Continental
The Future in Motion

MIL/SIL/PIL Approach
A new paradigm in Model Based Development

Narayanamurthy Srinivas, Narendrakumar Panditi

Stefan Schmidt, Ralf Garrelfs

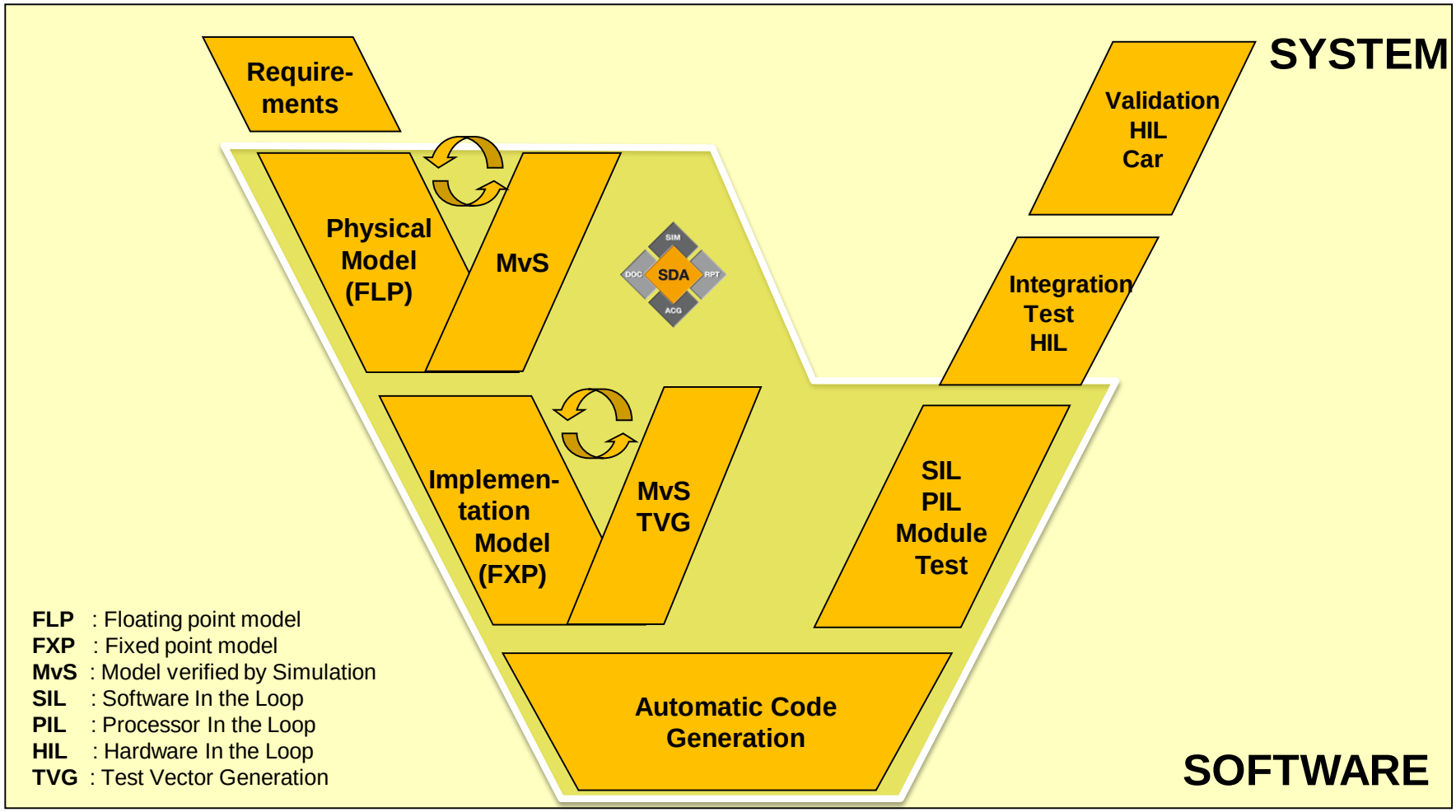
Agenda

1. Motivation - Model Based Development (MBD)
2. Model verified by Simulation (MvS)
3. Case study on MIL/SIL/PIL
4. MIL/SIL/PIL Simulation results in SDA
5. Comparison of MIL/SIL/PIL results
6. Conclusion

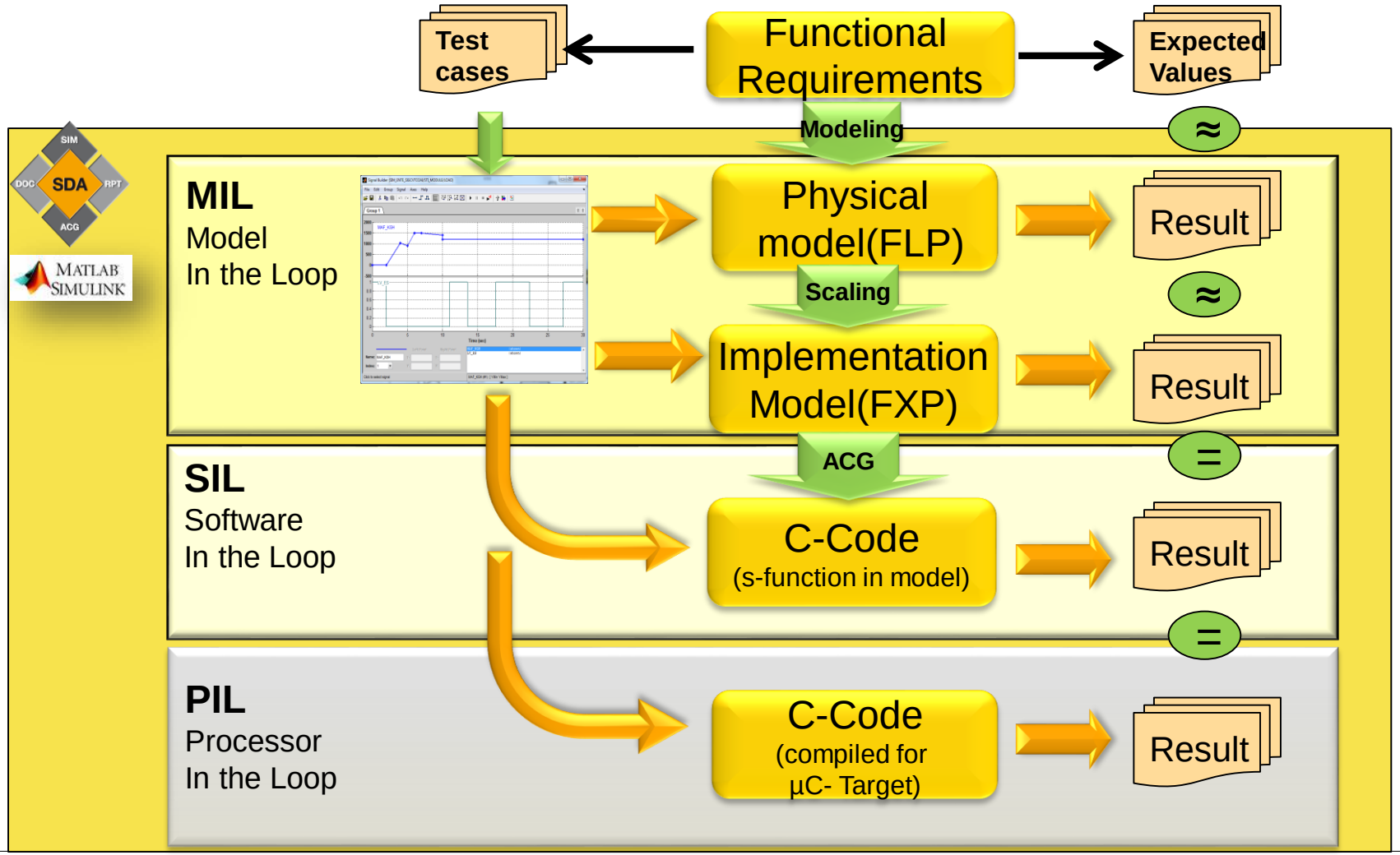
Motivation - Model Based Development (MBD)

	Manual	Model In the Loop (MIL)	Model In the Loop (MIL) Software In the Loop (SIL) Processor In the Loop (PIL)
Specification Design	Manual in the form of document	Model design using MBD MIL: Model verification	Model design using MBD MIL: Model verification
Coding	Manual coding	Auto code generation (ACG)	Auto code generation (ACG)
Code Verification	Manual prepared test cases to perform Unit Testing	Tool generated test cases to perform unit testing	Reuse MIL test cases SIL : Software verification PIL : Software verification on Target processor or equivalent instruction set simulator

Model Based Development: V- Cycle



Model Verified by Simulation (MvS)

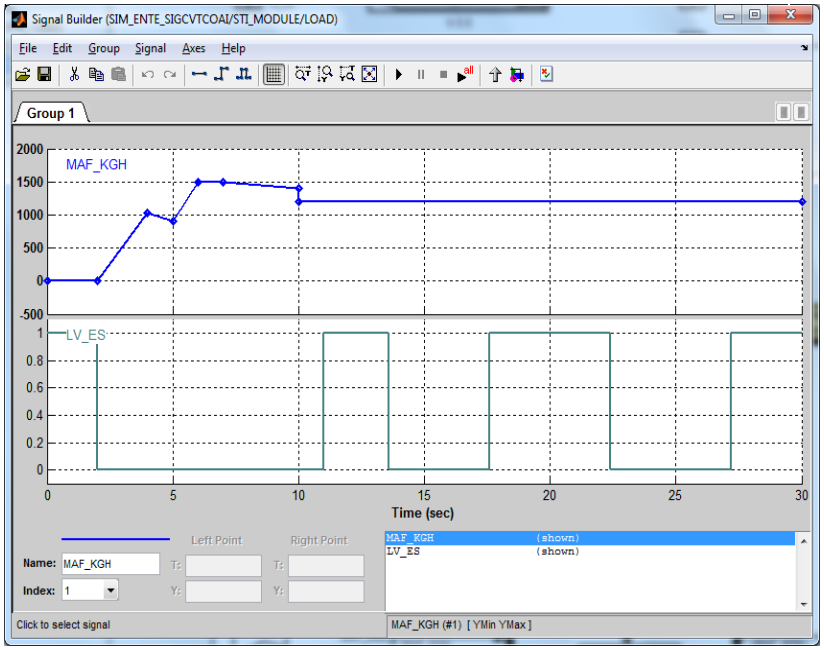


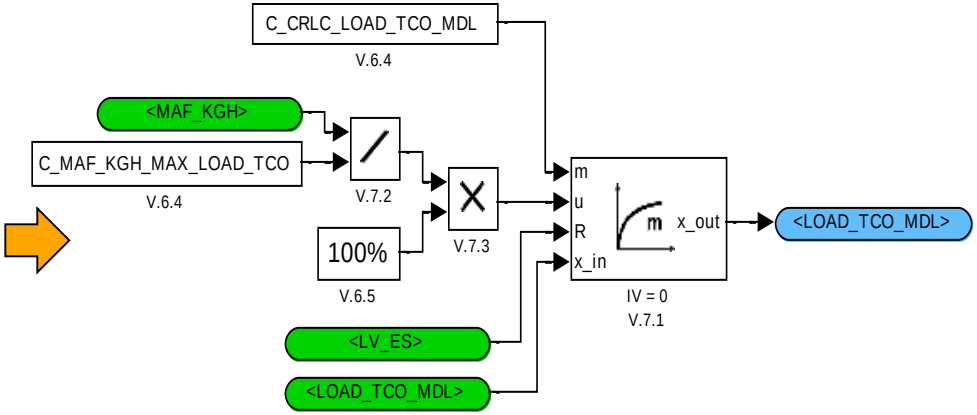
Definition – MIL/SIL/PIL

MIL Model In the Loop	SIL Software In the Loop	PIL Processor In the Loop
Refers to the kind of testing done to verify the accuracy / acceptability of a plant model or a control system. MIL testing means that the model and its environment are simulated in the modeling framework without any physical hardware components.	Refers to the kind of testing done to validate the behavior of the auto generated code used in the controller. The embedded software is tested within a simulated environment model but without any hardware.	Refers to the kind of testing done to validate the referenced model by generating production code using the model reference target. The code is cross-compiled for and executed on a target processor or an equivalent instruction set simulator.
MIL allows testing at early stages of the development cycle.	SIL also allows to verify the code coverage.	PIL level of testing can reveal faults that are caused by the target compiler or by the processor architecture.

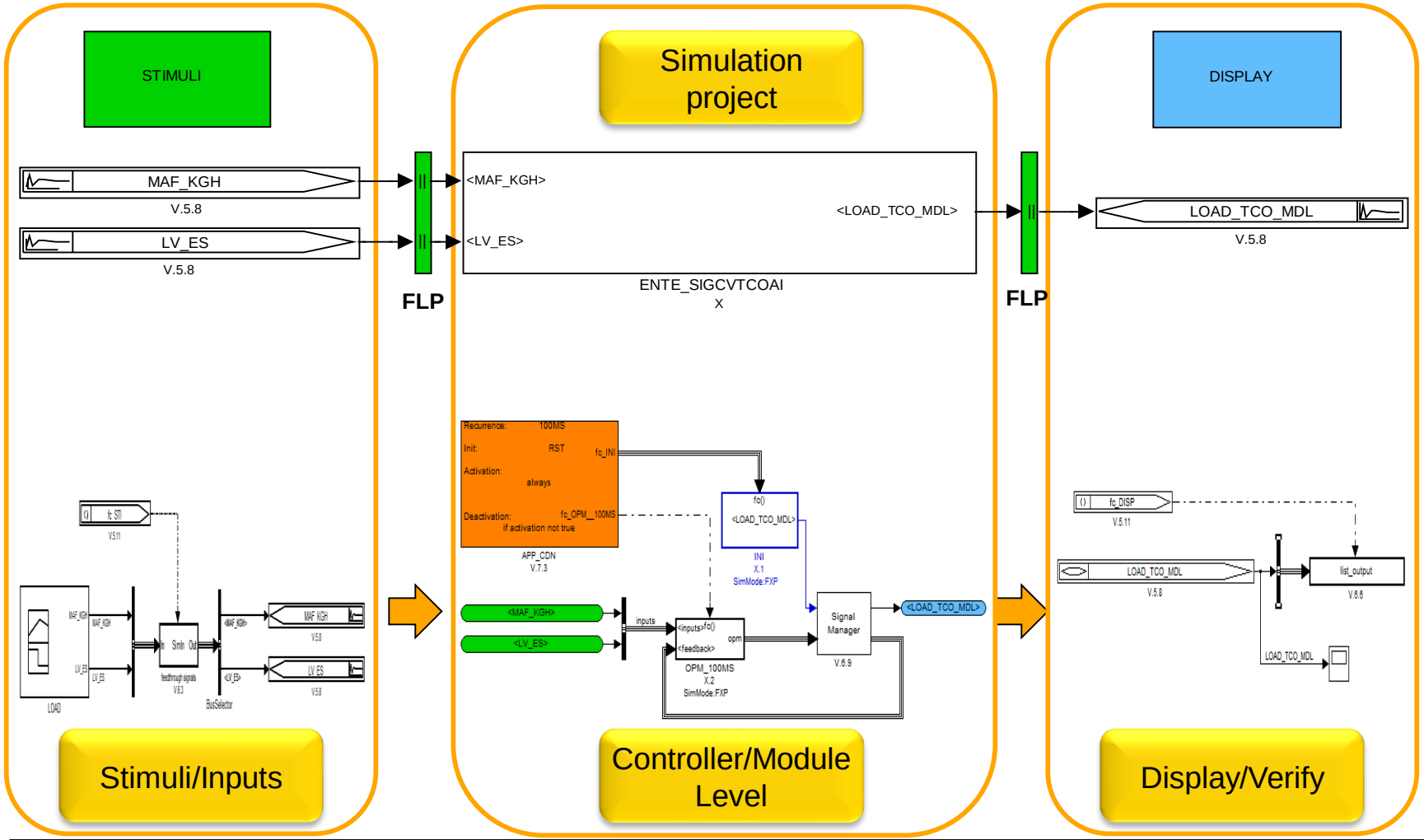
Case study on Engine Temperature function

1 Test suite for calculation of load information for coolant temperature model

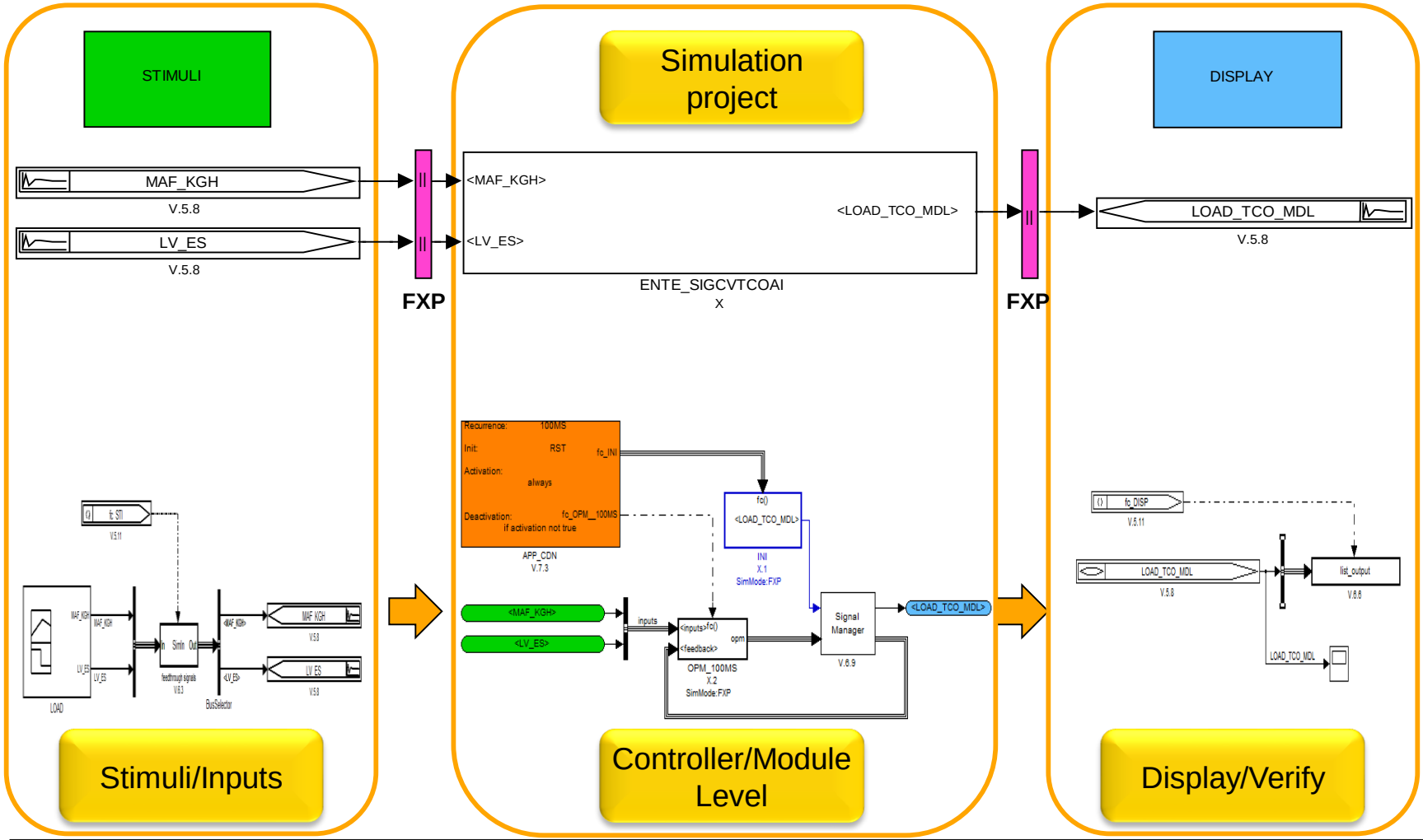




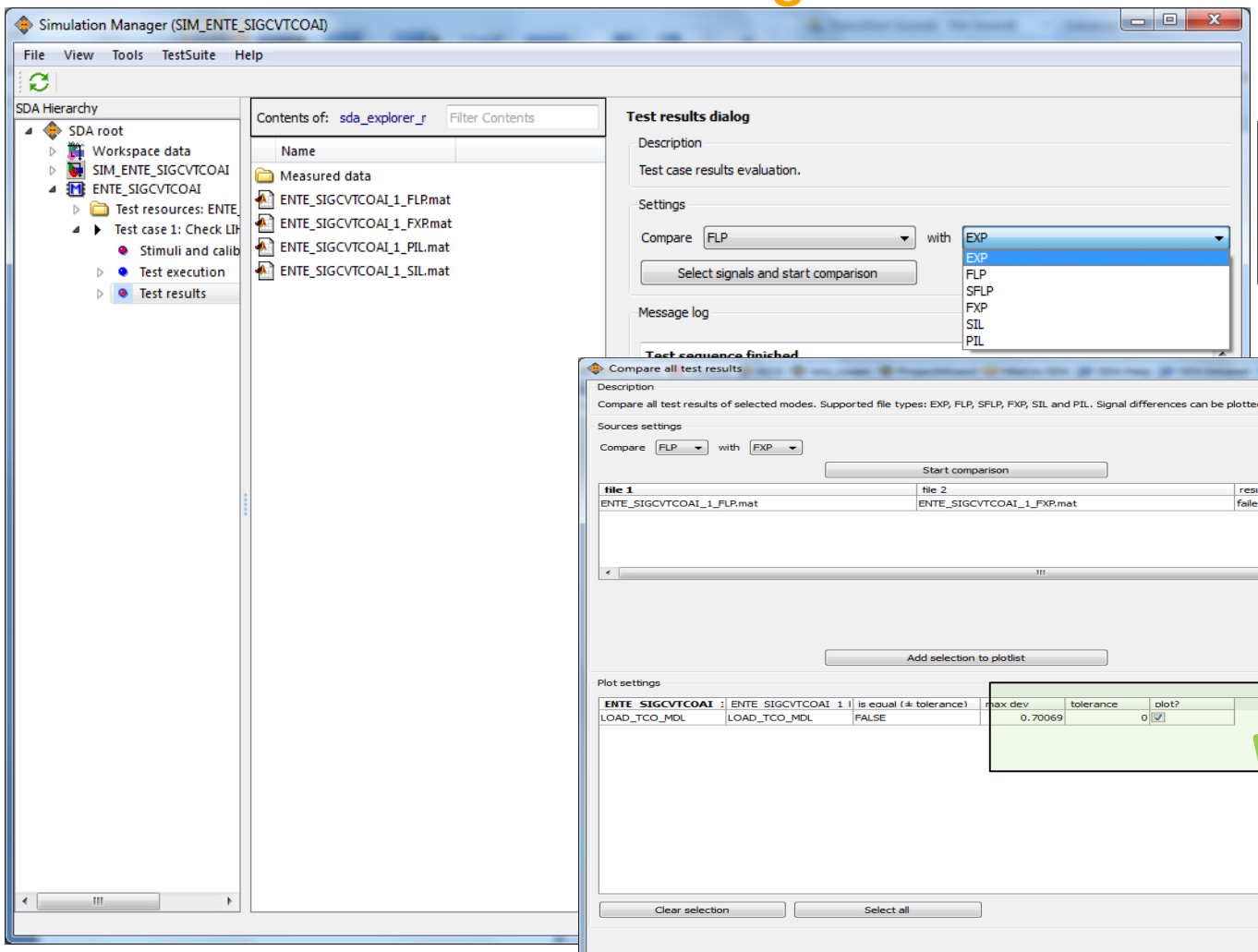
Model In the Loop (MIL): Floating point model



Model In the Loop (MIL): Fixed point model

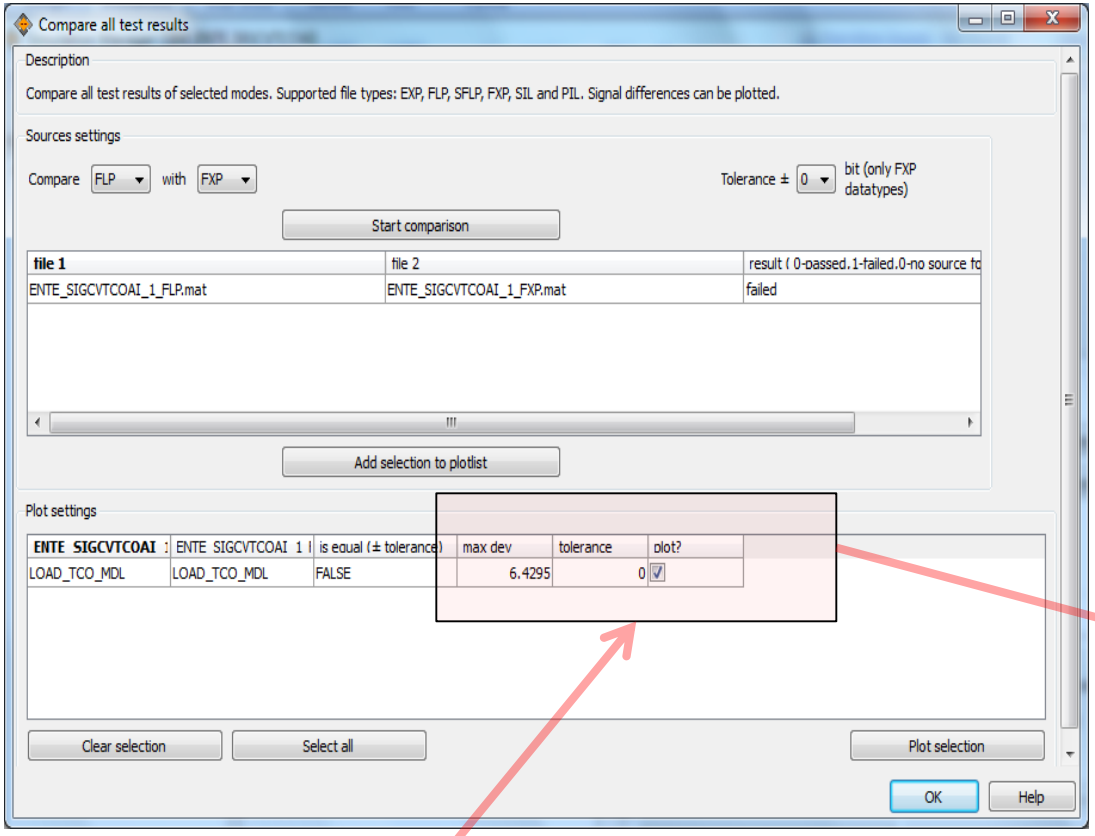


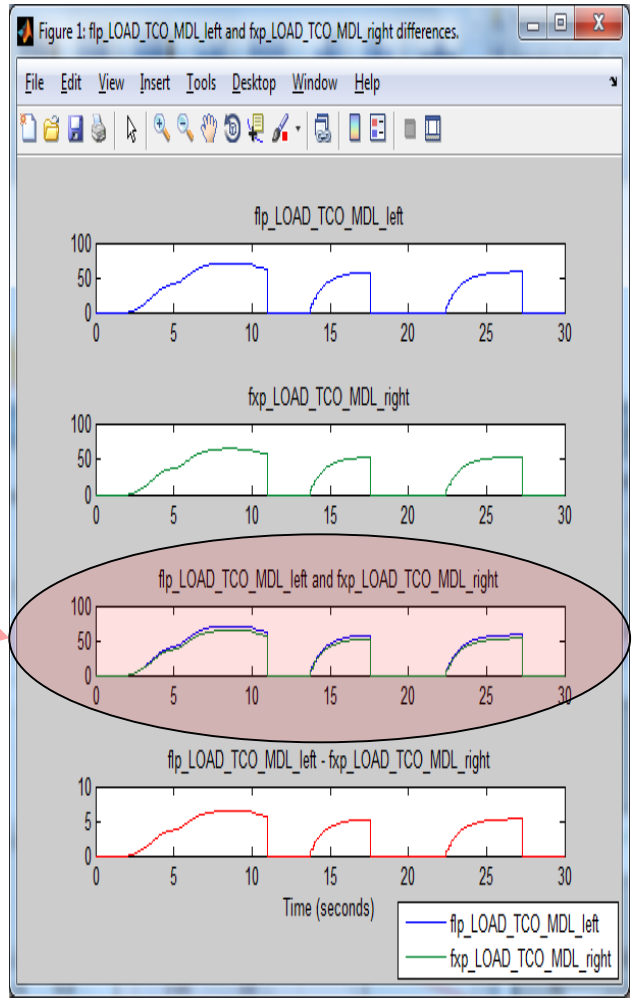
MvS: SDA Simulation Manager



Deviations can be detected and can be solved at early stages

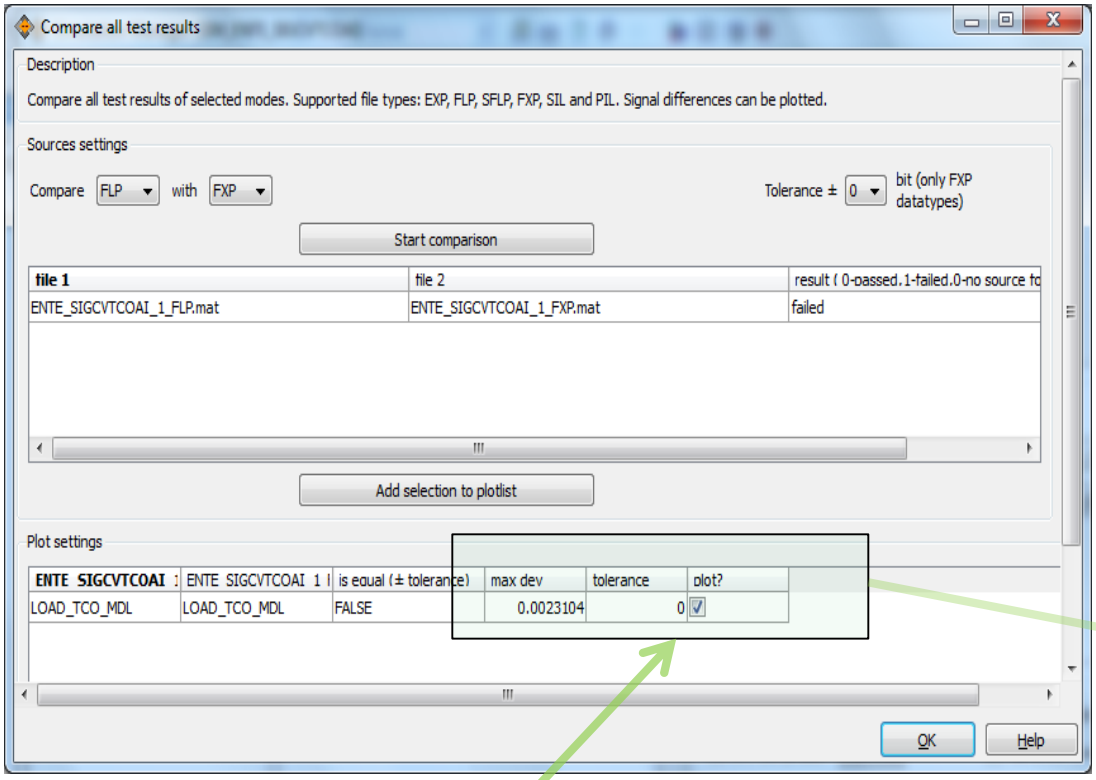
Comparison results MIL - FLP/FXP (Error)

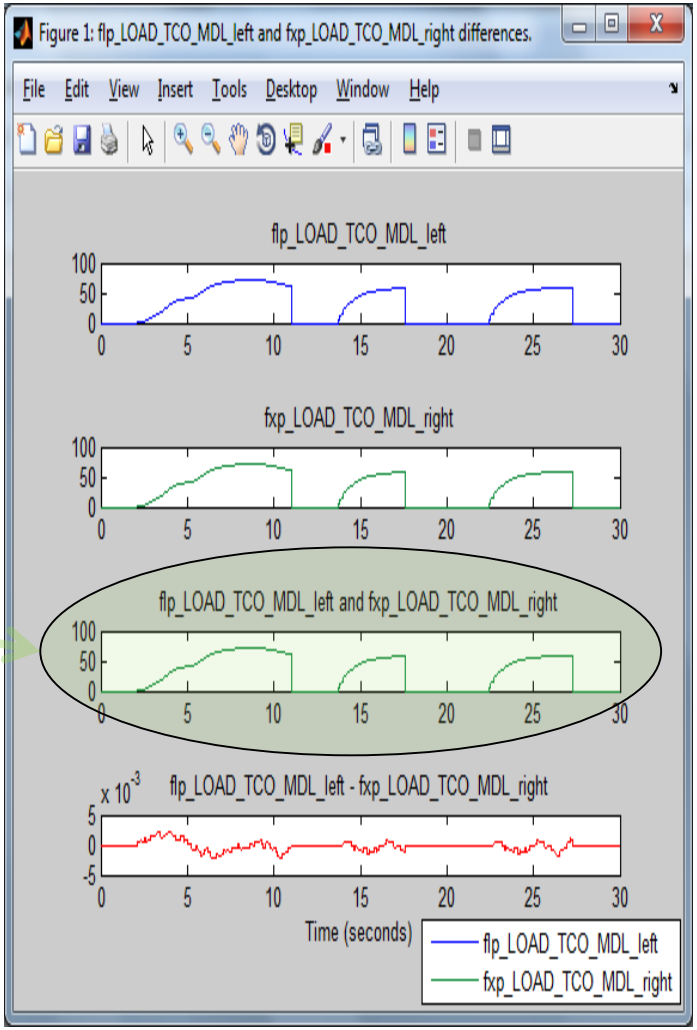




Deviations due to wrong scaling

Comparison results - FLP/FXP (Corrected Case)





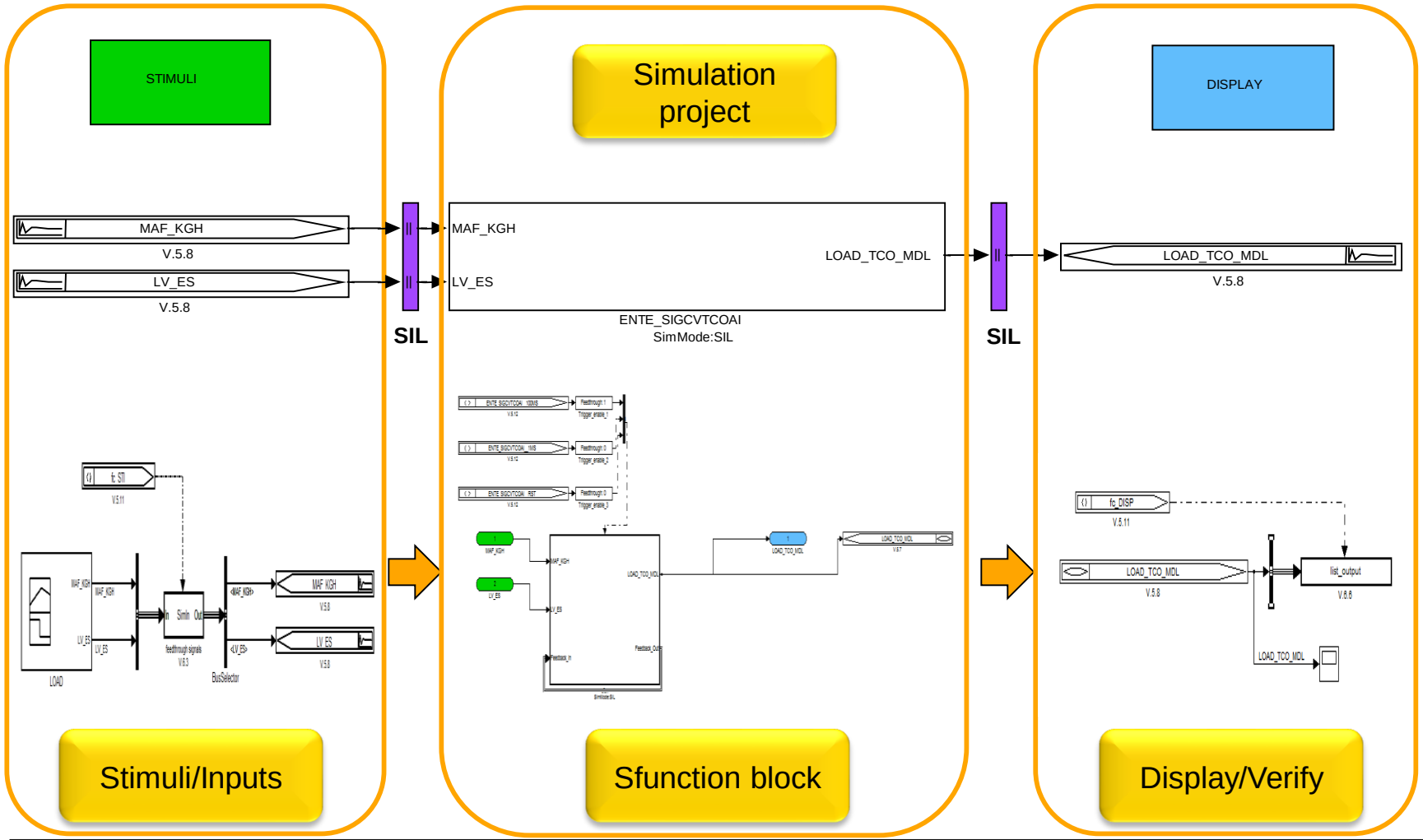
Deviations are with in the resolution

Present Situation after MIL

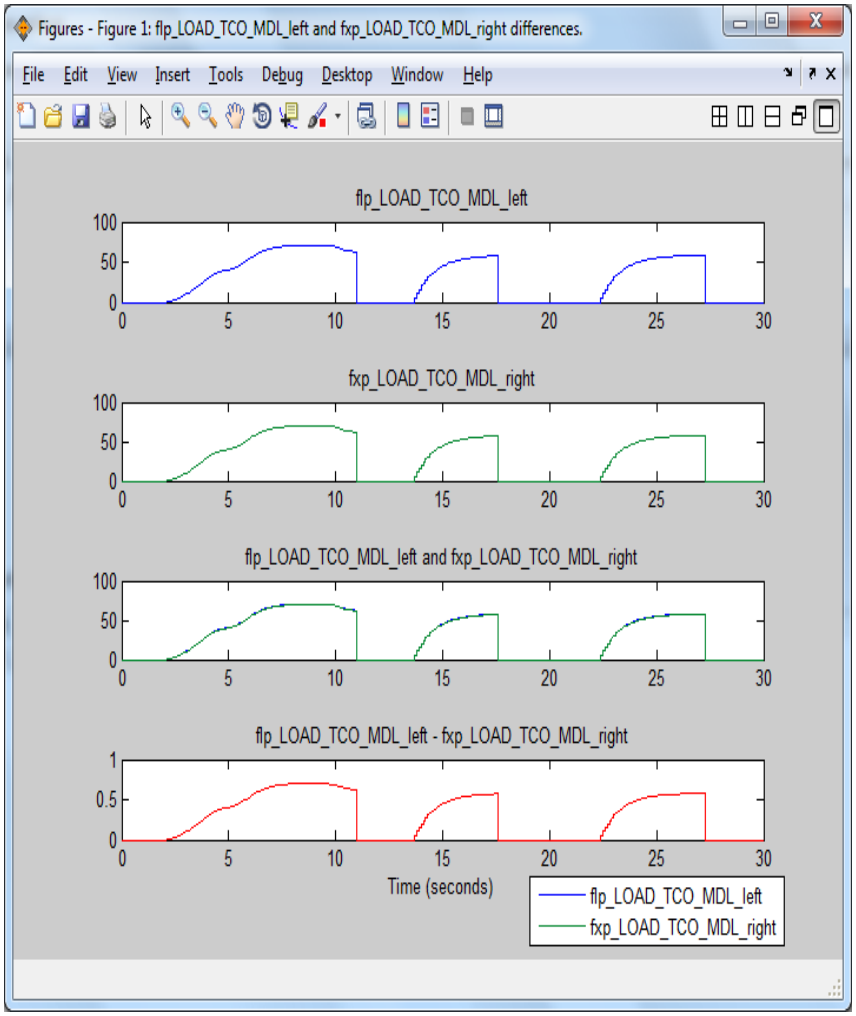
1. Random test cases are generated to test production code.
2. Execute generated test cases in the project environment.
3. More effort is required to prepare test cases to verify production code.
4. Completely different test cases are used to verify model and generated code.

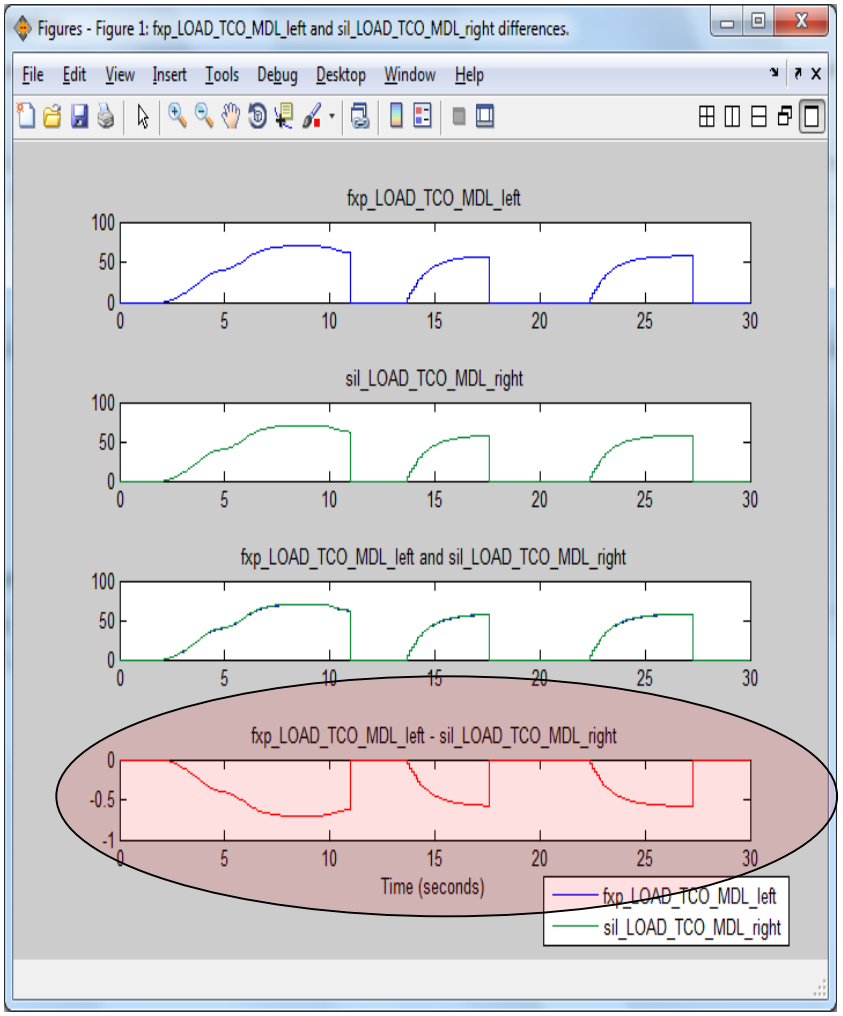
1 Wouldn’t it be nice to reuse the MIL test cases for test of the Automatically Generated Code ?

Software in the Loop: SIL

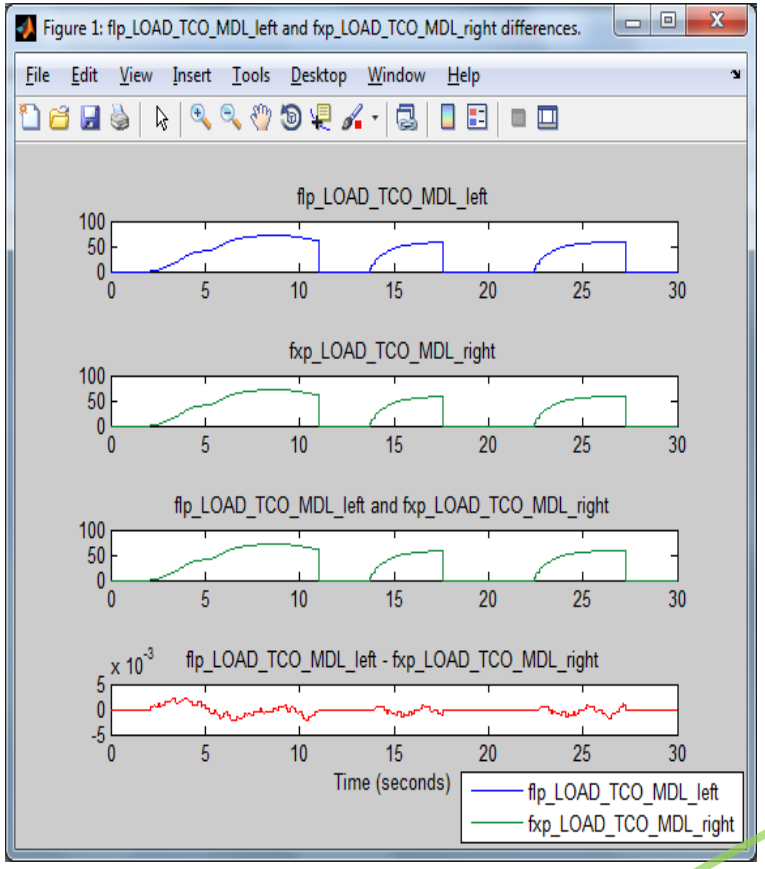


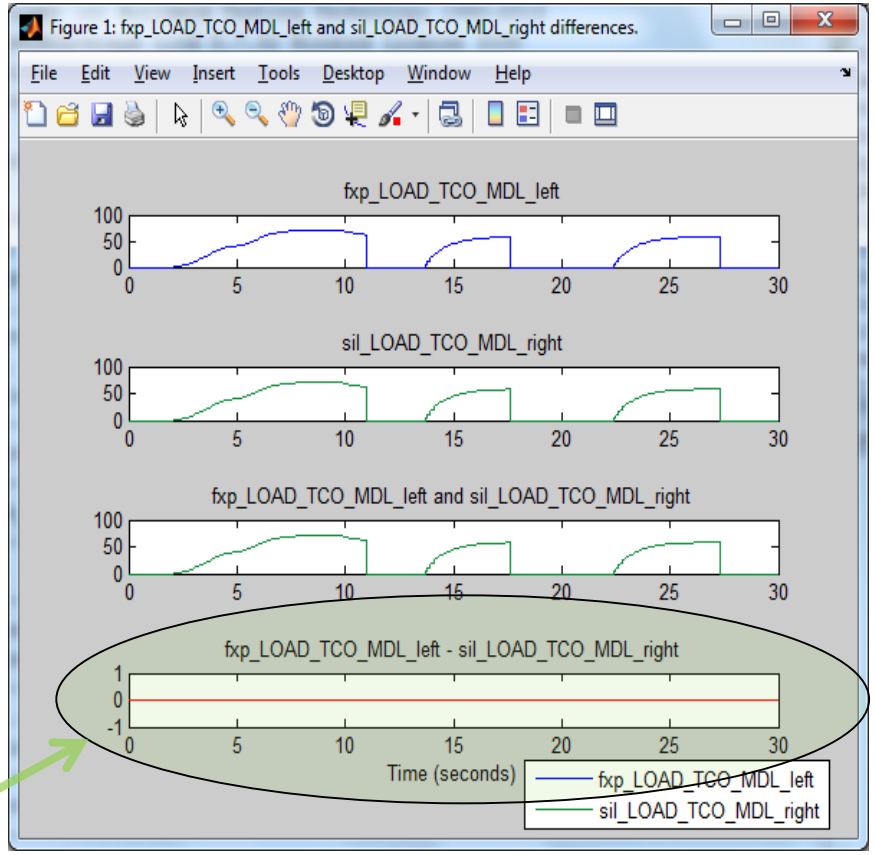
Comparison results - MIL/SIL (wrong case)





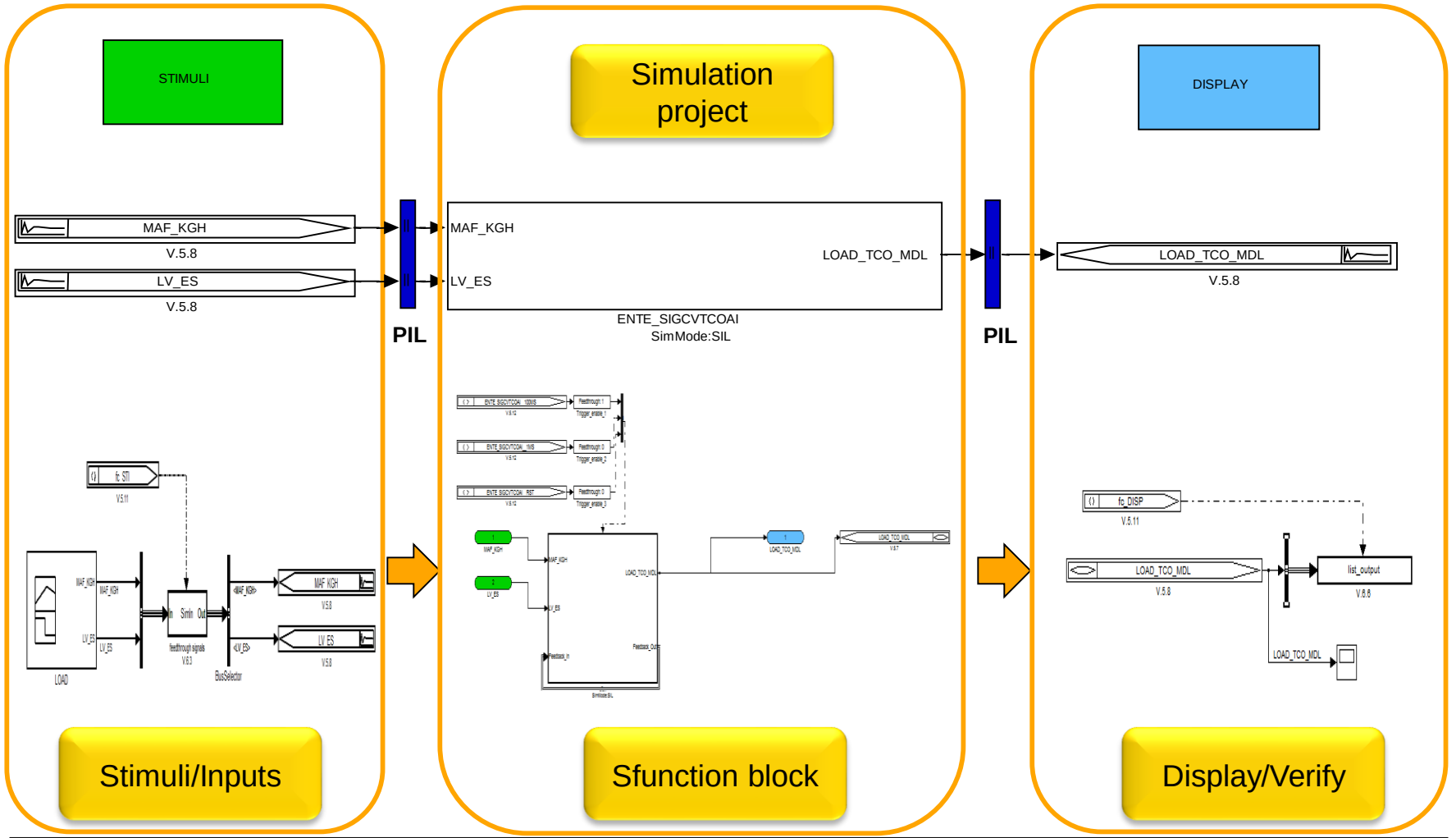
Comparison results - MIL/SIL (correct case)



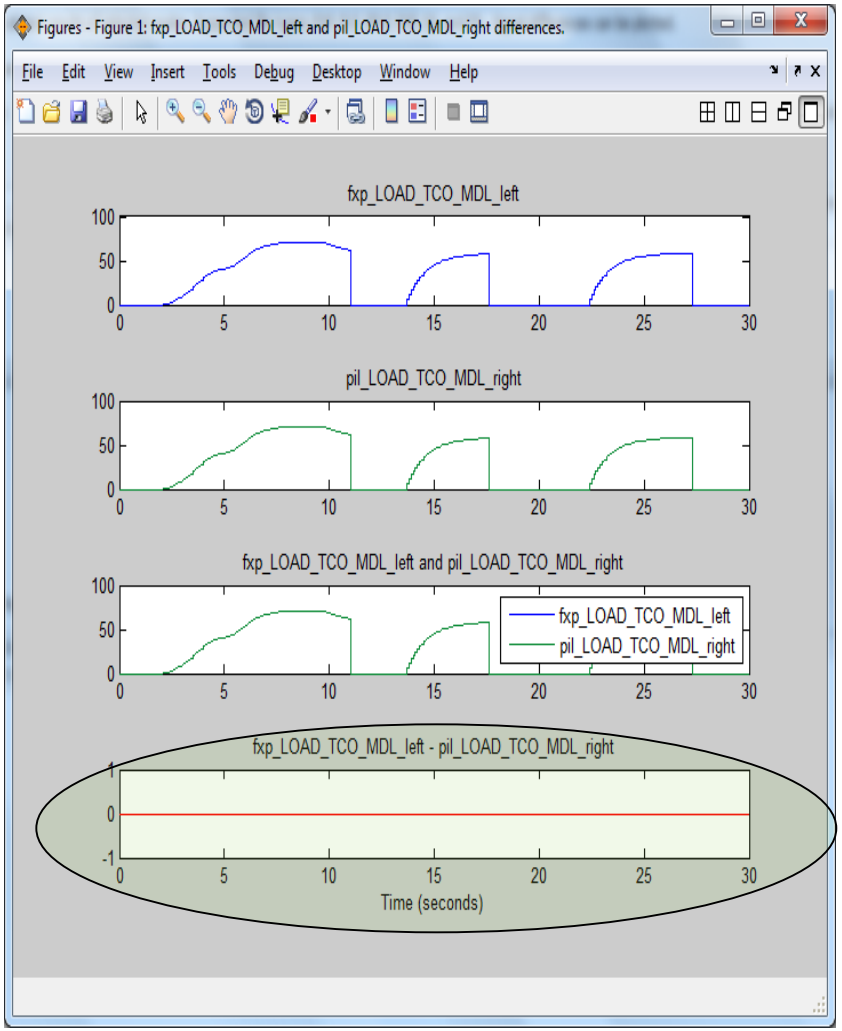


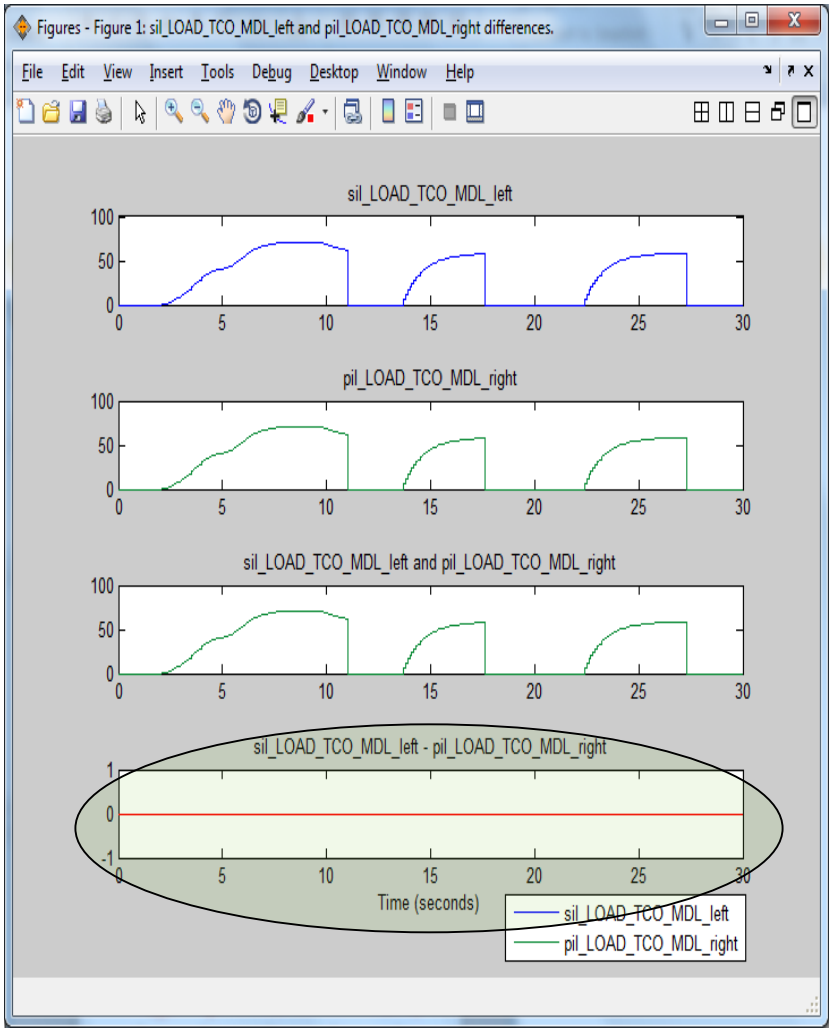
No Deviation

Processor In the Loop: PIL



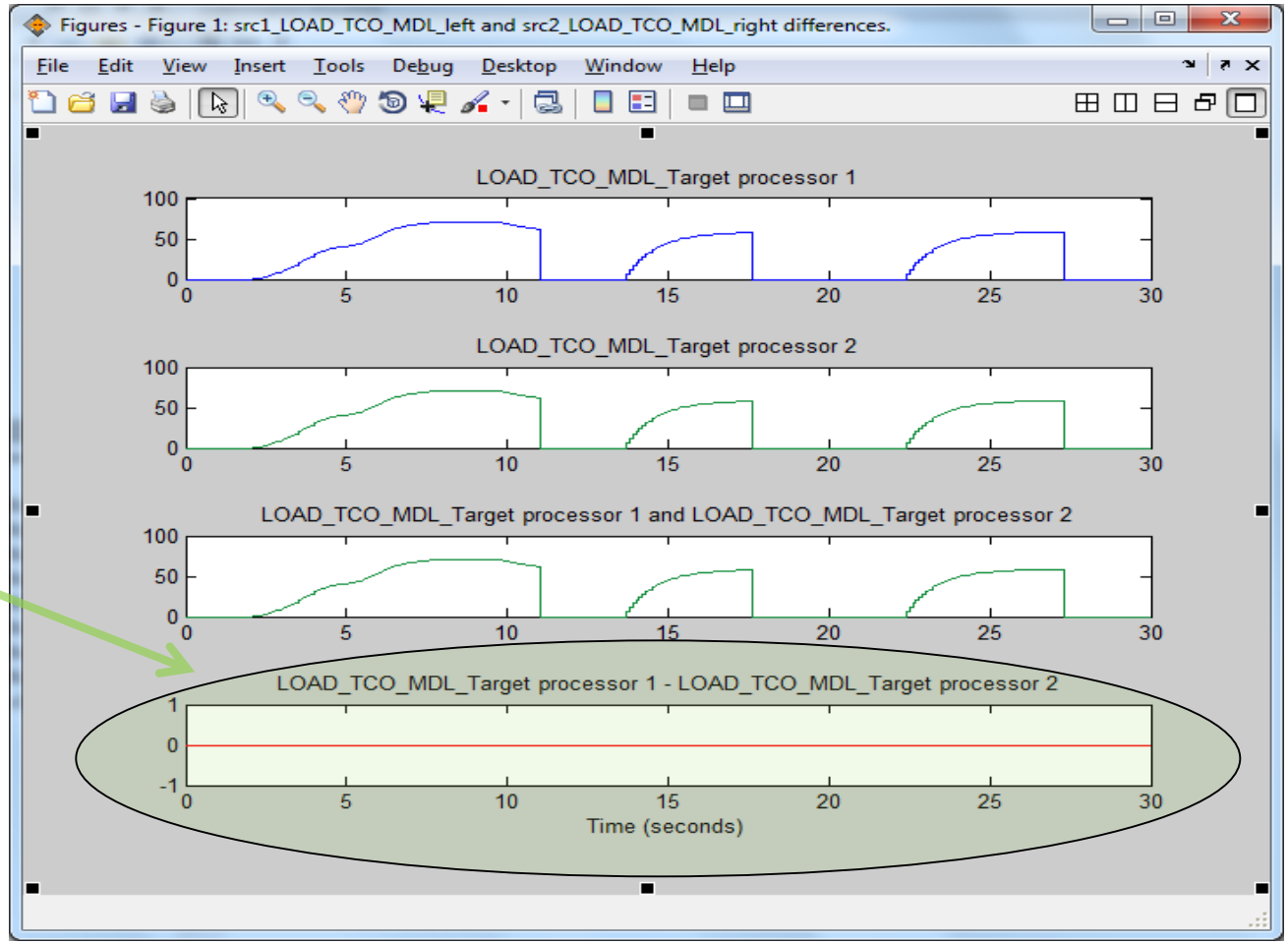
Comparison results - SIL/PIL





PIL results for different target processors - Reusability

No Deviations



Conclusion

1. Necessary test effort can be essentially minimized across simulations.
2. Tests suites are portable and reusable.
3. Cost-efficient consistent testing for all phases of the development:
   One test suite for all development phases (MIL, SIL, PIL).
4. Early malfunction detection.
5. Eases the updating of test suites for changed requirements.
6. Shorter development process resulting in significant time-to-market advantage.

Thank you
for your attention!