Trace Debugging in Academics – the Future of Multicore Debugging?

Philipp Wagner

Institute for Integrated Systems
Prof. Dr. Andreas Herkersdorf

Arcisstraße 21
80333 München
Germany

http://www.lis.ei.tum.de

November 14, 2013

Well-known roads

http://en.wikipedia.org/wiki/File:Hume_Freeway_Craigieburn_Bypass.jpg
Wikipedia user Melburnian, CC-BY-SA

Road Signs: Where are we going?

http://www.flickr.com/photos/schoschie/1258647676
flickr user "schoschie", CC-BY

And a Glimpse Beyond?

http://www.flickr.com/photos/dougtone/6172050448
flickr user "dougtone", CC-BY-SA (edited)

The Road to Multicore Debug – Overview

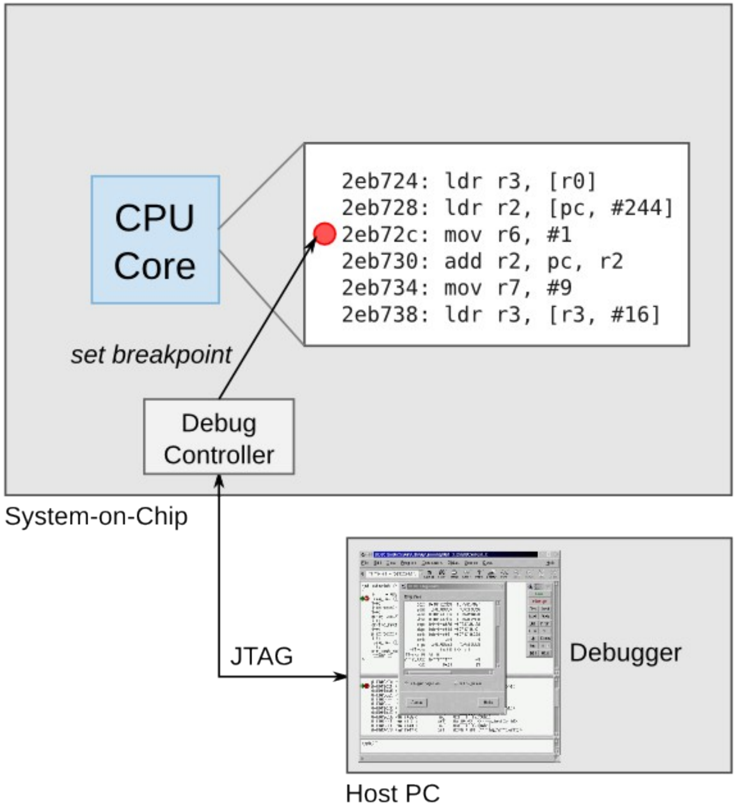
- How we're debugging now
- Tracing – the new way of Multicore Debugging
  - How we're doing it now
  - How we might do it in the future

Proven and Tested:

The way we debug today.

Run-Control Debug

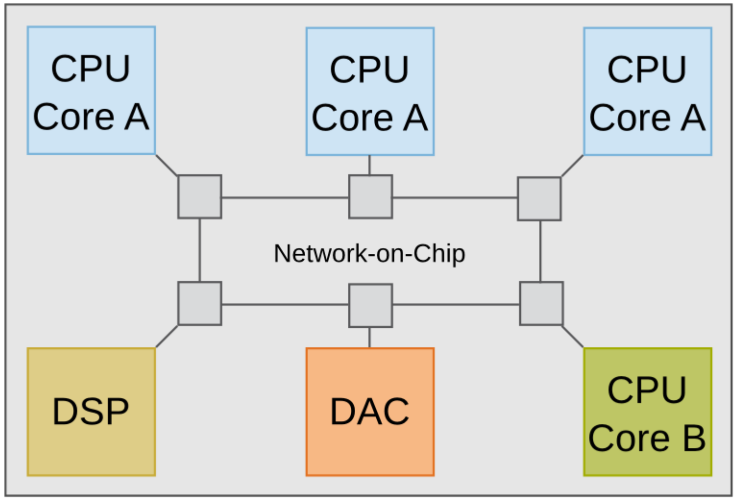
- It's easy
- It's well-known
- It works great
  - for programs that have no/little concurrent parts
- Today that's the majority of software!



```
2eb724: ldr r3, [r0]
2eb728: ldr r2, [pc, #244]
2eb72c: mov r6, #1
2eb730: add r2, pc, r2
2eb734: mov r7, #9
2eb738: ldr r3, [r3, #16]
```

Run-Control Debug

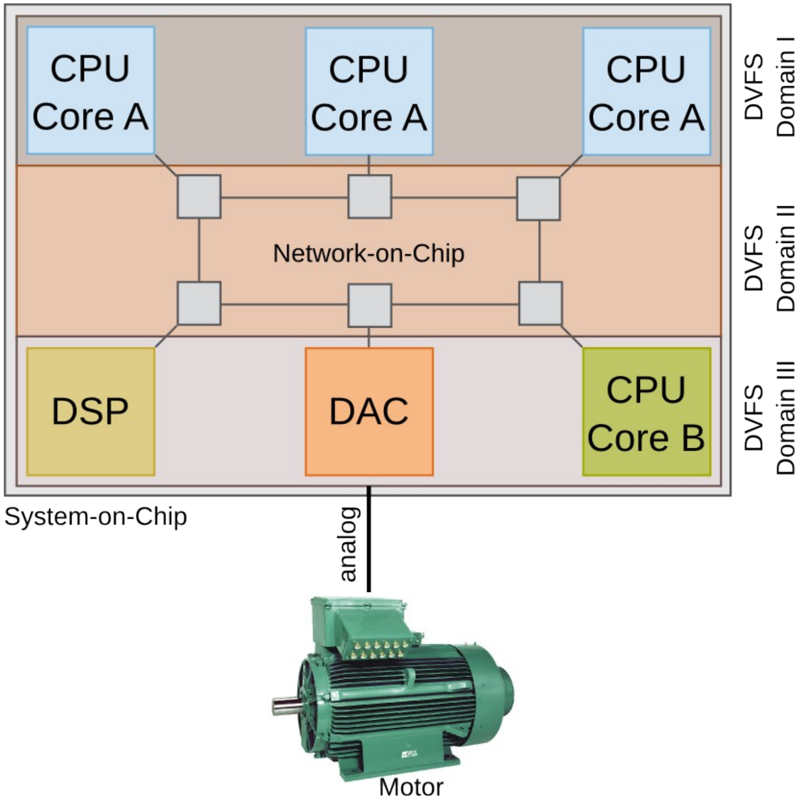
- What to do with
  - complex, heterogeneous systems?
  - concurrency problems?



System-on-Chip

Run-Control Debug

- What to do with
  - complex, heterogeneous systems?
  - concurrency problems?
- Different clock domains
- Real-time?
- Unattended debug?



Wind turbine:
http://images.cdn.fotopedia.com/flickr-185488411-original.jpg
User “phault” on Flickr, CC-BY

A Road Sign: Tracing

- Can find concurrency bugs
- Can work in heterogeneous environments
- Can be non-intrusive
- Can be unattended

http://www.flickr.com/photos/countylemonade/5321535677
User “Garrett” on flickr, CC-BY (edited)

Tracing can solve the problems of run-control debugging!

Road Bumps

Data Trace for a DDR3 Memory [1]

DDR3-1333 Peak Data Rate: ~ 10 GByte/s

Compression for an average memory trace:
~ 50 % reduction

NEXUS 5001 Trace Port [2]

Freescale MPC5777M

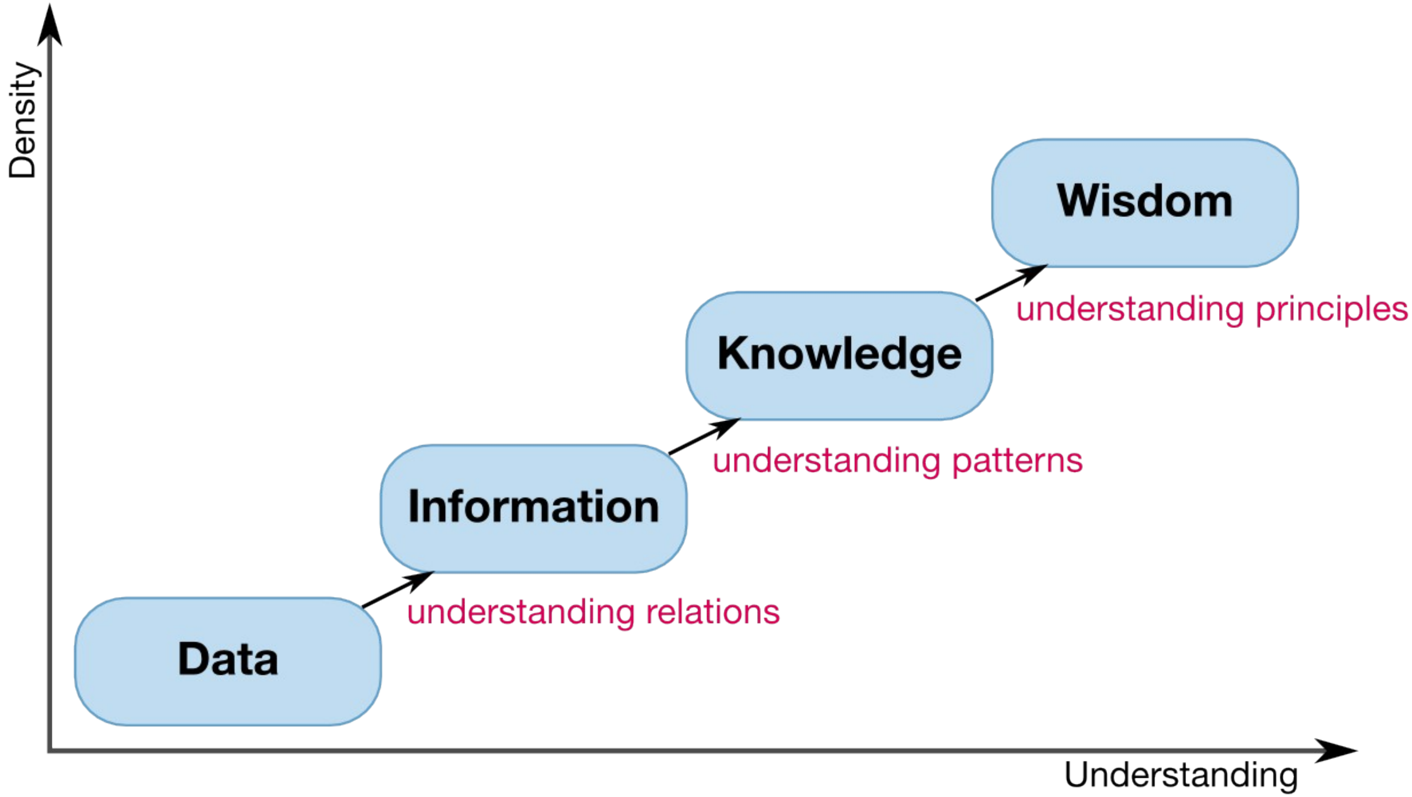
4 lanes of 1 GBit/s Xilinx Aurora trace port

40 GBit/s data generation ↔ **4 GBit/s off-chip**

[1] Data compression rate: Christian Morgenstern, “Collection and Compression of Memory Traces for Manycore System-On-Chip”, Bachelor Thesis, 2013

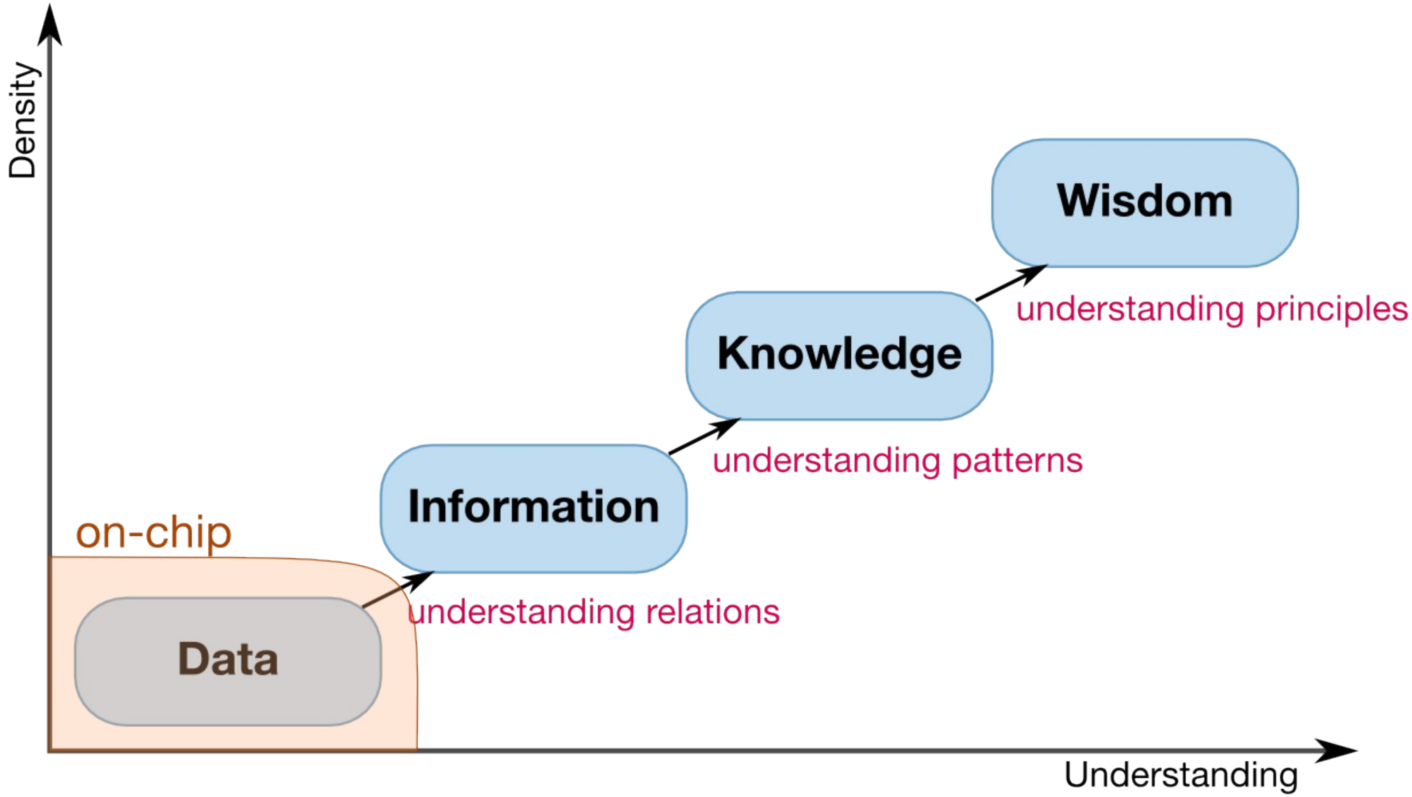
[2] Only on the emulation device, 4 lanes w/ 1.25 GBit/s data rate and 8b10b encoding. Ignoring all NEXUS 5001 message overhead.

Think! On Chip.

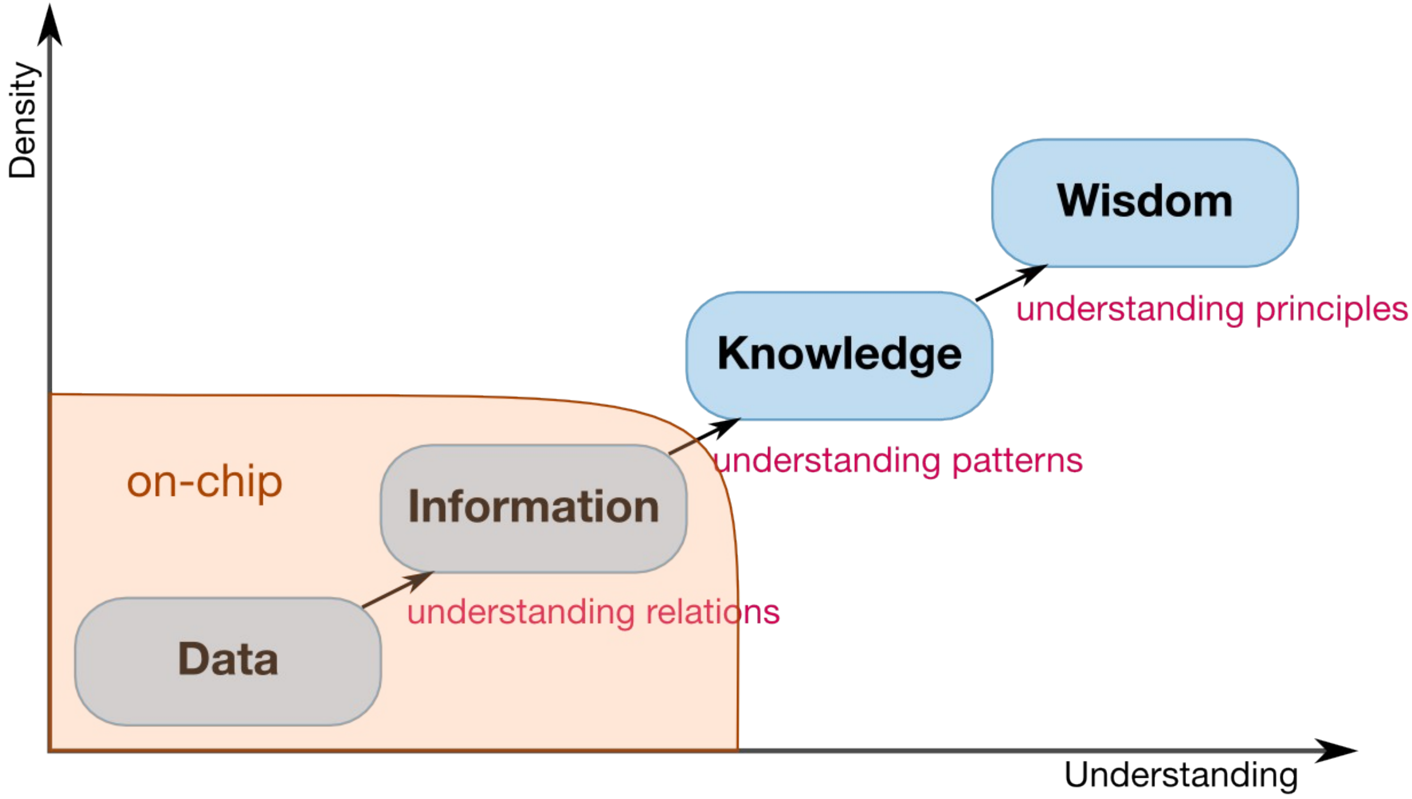


DIKW Chain (Ackoff, 1989). This representation is inspired by Gene Bellinger, “Knowledge Management—Emerging Perspectives”. http://www.systems-thinking.org/kmgmt/kmgmt.htm. 2004

Think! On Chip.



Think! On Chip.



Attach meaning to data on chip
Reduce required off-chip transfer bandwidth

From Data to Information: First Road Signs

- Information from the software developer
  - ARM CoreSight System Trace Macrocell (STM), NEXUS 5001-2008 Data Acquisition Messaging (DQM)
  - like printf()/kprintf() with hardware support
- Configurable trace collection
  - Infineon MCDS
- Recognizing known bugs
  - On-chip Data Race Detection [Wen et. al, 2012]

C.-N. Wen, S.-H. Chou, C.-C. Chen, and T.-F. Chen, "NUDA: A Non-Uniform Debugging Architecture and Nonintrusive Race Detection for Many-Core Systems," *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 199 – 212, Feb. 2012.

From Data to Information: Take it a Step Further

1. Give the developer a way to express the meaning of the data
2. Add on-chip hardware to actually extract information out of data

Express the Meaning of Data

```
# increment a drop counter for every
# location we drop a packet at
probe kernel.trace("kfree_skb") {
  locations[$location] <<< 1
}

# Every 5 seconds report our drop locations
probe timer.sec(5) {
  printf("\n")
  foreach (l in locations-) {
    printf("%d packets dropped at %s\n",
           @count(locations[l]), symname(l))
  }
  delete locations
}
```

Data point: Linux kernel function `kfree_skb()` is called

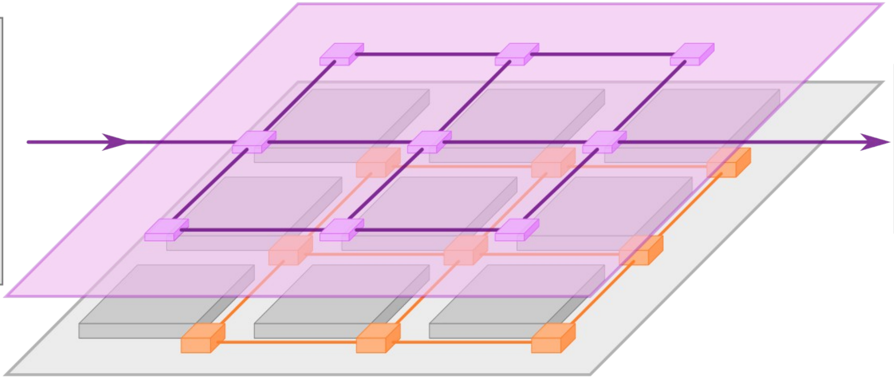
Information: dropped TCP packets

```
#> stap dropwatch.stp
3 packets dropped at 0xffffffff81495cfb

9 packets dropped at 0xffffffff81495cfb
1 packets dropped at 0xffffffff8154da4c

4 packets dropped at 0xffffffff81495cfb
2 packets dropped at 0xffffffff814f2100
```

SystemTap example from https://sourceware.org/systemtap/SystemTap_Beginners_Guide/useful-systemtap-scripts.html

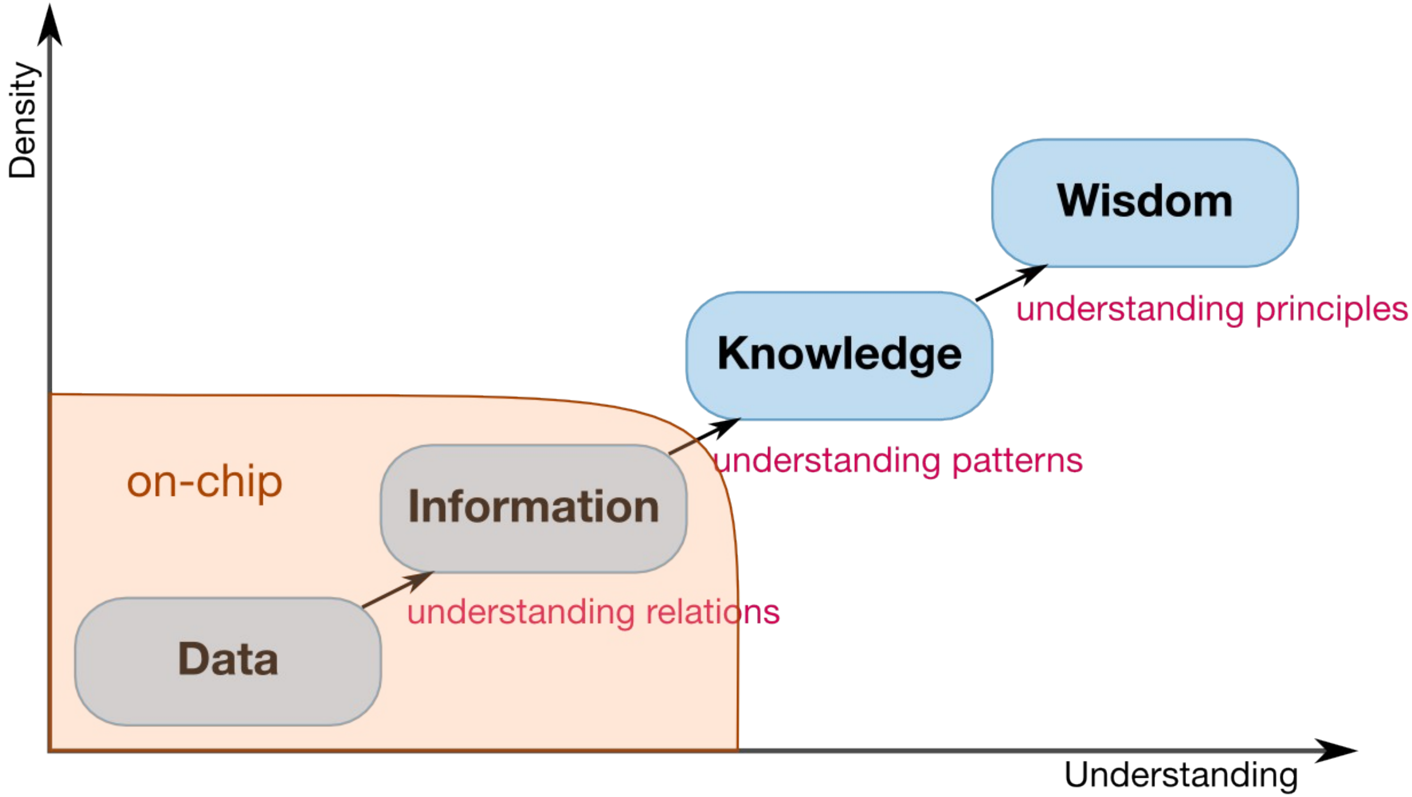
Now put it in hardware

```
# increment a drop counter for every
# location we drop a packet at
probe kernel.trace("kfree_skb") {
  locations[$location] <<< 1
}

# Every 5 seconds report our drop locations
probe timer.sec(5) {
  printf("\n")
  foreach (l in locations-) {
    printf("%d packets dropped at %s\n",
           @count(locations[l]), symname(l))
  }  delete locations}
```

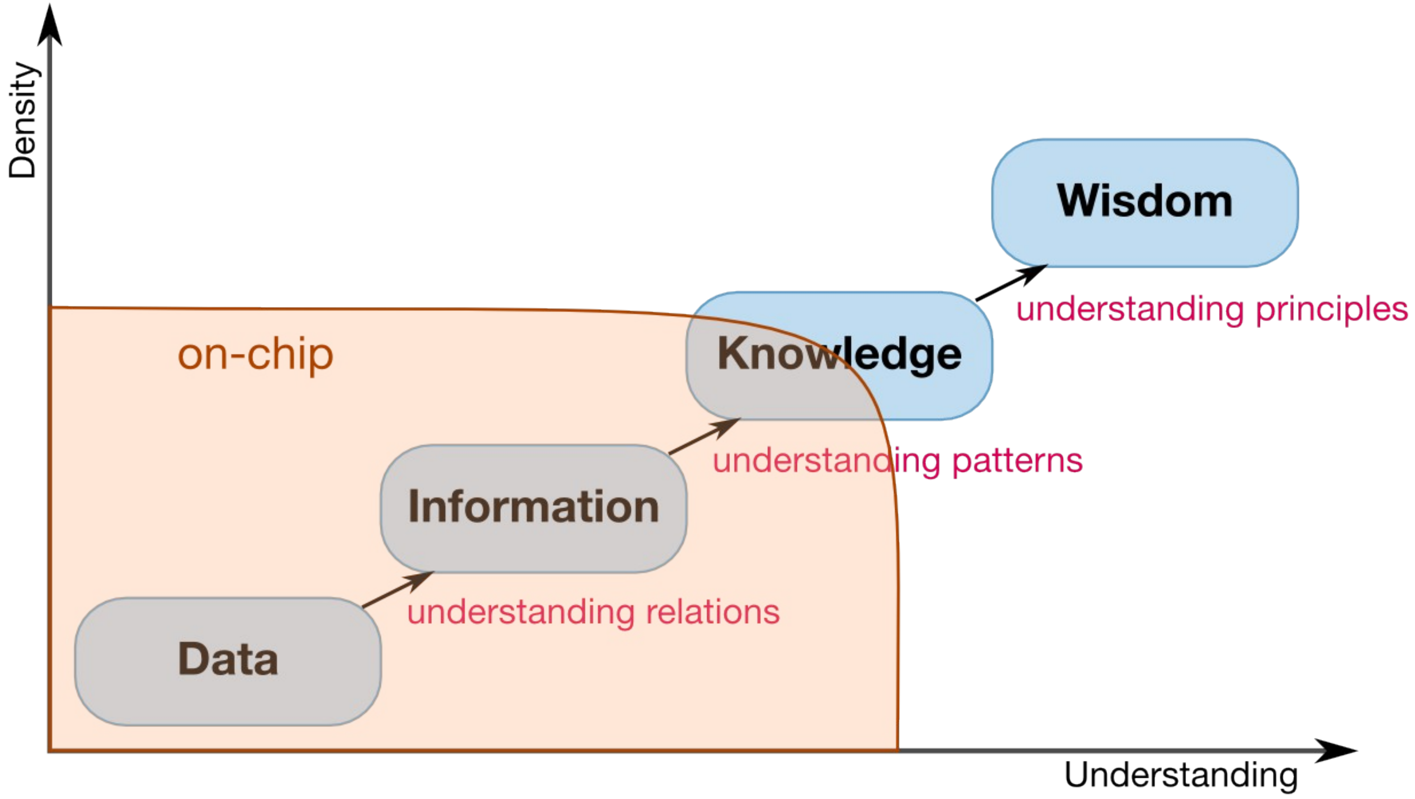


```
#> stap dropwatch.stp
3 packets dropped at 0xffffffff81495cfb

9 packets dropped at 0xffffffff81495cfb
1 packets dropped at 0xffffffff8154da4c

4 packets dropped at 0xffffffff81495cfb
2 packets dropped at 0xffffffff814f2100
```

Think! On Chip.

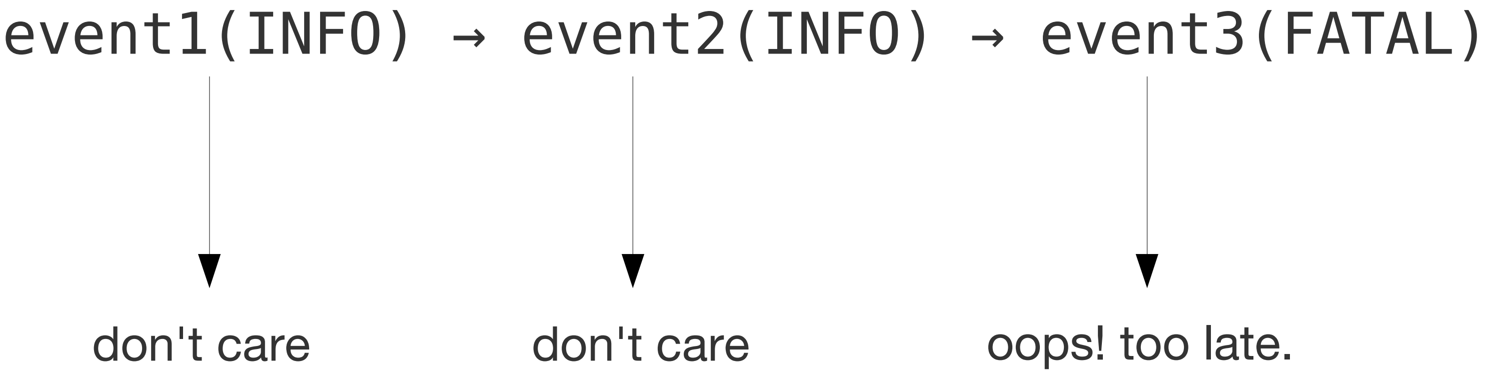


Think! On Chip.



Find patterns.
From information to knowledge?

From Information to Knowledge



`event1(INFO) → event2(INFO) → event3(FATAL)`

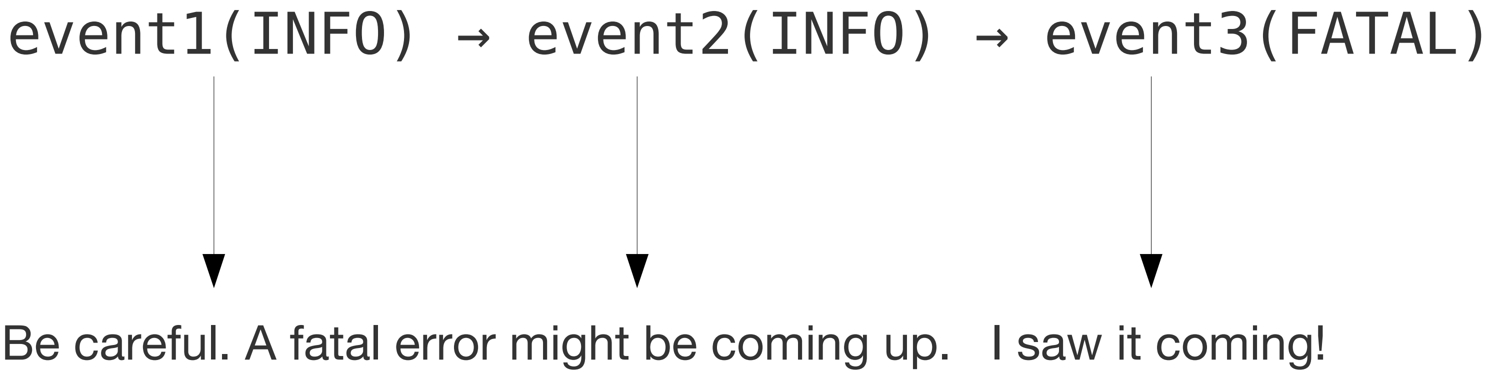
don't care

don't care

oops! too late.

this sequence repeats for a couple times

From Information to Knowledge



```
event1(INFO) → event2(INFO) → event3(FATAL)
```

Be careful. A fatal error might be coming up. I saw it coming!

Pawlow's Dog: Reinforcement Learning

http://www.flickr.com/photos/leartoon/3408952805
Flickr user "Nikola B", CC-BY

Can we bring machine learning into the chip?

Summary: The Road to Multicore Debug

- Less run-control debug, less full system tracing in the future
- Put debug intelligence inside the chip!
- Give meaning to data: “Debug Coprocessors”
- Find recurring patterns: Machine learning

Thank you for your attention.

Your Thoughts?

Philipp Wagner
Institute for Integrated Systems, TUM
philipp.wagner@tum.de