TFE4171 - Design of Digital Systems 2
Laboratory 1

Norwegian University of Science and Technology
Department of Electronic Systems
Spring 2022

1 Submission

DEADLINE: MON**DAY 14th** MARCH **23:59**
Submit on Blackboard under Labs/Lab 1/Lab 1 SUBMISSION. Upload in a .zip file the following:

- A **report** which shows:
  - **Results** achieved
  - **Discussion** of results
  - **Explanation** of how results are achieved
  - **Answers to the questions** asked in the exercise
- All **codes** written by the group. Codes must be ready to run without any errors.

Only one group member needs to submit. The work must have been performed only by the group members. Groups may get questions in the lab about their work after submission on Blackboard. Plagiarism will be reported to the coordinator of the course.
Lab work counts 20% of the final grade, and it is distributed as follows:

- 4 exercises (5% each)

All group members get the same grade.

When you are finished with the codes, you may explore different alternatives to download the codes in the server into your local computer, e.g., sftp, scp. After logging in using putty, zip the ex-1 directory you created in the first lab as follows:
zip -r ex-1.zip ex-1
Use ls command to check that a .zip file was created. Next steps are to download the .zip file from the remote server to your local computer.
For macOS users (following steps have only been tested in macOS): open a **Terminal** and sftp your user in the server as follows:

sftp user_name@aurora.iet.ntnu.no
After entering your account password, download the file typing the following command also in the terminal:

`get ex-1.zip /Users/YourUserNameOnMac/Desktop/`
Now the .zip file should be on the desktop of your local computer.
If you want to upload any file from your local computer to the server with sftp, use the command `put`. For example, to upload a file stored in the desktop of your local computer:
`put /Users/YourUserNameOnMac/Desktop/file_name.txt`
For Windows or Ubuntu users: follow for example this tutorial.

2 Connection to the server

For communication with the linux servers **putty** can be used for simple text interface. There are other alternatives, which might not be necessary at least for this exercise, such as **XWin32** and **CygWin** to get a graphical interface.
For Windows and Ubuntu users: downloading and installation of putty should be straightforward.
For macOS users: getting putty up and running might take more time in macOS. Follow these steps in this tutorial. If after completing the steps indicated in the tutorial, putty cannot get correctly executed, then try the following steps. First, download and install **XQuartz**. Once installed, launch XQuartz and then make right click on the X icon. Click on *Applications*, and choose *terminal*. Type *putty*, and after that a putty window should have been open.

Once putty is up and running and you are also connected to NTNU's network (use VPN if you are out of campus following these instructions), connect to the server **aurora.iet.ntnu.no** as shown in Fig.1, and use the username and password provided to your group. After logging in you can change the password of your account with the command `passwd`. For future exercises, servers **venus.iet.ntnu.no** and **jupiter.iet.ntnu.no** might be used.

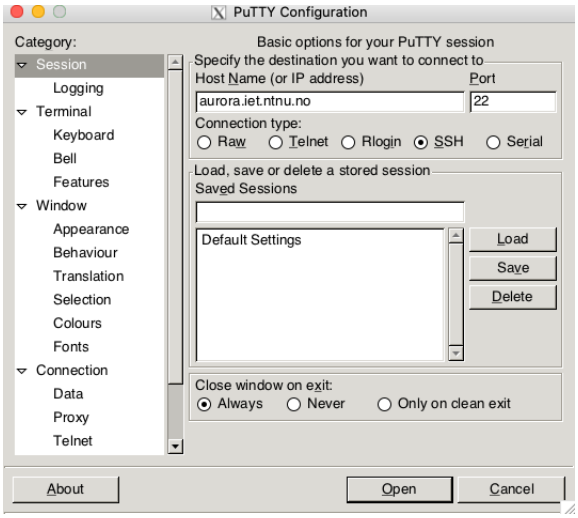

Figure 1: Connection to aurora.iet.ntnu.no using putty

3 Learning outcome

Testing out binding of assertions to modules, simple assertions with and without implication, how overlap in SVA operators work. Then you will test your basic knowledge through instrumenting two simple examples with a set of assertions for given conditions, one FIFO and one counter and comparing with expected results. You will also be using concurrent assertions to check requirements for protocol verification and also functional verification of state machine based controllers.

4 Reference literature

[1, 2]

5 Labs

5.1 Lab 1 - Binding design/property module

First, set up your user account for using *Mentor Questa* tools. For this you need to copy a *.bashrc* file which defines necessary environment variables:
`cp -r /home/courses/desdigsys2/2022s/dd2master/.bashrc .`
Execute the commands in this file by:
`source .bashrc`
You only need to do this once, next time you log in this file will be automatically executed and the environment set up as required.
Now create a directory for this exercise 1 in your home directory:
`mkdir ex-1`
Enter this directory by:
`cd ex-1`
Now copy the codes for lab 1 executing this command:
`cp -r /home/courses/desdigsys2/2022s/dd2master/ex-1/lab1 .`
Enter the lab 1 directory:
`cd lab1`
Use following command to see the files you will be using for this lab:
`ls -a`
For opening the files you can use the editor you prefer: *vi*, *vim*, *gedit*, *nano*, *emacs*, etc. If interested in *vim*, follow this tutorial.

For the rest of this lab 1 you need to follow the instructions given in the PDF *SVALAB_LINUX_LAB1*, which can be found on Blackboard and also in the lab1 directory. The PDF states that there are some questions to be answered. Ignore these questions and ONLY ANSWER THE QUESTIONS BELOW (make sure to answer all of them to get full points):

1. Follow task 2 in the PDF to bind the property module with the DUT.
   - Show how you bind the modules.
2. Follow task 3 in the PDF about the *no_implication*. Remember to run it doing: *./run_no_implication*. Once you get the FAILS and PASSES, you can compare if what you are getting is correct by having a look at the solution logs which are available for all the labs. Execute the following commands:
   `ls -a`
   `cd .solution`
   `ls`
   The logs you will see in the solution directory are the reference so that you know which logs you

are expected to get when you run your properties. For this task, after you have checked you are getting the same log for your no-implication property (if not, check the property again), answer the following questions:

- Would you get the same execution results when running *no_implication* if when binding the modules you use signals *sys_clk*, *sys_req*, and *sys_gnt* instead of using *clk*, *req*, and *gnt*?
- Why is there neither FAIL nor PASS at time 30?
- Why is there a FAIL at time 50?
- Why is there a FAIL and a PASS at time 70?
- Why are there 2 FAILS at time 130?

3. Follow task 4 in the PDF about the *implication*. Once you get the same FAILS and PASSES as in the respective solution log, answer the following:
   - What is a vacuous pass?
   - Most of the passes are vacuous, so which passes are NOT vacuous? State the times in which the non-vacuous passes occur, and explain why they occur in these specific clock cycles.
   - Why are there 2 PASSES at time 70?
   - Why is there a PASS and a FAIL at time 130?
4. Follow task 5 in the PDF. After checking you get the same as in the respective solution log, answer the following:
   - Why do you only get in total 1 PASS and 1 FAIL? Why you don't get more PASSES and FAILS in the other clock cycles?

5.2 Lab 2 - Overlap and non-overlap operators

Go back to your *ex-1* directory and copy the codes for lab 2 executing this command:
`cp -r /home/courses/desdigsys2/2022s/dd2master/ex-1/lab2 .`
Enter the directory and explore its files by using the commands `cd` and `ls`.
For this lab 2 you need to follow the instructions given in the PDF
SVALAB_LINUX_LAB2, which can be found on Blackboard and also in the lab2 directory. The PDF states that there are some questions to be answered. Ignore these questions and ONLY ANSWER THE QUESTIONS BELOW (make sure to answer all of them to get full points):

1. Follow task 2 in the PDF to explore how the overlap implication operator works. Once you get the same FAILS and PASSES as in the solution log shown in Fig.2 (this log in particular is not available in the server), answer the following questions:
   - Why does the property FAIL at 30?
   - If both *cstart* ad *req* are 1 at 50, why we don't get a PASS at 50?
   - Why does the property PASS at 90?
   - Why does the property PASS at 150?
   - Why does the property FAIL at 170?
   - Why does the property FAIL at 190?
2. Follow task 3 in the PDF to explore how the non-overlap implication operator works. Once you get the same FAILS and PASSES as in the solution log shown in Fig.3 (this log in particular is not available in the server), answer the following questions:

- If *cstart* is 1 and *req* is 0 at 30, why we don't get a FAIL at 30?
- Why does the property FAIL at 70?
- Why does the property PASS at 90?
- Why does the property FAIL at 170?
- Why does the property FAIL at 190?
- Why does the property PASS at 210?

```
 10  clk=1 cstart=0 req=0 gnt=0
 30  clk=1 cstart=1 req=0 gnt=0
 30             test_overlap_nonoverlap.reqGnt FAIL
 50  clk=1 cstart=1 req=1 gnt=0
 70  clk=1 cstart=0 req=0 gnt=0
 90  clk=1 cstart=0 req=0 gnt=1
 90             test_overlap_nonoverlap.creqGnt PASS
110  clk=1 cstart=1 req=1 gnt=0
130  clk=1 cstart=1 req=1 gnt=0
150  clk=1 cstart=1 req=1 gnt=1
150             test_overlap_nonoverlap.creqGnt PASS
170  clk=1 cstart=0 req=1 gnt=0
170             test_overlap_nonoverlap.reqGnt FAIL
190  clk=1 cstart=0 req=0 gnt=0
190             test_overlap_nonoverlap.reqGnt FAIL
210  clk=1 cstart=0 req=0 gnt=1
```

Figure 2: Solution log for overlap implication operator

```
 10  clk=1 cstart=0 req=0 gnt=0
 30  clk=1 cstart=1 req=0 gnt=0
 50  clk=1 cstart=1 req=1 gnt=0
 70  clk=1 cstart=0 req=0 gnt=0
 70              test_overlap_nonoverlap.reqGnt FAIL
 90  clk=1 cstart=0 req=0 gnt=1
 90              test_overlap_nonoverlap.creqGnt PASS
110  clk=1 cstart=1 req=1 gnt=0
130  clk=1 cstart=1 req=1 gnt=0
150  clk=1 cstart=1 req=1 gnt=1
170  clk=1 cstart=0 req=1 gnt=0
170             test_overlap_nonoverlap.reqGnt FAIL
190  clk=1 cstart=0 req=0 gnt=0
190             test_overlap_nonoverlap.reqGnt FAIL
210  clk=1 cstart=0 req=0 gnt=1
210             test_overlap_nonoverlap.creqGnt PASS
```

Figure 3: Solution log for non-overlap implication operator

5.3 Lab 3 - FIFO

Go back to your *ex-1* directory and copy the codes for lab 3 executing this command:
`cp -r /home/courses/desdigsys2/2022s/dd2master/ex-1/lab3 .`
Enter the directory and explore its files by using the commands `cd` and `ls -a`.
For this lab 3 you need to follow the instructions given in the PDF
SVALAB_LINUX_LAB3, which can be found on Blackboard and also in the lab3 directory.

1. Follow task 2 in the PDF to explore and understand how the FIFO works. Remember that no assertions are active yet.
2. Follow from task 3 until 9 by adding one by one your properties according to the indications given in both the PDF and also in the comments in the properties code. You only need to remove the DUMMY code and replace it by your property code. Properties are already asserted in the code. Run the check for each property and compare the log you get with the respective solution log. If you do not get the same log, change your property code until you are able to get the same log. It is important that you understand the changes you may have made to your property to get the same log. For each of the 7 checks answer the following questions:
   - Which property did you use? Explain it.
   - Explain the FAILS (or WARNINGS) you get in your log.

5.4 Lab 4 - Counter

Go back to your *ex-1* directory and copy the codes for lab 4 executing this command:
`cp -r /home/courses/desdigsys2/2022s/dd2master/ex-1/lab4 .`
Enter the directory and explore its files by using the commands `cd` and `ls -a`.
For this lab 4 you need to follow the instructions given in the PDF
SVALAB_LINUX_LAB4, which can be found on Blackboard and also in the lab4 directory.

1. Follow task 2 in the PDF to explore an understand how the counter works. Remember that no assertions are active yet.
2. Follow from task 3 until 5 by adding one by one your properties according to the indications given in both the PDF and also in the comments in the properties code. You only need to remove the DUMMY code and replace it by your property code. Properties are already asserted in the code. Run the check for each property and compare the log you get with the respective solution log. If you do not get the same log, change your property code until you are able to get the same log. It is important that you understand the changes you may have made to your property to get the same log. For each of the 3 checks answer the following questions:
   - Which property did you use? Explain it.
   - Explain the FAILS you get in your log.

5.5 Lab 5 - Bus protocol

Go back to your *ex-1* directory and copy the codes for lab 5 executing this command:
`cp -r /home/courses/desdigsys2/2022s/dd2master/ex-1/lab5 .`
Enter the directory and explore its files by using the commands `cd` and `ls -a`.
For this lab 5 you need to follow the instructions given in the PDF
SVALAB_LINUX_LAB5, which can be found on Blackboard and also in the lab5 directory.

1. Follow task 2 in the PDF to explore and understand how the BUS protocol works. Remember that no assertions are active yet.
2. Follow from task 3 until 5 by adding one by one your properties according to the indications given in both the PDF and also in the comments in the properties code. You only need to remove the DUMMY code and replace it by your property code. Properties are already asserted in the code. Run the check for each property and compare the log you get with the respective solution log. If you do not get the same log, change your property code until you are able to get the same log. It is important that you understand the changes you may have made to your property to get the same log. For each of the 3 checks (*check1*, *check2*, and *check3*) answer the following questions:
   - Which property did you use? Explain it.
   - Explain the FAILS you get in your log.

   After running the *checkall*:
   - Explain the FAILS you get in your log.

5.6 Lab 6 - PCI read protocol

Go back to your *ex-1* directory and copy the codes for lab 6 executing this command:
`cp -r /home/courses/desdigsys2/2022s/dd2master/ex-1/lab6 .`
Enter the directory and explore its files by using the commands `cd` and `ls -a`.
For this lab 6 you need to follow the instructions given in the PDF
SVALAB_LINUX_LAB6, which can be found on Blackboard and also in the lab6 directory.

1. Follow from task 2 until 6 by adding one by one your properties according to the indications given in both the PDF and also in the comments in the properties code. You only need to remove the DUMMY code and replace it by your property code. Properties are already asserted in the code. Run the check for each property and compare the log you get with the respective solution log. If you do not get the same log, change your property code until you are able to get the same log. It is important that you understand the changes you may have made to your property to get the same log. For each of the 5 checks answer the following questions:
   - Which property did you use? Explain it.
   - Explain the results you get in your log.

References

[1] IEEE Standard for SystemVerilog-Unified Hardware Design, Specification, and Verification Language
[2] Ashok B. Mehta, SystemVerilog Assertions and Functional Coverage, 3rd edition, Springer, 2020.
[3] Ashok Mehta, Introduction to SystemVerilog, Springer, 2021.
[4] Eduard Cerny, Surrendra Dudani, John Havlicek, Dmitry Korchemny, et al. *SVA: the power of assertions in systemVerilog (chapters 4, 5, 6.4.3, and 10.6).* Springer, 2015.