

Methods and tools for porting legacy software to multi-core platforms

Stefan Resmerita

University of Salzburg and Chrona GmbH

Joint work with A. Naderlinger, A. Poeltzleitner, S. Lukesch and
W. Pree

Overview

- ▶ Verification of data consistency in multi-core software
 - Concurrency Analysis
- ▶ Correct-by-construction methods
 - Logical Execution Time
- ▶ Validation of system behavior by platform-aware simulation
 - Software in the loop, hardware in the loop
- ▶ Test-centered porting of legacy SW to multi-core
- ▶ Testing the tools
 - Automatic generation of test cases
- ▶ Use cases
- ▶ Demo

Data issues in single-core versus multi-core software

- ▶ Data consistency (DC)
 - Access constraints to shared memory
 - Non-blocking: buffering
 - Blocking: mutual exclusion
- ▶ Data dependency (DD)
 - Ordering constraints for executions of functions representing data producers and consumers
 - Stay close to original legacy DD

Verification of Data Consistency: Problematic Access Patterns

- ▶ Take a sequence of three atomic accesses to the same shared memory location a in two concurrent processes P and Q , of the form:

$$\alpha : [r \mid w]^P(a) \rightarrow w^Q(a) \rightarrow r^P(a)$$

- ▶ Notations:
 - Sequence of statements executed in P between the two accesses to a : s_1, s_2, \dots, s_n
 - Statement in Q where a is accessed: s^*
 - Set of locks held by process P at statement s : $L^P(s)$
- ▶ Sequence α is a **PAP** if:

$$\exists k \in \{1, \dots, n\} \text{ s.t. } L^P(s_k) \cap L^Q(s^*) = \emptyset$$

PAP tool: Auxilia

- ▶ Static analysis of source code
- ▶ Sound (conservative)
- ▶ Measures to reduce number of false positives
 - Use timing information
 - Filter out initialization functions
- ▶ Integrated in Eclipse
- ▶ Lean, fast
- ▶ Simple user interface
- ▶ Easy to integrate in existing tool-chains
- ▶ Output in XML and PDF

Auxilia snapshots

Auxilia concurrency analysis

Basic configuration

Select the destination project.
autodemo_mc

Input file for top level function definitions
C:\Data\work\chrona\runtime-EclipseAppli

Folder for result files
C:\Data\work\chrona\runtime-EclipseAppli

Pattern for Locking

Pattern for Unlocking

Pattern for DI (optional)

Pattern for EI (optional)

Types of analysis
 Shared variables Problematic Access

? < Back

Auxilia concurrency analysis

Advanced configuration

Filters

Function filter (exclude)

Variable filter (include)

Module filter

Run time optimization
 Reuse previous Lockset model Keep temporary files in temp folder

Call graphs for PAPs
 No call graphs
 One call graph per PAP
 All call graphs

Output format
 LateX+PDF
 XML

Output actions
 Open output file(s)
 Open output folder

? < Back Next > Finish Cancel

Auxilia output example (pdf)

The Write reference in `ApplicationTasks.c:checkGate()`

```
28 gateState = -1;
```

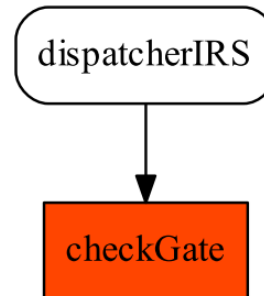
can write between Write reference in `Gear.c:Gear_gear_state()`:

```
111 *p = -1;
```

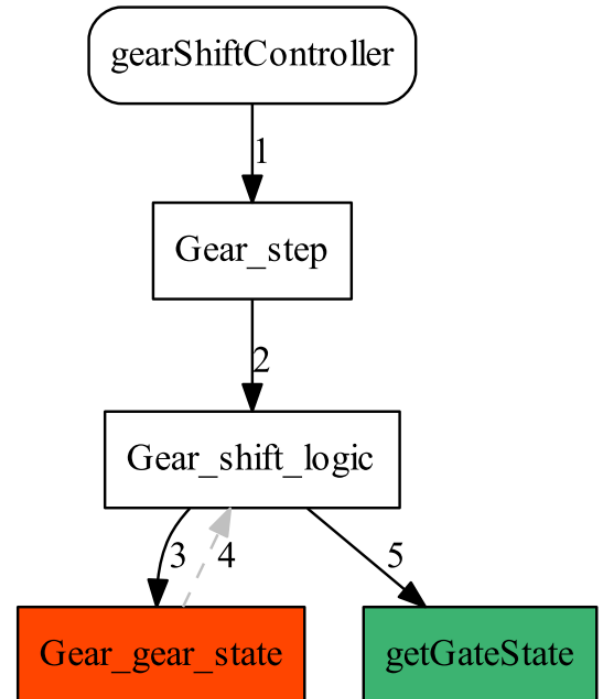
and Read reference in `Gear.c:getGateState()`:

```
222 i=gateState;
```

Key reference

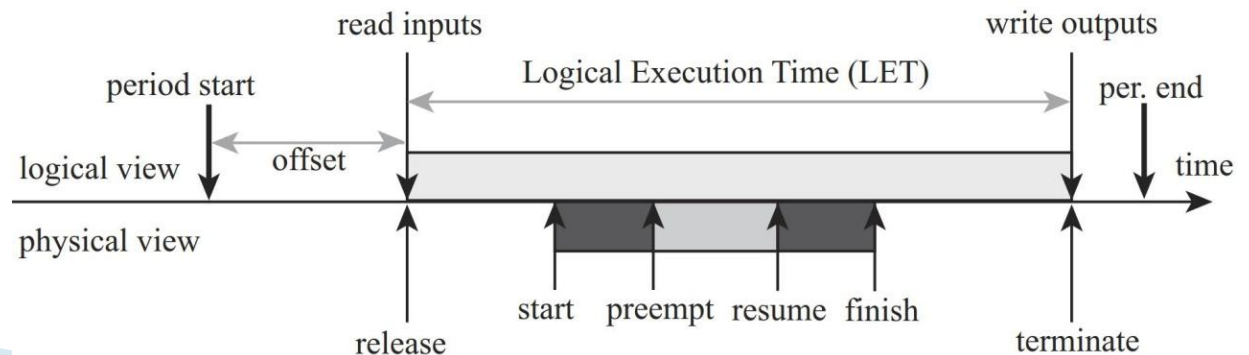


Other references

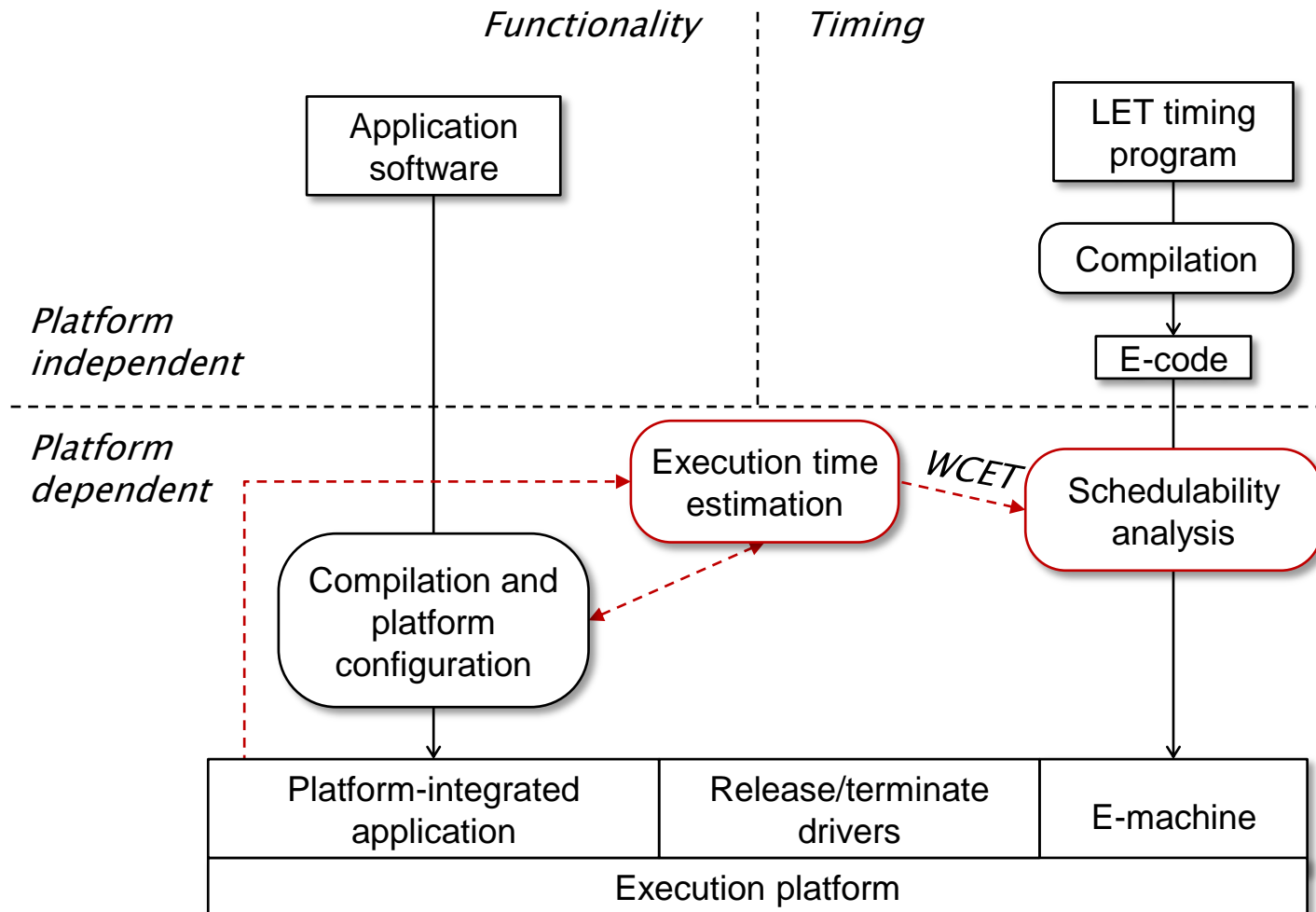


Correct-by-construction design: The LET Programming Model

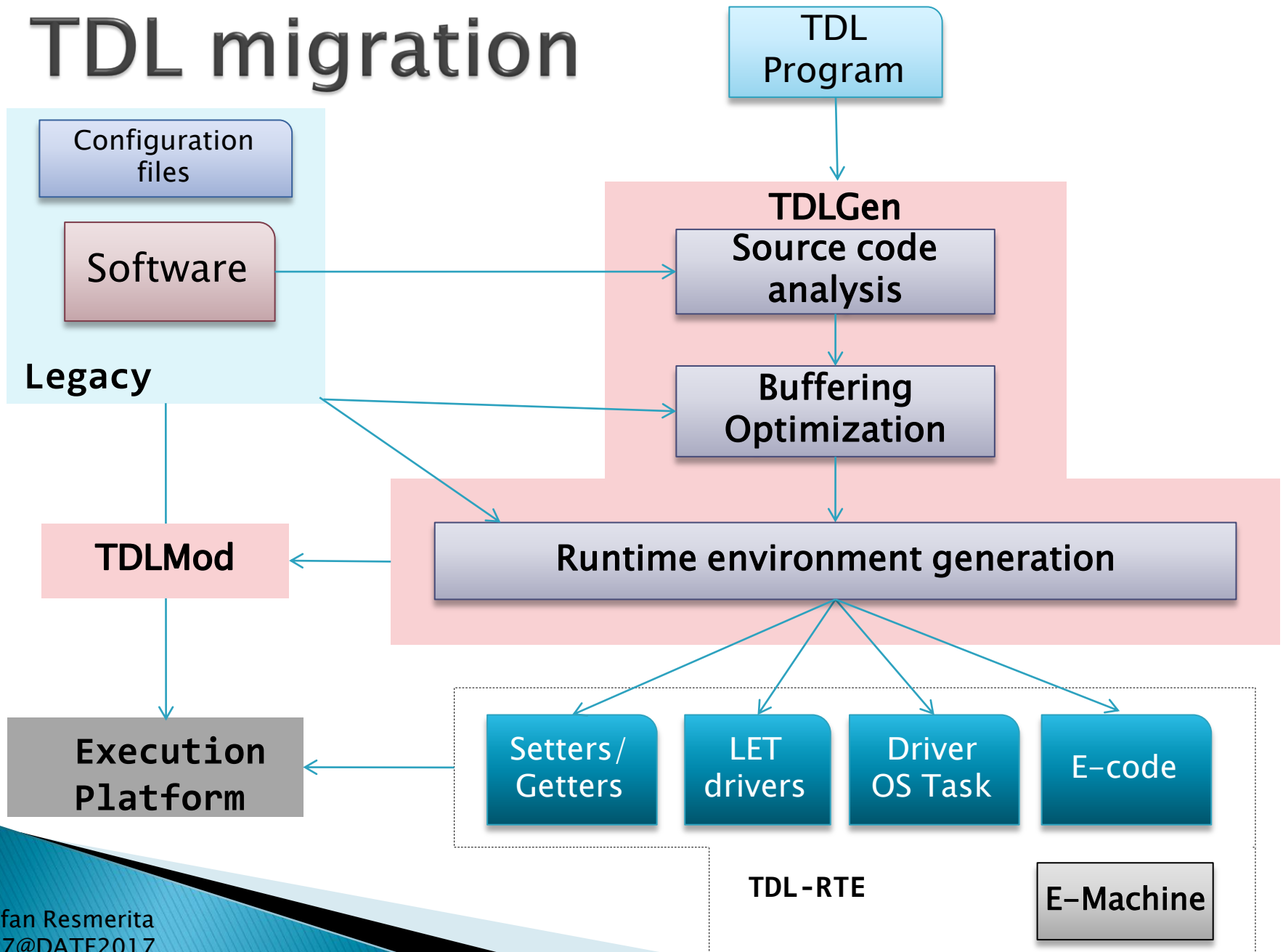
- ▶ Logical execution time (LET) for periodic software functions
- ▶ Inputs are read at the start of the LET
- ▶ Outputs are made available at LET end
- ▶ Physical execution (managed by platform scheduler) takes place within the LET
- ▶ Deals with **both** DC and DD!



LET-based design



TDL migration



TDLGen

runtime-EclipseApplication - Ovid - - Eclipse

Quick Access

Pro

Demo

- Includes
- src
- TDL demo.tdxl

TDL demo.tdxl

TDL Tasks

TDL Task	Function
▼ T1	
	Spark_step
▶ T2	
▼ T3	
	Throttle_step
▶ T4	

Modules

Module	Mode	Period	Time Unit	Initial	Invocation	Offset	Period	LET
▼ Module1								
	▼ Mode1C1	20 ms		true	T2	1	20	2
					T3	4	20	5
▼ Module2								
	▼ Mode1C2	10 ms		true	T1	0	10	2
					T4	8	10	2

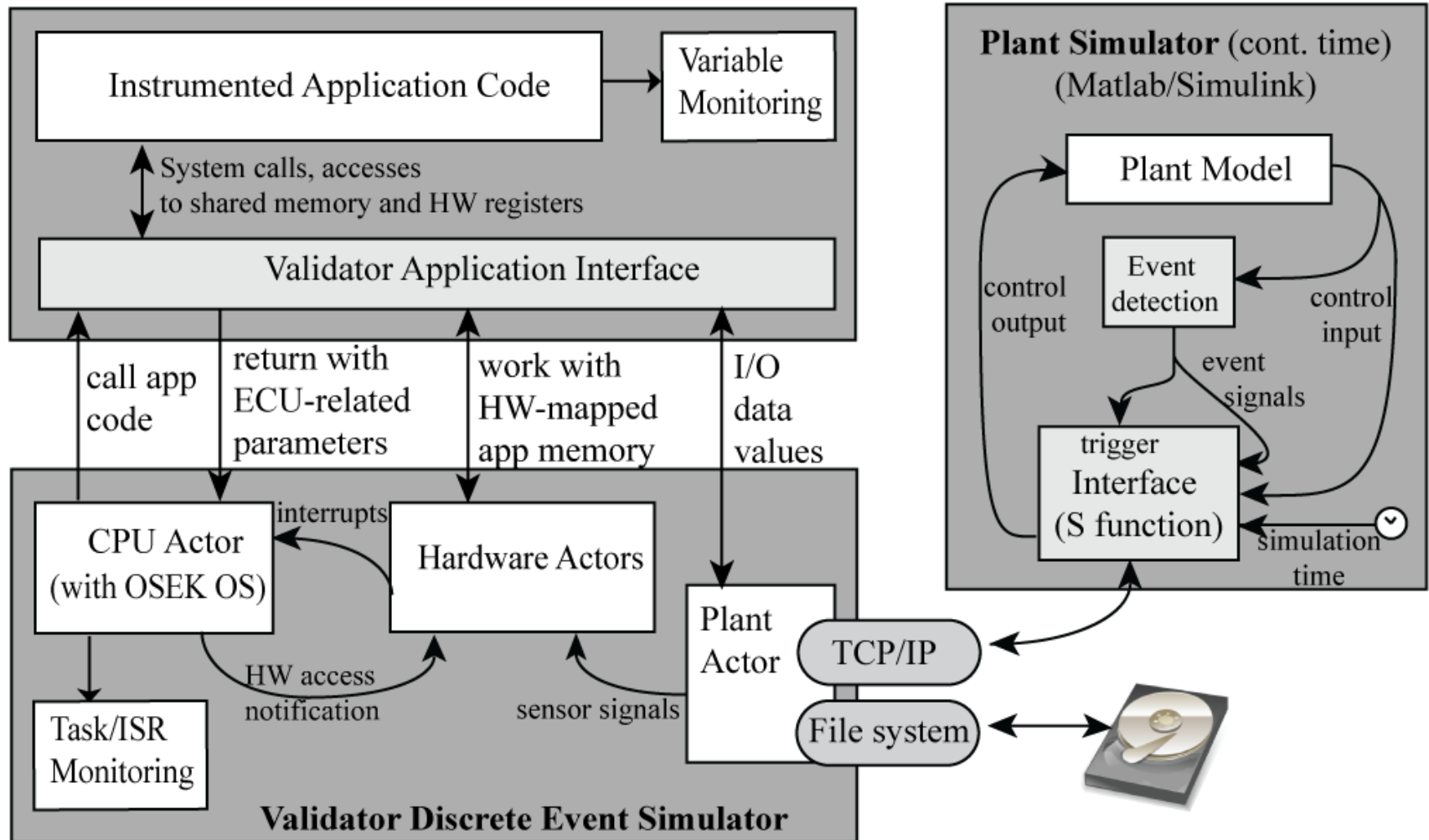
Application | Target | Program | Transitions | Mappings | XML

Program Timing: Program -> Aurix

0 items selected

392M of 741M

Validation: The Validator tool



Validator snapshot

The screenshot displays the Eclipse IDE interface for the AUTOMODEL/Autoexample-model.ovd project. The main workspace shows a block diagram with the following components and connections:

- SimulinkBridge**: Has an input port `CA10Deg` and an output port `trigger1ms`.
- InterruptController**: Has an input port `IRQ_trigger1ms` and an output port `internal...`.
- CPUScheduler**: Has an input port `interrupts` and an output port `hardware...`.
- Crank Timer**: Has an input port `input10Deg` and an output port `toIntC`.
- RTU**: Has an input port `toIntC` and an output port `fromCPU`.

Connections in the diagram:

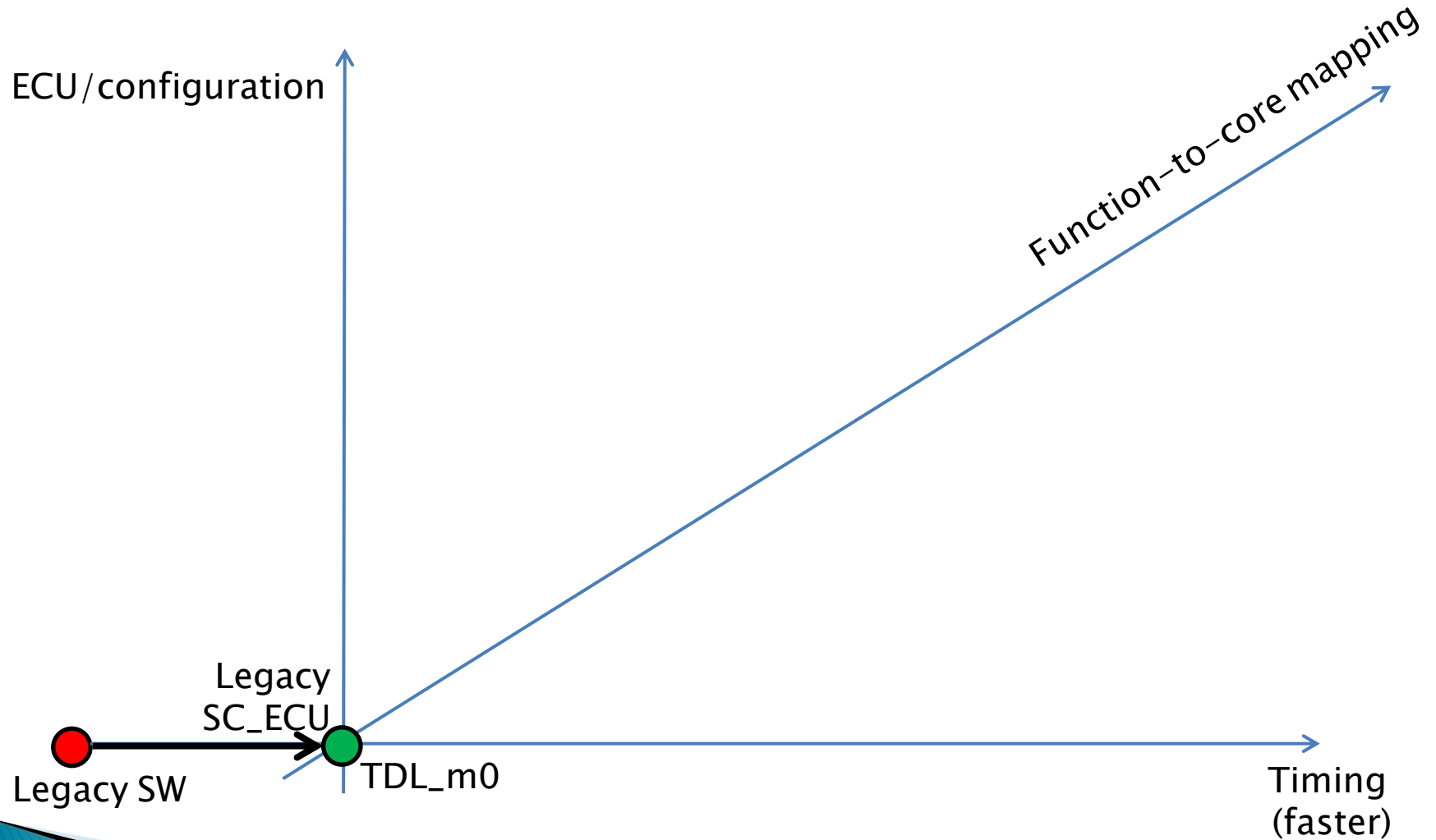
- `CA10Deg` (SimulinkBridge) connects to `input10Deg` (Crank Timer).
- `trigger1ms` (SimulinkBridge) connects to `IRQ_trigger1ms` (InterruptController).
- `input10Deg` (Crank Timer) connects to `toIntC` (RTU).
- `toIntC` (RTU) connects to `toIntC` (InterruptController).
- `toIntC` (InterruptController) connects to `interrupts` (CPUScheduler).
- `hardware...` (CPUScheduler) connects to `fromCPU` (RTU).
- `fromCPU` (RTU) connects to `fromCPU` (InterruptController).

The left sidebar shows the Project Explorer and Outline. The right sidebar shows the Palette and Actors. The bottom pane shows the Properties view for the `IdleSpeedControllerTask`.

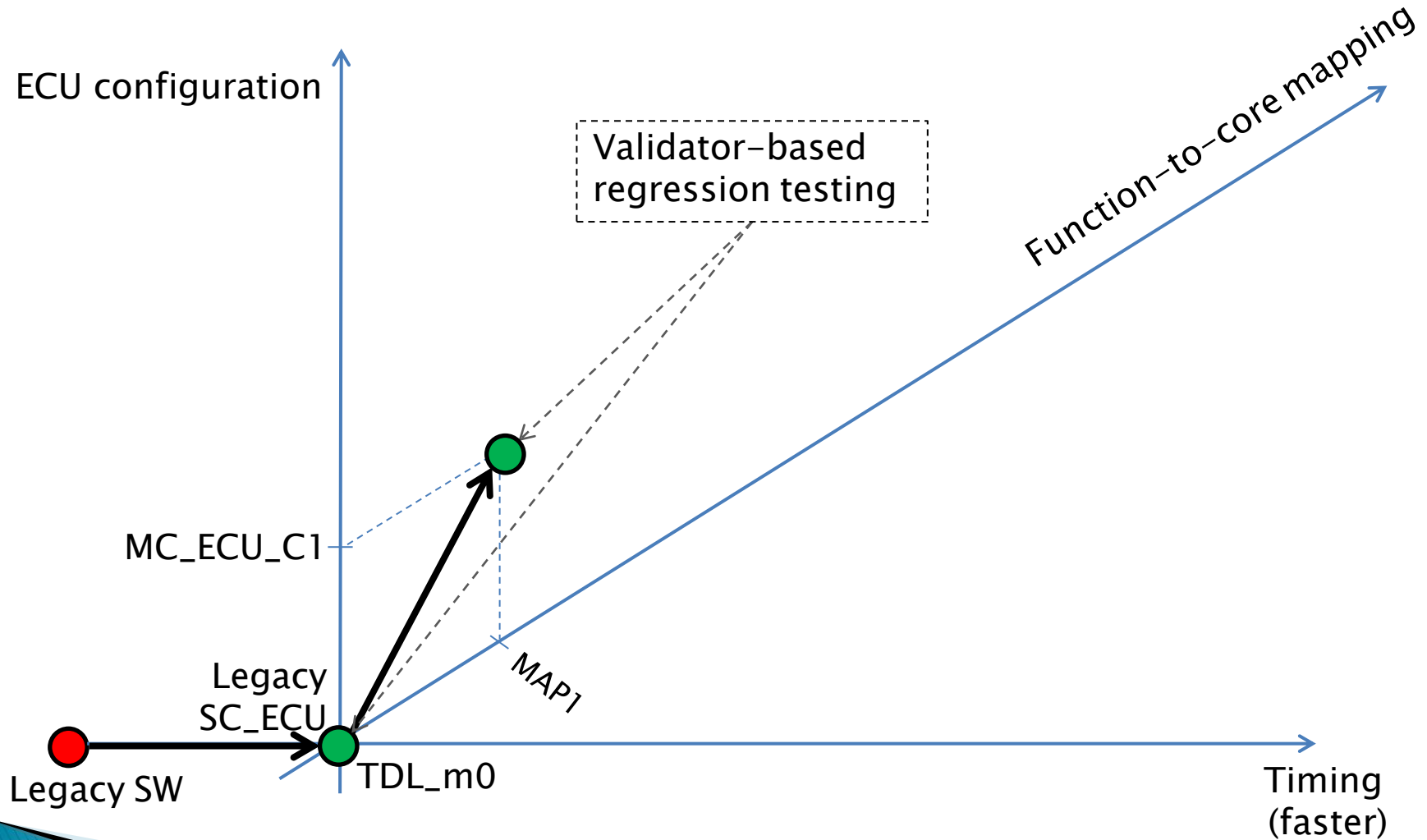
Properties view for IdleSpeedControllerTask:

Property name	Property value
priority	11
ID	6
CFunction	idleSpeedController
Autostart	false
Activation	1

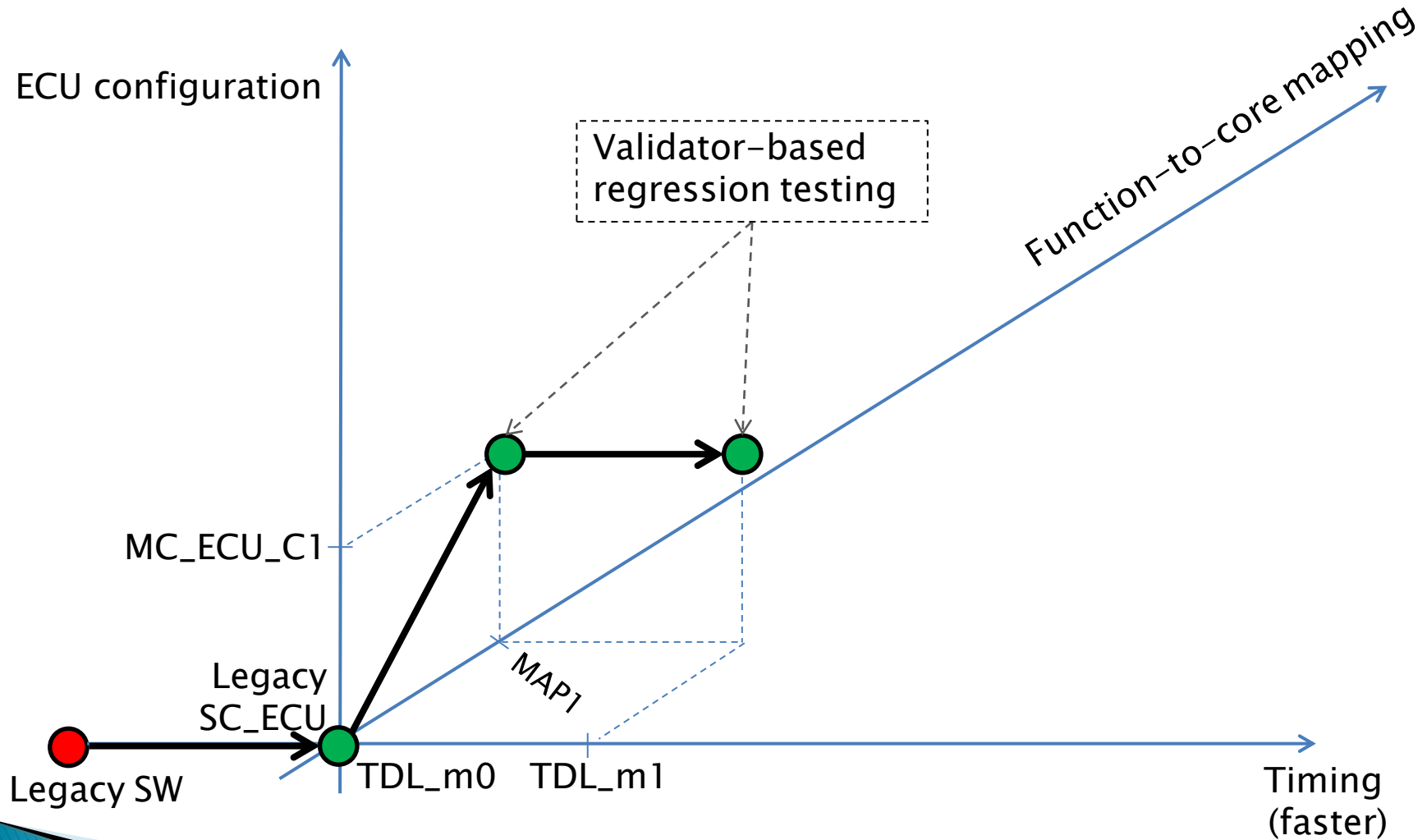
A LET-based porting process (1)



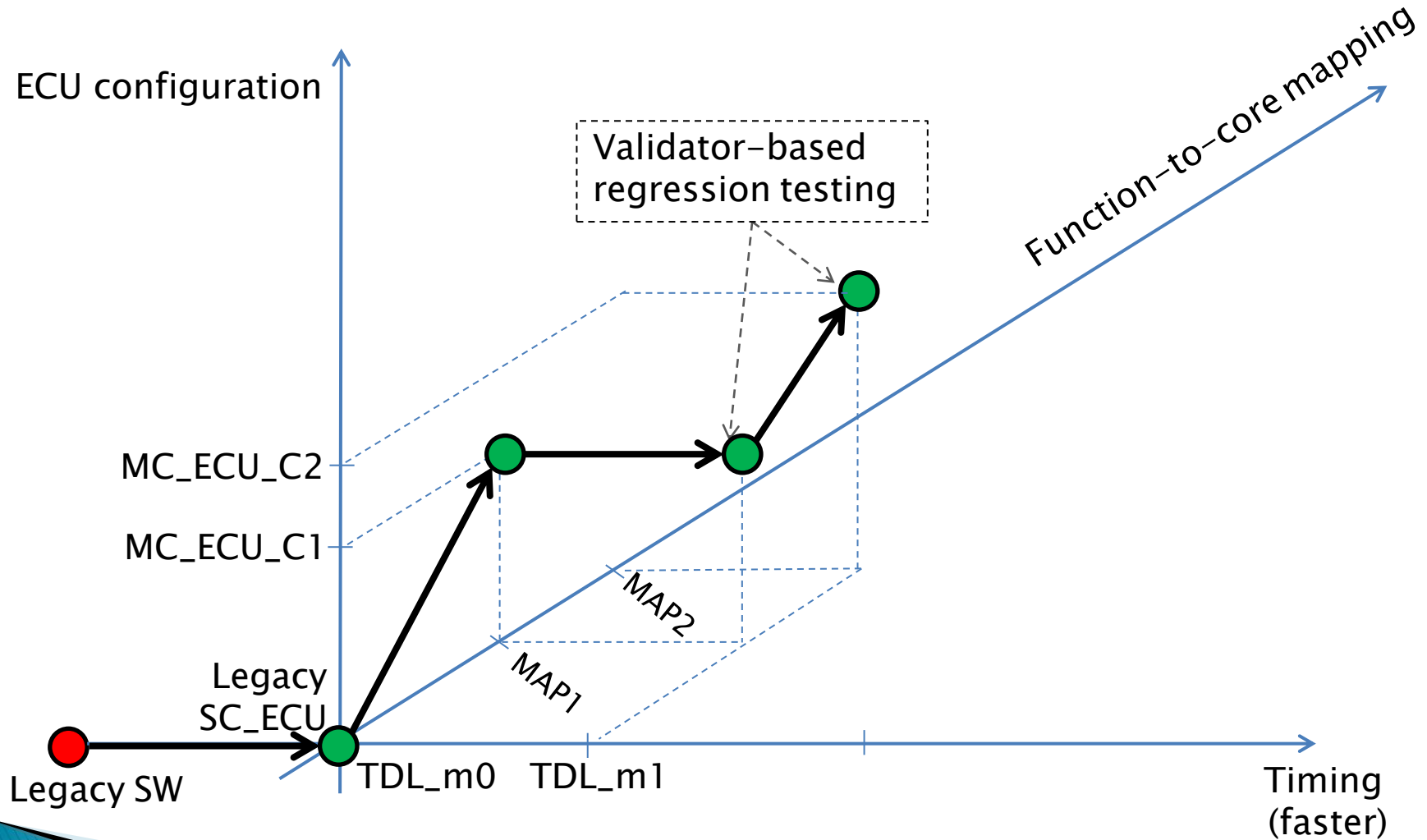
A LET-based porting process (2)



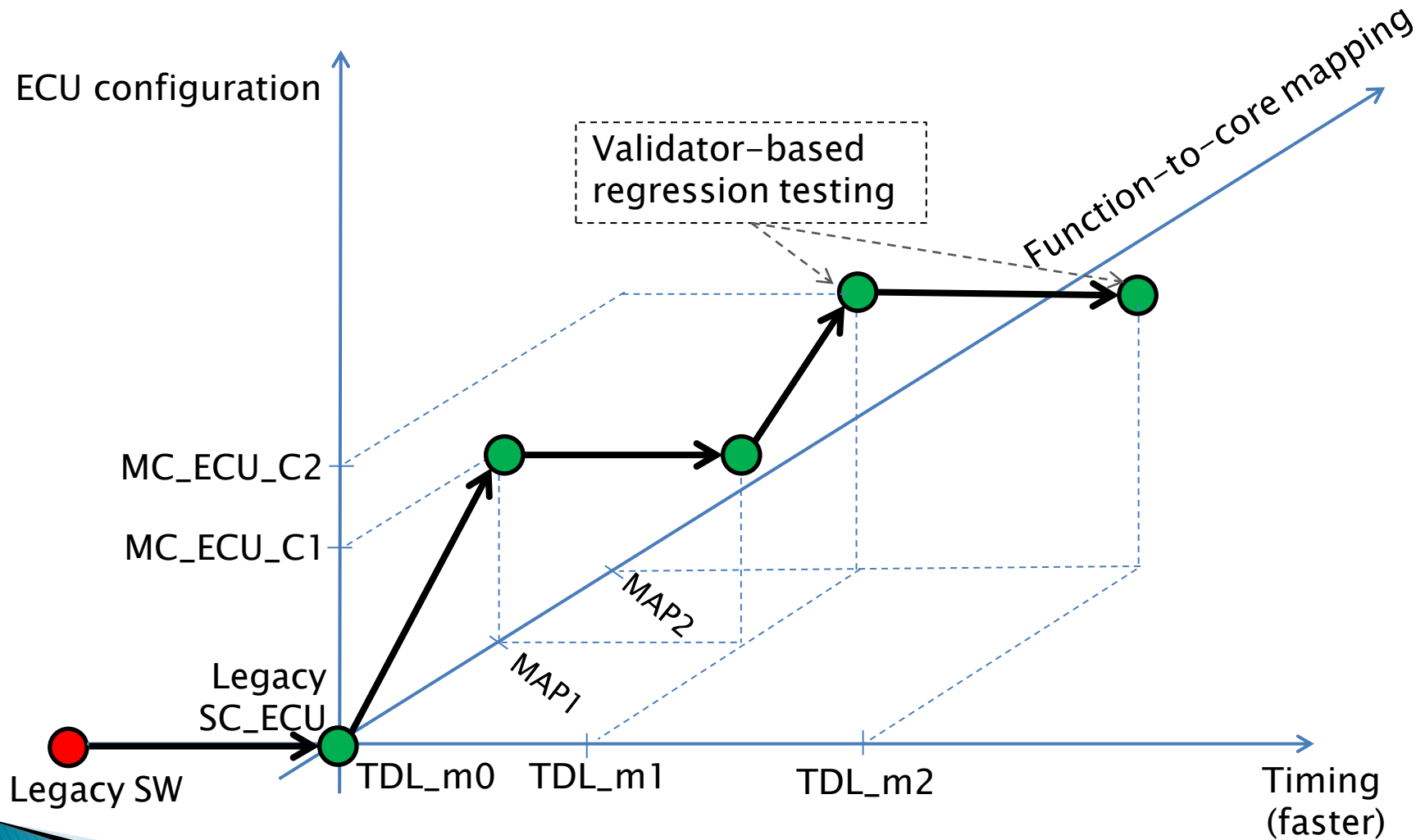
A LET-based porting process (3)



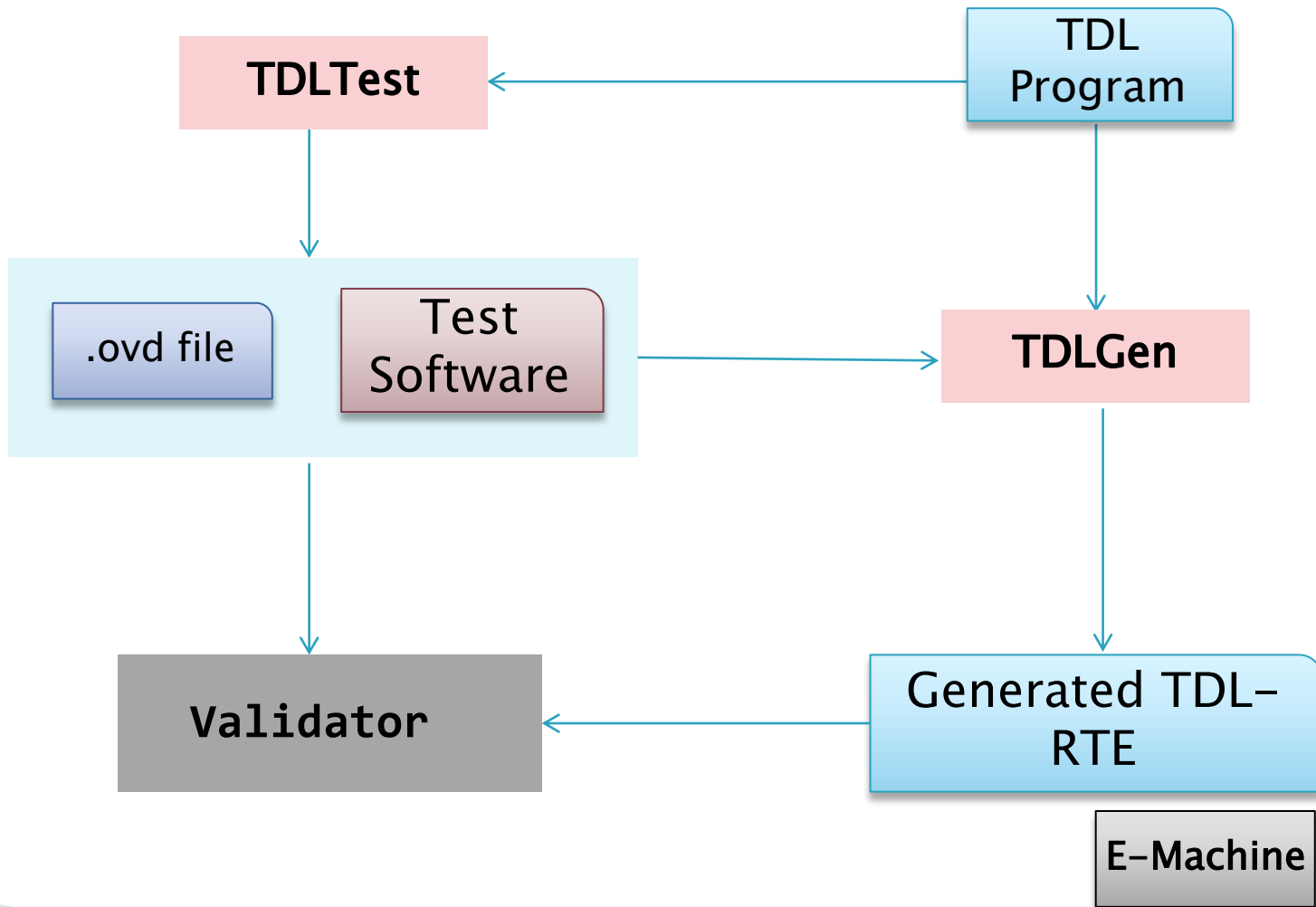
A LET-based porting process (4)



A LET-based porting process (5)



Testing: the TDLTest tool



Application: Engine Control SW on single- and multi-core

- ▶ Migration of single-core SW to TDL (2014-2015)
 - Migration tools: TDLGen and TDLMod
 - Verification: Validator and HIL simulations
 - Low additional computational costs
- ▶ Concurrency analysis: Auxilia (2016-2017)
 - Analysis time: under an hour

Application: Powertrain Control SW on Autosar and multi-core

- ▶ TDLGen (2015-2016)
 - Applied at OEM and Tier 1 (ECU supplier)
- ▶ Auxilia (2016)
 - Applied at ECU supplier
- ▶ TDLTest & Validator (2017)
 - Applied at TDLGen supplier

Thank you!

