FTF | FREESCALE TECHNOLOGY FORUM POWERING INNOVATION

Increase Application Performance and Shorten Debug Time Using QorIQ Processors Advanced Debug Capabilities: CodeWarrior Software Analysis

FTF-ENT-F0104

Ed Martinez
CodeWarrior Software Analysis

June 2012

Agenda

- **High Level introduction to the QorIQ Debug IP**
  - Trace support
  - Performance support
- **Freescale tools**
  - Trace tools
  - Performance analysis tools
- **Common Use Cases**
  - Cores
  - Memory related
  - Other SoC components (DPAA)
- **Q&A**

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Introduction to the QorIQ Debug IP

Stage – What Do We Need to Think About?

- Accessibility
- Concurrency
- Bandwidth
- Intrusiveness
- Security
- The answer?

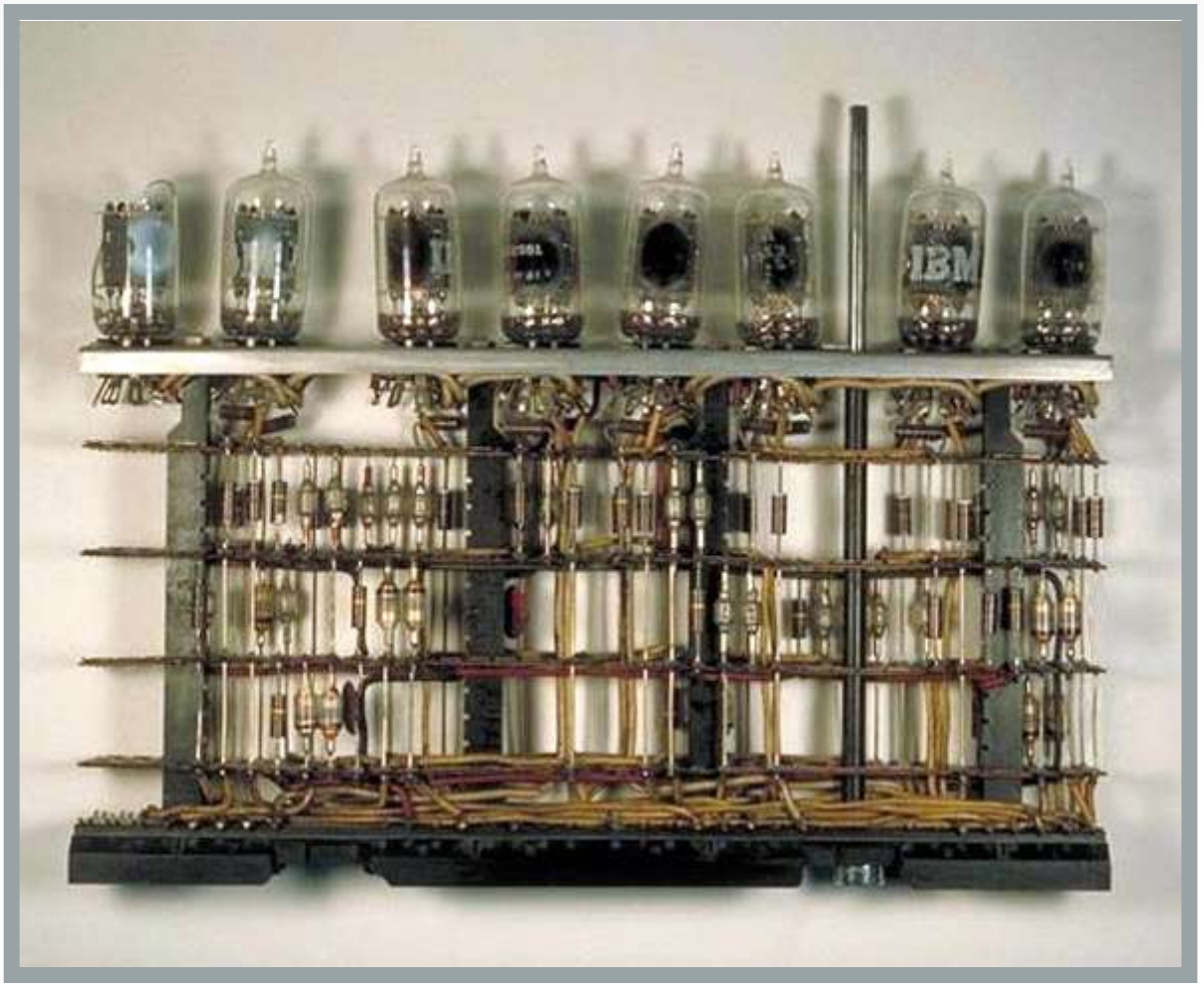
FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Accessibility

Physical Access

One Bit

Physical Access

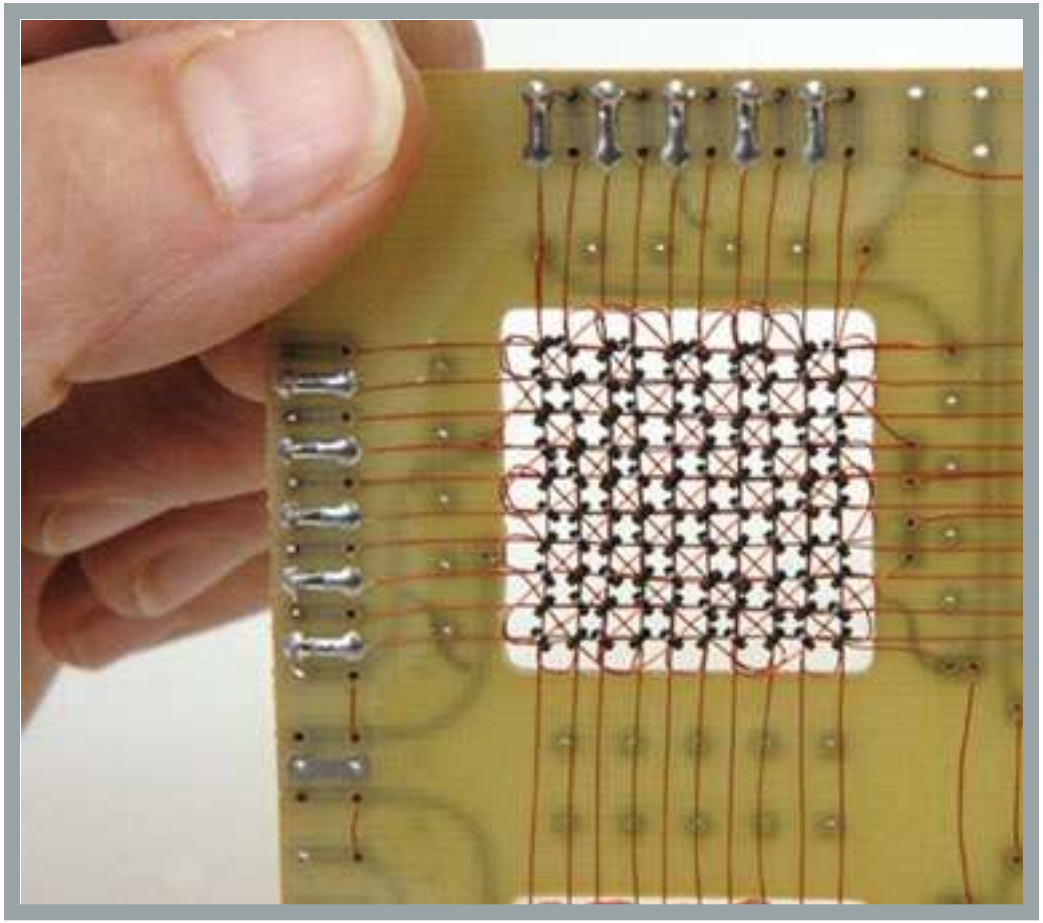


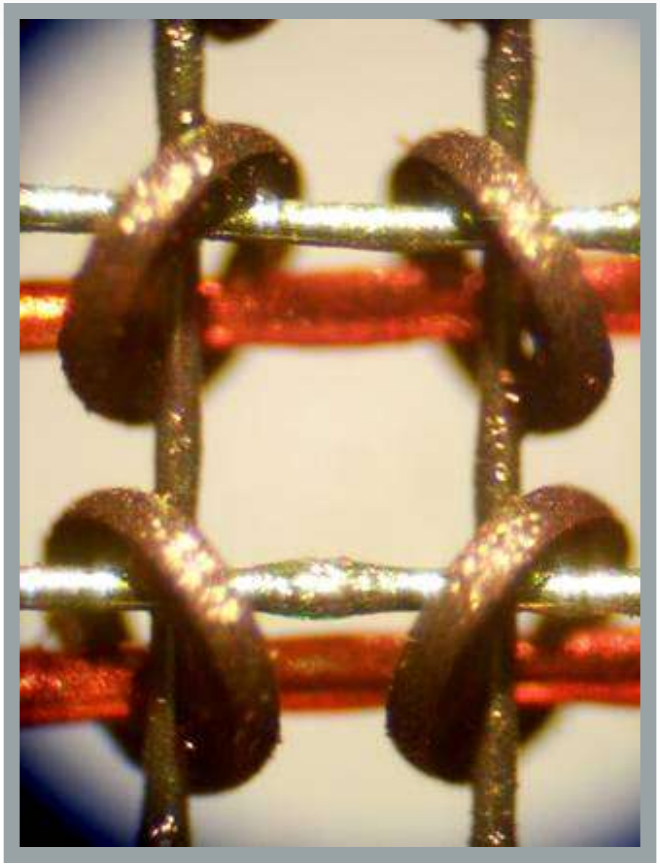
One Byte

Physical Access

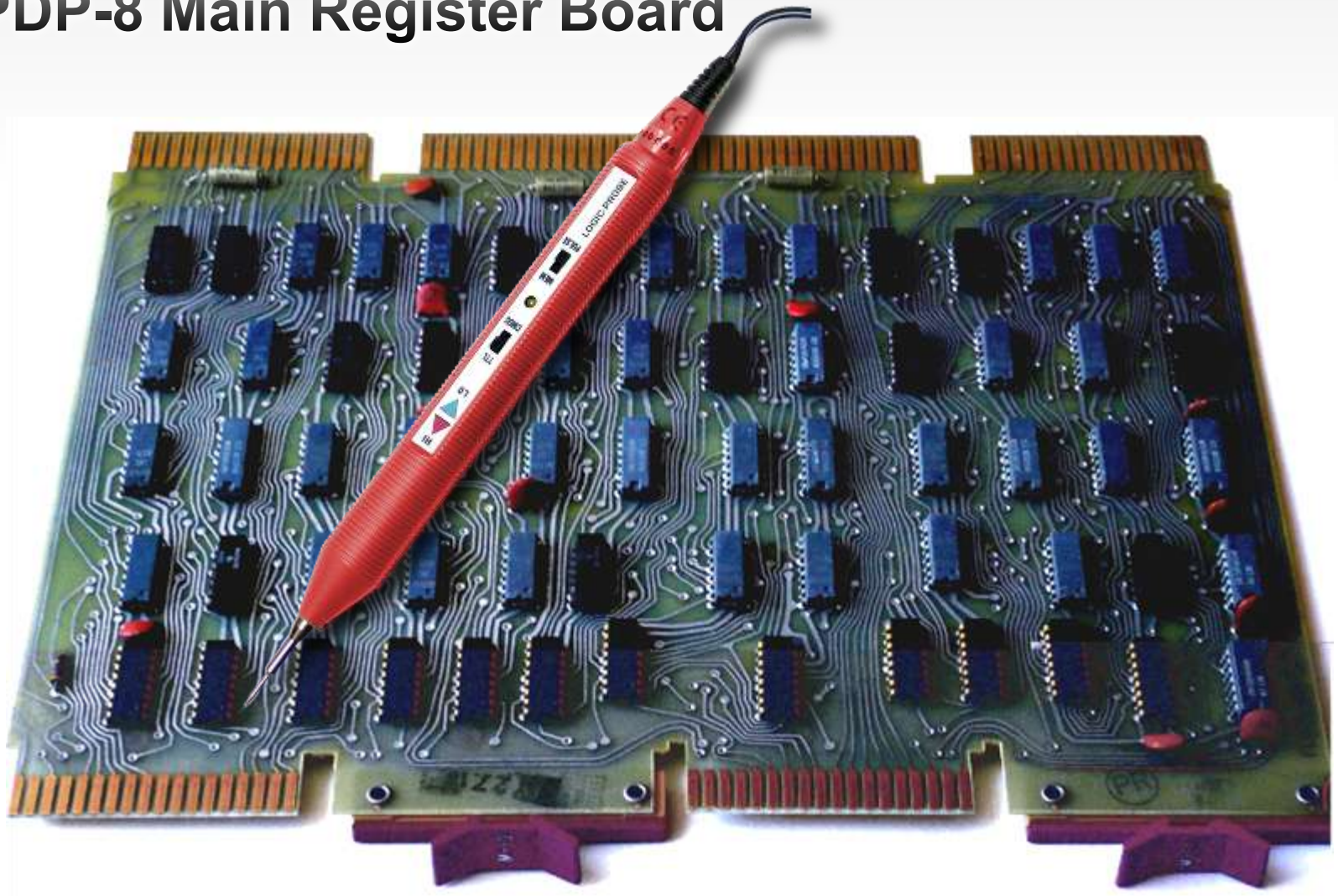
One Byte

Physical Access: Magnetic Core Memory



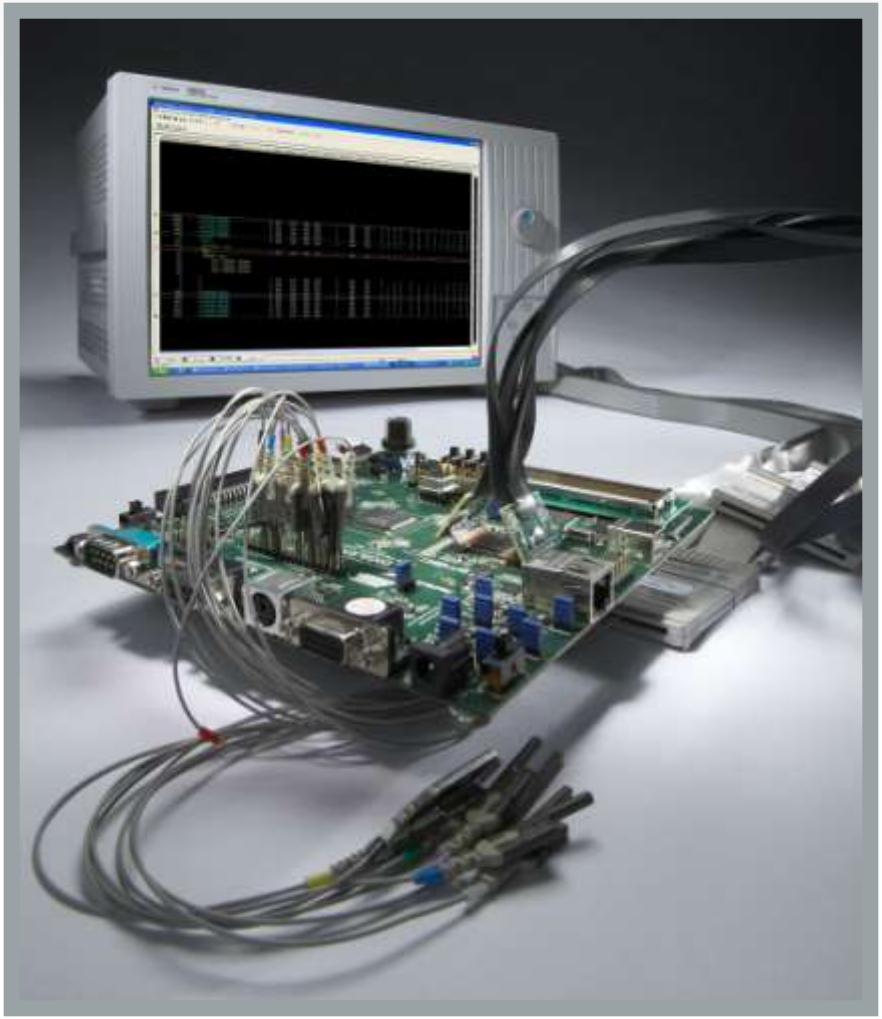


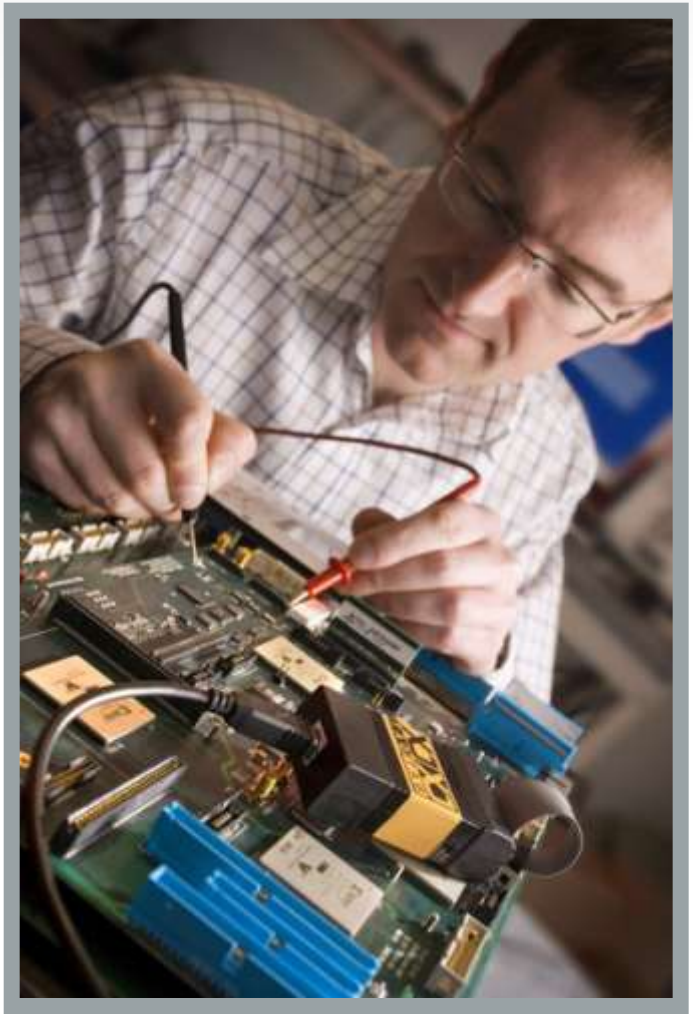
PDP-8 Main Register Board



External Cache and Math Coprocessors

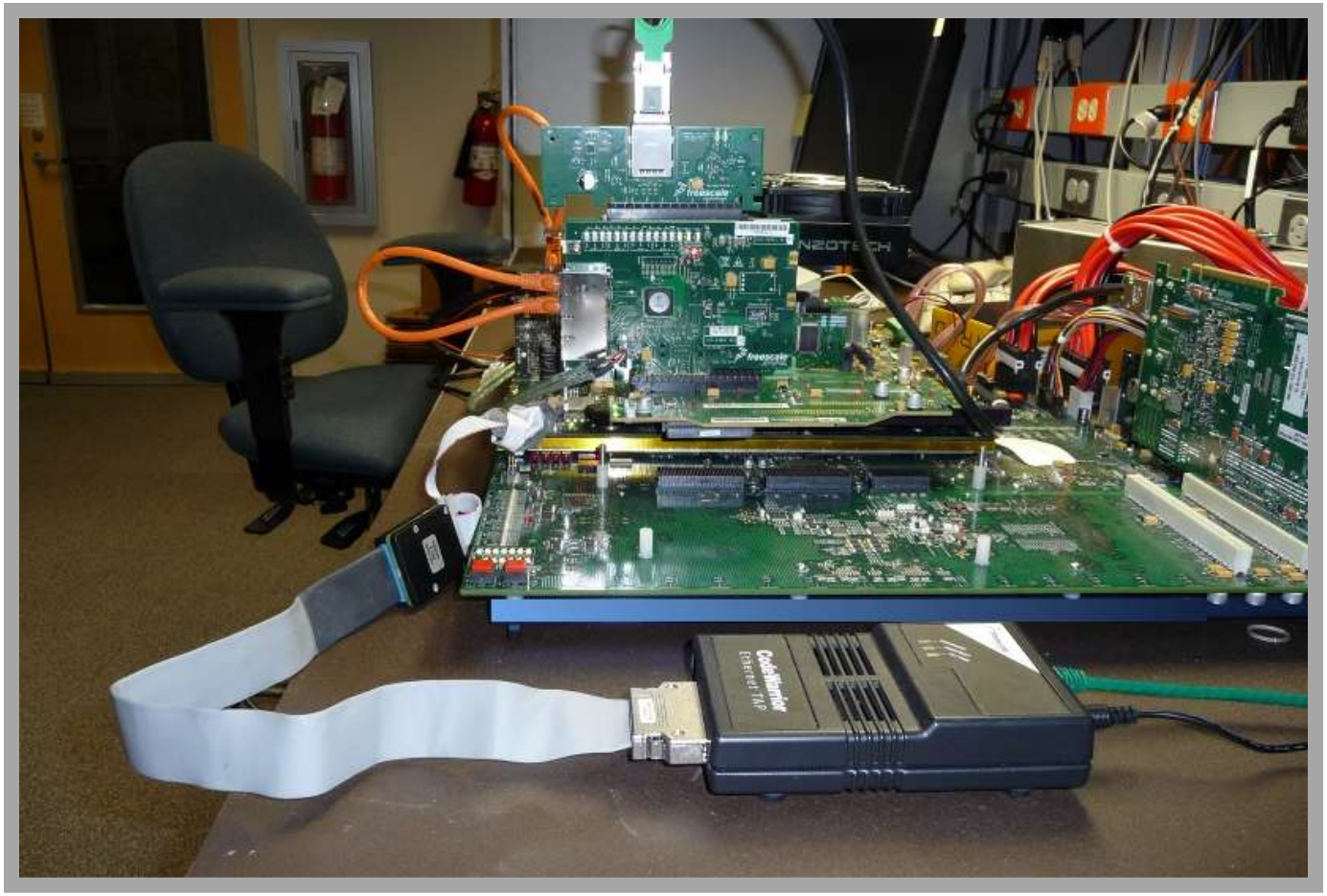
Board-Level Access



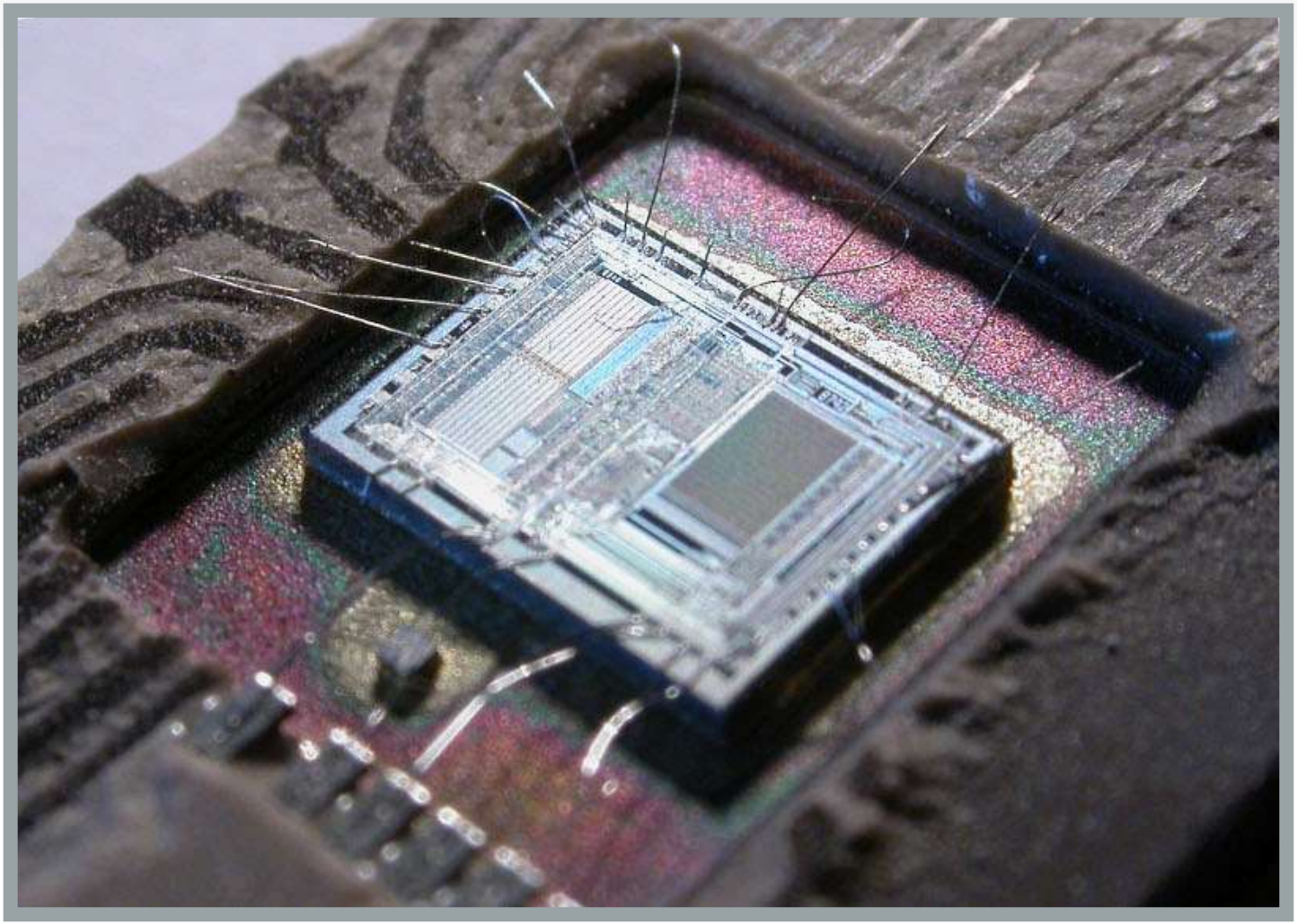


Ethernet or USB JTAG Probe

- **Pros:** Ubiquitous. Inexpensive, Single-socket, Daisy-Chainable



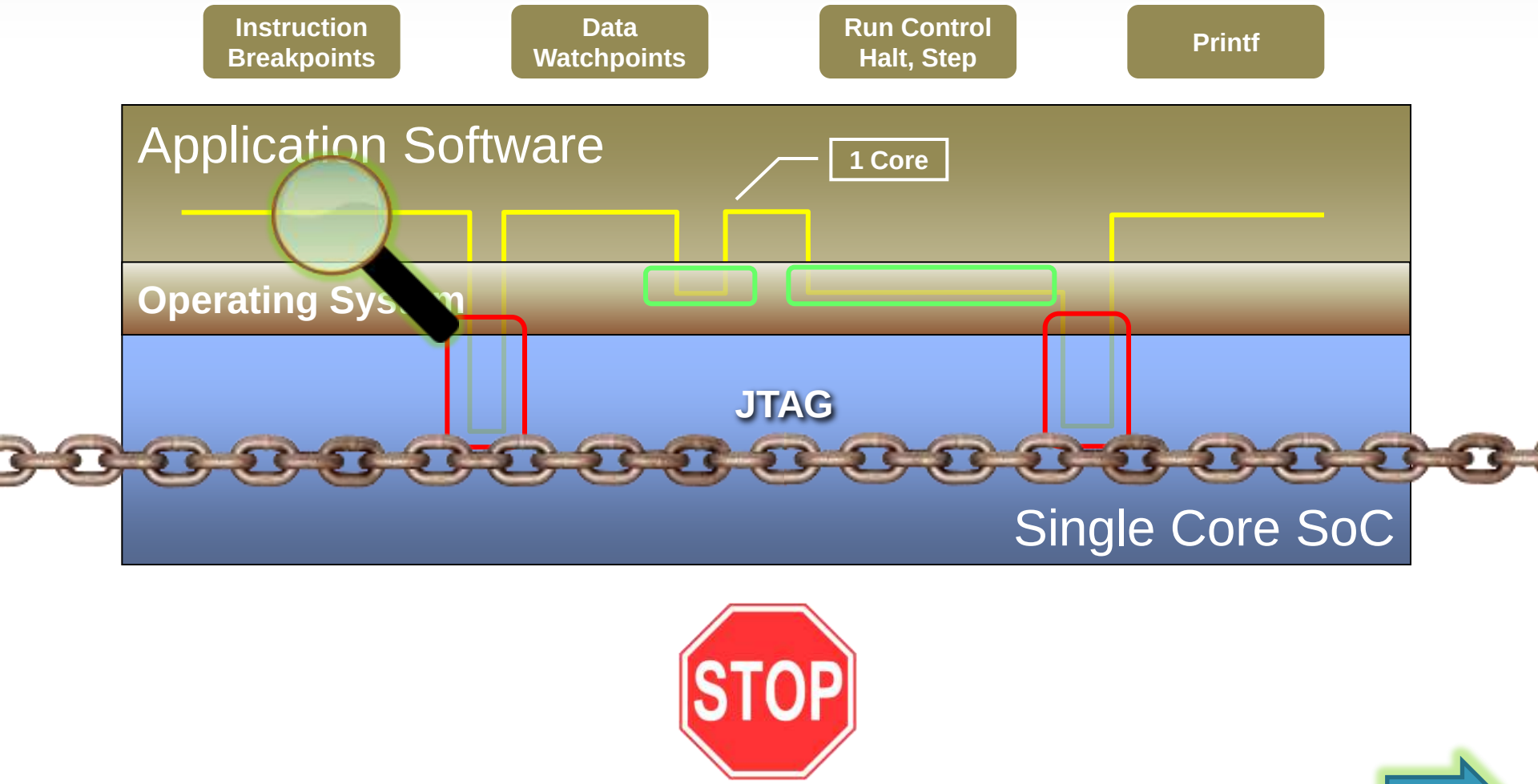
All on One Die



All on One Die

Next

Single Core – Software Debug

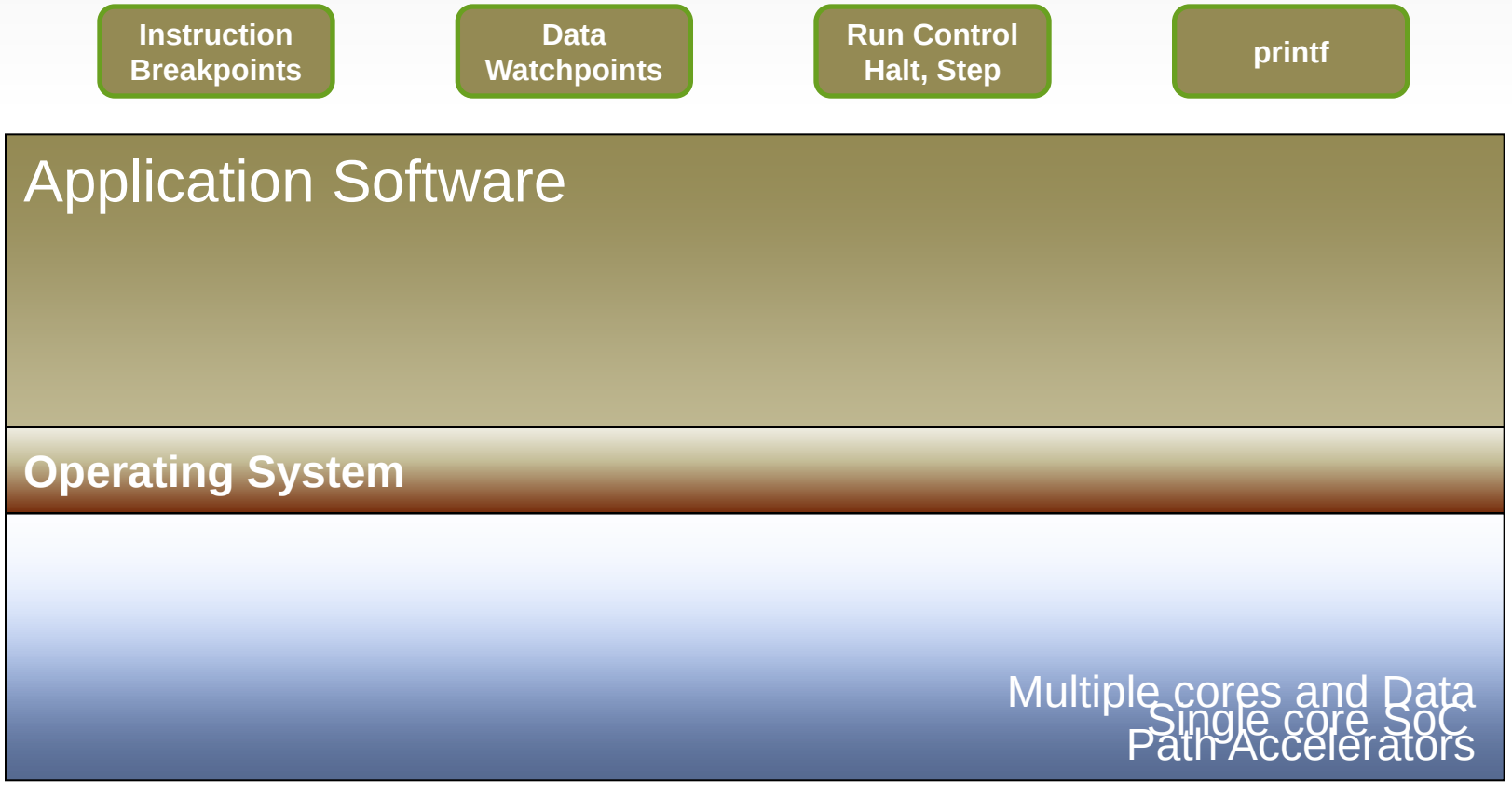


Next

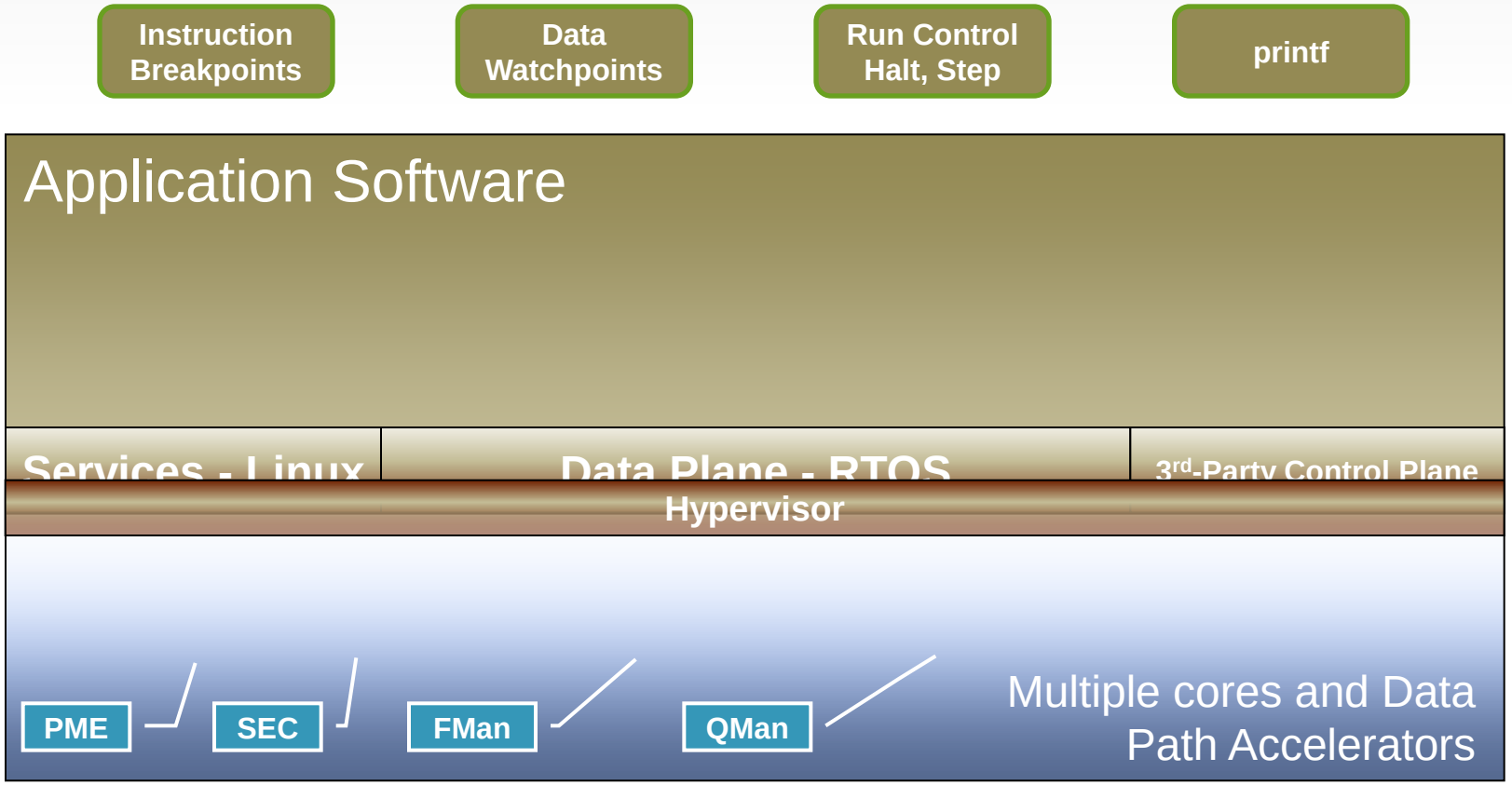
FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Concurrency

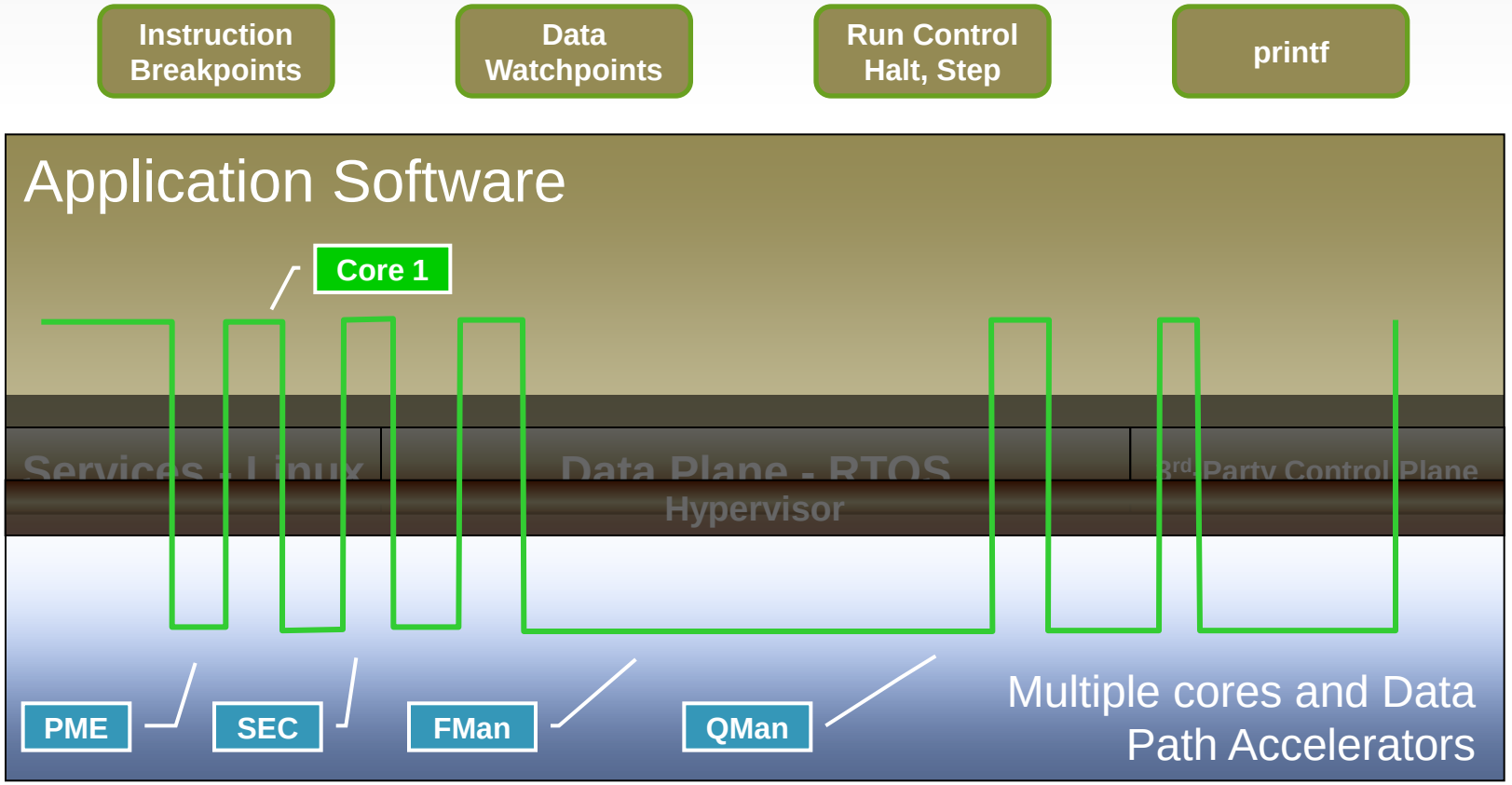
Multicore Application Migration



Multicore Application Migration



Multicore Application Migration



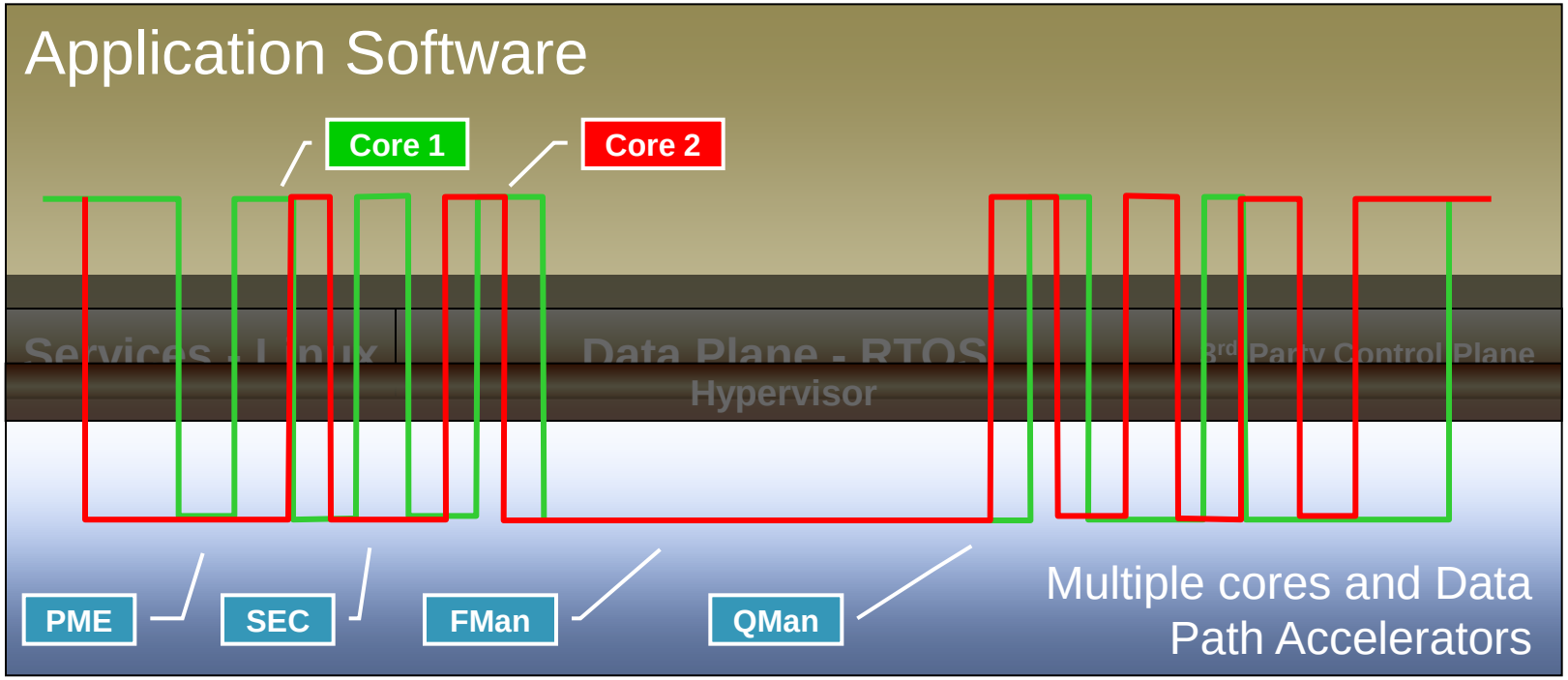
Multicore Application Migration

Instruction Breakpoints

Data Watchpoints

Run Control Halt, Step

printf



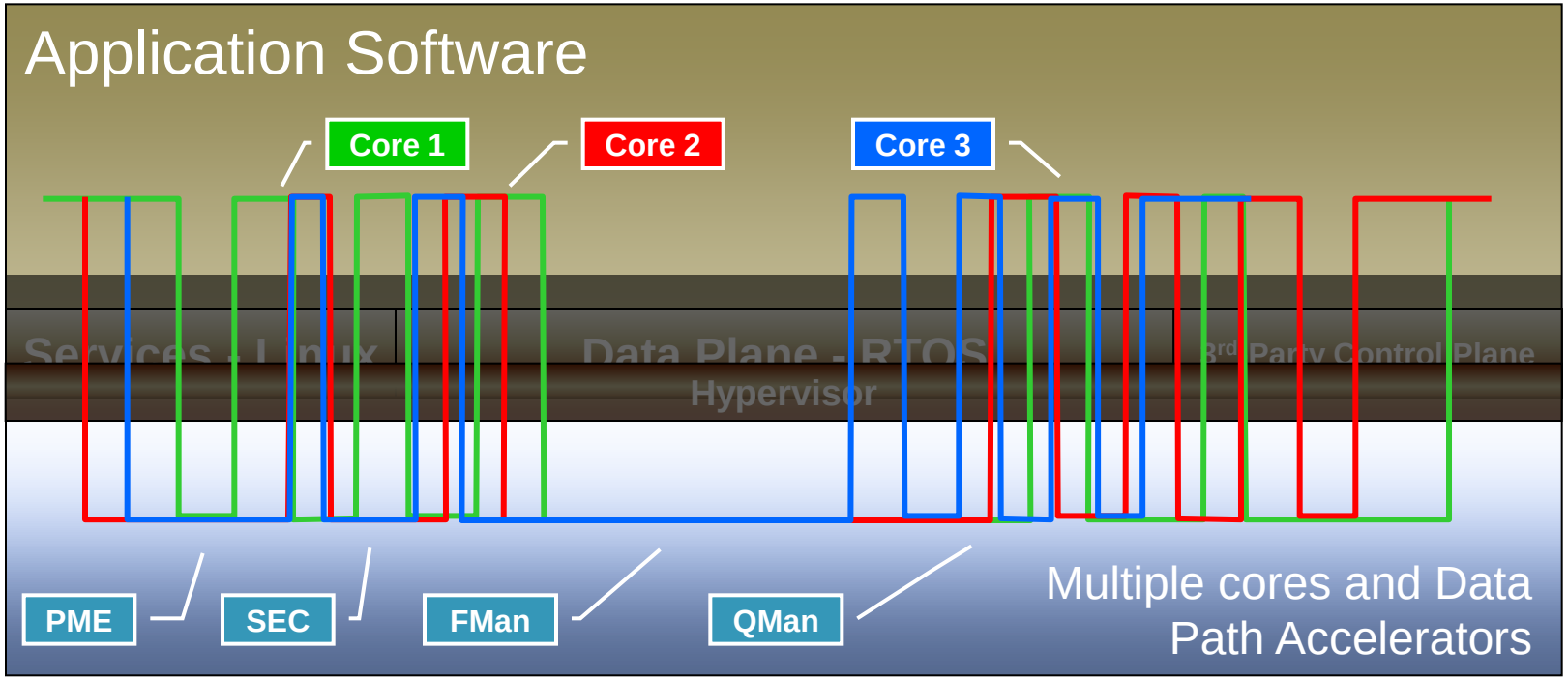
Multicore Application Migration

Instruction Breakpoints

Data Watchpoints

Run Control Halt, Step

printf



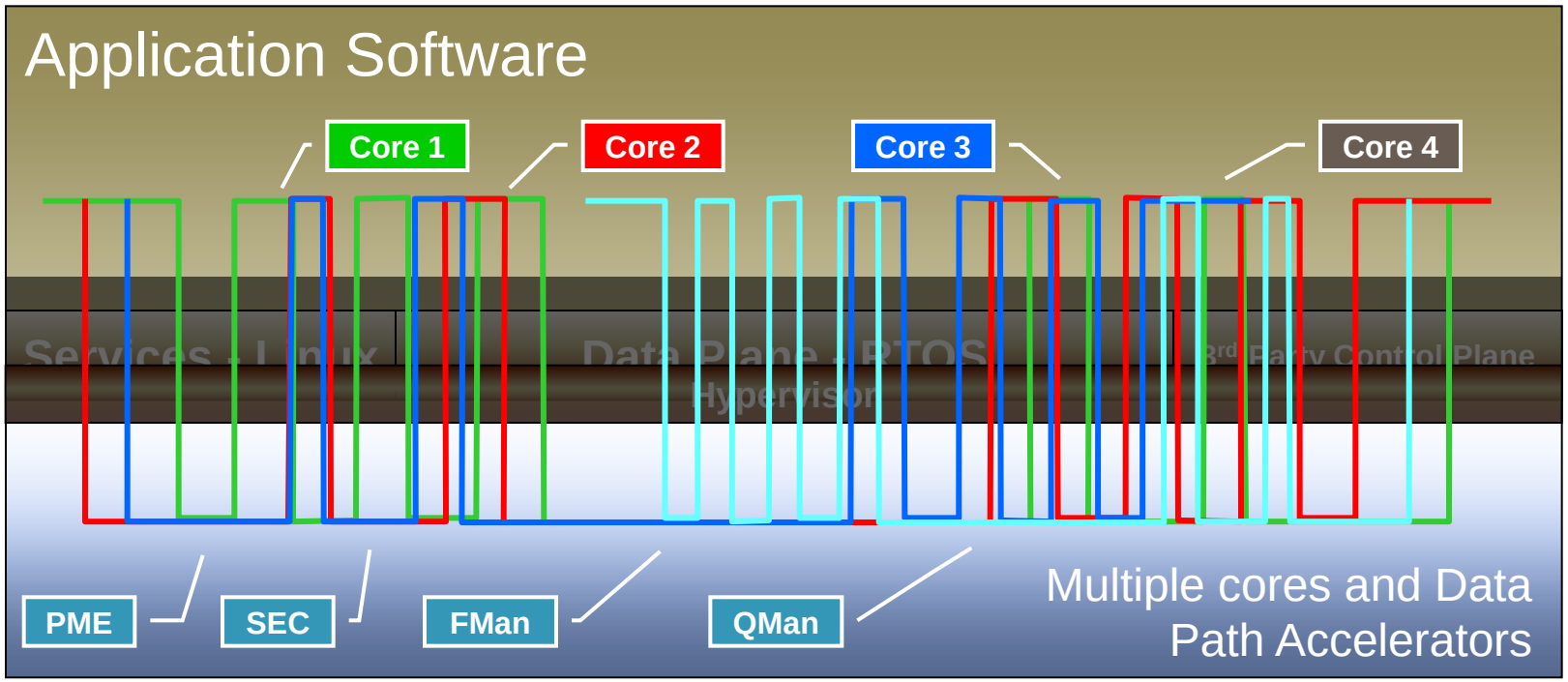
Multicore Application Migration

Instruction Breakpoints

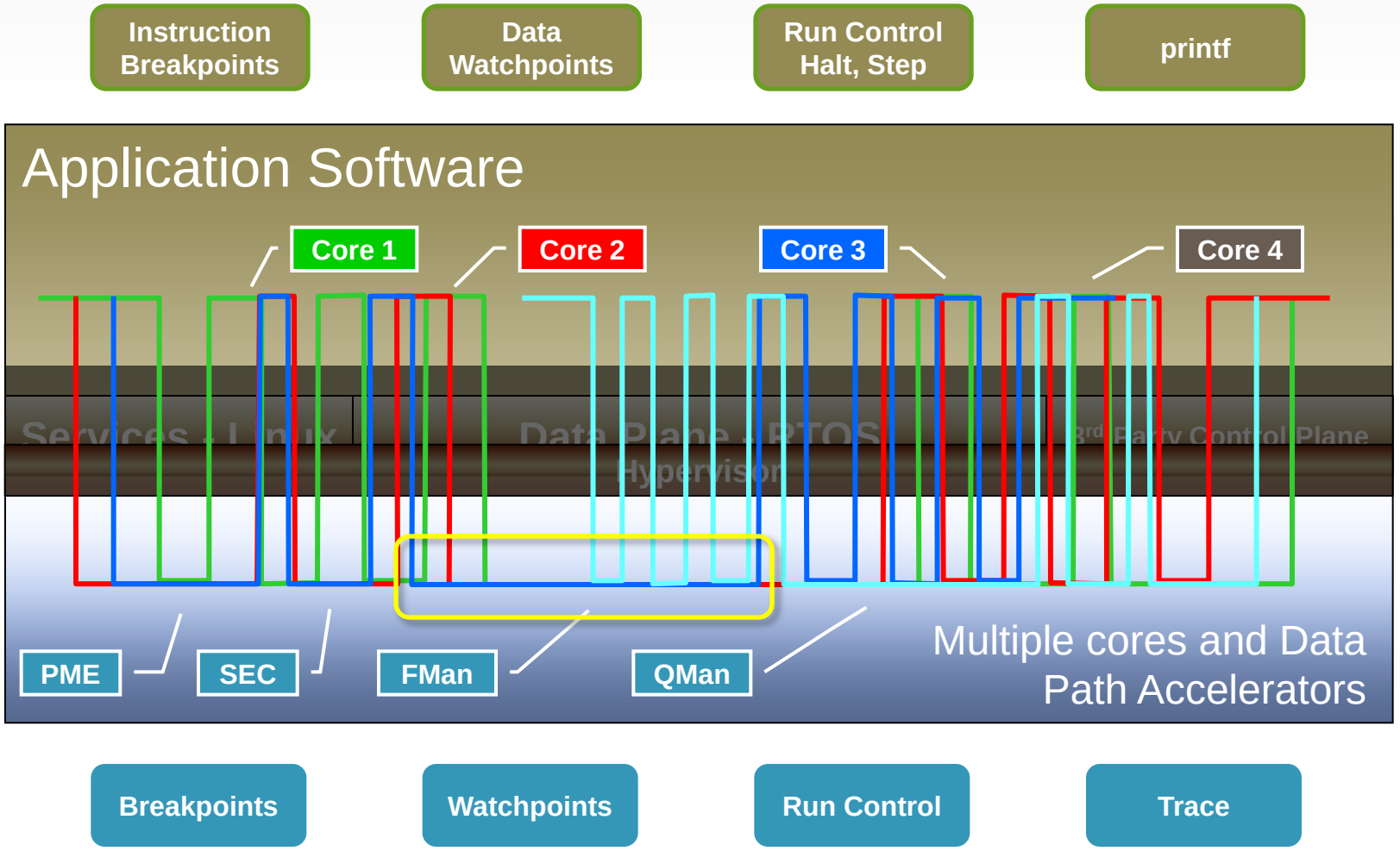
Data Watchpoints

Run Control Halt, Step

printf

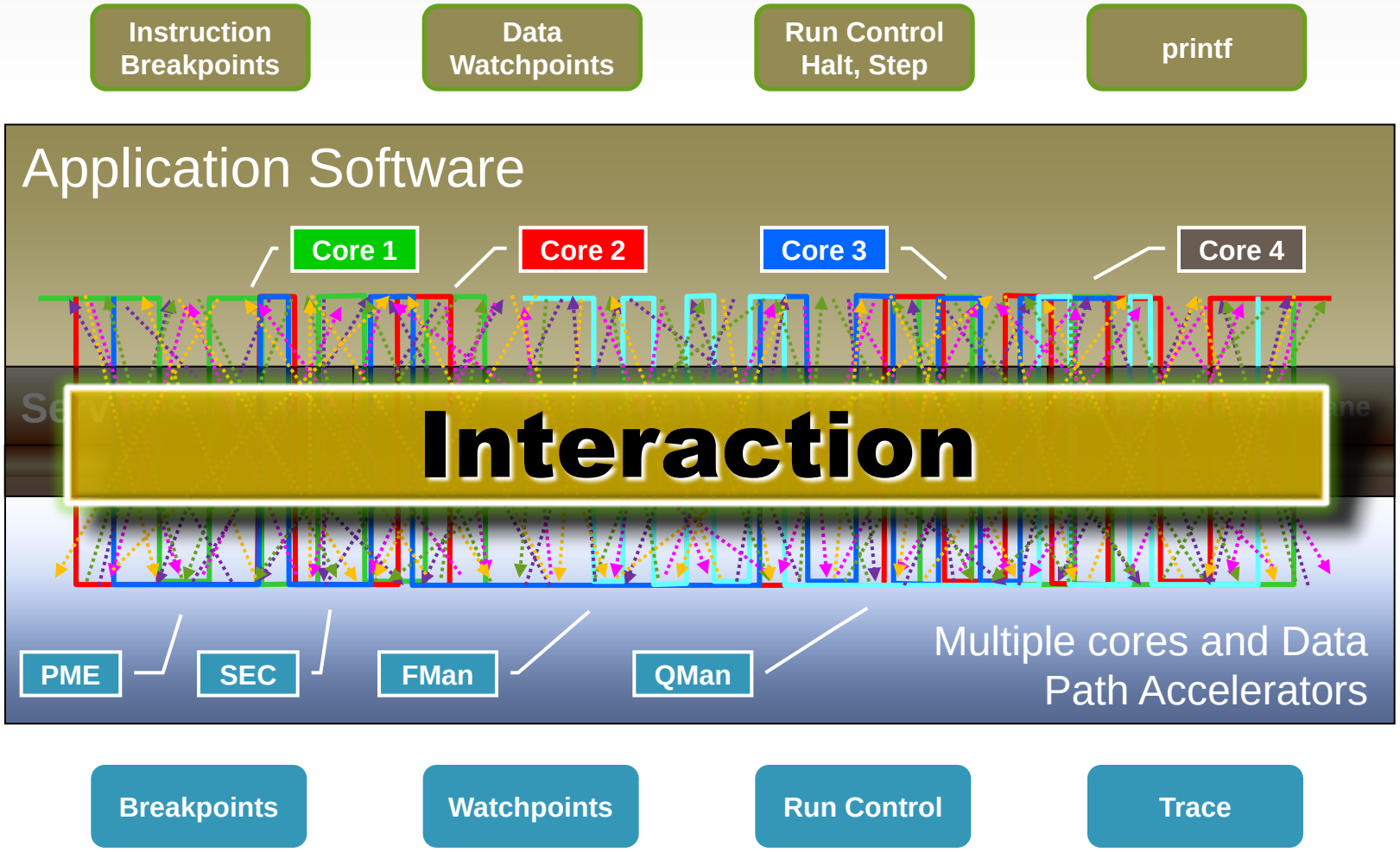


Multicore Application Migration



Next

Multicore Application Migration



Next

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Bandwidth and Intrusiveness

JTAG Limitations

- **JTAG**
  - Sample rate
  - Bandwidth
- **P4080 Debug Trace Sources**
  - 8 x Cores at 1.5 GHz
  - 2 x DDR controllers
  - L1, L2, L3 caches
  - 2 x Frame Managers
    - 1 x 10G Ethernet controllers
    - 4 x 1G Ethernet controllers
  - Queue Manager
  - 3 x PCIe controllers
  - 2 x serial RapidIO™ controllers

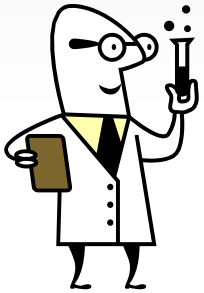
Next

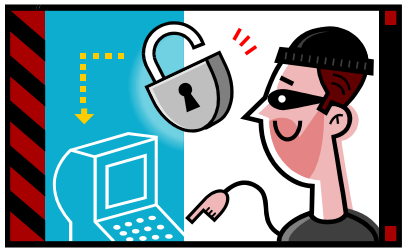
FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Security

Two Sides to Accessibility

- **Accessibility is a double-edged sword**
  - Great during development process
  - Great during lab or field debug
- **However**
  - Provides an attack surface for hackers
- **Access Methods**
  - Debugger
    - High-Speed Serial
    - JTAG
  - Target-Resident Code Instrumentation

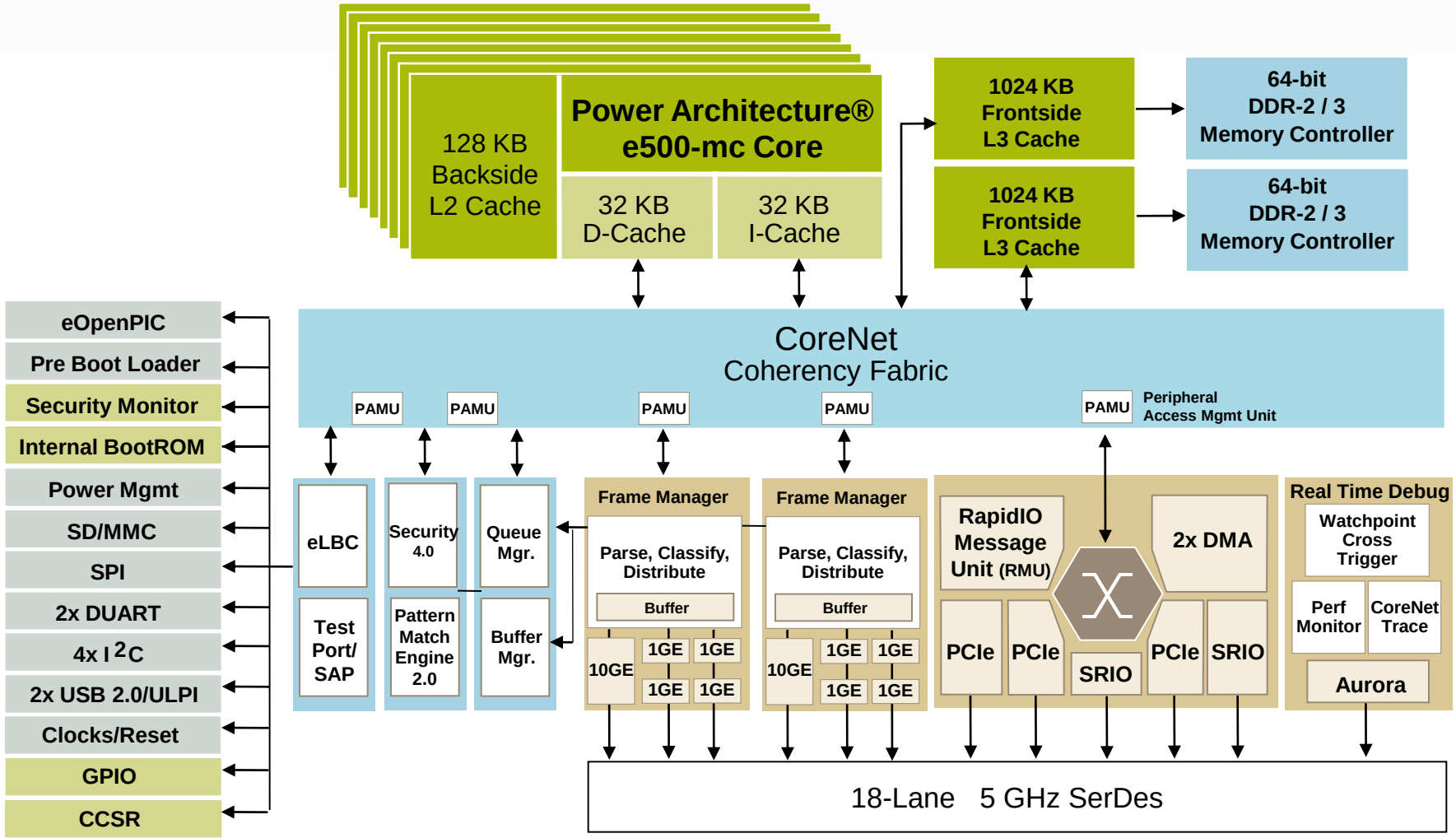




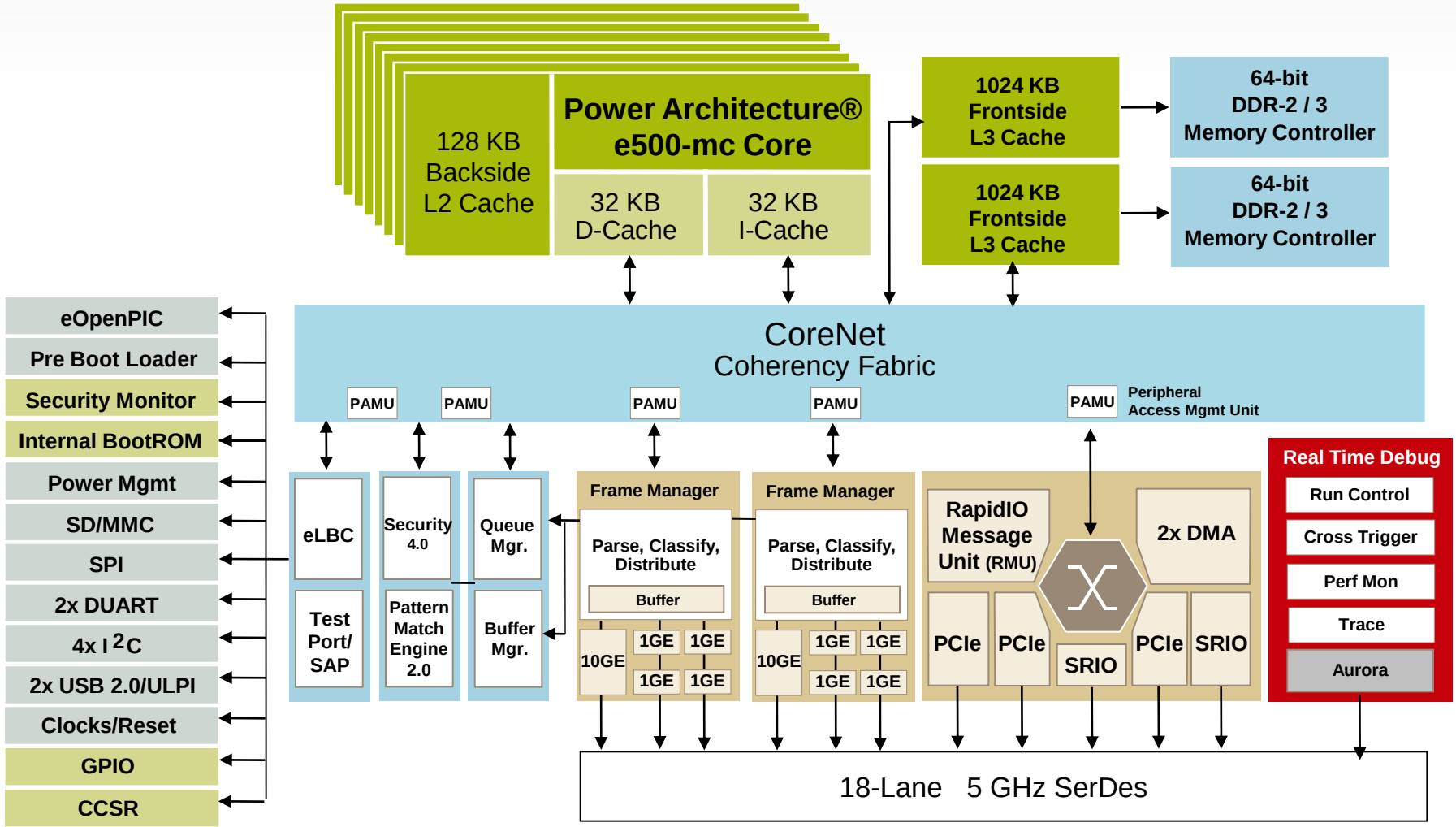
FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

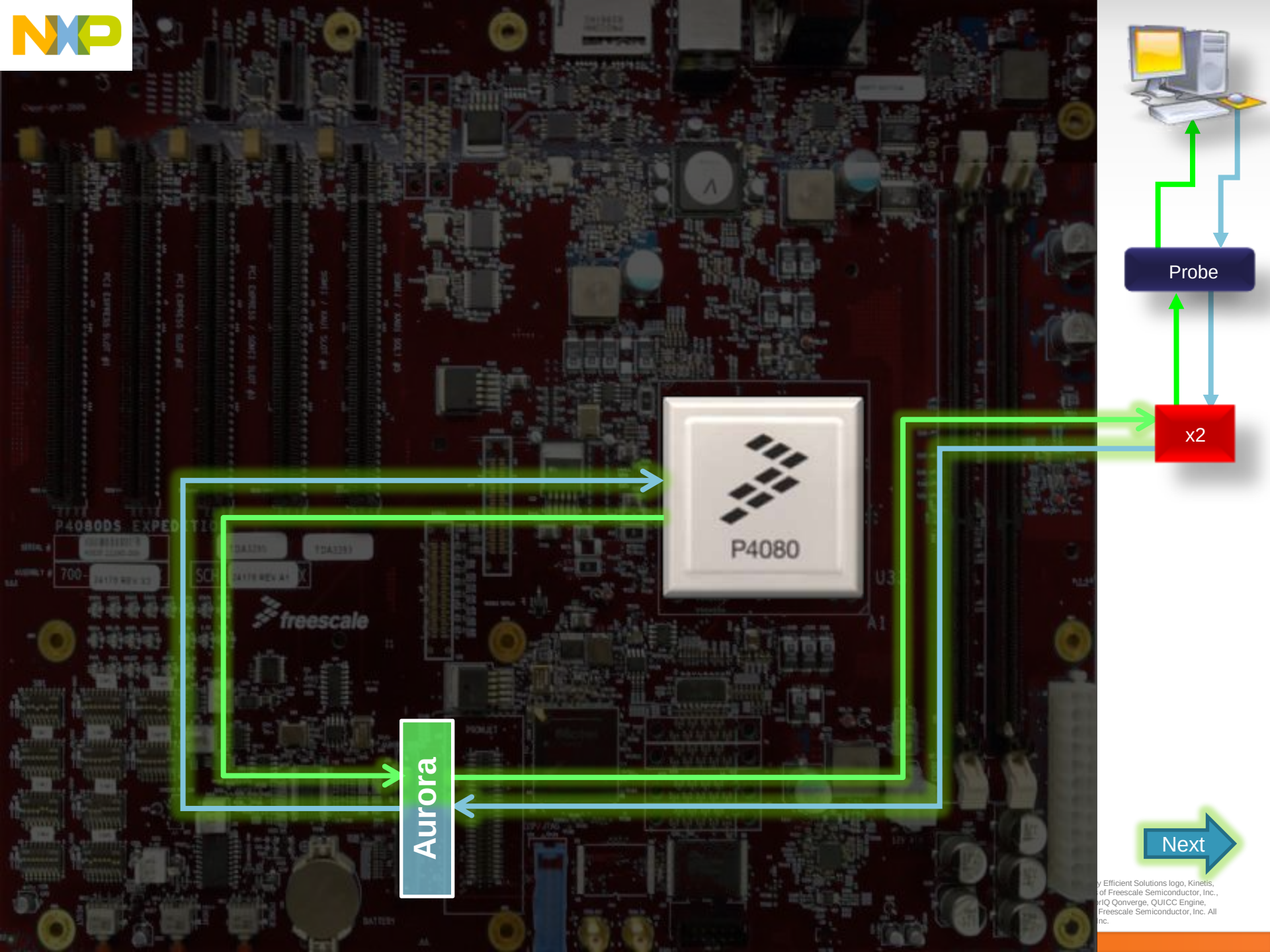
Solution: Advanced QorIQ Debug and Performance Monitoring Architecture

QorIQ P4 Series P4080 Block Diagram

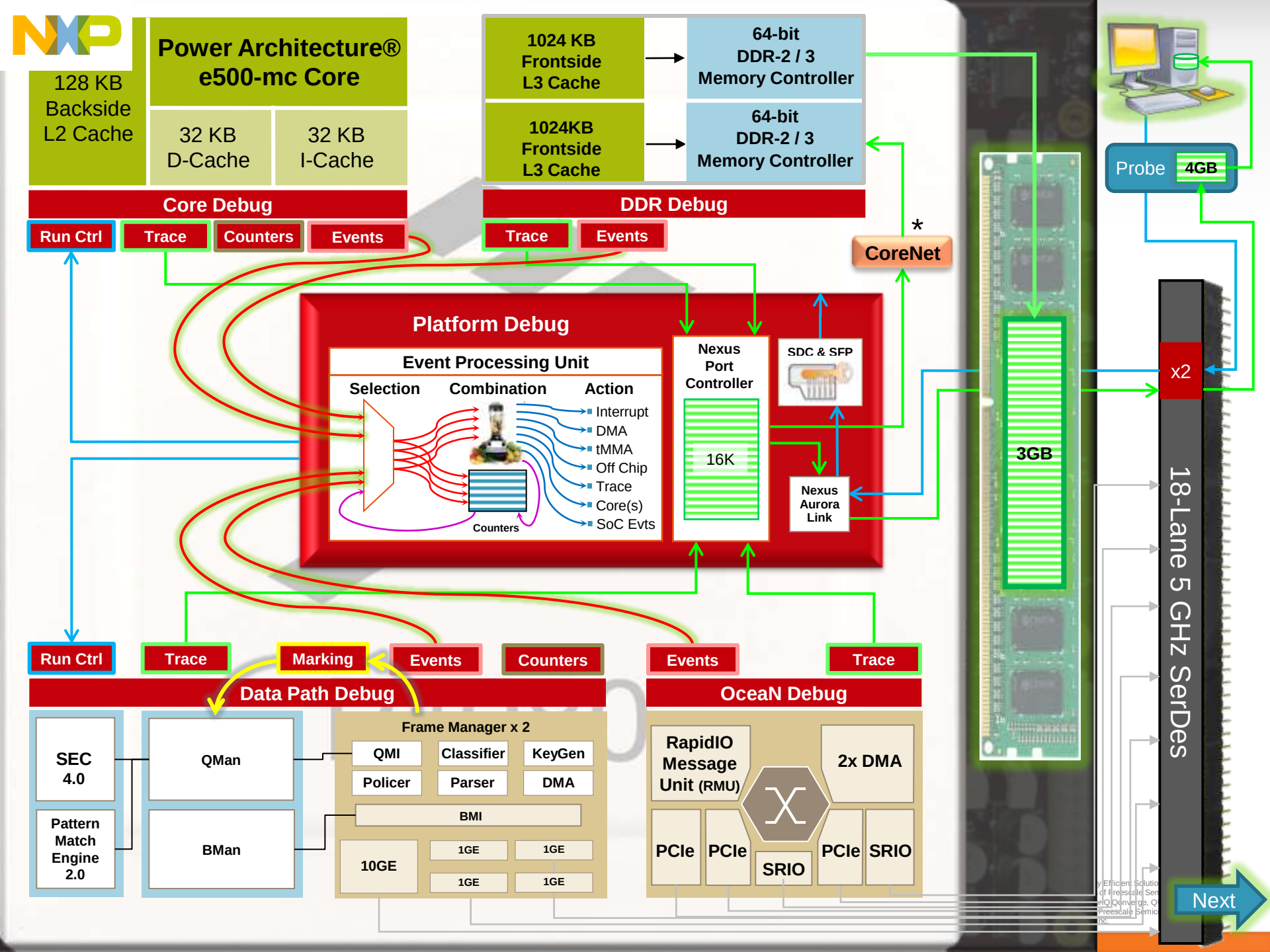


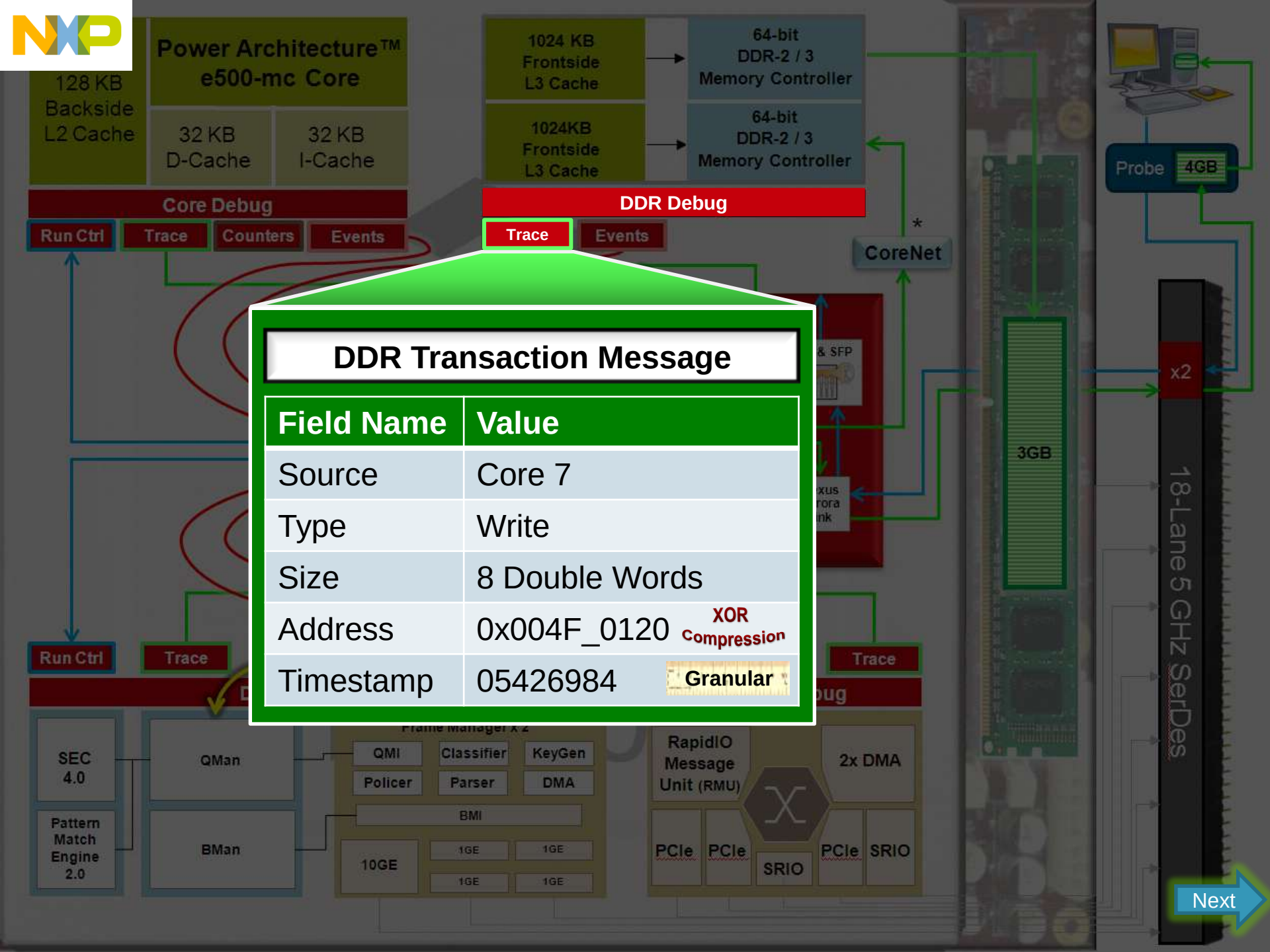
QorIQ P4 Series P4080 Block Diagram





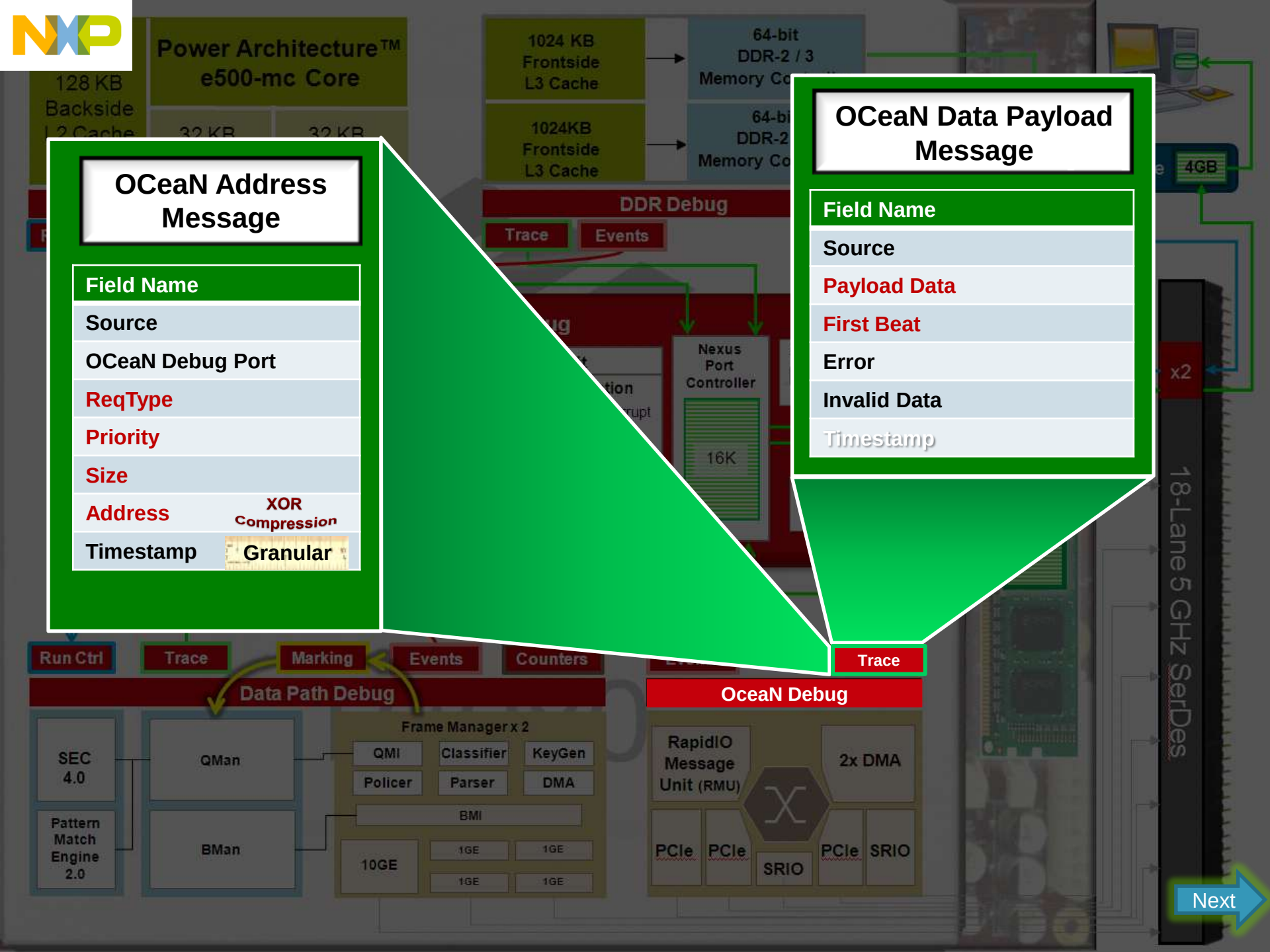
Next

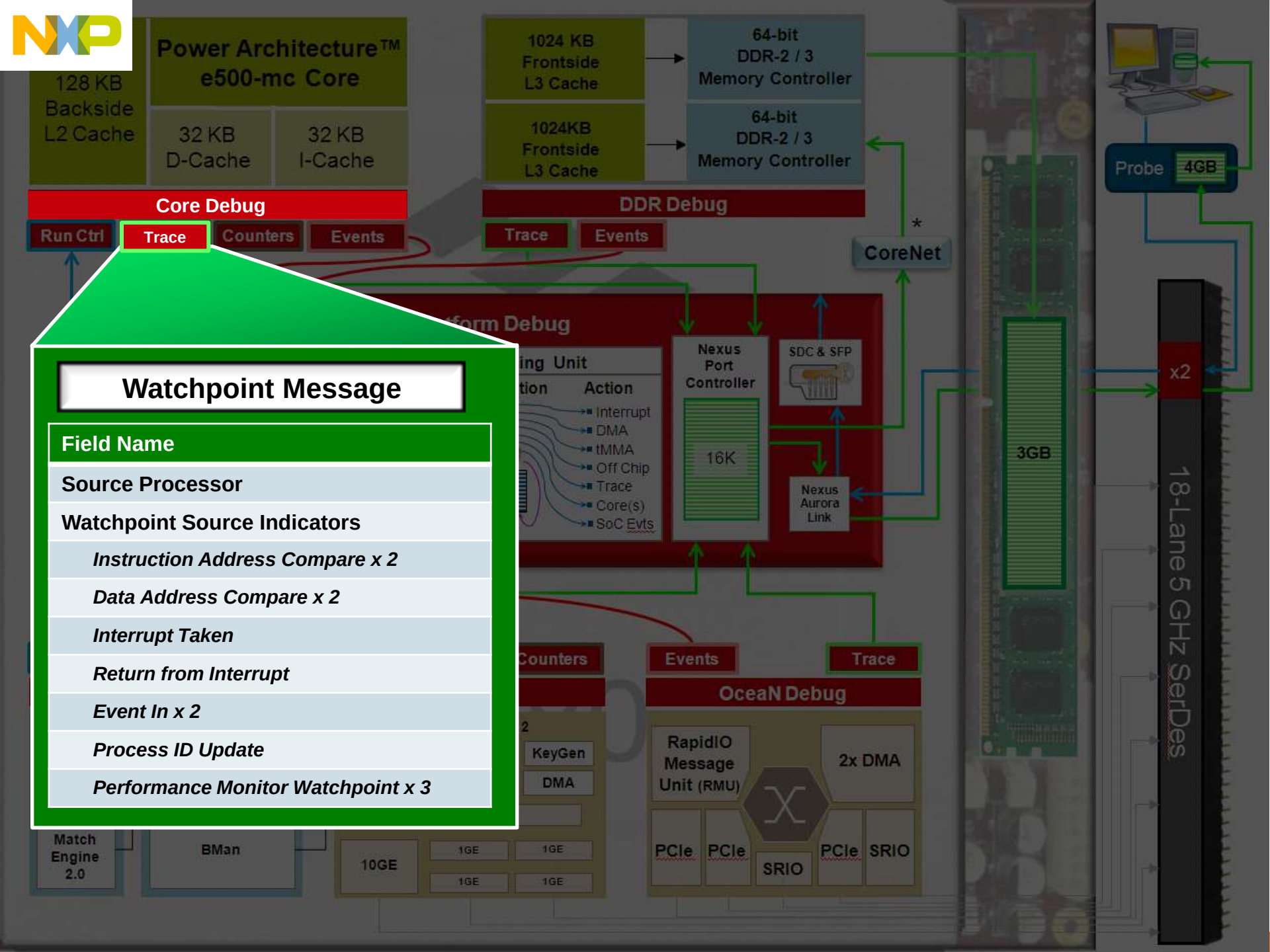

Power Architecture® e500-mc Core
128 KB Backside L2 Cache
32 KB D-Cache
32 KB I-Cache
1024 KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
1024KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
Core Debug
Run Ctrl
Trace
Counters
Events
DDR Debug
Trace
Events
CoreNet
Platform Debug
Event Processing Unit
Selection
Combination
Action
Interrupt
DMA
tMMA
Off Chip
Trace
Core(s)
SoC Evts
Counters
Nexus Port Controller
16K
SDC & SFP
Nexus Aurora Link
Probe
4GB
3GB
x2
18-Lane 5 GHz SerDes
Run Ctrl
Trace
Marking
Events
Counters
Events
Trace
Data Path Debug
SEC 4.0
Pattern Match Engine 2.0
QMan
BMan
Frame Manager x 2
QMI
Classifier
KeyGen
Policer
Parser
DMA
BMI
10GE
1GE
1GE
1GE
1GE
OceaN Debug
RapidIO Message Unit (RMU)
2x DMA
PCIe
PCIe
SRIO
PCIe
SRIO
Next



DDR Transaction Message

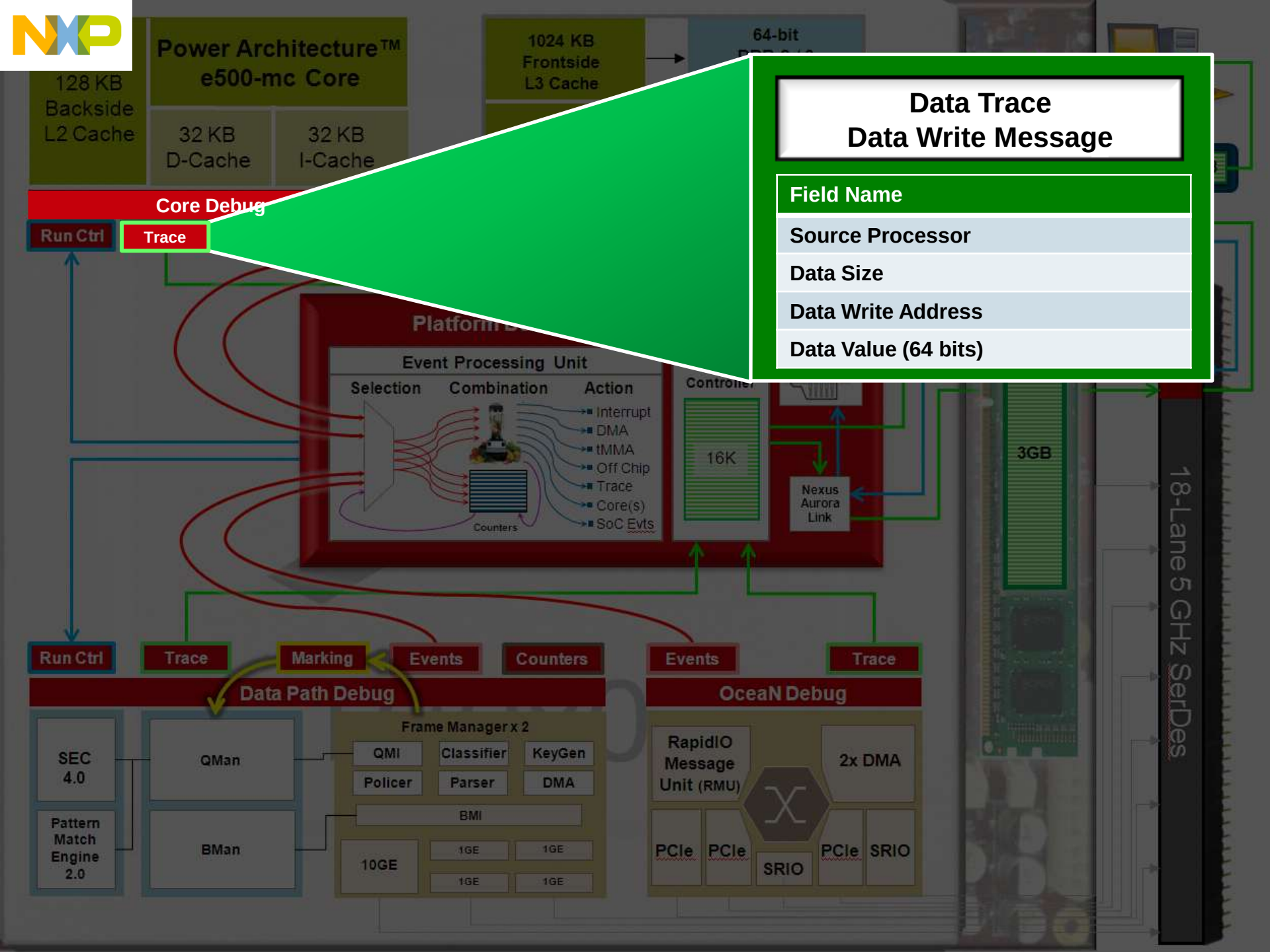
Field Name	Value
Source	Core 7
Type	Write
Size	8 Double Words
Address	0x004F_0120 XOR Compression
Timestamp	05426984 Granular


Power Architecture™ e500-mc Core
128 KB Backside L2 Cache
1024 KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Co
1024KB Frontside L3 Cache
64-bi DDR-2 Memory Co
DDR Debug
Trace
Events
Nexus Port Controller
16K
4GB
x2
18-Lane 5 GHz SerDes
OCeaN Address Message
Field Name
Source
OCeaN Debug Port
ReqType
Priority
Size
Address
XOR Compression
Timestamp
Granular
OCeaN Data Payload Message
Field Name
Source
Payload Data
First Beat
Error
Invalid Data
Timestamp
Run Ctrl
Trace
Marking
Events
Counters
Data Path Debug
SEC 4.0
Pattern Match Engine 2.0
QMan
BMan
Frame Manager x 2
QMI
Classifier
KeyGen
Policer
Parser
DMA
BMI
10GE
1GE
1GE
1GE
1GE
Trace
OceaN Debug
RapidIO Message Unit (RMU)
2x DMA
PCIe
PCIe
SRIO
PCIe
SRIO
Next



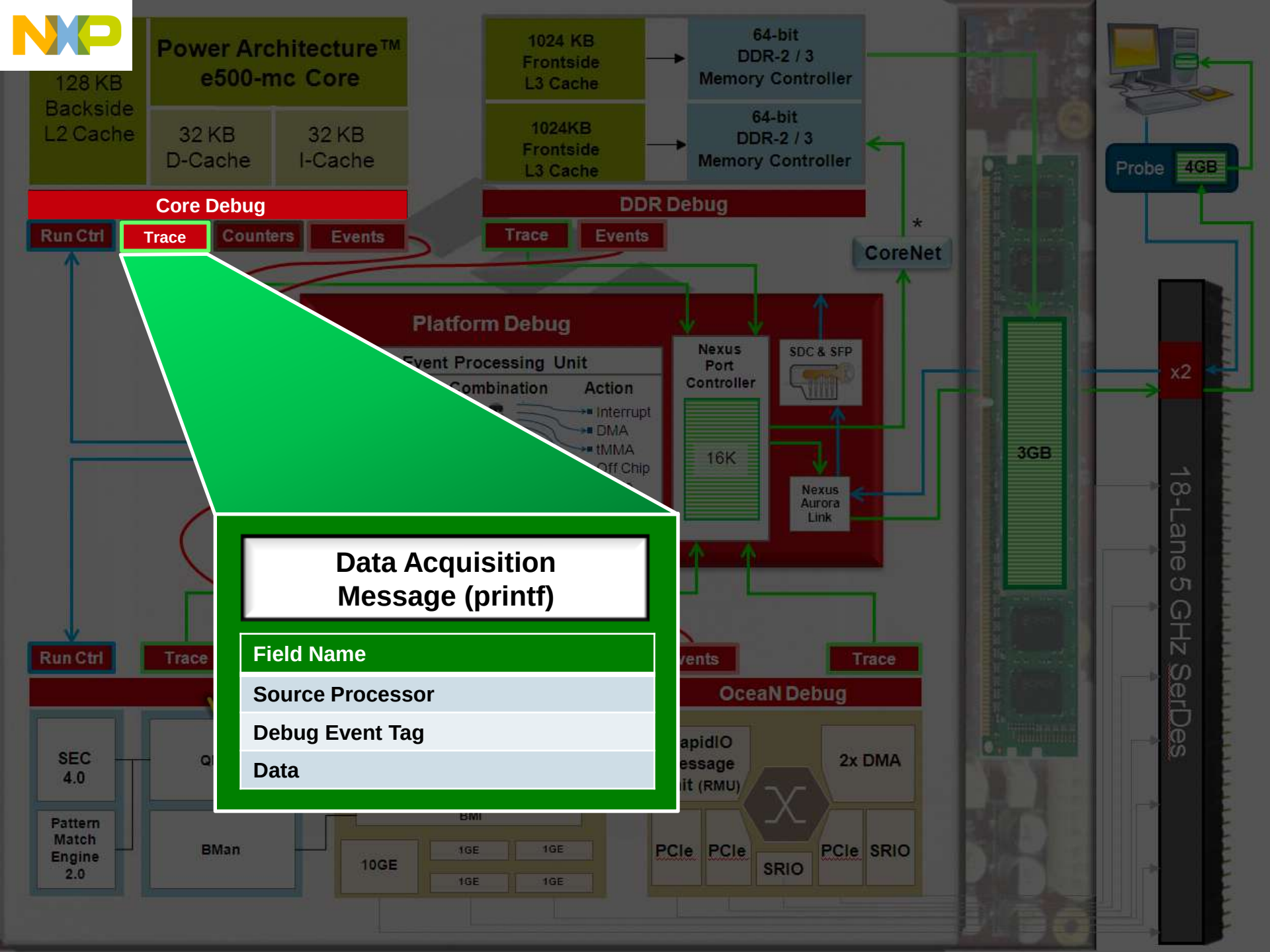
Watchpoint Message

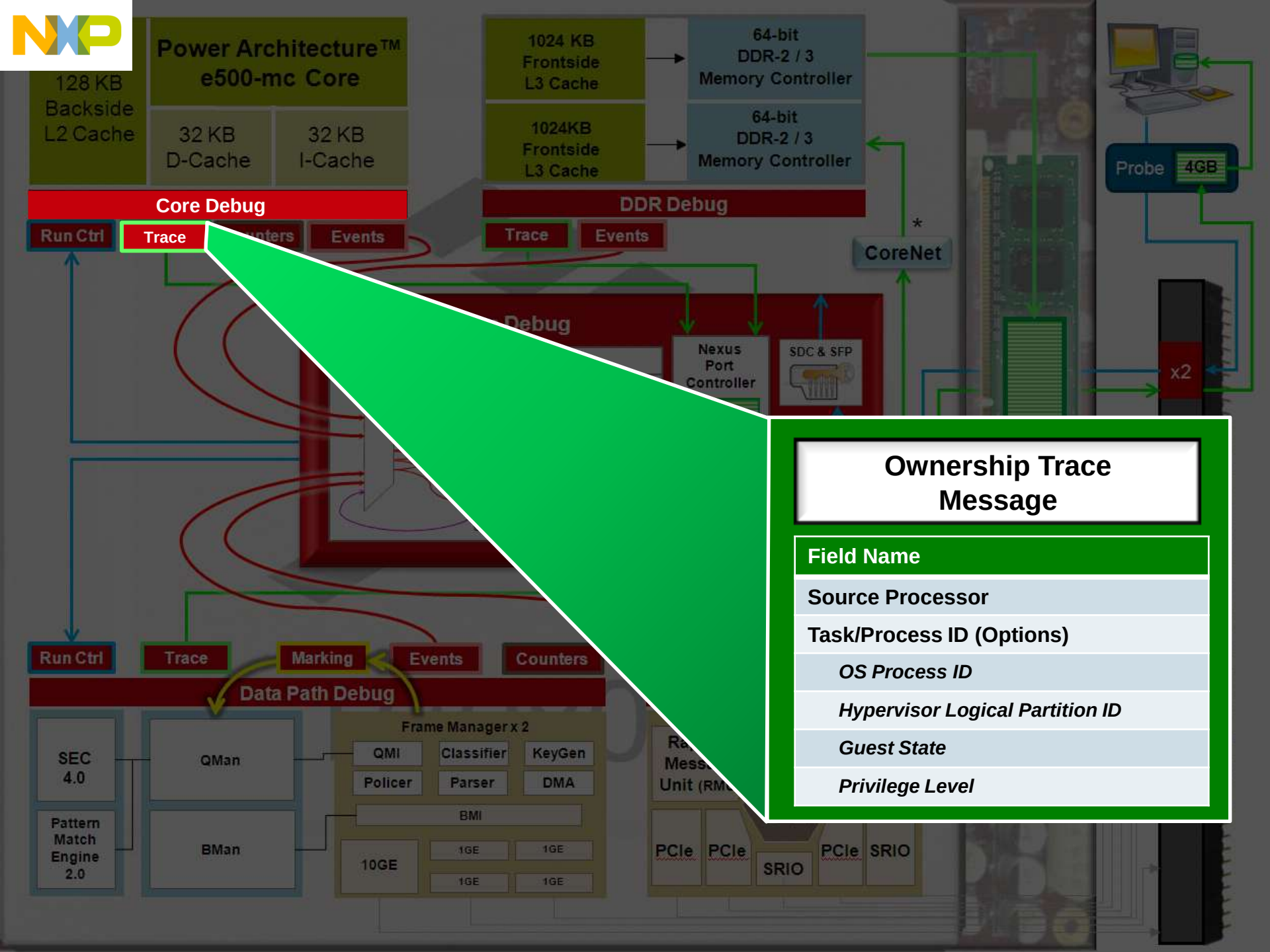
Field Name
Source Processor
Watchpoint Source Indicators
Instruction Address Compare x 2
Data Address Compare x 2
Interrupt Taken
Return from Interrupt
Event In x 2
Process ID Update
Performance Monitor Watchpoint x 3



Data Trace
Data Write Message

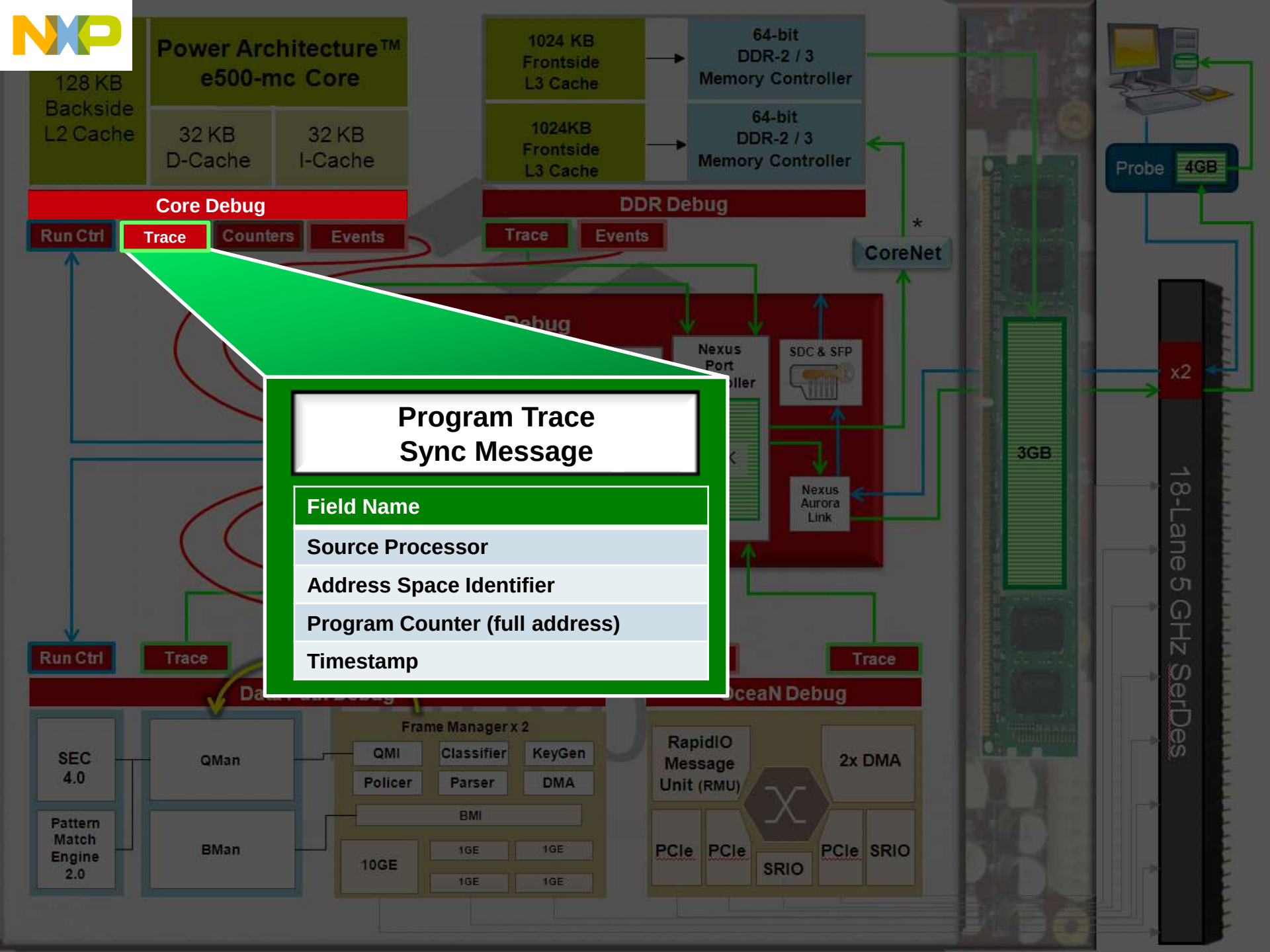
Field Name
Source Processor
Data Size
Data Write Address
Data Value (64 bits)

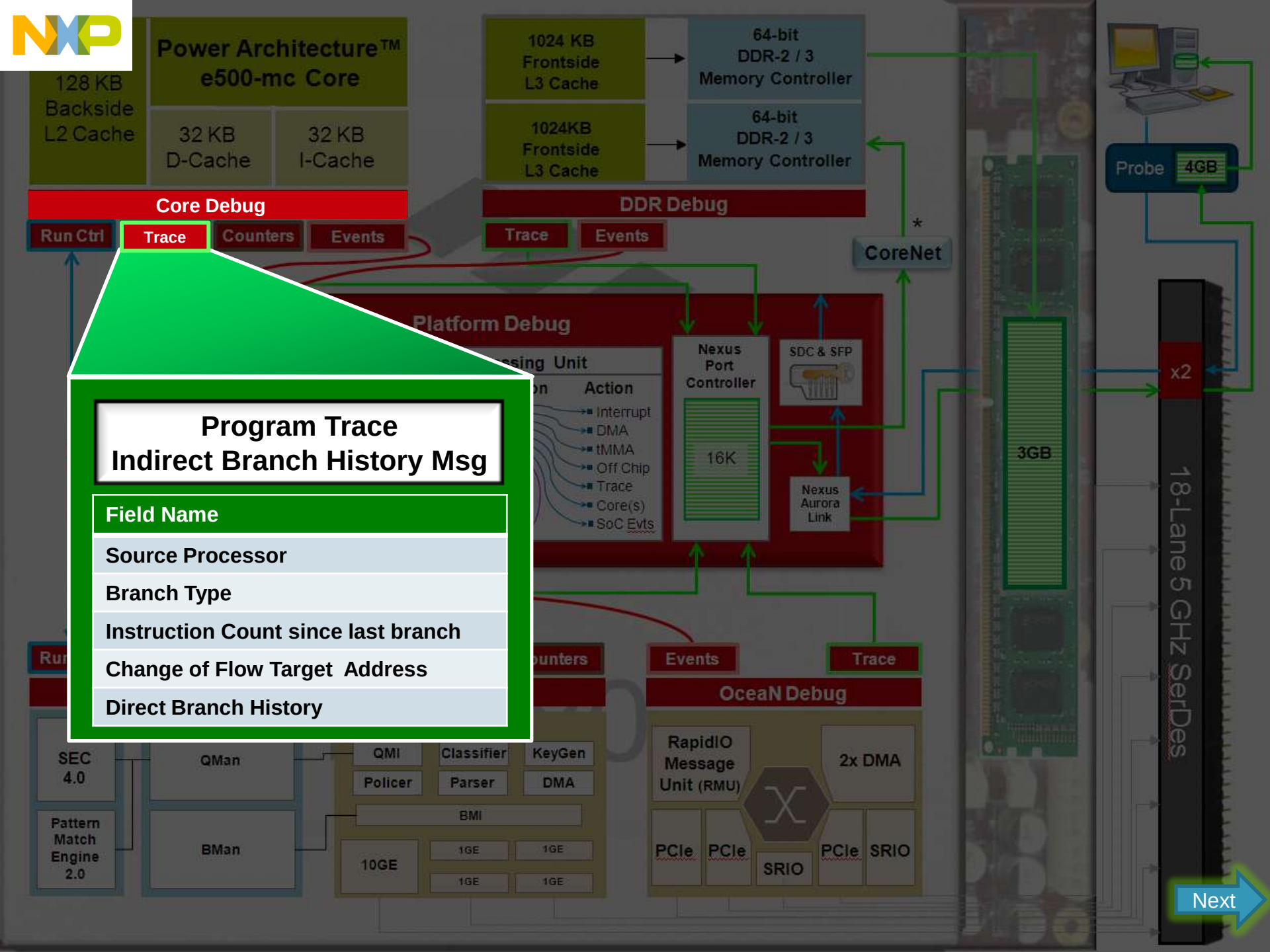

Power Architecture™ e500-mc Core
128 KB Backside L2 Cache
32 KB D-Cache
32 KB I-Cache
1024 KB Frontside L3 Cache
1024KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
64-bit DDR-2 / 3 Memory Controller
Core Debug
Run Ctrl
Trace
Counters
Events
DDR Debug
Trace
Events
CoreNet
Platform Debug
Event Processing Unit
Combination
Action
Interrupt
DMA
tMMA
Off Chip
Nexus Port Controller
16K
SDC & SFP
Nexus Aurora Link
3GB
Probe
4GB
x2
18-Lane 5 GHz SerDes
Run Ctrl
Trace
Trace
OceaN Debug
SEC 4.0
Pattern Match Engine 2.0
BMan
10GE
1GE
1GE
1GE
1GE
2x DMA
PCIe
PCIe
SRIO
PCIe
SRIO
Data Acquisition Message (printf)
Field Name
Source Processor
Debug Event Tag
Data



Ownership Trace Message

Field Name
Source Processor
Task/Process ID (Options)
OS Process ID
Hypervisor Logical Partition ID
Guest State
Privilege Level

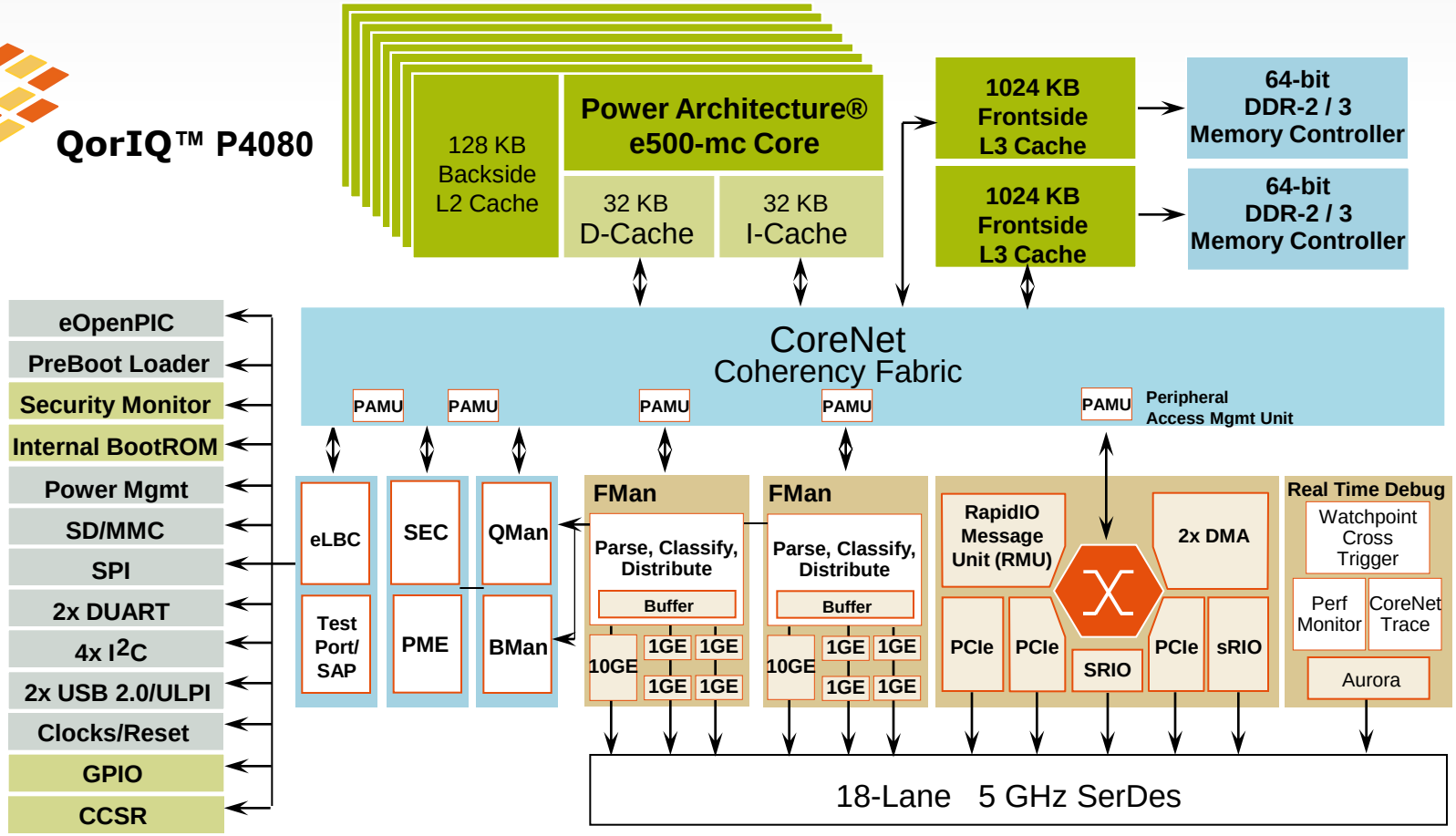

128 KB Backside L2 Cache
Power Architecture™ e500-mc Core
32 KB D-Cache
32 KB I-Cache
1024 KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
1024KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
Core Debug
Run Ctrl
Trace
Counters
Events
DDR Debug
Trace
Events
CoreNet
Probe
4GB
Nexus Port
SDC & SFP
Nexus Aurora Link
3GB
x2
18-Lane 5 GHz SerDes
Program Trace Sync Message
Field Name
Source Processor
Address Space Identifier
Program Counter (full address)
Timestamp
Run Ctrl
Trace
Trace
SEC 4.0
Pattern Match Engine 2.0
QMan
BMan
Frame Manager x 2
QMI
Classifier
KeyGen
Policer
Parser
DMA
BMI
10GE
1GE
1GE
1GE
1GE
RapidIO Message Unit (RMU)
2x DMA
PCIe
PCIe
SRIO
PCIe
SRIO


128 KB Backside L2 Cache
Power Architecture™ e500-mc Core
32 KB D-Cache
32 KB I-Cache
1024 KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
1024KB Frontside L3 Cache
64-bit DDR-2 / 3 Memory Controller
Core Debug
Run Ctrl
Trace
Counters
Events
DDR Debug
Trace
Events
CoreNet
Probe
4GB
Platform Debug
Action
Interrupt
DMA
tMMA
Off Chip
Trace
Core(s)
SoC Evts
Nexus Port Controller
16K
SDC & SFP
Nexus Aurora Link
3GB
x2
18-Lane 5 GHz SerDes
Program Trace
Indirect Branch History Msg
Field Name
Source Processor
Branch Type
Instruction Count since last branch
Change of Flow Target Address
Direct Branch History
Events
Trace
OceaN Debug
SEC 4.0
QMan
Pattern Match Engine 2.0
BMan
QMI
Classifier
KeyGen
Policer
Parser
DMA
BMI
10GE
1GE
1GE
1GE
1GE
RapidIO Message Unit (RMU)
2x DMA
PCIe
PCIe
SRIO
PCIe
SRIO
Next

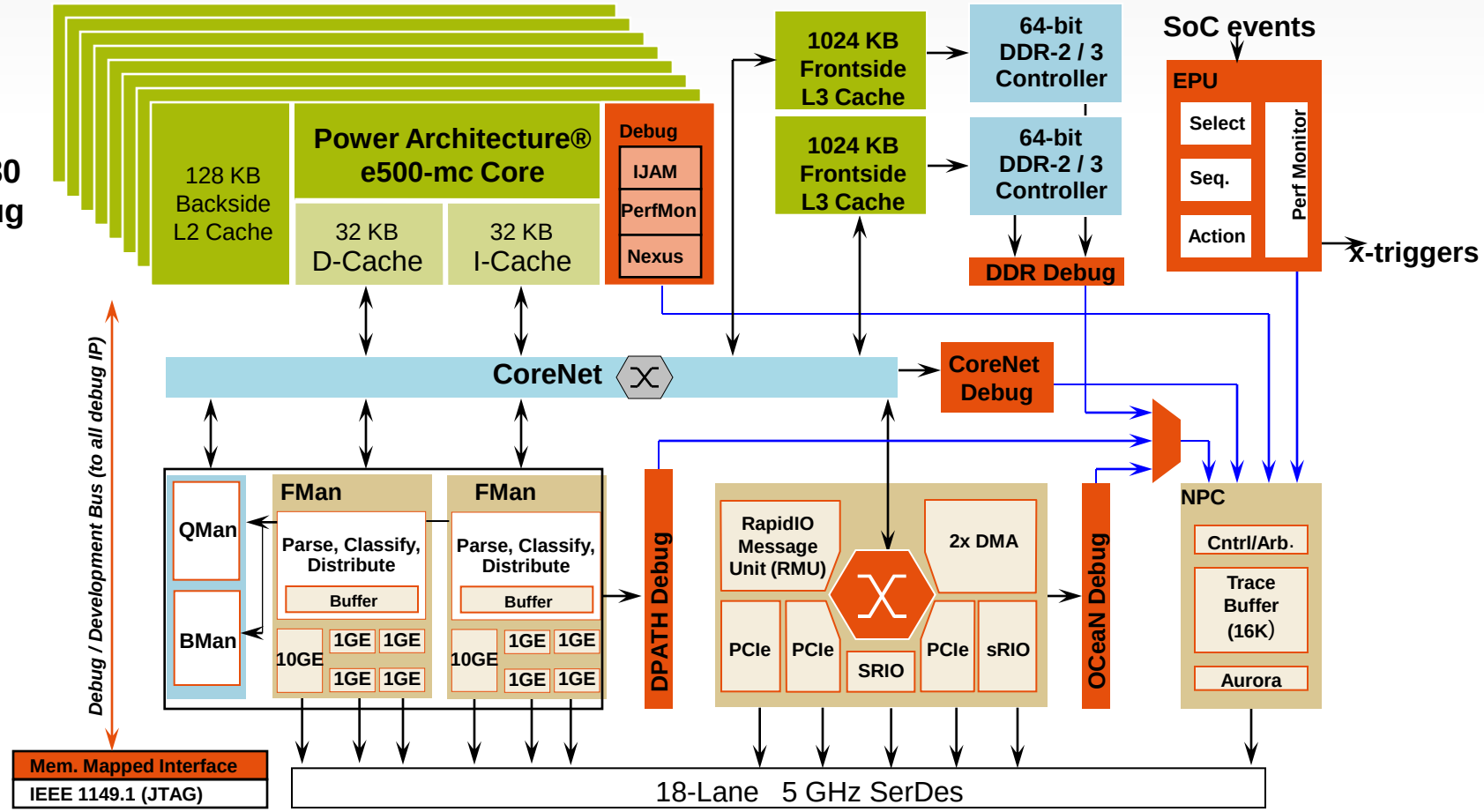
FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Debug IP Details

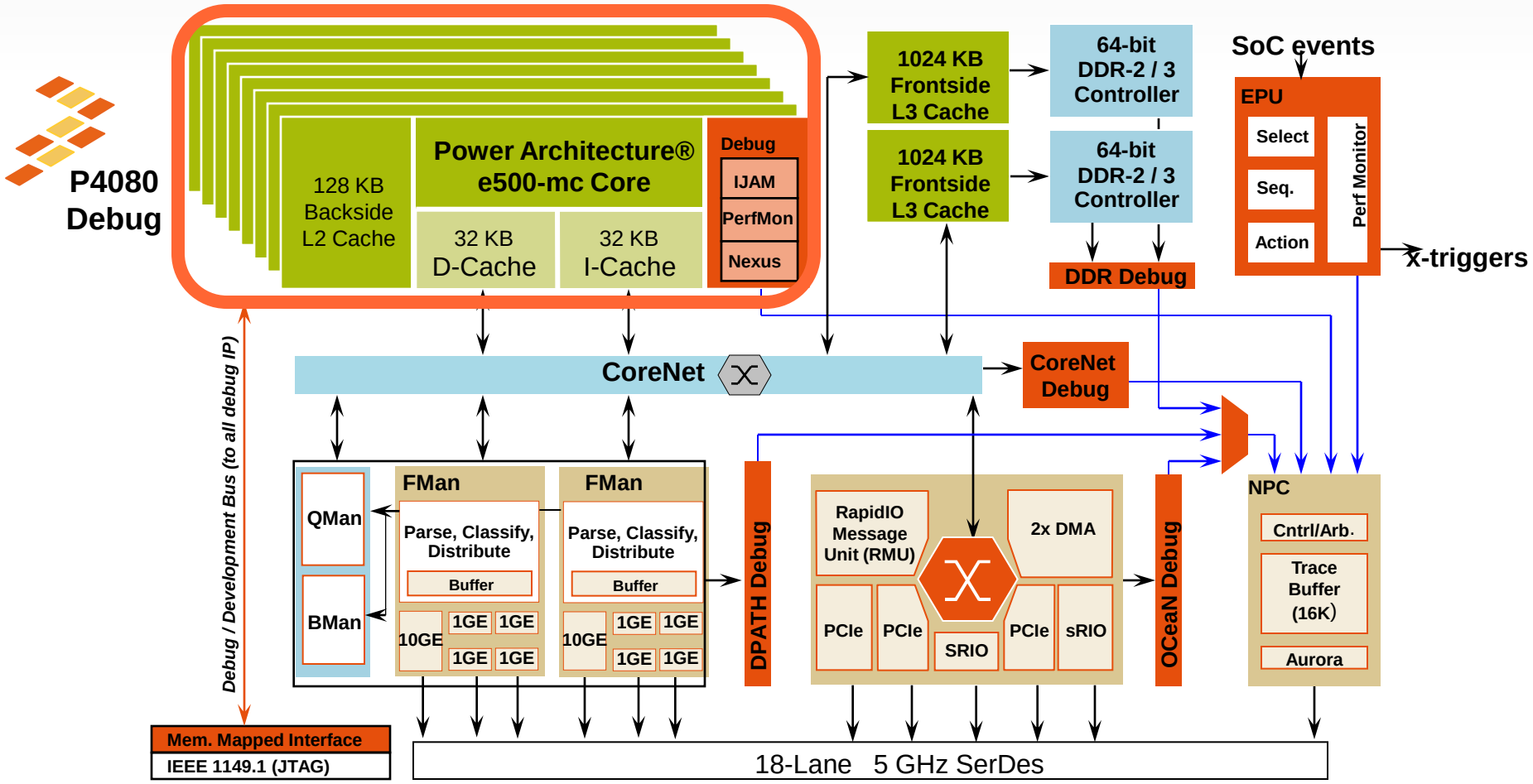
QorIQ™ P4080



P4080 Debug



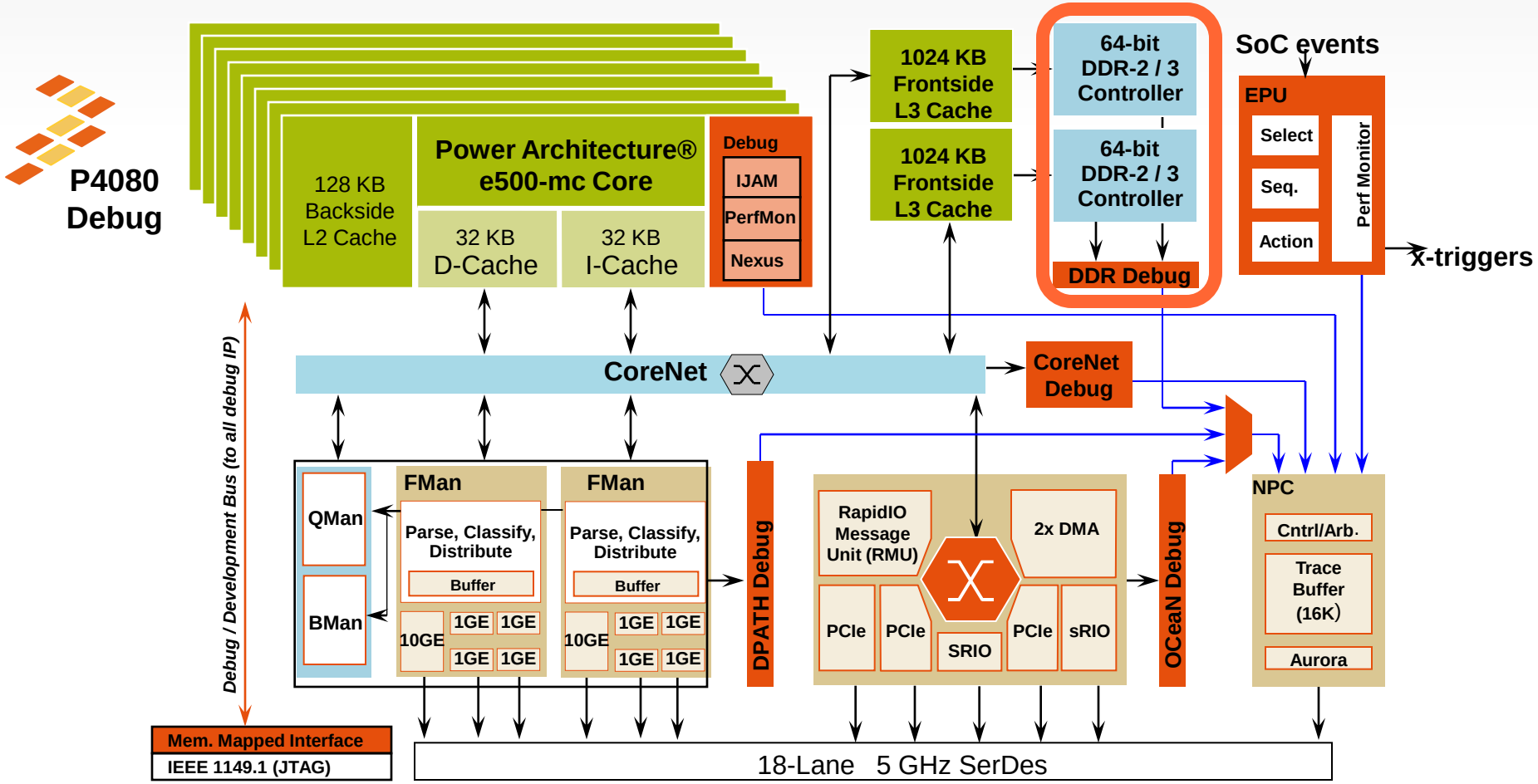
e500mc Debug



e500mc Debug

- **Static Debug Features (“Classic” Debug):**
  - Breakpoints and watchpoints
  - Execute instructions from a halted state (instruction jamming)
  - Software access to debug resources through memory mapped bus architecture
  - Hardware access to memory mapped resources through JTAG (or high-speed serial)
- **Real-Time Debug Features:**
  - Program (instruction) and Data Trace, Ownership trace, Data acquisition.
  - Watchpoint (event) messages and triggering
  - Status messages
  - Trace queue management policy
  - Timestamp capability
  - All features (static/real-time) are programmable by external development tools (memory mapped) or by embedded software
- **Performance Monitor**
  - Each core provides 4x 32-bit performance monitor counters (PMCs)
  - PMC overflows can generate events to SoC event processing module (x-triggering)
  - PMC count values can be captured into shadow registers in response to a trigger. The shadow registers can be accessed non-intrusively

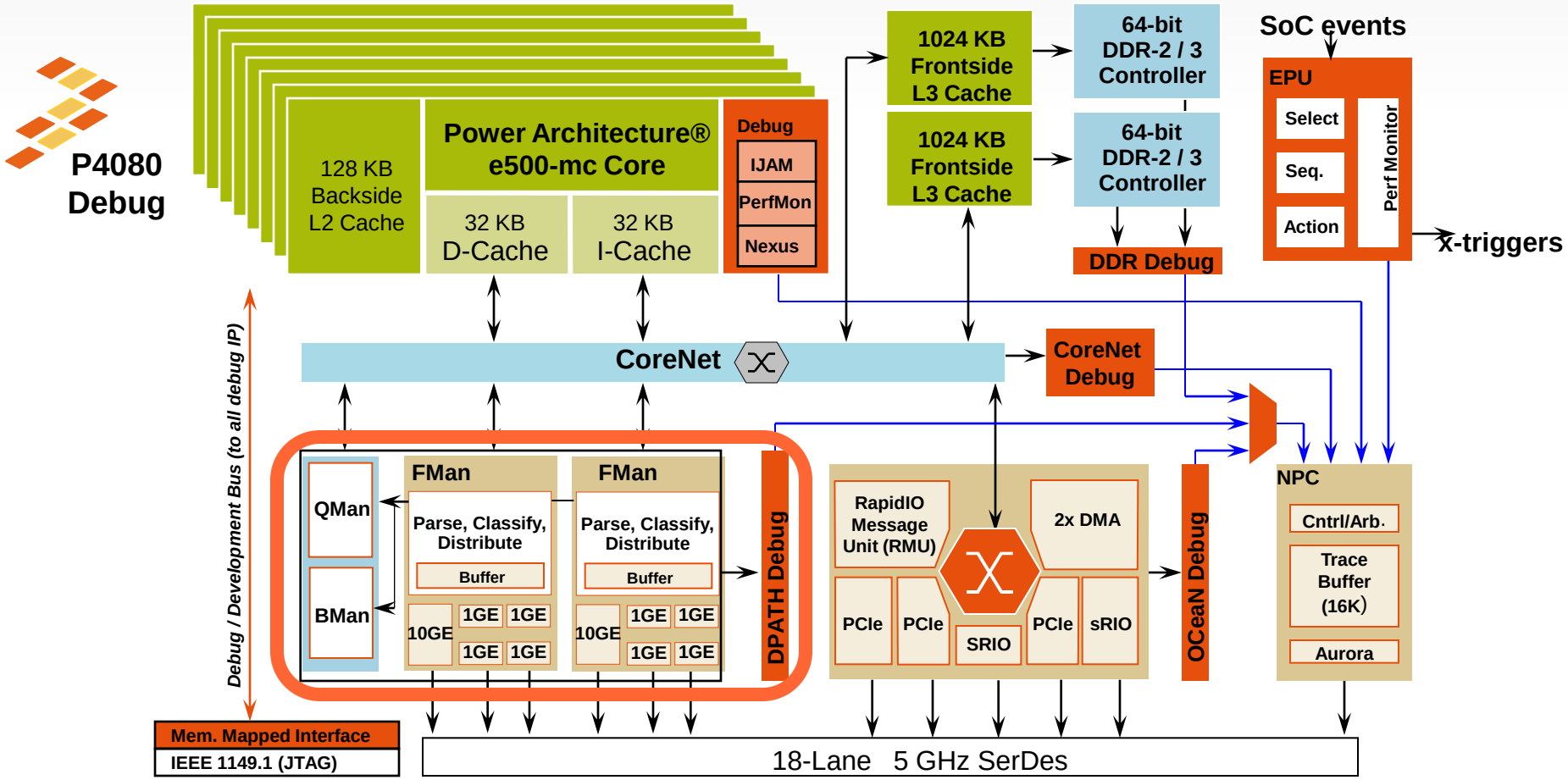
DDR Debug



DDR Debug

- **Provides visibility to DDR transactions by monitoring DDR memory controllers (2)**
  - Source ID
  - Transaction type, size, and address
  - Other transaction attributes

DataPath Debug



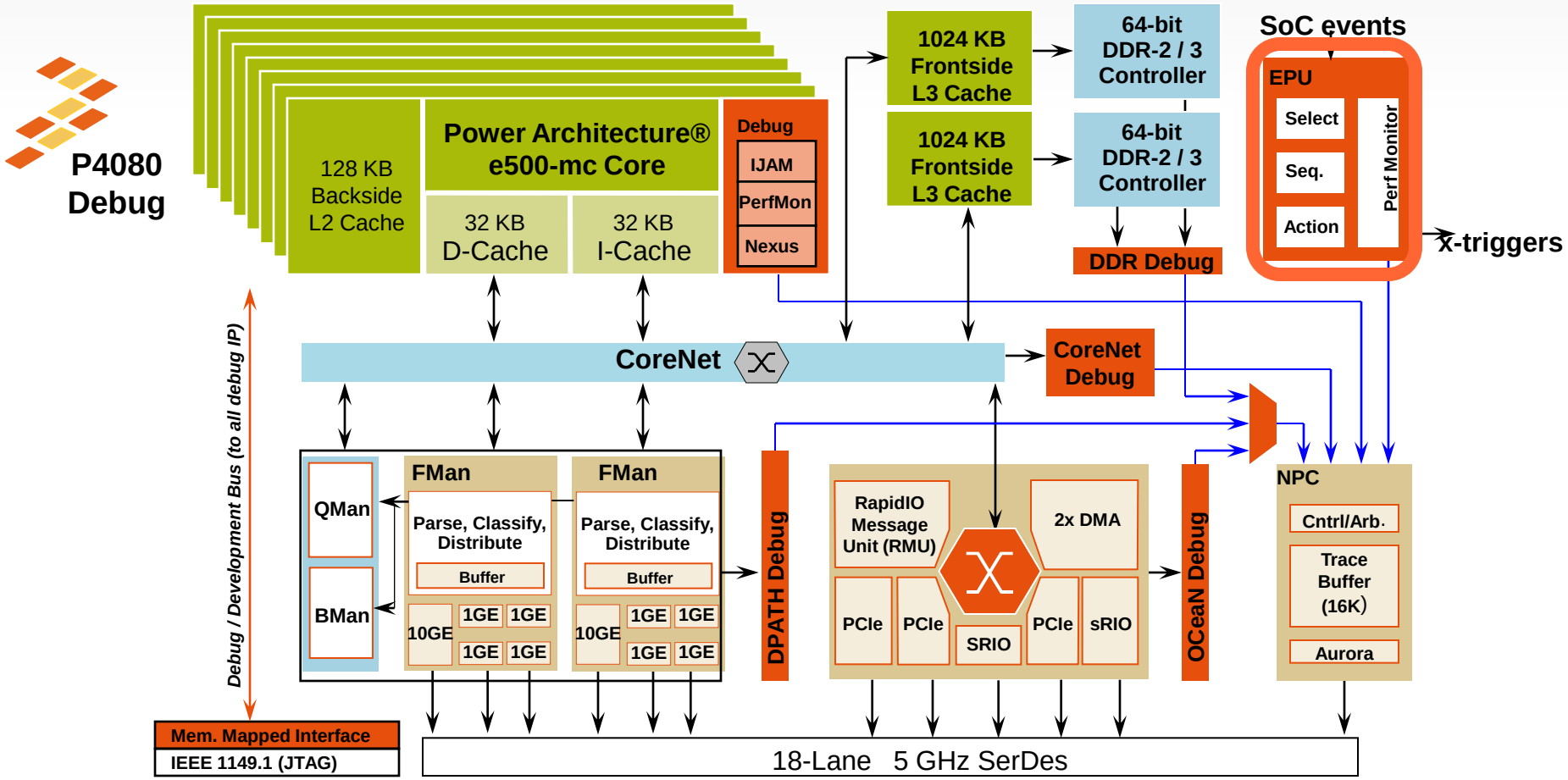
DataPath Debug

- **Provides visibility to queuing operations in architected Queue Manager (QM)**
  - Frame queue ID, queue type (enqueue/dequeue)
  - Channel number
  - Portal number, portal type

- **Provides visibility to debug context info within each Frame Manager (FM)**
  - Trace data from each processing stage (BMI, QMI, KeyGen …)
  - Sophisticated frame comparators for each sub-block
  - Fine granularity frame tracing control

- **Generates performance events from all DataPath blocks (FM, QM, BM, PME)**
- **Supports event controlled halting**
- **Generates watchpoint (event) trace**

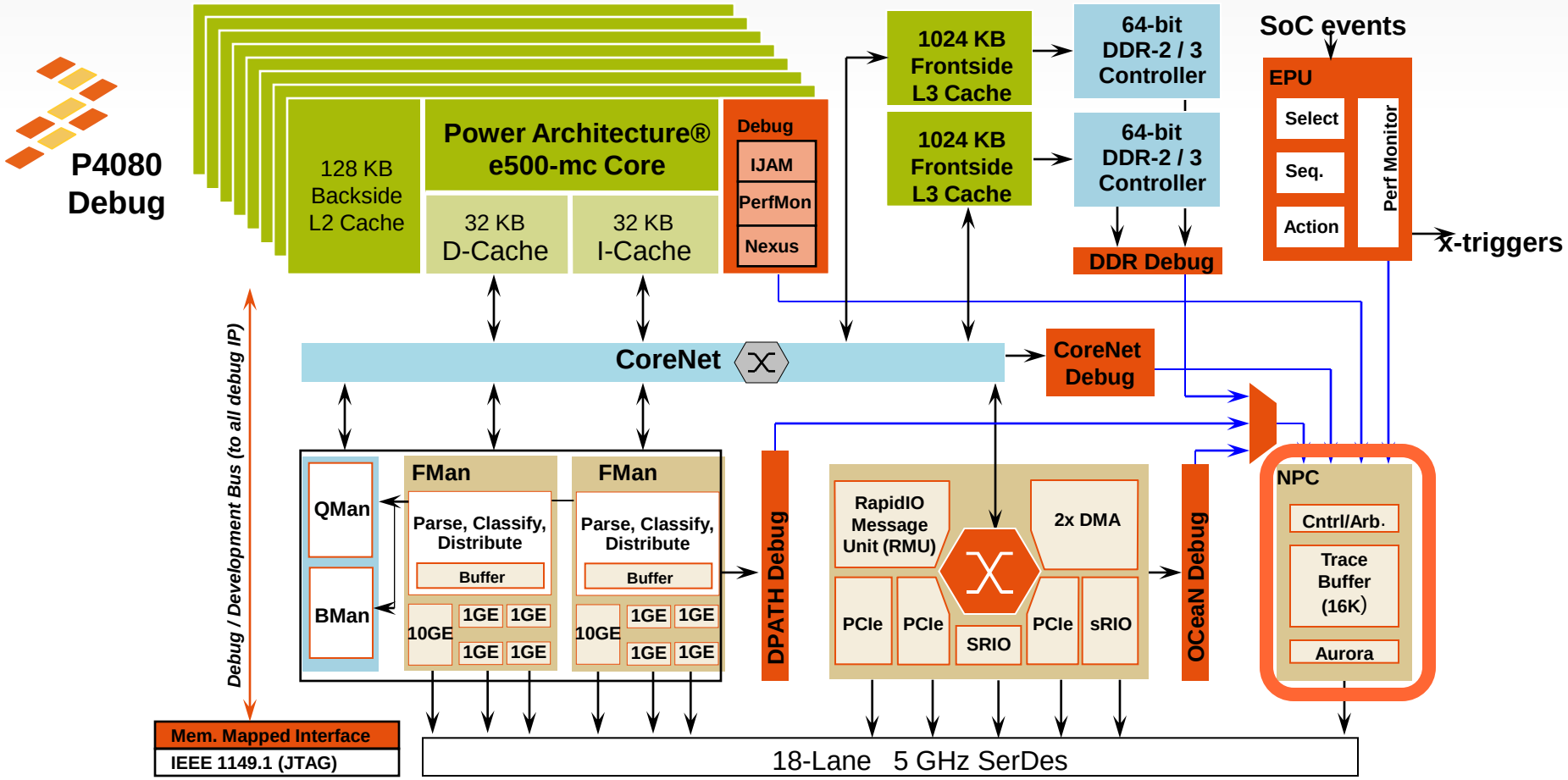
Event Processing Unit (EPU)



EPU Block Diagram

- The Event Processing Unit (EPU) receives event notifications from CPUs, SoC blocks or off-chip and generates appropriate watchpoint cross-triggers.
- Programmable front-end mux pares down the many signals/events to be processed
- Sequencer module within the EPU applies logical functions (and delays) to incoming events to detect certain conditions and provide the appropriate cross-triggers
- EPU can trigger by matching event/transaction types, addresses
  - Sources: CPU cores, SoC blocks, perfmon. counters, external triggers
  - Targets: CPU cores, SoC blocks, Nexus trace output
- Example Usage:
  - Allow breakpoint on one core to halt any combination of other cores
  - Correlated breakpoint groups supported (ex. 0-8 cores per group)

Nexus Port Controller (NPC)



Nexus Port Controller (NPC)

- **Provides top level control of trace messages from all clients**
  - Front-end buffers and arbitration logic to capture multi-client trace data
  - Data received by NPC is lossless (barring loss of high-speed link)

- **Provides internal trace buffer (16 KB for QorIQ P4080)**
  - Mechanism to capture data for probeless environment (or production)
  - Selectable client trace captured into buffer (ex. capture trace from CPU #1, 2, 7)

- **Provides interface to high-speed serial link trace transmission off chip**
  - Xilinx Aurora - link layer communications protocol for point-to-point serial
  - Two (2) bidirectional SerDes lanes dedicated for Nexus
  - NPC formats (pre-translates) data into octet pairs - sent to SerDes

- **Provides support for alternative egress mechanisms (DDR, parallel AUX)**
- **Generates Global watchpoint event used to synchronize timestamps**

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

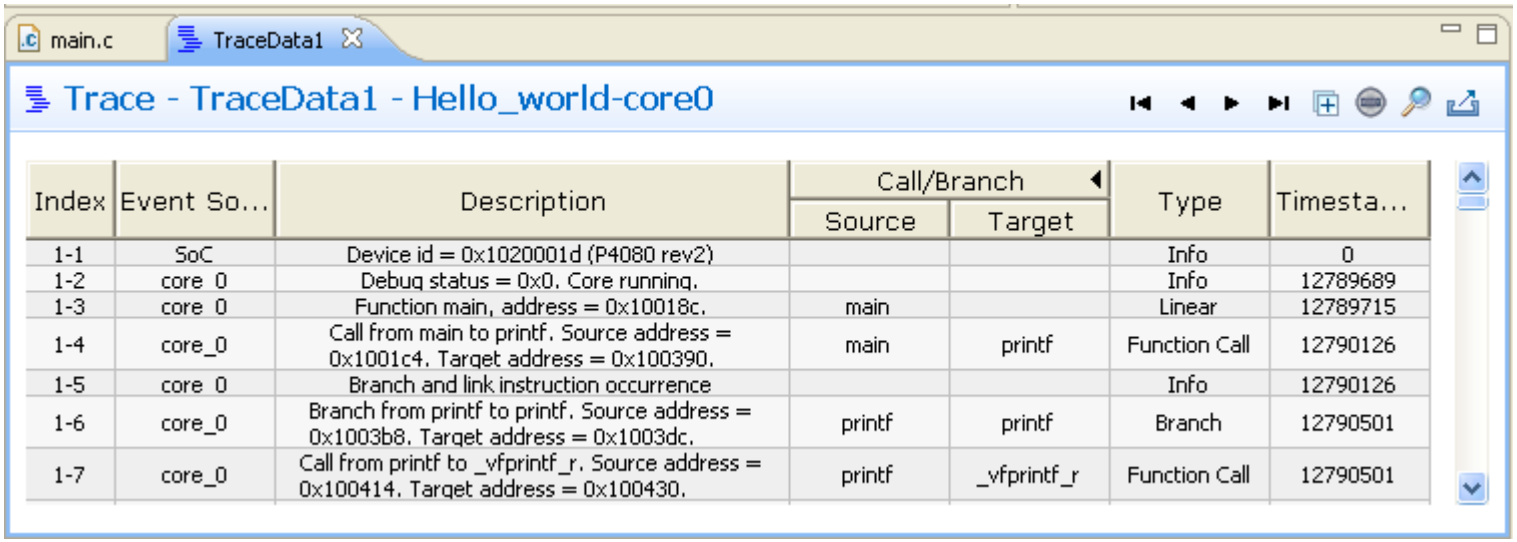
Freescale Tools

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

CodeWarrior PA10 Trace Tools

PA10 Trace Viewer

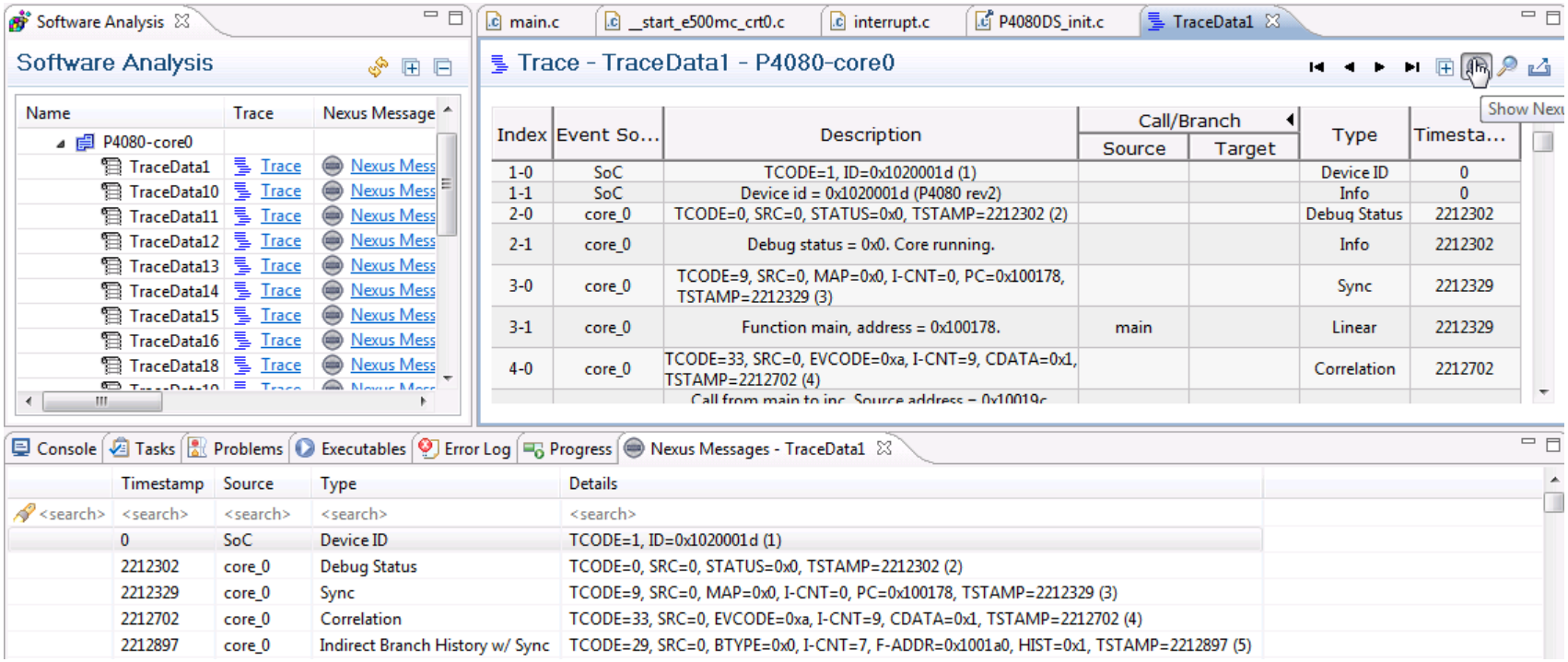
- Displays decoded trace
- Trace can be collected internally (Nexus Buffer or DDR) or external via the Gigabit TAP
- Supports all cores, DDR, Ocean, FM and QM Manager
- Captured data is decoded and presented in trace viewer.
- Core (Program) Trace presented in either source, assembly or mixed views



Index	Event So...	Description	Call/Branch		Type	Timesta...
			Source	Target		
1-1	SoC	Device id = 0x1020001d (P4080 rev2)			Info	0
1-2	core 0	Debug status = 0x0. Core running.			Info	12789689
1-3	core 0	Function main, address = 0x10018c.	main		Linear	12789715
1-4	core_0	Call from main to printf. Source address = 0x1001c4. Target address = 0x100390.	main	printf	Function Call	12790126
1-5	core 0	Branch and link instruction occurrence			Info	12790126
1-6	core_0	Branch from printf to printf. Source address = 0x1003b8. Target address = 0x1003dc.	printf	printf	Branch	12790501
1-7	core_0	Call from printf to _vfprintf_r. Source address = 0x100414. Target address = 0x100430.	printf	_vfprintf_r	Function Call	12790501

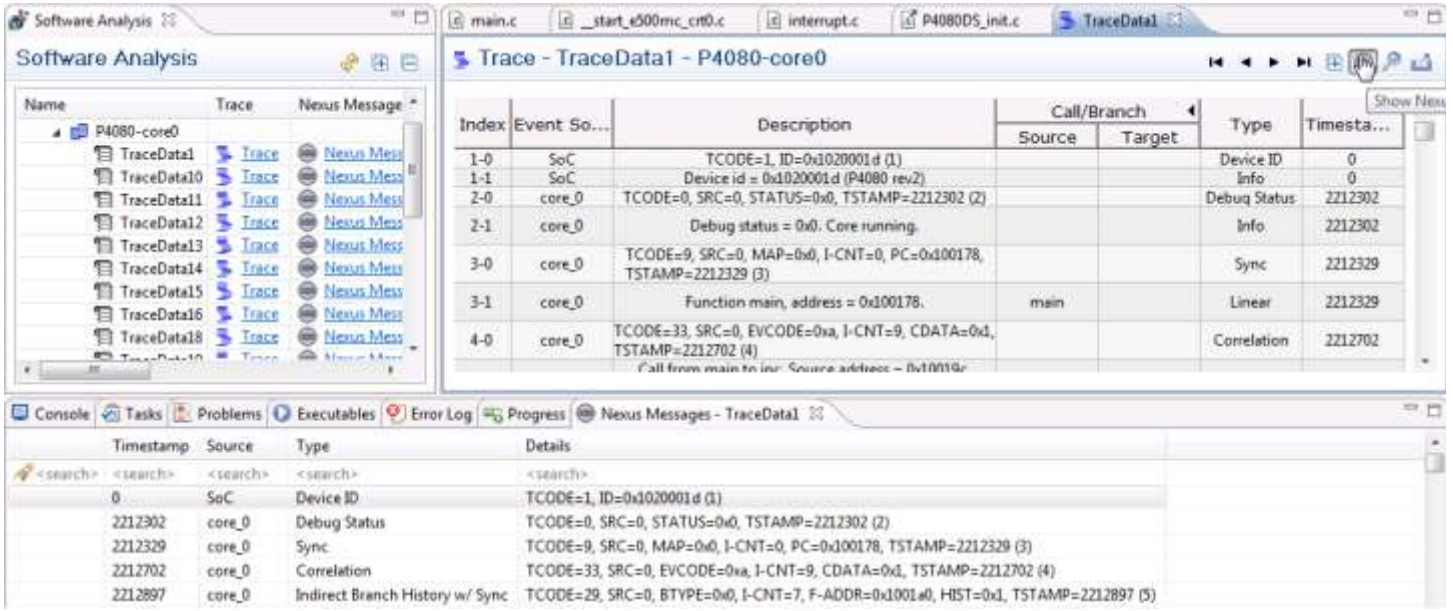
PA10 Trace View: Raw Nexus Messages

- For advanced debug cases, developers need to see the ‘raw’ Nexus messages, no decoding needed.
- CodeWarrior PA 10.1.2 has a view that allows viewing these raw Nexus messages.



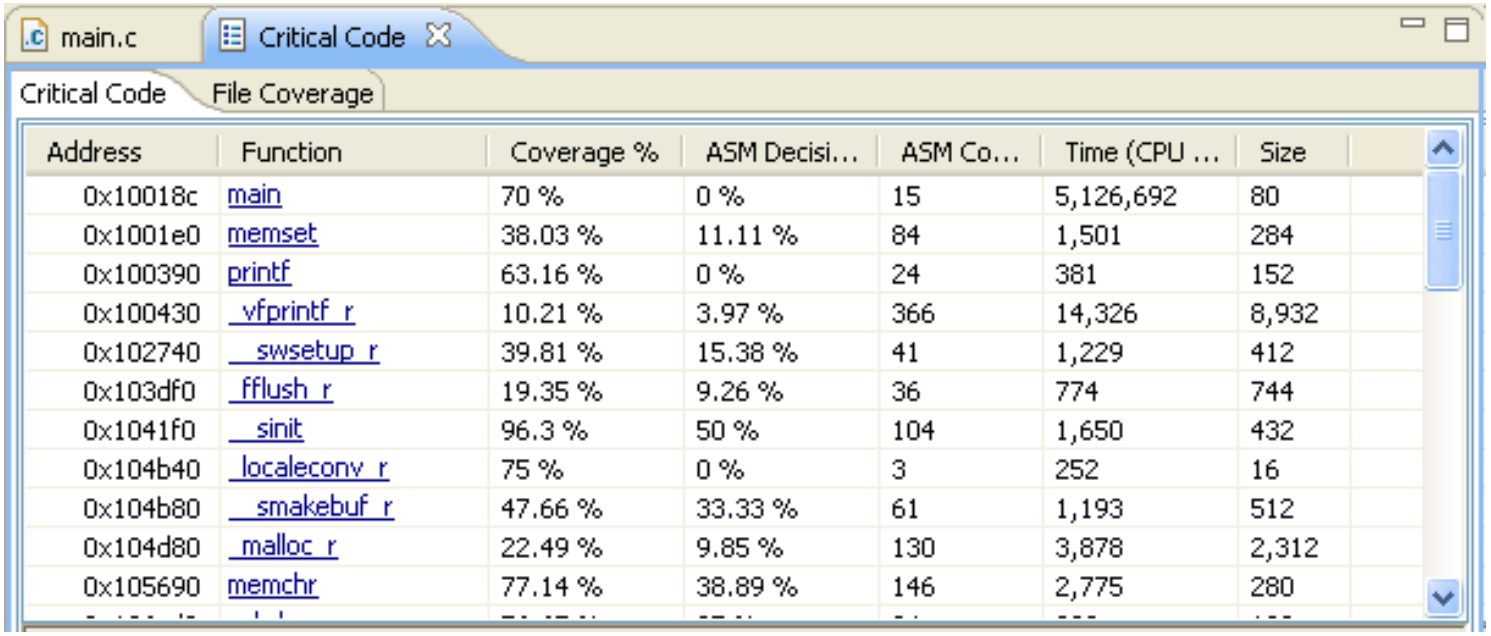
Nexus (Raw) Trace View

- Ability to turn on interleaved Nexus Messages in the existing Trace Editor.
- ‘Nexus Messages’ view, opened from Software Analysis view.



Critical Code View – Bare Metal Only

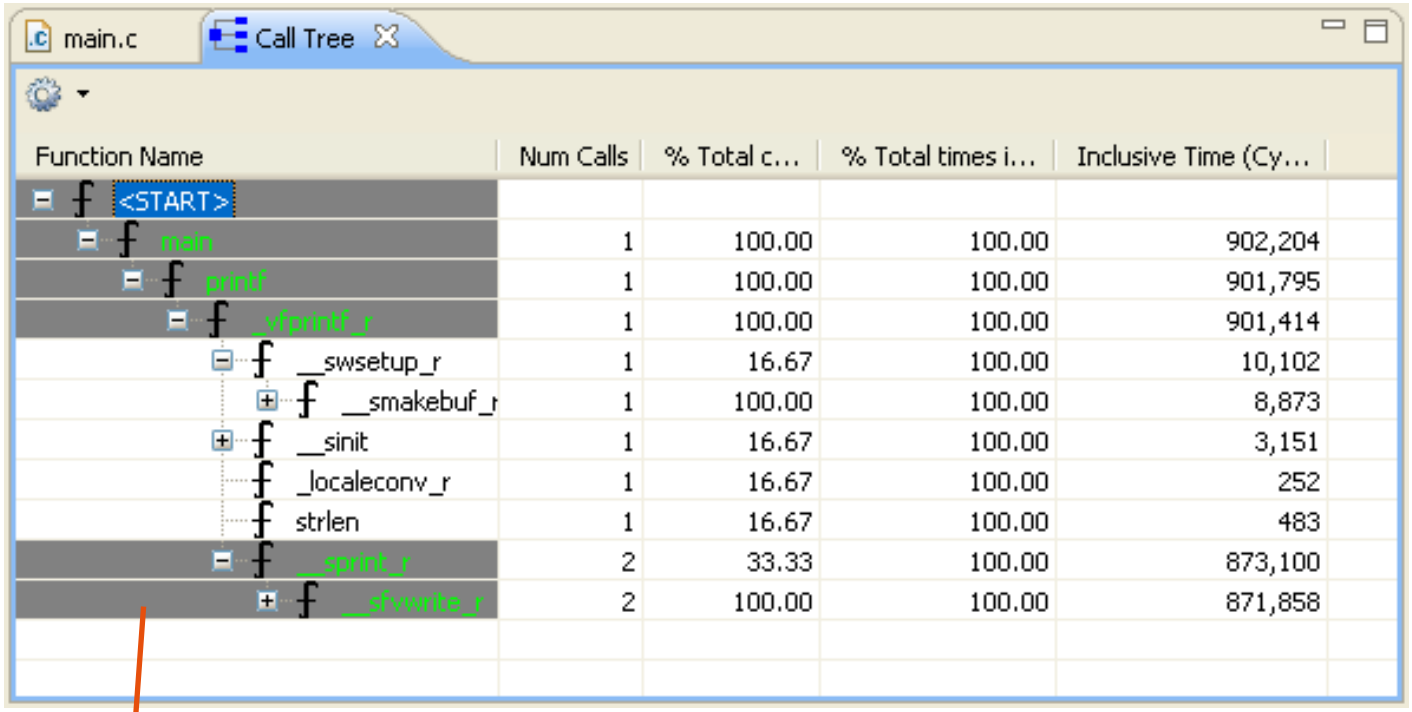
- Flat Profiler (counts each function execution)
- Allows analysis of the flat profile of your application either at a functional or file level



Address	Function	Coverage %	ASM Decisi...	ASM Co...	Time (CPU ...	Size
0x10018c	main	70 %	0 %	15	5,126,692	80
0x1001e0	memset	38.03 %	11.11 %	84	1,501	284
0x100390	printf	63.16 %	0 %	24	381	152
0x100430	_vfprintf_r	10.21 %	3.97 %	366	14,326	8,932
0x102740	__swsetup_r	39.81 %	15.38 %	41	1,229	412
0x103df0	_fflush_r	19.35 %	9.26 %	36	774	744
0x1041f0	__sinit	96.3 %	50 %	104	1,650	432
0x104b40	_localeconv_r	75 %	0 %	3	252	16
0x104b80	__smakebuf_r	47.66 %	33.33 %	61	1,193	512
0x104d80	_malloc_r	22.49 %	9.85 %	130	3,878	2,312
0x105690	memchr	77.14 %	38.89 %	146	2,775	280

Call Tree View – Bare Metal

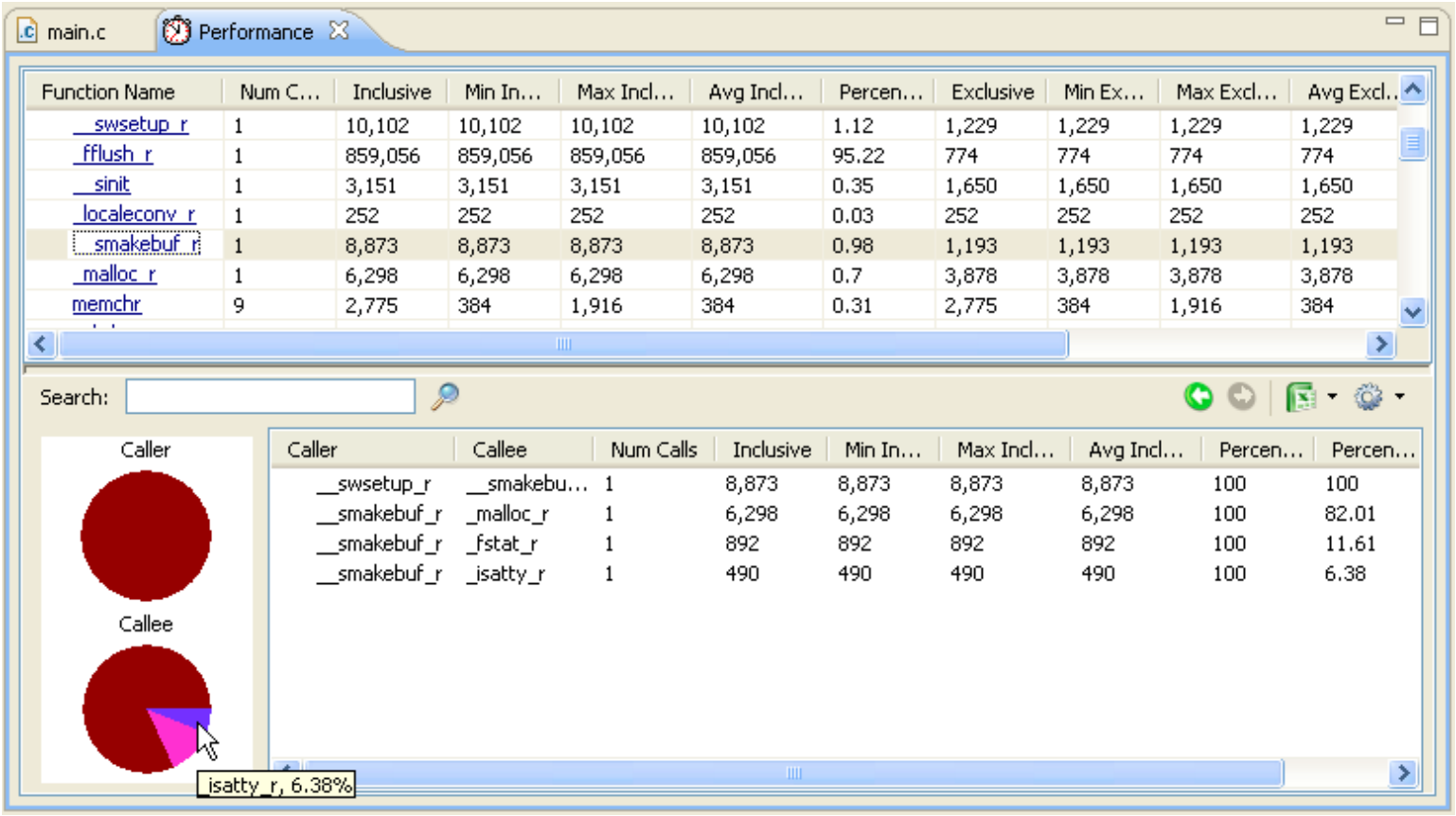
- Displays a tree view of your code using a tree view
- Synchronized to source



Dark gray nodes indicates the functions that consume the most time

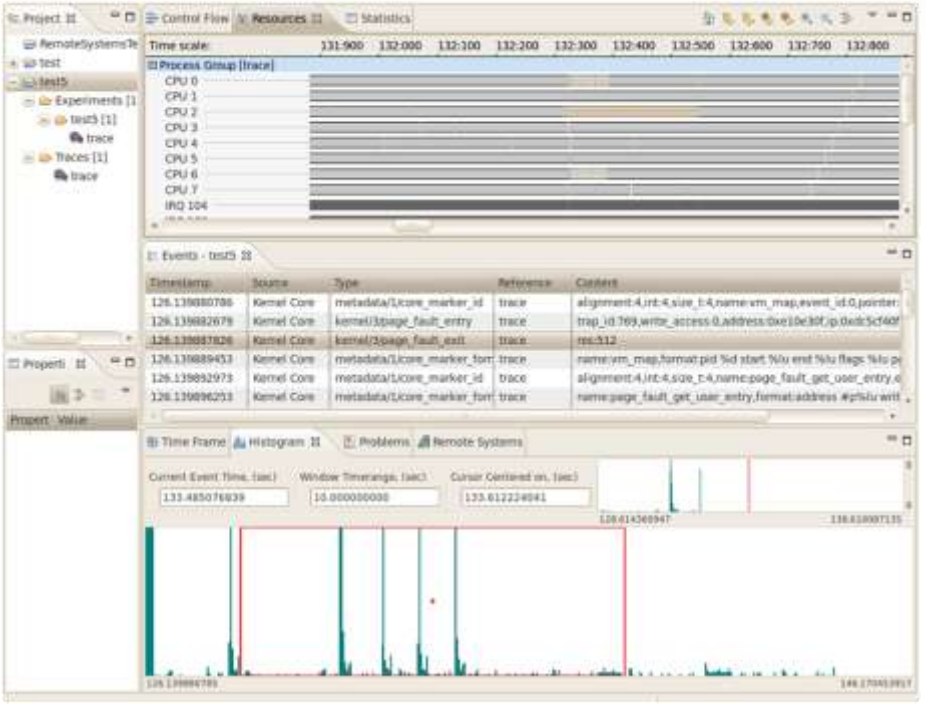
Performance Data View (Bare Metal Only)

- **Hierarchical profiler**
- **Presents**
  - Self time (exclusive time for a function)
  - Hierarchical time - time spent in that function and its children



LTTng Trace: PA10 LTTng Viewer

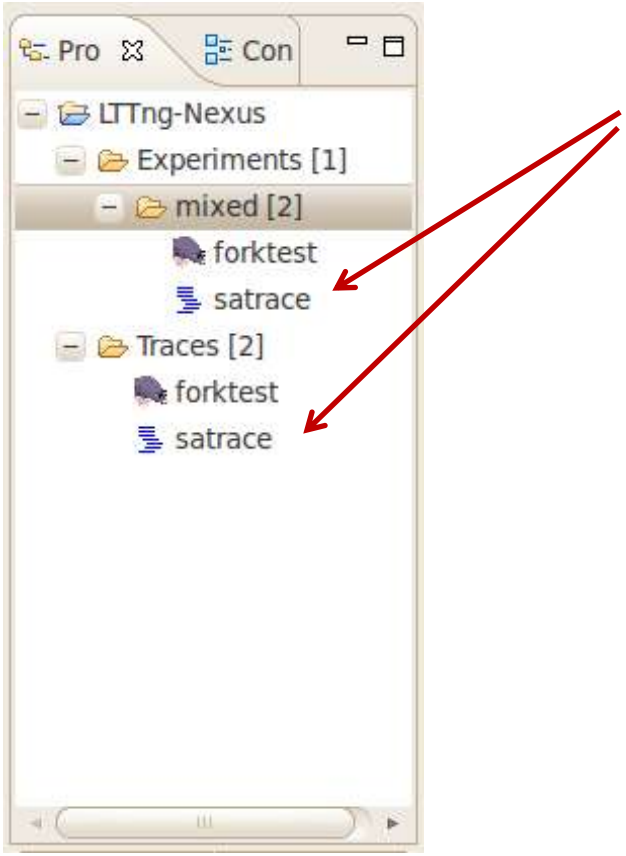
- **Features**
  - **Project View** - Project management
  - **Events View** - Table view raw events
  - **Time Frame View** - Time-based navigation in the traces
  - **Histogram View** - Event distribution analysis
  - **Control Flow View** - Processes state analysis
  - **Resource View** - System resources state analysis
  - **Statistics View** - General traces statistics
  - **Integrated Help**
  - **GUI Based Configuration** – GUI based trace configuration



- Provides visualization for the Linux® Tracing Toolkit, next generation – LTTng

PA10 Linux Trace Support: LTTngX

- Freescale's enhancement to the standard Eclipse Linux LTTng Tools Viewer
- Displays LTTng Trace events together with (synchronized) Nexus events in CodeWarrior's LTTng views.
- With this feature it is possible to capture both LTTng kernel tracing and Nexus program trace, and then view the data, time synchronized, in the LTTng views.



- SA Nexus traces can be imported into LTTng projects and added to experiments

LTTngX

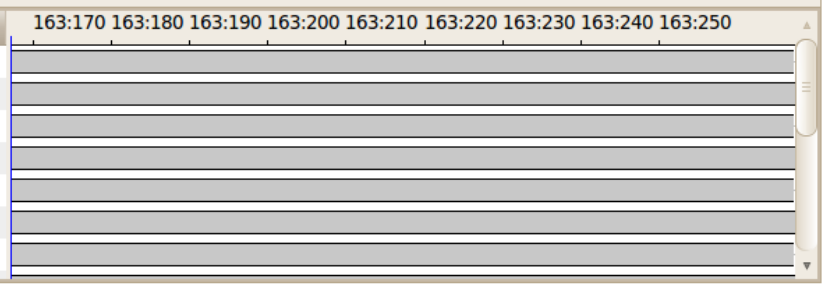
Events - mixed

	Timestamp	Source	Type	Reference	Content
N	163.167050600	e500mc_core_0	Info	TraceData2 (test2-core0)	Debug status - 0x0. Core running.
	163.167050790	Kernel Core	metadata/0/core_marker_id	forktest	alignment:0,size_t:4,int:4,name:page_fault_get_user_entry,pointer:4
N	163.167052896	e500mc_core_0	Info	TraceData2 (test2-core0)	Branch and link instruction occurrence
N	163.167055892	e500mc_core_0	Info	TraceData2 (test2-core0)	Branch and link instruction occurrence
	163.167058043	Kernel Core	metadata/0/core_marker_forr	forktest	name:page_fault_get_user_entry,format:address #p%lu write_acce
	163.167060710	Kernel Core	metadata/0/core_marker_id	forktest	alignment:0,size_t:4,int:4,name:swap_out,pointer:4,event_id:3,long:
	163.167063163	Kernel Core	metadata/0/core_marker_forr	forktest	name:swap_out,format:pfn %lu filp %p offset %lu,channel:mm
N	163.167064721	e500mc_core_0	Branch	TraceData2 (test2-core0)	Branch from ? to printf. Source address = ?. Target address = 0x103
	163.167065510	Kernel Core	metadata/0/core_marker_id	forktest	alignment:0,size_t:4,int:4,name:bio_complete,pointer:4,event_id:12,
	163.167068417	Kernel Core	metadata/0/core_marker_forr	forktest	name:bio_complete,format:sector %llu size %u rw(FAILFAST_DRIVE

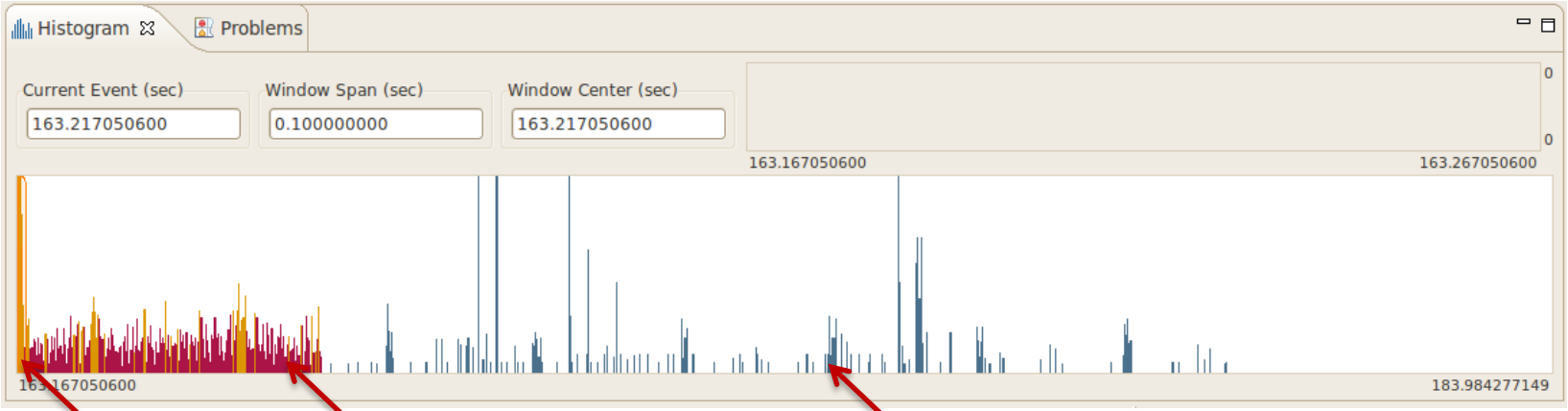
- The events and control views displays color-coded Nexus events intermixed with the LTTng events (time ordered)

Control Flow | Resources | Statistics

Process	Brand	PID	TGID	PPID	CPU	Birth sec	Birth nsec	TRACE
		0	0	0	0	0	000000000	forktest
		0	0	0	0	0	000000000	satrace
		0	0	0	1	0	000000000	forktest
		0	0	0	1	0	000000000	satrace
		0	0	0	2	0	000000000	forktest
		0	0	0	2	0	000000000	satrace
		0	0	0	3	0	000000000	forktest



LTTngX



Orange: mixed LTTng and Nexus events

Red: Nexus events

Blue: LTTng events

- The histogram view displays color-coded events

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

CodeWarrior Performance Analysis

PA10 Performance Analysis Tools

Goals:

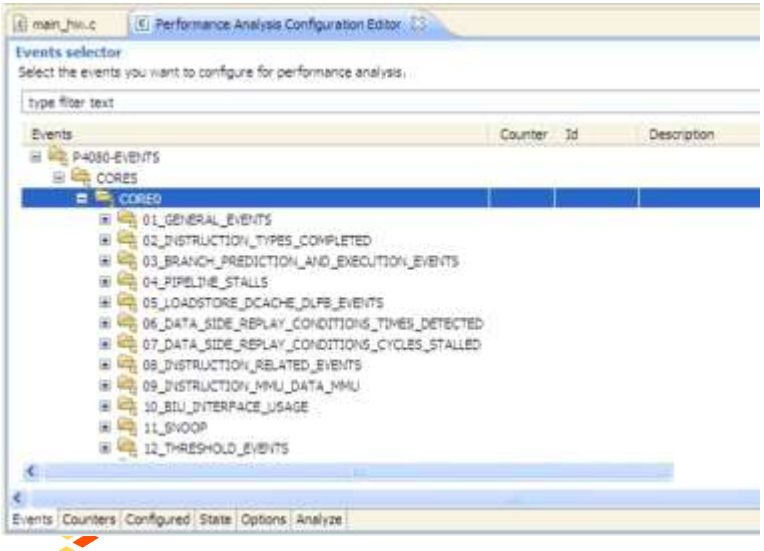
- Investigate performance problems related to Core and SoC Components
- Improved Application Analysis capability – beyond standard Trace and Profile.
- Leverages the QorIQ Debug IP (P4080, P3041, P5020, P5021 and P5040)

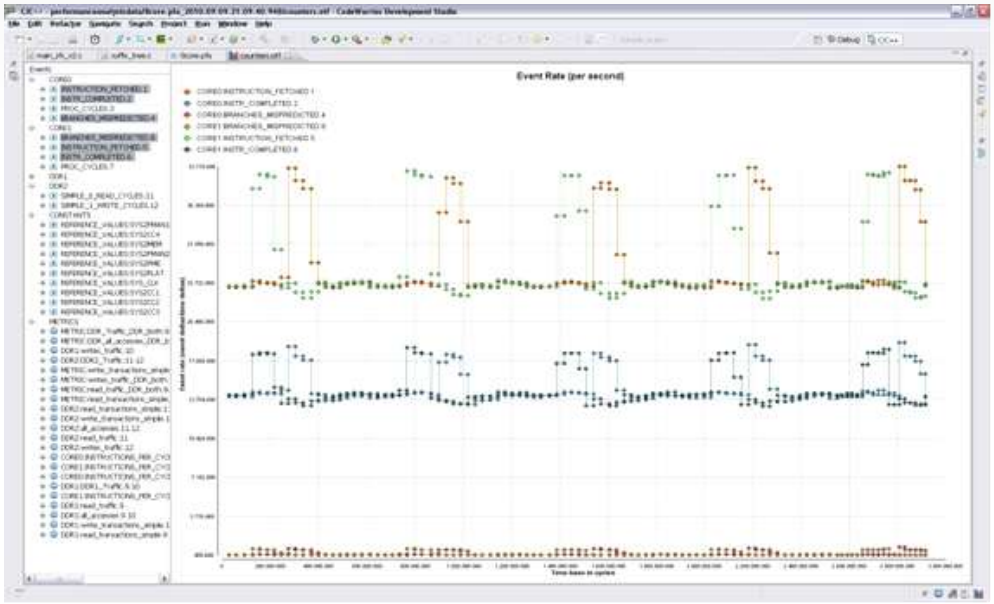
First Available in PA 10.0

- Support for P4080, P3041,P5020, P5021 and P5040 devices

Scenario Tool available

- Stand alone, simplified version that allows user to run common “canned” performance configurations.



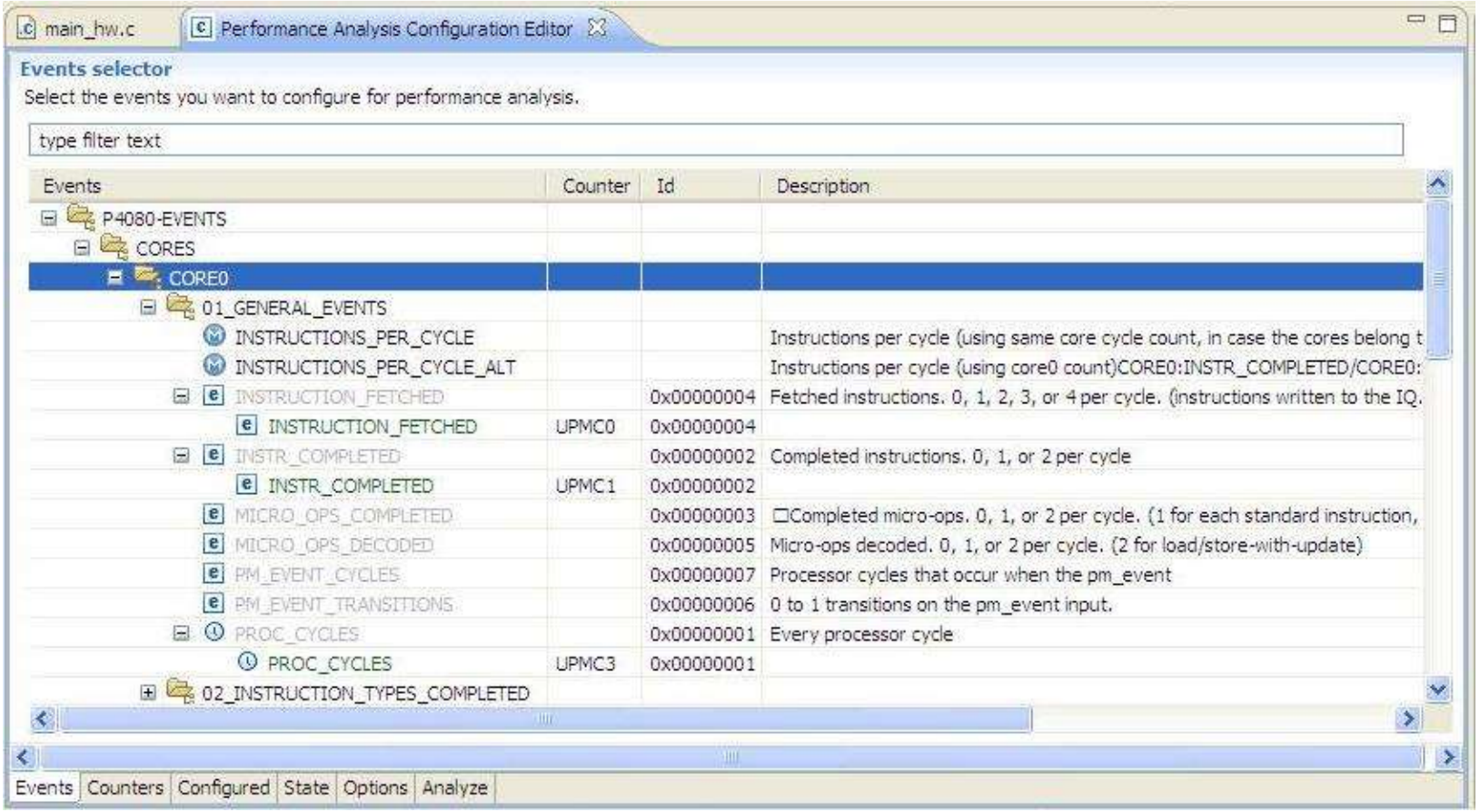


Capabilities

- Captures real-time data from Performance Analysis registers while target code is running
- Captured data are then displayed in both graphical and tabular formats
- Ability to export data for use in a spreadsheet.
- Any and all PerfMon registers are available for data capture
- Data captured via USB TAP, Ethernet TAP or Gigabit TAP
- Graphical Data display

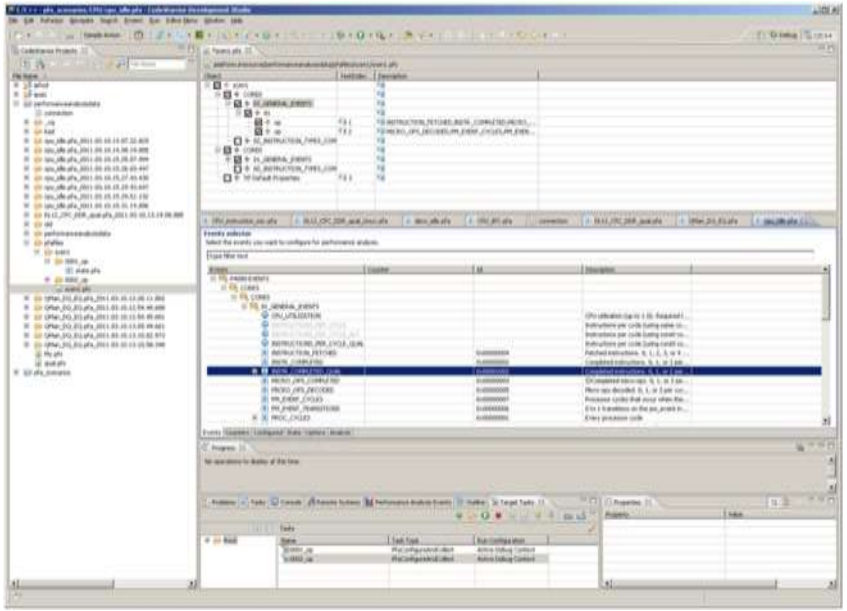
freescale™

Performance Monitor Configuration

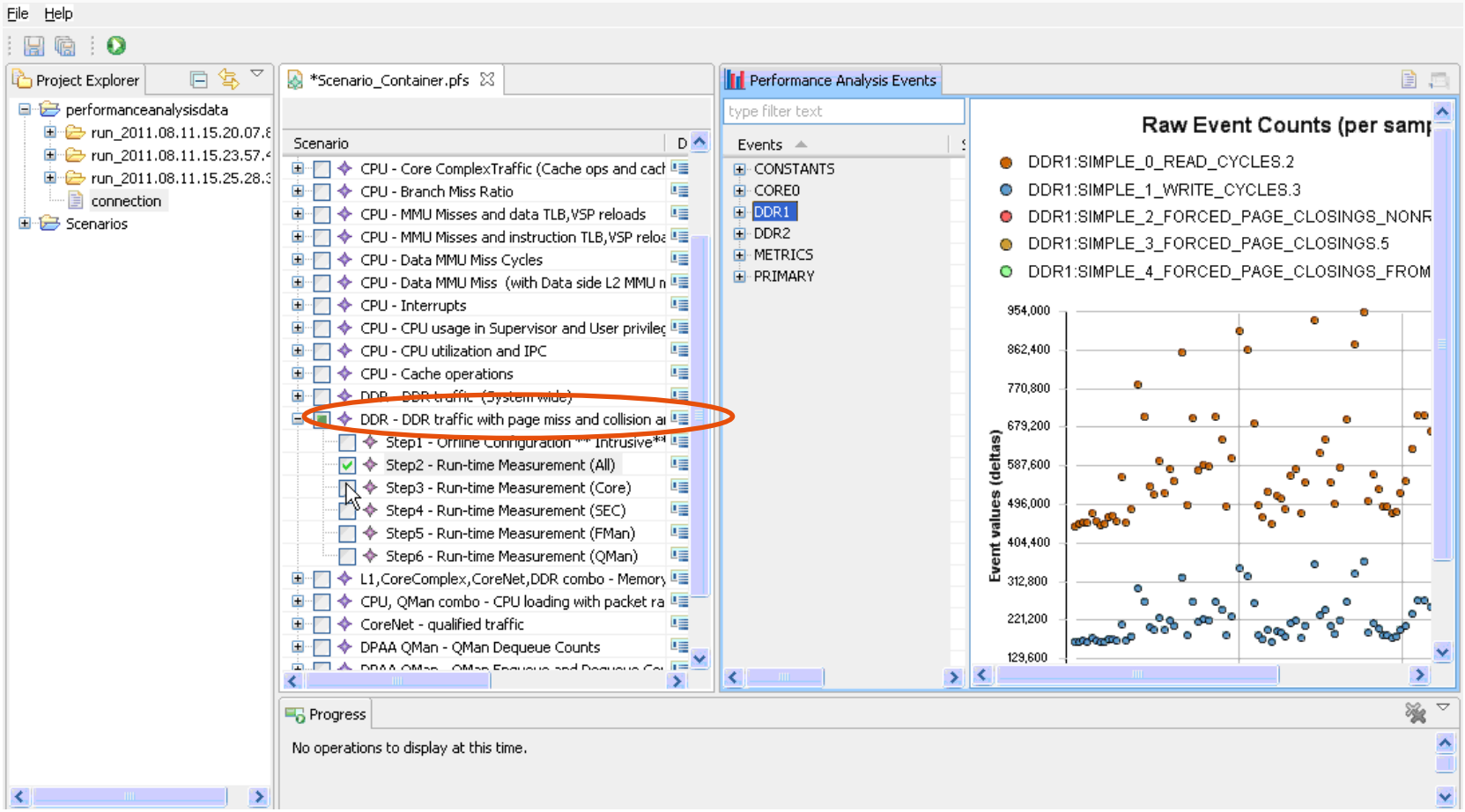


Scenarios Tool

- **What**
  - Stand alone version of PA10 performance scenarios.
  - Tool allows for quick selection of performance scenario), application of scenario and results display
- **Approach:**
  - Delivered as standalone component (no CodeWarrior needed)
  - Runs in both Windows and Linux Hosts
  - Can be easily updated in the field with additional scenarios.
- **Schedule**
  - Version 1.1 Available today – download from http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PE_QORIQ_OPTI_SUITE&fsrch=1&sr=4



Performance Analysis Scenarios



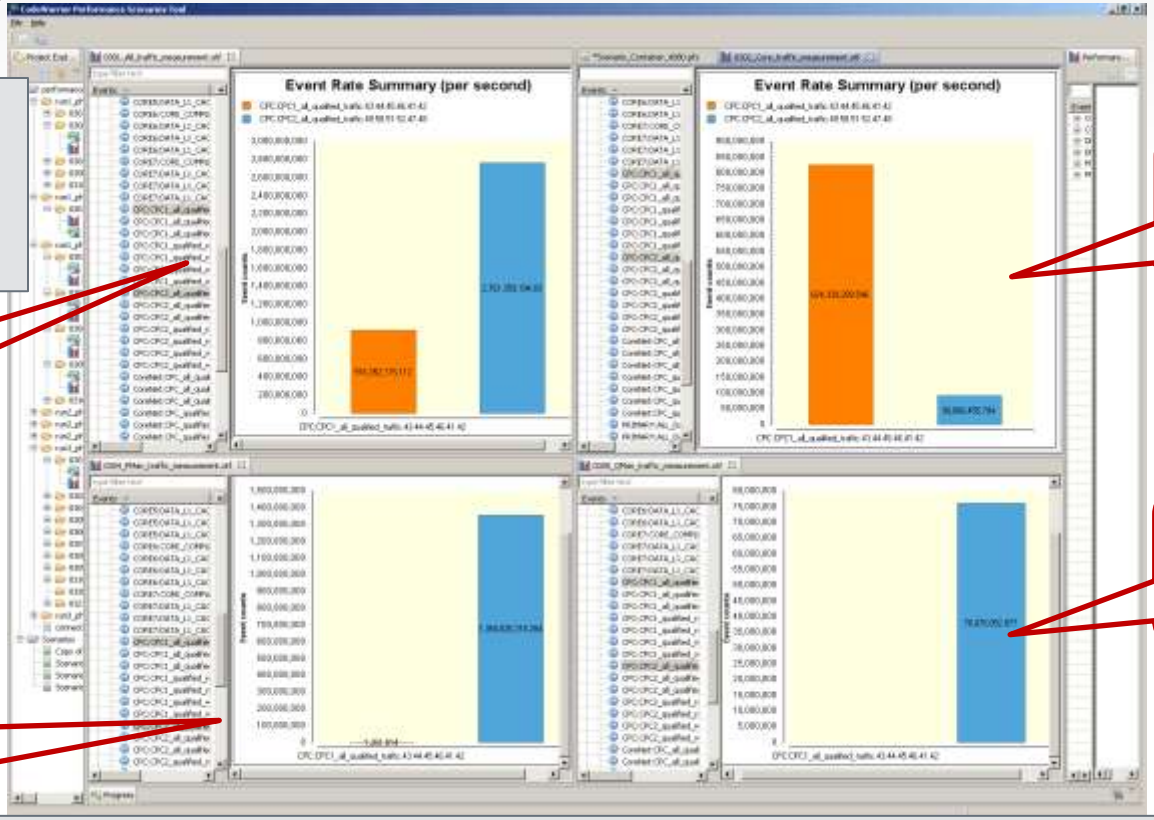
Identifying Cache Performance Imbalance

Selected scenario captures overall traffic as well as individual subsystem traffic including traffic from cores, Fman and Qman

Total traffic to CPC1 and CPC2

Unbalance between platform caches

Fman CPC2 Access



CPU Accesses

Qman CPC2 Access

- Imbalance can be seen by analyzing these subsystems:
  - Majority of CPC1 accesses are made by the CPU (top right)
  - Majority of CPC2 access are from Fman and Qman
  - Remainder of the traffic is due to PCI
- Analysis suggests that DDR interleaving is not configured properly.
- Fixing the configuration provides a performance boost.

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Additional Tools

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

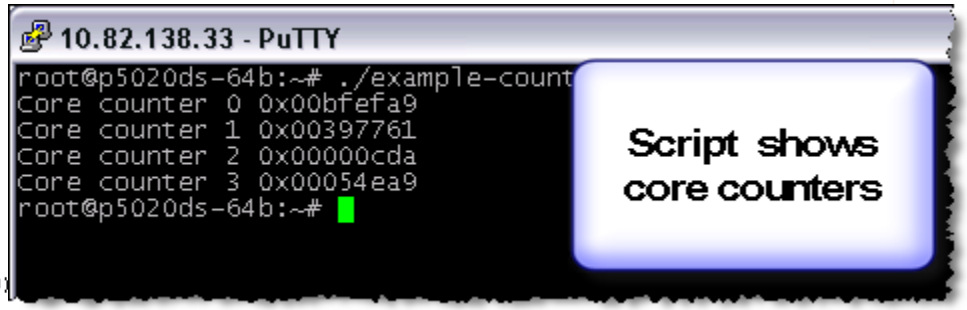
qoriq-dbg

Qoriq-dbg Tools

- **What:**
  - qoriq-dbg – a Linux kernel module mapping QorIQ DCSR memory-mapped registers into the debugfs pseudo-file system.
  - Also maps the cores' performance monitor registers (PMR based) allowing access to both core and SoC debug facilities.
  - Bridges between user space and kernel space especially for core performance register access
  - Easily accessed from user-space scripts or applications via standard file system operations.
- **Approach**
  - Present tools in a manner that is familiar to Linux developers
  - Ease of access from user scripts or applications via standard file system operations.
- **Schedule**
  - Available now as part of Freescale Linux SDK 1.1

Accessing Counters Using qoriq-dbg

```
1  #!/bin/sh
2
3  # define the base directory
4  QORIQ_DBG=/mnt/debugfs/qoriq-dbg
5
6  # Example to read some core counters.
7
8  # set counter 0 to count clocks (1)
9  echo 0x00010000 > ${QORIQ_DBG}/cpu0/pmlca0
10
11 # set counter 1 to instr complete (2)
12 echo 0x00020000 > ${QORIQ_DBG}/cpu0/pmlca1
13
14 # set counter 2 to number interrupts taken (86)
15 echo 0x00560000 > ${QORIQ_DBG}/cpu0/pmlca2
16
17 # set counter 3 to number L2 Cache accesses (110)
18 echo 0x006E0000 > ${QORIQ_DBG}/cpu0/pmlca3
19
20 # enable all counters
21 echo 0x00000000 > ${QORIQ_DBG}/cpu0/pmgc0
22
23 # read the 4 counters
24 for i in `seq 0 3`;
25 do
26     cnt=`cat ${QORIQ_DBG}/cpu0/pmc${i}`
27     echo Core counter ${i} ${cnt}
28 done
29
30
31 #stop counting
32 for i in `seq 0 3`;
33 do
34     cnt=`echo 0 > ${QORIQ_DBG}/cpu0/pmlca${i}`
35 done
36 # freeze all counters
37 echo 0x80000000 > ${QORIQ_DBG}/cpu0/pmgc0
```



Simple Trace With qoriq-dbg

```
#!/bin/sh

# define the base directory
QORIQ_DBG=/mnt/debugfs/qoriq-dbg
TRACE_FILE=example-trace.out

# Example to collect Nexus trace

# Disable the Aurora link
echo 0x40000000 > ${QORIQ_DBG}/nal/nalgcr

#reset the NPC (Erata: DEBUG M)
echo 0x80800001 > ${QORIQ_DBG}/npc/ncr

#reset the NST state
echo 0xffffffff > ${QORIQ_DBG}/npc/nst

# define EPU action to control NPC suppression
echo 0x00000020 > ${QORIQ_DBG}/epu/epacr1

# setup global suppression trigger
echo 0x80000000 > ${QORIQ_DBG}/npc/stcr4

# enable the NPC (suppressed) with internal trace buffer (wrapped)
echo 0x80800002 > ${QORIQ_DBG}/npc/ncr

# set core 0 to send ownership trace with NPIDR, DAQ
echo 0x01002021 > ${QORIQ_DBG}/cpu0/dc1
```

Simple Trace With qoriq-dbg

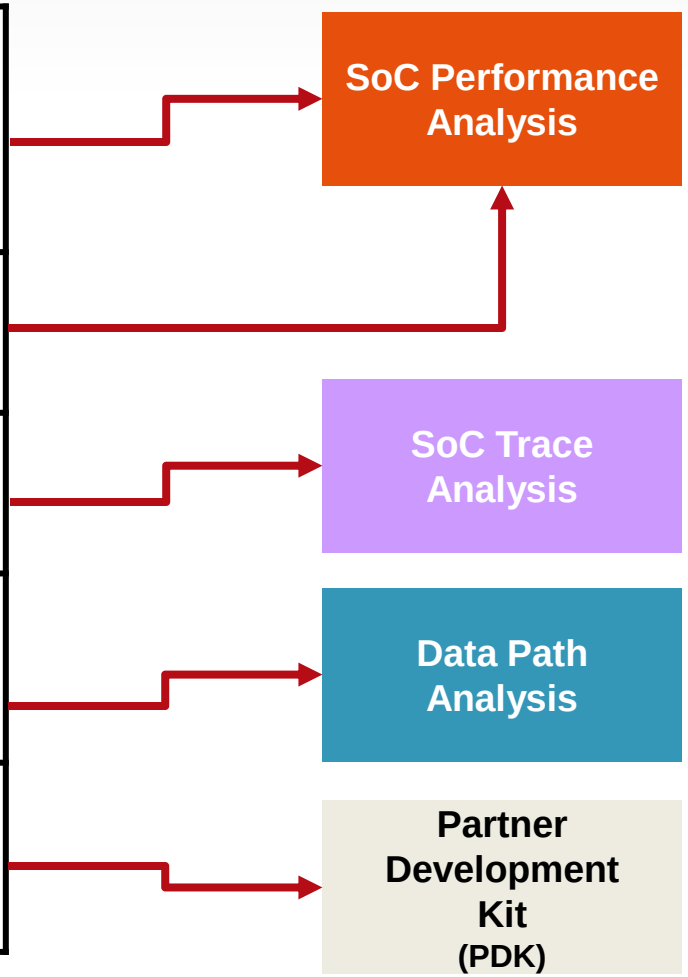
```
echo 0x01002021 > ${QORIQ_DBG}/cpu0/dc1

# Start capturing trace
echo 0x00000001 > ${QORIQ_DBG}/epu/epecr1
echo Starting trace

# Send DAQ
echo 0xFEED0001 > ${QORIQ_DBG}/cpu0/ddam

# Send NPIDR
echo 0xF00DFACE > ${QORIQ_DBG}/cpu0/npidr

# collect some ownership events for a second
sleep 1s

# Stop collecting trace
echo 0x00000001 > ${QORIQ_DBG}/epu/epecr1
echo Stopping trace

# retrieve the trace to a file
echo Capturing the trace
cat ${QORIQ_DBG}/npc/trace_buffer > ${TRACE_FILE}

# reset core 0 tracing
echo 0x00000000 > ${QORIQ_DBG}/cpu0/dc1
echo Done
```

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Use Cases

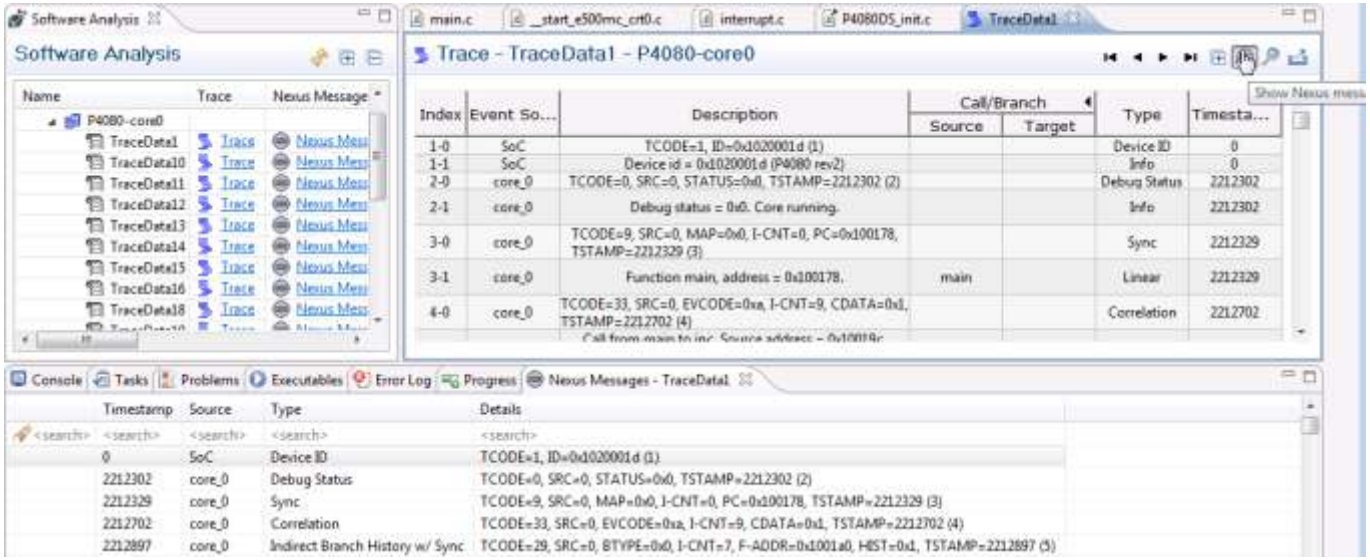
Performance and Optimization Tools – What Problems do We Address?

Complex Performance Metrics	▪ Because SoC is complex, measuring and understanding system performance is also a difficult and complex task. ▪ Setting up device (counters, registers) is sometimes a non-trivial task. It requires access to extensive documentation and information that sometimes is not accessible outside Freescale
Intrusive Performance Monitoring	▪ Many of today's solutions require use of agents on the target. ▪ Some solutions require that the devices be stopped to access performance information.
Extended Trace Analysis	▪ Raw decoded trace is important but needs special expertise to understand. ▪ Sophisticated correlation of the traces from multiple cores.
Debugging the Data Path	▪ Data path applications require core and IP blocks interaction. ▪ There is no way to "single-step" through IP blocks while they operate on data. We must access the debug IP hardware.
Optimized Enablement for Partners	▪ Most 3rd parties do not have access to same information as we do. ▪ Most 3rd parties are not willing to invest the time that it would take to learn the intricacies of the debug IP in our Silicon



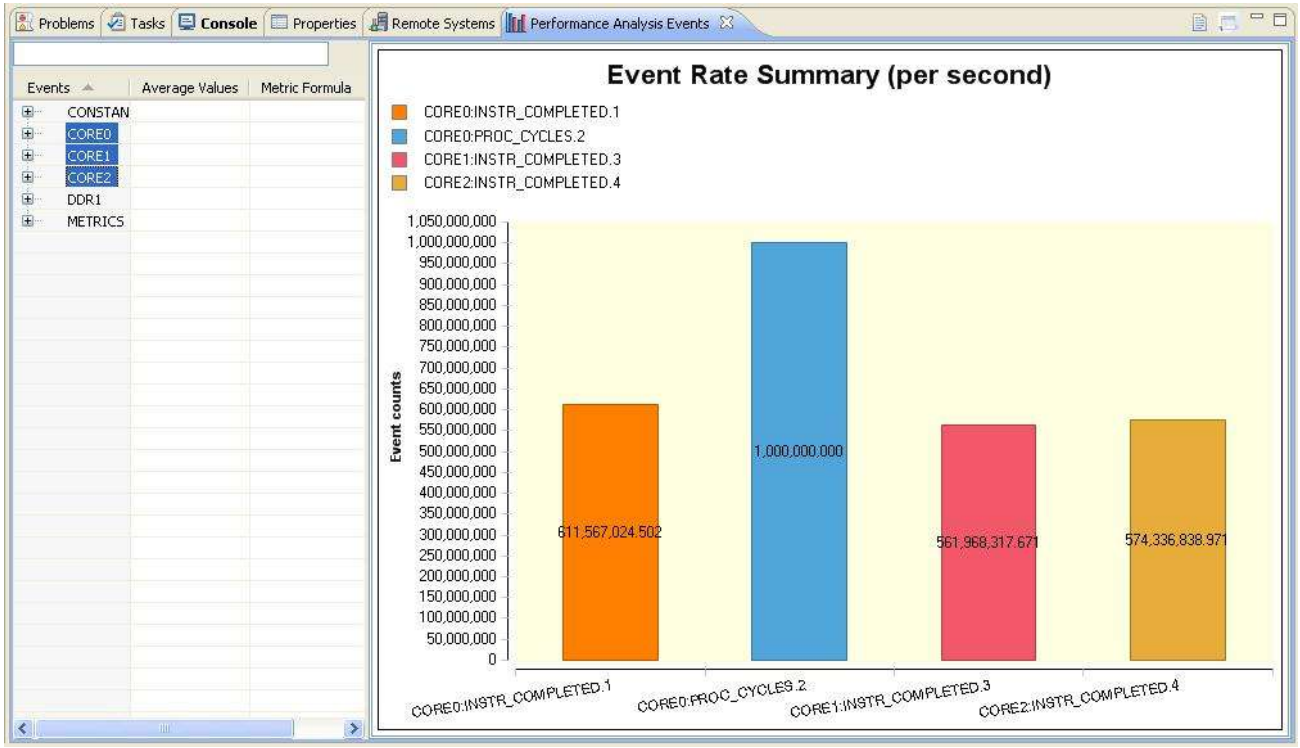
Solving Multicore SoC Programming Issues

Software Problem	CodeWarrior Trace Solution
Code using SoC Hardware capabilities is not producing the expected result. Where is the error?	See what is happening “under the surface” with non-intrusive Nexus trace.
Bugs related to multiple cores accessing shared resources can be difficult to solve.	Multi-core program trace shows you which cores are accessing various resources in chronological order.
What correlation is there between kernel calls, user-space actions and SoC block events?	• Combines open-source Linux kernel trace from LTTng with Nexus trace • Exposes possible correlations • Support for single process user-space trace is available
My multi-core SoC SW crashed. What was happening on the cores and in the SoC blocks immediately before the exception?	• Post-mortem configuration supported • See the last events that occurred before the crash



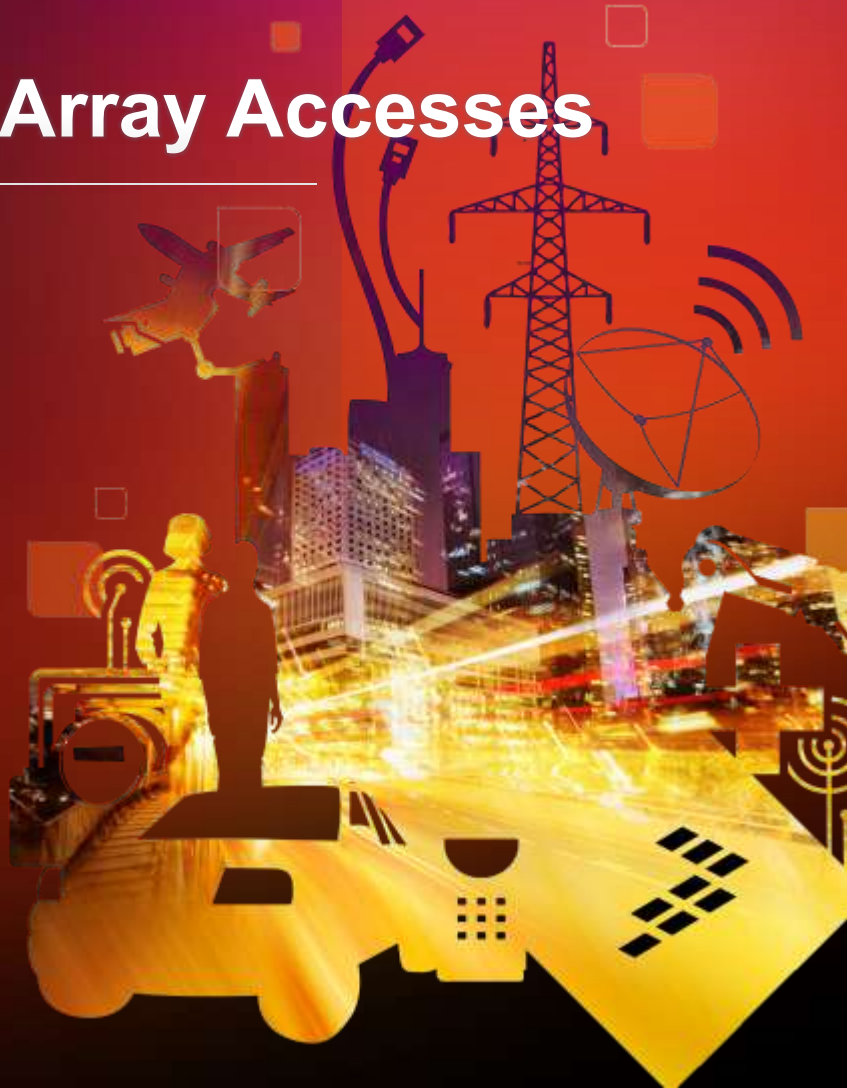
Solving Performance Problems

Software Problem	CodeWarrior Performance Analysis Solution
Program is functionally correct and utilizing SoC features, but not providing the hoped for performance	Use on-chip event counters to identify bottlenecks
Unsure of which events should be measured and how they should be used to identify performance issues	CodeWarrior performance analysis tools provide “canned” scenarios that configure events and counters to help identify common user performance issues



FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Optimizing Array Accesses



Optimizing Array Accesses - 1

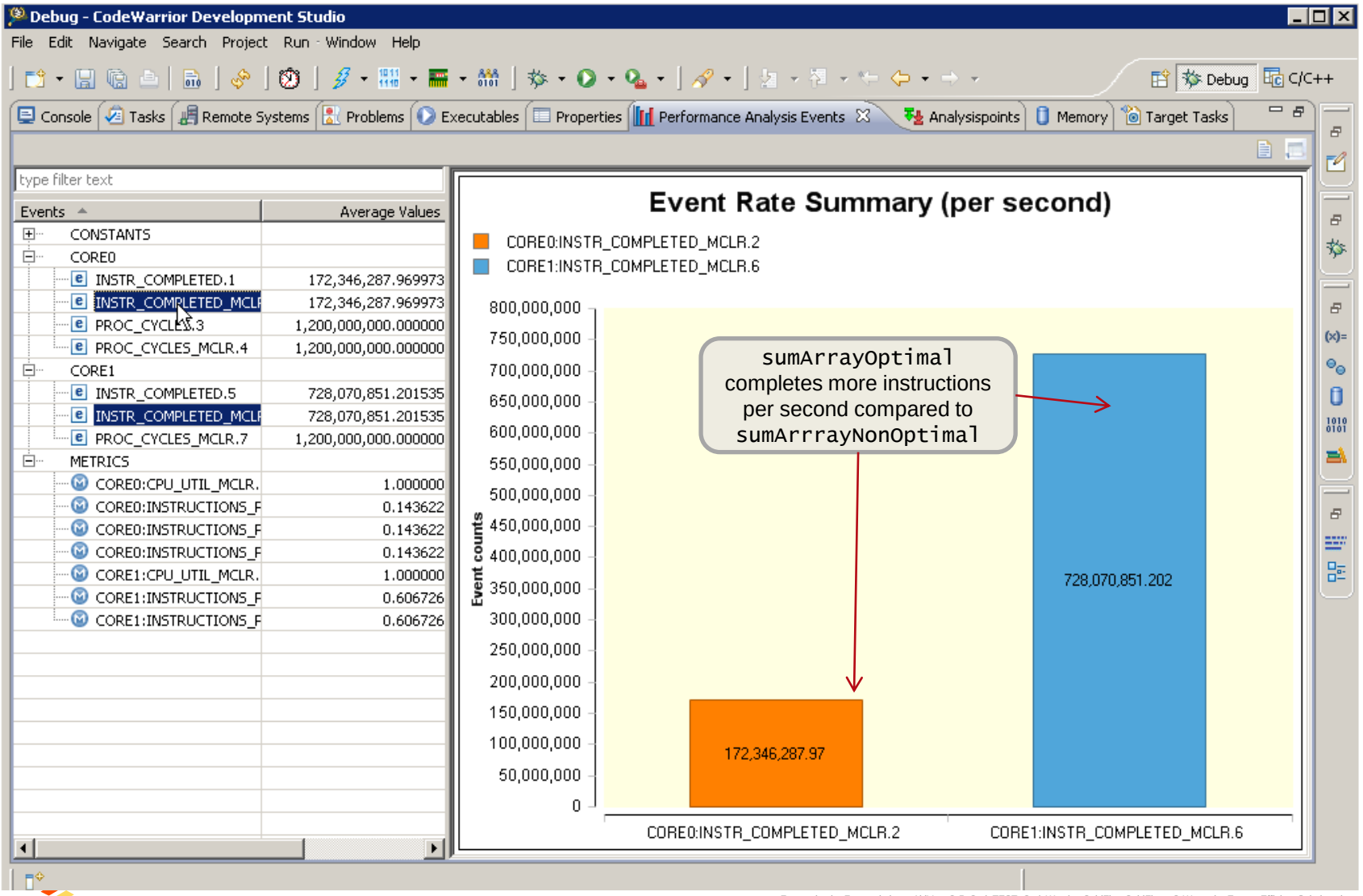
Two similar functions that are manipulating an array in memory

```
unsigned long long sumArrayNonOptimal (int
*array, int w, int h, int n)
{
  unsigned long long ret = 0;
  int i, x, y;
  for (i = 0; i < n; i++)
     for (x = 0; x < w; x++)
          for (y = 0; y < h; y++)
          ret += array[y*w + x];
          return ret;
}
```

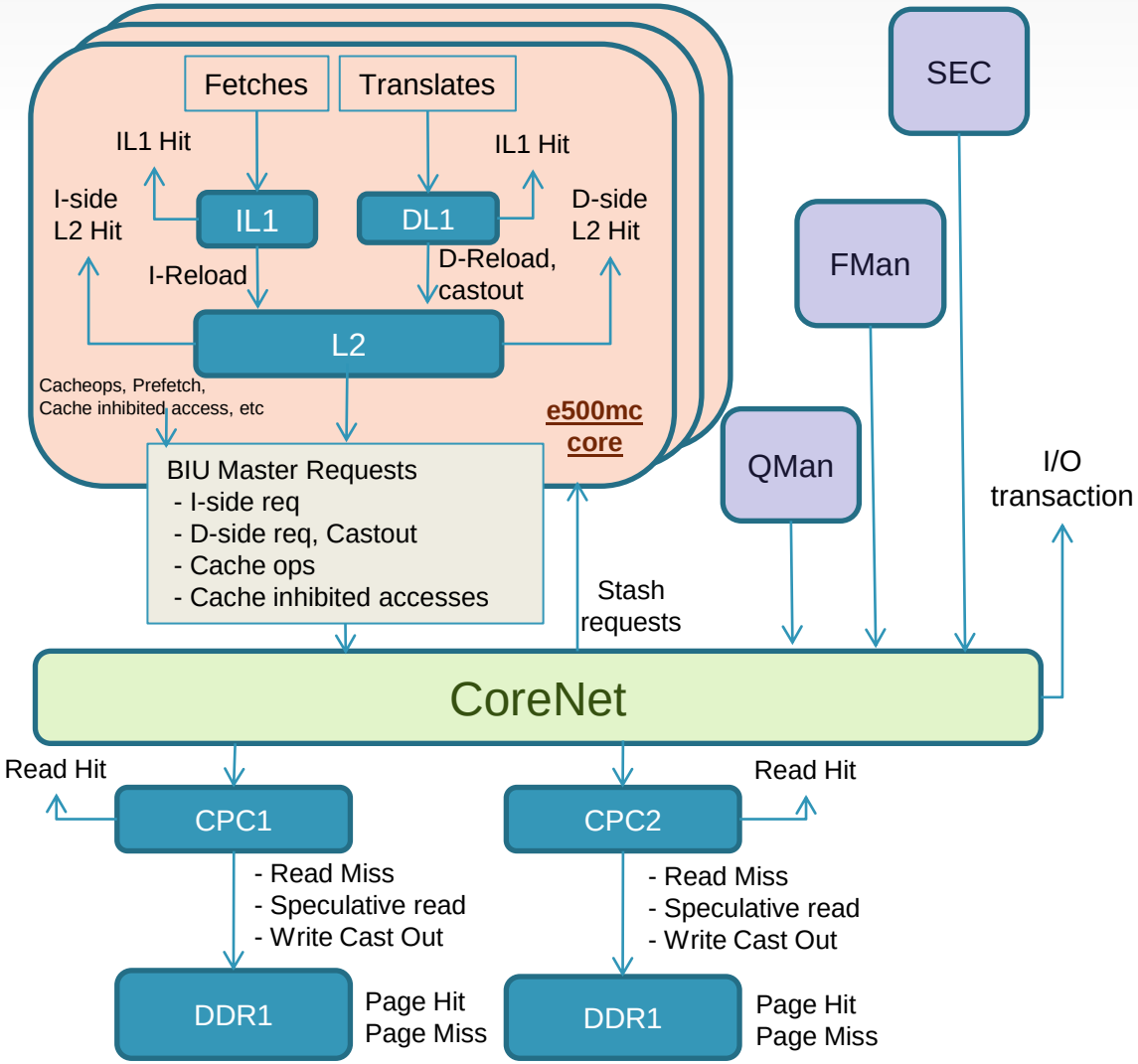
```
unsigned long long sumArrayOptimal (int
*array, int w, int h, int n)
{
  unsigned long long ret = 0;
  int i, x, y;
  for (i= 0; i< n; i++)
     for (y = 0; y < h; y++)
          for (x = 0; x < w; x++)
          ret += array[y*w + x];
          return ret;
}
```

Both functions algorithmically accomplish the same result. But when we run them and measure their performance things are not quite the same

Optimizing Array Accesses – How is the CPU Doing?



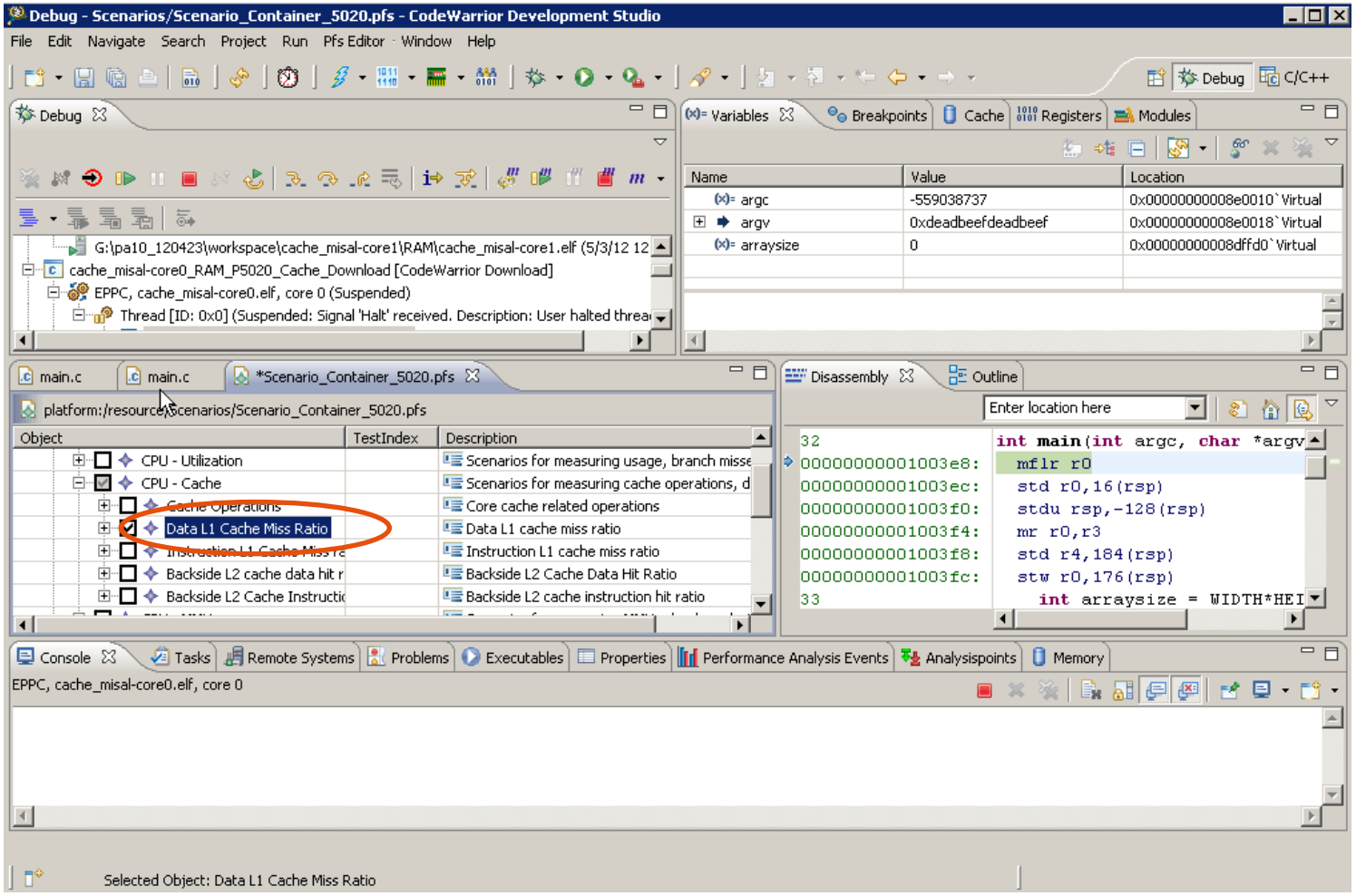
Memory Subsystem Performance Analysis



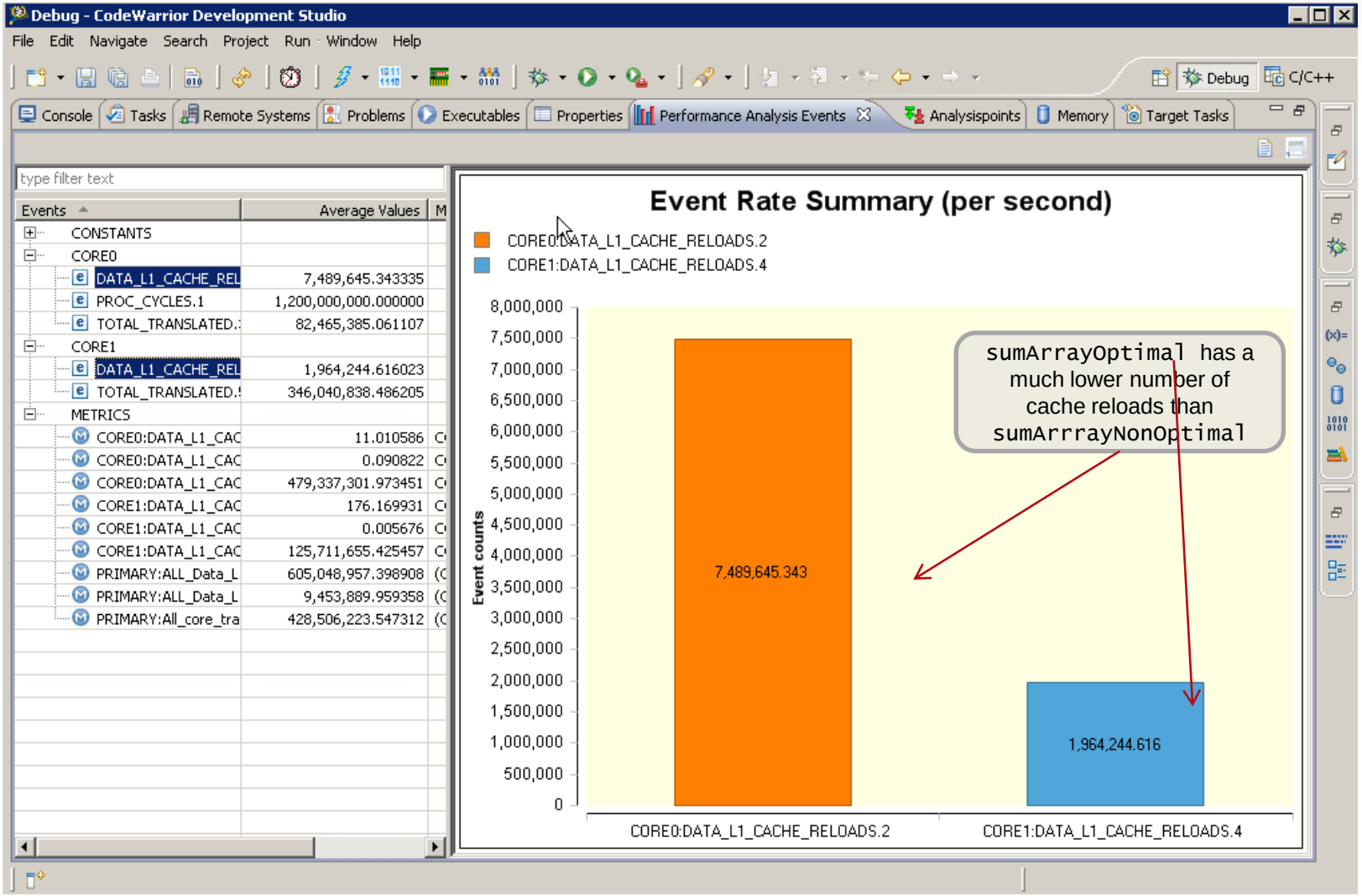
- **Category: Core**
  - IL1
    1. Instruction fetched
    2. IL1 cache reload
  - DL1
    3. Total translated
    4. DL1 cache reload
  - L2
    5. L2 I-side access
    6. L2 I-side hits
    7. L2 D-side access
    8. L2 D-side hits
  - BIU
    9. BIU master requests
    10. I-side BIU requests
    11. D-side BIU requests
    12. D-side BIU castout requests
  - Stash
    13. Stash requests
- **Category CoreNet**
  14. I/O transactions
- **Category CPC**
  15. CPC read hit
  16. CPC read miss
  17. CPC write hit
  18. CPC write miss
  19. CPC other hit
  20. CPC other miss
  21. CPC castout
- **Category DDR**
  22. DDR page hit access
  23. DDR page miss access

Optimizing Array Accesses

- PA10 Performance Analysis – Measure Cache reloads



Optimizing Array Accesses



Optimizing Array Accesses – Analysis

- It is obvious from previous analysis that even though both functions compute “answer” correctly, there are differences.
- One routine (nonOptimal)
  - Is not very efficient CPU-wise
  - Shows inefficiencies in the use of the cache
- Why?

Optimizing Array Accesses – Analysis

- This code accesses memory walking down the rows
- Each of these accesses loads the cache with data, but it only uses the first element in array
- Subsequent accesses cause cache to be reloaded.

1,1	1,2	…	1,n
2,1	2,2	…	2,n
….	…	…	…
n,1	n,2	…	n,n

- This code accesses memory by selecting a row
- Each of these accesses loads the cache with data
- It then walks the array by columns
- The cache is not reloaded until you get to the end of the line.

```
for (i = 0; i < n; i++)
     for (x = 0; x < w; x++)
          for (y = 0; y < h; y++)
          ret += array[y*w + x];
          return ret;
}
```

Non-Optimal

```
for (i= 0; i< n; i++)
     for (y = 0; y < h; y++)
          for (x = 0; x < w; x++)
          ret += array[y*w + x];
          return ret;
}
```

Optimal

References

- DPAA Primer: http://www.freescale.com/webapp/search/Serp.jsp?QueryText=qoriqdpaawp&assetIdResult=&fsrch=1&sessionChecker=qgZmPRhSbqPVPR2hffsC08PGLCmVhGXLnglSWbJPhm9N9tJXfxnS%21880891312%2113391056500 63&attempt=0&showCustomCollateral=false&RELEVANCE=true&fromTrng=false&showAllCategories=false&fromMobile=false&isResult=false&isFromFlex=false&isTree=false&pageSize=25&fromASP=false&isAdvanceSearch=false&getTree=false&fromPSP=false&lastQueryText=qoriqdpaawp&iteration=1&assetLocked=false&assetLockedForNavigation=false&fromCust=false&getFilter=false&fromDAP=false&fromWebPages=false&getResult=false&isComparison=false&SelectedAsset=Documentation

Q&A

Facebook.com/Freescale
Tag yourself in photos
and upload your own!

Tweeting?
Please use hashtag
#FTF2012

Session materials will be posted @ www.freescale.com/FTF
Look for announcements in the FTF Group on LinkedIn or follow Freescale on Twitter

freescale™


freescale™