Real-time Simulation and CHiL Testing of Power System Stabilizers on Microcontrollers with Modelica and eFMI

Luigi Vanfretti[1] **Christoff Bürger**[2] **Joseph Pizzimenti**[1] **Kyle Wilt**[1] **Hao Chang**[1]

[1] Electrical, Computer and Systems Engineering Dept., Rensselaer Polytechnic Institute, USA

[2] Dassault Systèmes AB, Lund, Sweden

American Modelica & FMI Conference 2026

Georgia Tech, Atlanta · 12–14 October 2026

LUIGI VANFRETTI

Professor, ECSE, Rensselaer Polytechnic Institute

vanfrl@rpi.edu

alsetlab.com

github.com/ALSETLab/efmi-pss-use-case

The problem: Why damping control must be redesignable?

Grid incidents keep coming back to damping

- **ENTSO-E 2016 and 2017** — inter-area oscillations across the European system.
- **Iberian grid, April 2025** — complex **0.63 Hz** and **0.2 Hz** oscillations. The report blames poor coordination and the *absence of detailed models of active damping controls*, including existing PSS, for limiting operators' ability to foresee and contain it.

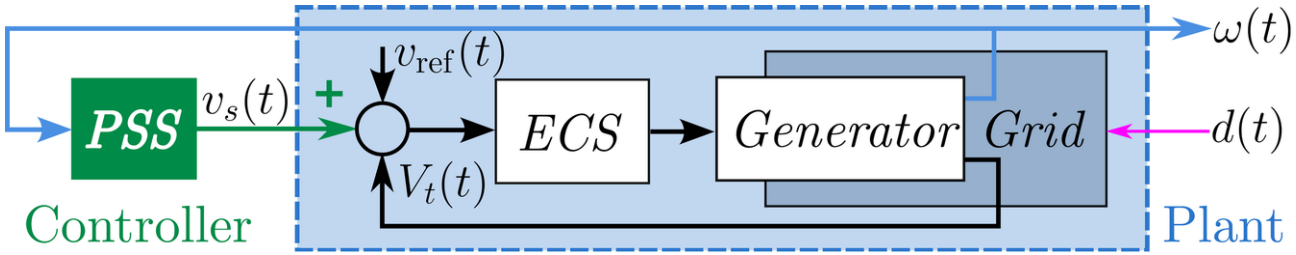
The controller is fixed; the grid is not

- **A PSS has a fixed structure** — a washout, two lead-lag stages, a gain and a limiter. Only its parameters are designed.
- **Those parameters are set once**, in offline design studies before commissioning, and rarely revisited.

⇒ Damping controllers must be redesigned and redeployed throughout their operational life.



ABB Unitrol 1020 Excitation Control System (ECS)
PSS included as part of the ECS firmware.

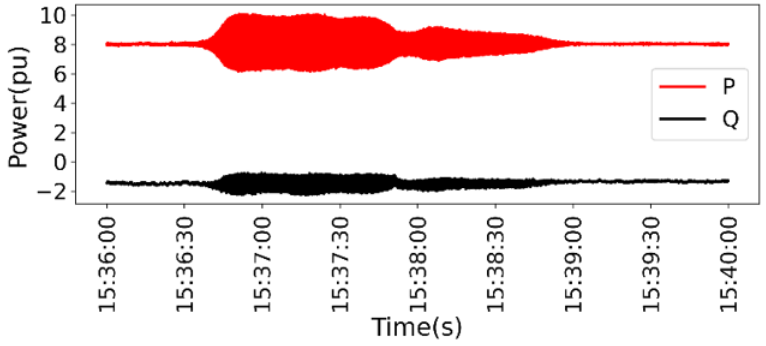


Generator with a PSS ahead of the ECS, interconnected with a grid

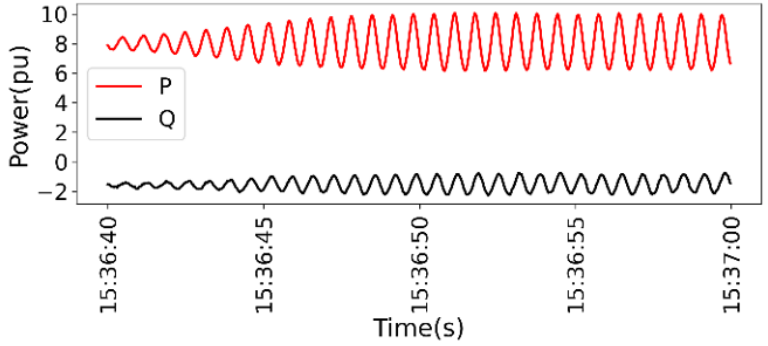
Vocabulary for the rest of the talk

- **controller** — the PSS alone
- **generator unit** — the generator and its excitation control system (ECS), with or without a PSS
- **plant** — the generator, the ECS and the rest of the grid, w/o the PSS
- `d(t)` — grid disturbances: faults, line trips

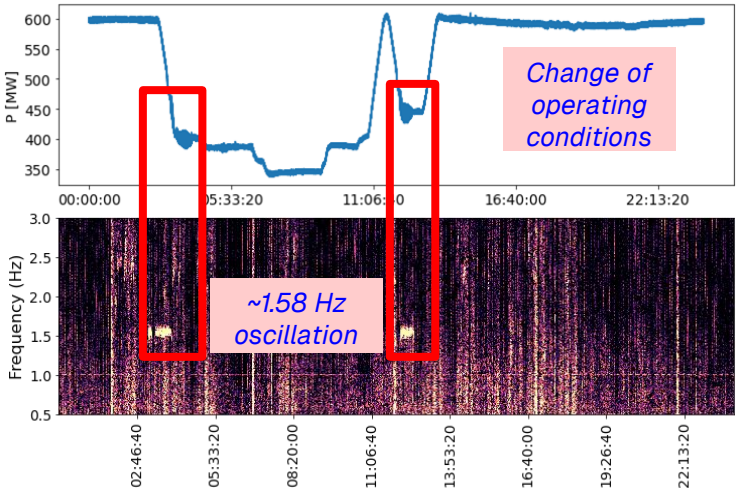
The Evidence: Local oscillations — Need for new PSS and Re-design



PMU measurements of P and Q from the plant: entire test window



Zoom — the limit cycle in both P and Q



Control design tested at several power points; the mode is excited at low power dispatch

- **A change of voltage set-point** moved the excitation system and AVR dynamics into the leading power-factor region.
- **Visible in both P and Q** — P participates through the P–V coupling of the synchronous machine.
- **A limit cycle, not a divergence** — the over-excitation limiter bounds it; the ≈1.58 Hz mode is excited only in the low-P conditions where the design fails.

The PSS was not tuned for light load in the leading-p.f. region.

New controller functions will require to update the parameters with changes in operating conditions.

C. Mishra, L. Vanfretti, M. Baldwin, J. Delaree Jr. and K. D. Jones, "Analysis of Generator Forced Oscillations During MOD 25 Testing Exploiting Wavelets," HICSS-57, 2024, doi:10.24251/HICSS.2024.364.

The gap: from control design to CHiL testing

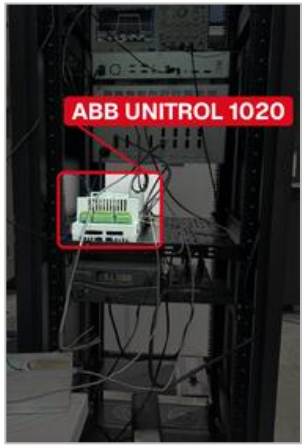
ALSETlab

What it takes today

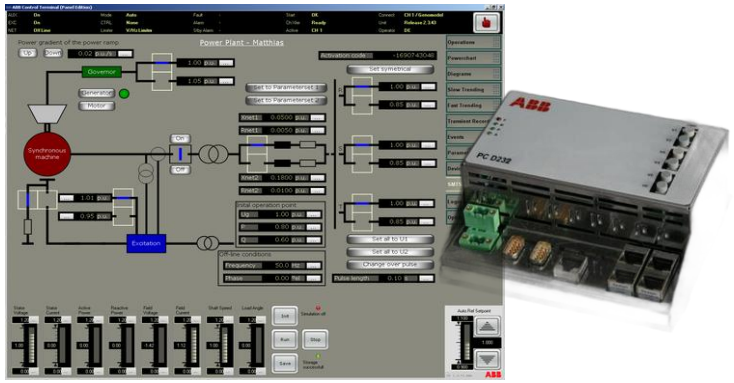
- **Offline tools cannot cross over** — power system models are monolithic off-line simulators, PSS®E and PSCAD®, models are not usable for real-time simulation.
- **Manual re-implementation breaks traceability** — the deployed code stops matching the physics model.
- **Control code is hardware-bespoke** — PSS implementations are written against embedded hardware specific platforms.
- **Real-time simulation platforms are expensive:**
  - **Require real-time capable plant models, costly to develop.**
  - General Purpose: dSPACE®, Typhoon HIL®, OPAL-RT®: typically, ≫ $50,000, plus license stack, requiring to re-built plant model.
  - Vendor Specific Platform: single purpose RTS (plant cannot be modified), ABB SMTS-RT 6000, limited to specific vendor controllers.

What we do instead

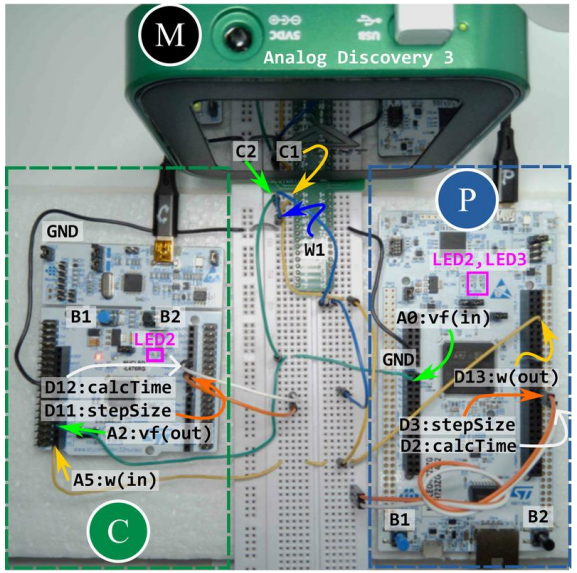
- **One acausal Modelica model** is the single source of truth — *for both* **controller** and **plant**
- **eFMI generates the code** — GALEC, then production C that is MISRA C:2023 and SEI CERT C compliant.
- **Validated at every level** — MiL → SiL → CHiL.
- **Deployed on < $50 STM32 boards** — **controller** and real-time **plant**



Own bench: ABB UNITROL 1020 interfaced with Typhoon HIL®



ABB's Synchronous Machine Transient Simulator SMTS-RT 6000



Two NUCLEO boards replace the real-time platform:

(left, C) one runs the **controller**,
(right, P) one runs the **plant**

(top, M) oscilloscope and signal generator for data acquisition and signal injection

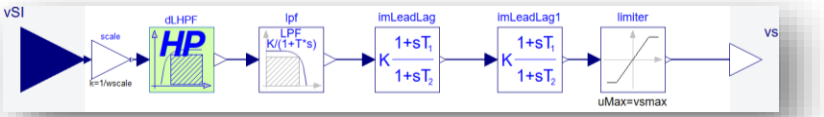
One source of truth with Modelica · traceability by design with eFMI · ≫ $50,000 platform → < $50 microcontroller

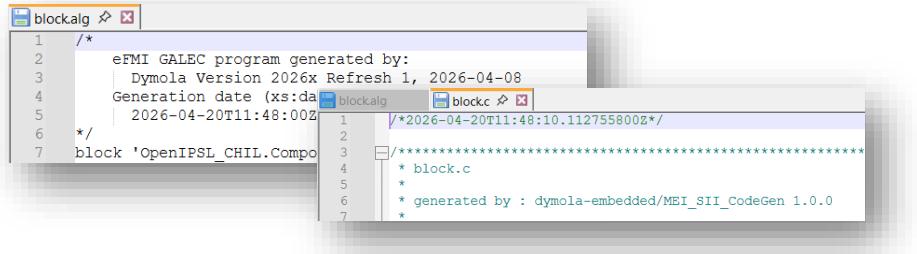
Our approach: one source of truth, generated down

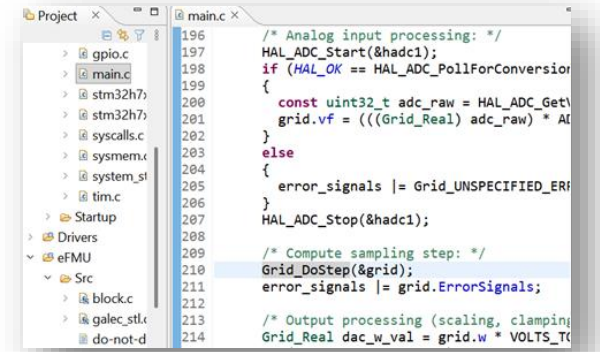
Modelica® model
acausal physics · controller + plant

eFMI® eFMU
GALEC → production C
MISRA C:2023 · SEI CERT C

STM32 microcontrollers
NUCLEO-L476RG + H723ZG
0.2 ms · 5 kHz



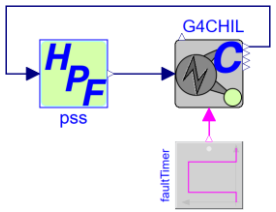


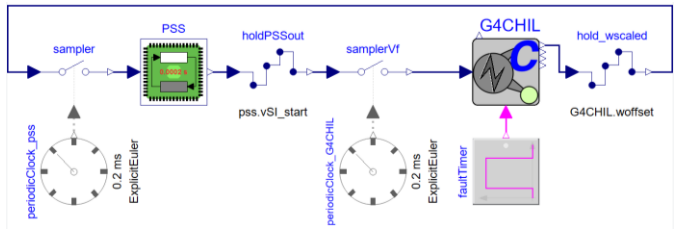


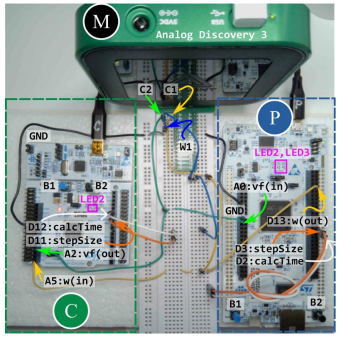
MiL
model-in-the-loop

SiL
software-in-the-loop

CHiL
controller-hardware-in-the-loop







- **Physics-based Modelica** is the single source of truth — **controller** *and* **plant**.
- **eFMI generates** production-ready, safety-critical C from that same model.
- **Validate at every abstraction level** — MiL → SiL → CHiL.

eFMI® in one slide

Algorithm Code (GALEC)

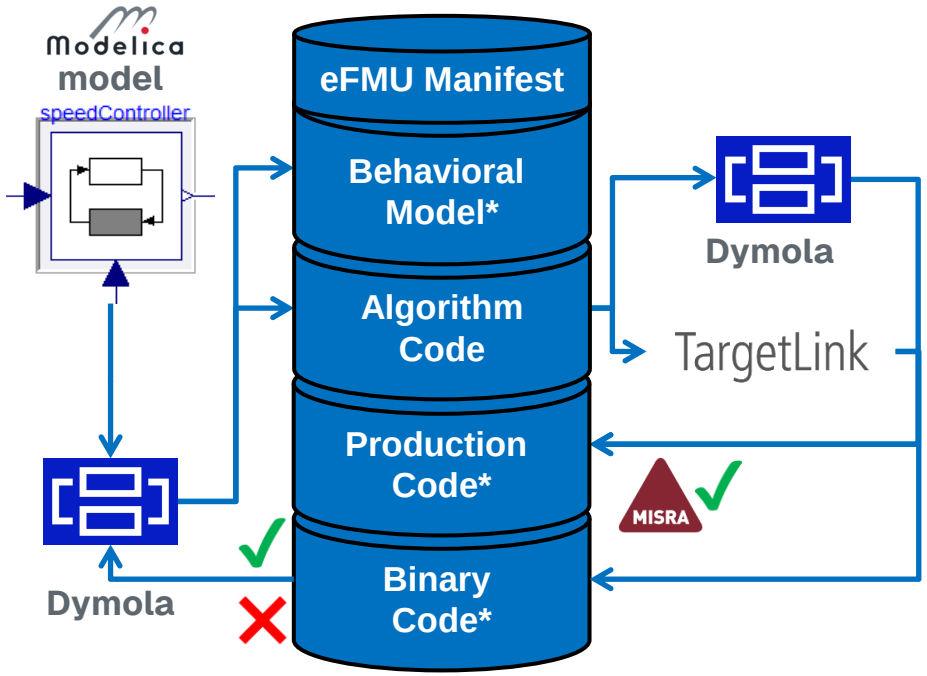
- GALEC — guarded algorithmic language for embedded control
- Standardized block interface and life-cycle
- High abstraction level (multi-dimensional arithmetic, rich set of built-in functions, e.g. interpolation 1-2-3D, solve linear equations, etc.)
- Guaranteed termination and maximum memory consumption, safe indexing
- Strict error handling (within normal control-flow, no exceptions)
- Self contained – no foreign function interface
- Requires solver to be inline-integrated

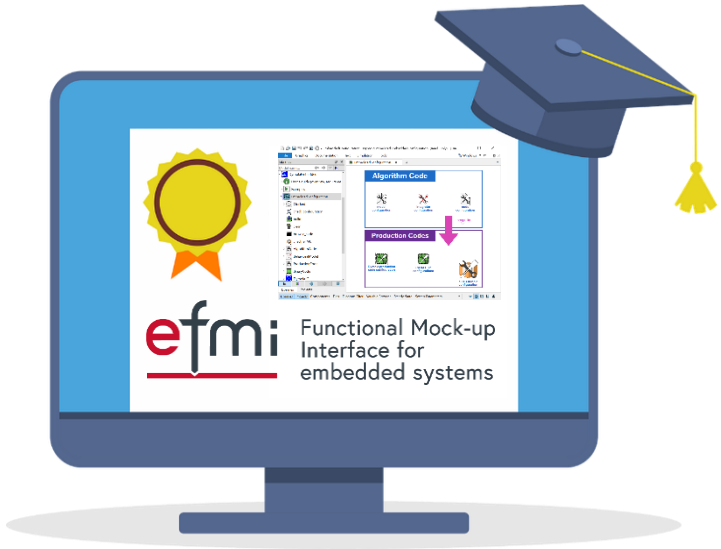
Production Code – Dymola's Features

- C17, generated from GALEC
- Compliant with MISRA C:2023, SEI CERT C
- 32- and 64-Bit floating-point precision
- No use of external libraries, only C Standard Library (mainly `math.h`)

Behavioral Model

- Manifest file with test scenarios, units, tolerances
- Variable or equidistant time grids
- Reference trajectories in CSV (described in manifest)





- **An eFMU is an open workspace container, *not a sealed artefact*** — it stays in development alongside the model.
- **GALEC is the pivot** — simulation tools target it; embedded code generators consume it, and benefit from its guarantees.

Every container links back to the same standardised block interface!

Traceability is structural, not documentary!

More details: https://www.efmi-standard.org/

OpenIPSL_CHIL: OpenIPSL for embedded real-time

ALSETlab

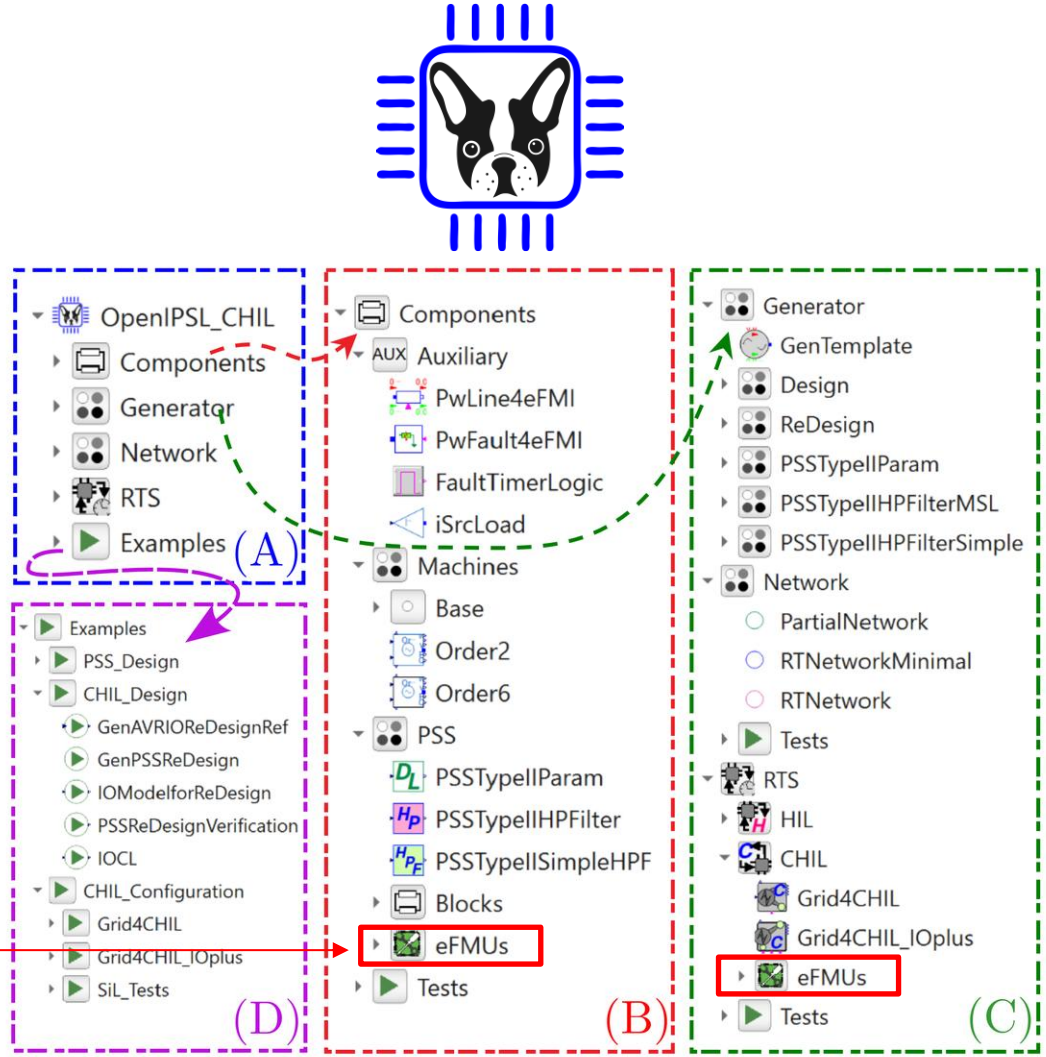
`OpenIPSL 3.1.0` — the starting point

- Open-source Modelica library for phasor-domain transient (RMS) power system simulation.
- Transmission, distribution with unbalanced three-phase, and mixed EMT / TS modelling.

`OpenIPSL_CHIL` — what this paper adds

- Extensions of OpenIPSL for embedded real-time applications.
- **(B) Components** — PSS variants, specialised blocks, and the OpenIPSL machine, line and fault models that needed modification for eFMU export.
- **(C)Generator / Network** — generator-unit and grid variants per design stage: continuous, redesign, embedded.
- **(C) RTS** — plant models for HiL and CHiL targets, with their eFMU configurations.
- **(D) Examples** — whole-system closed-loop MiL and SiL experiments.

Every sub-package ships its eFMU generation configuration (when appropriate) and unit tests.



OpenIPSL_CHIL library overview

Re-factoring the controller for eFMI Export: need → μ-controller

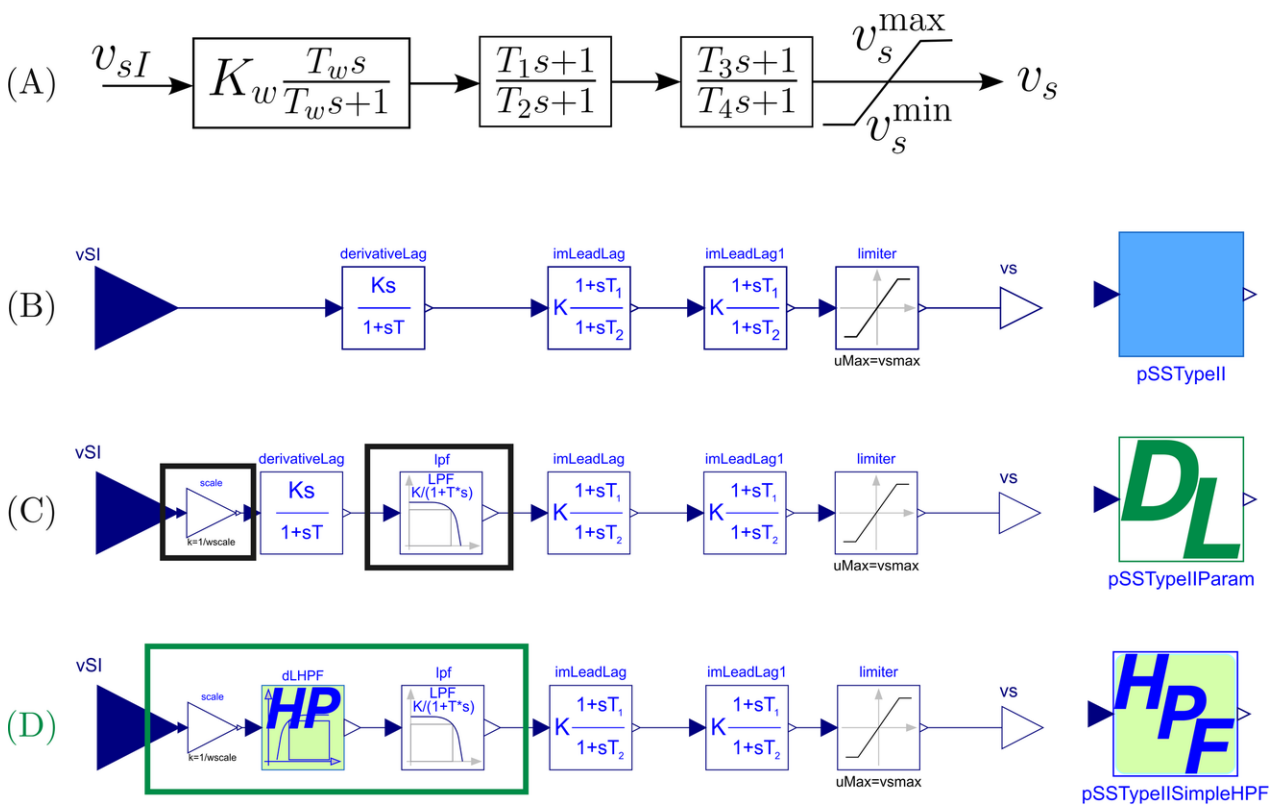
From textbook block to deployable model

- **(A)** is the textbook PSAT Type II, equivalently PSS®E STAB1
- **(B)** PSAT Type II / PSS®E STAB1: gain, washout, two lead-lag stages, limiter. Implemented in OpenIPSL as `PSSTypeII`.
- **(C) Input-dependent initialization:**
  - `vSI_start` as a tunable parameter;
  - Scaling: ×10 input scaling to use the DAC range;
  - LPF: a 5 Hz low-pass against analog noise.
- **(D) washout →** `HighPassSimple` — implements a first-order Butterworth high-pass to replace the "washout" (`derivativeLag`) block
  - one state, one differential equation,
  - one algebraic output, same
  - Benefits:
    - same parametrisation,
    - no differentiation!

```
block HighPassSimple
  parameter Real K = Kw*Tw;
  parameter Real T = Tw;
equation
  der(x) = (u - x)/T;
  y      = K/T*(u - x);
end HighPassSimple;
```

⇒ Every change is at the model-level: MiL test can show the effects of every change.

⇒ The generated code is never edited.



PSS variants: (A) generic block, (B) OpenIPSL, (C) modified for eFMI export, (D) final variant with the simplified high-pass filter

Re-factoring the Plant for eFMI Export: need → RT Simulator

Three changes for GALEC code generation:

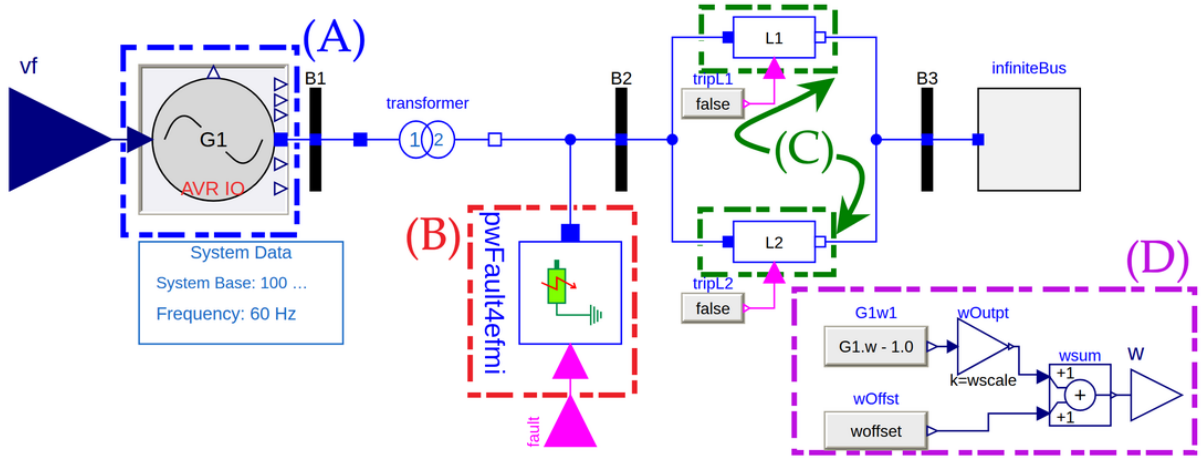
- **(A) generator unit**: `atan3` → `atan2` in `baseMachine`
  - GALEC has no `atan3`;
  - Mathematically equivalent since the phase angle φ is always 0 in OpenIPSL.

- **(B)/(C) fault and line-openings:** used to disturb the network and assess the controller's performance.

  Why this needs modifications?
  No global clock exists in CHiL, and fault to be triggered by user

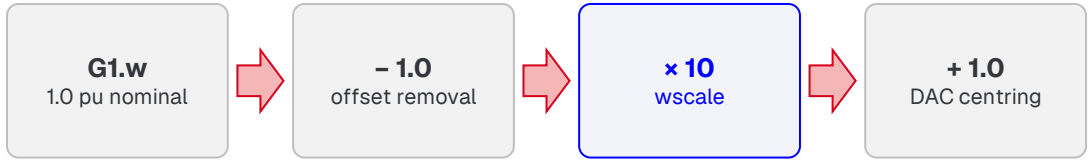
  Solution: implicit `time`-based switching → **explicit Boolean inputs**

- **(D) signal conditioning:**
  - STM32 boards equipped with 12-Bit, 0–3.3 V DAC
  - We need to:
    remove the 1.0 pu offset,
    scale by `wscale = 10`,
    re-offset to 1.0.
  - Speed deviations rarely exceed the third decimal — unscaled,
    →the signal would sit on the quantization floor.

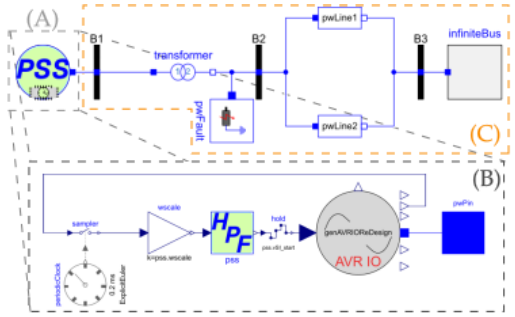


*The plant model: generator with ECS interconnected with a power grid — *.RTS.CHIL.Grid4CHIL*

Signal conditioning (D) — making the deviation visible to a 12-Bit DAC



MiL: five variants, one answer – understanding re-factoring impact



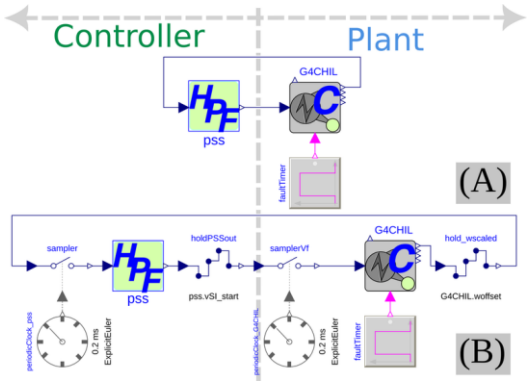
*Reference Model (**Sim1**)*

(A) Generator in power grid
(B) Inside the generator unit, showing the PSS
(C) Reminder of the grid

CHiL Models used for Simulations 2-5

(A) Continuous time representation

(B) Clocked Representation



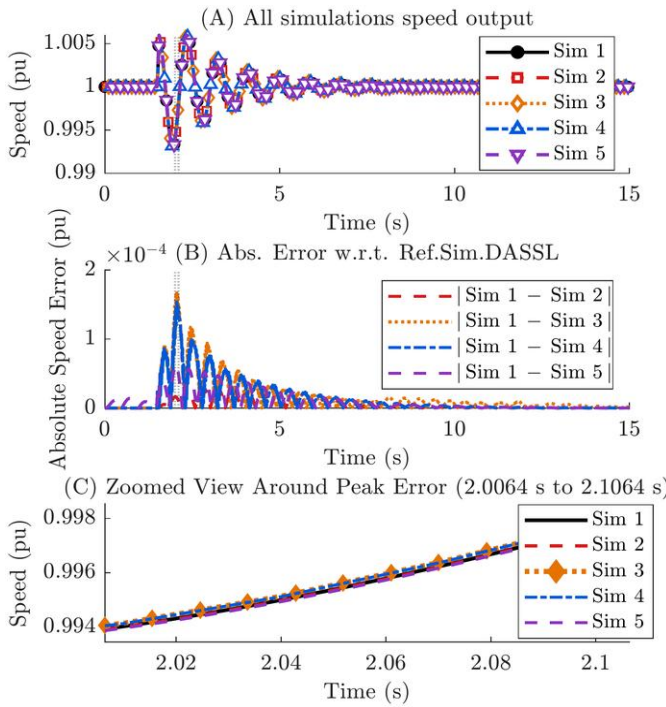
Models:

- ***Reference (Ref.) Model:*** represents the deployment of the digital control in a "real-world" power plant
- ***CHiL-Models:*** representation of the experimental controller-hardware-in-the-loop setup we will build with HW → continuous and clocked representation (both controller **and plant**)

Five cases, same experiment

- ***Sim 1:*** Ref. with DASSL, `Interval = 0.0002`
- ***Sim 2:*** Ref. with Exp. Euler, 0.2 ms fixed step
- ***Sim 3:*** CHiL (A) with DASSL, `Interval = 0.0002`
- ***Sim 4:*** CHiL (A) with Exp. Euler, 0.2 ms fixed step
- ***Sim 5:*** CHiL (B) ***clocked representation***, with DASSL, `Interval = 0.0002`

Explicit-Euler baseline penalty alone: max 3.02×10^{-5} pu, RMS 9.23×10^{-6} pu (in other words: noise is >>)



MiL simulations: all cases vs the DASSL reference

- **Visually indistinguishable** (RHS Fig) — the five cases overlay in panel (A); the errors are in (B) and (C).
- **Peaks in recovery, not in the fault** — the worst error occurs near t ≈ 2.06 s, during post-fault recovery, ***which confirms the numerical stability of the eFMI-targeted variants.***
- **RMS an order of magnitude below the maxima** — these are brief numerical transients, not a systematic bias.

1.69×10^{-4} pu
worst-case deviation, all variants

8×10^{-4} pu
12-Bit DAC quantisation floor

4.7 ×
margin — the refactoring is invisible to the hardware

eFMU generation and SiL testing using Dymola

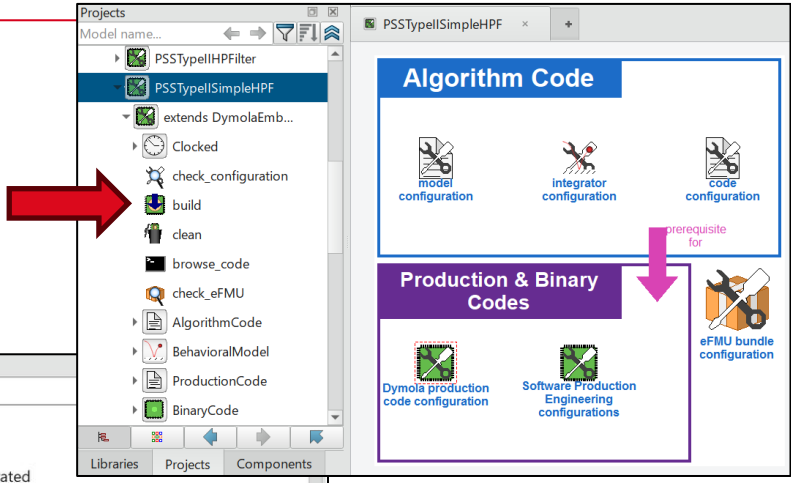
Configuration is just a Modelica package

- `DymolaEmbedded.EmbeddedConfiguration`:
  - *Model Configuration:* define the model subject to eFMU export
  - *Integrator Configuration:* define fixed sampling period (mandatory), and optional solver method for inline integration.
    Available solvers: Explicit Euler (mature), implicit Euler and Ronsenbrock (under dev).
  - *Code Configuration:* optional options to obfuscate GALEC code, keep multi-dimensional entries intact, etc.
  - *Software Production Engineering Configurations:* define the floating-point precision (fpp), target architectures.
- **Fully scriptable automation fuctions** in `*.mos`: `build()`, `check_codes()`, `build_tests()`.

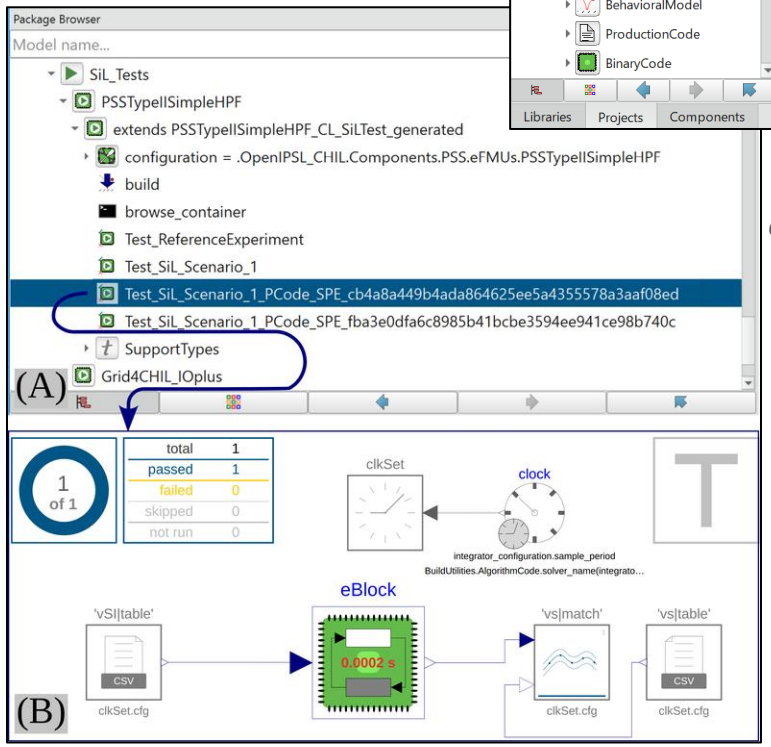
SiL Experiment Packages

- Derived from the MiL experiments.
- **No new test authoring** — reference trajectories come from the existing MiL experiment; tolerances are the only user input
- **Deviations:**
  - Controller ≈3.65 × 10^{-4} (32-Bit fpp) / ≈1.39 × 10^{-16} (64-Bit fpp)
  - Plant ≈1.61 × 10^{-4} (32-Bit fpp) / ≈1.78 × 10^{-14} (64-Bit fpp)

32-Bit fpp goes on the hardware: the NUCLEO-L476RG (controller) accelerates only 32-Bit fpp, and 64-Bit fpp plant code left too little real-time headroom.

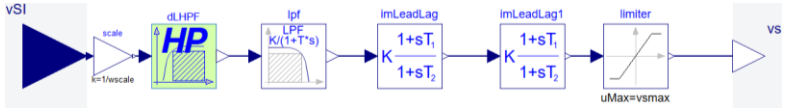


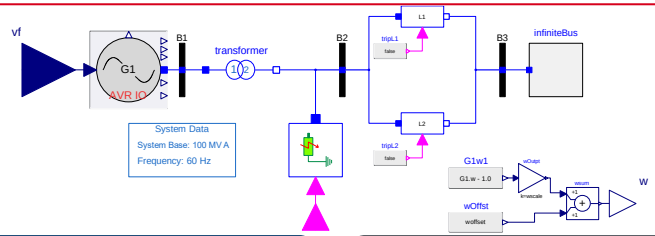
`DymolaEmbedded.EmbeddedConfiguration` — eFMU configuration for the PSS controller



SiL experiment package derived from the MiL experiment, & test model for the 32-Bit production code

eFMI generated code and exchange with FMI: 175× smaller FMUs





31 kB
controller, eFMU source code FMU

5.41 MB
controller, ordinary source code FMU

38 kB
plant, eFMU source code FMU

5.49 MB
plant, ordinary source code FMU

Compiled binaries, the same story

- *eFMU-backed FMU:* 176 kB controller, 220 kB plant
- *Ordinary Dymola binary FMU:* 2.84 MB and 2.93 MB

Why it matters

- ***It is what lets the code fit a microcontroller!***
- No dependency beyond `math.h` — portable by construction

≈175× smaller than the same model exported as an ordinary source-code FMU!

- **eFMU exported as source code FMU, the same artefact imports** into dSPACE ConfigurationDesk®, TwinCAT® or Silver®, which compile and deploy it themselves — *so eFMI reaches the established real-time platforms.*
- **Exports also as** a Simulink® C Function block or an Arduino® sketch, *with the production code used as is.*

Production code quality: MISRA C:2023 and SEI CERT C

The analysis is deliberately as strict as it goes

- **Cppcheck Premium 26.3.0** — exhaustive check level, style and information analyses, inconclusive reports included, MISRA C:2023 and SEI CERT C enabled
- **clang-tidy (Clang Tools 19.1.5)** — all checks except four, each excluded for a stated reason: short physics variable names, magic numbers in equations, loop unrolling left to the compiler, and struct alignment left to the compiler

204 reports, none of them a defect

- **The only reports clang-tidy raises** are 64-Bit literals narrowed into 32-Bit variables — intended, but Dymola omits the `F` suffix that would make it explicit — and false-positive uninitialized locals

One genuine-looking MISRA violation: "A loop counter shall not have essentially floating type", false-positive, Cppcheck bug — fixed upstream in 26.6.0 based on our report

Defects: galec_stl.c

premium-misra-c-2023-8.7 268
premium-misra-c-2023-8.7 279
premium-misra-c-2023-8.7 290
premium-misra-c-2023-8.7 304
premium-misra-c-2023-8.7 315
premium-misra-c-2023-8.7 336
premium-misra-c-2023-8.7 343
premium-misra-c-2023-8.7 355
premium-misra-c-2023-8.7 369
premium-misra-c-2023-8.7 395
premium-misra-c-2023-8.7 407
premium-misra-c-2023-8.7 421
premium-misra-c-2023-8.7 433
premium-misra-c-2023-8.7 447
premium-misra-c-2023-8.7 459

```
2370 /*
2371         Are the values of multi-dimension x finite and strictly monotonically
2372         increasing?
2373 */
2374 static inline bool SPE_isIncreasingMD(
2375           const SPE_Real_H225c1baf6cf5a31bc9b0c38998c32298f6f0531c_cb4a8a449b
2376         , const size_t r_size)
2377 {
2378         SPE_Real_H225c1baf6cf5a31bc9b0c38998c32298f6f0531c_cb4a8a449b4ada8646
2379         size_t r_index = ((size_t) 0);
2380         while ((r_index < r_size)
2381                 && SPE_isFinite_H225c1baf6cf5a31bc9b0c38998c32298f6f0531c_cb4
2382                 && (0 != isgreater(r[r_index], last)))
2383         {
2384                 last = r[r_index]; <--- A loop counter shall not have essentia
2385                 r_index++;
2386         }
2387         return r_index == r_size;
2388 }
```

Excerpt of the Cppcheck Premium report for the 32-Bit floating-point precision production code generated for the PSS

159 × MISRA 8.7

- Advisory: external linkage in one translation unit
- All are GALEC built-ins, intended as a shared library
- **Can be ignored.**

23 × SEI CERT DCL06-C

- Use symbolic constants for literals
- Stem from magic numbers in physics equations
- **Can be ignored too.**

3 × SEI CERT INT30-C

- Unsigned wrap
- Impossible: `index - 1` ≥ 0
- GALEC indexes from 1
- **False positives.**

STM32 integration and the CHiL bench

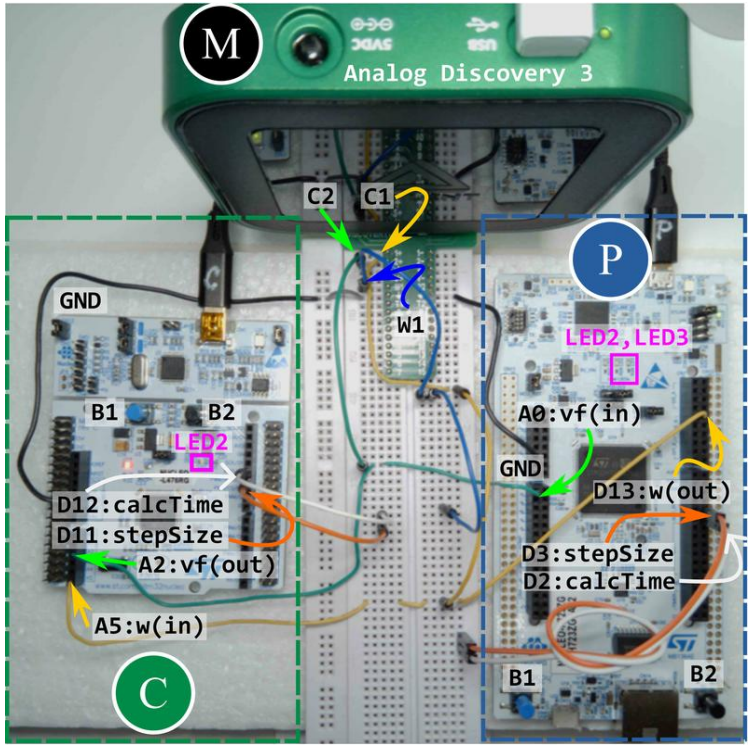
Controller
- NUCLEO-L476RG
- 80 MHz Arm Cortex-M4
- Hardware FPU, 32-Bit FPP

Plant
- NUCLEO-H723ZG
- 550 MHz Arm Cortex-M7
- For the matrix algebra

```
freq = (550/2 MHz) / ((ARR+1)(PSC+1))
     = (275 MHz) / ((199+1)(274+1))  =  5 kHz
```

The board timer must match the eFMU sampling period exactly

- **Production Code containers are symbolically linked** into the STM32CubeIDE projects:
  - re-export from Dymola, just refresh and rebuild
  - No manual re-import → makes the prototyping loop fast.
- **Integration is confined to** `main.c`:
  - include the block header, call `Startup()` and `DoStep()` on the timer tick, and
  - handle ADC/DAC scaling and clamping around it.
- **Interactive testing: —**
  - onboard buttons inject the plant fault and engage or disengage the PSS;
  - LEDs signal overruns and eFMU error signals.



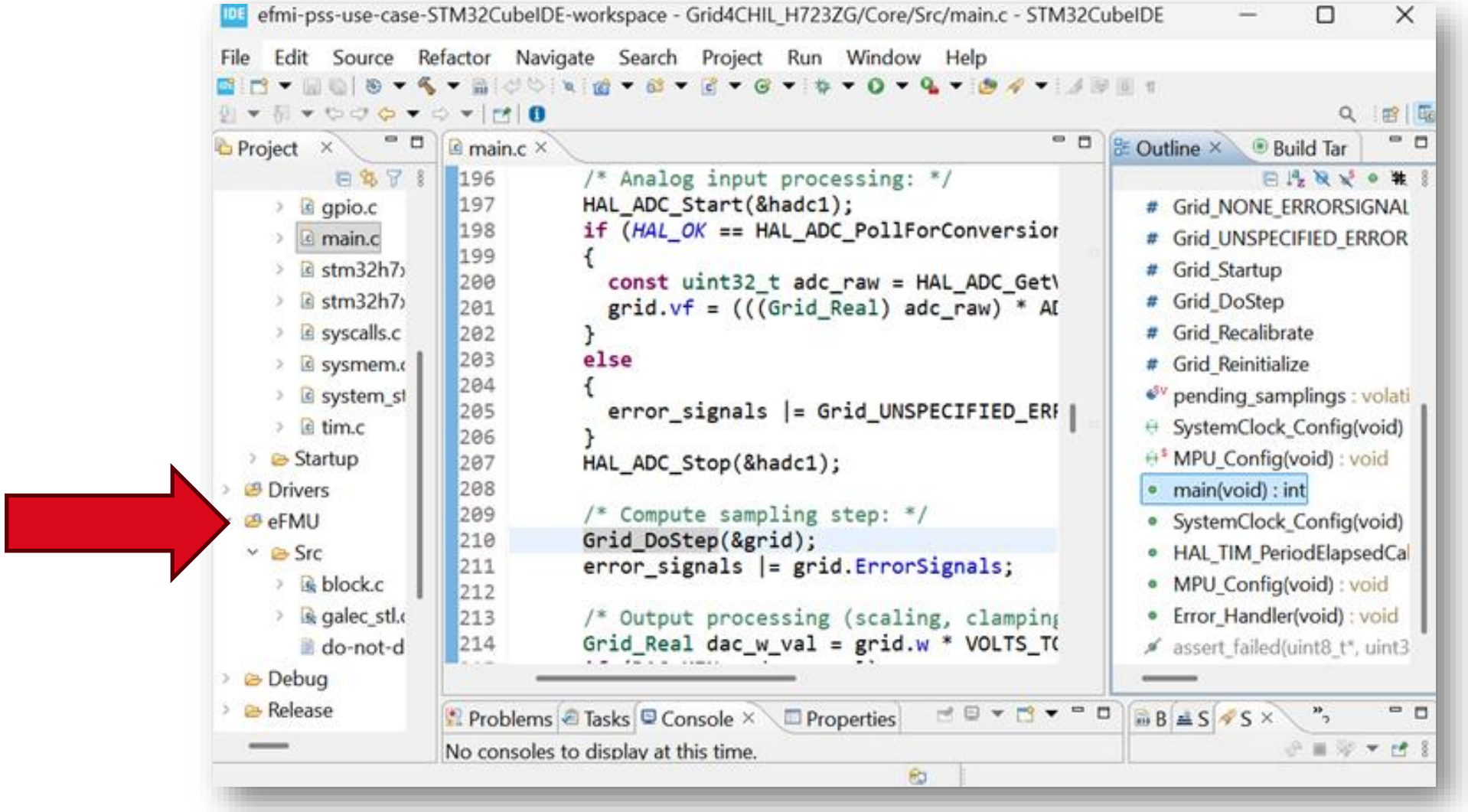
CHiL test bench: (C) controller, (P) plant, (M) Analog Discovery 3

Total hardware cost: < $90.00*
- NUCLEO-L476RG - $21.30 at DigiKey
- NUCLEO-H723ZG - $40.81 at DigiKey
- Break boards + patch cables: ~$11.00-$25.00

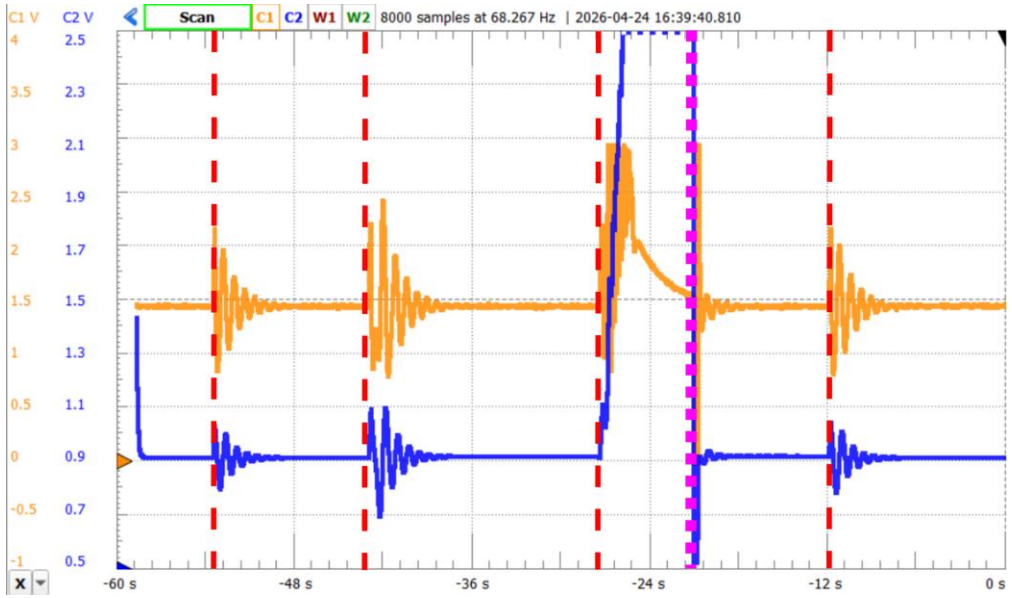
* Excl the Analog Discovery 3 ~$379.00

STM32 System Integration Code

All details in the paper!

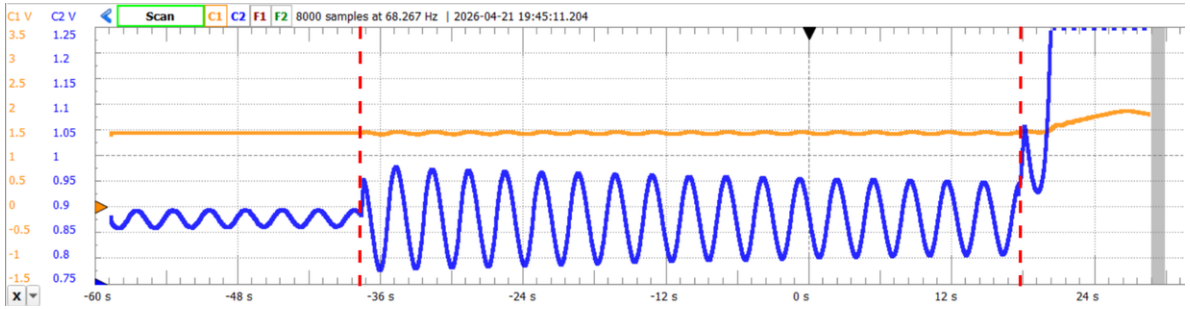


CHiL results: the controller does its job



PSS engaged — damped within seconds.

- *Legend:*
  - **Blue:** scaled speed from NUCLEO-H723ZG, output of plant
  - **Orange:** PSS output voltage (pu) from NUCLEO-L476RG, input to plant
  - **Red:** fault applied via push-button B1 on NUCLEO-H723ZG
  - **Magenta:** reset applied on both boards via push-button B2.



PSS disengaged — sustained oscillations, then instability.

- **Engaged:**
  - The controller voltage provides immediate active damping
  - Generator speed damps to steady state within seconds.

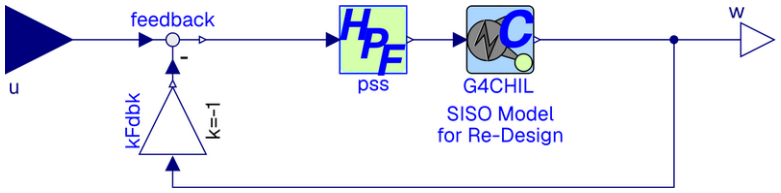
- **The limit is real**:
  - Faults were held long enough for the speed to run away,
  - The controller has a finite window of operating speeds it can stabilise

- **Disengaged:**
  - The plant behaves as an undamped pendulum;
  - A later fault excites it further and it becomes unstable.

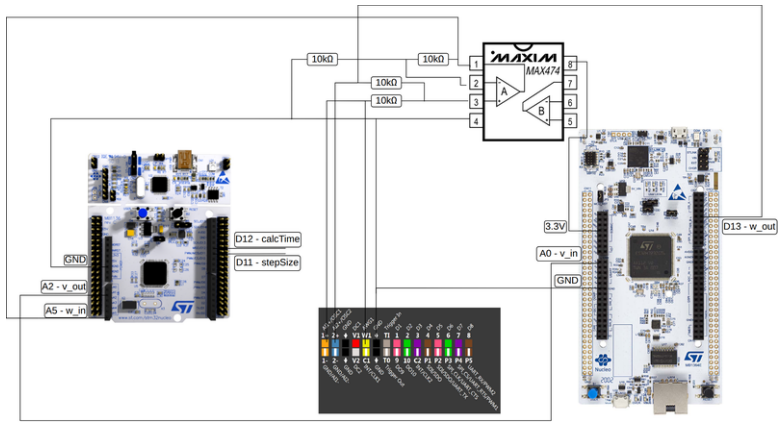
±0.2 pu
recovers, PSS engaged

±0.1 pu
unstable, PSS off

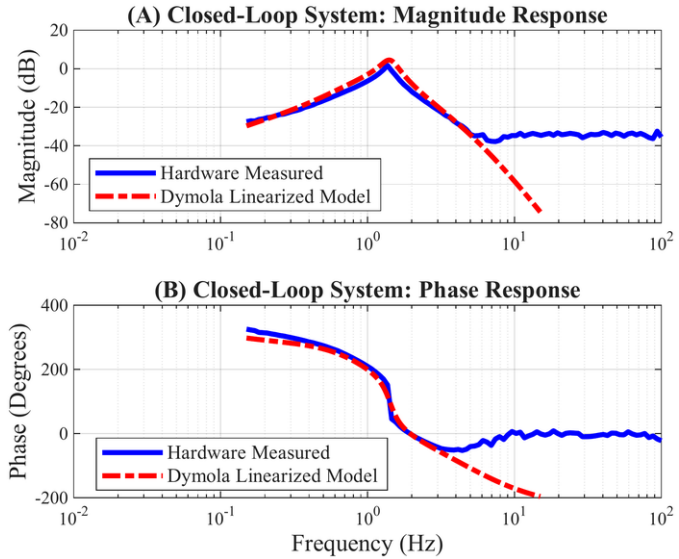
Closed-loop frequency response: model against hardware



Dymola closed-loop model with input and output interfaces



Closed-loop wiring for network analysis with the Analog Discovery 3

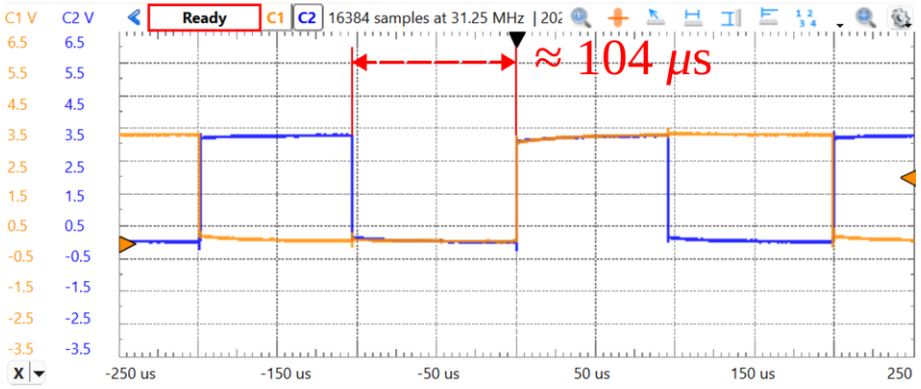


Closed-loop response: hardware (blue) vs Dymola linearisation (red)

- **Logarithmic sweep 0.15–100 Hz at 50 mV** — amplitude chosen so the plant Excitation Control System (AVR limiters) are never reached.
- **A summing amplifier was needed:**
  - the STM32 ADC cannot process negative voltages,
  - A MAX474 biases the ±50 mV stimulus onto the plant's flat 1.0 V DC speed output, giving 0.95–1.05 V into the controller.

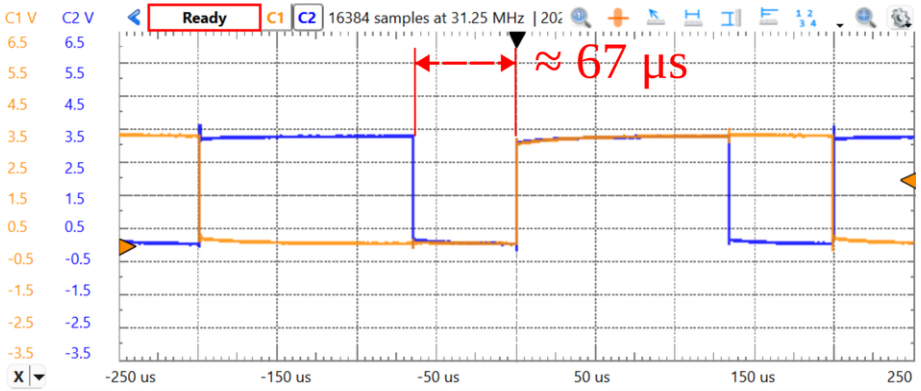
The eFMU reproduces the offline model in closed loop; the flattening near 5 Hz is the instrument noise floor and the onboard converters, not the model.

Real-time headroom: no overruns, with margin



(A)

52 %
plant headroom
steady state
(96 µs)



(B)

33.5 %
plant headroom
sustained fault
(133 µs)

Computation time (blue) vs sampling timer (orange) of the NUCLEO-H723ZG, Runs the plant model

- **Headroom is measured, not estimated —**
  - a GPIO pin is driven high at the start of a sampling and low at the end;
  - a second pin toggles on every timer tick, giving the oscilloscope a reference grid.
- **Why the fault case is the real test —**
  - plant's GALEC code is dominated by `solveLinearEquations(A, b);`
  - without a fault, A and b are sparse → generated solver skips the zeros.
- **For the plant, the debug build overruns:**

  → This is why it is important to have the ability to do SiL testing

- **What about the PSS controller headroom?**
  - Release build: 92.17 % headroom (15.67 µs)
  - Same for steady state or fault tests

What that buys for the PSS ?

- Controller Debug build also deployable: 82.8 % headroom
- Room for richer stabilizers, and for -O0 where -O3 is not allowed

What does not fit on the plant board (NUCLEO-H723ZG)?

- Plant, Debug build: 239 µs — 19.5 % overrun
- Plant, 64-Bit FPP Release: 180 µs — only 9 % headroom

Conclusions — and the repository is open-source

What we showed

- **A complete MBSE chain** from acausal Modelica physics to MISRA-compliant embedded C, traceability preserved by design thanks to eFMI std.
- **Validation at every level** — once SiL passes, hardware integration is mostly memory fit and timing.
- **eFMI is viable for the plant too**, not only the controller — this surprised us. We planned to use a real-time simulator for the plant but tried eFMI out of need, as the RTS vendor has poor FMU support (only works with models exported from a specific tool).
- **≫ $50,000 platform → < $100 microcontrollers**, with free tooling apart from Dymola.

What comes next

- More PSS models and board templates; online recalibration for adaptive tuning; eFMU integration on dSPACE, Typhoon HIL and OPAL-RT.

`github.com/ALSETLab/efmi-pss-use-case` · **3-Clause BSD**

Scan for models, firmware, measurements and analysis scripts

Models and tests

- `models/` — OpenIPSL_CHIL:
  PSS variants,
  real-time-suited plant models,
  eFMU configurations,
  unit tests
  etc

Firmware

- `firmware/` — STM32CubeIDE projects for controller and plant, with the production code symbolically linked

Evidence

- `measurements/`, `postprocessing/`, `logs/check-code/` — raw CHiL captures, the scripts that reproduce every figure, and the full clang-tidy and Cppcheck reports

Thank you — Questions?



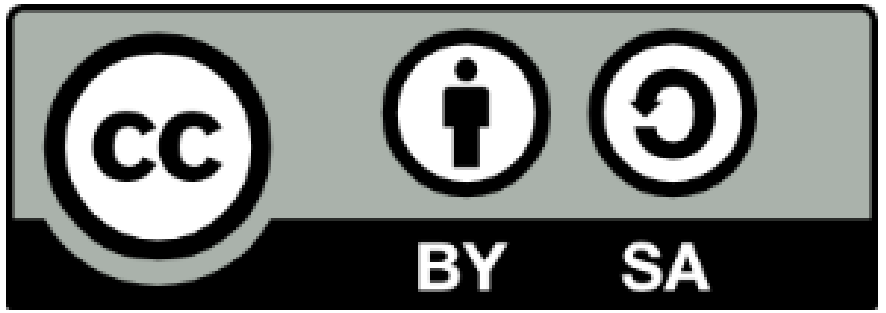
Luigi Vanfretti · Professor, ECSE · Rensselaer Polytechnic Institute
vanfrl@rpi.edu

`github.com/ALSETLab/efmi-pss-use-case`





© 2021-2026, Luigi Vanfretti, Christoff Bürger, Modelica Association and contributors.



This work is licensed under a *CC BY-SA 4.0* license.

Modelica® is a registered trademark of the Modelica Association.
eFMI® is a registered trademark of the Modelica Association.
FMI® is a registered trademark of the Modelica Association.

Third party marks and brands are the property of their respective holders.