Automatic testing of digital hardware designs

Helder Daniel

Departamento de Engenharia Electrónica e Informática
Faculdade de Ciências e Tecnologia
Universidade do Algarve

Setember 23, 2021

Abstract

Hardware reliability is as critical as software reliability. Black box testing of hardware can be preformed by sending signals to the inputs and comparing the output signals with the expected according to specifications. The reliability of hardware can be tested at the design level, before the actual hardware be created. There are applications that can be used to design digital hardware, design a digital circuit. The circuits can then be converted to and hardware definition language specification, in Verilog, VHDL or other, that can be used to generate the actual hardware. In this paper it is proposed a way to extend Mooshak programming contest manager (Leal and Silva [2003]) to accept also hardware specifications and perform black box testing on it, using a freely available, digital logic designer and circuit simulator tool: Digital (Neeman [2021]).

1 Introduction

Hardware reliability of digital circuits design can be preformed right at the design level, before the actual hardware be generated. One way to test the design is by sending digital signals to the inputs and comparing the output signals with the expected according to specifications, in a black box fashion.
Taking advantage of a software testing tool and a digital design application working together it is possible to automatize the testing of digital hardware designs and receiving back information on the testing. This way when generating the hardware specification in Verilog, VHDL or other, the design has been already tested for reliability.

In this proposed work, as testing tool it is used the Mooshak (Leal and Silva [2003]) programming contests manager. This free testing tool was designed according to International Collegiate Programming Contest rules (ICPC [2019]), to manage software programming contests, being unsuitable to test hardware designs. However it has a web portal where users can submit software to be tested and receive feedback on that testing, and can also perform basic black box testing.
As the hardware testing tool it is used a freely available, digital logic designer and circuit simulator tool: Digital (Neeman [2021]).
Combining the submission management and basic black box features of Mooshak with the hardware design testing provided by Digital simulator it is possible to automatize the testing as it is described in the next sections. These features are developed as a standalone Mooshak extension without changing the original code of Mooshak and Digital simulator.

2 Mooshak and Digital simulator testing operation

The default Mooshak's 1.x.x software testing process has 2 steps:

1. If needed, the submitted **solo source code** file is compiled or assembled first.
2. The **solo file** application is executed and tested.

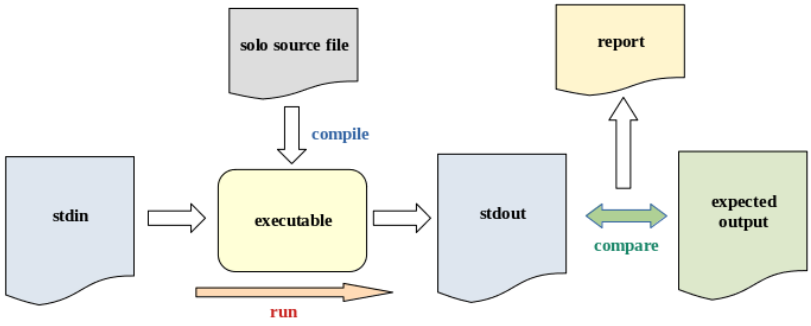

Figure 1: Mooshak default testing operation on solo files

Mooshak performs black box software testing, running several test cases on the software and generating a report (Daniel [2019]), see figure 1.

To use Mooshak together with Digital simulator hardware design testing facilities, only one test case is needed to define in Mooshak. This test case compares the output generated by Digital simulator, with an expected output where all Disital Simulator tests have be successful.
Further, no compilation step is needed, so the first step of Mooshak testing process is not needed, being Digital Simulator run in the second step.

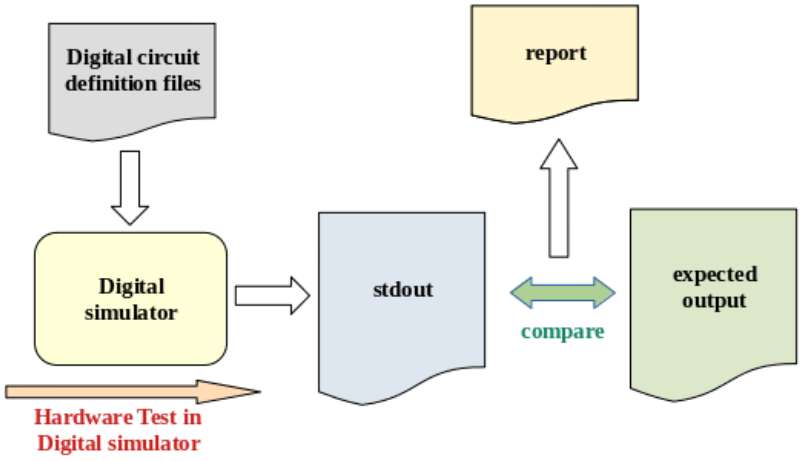

Figure 2: Mooshak and Digital simulator testing operation

3 Extending Mooshak to test digital circuits designs

Digital simulator can perform black box testing on circuits on its own. Digital signals are injected at the circuit inputs, the circuit is then simulated, being stored the output signals. After simulation the output signals is compared with the expected output signals. Figure 3 shows the testing process. The **test data** window shows the testing setup. Each line is a test case, where the input and expected output signals are specified. This testing setup supports also a small language to generate simple test cases. The **Test results** window shows the result of each test case. If background is green the test has passed.
Digital allows also the testing to be performed in the command line, where the result data are printed to the stdout.
This extension uses Mooshak to manage the submission of the circuit specification file and then sends it to Digital simulation to be tested. The stdout of Digital is redirected to be compared by Mooshak with an expected output file, that determines if all Digital testing have passed or not.

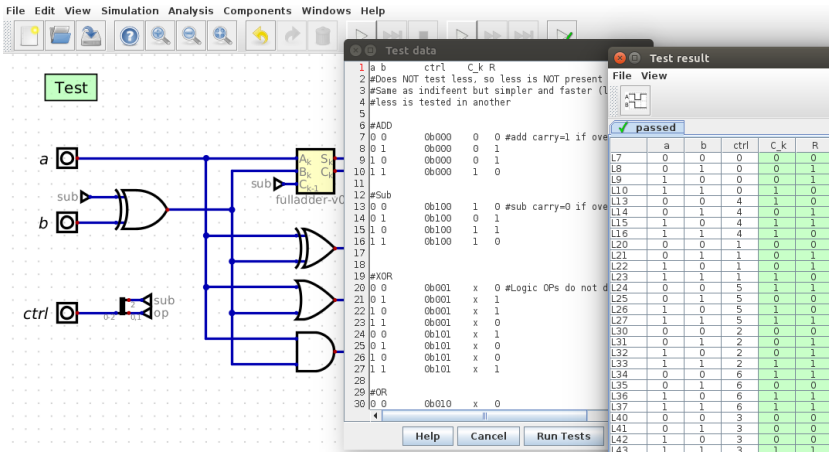

Figure 3: Digital logic simulator

Mooshak testing process first step (see figure 1) is not needed since there is no need to compile the circuit. Digital is called on the second step, being specified at language settings page on Mooshak's GUI, **"Execute"** field, figure 4.

Name	Digital circuit simulator
Extension	dig
Compiler	Digital
Version	0.25
Compile	/bin/true
Execute	/bin/bash $home/contrib/mydigital.sh $file
Data	
Fork	
Omit	
UID	

Figure 4: Mooshak language settings

At this field, is called a script to perform the Digital testing and capture Digital's stdout to be compared. This is a bash script that redirects stdin/stdout as needed and need only information of the filename of the circuit definition to be tested. Mooshak makes available some operation variables, (Daniel [2019]),

namely the filename submitted, that can be passed as argument to the script:

```
digitalext.sh $file
```

The main steps of this Mooshak extension are presented below.

Algorithm 1: Extended Mooshak client-server testing (simplified)

$folder \leftarrow unpack(submission)$;
$testCircuitWithDigital(folder)$;

Digital simulator is a Java multi threaded application. To execute RARS from Mooshak in step 2, it is needed to raise the thread limit in the Mooshak's sandbox **"safe execution environment"**. This can be configured on the overall language setting, figure 5, field **MaxExecFork**.

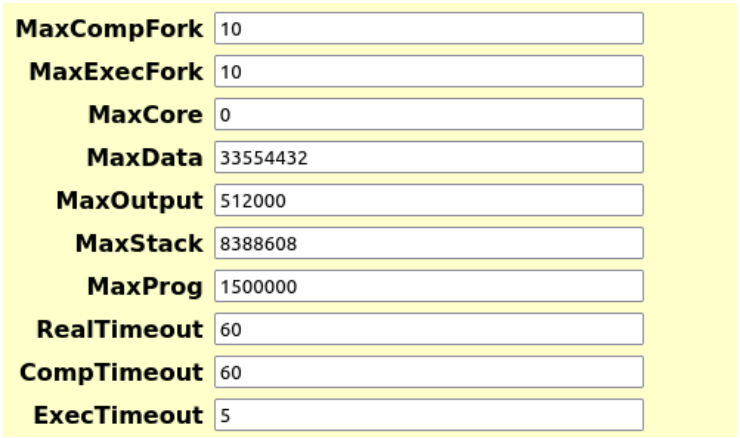

Figure 5: Mooshak all languages settings

This can also be done in Mooshak's language settings page, figure 4, field **Fork**. which takes precedence.

It is also needed to define a suitable time to allow Digital to be launched, simulate and test circuit. About 5 times should be enough on most systems, for fair complex circuits. If the test case battery is large it may be needed to increase this time. This can be defined in the **ExecTimeout** field of the overall language setting page.

4 Conclusion

Mooshak was designed to perform black box testing on simple programs, being unsuitable to test digital circuits design. Combining the circuit testing abilities of Digital logic simulator with the Mooshak's ability to receive and perform basic black box testing on source code, it is possible to extend Mooshak to test also digital circuits. There are other logic simulator options suitable to the same purpose, the one chosen is a free and open source tool.
This extension is an add on to Mooshak, that establishes an interface with Digital simulator, at the command line level, without any changes to Mooshak and Digital original source code.

References

Helder Daniel. Java unit testing framework, 2019. URL `https://junit.org`. [Online; accessed 7-June-2020].

ICPC. International collegiate programming contest, 2019. URL `https://icpc.global/worldfinals/rules`. [Online; accessed 08-November-2019].

José Paulo Leal and Fernando Silva. Mooshak: a web-based multi-site programming contest system. *Software Practice & Experience*, 33(6):567–581, 2003. URL `http://www.ncc.up.pt/mooshak/`.

Helmut Neeman. A digital logic designer and circuit simulator, 2021. URL `https://junit.org`. [Online; accessed 23-September-2021].