Functional & timing in-hardware verification of FPGA-based designs using unit testing frameworks

Julián Caba, Fernando Rincón and Julio Daniel Dondo
Department of Technology and Information Systems, University of Castilla-La Mancha, Spain
email: julian.caba@uclm.es

***Abstract*—In this PhD dissertation, we propose a new testing approach for effectively managing hardware development risks, producing hardware designs with enough quality and reliability. Our proposal is based on the combination of high-level modelling and a unit testing framework in order to generate real hardware implementations for validating the designer intent, in order to keep a high cycle-accuracy and a low design effort. Such real hardware implementations are based on FPGAs, whose reconfigurability are key to provide a flexible verification environment, whereas unit testing frameworks have been extended to consider new testing requirements beyond pure functionality, such as timing analysis. Moreover, we provide a hardware library with two different types of components: 1) monitors to check internal variables at run time, keeping the errors to later trace them, and 2) double functions to reduce third-party dependencies.**

I. Introduction

NOWADAYS, high-level modelling (HLM) is widespread to build hardware designs, providing an early understanding of the design impact decisions, and allowing an effective design space exploration, which results in a higher design productivity and improves the likelihood of finding the most-optimal implementation. In order to fill the gap between development tools and the capabilities offered by the technology, FPGA vendors are making a high effort to include High-Level Synthesis (HLS) as a solution [1]. However, the verification stage still entails an amount of non-trivial problems, that are summarised in the following points and coincide with our motivation.

- The trade off between simulation effort and simulation accuracy depends on the design abstraction level. Using HLM provides the lowest simulation effort but results in a very poor accuracy, whereas a real hardware prototype would be the perfect environment for an accurate verification but it implies an important verification effort [2], and even more the use of real hardware devices introduces a new problem: the exponential synthesis time.
- Each testing-level stage induces rewriting tests, which is prone to human errors and time consuming. The test translation process may result in wrong decisions while developers try to modify them according to the new verification level, instead of just working in production code with the only aim to pass their tests at any level.
- The time spent in verification accounts for roughly 80 percent of the development life-cycle, considering this task as the bottleneck of most projects, and even verification engineers are not the only staff to check the design [3]. Nowadays, the biggest challenge in design and verification is identifying solutions to reduce the verification gap, optimising the *time-to-market* and the product *reliability/quality*.
- Timing is a major issue in hardware projects, which makes the verification process more complex. Timing results in-hardware verification usually differ from verification models used during simulation, where engineers do not care about available resources and their location.
- Maybe the most hard task is to integrate a new component into a test environment fulfilling all its dependencies, since several ones will not be already implemented or simply not available (for the case of third-party components, for example). Many vendors provide simulation models for their products, but they rarely can be synthesised. In addition, a part of a hardware component can be implemented in different ways, and that decision should be taken at implementation time, where it may prove not to be the correct one.

II. Objectives

The main objective of this dissertation is to propose an integral solution that provides solutions to the mentioned verification problems in digital hardware design. This solution is presented as a novel hardware testing framework using the new generation tools provided by FPGA vendors. At the same time, the dissertation should focus on a solution that considers two important verification aspects: functional and timing factors; whose precision should be closed to a real scenario. Thereby, hardware designs described by a high-level programming language must be verified through an automatic infrastructure allowing transparent verification using a real hardware device. The expected contributions of our work are:

- To propose techniques for checking the correctness of a hardware design, contrasting the developer intent and the result obtained, including the timing factor too.
- To ease the verification stage without building the whole verification environment, and reusing the same test at different abstraction level.
- To reduce the third-party dependencies and allow to increase the intermediate results visibility.

III. Our Approach

To provide a full verification framework for hardware components, and to overcome the limitations related to hardware/-software communication, high-level synthesis tools, and unit

This work is supported by *Ministry of Economy and Competitiveness of Spain* under project `REBECCA (TEC2014-58036-C4-1R)`, and *European Regional Development Fund* under project `SAND (PEII11-0277-0070)`

testing principles, we explore an alternative based on Remote Method Invocation (RMI). Each function to test must be accessed individually and checked also individually. In addition, RMI provides a way to communicate components within different implementation domains, whose main advantage is reusing the same test at any abstraction level.

The synthesis process requires high computing power and, thus is quite time-consuming. In order to improve and minimise synthesis process, we include an interesting feature available in some FPGAs called Dynamic Partial Reconfiguration (DPR). One benefit enabled by DPR is the fact that the designer can dynamically insert new functionality without redesigning the whole system. Furthermore, it is possible to adapt the FPGA to different scenarios, by simply replacing a few components. In our approach, we use DPR in those parts of the system that are susceptible to be changed, mainly the DUTs. The DUT area will be updated everytime a new version is available, thus an engineer can exercise the DUT on real hardware in a short time.

HLS tools introduce a new problem: the visibility of the generated design. These tools produce complex designs whose traceability is quite limited, requiring developers to trace the signal(s) manually when the simulation fails. Therefore, our approach must provide some facilities to improve visibility of internal variables or intermediate results.

On the other hand, the integration of a new component with its dependencies in a test environment is a hard task, since some may not be implemented or may not be available at that time. Thus, the use of test doubles can be an appropriate solution to perform testing efficiently and effectively, reducing the dependencies with third-party components.

Summarising, our approach provides a transparent testing service through a hardware platform where a DUT is deployed into a dynamic area. DUTs are generated using HLS tools, and are verified through unit testing, checking its behavioural and timing correctness. These tests are the same at any abstraction level. The testing process is transparently automated; an engineer commits his design code and unit tests written in a high-level language, such as C, into a repository, and automatically the testing service is able to synthesise the design code, deploy the DUT remotely into an FPGA and exercise it with the original unit tests, reporting the testing result to the engineer. In addition, we provide some facilities to reduce third-party dependencies and to increase the intermediate results.

IV. Current Status and Future Work

To communicate a hardware component with other hardware or software components we built a communication mechanism based on RMI that is able to route the required messages. This mechanism can be used on any bus, such as AXI. In addition, the hardware component is wrapped to ensure individual function access, fulfilling the unit testing principles and HLS restrictions. To get a solution based on RMI technology, addressing these limitations, we follow the research line explained in [4] and [5]. The testing framework chosen is *Unity*, and has been extended to fit hardware timing features.

To facilitate the DPR process we built a fast reconfiguration component which is able to deploy new functionality without

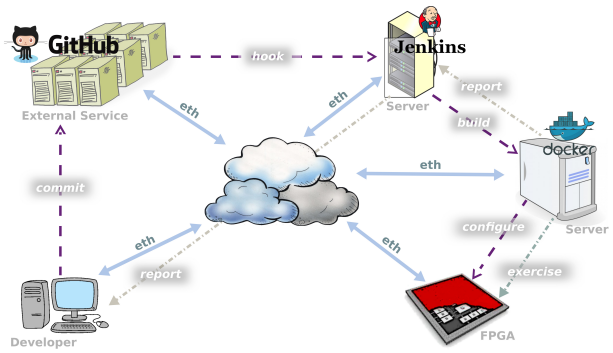

Figure 1: Overview of Remote Testing Service

to stop the whole design. Thus we only must synthesise just the logic of DUT instead, reducing the computation time and power consumption. The hardware component developed achieves the maximum theoretical speed up to deploy a partial bitstream though the ICAP component (about 400 MB/s). For the experiments we use a Zedboard from Xilinx.

We addressed the visibility problem through the implementation of hardware asserts, providing a hardware library which contains a set of functions that are able to check intermediate results. Moreover we have included a test double technique to reduce or eliminate the third-party dependencies. Both solutions provide extra functionality to express how often, with which arguments and the relative time when the method shall be called, this information is stored internally and can be accessed from unit tests.

Finally, we have integrated our approach using *Jenkins* as a continuous integration platform connected to a repository, such as *GitHub* which stores the source code: the DUT and its tests. Thereby, when a change is committed to the repository, the *Jenkins* framework triggers the building of the new DUT version in a remote node through a *Docker* container. After the synthesis process, the container sends the partial bitstream to the real hardware and it deploys the new functionality through the DPR capabilities of the FPGA. Then the container stimulates the DUT using the tests defined by the developer. Finally, the developer is able to observe the correctness of the new version in a real hardware. (see Figure1)

For the subsequent steps, we plan to complete some experiments using real-life applications to address a verification stage lead by our proposal. Nowadays, we are using a case study based on the histogram of oriented gradients to validate our proposal [6].

References

[1] J. Cong et al. *"High-Level Synthesis for FPGAs: From Prototyping to Deployment"*, Computer-Aided Design of Integrated Circuits and Systems, 2011.

[2] L. Gong and O. Diessel *"Functional Verification of Dynamically Reconfigurable FPGA-based Systems"*, Springter, 2015.

[3] H. Foster *"The 2016 Wilson Research Group Functional Verification Study"*, Mentor Graphics, 2016.

[4] F. Rincón et al. *"Transparent IP Core Integration Based on the Distributed Object Paradigm"*, Intelligent Technical Systems, 2009.

[5] J. Barba et al. *"OOCE: Object-Oriented Communication Engine for SoC Design"*, DSD, 2007.

[6] N. Dalal and B. Triggs *"Histograms of Oriented Gradients for Human Detection"*, CVPR, 2005.