Automatic Firmware Verification for Automotive Applications

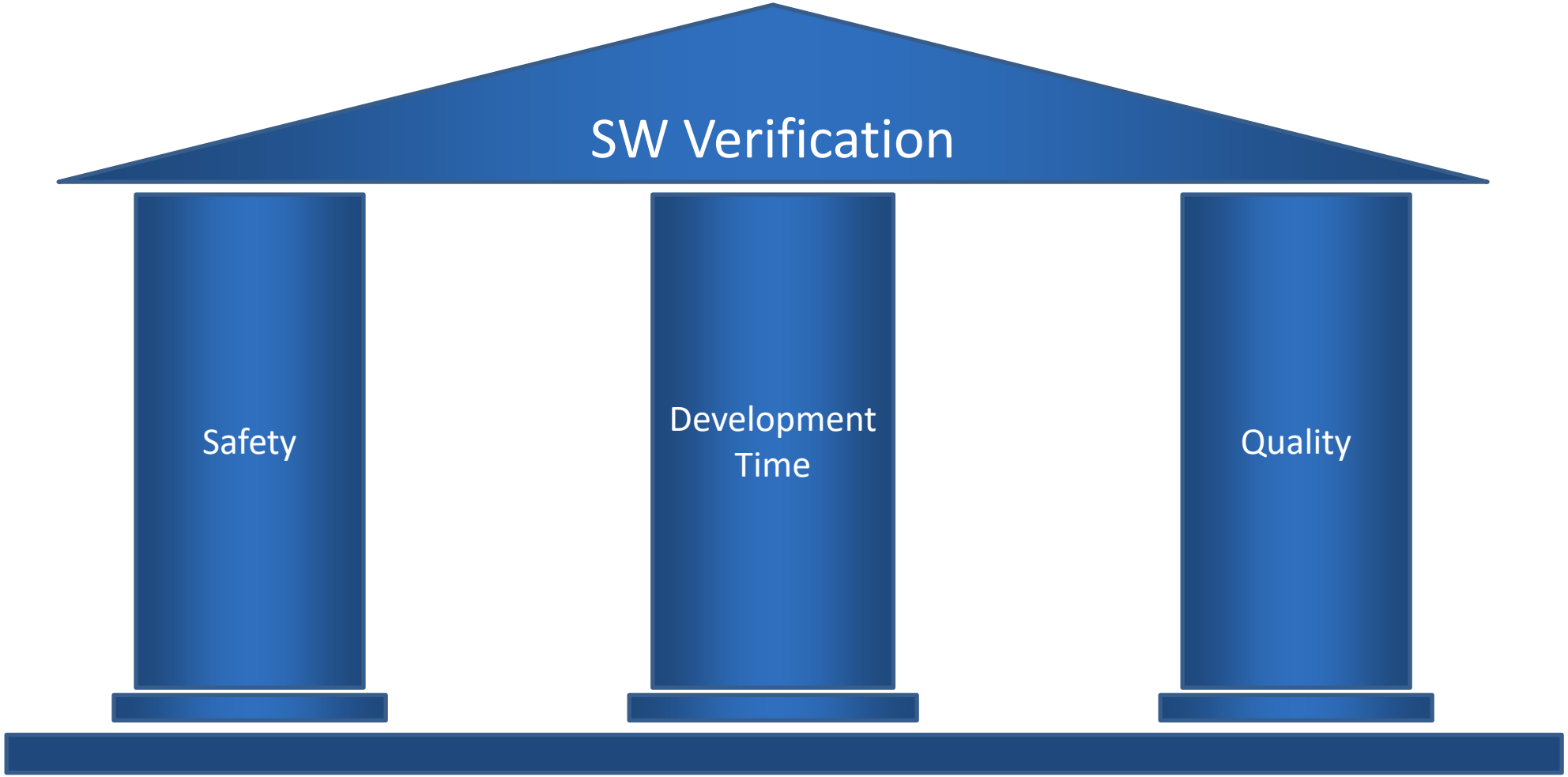
Dr. Torsten Andre, Daniel Valtiner

Infineon Technologies Austria AG

accellera SYSTEMS INITIATIVE

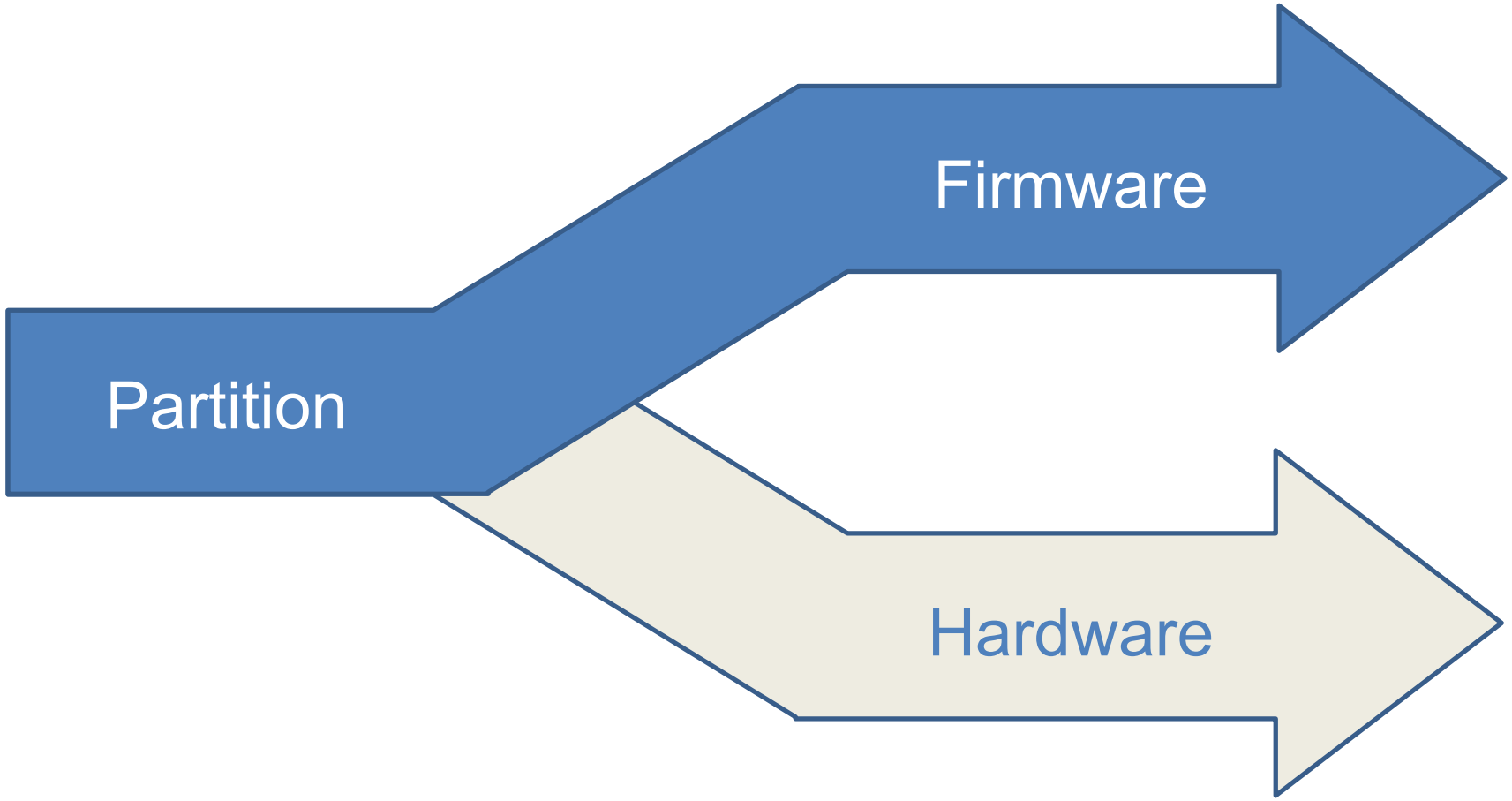
2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Motivation



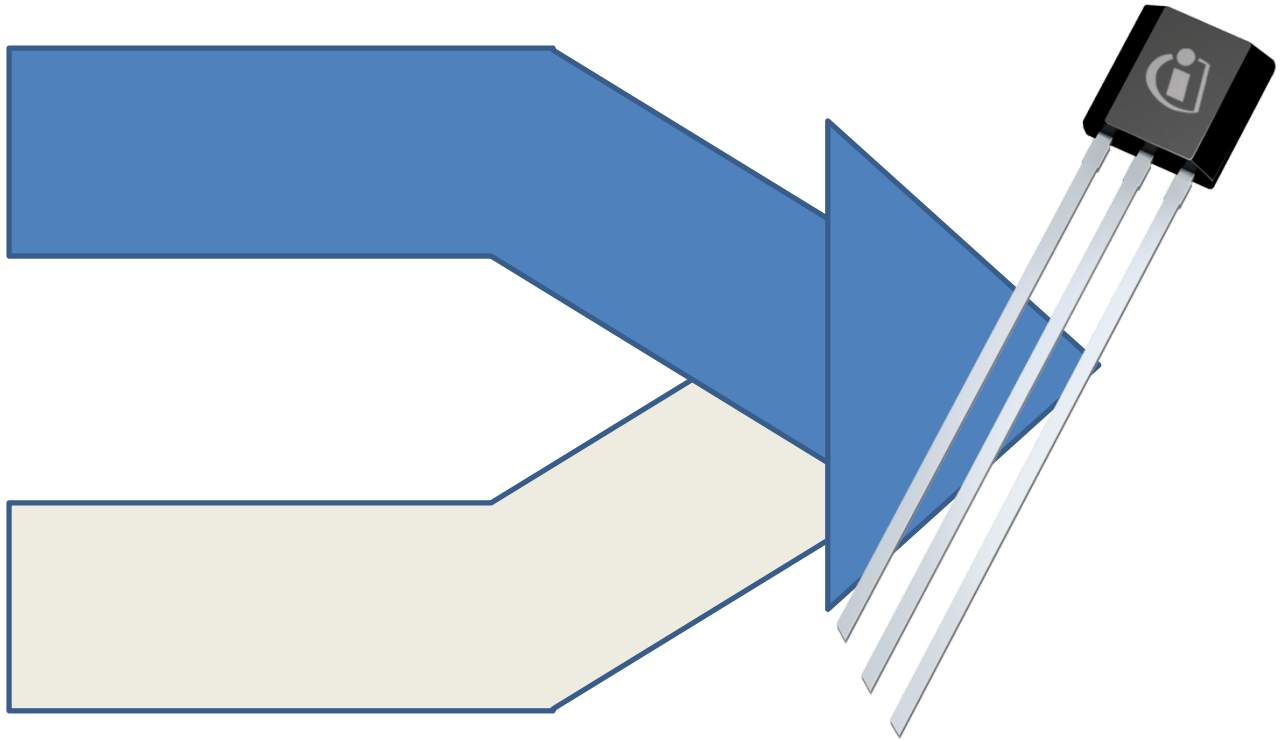
© Accellera Systems Initiative

Development Time



© Accellera Systems Initiative

Parallel Development



- **Parallel** development
- **Influencing** development
  - Confirm feasibility of partitioning
  - React quickly if unfeasible

© Accellera Systems Initiative

Quality of Verification

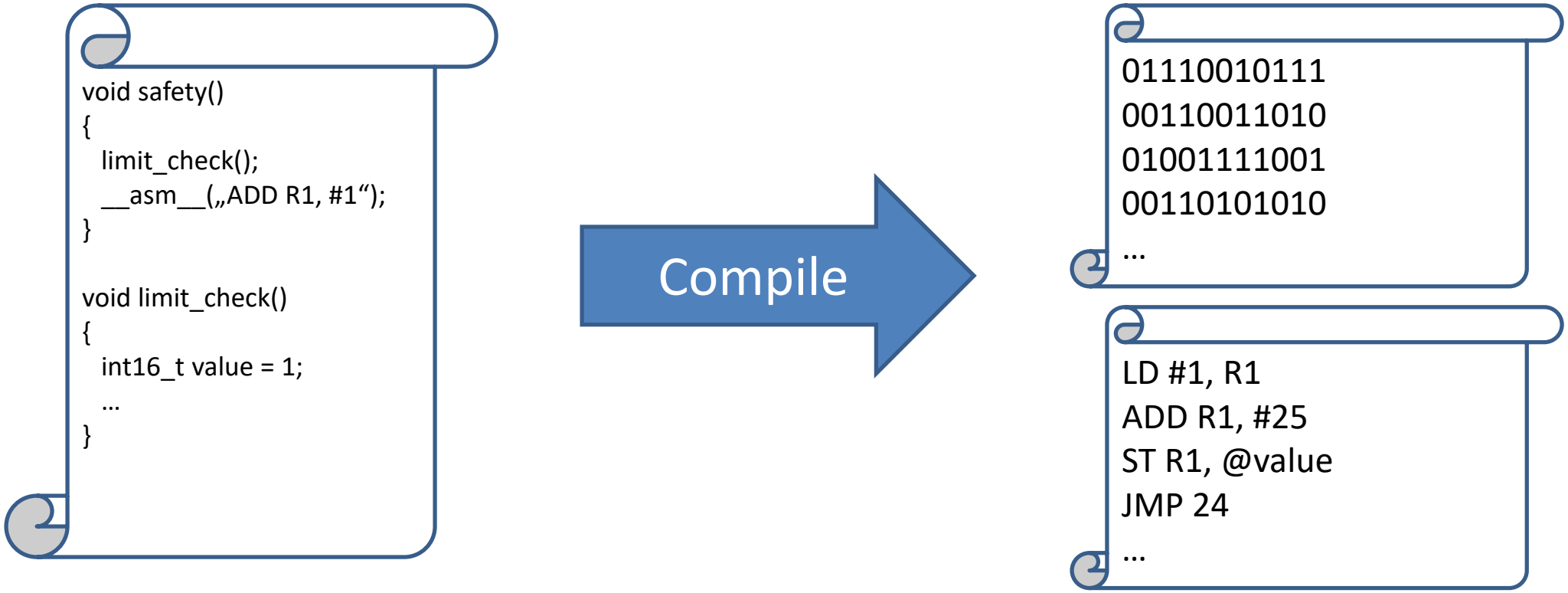
- Functional requirements
  - Behavior
  - Non-behavior
- Non-functional requirements
  - Resource constraints (e.g. timing, memory size)

✓ Fault tolerant time interval (FTTI)

✓ Timing Limitations
- Processing delay
- Acceptable jitter
- Refresh rates

© Accellera Systems Initiative

Host vs. Target Verification



© Accellera Systems Initiative

ISO 26262

ISO 26262-6:2011 9.4.6

"The test environment for software **unit testing** shall correspond as closely as possible to the target environment. […]"

ISO 26262-6:2011 10.4.8

"The test environment for software **integration testing** shall correspond as closely as possible to the target environment. […]"

accellera SYSTEMS INITIATIVE

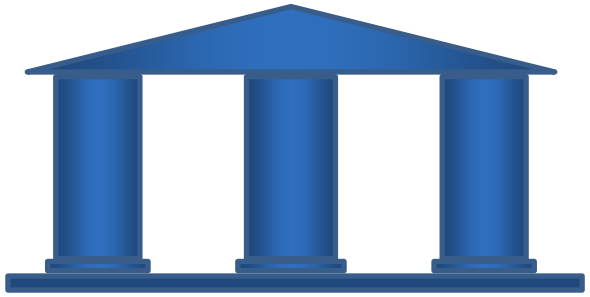
© Accellera Systems Initiative

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Implications in a Nutshell

Pro's Host Verification

- Many tools to choose from
- Verify C code



Pro's Target Verification

- Enable parallel development
- Non-functional requirement verification
- Supports verification of native assembler statements
- ISO 26262
- Verify machine code (do not depend on compiler)

© Accellera Systems Initiative

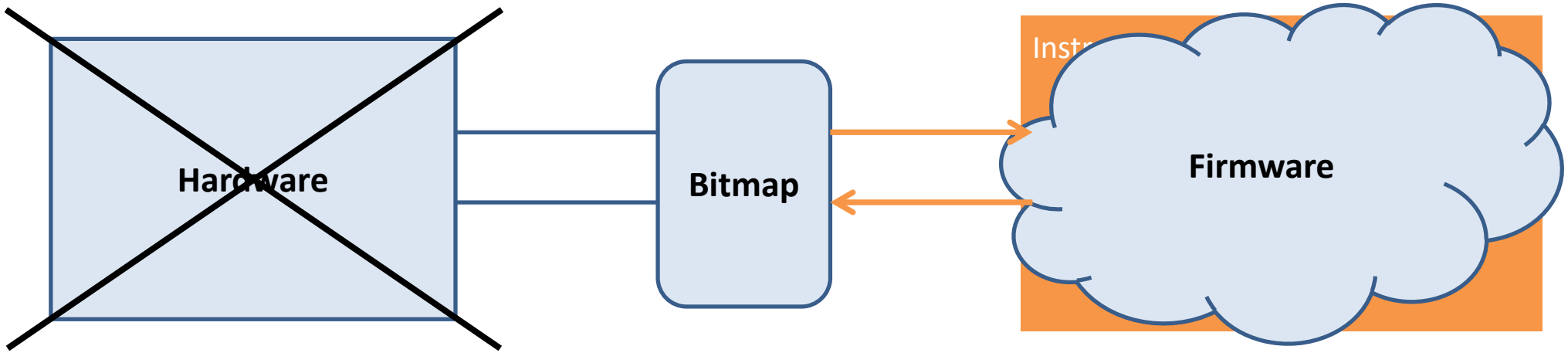
Evaluation Criteria

- Comprehensive verification
- Easy to use
- Low overhead
- Fast execution
- Support regression
- FW Unit verification
- FW Unit Integration verification
- Independent of hardware development
- High level of reuse
- Reliability tooling
- Simple result interpretation

© Accellera Systems Initiative

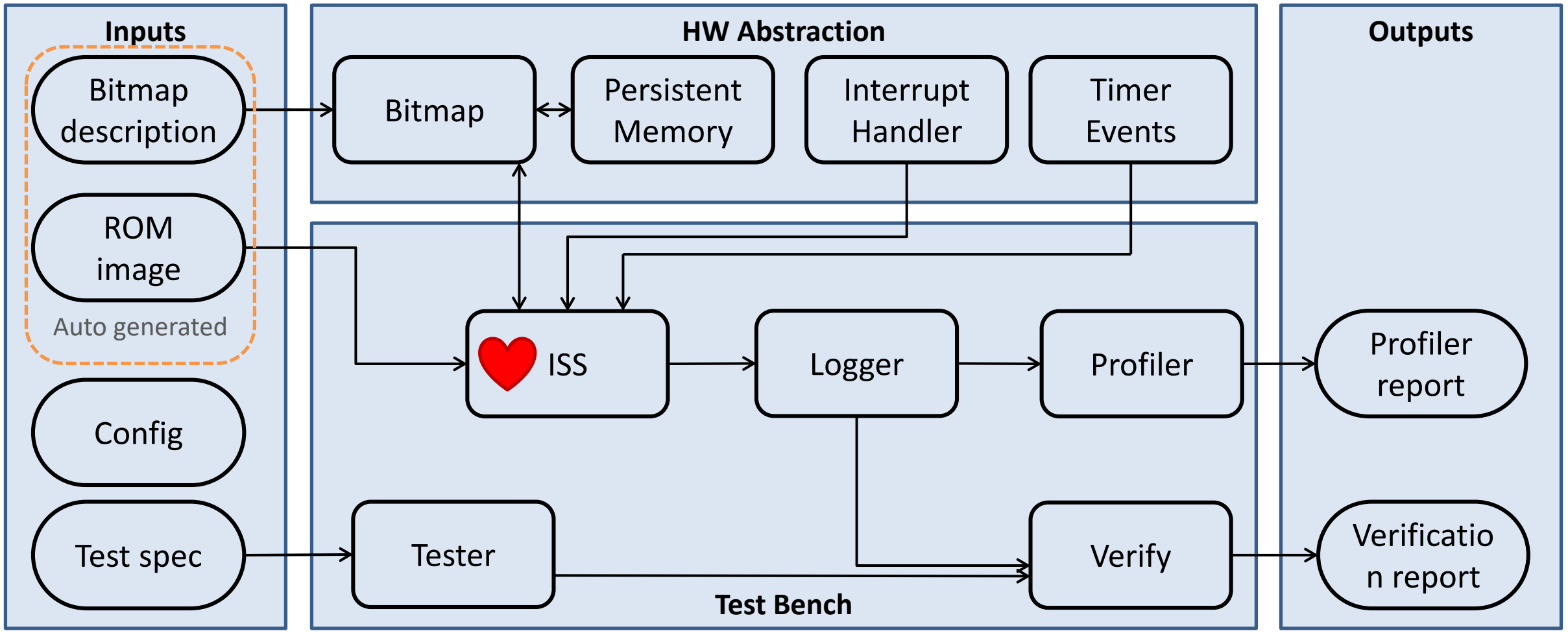
Hardware Abstraction

- **Bitmap:** All registers accessible by HW and FW



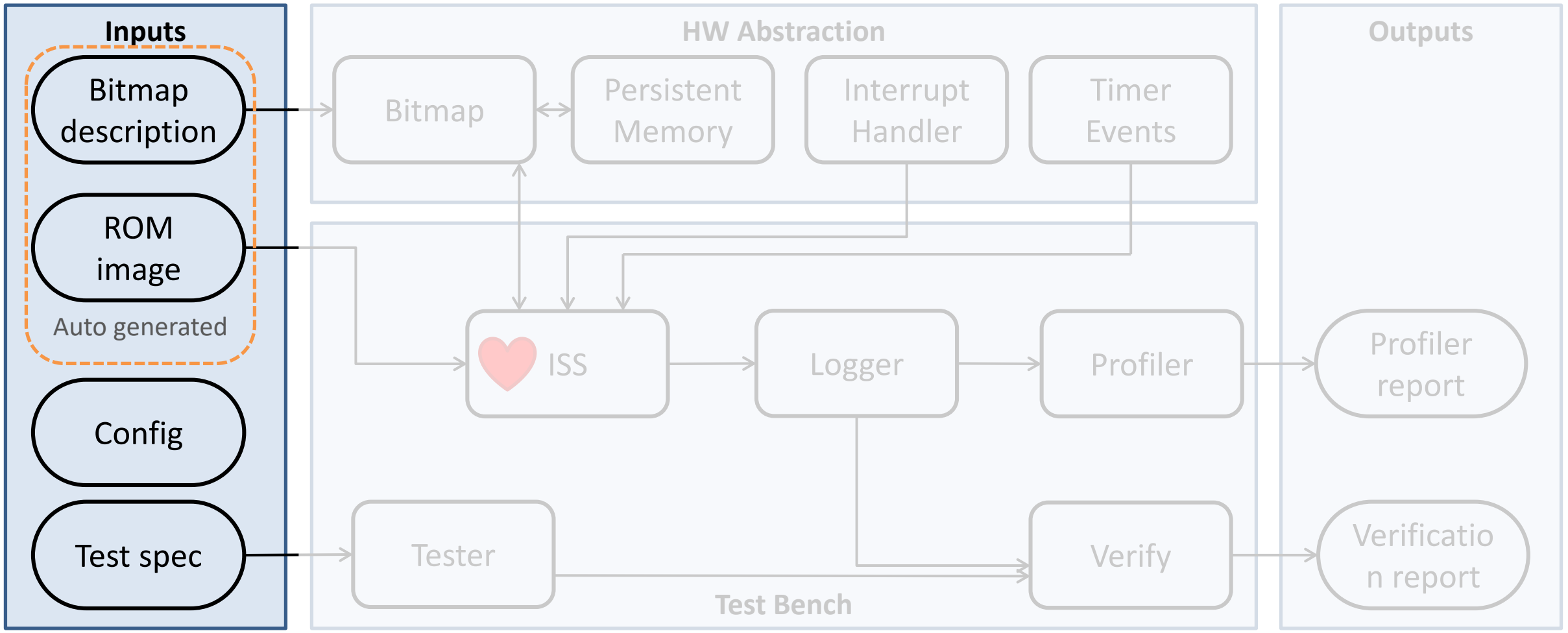
© Accellera Systems Initiative

Verification Tool Architecture



© Accellera Systems Initiative

Verification Tool Architecture



© Accellera Systems Initiative

Inputs

- **Bitmap Description**
  - Register specification
  - Auto-generated from central specification
- **ROM image**
  - Binary FW image
  - Compiled from high level language
- **Config**
  - Project specific adaptation
  - Specified once per project

© Accellera Systems Initiative

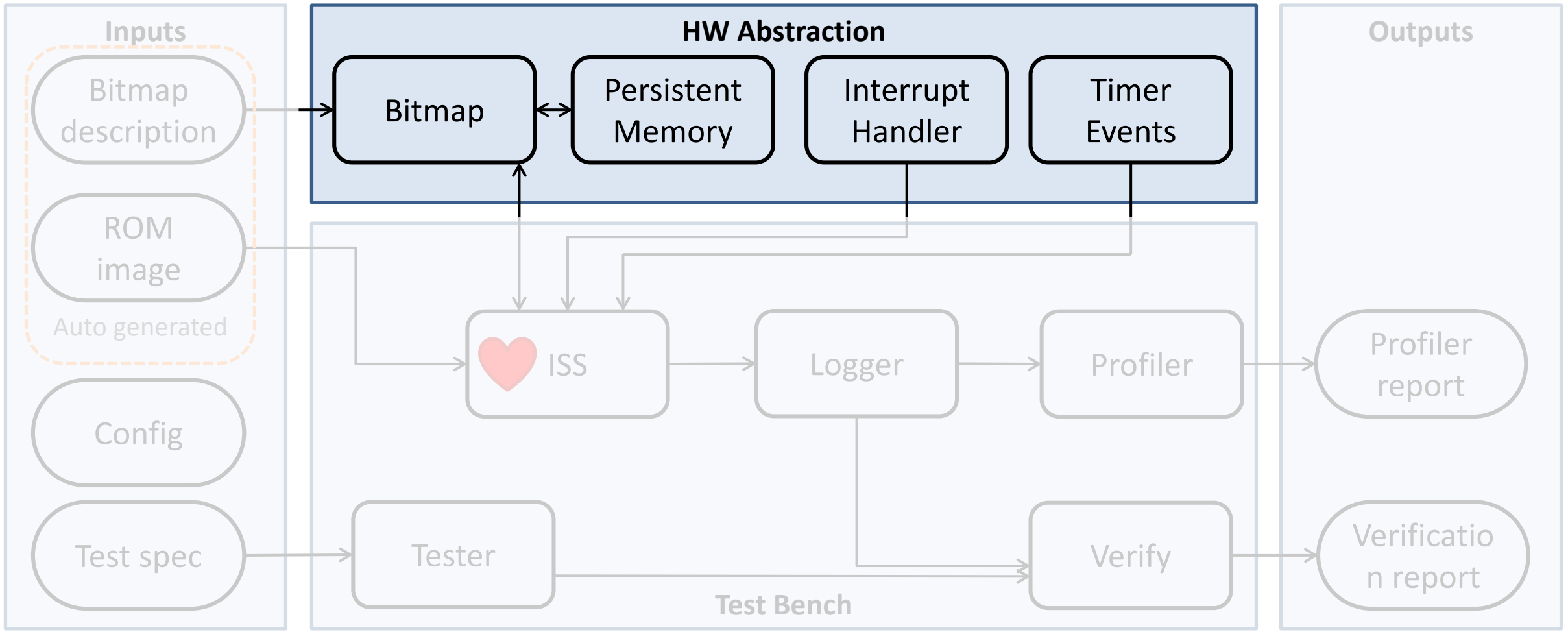
Test Specification

- Input stimuli
- Expected values
- Test definition in Microsoft Excel
  - Well-known tool
  - Reliable libraries available to enable parsing by framework
  - Reference implementation using Excel formulas
  - CSV import from other tools for reference implementation (e.g. Matlab)

	A	B	C	D
1	Inputs			Expected Output
2	algo_value	Lower limit	Upper limit	Limit error
3	-15	-25	25	0
4	-35	-25	25	1
5	…	…	…	…

© Accellera Systems Initiative

Verification Framework Architecture



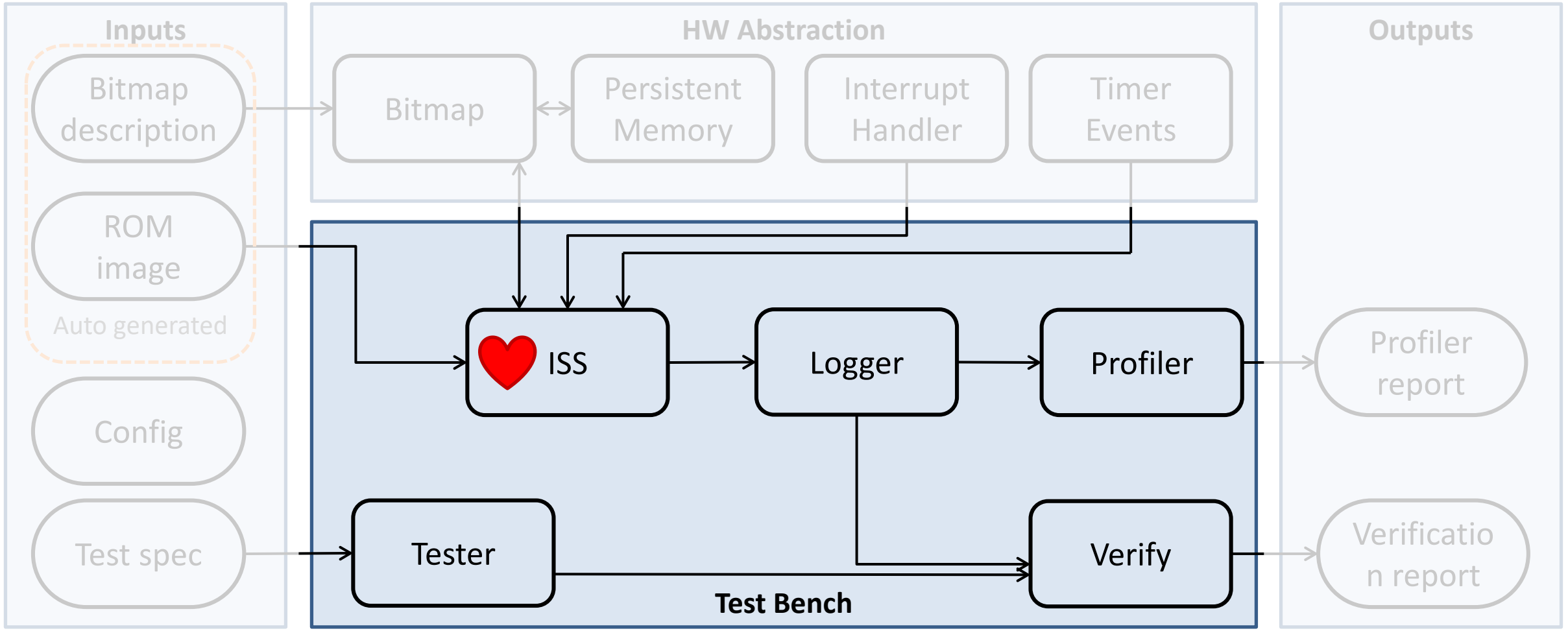
© Accellera Systems Initiative

HW Abstraction

- **Bitmap**
  - Automatically generated from bitmap specification
- **Persistent memory**
  - Models specific behavior to retrieve data, e.g. EEPROM paging
- **Interrupt/Event handler**
  - Register behavior based on conditions/time

© Accellera Systems Initiative

Verification Tool Architecture



© Accellera Systems Initiative

Test Bench

- **Instruction Set Simulator (ISS)**
  - Allows execution of native FW image
  - Cycle accurate
- **Tester**
  - User defined
  - Maps input stimuli to FW units

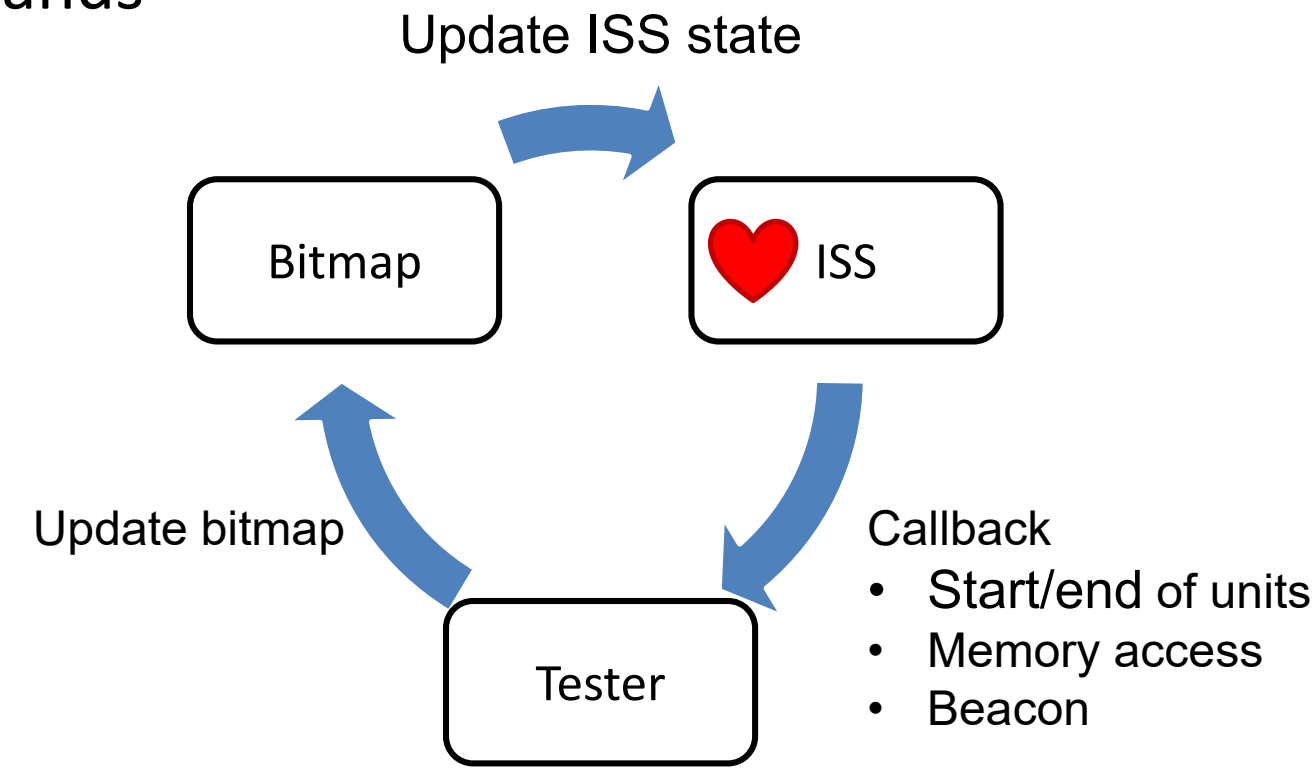
© Accellera Systems Initiative

Instruction Set Simulator (ISS)

- FW enriched with triggers/commands

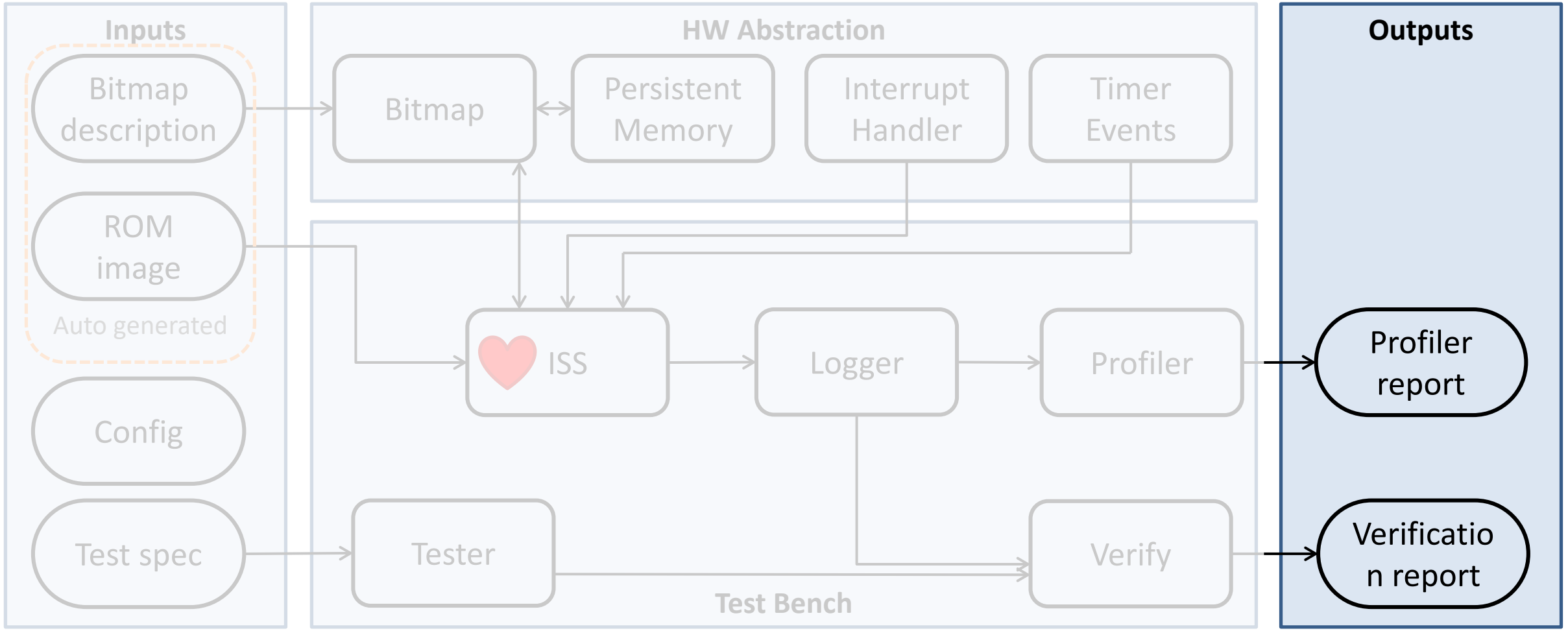
```
void safety()
{
  VERIFICATION_CMD(SAFETY_START);
  limit_check();
  regbist();
}

Void limit_check()
{
  VERIFICATION_CMD(LIMIT_CHK_START);
  int16_t a = GET(EEP1_LOWER_LIMIT);
  …
}
```



© Accellera Systems Initiative

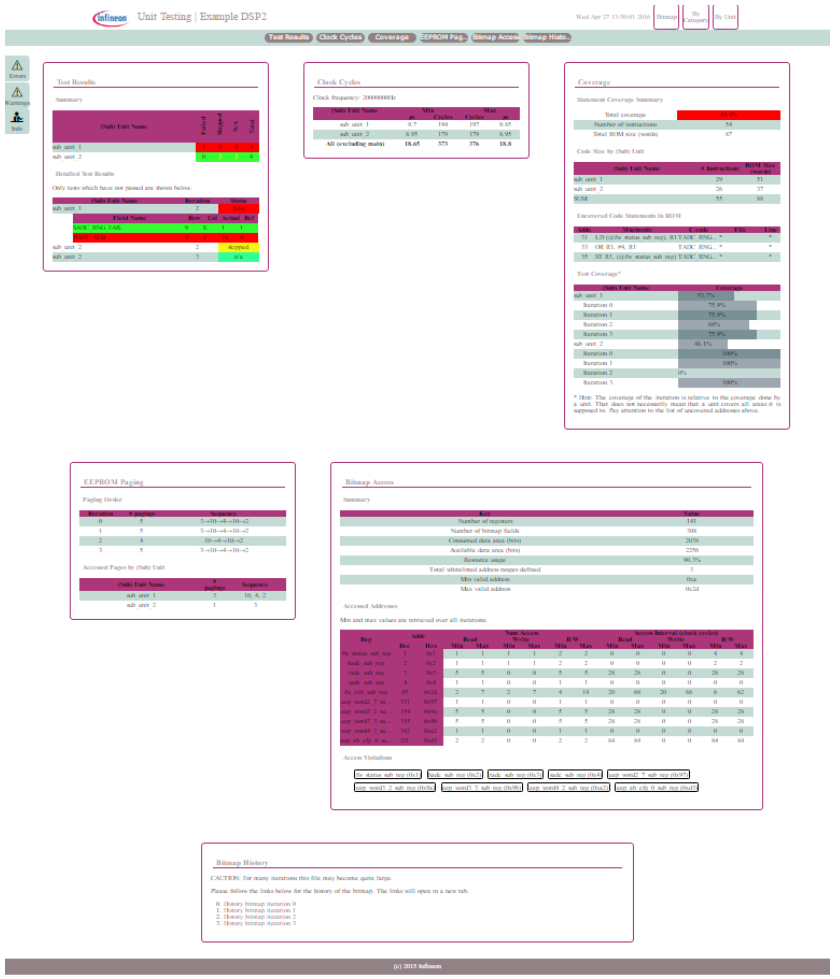
Verification Tool Architecture



© Accellera Systems Initiative

Outputs

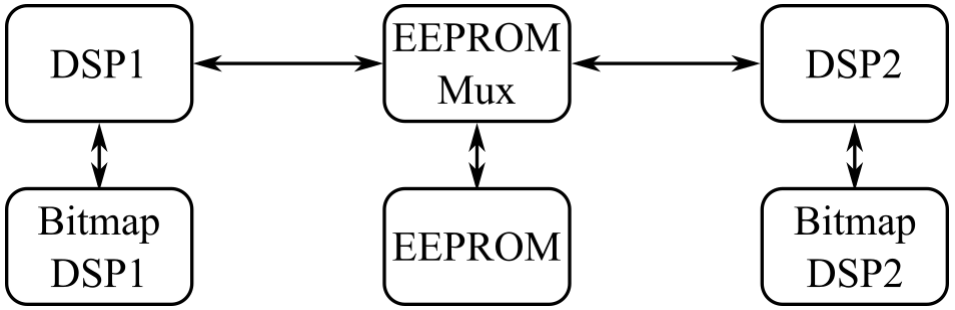
- **Profiler Report**
  - Statement coverage
  - Branch coverage
- **HTML Verification Report**
  - Test summary (skipped, pass, fail)
  - Profiling information (# opcodes, execution time)
  - Access order to persistent memory
  - Bitmap access (when, how often, min/max interval)
  - Memory access violations

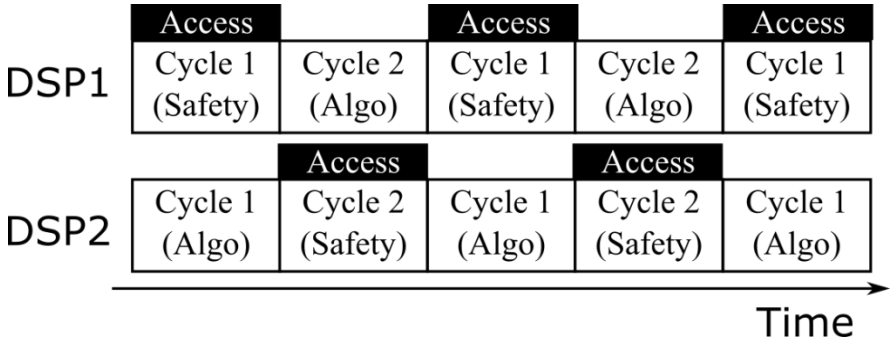


© Accellera Systems Initiative

Application Example

- Application example in the paper
- Two units (algo, safety)





© Accellera Systems Initiative

Evaluation Criteria

- ✓ Comprehensive verification
- Easy to use
- ✓ Low overhead
- ✓ Fast execution
- ✓ Support regression
- ✓ FW Unit verification
- ✓ FW Unit Integration verification
- ✓ Independent of hardware development
- ✓ High level of reuse
- ✓ Reliability tooling
- ✓ Simple result interpretation

© Accellera Systems Initiative

Conclusions

- Verification tool for target architecture
- Safe and high quality verification
- Fast and parallel development

Questions

© Accellera Systems Initiative