HyperSSE: Cross-Domain Static Analysis of Partitioned Real-Time Hypervisor Systems (Artifact)

Andreas Kässens
Leibniz Universität Hannover, Germany

Mareike Burg
Leibniz Universität Hannover, Germany

Daniel Lohmann
Leibniz Universität Hannover, Germany

Abstract

Timing analysis for embedded real-time systems is crucial to guarantee the correct behavior and to calculate the *Worst-Case Response Time (WCRT)* of safety-critical applications. With the increasing requirements of such systems in automotive, industrial or avionic industries, consolidation of multiple real-time and general-purpose operating systems on a single high-performance *Multiprocessor System-on-Chip (MPSoC)* platform using *Static Partitioning Hypervisors (SPHs)* is becoming more prevalent. Although the strong separation is well suited to reduce interference between isolated domains in such mixed-criticality systems, cross-domain interactions must still be considered in real-time analysis. Previous work has focused on dynamic monitoring and enforcement of timing constraints in virtualized environments.

In this paper, we present HyperSSE, the first approach for the *static* analysis of cross-domain interactions in hypervisor-based real-time systems. By hierarchically combining existing domain-local static analyses, and synchronizing the control flow at cross-domain interactions, we enable control-flow–sensitive whole-platform analysis including multiple real-time domains. Using abstract task models, this approach can integrate a coarser analysis of general-purpose operating systems, accelerators, and coprocessors with the precise timing analysis. We demonstrate the applicability of HyperSSE in an automotive case study with mixed-criticality software stacks running on Xen in a static partitioning configuration. The resulting *Hypervisor State Transition Graph (HSTG)* exposes deep knowledge about the platform interactions, enabling cross-domain timing analysis with reduction of pessimistic WCRT calculations. Additionally, HyperSSE can be used for placement optimizations for better predictability, verification of critical interaction paths, and is the foundation for further platform-level analyses, such as analysis of implicit interactions through transparent resource sharing.

2012 ACM Subject Classification Computer systems organization → Real-time operating systems; Software and its engineering → Automated static analysis

Keywords and phrases Static Analysis, Hypervisor, Real-Time Operating System, Cross-Domain

Digital Object Identifier 10.4230/DARTS.12.2.2

Funding This work was partly supported by the German Research Foundation (DFG) under grant no. LO 1719/4-1.

Related Article Andreas Kässens, Mareike Burg, and Daniel Lohmann, "HyperSSE: Cross-Domain Static Analysis of Partitioned Real-Time Hypervisor Systems", in 38th European Conference on Real-Time Systems (ECRTS 2026), LIPIcs, Vol. 375, pp. 23:1–23:27, 2026.
https://doi.org/10.4230/LIPIcs.ECRTS.2026.23

Related Conference 38th European Conference on Real-Time Systems (ECRTS 2026), July 7–10, 2026, Lund, Sweden

1 Scope

To obtain and reproduce the evaluation results of the HyperSSE research paper [1], we provide this artifact as docker image.

2 Content

The prebuilt docker image includes:

- the cloned source code from the open source repository of HyperSSE [3],
- all dependencies required to build and run the test cases,
- a Python script that executes all test cases and generates the evaluation results.

Option A: Artifact setup

To setup the artifact, the following steps are required:

```
docker load -i ecrts26_image.tar.gz

# run container
docker run --name parrot --rm -it parrot:hypersse-ecrts26
cd parrot
```

Option B: Building the artifact from source

Alternatively, the artifact can also be built from the Dockerfile:

```
# clone, build and run the container locally
git clone --depth 1 https://github.com/luhsra/parrot --branch ecrts26
cd parrot

# build container
docker build -t scm.sra.uni-hannover.de:5050/research/parrot ./Docker

# run container
./Docker/run-user-docker.sh

# configure meson, might take some minutes because of zephyr's toolchain
meson setup build --native-file native-debian.ini -Denable_posix=true \
 -Denable_arm=true -Dbuild_hypersse_xen=true
```

Reproducing Results

All test cases of the HyperSSE are wrapped into a meson test suite. It executes all tests in parallel, compiling each case's source code into LLVM IR code, performing preprocessing, and analyzing it using the HyperSSE in the ARA framework.
The results are placed in the folder `parrot/build/dumps/`, of interest are each test case's State-Transition-Graphs named `HyperSSE.*.hstg.dot`, and the control-flow graphs in Atomic-Basic-Block notation in the `DumpCFG.*.dom*.abb.dot` files.
The source code of the applications is located in `parrot/subprojects/ara/appl/Xen/`.
The `paper.py` helper script provides additional options for executing the test suite and further compares the generated state transition graphs with a list of graphs we manually verified. Furthermore, it creates SVGs of the dot graphs and the figures in the appendix. Each run creates a folder in the directory `parrot/subprojects/ara/test/xen-eval`, creating the `ecrts26-*.pdf` and `res-ara.csv` with statistics about the analysis. Sometimes, single test cases may fail because of concurrency issues in meson, repeat the command in that case.

```
# compile required targets
cd build
# might take ~5 minutes
meson compile

# optional: run 99% hypersse test suite (~5s per case, 40 parallel cases)
meson test --suite hypersse_xen
# optional: run complete hypersse test suite (~100s for special case)
meson test --suite hypersse_xen_full

# reproduce paper graphs
cd ../subprojects/ara/test/xen-eval
python3 paper.py --ecrts26 true

# extract results from container (Option A)
docker cp parrot:/parrot/subprojects/ara/test/xen-eval/ .
```

The most interesting files are in the ARA subproject [2]: the new Xen model `ara/os/xen.py` and the main control flow in `ara/steps/hypersse.py`, which uses the `ara/steps/multisse.py`. The hypercall header file is located at `libs/os/xen/events.h` and the graph-tool/cython datastructures are in `ara/graph/`.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://zenodo.org/records/19844317`.

The source code of HyperSSE, integrated into the ARA framework, its related subprojects, and test cases are available on GitHub [3].

4 Tested platforms

The artifact can be executed using Docker, or any Linux system which supports all required dependencies (see Dockerfile for details). A maximum of 16 GB disk space is necessary to compile and execute all testcases.

5 License

The artifact is available under license GNU General Public License v3.0 or later.

6 MD5 sum of the artifact

6ba8d1910d26d1b0a6e7996340fb565f

7 Size of the artifact

4.2 GB

References

1 Andreas Kässens, Mareike Burg, and Daniel Lohmann. HyperSSE: cross-domain static analysis of partitioned real-time hypervisor systems. In *Proceedings of the 38th Euromicro Conference on Real-Time Systems (ECRTS '26)*, 2026. doi:10.4230/LIPIcs.ECRTS.2026.23.

2 LUH SRA. ARA – Automatic Real-time System Analyzer. URL: `https://github.com/luhsra/ara`.

3 LUH SRA. PARROT: Projects regarding ARA combined with Real-time Operating system applications and their Toolchains. URL: `https://github.com/luhsra/parrot`.