GH09.B.5.bridge - Bridge Conversion

General Description

This design implements a protocol conversion between a serial bus and a parallel bus. The scenario assumes that the design is a master controlling a serial input bus and the device is a slave on a parallel bus. Requests made on the parallel bus (by external masters) are communicated and converted to request on the serial bus to an external slave. This includes buffering incoming packets from the Serial Peripheral Interface (SPI) protocol for the serial bus and using a Wishbone compliant bus to receive and transmit packets on the bus complying to external master requests.

Features

- The Bridge Chip is a slave on a Wishbone direct connection to one master. This simplifies the details for the external environment.
- The bridge chip is a master on the SPI that makes read and write requests to a slave device.
- Wishbone compliant bus interface
- SPI serial bus interface

Block Diagram

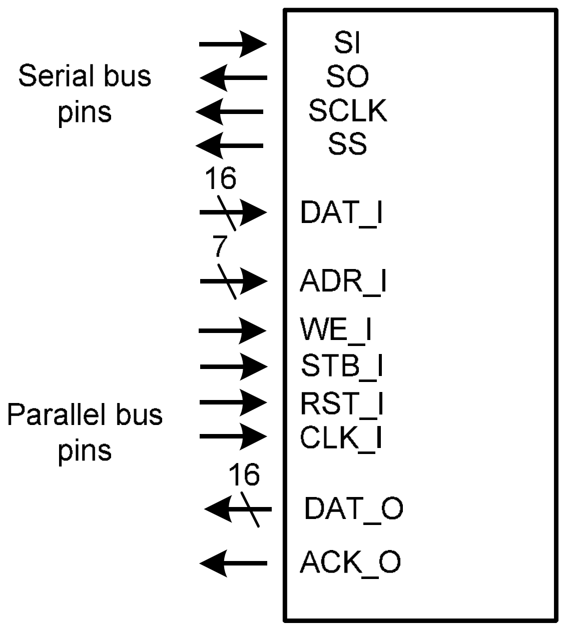

Figure 1 – Block diagram of how the design operates as a chip in a system

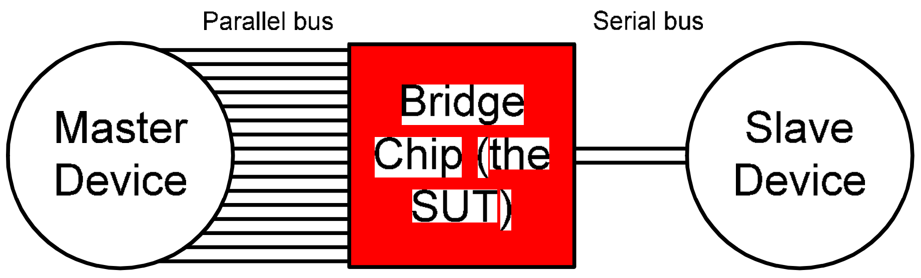

Figure 2 – A diagram of how the bridge device fits in the system view. Note that not all the signals are included in this diagram.

Details

There are three sections to the bridge chip details. They describe the chip behaviour based on the following:

1. Parallel Bus Communication – This section describes the WISHBONE compliant parallel bus, and how this Bridge device operates as a slave on this bus.

2. Serial Bus Communication – This section describes the Serial Peripheral Interface bus, and how this Bridge device operates as a master on this bus.

3. Parallel and Serial Bus bridging – This section describes how requests on the Parallel bus are translated to the Serial bus for both a read and write to the slave device.

Parallel Bus Communication

This section describes the behaviour of the slave Wishbone compliant parallel bus. This includes a description of how the parallel bus operates, and a section on the rules satisfied for the WISHBONE specification.

WISHBONE Slave Operation

The bridge chip must handle read, write, and reset on the parallel bus. Table 1 shows the basic data sheet for the WISHBONE compliant part of the parallel bus.

Table 1 – Summary of Wishbone specification

Wishbone Datasheet for a 16-bit output port with 16-bit granularity	
Description	**Specification**
General Description:	SLAVE, READ/WRITE
Data port size: Data port granularity: Data port Maximum operand: Data transfer ordering Data transfer sequencing:	16-bit 16-bit 16-bit Big endian and/or little endian Undefined
Supported signal list and cross reference to equivalent WISHBONE signals:	Matched

We now take a more in depth look at the read and write timing diagrams from the perspective of the slave. These details are taken from Section 3, titled "Wishbone Classic Bus Cycles", of the wishbone b3 specification. Our description does not include details of timing, which assumes that based on the clock speed of the design (not shown on the block diagram) and the parallel bus clock (CLK_I).

Read Cycle

The read cycle is defined by the following timing diagram:

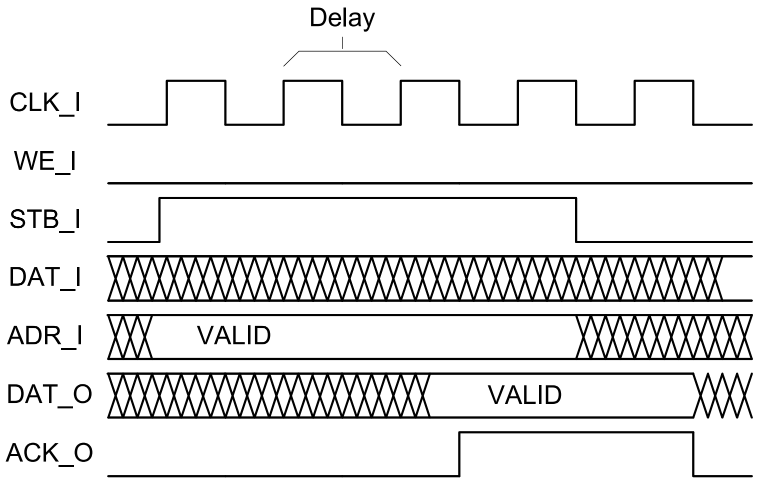

Figure 3 – The read signals for the parallel bus.

The reader should notice the delay clock cycle in the Figure 3 timing diagram marked by "Delay". This delay is controlled and can be extended over multiple clock cycles by not asserting the ACK_O signal. This controls the handshaking between the master and slave to acknowledge when the read is complete, and will be used by the bridge chip while the read request is being sent to the slave device.

Write Cycle

The write cycle is similar to the read and is defined by the following timing diagram

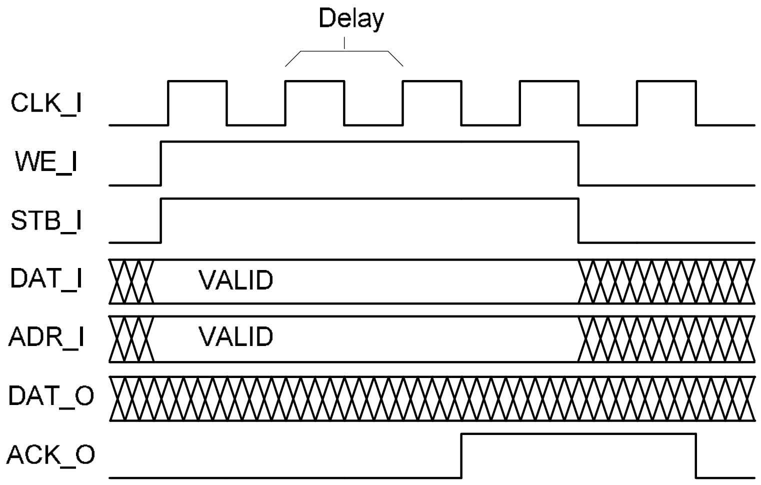

Figure 4 – The write signals for the parallel bus.

The write cycle, in Figure 4, also has the ability to slave delay the writing by not sending the ACK_O signal until the data is latched (as described above). Both of these delay handshakes in the read and write are important for this Bridge design since the conversion of the protocol and writing to the actual slave on the serial bus may need delays to complete the serial bus writes and reads to the slave device (depending on clock domains).

Reset

The RESET_I signal is used to reset the parallel bus interface, and can happen at any time. The reset is triggered on the rising edge of the clock.

WISHBONE Compliance

The following list describes the wishbone specification rules for this instance of a wishbone bus. These details need to be satisfied to meet the Wishbone specification (see included document wishbone_spec_b3.pdf in the RELATED_DOCUMENTS directory). Not that we provide only some of the rules:

2.1.1

RULE 2.00

Included in the distribution is the datasheet this information was created from

RULE 2.15

(1) Wishbone specification b3

(2) Slave

(3) CLK_I, DAT_I(15..0), DAT_O(15..0), ADR_I(6..0), RST_I, ACK_O, STB_I, WE_I

(4) ERR_I not supported

(5) RTY_I not supported

(6) No tag signals

(7) 16-bit port port size

(8) 16-bit granularity size

(9) 16-bit operand size

(10) Doesn’t matter since port size and granularity size are equal

(11) Undefined

(12) No constraints on CLK_I

2.1.2

RULE 2.20

All signals must adhere to naming above

2.1.3

RULE 2.30

All Wishbone interface signals MUST use active high logic

Serial Bus Communication

The Serial Peripheral Interface (SPI) interface has the following specifications for this Bridge chip acting as the Master on this bus.

Write to slave

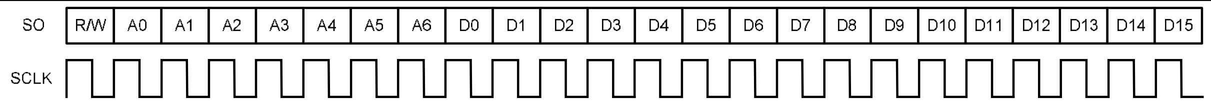

Figure 5 – A write message for the slave

The slave has 2^7 (128) registers to write to. The master can write to any of the registers with a packet similar to the one in Figure 5. The first bit is set to indicate a write message (and the chip designer can choose whether this is a 1 or 0). Address bits A6-A0 tell the slave which register is to be written to. D15-D0 contains the bits that are to be written.

The pin SS needs to be set to tell the device that the packet on the bus is meant for it. This is not shown in the timing diagram in Figure 5.

Read from slave

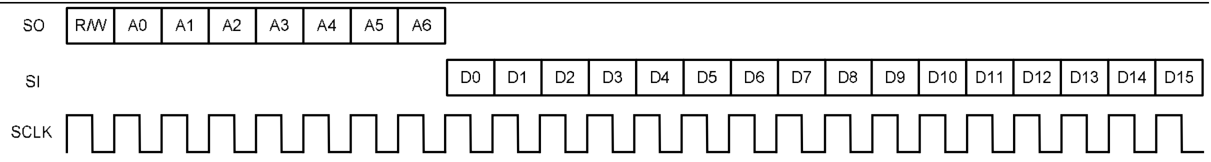

Figure 6 – A read message for the slave and reply to the master (this chip)

The read on the serial bus is shown in Figure 6. Again, the master sends a bit to indicate a read and the address of the register to be read in bits A6-A0. Once the 8 bits are sent by the master (Bridge Chip), the slave will respond on the clock with the 16 bits (D15-D0) of the data contained in the requested register.

The slave knows that this read request is for it based on the setting of the SS signal. In this chip there is only one SS signal assuming there is only one slave.

Parallel and Serial Bus bridging

The bridge chip is responsible for moving data between the serial and parallel bus. The assumption is that a Master on the parallel bus (external from the bridge chip) will make a request. This request is intended for a chip that uses a serial interface, so the bridge chip is responsible for passing on the request (either a write to a register in the slave or a read from a register in the slave).

There are two basic requests that can be made by an external master on the parallel bus:

1. Read from a register on the slave device

2. Write to a register on the slave device

We will describe each one of these scenarios in more detail.

Read from Master to Slave

The read from master to slave is initialized by a read cycle as illustrated in Figure 3. The bridge chip is then required to convert the address (ADDR_I) of the read into the form for serial bus as in Figure 6, and send these signals serially with the SCLK and SS. The slave device on the serial bus then responds with the data, as seen in Figure 6 marked D0-D15, which then needs to be buffered, and finally, sent back to the master as in Figure 3 on the DAT_O port. In each transmission, the appropriate handshaking signals need to be properly set on the buses.

Write from Master to Slave

The write from Master to slave is initialized by a write cycle as illustrated in Figure 4. The bridge chip then converts both the address (ADDR_I) and the data (DAT_I) into a serial packet as in Figure 5. This packet is then sent to the slave device with the SCLK and SS signals. Once the transmission is complete, the bridge chip completes the signals in Figure 5 to signal that the write has happened.