Gisselquist
Technology, LLC

WISHBONE SCOPE SPECIFICATION

Dan Gisselquist, Ph.D.
dgisselq (at) opencores.org

June 22, 2015

Revision History

Rev.	Date	Author	Description
0.1	6/22/2015	Gisselquist	First Draft

Contents

Figures

Tables

Preface

This, then, was and is the genesis of this project.

Dan Gisselquist, Ph.D.

1. Introduction

The wishbone Scope is a debugging tool for reading results from the chip after events have taken place. In general, the scope records data until some trigger has taken place, and then continues for some user programmable hold off. Once the holdoff has been reached, the scope stops recording and asserts an interrupt. At this time, data may be read from the scope in order from oldest to most recent. That's the basics, now for two extra details.

First, the trigger and data that the scope records is implementation dependent. The scope itself is designed to be easily reconfigurable from one build to the next so that the actual configuration may even be build dependent.

Second, the scope is built to be able to run off of a separate clock from the bus that commands and controls it. This is configurable, set the parameter "SYNCHRONOUS" to '1' to run off of a single clock. When running off of two clocks, it means that actions associated with commands issued to the scope, such as manual triggering or being disabled or released, will not act synchronously with the scope itself.

Third, the data clock associated with the scope has a clock enable line associated with it. Depending on how often the clock enable line is enabled may determine how fast the scope is primed, triggered, and eventually completes its collection.

2. Operation

So how shall one use the scope? First, before using the scope, the holdoff needs to be set. The scope is designed so that setting the scope control value to the holdoff alone will reset the scope from whatever condition it was in, freeing it to run. Once running, then upon every clock enabled clock, one sample of data is read into the scope and recorded. Once every memory value is filled, the scope has been `PRIMED`. Once the scope has been `PRIMED`, it will then be responsive to its trigger. Should the trigger be active on a clock–enabled input, the scope will then be `TRIGGERED`. It will then count for the number of clocks in the holdoff before stopping collection, placing it in the `STOPPED` state. If the holdoff is zero, the last sample in the buffer will be the sample containing the trigger.

There are two further commands that will affect the operation of the scope. The first is the `MANUAL` trigger command/bit. This will cause the scope to trigger immediately if set. If coupled with a `RESET` command, the trigger will first wait until it is `PRIMED` before the manual trigger takes effect.

The last command that can affect the operation of the scope is the `DISABLE` command/bit. This command will prevent the scope from triggering, or if triggered, prevent the scope from generating an interrupt.

3. Registers

This scope core supports two registers, as listed in Tbl. 3.1. The first is the control register, whereas the second is the data register. Each register will be discussed in detail in this chapter.

Name	Address	Width	Access	Description
WBSCOPE	0	32	R/W	Configuration, control, and status of the scope.
WBSCOPEDATA		32	R	Read out register, to read out the data from the core.

Table 3.1: List of Registers

3.1 Control Register

3.2 Data Register

This is perhaps the simplest register to explain. Before the core stops recording, reads from this register will produce reads of the bits going into the core, save only that they have not been protected from any meta-stability issues. This is useful for reading what's going on when the various lines are stuck. After the core stops recording, reads from this register return values from the stored memory. Further, after recording has stopped, every read increments an internal memory address, so that after N reads (for however long the internal memory is), the entire memory has been returned over the bus. If you would like some assurance that you are reading from the beginning of the memory, check the control register's `RZERO` flag.

Bit #	Access	Description
31	R/W	RESET_n. Write a '0' to this register to command a reset. Reading a '1' from this register means the reset has not finished crossing clock domains and is still pending.
30	R	STOPPED, indicates that all collection has stopped.
29	R	TRIGGERRED, indicates that a trigger has been recognized, and that the scope is counting for holdoff samples before stopping.
28	R	PRIMED, indicates that the memory has been filled, and that the scope is now waiting on a trigger.
27	R/W	MANUAL, set to invoke a manual trigger.
26	R/W	DISABLE, set to disable the internal trigger. The scope may still be triggered manually.
25	R	RZERO, this will be true whenever the scope's internal address register is pointed at the beginning of the memory.
20–24	R	LGMEMLEN, the base two logarithm of the memory length. Thus, the memory internal to the scope is given by 1¡¡LGMEMLEN.
0–19	R/W	Unsigned holdoff

Table 3.2: Control Register

4. Clocks

5. Wishbone Datasheet

Tbl. 5.1 is required by the wishbone specification, and so it is included here. The big thing to notice

<table>
<tr><th>Description</th><th>Specification</th></tr>
<tr><td>Revision level of wishbone</td><td>WB B4 spec</td></tr>
<tr><td>Type of interface</td><td>Slave, Read/Write, pipeline</td></tr>
<tr><td>Port size</td><td>32–bit</td></tr>
<tr><td>Port granularity</td><td>32–bit</td></tr>
<tr><td>Maximum Operand Size</td><td>32–bit</td></tr>
<tr><td>Data transfer ordering</td><td>(Irrelevant)</td></tr>
<tr><td>Clock constraints</td><td>None.</td></tr>
<tr><td>Signal Names</td><td><table>
<tr><th>Signal Name</th><th>Wishbone Equivalent</th></tr>
<tr><td>i_wb_clk</td><td>CLK_I</td></tr>
<tr><td>i_wb_cyc</td><td>CYC_I</td></tr>
<tr><td>i_wb_stb</td><td>STB_I</td></tr>
<tr><td>i_wb_we</td><td>WE_I</td></tr>
<tr><td>i_wb_addr</td><td>ADR_I</td></tr>
<tr><td>i_wb_data</td><td>DAT_I</td></tr>
<tr><td>o_wb_ack</td><td>ACK_O</td></tr>
<tr><td>o_wb_stall</td><td>STALL_O</td></tr>
<tr><td>o_wb_data</td><td>DAT_O</td></tr>
</table></td></tr>
</table>

Table 5.1: Wishbone Datasheet

is that this core acts as a wishbone slave, and that all accesses to the wishbone scope registers become 32–bit reads and writes to this interface.

6.

IO Ports

The ports are listed in Table. 6.1.

Port	Width	Direction	Description
i_clk	1	Input	
i_ce	1	Input	Clock Enable. Set this high to clock data in and out.
i_trigger	1	Input	
i_data	32	Input	
i_wb_clk	1	Input	The clock that the wishbone interface runs on.
i_wb_cyc	1	Input	Indicates a wishbone bus cycle is active when high.
i_wb_stb	1	Input	Indicates a wishbone bus cycle for this peripheral when high.
i_wb_we	1	Input	Write enable, allows configuring the part.
i_wb_addr	1	Input	A single address line, set to zero to access the configuration and control regiseter, to one to access the data register.
i_wb_data	32	Input	Data used when writing to the control register, ignored otherwise.
o_wb_ack	1	Output	Wishbone acknowledgement. This line will go high on the clock after any wishbone access, as long as the wishbone `i_wb_cyc` line remains high (i.e., no ack's if you terminate the cycle early).
o_wb_stall	1	Output	Required by the wishbone spec, but always set to zero in this implementation.
o_wb_data	32	Output	Values read, either control or data, headed back to the wishbone bus.

Table 6.1: List of IO ports