Virtual Test Driving in the Development Process – New Methods and Tools for Current Challenges
Episode 1: Hardware-in-the-Loop
Current and Future Challenges for Function Verification with HIL

- Highly networked functions
- Platform ECUs/„Super ECUs“
- Sensor data fusion
- Neural networks
  Artificial intelligence
- Automated Driving

© IPG Automotive GmbH, 2018
Existing Implementation with CarMaker

Databases

- .dbc
- .xml
- .arxml

CANiogen cmdline tool

- .xml
- .arxml

Fibex2flex cmdline tool

C code / Infofile editing

APP_VER = "Car_Generic
<insert.your.version.no>"
APP_NAME = CarMaker.$(ARCH)$(EXE_EXT)
OPT_CFLAGS = -g -O1
LD_LIBS = $(CAR_LIB) \n$(CARMAKER_LIB)
$(DRIVER_LIB) $(ROAD_LIB) $(TIRE_LIB)
$(TAME_LIB)
OBJJS = CM_Main.o CM_Vehicle.o
User.o
OBJJS_xeno = IO.o $(OBJJS_CANIOGEN)
$(OBJJS_CANIOGEN_USER)
OBJJS_linux =
OBJJS_linux64 =
OBJJS_win32 =
OBJJS_win64 =

# Prepend local include/library directory
to include path:
# PREINC_CFLAGS +=
-# ../include -

C code

- (IO_CAN.c,...)

Infofiles

- (FlexRay Parameters, ...)

void IO_CAN_Cam_User_Out(
    const Unsigned int CycleNo)
{
    // Modify/Delete following lines, assign values to CAN signal variables
    /*
    ECU CAN Out */
    /* CAN Msg 0x550 (CAMcommon) */
    IO_CAN_Cam_Timings.SendPeriod = 1;
    /*Periodenzeit in Millisekunden*/
    IO_CAN_Cam_Timings.SendDistrib = 0;
    /*Zeitversatz der Botschaft*/
    IO_CAN_Cam_EPM.MEcommon_DLC = IO_CAN_Cam_MEcommon_DLC; /*Data length coding gibt Länge der Botschaft an*/
    IO_CAN_Cam.Blinker_Left
    = VehicleControl.Lights.IndL;
    IO_CAN_Cam.Blinker_Left_State
    = IO_CAN_Cam_SigStates[IO_CAN_Cam_SState_
Bus Configurator
For rest bus simulation

Databases
- .dbc
- .xml
- .arxml

Infofiles
- FlexRayParameters
- RBSParameters
- FlexCardParameters

Infofiles
- CANParameters
- RBSParameters

LIVE DEMO
BUS CONFIGURATOR
1. Choose your database

2. Configure your restbus configuration

3. Map and refine your signals

4. Save and run your test
Features at a glance

✓ Full FlexRay support (FIBEX, AUTOSAR)
✓ Full CAN support (FIBEX, AUTOSAR, Vector DBC)
✓ Automatic detection of database type
✓ Automatic generation of frames, PDUs and signals
✓ GUI-based mapping of signals to Data Dictionary, Constants, RollingCount, E2E_P2, CRC
✓ Additional functions for usability (e.g. filter, hooks, trigger…)
✓ Infofile approach enables simulation start on-the-fly
Automotive Sensors

- Radar
- Lidar
- Front View Kamera
- Surround View Kamera
- Rear View Kamera
- Ultrasonic
- GPS
Hardware-in-the-Loop Environment

Sensors

ECU

Aero

Chassis

z(.)

m(.)

k(.)

d(.)

q(.)

y(.)

p(.)

x(.)

a(.)

h(.)

j(.)

i(.)

f(.)

b(.)

c(.)

...
Object List Based Injection

Bypass (object lists)

ECU

Restbus simulation

Sensors

Device under test
Radar Sensor
Physical Sensor Integration

- Device under test
- Sensor stimuli
- Sensors
- ECU
- Restbus simulation
- Xpack4 RT-System
Example: Ultrasonic Sensor Box

Device under test

ECU

Restbus simulation

HIL TEST SYSTEM

CarMaker

Xpack4 RT-System
Example: Ultrasonic Sensor Box

Device under test

HIL TEST SYSTEM

Xpack4 RT-System

CarMaker

Restbus simulation

ECU
Emulation of Sensor Signals

Device under test

Sensor emulation

Sensors

ECU

Restbus simulation

CarMaker

Xpack4 RT-System
Example: Video Interface Box

1. Sending the virtual scenario
2. Sending the data to device under test
3. Sending back information to CarMaker
4. System reaction in CarMaker

Closed Loop Testing

Xpack4 system with CarMaker

Device under test
### ADAS Sensor Interfaces

#### Overview

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Physical (sensor-in-the-loop)</th>
<th>Emulation</th>
<th>Object list based approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>Screen capturing = Monitor-HIL</td>
<td>Camera RSI</td>
<td>Object Sensor / Line Sensor / Traffic Sign Sensor</td>
</tr>
<tr>
<td>Ultrasonic</td>
<td>Ultrasonic sensor box</td>
<td>Ultrasonic RSI</td>
<td>Object Sensor</td>
</tr>
<tr>
<td>Lidar</td>
<td>Lidar target simulator with partners of IPG Automotive</td>
<td>Free Space Sensor / (Lidar RSI – coming soon)</td>
<td>Object Sensor</td>
</tr>
<tr>
<td>Radar</td>
<td>Radar target simulator with partners of IPG Automotive</td>
<td>Radar RSI</td>
<td>Object Sensor / Radar HiFi Sensor</td>
</tr>
</tbody>
</table>

Every sensor type matches at least one HIL solution

Solution depends on requirements and available information
LIVE DEMO

HARDWARE-IN-THE-LOOP
Drive PX2 HIL Demo System

Device under Test

Restbus simulation / Radar data

Video Interface Box

NVIDIA Drive PX2

GMSL

HDMI

Radar

RSI

CarMaker

Xpack4 RT-PC
Drive PX2 HIL Demo System

Device under Test

NVIDIA Drive PX2

Restbus simulation / Radar data

Video Interface Box

HIL TEST SYSTEM

Radar RSI

CarMaker Xpack4 RT-PC

HDMI

GMSL

Device under Test

NVIDIA Drive PX2

Restbus simulation / Radar data

Video Interface Box

HIL TEST SYSTEM

Radar RSI

CarMaker Xpack4 RT-PC

HDMI

GMSL
CarMaker Runs on Different Hardware Platforms

Function under test integrated in CarMaker HIL software as middleware running on Hardware platform (i.e. Xpack4, National Instruments PXI, dSPACE Scalexio & DS1006, ETAS Labcar)
Episode 2: Virtual ECUs
- FMUs allow a seamless and easy integration of own or third-party models into CarMaker
- Automatic conversion of physical units between connected signals
  - GUI proposes factor and offset for conversion automatically based on given units
- Tunable parameters allow a runtime influencing of the FMU
Silver Virtual ECUs

Front-loading validation and testing during automotive software development
What is a virtual ECU? – Runs on a PC „like the ECU in the car“

vECU and ECU share the same code
- C sources or
- Target HEX code

vECU and ECU have same interfaces
- Calibration – CANape, INCA
- Configuration files – PAR, DCM
- Networking – CAN, LIN, FlexRay
Software Development Process

Model-based development automotive software

Feedback for developers

ECU test and validation

Build for target CPU

Flash

HIL

ECU

C code libs

Code gen. and hand coding

 mdl

.C code libs

Build for target CPU

.hex

Flash

ECU

Feedback for developers

ECU test and validation

Build for target CPU

Flash

HIL

ECU

.C code libs

Code gen. and hand coding

.mdl
Software Development Process

Model-based development automotive software

Code gen. and hand coding

C code libs

Build for target CPU

.hex

Flash

Few and expensive: BOTTLENECK!

Feedback for developers

Days or weeks: LATE!

ECU test and validation

Build for target CPU

Few and expensive: BOTTLENECK!

ECU

HIL
Silver Virtual ECUs – Software-in-the-Loop

Benefit of SIL: same as MIL plus
- More SW-in-the-loop
- Behavior closer to real ECU
- Just 1 process for code generation

Minutes or days!

ECU test and validation

Feedback for developers

Build

C code libs

Build for target CPU

Flash

.vmdl

Code gen. and hand coding

Libs

vECU

ECU test and validation

ECU
Silver Virtual ECUs – Chip Simulation

Benefit of vPIL: same as SIL plus
+ No model or C code required

Minutes or days!

ECU test and validation

Feedback for developers

Build

vECU

C code libs

Build for target CPU

Code gen. and hand coding

Libs

vPIL

.ml
Build a Silver Virtual ECU

Configuration file for Adaptive Cruise Control

```plaintext
01 # set up compiler and files
02 visualStudioVersion("100") # use Visual Studio 10.0 compiler
03 inc_dir("$(PROJECT)\src") # location of C header files
04 src_files("$(PROJECT)\src\.+\..c") # C source files
05
06 # tasks of the virtual ACC
07 task_initial(task_init)
08 task_periodic(task_10ms, 0.01)
09
10 # interface of the virtual ACC
11 input(ACCIsActive)
12 input(ACC_DesiredSpd)
13 input(DM_Brake)
14 input(DM_Gas)
15 output(VC_Brake)
16 output(VC_Gas)
```

Virtual ACC

- ECU functions
- RTOS emulation
- 4 GB virtual memory
- A2L conversion
- XCP flash

.c, .h, .lib
.a2l, .dbc
.ldf
2. Import Silver FMU into CarMaker 7.0

Import is straightforward
- Just edit the configuration
- No need to rebuild CarMaker.exe
Virtual Test Driving: CarMaker & Silver Virtual ECU

Entire Silver setup acts as FMU
- One or many networked vECUs
- User interface
- Optional plant models

Validation and test
- Use TestWeaver or any other test automation solution available for CarMaker and/or Silver
Silver offers

- PC execution platform for virtual ECUs
- Proven build process for virtual ECUs
- Co-simulation with CarMaker 6.0.4 and later

Benefit

- Extremely short edit/run cycle for agile development
- Source-level debugger on PC
- Used to front-load validation and testing
Episode 3: Automated Driving
ROS in General

**Robot Operating System (ROS)**
- Open-source software framework
- Background in robotics
- Rising interest in the ADAS and AD community
- Autonomous software stack available
- Flexible architecture, very useful especially for big, complex systems
- Supports message-based inter-process communication even across hardware boundaries
- Hardware abstraction, device control and package management available
CarMaker / TruckMaker
ROS (Robot Operating System) support

- There are numerous trade-offs in creating a ROS AD architecture in combination with the simulation environment
- ROS node as plugin in CarMaker gives you a fast start
- CarMaker builds the perfect completion for a ROS-based virtual vehicle development
  - Simulates the missing environment
  - Provides the required flexibility for typical system and test designs
  - Straightforward data exchange between CarMaker and ROS
Possible Architecture for Automated Driving
Closed loop for automated driving function testing

Autonomous software stack
CarMaker environment

- Radar data processing
  - Radar data
  - /ROS node
  - Processed sensor data
- Camera data processing
  - Camera data
  - /ROS node
  - Processed sensor data
- Map data processing
  - Road data
  - /ROS node
  - Processed sensor data
- Sensor data fusion
  - /ROS node
  - Processed sensor data
  - Environment model
- Decision making
  - /ROS node
  - Desired maneuver
- Trajectory planning
  - /ROS node
  - Planned trajectory
- Control signal generation
  - /ROS node
  - Control signals

CarMaker environment

ROS node
Possible Architecture for Automated Driving
Closed loop for automated driving function testing

Autonomous software stack

- Radar data processing
- Camera data processing
- Map data processing
- Sensor data fusion
- Decision making
- Trajectory planning
- Control signal generation

CarMaker environment

/ROS node

- Radar data
- Camera data
- Road data

Processed sensor data

Environment model

Desired maneuver

Planned trajectory

Control signals
Demo Architecture for ACC Control
Closed loop for ADAS function testing

Autonomous software stack

ACC control algorithm
/ROS node

Object sensor data + vehicle data

Control signals

CarMaker environment

/ROS node

CarMaker
LIVE DEMO CarMaker & ROS
Proof of Concept
Episode 4: High-Performance Computing
Parallelization Analysis

**Single-domain**
- Evenly distributed atomic units
- Constant model complexity
- Medium serialization share

**Multi-domain**
- Uneven model split
- Purpose driven fidelity – variable model complexity
- High serialization share
The Challenge of Speeding Up the Simulation

According to Amdahl's law

\[ \eta_s = \frac{T}{ts + to + tp/np} \]

Number of cores

Parallel portion
- 50%
- 95%
Parallelization on TestRun Level
Parallelization on TestRun Level
The Solution
According to Amdahl's and Gustafson's law

\[ \eta_s = \frac{T}{ts + to + tp/np} \]
\[ \eta_s = (1 - tp) + n_p * tp \]

Acceleration factor

<table>
<thead>
<tr>
<th>Parallel portion</th>
<th>50%</th>
<th>95%</th>
<th>99.5%</th>
</tr>
</thead>
</table>

Number of cores

0   4   16   32   64   128   256   512   1024
HPC Classes

Number of cores

few

Test Manager HPC config

HPC with scheduler

HPC light

many
## HPC Light

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduler</td>
<td>Embedded in CarMaker Test Manager</td>
</tr>
<tr>
<td>HPC Scalability</td>
<td>1 node &lt; 100 cores</td>
</tr>
<tr>
<td>Dedicated Hardware</td>
<td>Single PC, multi-core</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Limitation of parallel processes</td>
</tr>
<tr>
<td>Resiliency</td>
<td>No special implementation</td>
</tr>
<tr>
<td>Health Checks</td>
<td>Process monitoring (tbi)</td>
</tr>
</tbody>
</table>
Re-Use Your Company Infrastructure
HPC with scheduler
## HPC with Scheduler

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scheduler</strong></td>
<td>With external scheduler (e.g. PBS professional)</td>
</tr>
<tr>
<td><strong>HPC Scalability</strong></td>
<td>beyond 50,000 nodes, millions of cores</td>
</tr>
<tr>
<td><strong>Dedicated Hardware</strong></td>
<td>Clusters, clouds, and supercomputers</td>
</tr>
<tr>
<td><strong>Scheduling</strong></td>
<td>Balancing job turnaround time and utilization with optimal job placement</td>
</tr>
<tr>
<td><strong>Resiliency</strong></td>
<td>Includes automatic fail-over architecture with no single point of failure</td>
</tr>
<tr>
<td><strong>Health Checks</strong></td>
<td>Monitors and automatically mitigates faults with a comprehensive health check framework</td>
</tr>
</tbody>
</table>
Runtime Licenses

- Main GUI, APO SDK with C-interface
  CarMaker for Simulink

- FMI, SL plugins

- Maneuver, road, traffic, driver GUIs

- Vehicle GUI with encryption

- Test Manager
  Test Configurator

- Simulation backend

- Remote control

- IPGMovie, IPGControl, Instruments

Office

HIL
Runtime Licenses

- Create simulation frameworks
- Integrate models
- Build scenarios with road networks and traffic
- Create and encrypt virtual prototypes
- Create test catalogs
- Interactively run simulations
- Execute pre-defined test catalogs
- Automation interface
- No GUI
Use Case Scenarios
Office simulations

CarMaker/Office
Full functionality

Simulation framework established

CarMaker/Office Operate
For interactive operation by humans

CarMaker/Office Runtime
For automatic execution

Third-Party Software

© IPG Automotive GmbH, 2018
Use Case Scenarios

HIL simulations

CarMaker/HIL
Full functionality

Simulation framework established

CarMaker/HIL Operate
For interactive operation by humans

CarMaker/HIL Runtime
For automatic execution

Third-Party Software

Full functionality Simulation framework established
HPC Applications
Just the first ideas

- Use HPC to test millions of test kilometers in simulation using test automation
- Continuous integration - Robustness checks of daily builds
- Stochastic vs. deterministic traffic simulation - Simulate the unexpected
- Generate simulation data to train neural networks

- Identify the best vehicle dynamics setup
- Calibrate and analyze emissions and energy consumption with design of experiments (DoE)
IPGDriver is now able to identify a driving strategy for the fastest ride in double lane change and soon in slalom maneuvers.