Dynamic Fault Injection Library Approach for SystemC AMS

Thomas Markwirth – Fraunhofer IIS
Paul Ehrlich – COSEDA Technologies
Dominik Matter – Hella Aglaia Mobile Vision GmbH
Cross-domain system level modelling

- Exploration and validation of the overall system
- Concept engineering support and system level verification
- Combined simulation HW (an.+dig.), SW and non-electrical environment
  - Trade-off: high performance and model accuracy

SystemC/ SystemC AMS is the solution!

© Accellera Systems Initiative
Ensuring functional safety

• Safety integrity verification by fault simulation
• Checking correct and complete implementation of technical safety requirements (ASIL)

  • Improvement of diagnostic coverage and design robustness
  • State of the art: fault injection by direct integration
  • Introduced: Approach for dynamical fault injections for SystemC AMS
Dynamical Fault Injection approach

Preparations e.g.:
- create module hierarchy
- port bindings

Execution of a SystemC simulation

Test case simulation run

Design Under Test (DUT) in SystemC/ SystemC AMS

Test description
Dynamical Fault Injection approach

- Design Under Test (DUT) in SystemC/ SystemC AMS
- Test description

Execution of a SystemC simulation

- stimuli
- checker

Test case simulation run
Dynamical Fault Injection approach

- Separation between DUT and test environment
- No DUT code change
Dynamical Fault Injection approach

• How to create fault descriptions?
Fault descriptions:

• Manually defined...

For each fault:

Definition of:
• location (port inside the design hierarchy)
• Type of fault model
• optional configurations
Fault descriptions:

• Manually defined...

For each fault:

Definition of:
• location (port inside the design hierarchy)
• Type of fault model
• optional configurations

• ...or alternatively:

Definition of used fault type, and...

...scanning Design hierarchy e.g. for:
• Naming conventions
• Port MoC
• Port type
Fault injection library structure

• Low level failure structures depend on MoC

• Established fault models
  – (stuck-at, crosstalk, open/short, delay, bridge, glitch)

• Scenarios
Fault occurrence configuration

- Periodical

- Statistical variations of ...
  - ...location
  - ...time
Fault value configuration

• Continuous value sweep

• Sequence of simulations (fault parameter sweep)
Battery management application (IKEBA)

- Functional validation - nominal
Battery management application

- Balancing-network
- Chain of monitoring and balancing ICs
- Temperature sensor
- Controller with management software
- Battery model
- Driving cycle as load current
- Current sensor
Battery management application

- Controller model with software applications
Investigation of battery faults

- Battery with injected faults
Battery management application (IKEBA)

• Functional validation - HiL

Model Input

HW demonstrator

Model Output

Validation

1). Accuracy of estimation
2). Cell symmetry
3). Clock time & robustness

Range, safety

Fault Injections

DSpace

Batt. model

Battery Monitoring IC

Battery Monitoring IC

Battery Monitoring IC

Battery Monitoring IC

µC

BMS SW

Battery Monitoring IC

Battery Monitoring IC

Battery Monitoring IC

Current, Temperature

Range, safety
Approach used as stimulus generator

1. Ramping generator

2. Insertion of nonidealities
Conclusion

• Dynamical fault injection approach
  – Clear separation between DUT and test environment
  – Avoids code changes in the DUT
  – Prepared and user defined fault models
  – Applicable in real-time systems (HiL)

• Usage as in-depth stimulus generator possible
  – Further investigation in process
  – Enables new testbench concepts

• Wide range of investigation on system level enabled
  – Diagnostic coverage improvement
  – Software testing
  – Variation testing
Outlook

• Increase user friendliness
  – Automatical generation of the fault injection stimulus model (requirement management tools)
  – Alternative GUI-based approach

• Extensions of available fault models
Questions ?
What is the injection approach in details?

• SystemC callback used: *before_end_of_elaboration*

• Searching in design hierarchy for target port

• reconnecting port

• Instantiate and insert failure structure