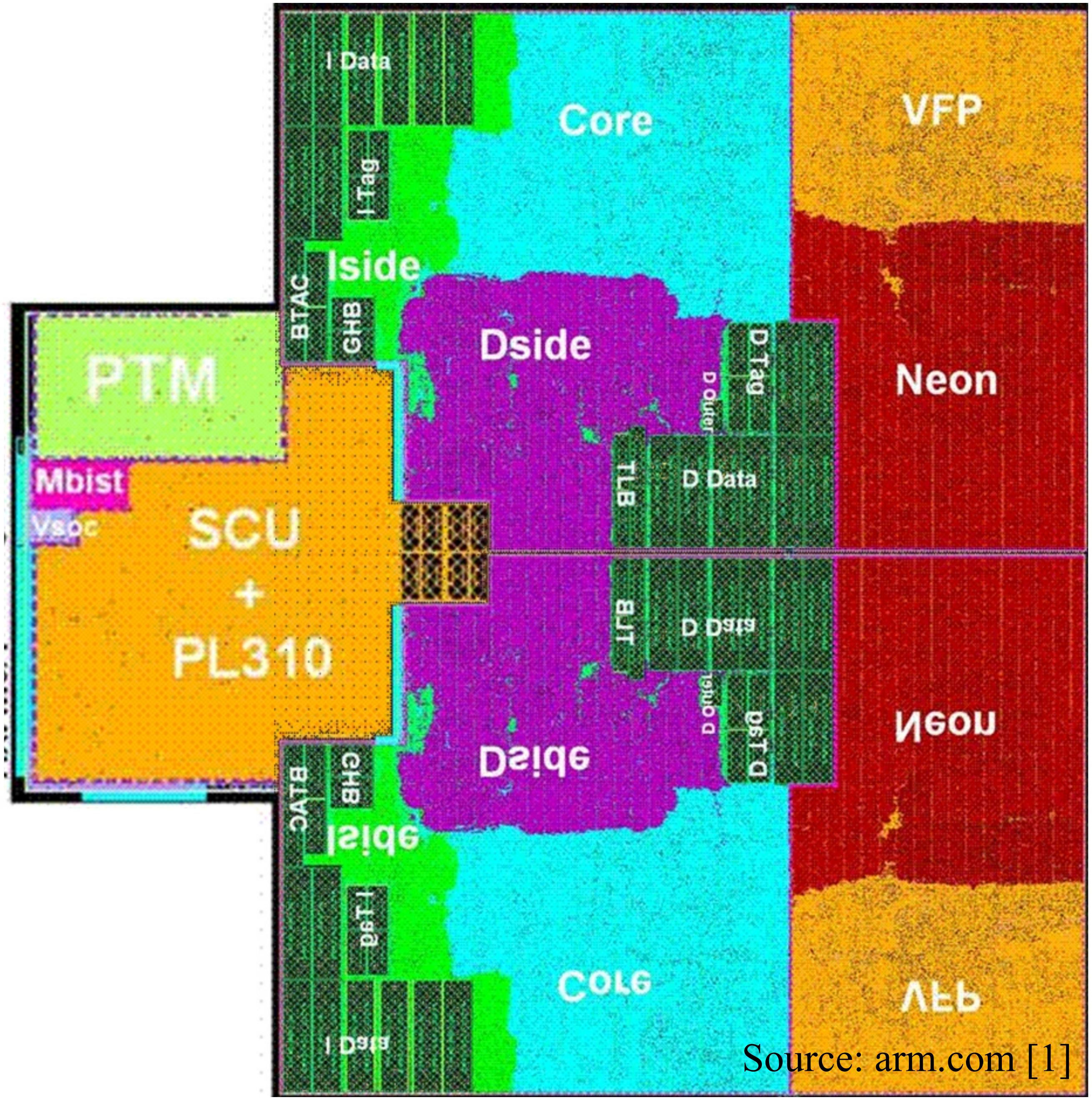
Embedded Linux Conference Europe 2013

# Bridging the gap between hardware and software tracing

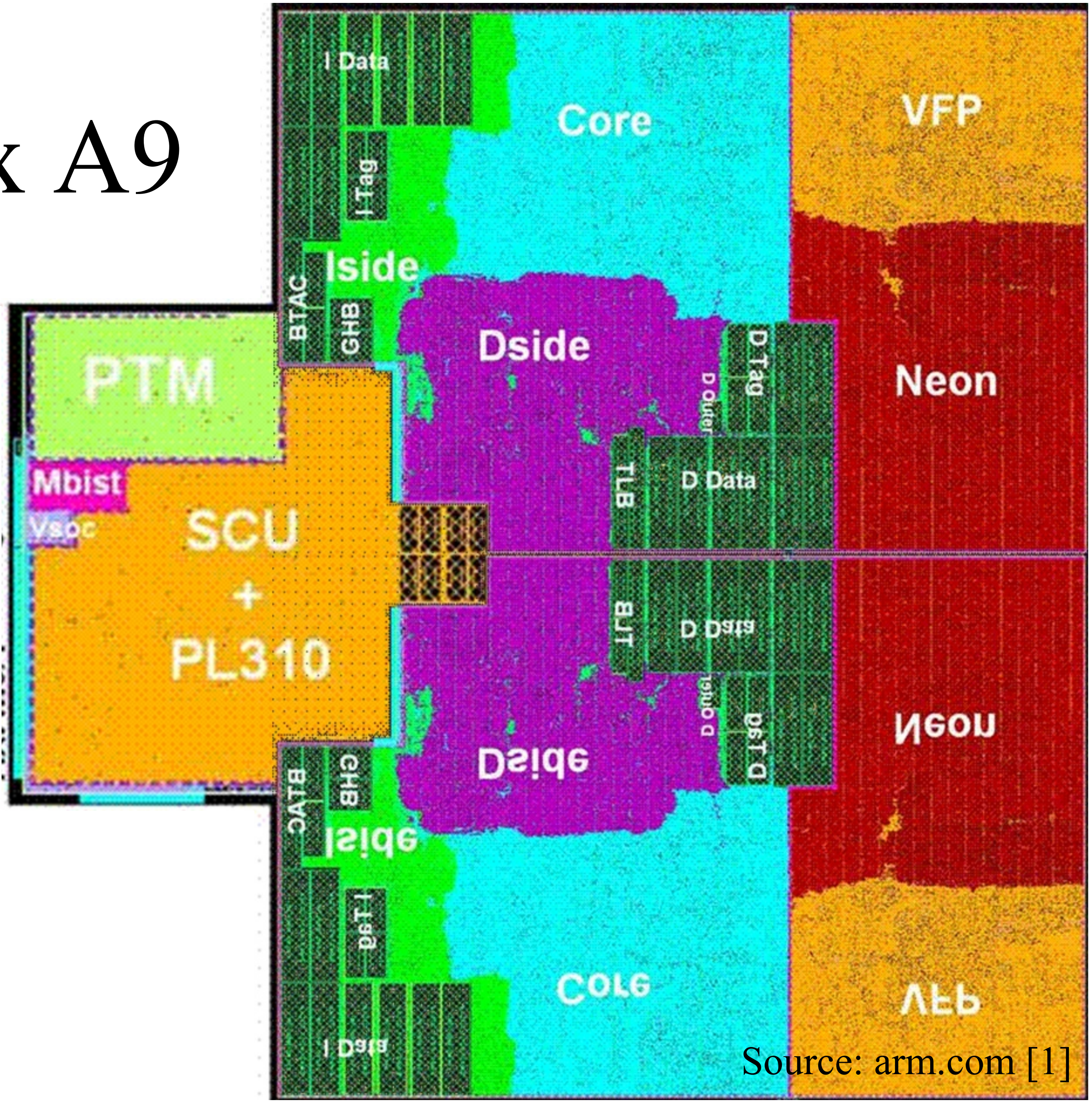
EfficiOS

christian.babeux@efficios.com
@c_bab



Source: arm.com [1]

# ARM Cortex A9

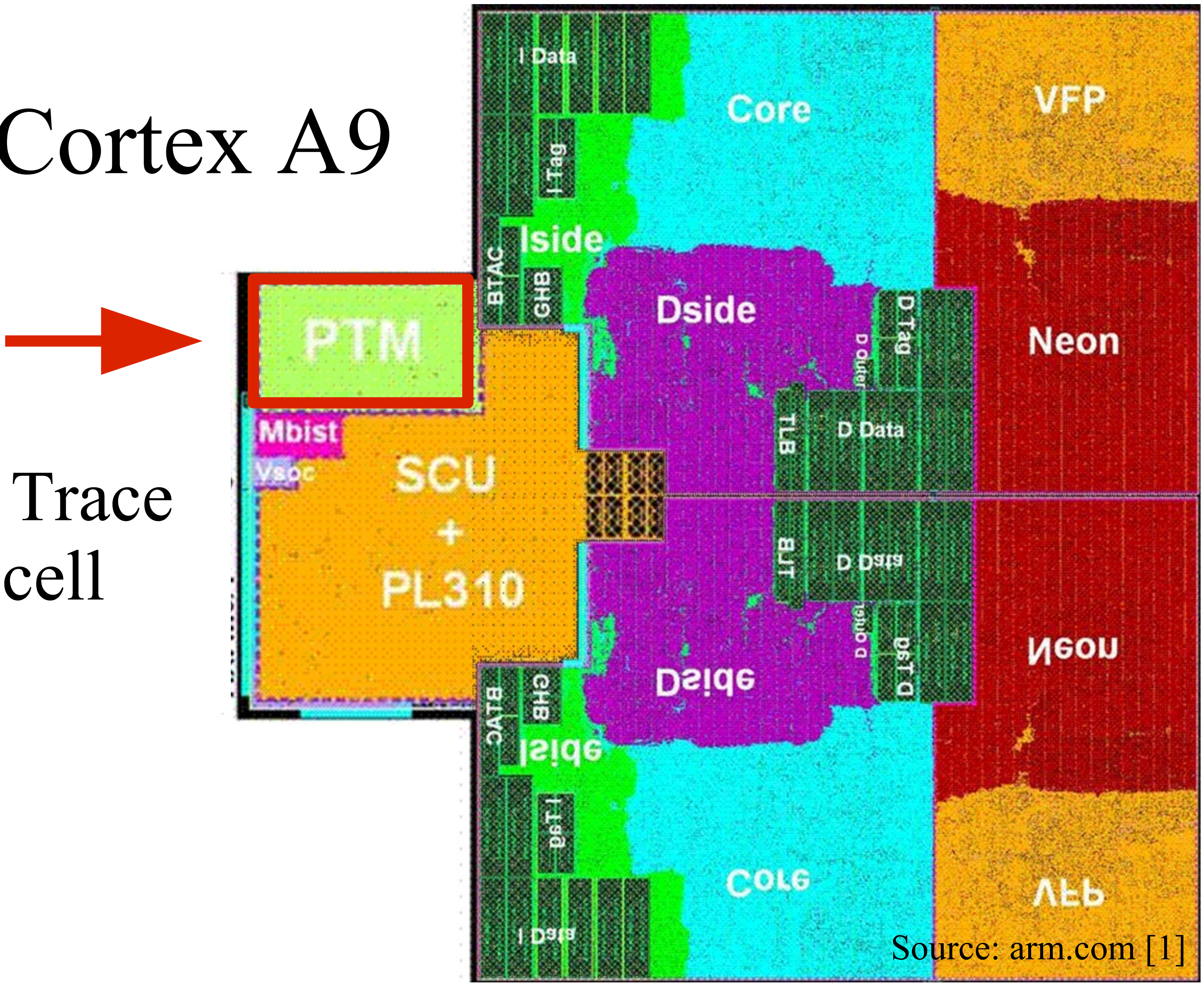


Source: arm.com [1]

## ARM Cortex A9

PTM

Program Trace Macrocell



Source: arm.com [1]

## whoami

- Christian Babeux, Software Developer, EfficiOS,
- Background in embedded and ASIC tools,
- Active contributor to the LTTng projects:
  - lttng-tools, lttng-ust, babeltrace,
  - CI infra, Website, Twitter.
- AUR package maintainer for Arch Linux.

## Content

- What is hardware tracing?,
- Why is it useful?,
- ARM Coresight & ETM,
- Freescale QorIQ & Nexus tracing,
- LTTng & hardware tracing.

## What is hardware tracing?

## What is hardware tracing?

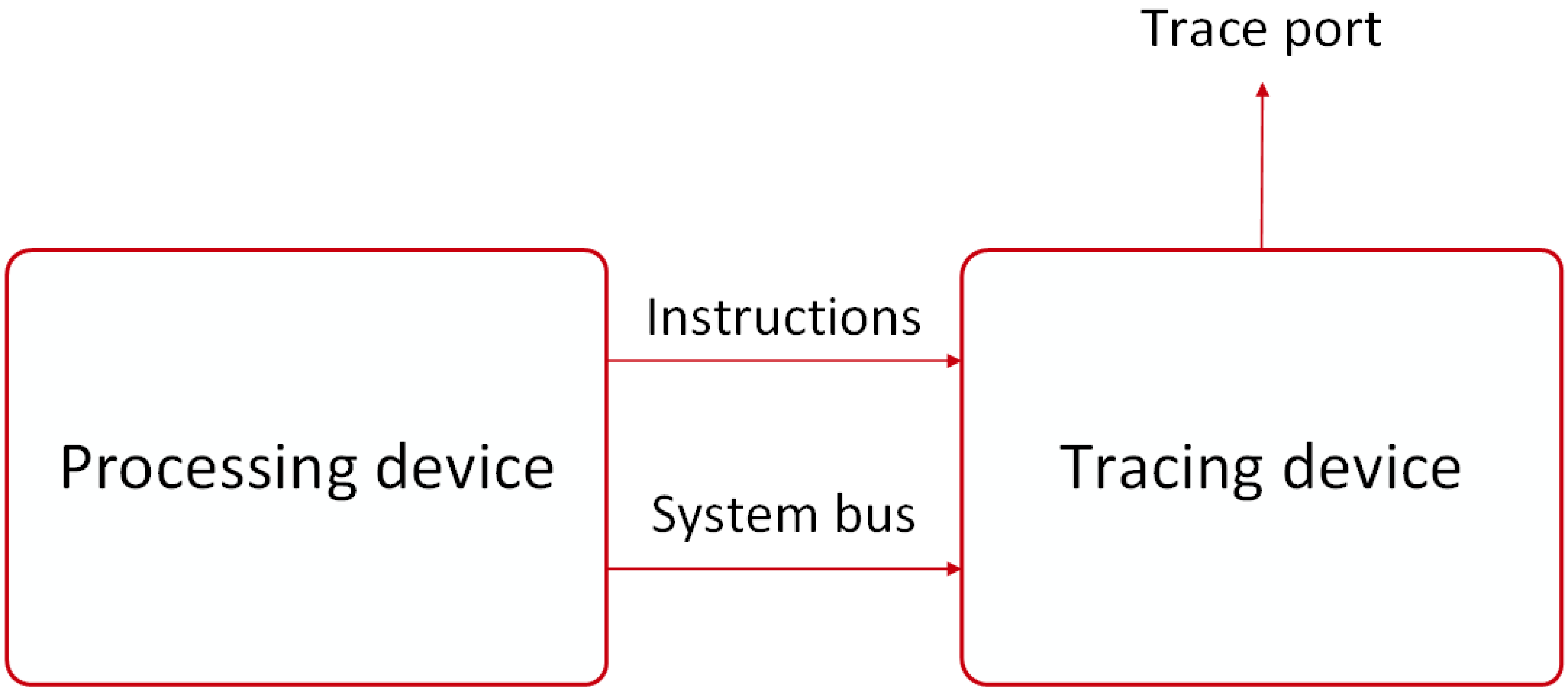
Hardware component(s) used to traces instructions and data movement of a processing device

Real-time observation

Low intrusiveness

## What is hardware tracing? Cont.

- External trace



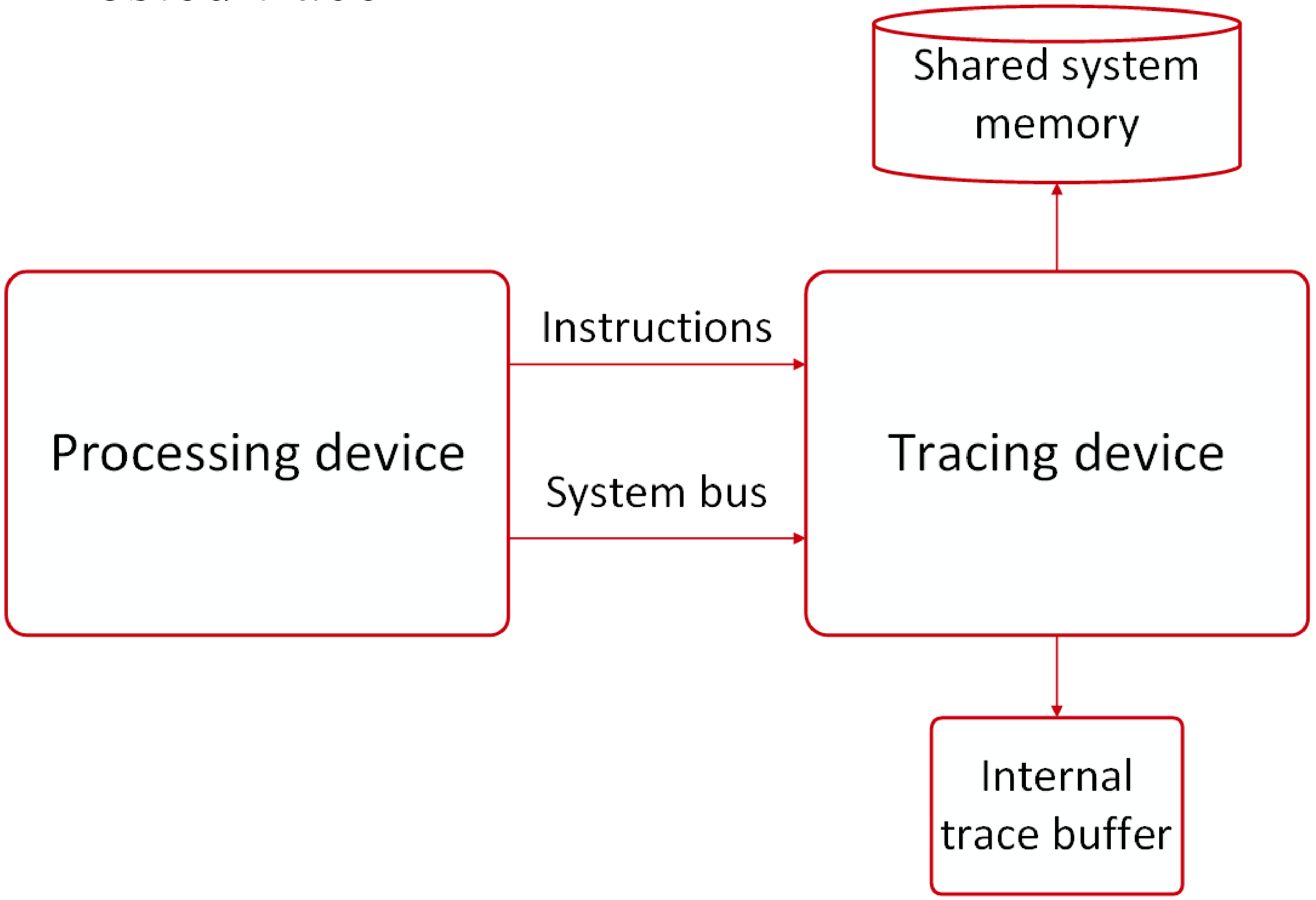
## What is hardware tracing? Cont.

- Pros
  - Can accommodate high data bandwidth
  - Minimal impact on system performance

- Cons
  - Trace port not always available
  - Custom hardware needed

## What is hardware tracing? Cont.

- Self-hosted trace



## What is hardware tracing? Cont.

- Pros
  - Self-contained, facilities can be used by host OS
  - No need for special hardware
- Cons
  - Limited internal buffer space
  - Might impact system performance

## Hardware tracing support

- ARM
  - Embedded Trace Macrocell (ETM),
  - Program Trace Flow Macrocell (PTM),
  - System Trace Macrocell (STM)
- PowerPC
  - Freescale QorIQ with Nexus tracing
  - Branch History Rolling Buffer (BHRB)
- Intel
  - Intel Processor Tracing (PT)
  - Last Branch Record (LBR), Branch Trace Store (BTS)

## Hardware tracing vs. software tracing

- Software tracing:
  - Static instrumentation or dynamic code patching
  - Can be intrusive
  - Can be slow
  - Tracepoint level granularity

- Hardware tracing:
  - Tracing done on hardware
  - Instrumentation not required
  - Instruction level granularity

## Why hardware tracing is useful

## Why hardware tracing is useful

- Profiling
  - Very fine granularity profiling
- Performance measurement
- Code coverage
- Monitoring
  - Statistics on application currently running?

# Why hardware tracing is useful cont.

- Snapshot on crash/anomaly
  - Trace overwrites old data until anomaly detected

- Event trigger trace

- Hardware-assisted software tracing
  - Use hardware facility (ringbuffer)

## ARM Coresight & ETM

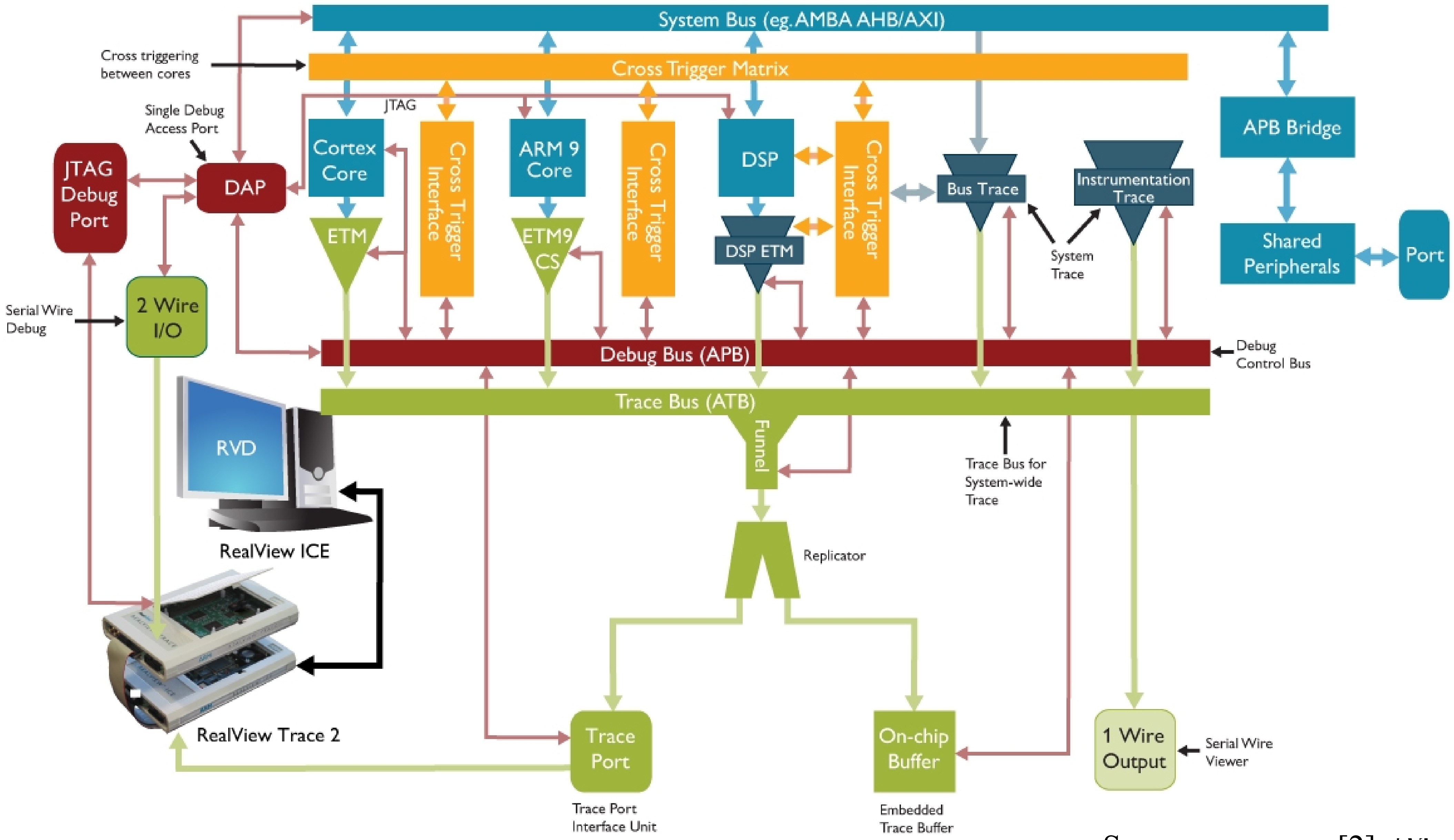
# ARM Coresight

- Coresight
  - Collection of hardware components
  - Trace and debug a complete SoC
  - Open architecture
- Trace source
  - Processing elements (CPU, DSP, etc.)
  - Buses
  - System trace (generated from software)

## ARM ETM

- ETM
  - Monitor the core internal bus
  - Instructions + data trace
  - Hardware filters and triggers
  - Trace stream compression
  - Traces can be saved in internal buffer (ETB) or shared system memory

# ARM Coresight



Source: arm.com [2]

## State of Coresight & ETM in Linux

- ETM tracer implementation available in Linux
    - Seems to work only on specific hardware
- Coresight support status
    - Framework patchset proposed by Pratik Pratel [3]
- Trace decoder availability?

## Interested about Coresight/ETM?

"Hardware Trace in the Kernel" BoF
by Pawel Moll from ARM

Today, 4:30 PM in Pentland

## Freescale QorIQ & Nexus tracing

## Freescale QorIQ

- PowerPC based platform targeted for high-performance communications usage
  - Multiple e500mc processors,
  - DPAA support (packet processing offloading),
  - Support the Nexus debugging & tracing standard.

Source: radiolocman [4]

## Nexus standard

- ISO standard for debugging embedded systems (IEEE-ISTO-5001-2003),
- Designed for low pin count, standard set of connectors (JTAG or Debug port),
- Multiple level of Nexus “compliance”.

## Nexus standard level

- Level 1:
    - Run time control only (run, stop, breakpoints, etc.)
    - Tracing not supported
- Level 2:
    - Ownership and program trace
- Level 3:
    - Data write trace & memory read/write on the fly
- Level 4:
    - Memory substitution
    - Trace triggering via a watchpoint

## Nexus standard format

- Packet based output format,
  - Standard defines public messages, vendors can define extensions (TCODES)
  - Fixed packet size per message, last packet can be of variable length
  - Message can have an optional timestamp

# Example decoded Nexus message

```
Message # 328
TCODE : 2 Ownership Trace Message
SRC ID  : 0 Core0 / CPU0 (Clst0:Core0:Thread0)
PID INDEX  : 0x02 - Sync PID
PID VALUE : 0x01e01bf59c - LPIDR, MSR[GS],
PID/NPIDR (ref:DC1[OTS]), MSR[PR]
TIMESTAMP : 1140061 (0x11655d)


Message # 329
TCODE   : 27 Resource Full Message
SRC ID    : 0 Core0 / CPU0 (Clst0:Core0:Thread0)
RCODE   : 0x8 - Timestamp counter
RDATA    : 0x00ffffff
TIMESTAMP : 0 (0x0)
```

## State of Nexus in Linux

- Nexus qoriq-debug kernel module,
  - Available in Freescale QorIQ SDK Yocto Layer
  - Implements a debugfs with memory mapped access to Nexus control register
  - cat /sys/kernel/debug/npc/trace_buffer > trace

# Nexus debugfs

```
root@p3041ds:~# ls -al /sys/kernel/debug/qoriq-dbg
[...]
drwxr-xr-x  2 root root 0 Aug  6 20:22 cpu0
drwxr-xr-x  2 root root 0 Aug  6 20:22 cpu1
drwxr-xr-x  2 root root 0 Aug  6 20:22 cpu2
drwxr-xr-x  2 root root 0 Aug  6 20:22 cpu3
drwxr-xr-x  2 root root 0 Aug  6 20:22 ddr1
drwxr-xr-x  2 root root 0 Aug  6 20:22 dpaa
[...]
drwxr-xr-x  2 root root 0 Aug  6 20:22 npc
drwxr-xr-x  2 root root 0 Aug  6 20:22 nxc
[...]
root@p3041ds:~# ls -al /sys/kernel/debug/qoriq-dbg/cpu0
[...]
-rw-rw-rw-  1 root root 0 Aug  6 20:22 dc1
-rw-rw-rw-  1 root root 0 Aug  6 20:22 dc2
-rw-rw-rw-  1 root root 0 Aug  6 20:22 dc4
--w--w--w-  1 root root 0 Aug  6 20:22 ddam
[...]
```

## State of Nexus in Linux

- Nexus decoder availability
  - Released as part of the babeltrace project

- Open-source software to reconstruct program flow not yet developed
  - Integrate such functionality in an IDE ? perf?

## LTTng & Hardware tracing

## Project overview

Tracers

Utilities

Viewers

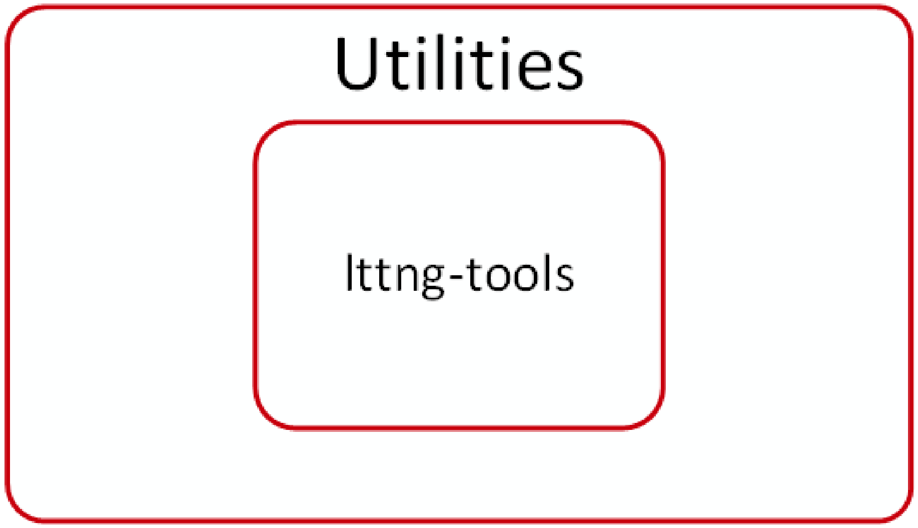
## Tracers

lttng-modules
Tracers
lttng-ust

- lttng-modules: kernel tracer module, compatible with kernels from 2.6.38* to 3.11,
- lttng-ust: user-space tracer, in-process library.

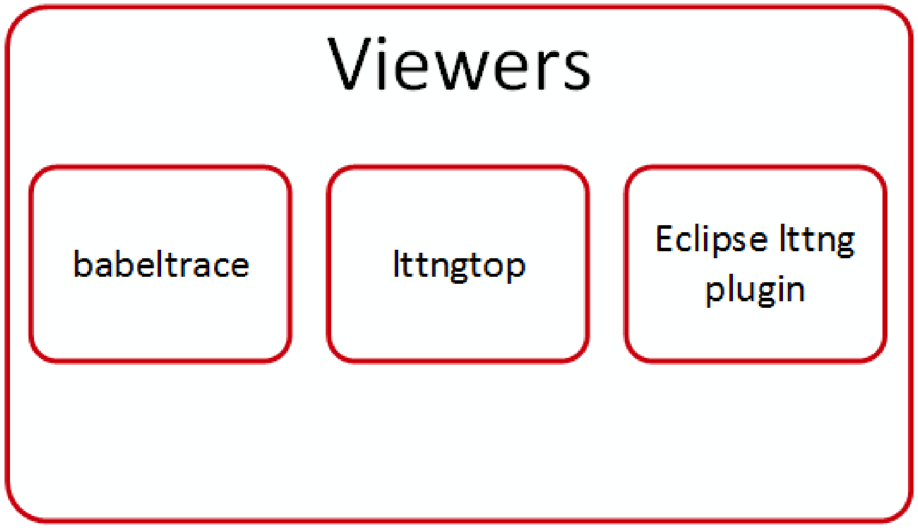
* Kernel tracing is now possible on 2.6.32 to 2.6.37 by backport of 3 Linux Kernel patches [1].

# Utilities



- lttng-tools: cli utilities and daemons for trace control,
  - lttng: cli utility for tracing control,
  - lttng-sessiond: tracing registry daemon,
  - lttng-consumerd: consume trace data,
  - lttng-relayd: network streaming daemon.

## Viewers



- babeltrace: cli text viewer, trace converter, plugin system,
- lttngtop: ncurse top-like viewer,
- Eclipse lttng plugin: front-end for lttng, collect, visualize and analyze traces, highly extensible.

## Hardware tracing support

- Initial attempt to support hardware tracing:
  - Babeltrace Nexus to CTF converter [5]
  - Goal: Leverage existing traces visualizer

- Encountered issues:
  - Traces are not self-contained, need sideband information
  - Internal trace buffer size limitation
  - Synchronization with other traces can be tricky

## Hardware tracing support demo

DEMO

## Future work

- Decoder/Converter for ARM ETM?
- Control hardware tracing facilities with the lttng-tools command-line?
- Custom views for hardware traces in Eclipse plugin

## Conclusion

- Availability and usefulness of hardware tracing
- Initial support for self-hosted hardware tracing
- Common abstraction for hardware tracing in the Linux kernel?

## Questions ?

?

EfficiOS

www.efficios.com

lttng.org

lttng-dev@lists.lttng.org

@lttng_project

## References

[1] - http://www.arm.com/images/processor/Cortex-A9-osprey.jpg

[2] - http://www.arm.com/images/CoreSight_Diagram_Tiny.jpg

[3] - https://lkml.org/lkml/2012/12/19/331

[4] - http://www.rlocman.ru/i/Image/2010/06/24/2.jpg

[5] - https://github.com/cbab/babeltrace/tree/nexus