# Fault Proof: Using Formal Techniques for Safety Verification and Fault Analysis



Adrian Traskov Thorsten Ehrenberg Sacha Loitz



Abdelouahab Ayari Avidan Efody Joseph Hupcey III





## Outline

#### **Introduction to Fault Injection**

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Example

**Safety Mechanism Validation Flow** 

Formal-based Fault Injection

Fault Coverage

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary





# ISO26262 Safety Validation



SYSTEMS INITIATIVE © Accellera Systems Initiative

IR.

## Random Faults and Safety Mechanism (SM)

Random Faults

□ physical defects that can occur in system components during system operation

- Purpose of Safety Mechanism
  - Control random faults
    - Detect all faults
    - Provide a deterministic and correct reaction to faults
  - Guarantee safety operation of the system
    - Recover the system, or
    - Go to a safe system state
- Validation/Verification of Safety Mechanism

#### Completeness

• Check the ability to detect and handle all possible (important, resp.) faults

#### Correctness

- Check that the safety mechanism specification/requirements are satisfied
- For example:
  - Check design behaves as without presence of faults only for a specified list of faults.
  - Check design goes to a safe state for a specified list of faults



# **Examples of Safety Mechanism**

- FSM safe encoding
- Triple Modular Redundancy (TMR)
- Error-correction code (ECC)
- Lock-step





### Safety Mechanism: Illustration Using SPI Master Core Example





© Accellera Systems Initiative

DESIGN AND VERIF

#### SPI Master Core: Assume Faults Occurs in Clock Generator



© Accellera Systems Initiative

SYSTEMS INITIATIVE

20

### **SPI Master Core:** Faults Could Affect Functional Safety



© Accellera Systems Initiative

SYSTEMS INITIATIVE

2016

#### SPI Master Core: Fail-Operational Safety Mechanism Handel Faults



© Accellera Systems Initiative

SYSTEMS INITIATIVE

# Safety Mechanism for Clock Generator





accellera

2016

IRCE

## Outline

#### **Introduction to Fault Injection**

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Examples

**Safety Mechanism Validation Flow** 

Formal-based Fault Injection

Fault Coverage

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary



DESIGN AND VE

## **Regular Simulation**







### Fault Simulation: Fault Detected & Corrected







## Fault Simulation: Fault Detected & Not Corrected







## Fault Simulation Fault Not Detected



• The fault is not detectable in this stimulus

➤ Is the fault not detectable at all?

> How to decide this?

If it is detectable, then which stimulus can be used for detection?



### **Fault Simulation Regression Results**

Sim

Not Detected

Sim

**Detected &** 

Corrected

- Regression using Questa EVP
  - Number of Tests 372
    - 4 Test cases (UVM)
    - Fault Models: Stuck-at-1
    - 372 Faults
  - Results
    - Non-Propagatable 67%

#### – Run Time 2H 16min



SYSTEMS INITIATIVE © Accellera Systems Initiative

### **Questa Formal Model for Fault Injection**





SYSTEMS INITIATIVE © Accellera Systems Initiative

accellera

### Fault Models

- **1. Permanent Faults** (Stuck at 0, Stuck at 1)
  - Irreversible component damage
- 2. Transient Faults (a.k.a. soft-errors, SEU and SET)
  - Environmental Conditions
  - Cause Erroneous States in the system
  - Do not cause permanent damage
  - Hardest to detect
- 3. Intermittent Faults

© Accellera Systems Initiative

SYSTEMS INITIATIVE

- Caused by unstable HW
- Often become **permanent** faults after a period of time



#### **Modelling Faults in Formal**



# Formal Model for Fault Injection





© Accellera Systems Initiative

accellera

SYSTEMS INITIATIVE

### Safety Mechanism Checkers

Questa Verify (/faulty_injection/spi/run_	_session_2/re	gression/config_12	/session	/results/	/formal/forn	nal_verif	y.db)
<u>File View Compile Verify Tools Reports Logs Layout M</u>	<u>/</u> indow <u>H</u> elp						
🖹 + 😅 🖬 🦈 🍜   🤾 🐚 🏙 🗠 斗   🖊 🖺 🥱					Formal	- 🏇	🖉 💓 🛃
Layout Last Settings 🗨				_			
🔢 Design 🔤 🛨 🗗 🖸	k h ngc/faulty	/_injection/spi/run_sessi	on_2/regre	ession/cont	fig_12/vlog/fau	ulty_injectio	n.sv - Default
* Instance	FF Ln#	🗇 🄿 🏠 🗞 🗞 🤧 🤅	<b>♦ <del>96</del> ♦</b>	u_correcti	on_logic_checl	ker	
Final faulty top (4)	108	module correction_1	.ogic_che	cker(			
ref spi (2)	109	input		RESE	TN		
	111	, input		IRO	1		
= spi (2)	112	, input[`SPI SS N	B-1:0]	ss pad	o 1		
🗾 shift	113	, input		sclk	_pad_o_1		
🛓 🔟 clgen (4)	114	, input		mosi	_pad_o_1		
claen 1	115	, input	D-1.01	IRQ_	2		
	117	, input	B-1.0]	ss_pau_	pad o 2		
clgen_2	118	, input		mosi	pad o 2		
🗾 clgen_3	119	);					
🖃 🗾 u correction logic (3)	120		_				
il tmr vote	121	default clocking co	(posed)	ge CLK );	endclocking	;	
	122	default disable iff	(I RESE	1'N ) ;			
I2_tmr_vote	124						
	125		1111111	/////			
intermittent_faulty_clgen_clgen_1_pos_edge	126	11					
u correction logic checker	127	// Check equivalence	e of out	puts			TD0 0
	<b>P</b> 128	check as pad o	assert	property	( IRQ_I	1 ==	IRQ_2
	<b>P</b> 130	check sclk pad o	assert	property	( sclk pad	0 1 ==	sclk pad o 2
	<b>P</b> 131	check mosi pad o	assert	property	( mosi pad	o 1 ==	mosi pad o 2
	132	endmodule				_	
🕮 Project 🗙 🌆 Library 🗶 🕞 Directives 🗶 🏥 Design 🗴 🎩	133						
A Properties 🗧							
Name			Туре	Radius	Clocks	Time	
u_correction_logic_checker.checke	k_IRQ		sva		PCLK		
u_correction_logic_checker.chec	k_mosi_pad_o		sva		PCLK		
u_correction_logic_checker.chec	u_correction_logic_checker.check_sclk_pad_o				PCLK		
u_correction_logic_checker.chec	u_correction_logic_checker.check_ss_pad_o				PCLK		
intermittent_faulty_clgen_clgen_	intermittent_faulty_clgen_clgen_1_pos_edge.cover_control_sig_toggle			4	PCLK	500ns	
intermittent_faulty_clgen_clgen_	intermittent_faulty_clgen_clgen_1_pos_edge.cover_intermittent			8	PCLK	900ns	
intermittent_faulty_clgen_clgen_	_faulty_clgen_clgen_1_pos_edge.assume_control_sig_1				PCLK		
intermittent_faulty_clgen_clgen_	1_pos_edge.ass	ume_control_sig_2	sva		PCLK		
intermittent_faulty_clgen_clgen_	1_pos_edge.ass	ume_design_sig	sva		PCLK		

SYSTEMS INITIATIVE

## Outline

#### **Introduction to Fault Injection**

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Examples

**Safety Mechanism Validation Flow** 

Formal-based Fault Injection

**Fault Coverage** 

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary



#### SPI Master Core Fault Coverage Results

Q	Questa Sir	m-64 QA Basel	ine Assertion:	10.5 Beta - 293	32224 (Covera	ge View)	
<u>F</u> ile <u>E</u> dit <u>V</u>	iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d <b>Verifica<u>t</u>ion Tracker</b>	T <u>o</u> ols Layo <u>u</u> t I	Boo <u>k</u> marks <u>W</u> indo	ow <u>H</u> elp			
🖹 • 🗃 🕻	] 🤣 🍈   🕺 🛍 🍭 았 🚉   🔕 - 🛤 🖺   Не	p 🛛 👔	Layout VMgm	t	ColumnLayo	out Testplan	
Precision 2	2 💴				(u		
			as 2006 222 B	• •	<b>A A I A</b>	•	
1						. <b></b> .	
🔁 / 🗄 - 1	h 🦇 🕪   🗊 📃 100 🖨 🖺 🖺 🛣 🍩   🕅	x 💼 😳 🥠	🛛 🦓 - 🛃 - 🥵 I	📲 <b>- 4</b>			
<u>₩</u> <u></u> <u></u>	Threshold 100 🜩 🗽 🖑 🛪 👒						
🄈 Verificatio	n Management Tracker						
Sec#	Testplan Section / Coverage Link	Type	Goal	Coverage %	of Goal Statu	us Weight	Link Sta
0	🖃 🔆 testplan	Testplan	-	50.87%	50.87%		1 Clean
1	🗄 🔆 SPI Simulation Fault Verification stuck_at_C	) Testplan	100%	42.1%	42.1%		1 Clean
2	🗄 🔆 SPI Simulation Fault Verification stuck_at_1	. Testplan	100%	89.47%	89.47%		1 Clean
3	🗄 🔆 SPI Simulation Fault Verification seu	Testplan	100%	21.05%	21.05%		1 Clean
2	Questa Sir	n-64 QA Basel	ine Assertion:	10.5 Beta - 293	32224 (Covera	ge View)	
<u>E</u> ile <u>E</u> dit <u>V</u>	iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d <b>Verifica<u>t</u>ion Tracker</b>	T <u>o</u> ols Layo <u>u</u> t I	Boo <u>k</u> marks <u>W</u> indo	ow <u>H</u> elp			
🖹 - 🚅 🕻	J 🕏 🍈   X 🐚 🍭 았 🗠   🖉 - 🛤 🖺   Hel	p 🛛 🖁	Layout VMgm	t	ColumnLayo	out Testplan	
Precision 2	2 💴		U		U		
					* T   👬 * 🕷	i 1 <b>0</b> 1	
Ъ 🔁 - 1	🕇 🦇 🛶   🗊 📃 100 🗣 🗉 🖺 🗱 🌋   🗴	K 😰 😨	🦷 📲 🗕 🥳 🖉	🚰 <b>-</b> 🕰			
±± ∓	Threshold 100 🍨 🗽 😴 🗶 🐲						
Verificatio	n Management Tracker						
Sec#	Testplan Section / Coverage Link	Туре	Goal Co	verage % of	Goal Status	Weight	Link Statu
0	🖃 👷 testplan	Testplan	-	100%	100%	1	Clean
1	🖃 🔆 SPI Formal Fault Verification stuck_at_0	Testplan	100%	100%	100%	1	Clean
2	🛓 🔆 SPI Formal Fault Verification stuck_at_1	Testplan	100%	100%	100%	1	Clean
3	🛓 🔆 SPI Formal Fault Verification seu	Testplan	100%	100%	100%	1	Clean
						DE	SIGN AND VER
era )							DVCI
		23				co	NFERENCE AND
(0)	Accellera Sveteme Initiative						

SYSTEMS INITIATIVE © Accellera Systems Initiative

**acc** 

## Outline

#### **Introduction to Fault Injection**

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Examples

**Safety Mechanism Validation Flow** 

Formal-based Fault Injection

**Fault Coverage** 

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary



# Summary

- Functional safety critical components are often small enough to be analyzed using formal techniques
- Formal fault injection is complete regarding legal design input pattern AND failure time points
- Still some work is need to compile formal pass/fail information to fault coverage.





## Questions





© Accellera Systems Initiative