Wiki »

EXERCISE: MM-BUS FUNCTIONAL TEST BENCH

Exercise Goals

This exercise will through simulation of realistic interfaces introduce you to:

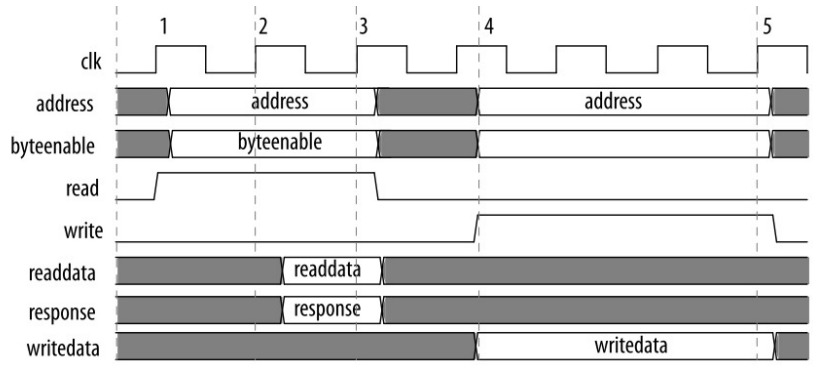
- Creation of sequential test benches in the ModelSim IDE
- Creation of stimuli procedures and response monitors
- Create of non-syntheziable VHDL code
- MM-Bus Interface Simulation

Prerequisites

- Quartus and ModelSim IntelFPGA software must be installed and working.
- Knowledge about response monitors, stimuli procedures and configurations
- Template code from the repository (exercise_seq_tb_register, see repository tab)

Introduction

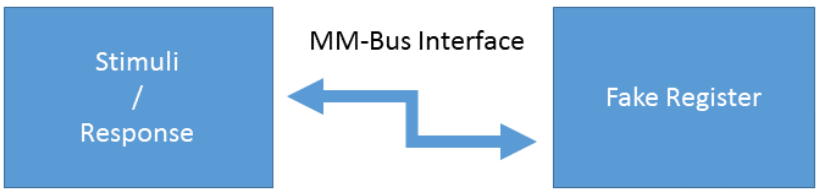
The ARM- and Nios processors have a memory mapped bus interface, that we can connect devices to. In this exercise, a register bank with this interface is given, and it will be your job to create a test bench that can provide the correct signalling to write- and read from it. The signalling is described in the Avalon Interface Specification (Memory Mapped Interface).



(write wait time: 2, read wait time: 1)

Exercise Steps

As mentioned, a fake_register.vhd is delivered and you must emulate the memory mapped bus interface to access it:



1 Inspect the existing code

A) Create a project in ModelSim and include the file supplied.

B) Inspect the fake_register.vhd file. Its code cannot be compiled to actual gates in the FPGA, as it uses ex. wait for xx ns (how would you want the compiler to implement that?)

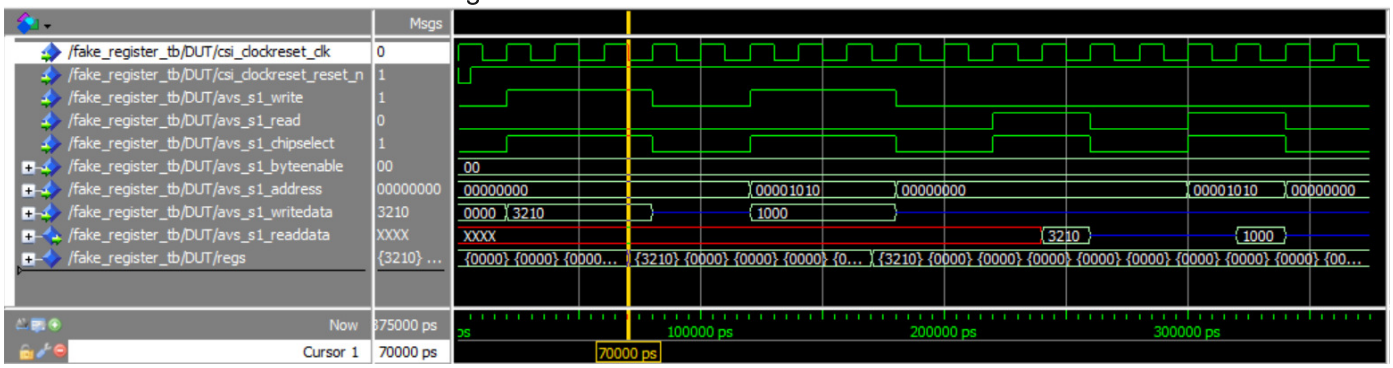
2 Create a Test Bench

A) Create a test bench for the fake_register.vhd.

B) Create stimuli signals to write to the fake_register. See the Avalon Interface Specification around figure 3-4 for more details, especially how read_wait_time and write_wait time works.

C) Create signals for reading back the value and verify that you can read the value back.

Your simulation should look something like:



3 Create a Stimuli Package

A) Create a new vhd file, mm_bus_sim.vhd

B) Create a package header with the following interface (for write):

```
package mm_bus_sim is

  procedure mm_write (
    constant write_wait_time : in  integer;
    constant wr_addr         : in  std_logic_vector(7 downto 0);
    constant wr_data         : in  std_logic_vector(15 downto 0);
    signal avs_s1_clk        : in  std_logic;
    signal avs_s1_cs         : out std_logic;
    signal avs_s1_write      : out std_logic;
    signal avs_s1_byteenable  : out std_logic_vector(1 downto 0); -- 16-bit
    signal avs_s1_address    : out std_logic_vector(7 downto 0);
    signal avs_s1_writedata  : out std_logic_vector(15 downto 0));

end mm_bus_sim;
```

C) Copy your (write) code from 2 into the body section. You may have to adapt it a bit.

D) Replace your (write) code in the stiuli process with a call to your new procedure and test...

E) Do B-D for read ;-)

D) (Optional) Create a response monitor that writes mm-bus transactions to a log file. (Ex. time:13123 ; wr ; addr:32432 ; data: 232432)

fake_register_block.png (4,46 KB) Peter Høgh Mikkelsen, 2017-02-07 20:50

fake_register_sim.png (107 KB) Peter Høgh Mikkelsen, 2017-02-07 20:50

mm-bus_spec.png (18,3 KB) Peter Høgh Mikkelsen, 2017-02-07 20:50