cādence®

Incisive Functional Safety Simulator

Fault injection capability for RTL and gate-level simulation supporting a comprehensive functional safety solution for ISO 26262 and IEC 61508

Part of the Cadence® System Development Suite, the Incisive® Functional Safety Simulator injects faults to verify the ability of your design to handle unexpected events. It can execute with the Incisive vManager™ solution to provide a comprehensive requirements tracing, functional verification, and safety verification solution.

Functional Safety in ISO 26262 and IEC 61508

Functional safety refers to the concept that an overall system will remain dependable and function as intended even in the event of an unplanned or unexpected occurrence. The IEC 61508 specification codified this concept into a standard, from which the automotive industry then derived the ISO 26262 standard. Compliance to these standards is now a requirement flowing from OEMs to Tier 1 integrators and to semiconductor providers. The benefit from implementing a functional safety methodology is a more dependable design that can improve all products.

Compliance with these safety standards consists of three elements identified in Figure 1.

The first element is quality processes, including requirements tracing and the tool confidence level (TCL). Requirements tracing connects enterprise requirements systems to the test input and result output from the verification process at all levels of abstraction through the semiconductor development flow, to document that all the design requirements were implemented and tested. The TCL assesses the error injection risk of each tool in the flow to document the confidence level for the data processing of each tool.

The second element is the quality measurement for both the overall design function and the function of the safety systems. Design function employs digital and analog/mixed-signal functional verification at multiple levels of abstraction. Safety verification inherits much of the functional verification environment to measure the response of the safety systems to injected errors.

The third element integrates all of the data from the first two into a safety manual that is provided with each semiconductor and/or integrated system.

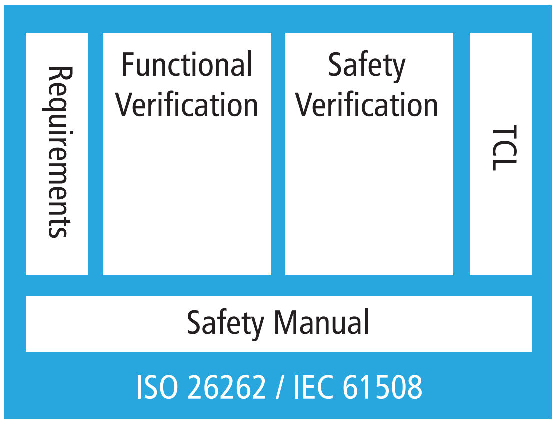


Figure 1: Elements of functional safety standards

Functional Safety Simulation

Simulation requires a new mechanism measure how a design responds to unexpected events. Those events can be modeled as faults. A fault is an error injected into the design to either temporarily or permanently change a logic value at any time during the simulation. Traditional fault simulators implemented for design for test (DFT) applications typically support part of this requirement. The traditional fault simulators may have IEEE language support limitations, timing (SDF) annotation issues, lack of mixed-signal support, and more.

The Incisive Functional Safety Simulator implements the fault models required by ISO 26262 and IEC 61508 within a modern verification flow. At elaboration time, it extracts a fault dictionary that can be localized to a particular hierarchy. A script or Incisive vManager solution is then used to iterate through the fault dictionary injecting permanent and temporary faults. The Incisive Functional Safety Simulator supports the stuck-at (SA0/1), single event upset (SEU), and single event transient (SET) models, in compliance with the ISO 26262 standard.

The simulation proceeds normally until the injection time is reached; the simulation will proceed from this time with the altered (injected) value and progate it through the design. Since the underlying simulator is the Incisive Enterprise Simulator, the fault can propagate through any element it can simulate, including analog transistor models, digital mixed-signal models, SystemC, assertions, etc. Strobe points are set to be able to detect the fault propagation. The value in the faulted or "bad machine" is compared to the value at the same strobe point in the "good machine" until a difference is detected or the simulation finishes. This process is repeated for the whole fault dictionary.

Fault Simulation Finish Types

Type	Fault Run Finished…
ON_TIME	At same time as good run
DELAYED	After good run
PREMATURE	Before good run
TIMEOUT	When specified timeout was reached
STOPPED	Due to the simulator stopping the run

Fault Detection Conditions

		Bad Machine			
		0	1	X	Z
Good Machine	0	U	D	P	P
	1	D	U	P	P
	X	U	U	U	U
	Z	U	U	U	U

Key:
D = Detected P = Potentially Detected U = Undetected

Figure 2: Temporal fault classification semantics

This is where the work in safety verification begins. As indicated in Figure 2, the detection condition for each fault is reported. Faults that are reported as undetected or potentially detected need further debug before they can be classified.

Undetected faults occur when the simulation finishes without any logic difference detected at the strobe point. Potentially detected faults occur when the logic state at the strobe point becomes unknown (X) or high impedance (Z).

Both undetected and potentially detected faults may be dangerous, so the debug task is to identify the reason for the detection condition. For example, a fault may be undetected if the combinatorial logic masks the fault. A fault may also be undetected if fault node is either not controllable from the stimulus or the stimulus used fails to exercise the logic at the faulted node. In each case, further simulation or formal analysis is needed. If the fault is masked or not controllable, it may be removed from the ratio calculation as safe. If it shows that the stimulus does cause the fault to propagate but the strobe fails to detect the fault, then the fault is designated as dangerous.

Faults of the third condition may also be designated as dangerous regardless of whether the simulation finishes ON_TIME, DELAYED, or PREMATURE. The actual designation occurs based on the processing between the strobe point and the safety system output. For example, if the ECC fails to correct the fault due to accumulated error or if the time between the fault insertion and detection by the comparator exceeds the safety goal, the fault could be dangerous-undetected. A SystemVerilog, PSL, or OVL assertion at the safety system output can contribute to automating the designation of detected faults.

The goal of the safety verification process is to identify the dangerous-detected and dangerous faults. The ratio of dangerous-detected faults to the total dangerous and dangerous-detected faults in the system is then used to calculate the Automotive Safety Integrity Level (ASIL).

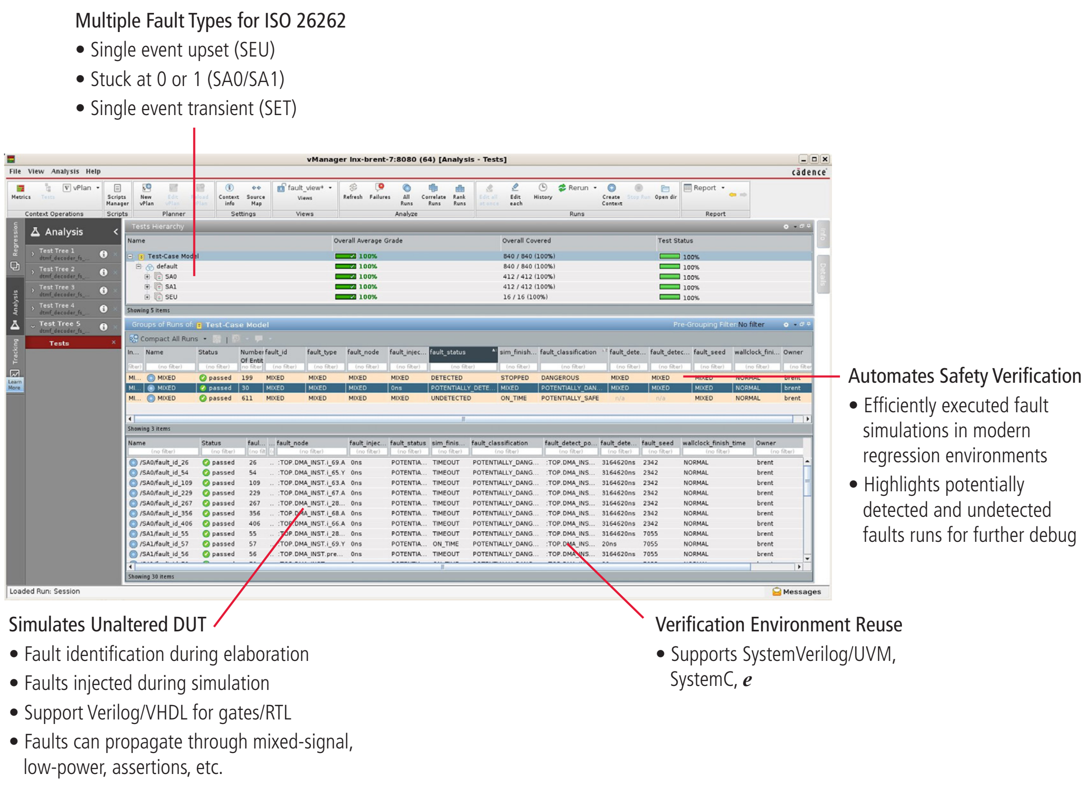
Comprehensive Solution for Safety

In a relatively small design—for example, under a few hundred thousand logic gates plus any analog circuitry—it may be possible to run safety verification using sampled input for the testbench and manually analyze the results. As system complexity increases, however, a more efficient methodology is needed. Safety verification should become part of the functional verification flow (Figure 3) so that sophisticated testbenches in modern functional verification can be used to control the fault injection and to support the debug process. Similarly, the same simulator should be used to eliminate the efficiency loss due to debugging result differences caused by the use of a modified DUT or a different simulation engine.

Given that the safety simulation process may involve hundreds of thousands or even millions of temporal faults, automated regression verification as established by metric-driven verification can both increase the efficiency of identifying the undetected and potentially detected fault simulations and automatically aggregate the safe from unsafe faults. Taken together, these techniques can reduce the effort for safety verification by up to 50%.

Benefits and Features

- Fault injection occurs on un-altered design after elaboration
- Faults can be injected on RTL or gate-level nets coded in Verilog or VHDL
- Stimulus is provided by the un-altered functional testbench in SystemVerilog, *e*, SystemC, or other languages
- Faults can propagate through any design structure that runs in the Incisive Enterprise Simulator, including SystemVerilog, VHDL, SystemC, analog transistor level, analog behavioral level, digital mixed signal models, and more
- Fault dictionary can be generated for the complete design or for a selected hierarchy
- Fault injection time can be randomized using the SystemVerilog engine in the Incisive Enterprise Simulator
- SA0 and SA1 faults force that value at any point in simulation time
- SEU faults modeled as a bit-flip at any point in simulation time
- SET faults modeled as a bit-flip and force at any point in simulation time


Multiple Fault Types for ISO 26262
• Single event upset (SEU)
• Stuck at 0 or 1 (SA0/SA1)
• Single event transient (SET)
Automates Safety Verification
• Efficiently executed fault simulations in modern regression environments
• Highlights potentially detected and undetected faults runs for further debug
Simulates Unaltered DUT
• Fault identification during elaboration
• Faults injected during simulation
• Support Verilog/VHDL for gates/RTL
• Faults can propagate through mixed-signal, low-power, assertions, etc.
Verification Environment Reuse
• Supports SystemVerilog/UVM, SystemC, e
vManager lnx-brent-7:8080 (64) [Analysis - Tests]
File View Analysis Help
Metrics
Tests
vPlan
Scripts Manager
New vPlan
Edit vPlan
Reload vPlan
Context info
Source Map
fault_view*
Views
Refresh
Failures
All Runs
Correlate Runs
Rank Runs
Edit all at once
Edit each
History
Rerun
Create Context
Stop Run
Open dir
Report
Context Operations
Scripts
Planner
Settings
Views
Analyze
Runs
Report
Regression
Analysis
Tracking
Learn More
Test Tree 1
Test Tree 2
Test Tree 3
Test Tree 4
Test Tree 5
dtmf_decoder_fs_...
Tests
Info
Details
Compact All Runs
Messages
Loaded Run: Session

Tests Hierarchy

Name	Overall Average Grade	Overall Covered	Test Status
Test-Case Model	100%	840 / 840 (100%)	100%
default	100%	840 / 840 (100%)	100%
SA0	100%	412 / 412 (100%)	100%
SA1	100%	412 / 412 (100%)	100%
SEU	100%	16 / 16 (100%)	100%

Showing 5 items

Groups of Runs of: Test-Case Model — Pre-Grouping Filter: No filter

In...	Name	Status	Number Of Entit	fault_id	fault_type	fault_node	fault_injec...	fault_status	sim_finish...	fault_classification	fault_dete...	fault_detec...	fault_seed	wallclock_fini...	Owner
MI...	MIXED	passed	199	MIXED	MIXED	MIXED	MIXED	DETECTED	STOPPED	DANGEROUS	MIXED	MIXED	MIXED	NORMAL	brent
MI...	MIXED	passed	30	MIXED	MIXED	MIXED	0ns	POTENTIALLY_DETE...	MIXED	POTENTIALLY_DAN...	MIXED	MIXED	MIXED	NORMAL	brent
MI...	MIXED	passed	611	MIXED	MIXED	MIXED	MIXED	UNDETECTED	ON_TIME	POTENTIALLY_SAFE	n/a	n/a	MIXED	NORMAL	brent

Showing 3 items

Name	Status	faul...	...	fault_node	fault_injec...	fault_status	sim_finis...	fault_classification	fault_detect_po...	fault_dete...	fault_seed	wallclock_finish_time	Owner
/SA0/fault_id_26	passed	26	..	:TOP.DMA_INST.i_69.A	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA0/fault_id_54	passed	54	..	:TOP.DMA_INST.i_65.Y	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA0/fault_id_109	passed	109	..	:TOP.DMA_INST.i_63.A	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA0/fault_id_229	passed	229	..	:TOP.DMA_INST.i_67.A	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA0/fault_id_267	passed	267	..	:TOP.DMA_INST.i_28...	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA0/fault_id_356	passed	356	..	:TOP.DMA_INST.i_68.A	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA0/fault_id_406	passed	406	..	:TOP.DMA_INST.i_66.A	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	2342	NORMAL	brent
/SA1/fault_id_55	passed	55	..	:TOP.DMA_INST.i_28...	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	7055	NORMAL	brent
/SA1/fault_id_57	passed	57	..	:TOP.DMA_INST.i_69.Y	0ns	POTENTIA...	ON_TIME	POTENTIALLY_DANG...	:TOP.DMA_INS...	20ns	7055	NORMAL	brent
/SA1/fault_id_56	passed	56	..	:TOP.DMA_INST.pre...	0ns	POTENTIA...	TIMEOUT	POTENTIALLY_DANG...	:TOP.DMA_INS...	3164620ns	7055	NORMAL	brent

Showing 30 items

Figure 3: Functional verification and safety verification flow

Cadence Design Systems enables global electronic design innovation and plays an essential role in the creation of today's electronics. Customers use Cadence software, hardware, IP, and expertise to design and verify today's mobile, cloud, and connectivity applications. **www.cadence.com**