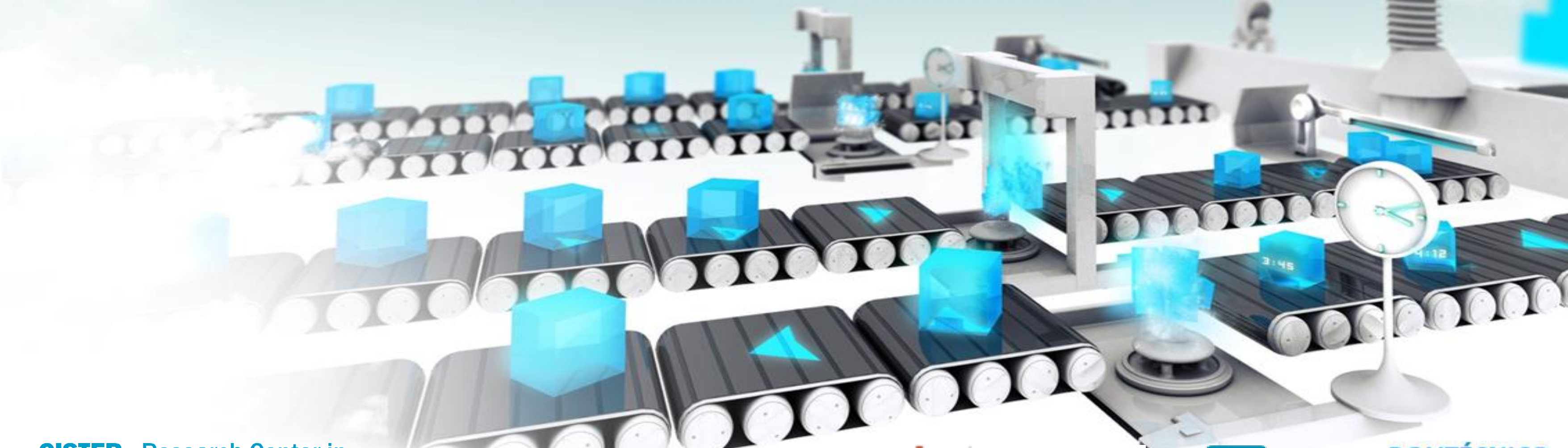
Towards Certifiable Resource Sharing in Safety-Critical Multi-Core Real-Time Systems

Benny Akesson
CISTER/INESC TEC and ISEP

Jan Nowotsch
Airbus Group Innovations



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

isep Instituto Superior de Engenharia do Porto

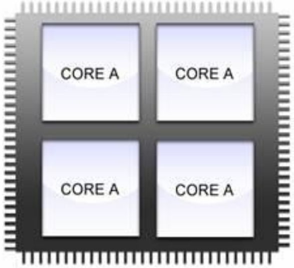
INESCTEC

POLITÉCNICO DO PORTO

Trends in Embedded Systems

- Embedded systems get **increasingly complex**
  - Increasingly complex applications (more functionality)
  - Growing number of applications integrated in a device
  - More applications execute concurrently
  - Requires increased system performance without increasing power

- The resulting complex platforms
  - are (heterogeneous) **multi-core systems** to improve performance/power ratio
  - Resources in the system are **shared** to reduce cost



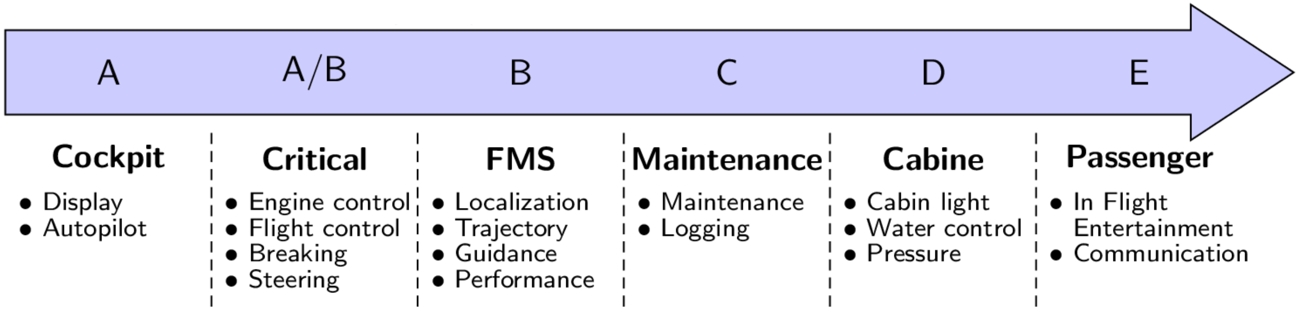
Safety-Critical Systems

→Some applications have **real-time requirements**

→Applications have different **design assurance levels** (DAL)

– DAL level determines required certification effort [1,2]

– High DAL levels are very expensive and time-consuming to certify



A A/B B C D E

Cockpit
- Display
- Autopilot

Critical
- Engine control
- Flight control
- Breaking
- Steering

FMS
- Localization
- Trajectory
- Guidance
- Performance

Maintenance
- Maintenance
- Logging

Cabine
- Cabin light
- Water control
- Pressure

Passenger
- In Flight Entertainment
- Communication

→Commercial-of-the-shelf (COTS) platforms are used

– Custom hardware not cost-effective with low volumes

[1] DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2012
[2] DO-254 Design Assurance Guidance for Airborne Electronic Hardware, 2000

Resource Sharing

- Increased integration implies **mixed-criticality systems**
  - Applications with different DALs share resources

- Resource sharing creates **interference** between applications
  - Makes it difficult to derive WCET of applications
  - Highest DAL of applications must be used unless there is **isolation** [1]

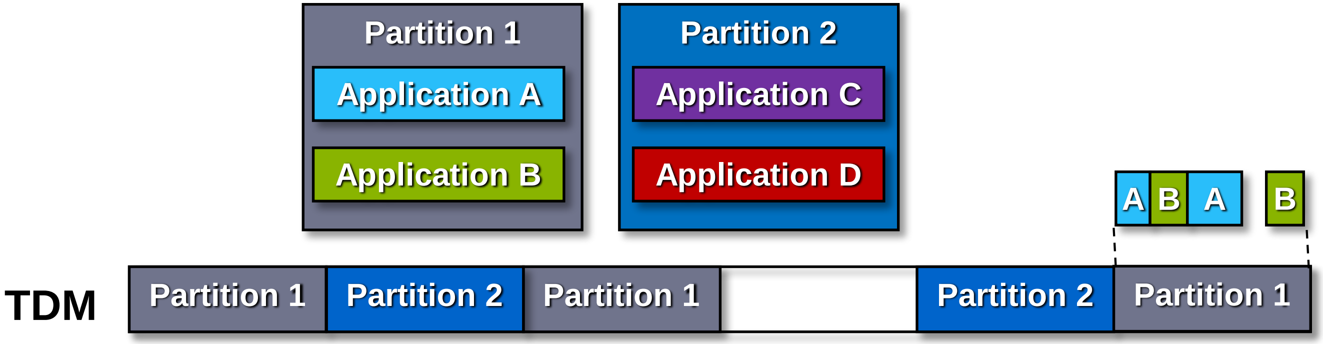
- Both **temporal and spatial isolation** is required [1]
  - Applications must be ”**sufficiently**” independent

[1] DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2012

Single-Core Isolation

- Isolation on single core is typically provided by operating system
  - E.g. based on ARINC-653 specification [3]
  - **”Robust” partitions** created for sets of applications

- Temporal isolation using time-division multiplexing (TDM)
  - TDM non-work-conserving (nwc) to eliminate interference
  - Application-level scheduling within a partition



[3] ARINC Specification 653, 2010

Problem Statement

How to ensure that applications sharing resources are isolated and that WCET of applications can be computed in certifiable mixed-criticality multi-core systems?

This presentation discusses this problem in a survey-like manner

Presentation Outline

Introduction

Time-Predictable Hardware

COTS Analysis Methods

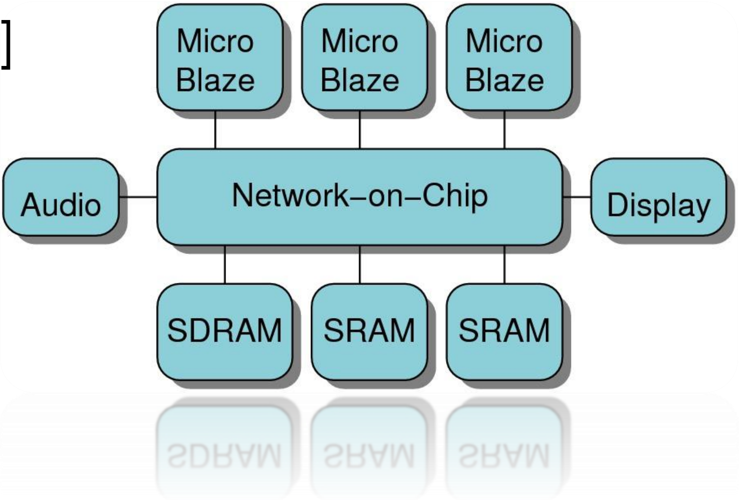
Airbus isWCET Approach

Conclusions

CompSoC Platform

- →**CompSoC** is a platform for real-time applications [4]
  - For independent app. development, verification, and execution

- →Components of **tiled architecture** [5]
  - Processor tiles with MicroBlaze cores
  - Æthereal network-on-chip
  - Memory tiles with SRAM or SDRAM
  - Peripheral tiles



- →Platform implementation in VHDL [6]

[4] http://compsoc.eu
[5] Goossens, Kees, et al. "Virtual execution platforms for mixed-time-criticality systems: The compsoc architecture and design flow." *SIGBED Review* 10.3, 2013.
[6] Goossens, Sven, et al. "The CompSOC design flow for virtual execution platforms." *Proceedings of the 10th FPGAworld Conference*. ACM, 2013.

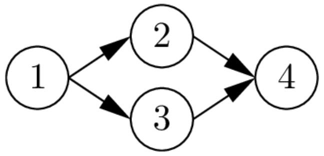
CompSoC

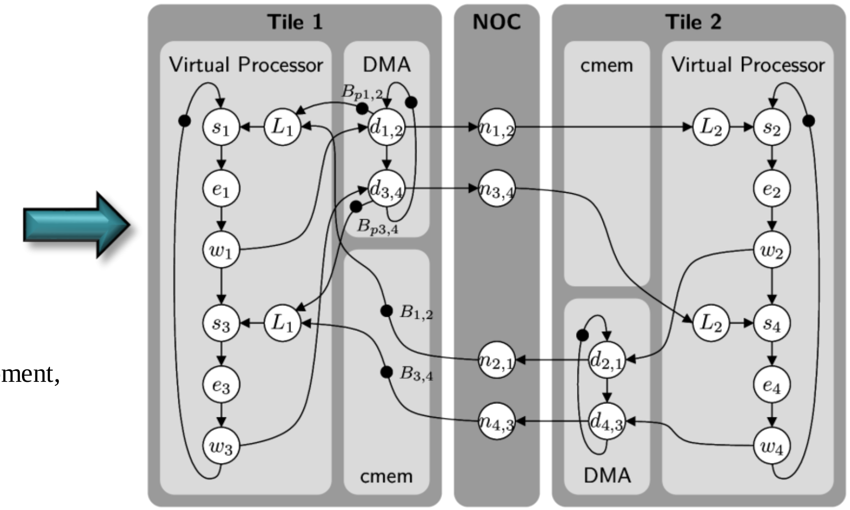
→**All resources are shared** [7]

- NWC TDM partition scheduling on CPU (ARINC-653)
- NWC pipelined TDM flit scheduling in network-on-chip
- NWC TDM trans. scheduling or any scheduler + delay for DRAM

→Performance analysis

- Data-flow models for all software/hardware components
- WCET for all tasks/transactions

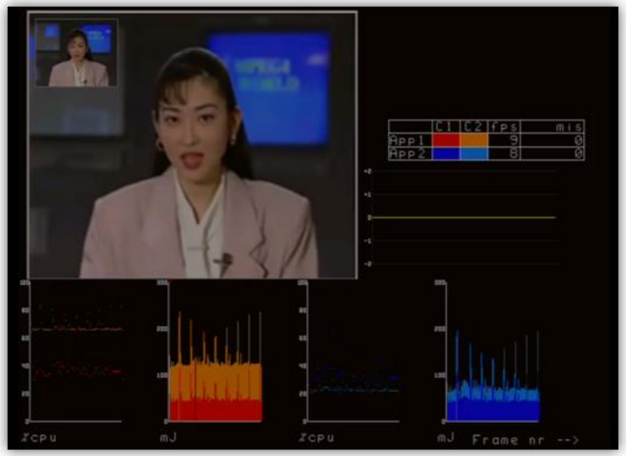




[7] Akesson, Benny, et al. "Composability and predictability for independent application development, verification, and execution." Chapter in *Multiprocessor System-on-Chip*, 2011.

Extreme Partitioning

- **Extremely robust partitioning** [7]
  - **Not a single cycle interference** from other partitions
  - Similar to PREcision-Timed Architectures (PRET) [8]



	C1	C2	fps	mis
App1	🟥	🟧	9	0
App2	🟦	🟦	8	0

[7] Akesson, Benny, et al. "Composability and predictability for independent application development, verification, and execution." Chapter in *Multiprocessor System-on-Chip*, 2011.
[8] Edwards, Stephen A., and Edward A. Lee. "The case for the precision timed (PRET) machine." *Proc. DAC*, 2007.

Summary

- It is possible to design time-predictable multi-core platforms
  - Extremely robust partitioning
  - WCET for all tasks/transactions, but
  - Average-case performance suffer

- Application domain is practically restricted to COTS platforms
  - Hardware is given
  - Transfering technology is very difficult
  - Most customers are oriented towards average-case performance

Presentation Outline

Introduction

Time-Predictable Hardware

COTS Analysis Methods

Airbus isWCET Approach

Conclusions

Modeling COTS Platforms

- Analytically modeling a COTS platform is very difficult
  - Hardware is optimized for average-case performance
  - No detailed documentation of implementation
  - Limited possibilities for measurements during validation
  - Difficult to guarantee correctness / conservativeness of model

- Often **pessimistic assumptions** about memory controller:
  - Unknown size of reorder buffer in memory controller [9]
  - Unknown work-conserving memory scheduler [10,11,12]
  - Bounds still useful?

[9] Kim, Hyoseung, et al. "Bounding memory interference delay in COTS-based multi-core systems." *Proc. RTAS*, 2014.
[10] Dasari, Dakshina, et al. "Response time analysis of COTS-based multicores considering the contention on the shared memory bus." *Proc. TRUSTCOM*, 2011.
[11] Nowotsch, Jan, et al. "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement." *Proc. ECRTS*, 2014.
[12] Schliecker, Simon, and Rolf Ernst. "Real-time performance analysis of multiprocessor systems with shared memory." *ACM Transactions on Embedded Computing Systems (TECS)* 10.2 (2010): 22.

Interference Analysis

- There is much work on bounding interference between tasks
  - Vary w.r.t. task model and (task/transaction) schedulers

- Common assumptions
  - **Single outstanding transaction**
  - No or partitioned caches
  - **Different path of worst-case memory accesses** (WMA)

- Abstraction of memory accesses
  - Number of memory accesses per task / block [11,13]
  - Minimum / maximum requests in interval [12] (for self / others)

[11] Nowotsch, Jan, et al. "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement." *Proc. ECRTS*, 2014.
[12] Schliecker, Simon, and Rolf Ernst. "Real-time performance analysis of multiprocessor systems with shared memory." *ACM Transactions on Embedded Computing Systems (TECS)* 10.2 (2010): 22.
[13] Yun, Heechul, et al. "Memory access control in multiprocessor for real-time systems with mixed criticality." *Proc. ECRTS*, 2012.

Memory Throttling

- Throttling popular to control memory interference [11,14,15,16]
  - Can be implemented at operating system level
  - Relies on good performance monitoring counters

- Basic idea:
  1. Assign memory access budgets
  2. Monitor number of memory accesses using performance counters
  3. Enforce budget by suspending tasks with depleted budgets

- Optionally, there are mechanisms for slack distribution
  - Observed slack [15] or proven slack [16]

[11] Nowotsch, Jan, et al. "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement." *Proc. ECRTS*, 2014.
[14] Inam, Rafia, et al. "The Multi-Resource Server for predictable execution on multi-core platforms." *Proc. RTAS*, 2014.
[15] Yun, Heechul, et al. "Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms." *Proc. RTAS*, 2013.
[16] Nowotsch, Jan, and Michael Paulitsch. "Quality of service capabilities for hard real-time applications on multi-core processors." *Proc. RTNS*, 2013.

Schedulability Analysis

- →New scheduling theory on top of memory throttling [13,17]
  - Respecting both memory budget and CPU scheduling
- →Theory requires knowledge about memory access times
  - Commonly done by assumption
  - Sometimes by measurements on platform [11,13]
  - Never done using validated analytical model

[11] Nowotsch, Jan, et al. "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement." *Proc. ECRTS*, 2014.
[13] Yun, Heechul, et al. "Memory access control in multiprocessor for real-time systems with mixed criticality." *Proc. ECRTS*, 2012.
[17] Behnam, Moris, et al. "Multi-core composability in the face of memory-bus contention." *ACM SIGBED Review* 10.3 (2013): 35-42.

Measurement-based Approaches

- Measurement-based approaches offer **pragmatic solution**

- Possible to use measurement-based WCET tools
  - E.g. RapiTime
  - Measure your way around things you cannot model

- Stressing shared resources
  - Possible using synthetic resource stressing tasks [18,19]

[18] Nowotsch, Jan, and Michael Paulitsch. "Leveraging multi-core computing architectures in avionics." *Dependable Computing Conference (EDCC), 2012 Ninth European*. IEEE, 2012.
[19] Radojković, Petar, et al. "On the evaluation of the impact of shared resources in multithreaded COTS processors in time-critical environments." *ACM Transactions on Architecture and Code Optimization (TACO)* 8.4 (2012): 34.

Presentation Outline

Introduction

Time-Predictable Hardware

COTS Analysis Methods

Airbus isWCET Approach

Conclusions

isWCET Analysis I

→Setup

- Freescale P4080 multi-core platform
- SYSGO Pike OS operating system
- AbsInt aiT static analysis tool
- EEMBC Automotive benchmarks

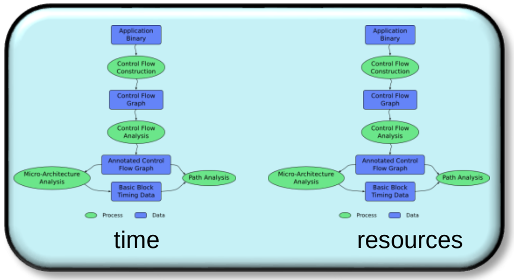
→Approach [11]

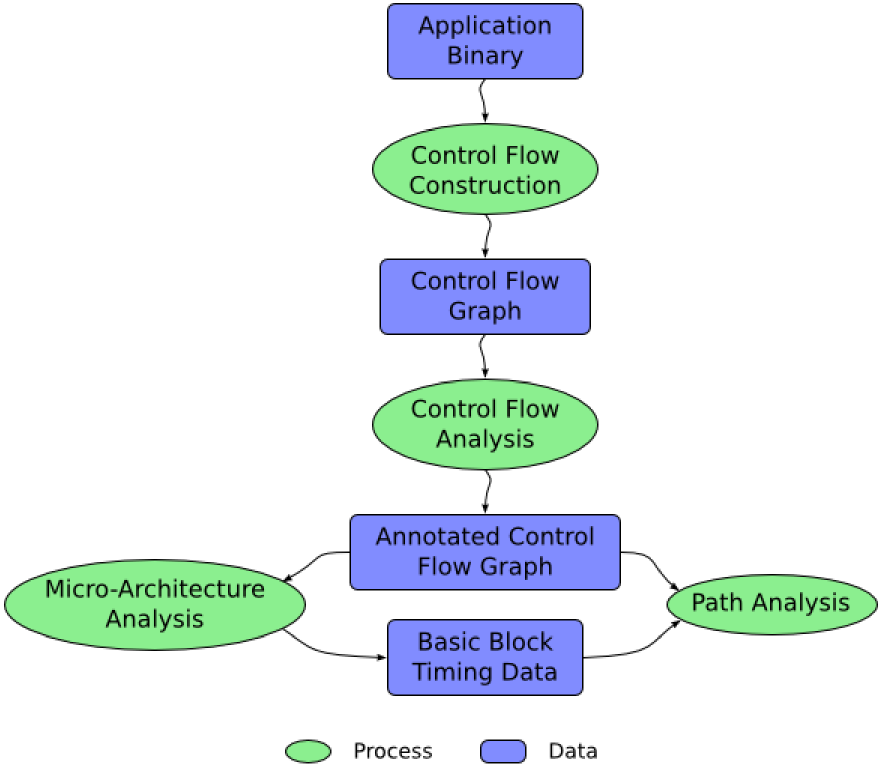
- Individual core-local and interference analyses
- Separation of timing and resource analyses

[11] Nowotsch, Jan, et al. "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement." *Proc. ECRTS*, 2014.

isWCET Analysis II

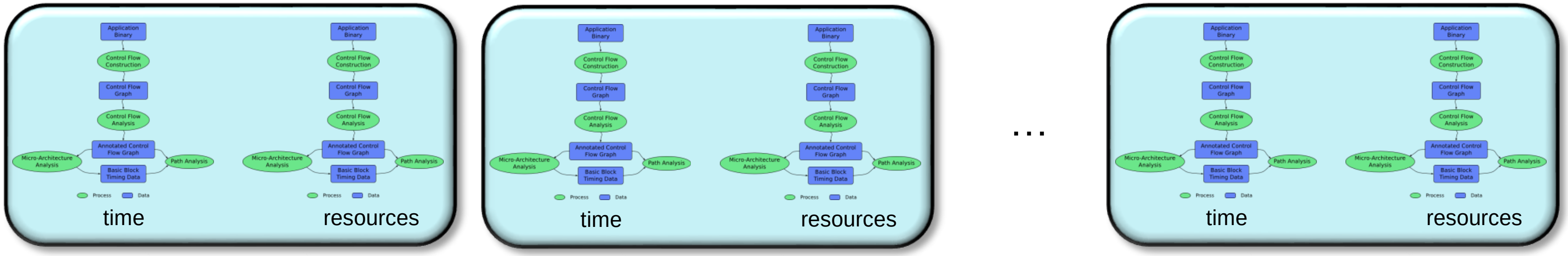
→Core-local Analysis





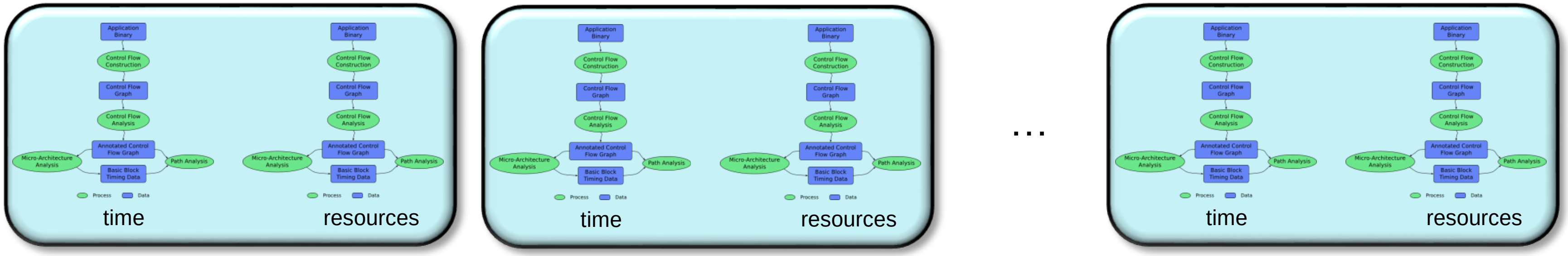
isWCET Analysis II

→Core-local Analysis

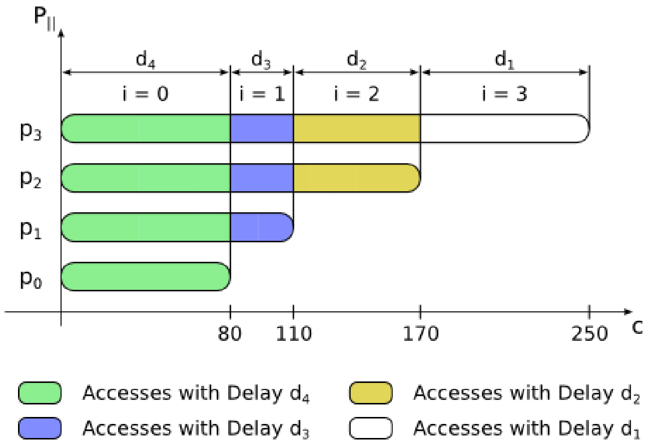


isWCET Analysis II

→Core-local Analysis



→Interference Analysis



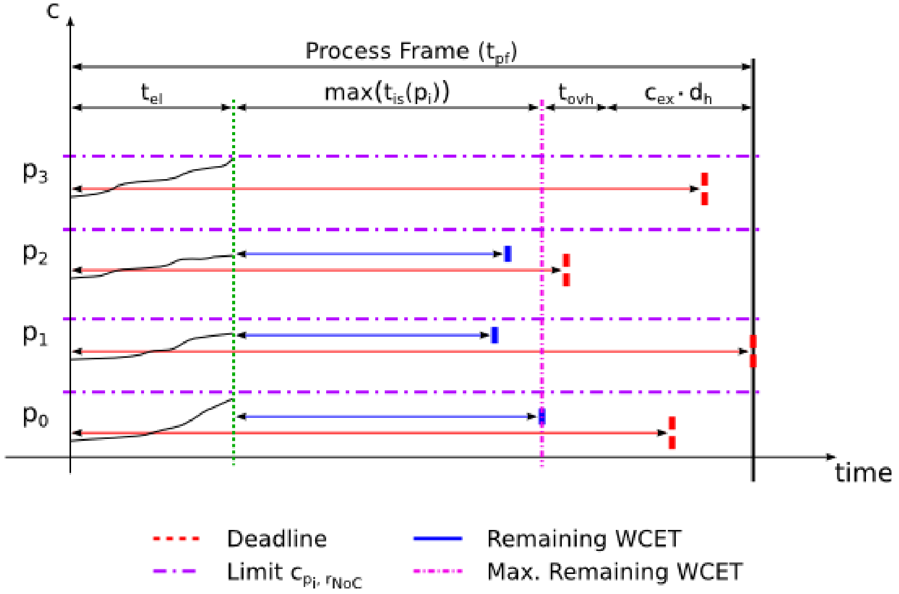
Evaluation

→Comparison to intuitive approaches (minimum t_{min} and maximum t_{max} contention)

Benchmark	T_{min}[ms]	T_{max}[ms]	T_{is}[ms]
cacheb	114	1996	493
iirflt	60	136	116
rspeed	233	4468	612
a2time	29	524	231
bitmnp	154	262	225
tblook	122	449	289
matrix	21	35	32
aifftr	11	11	11

Run-time Adaption

- Dynamic adaptation of resource budgets based on actual system progress
- Progress determined through monitoring



[16] Nowotsch, Jan, and Michael Paulitsch. "Quality of service capabilities for hard real-time applications on multi-core processors." *Proc. RTNS*, 2013.

Presentation Outline

Introduction

Time-Predictable Hardware

COTS Analysis Methods

Airbus isWCET Approach

Conclusions

Conclusions

- Increased integration drives transition to multi-core platforms
  - Resource sharing causes **interference** between applications
  - Nightmare w.r.t. certification
  - Problem to **isolate sharing applications and safely determine WCET**

- Time-Predictable Platforms have been demonstrated
  - Extremely robust partitioning and easy to determine WCET
  - Difficult to get commercial uptake of technology

- Analysis of COTS systems active research topic
  - Difficult to model analytically due to lacking openness
  - Community is finding the right models/abstractions
  - Most models remain unvalidated
  - Alternative is to use measurement-based techniques