

MCDS and MCD API

Multi-Core Debug Solutions setting the
Benchmark in the Automotive Industry

Multi-Core Expo, Tokio, Nov. 2008

Dr. Albrecht Mayer

Senior Principal Emulation Systems and Tooling



Never stop thinking

Outline

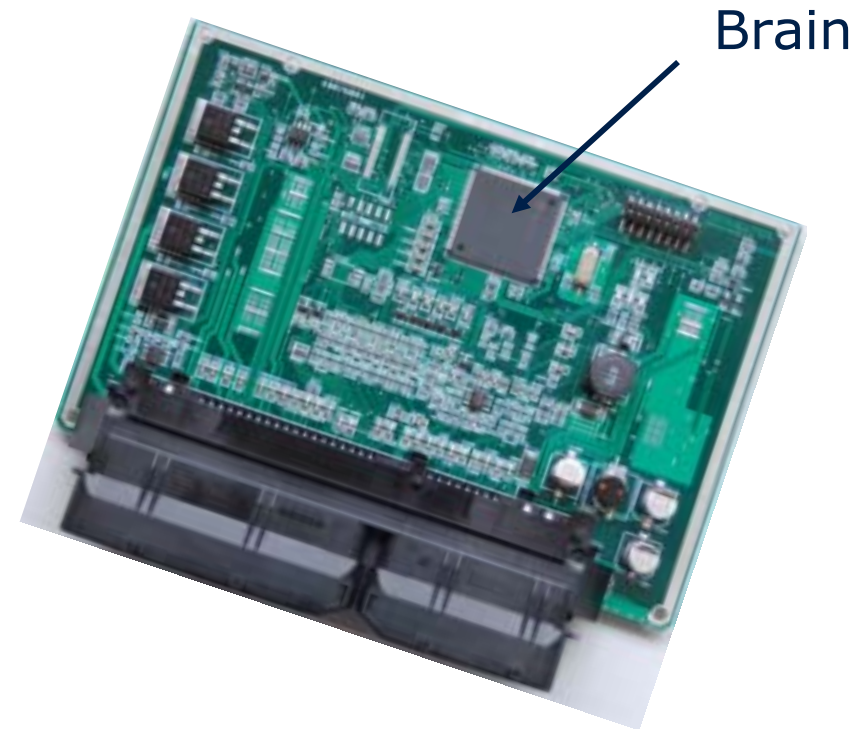
- Motivation
- MCDS (Multi-Core Debug Solution)
- MCDS IP
- MCDS Tooling
- MCD (Multi-Core Debug) API
- Summary

Automotive Microcontrollers

Muscles

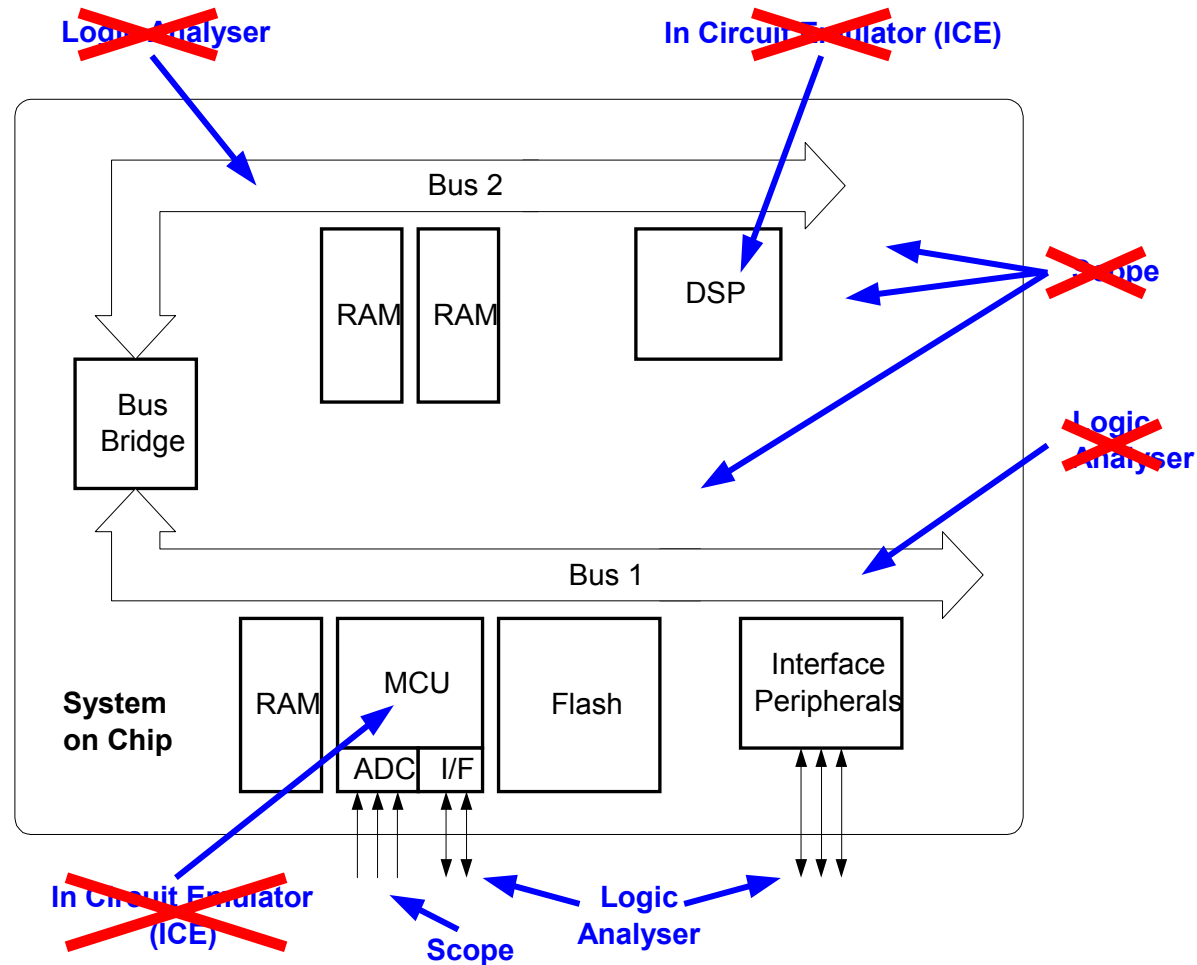


Head



ECU Electronic Control Unit

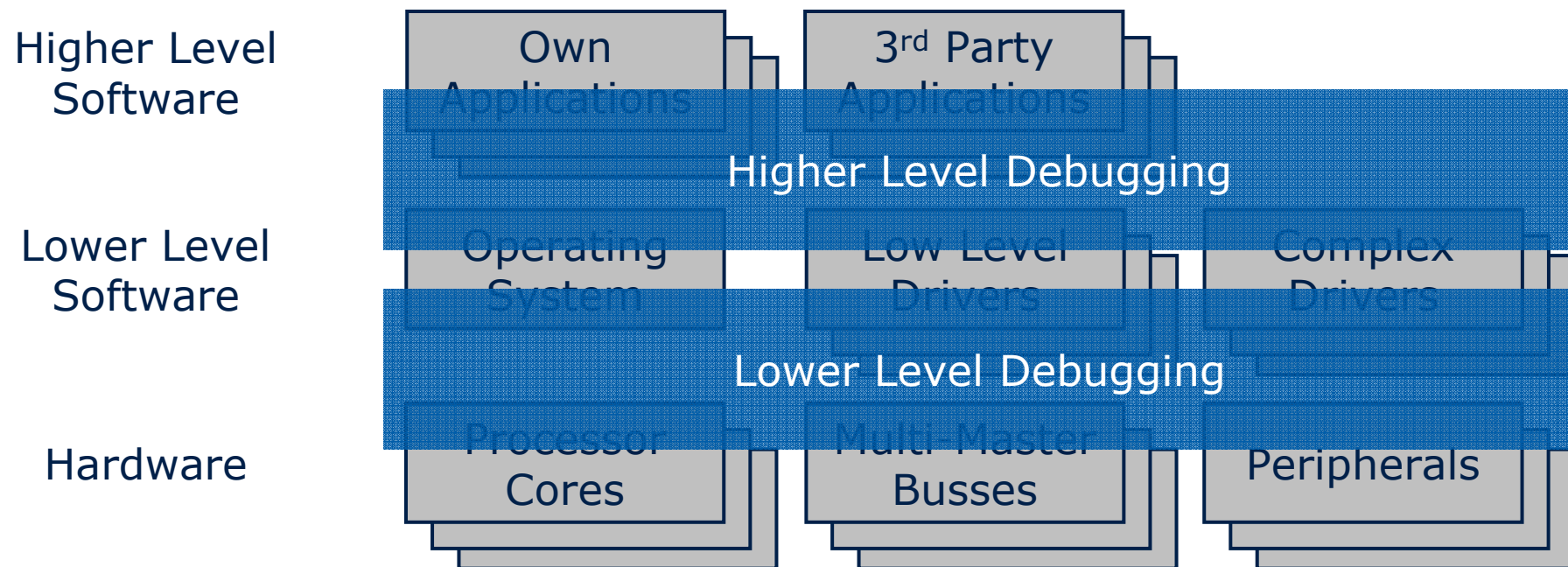
The Visibility Challenge: The Multi-Core SoC



On chip debug support with trace capabilities needed

State of the Art Embedded System Design

- System consists of independent, hierarchical structured, **bug free HW and SW components.**
- Debugging focuses on local interaction of components on one layer and the next higher level
- Verify all components earlier and more thoroughly



Challenges on the different Debugging Levels

Higher Level Debugging

- System level trace
- Performance analysis and optimization:
 - Global with statistical analysis
 - Local with cycle accurate trace and trace of performance relevant events (e.g. cache miss, bus conflict, etc.)

Lower Level Debugging

- Cycle accurate, parallel trace of:
 - multi-core (program and data)
 - multi-bus
 - special signals

Debug Challenge: Reluctance to Invest in Debug Support



- “Debug support is only needed by engineers who make errors!”
- Only a few users for hardware debug

But isn't it really:

Analysis and Performance Optimization (APO)

APO Tools, APO IP

The Chance

- Get it out of the lab and to thousands of software engineers
- Definitely needed for multi-core real time systems
- Unified solution for debug, calibration and measurement

Infineon Solution developed for Automotive: On-chip Emulator



Limitation is the capacity of the on-chip trace memory

Only relevant data has to be captured!

Solution:

- On-chip filter capabilities
- Trigger logic
- Trace data compression
- Performance counters

Multi Core Debug Solution (MCDS)

Introduction to Automotive Emulation Devices



ED = Production SoC
+ large overlay RAM
+ MCDS trigger/trace unit

ED has same package

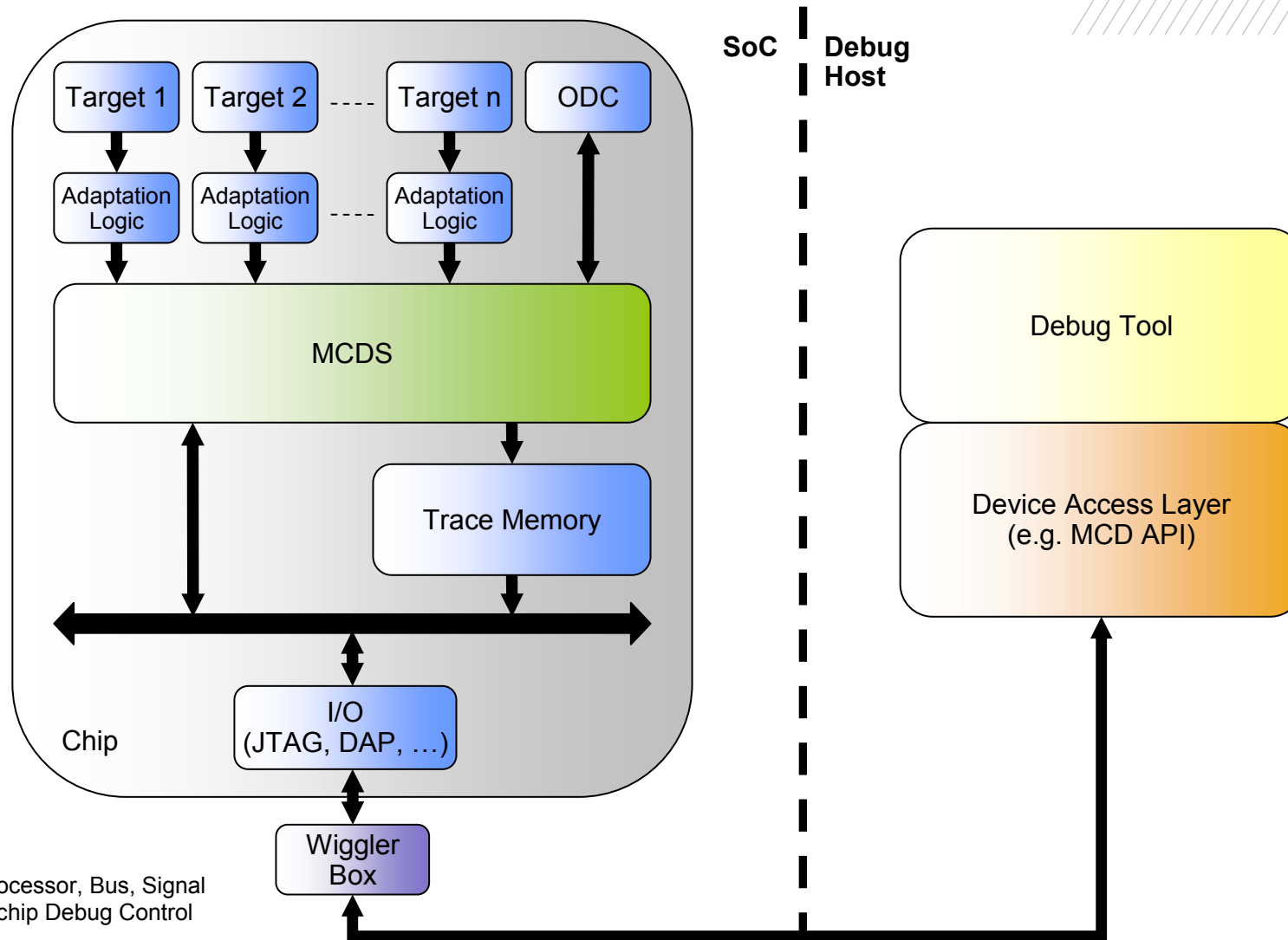
- Higher current consumption is the only difference in target system
- Reason for ED is the large flash overlay RAM for calibration
- Just MCDS + small trace RAM can be on production SoC → No ED required

MCDS Kernel Building Blocks

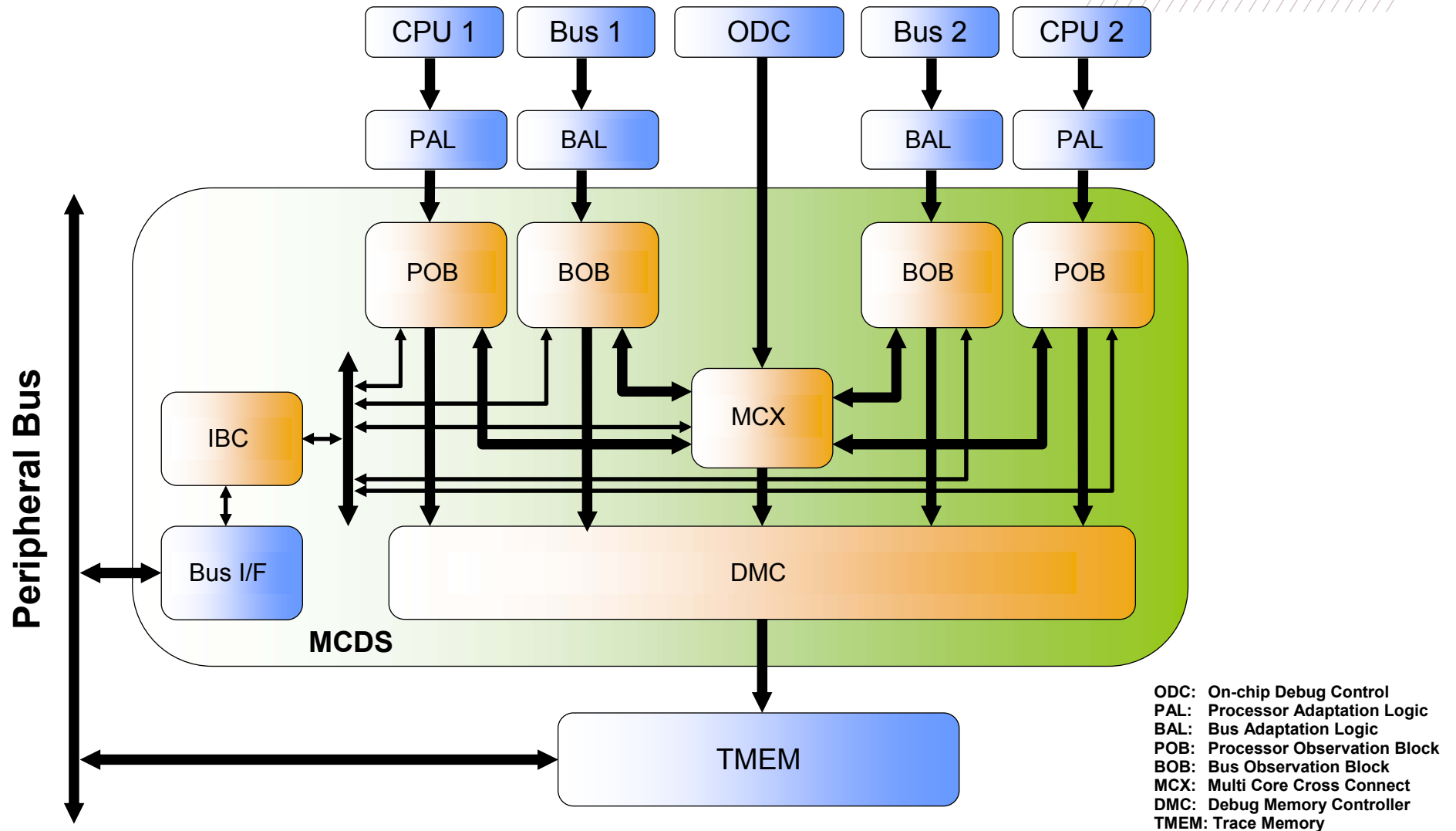
The IPextreme logo features the word "IPextreme" in a bold, black, sans-serif font. The letter "x" is highlighted in red. A red triangle points to the right, positioned at the end of the word.

IPextreme

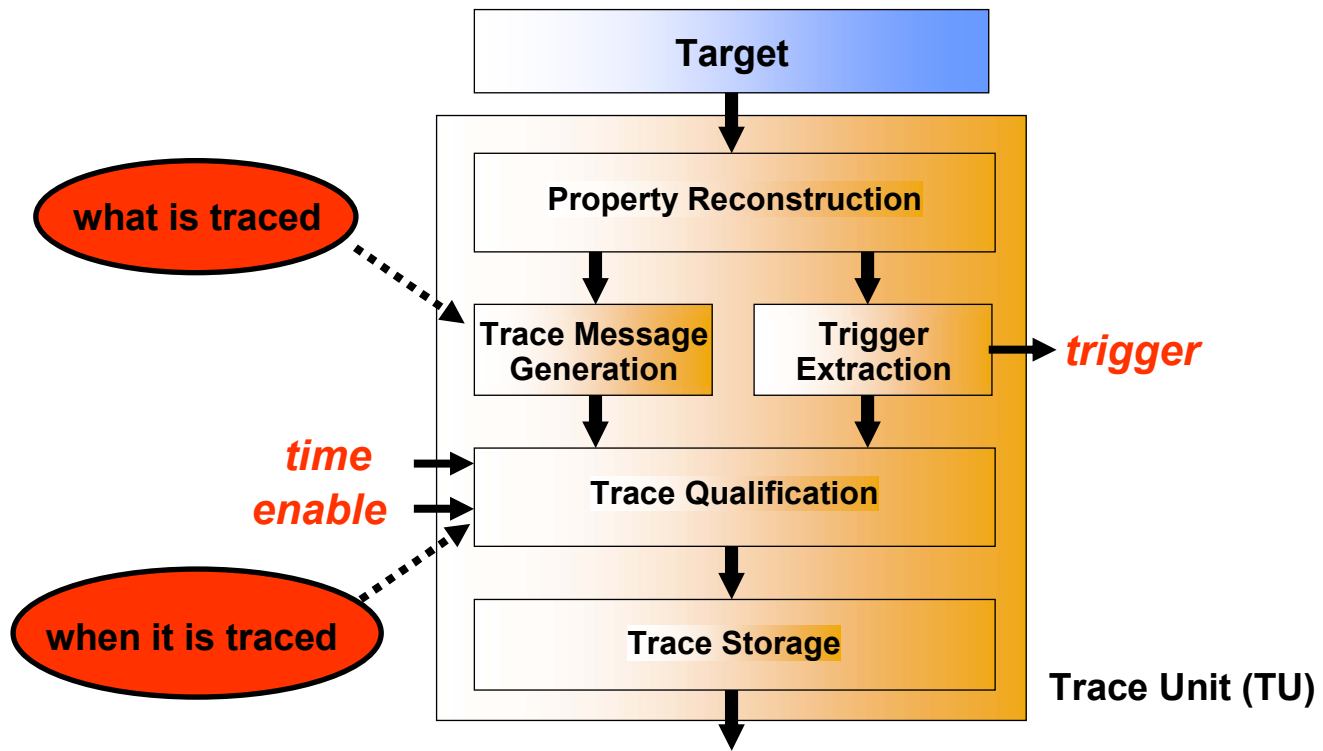
MCDS Debug Architecture



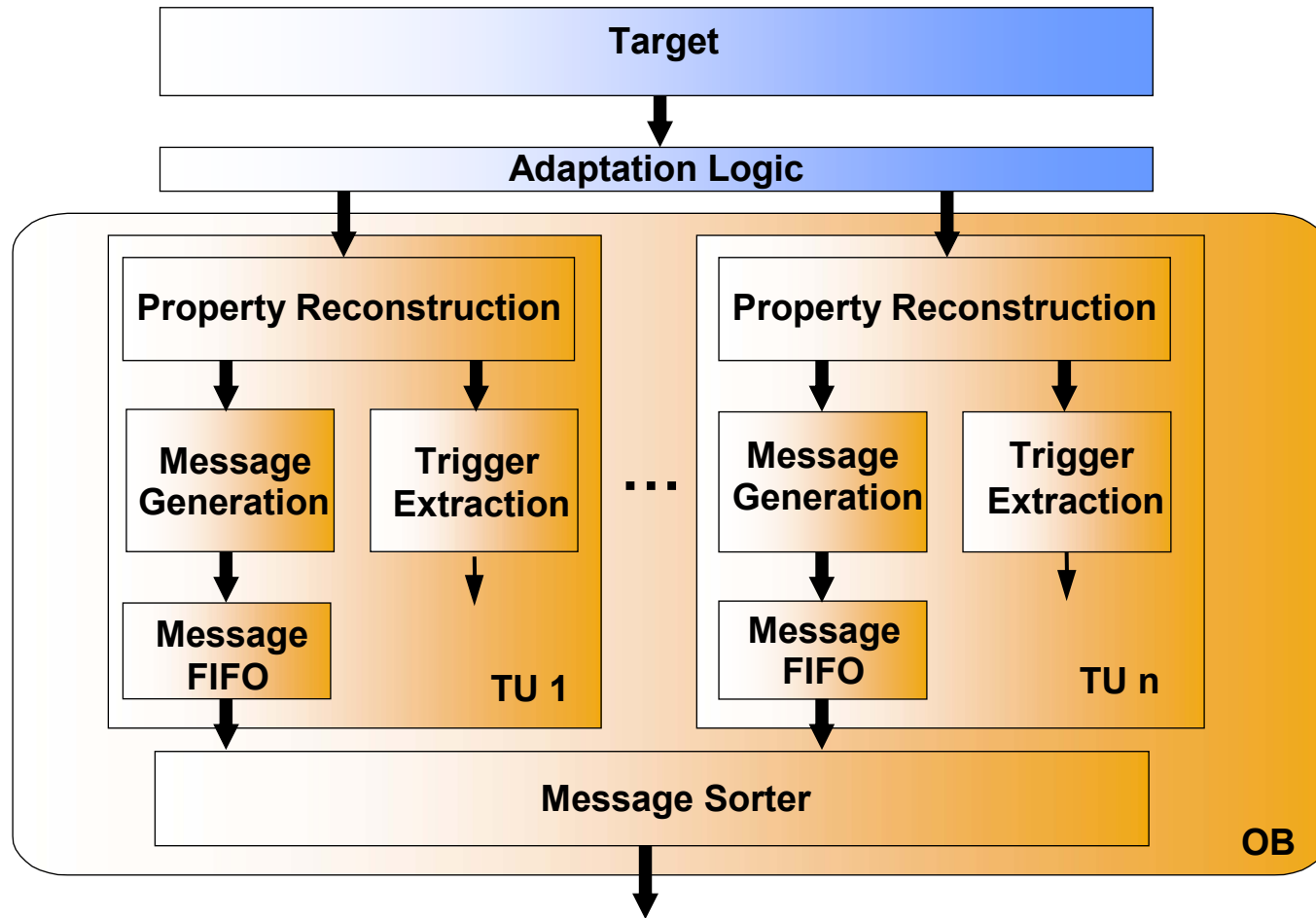
Sample MCDS System



Trace Unit



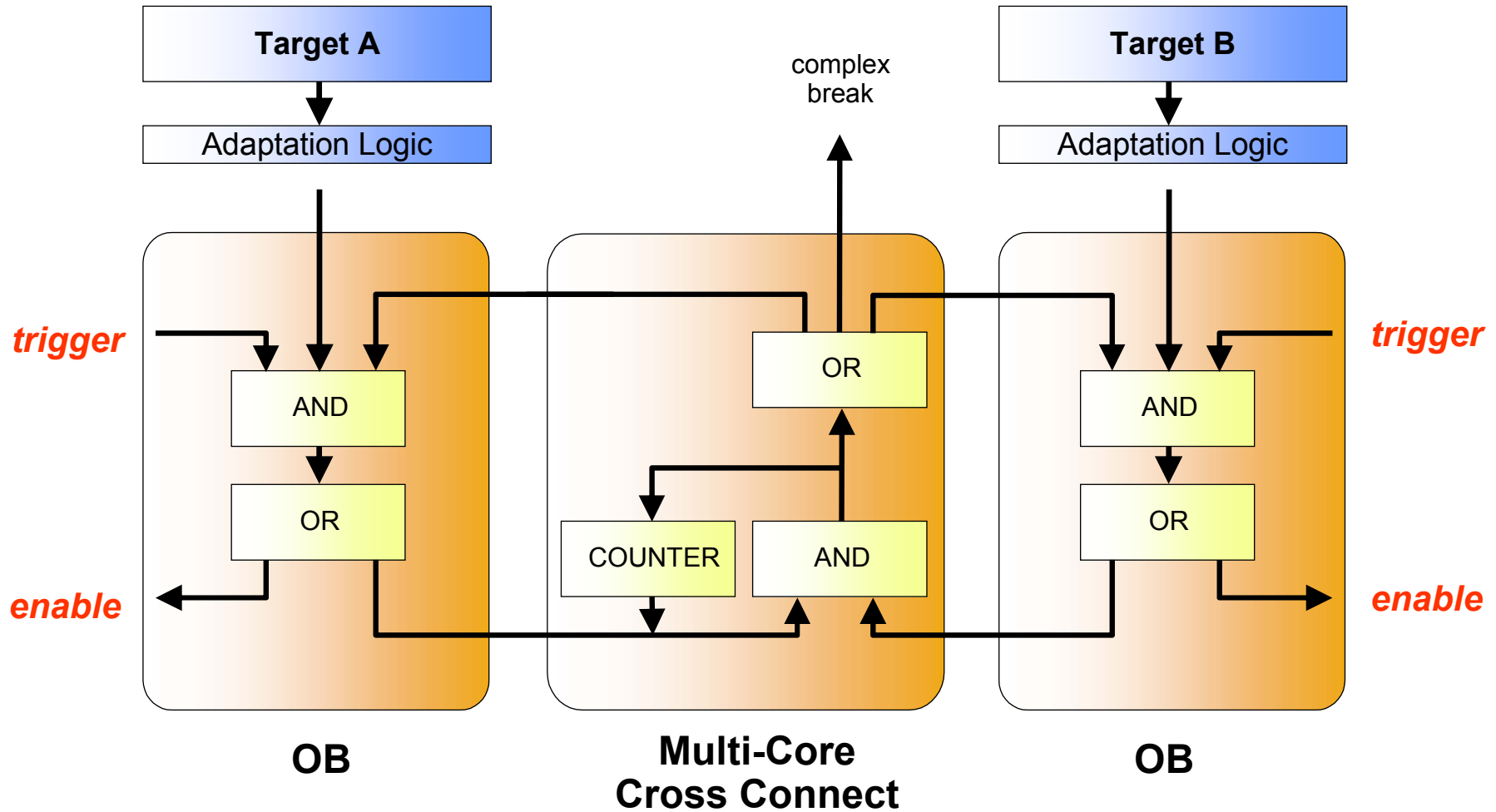
MCDS Observation Block



MCDS Building Blocks Details

- ▶ Adaption Logic
 - Translate target specific i/f to MCDS i/f
- ▶ Observation Block (OB)
 - Contains several Trace Units (of different types)
 - Reconstruction of traced information
 - Trace message generation
 - Trigger Extraction (triggers used for Trace Qualification)
 - Buffering (FIFO)
 - Trace Qualification
 - Message sorting from different Trace Units

Hierarchical Trace Qualification



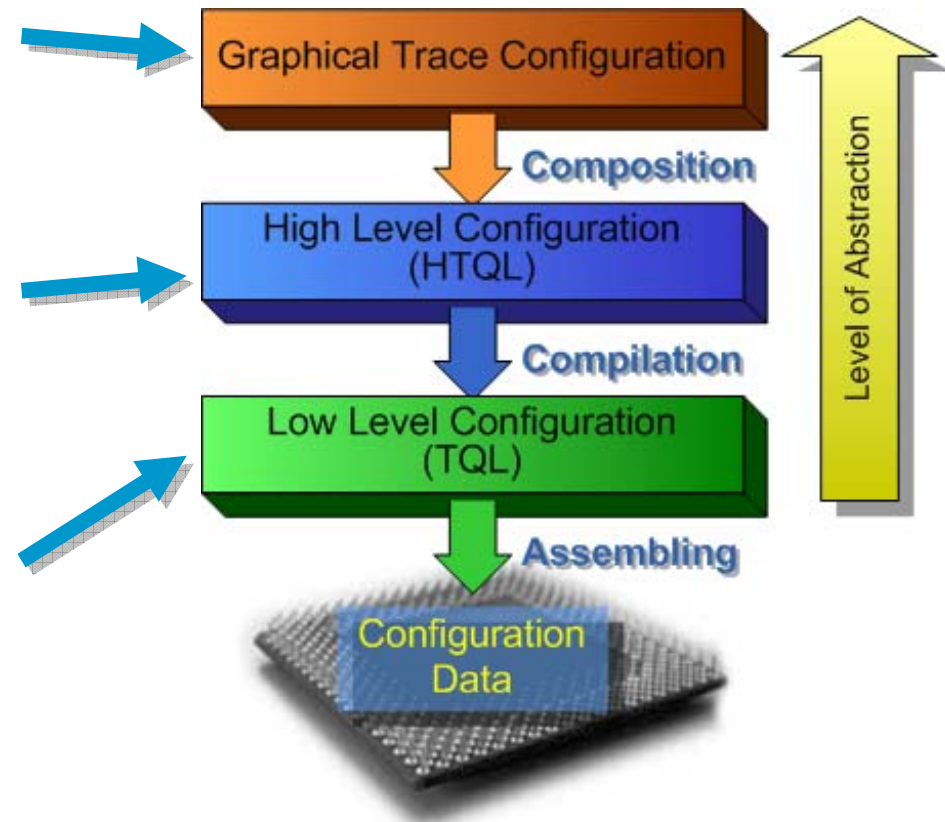
MCDS IP Summary

- ▶ Debugging of complex multi-core real-time SoCs
- ▶ Scales with Moore's Law and clock frequency trend
- ▶ No extra pins required
- ▶ Trace data compression
- ▶ Enabler for full visibility into the system
 - Non-intrusive
 - State-machine based complex triggers
 - Cross target triggers
 - Cycle accurate trace reconstruction
 - Exact time correlation between different trace sources

Universal Emulation Configurator for MCDS

Three level approach to:

- Hide the complexity:
Easy to use graphical editor for fast specification of trace / debug tasks
- Provide the global view:
Easy to use language to provide customizable building blocks for graphical editor, allows usage of state machines
- Keep the feature available:
Low-level language for tweaking often used / specialized trace / debug tasks



TriCore AUDDO-XX

High-End Automotive Powertrain Microcontroller

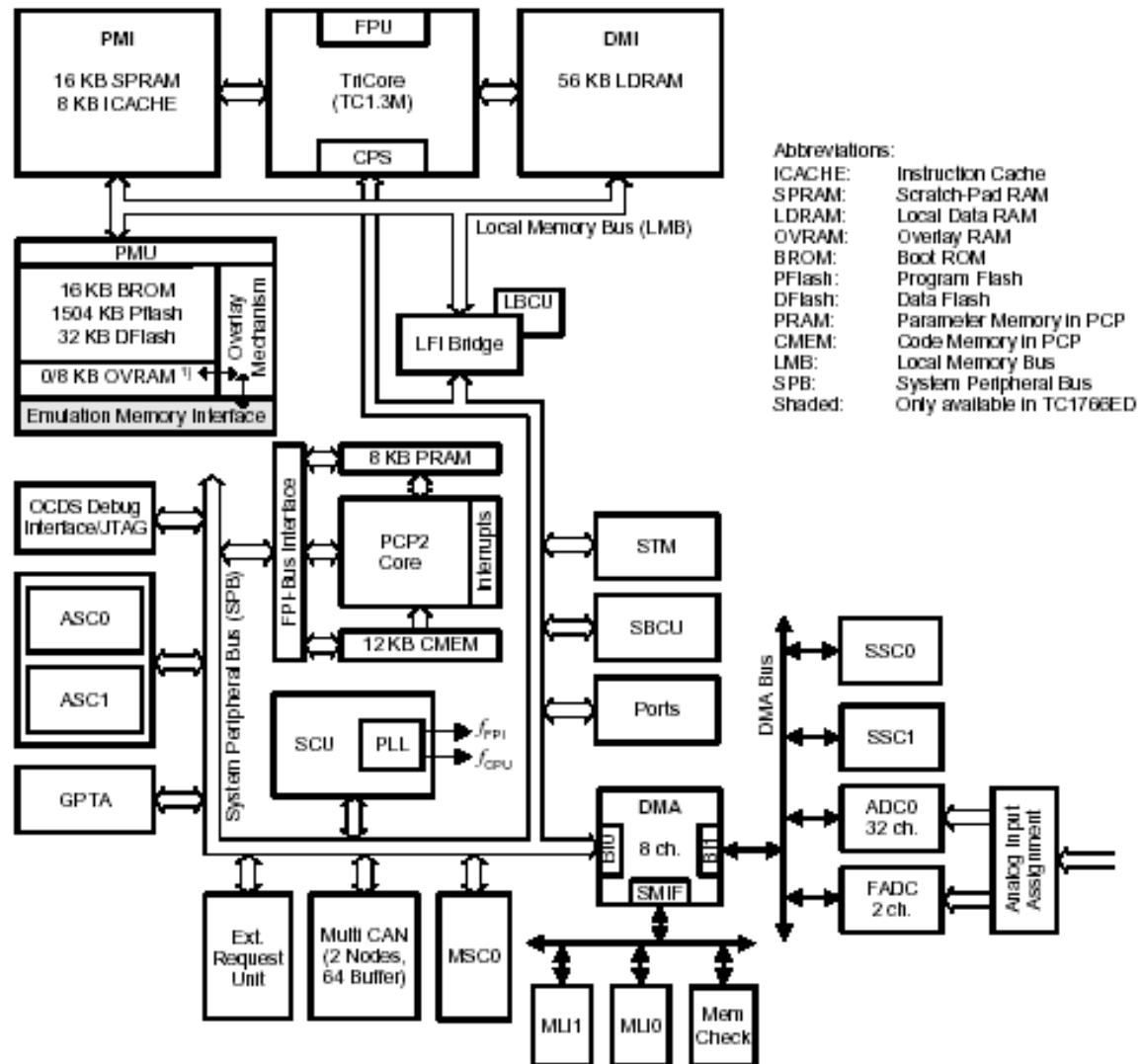


Market Share

- 50% Europe
- 30% WW

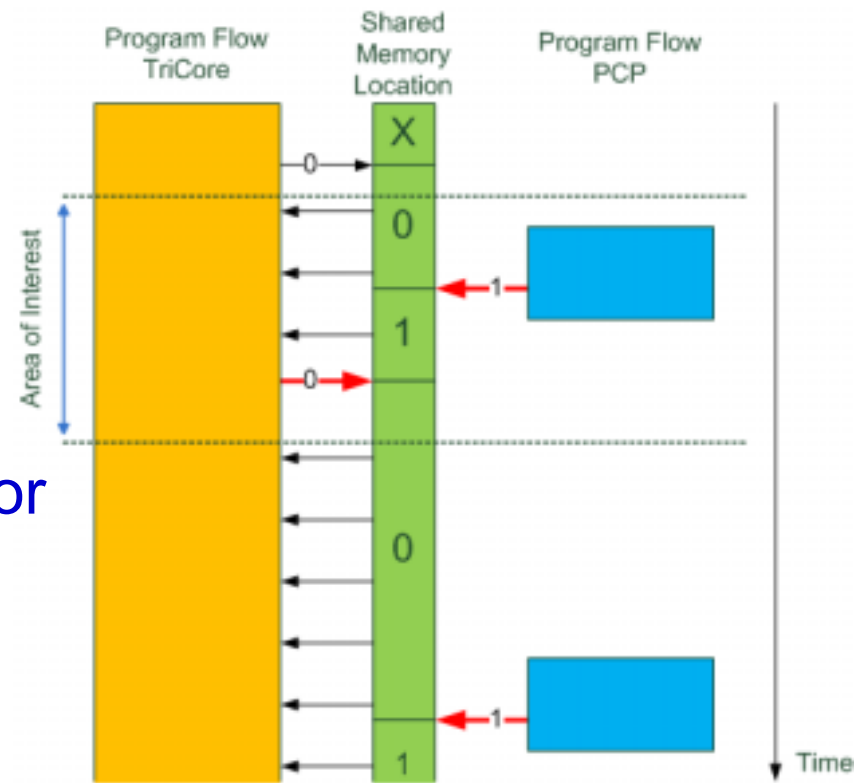
Complexity Level

- Multi-core
- Multi multi-master busses
- Hard real time



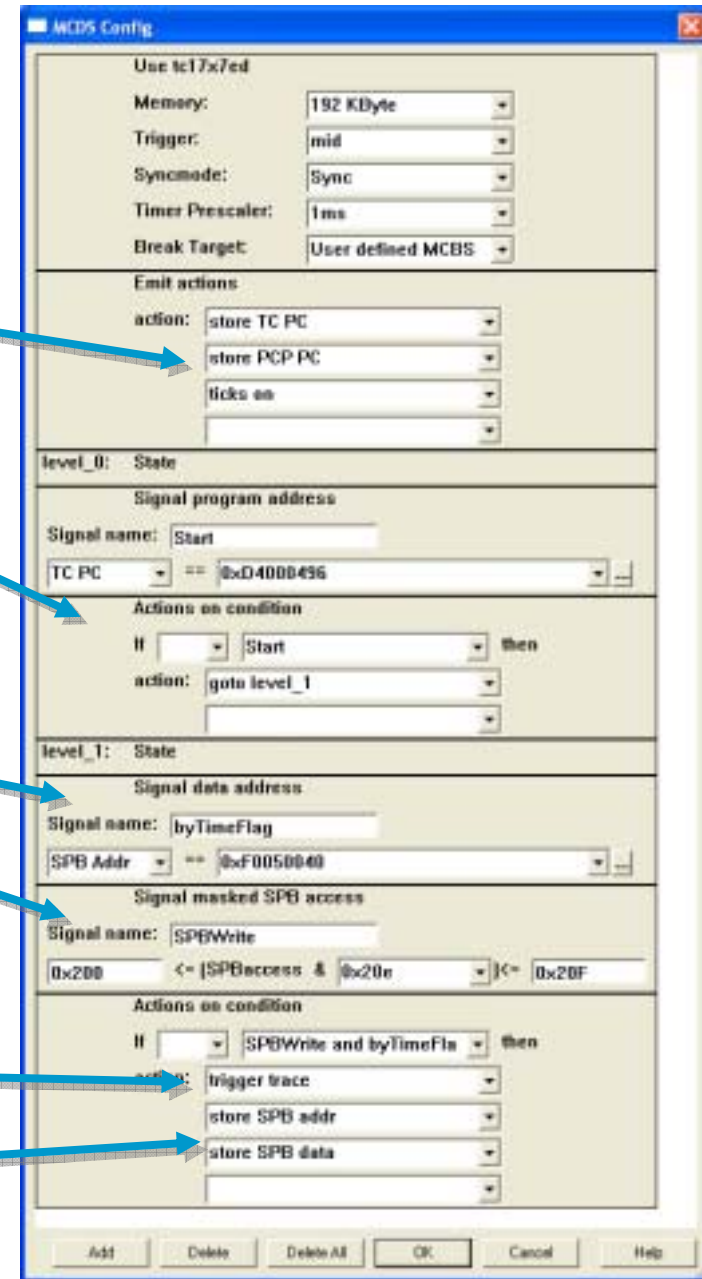
Example: TriCore / PCP Interaction

- Observation of write access to a memory location shared by the TriCore and PCP processor cores
- MCDS allows to observe cores and busses in parallel
 - All trace messages time aligned
 - Parallel program and data trace
- State machine is used to trace only the “Area of Interest”
 - HTQL compiler maps states to MCDS counters
- Make use of the graphical editor to compose the task



Example: TriCore / PCP Interaction

- Program trace all the time
- Observation of SPB bus transactions should begin after start event has occurred
- Recognize SPB bus access
 - Address of shared memory location
 - Write access of any bus master
- Mark area of interest in trace record using a trigger
 - All interesting things will happen around it
 - Make the data transfer visible



Example: TriCore / PCP Interaction (3)

Tick	Address	Data	Interpret	Source	Block	Trigger
-44	0xD40004A2		JNE d15, 0x1, 0xd400049a		Core	
-42	0xD400049A		MOVH.A a15, 0xf005	if (byTimeFlag == 1)	Core	
-37	0xF0060008		XOR R1, R4, cc_UC		PCP	
-36	0xF006000A		ST.F R1, [R2], 32	st.f r1, [r2], SIZE=32	PCP	
-35	0xF006000C		LDL.IL R0, #1	ldl.il r0, 0x0001	PCP	
-34	0xD400049E		LD.W d15, [a15] 0x40		Core	
-33	0xD40004A2		JNE d15, 0x1, 0xd400049a		Core	
-32	0xD40004A0		ST.F R0, [R3], 32	st.f r0, [r3], SIZE=32	PCP	
-28	0xD40004A2		JNE d15, 0x1, 0xd400049a		Core	
-26	0xD400049A		MOVH.A a15, 0xf005	if (byTimeFlag == 1)	Core	
-24	0xF0060012		EXIT EC=0, ST=0, EP=0, INT=0, cc_UC	exit EC=0, ST=0, INT=0,...	PCP	
-2	0xF0050040	0x00000001	Data: Write SVM PCP		SPB	
-18	0xF0060012		EXIT EC=0, ST=0, EP=0, INT=0, cc_UC	exit EC=0, ST=0, INT=0,...	PCP	
-13	0xD400049E		LD.W d15, [a15] 0x40		Core	
-12	0xD40004A2		JNE d15, 0x1, 0xd400049a		Core	
-12	0xD40004A6		MOVH.A a15, 0xf005		Core	
-9	0xD40004AA		MOV d15, 0		Core	
-8	0xD40004AC		ST.W [a15] 0x40, d15		Core	
-8	0xD40004B0		MOVH d3, 0xd000		Core	
-8	0xD40004B4		LD.W d2, [a10]		Core	
-6	0xD40004B6		MOVH d15, 0xcccd		Core	
-5	0xD40004BA		ADDI d15, d15, -0x3333		Core	
-4	0xD40004BE		MUL.U e0, d2, d15		Core	
-3	0xD40004C2		SH d15, d1, -0x3		Core	
-1	0xD40004C6		MUL d15, d15, 0xa		Core	
-1	0xF0050040	0x00000000	Data: Write SVM LPI bus bridge (co...		SPB	
0	0xD40004C2		SH d15, d1, -0x3		Core	
2	0xD40004CE		ADDI d15, d3, 0x2010		Core	
3	0xD40004D0		MOV.A a3, d0		Core	
3	0xD40004D2		ADDSC.A a15, a3, d15, 0		Core	
5	0xD40004DA		LD.W d15, [a10] 0		Core	
5	0xD40004DC		ST.B [a15] 0, d15		Core	
6	0xD40004DE		LD.W d2, [a10]		Core	
7	0xD40004DC		MOVH d15, 0xcccd		Core	
8	0xD40004E0		ADDI d15, d15, -0x3333		Core	
9	0xD40004E4		MUL.U e0, d2, d15		Core	
10	0xD40004E8		SH d15, d1, -0x3		Core	
12	0xD40004EC		MUL d15, d15, 0xa		Core	
13	0xD40004F0		SUB d15, d2, d15		Core	

PCPwrite

Trace records from different cores / busses

TriCore write

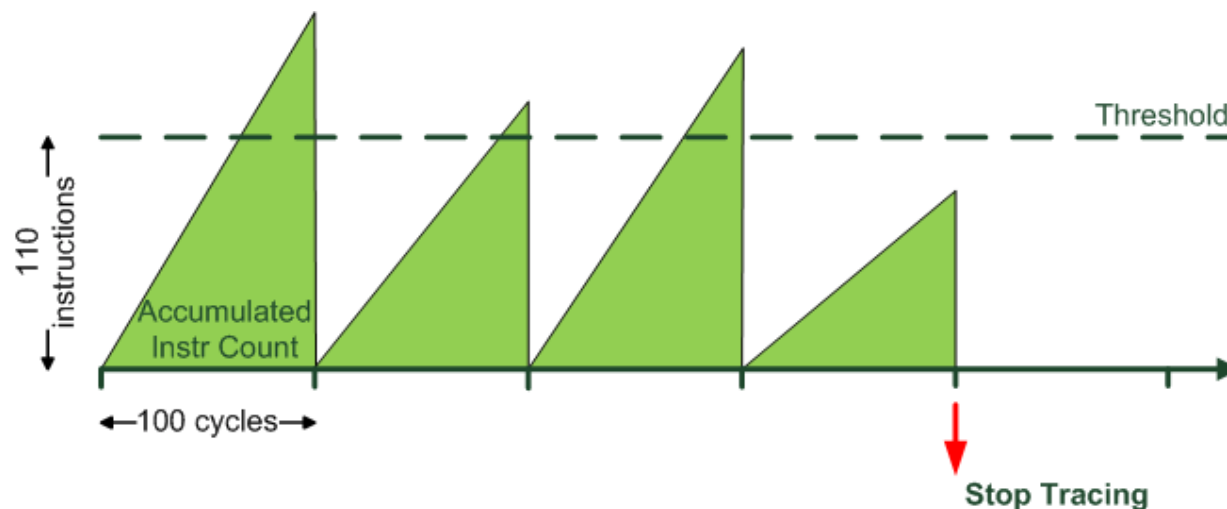
Trigger mark

Performance Measurement and Monitoring

- For cores and busses MCDS can be used for performance measurements and monitoring
- Counters inside MCDS are used (performance counters)
 - Counters accumulate incoming events
- Countable performance parameters:
 - Executed instructions
 - Stall cycles
 - Cache misses, cache hits
 - Interrupts
 - DMA transactions
 - ...

Example: Instructions per Cycle rate

- Monitoring the performance counter for executed instruction during a defined period of cycles
- When executed instructions per period drop below a threshold stop the instruction trace
- The instruction trace shows the program flow causing the IPC drop



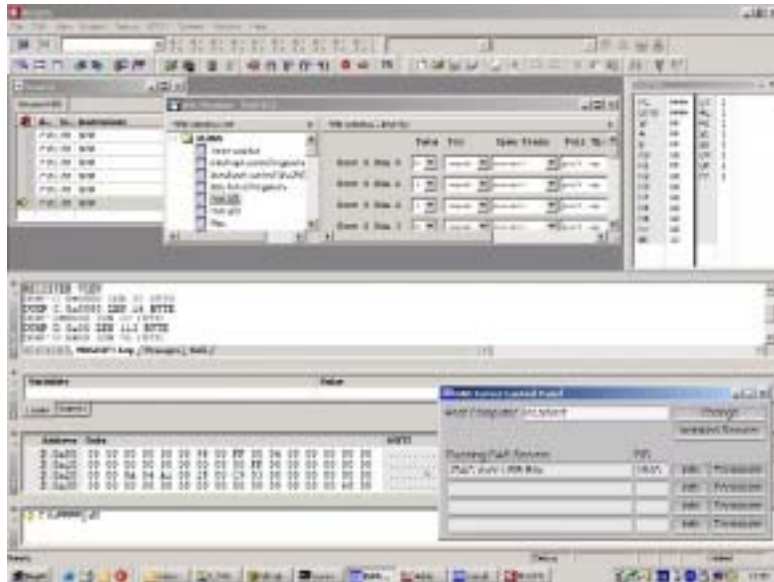
MCDS allows Performance Optimization of Application *and* Architecture



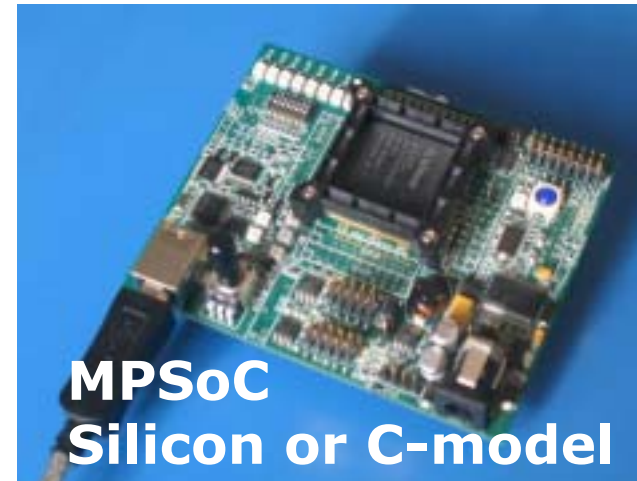
Improving Application Performance as shown on previous slides

Improving System Architecture by:

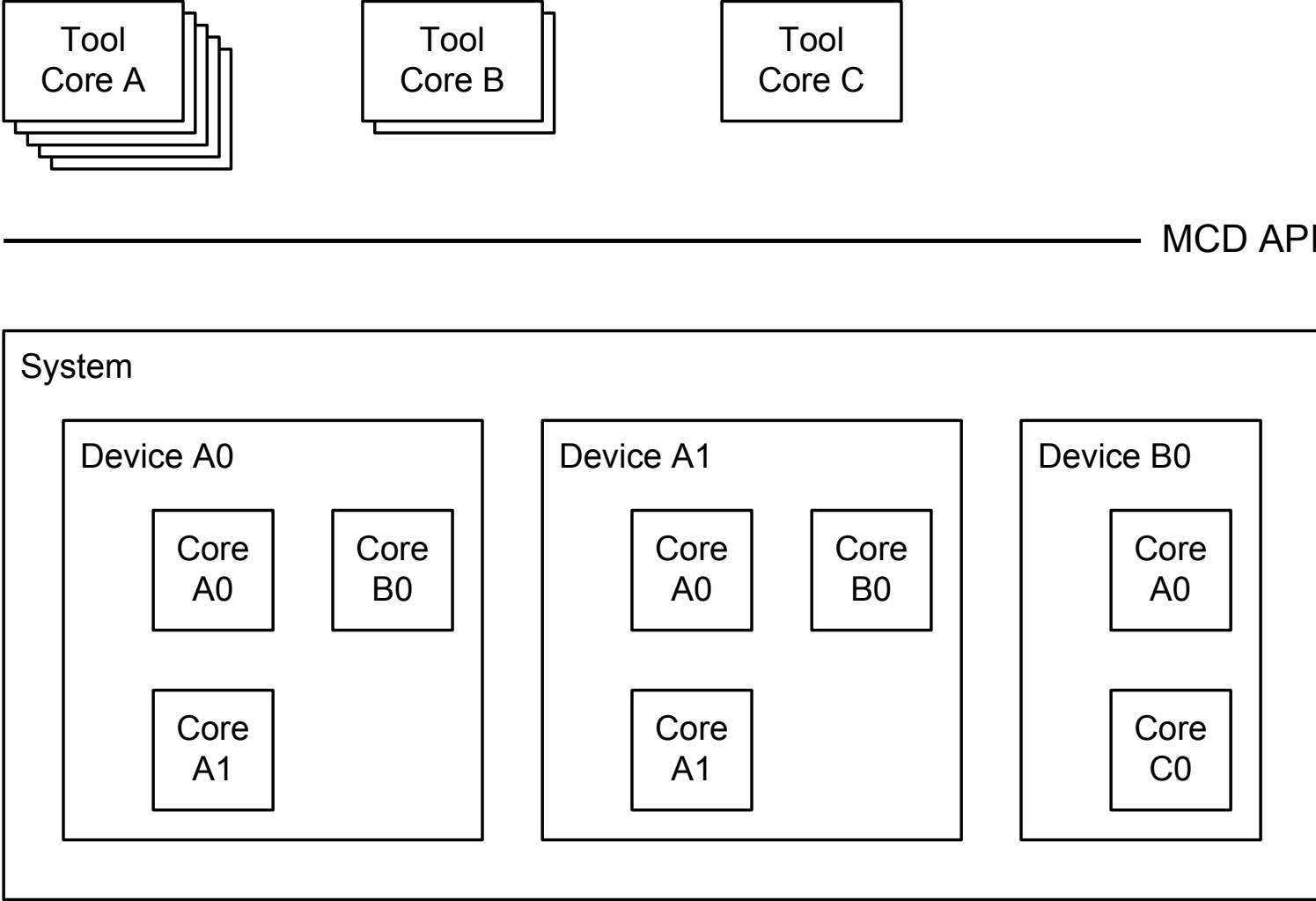
1. Tracing of all relevant system events for relevant applications
2. Scale event categories according to architecture changes (e.g. penalty in number of clock cycles for a cache miss)
3. Calculate new Instructions per Cycle (IPC) rate from weighted and scaled event categories



MCD API
=
Abstraction of
(physical) APO
tool interface

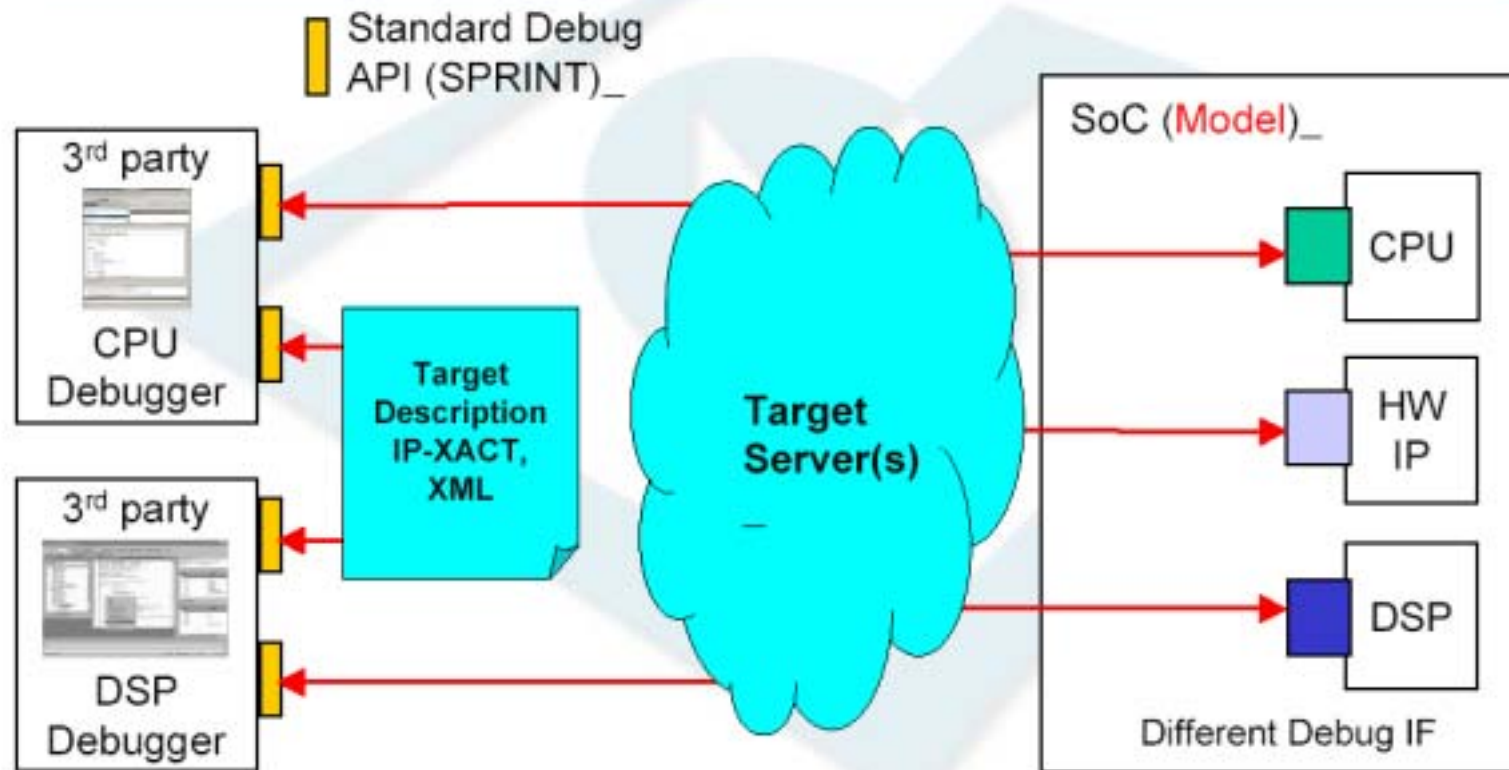


MCD API Complex Use Case



MCD_SYSTEM_BLOCK_DIAGRAM_COMPLEX

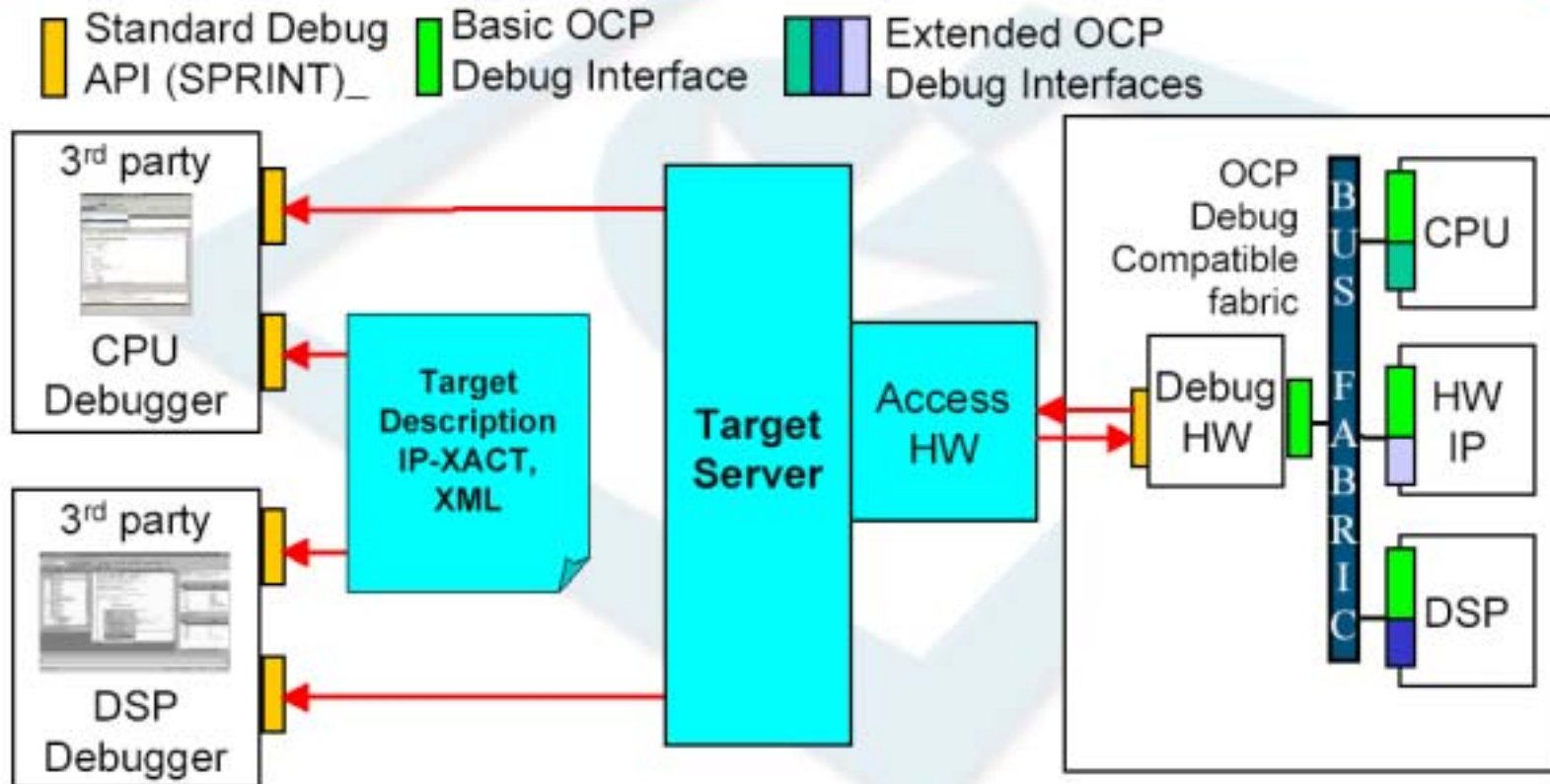
Debug Environment



Copyright ©2008 OCP-IP, All Rights Reserved

3

NEW OCP Debug Standard



The SPRINT software standard is related to our work as a general debug functionality description for multiple processors.

Copyright ©2008 OCP-IP, All Rights Reserved

4

SPRINT Work Package 4 Members

30



Analysis and Debug Tools

Summary

- MCD API standard I/F between tools and targets
 - For multi-core simulation models and silicon
 - Version V1.0 will be released end of 2008

- MCDS is the benchmark APO solution in the automotive industry
 - MCDS was introduced with Emulation Devices
 - Product chip solutions (also non automotive) are upcoming
 - Proven MCDS based tooling exists
 - pls Development Tools, Lauterbach
 - Automotive specific for calibration and measurement)
 - MCDS IP can be licensed from IPextreme



We commit.

We innovate.

We partner.

We create value.



Never stop thinking