LabEAD AutoTest: Online Tests of Hardware Designs

Victor T. Hayashi[1], **Wilson V. Ruggiero**[1], **Felipe V. de Almeida**[1]

[1] Escola Politécnica – Universidade de São Paulo (USP)

{victor.hayashi,wruggiero,felipe.valencia.almeida}@usp.br

Abstract. *The outsourcing of hardware production to third-party foundries (often overseas) for cost reduction presents additional challenges to the security of hardware devices. If Hardware Trojans are added to hardware devices during the fabrication phase, design-time approaches are not enough. LabEAD AutoTest is an open-source tool useful to perform automated tests in a hardware design in runtime. It is based on the MQTT protocol, the remote lab LabEAD and Python Notebook. An example of a glitch in an adder described in the VHDL Hardware Description Language was used to show the soundness of the proposed tool. LabEAD AutoTest may be useful to aid in the hardware device validation process.*

1. Introduction

The outsourcing of hardware production to third-party foundries (often overseas) for cost reduction presents additional challenges to the security of hardware devices [Fyrbiak et al. 2019, Hu et al. 2020]. If Hardware Trojans are added to hardware devices during the fabrication phase, design-time approaches are not enough [Kulkarni et al. 2016].

Another relevant attack scenario is a malware in an open-source tool used in the hardware development process [Hu et al. 2020]. It is feasible to include a malicious code in a specific open-source Integrated Development Environment (IDE) that adds a Hardware Trojan when generating the final bitstream, so that it becomes impossible to detect the incorrect behavior earlier (i.e., in the simulation phase) [Krieg et al. 2016].

Moreover, a static analysis approach may result in a NP-complete problem. For example, a proposed solution that aims to detect a Hardware Trojan in a larger hardware design using graph theory results in the subgraph isomorphism problem [Fyrbiak et al. 2019].

Considering the reasons presented, it is reasonable to propose an automated tool to test a hardware design in runtime to complement the static analysis approaches. Such open-source tool may be useful to aid in the hardware device validation process. Figure 1 presents some methods and tools one may use in the hardware development process, including the proposed testing tool.

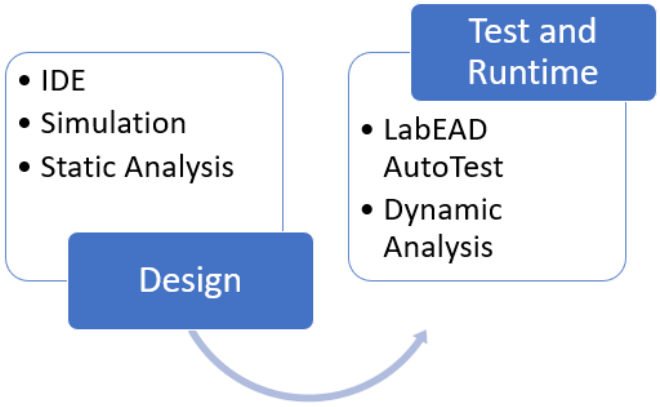

Figure 1. Tools and Mechanism in the Hardware Development Process

The text is organized as follows. In Section 2, the automated testing tool is presented. The demonstration is described in Section 3. Section 4 contains the open-source files, documentation and demonstration video. Final considerations are presented briefly in Section 5.

2. LabEAD AutoTest

2.1. Building Blocks

A Field-programmable gate array (FPGA) is an integrated circuit which can be configured to act as a myriad of different digital systems. An FPGA board contains the FPGA together with peripherals like keys, buttons and LEDs, which can be used to map the FPGA inputs and outputs.

One way to configure the FPGA is using a Hardware Description Language (HDL) such as VHDL. VHDL was developed by the US Department of Defense for the purpose of documenting digital systems. It was standardized latter by the IEEE, enlarging its usage by the circuit designers.

The advantages of using a FPGA together with HDL is how quickly we can describe, simulate, synthesize and test a digital system. It is even possible for the project designer to describe a complex system with a small number of code lines, leaving the work to the synthesizer, that will generate an equivalent system using components such as registers, multiplexers, adders and others.

LabEAD [Hayashi et al. 2020] is an educational project developed with the purpose of allowing teachers and students to interact with lab infrastructure remotely. Taking the digital system design as an example, students in their residences can interact with the FPGA board located in the lab using the developed architecture. The design cycle composed by VHDL description of the system, simulation, system syntheses and FPGA programming can all be done with remote commands.

We opted to use non-proprietary components and protocols in a way that our open-source project could be easily replicated. The main communication protocol used was the Message Queuing Telemetry Transport (MQTT) [OASIS 2014]. MQTT is a lightweight protocol based on the publisher-subscriber pattern. Each device connected to the network

that uses it is called a MQTT Client, and each client can publish or subscribe to a different number of topics. The topic here acts as a label, which the message router called MQTT Broker will use to route the messages between publishers and subscribers. Many programming languages have libraries that implement MQTT Clients. Python, for example, has the *paho-mqtt* library[1], which we used in the project.

A Python notebook is a virtual environment that is used for programming. There are different solutions that implement notebook platforms such as Jupyter Notebook or Google Colab. The native notebook extension (.ipynb) was developed to run Python code, which means that it is possible to create a MQTT Client by using the *paho-mqtt* library inside a notebook.

2.2. Architecture

The architecture of the proposed testing tool is illustrated in Figure 2. The testbench has a computer with a IDE for hardware simulation, synthesis and load to the FPGA (Device under Test). During tests, the FPGA may be monitored by a webcam and the I/O signals are given by a supporting Internet of Things device (in red), which communicates with the other devices using the MQTT protocol. Manual tests can be performed by the smartphone interface, which publishes commands in specific topics. Automated tests can be performed using the LabEAD AutoTest, which consists of two Python Notebooks. All commands from manual or automated tests are relayed to the specific testbench and consequently to the desired device under test by the MQTT Broker of the LabEAD.

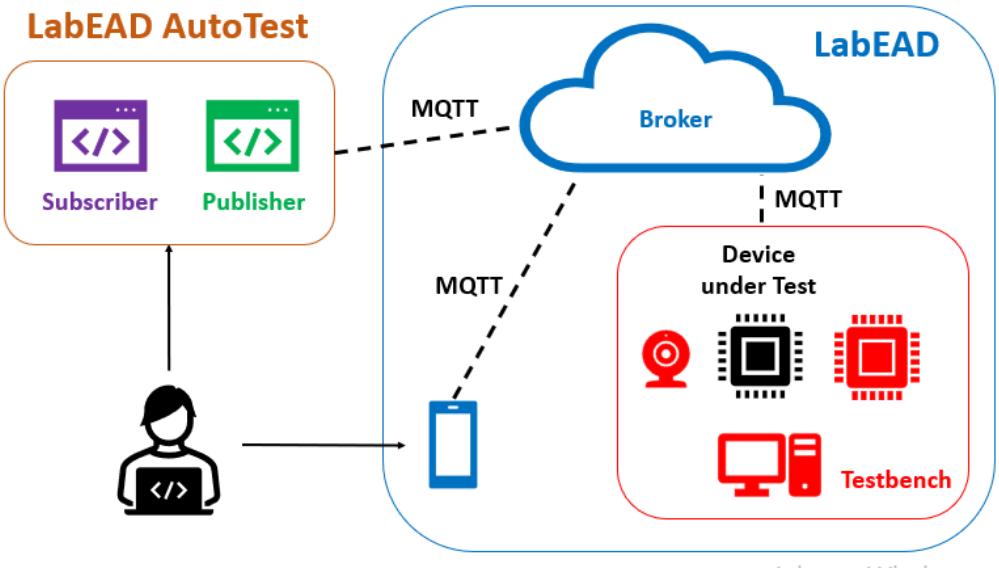

Figure 2. LabEAD AutoTest Architecture

All tools used in the development of the proposed testing tool are open-source.

2.3. Main Functions

Based on a test description, the Subscriber notebook connects to the MQTT broker and subscribes to the Device under Test (DuT) output topics of interest. The Publisher is con-

[1]https://pypi.org/project/paho-mqtt/

nected to the broker before the test initiation. For each test, the Publisher publishes in specific DuT input topics. These commands are relayed to the supporting IoT installed in the testbench, which changes the DuT inputs accordingly. The changes in the DuT's outputs are detected by the supporting IoT, which publishes the results in DuT output topics. The broker relays the results to the Subscriber Notebook. After the last publication, the Publisher sends the test id, and this information is sent to the Subscriber notebook. Upon receival, the Subscriber notebook compares the expected result of the test description and the obtained result to analyze if the behavior of the DuT is correct. This test sequence is portrayed in Figure 3.

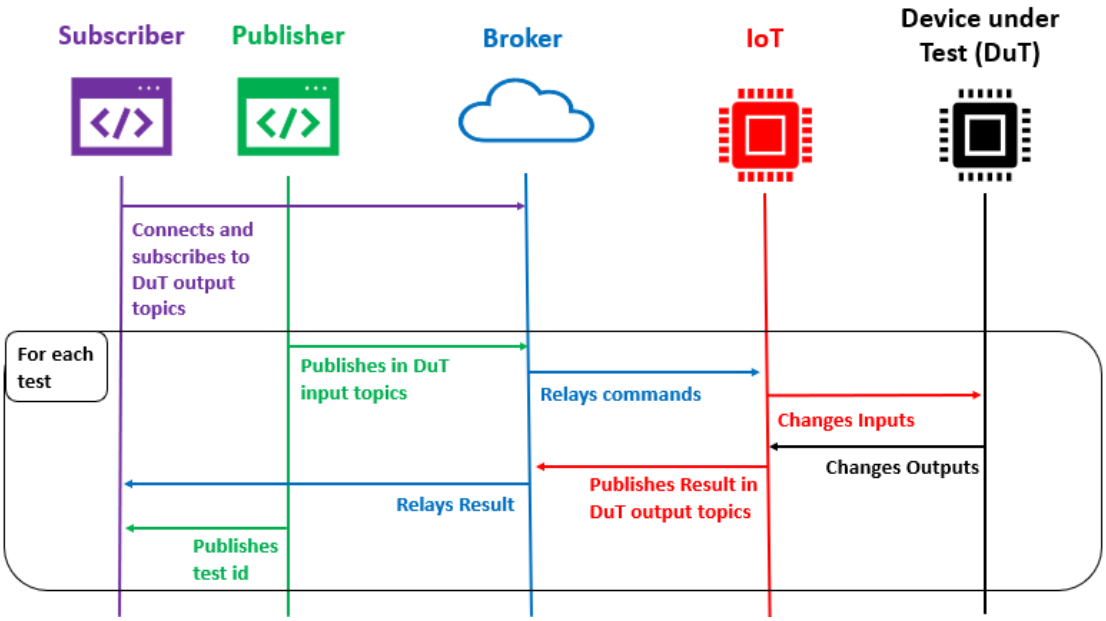

Figure 3. Test Sequence Diagram

2.4. Development Process

The developed digital system to test the soundness of the proposed tool was a 4-bit adder. An HDL structural description of the system was used, first describing the full-adder in a block, then connecting the blocks using the VHDL *port map* resource. Figure 4 shows the system Register Transfer Level (RTL) viewer.

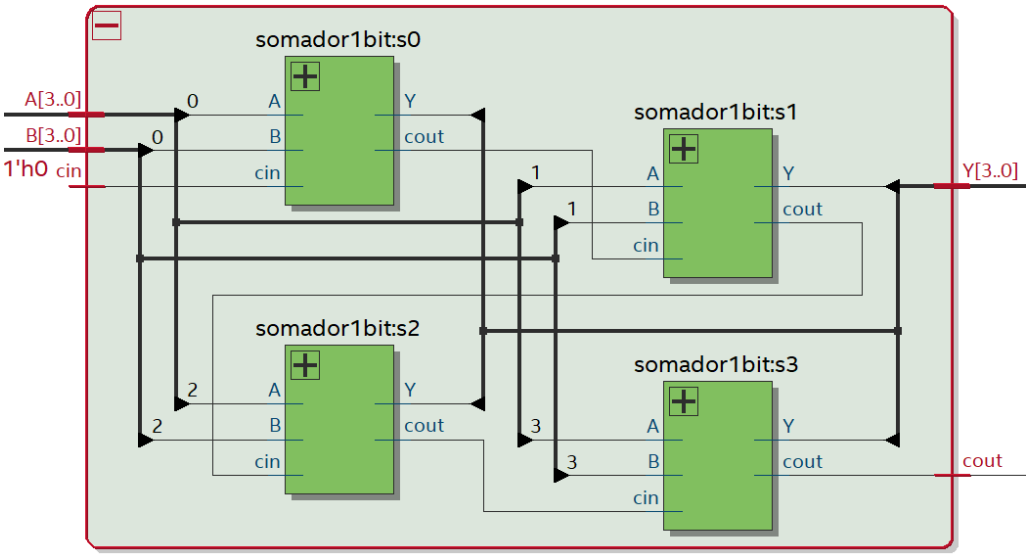

Figure 4. 4-bit adder RTL viewer

Two versions of the same system were developed. The first one is correct version, where the two 4-bit numbers are inputted in the system and the correct sum, with the carry out, is always outputted. The second one is the glitched version that creates a situation with the wrong sum output.

We opted to describe a comparator that checks if one of the inputs assumes one specific value for the situation that triggers the glitch. In this case, the comparator emits a signal responsible for inverting the output bits that correspond to the sum output (0 becomes 1 and 1 becomes 0), showing the result in a 1's complement notation.

Figure 5 shows a high-level vision of the digital system second version.

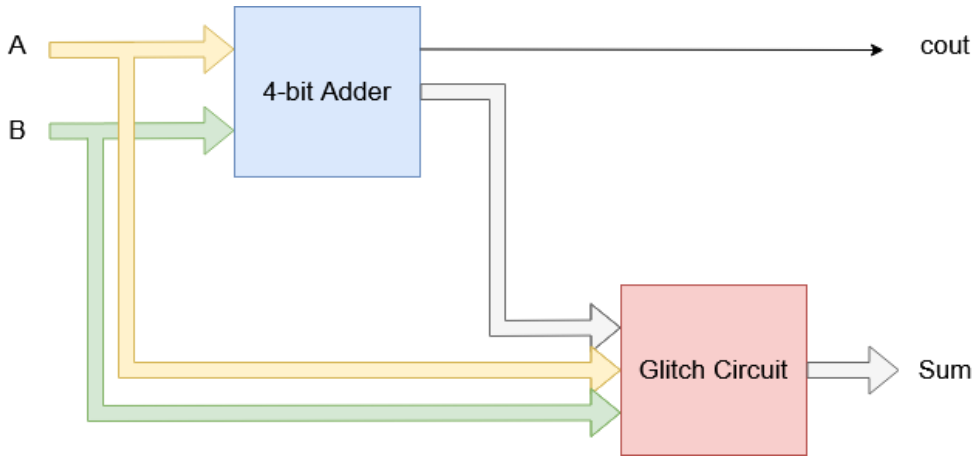

Figure 5. High-level view of the 4-bit adder glitched version

3. Demonstration

3.1. Required Materials

The required materials are:

- The correct 4-bit adder VHDL description available on this project's GitHub.
- The glitched 4-bit adder VHDL description available in this project's GitHub.
- The Subscriber and Publisher notebooks available in this project's GitHub.
- A MQTT broker with enabled authentication using user-password. It is possible to use the University of São Paulo's broker or build from the open-source HiveMQ community edition[2] or the Eclipse Mosquitto[3].
- A testbench of the LabEAD: computer with remote access and IDE for hardware development, webcam and an IoT device using the PubSub MQTT library[4]. For replication of LabEAD, please refer to its GitHub[5]. A demonstration with the LabEAD deployed in the University of São Paulo may also be scheduled.
- (Optional) a smartphone or web client of MQTT (e.g., HiveMQ MQTT client[6]) for complementary manual tests.

It is possible to use a cloud notebook tool, such as the AWS SageMaker or Google Colab, or the open-source Jupyter Notebook locally to run the Subscriber and Publisher notebooks.

3.2. Planned Demonstration

The planned demonstration is described in the following steps:

1. Set the correct 4-bit adder in the FPGA;
2. Begin MQTT connection of the Subscriber notebook;
3. Begin MQTT connection of the Publisher notebook;
4. Perform the tests using the Publisher notebook;
5. Disconnect the Subscriber and the Publisher notebooks from the MQTT broker;
6. Change to the glitched 4-bit adder in the FPGA and repeat steps (2) to (4).

The results of the correct and the glitched versions of the 4-bit adder using the LabEAD AutoTest tool are presented in Figures 6 and 7, respectively. The notebook results are depicted on the left side of the figures, the webcam views are presented in the center, and the MQTT mobile client screenshots (optional) are in the right side.

[2]https://www.hivemq.com/developers/community/
[3]https://mosquitto.org/
[4]https://github.com/knolleary/pubsubclient
[5]https://github.com/vthayashi/labead-labdig
[6]https://github.com/hivemq/hivemq-mqtt-client

Figure 6. Results of the correct version of the 4-bit adder

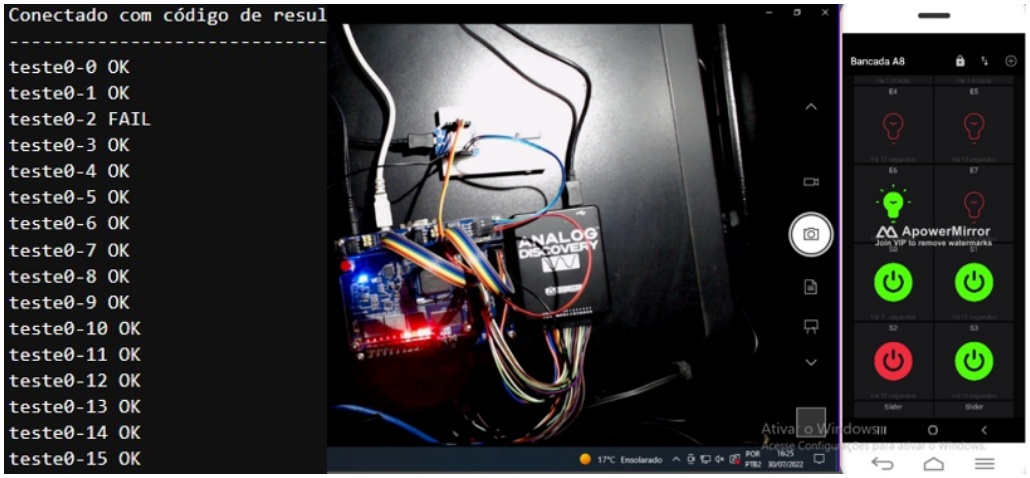

Figure 7. Results of the glitched version of the 4-bit adder

4. Open-source

4.1. Where is the code?

The code is available in this project's GitHub:

`https://github.com/vthayashi/labead-autotest`

4.2. Where are the docs?

The instructions are available also in the GitHub repository:

`https://github.com/vthayashi/labead-autotest`

4.3. Do you have a video?

A video with the demonstration is available in YouTube (Brazilian Portuguese):

`https://youtu.be/pjhnV2qPIEY`

5. Final Considerations

This paper presented LabEAD AutoTest, a tool for performing automatic online tests of hardware designs described in VHDL and deployed on a FPGA development board. It is an open-source tool based on the MQTT protocol, the remote lab LabEAD and Python Notebook. An example of a glitch in an adder described in the VHDL Hardware Description Language was used to show the soundness of the proposed tool. LabEAD AutoTest may be useful to aid in the hardware device validation process.

The limitation of the tool is related to its underlying testing approach. It is not reasonable to perform all possible tests to detect Hardware Trojans. However, in future work it is possible to integrate data collection in the Subscriber module. The data obtained in previous tests may be useful to optimize the testing strategies using a statistical learning method.

Another opportunity in future work is to expand the tool to perform a random testing approach to discover behaviors that are not in the scope of the known and expected functionalities (i.e., unknown behaviors beyond the device specification). One such example is a device whose specification is of a 4-bit adder, but that works as a subtractor given a specific trigger.

References

[Fyrbiak et al. 2019] Fyrbiak, M., Wallat, S., Reinhard, S., Bissantz, N., and Paar, C. (2019). Graph similarity and its applications to hardware security. *IEEE Transactions on Computers*, 69(4):505–519.

[Hayashi et al. 2020] Hayashi, V., Almeida, F., Arakaki, R., Teixeira, J. C., Martins, D., Midorikawa, E., Cugnasca, P. S., and Canovas, S. (2020). Labead: Laboratório remoto para o ensino de engenharia. In *Anais dos Workshops do IX Congresso Brasileiro de Informática na Educação*, pages 187–194. SBC.

[Hu et al. 2020] Hu, W., Chang, C.-H., Sengupta, A., Bhunia, S., Kastner, R., and Li, H. (2020). An overview of hardware security and trust: Threats, countermeasures, and design tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1010–1038.

[Krieg et al. 2016] Krieg, C., Wolf, C., and Jantsch, A. (2016). Malicious lut: A stealthy fpga trojan injected and triggered by the design flow. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE.

[Kulkarni et al. 2016] Kulkarni, A., Pino, Y., and Mohsenin, T. (2016). Svm-based real-time hardware trojan detection for many-core platform. In *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pages 362–367. IEEE.

[OASIS 2014] OASIS (2014). Mqtt version 3.1. 1. *URL http://docs.oasis-open.org/mqtt/mqtt/v3*, 1:29.