DS1WM Release Notes

Rev 101308

Introduction:

This release contains VHDL/Verilog source code for the DS1WM RTL, and Verilog source code for the testbench.

The document discusses the following topics:

- DS1WM source code and compile notes
- Testbench architecture and source code
- DS1WM programming model
- DS1WM functional verification testcases

DS1WM source code and compile notes

The DS1WM RTL VHDL source code is located under:

delivery/design/vhdl_src/ds1wm

The DS1WM RTL Verilog source code is located under:

delivery/design/verilog_src/ds1wm

A simulation flow is not included, however, the VHDL/Verilog should be compiled in the following order:

1. one_wire_io.vhd or one_wire_io.v
2. clk_prescaler.vhd or clk_prescaler.v
3. one_wire_interface.vhd or one_wire_interface.v
4. onewiremaster.vhd or onewiremaster.v
5. ds1wm.vhd or ds1wm.v

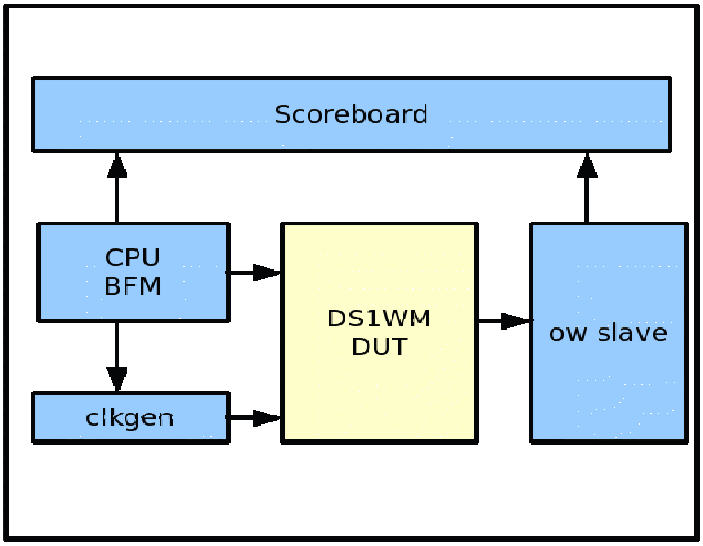

Illustration 1: Testbench Architecture

The testbench presented as Illustration 1 is comprised of the following source code file:

delivery/verification/verilog_src/testbench/tb_ds1wm/

tb_ds1wm.v
tc_ds1wm.v

delivery/verification/verilog_src/testbench/cpu_bfm/

cpu_bfm.v

delivery/verification/verilog_src/testbench/ow_slave/

ow_slave.v
cmd_ctrl.v
iox.v

delivery/verification/verilog_src/testbench/clkgen/

clkgen.v

delivery/verification/verilog_src/testbench/scoreboard/

scoreboard.v

The **stimulus.inc** file contains the test stimulus. It is included by the tc_ds1wm block, and should not be explicitly compiled.

DS1WM Functional Verification

Each DS1WM transaction consists of the following instruction sequence:

1. 1-Wire Reset
2. A device addressing instruction
3. A data access instruction

A device addressing instruction may one of the following:

- READ_ROM
- MATCH_ROM
- SKIP_ROM
- OD_SKIP_ROM
- OD_MATCH_ROM

A data access instruction may be one of the following:

- WRITE_SP
- READ_SP
- COPY_SP (command only)
- READ_MEM (command only)

This version of the testbench verifies that the slave correctly recognizes 1-wire commands issued by the testbench. Scratch Pad data integrity is alsochecked.

There are 3 fundamental 1-Wire transactions:

1. 1-wire reset
2. 1-wire write cycle
3. 1-wire read cyle

The DS1WM sets specific status bits in it's Interrupt Register to indicate that a 1-Wire transaction has completed. Interrupts are raised if the corresponding Interrupt Enable bit has be set.

The testbench uses these interrupts to control instruction sequencing.

Test Cases

These testcases each come with a run script for simulation with ncsim.

For usage, please see the README file in the 'tests' directory.

Testcases are located under delivery/verification/verilog_src/tests.

Here is a brief description of each testcase:

cmd_recognition –

> The command recognition checks all the basic commands like skip rom, match rom, search rom, write scratch pad, read scratch pad, and overdrive skip/overdrive match. It also checks one wire reset, overdrive reset, the strong pullup enable, active high/low interrupts, and all the divisor clock frequencies. This configuration contains only one slave device on the one wire network.

multi_ow_network –

> The multiple one wire network checks the Search Rom Accelerator in the ds1wm. This configuration contains four slave devices on the one wire network. The results show and verify that the correct four ROMIDs were found by using the Search Rom Accelerator.

single_search_rom –

> The single one wire network checks the Search Rom Accelerator in the ds1wm. This configuration contains only a single slave on the one wire network. The results show and verify that the correct single ROMID was found by the Search Rom Accelerator.

scratchpad_integrity –

> The scratchpad memory in the slave device is written too and read back using the WRITE_SP and READ_SP commands. The data being written is random. The data is stored at the bus functional model while also being written to the ds1wm block and sent across to the one wire slave device. The data written into the slave device is read back using ds1wm and compared to the previously stored value in the bus functional model.