

Software-Defined Vehicles and Real-Time Systems – like Oil and Water?

And is there any Dish Soap?

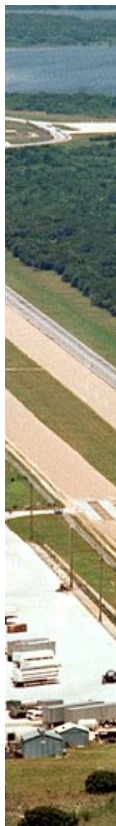
Short Inspirational Talk

Topics:

- The evil of complexity
- Throughput and latency
- Bandwidth (feed the compute monster)
- From classic ECUs to SDV
- Real-time with distributed software components
- An overview of timing analysis in SDV environments

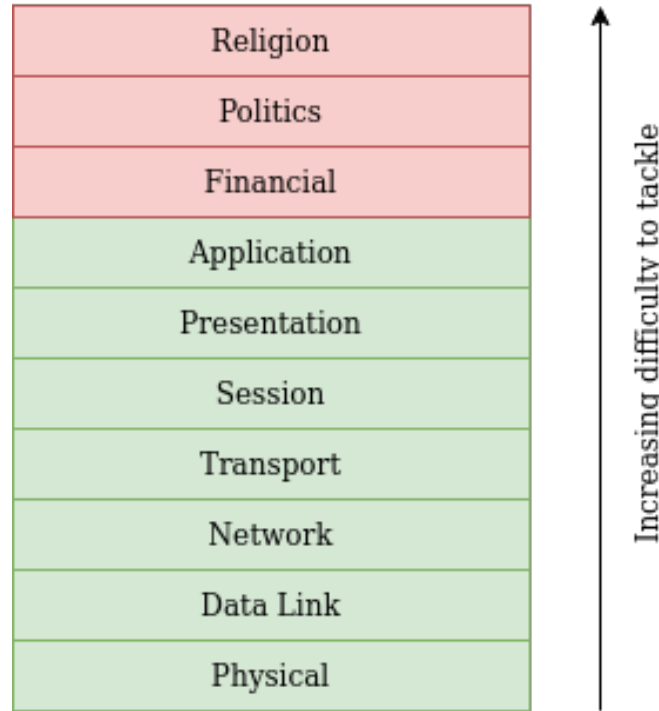


The Evil of Complexity





Complexity and OSI Layers 8 and 9



Source: Sven Vermeulen , <https://blog.siphos.be/2021/06/the-three-additional-layers-in-the-OSI-model/>

“The bitterness of poor quality remains long after the sweetness of low price is forgotten”

Benjamin Franklin

Why is Complexity Bad Today?

- Humans can only handle 5-7 items consciously at once
 - Shiffrin, Richard; Robert Nosofsky (April 1994). "Seven plus or minus two: A commentary on capacity limitations". Psychological Review.
- How does this relate so Software?
 - "There are no complex systems that are secure. Complexity is the worst enemy of security, and it almost always comes in the form a features or options."
 - "Complexity is a measure of **how many things interact at any one point.**"
 - See
Ferguson/Schneier: Practical Cryptography, Wiley 2003 /
Ferguson, Schneier, Kohno: Cryptography Engineering, Wiley 2010

How to (not) Handle Complexity?

- KISS – Keep it simple, stupid
 - The world needs more KISSing!
- Kahneman's System 1 (fast)
 - Low effort
 - Works until it doesn't, see Space Shuttle Challenger and Intercity-Express 884 Wilhelm Conrad Röntgen
- Kahneman's System 2 (slow)
 - High effort
 - Costly
 - Gets the job done right, for a price
- Break problem down into small parts (divide and conquer)
 - Set up clear and simple interfaces



Throughput vs. Latency:
Have your Cake **or** eat it

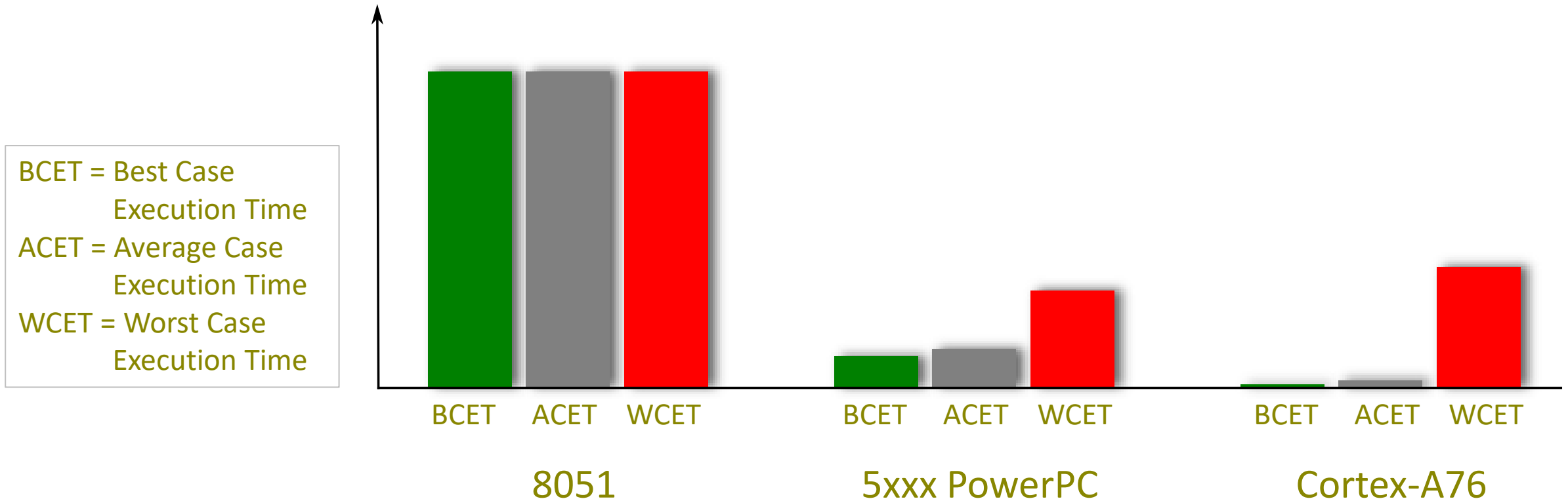
FTL: Faster Takes Longer

- Signal propagation delay in silicon exists
 - Only so many logic gates can be reached in n nanoseconds
 - Slice the task into consecutive steps
 - Pipeline them
 - Yeah! High clock rates
 - But: many steps for each single unit of work
 - Intel Pentium 4: bad performance for code with branches
- Product segmentation by pipeline length, example ARM:

- **A**pplication
- **R**ealtime
- **M**icrocontroller



Execution Times and how They Developed



Technical features for more throughput: a) higher clock rates b) pipelines c) caches d) etc.

The image features a white brick wall as the background. On the left side, there are four white, 3D L-shaped blocks arranged in a staggered pattern. The text "How to Feed the Compute Monster?" is positioned on the right side of the wall.

How to Feed the
Compute Monster?

The Powerful Next Project Hardware

Multi-GHz
Multi-Core
SIMD
DSP
HW Crypto
1000BASE-T



Man, that memory interface is expensive! Let's reduce the pin count, cost control ftw!



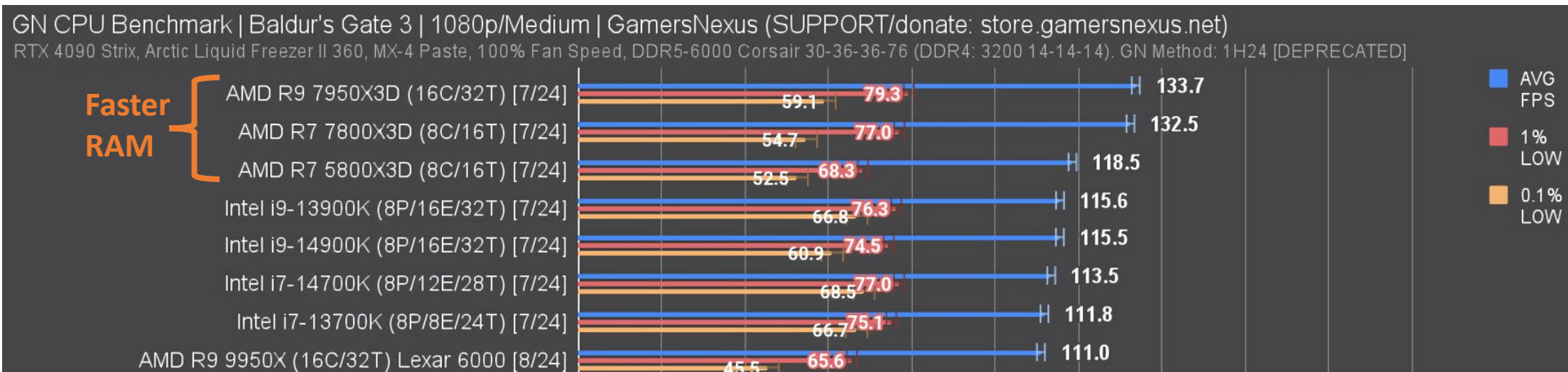
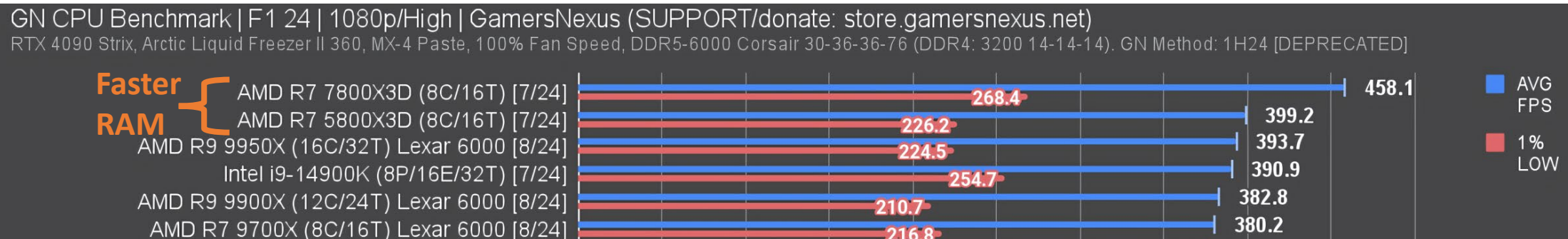
Why does this happen to my project?



A's in Math, C's in Reading

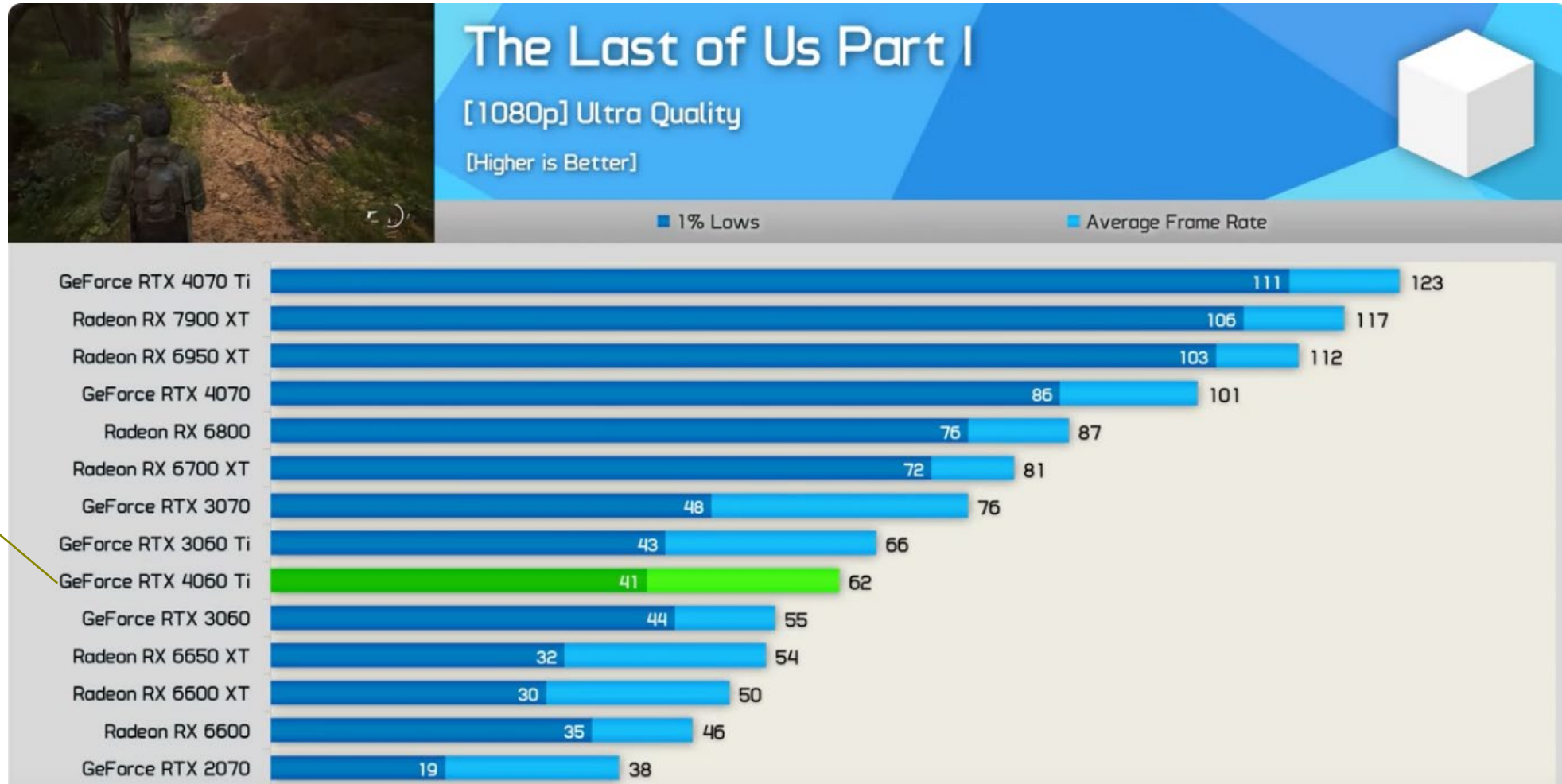
- Early 1980s: CPUs $<10\text{MHz}$, RAM $\geq 10\text{MHz}$
- Early/Mid 1990s: 486DX2
- 2000s: Breaking the CPU GHz barrier, going multi-core
- 2009: AES implementation by Emilia Käsper is faster in calculating S-BOX content on the fly than loading it from cache
- Today: CPUs and GPUs wait for memory to catch up
 - ADAS on the parking lot

Today: CPUs



Source: Gamers Nexus, <https://gamersnexus.net/megacharts/cpus>

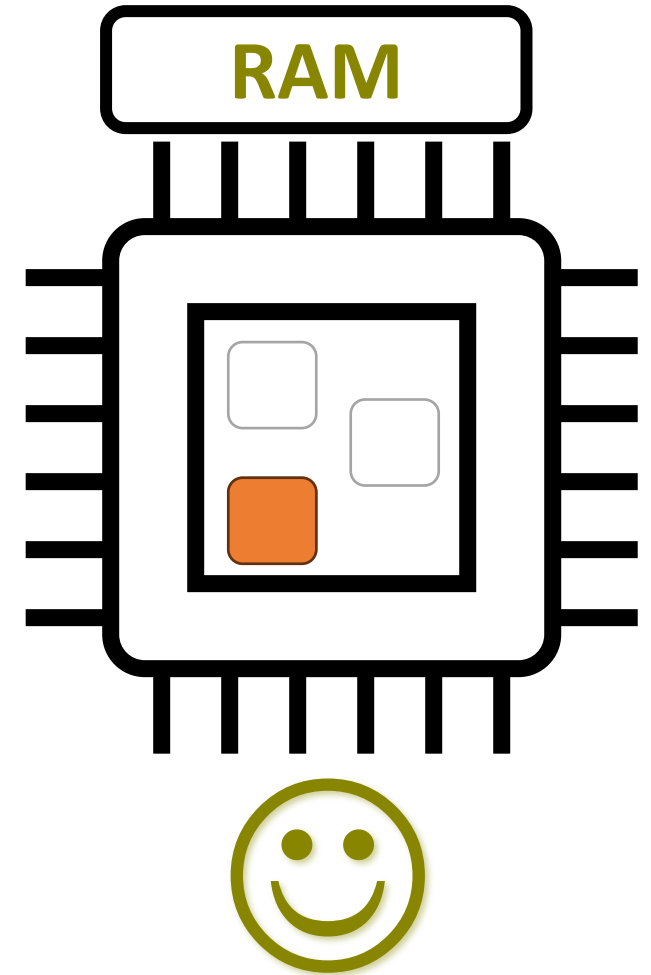
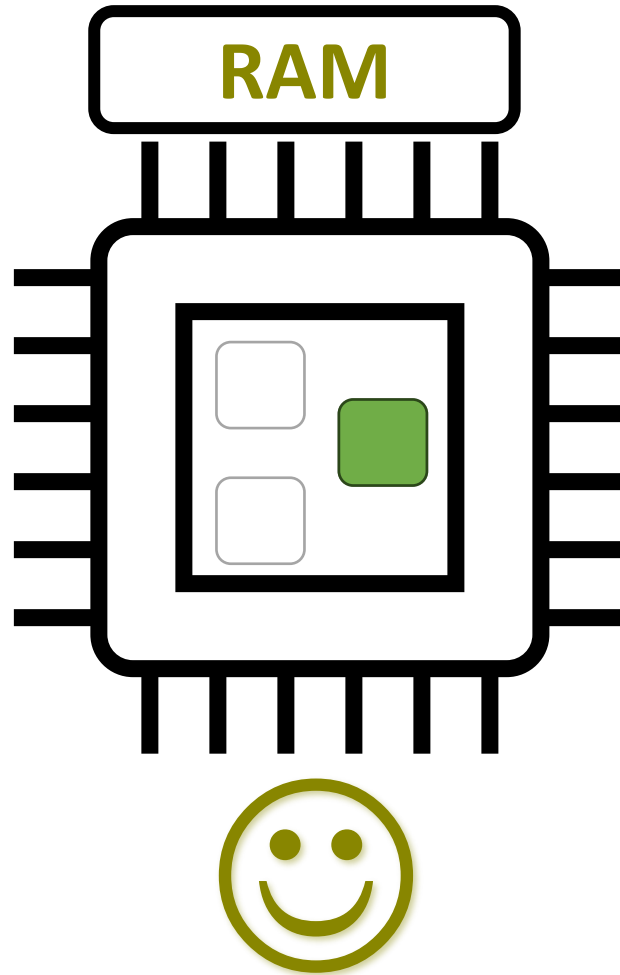
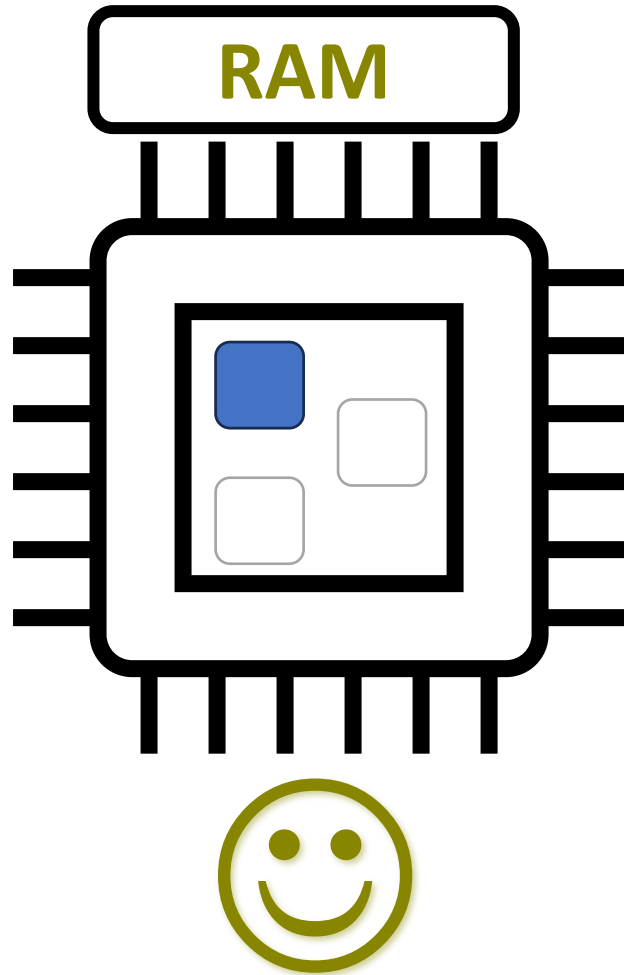
Today: GPUs



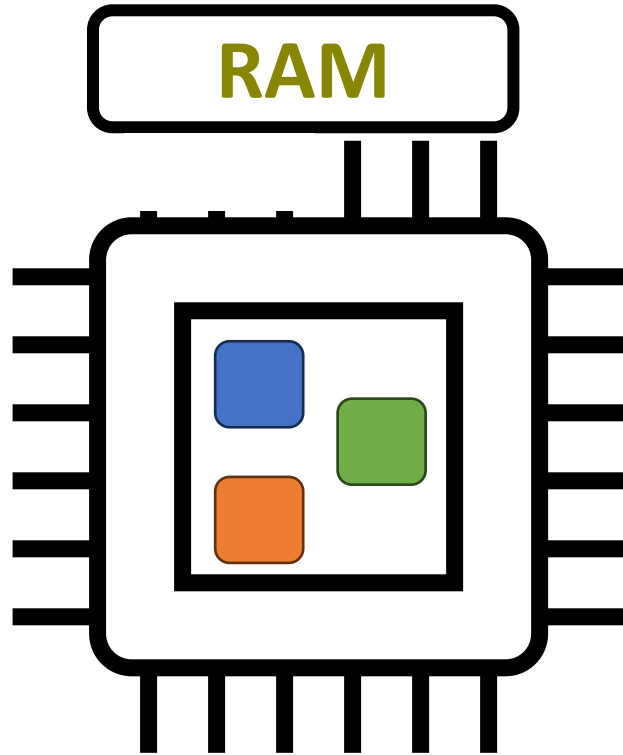
Losing against
previous model
due to slower
RAM access

Source: Hardware Unboxed, <https://youtu.be/WLk8xzePDg8?t=699>

ADAS in the Office, Divided and Conquered



ADAS in the Parking Lot



Religion
Politics
Financial
Application
Presentation
Session
Transport
Network
Data Link
Physical

Increasing difficulty to tackle

“Complexity is a measure of **how many things interact at any one point.**” (Niels Ferguson and Bruce Schneier)

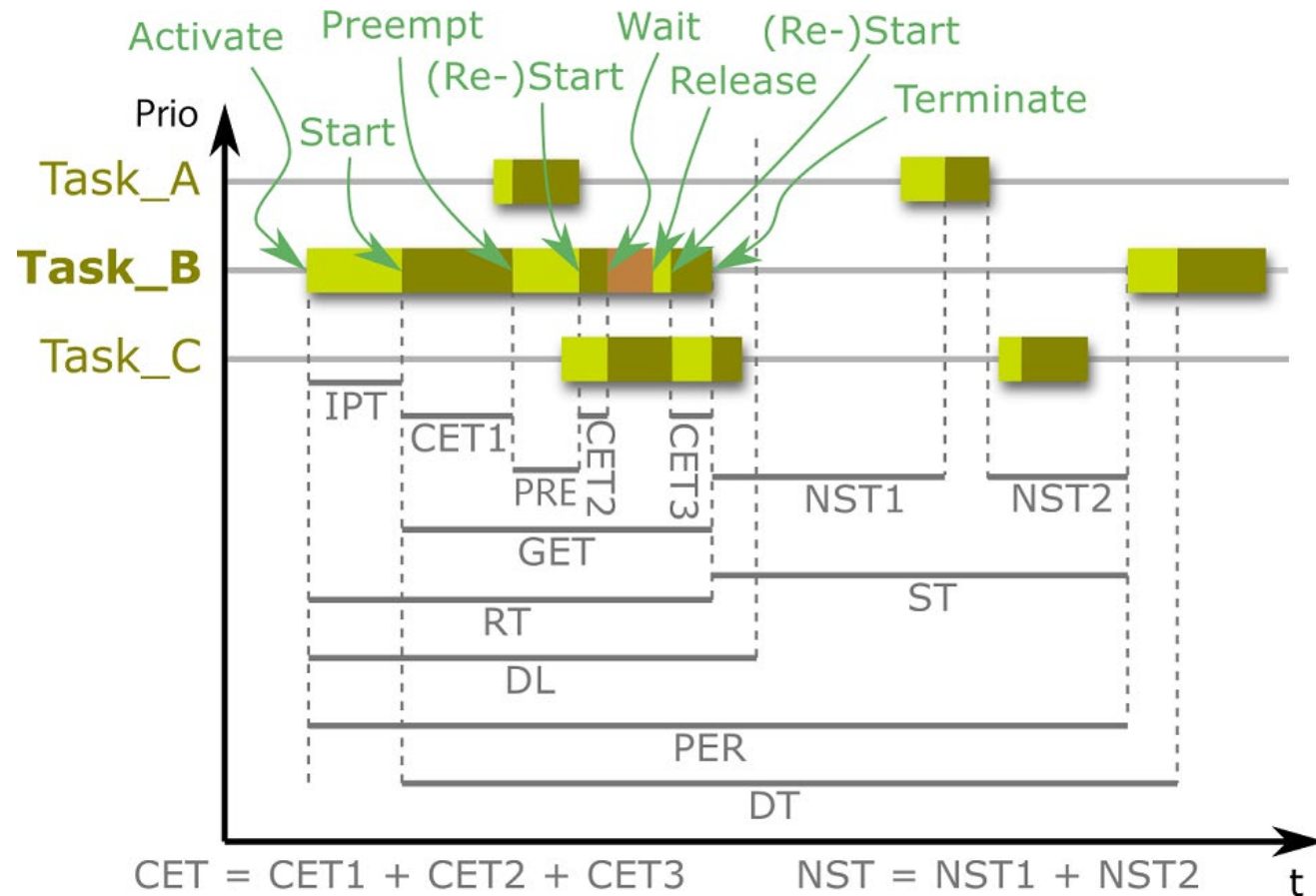
What to do?

- Promote data locality
- Promote cache friendliness
- Specify and monitor memory bandwidth requirements for software
- Top tier: “memory Tetris”
 - We did not understand the customer’s math PhD’s algorithms
 - We did not have to
 - We DMAed that data into local on-chip buffers when and where it was required and solved the RAM controller bottleneck
 - The ADAS got off the parking lot and into mass production
- Always, always monitor timing changes when adding new software

The background of the slide is a light-colored brick wall. On the left side, there are several white, three-dimensional rectangular blocks of varying sizes and orientations, some standing upright and others leaning against each other. The text is positioned on the right side of the slide, overlaid on the brick wall background.

Timing Requirements: from classic ECU to SDV

ECU: Timing Parameters



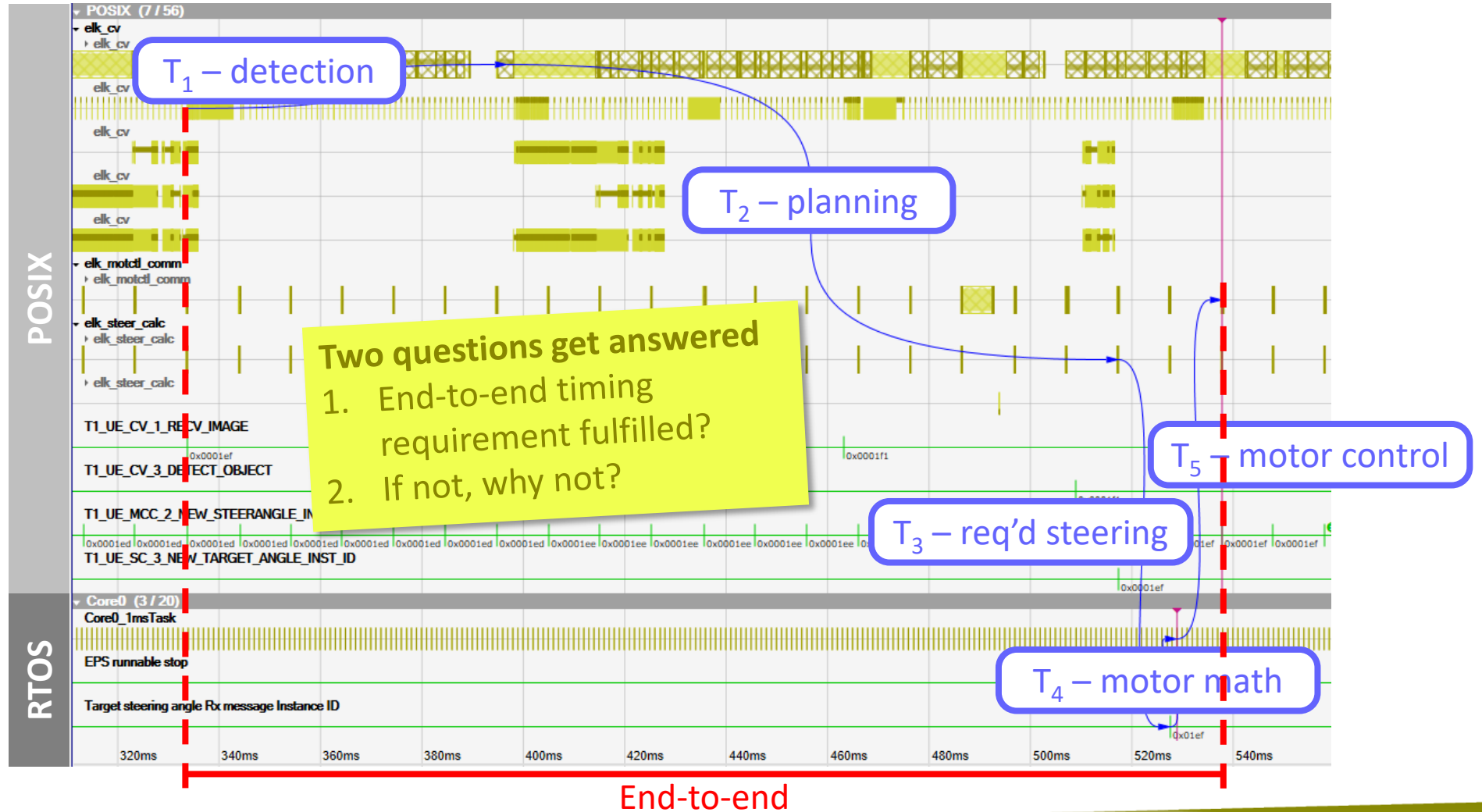
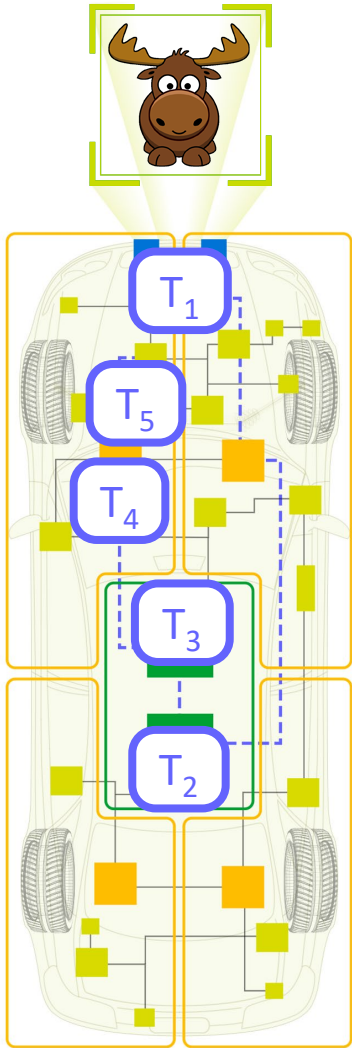
- **IPT (Initial Pending Time)**
Ready time before task starts
- **CET (Core Execution Time)**
Time spent in running state, i.e. executing
- **GET (Gross Execution Time)**
From start to termination (cf. pin toggle)
- **PRE (PREemption Time)**
Sum of ready times without IPT
- **RT (Response Time)**
cf. schedulability analysis; **DL (DeadLine)** = limit for RT
- **Period (PERiod)**
time difference between two subsequent activations
- **DT (Delta Time)**
time difference between two subsequent events of the same type; observed period; cf. jitter
- **ST (Slack Time)**
duration of the 'gap'
- **NST (Net Slack Time)**
headroom: time which could be added to CET

ECU->SDV

From Tasks and ISRs to Event Chains



Tracing/verification of event chains



The background is a white brick wall with a horizontal pattern. On the left side, there are four white 3D rectangular blocks of varying sizes and orientations, some standing upright and some lying flat, creating a modern architectural feel.

Best practices for distributed software components

Event Chain how-to, step by step

- Identify event chains
- Specify them, including end-to-end timing requirement
- system architecture & software architecture
- Decompose event chains (e.g. $T_1 + T_2 + T_3 + T_4 + T_5$)
- Verify event chains
- Supervise critical event chains in the final product

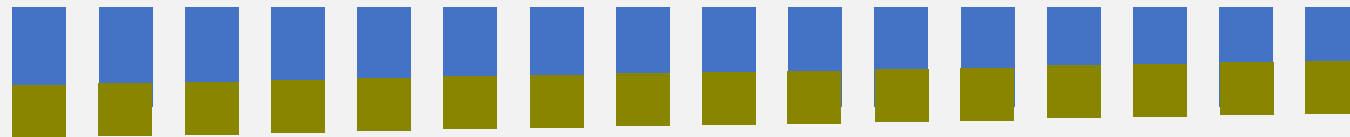
Start Early, Automate

Functional tests

(Timing tests)



Functional tests
including Timing tests



**Capture CPU load, execution times, jitter,
etc. 'on-the-fly' along with functional tests!**


Summary, advice w.r.t. timing

Don'ts

- Don't let timing just happen, be aware of it!
- Don't let task forces control your timing methodology.
- Don't burn your best experts up in timing debugging
- **If you fail to plan you plan to fail!**

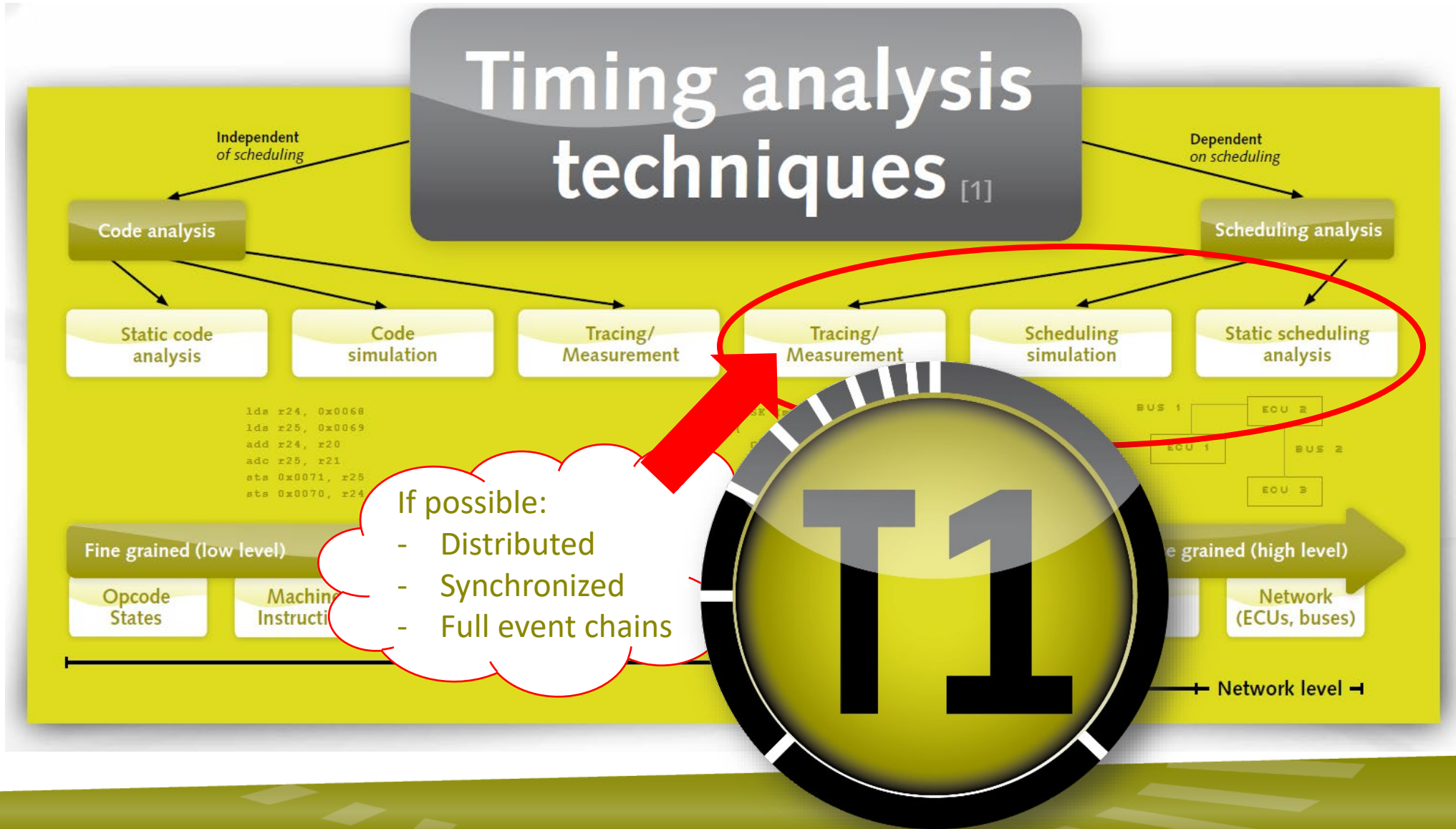
Dos

- Consider timing in all development steps.
- Every project must have a timing expert assigned.
- Establish **automated** timing tests (cf. functional tests)
- Get the timing infrastructure right in the **platform**.

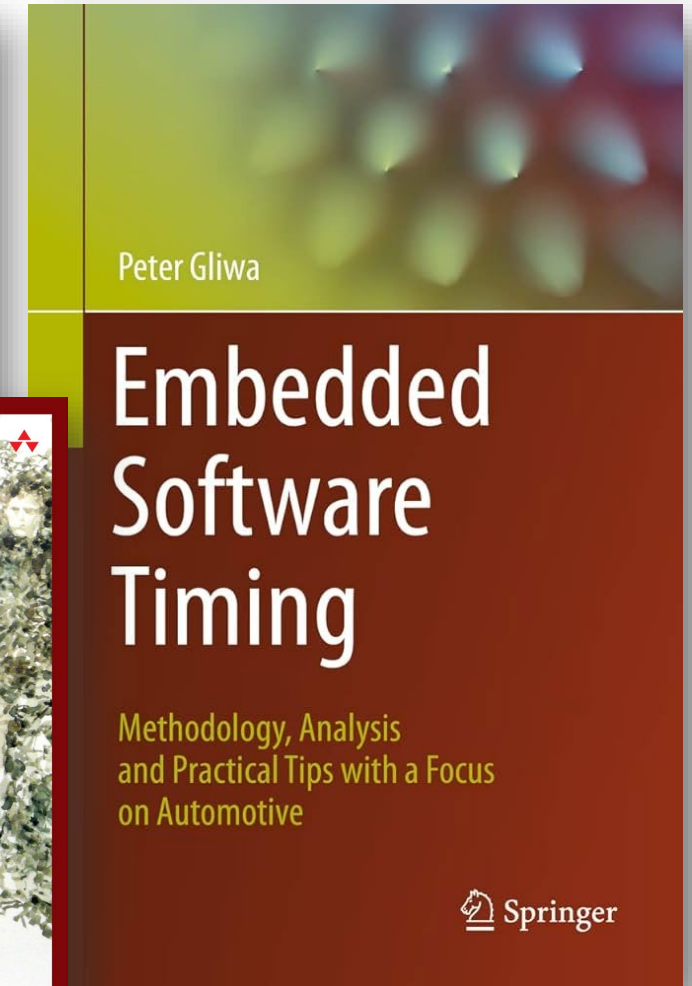
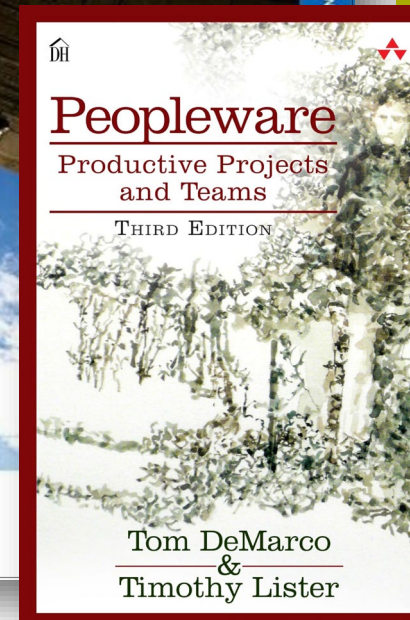
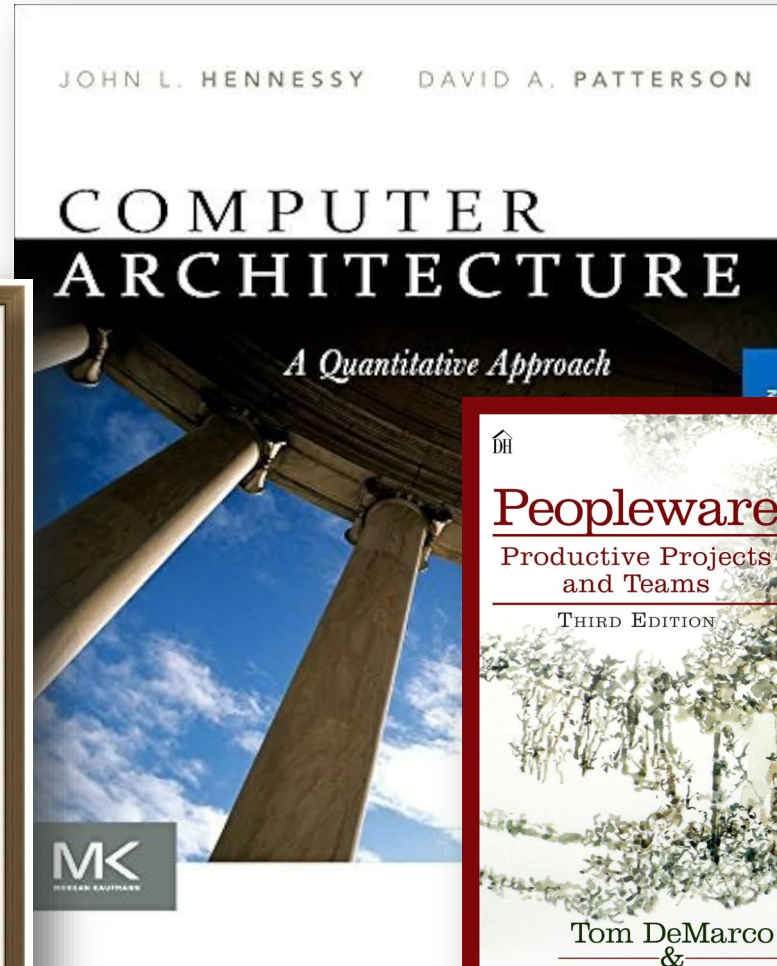
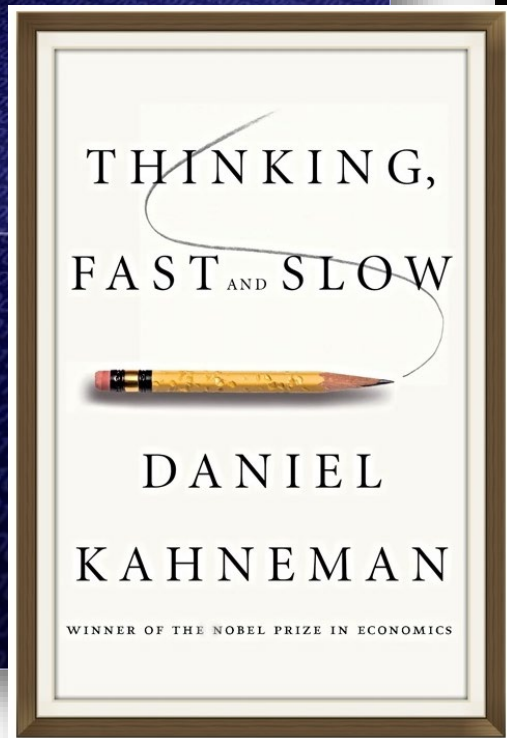
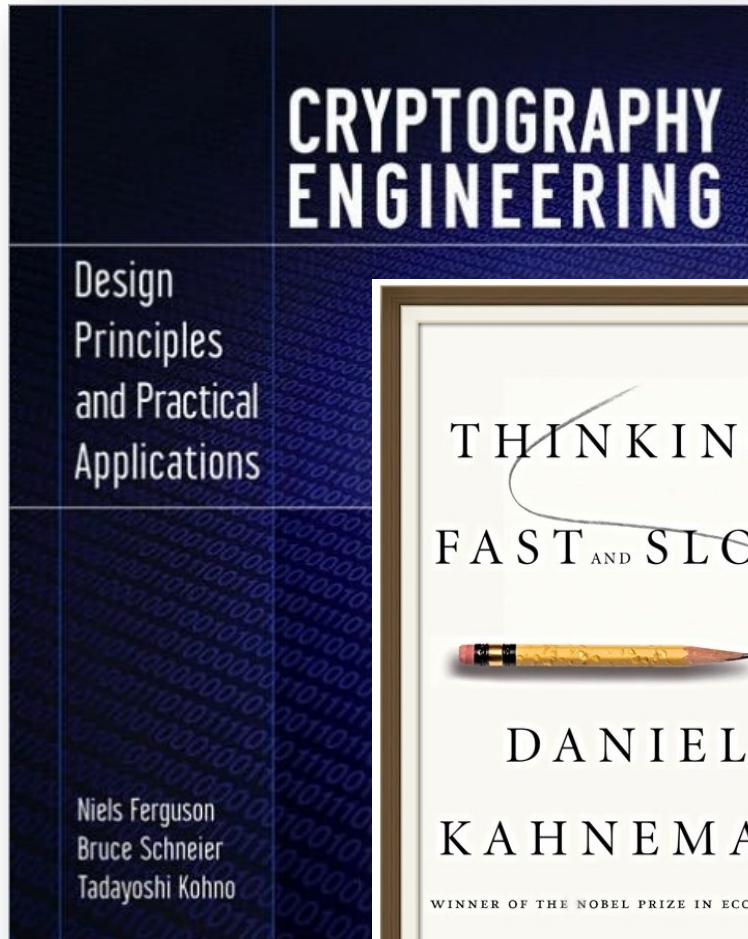
The background of the slide features a light-colored brick wall. On the left side, there are several white, three-dimensional rectangular blocks of varying sizes and orientations, some standing upright and others leaning against each other or the wall. The lighting is soft, creating subtle shadows on the wall and floor.

Timing Analysis Tools and Techniques Applicable to SDV Environments

Overview of timing analysis techniques



The Wisdom of the (not quite) Ancients





M.Sc., Dipl.-Ing.(BA)

Christian Wenzel-Benner

Director Training & Coaching

GLIWA GmbH & Co. KG

phone +49-881-13 85 22-82

email christian.wenzel-benner@gliwa.com

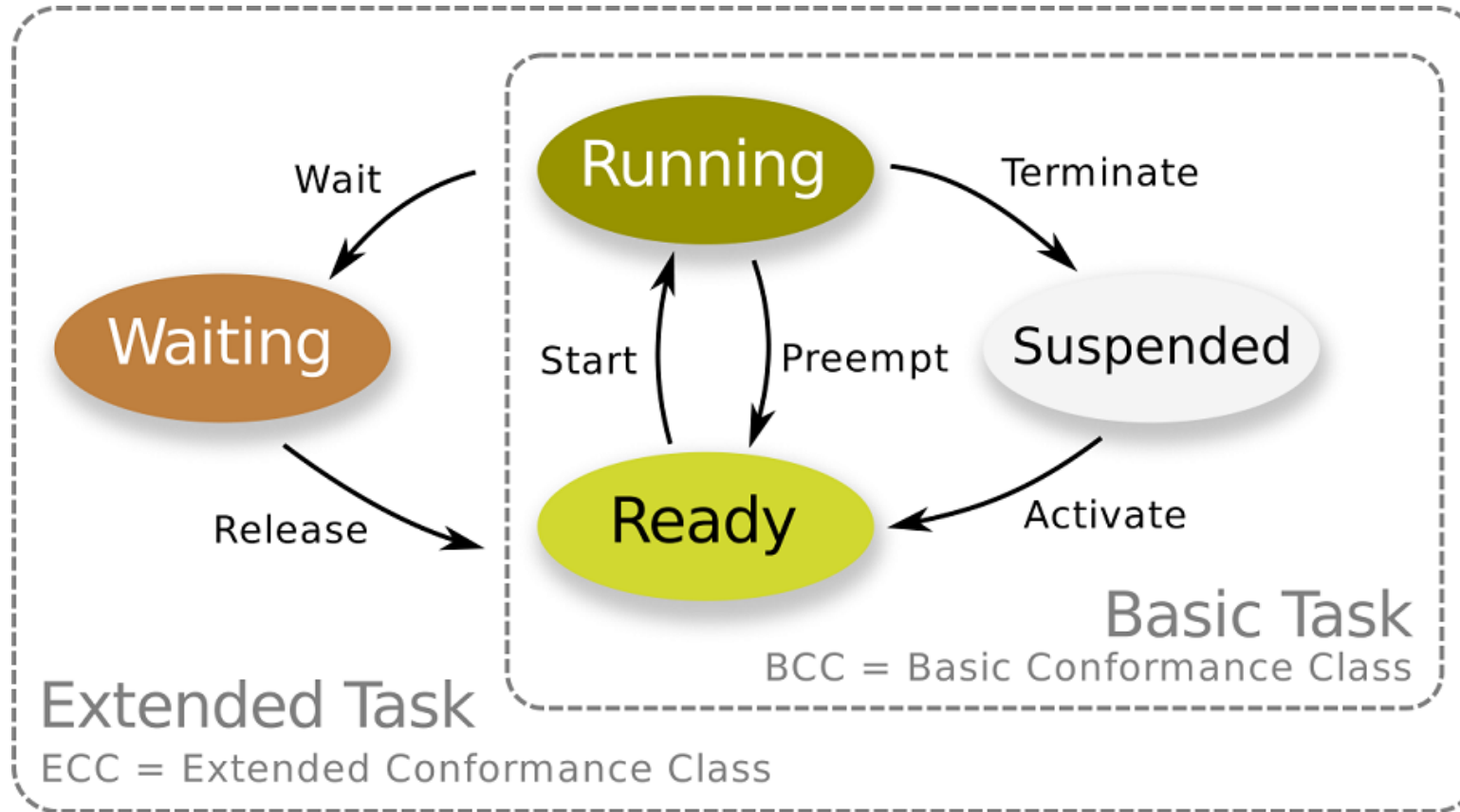
web gliwa.com

Thank you!

Timing dependencies, timing impact



ECU: AUTOSAR / OSEK task states



Must everyone become a timing expert?

- Not everyone needs to be a timing expert!
- My recommendation: **T-shaped** approach
- **Depth 1**
 - Awareness that timing is important
 - Know who the timing expert is
- **Depth 2**
 - Decent timing expertise
 - Timing tool knowledge
 - Timing methodology knowledge
 - Connection to super experts for the heavy lifting

