A VHDL 8254 Timer core

An
www.OpenCores.org
Project

hlefevre@opencores.org

Revision History

Revision	Date	Author	Description
0.1	3 Aug 2008	H LeFevre	Initial Release of source files
0.5	4 Aug 2008	H LeFevre	Add info about Timer Operation
1.0	9 Aug 2008	H LeFevre	Finished initial version of document

Table of Contents

1 Introduction

This core is designed to be a compatible with the Intel 8254 Timer.

Some differences:

The Processor interface is synchronous.

1.1 Purpose

- Educational – to learn more about SOC design (at lest for me…)
- To provide another example how to make use of the GH VHDL Standard Parts Library.

1.2 The VHDL 8254 Timer core License

2 The Timer Core

Until the document is done (and perhaps later as well), those interested in understanding the Timer better should consult the data sheets for the Intel 8254.

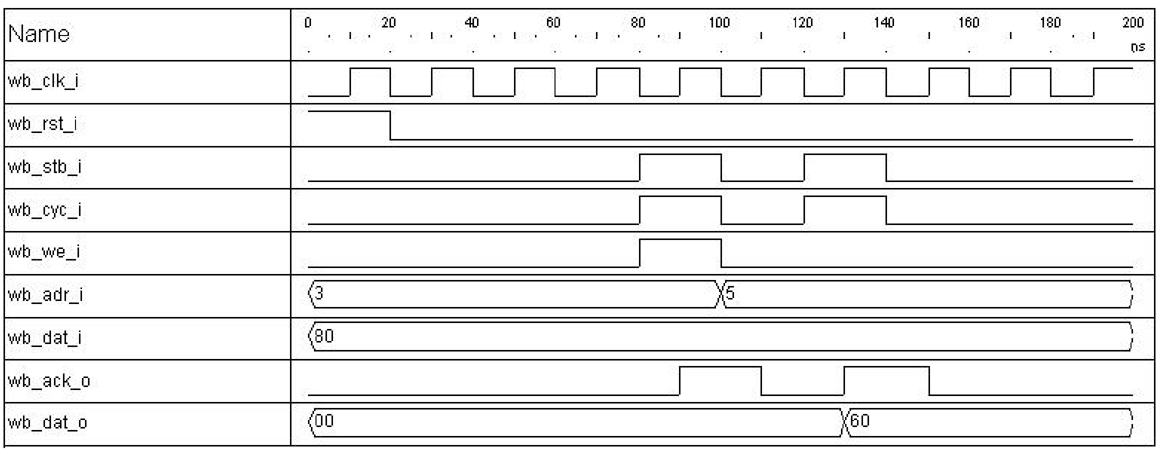
The gh_vhdl_library (another www.opencores.org project) is used extensively in this project, and needs to be downloaded separately.

The Timer core includes three identical timers. By default, the interface clock is expected to be asynchronous with the clocks used by the counters. Double clock buffering is used for the control signals crossing the clock domains. There are Boolean generics, independent for each counter, to bypass the double buffering if the same clock is used for the interface and a counter, or if a synchronous clock (different frequency, but with time aligned rising edges) is used.

2.1 The Wishbone Processor Interface

A Wishbone Interface is used to by one version of the core. File name: gh_timer_8254_wb.vhd

I/O		Function
wb_clk_i	I	Clock, primary clock used in core – not used for the baud rate generator, the Tx module, or the Rx module
wb_rst_i	I	Asynchronous Reset, active high
wb_stb_i	I	Data strobe
wb_cyc_i	I	Cycle in process
wb_we_i	I	Write enable
wb_adr_i (1 downto 0)	I	Address bus
wb_dat_i (7 downto 0)	I	Input data bus
wb_ack_o	O	Data transfer acknowledge
wb_dat_o (7 downto 0)	O	Output Data bus

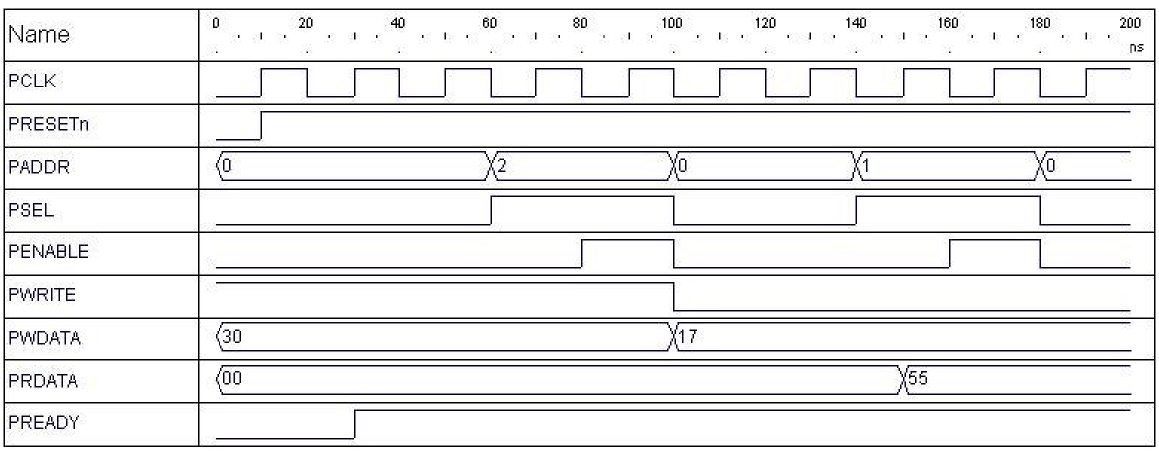


A Write cycle, followed by a Read cycle. The output data (wb_dat_o) is valid when wb_ack_o is active, and it is held until the next data transfer cycle.

2.2 AMBA APB Interface

This AMBA APB bus 3.0 is used by one version of the core. This is the Peripheral bus used by the ARM family of processors. File name: gh_timer_8254_AMBA_APB.vhd

I/O		Function
PCLK	I	Clock, primary clock used in core – not used for the baud rate generator, the Tx module, or the Rx module
PRESETn	I	Asynchronous Reset, active low
PSEL	I	Peripheral Select
PENABLE	I	Peripheral transfer Enable
PWRITE	I	Peripheral Write / READn Control
PADDR (1 downto 0)	I	Peripheral Address bus
PWDATA (7 downto 0)	I	Peripheral Data Write bus
PRDATA (7 downto 0)	O	Peripheral Data Read bus



A write cycle, followed by a (no wait state) read cycle.

2.3 Non Wishbone/AMBA APB Signals

The following signals are not part of a standard bus, but are part of the Timer core.

I/O		Function
clk0, clk1, clk2	I	Clock – each of the three counters has it's own clock
gate0, gate1, gate2	I	Gate – each of the three counters has it's own gate
out0, out1, out2	O	Out – each of the three counter has it's own out signal

2.4 Timer Operation

The 8254 timer has three identical programmable down counters. The operation of each counter is based upon how it is programmed. Prior to being programmed, operation is undefined.

For each counter, the control word must be written before the initial count is written. The initial count must follow the count format as specified by the control word (MSB only, LSB only, LSB and then MSB).

2.4.1 Address Map

Address	R/W	Function
0	R/W	Read counter 0
1	R/W	Read counter 1
2	R/W	Read counter 2
3	W	Control Word write

2.4.2 Control Word Format

7	6	5	4	3	2	1	0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC – Select Counter
00 Select Counter 0
01 Select Counter 1
10 Select Counter 2
11 Read Back Command (see read operations)

RW – Read/Write Control
00 Counter Latch Command (see read operations)
01 Read/Write LSB only
10 Read/Write MSB only
11 Read/Write LSB, then MSB

M – Mode
000 Mode 0
001 Mode 1
x10 Mode 2
x11 Mode 3
100 Mode 4
101 Mode 5

BCD
0 binary counter (16 bits)
1 Binary Coded Decimal Counter (4 decades)

2.4.3 Read Operations

There are three ways to read a counter:

1. A simple read (just read the counter – if counter changes during read, may have an invalid value)
2. The Counter Latch Command
3. The Read Back Command

The Counter Latch Command and the Read Back Command start with a write to the control word.

Data Word for Counter Latch Command

7	6	5	4	3	2	1	0
SC1	SC0	0	0	x	x	x	x

<u>SC – Select Counter</u>

00 Select Counter 0
01 Select Counter 1
10 Select Counter 2
11 Read Back Command

Data Word for Read Back Command

7	6	5	4	3	2	1	0
1	1	count	status	CNT 2	CNT 1	CNT 0	0

If count = 0, then a 1 in any of the CNT bits will latch that counter to be read
If Status = 0, then a 1 in any of the CNT bits will latch status for that counter

Status Word

7	6	5	4	3	2	1	0
out	Null Count	RW1	RW0	M2	M1	M0	BCD

D7 is state of counters out signal
D6 is NULL Count (0 = count is available for reading)
D5-D0 is Counter programmed mode

Counter Reads follow the format of Counter Write format. If a Read Back Command requests both status and count data, the first read is the Status Word. The counter data is either LSB only, MSB only, or LSB followed by MSB.

2.4.4 Counter Modes

The counters in the gh_timer_8254 uses only the rising edge of its clock – unlike the Intel original. A Trigger is defined as a rising edge of the counter's Gate input (as sampled by the counters clock). Note: for the non- periodic modes, when the counts reach zero, they will roll over to max value (xFFFF for binary, 9999 for BCD) and continue counting.

Mode 0

After the control word is written, OUT is initially low, and will remain low until the Counter reaches 0. OUT then goes high, and remains high until a new count or control word is written.

Gate = 1 enables counting, Gate = 0 disables counter (has no effect on OUT)

Mode 1

Once the control word (OUT will be high) and count value are written, a trigger will cause OUT to go low on the next clock pulse. Once the counter reaches zero, OUT will go high until the clock edge after the next trigger.

Mode 2

A periodic mode – operates as a divide-by-N counter. OUT will high until the counter reaches a count of 1. OUT will go low for one clock period, and will go high again as the counter reloads with the initial count.

Gate = 1 enables counting, Gate = 0 disables counter If low, OUT will go high. A trigger will reload the initial count.

Mode 3

Similar to Mode 2, except that the output has a 50 % duty cycle (for an odd count, the duty cycle will be one clock period off of 50 %).

Mode 4

Out will be initially high. When the initial count reaches zero, OUT will go low for one clock period, and go high again. Counting is "triggered" by writing the initial count.

Gate = 1 enables counting, Gate = 0 disables counter (has no effect on OUT)

Mode 5

Out will be initially high. When the initial count reaches zero, OUT will go low for one clock period, and go high again.

Gate = 1 enables counting, Gate = 0 disables counter (has no effect on OUT). A trigger will reload the initial count.

3 Custom Core Sub-modules

The gh_timer_8254 is an example of Modular design using VHDL. It is a combination of custom modules and parts from the gh_vhdl_lib library (which is another www.opencors.org project – which must be downloaded separately).

The top level includes the bus interface and the instantiation of three counter control modules (gh_counter_control.vhd). It should be noted that each of the counter control modules is designed to be counter 0 – the address and data bits are remapped, as required, at the top level.

The gh_vhdl_lib parts are described in the documentation for the library.

3.1 gh_counter_control.vhd

I/O		Function
clk_i	I	Interface clock
rst	I	Asynchronous Reset, active high
ics	I	Chip select
dwr	I	Write control
iA(1 downto 0)	I	Address bus
iD(7 downto 0)	I	Data bus
Dat_o (7 downto 0)	O	Data Read bus
clk	I	Counter clock
gate	I	Counter gate
rd_busy	O	Busy signal, used to delay the read data acknowledge (needed for when the counter clock is much slower than the interface clock)
Cout	O	Counter Out signal

The Counter control module holds the counter control registers. It also is where the control signals cross the clock domains (using the gh_edge_det_XCD_t.vhd module).

The control signals for the 16 bit binary/BCD counter are in this module.

3.2 gh_counter_down_16b_bb.vhd

This is the 16 bit binary/BCD down counter.

I/O		Function
clk	I	clock
rst	I	Asynchronous Reset, active high
BCD	I	1 = BCD counter, 0 = binary counter
CE	I	Count enable
LD	I	Load control
M_CMD	I	Mode command load signal
MODE(2 downto 0)	I	Counter Mode
DI(15 downto 0)	I	Count load value
Cout	O	Counter out signal
NULL_C	O	Counter Null bit
DO(15 downto 0)	O	Count out

The 16 bit counter is built will four, 4 bit binary/BCD down counters.

3.3 gh_counter_down_4b_bb

This is a 4 bit binary/BCD down counter.

I/O		Function
clk	I	clock
rst	I	Asynchronous Reset, active high
LOAD	I	Load control
BCD_EN	I	1 = BCD counter, 0 = binary counter
CE	I	Count enable
D(3 downto 0)	I	Count load value
TC	O	Thermal count for counter
Q(3 downto 0)	O	Count out

3.4 gh_div2_bcd_4digits.vhd

This module divides a 4 digit BCD number in half. This allows a BCD counter to have a 50 % duty cycle on the out signal.

I/O		Function
BCD_IN	I	Input data
BCD_OUT	O	Output data

3.5 gh_div_bcd2.vhd

This module will divide a single BCD digit by 2 – includes carry in/out so that it may be cascaded.

I/O		Function
CDi	I	carry down input
iBCD	I	Input data
oBCD	O	Output data
CDo	O	carry down out

3.6 gh_edge_det_XCD_t.vhd

I/O		Function
iclk	I	Input data clock
oclk	I	Output data closk
rst	I	Asynchronous Reset, active high
D	I	Input data signal (expected to be synchronous with iclk)
re	O	Rising edge on input data signal detected – synchronous with oclk

This module is similar to the gh_vhdl_lib part with a similar name. The basic difference is this module has two Boolean generics (same_clk, sync_clk). By default, both generics are false. When the two generics are false, the data signal is double registered as it crosses the clock domains – recommended when the interface clock is asynchronous with the counter clock.

If the same clock is used for the interface, as well as the counter, setting the same_clk generic to true, will bypass the double buffering, allowing the counter to respond to commands faster.

If the clocks are synchronous (the rising edges of the two clocks are aligned in time), but at different frequencies, the sync_clk generic may be set true.

The generics are set at the top level.

4 Core Notes

It could be argued the modules in the core should have a prefix of hal_. But, since it has a close relationship with the GH VHDL Standard Parts Library, it seems reasonable to use the gh_ prefix, and save the ego trip for better use elsewhere.