EVIDENCE®

EMBEDDING TECHNOLOGY

Open-source and Real-time in Automotive Systems: (not only) Linux, (not only) AUTOSAR

Paolo Gai, Evidence Srl
pj@evidence.eu.com

agenda

- something about Evidence
- something about ERIKA
- future plans

That is… something about the business of maintaining, adapting and selling a (non-Linux) open-source real-time operating system!

… plus some questions for the community!!!

something about Evidence

…just a quick introduction, don’t worry!

the company

Founded in 2002 as spin-off company of the
Real-Time Systems Lab at Scuola Superiore S.Anna

- ~20 qualified people with an average age of 34 years
- 10+ years of experience in academic and industrial projects
  - One third of the company has a PhD degree

Our Mission:
design and development of software for small electronic devices

(some) customers and partners

OSEK, microcontrollers, schedulability analysis, code generation

Linux, SW devel.

Partnerships

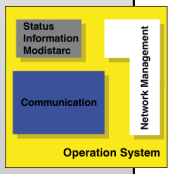
products and services

RTOS and MCU skills

- OSEK/VDX, AUTOSAR,
- Automatic code generation

ERIKA ENTERPRISE

E4 CODER



Embedded Linux skills

- 8 Yrs experience in custom BSPs, U-Boot, kernel drivers,
- Initial developers of the SCHED_DEADLINE patch

EVELIN

Application Development

something about ERIKA

current features… and a bit of its history!

Something about ERIKA Enterprise

http://erika.tuxfamily.org

- ERIKA Enterprise is an RTOS OSEK/VDX certified
- ERIKA Enterprise implements an API inspired to a subset of the AUTOSAR API
- With a suitable open-source license allowing static linking of closed source code
- Typical footprint around 2-4KB Flash

OSEK/VDX for dummies

OSEK/VDX for ~~dummies~~ for a Linux hacker

Let's compare OSEK/VDX with a typical Linux setup

What	Linux	ERIKA - OSEK/VDX
Birth date	90s, UNIX in early 70s	OSEK: 90s ERIKA: 2002
Target	General purpose OS	Automotive minimal OS
Initial Target HW	PC	8 bit microcontroller
Development method	Open-source community	Automotive companies
Goal	Create a great free OS?	Cost reduction
API	POSIX / pthreads 300 (?) functions	OSEK/VDX, 35 functions http://www.osek-vdx.org
IRQ response time	Well…big ☹	3-10 usec on a small micro

OSEK/VDX for a Linux hacker (2)

What	Linux	ERIKA - OSEK/VDX
Flash Footprint	4-32 MB Flash for a minimal system	2-4 KB Flash
RAM	8-64 MB	Hundreds of Bytes
Static/Dynamic approach	Dynamic	Static, configured with an OIL file or AUTOSAR XML
CPU support	32-64 bit	Down to 8 bit MCUs
Filesystem	Yes	No
MMU support	Yes	No (Yes for AUTOSAR)
Device Drivers	Yes	No (Yes for AUTOSAR, but configured «more statically»)
Execution from flash	No	Yes

OSEK/VDX for a Linux hacker (3)

What	Linux	ERIKA - OSEK/VDX
Certification suite	No	Yes
Real-time support	Available through patches (RT-PREEMPT, RTAI, Xenomai, SCHED_DEADLINE…)	Native support for Fixed priority, Preemptive and non preemptive execution
Stack sharing	No	Yes
Immediate Priority Ceiling	Yes (with realtime priorities)	Native
Multicore support	Yes, SMP	Yes, Static Partitioning
IRQ handling	in the kernel	in the Application, exposed in the API
Blocking primitives	full support	limited support
Conform. classes	Kernel configurations	Yes, Used to limit footprint

ok… but where are the drivers?

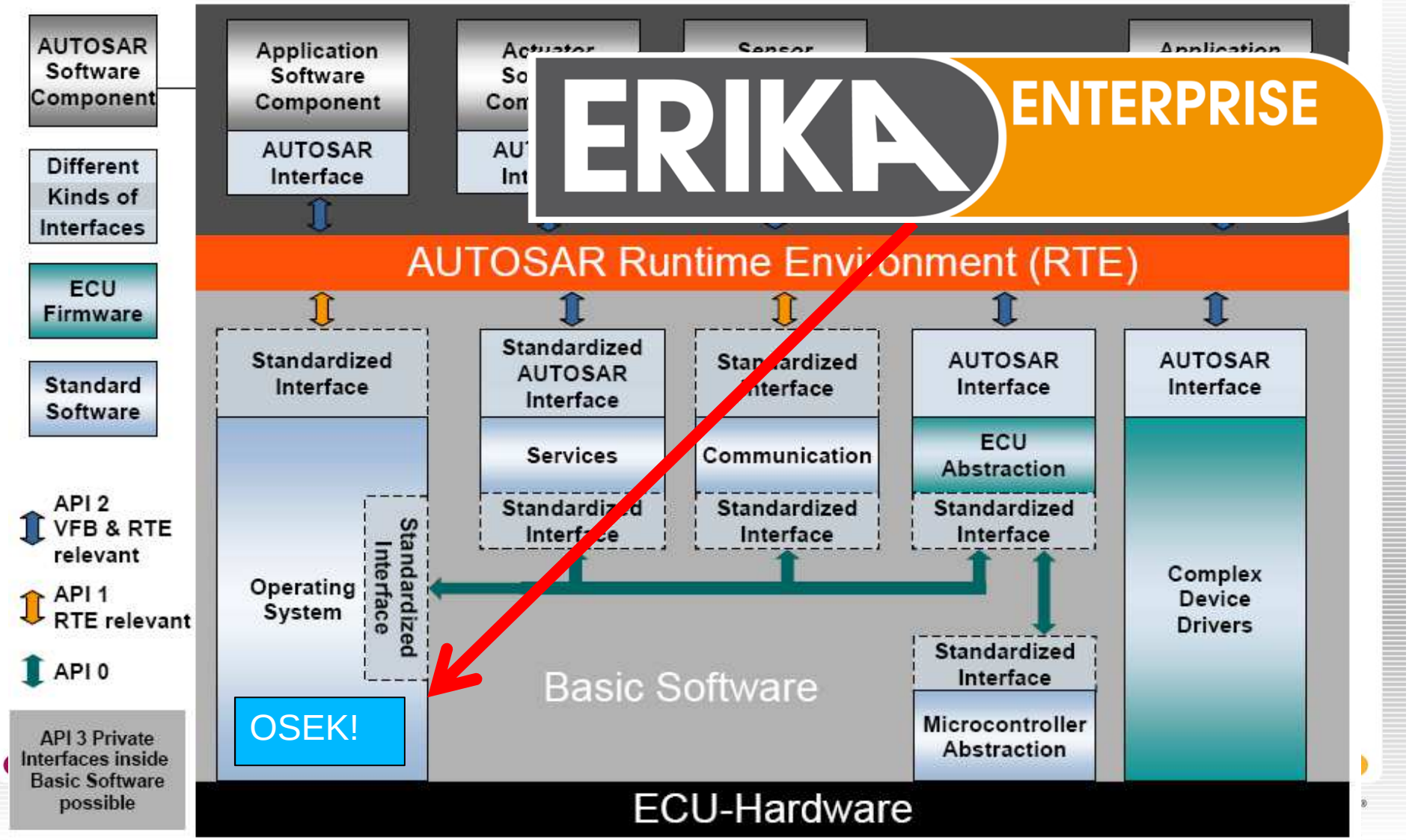
Most of the Linux code is made of drivers…

Where are the drivers in OSEK/VDX?

… they are in AUTOSAR!!!

(which is, from the OS point of view, an extension of OSEK/VDX)

AUTOSAR Architecture



Let’s go back to ERIKA

As you’ll see it’s not only the OSEK/VDX part

But where did everything started from?

… in other words… things never goes the way you plan them!

1999-2002 - Shark

At the beginning was … SHaRK!

A modular RTOS for PCs.

- With a modular scheduler
- Implementing around 10-15 different schedulers
- With pthread support
- With the «shadow» mechanism, equivalent to the “Linux proxies”
- Tickless

2000-2006 – Multicore and MSRP

ERIKA was born in 2000 to support the Janus dualcore

- Competitive advantage on the multicore part
- Started open-source on ST10, become closed source to leverage the multicore part
- But the project was canceled 6 months after founding Evidence! ☹

2000-2006 - Features

Features

- Tiny footprint
  - Initially as a reduction of SHaRK, from 50k to 10k
  - then I rewrote it from scratch, from 10k to <1k
- Static partitioning, shared stack
- 1 copy of the RTOS per core
- Porting for ARM7, ST10, Nios II
- In 2004 I implemented the OSEK/VDX Layer
  - Initially as a rename of the FP scheduling class
- But around 2004 AUTOSAR arrived, raising the barriers

2000-2006 – MSRP

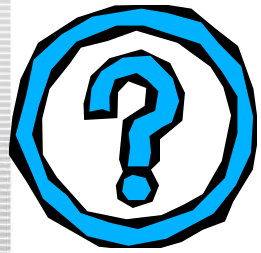
- MSRP algorithm, with queuing spinlocks implemented with the Graunke & Thakkar (1990) locks
  - Global resources identified automatically by the mapping at configuration time … not needed in AUTOSAR because the communication is handled by the RTE
- Then there was an allocation algorithm to place tasks into cores

AUTOSAR still does not have queuing spin locks…

- Which means we now removed the queuing spin-locks as default for production!

2000-2006 MSRP (2)

- …but what customers really want is to extend legacy OSEK/VDX code with MINIMAL changes



Question: is it possible to design an allocation algorithm that guarantees the minimum number of changes with respect to a given working software architecture?

No way industry will accept radical changes to an app due to allocation algorithms…

Probably a sub-optimal solution is ok too…

2006-2009: FLEX, open-source

- In 2006 it become clear it was impossible to sell ERIKA
- Collaboration with the ReTiS Lab to create a cheap board
- Flex boards!
  - dsPIC based
  - Port of ERIKA
  - Daughter boards
  - Created a collaborative platform

Idea… change the business model!

- we sell the hardware
- we give the RTOS for free

2006-2009: Licensing error!

RTOS for free?

- let's try Dual Licensing GPL + commercial
- In this way we'll be able to collaborate with Universities and labs!

Wrong choice:

- …. Are you going to make money out of the small snippet that I'm providing you GPL?
- … I could not make a commercial distro out of GPL code!

At the end, we reverted to… GPL2+Linking Exception

2006-2009: HW boards?

The FLEX boards were board in an environment where the cheapest evaluation board was around…

300€ ARM Evaluator7T

So a price around 100€ seemed to be fine!

But… at the same time came… Arduino!

- 30€ target
- Simple to blink a led, no RTOS

% of people needing…

- A multithreaded automotive minimal RTOS? → 0.x%
- Something simple to blink on a led? → a lot of … «Makers»!!!

2006-2009: Research Scheduling algs

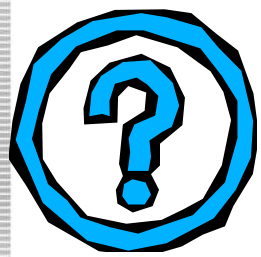
ERIKA implements:

- EDF with wraparound timers
  - In 300 bytes more than Fixed priority implementations!
- FRSH with resource reservation
  - In around 4k Flash on a Nios II
- (and in 2014) HR with hierarchical scheduling by Alessandro Biondi

2006-2009: Research Scheduling algs?

No way yet to get those results in production in automotive!

- They are too conservative, they protect saying the state of the art is in the standards! → AUTOSAR
- Better results on Linux ☺
  where SCHED_DEADLINE was merged in 3.14



Question: how this community can help in providing better scheduling algorithms to the automotive sector?

Note: I'm not talking about the usual RM vs. EDF!!!

2009-2012: automotive crisis

In 2009, after an automotive crisis, some companies started to look at open-source implementations of OSEK/VDX

Yes, still OSEK/VDX, as for some subsystems AUTOSAR was not required

Industrial usages: Cobra AT

The first one was Cobra AT

with:

2009 – feasibility for a OEM product (Freescale S12XS)

2012 – Cobra ParkMaster (Freescale S12G)

(integration work performed by Massimiliano Carlesso)

http://www.cobra-es.com/parkingaid_250.html

Magneti Marelli

Then came Magneti Marelli Powertrain Bologna

With support for:

- PPC MPC5674F (Mamba)
- MPC5668G (Fado)
- Tricore AURIX 27x and 26x
- AUTOSAR OS implementation (not yet open source)
- Other 2 MCUs (Cobra55 and K2)

then…

Aprilia Motor Racing on PPC Mamba

FAAM on S12XS

esi-RISC port (made by Pebble Bay)

(undisclosed)

TI Stellaris Cortex M4F, Renesas 2xx and AUTOSAR-like drivers

PPC Leopard
(paper submitted to SAE SETC2014)

2009-2012: …a right licensing is the enabler

The licensing model GPL2+Linking Exception turned out to be an advantage, because it allowed

- Customers to use the kernel in production
- Universities to contribute freely

The project was then moved to an independent site

http://erika.tuxfamily.org

And there was a business model change,
from a product business to a service business

2012-2014: OSEK/VDX certification

In 2012 we did the OSEK/VDX certification
funded by a research project on white goods

In 2012? Still someone cares about OSEK/VDX?

Well… yes!

- It gave a proof of quality of the code
- It gave automotive acceptance and visibility!

Then, we implemented the AUTOSAR OS specification for an AUTOSAR member, and various other ports

2012-2014: community…

We are getting reports of ERIKA used in various research projects and open-source initiatives:

- ZELOS3
- AMALTHEA
- MOTTEM
- INCOPBAT/eDAS
- eCOMPOSE
- ARAMIS
- P-SOCRATES (see later)
- Porting to the Arduino IDE and STM32F4Discovery

Lession learned…

- Business model change to get accepted…
- The market does not always go as you expect
  (… especially if you are an unknown spinoff company!)
- Software licensing is fundamental
- The platform is fundamental to aggregate a community
- Standards are not including the latest technology
  (see MSRP and queuing spin locks!)
- Customers follow the standards (no EDF!)
- Customers do not want to change, and have a huge amount of legacy code
  (allocation algorithms are not always applicable!)

Current status of ERIKA…

Hardware supported

ERIKA Enterprise supports the following microcontrollers:

Altera	Nios II
Atmel	AVR5, Arduino Uno, Arduino Nano
ARM	ARM7, Cortex M0/M3/M4
Ensilica	esi-RISC
Freescale	S12XS, S12G
Freescale	PPC z0, z4, z6, z7 (Mamba, FADO, Leopard)
Infineon	Tricore AURIX 26x, 27x
Lattice	Mico32
Microchip	PIC24, dsPIC, PIC32
Renesas	R21x
TI	MSP430, TI Stellaris Cortex M4

A Porting guide available on the ERIKA Wiki!

Compilers/IDE/debuggers support

- we support more than 10 compilers /development environments and in-circuit debuggers

In particular for Automotive:

TRACE32®

- Debug scripts automatically generated when compiling for PPC/AURIX/Nios II

WinIdea

- Directly supported by iSystem
  http://www.isystem.com/supported-rtos/erika

Lauterbach-Evidence press release!

PRESS RELEASE

LAUTERBACH DEVELOPMENT TOOLS

Lauterbach and Evidence Collaborate on OSEK/VDX Tool Chain

Hoehenkirchen-Siegertsbrunn, June 2014 – Lauterbach, the leading manufacturer of microprocessor development tools, and Evidence, a leader in open source embedded systems, announced a partnership, where Lauterbach's outstanding debugger TRACE32 supports seamlessly Evidence' ERIKA Enterprise, the first open source, royalty free, OSEK/VDX certified RTOS.

www.evidence.eu.com

EVIDENCE® EMBEDDING TECHNOLOGY

AUTOSAR compliance

- A non-public branch of ERIKA implements AUTOSAR OS 4.0.3 for an AUTOSAR Member (memory protection/multicore/scheduling tables);
- RT-Druid is capable of importing AUTOSAR XML produced by SystemDesk.

We developed a set of AUTOSAR-like MCAL for various architectures

- Cortex M4 Stellaris (DIO, DMA, GPT, MCU, PORT, SCI, SPI, WDG)
- Renesas R2xx
- MPC 56xx

available on the repository

Eclipse-based configurator available on

http://www.evidence.eu.com/products/eforms.html

MISRA C compliance and regression tests

A subset of ERIKA Enterprise has been checked for MISRA C compliancy

- tools used: FlexeLint 9.00h, configured using Magneti Marelli Lin 7.10, with some additional exceptions documented on the ERIKA Enterprise Wiki

Continuous integration test environment based on Jenkins

- Official OSEK/VDX conformance test suite
- Regression tests derived from the MODISTARC tests published on the OSEK/VDX website
  - See http://erika.tuxfamily.org/wiki/index.php?title=Main_Page#Regression_Tests

Benchmarks

Footprint statistics and Benchmarks have been published on the Wiki.

- A typical scenario of 16 tasks + resources + alarms uses 2-4 Kb flash depending on the MCU
- Timings of the primitives are in the range 2-10 usec

They are in line with other commercial offerings

community (2)

Website traffic doubled in the last 6 months!

Jan-May 2013

	Country / Territory	Sessions	% Sessions
1.	Italy	1,845	20.42%
2.	Germany	1,047	11.59%
3.	India	893	9.88%
4.	United States	773	8.56%
5.	France	409	4.53%

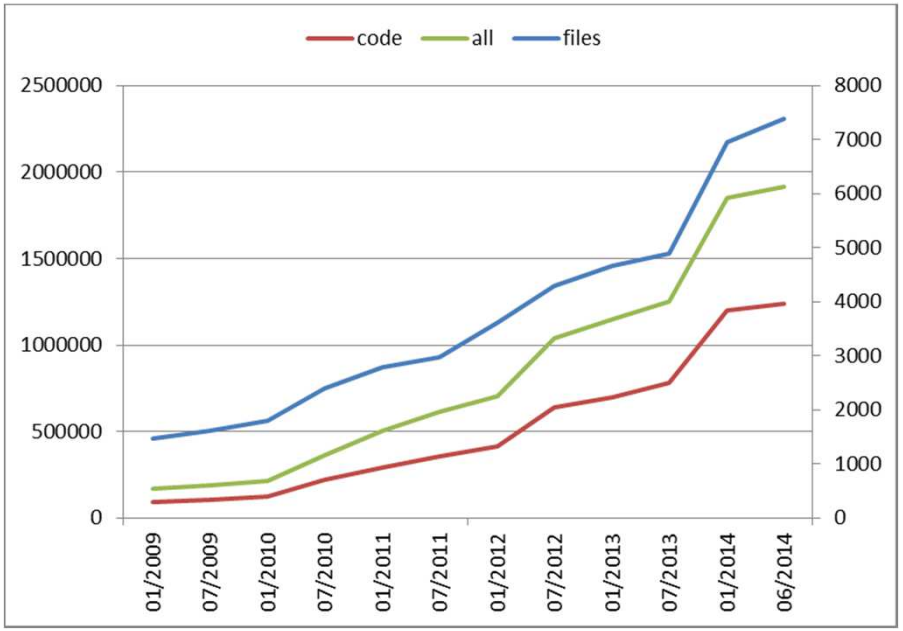
Jan-May 2014

	Country / Territory	Sessions	% Sessions
1.	Germany	2,596	16.46%
2.	Italy	2,292	14.53%
3.	United States	1,693	10.73%
4.	India	769	4.88%
5.	France	758	4.81%

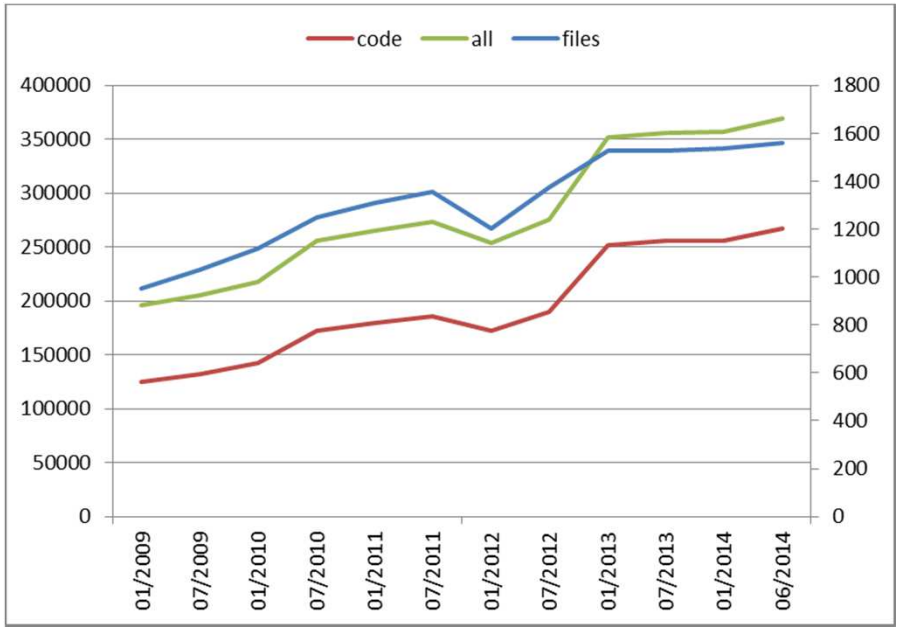
code size

The code base (mainly thanks to third party libraries) increased 3x from 2009 to 06/2014

ERIKA Enterprise



RT-Druid



The development community

http://erika.tuxfamily.org

- SVN repository open to the public
- Wiki and forum
- Application notes
  - Template system available in RT-Druid
- libraries for
  - console
  - uWireless (802.15.4 with beaconed mode / GTS support)
  - ScicosLab Libraries
  - Motor control
  - TCP/IP
  - CMOS Cameras, tracking
  - USB
  - various sensors
  - ball & plate, inverted pendulums, robot swarms

Let's try it… on a Virtual Machine!

In occasion of OSPERT, we released two VM:

ERIKA ENTERPRISE

LAUTERBACH DEVELOPMENT TOOLS

WIND RIVER

ERIKA + Diab Compiler + TRACE32 Instruction Set Simulator

ERIKA ENTERPRISE

Retis Real-Time Systems Laboratory

ERIKA + GCC compiler

Support for

- Arduino Uno / Nano
- STM32F4Discovery

http://www.erika-enterprise.com

future plans

what will be the project in 3-5 years?

A complete AUTOSAR implementation?

- Let’s consider what is currently happening in the automotive sector…

the basic idea

- Cost reduction is an important factor in automotive
- Every company is implementing (or buying) every time the same subsystems
  - RTOS (OSEK/VDX or AUTOSAR)
  - Device Drivers
  - Diagnostic protocols
- We always think in terms of **Make** or **Buy**…

there is an opportunity to

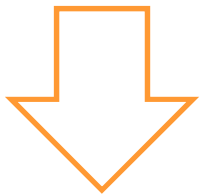
Share

software components
not in the core business

sharing in automotive

Sharing source code in automotive means:

nobody makes a free gift to competitors



we need a platform
where each company
adds a small part

first example: the Eclipse framework

The core business of tool makers
is on new functionalities,
not in the text editor!

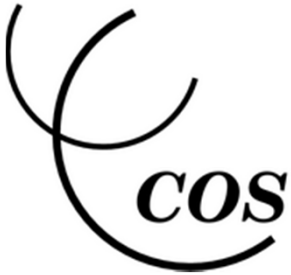
- The automotive world adopted Eclipse since years
- Artop is a common Tool Platform for AUTOSAR
  - why writing another AUTOSAR XML importer?
- Artop is based on EMF and Sphinx

http://www.eclipse.org/sphinx/

http://www.artop.org/

second example: code generation tools

We used the open source tool ScicosLab
as a base platform for providing
simulation and code generation
for control algorithms



http://www.e4coder.com

Composed by:

Code generator for embedded targets

Finite State Machines editor / FSM codegen

Simulation / code generation of GUI Panels

third example: Linux in infotainment

Many new infotainment systems on car are based on Linux and Android

Automotive Grade Linux - http://www.linuxfoundation.org

Tizen - https://www.tizen.org

Genivi - http://www.genivi.org/

… just take a look at this citation from

Oct 12th, 2012

> The Next Battleground for Open vs. Closed: Your Car

Wired, Oct 12th 2012

"A luxury automaker recently told me its IVI system contains about 1,900 use cases – "of which we only consider about 3 percent unique to our products; the other 97 percent are common across all car companies." Let me emphasize that: THREE percent. Can these companies really afford to pour a lot of time and money into such a small amount of differentiation?"

"But here's the paradox: The automotive industry is going to have to collaborate in order to differentiate."

"Competitors collaborate on the code and requirements to produce a common base, upon which they differentiate and compete with each other."

http://www.wired.com/opinion/2012/10/automakers-become-software-makers-the-next-battle-between-open-and-closed/

Let’s go back to the platform…

The FLEX boards helped creating a platform

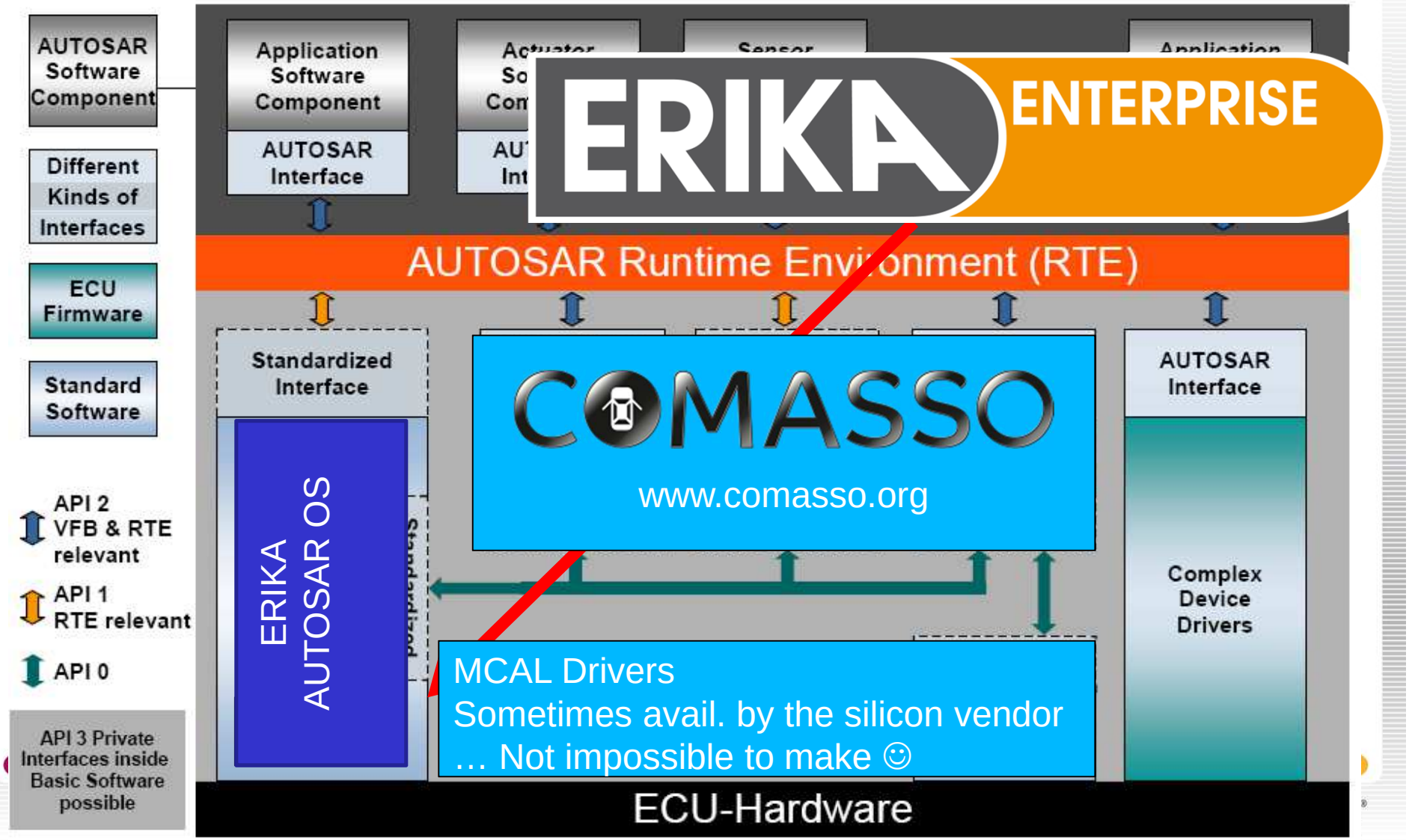
- Fundamental for the growth of every open-source project
- Think at Linux: Platform = x86 + GNU Project!!!

ERIKA could be the starting block for this platform!

Licensing is crucial to aggregate the users:

- Universities
  - No dual licensing, need for a low-cost platform
- Companies
  - Sharing to save costs on non differentiating features

(open?) AUTOSAR Architecture



A complete AUTOSAR implementation?

- The industry is converging to a common shared ecosystem of open or pseudo-open software
- ERIKA Enterprise could play the role of the RTOS…
- COMASSO is the way to go for the basic software (but remember it is not opensource
- The missing part is the RTE… which is still lacks an opensource implementation
- … plus, as usual, a lot of integration effort!

Question: any ideas from the realtime community on how to better converge this?

…hmm…. Maybe too related to the implementation?

ISO26262 qualification

The rise of the ISO26262 standard impose changes in the software and a whole new level of tests to the code

We are currently discussing possible qualification strategies for ERIKA Enterprise, including

- In-context qualification
- Out-of-context (SEOOC) qualification

It may happen in the next three years!

ISO26262: Freedom from interference

ISO26262 mandates (part 6, annex D) the testing of the freedom from interference.

- this Annex provides examples of possible mechanisms that can be considered for the prevention, or detection and mitigation of interference between components

- D.2.2 Timing and execution
  - blocking of execution;
  - deadlocks;
  - livelocks;
  - incorrect allocation of execution time;
  - incorrect synchronization between software elements.

ISO26262: Freedom from interference (2)

EXAMPLE Mechanisms such as cyclic execution scheduling, fixed priority based scheduling, time triggered scheduling, monitoring of processor execution time, program sequence monitoring and arrival rate monitoring can be considered.

D.2.3 Memory

[…]

- read or write access to memory allocated to another software element.

ISO26262: Freedom from interference (3)

D.2.4 Exchange of information

- repetition of information;
- loss of information;
- delay of information;
- insertion of information;
- masquerade or incorrect addressing of information;
- incorrect sequence of information;
- corruption of information;
- asymmetric information sent from a sender to multiple receivers;
- information from a sender received by only a subset of the receivers;
- blocking access to a communication channel.

ISO26262: questions

Question: Can the freedom of interference on the Timing and Execution be solved just with an execution budgeting control?

- Hey! This is «mixed criticality»!
- Hey! This can be solved using proper allocation of priorities, plus simulation and Schedulability Analysis!

ISO26262: questions (2)

…but… It's like someone telling you the world is make in a given way and you need to find the right point in the world where everything works…

Can we find:

- A proper scheduling algorithm which avoids the interference (resource reservation?)
- Plus some modeling framework to help designing the system in a proper way???

Multi-thread, Multi-core, … Multi-OS!

…yet another trend in the automotive for cost reduction!

The basic idea…

Automotive embedded systems changed over time

- 1985 – Isolated embedded architectures
- 1995 – Distributed architectures over CAN bus
- 2005 – Integrated architectures based on AUTOSAR
- 2015 – Distributed architectures based on Multicore AUTOSAR + Infotainment solutions

What's next?

…is cost reduction

- 2025 – Distributed architectures …
  …with small number of nodes

Need to:

- Integrate applications from different sources → AUTOSAR components
- Integrate applications with heterogeneous timing requirements → schedulability analysis
- Integrate applications with different safety levels → mixed criticality, mem. protection

… but then…

- Integrate applications with different semantics → ???

A static world…

- Static allocation of resources, Static software architecture, control
- No dynamic allocation of memory
- Hard realtime, safety critical
- Limited HW resources

Compared with a dynamic world:

- Infotainment has relaxed real-time constraints
- Works on Linux-based systems (or similar)
- GUI, Network, Graphical libraries, standard applications
- iPhone/Android integration, App stores

the “dynamic” side: Linux in infotainment

Many new infotainment systems on car are based on Linux and Android

Automotive Grade Linux - http://www.linuxfoundation.org

Tizen - https://www.tizen.org

Genivi - http://www.genivi.org/

ok, Linux is there… but…

…there are requirements of future IVI systems!

- Fast Boot
  - there must be a subsystem ready to go in a few ms
  - Linux boot times are usually in the order of seconds
- Real-Time support
  - there must be a subsystem with real-time performance
  - e.g. CAN Bus, motor control
- Quality of Service
  - IVI applications need soft-realtime support
  - for video/audio content

Infotainment, Linux, and multicores

- Next generation infotainment systems will be multi-core
- They can host more than one OS

What about

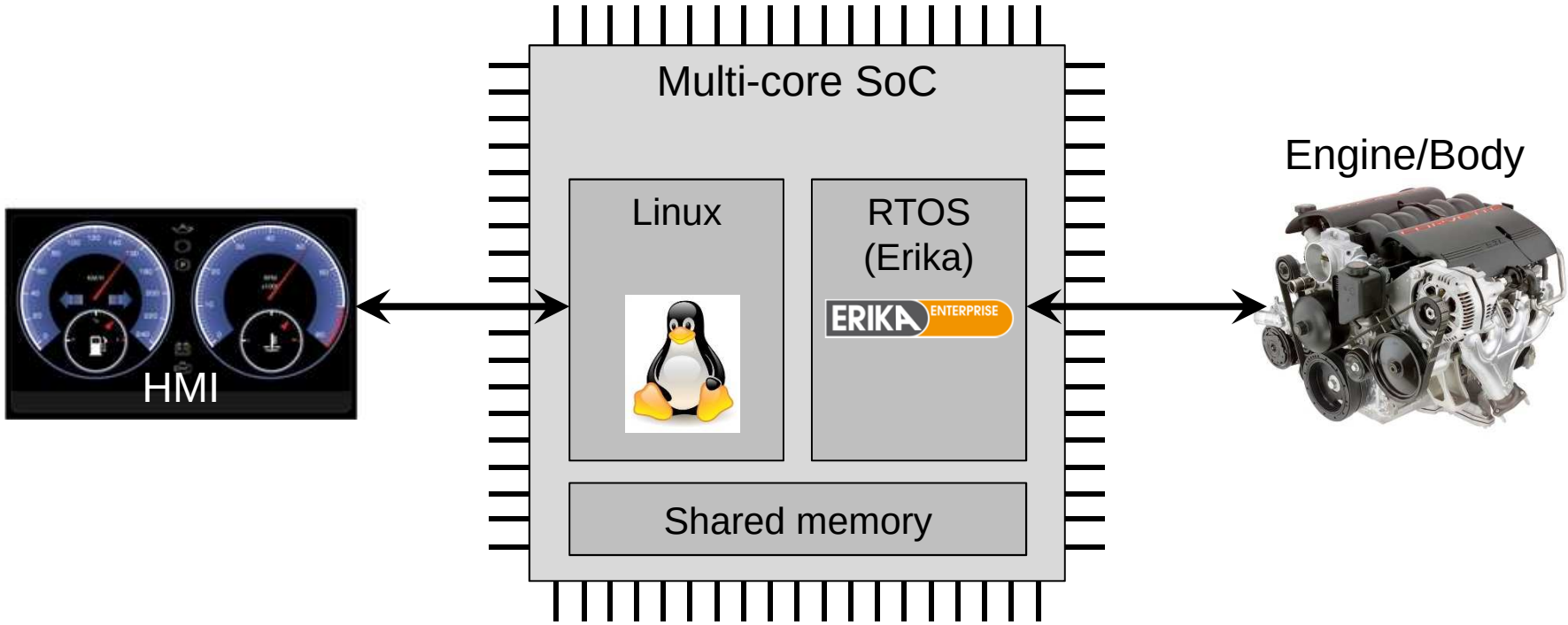
creating a complete
open-source environment
for automotive systems integrating

Infotainment + OSEK/VDX/AUTOSAR

on the same chip?

Towards a fully Open-Source platform

We envision the possibility to exploit multi-cores to run Linux and Erika Enterprise complementing each other!



Opportunity

Linux Embedded

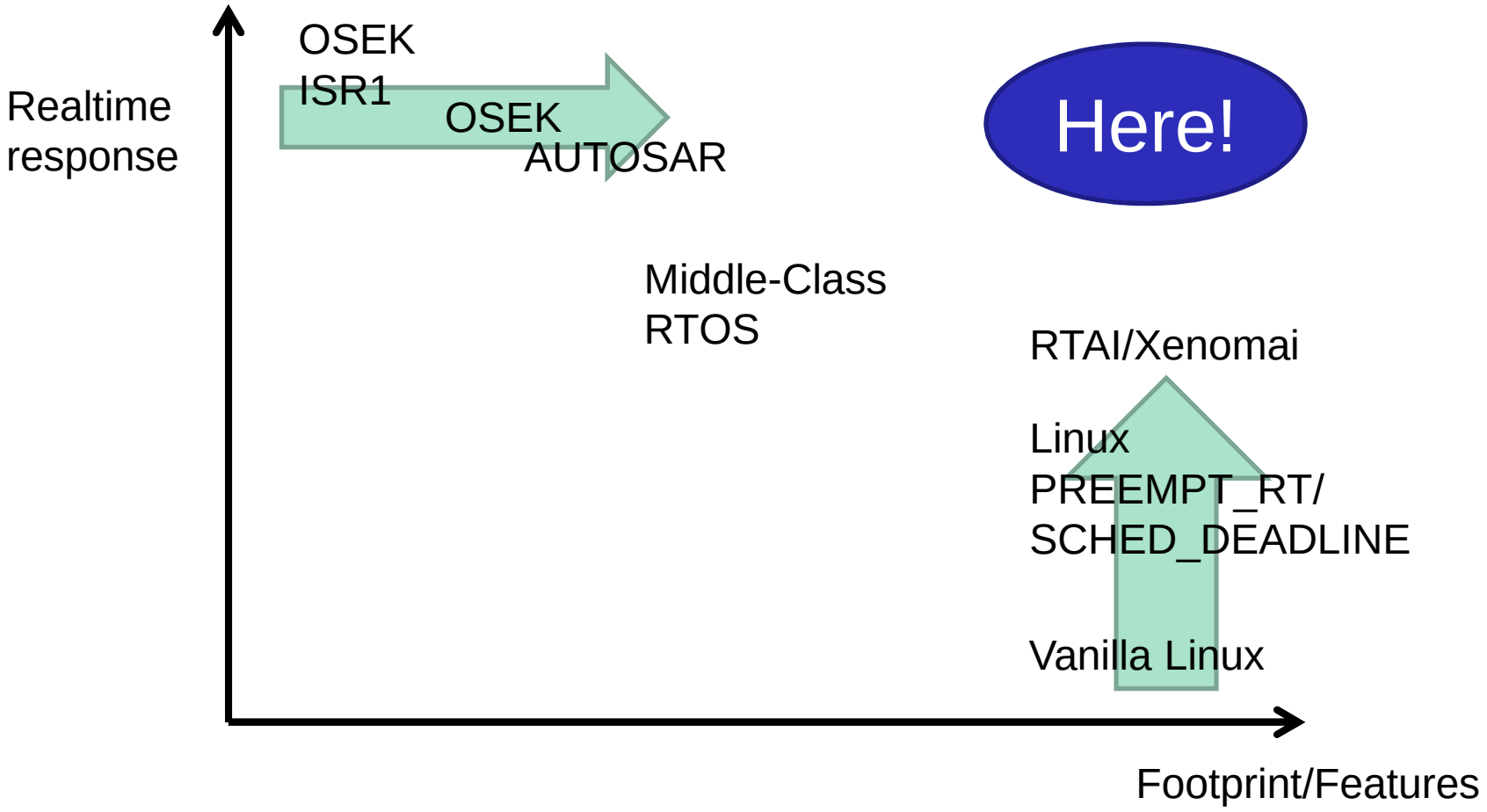
- Drivers, Displays, and communication infrastructure
- Soft Real-Time support using Linux and SCHED_DEADLINE

ERIKA Enterprise

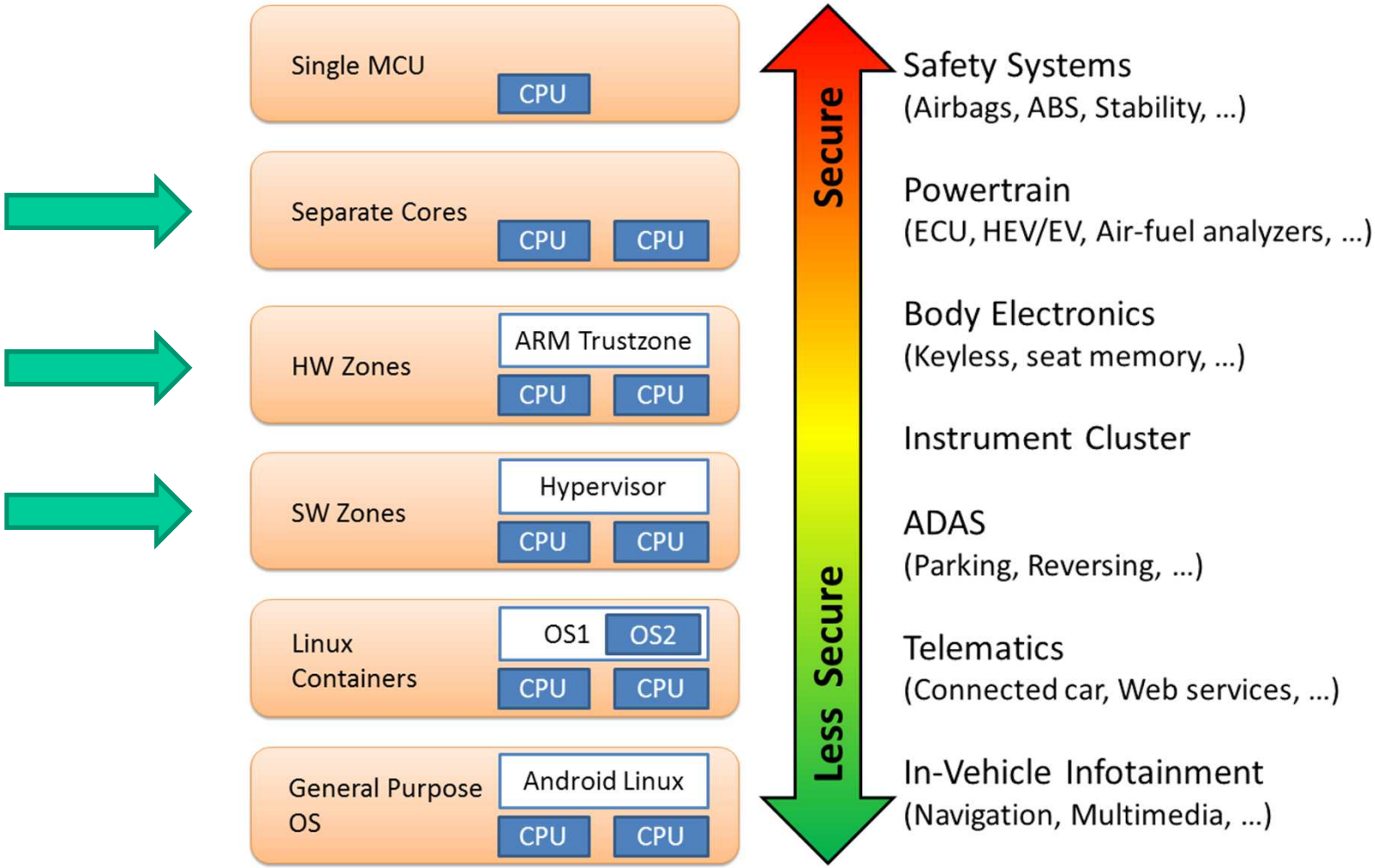
- Hard Real-Time support
- Open-source
- OSEK/VDX system, born for automotive

on a single multicore chip!!!

Footprint vs. Realtime



Integration at different levels…



Original source: Mentor Graphics,
Automotive Linux Conference Oct 2013

Three scenarios for the separate cores

1) Linux boots, ERIKA = special «device» for Linux

- slow! → ERIKA needs to wait for Linux boot

2) Hypervisor-like approach

- both ERIKA and Linux as hypervisor «clients»

3) ERIKA boots from U-Boot

- modified U-Boot to boot both ERIKA and Linux

HW Zones

- We could use the ARM TrustedZone
- ERIKA should be put in the Trusted Zone…

Advantage: good for automotive safety qualification!

…still an idea, nothing implemented by us so far…

Hypervisor → SW Zones

We need a good hypervisor… candidates are:

- XEN
- KVM
- NOVA ← 9000 lines! …good for certification!

Current work on XEN

- A master Thesis by Arianna Avanzini done in collaboration with UniMORE (Prof. Paolo Valente) is almost ended
- ERIKA will be hosted as a domU of XEN
- It is a first step towards having ERIKA as dom0

Demo on separate cores

Demo based on a Freescale iMX6

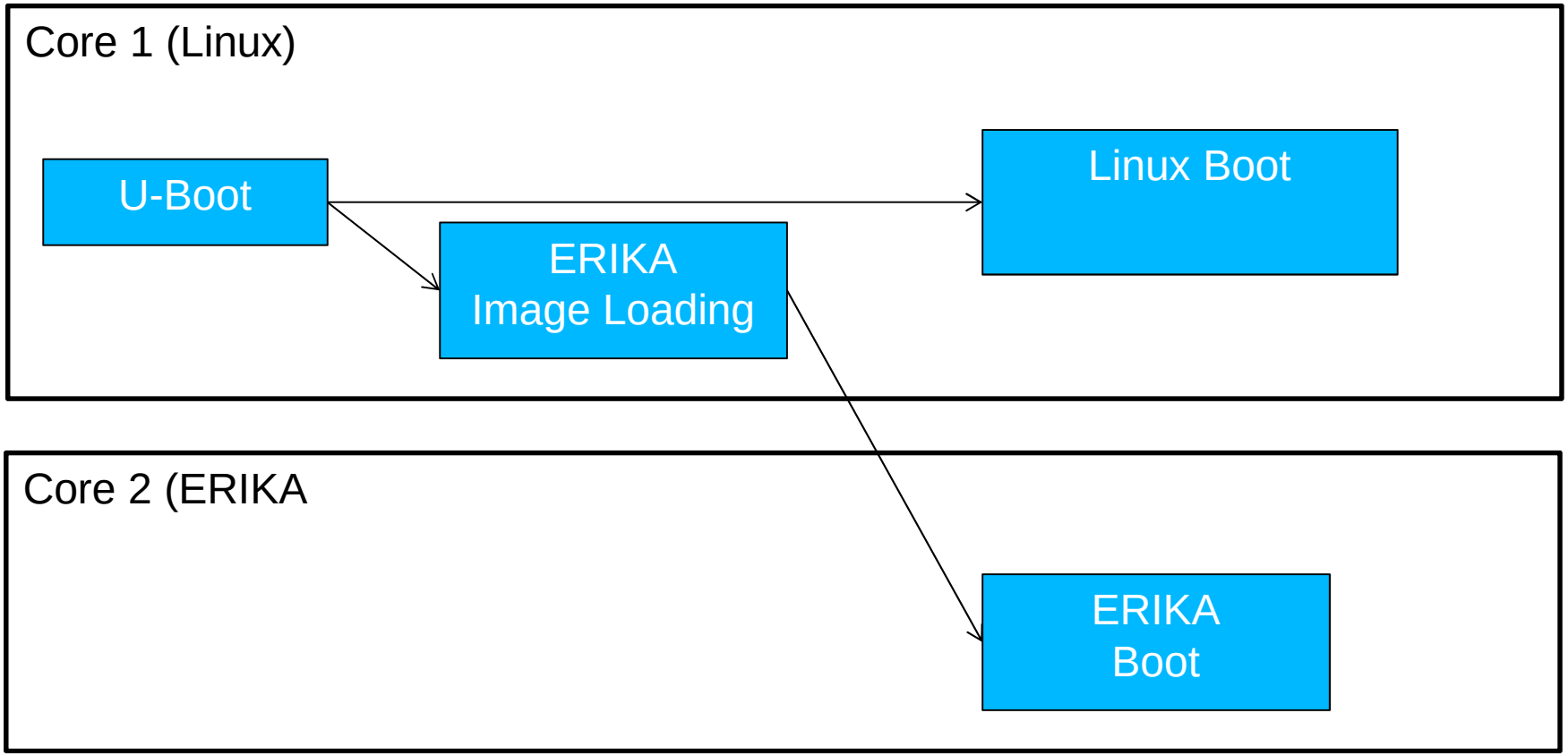
We let U-Boot handle the multicore boot

- ERIKA starts almost immediately
- Linux can start afterwards

No hypervisor

- could be useful in some cases to protect the behavior of misbehaving applications
- limited need because we statically allocate a CPU to each OS

The idea…



Interaction model

Linux → ERIKA

- Linux can trigger the following actions:
  - activate a task
  - set an event
  - start an Alarm
  - increment a counter

  (similar to those doable on a remote core of an AUTOSAR OS)
- Linux can stop and reload the ERIKA application

Linux ←→ ERIKA

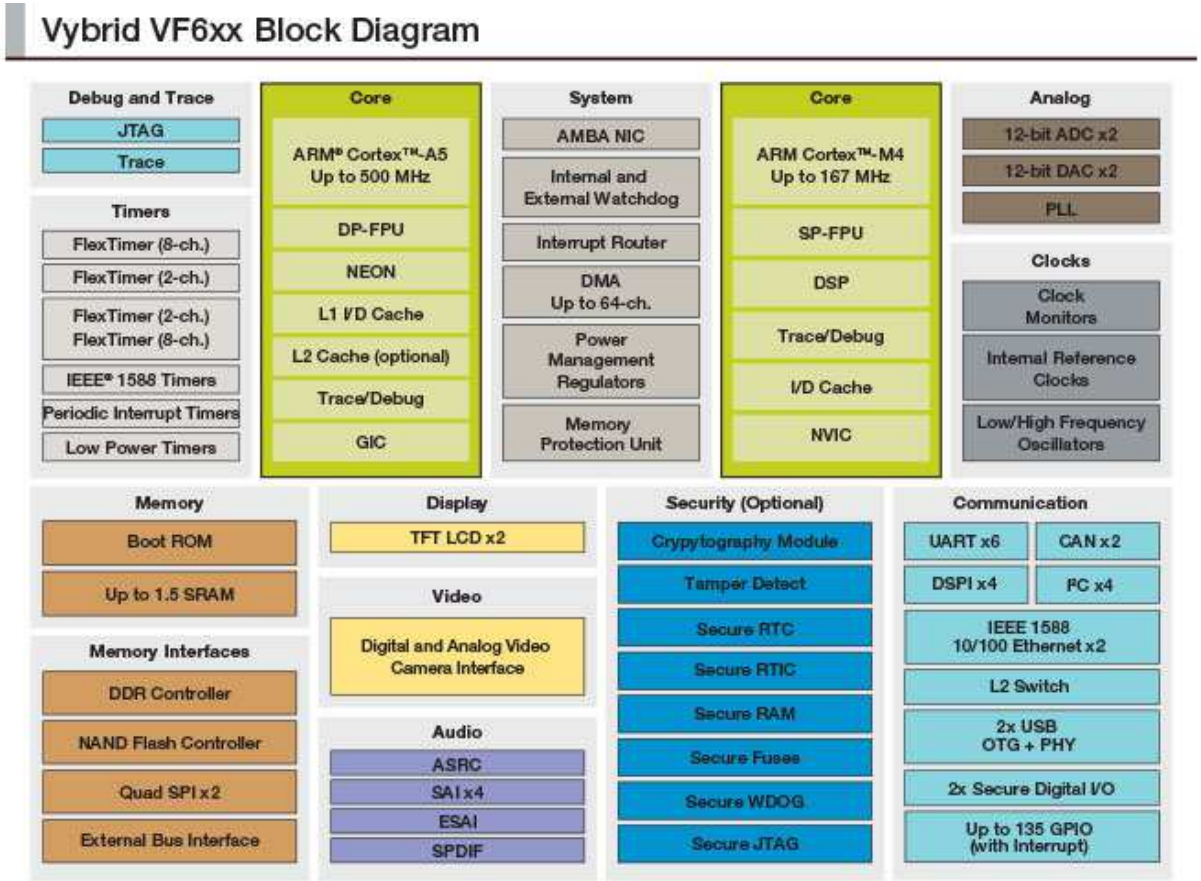
- Simple asynchronous message passing allowing asynchronous read/write of variable length buffers on predefined channels

The demo…

- Implemented on a iMX6 board from Engicam (http://www.engicam.com/prodotti/icorem6.html)
- U-Boot loads ERIKA, then Linux
- ERIKA generates a SawTooth signal
- Linux reads the message and displays the data
- A slider can be used to set the sawtooth signal amplitude
  - implemented through messages
- Simulated LED
  - implemented through interprocessor interrupt
  - there can’t be a demo without a Blinking Led!

...the future will not be SMP

Vybrid with Cortex A5 + Cortex M4



Software reuse…

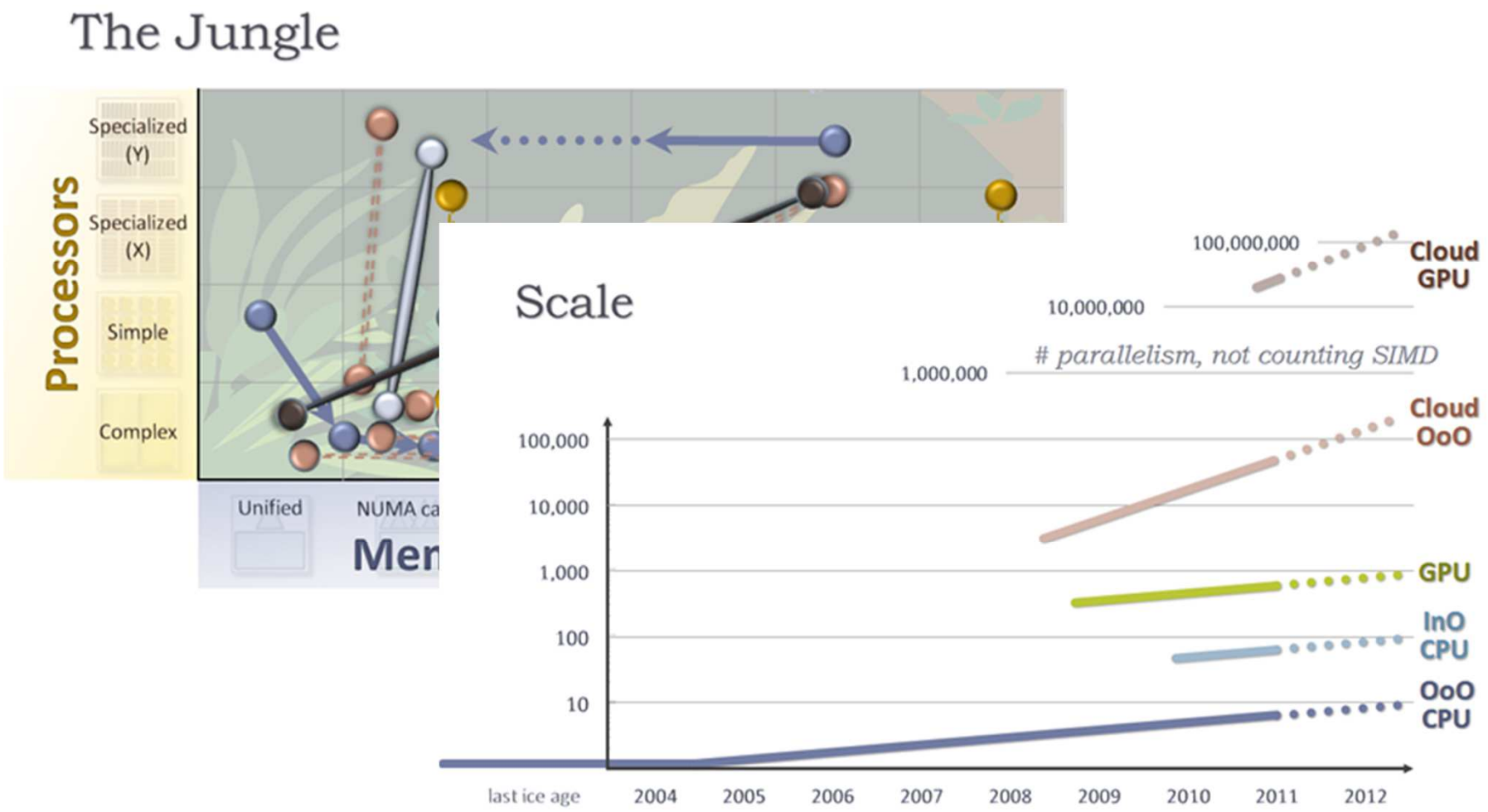
- The whole hypervisor thing is there because people want to reuse their software
  - There was a talk at SIES 2014 in Pisa of BMW using ERIKA with the ETAS Hypervisor in the context of the ARAMIS Project

What is the best way to recycle legacy code coming from previous AUTOSAR systems?

- Depends on the real architecture available
- But what is the best architecture available? It's a Jungle!

http://herbsutter.com/welcome-to-the-jungle/

The Jungle...



from http://herbsutter.com/welcome-to-the-jungle/

… in the next few yrs:

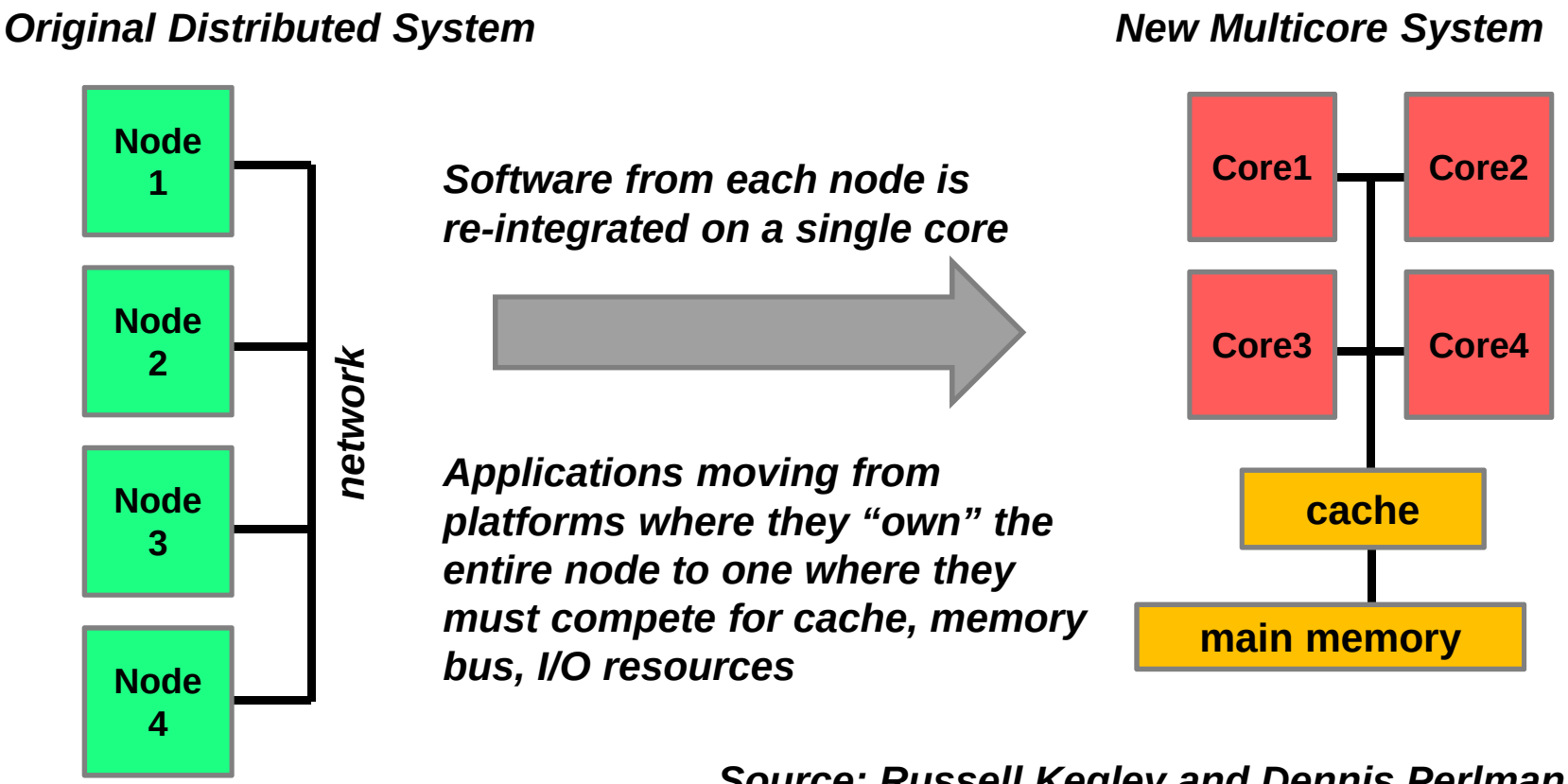
You can expect to have automotive architectures to be:

- Heterogeneous in strange ways…
  - Cortex A + Cortex M
  - Mixture of Lockstep cores and normal cores
  - Small CPUs as accelerators near peripheral buses
  - Seems like that they will fit a CPU where is space in the die…
- MPU and not MMU, Hypervisor extensions
- AUROSAR is good in implementing the instruments for building the system…
  … but you have to do the analysis yourself

multi-core equivalence (collab. UIUC)

"Single Core Equivalent (SCE) Architecture Framework for Safety Critical Multicore Systems"

http://rtsl-edge.cs.illinois.edu/SCE/



Source: Russell Kegley and Dennis Perlman

Multicore and beyond…

And to complicate things…

- Future automotive systems will require high computational load

The answer is probably to use a many-core platform

… think for example at:

- ST P2012 (killed!)
- Kalray MPPA
- Adapteva Parallela
- TI Keystone

many-cores: P-SOCRATES FP7 Project

P-SOCRATES

Parallel Software Framework for Time-Critical many-core Systems

P-SOCRATES (FP7-ICT-611016)

SEVENTH FRAMEWORK PROGRAMME

Oct 2013 – Oct 2016

http://www.p-socrates.eu

ERIKA will be hosted
on the KALRAY MPPA
(256 cores + 16 I/O cores!!!)

CISTER - Research Center in Real-Time & Embedded Computing Systems

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

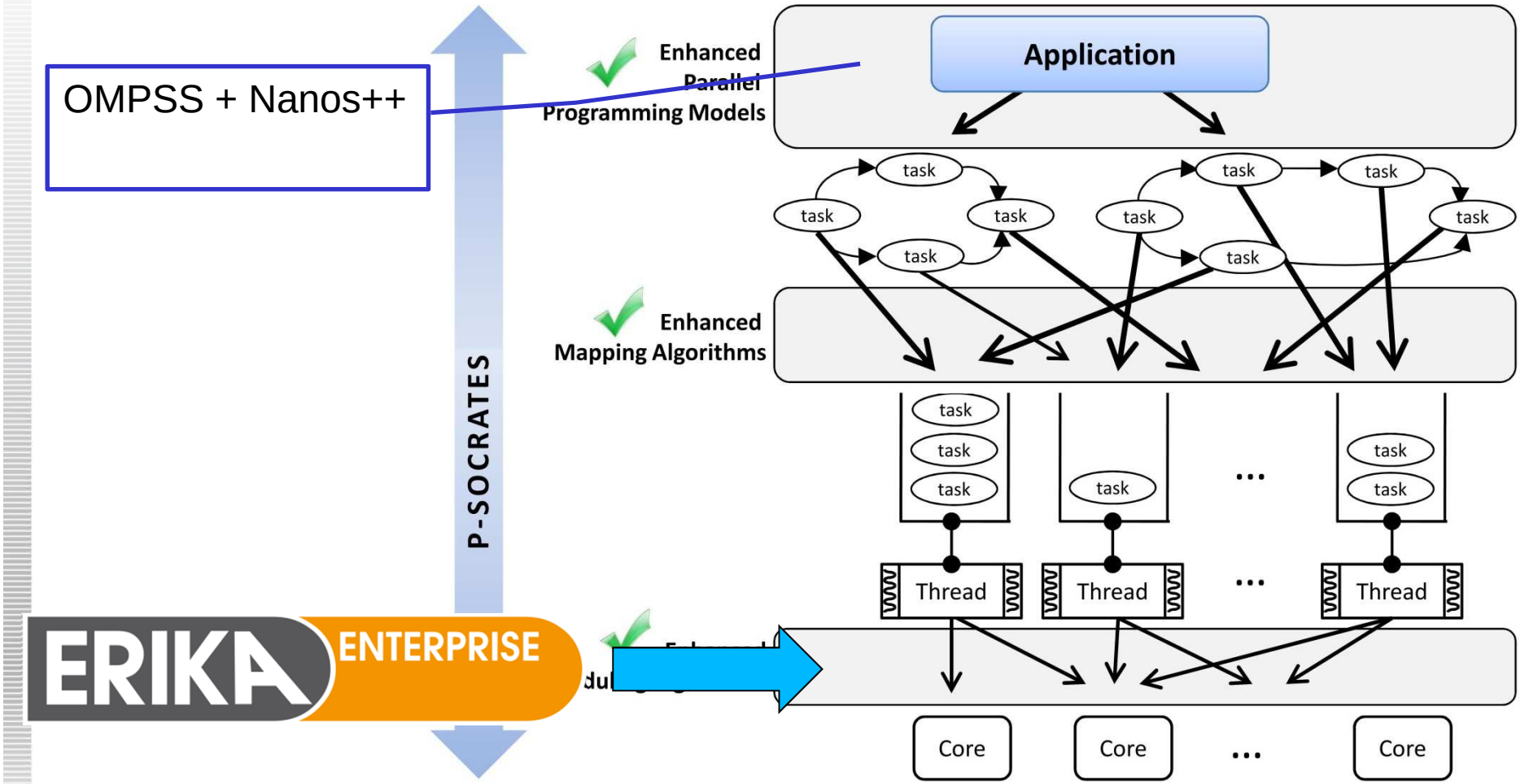
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

EVIDENCE®
EMBEDDING TECHNOLOGY

AT

Atos

P-SOCRATES Architecture



So other questions…

Again, in the short term, think at a lot of legacy code to be ported on a multicore machine with hypervisor.

- How to do schedulability analysis and placement?
- How the SMP schedulability bounds expand to these architectures?
- How should we model the overhead (preemption/communication/hypervisor)
- Should we use the hypervisor only to isolate the safety critical jobs?
  One guest OS per CPU, or more than one per CPU?
- Should the hypervisor be «transparent» to the choices of the guest RTOS, in a way to implement hierarchical scheduling?
- Can we do all this without hypervisor using a proper scheduling algorithm?

Finally consider powertrain applications

For example… powertrain applications

- Not only periodic tasks
  - Variable period tasks depending on the engine speed
  - Computation times dependent on speed with hysteresis
- Are deadlines really hard?
  - Just a few…
    inertia in the engine helps also with non optimal controllers
  - Tasks with high offset
    to check for detonation to avoid the knock phenomena
  - Oversampling… could be not dangerous to miss some activations (skip model?) But how many of them can I skip?
- How can we model and analyze these kind of applications?
- Some efforts done in WATERS to mix SysML with Matlab to give architecture definition…

conclusions

…a lot of opportunities

- Next years will be full of great opportunities especially in the multi-core area
- Still to be understood which is the best way to go
- Automotive systems are rather “static” and slow-moving, but I believe there will be good opportunities for research in that area!
- … if you are going to use ERIKA… give feedback, patches, and bugs to help future developments!

… some acknowledgements

- Thanks Björn!

- … and thanks to all people in Evidence and in the ReTiS Lab who worked with me in making all this possible!

Contacts

Paolo Gai
Evidence Srl
Via Carducci 56
56010 S.Giuliano Terme
Pisa - Italy

Web: http://www.evidence.eu.com
E-mail: pj@evidence.eu.com
Phone: +39 050 99 11 224 then 101

Thank you for listening !

Questions ?

Appendix A – details on the multicore implementation

U-Boot code changes

we added the `cpu` command to U-Boot

- (cherry pick from PPC to iMX6)

Multiprocessor CPU boot manipulation and release

`cpu <num> reset`

← Halts cpu <num>

`cpu <num> status`

← prints latest `<addr>` and `r0`, plus the status

`cpu <num> release <addr> [args]`

← Restart of cpu <num> at <addr> with a value for the r0 register

Linux code changes

- Linux runs on a subset of the available CPUs
  - 1 CPU dedicated to ERIKA
- IRQs are mapped statically to cores
  - additional boot parameter to map the GIC IRQs that Linux cannot use

```
git_skip_intid=142-147,152,180,205-220
```

Linux code changes

- a fixed amount of memory is allocated to ERIKA
  - ERIKA allocated in the first part of the RAM, Linux afterwards

- Idle time does not change CPU frequency
  - Linux by defaults reduces the CPU frequency on idle time

ERIKA code changes

- ERIKA is statically linked on the first 128 Mb of the available RAM
- the Memory Protection Unit (MPU) has been programmed to limit the possibility to write only inside the allocated memory
  - it will not destroy Linux!
- the OIL file used to configure ERIKA has been extended
  - Cortex A support
  - ORTI support through 
    - We can make an AMP cor[illegible] Linux on one core and ERIKA (with ORTI support) on the second core

OIL file extensions

```
CPU mySystem {
  OS myOs {
    MASTER_CPU = "master";      ← the ERIKA CPU

    CPU_DATA = CORTEX_AX {
      CPU_CLOCK = 660.0;
      APP_SRC = "main.c";
      COMPILER_TYPE = GNU;
      MODEL = A9;
      ID = "master";            ← the ERIKA CPU
    };

    CPU_DATA = LINUX;           ← the Linux CPU
```

OIL extensions

```
MCU_DATA = FREESCALE {          ← The MCU information
  MODEL = IMX6Q;
};
BOARD_DATA = ENGICAM_ICOREM6;

REMOTENOTIFICATION = USE_RPC; ← configuring RPC
USEREMOTETASK = ALWAYS;
USEREMOTEEVENT = ALWAYS;

ORTI_SECTIONS = ALL;    ← ORTI support through
                          Lauterbach Trace32
```

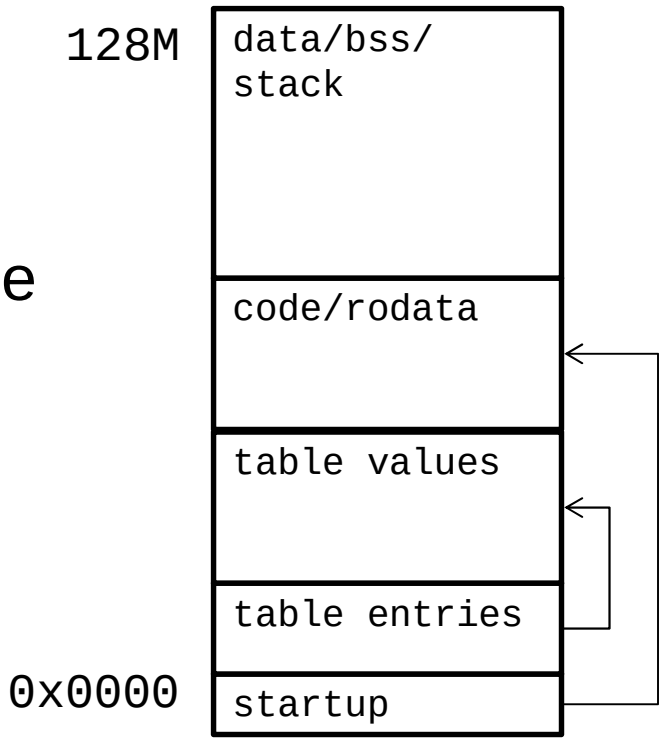
OIL extensions

```
    MESSAGE_BOX = TRUE {    ← Asynchronous message box
      NAME = "led_status";
      DIRECTION = OUT;
      MAX_MESSAGES = 5;
      MAX_MESSAGE_SIZE = 8;
    };
  };

  TASK Blinking_led_task {
    CPU_ID = "master";      ← Task mapping to the master CPU
    [...]
  };
};
```

ERIKA binary format

- we defined a custom binary format for the ERIKA images
- symbol table with a DB of the entities defined in the OIL configuration file
- a customized Linux driver reads the DB and publishes the data into `/sys` and `/dev`



Linux driver

a custom driver allows Linux to do the following actions:

- activate a task
- set an event
- start an Alarm
- increment a counter

These are remapped to interprocessor interrupts in a way similar to what specified by multicore AUTOSAR

In addition we implemented a simple asynchronous message passing primitive allowing asynchronous read/write of variable length buffers on predefined channels

/sys filesystem structure

```
/sys/class/
    EE_alarms
        [...]              ← information about alarms
    EE_buffers
        led_status             ← asynchronous message channel
            direction
            size
            [...]
    EE_tasks
        led_task           ← tasks defined in the OIL file
            activate       ← writing to this file remotely activates a task
        [...]
    mem_ex
        symbols            ← symbol table as read in the binary image
```

/sys/class/mem_ex/symbols

```
# cat /sys/class/mem_ex/symbols
                  rpc 0xb9b0003c   RPC     28
   Blinking_led_task 0xa9000000   TASK     0
      Saw_tooth_task 0xa9000000   TASK     1
   Activate_led_task 0xa9000000   TASK     2
         AlarmMaster 0xa9000000   ALARM    0
       CounterMaster 0xa9000000   COUNTER  0
          led_status 0xb9b00058   IN      40
      saw_tooth_data 0xb9b0006c   IN     400
 saw_tooth_data_max 0xb9b00080   OUT     16
```

/dev filesystem structure

```
/dev/
    mem_ex          ← ERIKA image write
    led_status      ← asynchronous message channel
    [...]
```

- it is possible to reprogram and restart the ERIKA application by writing on `/dev/mem_ex`
- asynchronous channels are inserted in the `/dev` filesystem automatically
  - you can read/write single messages
  - no remote notification – completely asynchronous

memory protection

- each core has its own Memory protection unit

ERIKA Enterprise

- single table with 1Mb pages
- ERIKA cannot write outside its own memory space
- currently we allocate 128 Mb (should be enough ☺)

Linux

- first available address after the end of the ERIKA image
- Linux can access ERIKA memory only through the driver

Other features

Spin Locks

- ERIKA and Linux use spin locks to guarantee mutual exclusion during the access to shared data structures
- the spin lock location resides in the ERIKA memory space

Interprocessor interrupt

- currently used Linux → ERIKA
  to implement remote notifications
- data exchange is implemented using asynchronous messages

Appendix B – more details on E4Coder

E4Coder - facts

E4 CODER is a toolset which is able to:

- simulate continuous time and discrete time designs
- simulate finite state machines
- GUI panel generation
- generate code without changing the design
- with and without RTOS
  - with support for microcontrollers without RTOS
  - with support for OSEK/VDX RTOS
- support for multi-rate designs

http://www.e4coder.com

key advantages

high-level simulation of a design

- you can simulate the design before generating the code
- the code generator preserves the model correctness

finite-state machines

- you can simulate and generate code for state machines

technical support

- for using the code generator
- for converting existing designs

Building Blocks

- ScicosLab
  - Simulation engine, http://www.scicos.org

- E4Coder Code Generator
  - Code generation for embedded targets

- SMCube
  - Simulation / code generation of Finite State Machines

- E4Coder GUI
  - Simulation / code generation of GUI Panels

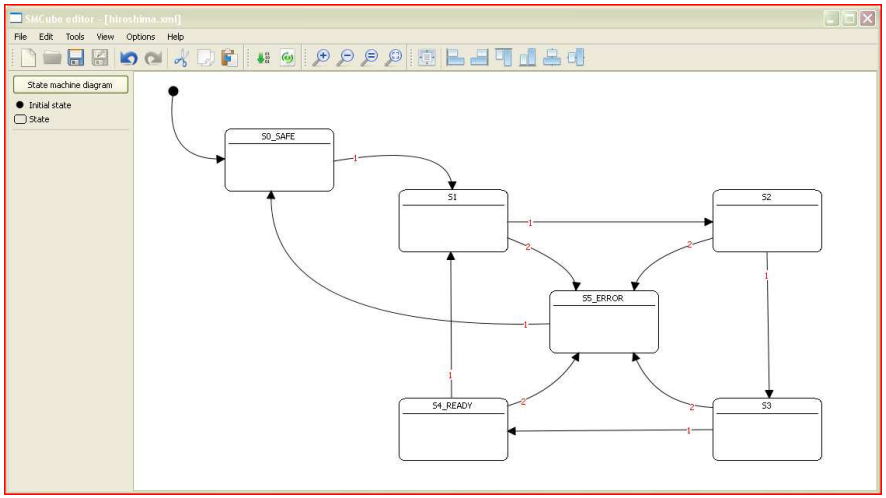
- E4Box

E4Coder CG

- ScicosLab blockset
- Optimized code generation
- Peripheral blocks independent from the target
- Same diagram used for simulation and code generation
- Multithread code generation support

SMCube

- stands for: SMCube is a State Machine System modeler
- Flat Discrete-time State Machine editor
- Simulation and Code generation of state machine diagrams
- Integrated in ScicosLab
- Hierarchical State Machines
- Junction points



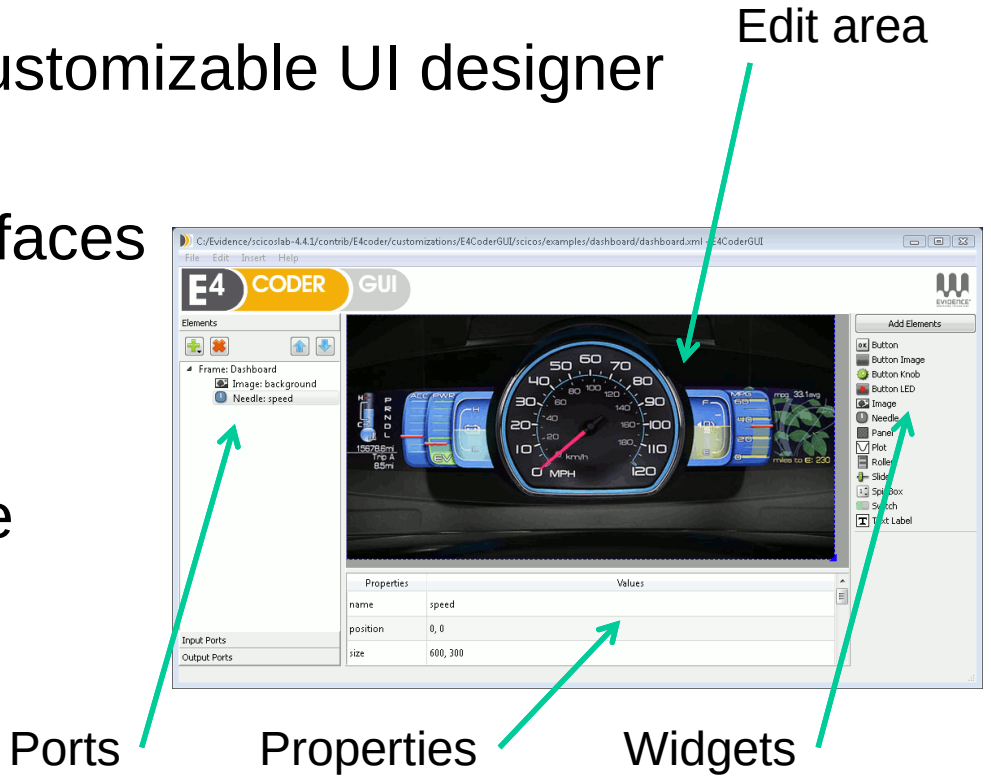
E4Coder GUI - customizable UI designer

E4Code GUI is a simple customizable UI designer which is able to:

- design simple user interfaces
- insert simple widgets
- simulate the interface
- generate QT target code

Example:

- Dashboard panel



E4Box

E4Box is a ready to use all-in-one embedded computing box

- Intel Atom processor
- NI PCI-6221 Data acquisition board
- Open Edition
  - Linux+RTAI+Comedi+ScicosLab open source software
- Professional Edition
  - Open Edition + E4Coder