up_spi.v

AUTHORS

JAY CONVERTINO

DATES

2024/04/29

INFORMATION

Brief

uP Core for interfacing with axis spi that emulates the ALTERA SPI IP.

License MIT

up_spi_master

```
module up_spi_master #(
parameter
ADDRESS_WIDTH
 =
32,
parameter
BUS_WIDTH
 =
4,
parameter
CLOCK_SPEED
 =
100000000,
parameter
```

```
SELECT_WIDTH
  =
16,
parameter
DEFAULT_RATE_DIV
  =
0,
parameter
DEFAULT_CPOL
  =
0,
parameter
DEFAULT_CPHA
  =
0
) ( input clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WI
```

SPI Master core with axis input/output data. Read/Write is size of BUS_WIDTH bytes. Write activates core for read.

Parameters

ADDRESS_WIDTH parameter	Width of the uP address port, max 32 bit.
BUS_WIDTH parameter	Width of the uP bus data port(can not be less than 2 bytes, max tested is 4).
CLOCK_SPEED parameter	This is the aclk frequency in Hz, this is the the frequency used for the bus and is divided by the rate.
SELECT_WIDTH parameter	Bit width of the slave select, defaults to 16 to match altera spi ip.
DEFAULT_RATE_DIV parameter	Default divider value of the main clock to use for the spi data output clock rate. 0 is 2 (2^(X+1) X is the DEFAULT_RATE_DIV)
DEFAULT_CPOL parameter	Default clock polarity for the core (0 or 1).
DEFAULT_CPHA parameter	Default clock phase for the core (0 or 1).

Ports

clk	Clock for all devices in the core
rstn	Negative reset
up_rreq	uP bus read request
up_rack	uP bus read ack
up_raddr	uP bus read address
up_rdata	uP bus read data
up_wreq	uP bus write request
up_wack	uP bus write ack
up_waddr	uP bus write address
up_wdata	uP bus write data
irq	Interrupt when data is received
sclk	spi clock, should only drive output pins to devices.
mosi	transmit for master output
miso	receive for master input
ss_n	slave select output

DIVISOR

```
localparam DIVISOR = BUS_WIDTH/2
```

Divide the address register default location for 1 byte access to multi byte access. (register offsets are byte offsets).

REG_SIZE

```
localparam REG_SIZE = 8
```

Number of bits for the register address

REGISTER INFORMATION

Core has 7 registers at the offsets that follow when at a full 32 bit bus width, Internal address is OFFSET >> BUS_WIDTH/2 (32bit would be h4 >> 2 = 1 for internal address).

RX_DATA_REG	h00
TX_DATA_REG	h04
STATUS_REG	h08
CONTROL_REG	h0C
RESERVED	h10
SLAVE_SELECT_REG	h14
EOP_VALUE_REG	h18
CONTROL_EXT_REG	h1C

RX_DATA_REG

```
localparam RX_DATA_REG = 8'h0 >> DIVISOR
```

Defines the address offset for RX DATA OUTPUT

RX DATA REGISTER	
31:N	N:0
UNUSED	RECEIVED DATA

Valid bits are from BUS_WIDTH*8-1:0, which are data.

TX_DATA_REG

```
localparam TX_DATA_REG = 8'h4 >> DIVISOR
```

Defines the address offset to write the TX DATA INPUT.

TX DATA REGISTER	
31:N	N:0
UNUSED	TRANSMIT DATA

Valid bits are from BUS_WIDTH*8-1:0, which are data.

STATUS_REG

```
localparam STATUS_REG = 8'h8 >> DIVISOR
```

Defines the address offset to read the status bits.

STATUS REGISTER								
31:10	9	8	7	6	5	4	3	2:0
UNUSED	EOP	E	RRDY	TRDY	TMT	TOE	ROE	UNUSED

Status Register, 1 is considered active.

EOP 9, This bit is active(1) when the EOP_VALUE_REG is equal to RX_DATA_REG or TX_DATA_REG.

E 8, Logical or of TOE and ROE (Clear by writing status).

RRDY 7, Receive is ready (full) when the bit is 1, empty when the bit is 0.

TRDY 6, Transmit is ready (empty) when the bit is 1, full when the bit is 0.

TMT 5, Transmit shift register empty is set to 1 when all bits have been output.

TOE 4, Transmit overrun is set to 1 when a TX_DATA_REG write happens whne TRDY is 1 (Clear by writing status reg).

ROE 3, Receive overrun is set to 1 when RRDY is 1 and a new received word is going to be written to RX_DATA_REG (Clear by writing status reg)

CONTROL_REG

```
localparam CONTROL_REG = 8'hC >> DIVISOR
```

Defines the address offset to set the control bits.

CONTROL REGISTER									
31:11	10	9	8	7	6	5	4	3	2:0
UNUSED	SSO	IEOP	IE	IRRDY	ITRDY	UNUSED	ITOE	IROE	UNUSED

Control Register, 1 is considered active. **All zeros on reset.**

SSO 10, Setting this to 1 will force all ss_n lines to 0 (selected).

IEOP 9, Generate a interrupt on EOP status bit going active if set to 1.

IE 8, Enable (1) or disable(0) all interrupts that are active.

IRRDY 7, Generate a interrupt on RRDY status bit going active if set to 1.

ITRDY 6, Generate a interrupt on TRDY status bit going active if set to 1.

ITOE 4, Generate a interrupt on TOE status bit going active if set to 1.

IROE 3, Generate a interrupt on ROE status bit going active if set to 1.

RESERVED

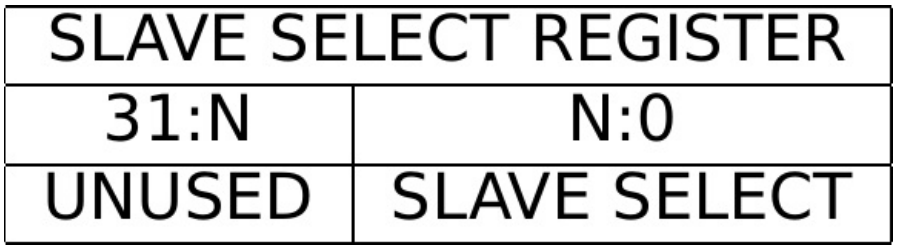
```
localparam RESERVED = 8'h10 >> DIVISOR
```

Defines the address offset that is not used.

SLAVE_SELECT_REG

```
localparam SLAVE_SELECT_REG = 8'h14 >> DIVISOR
```

Defines the address offset to set the slave select value



Valid bits are from SELECT_WIDTH-1:0, which are the slave select output lines to drive low during data transmission.

EOP_VALUE_REG

```
localparam EOP_VALUE_REG = 8'h18 >> DIVISOR
```

Defines the address offset to set the end of packet match value

Valid bits are from BUS_WIDTH*8:0, which are used to check for a word match between rx and/or tx and update status.

CONTROL_EXT_REG

```
localparam CONTROL_EXT_REG = 8'h1C >> DIVISOR
```

Defines the address offset for control register extensions

CONTROL REGISTER EXTENDED			
31:6	5	4	3:0
UNUSED	CPHA	CPOL	RATE DIV

Control Extension to add capabilities to Altera IP core.

CPHA	5, Clock Phase Bit, 0 or 1 per SPI specs (default value set by IP parameter).
CPOL	4, Clock Polarity bit, 0 or 1 per SPI specs (default value set by IP parameter).
RATE_TOP	3, Top bit for rate control. Divider values are 0 to 15 (2^X+1 where X is the divider value).
RATE_BOT	0, Bottom bit for rate control.

INSTANTIATED MODULES

inst_axis_spi

```
axis_spi_master #(
                                                                                .
CLOCK_SPEED(CLOCK_SPEED),
                                                                                .
BUS_WIDTH(BUS_WIDTH),
                                                                                .
SELECT_WIDTH(SELECT_WIDTH)
) inst_axis_spi_master ( .aclk(clk), .arstn(rstn), .s_axis_tdata(r_tx_wdata
```

SPI Master instance with AXIS interface