hidICE Introduction

Accemic GmbH & Co. KG

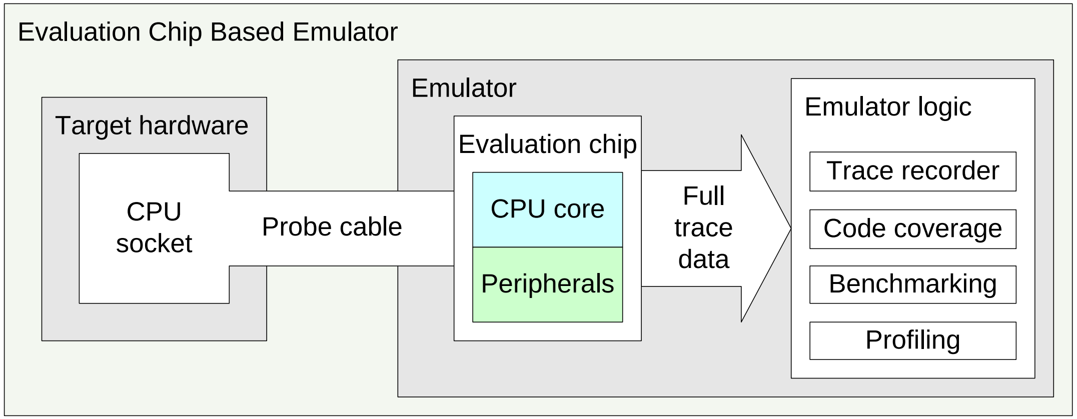
Agenda

1. **Today’s Trace Data Capturing Solutions**
2. hidICE – Capturing Full AND Continuous AND Real-time Trace
3. Comparison: Traditional Approaches vs. hidICE

Requests to Trace Data Capturing Solutions

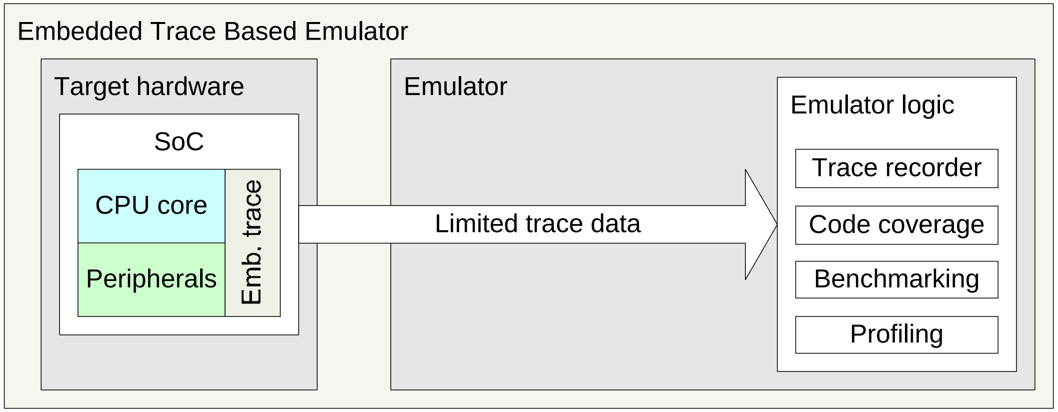
	FULL TRACE	▪ Executed instructions ▪ Data read access ▪ Data write access ▪ CPU registers (e.g. stack pointer) ▪ Bus cycles ▪ Cache activity ▪ Time stamps
AND	CONTINUOUS TRACE	▪ No CPU stop on full trace buffer
AND	REAL TIME OPERATION	▪ CPU runs at full speed
AND	TRACE FROM PRODUCTION SILICON	▪ Software test and certification on the same silicon as used in mass production
AND	MULTI CORE SUPPORT	▪ Support of multi core microcontroller

Evaluation Chip Based In-Circuit Emulator



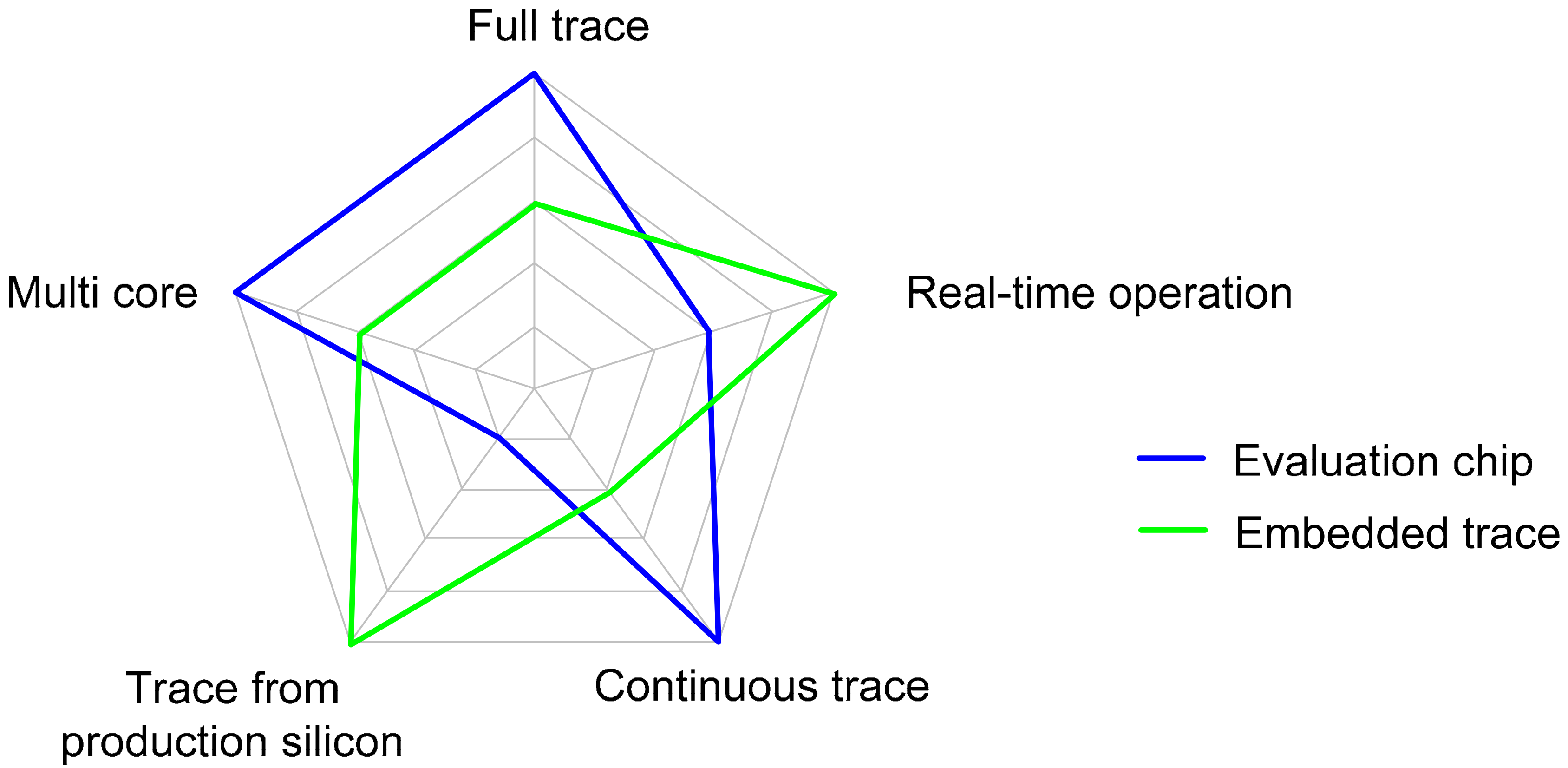
FULL TRACE	YES (depending on implementation)
CONTINUOUS TRACE	YES
REAL TIME OPERATION	NO (speed limitations on external bus)
PRODUCTION SILICON	NO
MULTI CORE SUPPORT	YES (depending on implementation)

Embedded Trace Based In-Circuit Emulator



FULL TRACE	LIMITED (bandwidth, trace buffer size)
CONTINUOUS TRACE	LIMITED (bandwidth)
REAL TIME TRACE	YES
TRACE FROM PRODUCTION SILICON	YES
MULTI CORE SUPPORT	LIMITED (bandwidth, trace buffer size)

Limitations of Traditional Approaches



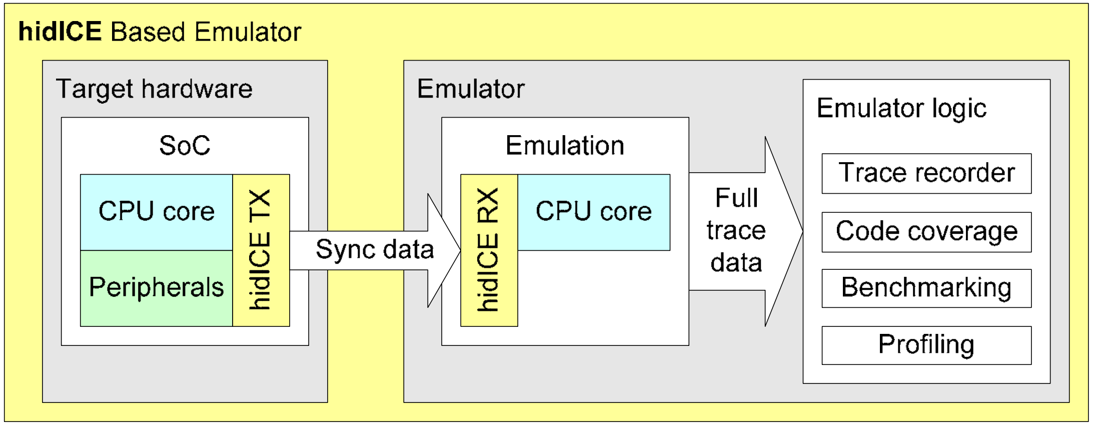
accelerate your microcontroller design

Agenda

1. Today’s Trace Data Capturing Solutions
2. **hidICE – Capturing Full AND Continuous AND Real-time Trace**
3. Comparison: Traditional Approaches vs. hidICE

The hidICE Approach

hidICE
Accemic hidden ICE technology



Key idea:

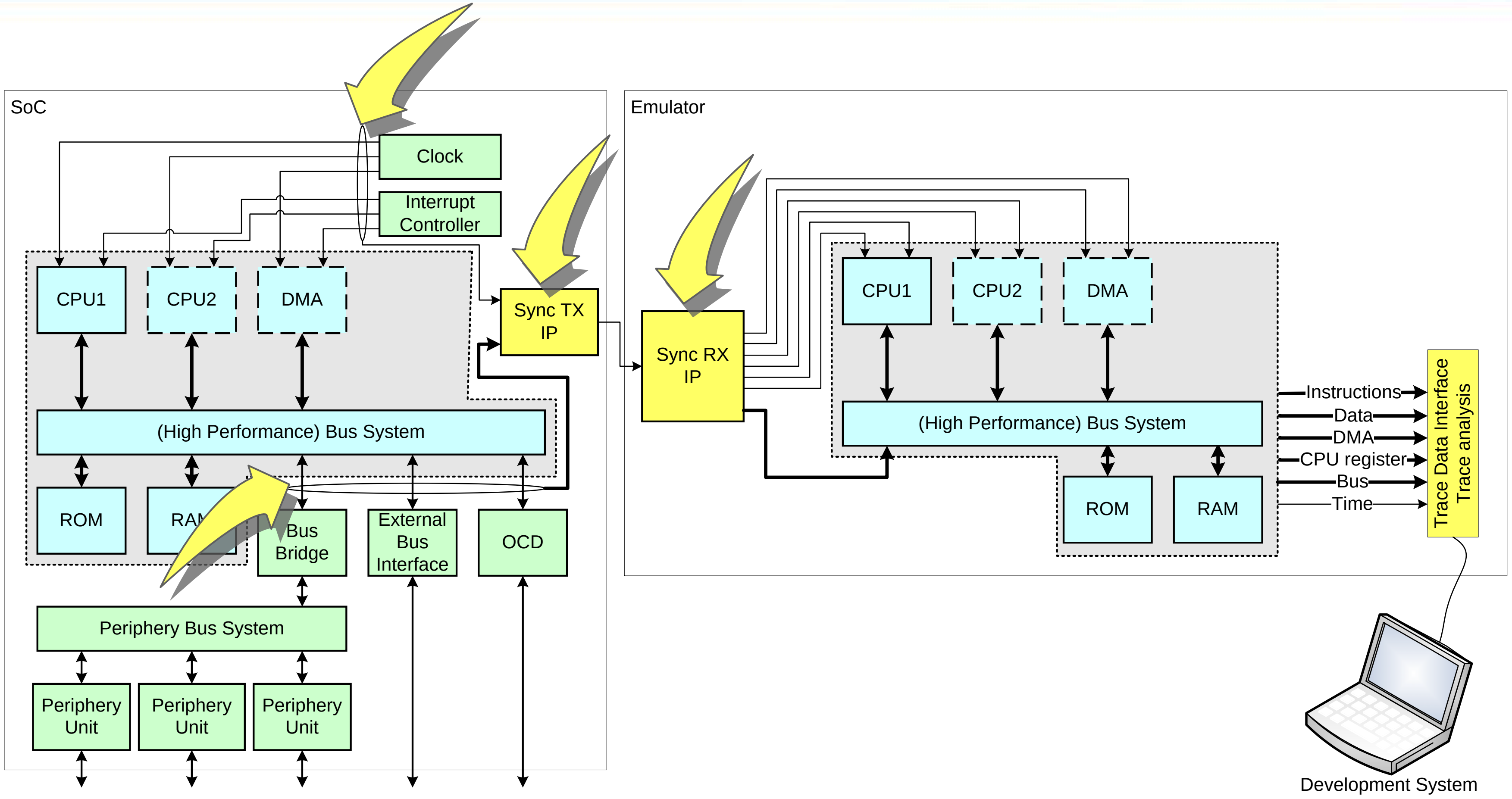
- Synchronization of a CPU subsystem inside the emulator

Requirements:

- Emulation must receive clock signals
- Emulation must receive the result of read operations
- Emulation must receive events (interrupts, DMA requests, wait states)
- Emulation must contain all bus masters
- Emulation must contain a superset of the memory

The emulation does not need to implement peripherals.

hidICE – Typical System Configuration



Synchronization

Signals to transmit

Serialization

Deserialization

hidICE – System Integrity Control

Will emulation and target always exhibit the same behavior?

We need to make sure!

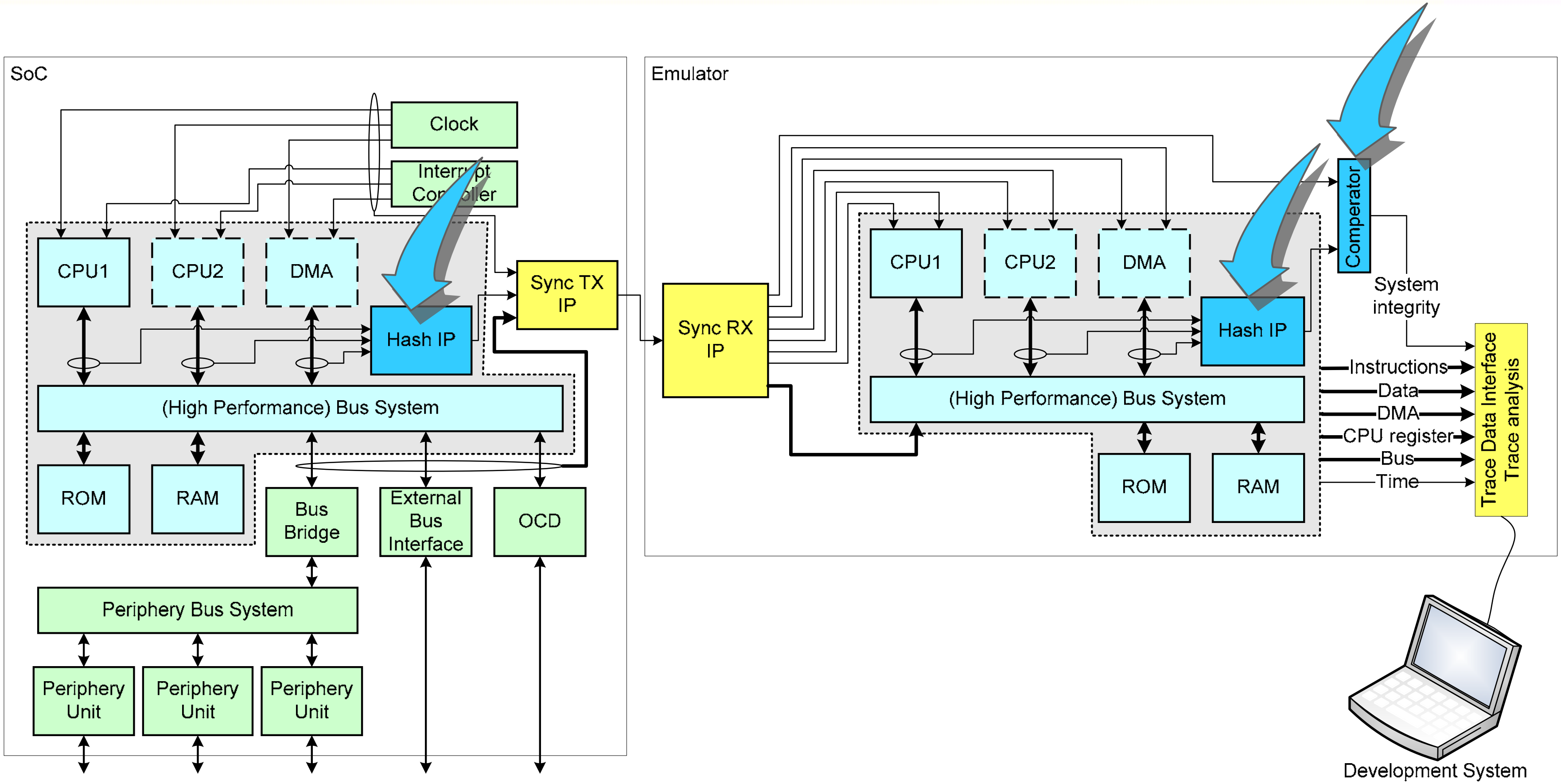
- Compute recursive hash over all relevant internal signals
- Transmit hash to emulation
- Compute hash locally in the emulation

Target hash and emulation hash differ

- Synchronization lost
- Further investigation required
  (and possible as trace is available)

accelerate your microcontroller design

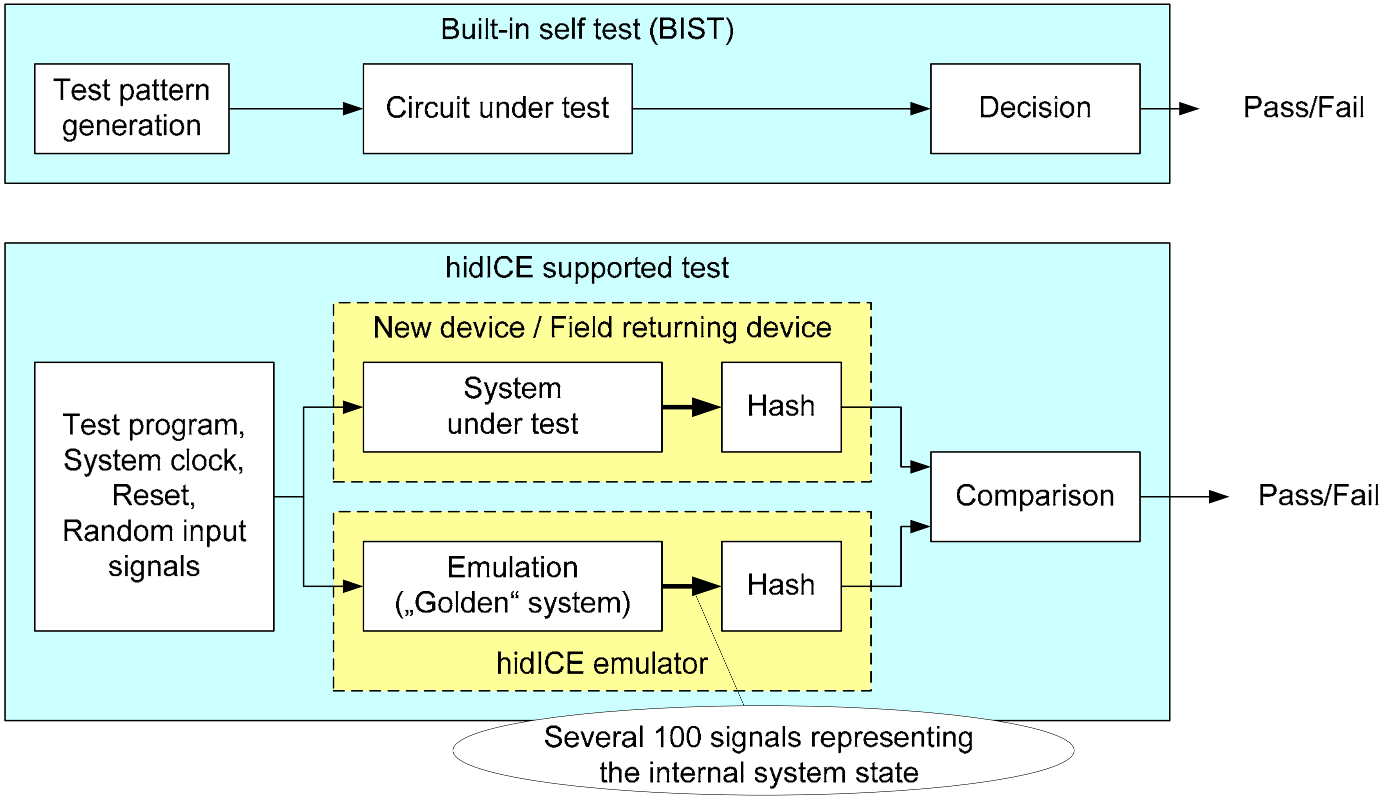
hidICE – System Integrity Control



Synchronization			System Integrity Control	
Signals to transmit	Serialization	Deserialization	Hash calculation	Hash check

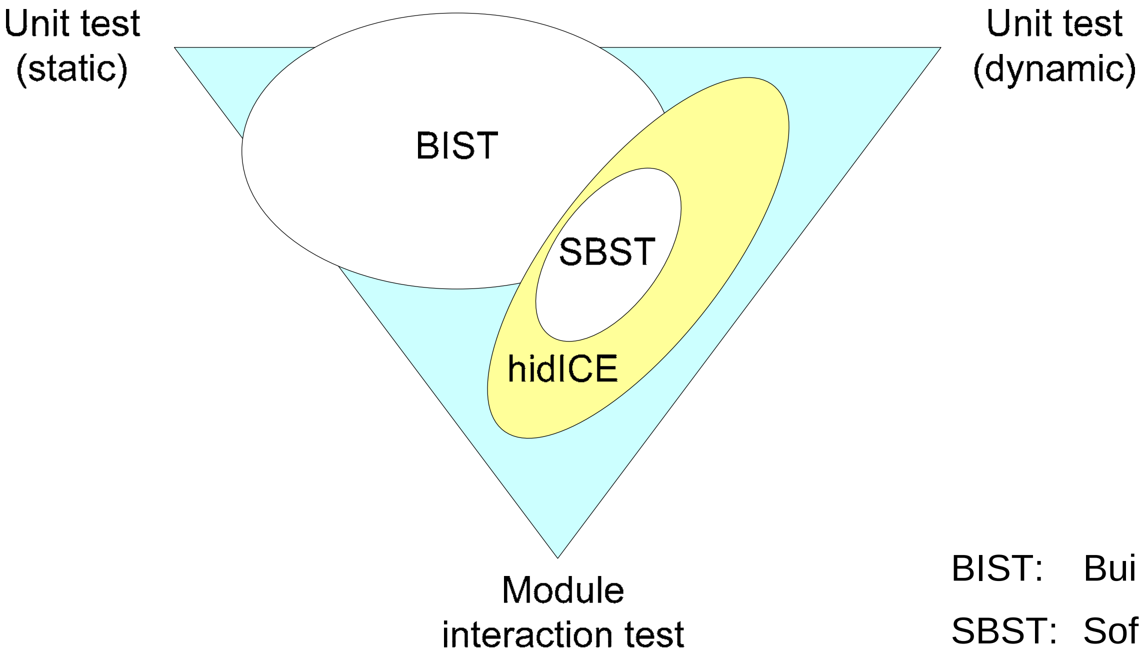
accelerate your microcontroller design

hidICE – System Integrity Control


Built-in self test (BIST)
Test pattern generation
Circuit under test
Decision
Pass/Fail
hidICE supported test
New device / Field returning device
Test program, System clock, Reset, Random input signals
System under test
Hash
Comparison
Pass/Fail
Emulation („Golden“ system)
Hash
hidICE emulator
Several 100 signals representing the internal system state

- Post manufacturing test
- Test of field-returning devices

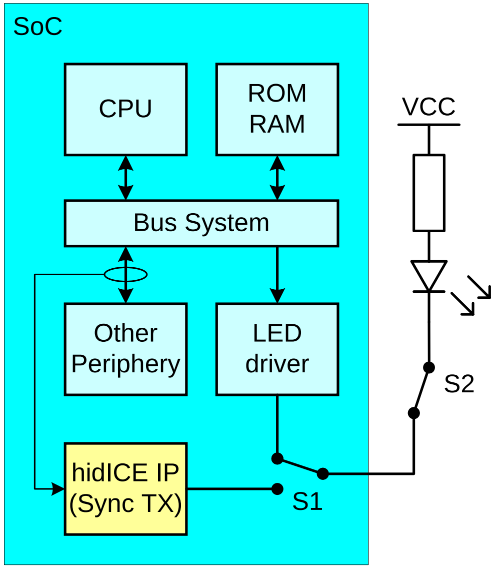
hidICE – System Integrity Control

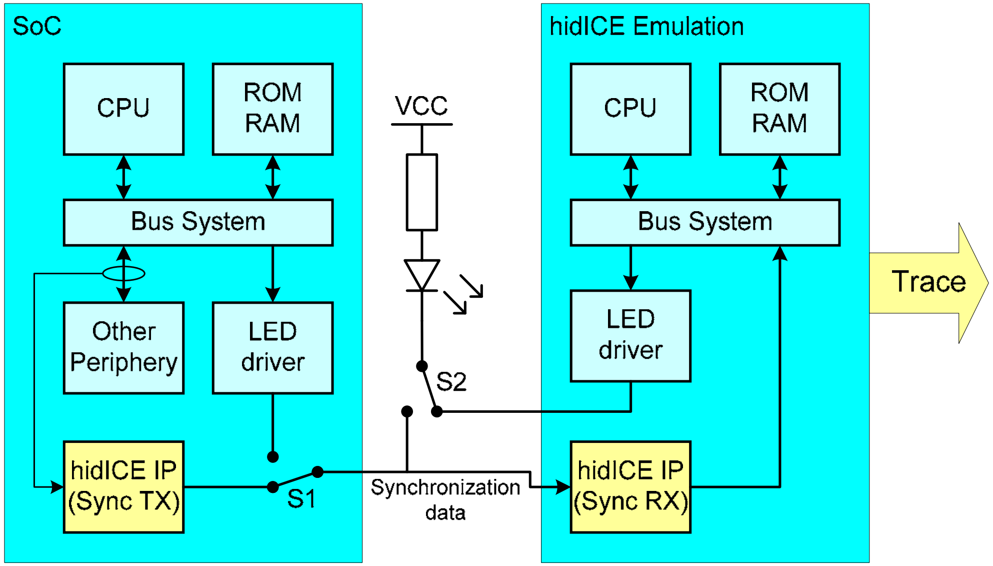


BIST:	Built-in self test
SBST:	Software based self test

Using system integrity control to increase the test coverage

hidICE – Port Reconstruction





„Normal“ mode

The port pins drives the LEDs.

“hidICE” mode

All pins are available for the application, even during recording of trace data.

Applicable for low speed output pins

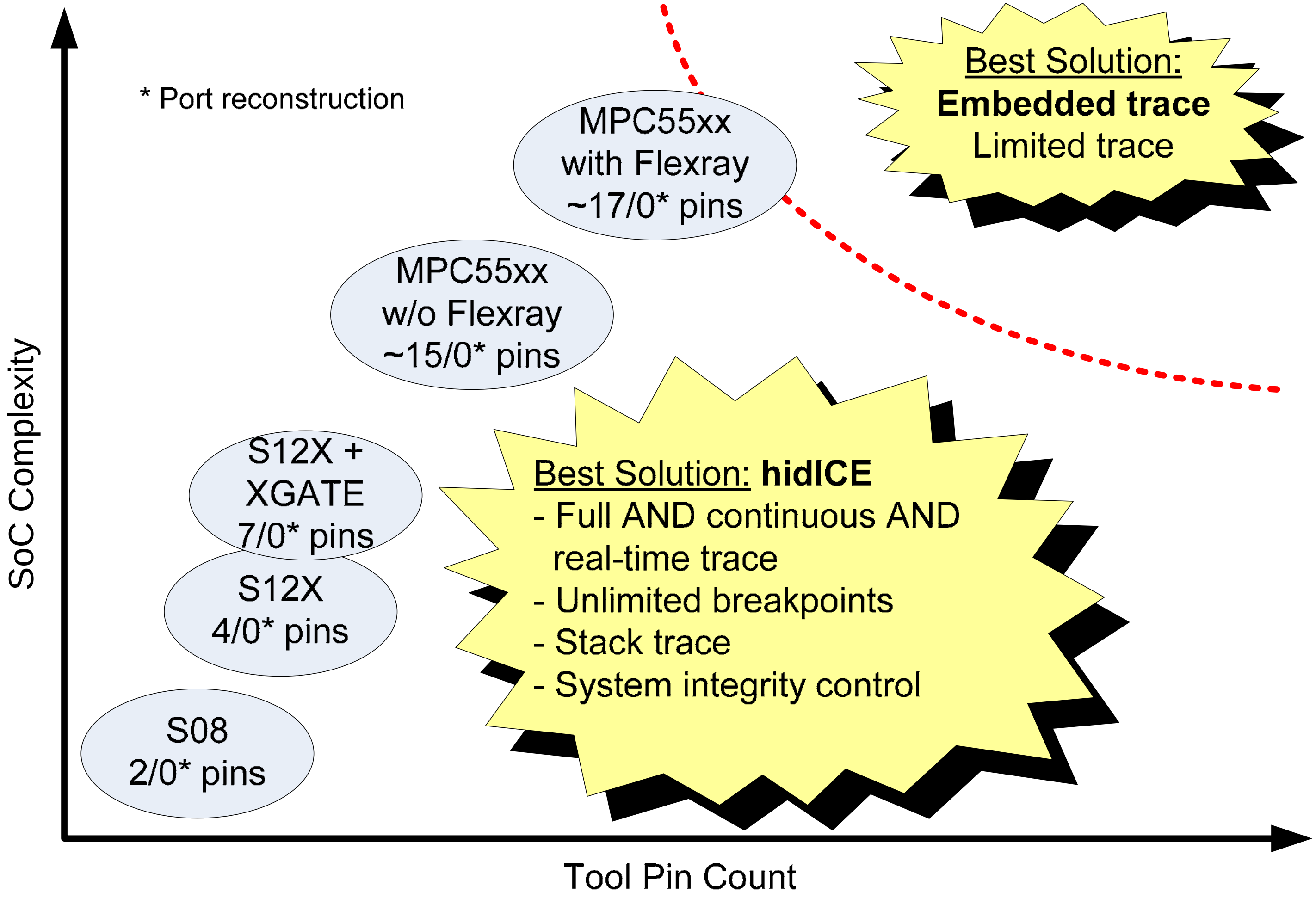
ACCEMIC
accelerate your microcontroller design

hidICE – Implementation Expenses

Target	Emulator
▪ 1k .. 10k gates (simple functionality: data collection and hash calculation) ▪ 2 .. 20 pins (without port reconstruction) ▪ 0 pins (with port reconstruction)	▪ No peripherals to implement ▪ One implementation supports all devices of the same family ▪ For low speed SoCs (< 100 MHz): FPGA implementation ▪ For high speed SoCs: ASIC implementation

Moving intelligence (and costs) from target to emulation

hidICE – Pin Count Estimation

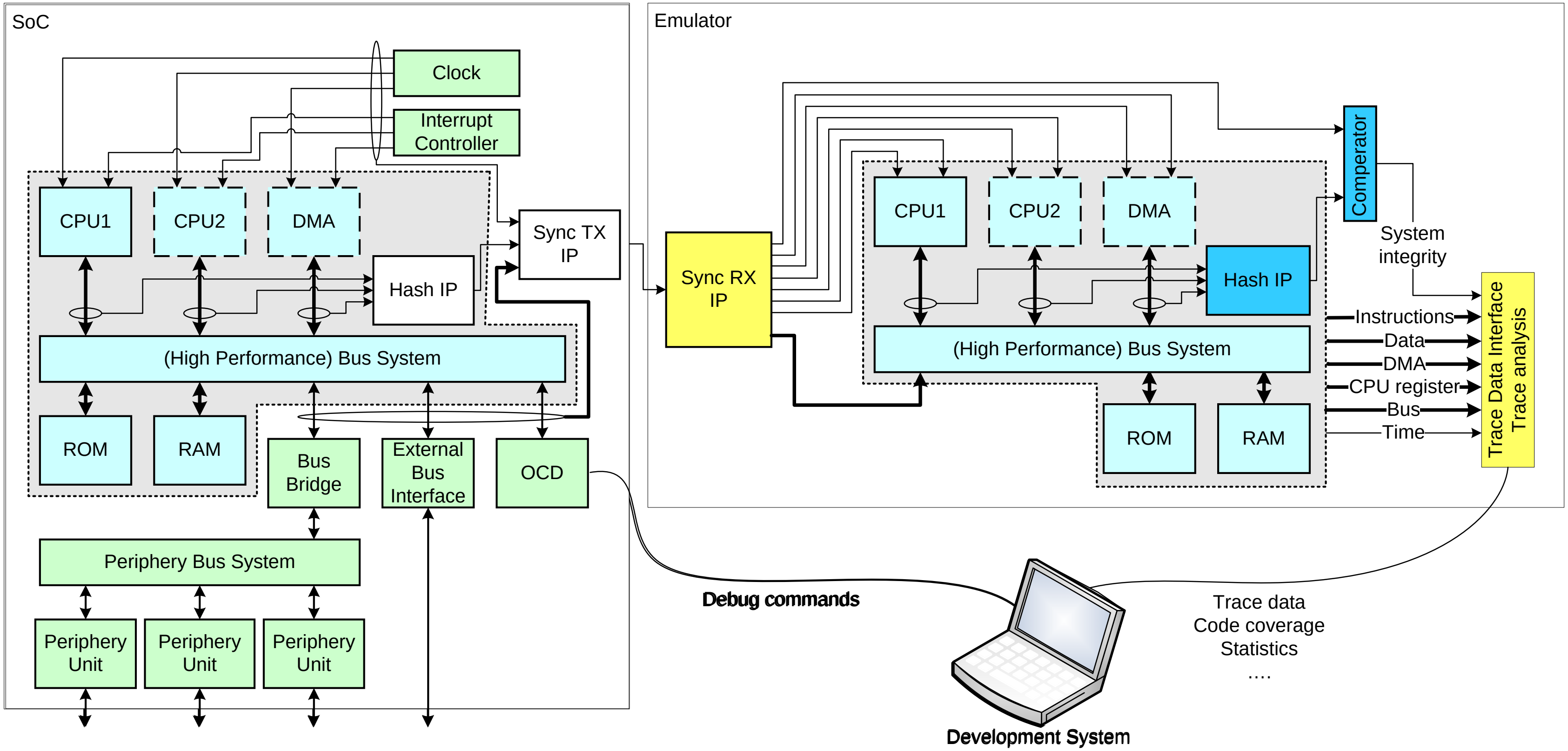


Best Solution: **Embedded trace**
Limited trace

Best Solution: **hidICE**
- Full AND continuous AND real-time trace
- Unlimited breakpoints
- Stack trace
- System integrity control

accelerate your microcontroller design

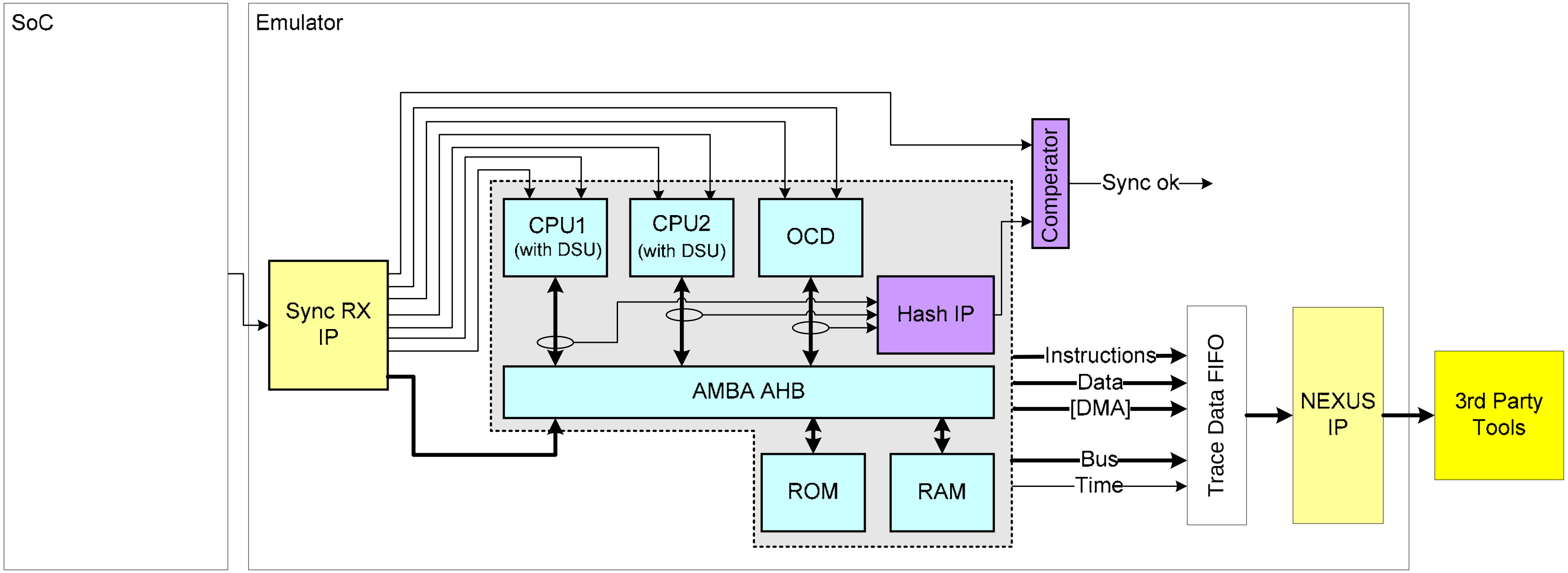
hidICE – Add-on Solution for On-Chip Debug Support



- Low end solution: On-chip debug support
- Mid/high end solution: Combination of OCDS and hidICE

ACCEMIC
accelerate your microcontroller design

Using 3rd party tools with hidICE

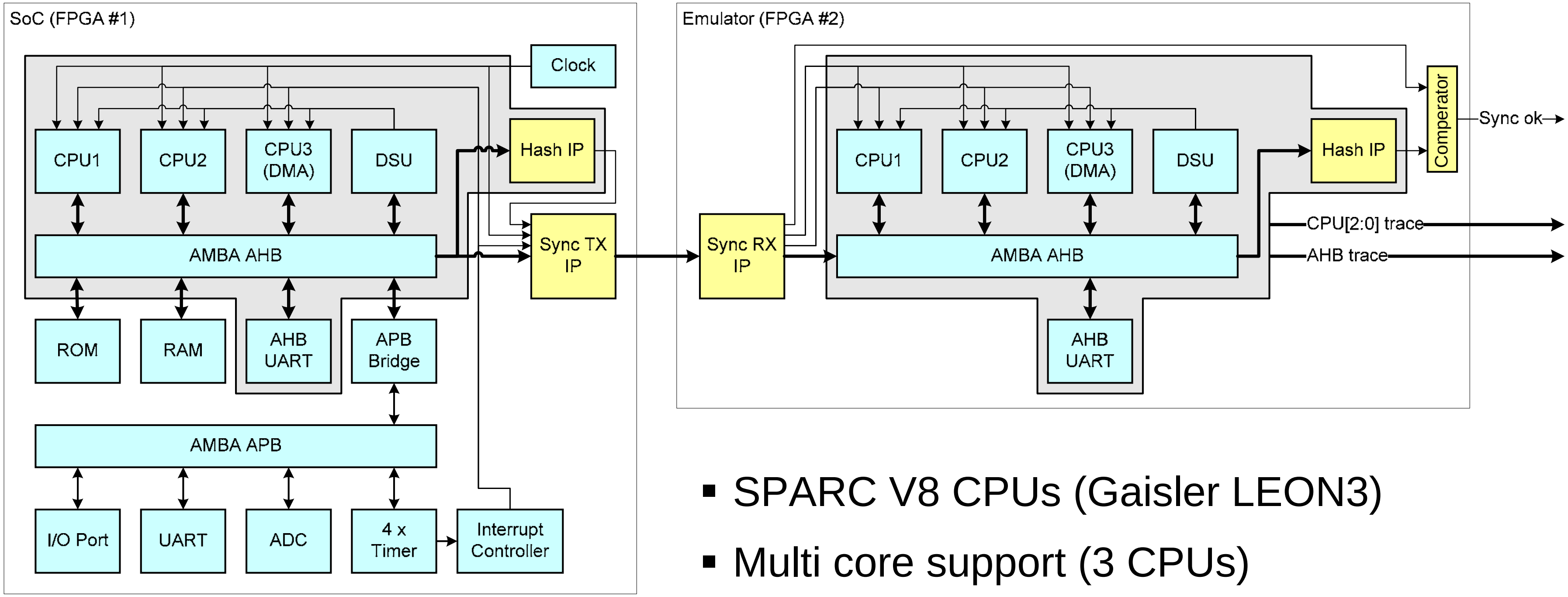


The trace data available inside the emulator can be accessed by standard 3rd party NEXUS compliant tool chains.

☺ Already available tool chains can be used

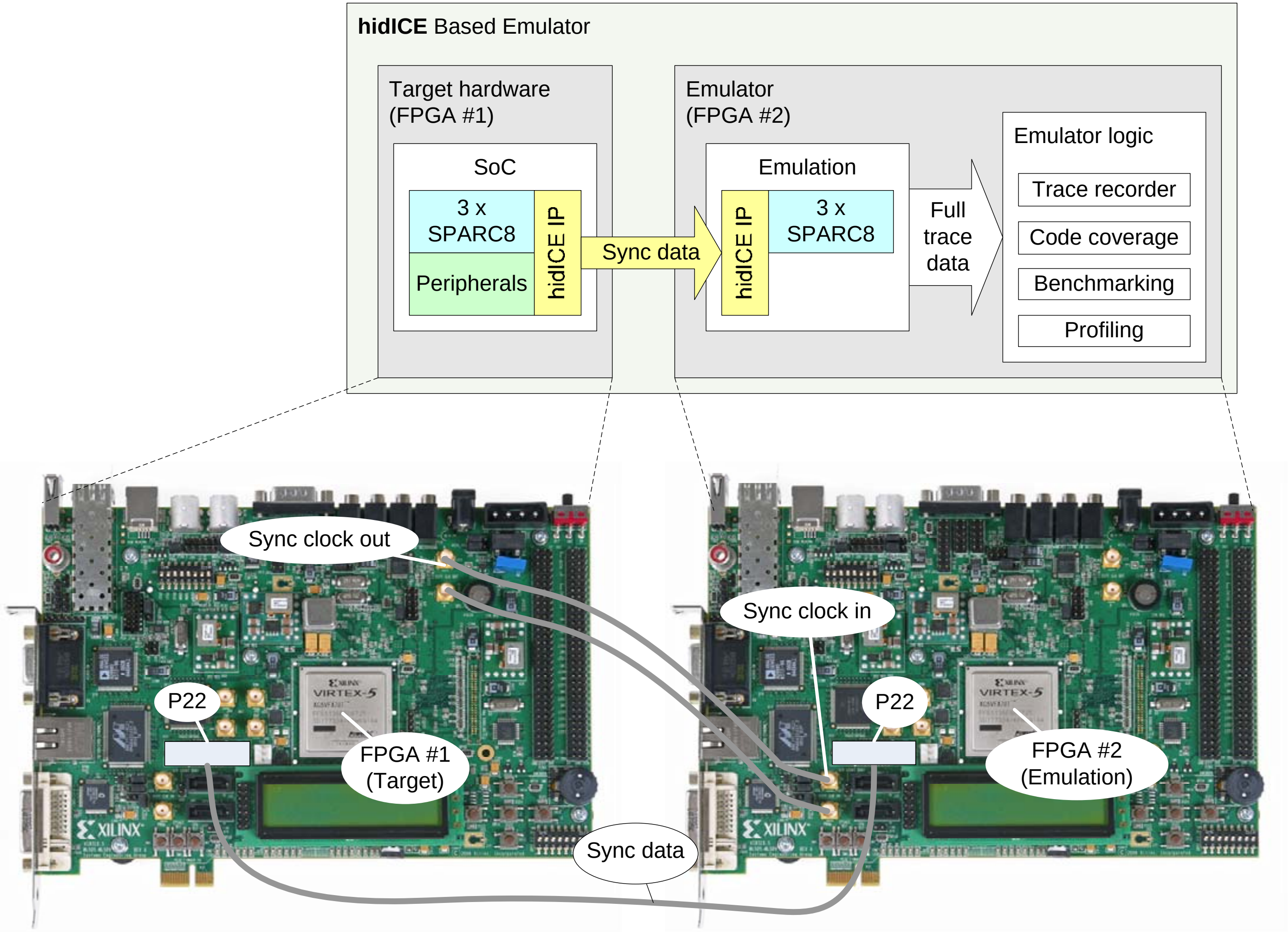
😐 Not all available trace information is used by NEXUS!

hidICE Demonstration System



- SPARC V8 CPUs (Gaisler LEON3)
- Multi core support (3 CPUs)
- AHB bus system (4 bus master)
- On-chip Debug support
- CPU / bus clock: up to 100 MHz (FPGA)
- Full source code and documentation available

hidICE Demonstration System

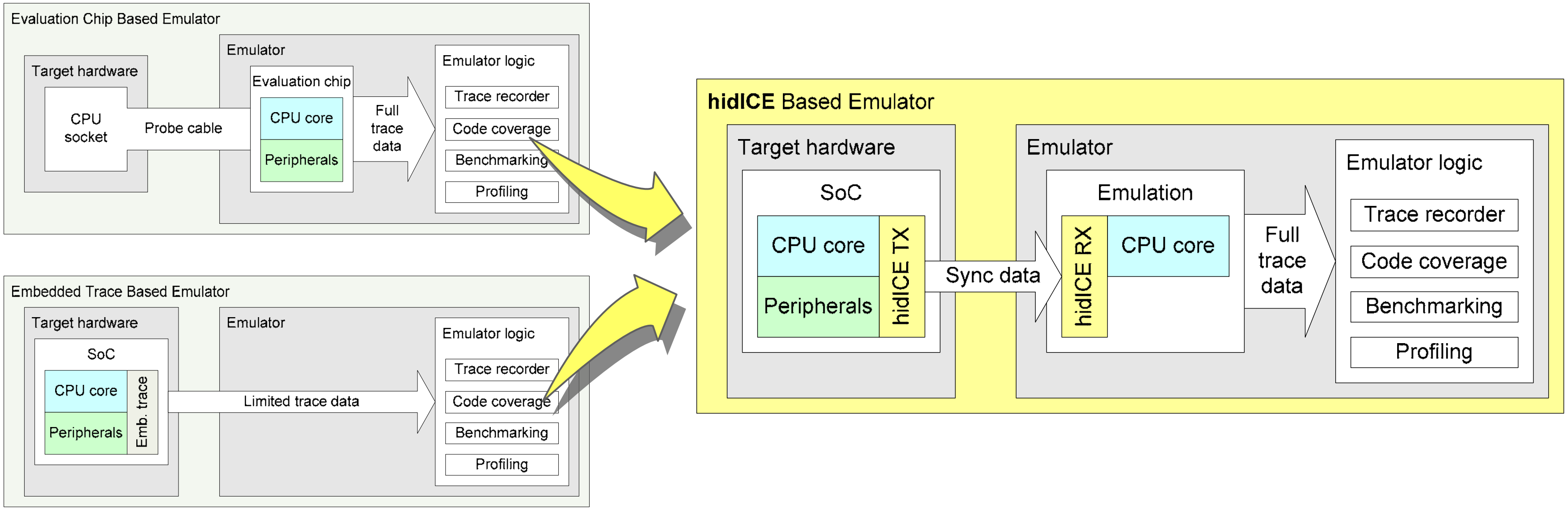


accelerate your microcontroller design

Agenda

1. Today’s Trace Data Capturing Solutions
2. hidICE – Capturing Full AND Continuous AND Real-time Trace
3. **Comparison: Traditional Approaches vs. hidICE**

hidICE - Combining the Advantages of Existing Technologies



- Full trace **AND** continuous trace **AND** real-time operation
- Applicable at high CPU frequencies (> 100 MHz)
- Very low implementation overhead (< 10k gates)
- Full trace support in mass production silicon
  Allows validation using the "real" chips, not any specifically designed ones (emulator chips)

ACCEMIC
accelerate your microcontroller design

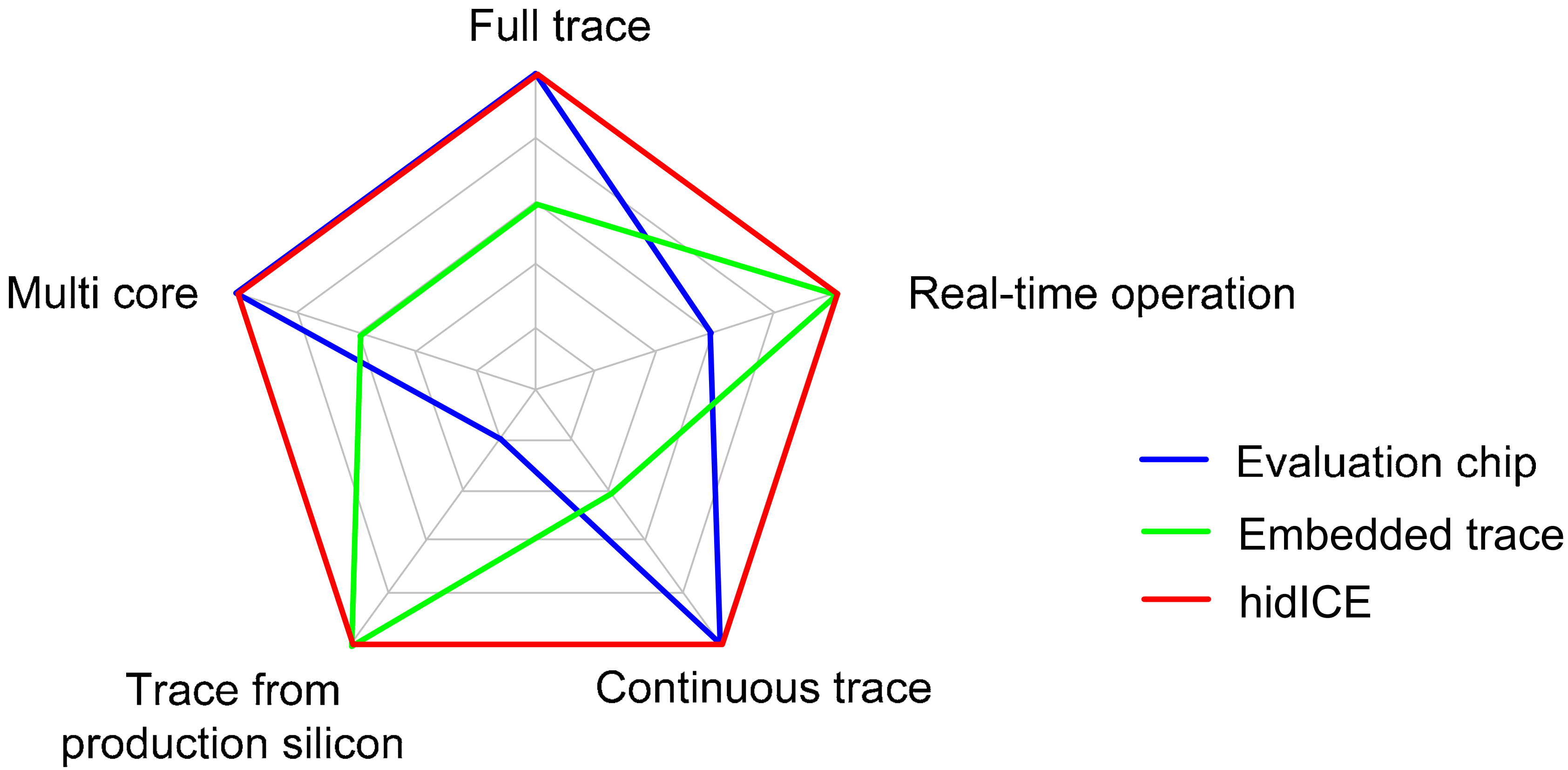
Comparison: Traditional Approaches vs. hidICE

	Evaluation Chip	Embedded trace	hidICE
Full AND continuous AND real-time trace	✓	✗	✓
Software test on mass production chips	✗	✓	✓
Gate count		10k ..100k + trace buffer	1k .. 10k
I/O port reconstruction	✗	✗	✓*
System integrity control	✗	✗	✓
Additional effort for full multi-core trace (n cores)		~ n	<< n

* Applicable for slow output ports

ACCEMIC
accelerate your microcontroller design

Comparison: Traditional Approaches vs. hidICE



hidICE advantages

	FULL TRACE	▪ Executed instructions	✓
		▪ Data read access	✓
		▪ Data write access	✓
		▪ CPU registers (e.g. stack pointer)	✓
		▪ Bus cycles	✓
		▪ Cache activity	✓
		▪ Time stamps	✓
AND	CONTINUOUS TRACE	▪ No CPU stop on full trace buffer	✓
AND	REAL TIME OPERATION	▪ CPU runs at full speed	✓
AND	TRACE FROM PRODUCTION SILICON	▪ Software test and certification on the same silicon as used in mass production	✓
AND	MULTI CORE SUPPORT	▪ Support of multi core microcontroller	✓
+	LOW RESSOURCES	▪ 1k .. 10k gates, low pin count	✓
+	EXTENDED TEST COVERAGE		✓

Our next steps…

- FPGA based implementations of major architectures
  - Busses: AXI, CoreConnect, …
  - CPUs: ARM, PowerPC, …
- Silicon implementations
- Interface specification
- Introducing hidICE to car makers and OEMs
- Exploring new applications:
  - SoC test and verification
  - Runtime verification
  - Continuous MC/DC analysis
  - Continuous bus performance analysis
  - High level language support for trace data analysis
  - Your ideas…

Accemic Services and Contact Information

- Estimation of required ressources
- hidICE implementation support
- hidICE based emulators
- hidICE IP licences

Thank You!

Project manager: Alexander Weiss
aweiss@accemic.com

Accemic GmbH & Co. KG
Hochriesstr. 2
83126 Flintsbach
Germany

www.accemic.com
Phone +49 8034 90993-0
Fax +49 8034 90993-27

accelerate your microcontroller design