR_Bit_RX_DMA_done 0x0200

// TX DMA Control Register
#define XSS_TX_DMA_CTL_EN 0x0001

// RX DMA Control Register
#define XSS_RX_DMA_CTL_EN 0x0001
```

6 Operations

7 Architecture

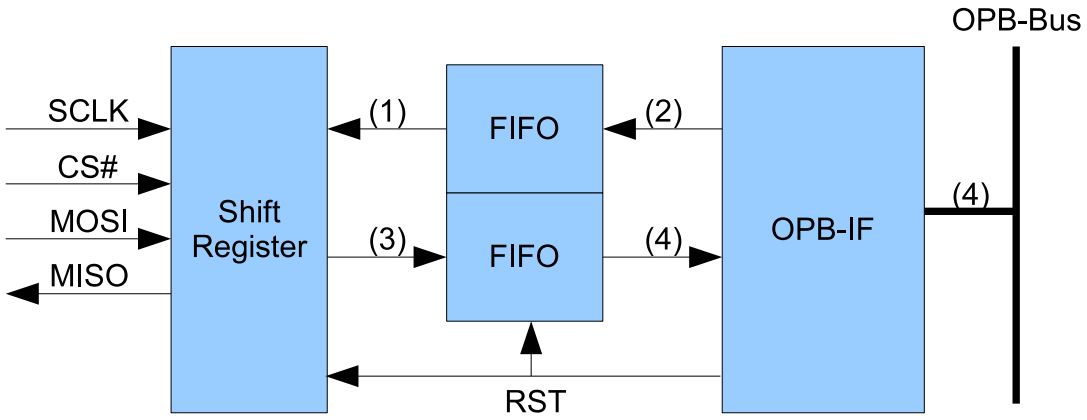

Figure 7.1: Blockdiagramm

7.1 Serial Shift Register

Signal	Direction	Description
Generics		
C_SR_WIDTH	-	Shift register width
C_MSB_FIRST	-	Transfer MSB First
Global		
rst	input	Async-Reset
External Interface		
sclk	input	Serial clock input
cs_n	input	Chip Select
mosi	input	Master Out Slave in
miso_o	output	Master in Slave out
miso_i	input	not used
miso_t	output	MISO Tristate
TX-FIFO (1)		
sr_tx_clk	output	FIFO-TX CLK
sr_tx_en	output	FIFO-TX enable
sr_tx_data[C_SR_WIDTH:0]	input	FIFO-TX Data
RX-FIFO (3)		
sr_rx_clk	output	FIFO-RX CLK
sr_rx_en	output	FIFO-RX enable
sr_rx_data[C_SR_WIDTH:0]	output	FIFO-RX Data

Table 7.1: Serial-Register

7.2 FIFO

Signal	Direction	Description
Generics		
C_FIFO_WIDTH	-	Shift register width
C_FIFO_SIZE	-	FIFO Size Width
C_SYNC_TO	-	FIFO Sync Flags to Clock
Global		
rst	input	Async-Reset
Write-Port (2,3)		
wr_clk	input	FIFO Write CLK
wr_en	input	FIFO Write enable
din[C_FIFO_WIDTH:0]	input	FIFO Write data
READ-Port (1,4)		
rd_clk	input	FIFO Read CLK
rd_en	input	FIFO Read enable
dout[C_FIFO_WIDTH:0]	output	FIFO Read data
Flags (4)		
empty	output	FIFO Emtpy
full	output	FIFO Full
overflow	output	FIFO Overflow
underflow	output	FIFO Underflow
prog_full_thresh[C_FIFO_WIDTH:0]	output	FIFO Programmable Full Threshold
prog_empty_thresh[C_FIFO_WIDTH:0]	output	FIFO Programmable Empty Threshold
prog_full	output	FIFO Programmable Full
prog_empty	output	FIFO Programmable Empty

Table 7.2: TX-FIFO

7.3 OPB_IF

Signal	Direction	Description
Generics		
C_BASEADDR	-	Base address for OPB SPI
C_HIGHADDR	-	High address for OPB SPI
C_OPB_AWIDTH	-	OPB address bus width
C_OPB_DWIDTH	-	OPB data bus width
C_FAMILY	-	Target FPGA Family
C_SR_WIDTH	-	Shift register width
C_FIFO_WIDTH	-	Shift register width
C_FIFO_SIZE	-	FIFO Size Width
C_NUM_FLG	-	Number of FIFO Status flags
C_NUM_INT	-	Number of IRQ Sources
OPB-Bus		
OPB_rst	input	Async-Reset
OPB_ABus[C_OPB_AWIDTH-1:0]	input	Adress-Bus
OPB_BE[C_OPB_DWIDTH/8-1:0]	input	Bytes Enables
OPB_Clk	input	Clock
OPB_DBus[C_OPB_DWIDTH-1:0]	input	Data-Bus to slave
OPB_RNW	input	Read/Write
OPB_Rst	input	Reset
OPB_select	input	Select
OPB_seqAddr	input	Sequential Adress Enable
Sln_DBus	output	Data-Bus to Master
Sln_errAck	output	Error Acknowledge
Sln_retry	output	Retry
Sln_toutSup	output	Timeout Suppression
Sln_xferAck	output	transfer Acknowledge
FIFO-PORT (2)		
opb_tx_en	output	FIFO-TX Write Enable
opb_tx_data[C_SR_WIDTH:0]	output	FIFO-TX Write Data
tx_thresh[(2*C_FIFO_SIZE)-1:0]	output	FIFO-TX Prog Thresholds
FIFO-PORT (4)		
opb_rx_en	output	FIFO-RX Read Enable
opb_rx_data[C_SR_WIDTH:0]	input	FIFO-RX Read Data
rx_thresh[(2*C_FIFO_SIZE)-1:0]	output	FIFO-RX Prog Thresholds
FIFO-Flags(2,4)		
opb_fifo_flg[C_NUM_FLG-1:0]	input	FIFO Flags
IRQ-Signals		
opb_dgie	output	Device Global IRQ Enable
opb_ier(C_NUM_INT-1:0)	output	IRQ Enable Register
opb_isr(C_NUM_INT-1:0)	input	IRQ Status Register
opb_isr_clr(C_NUM_INT-1:0)	output	Clear IRQ Flags

Table 7.3: OPB_IF

Bibliography

[1] Xilinx embedded development kit. http://www.xilinx.com/edk.

List of Figures

List of Tables