Next Generation ISO 26262-based
Design Reliability Flows

Jörg Große & Sanjay Pillay

www.onespin.com www.austemperdesign.com
Powerful Solutions Require Strong Technology and Apps

**Advanced Design Verification**
- Agile Design Evaluation
- Metric-Driven Verification
- Block Integ. Validation

**Innovative Specialized Solutions**
- FPGA Impl. Verification
- Safety Critical Verification
- C++/SysC Des. Verification

**High Performance, Comprehensive Technology Platform**
- Automated Inspection
- Design Exploration
- Sequential EC RTL
- Assertion Verification
- Operational Assertions
- Observation Coverage
- DV Apps
- X-Prop
- Connect
- Register
- Scoreboard
- Protocol Comp.
- Activation
- SystemC/C++
- Arithmetic
- Fault
- Propagation
- Fault
- Detection
- Fault
- Injection
- Security
- Verification
- Specification Verification

- **Proof Engines**
  - SystemVerilog
  - VHDL
  - SystemC
  - SVA
  - PSL

- **Formal Model**
- **Adv. Debug**
- **LaunchPad**

**Jörg Grosse – Product Manager Functional Safety**
• Provider of **End-to-End Functional Safety** Tool suite for Automotive, Industrial, Medical and Enterprise Markets.

• One-stop solution for ASIC vendors to **analyze, augment and verify** their designs for Functions Safety Compliance

• Based in Austin, TX, USA; Founded 2015

• Tools in production with customers

Sanjay Pillay
Founder & CEO

Previously responsible for:

- World wide enterprise SSD controller SoC development at HGST/STEC
- World wide SoC development at TRIDENT/NXP/CONEXANT
- Head of audio development at MAXIM
- Functional safety consultant
Agenda

- Introduction
- The design - AXI Crossbar
  - Austemper - Insert safety mechanism – STEP 1
  - OneSpin - prove that insertion did not corrupt main functionality
  - OneSpin - identify faults missed by safety mechanism
  - Austemper - insert additional safety mechanisms – STEP 2, STEP 3
  - OneSpin - prove that insertion did not corrupt main functionality
  - OneSpin - prove that diagnostic coverage has improved
  - OneSpin - identify/debug dangerous faults
- Integration with Fault Simulation
- Results and Conclusion

✓ Real design
✓ Hands-on tutorial
✓ Questions welcome
**Objective:**
Freedom from unacceptable risk of physical injury or of damage to the health of people either directly or indirectly

### Functional Safety Risks
- **Systematic Failures**
  - Design errors
  - Tool errors
- **Random failures**
  - Hard errors
  - Soft errors

### Risk drivers
- Continuous increase in flow and tool complexity
- Continuous increase in functionality
- Increasing density of the design process node
- Decreasing energy levels

### Risk management through functional safety standards
- Minimize systematic errors
- Safeguard against random errors

Safety mechanisms prevent/control random hardware failures
Types of Safety Mechanisms

SOFTWARE MECHANISMS
- Self-Test Routines
- Watchdog Timers

HARDWARE MECHANISMS
- Error Correcting Codes
- Parity bit
- Lockstep
- TRM with Voting Logic
- LBIST
Fault Classification and Metrics

Safe faults
- Not in safety relevant parts of the logic
- In safety relevant logic but unable to impact the design function (cannot violate a safety goal)

Single point faults
- Dangerous, can violate the safety goal and no safety mechanism

Residual faults
- Dangerous, can violate the safety goal and escape the safety mechanism

Multipoint faults
- Can violate the safety goal but are observed by a safety mechanism
- Sub-classified as “detected”, “perceived” or “latent”

Safe Faults: do not propagate to outputs
Detected Faults: propagate to outputs but detected by safety mechanisms
Dangerous Faults: propagate to outputs and missed by safety mechanisms
Observation and Diagnostic Points

Note: faults propagating to observation points but not to diagnostic points are definitely dangerous
The Candidate Design

AMBA AXI Fabric
- 2 Master ports
- 2 Slave ports
- Separate Read and Write channel FIFOs
- Configurable FIFO depth
- Single Clock Domain

Functional Safety
- None
# Austemper Safety Synthesis

## FEATURES

| Hamming code based $n$-bit detect/$m$-bit correct | ✓ | ✓ |
| Structures supported | RAM, ROM, Reg Files, FIFOS, stacks | Flip-Flop Banks |
| User-Defined Structure selection | ✓ | ✓ |
| Auto-Grouping of Structures | ✗ | ✓ |
| User selectable Option (Parity vs EDC vs ECC) | ✓ | ✓ |
| Multi-pass w/ incremental safety insertion mode | ✓ | ✓ |

## ERROR DETECTION & CORRECTION

<table>
<thead>
<tr>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redundancy</td>
</tr>
<tr>
<td>Duplication/Triplication</td>
</tr>
<tr>
<td>Multi clock designs</td>
</tr>
<tr>
<td>Auto-Identification</td>
</tr>
</tbody>
</table>

## PROTOCOL CHECKS

| Covered Items | Interface Parity/protocol, FIFO overflow/underrun | FSM Valid states & transitions |

© Accellera Systems Initiative
Safety Synthesis Steps

**STEP 1**
- Use Annealer to insert end-to-end datapath parity

**STEP 2**
- Use RadioScope to insert parity protection on selected state elements

**STEP 3**
- Use RadioScope to duplicate register blocks
STEP 1 : Annealer

TOOL INPUTS

DESIGN FILES : Step 1 Verilog
DESIGN TYPE : RTL
SYNTHESIS FILE : List of Datapath Signals

TOOL OUTPUTS

DESIGN FILES : Verilog with E2E parity inserted and built-in safety alarms
DESIGN TYPE : RTL
ERROR CHECK : Verilog Test bench and Test cases.
EQUIVALENCE CHECK : script to verify absence of corruption with third party tool
Annealer

Annealer -T 15
-F ../Inputs_AN/axi_cross.f
--clkdef ../Inputs_AN/axi_cross.clk
-u -l
-top-an-no

-intfparlist Inputs_AN/IntfPar.list

ATDOpIntParityMod_axi_cross #(.GENWIDTH(DW+1)) axi_i0_wdata_par_GenInst (.op_sigs({axi_i0_wdata}), .op_parity(axi_i0_wdata_par));
ATDOpIntParityMod_axi_cross #(.GENWIDTH(DW+1)) axi_i1_wdata_par_GenInst (.op_sigs({axi_i1_wdata}), .op_parity(axi_i1_wdata_par));
ATDOpIntParityMod_axi_cross #(.GENWIDTH(DW+1)) axi_s0_rdata_par_GenInst (.op_sigs({axi_s0_rdata}), .op_parity(axi_s0_rdata_par));
ATDOpIntParityMod_axi_cross #(.GENWIDTH(DW+1)) axi_s1_rdata_par_GenInst (.op_sigs({axi_s1_rdata}), .op_parity(axi_s1_rdata_par));

input ATD_safety_enable_axi_cross, //Interface Parity to data path
input ATD_safety_err_inject_axi_cross,
input [1:0] ATD_safety_err_clear_axi_cross,
output [1:0] ATD_safety_err_out_axi_cross;
STEP 2 : RadioScope

TOOL INPUTS

DESIGN FILES : Verilog
DESIGN TYPE : RTL
SYNTHESIS FILE : List of Target State Elements

TOOL OUTPUTS

DESIGN FILES : with parity inserted and built-in safety alarms
DESIGN TYPE : Verilog RTL
ERROR CHECK : Verilog Test bench and Test cases.
EQUIVALENCE CHECK : script to verify absence of corruption with third party tool
STEP 2 Output

axi_cross_wrapper_ATD

Master0 RD Port
Master1 RD Port
RD_ARBITER
RD_CHANNEL
Data FIFO
WG_ARBITER
WR_CHANNEL
Data FIFO
Master0 WR Port
Master1 WR Port
Slave0 RD Port
Slave1 RD Port
Slave0 WR Port
Slave1 WR Port

axi_cross

Safety Err Out

Control/Observe Parity

PGC
PGC
PGC
PGC

Austemper Additions
Austemper Demo– Parity Insertion
Verify Safety Mechanism

• Original design functionality corrupted?
  – Use Combinational/Sequential Equivalence Checking
Verify Safety Mechanism

- Safety Mechanism detects enough faults?
  - Verify diagnostic coverage
Formal Fault Analysis Flow

- Safety-Critical Function
  - Fault
  - Activate
  - Propagate
- Hardware Safety Mechanism
- Observation Points
- Diagnostic Points
- Fault List
- Input Constraints
- RTL / Netlist
- Debug Dangerous Faults
- FDA
- Fault Classification Report/DB
DEMO ONESPIN
DEMO ONESPIN – EC

• Show setup
• Highlight automated mapping of inputs and outputs
  • New ports remain unmapped
• Run compare
  • Highlight automatic identification/mapping of states not related to SMs
• Explain compare result: designs are equivalent
DEMO ONESPIN – EC

source fa_setup/elaborate_design.inc
source fa_setup/elaborate_orig_design.inc
compile -both
set-mode ec
map -input -output
compare

Finished mapping of Inputs/Outputs in 0.00 sec.
- Mapped inputs: 398 (unmapped: 4/0),
- Mapped outputs: 397 (unmapped: 2/0),
- Generating combined model.
- Reduced model after mapping and constraining:
  - Inputs: 402 x 398
  - Primary Clocks: 1 x 1
  - Data Inputs: 401 x 397
  - States: 567 x 380
  - Outputs: 399 x 397
  - Assertions: 0 x 0

The designs are not fully equivalent as there are unmapped outputs in the golden design.

ec> add_ignored_output -golden { ATD_safety_err_out_axic[0] } { ATD_safety_err_out_axic[1] }
- Number of outputs to ignore added: 2
ec>

ec> -R- Outputs: HOLD=397
- R- States: HOLD=423
- R- The designs are equivalent.
ec>
DEMO ONESPIN – FDA

- Read RTL design (Parity SM only)
- Show/explain observation points file
- Show/explain diagnostic points file
- Show/explain fault population
  - Note that we are only injecting faults on states (not on nets)
- Explain that we only consider faults propagating to observations points
  - Faults propagating to SM only are not relevant
- Explain using fault sampling for quick analysis
- Run Fault Detection App (FDA)
- Explain results
  - Only ~30% of the states are protected, as expected
Results

• OneSpin 360 EC: we have proven that functionality has not been corrupted
• OneSpin 360 DV: low fault coverage

➢ Additional safety mechanism might be required
STEP 3: RadioScope

**TOOL INPUTS**

- DESIGN FILES: Step 2 Verilog
- DESIGN TYPE: RTL
- SYNTHESIS FILE: List of registers to duplicate

**TOOL OUTPUTS**

- DESIGN FILES: Verilog with duplication and Checkers with built-in alarms
- DESIGN TYPE: RTL
- ERROR CHECK: Verilog Test bench and Test cases.
- EQUIVALENCE CHECK: script to verify absence of corruption with third party tool
DEMO AUSTEMPER
Statistics for SM insertion (FIFO depth = 16)

<table>
<thead>
<tr>
<th></th>
<th>OriginalDesign</th>
<th>DataParity</th>
<th>DataParity+RegParity</th>
<th>DataParity+RegParity+RegDup</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumofFF'sInDesign</td>
<td>1328</td>
<td>1512</td>
<td>1535</td>
<td>1605</td>
</tr>
<tr>
<td>NumofNodes</td>
<td>6029</td>
<td>6635</td>
<td>7448</td>
<td>7985</td>
</tr>
<tr>
<td>TransCount</td>
<td>99302</td>
<td>109096</td>
<td>115191</td>
<td>121667</td>
</tr>
<tr>
<td>% Increase in FFCount</td>
<td></td>
<td>13.86%</td>
<td>15.59%</td>
<td>20.86%</td>
</tr>
<tr>
<td>% Increase in NodeCount</td>
<td></td>
<td>10.05%</td>
<td>23.54%</td>
<td>32.44%</td>
</tr>
<tr>
<td>% increase in TransCount</td>
<td></td>
<td>9.86%</td>
<td>16.00%</td>
<td>22.52%</td>
</tr>
</tbody>
</table>
Verify Safety Mechanisms

- Original design functionality corrupted?
  - Use Combinational/Sequential Equivalence Checking
• Safety Mechanism detects **enough** faults?
  – Verify diagnostic coverage
DEMO ONESPIN
DEMO ONESPIN – EC

• Quick rerun
DEMO ONESPIN – FDA

• Show additional SMs added to diagnostic points file
  • Datapath parity and register duplication
• Rerun FDA on previous sample using “buggy design”
  • All looks OK
• Show results on FDA on ALL states (not sample)
  • 98.5% fault coverage – expected 100%
  • Show list of non-detected states
• Show email from Arun

Hi Jorg,

I reviewed the design for below signals, below are my comments:
1. state bits made sense, as we use only bits[1,0] is u_axi_cross_wr.ch and bit[0] in u_axi_rd.ch
   a. the remaining bits are optimized away
2. localgrant this is a user error, in the RTL I generated I missed adding these signals to duplication list.
   a. Issue with working in different directories [Sorry, my bad]

• ? Run stick checks to prove constant states in FSM (optimized away)
DEMO ONESPIN – FDA

• Rerun “localgrant” faults on fixed design
  • Highlight how quick it is to just modify Austemper script and regenerate design
• Show final table below
  • 85% coverage when considering FIFO Depth 16
  • Other figures refer to FIFO Depth 2
• Explain why coverage is lower when considering nets

<table>
<thead>
<tr>
<th></th>
<th>State only</th>
<th>RTL with nets</th>
<th>Gate level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number ST faults</td>
<td>~700</td>
<td>~40000</td>
<td>100000</td>
</tr>
<tr>
<td>SPF</td>
<td>100%</td>
<td>~85%</td>
<td>~80%</td>
</tr>
<tr>
<td>Runtime</td>
<td>~1 hour</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
DEMO ONESPIN – FDA

• Focus on debug
• Show a dangerous (non-detected) fault (a net or gate input)
• Show the driving FF case been detected
• Explain that some fanout logic of some FF is not coverage by SMs
  • Perhaps Austemper could suggest further improvements to achieve ASIL
Results

- OneSpin 360 EC: we have proven that functionality has not been corrupted
- OneSpin 360 DV: additional safety mechanisms detect previously undetected faults
- OneSpin 360 FPA: uncovered a usage mistake
- OneSpin 360 DV: identify/debug dangerous faults
Integrating Formal with Fault Simulation

• Analysis of software safety mechanisms requires fault simulation
  – Formal tools cannot read self-test software routines

• Analysis of large SoCs requires fault simulation
  – Formal tools have capacity limitations

• Can formal verification still help in these circumstances?
  – Yes!
• Two-mode approach fits well with simulation flow
KaleidoScope: Austemper Fault Simulator

- Safety Context derived entirely from RTL simulations
- Concurrent fault propagation without restrictions
- ~4 orders of magnitude faster than GLS
- Auto-classification of Fault outcomes
- Integration with Analysis front-end for computing DC
- Smart Fault injector automates the process
- Unresolved faults dispatched via either

- Hybrid Simulation
- Formal – Deep Analysis
KaleidoScope: Austemper Fault Simulator

- Analog IP, High-level models, ...
- Insufficient Safety Context, Deep Faults, ...

Further Analysis

Detected

Propagatable

Parallel Fault Injector

VCD File

Design Files RTL, Netlist

Designated Safety Alarms

RTL Simulation

KaleidoScope

HSE

onespin
Conclusions

• Hardware safety mechanisms detect random hardware faults

• Hardware safety mechanisms must be verified
  – Do not corrupt normal functionality
  – Detect enough faults, depending on target SIL

• Austemper tools automatically insert a variety of safety mechanisms
• OneSpin Safety-Critical Solution automates verification tasks

• Efficient and streamlined flow to ISO 26262 Certification
References


2. S. Marchese - J. Grosse – Formal fault propagation analysis that scales to modern automotive SoCs, DVCon Europe 2017

3. S. Marchese - Using formal to verify safety-critical hardware for ISO 26262, OneSpin Solutions White Paper

A note to offline readers: to receive a video of the demo parts of this tutorial please contact

Joerg.Grosse@onespin.com
Sanjay.Pillay@austemperdesign.com
Questions?