Lab 1

Creating a testbench for a memory design

Due date

Sep 3, 2024 11:59 PM CST

Table of content

Academic Integrity

The following actions are strictly prohibited and violate the honor code. The minimum penalty for plagiarism is a grade of zero and a report to the Aggie honor system office.

- Sharing your solutions with a classmate.
- Uploading assignments to external websites or tutoring websites
- Copying solutions from external websites or tutoring websites
- Copying code from a classmate or unauthorized sources and submitting it as your

Introduction

In this assignment, we will explore the specifications of a memory system. This memory works with a clock, ensuring read, write, and reset. For writing, we use the rising edge of the clock and an active write-enable signal. Reading, on the other hand, will occur under the same clock conditions, but when a read enable signal is active instead. Remember, we can't read and write at the same time. And if a reset is initiated, all memory slots promptly reset to zeros.

Design Under Test

The design under test is a memory with specifications that define its functionality. A memory operates as a storage unit capable of holding data or instructions. The memory has the following specifications.

- Memory is 5 bits wide, and the address range is from 0 to 31.
- Memory access is synchronous with asynchronous reset (rst_n).
- Memory Data width is 8-bit.
- Write data_in into the memory on the positive edge of the clock only when wr_en=1.
- Read from memory[addr] onto data_out bus on the positive edge of the clock only when rd_en=1.
- Read and write requests cannot be placed simultaneously
- Reset would write zeros to all the addresses

Full description

- Width and Address Range: The memory has a width of 5 bits, meaning it can store binary
- Values are up to 5 bits long. It also features an address range spanning from 0 to 31, enabling access to 32 memory locations.
- Synchronous Access and Reset: Memory access is synchronized with a clock signal, ensuring that read and write operations occur at specific times during the clock cycle. Additionally, the memory includes an asynchronous reset signal (rst_n), which, when triggered, clears the memory contents and sets all memory locations to zeros.

- Data Width: Each memory location can store 8 bits of data. Each address in the memory can hold an 8-bit binary value.
- Write Operation: The memory supports writing data into its locations. Data input (data_in) is written into the memory on the positive edge (posedge) of the clock signal, but only when the write enable signal (wr_en) is active (1).
- Read Operation: Reading data from the memory is also possible. When the read enable signal (rd_en) is active (1) and the clock signal's positive edge occurs, the data stored at the memory location specified by the address (addr) is output onto a data output bus (data_out).
- Simultaneous Requests: The memory system enforces a rule that read and write requests cannot be initiated simultaneously. This ensures orderly and controlled access to the memory.

Environment Setup

1. Log in to the Linux server (Refer to the steps explained in Lab 0 if needed). Use a secure shell to remote login to one of the following server hostnames.
    - olympus.ece.tamu.edu

Start SSH with X11 Forwarding enabled (-X). X11 forwarding allows you to run graphical applications such as Cadence tools from the remote server and have their graphical user interfaces (GUIs) displayed on your local machine.

```
ssh -Y <netid>@olympus.ece.tamu.edu
load-csce-616
```

2. Accept the assignment's repository on GitHub Classroom: https://classroom.github.com/a/sRdwJHon .
3. Clone your lab repository on the Linux server.
4. cd in the lab directory

```
cd lab-1
```

5. Change the directory to the design testbench

```
cd work/
```

6. Open file tb/simple_mem_tb.sv and complete all the TO DOs. Hints are provided.

Suggestion: Complete the mem_read and mem_write task code and call these tasks in the main initial begin block.

The specifications highlight all the signals/ports to be defined in bold.

7. Source setup bash from sim directory; then compile and simulate the design.

```
source ../../setupX.bash
xrun -f run.f
```

8. Debug eventual errors showing during compilation.
9. There is a bug in the HDL design, which you are expected to find out with your stimulus. Try to put debug statements to print expected vs observed values.

Deliverables

Commit and push all your changes to your remote repository.

Your repository must include the following:

1. design directory
2. sim directory
3. tb directory containing the updated testbench simple_mem_tb.sv

Important note: To get full credit, you must upload all the required files and directories and strictly name your files according to the requirements.