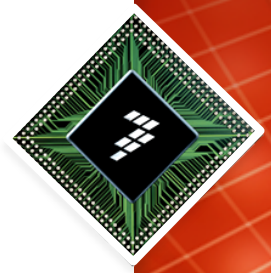
LTTng and Nexus Trace for Freescale QorIQ Devices

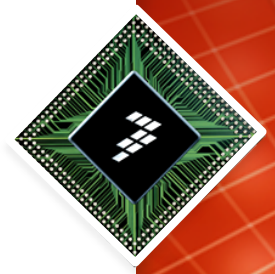
Ed Martinez

CodeWarrior Software Analysis

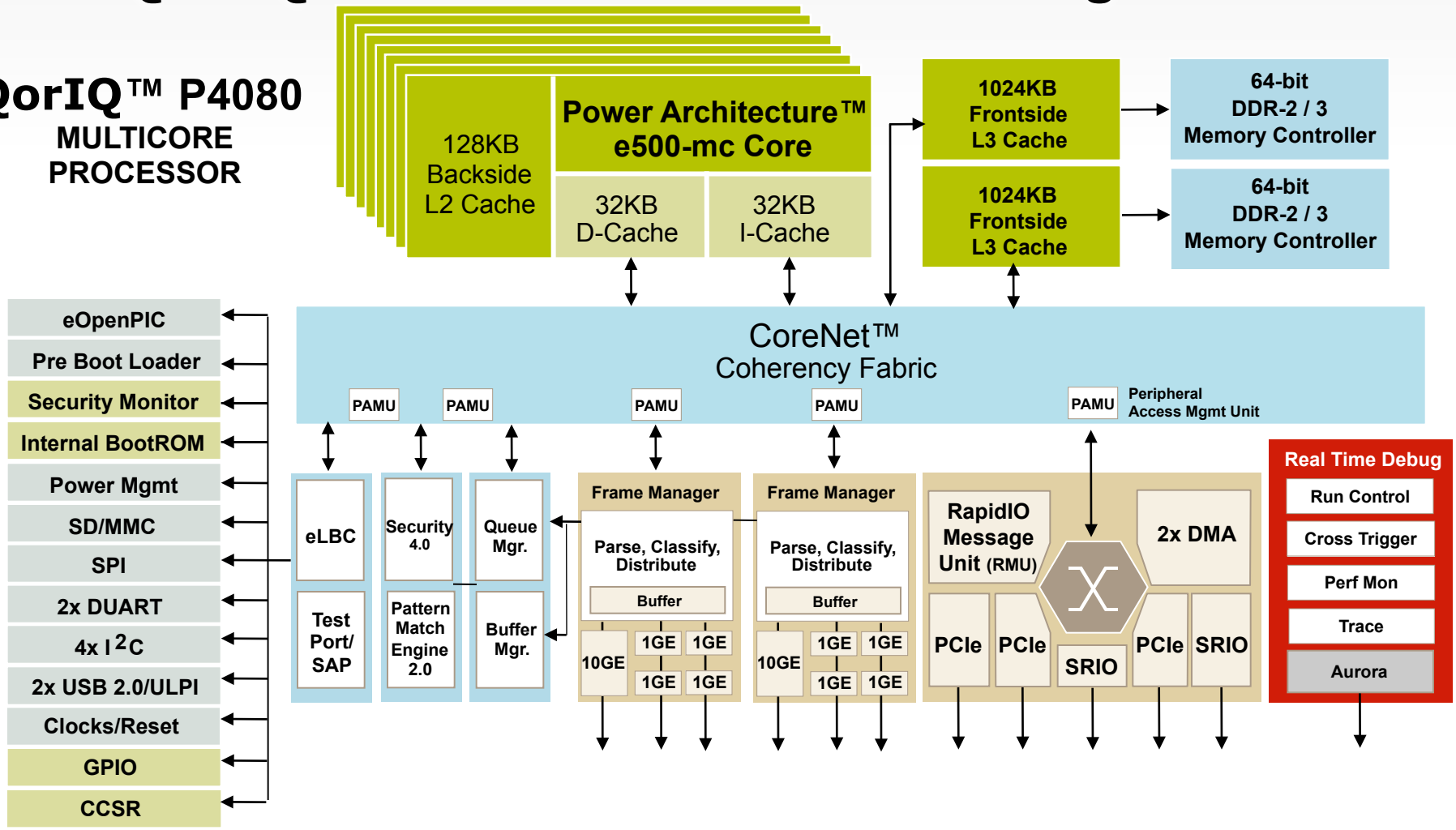


August 26, 2012

Brief Introduction to QorIQ Debug Architecture

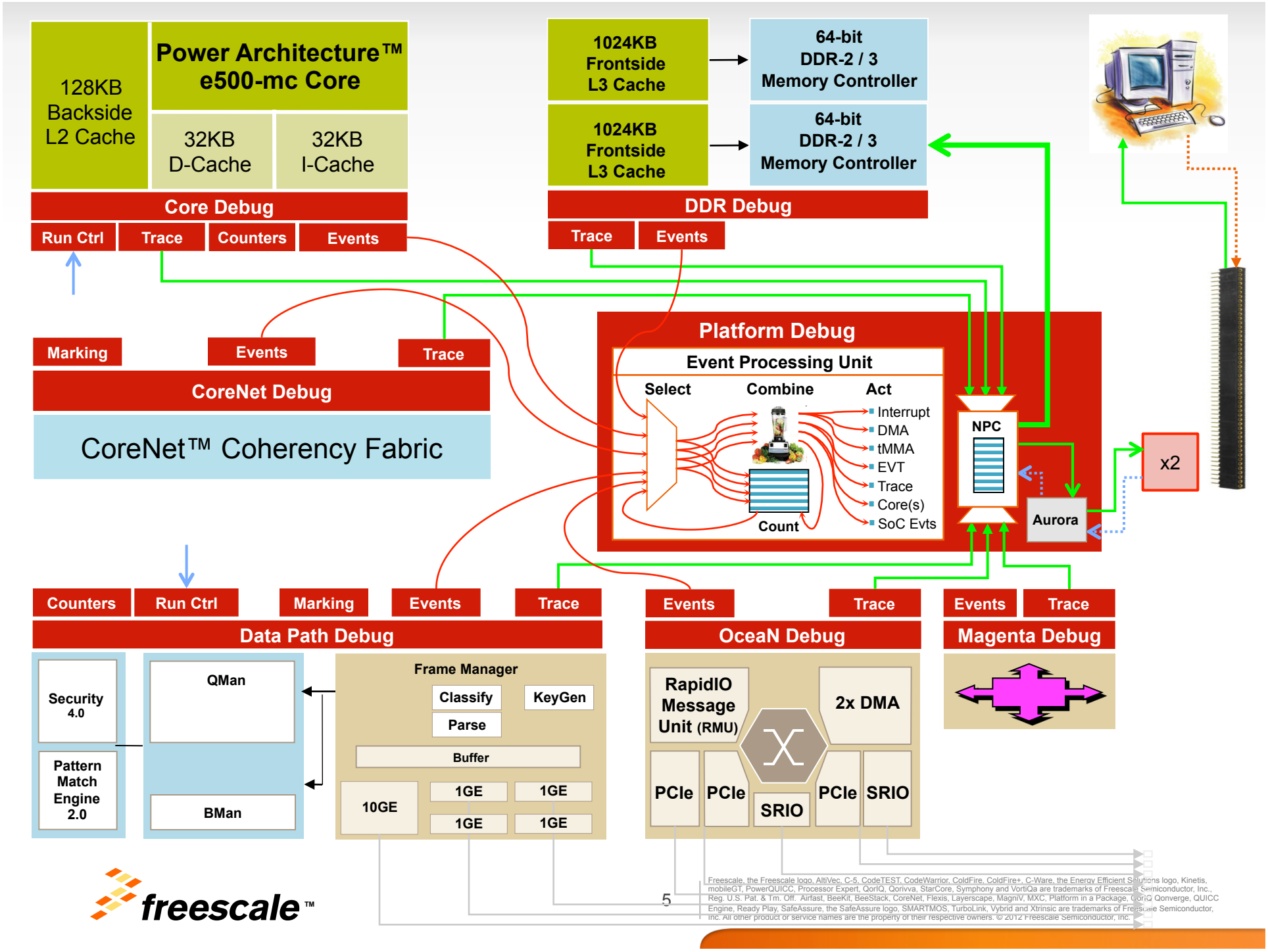


QorIQ™ P4 Series P4080 Block Diagram

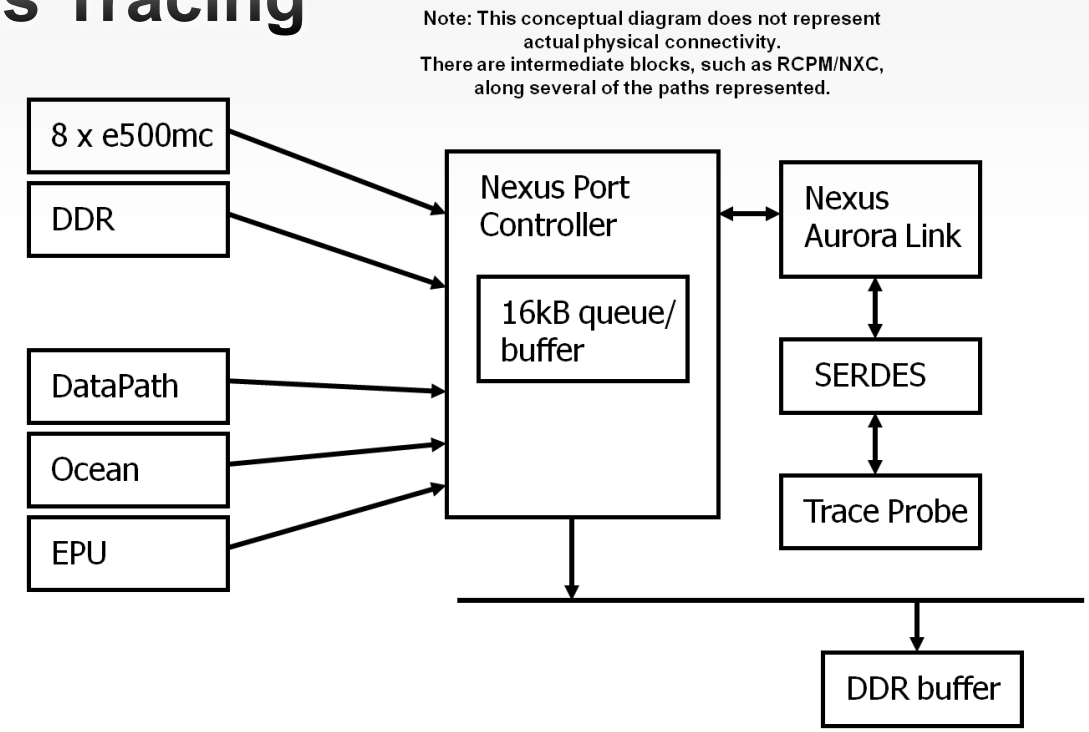


Debug Capabilities Overview (very brief!)

- **Run Control**
  - Provides access to registers and memory and can start or stop the cores as needed
- **Real-time Trace**
  - Provides an nonintrusive log of instructions, data and various events that happens in the SoC
- **Performance Monitoring**
  - Supports the ability to count a variety of events on the cores and various SoC blocks.
- **Cross-Triggering**
  - Allows debug events from various functional areas in the device to be selected, filtered and combined to trigger a variety of SoC, Core, and Off-Chip actions

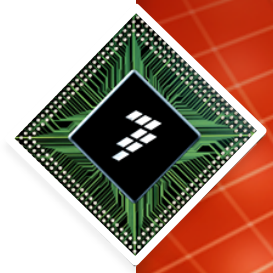


Nexus Tracing



- QorIQ devices can capture Nexus trace to one or more of the following configuration buffer locations (certain tradeoffs apply)
  - Internally in the Nexus Port Controller (NPC) Trace Buffer
  - A trace buffer defined within the SOC's DDR memory
- Trace can be captured by
  - Using Freescale CodeWarrior
  - Python scripts (provided) These scripts configure the trace logic and perform a simple trace collection by using either of the two trace buffer locations.

Freescale LTTngX



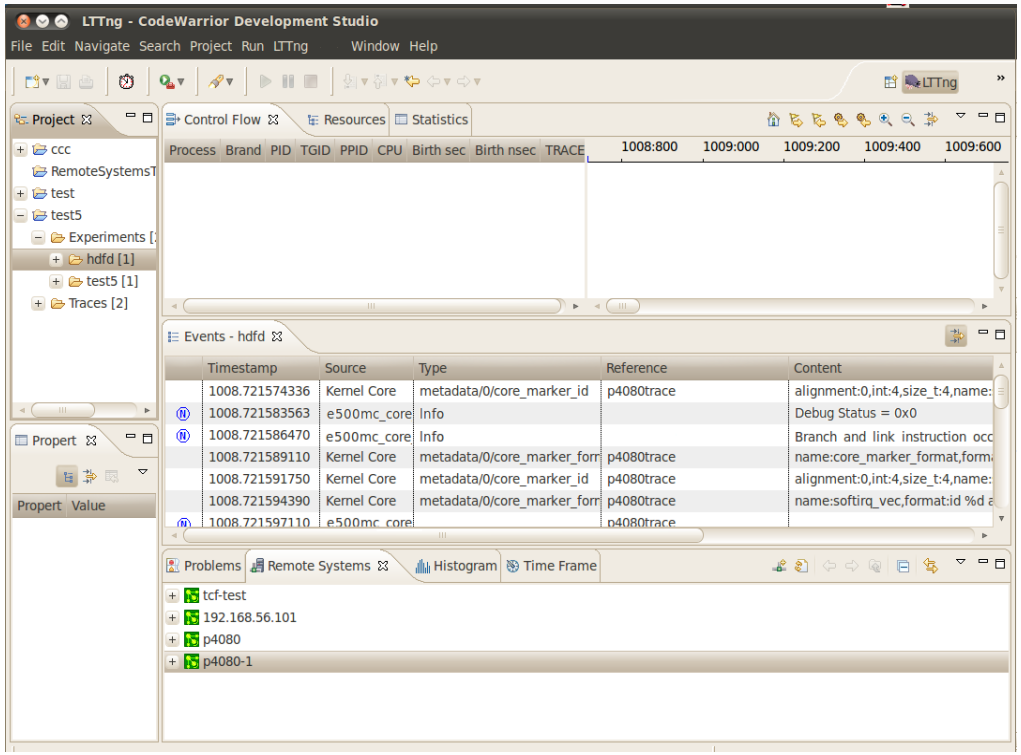
PA10 LTTngX

What

- Freescale's enhancement to the Eclipse Community LTTng 1.0 Toolkit
- Provides a consolidated view of LTTng trace information and QorIQ Nexus Trace information.
- Designed to allow users to include Nexus events in the LTTng Views

Features

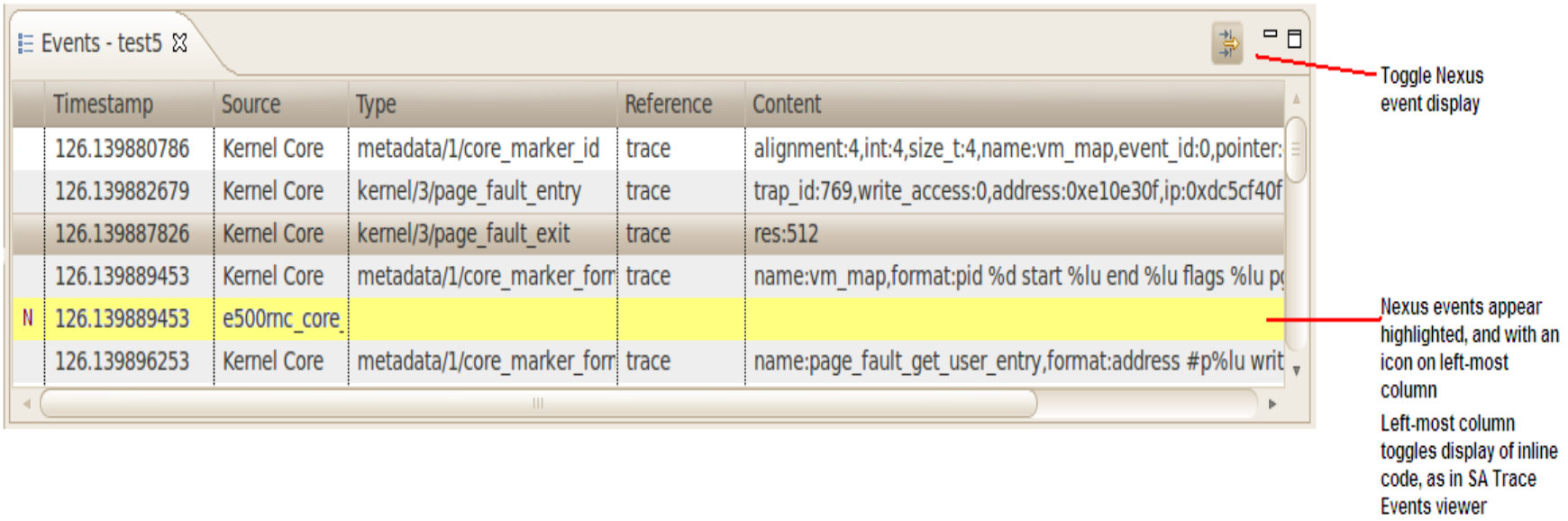
- Import existing CodeWarrior Nexus traces.
- Synchronize LTTng traces to Nexus Trace.
- Synchronized display of LTTng and Nexus traces.
- Display of LTTng and Nexus events side by side.
- Users can observe how the LTTng events correlate with Nexus events.
- Ability to display Nexus trace events together with LTTng events in the LTTng views.



Design Constraints

- One of the objectives of the integration was to be able to display the Nexus event information in the LTTng events view.
  - The CodeWarrior (CW)Trace views has features that are not present in the LTTng Events view.
  - Integration effort involved adding features to the LTTng views to make if feature-wise equal to Freescale's CodeWarrior View.
  - The missing features in the LTTng event view were search, filter, and disassembly display.
- We wanted to allow the user to display CW trace data in the LTTng views even if no LTTng trace data is available.
  - In this way, we can leverage the LTTng views like the Histogram to give users additional representations of the CW trace data.

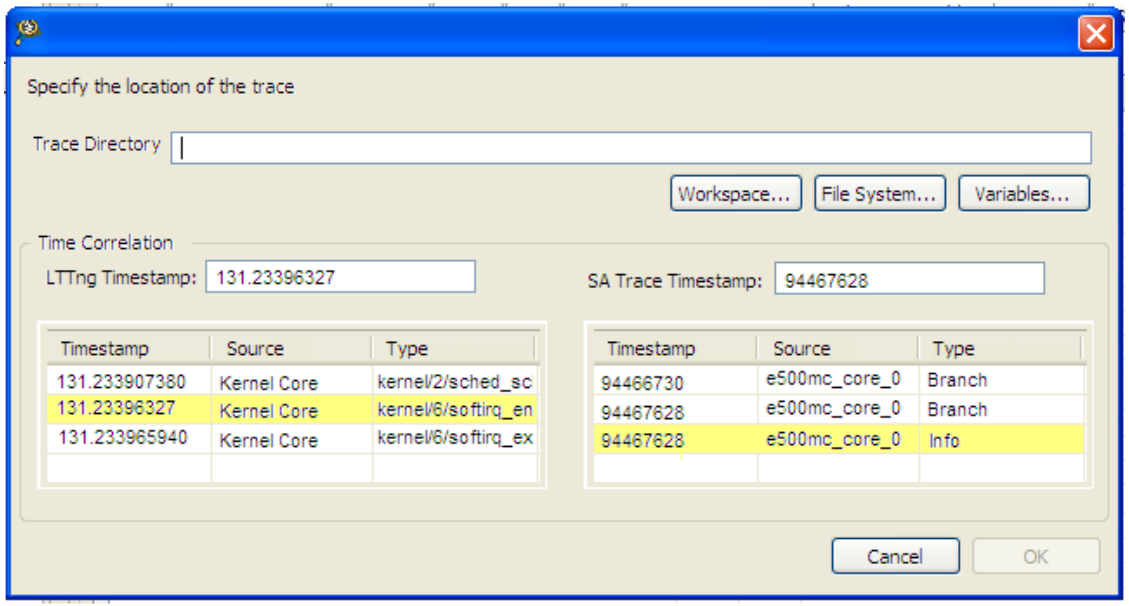
LTTng Events View Integration



- Figure above shows LTTng view and the initial set of changes to the view.
- When adding Nexus trace events to the view, it should be obvious to the user which events are Nexus and which ones come from LTTng

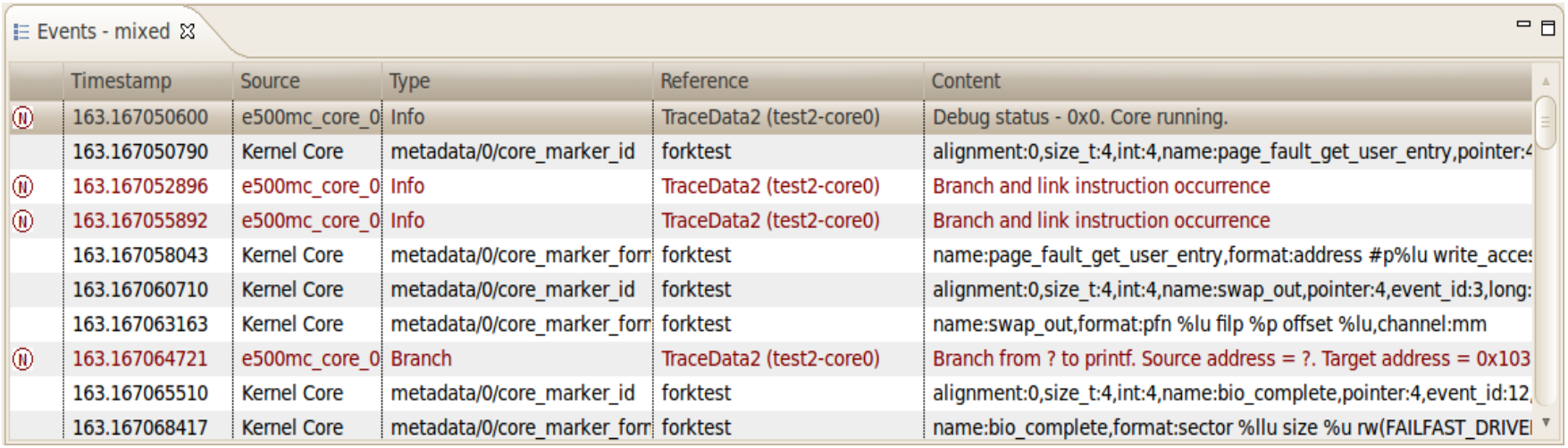
LTTng and Nexus Trace Synchronization

- The first time the toggle button is pressed, a dialog will be presented to the user to select the Nexus trace of interest.
- Since the timestamps for the events will likely differ, it will be necessary to allow the user to specify a common reference point on which to base the timestamps from the various sources:



- This dialog allows selection of Nexus trace to be added to the LTTng view.

Viewing Nexus and LTTng Trace Data Together

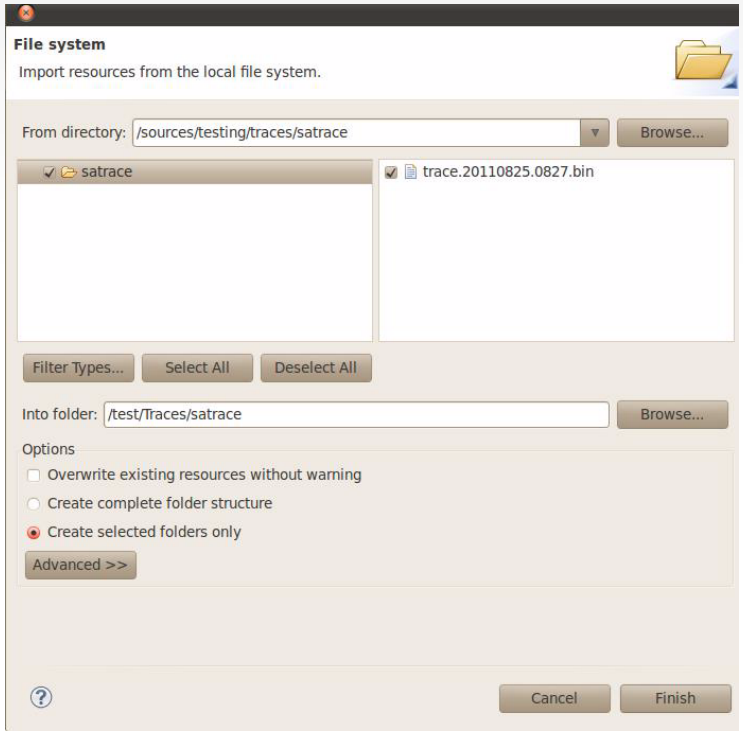


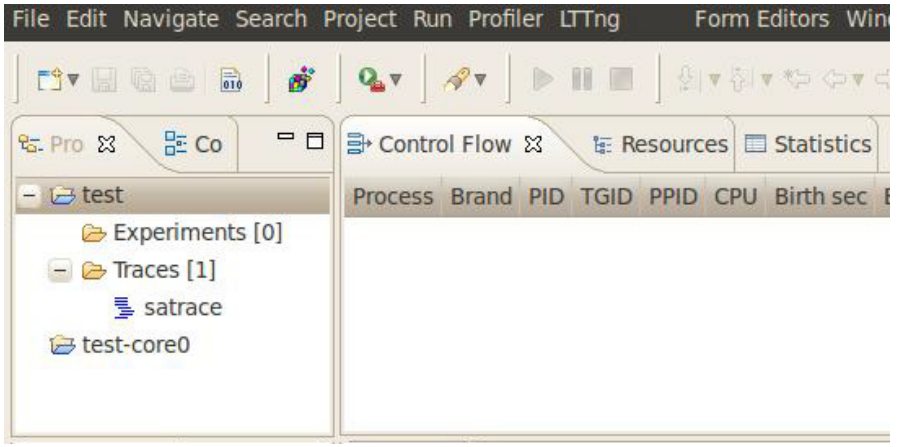
	Timestamp	Source	Type	Reference	Content
N	163.167050600	e500mc_core_0	Info	TraceData2 (test2-core0)	Debug status - 0x0. Core running.
	163.167050790	Kernel Core	metadata/0/core_marker_id	forktest	alignment:0,size_t:4,int:4,name:page_fault_get_user_entry,pointer:4
N	163.167052896	e500mc_core_0	Info	TraceData2 (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)
- You can observe how the Nexus events correlate with the LTTng generated events

Importing Nexus Trace

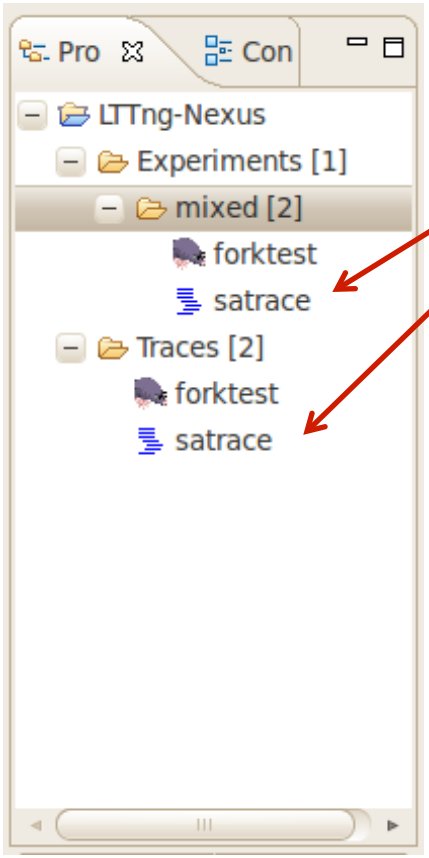
- Import Nexus trace data to the LTTng projects through the same mechanism as used for importing LTTng trace.
  - Select your project in the **Project view, and right-click the Traces folder.**
  - Select **Import Trace from the context menu.**
  - The **Import dialog box appears.**
  - Click **Browse against the From directory field, and select a directory from where you want to import** the trace data.
  - Click **OK.**
- The trace appears in the **Import dialog box.**





PA10 Linux Trace Support: LTTngX

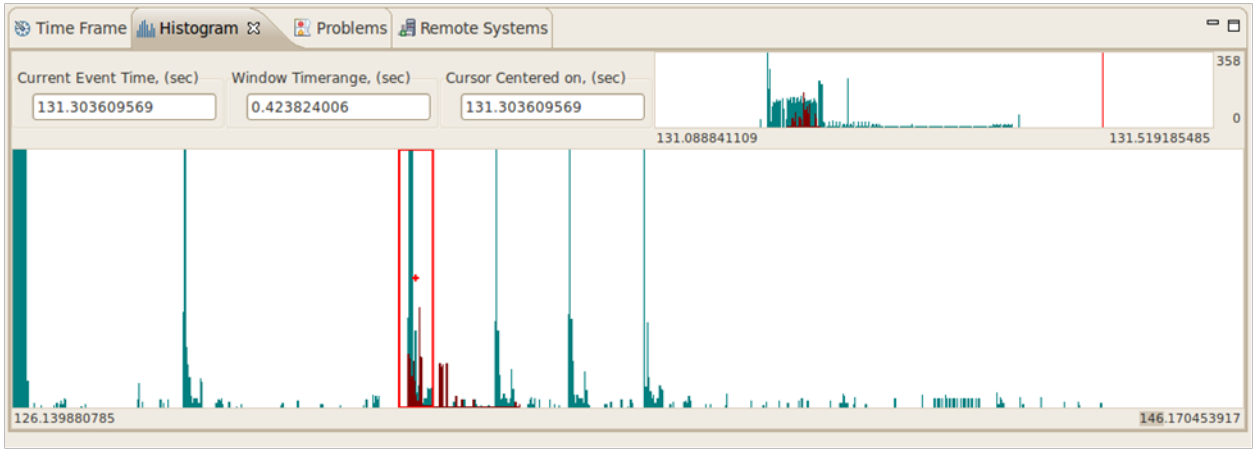
- You can differentiate between a Nexus trace and a LTTng trace in the LTTng project by the different icons.



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

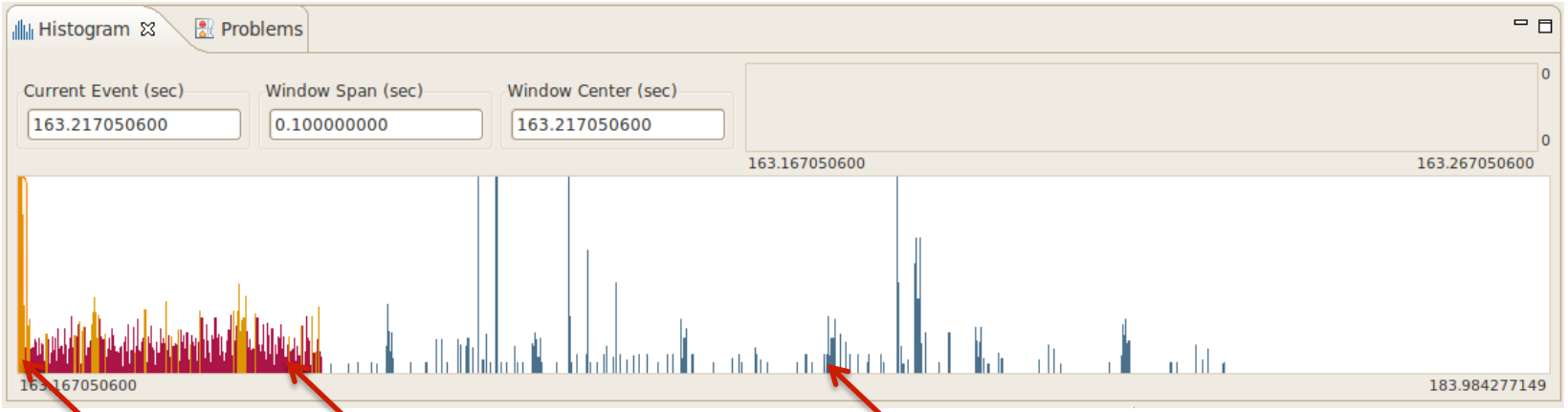
Viewing Nexus and LTTng Trace in the Histogram

- We modified the histogram view to allow users to view Nexus trace events and see the correlation of those events with the LTTng events being monitored.
- When the user enables the display of Nexus trace events in the events view, the other LTTng views will adjust accordingly. The histogram might look as the following screen shots depict:



The Nexus events being displayed in the picture above are drawn in a way that the user can easily differentiate between them and the LTTng events.

LTTngX



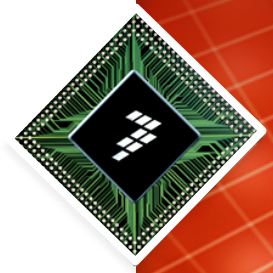
Orange: mixed LTTng and Nexus events

Red: Nexus events

Blue: LTTng events

- The histogram view displays color-coded events

Quick Use Case Description



Brief discussion of an LTTngX Use Case

- Scenario
  - A user is investigating an issue in their embedded system that is causing network performance problems intermittently. There are multiple applications running in user space. The user is unsure whether the problem is with the applications, or with the network driver.

- The user can:
  - Enable LTTng kernel tracing in the system, and instrument user space applications with LTTng UST.
  - Enable LTTng kernel tracing in the system and the collection of Nexus Trace

- If the source code is available, second option available more compelling as Nexus trace can replace LTTng UST providing source code reconstruction and would show synchronization of LTTng kernel events to the user's source code.

LTTngX usecase walkthrough

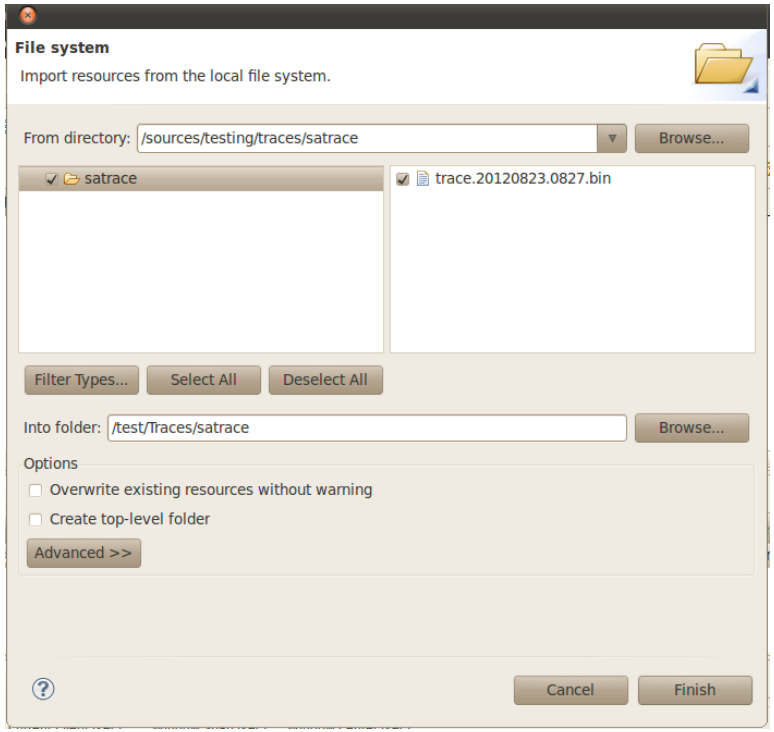
- Step 1: enable LTTng kernel tracing on target system
- Step 2: build and deploy nexus based userspace tracing modules and libraries
- Step 3: capture traces

```
root@p5020ds-32b:~# lttctl -C -w /tmp/trace1 trace1
root@p5020ds-32b:~# python ntrace.py -cmdline=./networkingapp -prog -filterprograce -tstamp -
cpu=[0] -ddr
…
root@p5020ds-32b:~# lttctl -D trace1
```

- Step 4: view and analyze

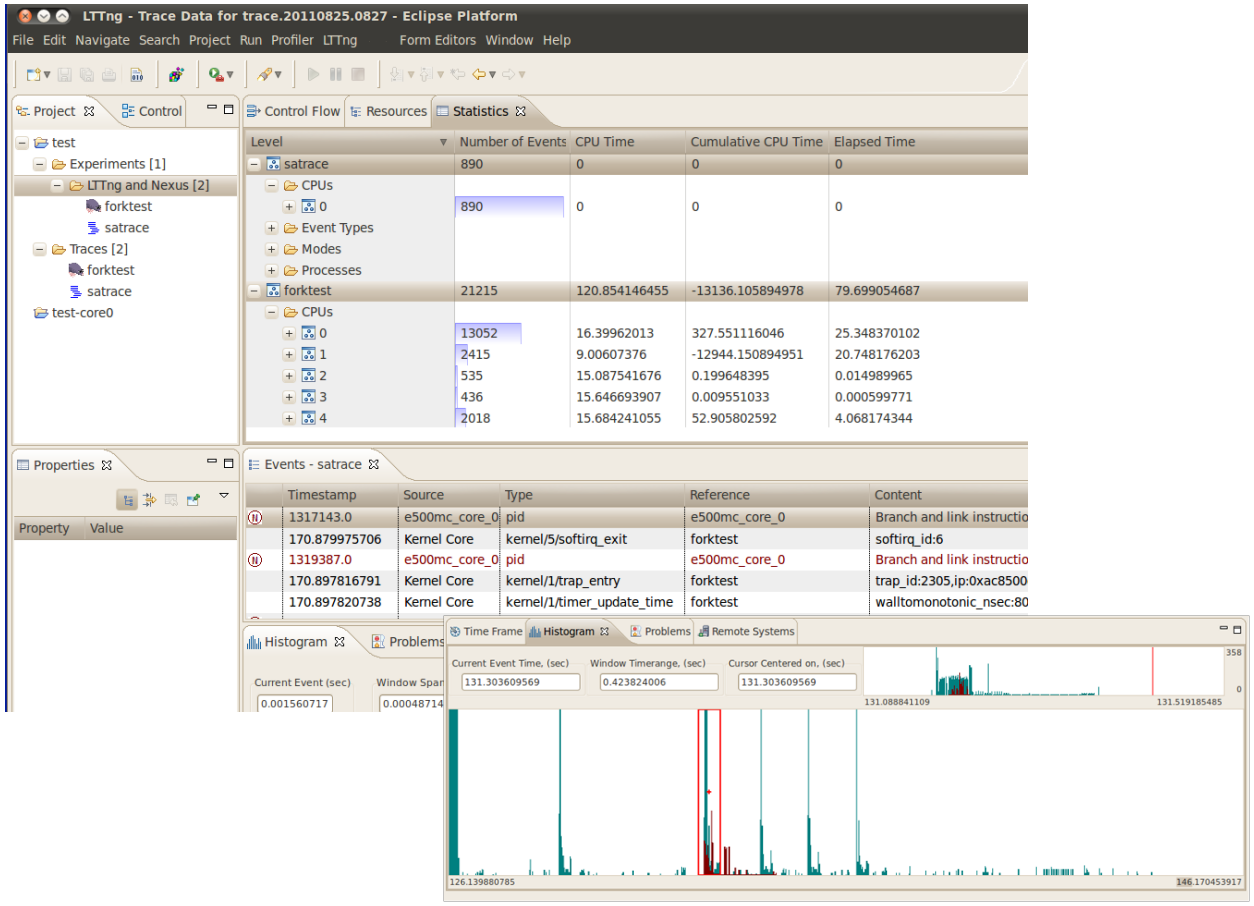
LTTngX use case walkthrough

Importing the nexus trace follows the same process as importing LTTng traces:



LTTngX use case walkthrough

Users can analyze both userspace and kernel traces together as with LTTng UST



What is next?

- Improve support for multicore.
- Add support (to the viewers) for viewing coordinated, synchronized Nexus trace from core(s) and SoC Nexus clients
- Add support for mixing trace events along with performance events from SoC blocks
- Add support for newer devices that we have coming out.
- Migrate to LTTng 2.0 – currently we are on LTTng 1.0. We plan to completely review our current LTTng and LTTngX support based on LTTng 2.0

freescale™

Thanks