Continental
The Future in Motion

Mastering Unexpected Situations Safely

SensePlanAct

Chassis & Safety | Vehicle Dynamics

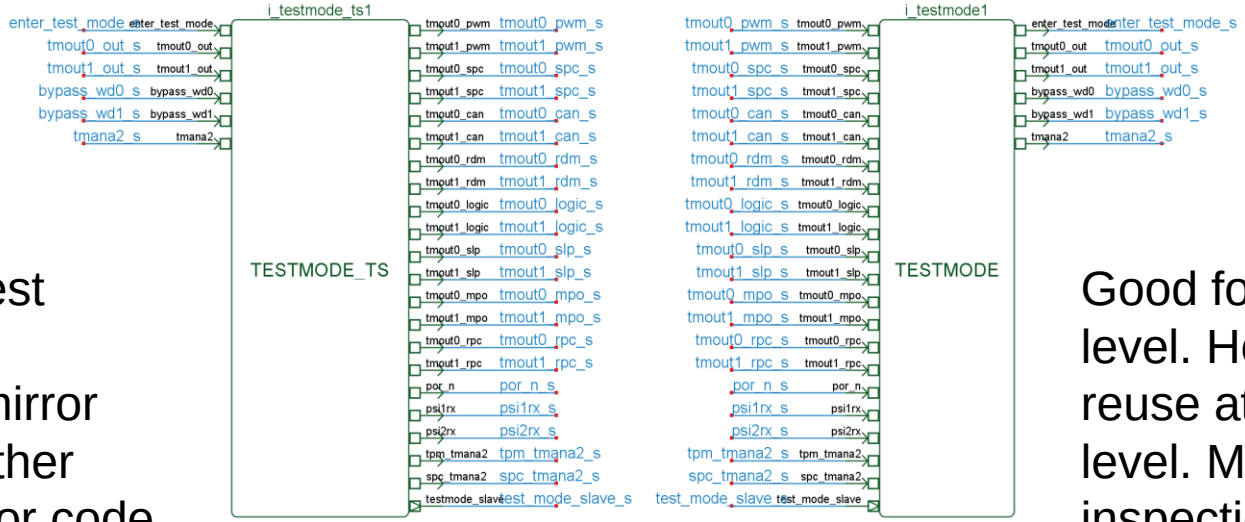
System Evaluation of UVM-SystemC

Coside Usergroup Meeting 18.10.2016

Agenda

SystemC IP Validation before UVM-SystemC

› Module Level

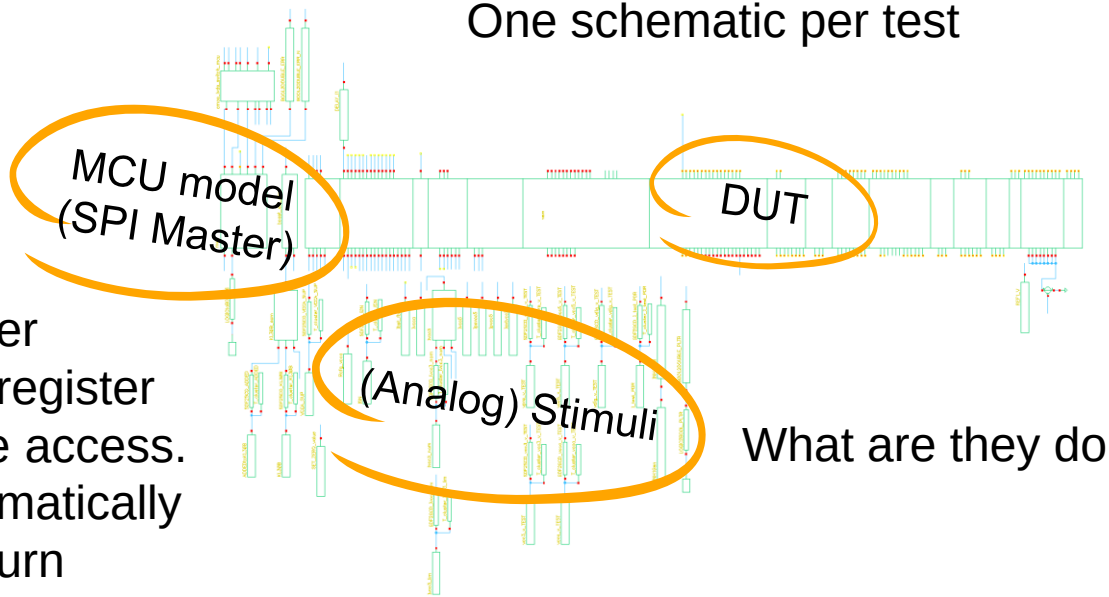


Describe test stimuli in a SystemC mirror module. Either schematic or code.

Good for module level. However, no reuse at system level. Mainly visual inspection.

SystemC Validation before UVM-SystemC

One schematic per test



SPI Master provides register read/write access. Can automatically check return values.

What are they doing?

Requirements to next testbench generation

- Standardized format for stimuli and result checks
- No secret waveform generators
- Can run in regression
- Module level tests (partly) reusable at toplevel

Agenda

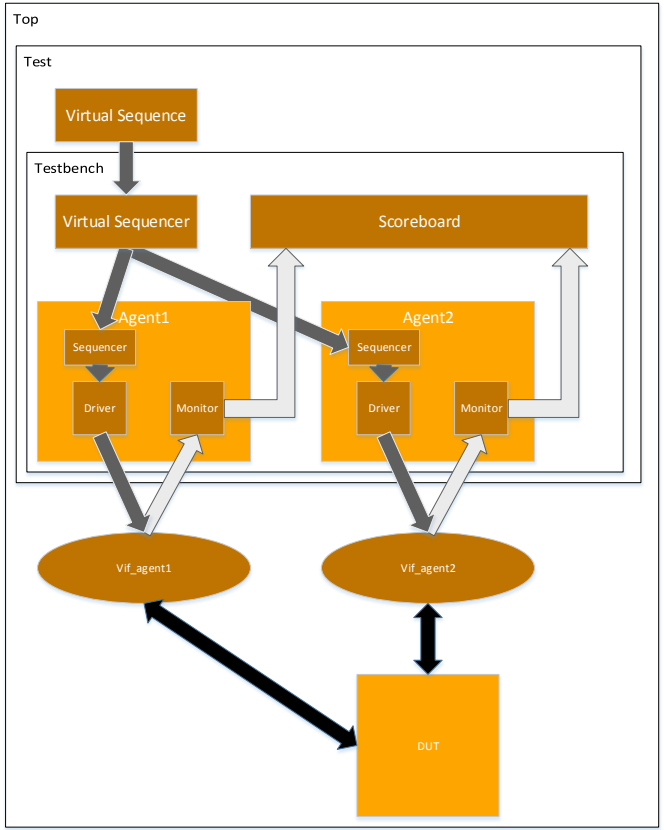
What is UVM?

- **U**niversal **V**erification **M**ethodology to create **modular, scalable, configurable and reusable test benches** based on verification components with standardized interfaces
- **Class library** with features dedicated to verification, e.g.,
  - phasing
  - component overriding (factory)
  - configuration
  - scoreboarding
  - reporting

Main concepts in UVM

- Clear **separation** of test stimuli and test bench
  - So stimuli can be developed and reused independently
- Test bench **abstraction levels**
  - Test bench components communicate in TLM
- Non-intrusive test bench **configuration and customization**
  - configuration and resource database
  - Factory design pattern
- Well defined **execution** (phases) and **synchronization** (objections) process
- **Reusable verification components**

General UVM setup

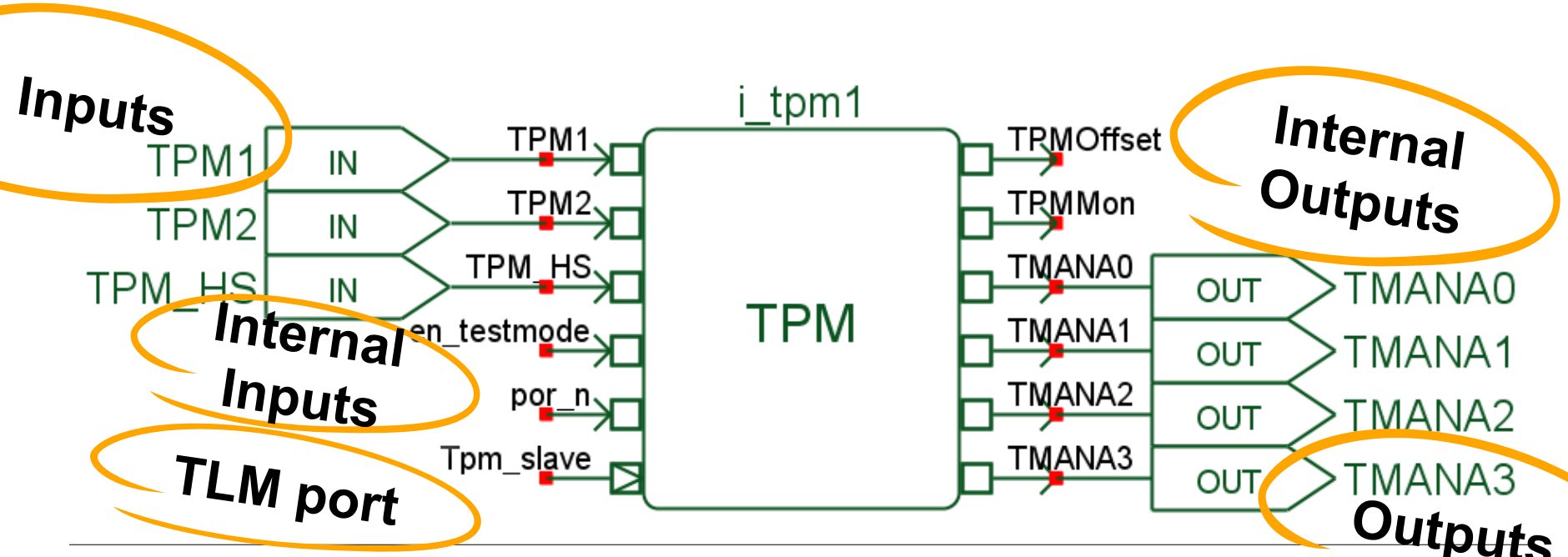


UVM-SystemC

- Developed in the EU 7th framework programme Verdi (2011 – 2014)
  - Continental participated in Verdi
- Donated to accelera public preview release available since December 2015
- At Continental currently work is ongoing to put UVM-SystemC into industrial use.

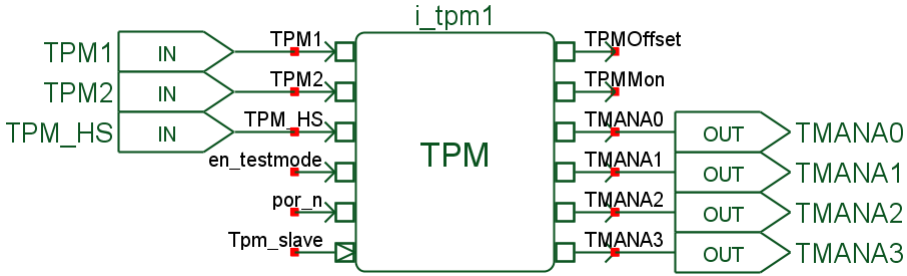
Agenda

Detailed view on a Module



Module level test setup with UVM-SystemC

› Separate interfaces to ease later reuse

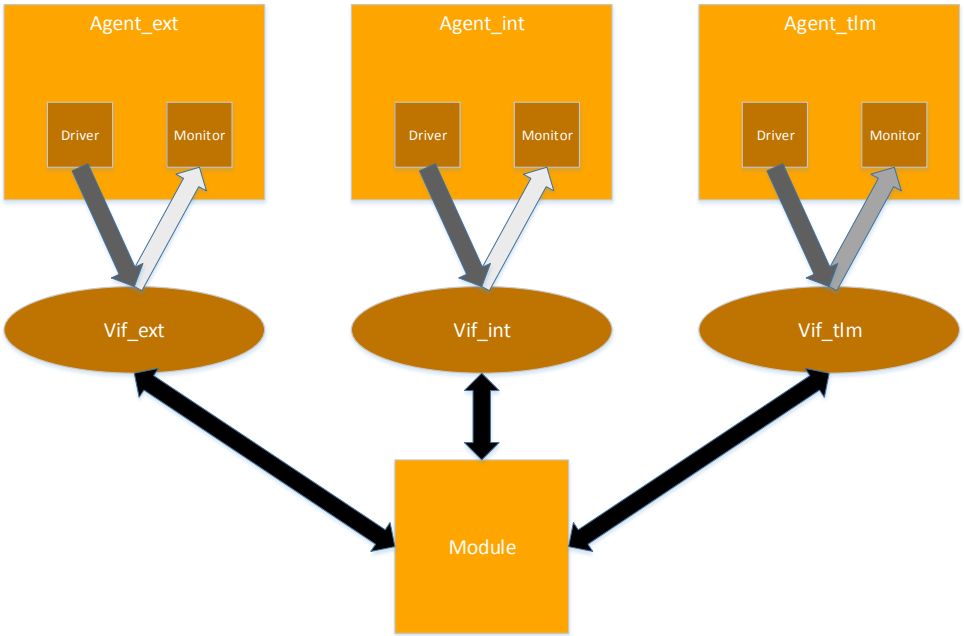


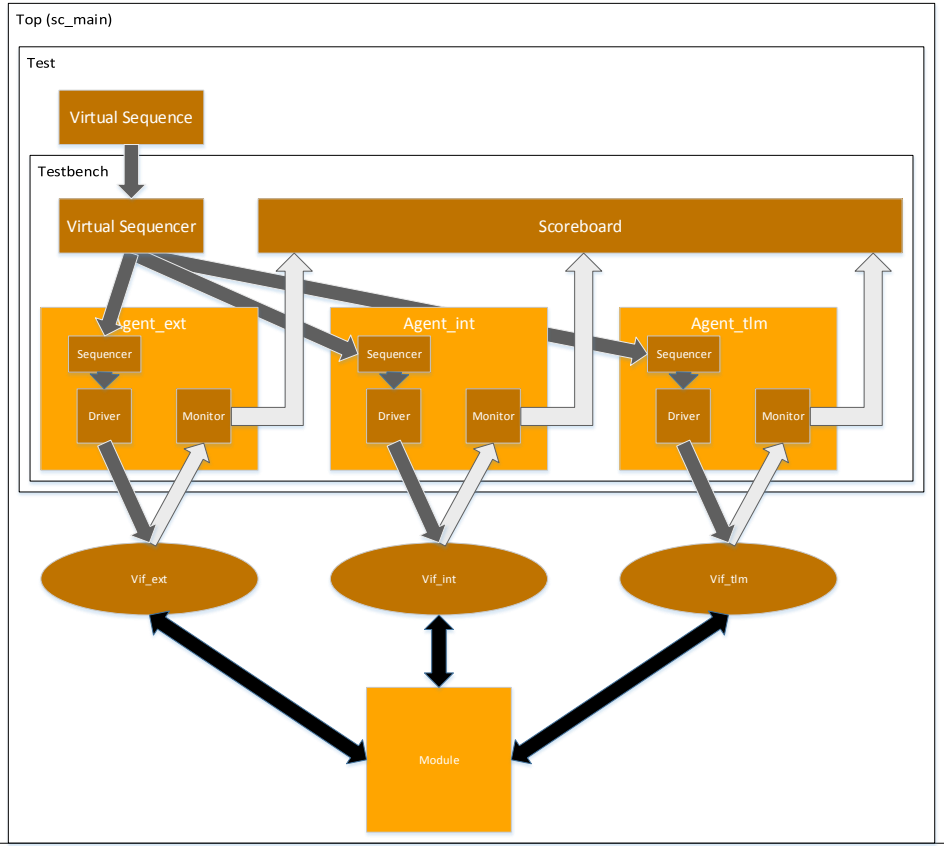

```
class vif_ext:public sc_core::sc_module
{
public:
   sc_core::sc_signal<double          > TPM1;
   sc_core::sc_signal<double          > TPM2;
                        ...
}

class vif_int:public sc_core::sc_module
{
public:
   sc_core::sc_signal<int             > en_testmode;
   sc_core::sc_signal<bool            > por_n;
                        ...
}

class vif_tlm:tlm::tlm_bw_transport_if<>,public sc_core::sc_module
{
public:
   tlm::tlm_initiator_socket< > Tpm_slave;
                        …
}
```

Agents for Module Level Test Setup

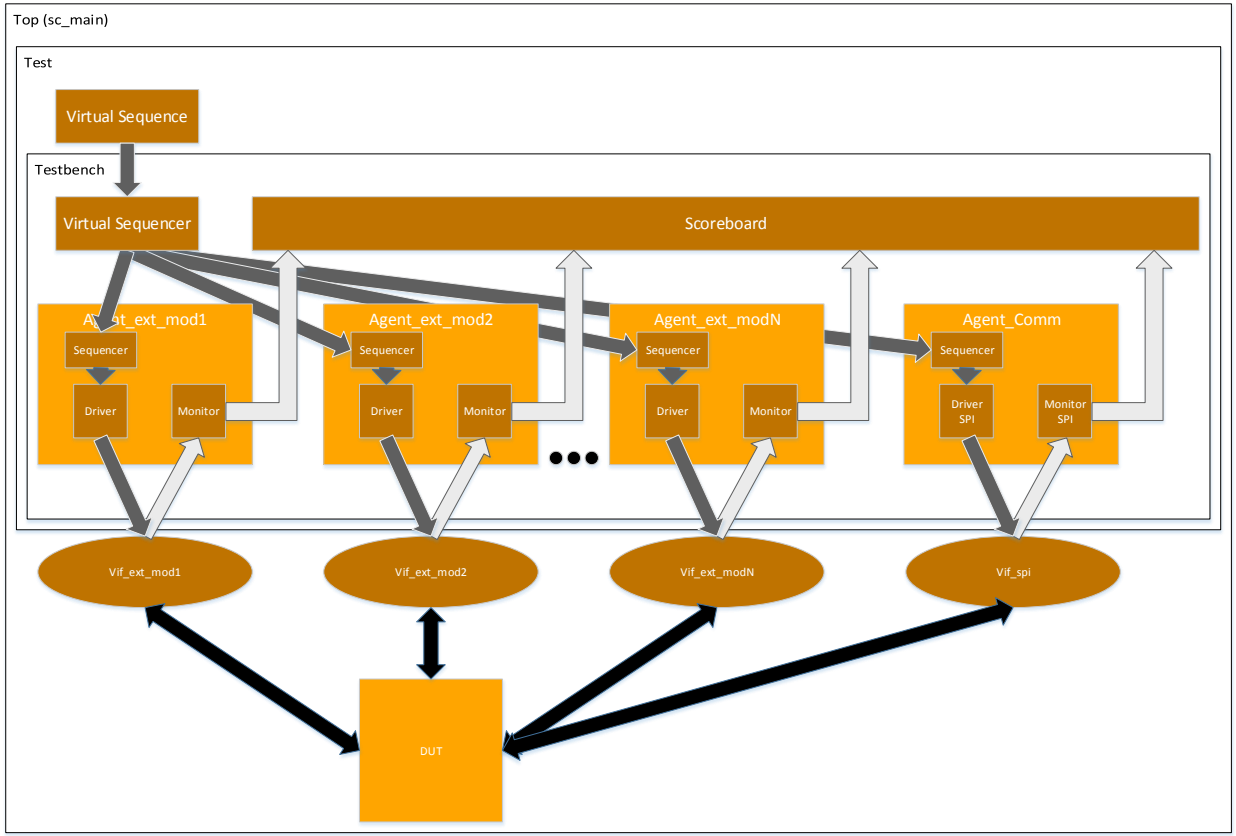



Top (sc_main)
Test
Virtual Sequence
Testbench
Virtual Sequencer
Scoreboard
Agent_ext
Sequencer
Driver
Monitor
Agent_int
Sequencer
Driver
Monitor
Agent_tlm
Sequencer
Driver
Monitor
Vif_ext
Vif_int
Vif_tlm
Module

Agenda

Going up to System-Level

- External pins also exist at system level
  - Reuse Agent_ext
- Internal pins are not accessible at system level
  - Outputs are not directly observable
  - Inputs need to be provided by other modules
- TLM bus not accessible at system level
  - Access handled by e.g. SPI interface
  - Agent on communication interface shall be able to process the same sequences as Agent_tlm


Top (sc_main)
Test
Virtual Sequence
Testbench
Virtual Sequencer
Scoreboard
Agent_ext_mod1
Sequencer
Driver
Monitor
Agent_ext_mod2
Sequencer
Driver
Monitor
Agent_ext_modN
Sequencer
Driver
Monitor
Agent_Comm
Sequencer
Driver SPI
Monitor SPI
Vif_ext_mod1
Vif_ext_mod2
Vif_ext_modN
Vif_spi
DUT

Agenda

Results

- Better test coverage
  - No need of extra stimulus modules or test benches to test some special cases
- No need of dedicated test stimulus modules
  - Still need agents
- Reuse: agents, interfaces, monitors from module level to system level test setup
  - Little extra effort in creating system level test setup
  - Module level test sequences abstracting some bit transactions can be reused at system level

Results

- Structured verification environment
  - Easy for new/other teammates to learn & contribute
- Configurability
- Analog pins can be easily observed
  - Instead of manual waveform analysis

Coside UVM Generator

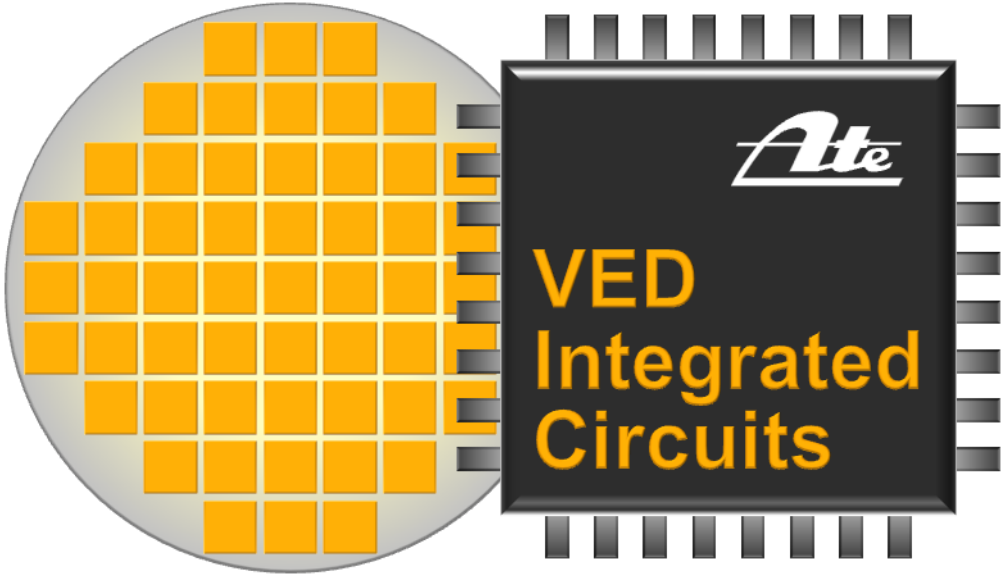
- In a first trial we manually created our UVM test bench.
  - High manual effort 1-2 PM
  - Most of the time goes in generation and debug of framework
- Recreation with alpha version of UVM Generator
  - Framework generation within one day
  - Test setup within one week

Agenda

Future Work

- Generation of possible top level sequences
  - From module level sequences we can directly reuse external and TLM part of the sequence
  - What about internal signals?
  - Idea:
    - Detemine module level sequences driving this signal as expected.
    - Combine sequences to system level sequences.

Thank you
for your attention!



ASIC solutions for Vehicle Dynamics

Safe and Dynamic Driving towards Vision Zero

SensePlanAct
Chassis & Safety

Continental