



# FBIS Development

Sven Stefan Krauss, Christian Sommer, Jan Brunner  
ZHAW Zurich University of Applied Sciences, Switzerland

# FBIS Development Introduction

## Introduction

### Name

- Sven Stefan Krauss
- Computer Engineering
- Certified Functional Safety Engineer (TÜV Rheinland)

### Role

- Project Manager HIL-Systems
- Technical Lead HIL-Systems
- QA



Many thanks to Annika for taking this picture ☺  
during ESS construction site visit for PSS in  
November 2016

# Agenda

FBIS  
Development

V&V

HiL Test  
System

Software  
Framework

Case  
Study

## FBIS Development

### Mission

- Development of FBIS is guided by IEC 61508 ed. 2

### Strategy

- FBIS is not considered as safety-related system, but a protection system
- Adaption of safety to protection
- Main focus on reliability and availability, i.e. hardware metrics and architecture constraints

### Protection System

- PIL – Protection Integrity Level
- PF – Protection Function

### Hardware Metrics

- PFH – Probability of Failure per Hour
- HFT – Hardware Fault Tolerance
- SSF – Safe Failure Fraction

## FBIS Development

### FBIS

- Is part of the Machine Protection System (MPS)
- Implements or is part of Protection Functions

### E/E/PE System

- Protection related logic will be realized in FPGA firmware → programmable electronic system

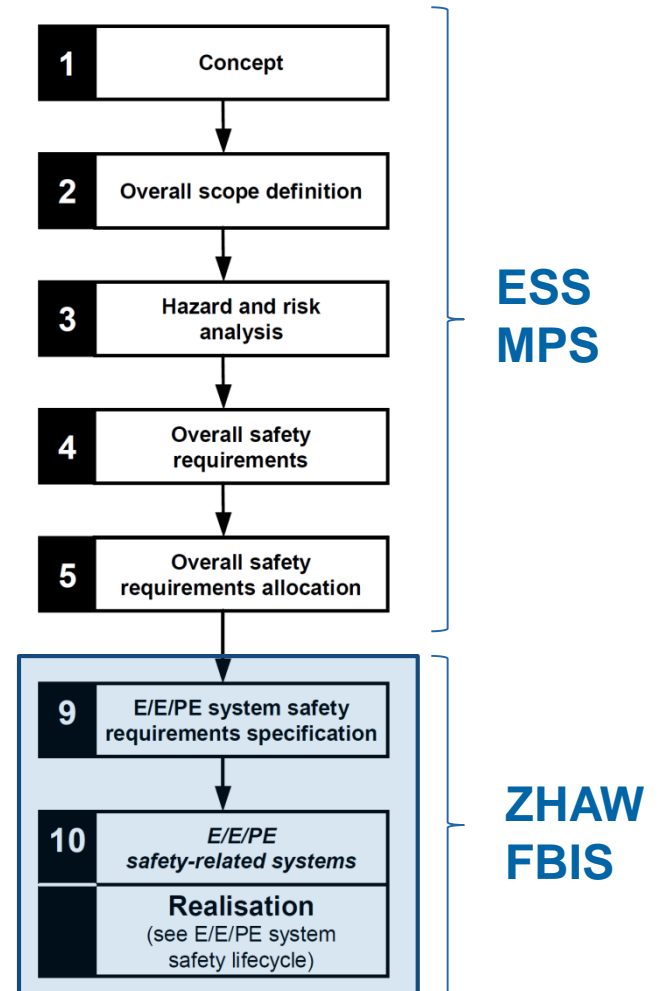
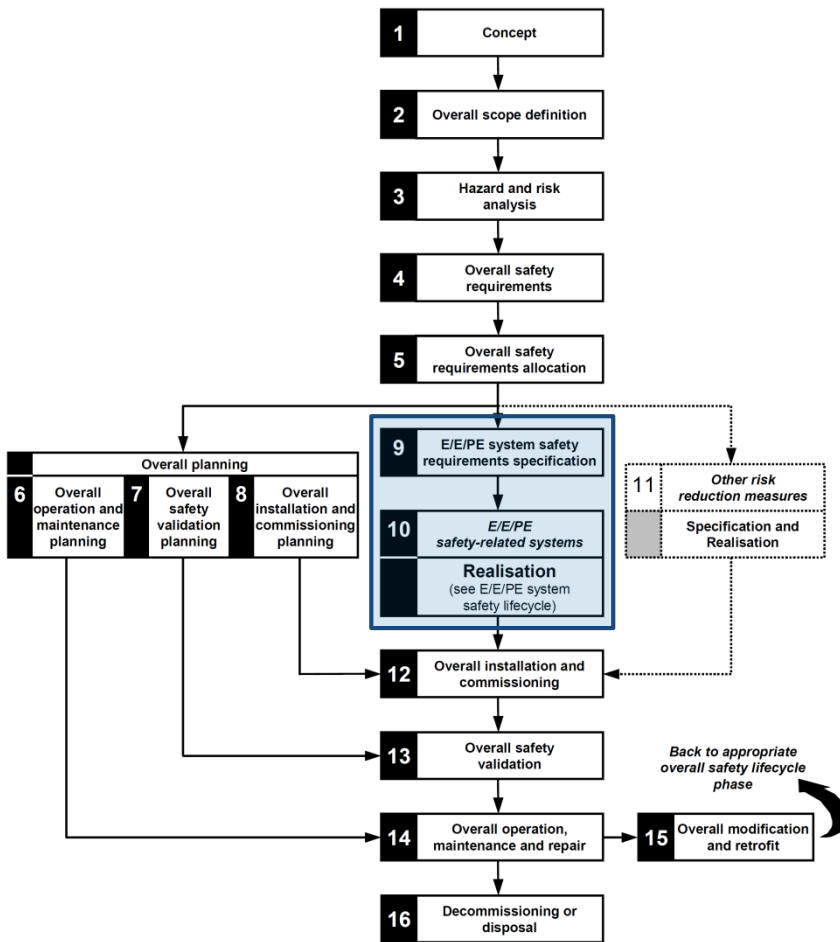
### Standard

- IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems
- Part 1: General Requirements
- Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems

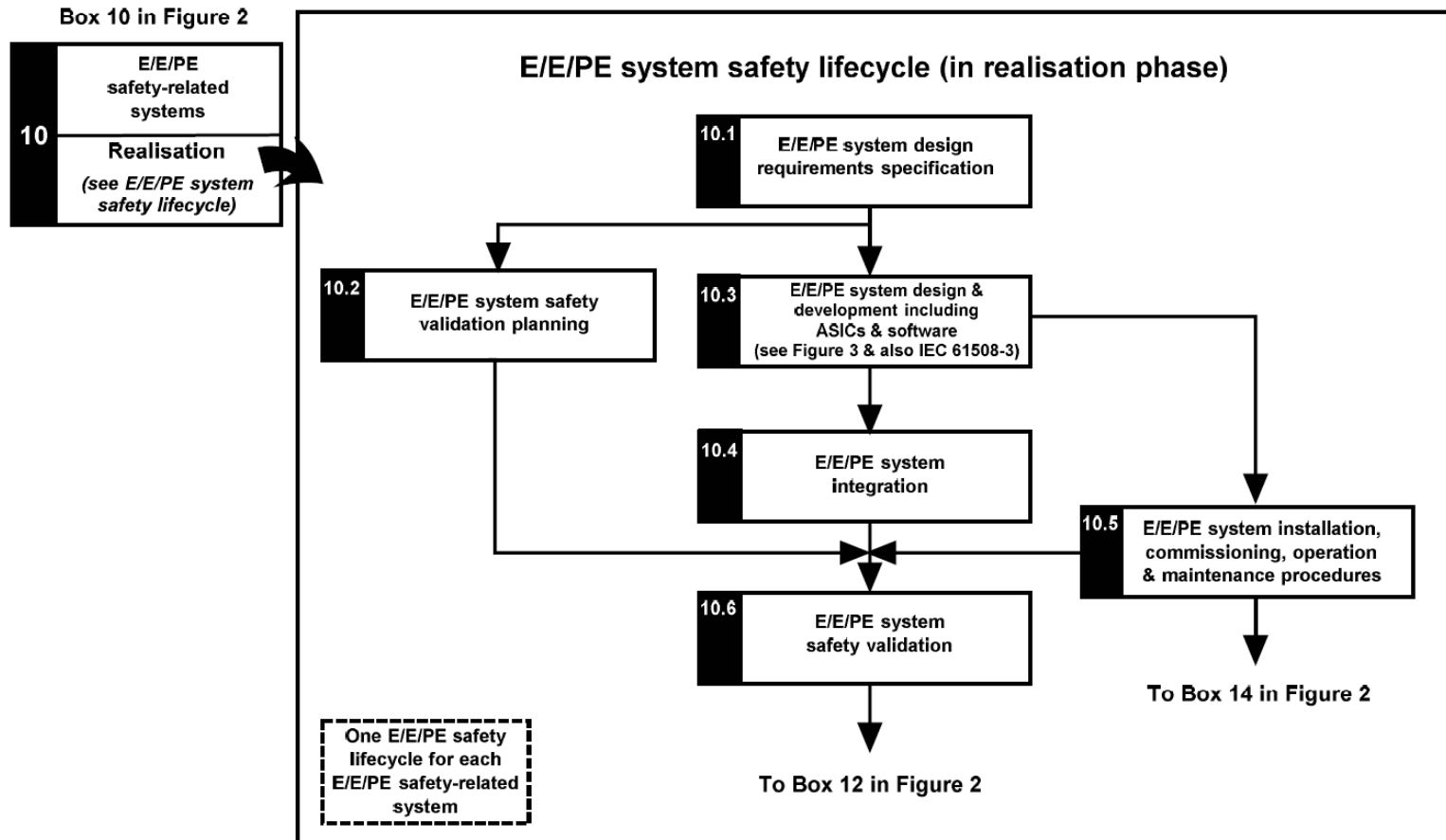
### Compliance Planning

- Tailoring of IEC 61508-1 and IEC 61508-2
- Compliance Matrix (work in progress)

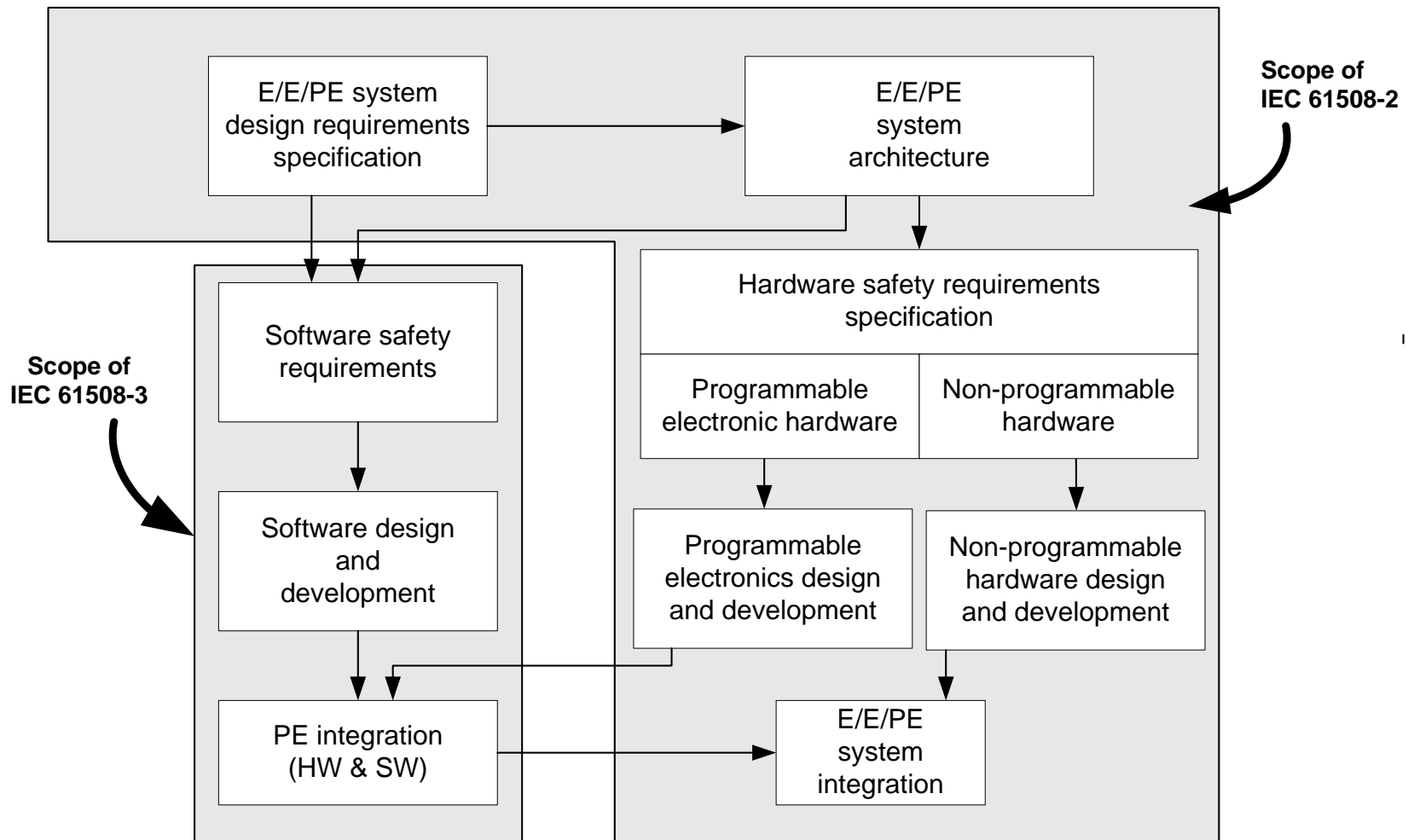
# FBIS Development IEC61508 Lifecycle



# FBIS Development IEC61508 Lifecycle

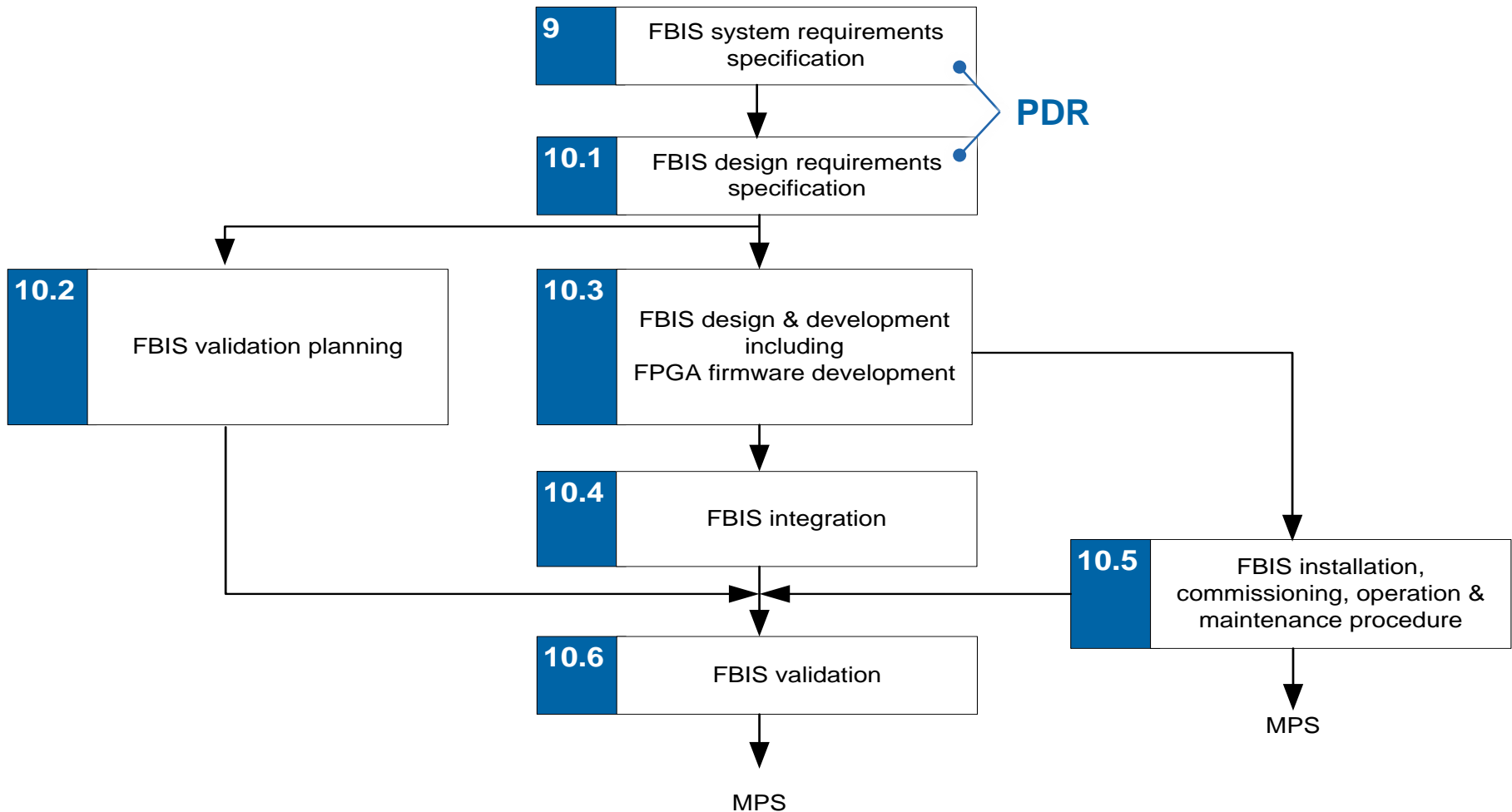


## Figure 5 Relationship and scope for IEC 61508-2 and IEC 61508-3





# FBIS Development FBIS Lifecycle



# FBIS Development System Requirements Specification

## FBIS-SRS

### System Requirements Specification


Document which collects FBIS

- Functional Requirements
- Non-Functional Requirements

### Purpose

- Provides a common point of reference for FBIS expectations and limits
- Used for further development
- Used for verification activities

ZHAW – Institute of Applied Mathematics and Physics



## 7 Functional System Requirements

### 7.1 Interfaces

#### 7.1.1 LPSID Interface

[#ISSUE:62862](#)

Requirement	The FBIS shall have an interface for the LPSID according to the specification in <a href="#">/FBIS-LPSID-IDD/</a> .
Explanation	The LPSID may request a beam-switch off via FBIS and is hence considered to be a Sensor System in this document.

#### 7.1.2 LPSVAC Interface

[#ISSUE:63711](#)

Requirement	The FBIS shall have an interface for the LPSVAC according to the specification in TBD.
Explanation	The LPSVAC may request a beam-switch off via FBIS and is hence considered to be a Sensor System in this document.

#### 7.1.3 LPSMAG Interface

[#ISSUE:63819](#)

Requirement	The FBIS shall have an interface for the LPSMAG according to the specification in TBD.
Rationale	The ACCT Digital Processing Boards may request a beam-switch off via FBIS and are hence considered to be Sensor Systems in this document.
Explanation	The LPSMAG is considered to be a Sensor System in this document.

#### 7.1.4 ACCT Digital Processing Board Interface

[#ISSUE:63819](#)

Requirement	The FBIS shall have interfaces to the ACCT Digital Processing Boards according to the specification in TBD.
-------------	---

#### 7.1.5 RF Fast Interlock Module Interface

[#ISSUE:63828](#)

Requirement	The FBIS shall have interfaces to the RF Fast Interlock Modules according to the specification in TBD.
Explanation	The RF Fast Interlock Modules may request a beam-switch off via FBIS and are hence considered to be Sensor Systems in this document.

#### 7.1.6 Fast Gate Valve Interface

[#ISSUE:63848](#)

Requirement	The FBIS shall have interfaces to the Fast Gate Valves according to the specification in TBD.
Explanation	The Fast Gate Valves may request a beam-switch off via FBIS and are hence considered to be Sensor Systems in this document.

#### 7.1.7 ESS Timing System Interface

[#ISSUE:63278](#)

Requirement	The FBIS shall have an interface to the ESS Timing System according to the specification in <a href="#">/FBIS-TS-IDD/</a> .
Explanation	The ESS Timing System has a dual role. It provides timing and configuration information and is used to inhibit the proton beam. Hence, it is considered to be an Actuation Systems and a Higher-Level Safety and Control System in this document.

Page: 20 / 73    Version: 1.0

# FBIS Development

## System Requirements Specification

## FBIS-SRS

### Requirements Management

- We use CodeBeamer for requirements management
- SRS is realized as issue tracker
- Requirements are realized as tracker items
- FBIS-SRS is a generated document

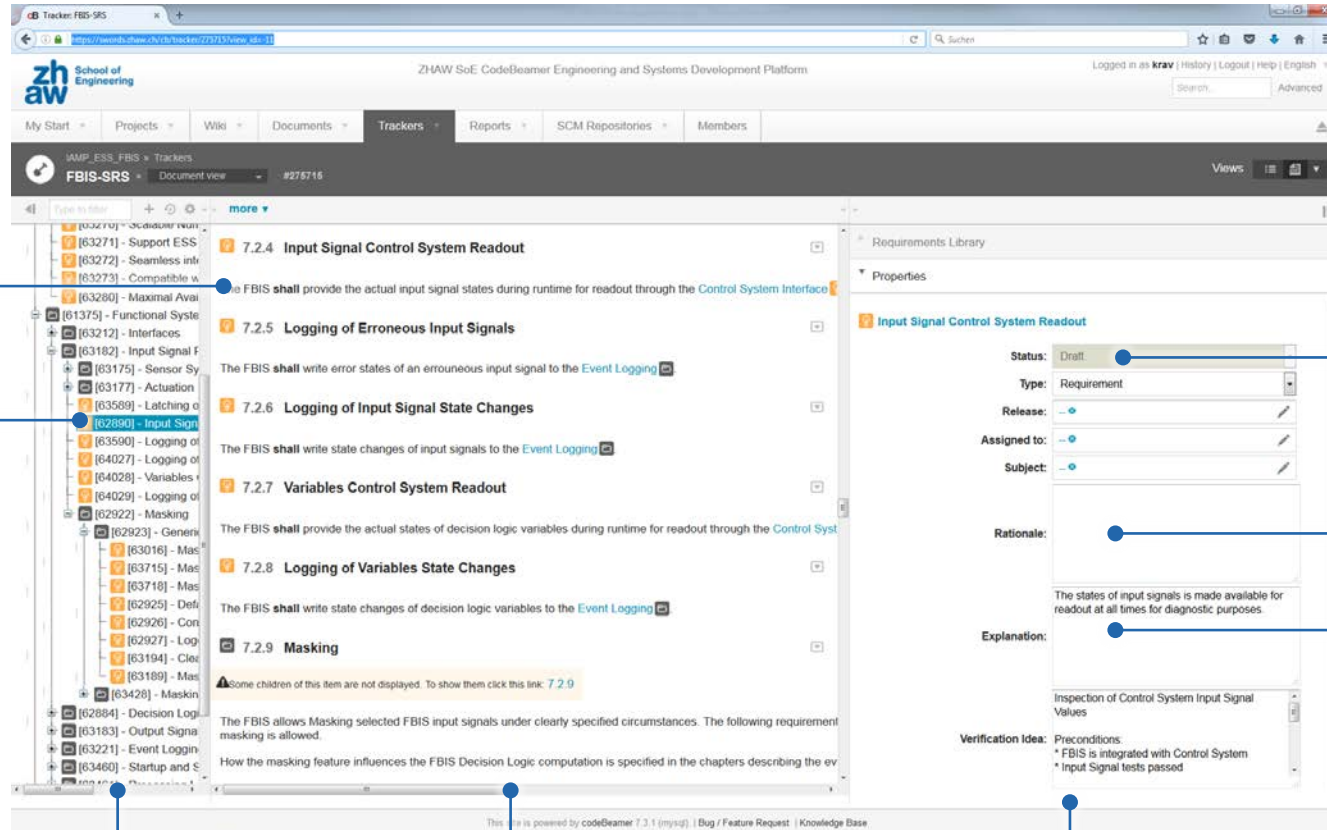
### CodeBeamer

Using CodeBeamer for Requirements Management provides per requirement:

- Automatic unique ID
- Status
- Log
- Traceability
- Workflow

# FBIS Development System Requirements Specification

## FBIS-SRS in CodeBeamer



Description

ID

Document Structure

Document View

Requirement Properties

Status

Rationale

Explanation

## FBIS-SRS

### Requirements Guidelines

- Wiki [Requirement Guidelines](#)

### Purpose

- Provides a common language templates for requirement texts
- Provides quality criteria for requirement and set of requirements
- Adapted from EARS: The Easy Approach to Requirements Syntax, Alistair Mavin et al (2009)

### Quality Criteria

#### Single requirement

- Identifiable
- Atomic
- Clear
- Precise
- Feasible

#### Set of requirements, i.e. SRS

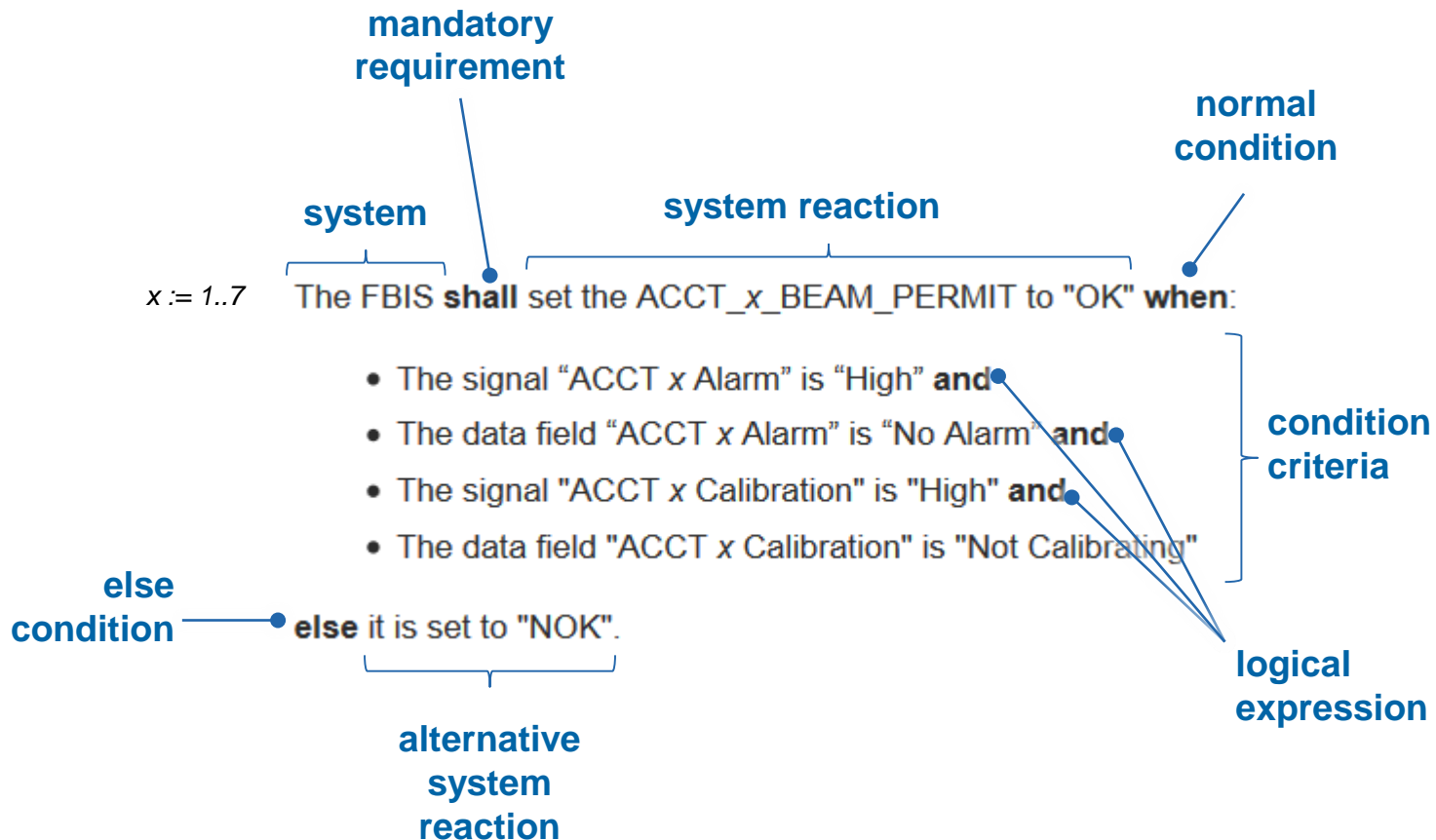
- Completeness
- Consistency
- Freedom from contradiction

# FBIS Development

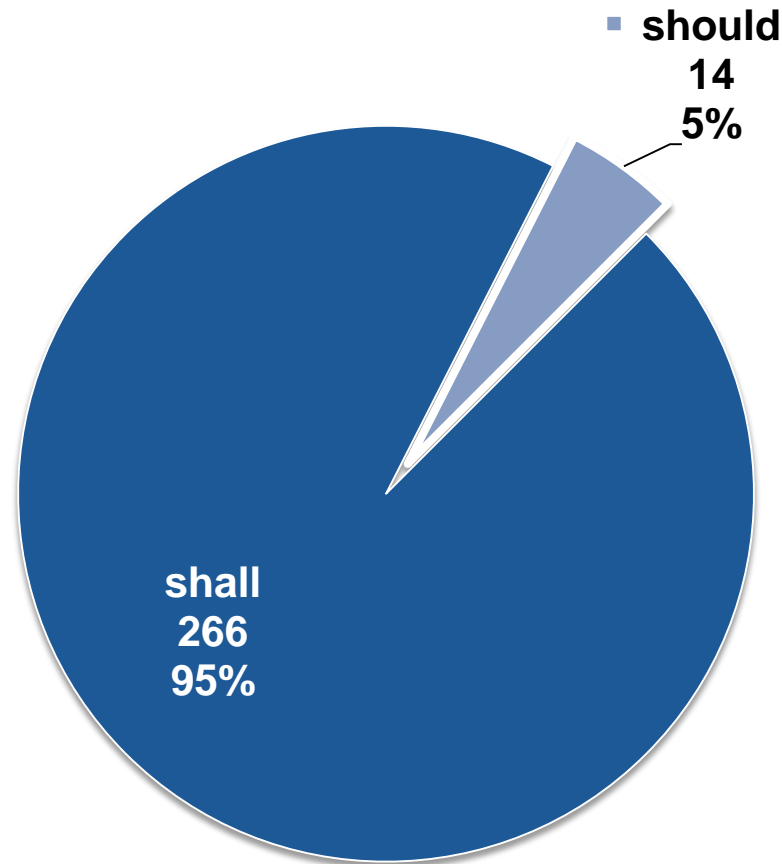
## System Requirements Specification

### FBIS-SRS

Example requirement



# FBIS Development System Requirements Specification



shall: 266 mandatory requirements, needs to be verified and validated  
should: 14 design goals

# Agenda

FBIS  
Development

V&V

HiL Test  
System

Model  
Based  
Testing

Case  
Study



# FBIS Verification & Validation

## Terms and Definitions

### Verification

- “[...] verification is the activity of demonstrating for each phase of the relevant safety lifecycle (overall, E/E/PE system and software), by analysis, mathematical reasoning and/or tests, that, for the specific inputs, the outputs meet in all respects the objectives and requirements set for the specific phase.”

IEC61508-4 3.8.1

#### **verification**

confirmation by examination and provision of objective evidence that the requirements have been fulfilled

# FBIS Verification & Validation

## Terms and Definitions

### Validation

- “[...] Validation is the activity of demonstrating that the safety-related system under consideration, before or after installation, meets in all respects the safety requirements specification for that safety-related system.”

IEC61508-4 3.8.2

#### **validation**

confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled

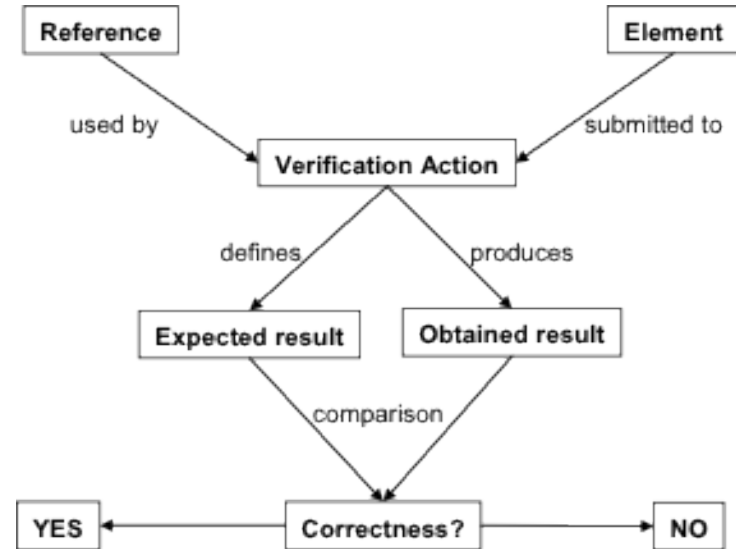
# FBIS Verification & Validation

## Terms and Definitions

### Verification Method

- Test
- Analysis
- Review
- Demonstration
- Inspection
- Simulation

Main focus on IEC61508 terms and definitions, some terms and definitions from other standards are also used or adapted.



Source: [http://sebokwiki.org/wiki/System\\_Verification](http://sebokwiki.org/wiki/System_Verification)



TARDIS  
Time And Relative Dimension(s) In Space  
From BBC TV-Show "Doctor Who"  
Source: <https://de.wikipedia.org/wiki/TARDIS#/media/File:TARDIS2.jpg>

# FBIS Verification & Validation

## Terms and Definitions

## Test

### Purpose

- To demonstrate that the system under test reacts/performs like expected.

### Typical for

- Functional Requirements
- Performance Requirements

### IEC61508-7

- B.5.1 Black box testing
- B.6.1 Fault insertion testing
- B.6.9 Worst-case testing
- C.5.27 Model based testing (test case generation)

### MIL-STD-961E

#### Test

An element of verification in which scientific principles and procedures are applied to determine the properties or functional capabilities of items.

### Example

- Stimulate inputs and observe outputs, compare actual output with expected output.

# FBIS Verification & Validation

## Terms and Definitions

## Analysis

### Purpose

- Show by analysis / calculation that the system will meet its requirements.

### Typical for

- Reliability Requirements
- Availability Requirements

### IEC61508-7

- B.6.6.5 Fault tree analysis (FTA)
- B.6.6.6 Markov models
- B.6.6.7 Reliability block diagrams (RBD)

### MIL-STD-961E

#### Analysis

An element of verification that uses established technical or mathematical models or simulations, algorithms, charts, graphs, circuit diagrams, or other scientific principles and procedures to provide evidence that stated requirements were met.

### Examples

- Calculate hardware metrics (MTBF, SFF, PFH) using Reliability Block Diagram, Fault Tree Analysis or Markov analysis.
- STPA for qualitative analysis of a proposed functional architecture

# FBIS Verification & Validation

## Terms and Definitions

## Review

### Purpose

- To check whether work products (specification, design documents, code, etc.) are correct and complete.

### Typical for

- Documents
- Source Code

### IEC61508-7

- B.2.6 Inspection of the Specification
- B.3.7 Inspection (reviews and analysis)
- B.3.8 Walk-through
- C.5.14 Formal inspections

Adapted from IEEE-Std. 1028\*

### 3.5 Review

A process or meeting during which a work product, set of work products, or a lifecycle process is presented to project personnel, managers, users, customers, user representatives, auditors or other interested parties for examination, comment or approval.

### Examples

- Review of a Requirements Specification Document to check whether requirements are complete, feasible and verifiable.
- Formal Code Inspection\*\* to check whether an algorithm is correctly implemented.

\* Original text: "A process or meeting during which a software product, set of software products, or a software process [...]"

\*\* Code Inspection is a "review" verification method and not a "inspection" verification method although the name contains the term "inspection".

# FBIS Verification & Validation

## Terms and Definitions

## Demonstration

### Purpose

- To demonstrate that the system can handle specific scenarios.

### Typical for

- Environmental Requirements
- Human Factor Requirements
- Performance Requirements

### IEC61508-7

- B.6.1 Functional testing under environmental conditions
- B.6.2 Interference surge immunity testing
- B.5.3 Statistical testing

### MIL-STD-961E

#### Demonstration

An element of verification that involves the actual operation of an item to provide evidence that the required functions were accomplished under specific scenarios. The items may be instrumented and performance monitored.

### Examples

- Electromagnetic Interferences (EMI), ESD
- Shock and Vibration
- Temperature
- Network load tests

# FBIS Verification & Validation

## Terms and Definitions

## Inspection

### Purpose

- To check that the system will meet its specified properties.

### Typical for

- Physical requirements (dimensions, weight, etc.)
- Data requirements

### IEC61508-7

- B.6.4 Static analysis

MIL-STD-961E

### Inspection

An element of verification that is generally nondestructive and typically includes the use of sight, hearing, smell, touch, and taste; simple physical manipulation; and mechanical and electrical gauging and measurement.

### Examples

- Check if the system under verification fits into a 19" rack.
- Check if the system contains a specific log entry in a specific format



# FBIS Verification & Validation

## Terms and Definitions

## Simulation

### Purpose

- To display significant aspects of the behavior of the system

### Typical for

- Hardware circuit design
- FPGA Timing

### References

- B.3.6 Simulation

IEC61508-7 B.3.6

### Simulation

To carry out a systematic and complete inspection of an electrical/electronic circuit, of both the functional performance and the correct dimensioning of the components.

### Examples

- Simulate hardware circuits for part stress data
- Simulate FPGA timing behavior using a virtual testbench

# FBIS Verification & Validation

## Terms and Definitions

## Simulation (= Animation)

### Purpose

- To display significant aspects of the behavior of the system

### Typical for

- System operational behavior
- Test case and data generation

### IEC61508-7

- C.5.17 Prototyping/animation
- C.5.26 Animation of specification and design
- C.5.27 Model based testing (test case generation)

IEC61508-4 3.8.14

### Animation

simulated operation of the software system (or of some significant portion of the system) to display significant aspects of the behaviour of the system, for instance applied to a requirements specification in an appropriate format or an appropriate high-level representation of the system design.

### Examples

- Simulate behavior using final state machine
- Simulate behavior using MATLAB/Simulink
- Generate test cases using MATLAB/Simulink

# FBIS Verification & Validation

## Verification Cross Reference Matrix

## FBIS-VCRM

### Verification Cross Reference Matrix

VCRM defines for each requirement

- verification method
- verification idea

### Purpose

- Defines starting point for detailed verification and validation plans
- Defines acceptance criteria for each requirement
- Identifies required verification resources

ZHAW – Institute of Applied Mathematics and Physics

**zhaw**

	<b>Required Interfaces:</b> <ul style="list-style-type: none"><li>• Serial Datalink</li></ul> <b>Precondition:</b> <ul style="list-style-type: none"><li>• n/a</li></ul> <b>Procedure:</b> <ul style="list-style-type: none"><li>• Send cyclic messages over serial datalink</li><li>• Change message datafields for "LPSVAC Proton Beam Mode"</li></ul> <b>Acceptance Criteria:</b> <ul style="list-style-type: none"><li>• LPSVAC_PROTON_BEAM_DESTINATION is set according to message datafields</li></ul>
--	--

**4.2.1.2.8 LPSVAC Proton Beam Destination Errors**  
[#ISSUE:64088](#)

<b>Requirement</b>	LPSVAC Proton Beam Destination Errors
<b>Verification Technique</b>	Test
<b>Verification Idea</b>	Simulate LPSVAC using HIL-Simulator <b>Required Interfaces:</b> <ul style="list-style-type: none"><li>• Serial Datalink</li></ul> <b>Preconditions:</b> <ul style="list-style-type: none"><li>• LPSVAC_PROTON_BEAM_DESTINATION is set to "Target"</li></ul> <b>Procedure:</b> <ul style="list-style-type: none"><li>• Send messages using HIL-Simulator</li><li>• Change message datafields to simulate error conditions</li></ul> <b>Acceptance Criteria:</b> <ul style="list-style-type: none"><li>• LPSVAC_PROTON_BEAM_DESTINATION is set to "None" for error conditions</li></ul>

**4.2.1.2.9 LPSVAC Proton Beam Mode Input**  
[#ISSUE:63714](#)

<b>Requirement</b>	LPSVAC Proton Beam Mode Input
<b>Verification Technique</b>	Test
<b>Verification Idea</b>	Covered by testing of related requirements <ul style="list-style-type: none"><li>• <a href="#">LPSVAC Proton Beam Mode State</a></li><li>• <a href="#">LPSVAC Proton Beam Mode Errors</a></li></ul>

**4.2.1.2.10 LPSVAC Proton Beam Mode**  
[#ISSUE:65149](#)

<b>Requirement</b>	LPSVAC Proton Beam Mode
<b>Verification Technique</b>	Test
<b>Verification Idea</b>	Covered by testing of related requirements <ul style="list-style-type: none"><li>• <a href="#">LPSVAC Proton Beam Mode State</a></li><li>• <a href="#">LPSVAC Proton Beam Mode Errors</a></li></ul>

**4.2.1.2.11 LPSVAC Proton Beam Mode State**  
[#ISSUE:65152](#)

<b>Requirement</b>	LPSVAC Proton Beam Mode State
<b>Verification Technique</b>	Test
<b>Verification Idea</b>	Simulate LPSVAC using HIL-Simulator

Page: 21 / 89    Version: 1D

# FBIS Verification & Validation

## Verification Cross Reference Matrix

### FBIS-VCRM

Example VCRM record

Reference to RequirementID

Verification method

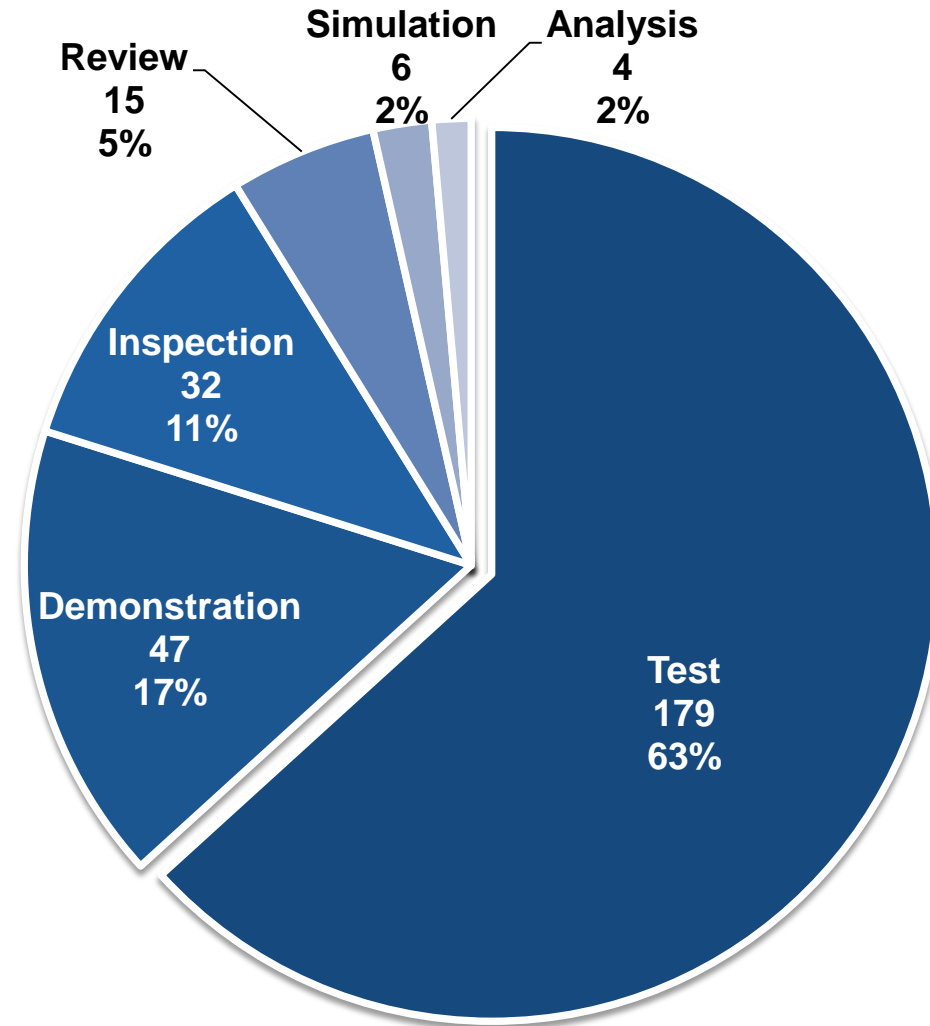
4.3.1.1.8 ACCT\_5 Beam Permit Evaluation

[#ISSUE:63625](#)

Requirement	ACCT_5 Beam Permit Evaluation
Verification Technique	Test
Verification Idea	<p>Simulate ACCT_5_BEAM_PERMIT using HLL Simulator</p> <p>Required Interfaces: <b>Required interfaces</b></p> <ul style="list-style-type: none"> <li>Discrete</li> </ul> <p>Preconditions: <b>Preconditions</b></p> <ul style="list-style-type: none"> <li>Input is configured to "No Masking"</li> </ul> <p>Procedure: <b>Procedure</b></p> <ul style="list-style-type: none"> <li>For each proton beam destination do:</li> <li>Reset</li> <li>Set ACCT_5_BEAM_PERMIT to "OK"</li> <li>Wait</li> <li>Set ENFORCED_PROTON_BEAM_DESTINATION to new proton beam destination</li> <li>Set ACCT_5_BEAM_PERMIT to "NOK"</li> <li>Reset</li> <li>Loop</li> </ul> <p>Acceptance Criteria: <b>Acceptance Criteria</b></p> <ul style="list-style-type: none"> <li>GLOBAL_BEAM_PERMIT<sub>k</sub> is set according table.</li> </ul>

# FBIS Verification & Validation

## Verification Cross Reference Matrix



## Testing

### Goal

- Verification and validation of FBIS functional requirements
- Verification of performance requirements

### Challenges

- Actual behavior of the Device-under-Test (DuT) needs to be observable
- Observed behavior needs to be checked against expected behavior
- Test oracle problem: how to get the expected results for defined inputs
- Creation of meaningful test cases and test data, i.e. test coverage
- Timing Measurement

## Testing

### Vision

Use Model Based Testing for FBIS verification (testing)

### Strategy

- Build a Hardware-in-the-Loop (HiL) Simulator to support verification and validation activities
- Combine testing techniques for Model Based Testing running on the HiL Simulator
- Demonstrate planned approach on CERN BIS
- Apply approach on ESS FBIS

### IEC61508-7 Techniques and Measures

- C.5.27 Model Based Testing
- B.5.1 Black box testing
- B.6.1 Fault insertion testing
- B.6.9 Worst-case testing
- C.5.17 Prototyping/animation
- C.5.26 Animation of specification and design

# Agenda

FBIS  
Development

V&V

HiL Test  
System

Model  
Based  
Testing

Results



# FBIS Verification & Validation Test System

## Hardware-in-the-Loop

### Applications

- Testing of Cyber-Physical Systems
- Testing of Reactive Real-Time Systems

### Goal

Testing of Fast Beam Interlock System (FBIS)

- Functional Requirements, i.e. logic
- Performance Requirements, i.e. timing



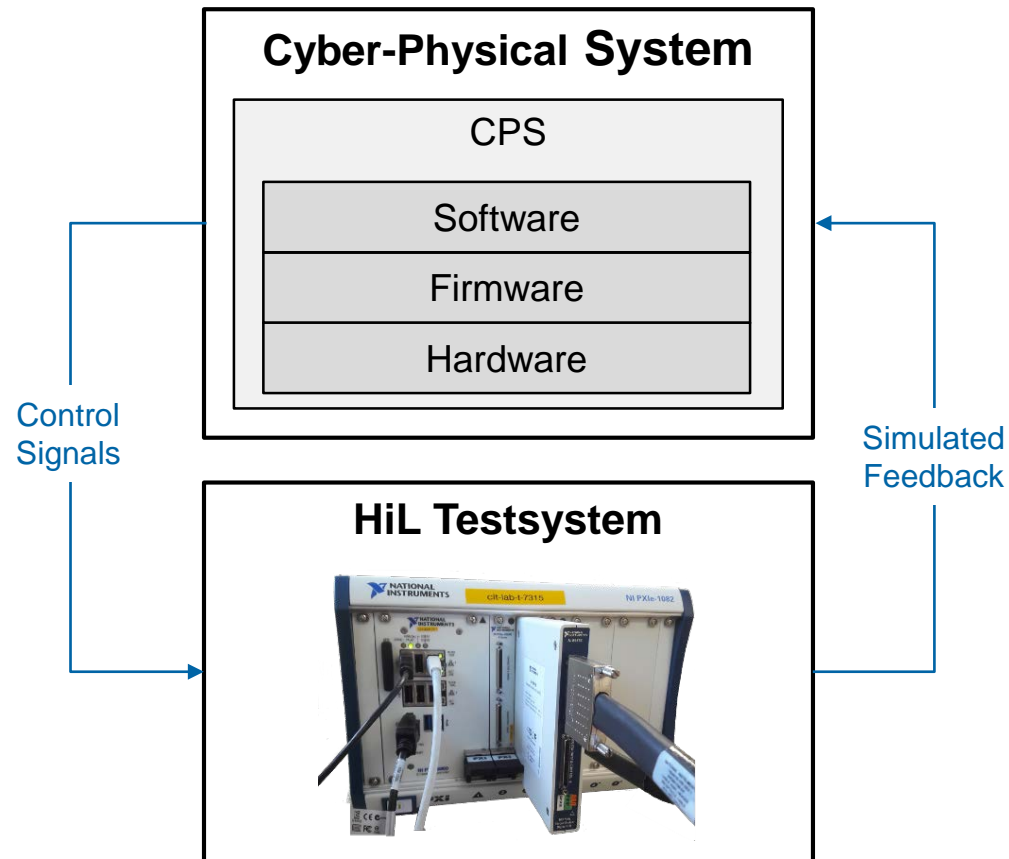
## Hardware-in-the-Loop

### Cyber-Physical System

- integrations of computation and physical processes
- Reads sensor feedback signals
- Provides actuator control signals

### HiL Simulator

- Reads actuator control signals
- Emulates physical process (plant)
- Provides simulated sensor feedback



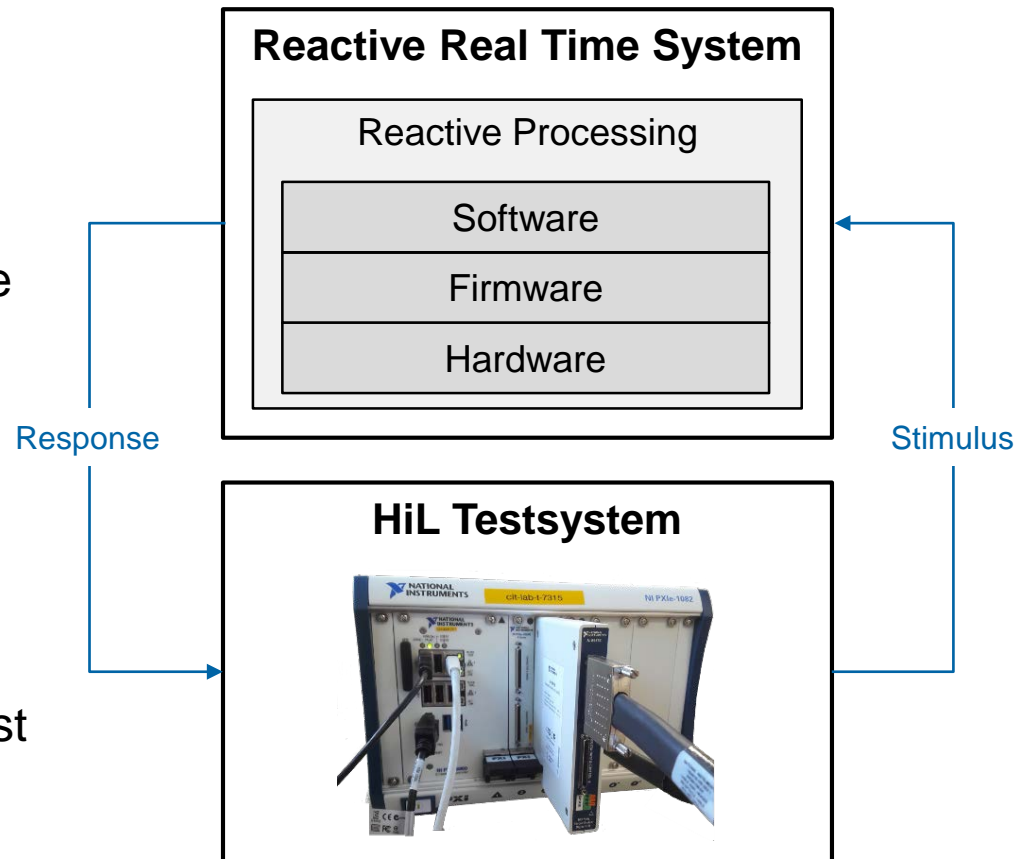
## Hardware-in-the-Loop

### Reactive Real Time System

- Acts on a stimulus and provides a response based on the current state
- Reads stimulus
- Processes information
- Provides response

### HiL Simulator

- Provides simulated stimulus
- Evaluates received response against expectation



# FBIS Verification & Validation Test System

## Hardware-in-the-Loop

### Fast Beam Interlock System (FBIS)

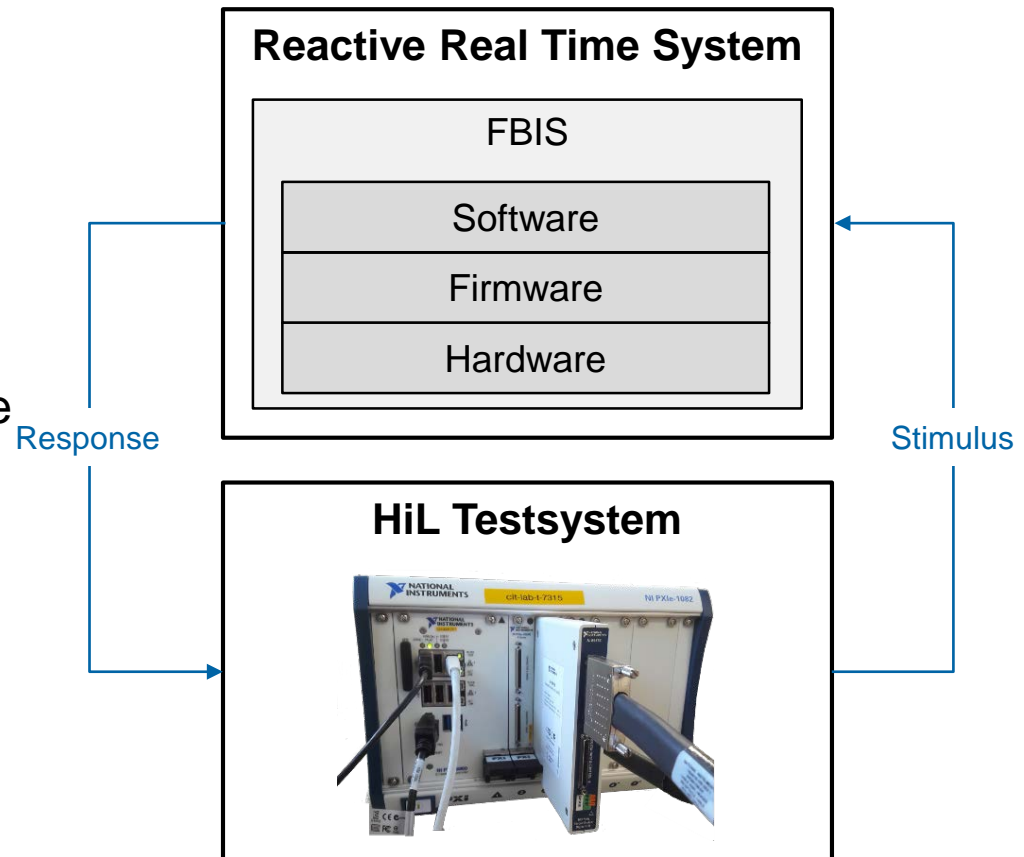
- FBIS is a Reactive Real Time System
- Acts on a stimulus and provides a response based on the current state

### Stimulus

- Simulate state change of input GLOBAL\_BEAM\_PERMIT<sub>k</sub> from “OK” to “NOK”

### Response

- Observe state change of output GLOBAL\_BEAM\_PERMIT, expected to change from “OK” to “NOK” within [TBD: time].

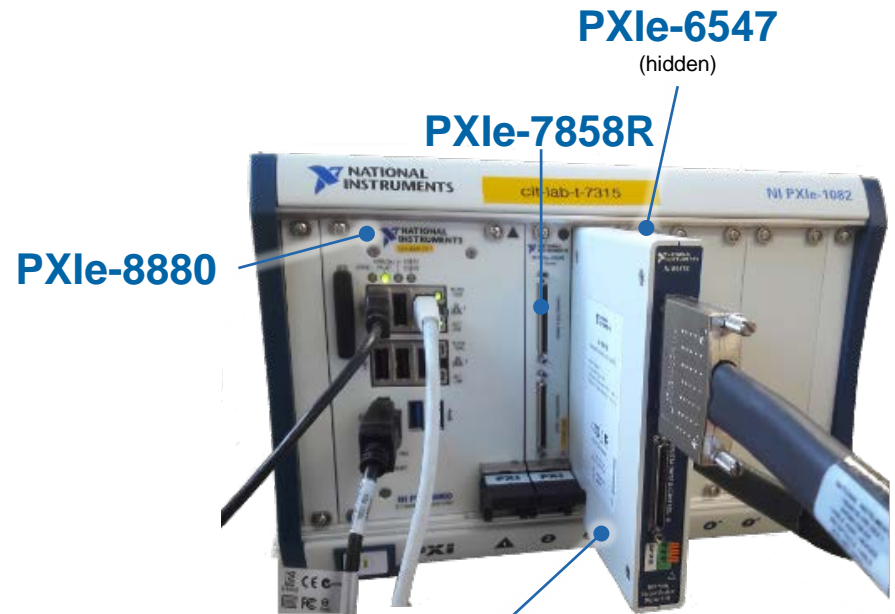


# FBIS Verification & Validation Test System

## Hardware-in-the-Loop

### Demo Test Rig

- NI PXIe-1082  
8-Slot 3U PXI Express Chassis
- PXIe-8880 RT  
Intel® Xeon® E5-2618L v3 Octa-  
Core Processor 2,3 GHz
- PXI-7858R  
NI Multifunction RIO with Kintex-7  
325T FPGA, 1MS/s AI, DRAM
- PXIe-7975R  
NI FlexRIO FPGA Kintex7 2GB
- PXIe-6547-64  
HSDIO 64 DIGIO 100MHz



PXIe-7975R

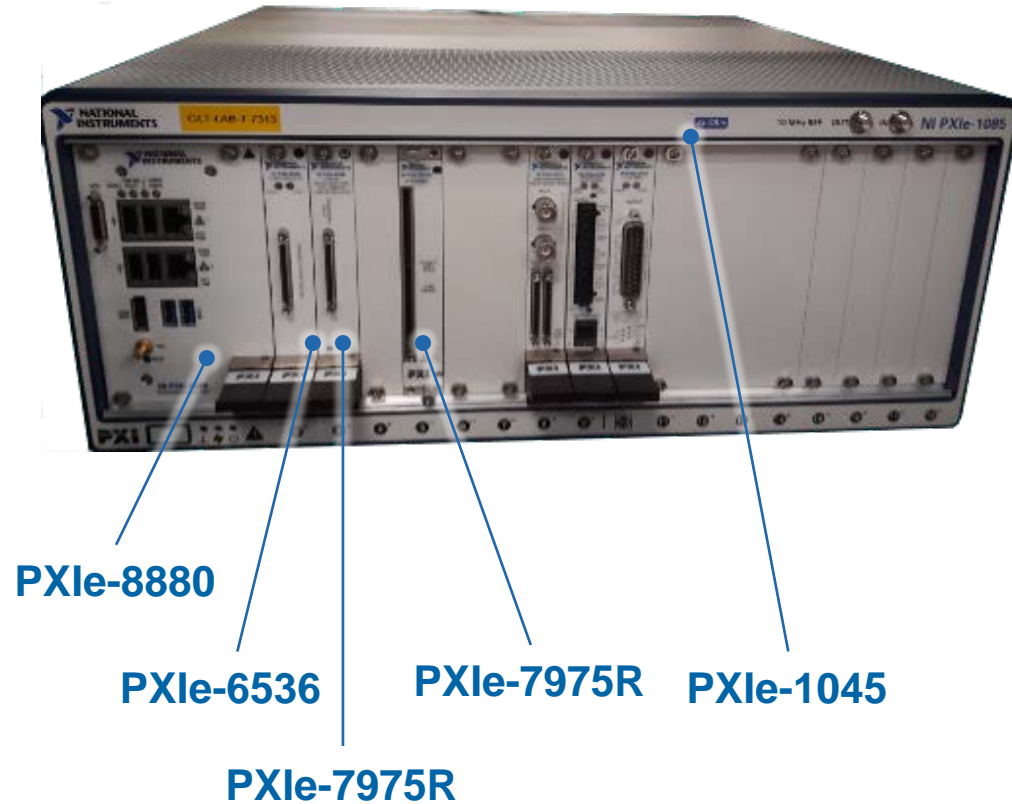


# FBIS Verification & Validation Test System

## Hardware-in-the-Loop

### Second Test Rig (1/2)

- NI PXIe-1045  
18-Slot 3U PXI Express Chassis
- PXIe-8880 RT  
Intel® Xeon® E5-2618L v3 Octa-  
Core Processor 2,3 GHz
- PXIe-6536  
High-Speed Digital I/O
- PXIe-6366  
X Series Multifunction DAQ
- PXIe-7972R  
FlexRIO FPGA Module

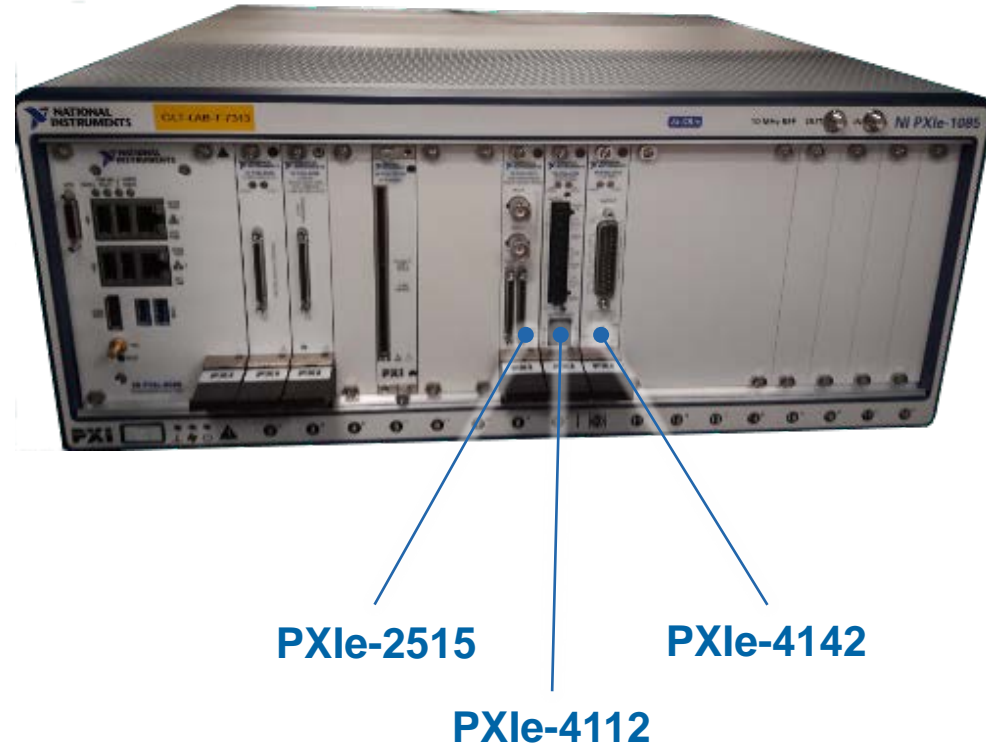


# FBIS Verification & Validation Test System

## Hardware-in-the-Loop

### Second Test Rig (2/2)

- PXIe-2515  
High-Speed Digital I/O Signal  
Insertion Switch
- PXIe-4112  
2 Channel Power Supply
- PXIe-4142  
4-channel Source-Measurement Unit



# FBIS Verification & Validation Test System

## Hardware-in-the-Loop

### Test applications out of the box

- Digital signal generation and acquisition
- Analogue signal generation and acquisition
- Fault injection tests
- Verification of control algorithms
- Comparison behavior system-under-test with Simulink control reference model
- ...





# FBIS Verification & Validation Test System

## Software

### NI VeriStand

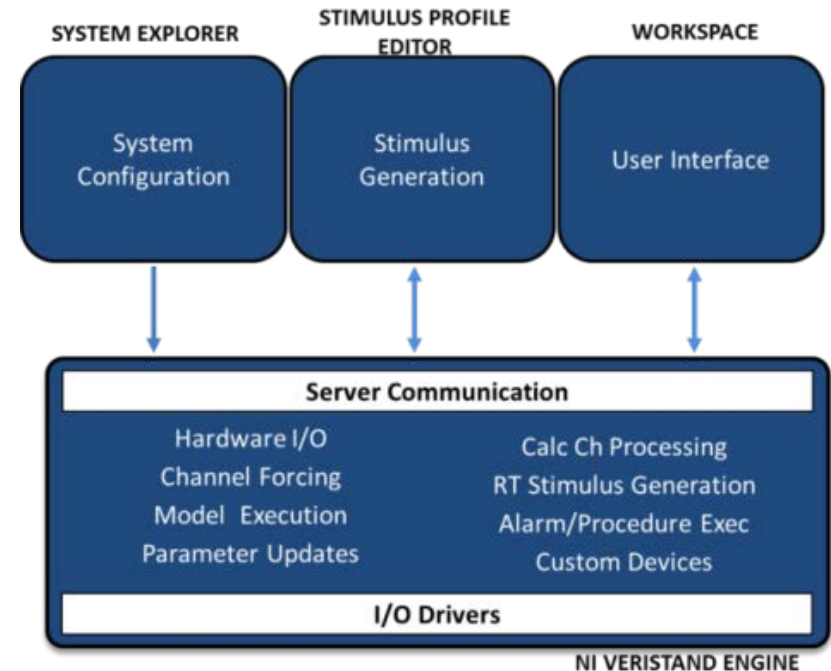
- Extendable software environment for configuring real-time test applications

### System Explorer

- Modular system definition
- Hardware configuration
- Model configuration
- Channel definition and configuration

### NI VeriStand Engine

- Handles channel communication
- Handles execution
- Extendable with custom devices



Configuring Real-Time Testing Applications  
<http://www.ni.com/white-paper/13068/en/>

# FBIS Verification & Validation Test System

## Software Defined Test System

### Simulink Model

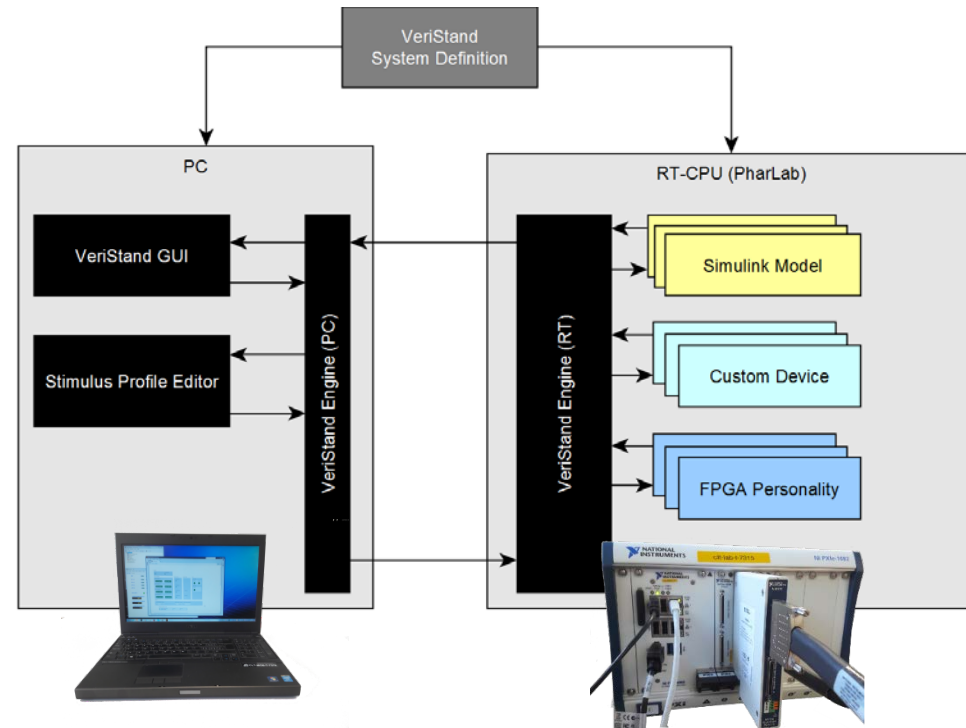
- Integration of Simulink models

### Custom Device

- Custom implemented device
- Developed in LabVIEW
- Communicates via channels
- Inline Custom Device
- Asynchronous Custom Device

### FPGA Personality

- Execution of a custom made FPGA code written in LabVIEW FPGA



# FBIS Verification & Validation Test System

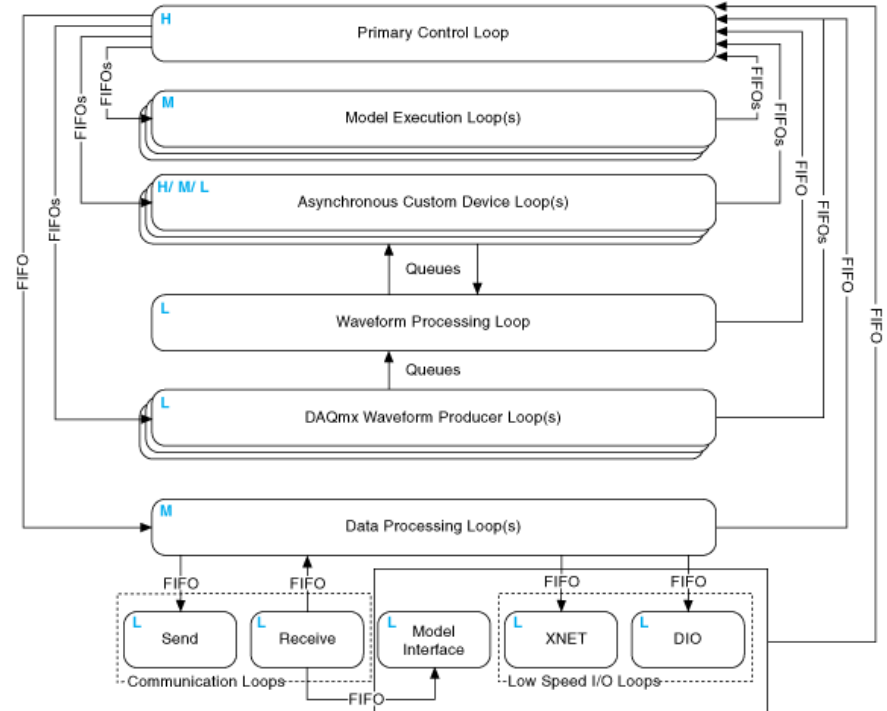
## Software Defined Test System

### NI VeriStand Engine

- Manages channel communication
- Manages execution loops

### Loops

- Primary Control Loop
- Model Execution Loop
- Asynchronous Custom Device Loop(s)



NI VeriStand Engine Architecture  
<http://www.ni.com/product-documentation/13033/en/>

# FBIS Verification & Validation Test System

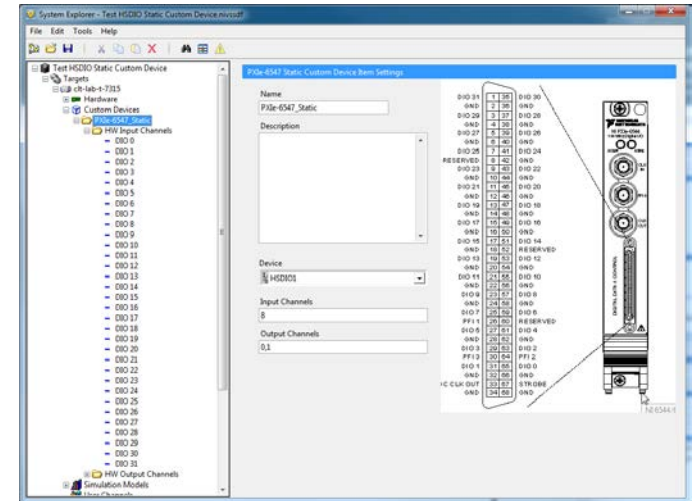
## Software Defined Test System

### Custom Device for Logic Testing

- Driver for PXIe-6547
- Testing of logic behavior
- Digital stimulus-response testing based on channel data
- Channels for digital output
- Channels for digital input
- Optimized for integration with Simulink models

### Channels

- 32 Input Channels
- 32 Output Channels



# FBIS Verification & Validation Test System

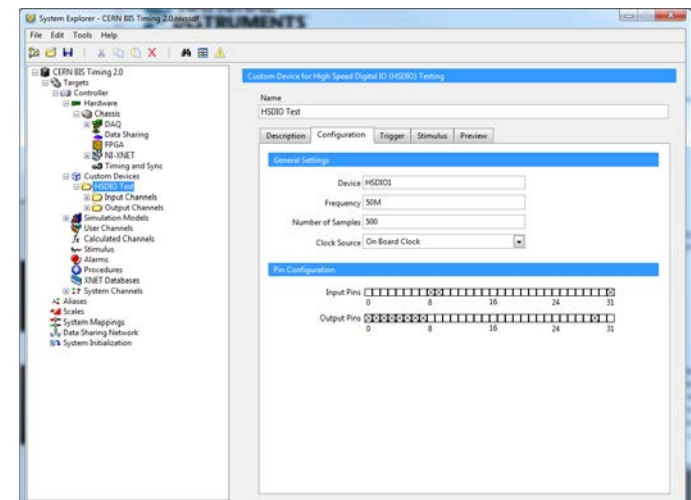
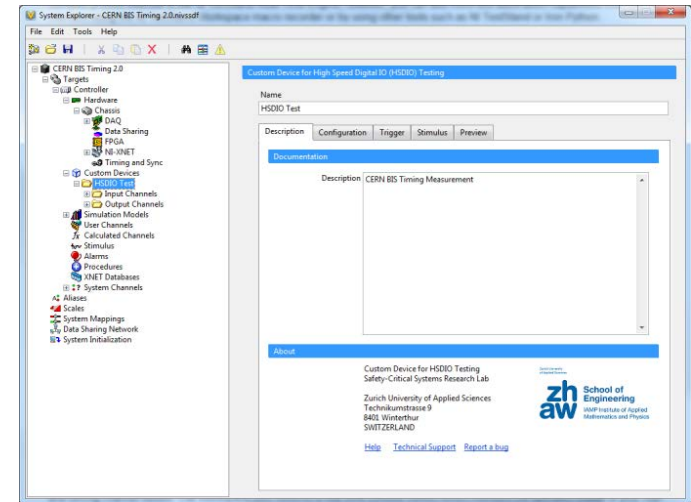
## Software Defined Test System

### Custom Device for HSDIO Testing

- Driver for PXIe-6547
- Digital stimulus-response testing
- Playback of predefined stimulus files
- Signal generation and acquisition up to 50 MHz
- Integrated timing measurement
- Trigger configuration

### Channels

- Control Channels
- Status Channels
- Timing Channels



# FBIS Verification & Validation Test System

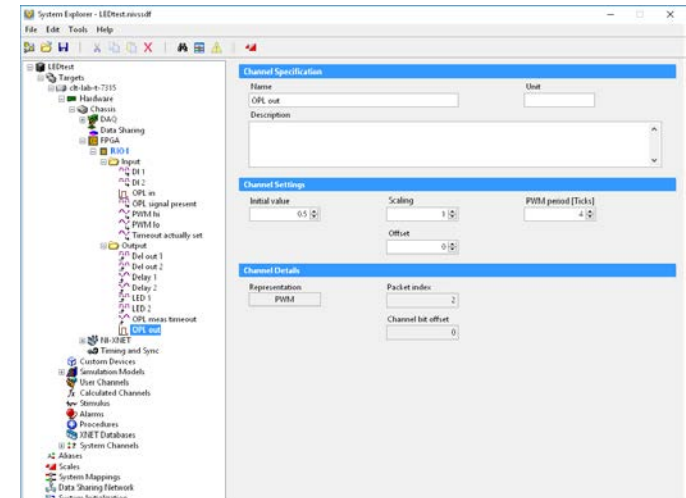
## Software Defined Test System

### FPGA Personality for Optical Protection Line (OPL)

- Generation of square-wave signal up to 10 MHz
- Reading of square-wave signal
- OPL break detection
- Digital out signal for OK/NOK
- Work in progress

### Channels

- Control Channels
- Status Channels



# Agenda

Basics

Approach

HiL Test  
System

Model  
Based  
Testing

Case  
Study

### Model Based Testing (MBT)

- black-box approach in which common testing tasks such as test case generation (TCG) and test results evaluation are based on a model of the system (application) under test (SUT).
- Model-based testing is the automatic generation of efficient test cases/procedures using models of system requirements and specified functionality

IEC61508-7 C.5.27

#### Model based testing (test case generation)

Aim: To facilitate efficient automatic test case generation from system models and to generate highly repeatable test suites.



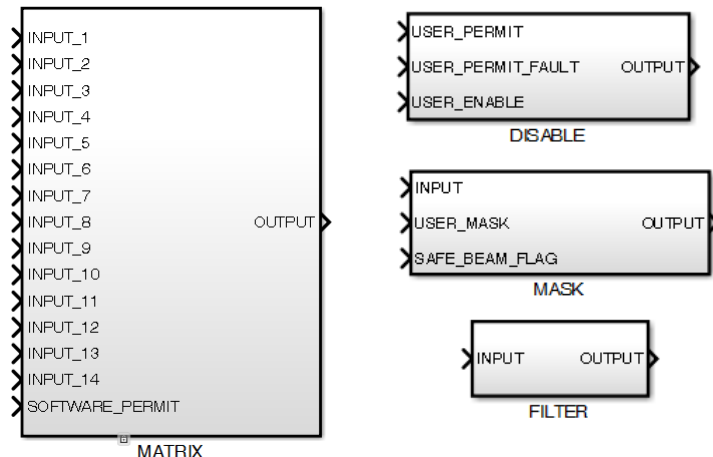
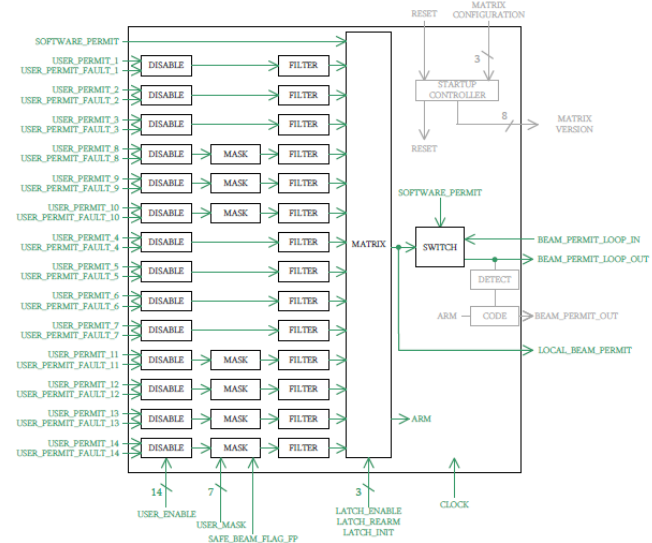
# FBIS Verification & Validation

## Model Based Testing

### Model Based Testing (MBT)

#### Necessary Building Blocks for MBT

- Functional blocks are the building blocks to build a system model
- Functional blocks needs to be verified
- A bottom-up approach is used, i.e. a number of verified functional blocks will be integrated into a system model
- Verified system model will be used as a test oracle for comparison with realized hardware



# FBIS Verification & Validation

## Model Based Testing

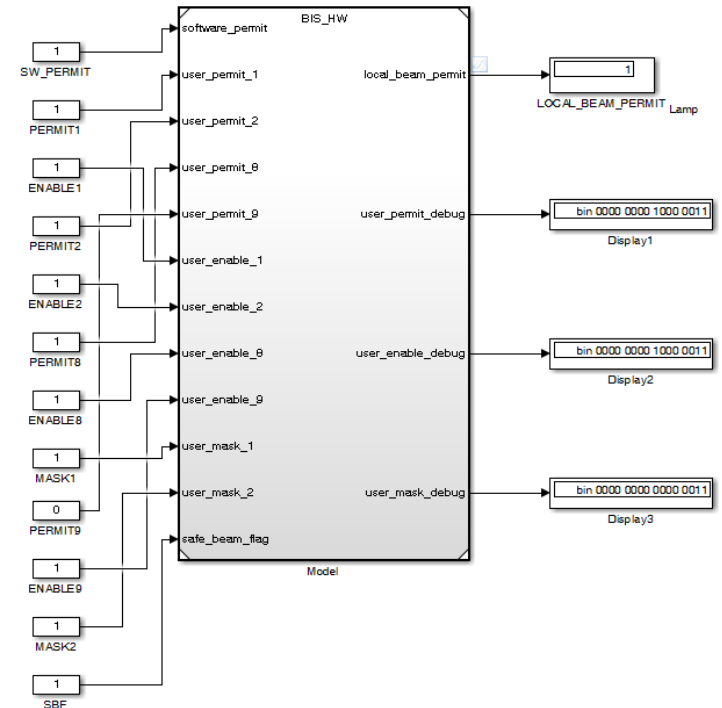
## System Model

Use a bottom up approach to create a system model:

1. Create a verified functional block
2. Integrate verified functional block in system model
3. Check with SDV if system model is verifiable
4. Generate testcases and testharness
5. Run simulation
6. Review test coverage report
7. Repeat steps 1-6 as necessary



Executable verified system model  
as test oracle



# FBIS Verification & Validation

## Model Based Testing

### Verified Functional Block

For a verified functional block do:

1. Get requirements subset
2. Create functional block in Simulink
3. Add verification block
4. Create verification script in MATLAB
5. Add test objectives for Simulink  
Design Verifier
6. Generate testcases and testharness
7. Run simulation
8. Review test coverage report

Requirements

Functional Block

Verification Block

Verification Script

Test Harness

Test Cases

Test Report

Test Coverage



Verified Functional Block

# FBIS Verification & Validation

## Model Based Testing

Requirements

Functional  
BlockVerification  
BlockVerification  
Script

## Requirements

- The System Requirements Specification (SRS) defines the desired behavior using textual and tabular functional requirements.
- A subset of related requirements is selected from the SRS and from detailed requirement specifications which allows the tester to build a functional block.

### 2.1.1 DISABLE

The disable output allows a USER\_PERMIT to be ignored when the relevant USER\_ENABLE is TRUE. This allows any channel that is not being used to be deactivated.



Figure 5 : DISABLE Functional Block

The required functionality corresponds to the following truth table:

USER_PERMIT	USER_PERMIT_FAULT	USER_ENABLE	OUTPUT
TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE
x'	x'	FALSE	TRUE

Table 4 : Truth Table of DISABLE Behaviour

Essentially this means that the OUTPUT of the DISABLE block is forced TRUE when the relevant USER\_ENABLE is FALSE. If the USER\_ENABLE is TRUE, the OUTPUT is only TRUE when USER\_PERMIT is TRUE and USER\_PERMIT\_FAULT is FALSE.

# FBIS Verification & Validation

## Model Based Testing

Requirements

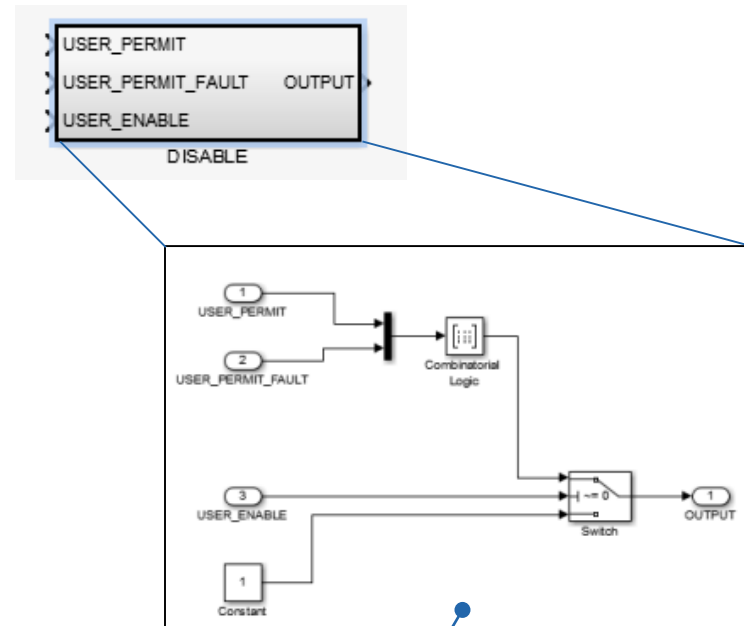
Functional  
Block

Verification  
Block

Verification  
Script

## Functional Block

- A functional block is created in Simulink for the set of related functional requirements.
- The functional block provides a executable specification for these requirements.
- Simulation of the functional block allows the tester to verify intended behavior versus specified behavior to reveal specification errors.



DISABLE.slx

Functional Block implementation

# FBIS Verification & Validation

## Model Based Testing

Requirements

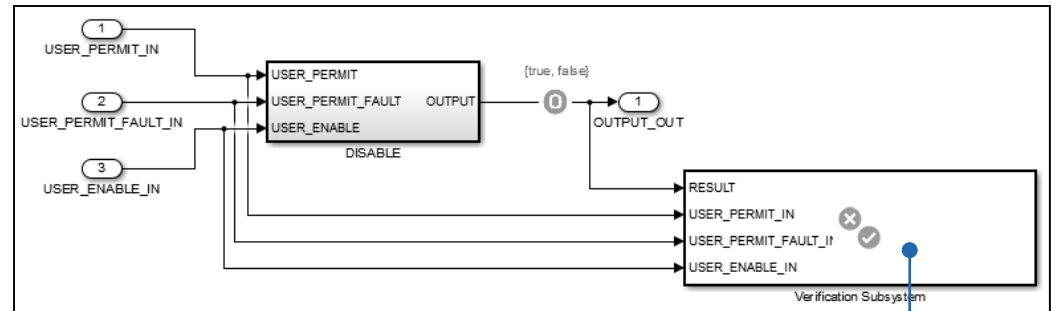
Functional  
Block

Verification  
Block

Verification  
Script

### Verification Block

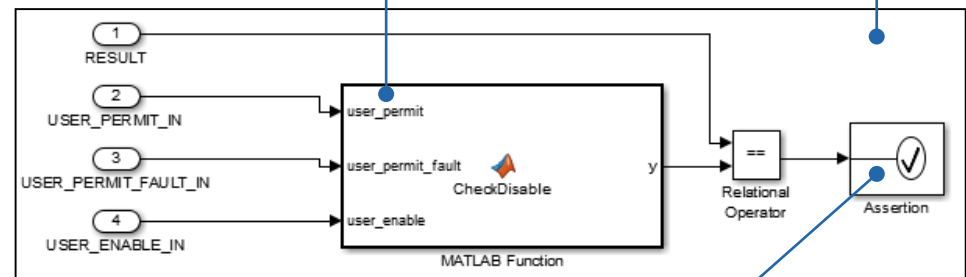
- A verification block is added to test the functional block using black box testing.
- The verification block compares the result of the functional block with the result calculated by a verification script.
- The assertion block will stop simulation with an error message when results are not identical.



DISABLE\_UUT.slx

Verification Script

Verification block



DISABLE\_UUT.slx: Verification Subsystem

Assertion block

# FBIS Verification & Validation Model Based Testing

Requirements

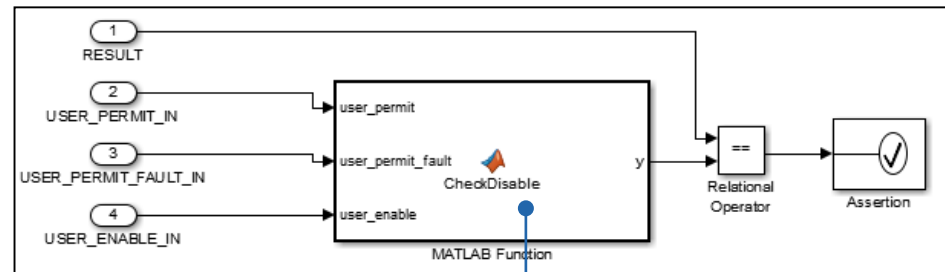
Functional  
Block

Verification  
Block

Verification  
Script

## Verification Script

- The verification script is a diverse implementation of the selected functional requirements.
- Elements within the Verification block are not part of code generation and can therefore use any available toolboxes within MATLAB for verification.
- For example, a requirement specified with a truth table can be verified in that way.



DISABLE\_UUT.slx: Verification Subsystem

## Verification Script

```
function y = CheckDisable(user_permit,user_permit_fault,user_enable)
if (user_permit==true) && (user_permit_fault==true) && (user_enable==true)
    x = false;
elseif (user_permit==false) && (user_permit_fault==true) && (user_enable==true)
    x = false;
elseif (user_permit==true) && (user_permit_fault==false) && (user_enable==true)
    x = true;
elseif (user_permit==false) && (user_permit_fault==false) && (user_enable==true)
    x = false;
elseif (user_enable==false)
    x = true;
else
    x = false;
end
y = x;
```

DISABLE\_UUT.slx: Verification Subsystem / MATLAB Function

# FBIS Verification & Validation

## Model Based Testing

Test Objectives

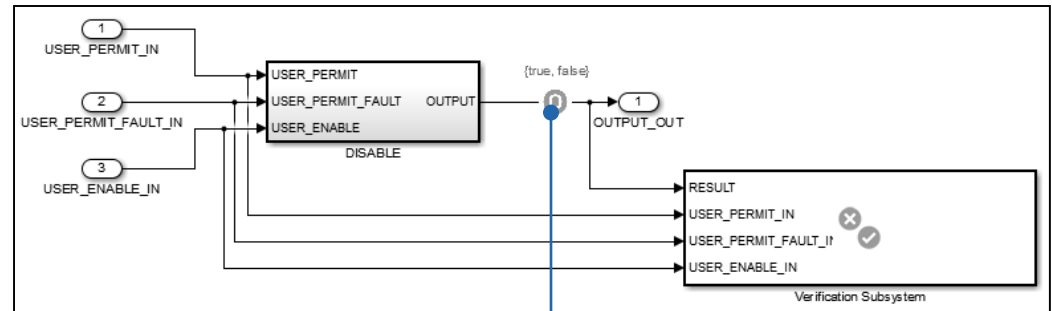
Test Harness  
Test Cases

Simulation  
Run

Test  
Coverage

## Test Objectives

- Test objectives are added to the Simulink model to guide Simulink Design Verifier's Test Case Generator to find inputs which will set the output to the specified values.
- Simulink Design Verifier (SDV) uses formal model checking techniques to find test cases.
- See formal methods presentation for details



DISABLE\_UUT.slx

Test Objective



# FBIS Verification & Validation

## Model Based Testing

Test Objectives

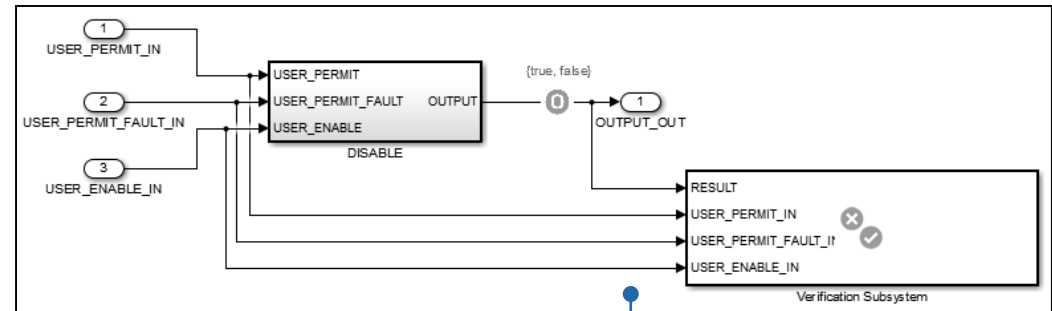
Test Harness  
Test Cases

Simulation  
Run

Test  
Coverage

### Test Harness

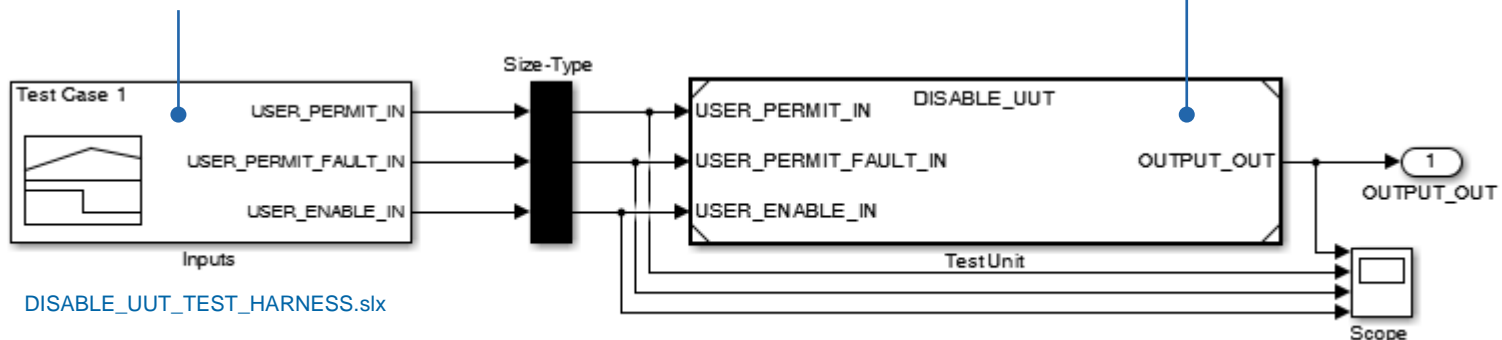
- Test harness and test cases are created by the SDV.
- SDV tries to optimize test case generation for test coverage based on the functional block.



DISABLE\_UUT.slx

Unit under Test (UUT)

Signal Builder Block



DISABLE\_UUT\_TEST\_HARNESS.slx

# FBIS Verification & Validation Model Based Testing

Test Objectives

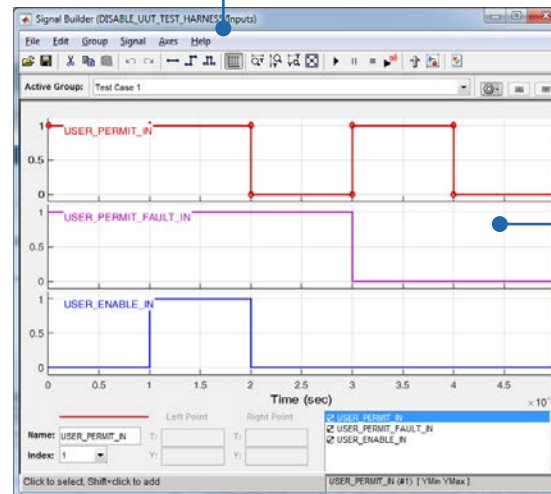
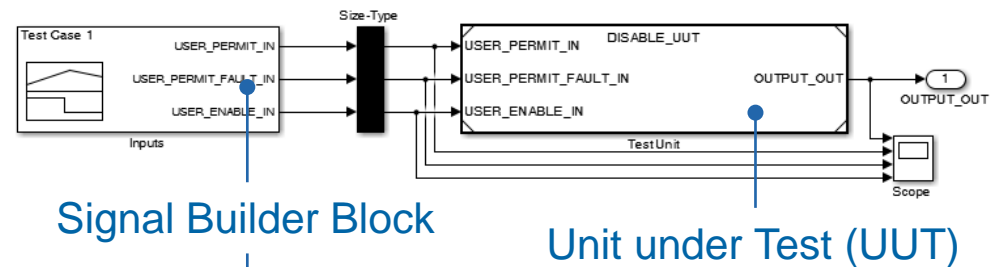
Test Harness  
Test Cases

Simulation  
Run

Test  
Coverage

## Simulation Run

- The test harness includes a signal builder block.
- The simulation run will use the signals generated by SDV.
- The signals will form input combinations and are used as input for the unit under test.
- The verification block will stop the simulation run when it detects abnormal behavior.



Generated  
Signals

# FBIS Verification & Validation Model Based Testing

Test Objectives

Test Harness  
Test Cases

Simulation  
Run

Test  
Coverage

## Test Coverage

- SDV will report test coverage metrics for the generated test cases after the simulation run.
- When test coverage does not meet required criteria, the test cases needs to be adjusted.

### Metric

Cyclomatic Complexity  
Condition (C1)  
Decision (D1)

### Coverage

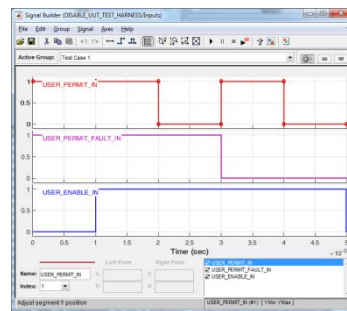
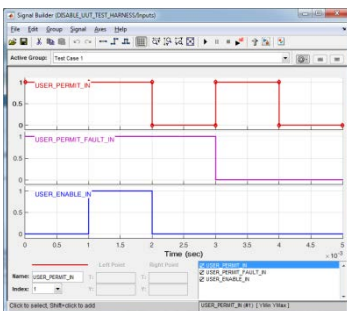
3  
50% (2/4) condition outcomes  
25% (1/4) decision outcomes

### Decisions analyzed

integer index value	25%
calculated to 0 based on inputs FF (output row 1)	0/1
calculated to 1 based on inputs FT (output row 2)	0/1
calculated to 2 based on inputs TF (output row 3)	0/1
calculated to 3 based on inputs TT (output row 4)	1/1

### Conditions analyzed

Description	True	False
input(1)	1	0
input(2)	1	0



# Agenda

Basics

Approach

HiL Test  
System

Software  
Framework

Case  
Study

## CERN BIS Case Study

- Demonstrate approach on CERN BIS
- Case study is based mainly on Engineering Specification Standard CIBM Matrix Specification [3]
- All CERN BIS related images are taken from [1-3]

## References

1. Todd, B., Dinius, A., Nouchi, P., Puccio, B., & Schmidt, R. (2005, October). The architecture, design and realisation of the LHC beam interlock system. In Proceedings of the 10th ICALEPCS International Conference on Accelerator & Large Experimental Physics Control System, Geneva Switzerland.
2. Puccio, B. et. al. (2005). Engineering Specification THE BEAM INTERLOCK SYSTEM FOR THE LHC, LHC-CIB-ES-0001-00-10
3. Todd, B. et. al. (2007). Engineering Specification BEAM INTERLOCK SYSTEM STANDARD CIBM MATRIX SPECIFICATION, AB-CO-MI 11-2007

## 8. OPERATION OF THE BEAM INTERLOCK SYSTEM

### 8.1 EXPLOITATION MODES OF THE BEAM INTERLOCK SYSTEM

The modes of the LHC Beam Interlock System (see fig.6) for each of the two LHC beams are:

- **OPERATION**
  - BEAM PERMIT
  - NO BEAM PERMIT
- **TEST**

#### 8.1.1 BEAM PERMIT

If `BEAM_PERMIT = TRUE` for one of the LHC beams, extraction of this beam from the SPS and subsequent transfer and injection into the LHC is permitted, provided that SPS extraction interlocks and transfer line interlocks give permission. When the beam permit disappears (transition from `TRUE` to `FALSE`), this beam is dumped by the Beam Dumping System.

#### 8.1.2 NO BEAM PERMIT

If `BEAM_PERMIT = FALSE` for one of the LHC beams, injection of this beam is inhibited. In this mode there should never be circulating beam in the LHC, since the transition from `TRUE` to `FALSE` should always dump the beam before. Since it takes some time to transmit the signals to the SPS extraction and LHC injection, there is a dead time of about 100  $\mu$ s: when the beam dump fires, injection will be disabled only after such dead time.

#### 8.1.3 TEST MODE STAND ALONE

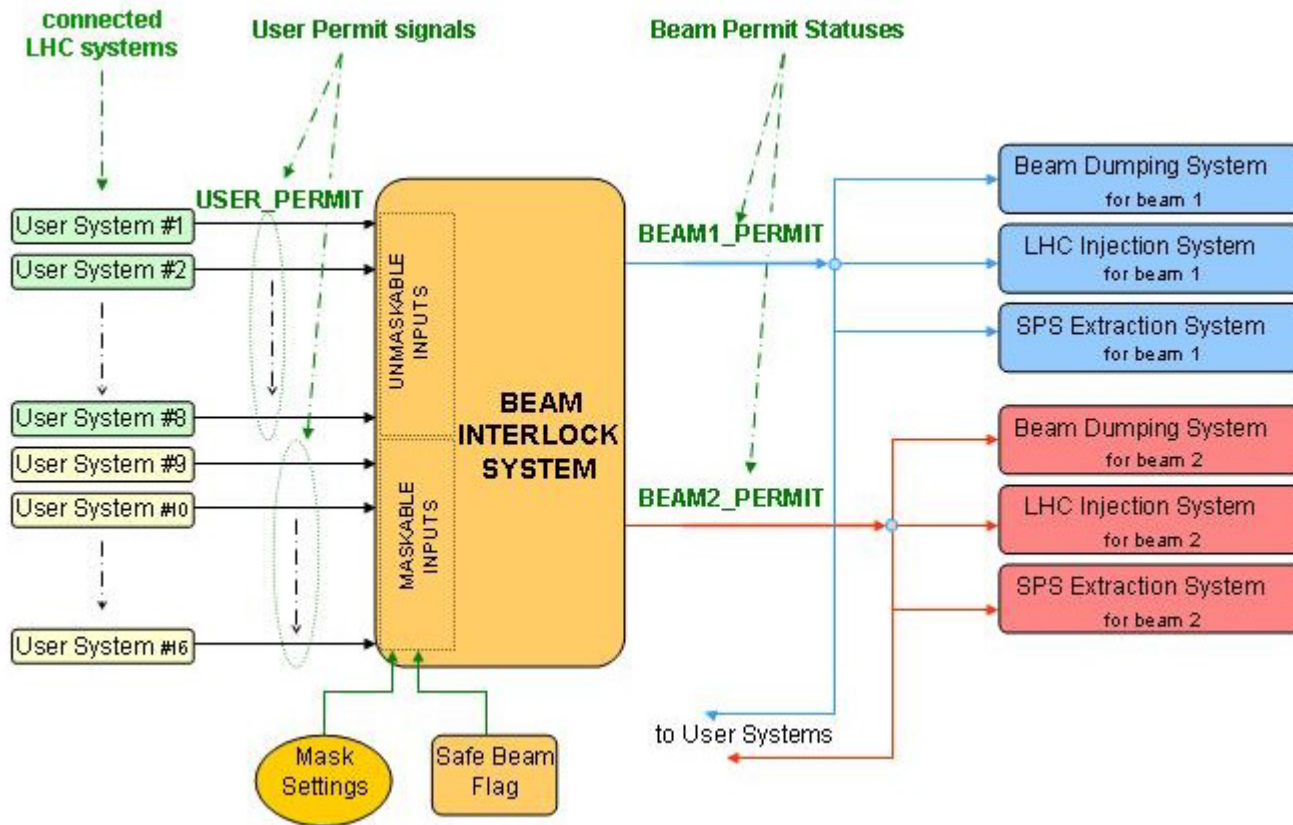
There is a test mode for each of the beams. In this mode it shall not be possible to give general beam permit. It is possible to close only one of the two `BEAM PERMIT LOOPS` for each beam in test mode:

- The *Beam Interlock User Interface* sets the `USER_PERMIT = TRUE` for one of the two redundant branches. This is done for all users, and for each Beam Interlock Controller.
- This will enable the 10 MHz signal circulating in one `BEAM PERMIT LOOP`.
- In this mode the second loop will be forced to remain open. It must never be possible to close both loops in test mode at the same time.

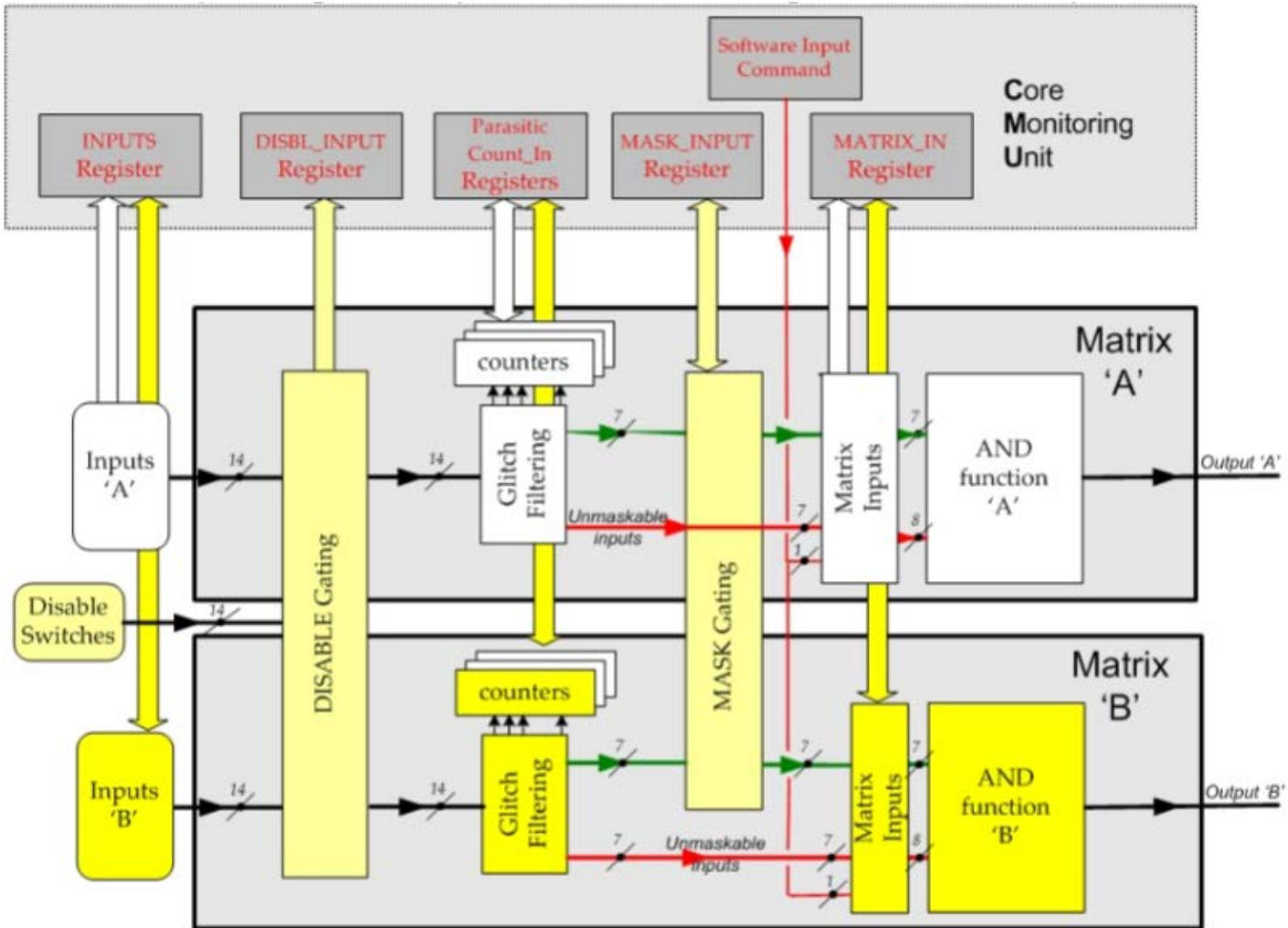
[Extract of the BIS specification \[2\]](#)

# CERN BIS Case Study

## Principle functionality of CERN's Beam Interlock System (BIS) [2]



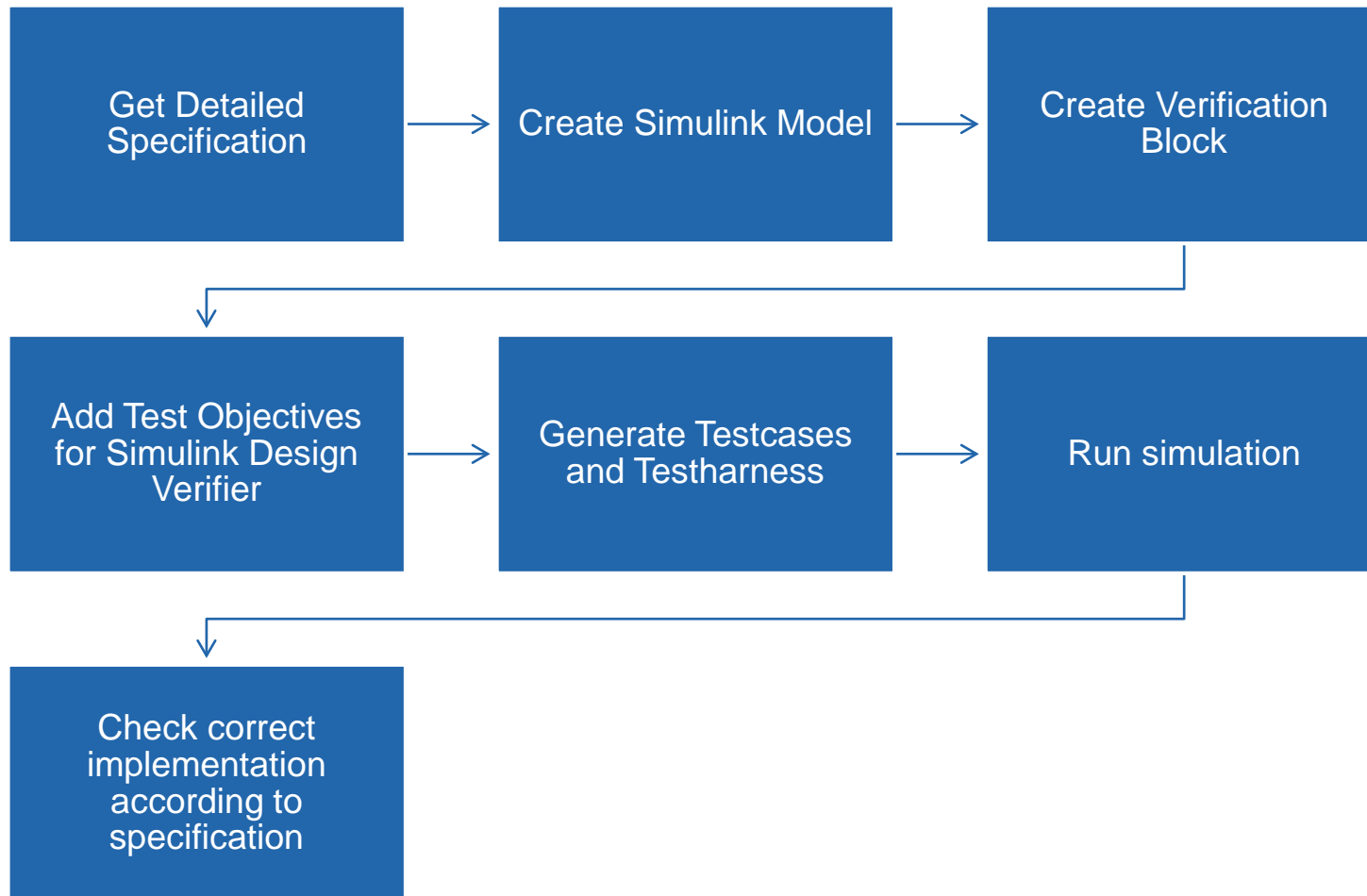
# CERN BIS Case Study



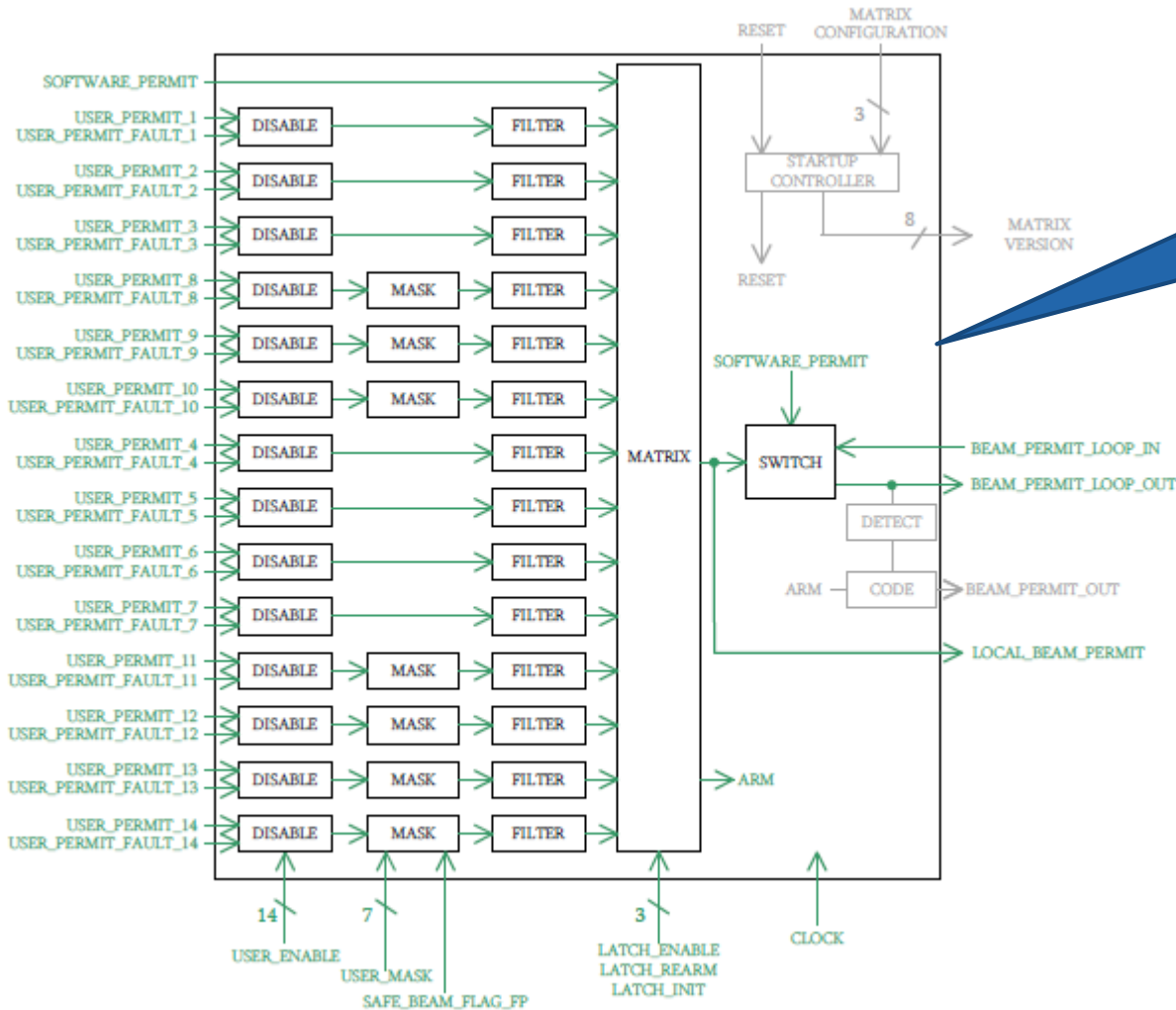
Matrixes Logic diagram with involved MMU registers



# CERN BIS Case Study



# CERN BIS Case Study



Simulink model for BIS behavior is based on this diagram

Logical Block Diagram [3]

## DISABLE Logic Specification [3]

### 2.1.1 DISABLE

The disable output allows a USER\_PERMIT to be ignored when the relevant USER\_ENABLE is TRUE. This allows any channel that is not being used to be deactivated.



Figure 5 : DISABLE Functional Block

The required functionality corresponds to the following truth table:

USER_PERMIT	USER_PERMIT_FAULT	USER_ENABLE	OUTPUT
TRUE	TRUE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE
'x'	'x'	FALSE	TRUE

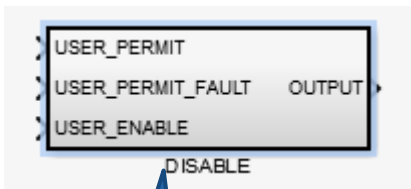
Table 4 : Truth Table of DISABLE Behaviour

Essentially this means that the OUTPUT of the DISABLE block is forced TRUE when the relevant USER\_ENABLE is FALSE. If the USER\_ENABLE is TRUE, the OUTPUT is only TRUE when USER\_PERMIT is TRUE and USER\_PERMIT\_FAULT is FALSE.

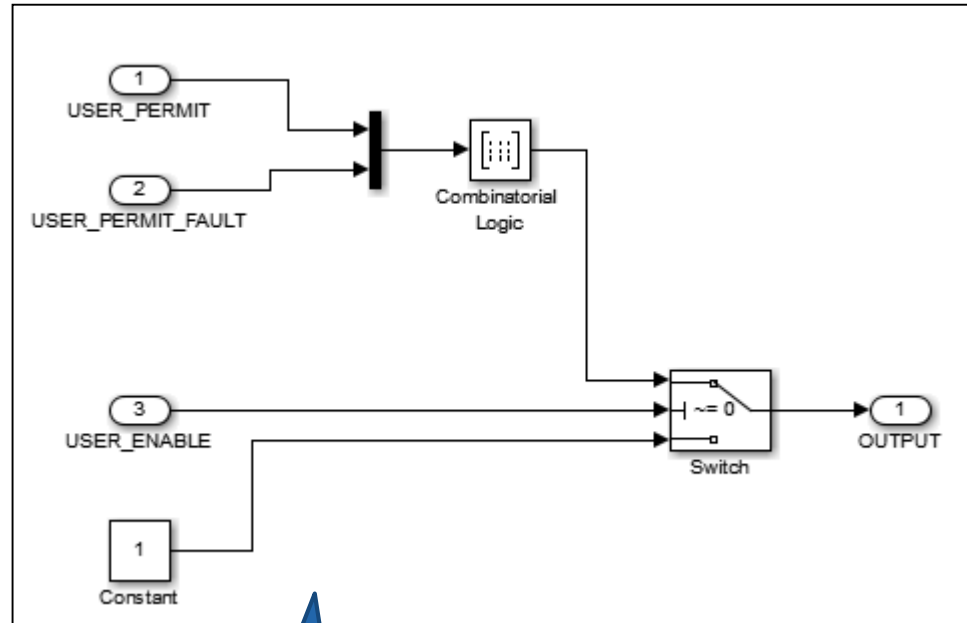
Specification of the DISABLE functional block [3]

# CERN BIS Case Study

## DISABLE Simulink Block



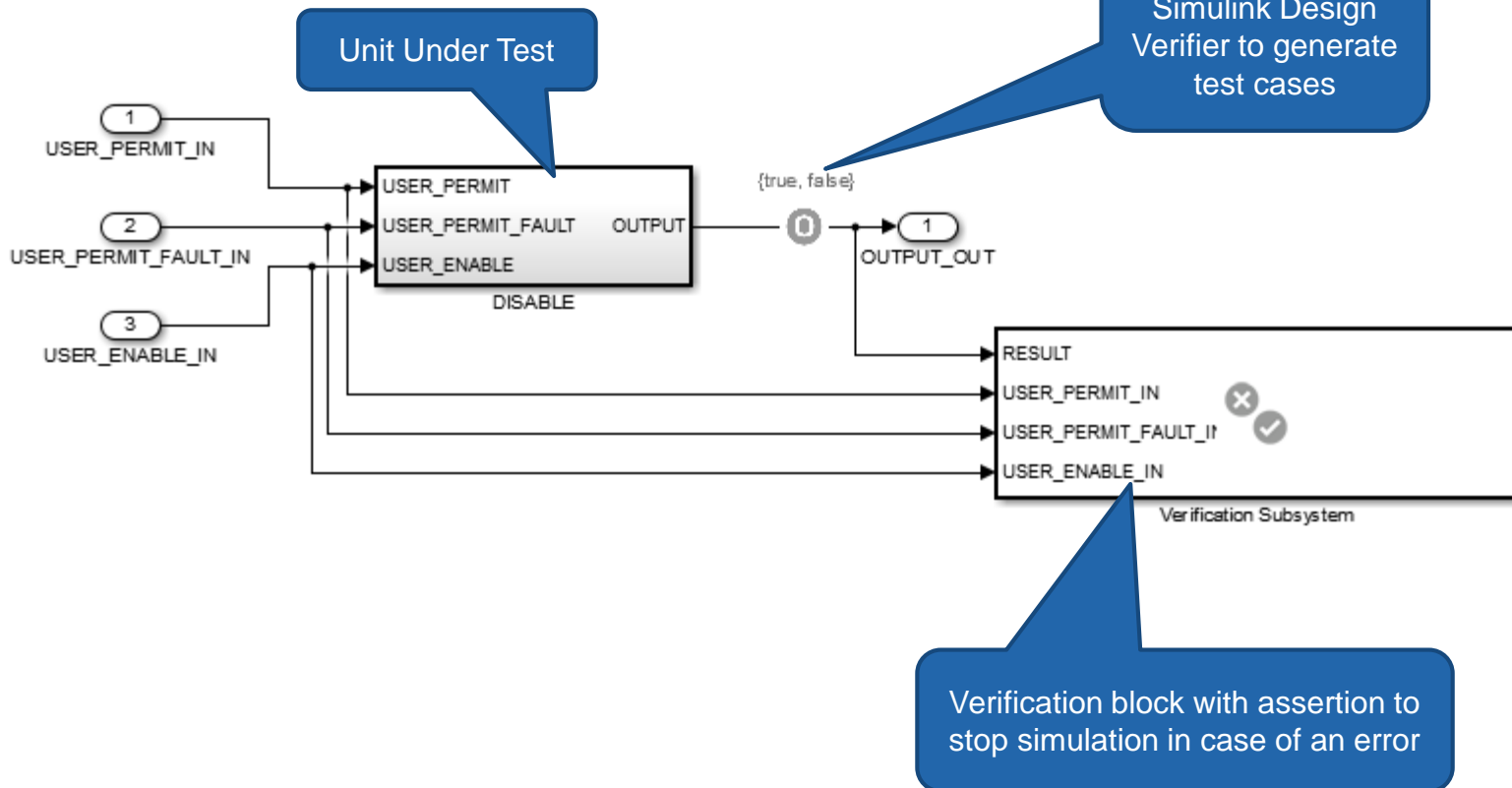
Functional  
Block



Simulink  
Implementation

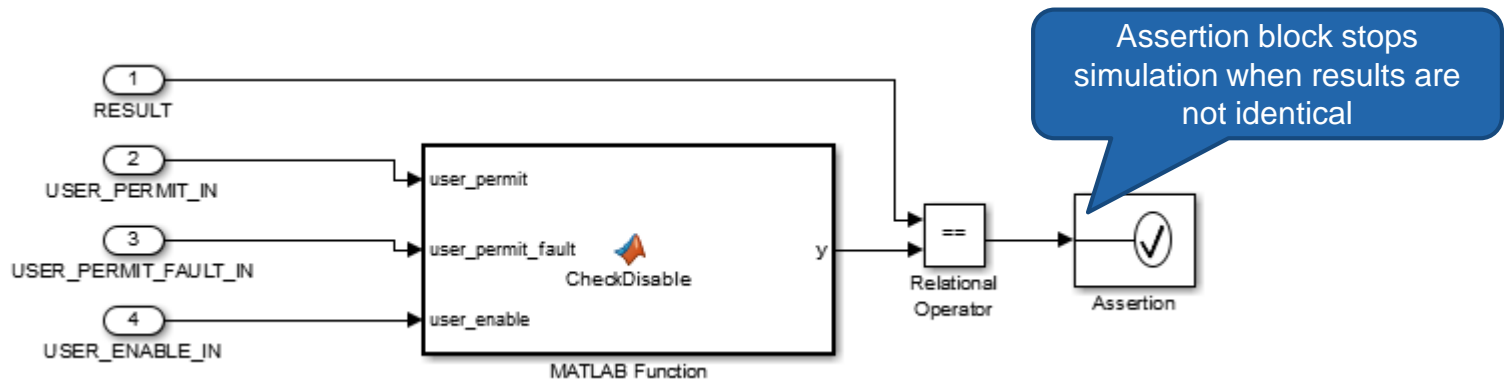
# CERN BIS Case Study

## DISABLE Unit Under Test (UUT)



# CERN BIS Case Study

## DISABLE Verification Subsystem

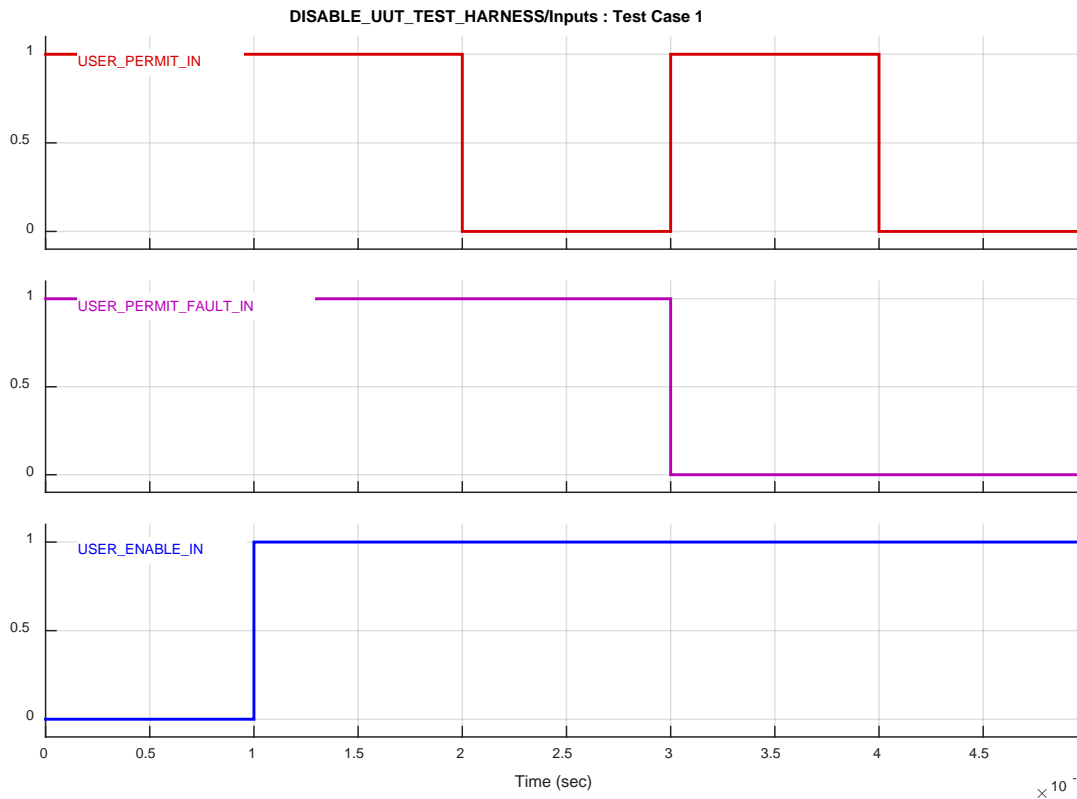


Verification script in  
MATLAB

```
function y = CheckDisable(user_permit,user_permit_fault,user_enable)
if (user_permit==true) && (user_permit_fault==true) && (user_enable==true)
    x = false;
elseif (user_permit==false) && (user_permit_fault==true) && (user_enable==true)
    x = false;
elseif (user_permit==true) && (user_permit_fault==false) && (user_enable==true)
    x = true;
elseif (user_permit==false) && (user_permit_fault==false) && (user_enable==true)
    x = false;
elseif (user_enable==false)
    x = true;
else
    x = false;
end

y = x;
```

## DISABLE SDV Generated Test Cases



SDV builds test harness model and input stimuli based on Test Objectives and coverage criteria

USER\_ENABLE\_IN modified for 100% coverage

Child Systems: [DISABLE](#)

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	1	6
Condition (C1)	NA	100% (4/4) condition outcomes
Decision (D1)	NA	100% (6/6) decision outcomes
Test Objective	NA	100% (2/2) objective outcomes



Simulation  
without  
Errors

# CERN BIS Case Study

## 2.1.2 MASK

A sub-set of the USER\_PERMIT signals can be ignored if the SAFE\_BEAM\_FLAG\_FP is TRUE and the operator chooses to do so by setting the relevant bit in the USER\_MASK register.



Figure 6 : MASK Functional Block

Thus the output of the block follows the input according to the following truth-table:

USER_PERMIT	MASK	SAFE_BEAM_FLAG	OUTPUT
'x'	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE
TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

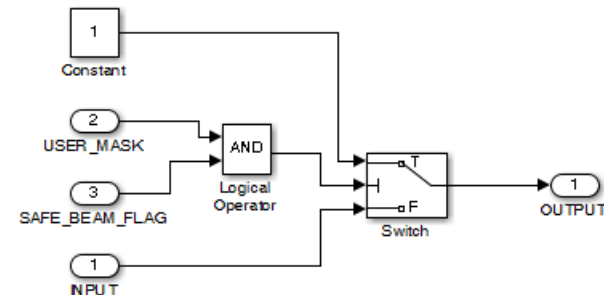
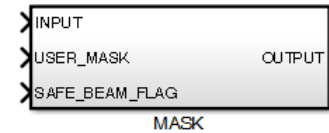
Table 5 : Truth Table of MASK Behaviour

Thus the OUTPUT of the MASK is TRUE when the relevant USER\_MASK is TRUE and the SAFE\_BEAM\_FLAG\_FP is TRUE, otherwise the OUTPUT of MASK is the same as the INPUT.

The MASK signals are only applied to half of the inputs, USER\_PERMIT 1 through 7 cannot be masked, 8 through 14 map to the USER\_MASK vector as follows:

USER_PERMIT index	USER_MASK index
8	1
9	2
10	3
11	4
12	5
13	6
14	7

Table 6 : Mapping USER\_PERMIT to USER\_MASK





## 2.1.3 FILTER

This block is very simple, having a single input and output



Figure 7 : FILTER Functional Block

The OUTPUT is a filtered version of the INPUT, with glitches removed from the input signal. A glitch is considered by the following definition:

This is shown in the following diagram

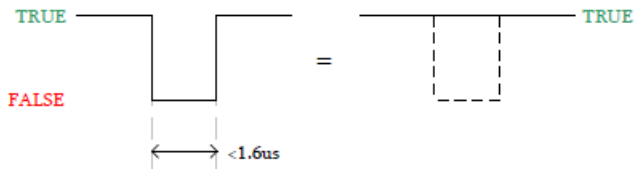
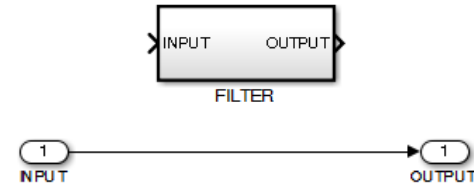


Figure 8 : Definition of a Glitch

When the glitch filter's input is TRUE, any change to FALSE followed by a return to TRUE with a FALSE duration lasting less than 1.6 microseconds is to be considered as a glitch.

The glitch filter is to be applied to every USER\_PERMIT signal, after the DISABLE and MASK functions have been applied.



No action required, because model freq is 1kHz

# CERN BIS Case Study

## 2.1.4 MATRIX

MATRIX takes the USER\_PERMIT signals that have been passed through the DISABLE, MASK and FILTER blocks as inputs and derives LOCAL\_BEAM\_PERMIT, which is only TRUE when all USER\_PERMITS are TRUE.

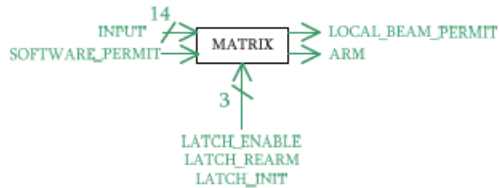


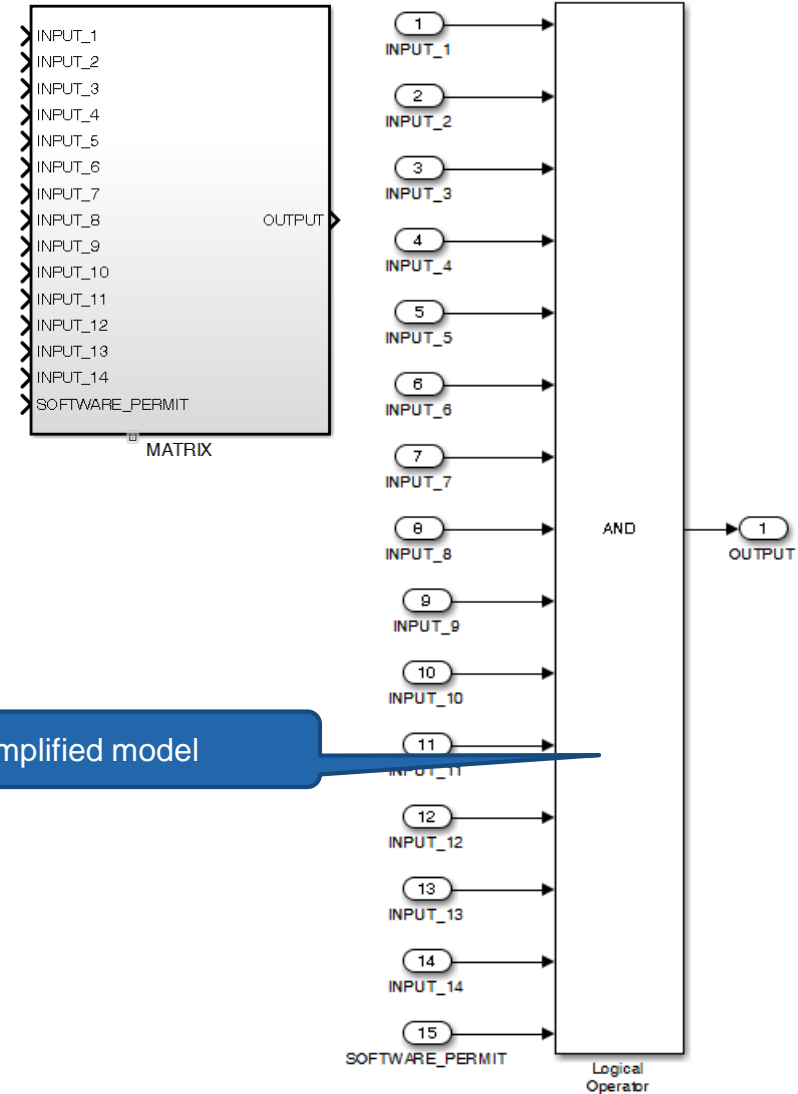
Figure 9 : MATRIX Functional Block

The ARM output is TRUE when the MATRIX has been initialised using LATCH\_INIT and has received a LATCH\_REARM signal.

LOCAL\_BEAM\_PERMIT is only TRUE when SOFTWARE\_PERMIT and all of the connected INPUTS and are TRUE, and when the MATRIX is correctly ARMED.

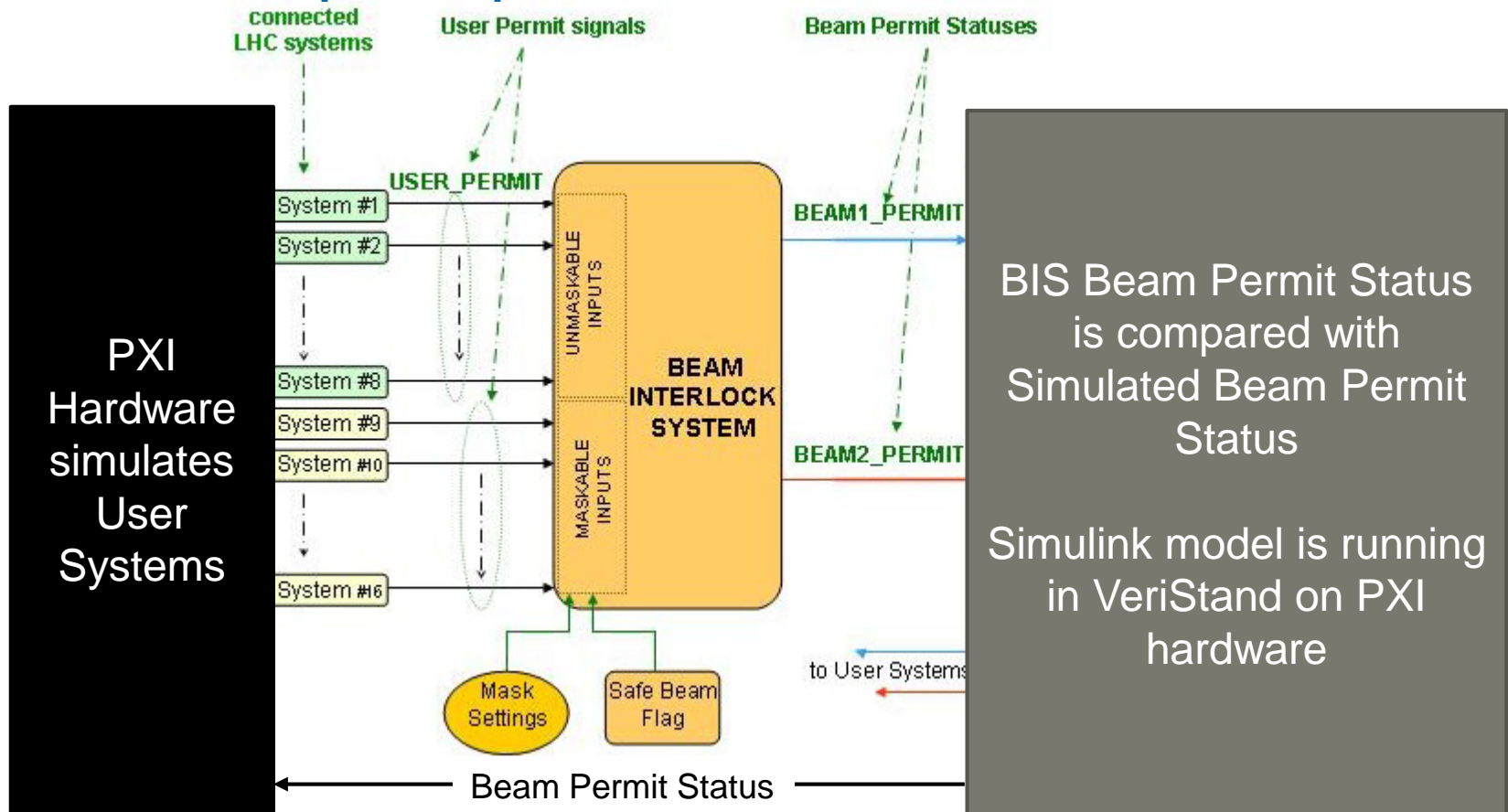
When the MATRIX is first powered, LOCAL\_BEAM\_PERMIT must be set FALSE; a transition to TRUE cannot take place until LATCH\_INIT has been set TRUE.

When LATCH\_ENABLE is TRUE every LOCAL\_BEAM\_PERMIT transition from TRUE to FALSE will result in the LOCAL\_BEAM\_PERMIT being held FALSE. A return to TRUE will only be permitted after a LATCH\_REARM command has been received.



# CERN BIS Case Study

## Hardware in the Loop Concept



Idea: Compare BIS Beam Permit Status with simulated Beam Permit Status

# CERN BIS Case Study

## Hardware



Demo Test Rig

USER\_PERMIT1 A B

USER\_PERMIT2 A B

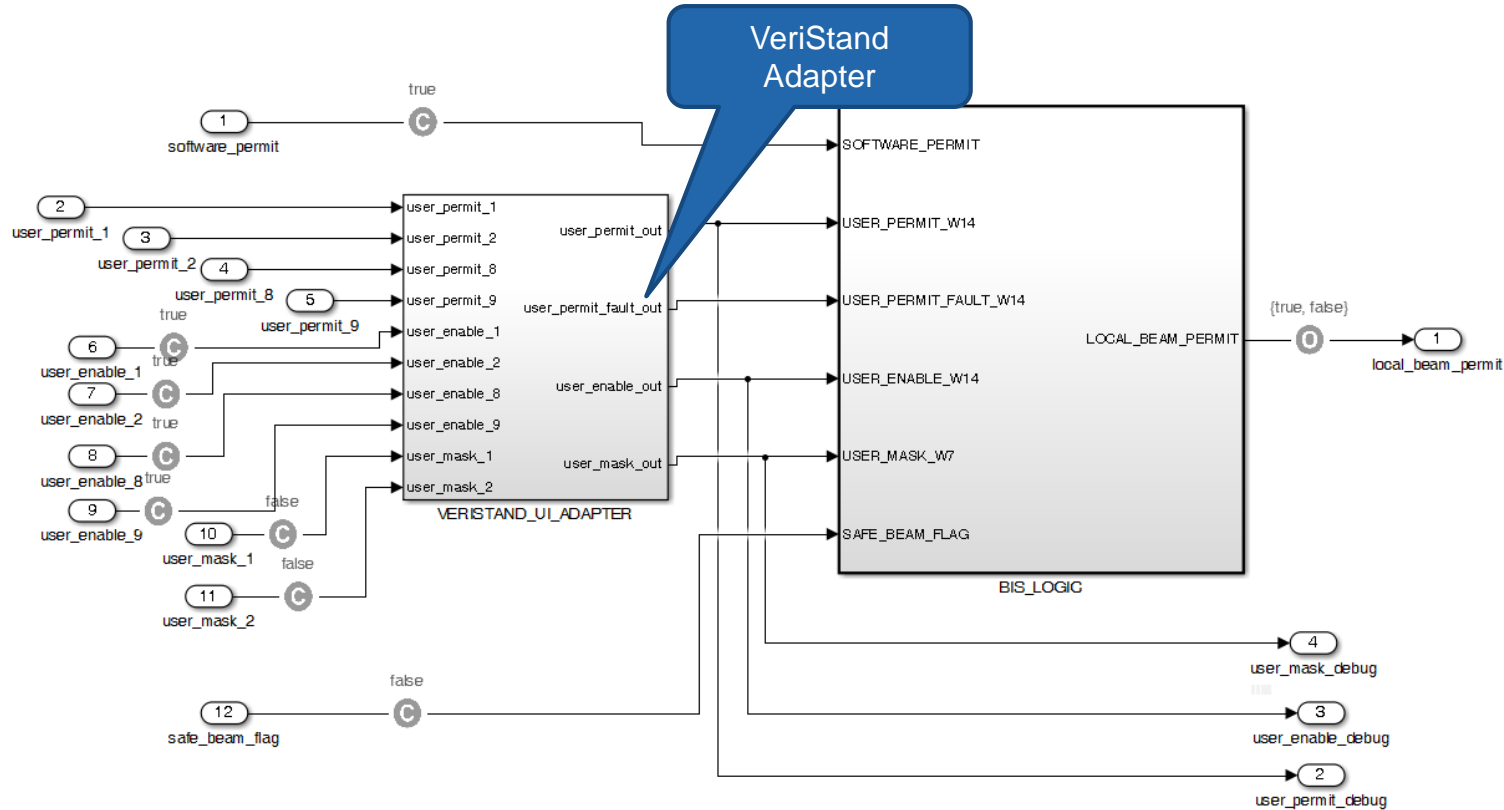
USER\_PERMIT8 A B

USER\_PERMIT9 A B

} Maskable

CIBM

## Simulink Model with VeriStand Adapter

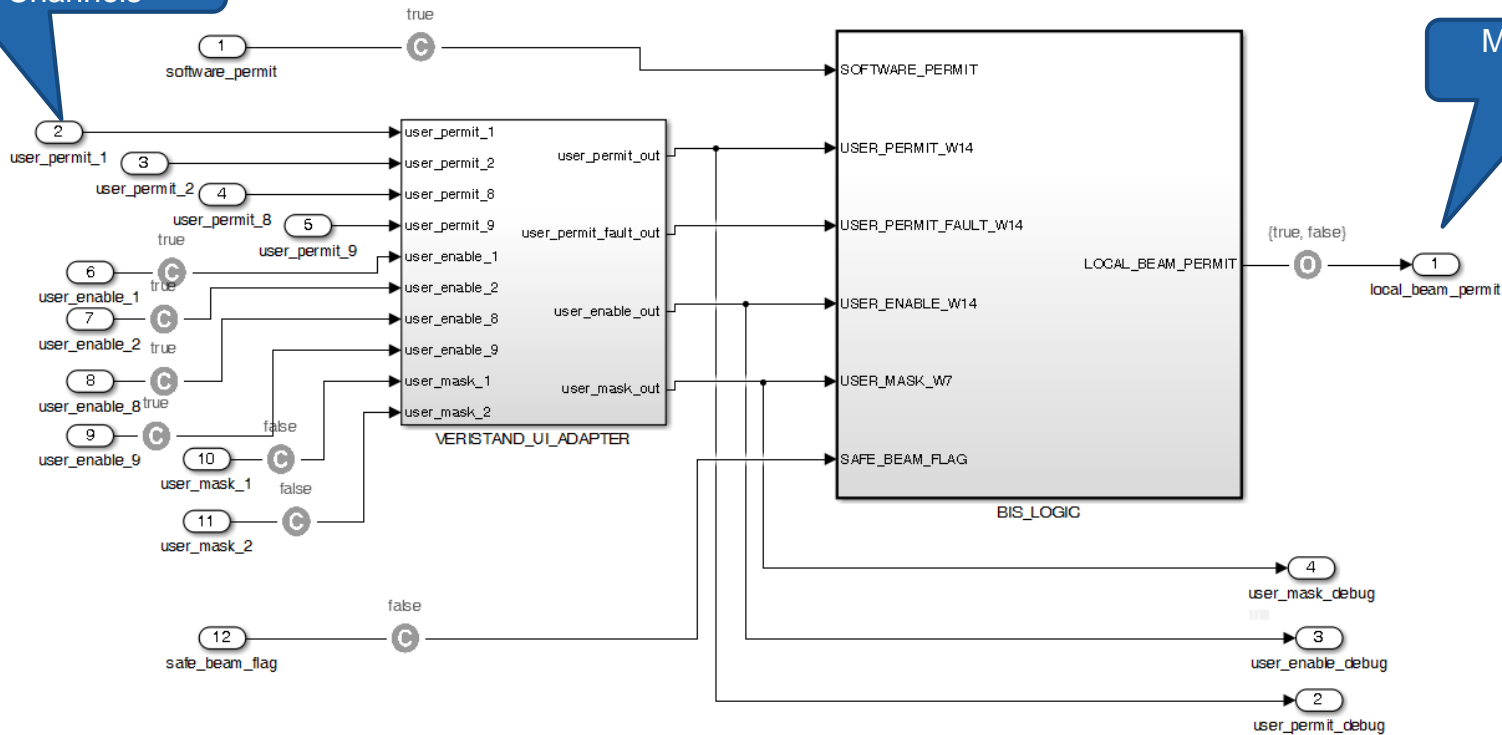


The VeriStand adapter will reflect the hardware setup.  
This helps to identify untestable logic for this hardware configuration.

# CERN BIS Case Study

## Simulink Model VeriStand Channel Interfaces

Model Input Channels

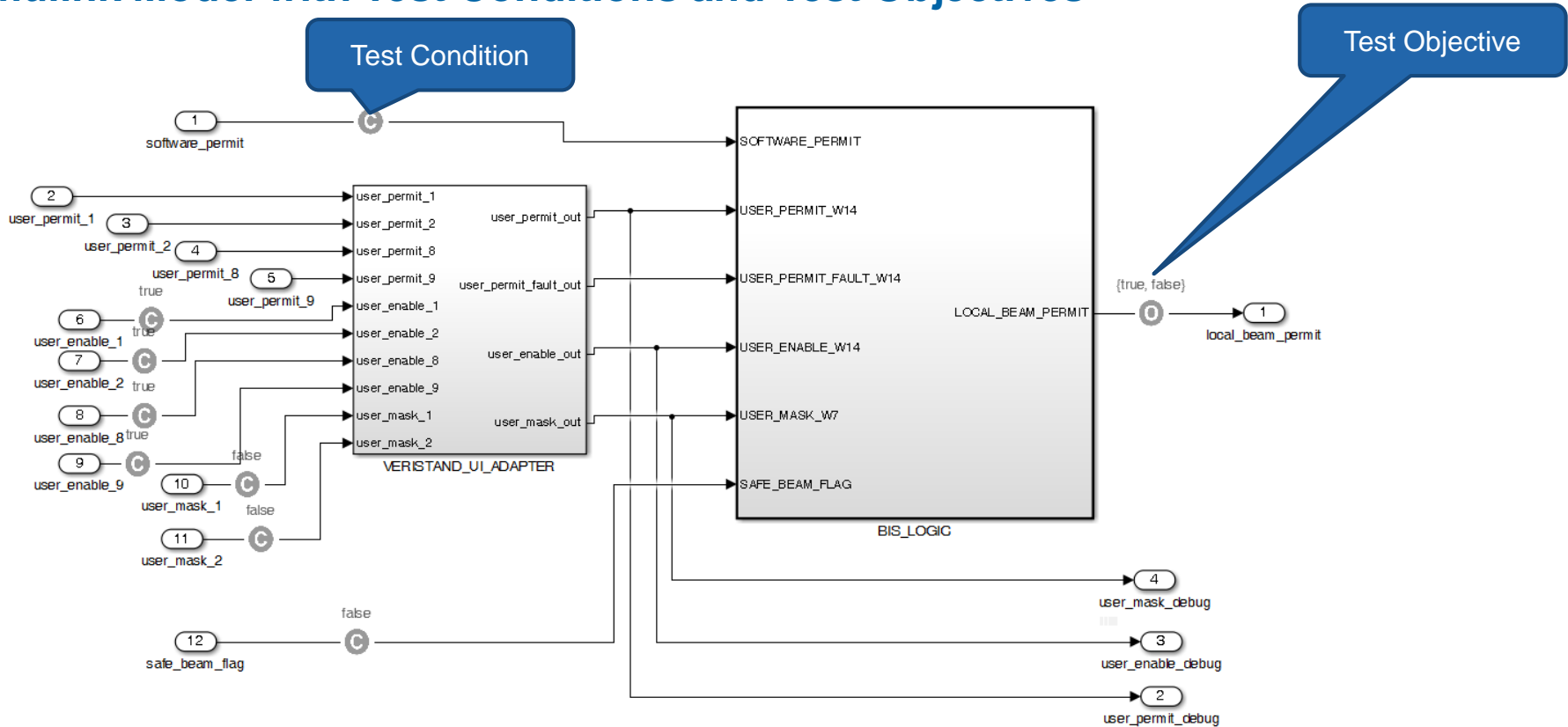


Model Output Channel

The input and output ports will be available in VeriStand's System Explorer. Use the Stimulus Profile Editor for hand written test cases.

# CERN BIS Case Study

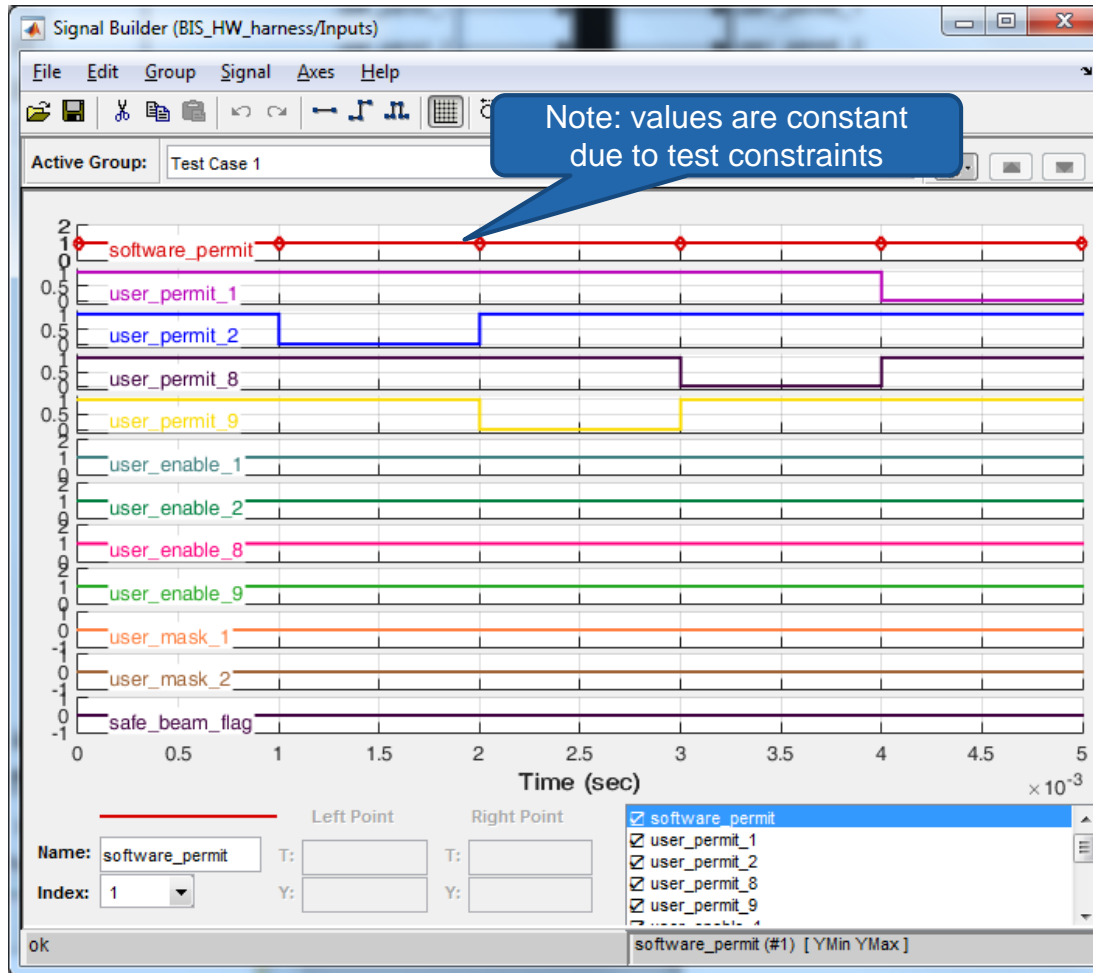
## Simulink Model with Test Conditions and Test Objectives



Test conditions are used to limit test cases to tests which can be performed with connected hardware without manual input or input via serial interface. Simulink Design Verifier used to generate test cases.

# CERN BIS Case Study

## SDV Generated Test Cases























































# CERN BIS Case Study

## Simulink Test Coverage Report

### Summary

Model Hierarchy/Complexity		D1		C1		Test 1	Test Condition	Test Objective
						MCDC		
1. <a href="#">BIS_HW</a>	162	30%		39%		14%		100%
2. ... <a href="#">BIS_LOGIC</a>	159	30%		39%		14%	NA	NA
3. .... <a href="#">DISABLE_BLOCK</a>	121	26%		21%		NA	NA	NA
4. .... <a href="#">DISABLE_1</a>	5	50%		75%		NA	NA	NA
5. .... <a href="#">DISABLE_10</a>	5	17%		0%		NA	NA	NA
6. .... <a href="#">DISABLE_11</a>	5	17%		0%		NA	NA	NA
7. .... <a href="#">DISABLE_12</a>	5	17%		0%		NA	NA	NA
8. .... <a href="#">DISABLE_13</a>	5	17%		0%		NA	NA	NA
9. .... <a href="#">DISABLE_14</a>	5	17%		0%		NA	NA	NA
10. .... <a href="#">DISABLE_2</a>	5	50%		75%		NA	NA	NA
11. .... <a href="#">DISABLE_3</a>	5	17%		0%		NA	NA	NA
12. .... <a href="#">DISABLE_4</a>	5	17%		0%		NA	NA	NA
13. .... <a href="#">DISABLE_5</a>	5	17%		0%		NA	NA	NA
14. .... <a href="#">DISABLE_6</a>	5	17%		0%		NA	NA	NA
15. .... <a href="#">DISABLE_7</a>	5	17%		0%		NA	NA	NA
16. .... <a href="#">DISABLE_8</a>	5	50%		75%		NA	NA	NA
17. .... <a href="#">DISABLE_9</a>	5	50%		75%		NA	NA	NA
18. .... <a href="#">MASK_1</a>	2	50%		50%		0%	NA	NA
19. .... <a href="#">MASK_2</a>	2	50%		50%		0%	NA	NA
20. .... <a href="#">MASK_3</a>	2	50%		50%		0%	NA	NA
21. .... <a href="#">MASK_4</a>	2	50%		50%		0%	NA	NA
22. .... <a href="#">MASK_5</a>	2	50%		50%		0%	NA	NA
23. .... <a href="#">MASK_6</a>	2	50%		50%		0%	NA	NA
24. .... <a href="#">MASK_7</a>	2	50%		50%		0%	NA	NA
25. .... <a href="#">MATRIX</a>	1	NA		63%		27%	NA	NA

# CERN BIS Case Study

## Simulink Test Coverage Report

MC/DC analysis (combinations in parentheses did not occur)

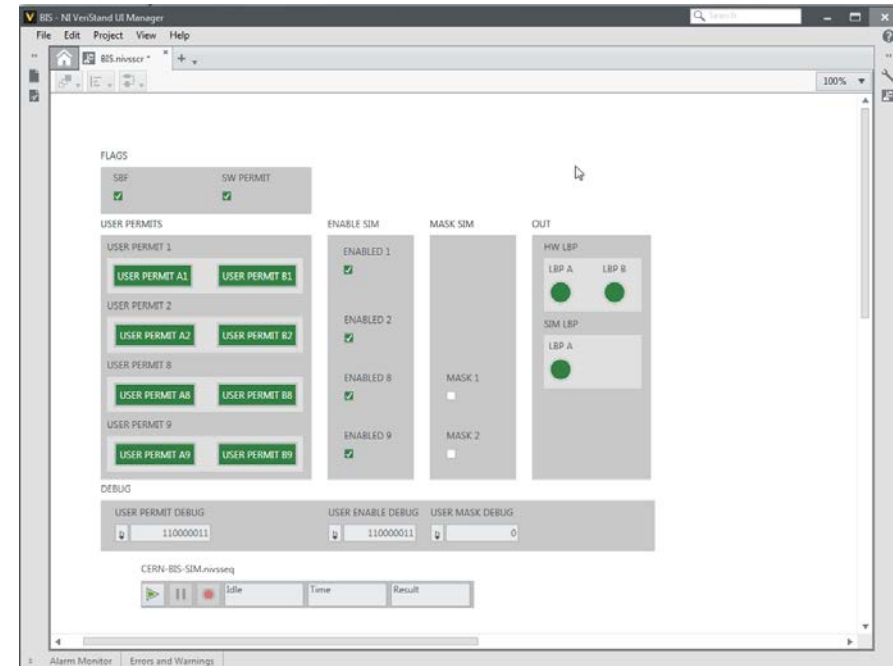
Decision/Condition	True Out	False Out
expression for output		
input port 1	TTTTTTTTTTTTTT	FTTTTTTTTTTTTT
input port 2	TTTTTTTTTTTTTT	TFTTTTTTTTTTTT
input port 3	TTTTTTTTTTTTTT	(TTFTTTTTTTTTTT)
input port 4	TTTTTTTTTTTTTT	(TTFTTTTTTTTTTT)
input port 5	TTTTTTTTTTTTTT	(TTFTTTTTTTTTTT)
input port 6	TTTTTTTTTTTTTT	(TTFTTTTTTTTTTT)
input port 7	TTTTTTTTTTTTTT	(TTFTTTTTTTTTTT)
input port 8	TTTTTTTTTTTTTT	TTTTTTTTFTTTTT
input port 9	TTTTTTTTTTTTTT	TTTTTTTTFTTTTT
input port 10	TTTTTTTTTTTTTT	(TTTTTTTTFTTTTT)
input port 11	TTTTTTTTTTTTTT	(TTTTTTTTFTTTTT)
input port 12	TTTTTTTTTTTTTT	(TTTTTTTTFTTTTT)
input port 13	TTTTTTTTTTTTTT	(TTTTTTTTFTTTTT)
input port 14	TTTTTTTTTTTTTT	(TTTTTTTTFTTTTT)
input port 15	TTTTTTTTTTTTTT	(TTTTTTTTFTTTTT)

## Testing of Logical Behavior

### Results

CERN BIS logical behavior is **identical** ✓  
to simulated behavior, when

- 1. Matrices have been initialized
- 2. Matrices have been rearmed
- 3. Software Permit has been set using nodal script via serial interface



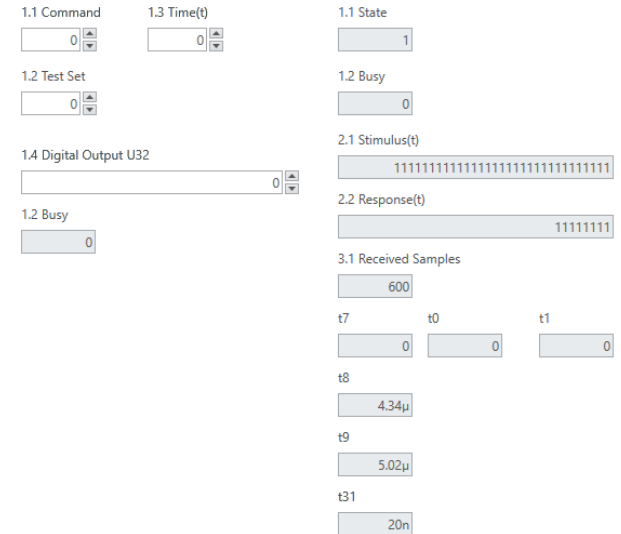
## Timing Measurement

### Measurement Configuration

- Sample Clock 50 MHz
- Resolution 20ns
- Trigger OK → NOK

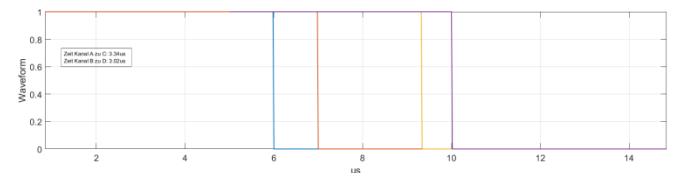
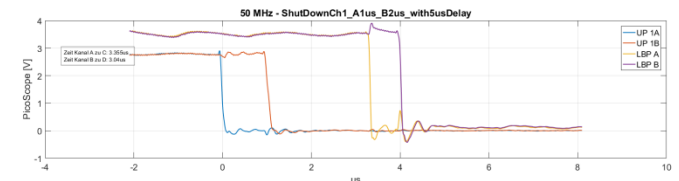
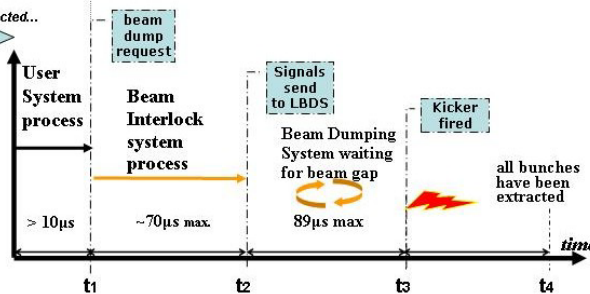
### Results

- 4.34μs for LBP A
- 5.02μs for LBP B



USER\_PERMIT signal changes from TRUE to FALSE

a failure has been detected...



## Testing

- Only few hardware ports are connected → low coverage
- Test automation must take into account serial interface commands → design to test in new FBIS implementation
- Current Simulink model implementation is limited to test functional behavior of a reactive system
- Concept required to include timing constraints in Simulink model

## Experiences

- Simulink design verifier can be used to generate test cases, best practice is to use a bottom up approach
- Test cases combined with verification scripts to verify Simulink functional blocks against specification
- Model integration in NI VeriStand is straight forward, but needs to reflect real hardware connections.



Contact:



Sven Stefan Krauss  
[svenstefan.krauss@zhaw.ch](mailto:svenstefan.krauss@zhaw.ch)



Martin Rejzek  
[martin.rejzek@zhaw.ch](mailto:martin.rejzek@zhaw.ch)



Christian Hilbes  
[christian.hilbes@zhaw.ch](mailto:christian.hilbes@zhaw.ch)

<http://www.iamp.zhaw.ch/sks>