**Design Verification Strategy**

**SPI**

**Rev. A**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision History** | **Description** | **Author** |
| **25.11.2022** | **A** | Initial Draft |  |
|  | **B** | 1.Design specifications  2.Feature extraction  3.DV Plan Strategy  4.TestBench Architecture |
|  | **C** | Test Cases waveform  1.  2.  3.  4.  5.  6. |
|  | **d** | functional coverage |
|  | **e** | Assertion |
|  | **f** | Code Coverage(after regression) |

**Table of content page no**

1. Design specifications ………………………………………………… 3

1.1

1.2

1.3.

2. Feature extraction.………………………………………………… 4

2.1

2.2

3. DV Plan Strategy…………………………………………… 5

4. Testbench Architecture ………………………………… 7

5. Test case screenshots ………………………………… 8

6. Functional coverage ……………………………………… 12

7. Assertion …………………………………………………… 12

8.Code coverage ……………………………………………… 13

**1.Design specifications :**

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:

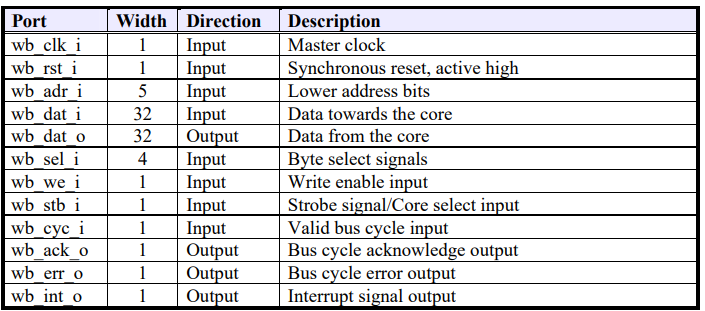
* SPI (a trademark of Motorola Semiconductor)
* Microwire/Plus (a trademark of National Semiconductor)

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

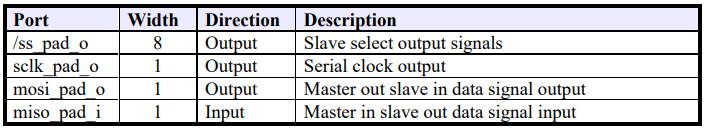
The SPI Master core is compatible with both above-mentioned protocols as master with some additional functionality. At the hosts side, the core acts like a WISHBONE compliant slave device

**1.1 IO ports**

1.1 WISHBONE interface signals

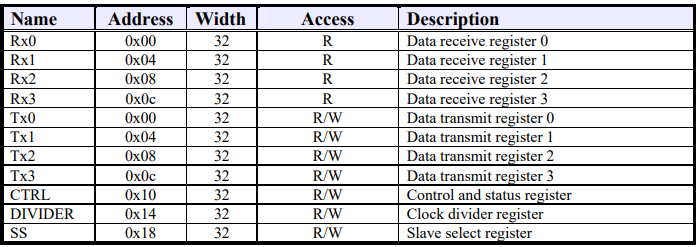


1.2 SPI external connection

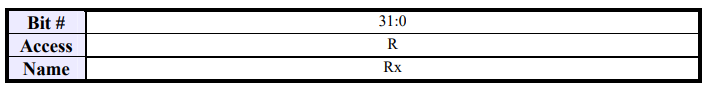


**Registers**

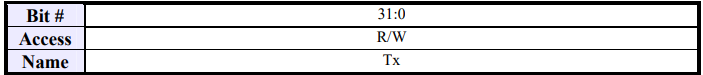
1 Core Registers list



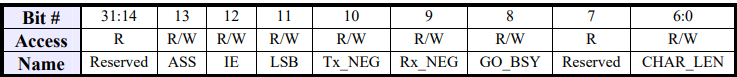
**Data receive registers[RxX]**

****

**Data transmit register [TxX]**

****

**Control and status register [CTRL]**

****

**Reset Value: 0x00000000**

**ASS**

If this bit is set, ss\_pad\_o signals are generated automatically. This means that slave select signal, which is selected in SS register is asserted by the SPI controller, when transfer is started by setting CTRL[GO\_BSY] and is de-asserted after transfer is finished. If this bit is cleared, slave select signals are asserted and de-aserted by writing and clearing bits in SS register.

**IE**

If this bit is set, the interrupt output is set active after a transfer is finished. The Interrupt signal is deasserted after a Read or Write to any register.

**LSB**

If this bit is set, the LSB is sent first on the line (bit TxL[0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit RxL[0]). If this bit is cleared, the MSB is transmitted/received first (which bit in TxX/RxX register that is depends on the CHAR\_LEN field in the CTRL register).

**Tx\_NEG**

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.

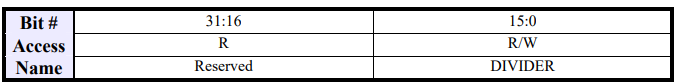
**Rx\_NEG**

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\_BSY

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

**Divider register [DIVIDER]**

****

Reset Value: 0x0000ffff

**Slave select register [SS]**

****

****

**SS**

If CTRL[ASS] bit is cleared, writing 1 to any bit location of this field sets the proper ss\_pad\_o line to an active state and writing 0 sets the line back to inactive state. If CTRL[ASS] bit is set, writing 1 to any bit location of this field will select appropriate ss\_pad\_o line to be automatically driven to active state for the duration of the transfer, and will be driven to inactive state for the rest of the time

**2. Feature extraction:**

* Full duplex synchronous serial data transfer
* Variable length of transfer word up to 128 bits
* MSB or LSB first data transfer
* Rx and Tx on both rising or falling edge of serial clock independently
* 8 slave select lines
* Fully static synchronous design with one clock domain
* Technology independent Verilog
* Fully synthesizable

**3.DV Plan Strategy**

1. **verify the basic write and read transaction between master and slave.**

1. The master drives the address and control signals onto the bus after the rising edge of HCLK.

2. The slave then samples the address and control information on the next rising edge of HCLK.

3. After the slave has sampled the address and control it can start to drive the appropriate HREADYOUT response. This response is sampled by the master on the third rising edge of HCLK.

1. **4 Beat Burst Write Read transaction:**

**a). Four Beat Burst Write Transaction**

for example:

first clk: HADDR = 32’H0000700

HWDATA=32”0000001C

HWRITE =1 HIGH

and HREADY goes high i.e. 1’h1 .

**HBRUST is 3’h3 is increment 4** , the **HTRANS** is Non-sequential at first clk pulse and sequential from next clk pulse onwards.

Response is 1’h0 which represents the response i.e. HRESP as OKAY.

**b).Four Beat Burst Read Transaction**

4 Beat Burst Read transaction where for the first clk the HADDR is at 32’h0000700 as the data is written on a particular address, after the data is written on the address the data is to read on the same address, so at HADDR 32’h0000700 the same data is read on same address location i.e. when HREADY is 1’h1 and with response i.e. HRESP as OKAY

1. **8 Beat Burst Write Read transaction:**

**a). 8 Beat Burst Write Transaction**

for example:

first clk: HADDR = 32’H0000700

HWDATA=32”0000001C

HWRITE =1 HIGH

and HREADY goes high i.e. 1’h1 .

**HBRUST is 3’h5 is increment 8,** the **HTRANS** is Non-sequential at first clk pulse and sequential from next clk pulse onwards.

Response is 1’h0 which represents the response i.e. HRESP as OKAY.

**b).8 Beat Burst Read Transaction**

4 Beat Burst Read transaction where for the first clk the HADDR is at 32’h0000700 as the data is written on a particular address, after the data is written on the address the data is to read on the same address, so at HADDR 32’h0000700 the same data is read on same address location i.e. when HREADY is 1’h1 and with response i.e. HRESP as OKAY

1. **4 beat wrapping burst**

Wrapping bursts wrap when they cross an address boundary.

The address boundary is calculated as the product of the number of beats in a burst and the size of the transfer. The number of beats are controlled by HBURST and the transfer size is controlled by HSIZE.

For example,

a four-beat wrapping burst of word (4-byte) accesses wraps at 16-byte boundaries. Therefore, if the start address of the burst is 0x34, then it consists of four transfers to addresses **0x34, 0x38, 0x3C, and 0x30.**

1. **VERIFY the Slave transfer responses**

A slave must provide a response that indicates the status of the transfer when it is accessed. The transfer status is provided by the **HRESP** signal.

**HRESP**: 0 : OKAY The transfer has either completed successfully or additional cycles are required for the slave to complete the

request.

1 ERROR An error has occurred during the transfer. The error condition must be signaled to the master so that it is aware the transfer has been unsuccessful.

**6.verify INVALID transaction like IDLE to SEQ /IDLE to BUSY /BUSY to NONSEQ**

**4.Testbench Architecture :**

**Verification Components**

uvm verification component classes are derived from uvm\_component class

Following are some of the uvm component classes

 uvm\_agent

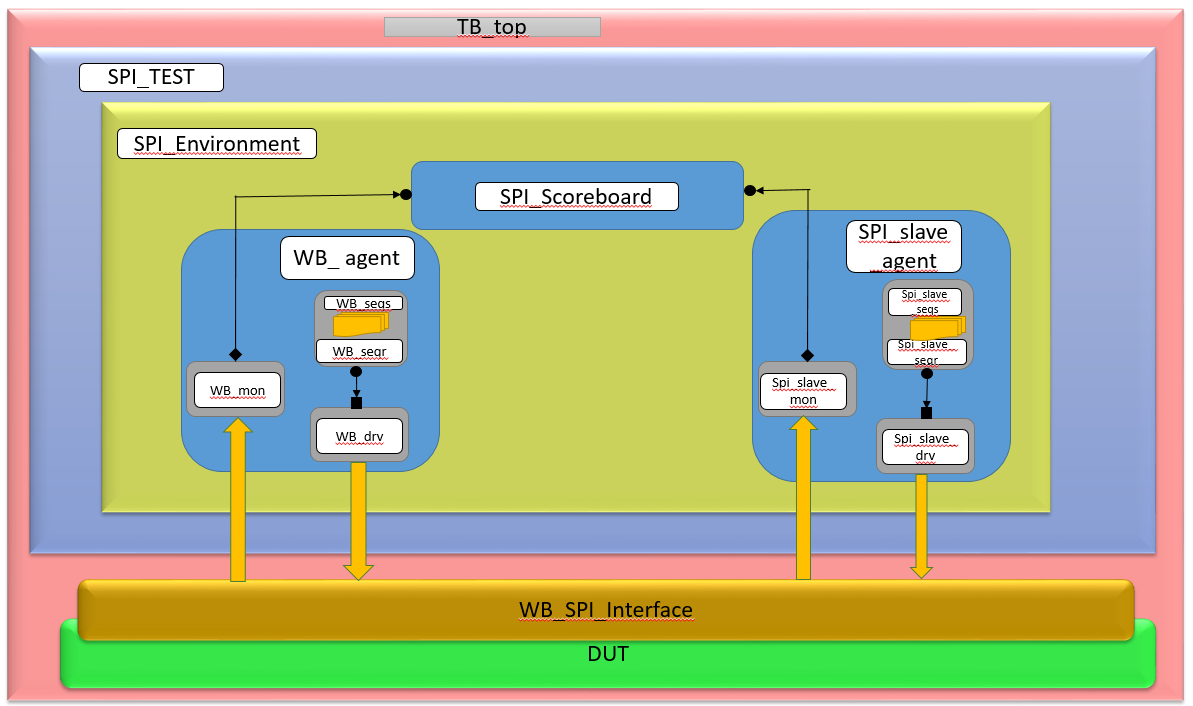
 uvm\_monitor

 uvm\_scoreboard

 uvm\_driver

 uvm\_sequencer

NOTE: uvm\_env and uvm\_test are also extended from uvm\_component.



**TEST CASES:**