Bending microarchitectural weird machines towards practicality

Ping-Lun Wang, Riccardo Paccagnella, Riad S. Wahby, and Fraser Brown, *Carnegie Mellon University*

https://www.usenix.org/conference/usenixsecurity24/presentation/wang-ping-lun

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

USENIX Security '24 Artifact Appendix: Bending microarchitectural weird machines towards practicality

Ping-Lun Wang, Riccardo Paccagnella, Riad S. Wahby, and Fraser Brown
Carnegie Mellon University

A Artifact Appendix

A.1 Abstract

This artifact provides the source code of our weird machine compiler that can generate Flexo weird machines, as well as our weird machine packer, UPFlexo. Our weird machines support eight x86_64 processors from AMD and Intel, and these processors are available on the AWS EC2 cloud platform. It takes roughly 12 hours to reproduce the experiment results in our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

The stable reference to our artifact is accessible via: `https://github.com/joeywang4/Flexo/tree/00186b46205497e87db78add5f2c86e69593a2e7`.

A.2.3 Hardware dependencies

Our weird machines are tested on eight AWS EC2 instances running x86_64 processors from AMD or Intel. Table 1 lists the microarchitecture, instance type, and processor of these eight instances.

Microarchitecture	Instance Type	Processor
Zen 1	t3a.xlarge	AMD EPYC 7571
Zen 2	c5a.xlarge	AMD EPYC 7R32
Zen 3	c6a.xlarge	AMD EPYC 7R13
Zen 4	m7a.xlarge	AMD EPYC 9R14
Skylake	c5n.xlarge	Intel Xeon 8124M
Cascade Lake	m5n.xlarge	Intel Xeon 8259CL
Icelake	m6in.xlarge	Intel Xeon 8375C
Sapphire Rapids	m7i.xlarge	Intel Xeon 8488C

Table 1: The AWS EC2 instances and processors we use to perform our evaluation.

Our weird machines may be able to run on other x86_64 processors with similar microarchitectures as these eight instances. However, we cannot guarantee the accuracy when running our weird machine on processors not listed in Table 1, since it may require further tuning for our weird machines to work on other types of processors.

A.2.4 Software dependencies

Any 64-bit Linux operating system can run our artifact, while we suggest using Ubuntu 22.04 or a later version. To build our compiler and run the evaluation script, `podman` and `python3` are required.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Clone the git repository of our artifact (including the submodules), and install the dependencies: `podman` and `python3`. After that, run the bash script located at `reproduce/scripts/build_all.sh` to install our compiler and compile the weird machines. The installation takes roughly one hour, and it consumes about ten gigabytes of disk space.

A.3.2 Basic Test

To test the installation, run the Python script located at `reproduce/scripts/test_install.py`.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Size, accuracy, and runtime of the circuits Our artifact can reproduce the results in Table 2, 3, 4 and 5 of our paper. These tables contain the circuit size, accuracy, and runtime of our Flexo circuits.

(C2): Runtime of the packer Our artifact can reproduce the results in Table 6 of our paper. This table contains the unpacking time of our weird machine packer.

A.4.2 Experiments

(E1): *[Circuits] [5 human-minute + 8 compute-hour]:*
How to: To run the circuits, execute the Python script located at `reproduce/scripts/run_WM.py`. The script will output the accuracy and runtime for each circuit.
Preparation: Make sure the installation is successful before running the Python script.
Execution: Run the Python script. For more information about how to configure the experiment or the command line arguments for the script, please refer to the read me file located at `reproduce/README.md`.
Results: The script will output the accuracy and runtime for each circuit. The results are also recorded in the results folder located at `reproduce/results/`. Run the Python script with `-r` command line argument to print out the results in the folder. The circuit size is recorded at `reproduce/build/circuits.log`. The results generated in this experiment should be similar to the results in Table 2, 3, 4, 5 in our paper.

(E2): *[Packer] [5 human-minute + 4 compute-hour]:*
How to: Run the Python script located at `reproduce/scripts/run_packed.py`. The script will output the runtime for each packer.
Preparation: Make sure the installation is successful before running the Python script.
Execution: Run the Python script. For more information about the command line arguments for the script, please refer to the read me file located at `reproduce/README.md`.
Results: The script will output the results. The results are also recorded in the results folder located at `reproduce/results/`. Run the Python script with `-r` command line argument to print out the results in the folder. The results generated in this experiment should be similar to the results in Table 6 in our paper.

A.5 Notes on Reusability

Compiling a custom circuit It is possible to use our compiler to create a new weird machine using a custom circuit. Please refer to the read me file for instructions about how to use our compiler. The `circuits` folder provides several examples about how to write a weird machine circuit in C/C++.

Using a custom encryption scheme for UPFlexo UPFlexo currently supports Simon and AES encryption schemes. To customize the encryption scheme for UPFlexo, add a new weird machine under the `UPFlexo/WM/` folder.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.