CellIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL

Flavien Solt, *ETH Zurich;* Ben Gras, *Intel Corporation;* Kaveh Razavi, *ETH Zurich*

https://www.usenix.org/conference/usenixsecurity22/presentation/solt

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

A Artifact

A.1 Abstract

In our Artifact, we provide the source code of CellIFT, a native RISC-V toolchain, and other dependencies. We also provide the framework for performing all the experiments described in this paper and analyzing the obtained results. Everything is packaged as a Docker image to allow for optimal reproducibility. To reproduce the experiments, we expect a machine with 256 GB memory and 500 GB of free storage.

A.2 Artifact checklist

- **Algorithm:** CellIFT is a newly developed algorithm to efficiently generate IFT shadow logic as part of a Yosys pass.
- **Program:** We use a set of five external RISC-V CPU designs (Ariane, BOOM, Ibex, Rocket, PULPissimo) as evaluation targets, as well as benchmarks from the RISC-V Architectural testing framework. All of this code is included in our artifact.
- **Compilation:** We include the required compilers and interpreters.
- **Transformations:** We include the required Verilog transformations (CellIFT and GLIFT), implemented as Yosys passes.
- **Binary:** We include prebuilt Verilator binaries of the five CPU designs in all instrumentation modes (i.e., vanilla, CellIFT, and GLIFT) where possible. Note that GLIFT instrumentation or synthesis sometimes fails, as explained in Section 7.2.
- **Run-time environment:** The bulk of our artifact is a Docker image that runs on Linux. We tested our image on an Ubuntu 22.04 system with 5.15.0-37-generic kernel.
- **Hardware:** We do not require any special hardware, but do need a relatively large amount of DRAM (256 GB) to run all the experiments.
- **Metrics:** The experiments record runtime performance and IFT precision for microbenchmarks for CellIFT as well as GLIFT. Further experiments record execution time and memory footprint of the instrumentation and synthesis process for all instrumentation modes. We also measure the simulation performance on for all instrumentation modes. Lastly, we show resource usage and clockable frequency after FPGA synthesis for all the five CPU designs under all instrumentation modes.
- **Output:** For all experiments used in the Evaluation section of this paper (Section 7), we include code to regenerate the charts. Also, we include code to reproduce all results in the Scenarios section of this paper (Section 8).
- **Experiments:** With the exception of the FPGA results, all experiments are executed automatically when building the Docker image. This means the way to reproduce all experiments is encoded in the Dockerfile, and a Docker container based on this Dockerfile would contain the generated results, and can be used to re-run individual experiments if desired.
- **How much disk space required:** The docker image with all the layers is 330 GB, and Xilinx Vivado requires around 150 GB for downloading and installation. In total, we estimate a total of 500 GB of free storage is required.
- **How much time is needed to prepare workflow:** To prepare the workflow, conscious effort is only needed to retrieve the Git repository and the Docker image, which should take only a few minutes.
- **How much time is needed to complete experiments:** Reproducing the experiments takes approximately 3 days.
- **Publicly available:** Stable URL: https://github.com/comsec-group/cellift-artifacts/commit/eaa9a26ae85fd6a7ae8cd248416315414ae4c135. The README points to a stable (sha256-verified) Dockerhub Docker image that contains the rest of the code and data, namely docker.io/ethcomsec/cellift-artifact-evaluation@sha256:9a15d4070d321026ad4d5d9ba5a236842c6c456279f9c08f4fa4132de7b399ce.
- **Code licenses:** CellIFT is licensed under GPL3.
- **Workflow frameworks used:** Docker, Make, Luigi.

A.3 Description

A.3.1 How to access

The project is located at https://comsec.ethz.ch/cellift. Our artifact is a single Git repository designed primarily to build a Docker image that has run all the experiments automatically. This Git repository is hosted at the ‘Publicly available’ checklist entry. The README.md in that repository contains further instructions to obtain the prebuilt Docker image from Dockerhub.

A.3.2 Hardware dependencies

The artifact will run all experiments on a machine with 256 GB of memory.

A.3.3 Software dependencies

We tested the Docker image on Ubuntu 22.04 LTS kernel 5.15.0-37-generic, but we expect it to work on a wide range of Linux distributions.

To reproduce the FPGA experiments in the paper, we furthermore depend on the Xilinx Vivado FPGA synthesis tool (version 2019.3).

A.4 Installation

The installation of our artifact requires the following two steps:

1. Cloning the git repository specified in the checklist and using its README.md to pull the Docker image artifact hosted on Dockerhub.
2. Reproducing the FPGA experiments, requires the installation of the full edition of Vivado 2019.3 from the Xilinx website and a license.

A.5 Experiment workflow

Follow the instructions in the git repository README.md that specifies in detail how to start a Docker container with the image, and how to reproduce each experiment, and examine the results.

In principle, cloning the git artifact repository and rebuilding the Docker image using the Dockerfile in the git repository will rebuild all CellIFT code and designs from scratch and perform the experiments (except the FPGA experiments). For maximum reliability, we also provide the prebuilt Docker image with all code, binaries and results that we have found to work, which can be used to reproduce all the experiments (and use CellIFT in general if desired).

To run the FPGA experiments, first source the settings64.sh file from the Vivado installation dir, and follow the instructions in the Artifact README.md.

A.6 Evaluation and expected results

The key results from our experiments are as follows. For each result, we point to scripts (Python or bash) that drive the experiments and show the analysis.

1. Instrumented designs that we can synthesize to C++ (i.e. be compiled) for all five RISC-V CPU designs, contrary to GLIFT, and with less CPU time and memory (follows from plot_instrumentation_performance.py and plot_rss.py), and with higher tainting precision (follows from plot_num_tainted_states_ibex.py).
2. For the designs that can be compiled in all instrumentation modes, we show that CellIFT has lower performance overhead than GLIFT (follows from plot_benchmark_performance.py).
3. The Meltdown and Spectre simulations reproduce Figure 11, showing they can both be detected (follows from plot_tainted_elements.py).
4. We show several bug scenarios detected by CellIFT (run_scenarios.sh).
5. We show FPGA synthesis results, showing that CellIFT instrumented designs can be synthesized, with fewer resources than the GLIFT instrumented designs.

We refer to the README.md of the artifact git repository for the detailed steps to reproduce each of the key results described above.

A.7 Experiment customization

There is ample customization opportunity in the Docker image, because the code of the instrumentation tool as well as the target designs are there and can be modified and rebuilt. This does require a deeper knowledge that goes beyond this appendix.

A.8 Version

Based on the LaTeX template for Artifact Evaluation V20220119.