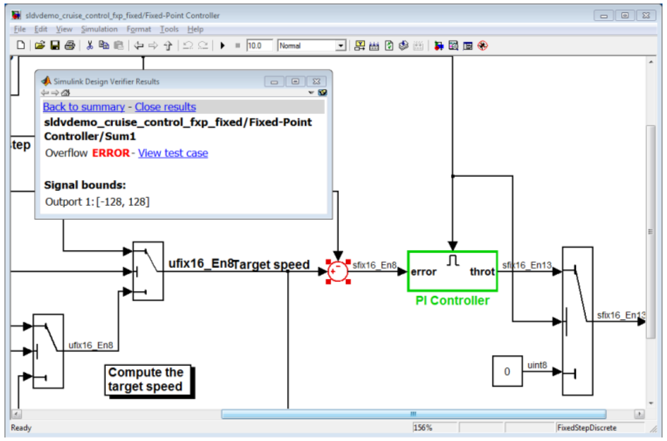
Simulink Design Verifier 2.0

Product Presentation


sldvdemo_cruise_control_fxp_fixed/Fixed-Point Controller
File Edit View Simulation Format Tools Help
Simulink Design Verifier Results
Back to summary - Close results
sldvdemo_cruise_control_fxp_fixed/Fixed-Point Controller/Sum1
Overflow ERROR - View test case
Signal bounds:
Outport 1: [-128, 128]
ufix16_En8
Target speed
sfix16_En8
error
throt
PI Controller
sfix16_En13
Compute the target speed
uint8
Ready
156%
FixedStepDiscrete

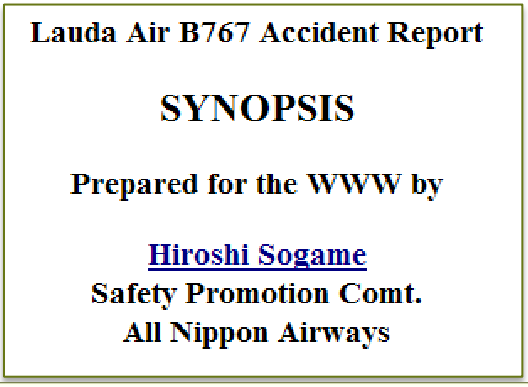
Denizhan Alparslan, Ph.D.

Agenda

- Introduction: Design Verification Challenge
  - Discover unanticipated functionality
- Part 1: Identifying Design Errors Early
- Part 2: Verifying Design Against Requirements
  - Model and validate requirements using models
  - Prove design correctness
- Part 3: Model Coverage Analysis
  - Generate test vectors
  - Measure model coverage
- Part 4: What’s New in Simulink Design Verifier

Design Verification Challenge
Discover Unanticipated Functionality

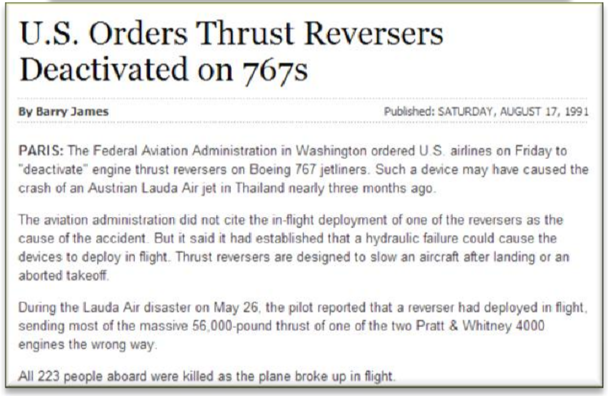
- Test for unanticipated (unwanted) functionality
  - Example: Thrust reversers shall not (*never, by design*) deploy during flight
- Help:
  - Process: Industry standards such as DO-178B, ISO 26262
  - Rigor: Systematic testing (conditions, decisions, MC/DC)
  - Math: Formal methods


Lauda Air B767 Accident Report

SYNOPSIS

Prepared for the WWW by

Hiroshi Sogame
Safety Promotion Comt.
All Nippon Airways


U.S. Orders Thrust Reversers Deactivated on 767s

By Barry James Published: SATURDAY, AUGUST 17, 1991

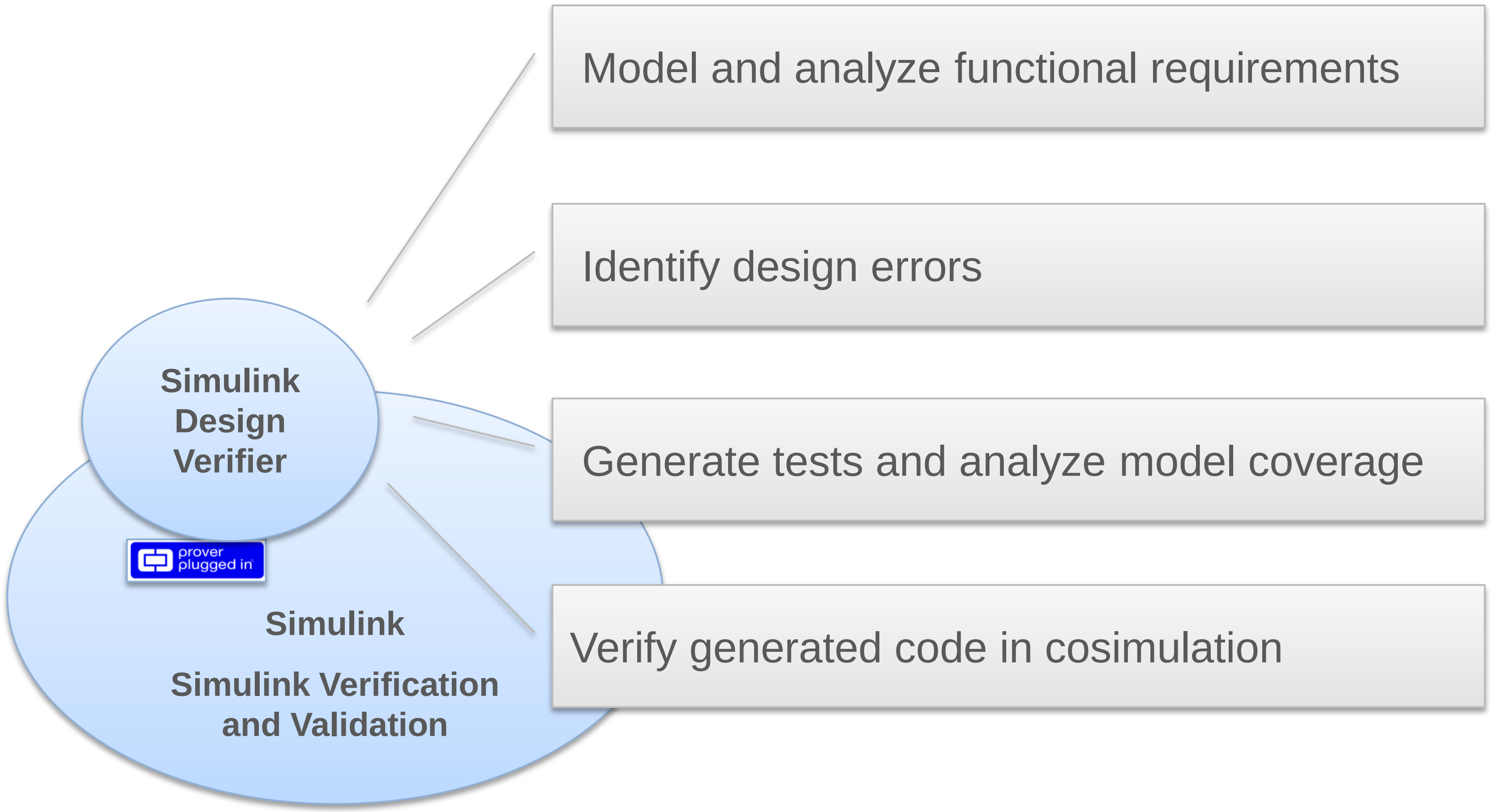
PARIS: The Federal Aviation Administration in Washington ordered U.S. airlines on Friday to "deactivate" engine thrust reversers on Boeing 767 jetliners. Such a device may have caused the crash of an Austrian Lauda Air jet in Thailand nearly three months ago.

The aviation administration did not cite the in-flight deployment of one of the reversers as the cause of the accident. But it said it had established that a hydraulic failure could cause the devices to deploy in flight. Thrust reversers are designed to slow an aircraft after landing or an aborted takeoff.

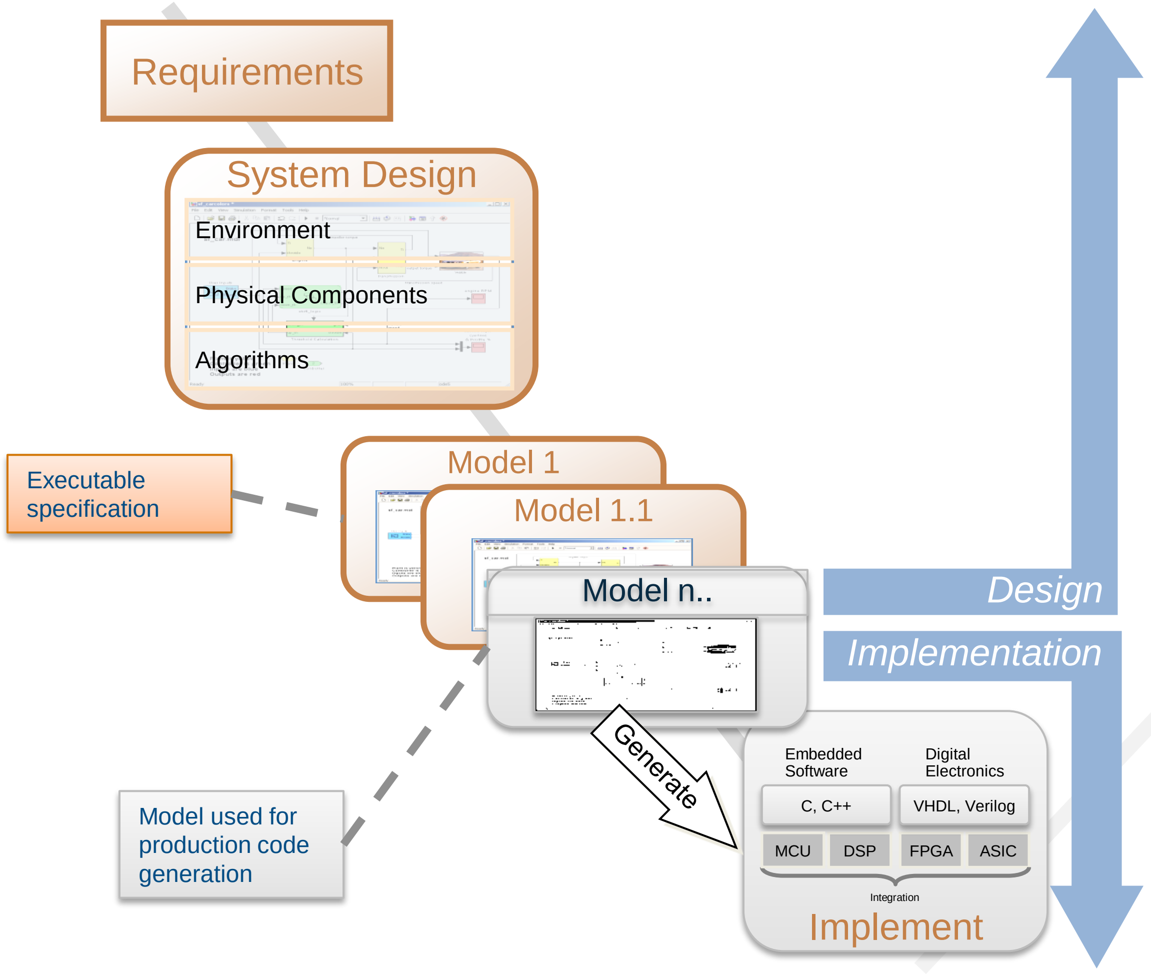
During the Lauda Air disaster on May 26, the pilot reported that a reverser had deployed in flight, sending most of the massive 56,000-pound thrust of one of the two Pratt & Whitney 4000 engines the wrong way.

All 223 people aboard were killed as the plane broke up in flight.

How can Simulink Design Verifier help?


Simulink Design Verifier
prover plugged in
Simulink
Simulink Verification and Validation
Model and analyze functional requirements
Identify design errors
Generate tests and analyze model coverage
Verify generated code in cosimulation

Early Design Verification with Simulink Design Verifier



Early design verification

- Requirements-based analysis
- Design error detection
- Property proving

Later – design done

- Test coverage analysis
- Code verification (SIL/PIL)

TRW Automotive Develops and Tests Electric Parking Brake Using Simulink and Simulink Design Verifier

Challenge

Design tests for an electric parking brake control system

Solution

Use Simulink Design Verifier to automatically generate tests that maximize model coverage and enable systematic design verification

Results

- Test development time reduced from days to hours
- 100% model coverage achieved
- Formal testing begun two months into the project

Electronic parking brake control system.

"Everyone knows that errors are much less expensive to fix when you find them early. With Simulink Design Verifier, we build on the advantages of Model-Based Design by performing formal testing in the first phases of development."

Christoph Hellwig
TRW

Link to user story

Simulink Design Verifier 2.0

Key Features

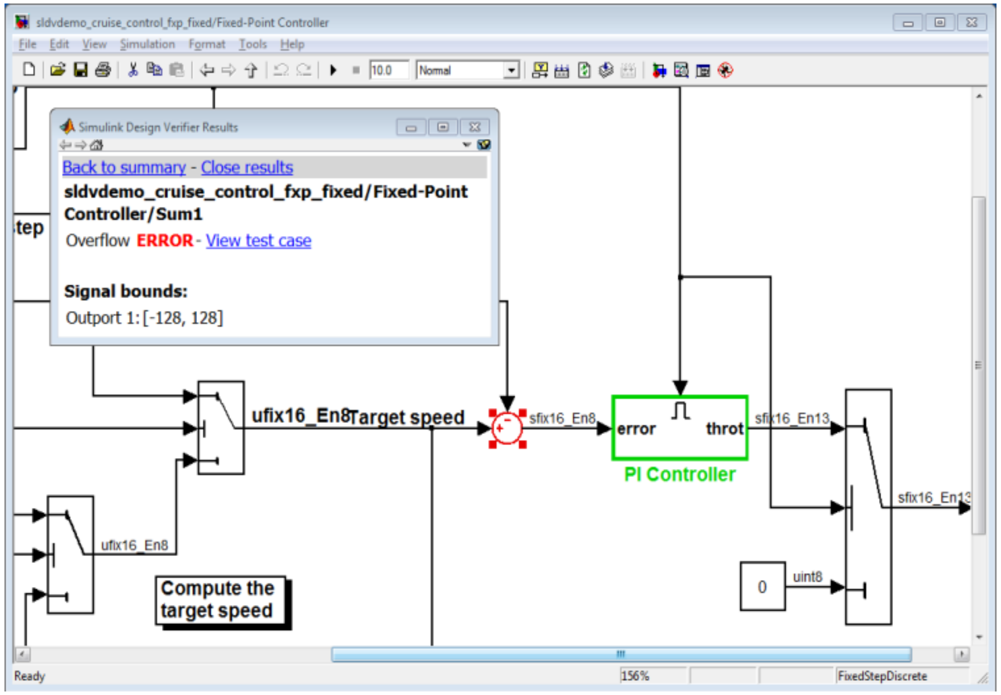
- Polyspace and Prover Plugin formal analysis engines
- Detection of dead logic, integer and fixed-point overflows, division by zero, and violations of design properties
- Blocks and functions for modeling functional and safety requirements
- Test vector generation from functional requirements and model coverage objectives, including condition, decision, modified condition/decision (MC/DC), and signal range
- Property proving, with generation of violation examples for analysis and debugging
- Fixed-point and floating-point model support

Identifying Design Errors Early

Identifying Design Errors Early

Automatic identification of hard-to-find design inconsistencies in the model without running simulation

- Integer overflow
- Division by zero
- Dead logic
- Assertion violation



Back to summary - Close results

sldvdemo_cruise_control_fxp_fixed/Fixed-Point Controller/Sum1

Overflow **ERROR** - View test case

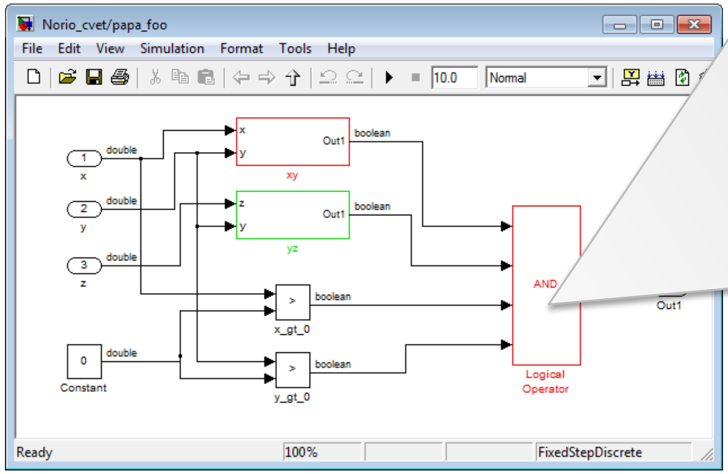
Signal bounds:

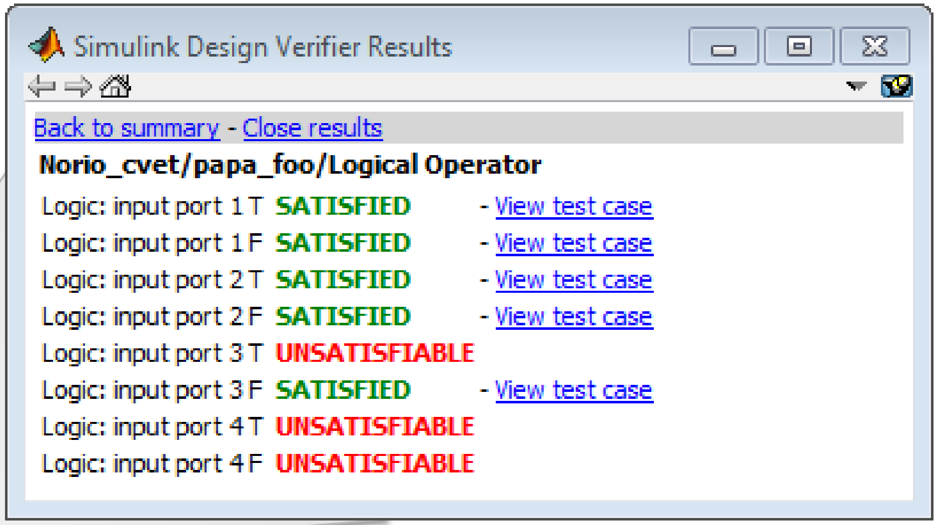
Outport 1: [-128, 128]

Example Design Error Found

Dead Logic

- Certain designed functionality can *never* be activated.
- Typical implications:
  - Design can't meet requirements.
  - Design generates dead code.





Back to summary - Close results

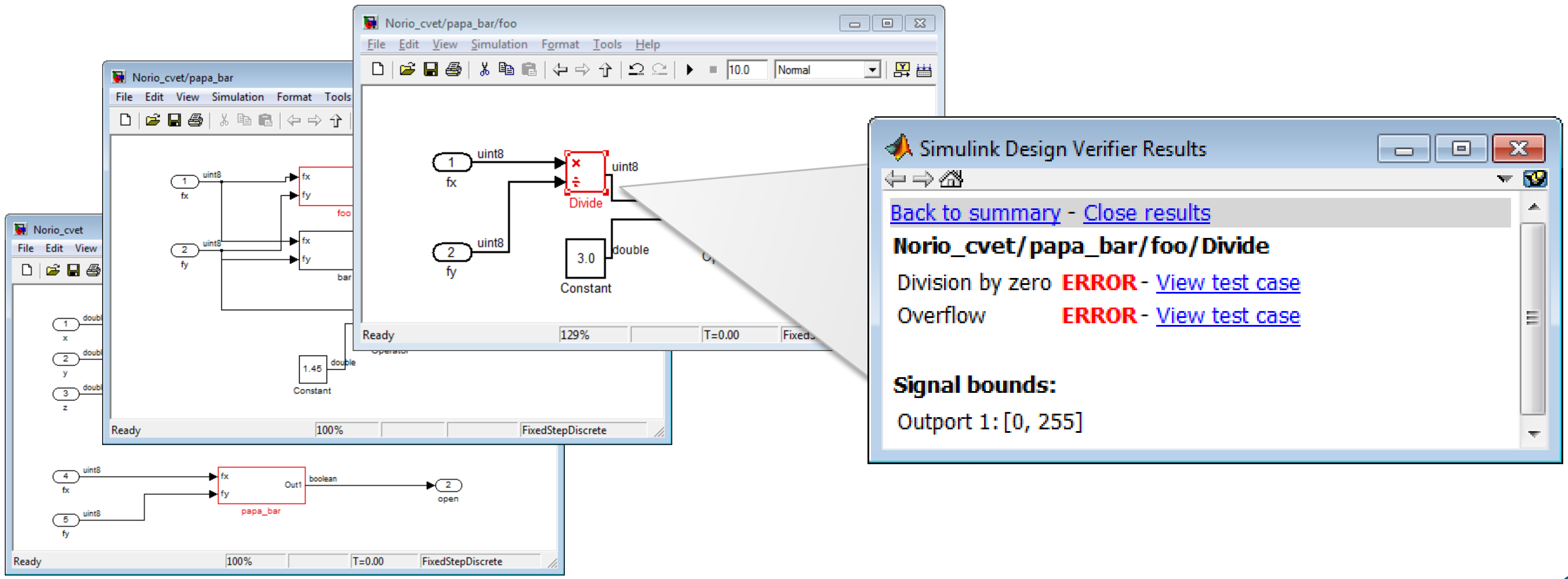
Norio_cvet/papa_foo/Logical Operator

Logic: input port 1 T	SATISFIED	- View test case
Logic: input port 1 F	SATISFIED	- View test case
Logic: input port 2 T	SATISFIED	- View test case
Logic: input port 2 F	SATISFIED	- View test case
Logic: input port 3 T	UNSATISFIABLE	
Logic: input port 3 F	SATISFIED	- View test case
Logic: input port 4 T	UNSATISFIABLE	
Logic: input port 4 F	UNSATISFIABLE	

Example Design Error Found

Division by Zero, Overflow

- Certain valid input data can cause non-deterministic behavior or exceptions.
- Typical implication:
  - Incomplete or incorrect specification



Back to summary - Close results

Norio_cvet/papa_bar/foo/Divide

Division by zero **ERROR** - View test case

Overflow **ERROR** - View test case

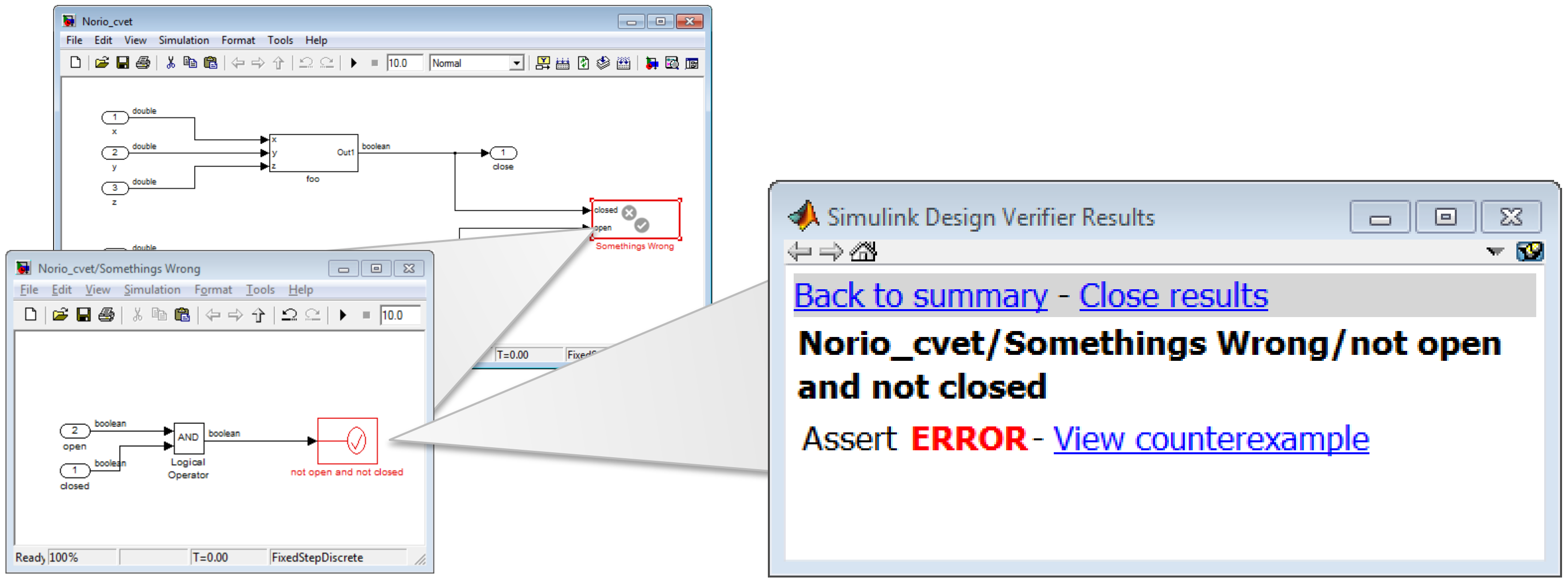
Signal bounds:

Outport 1: [0, 255]

Example Design Error Found

Assertion Violation

- Assertions are blocks you can add to your design to:
  - Detect faulty behavior
  - Monitor design and generated code running in simulation
- Simulink Design Verifier can provide you with the test cases that can trigger assertions



Norio_cvet
File Edit View Simulation Format Tools Help
double
x
y
z
foo
Out1
boolean
close
closed
open
Somethings Wrong
Norio_cvet/Somethings Wrong
open
closed
AND
Logical Operator
not open and not closed
Ready 100% T=0.00 FixedStepDiscrete
Simulink Design Verifier Results

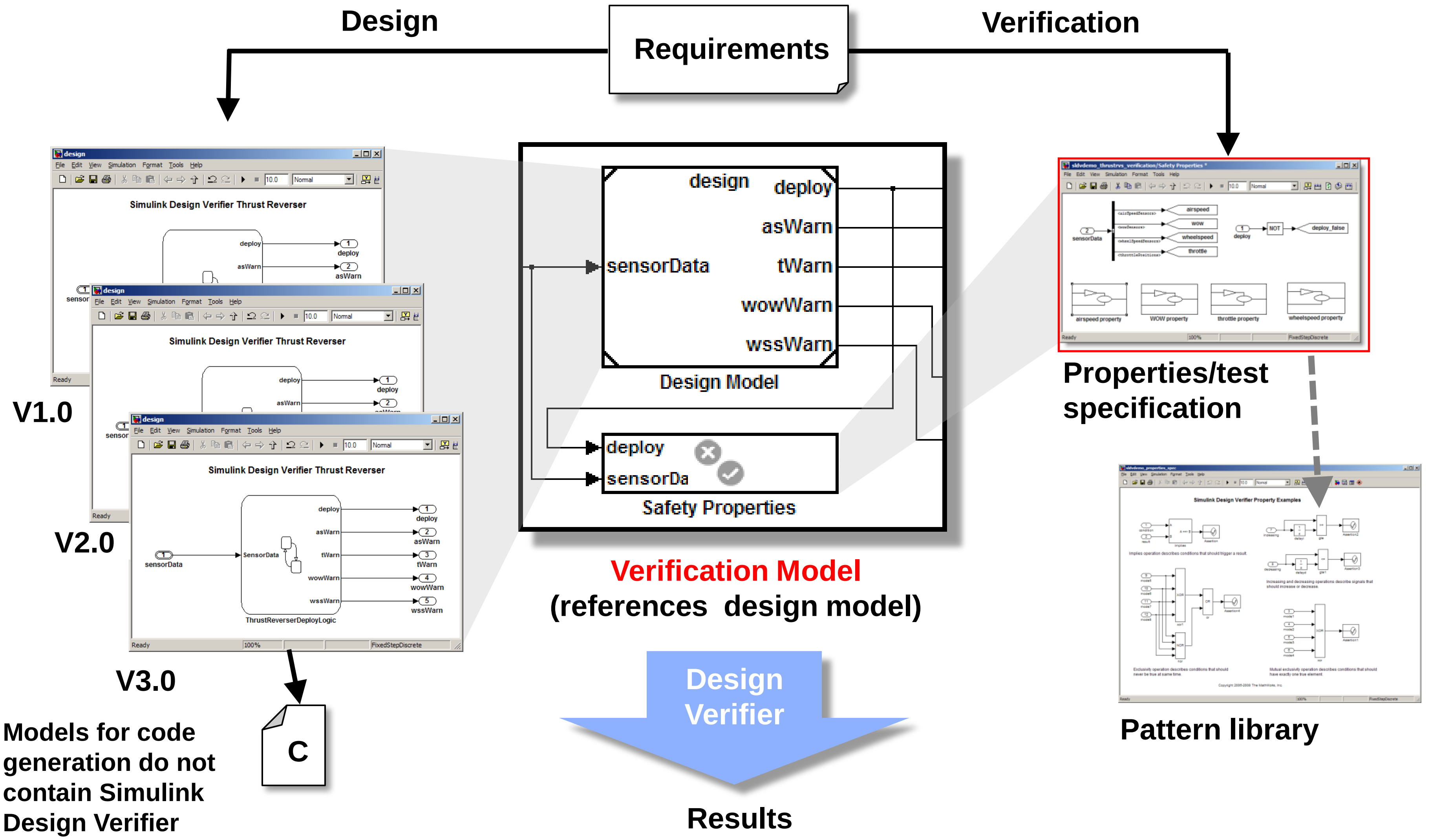
Back to summary - Close results

Norio_cvet/Somethings Wrong/not open and not closed

Assert **ERROR** - View counterexample

Verifying Design Against Requirements

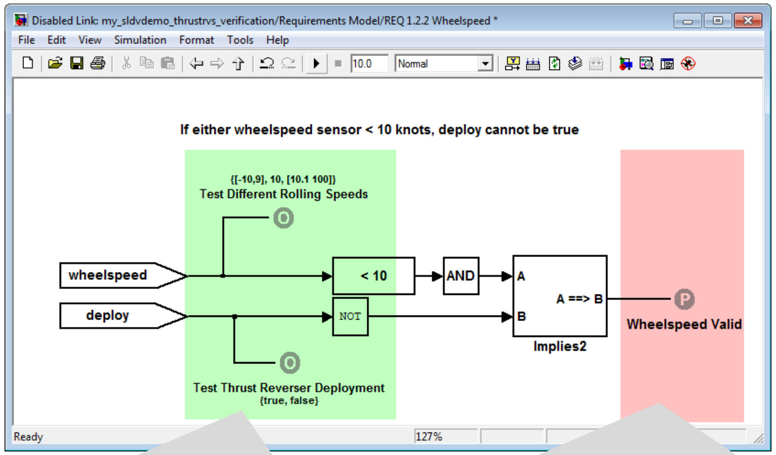
Working with Formal Requirements



Models for code generation do not contain Simulink Design Verifier blocks

Formalizing Requirements Into Properties

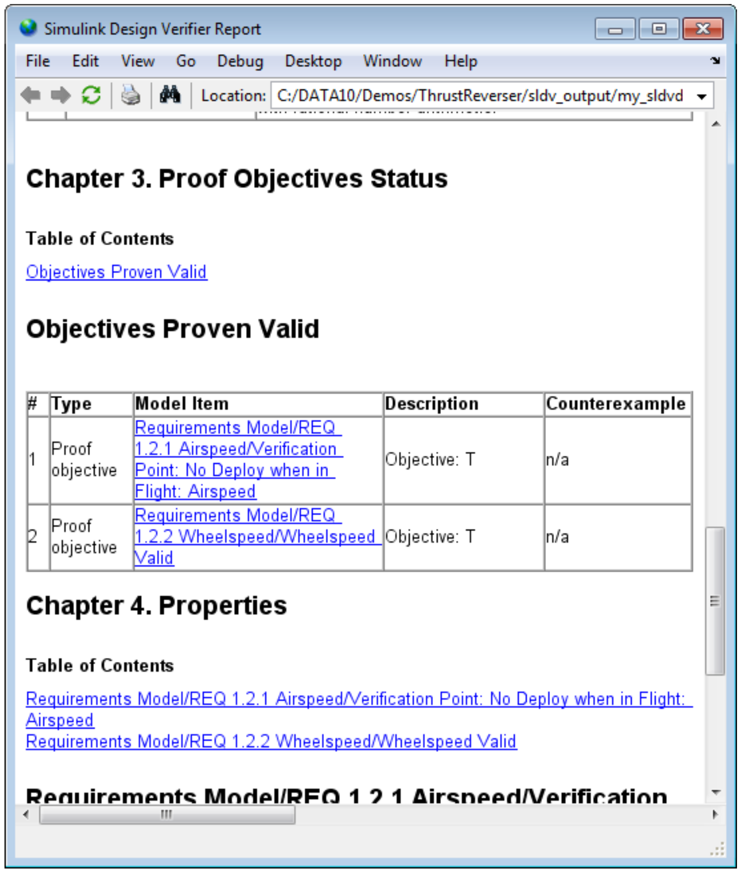
Model Requirements → Prove Design Correctness



Disabled Link: my_sldvdemo_thrustrvs_verification/Requirements Model/REQ 1.2.2 Wheelspeed *
File Edit View Simulation Format Tools Help
If either wheelspeed sensor < 10 knots, deploy cannot be true
{[-10,9], 10, [10.1 100]}
Test Different Rolling Speeds
wheelspeed
deploy
< 10
AND
NOT
A ==> B
A
B
Implies2
Wheelspeed Valid
Test Thrust Reverser Deployment
{true, false}
Ready
127%

Must do

Must never do



Simulink Design Verifier Report
File Edit View Go Debug Desktop Window Help
Location: C:/DATA10/Demos/ThrustReverser/sldv_output/my_sldvd

Chapter 3. Proof Objectives Status

Table of Contents

Objectives Proven Valid

Objectives Proven Valid

#	Type	Model Item	Description	Counterexample
1	Proof objective	Requirements Model/REQ 1.2.1 Airspeed/Verification Point: No Deploy when in Flight: Airspeed	Objective: T	n/a
2	Proof objective	Requirements Model/REQ 1.2.2 Wheelspeed/Wheelspeed Valid	Objective: T	n/a

Chapter 4. Properties

Table of Contents

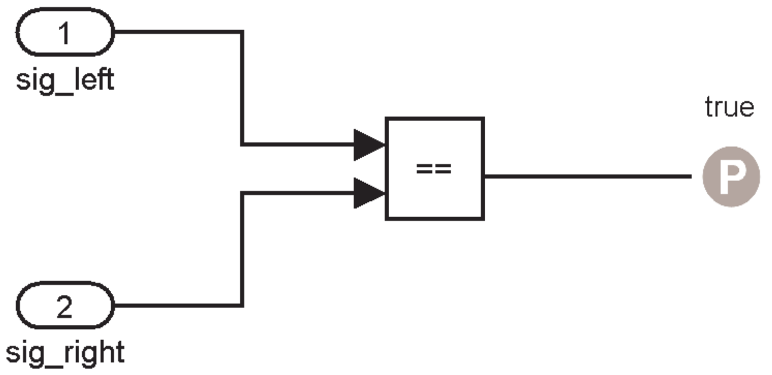
Requirements Model/REQ 1.2.1 Airspeed/Verification Point: No Deploy when in Flight: Airspeed
Requirements Model/REQ 1.2.2 Wheelspeed/Wheelspeed Valid

Requirements Model/REQ 1.2.1 Airspeed/Verification

Examples of Formalized Requirements

Explicit descriptions of required behavior – functional or safety requirement

- Primitive:
  - Objectives, proofs
- Invariant:
  - A > B => C=0
- Temporal:
  - A > B for 10 time steps => C=0 within 5 time steps
- Complex, state-based
  - Stateflow, MATLAB functions



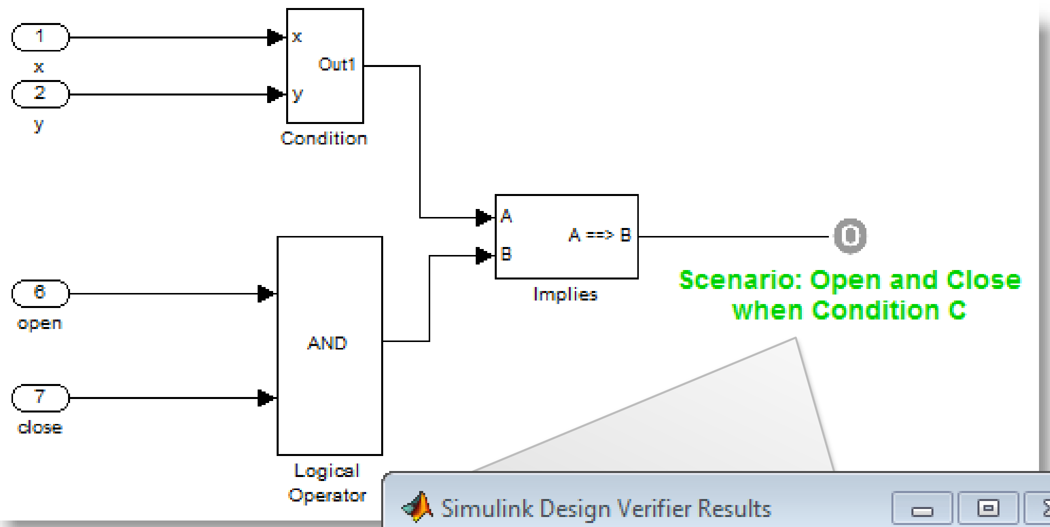
Example invariant:

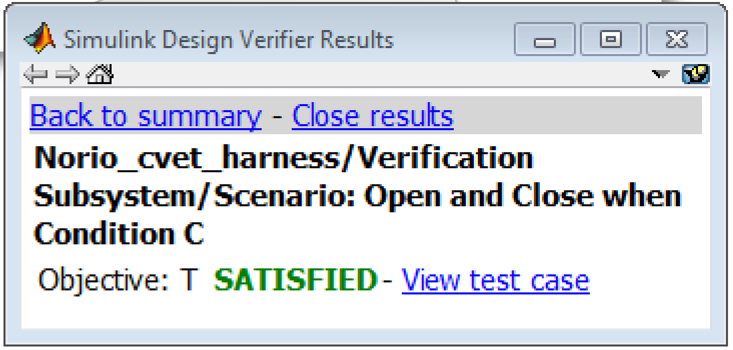
- Always identical, for every time step, including initialization, all modes of operation

Functional Requirements

Must Do (Test Case)

Must demonstrate opening and closing the valve when *[Condition]*

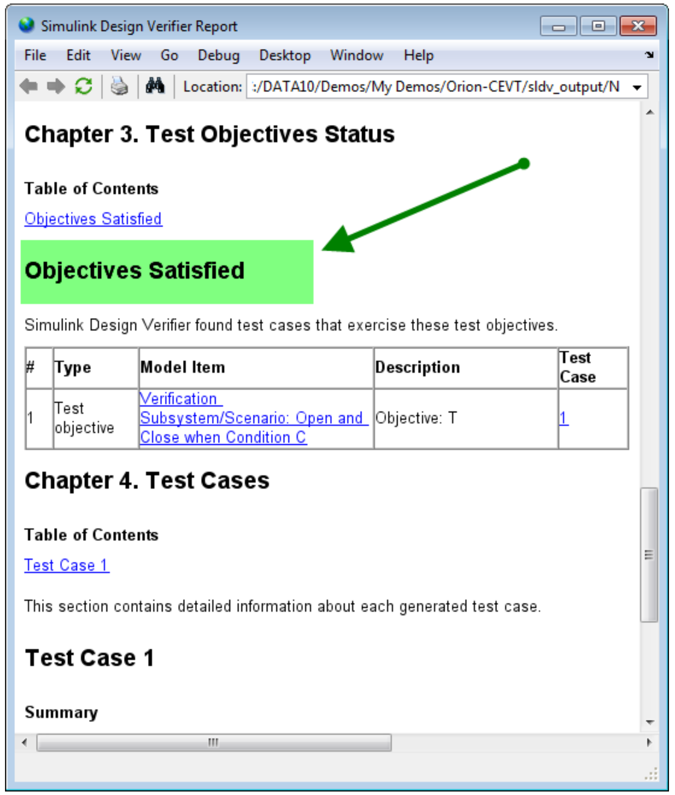




Back to summary - Close results

Norio_cvet_harness/Verification Subsystem/Scenario: Open and Close when Condition C

Objective: T **SATISFIED** - View test case


Simulink Design Verifier Report
File Edit View Go Debug Desktop Window Help
Location: :/DATA10/Demos/My Demos/Orion-CEVT/sldv_output/N

Chapter 3. Test Objectives Status

Table of Contents

Objectives Satisfied

Objectives Satisfied

Simulink Design Verifier found test cases that exercise these test objectives.

#	Type	Model Item	Description	Test Case
1	Test objective	Verification Subsystem/Scenario: Open and Close when Condition C	Objective: T	1

Chapter 4. Test Cases

Table of Contents

Test Case 1

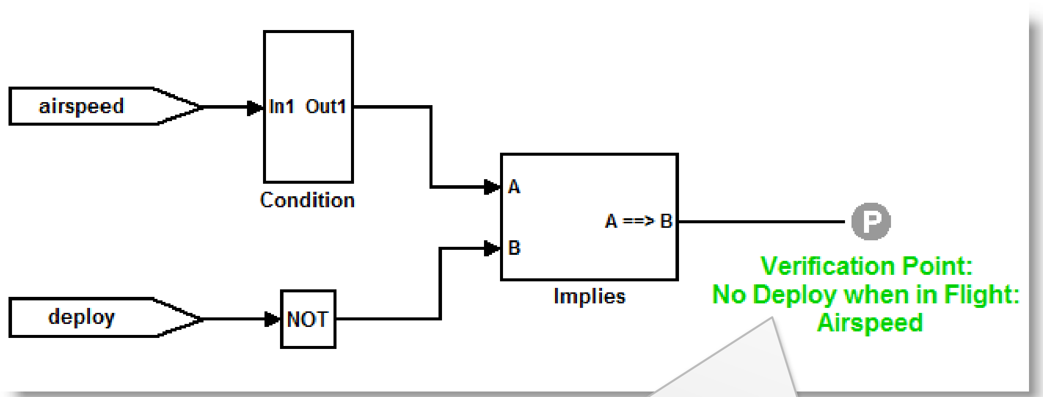
This section contains detailed information about each generated test case.

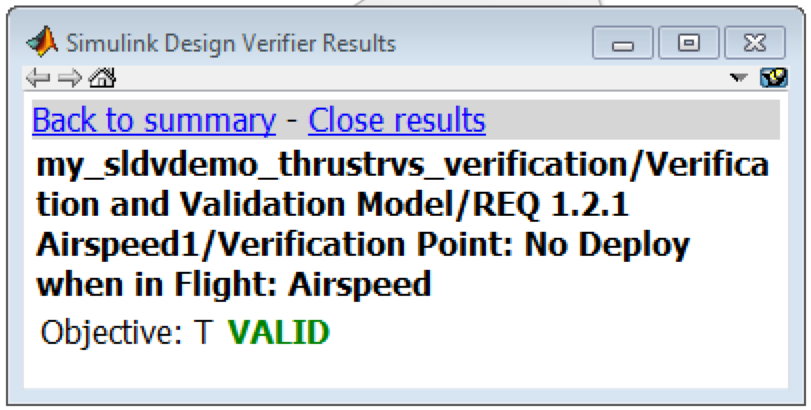
Test Case 1

Summary

Safety Requirements
Must Never Do (Proof)

Thrust reverser shall not deploy when *[Condition]*

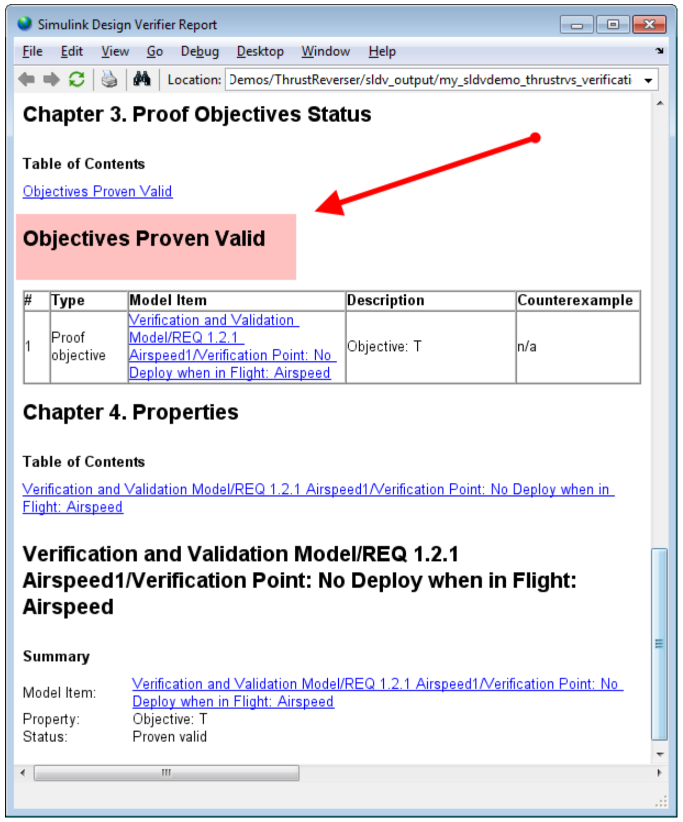




Back to summary - Close results

my_sldvdemo_thrustrvs_verification/Verification and Validation Model/REQ 1.2.1 Airspeed1/Verification Point: No Deploy when in Flight: Airspeed

Objective: T **VALID**



Simulink Design Verifier Report
File Edit View Go Debug Desktop Window Help
Location: Demos/ThrustReverser/sldv_output/my_sldvdemo_thrustrvs_verificati

Chapter 3. Proof Objectives Status

Table of Contents

Objectives Proven Valid

Objectives Proven Valid

#	Type	Model Item	Description	Counterexample
1	Proof objective	Verification and Validation Model/REQ 1.2.1 Airspeed1/Verification Point: No Deploy when in Flight: Airspeed	Objective: T	n/a

Chapter 4. Properties

Table of Contents

Verification and Validation Model/REQ 1.2.1 Airspeed1/Verification Point: No Deploy when in Flight: Airspeed

Verification and Validation Model/REQ 1.2.1 Airspeed1/Verification Point: No Deploy when in Flight: Airspeed

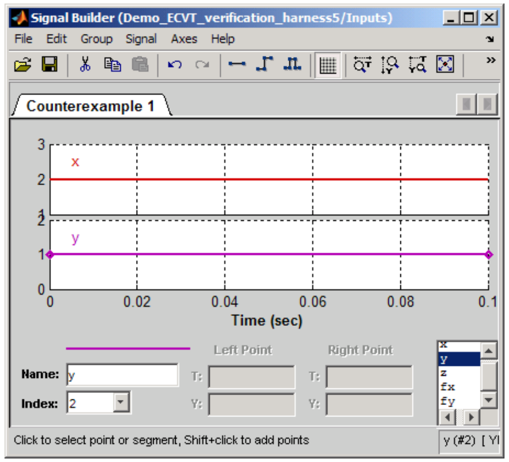
Summary

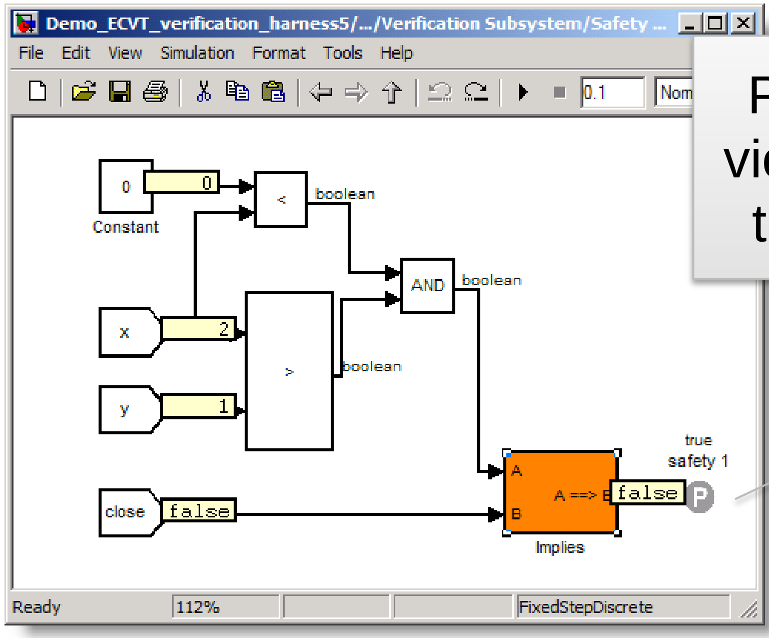
Model Item:	Verification and Validation Model/REQ 1.2.1 Airspeed1/Verification Point: No Deploy when in Flight: Airspeed
Property:	Objective: T
Status:	Proven valid

Validation of Formal Verification Results

Simulation / Debugging

Requirement models (properties) cosimulate with the design. Simulation driven by counterexamples.


Signal Builder (Demo_ECVT_verification_harness5/Inputs)
File Edit Group Signal Axes Help
Counterexample 1
x
y
Time (sec)
Left Point
Right Point
Name: y
Index: 2
Click to select point or segment, Shift+click to add points

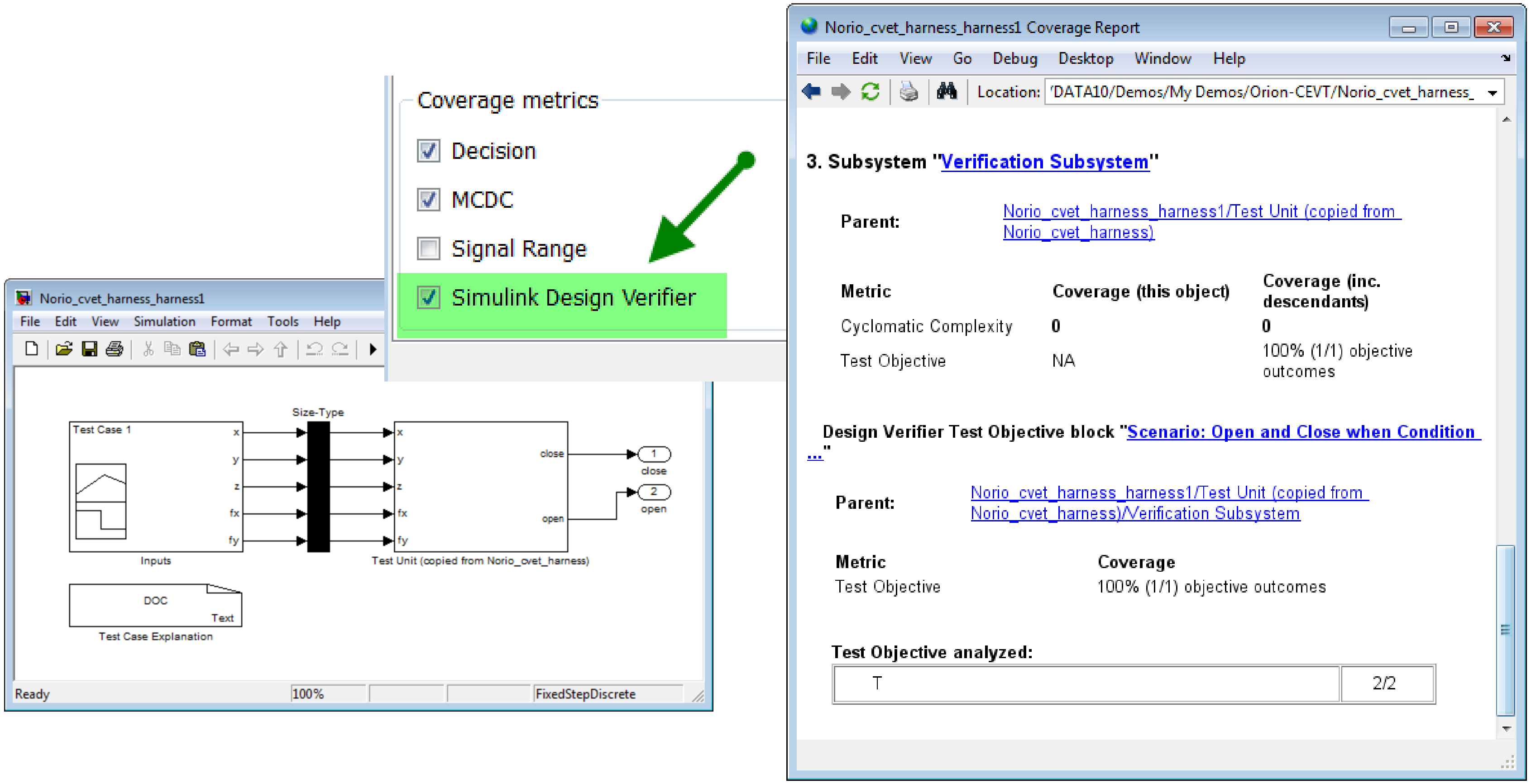

Demo_ECVT_verification_harness5/.../Verification Subsystem/Safety ...
File Edit View Simulation Format Tools Help
Constant
boolean
AND
x
y
close
false
Implies
true
safety 1
false
Ready
112%
FixedStepDiscrete

Proof objective violation stopped the simulation.

Validation of Property Proving Results

Simulation / Model Coverage

Model coverage of Simulink Design Verifier objectives


Norio_cvet_harness_harness1
File Edit View Simulation Format Tools Help
Test Case 1
x
y
z
fx
fy
Inputs
Size-Type
Test Unit (copied from Norio_cvet_harness)
close
open
1
2
DOC
Text
Test Case Explanation
Ready
100%
FixedStepDiscrete
Norio_cvet_harness_harness1 Coverage Report
File Edit View Go Debug Desktop Window Help
Location: 'DATA10/Demos/My Demos/Orion-CEVT/Norio_cvet_harness_

Coverage metrics

- [✓] Decision
- [✓] MCDC
- [ ] Signal Range
- [✓] Simulink Design Verifier

3. Subsystem "Verification Subsystem"

Parent: Norio_cvet_harness_harness1/Test Unit (copied from Norio_cvet_harness)

Metric	**Coverage (this object)**	**Coverage (inc. descendants)**
Cyclomatic Complexity	0	0
Test Objective	NA	100% (1/1) objective outcomes

Design Verifier Test Objective block "Scenario: Open and Close when Condition ..."

Parent: Norio_cvet_harness_harness1/Test Unit (copied from Norio_cvet_harness)/Verification Subsystem

Metric	**Coverage**
Test Objective	100% (1/1) objective outcomes

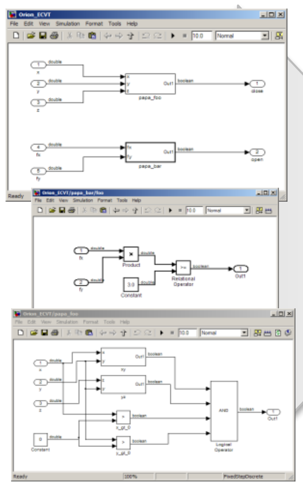
Test Objective analyzed:

T	2/2

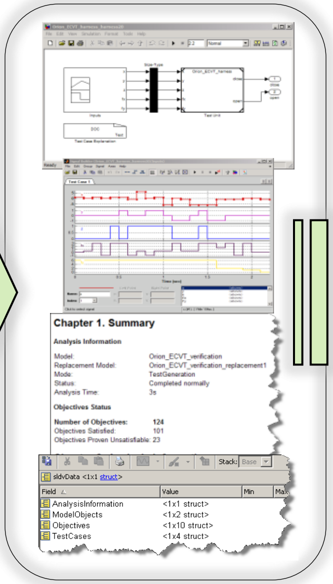
Model Coverage Analysis

Model Coverage Analysis

Generate Tests → Measure Coverage in Simulation



Formal Methods



Chapter 1. Summary

Analysis Information

Model:	Orion_ECVT_verification
Replacement Model:	Orion_ECVT_verification_replacement1
Mode:	TestGeneration
Status:	Completed normally
Analysis Time:	3s

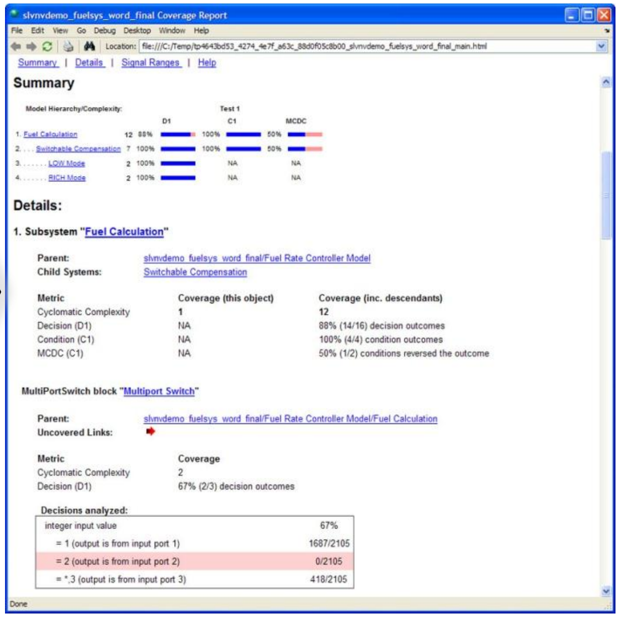
Objectives Status

Number of Objectives:	**124**
Objectives Satisfied:	101
Objectives Proven Unsatisfiable:	23

sldvData <1x1 struct>

Field	Value	Min	Max
AnalysisInformation	<1x1 struct>		
ModelObjects	<1x2 struct>		
Objectives	<1x10 struct>		
TestCases	<1x4 struct>		

Simulation



slvnvdemo_fuelsys_word_final Coverage Report

Location: file:///C:/Temp/tp4643bd53_4274_4e7f_a63c_88d0f05c8b00_slvnvdemo_fuelsys_word_final_main.html

Summary | Details | Signal Ranges | Help

Summary

Model Hierarchy/Complexity:		Test 1 D1		C1	MCDC
1. Fuel Calculation	12	88%		100%	50%
2. . . . Switchable Compensation	7	100%		100%	50%
3. . . . . . . LOW Mode	2	100%		NA	NA
4. . . . . . . RICH Mode	2	100%		NA	NA

Details:

1. Subsystem "Fuel Calculation"

Parent: slvnvdemo_fuelsys_word_final/Fuel Rate Controller Model
Child Systems: Switchable Compensation

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	**1**	**12**
Decision (D1)	NA	88% (14/16) decision outcomes
Condition (C1)	NA	100% (4/4) condition outcomes
MCDC (C1)	NA	50% (1/2) conditions reversed the outcome

MultiPortSwitch block "Multiport Switch"

Parent: slvnvdemo_fuelsys_word_final/Fuel Rate Controller Model/Fuel Calculation
Uncovered Links:

Metric	Coverage
Cyclomatic Complexity	2
Decision (D1)	67% (2/3) decision outcomes

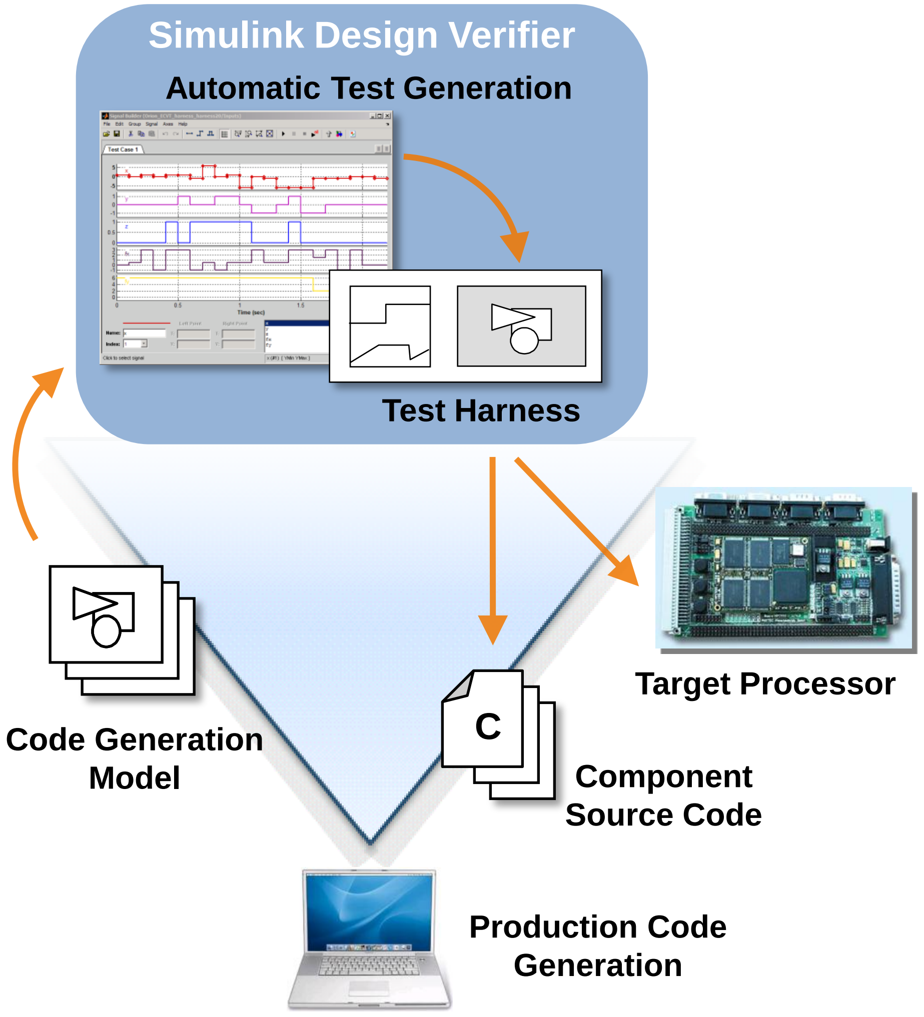
Decisions analyzed:

integer input value	67%
= 1 (output is from input port 1)	1687/2105
= 2 (output is from input port 2)	0/2105
= *,3 (output is from input port 3)	418/2105

Done

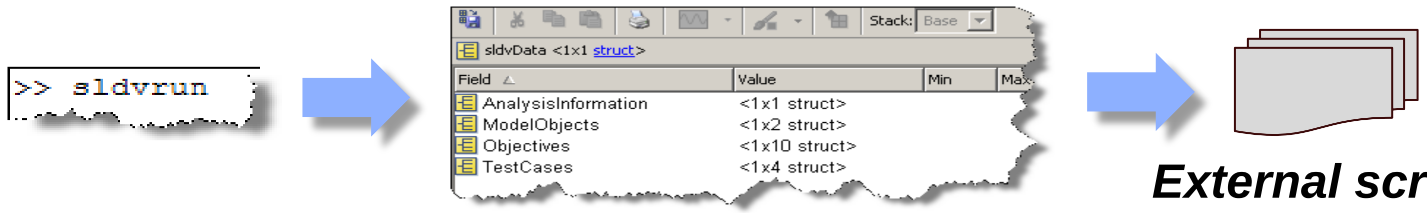
Verify Generated Code

1. Run requirements-based tests
2. Generate test vectors for missing model coverage objectives
3. Review generated test vectors
4. Update requirements-based tests
5. Execute tests on the generated code in SIL and PIL
6. Compare results



Using Test Vectors for Code Verification

A. Exporting test data into code testing tools



```
>> sldvrun
```

sldvData <1x1 struct>

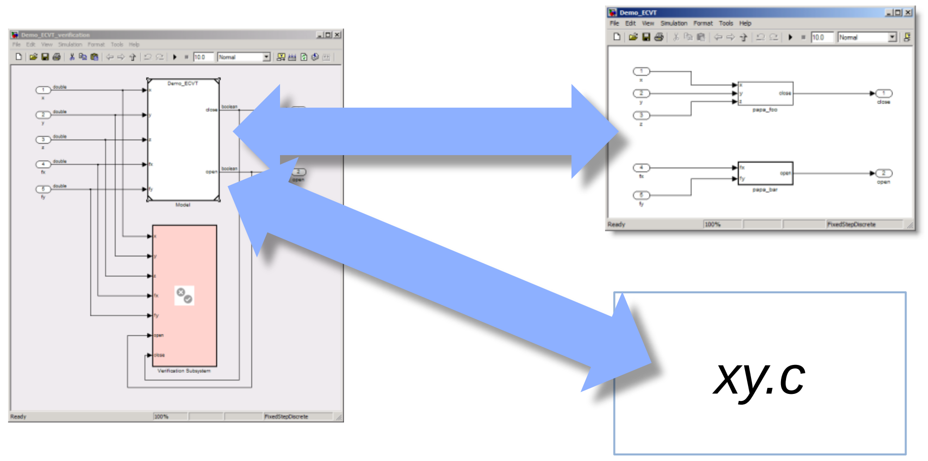
Field	Value	Min	Max
AnalysisInformation	<1x1 struct>		
ModelObjects	<1x2 struct>		
Objectives	<1x10 struct>		
TestCases	<1x4 struct>		

`sldvData.mat`

External script/ testing tool

B. Cosimulating via S-function wrappers

Verification model



Model algorithm

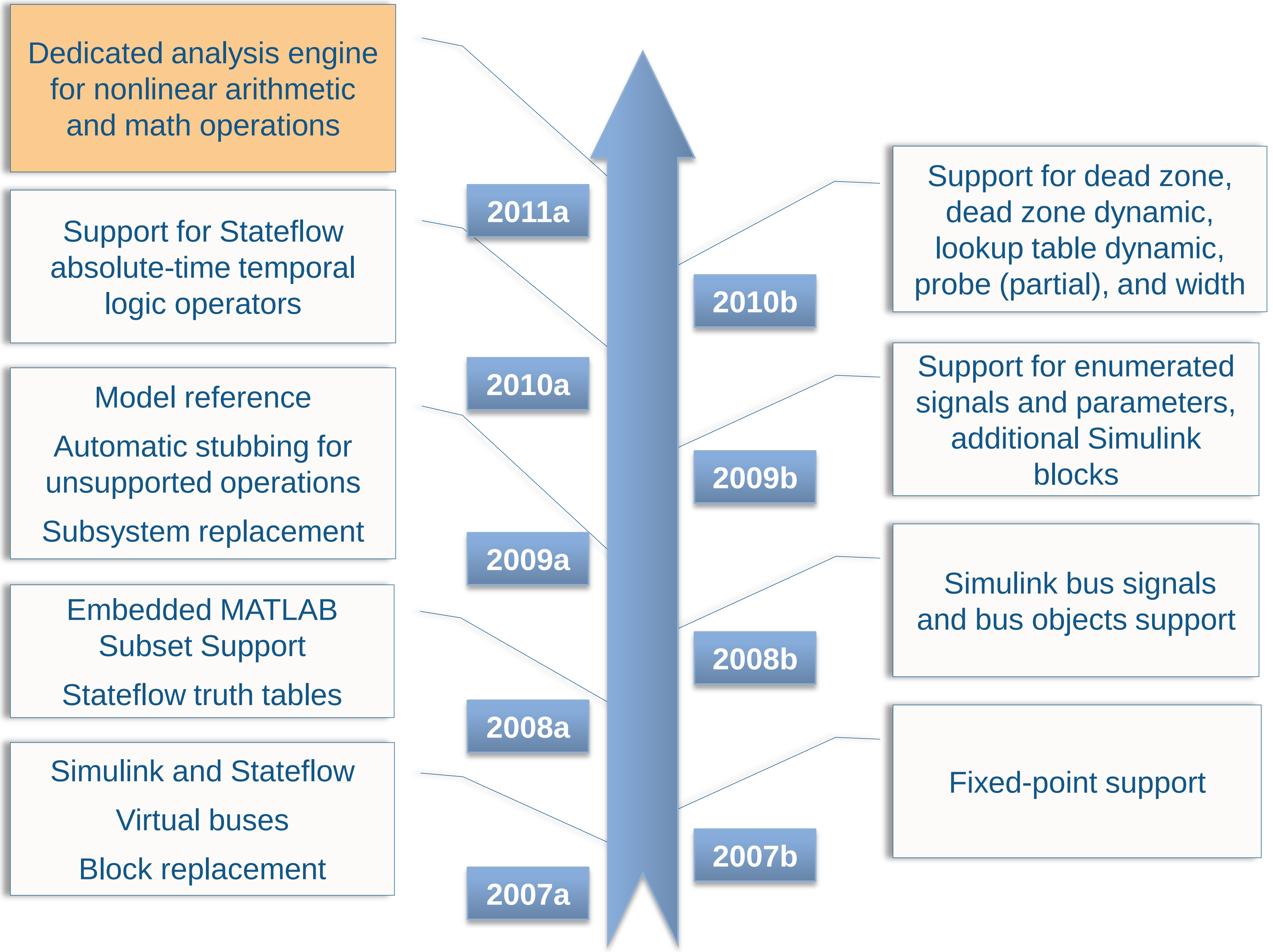
Code algorithm

Applying Formal Methods
Technology Limitations

- *Simulink Design Verifier supports discrete time systems only*
- Use of non-linear functions and long timers may require abstraction, time scaling, or other additional strategies.
  - In some cases, theorem prover requires additional information to solve the problem.
- Proof of property provides levels of confidence beyond test generation and violation detection, but it is also harder to achieve. Complete proof may require optimization of the design for the purpose of verification.

What’s New in Simulink Design Verifier

Extending Simulink Block Support



2011a

Dedicated analysis engine for nonlinear arithmetic and math operations

2010b

Support for dead zone, dead zone dynamic, lookup table dynamic, probe (partial), and width

2010a

Support for Stateflow absolute-time temporal logic operators

2009b

Support for enumerated signals and parameters, additional Simulink blocks

2009a

Model reference

Automatic stubbing for unsupported operations

Subsystem replacement

2008b

Simulink bus signals and bus objects support

2008a

Embedded MATLAB Subset Support

Stateflow truth tables

2007b

Fixed-point support

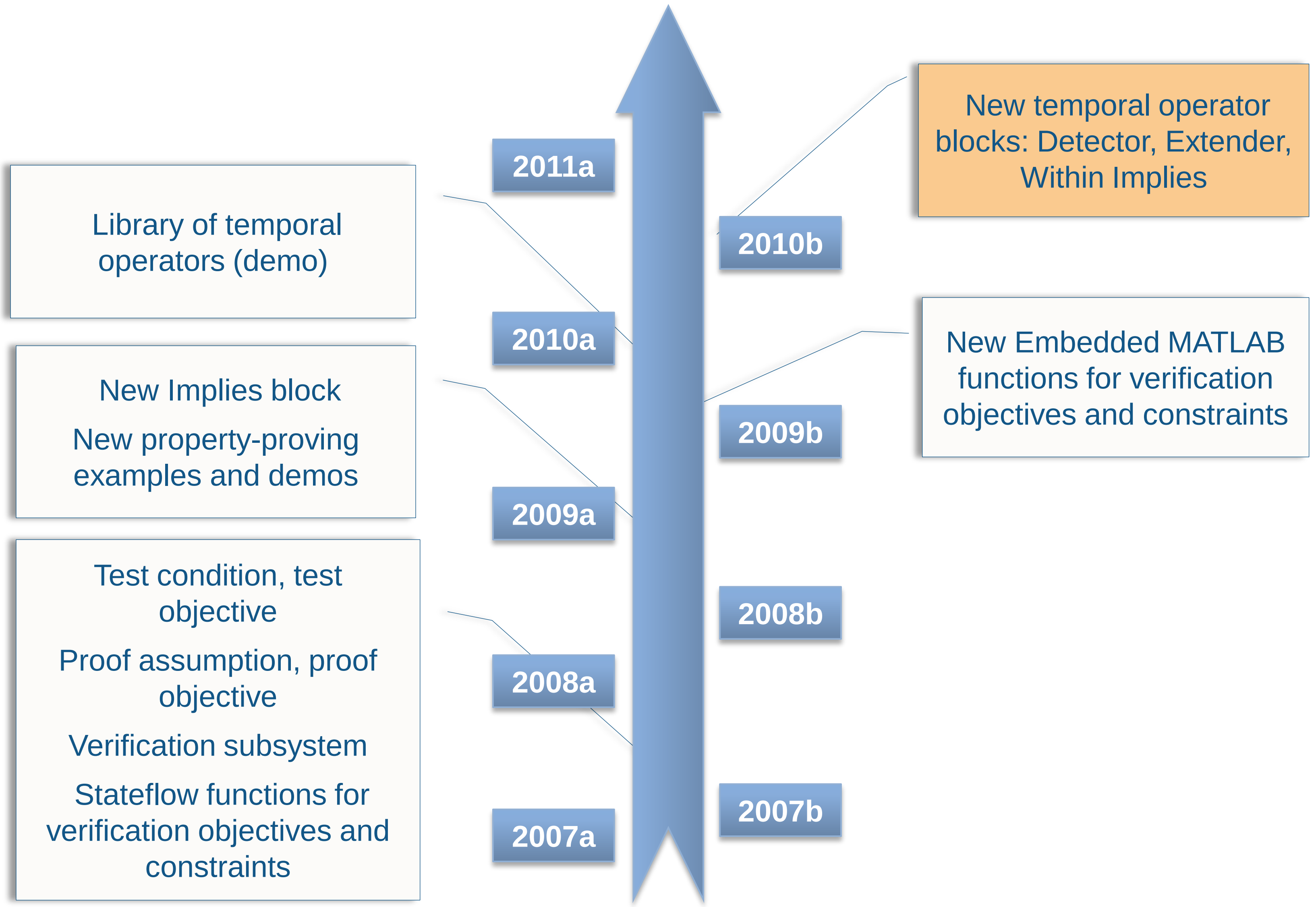
2007a

Simulink and Stateflow

Virtual buses

Block replacement

Making Definition of Verification Objectives Easier



New temporal operator blocks: Detector, Extender, Within Implies

Library of temporal operators (demo)

New Embedded MATLAB functions for verification objectives and constraints

New Implies block

New property-proving examples and demos

Test condition, test objective

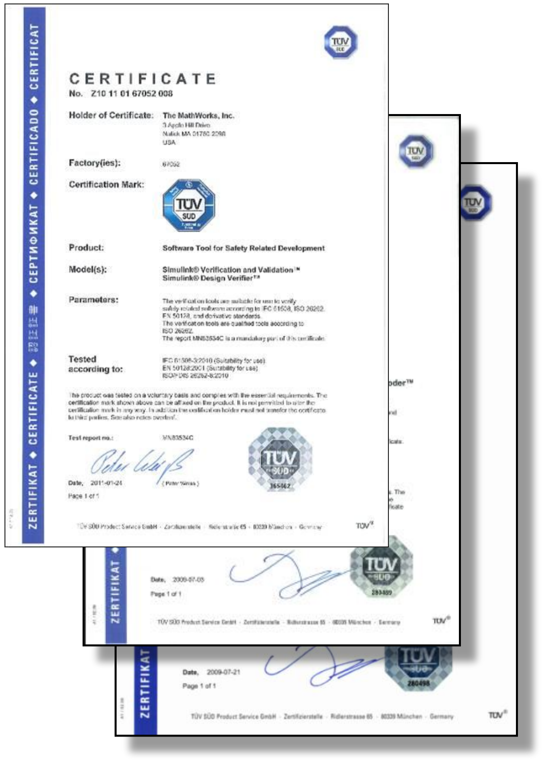
Proof assumption, proof objective

Verification subsystem

Stateflow functions for verification objectives and constraints

TÜV Certification of Simulink Design Verifier

- TÜV SÜD certified:
  - Embedded Coder
  - **Simulink Design Verifier**
  - **Simulink Verification and Validation**
  - Polyspace products for C/C++
- For use in development processes which need to 61508, ISO 26262, or EN 50128


ZERTIFIKAT ◆ CERTIFICATE ◆ 認証証明書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

CERTIFICATE
No. Z10 11 01 67052 008

Holder of Certificate:	The MathWorks, Inc. 3 Apple Hill Drive Natick MA 01760-2098 USA
Factory(ies):	67052
Certification Mark:	
Product:	Software Tool for Safety Related Development
Model(s):	Simulink® Verification and Validation™ Simulink® Design Verifier™
Parameters:	The verification tools are suitable for use to verify safety related software according to IEC 61508, ISO 26262, EN 50128, and derivative standards. The verification tools are qualified tools according to ISO 26262. The report MN83534C is a mandatory part of this certificate.
Tested according to:	IEC 61508-3:2010 (Suitability for use) EN 50128:2001 (Suitability for use) ISO/FDIS 26262-8:2010

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: MN83534C

Date, 2011-01-24 (Peter Weiss)

Page 1 of 1

365462

TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstraße 65 · 80339 München · Germany

Date, 2009-07-03

Page 1 of 1

Date, 2009-07-21

Page 1 of 1

TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstraße 65 · 80339 München · Germany

Note: The products listed above were not developed using certified processes.

MathWorks announcements:

www.mathworks.com/company/pressroom/articles/article17790.html (Initial certification)

www.mathworks.com/company/pressroom/articles/article39270.html (Recertification, ISO 26262 support)

TÜV SÜD certificate database:

http://193.30.192.53:8080/CertDetail_eng.aspx?CertNo=Z10%2009%2006%2067052%20002&CertTyp=no

http://193.30.192.53:8080/CertDetail_eng.aspx?CertNo=Z10 09 07 67052 003&CertTyp=no

http://193.30.192.53:8080/CertDetail_eng.aspx?CertNo=Z10%2011%2001%2067052%20008&CertTyp=no

Conclusion

- Simulink Design Verifier can automatically discover the following types of design errors:
  - Division by zero
  - Integer overflow
  - Dead logic
  - Assertion violations

- Definition of functional test objectives and design properties using the supplied operator blocks enable formal requirements modeling and verification