In-Circuit FPGA Debugging using Parameterised Reconfigurations

ALEXANDRA KOURFALI, Ghent University
DIRK STROOBANDT, Ghent University

Reassuring fault tolerance in computing systems that contain FPGA devices is the most important problem for mission critical space components. With the rise in interest of commercial SRAM-based FPGAs, it is crucial to provide runtime reconfigurable recovery from a failure. In this paper, we propose a superimposed virtual coarse-grained reconfigurable architecture, embedded with on-demand three level fault-mitigation technique. The proposed method performs run-time recovery via discrete microscrubbing. This approach can provide up to 3ÃŮ faster runtime recovery with 10.2ÃŮ less resources in FPGA devices, by providing integrated layers of fault mitigation.

Additional Key Words and Phrases: FPGA, signalrank, gaterank, debug, parameterised configuration

ACM Reference format:
Alexandra Kourfali and Dirk Stroobandt. 2017. In-Circuit FPGA Debugging using Parameterised Reconfigurations. 1, 1, Article 1 (July 2017), 3 pages.
https://doi.org/http://dx.doi.org/10.1145/3060558.3079422

1 INTRODUCTION

Ensuring functional correctness despite the rising levels of design complexity has been a major focus of research and development since the beginning of digital system design. This focus has led to significant advances in the verification and debugging of digital designs. While failures can be caused by various defects, such as logic, timing, layout, there is no straightforward way to locate the root cause of these errors. Designers have been using simulation to verify their designs. However, simulation can be incomplete time-consuming and bugs can potentially escape into silicon. A growing trend is to prototype the Design Under Test (DUT) using FPGA emulation, as it enables higher verification coverage compared to simulation and can operate orders of magnitude faster. The main drawback of FPGA emulation is the increased FPGA resources and its limited internal signal observability. While simulation gives full visibility, FPGA emulation allows the designer to observe only the signals that are driven through the scarce output pins.

2 IN-CIRCUIT PARAMETERISED DEBUGGING FLOW

We introduce a debugging flow, which is automated and integrated within the normal FPGA flow. It offers low-overhead accelerated debugging, with enhanced internal signal visibility by using parameterised configurations. This allows us to implement parameterised DUTs, with parameters that define different circuit instances that can be optimised on the fly by reconfiguring for a current set of parameters. Our debugging flow follows the typical

The first author is sponsored by a Ph.D. grant of the Flemish Fund for Scientific Research (FWO). This work was supported by the European Commission in the context of the H2020 FETHPC EXTRA project (671653).
Authors' addresses: Department of Electronics and Information Systems (ELIS), Ghent University, iGent, Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium.

XXXX-XXXX/2017/7-ART1
https://doi.org/http://dx.doi.org/10.1145/3060558.3079422

stages of the FPGA flow (synthesis technology mapping, place and route) and consists of two phases: the design phase and the verification phase.

During the *design phase*, *GateRank* is introduced, where all signals that can be used for debugging are automatically selected, and finite graphs with its adjacency matrices are created that describe the gate-level DUT. This is then balanced, ranked and plotted in such way that the designer can have an indication of the signals that are highly utilised (high GateRank) and of the amount of on-chip memories and trace cycles that are needed. Then, extra parameterised debugging support is added at the high GateRanked signals, as they will be the ones traced during debugging. Then, the mapping is fixed and the debugging infrastructure is optimized alongside the original DUT. During *Place & Route* we use the routing of the circuit in such way that its routing resources can be reused during debug. The parameterised infrastructure can hence be implemented in the FPGA's reconfiguration resources. This drastically reduces the area usage. Finally, a virtual intermediate FPGA configuration is created, with the debugging infrastructure integrated in the DUT. This allows most signals to be connected with on-chip memories that trace signal changes during the verification phase.

During the *verification phase*, for each debugging turn, the DUT can be reconfigured with different GateRanked signal sets. The signals that are not traced at the same time can share routing resources (based on their parameter settings). For each debugging cycle the new signal selection translates directly into a new evaluation of a Boolean function that represents the selected signals. Then, the new network is partially reconfigured with the exact signals the designer wishes to trace.

3 PRELIMINARY RESULTS

The first experiments with the ITC benchmarks indicate that we only need the sum of areas of the DUT and the optimised on-chip memories, instead of the sum of areas of the DUT and the added infrastructure (fixed on-chip memories and fixed trace infrastructure), as is the case in related work. In that way, we drastically reduce the resource overhead compared to related work, with a small area penalty (5-10%) on the DUT. The critical path delay of the added functionality also remains the same with the original DUT for most benchmarks. The logic depth (inversely related to clock speed) of the DUT changed after adding the extra debugging infrastructure by 10% in the worst case scenario.

The runtime overhead depends on the number of times the FPGA needs to be reconfigured and on the time to evaluate the parameterised configuration and to reconfigure the bits that changed. This is maximum 50 μs. Thus, each parameterised configuration can be up to 3 orders of magnitude faster than a full reconfiguration (176 milliseconds for a Xilinx Virtex-V FPGA). This describes the time needed for one debugging turn. The GateRank algorithm can drastically alter the number of debugging turns needed and the number of cycles each signal is being traced for. This has a large impact on the debugging overhead. The automation of the exact percentage of the high-GateRanked signals to be simultaneously traced to reduce the overhead below the 5% threshold and their impact on the FPGA debugging turns, will be future work.

4 CONCLUSION

An in-circuit debug methodology is proposed. The main parameterised FPGA flow is presented, enhanced with a signal ranking algorithm and parameterised low overhead added infrastructure, for increased design observability. The added infrastructure is optimised alongside the original design and is invoked only when a parameterised trigger is set. The area needed is found by introducing parameterized reconfiguration in the design. Hence, thanks to the fact that there is low overhead over the original implementation, we can incrementally add the debugging functionality almost for free.

ACKNOWLEDGMENTS

The first author is sponsored by a Ph.D. grant of the Flemish Fund for Scientific Research (FWO). This work was supported by the European Commission in the context of the H2020 FETHPC EXTRA project (671653).