Ensuring non-functional safety properties on embedded multicore systems

Simon Wegener, AbsInt Angewandte Informatik GmbH

Outline

- Introduction

  Why are we interested in non-functional properties?
- Timing

  Why are multicores hard to analyze?
- Memory

  How can we ensure safety properties by construction?
- Conclusion

AbsInt

AbsInt Angewandte Informatik GmbH

- Provides advanced development tools for embedded systems, and tools for validation, verification, and certification of safety-critical software.
- Founded in February 1998 by six researchers of Saarland University, Germany, from the group of programming languages and compiler construction of Prof. Dr. Dr. h.c. mult. R. Wilhelm.
- Privately held by the founders.
- 45 + employees.
- Selected customers:

AbsInt

Key Products

Astrée

- Detects all runtime errors, data races, deadlocks, and other critical errors

RuleChecker

- Checks coding guidelines (MISRA, CERT, ...)

CompCert

- Formally verified optimizing C compiler

StackAnalyzer

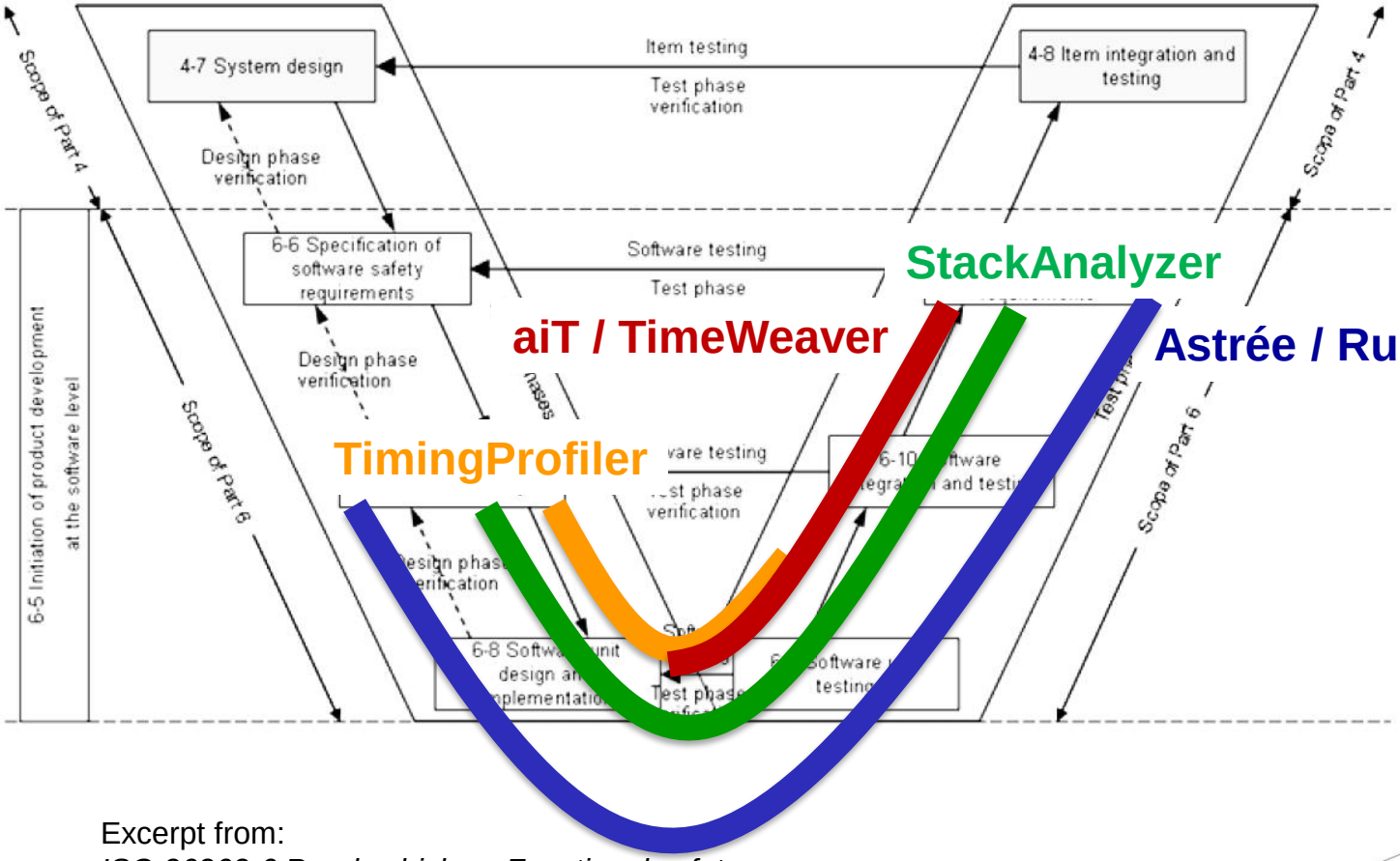
- Safe upper bounds on maximal stack: **no more stack overflows**

Timing Analysis Solutions

- aiT WCET Analyzer: Safe upper bounds on worst-case execution time: **timing guarantees**
- TimeWeaver: Hybrid worst-case timing analysis for High-End CPUs: **combines tracing with static analysis**
- TimingProfiler: Timing estimates: **continuous timing feedback** and optimization during early SW-development

AbsInt

Development Process



Excerpt from:
ISO 26262-6 Road vehicles - Functional safety –
Part 6: Product development: Software Level, 2011.

AbsInt

Functional Safety

- Demonstration of functional correctness
  - Well-defined criteria
  - ➢ Automated and/or model-based testing
  - ➢ Formal techniques: model checking, theorem proving

Required by DO-178B / DO-178C / ISO-26262, EN-50128, IEC-61508

- Satisfaction of safety-relevant non-functional requirements
  - No runtime errors (e.g. division by zero, overflow, invalid pointer access, out-of-bounds array access)
  - Resource usage:
    - Timing requirements (e.g. WCET, WCRT)
    - Memory requirements (e.g. no stack overflow)

Required by DO-178B / DO-178C / ISO-26262, EN-50128, IEC-61508

  - Robustness / freedom of interference (e.g. no corruption of content, incorrect synchronization, illegal read/write accesses)
  - ➢ Insufficient: Tests & Measurements
    - No specific test cases, unclear test end criteria, no full coverage possible
    - "Testing, in general, cannot show the absence of errors." [DO-178B]
  - Formal technique: abstract interpretation.

AbsInt

Trends in Automotive

- Code size explosion
  - LOC per car 2010: up to 10 million
  - LOC per car 2016: up to 150 million (factor 15)
- Software becomes key value proposition
  - Cars in transition from hardware driven machines to software-driven electronic devices
- Snowballing complexity is causing significant software-related quality issues.
- Quality and security of vehicle software and electronics are key requirements to guarantee safety.


McKinsey&Company

MCKINSEY CENTER FOR FUTURE MOBILITY

RETHINKING CAR SOFTWARE AND ELECTRONICS ARCHITECTURE

February 2018

As the car continues its transition from a hardware-driven machine to a software-driven electronics device, the auto industry's competitive rules are being rewritten.

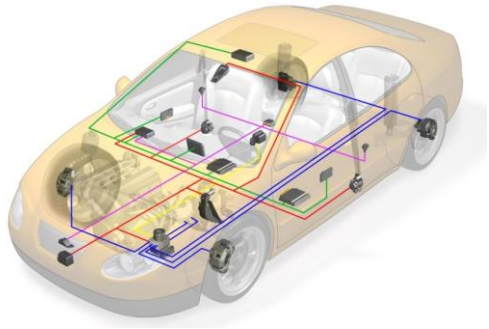
Excerpts from:
Ondrej Burkacky, Johannes Deichmann, Georg Doll, Christian Knochenhauer. Rethinking car software and electronics architecture. Report McKinsey & Company, Feb. 2018.

AbsInt

PART I – Timing Analysis

Safety-Critical Hard Real-Time Software

- Controllers in planes, cars, plants, … are expected to finish their tasks within reliable time bounds.
- Timing analysis must be performed.



AbsInt

Automotive: ISO-26262

Table 1 — Topics to be covered by modelling and coding guidelines

	Topics	ASIL A	ASIL B	ASIL C	ASIL D
1a	Enforcement of low complexity	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++

[b] The objectives of method 1b are

- Exclusion of ambiguously defined language constructs which might be interpreted differently by different modellers, programmers, code generators or compilers.
- Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.
- Exclusion of language constructs which might result in unhandled run-time errors.

Criticality levels:
A(lowest)
to
D (highest)

7.4.17 An upper estimation of required resources for the embedded software shall be made, including:

a) the execution time;

b) the storage space; and

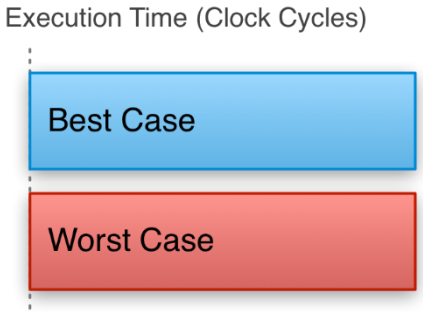
Excerpt from:
ISO 26262-6 Road vehicles - Functional safety – Part 6: Product development: software level, 2011

AbsInt

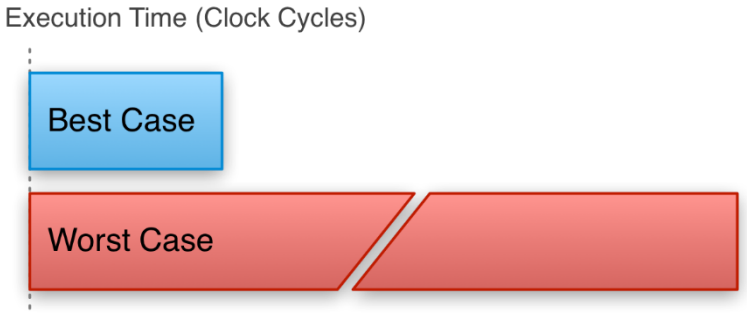
Execution Time Variablility (Singlecore)

```
LOAD    r2, _a
LOAD    r1, _b
ADD     r3,r2,r1
```

1990: 68020



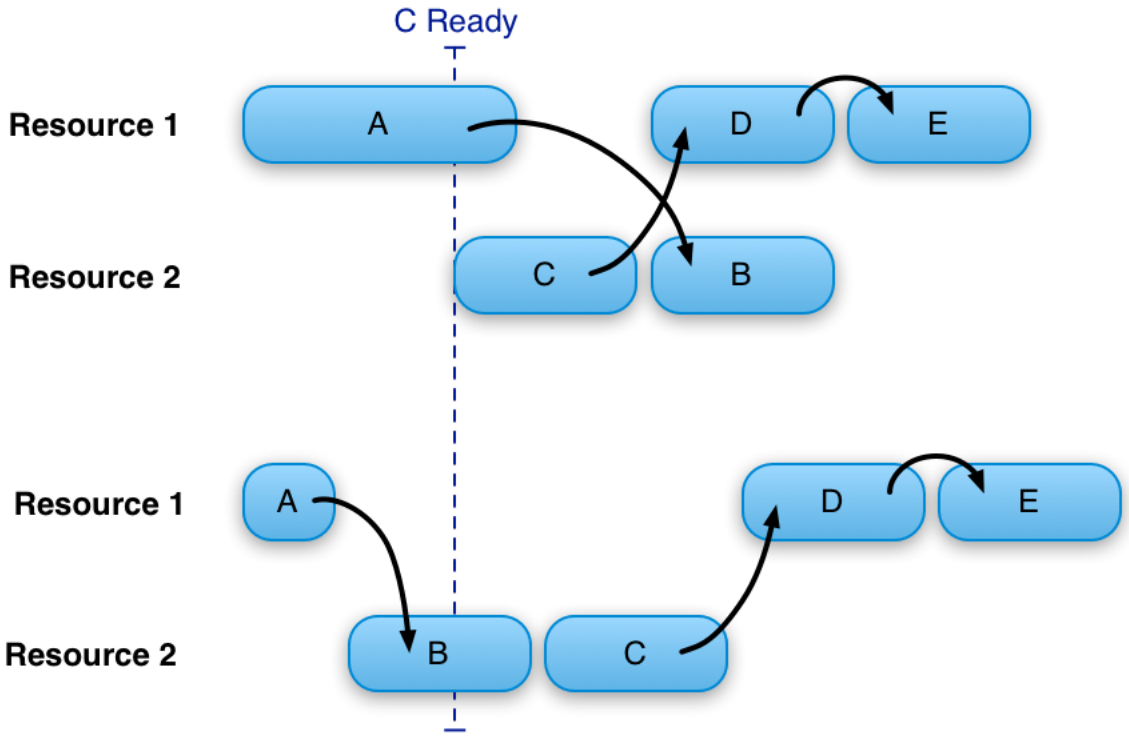
2001: MPC755



Up to a factor of 100 between best-case and worst-case!

AbsInt

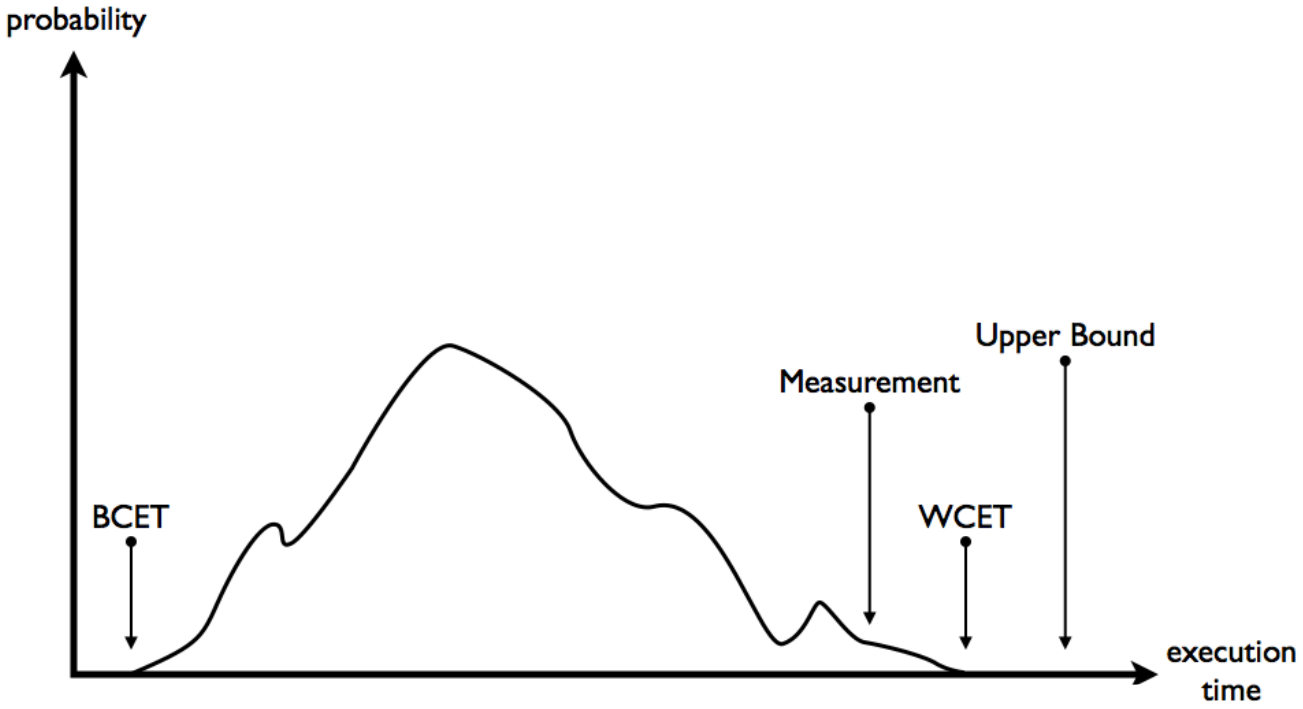
Timing Anomalies



Timing anomaly due to scheduling effects. When A hits the cache, the overall execution time is longer than in case of a cache miss.

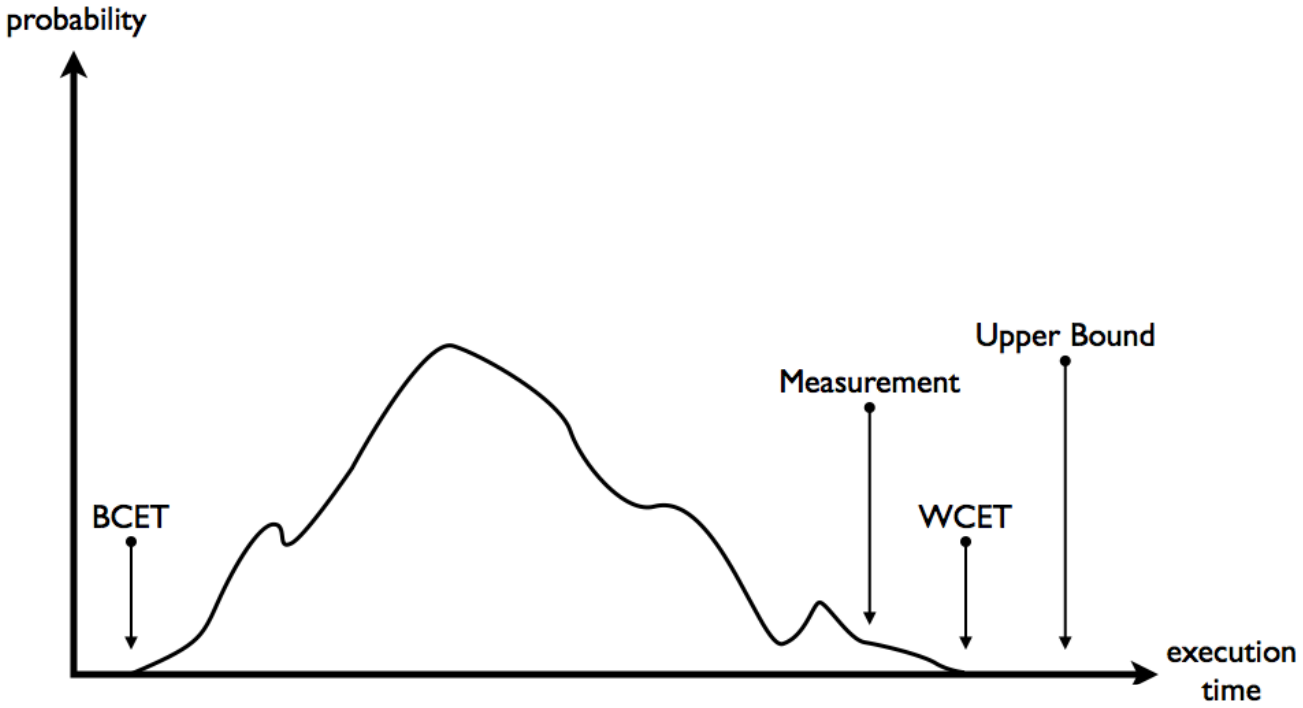
AbsInt

The Timing Problem



AbsInt

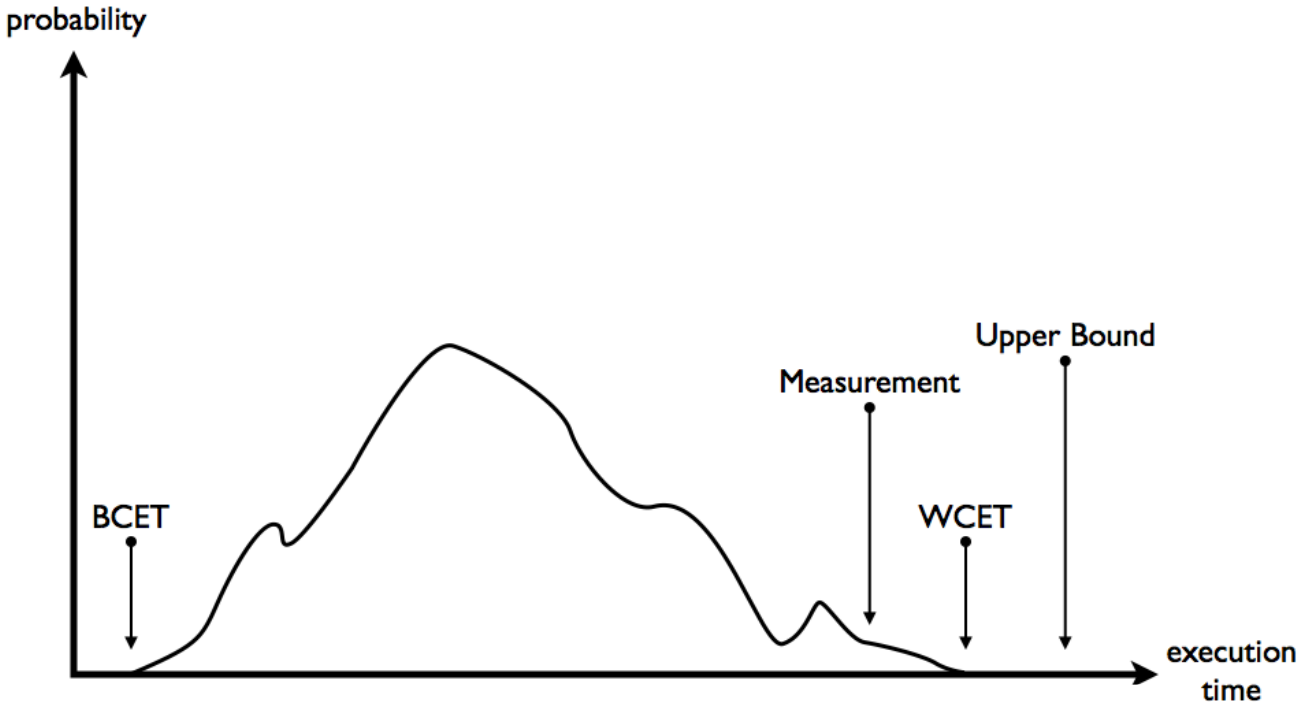
The Timing Problem



End-to-end measurements usually do not cover the worst case!

AbsInt

The Timing Problem

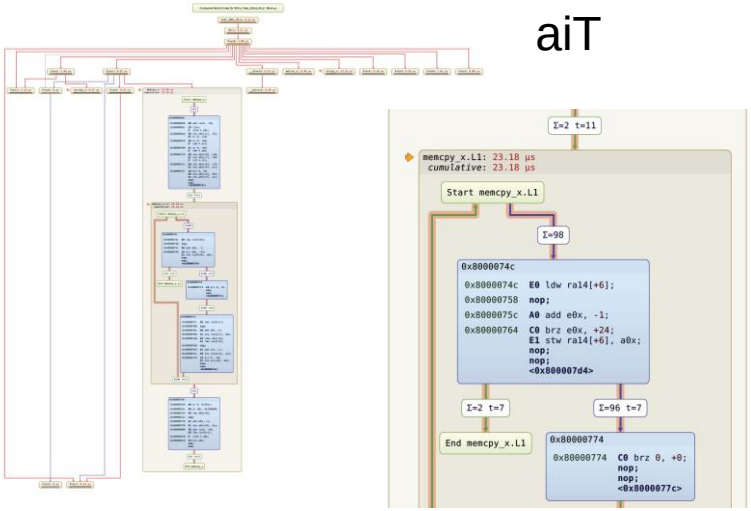


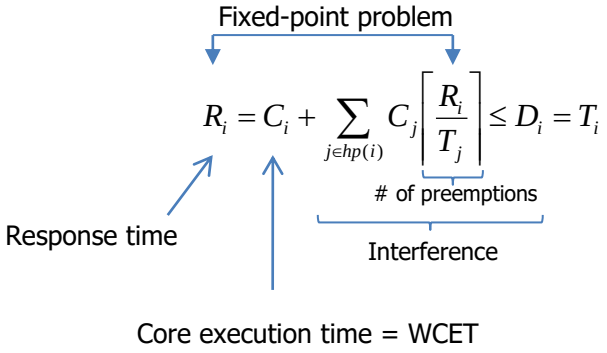
Static WCET analysis may produce unsatisfactory results for unpredictable architectures

AbsInt

Two Levels of Timing Analysis

- Code level
  - Single process, task, ISR
  - Focus on
    - Control flow
    - Processor architecture with pipelines and caches
    - WCET
- System level
  - Multiple functions or tasks
  - Focus on
    - Integration and scheduling
    - End-to-end timing
    - Worst-Case Response Time (WCRT)





AbsInt

Problem Solved?

Reinhard Wilhelm et al.:
The Worst-case Execution Time Problem
— Overview of Methods and Survey of Tools

CONCLUSIONS

„The problem of determining upper bounds on execution times for single tasks and for quite complex processor architectures has been solved. Several commercial WCET tools are available and have experienced very positive feedback from extensive industrial use.“

ACM Transactions on Embedded Computing Systems, Vol. 7, No. 3, April 2008.

AbsInt

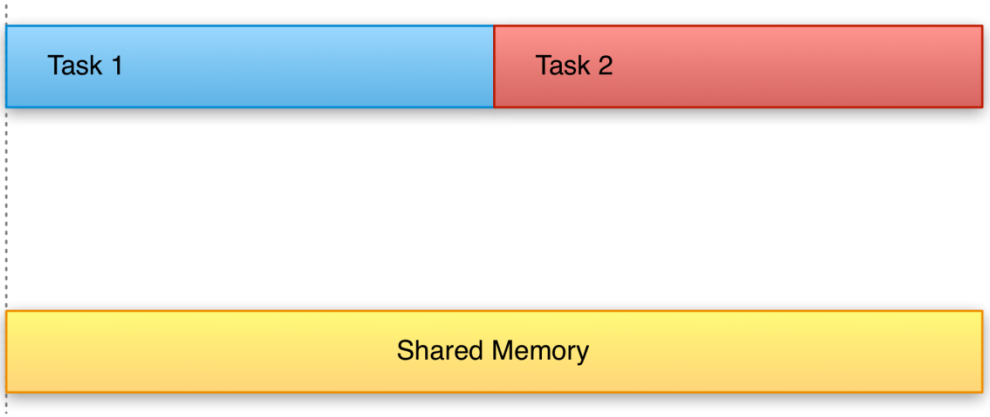
Problem Solved? No!

- The statement only addressed singlecore architectures.
- Many publications concerning multicore timing analysis, and several research projects investigated multicore timing analysis:
  - ARAMiS
  - ARAMiS II
  - ARGO
  - ASSUME
  - CERTAINTY
  - CONIRAS
  - PREDATOR
  - T-CREST
  - …

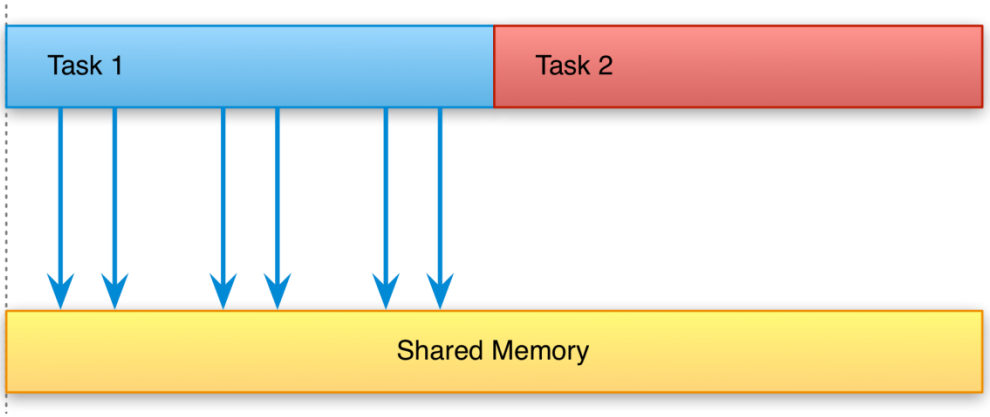
PART I – Timing Analysis

Resource Conflicts

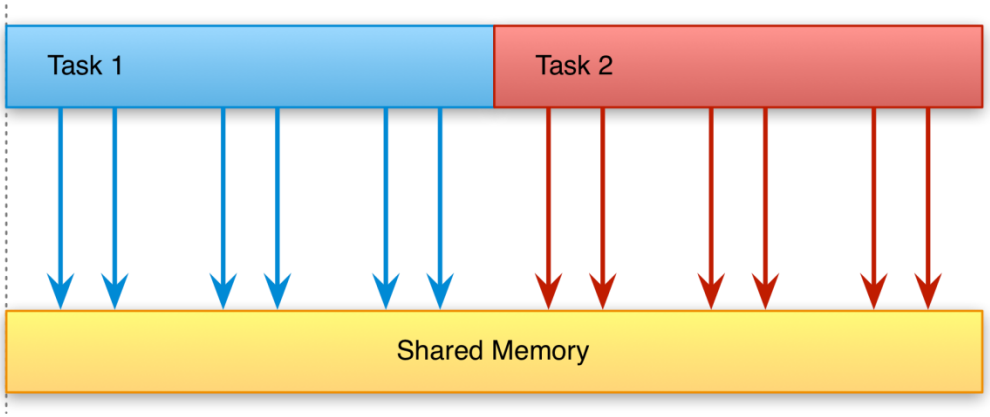
Singlecore



Singlecore

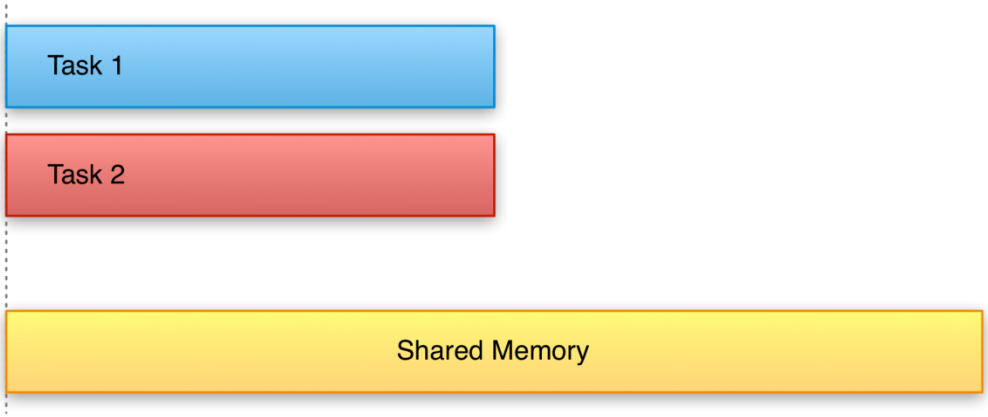


Singlecore



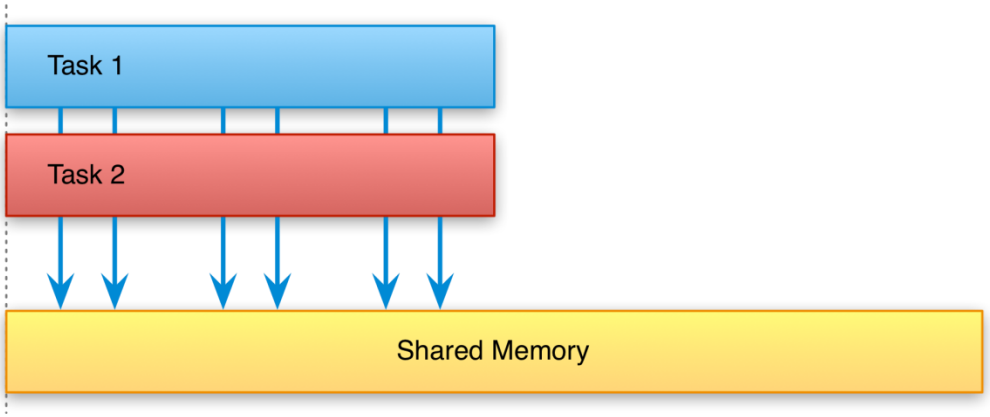
AbsInt

Multicore



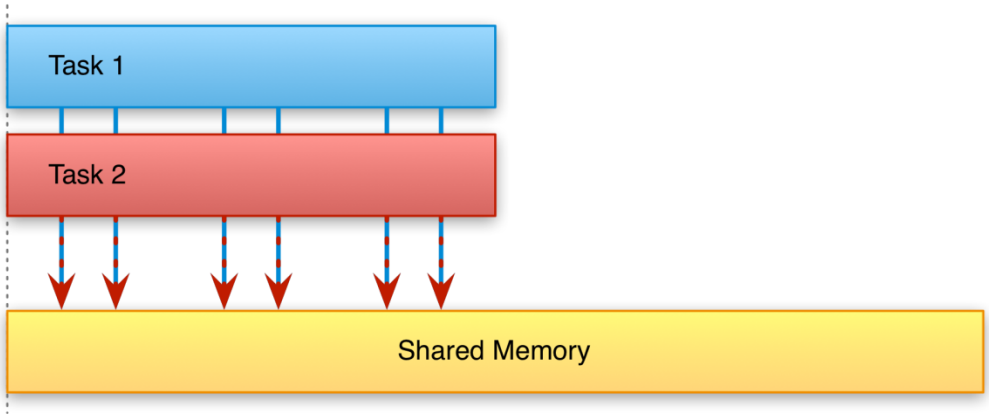
AbsInt

Multicore



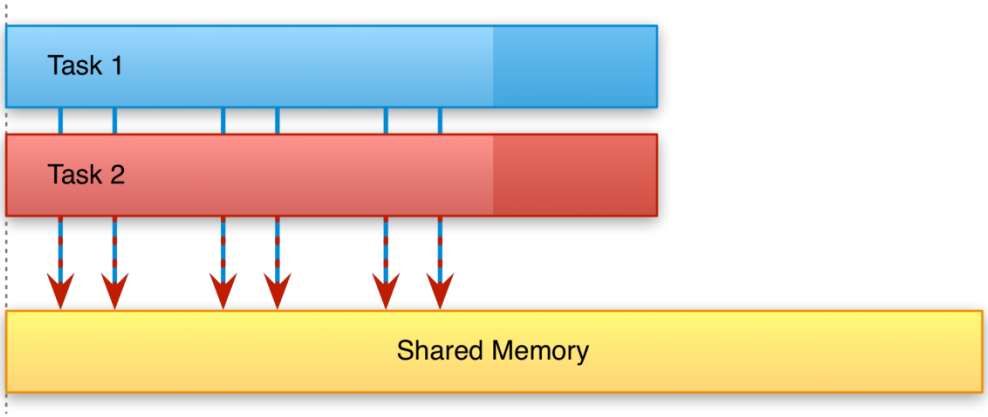
AbsInt

Multicore with Resource Conflicts



AbsInt

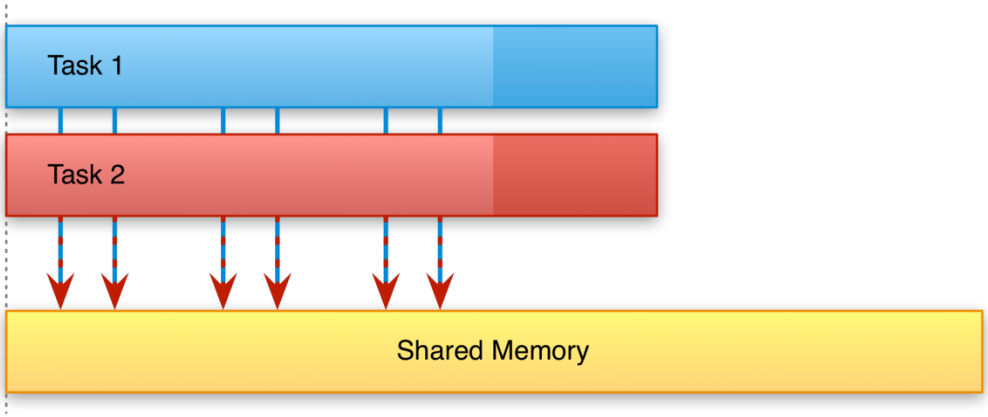
Multicore with Resource Conflicts



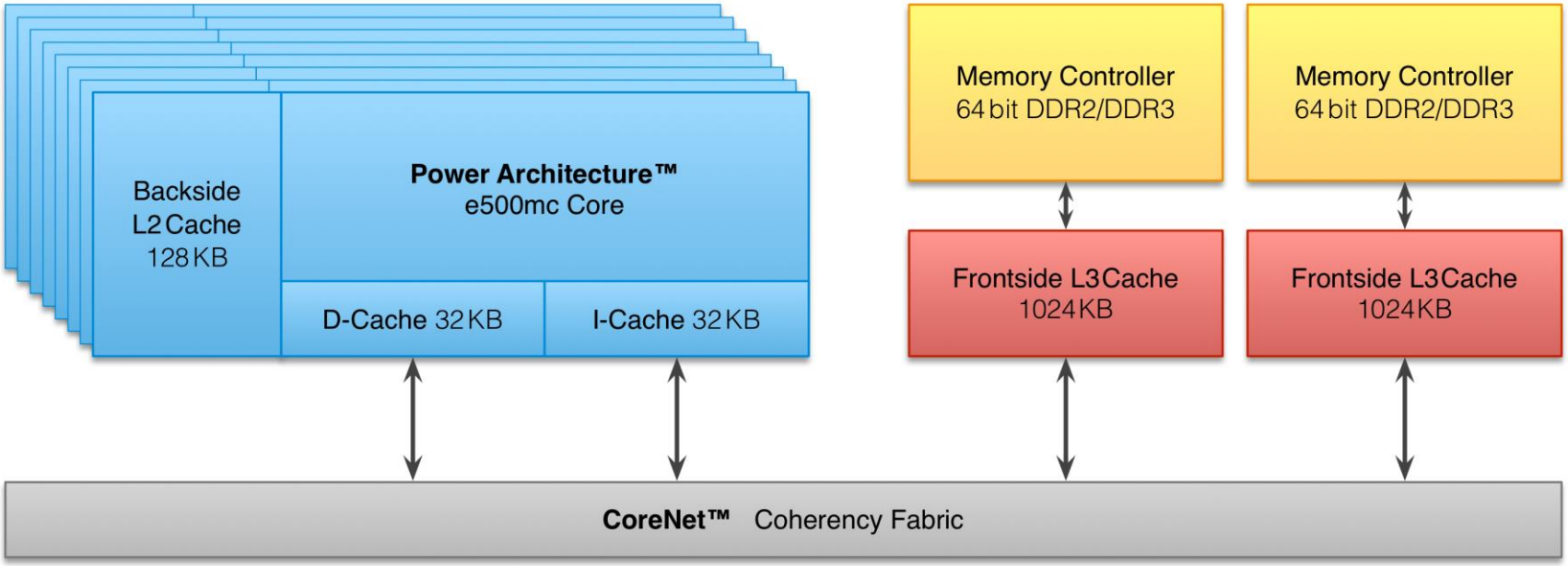
AbsInt

Problem: Resource Conflicts

- Any sound multicore WCET analysis must take interference delays into account!



Freescale QorIQ P4080



AbsInt

Freescale P4080 Access Latencies

- Derived through measurements

  Jan Nowotsch et al. *Multi-core interference-sensitive WCET analysis leveraging runtime resource capacity enforcement.* ECRTS 2014

TABLE I. P4080 MEMORY ACCESS LATENCIES FOR INCREASING NUMBER OF CONCURRENT CORES. LATENCIES USED FOR EVALUATION ARE MARKED BOLD.

	latency $[cycles]$							
cores	1	2	3	4	5	6	7	8
read	**41**	75	171	269	296	439	460	604
write	39	**164**	**245**	**463**	**517**	**737**	**784**	**1007**

- Write latency increased by 2550 % if all cores try to write concurrently to main memory

AbsInt

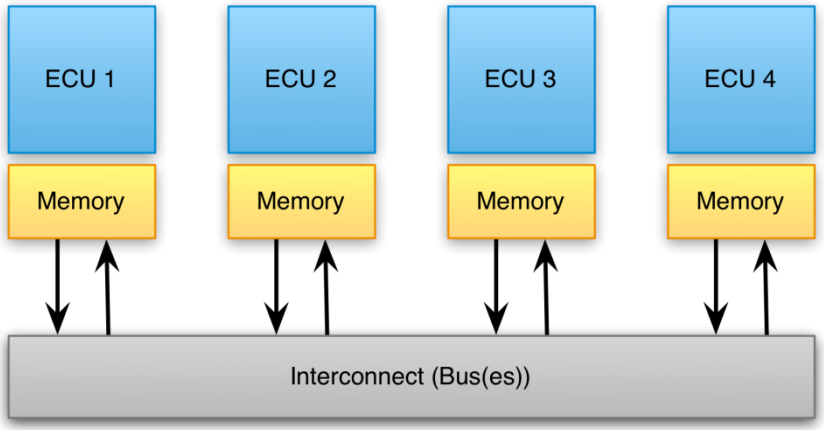
Assuming Full Interference

- e500mc core in very deterministic configuration
- Small benchmark program
- Measured (single core, no interference)
  - 610 ms
- Measured (all cores active)
  - 750 ms
- Safe WCET bound from aiT (single core)
  - 628 ms
- Safe WCET bound from aiT (assuming full interference)
  - 4571 ms

- Huge overestimation, safe WCET bound exceeds deadline

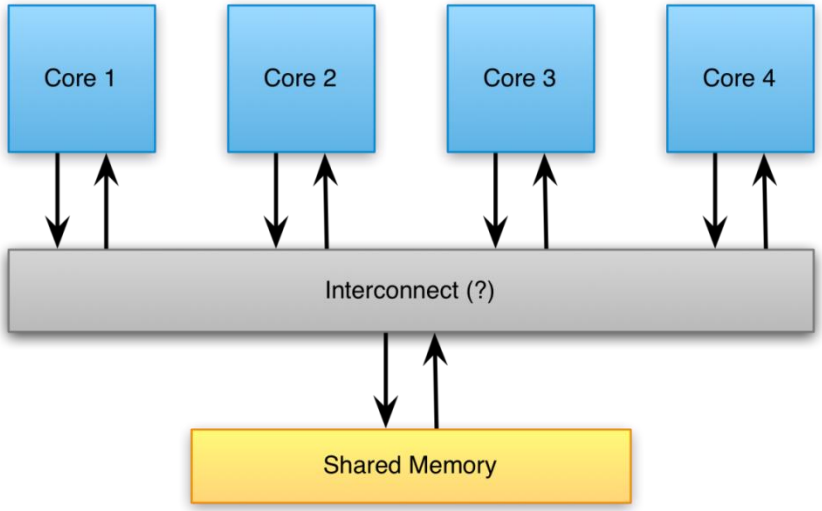
Architectural Differences

- Typical Embedded Multi-processing Platform



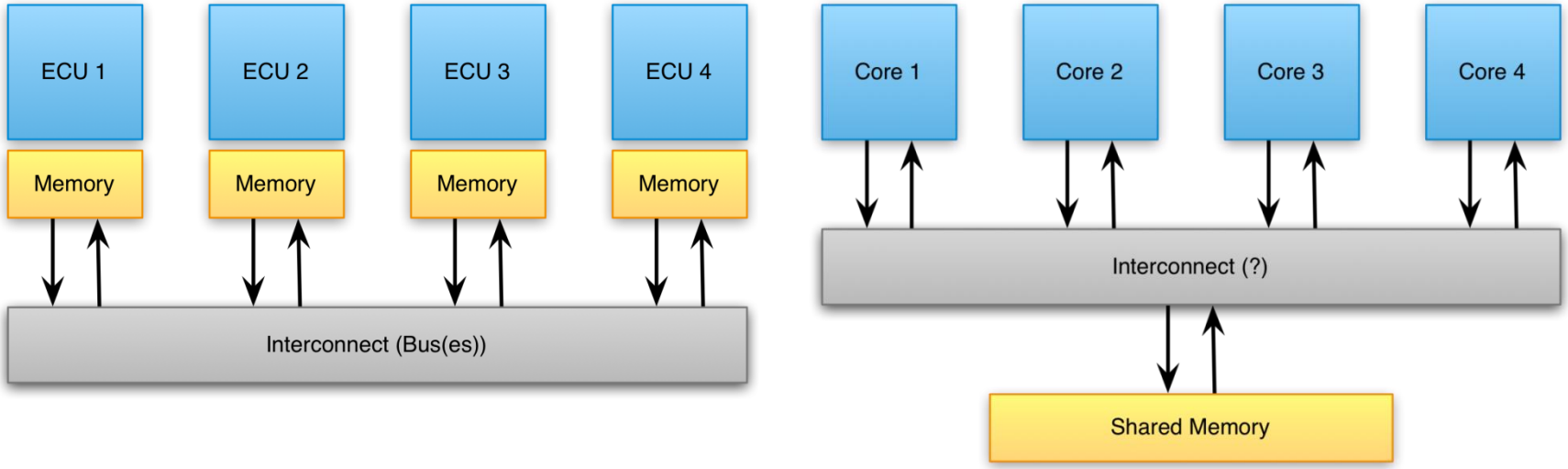
Architectural Differences

- Typical COTS Multi-core System



Architectural Differences

- 3 – 4 orders of magnitude more resource conflicts due to shared memory



AbsInt

PART I – Timing Analysis

Analysis Techniques

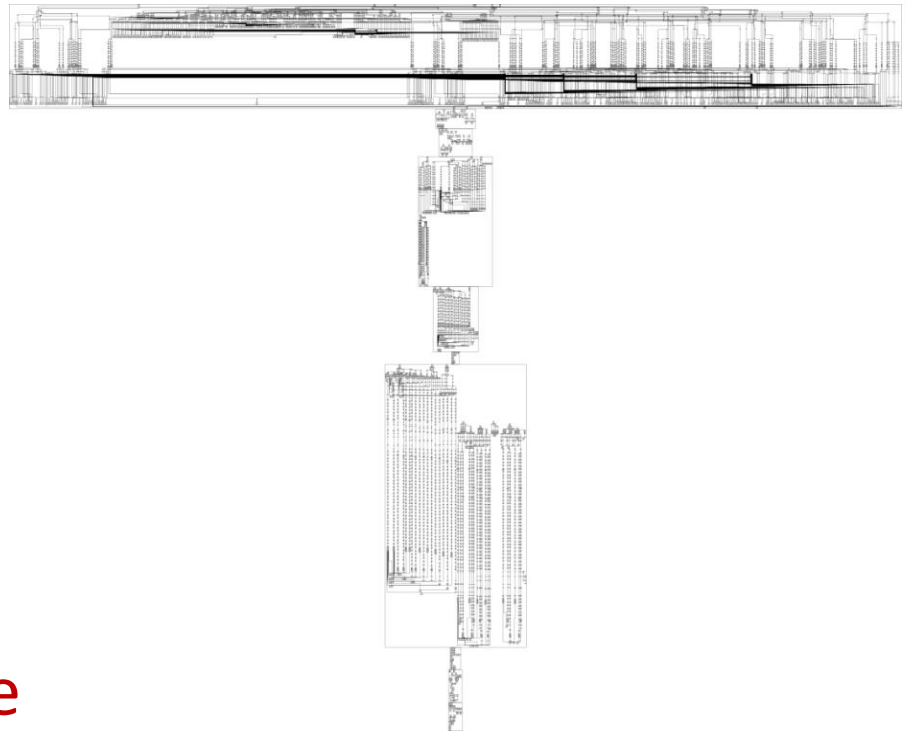
AbsInt

Analysis techniques

- Joint analysis (integrated code-level/system-level analysis) vs.
  separate WCET analysis for each core
- Static methods vs. measurement-based/hybrid methods
- (Probabilistic methods)

Joint Analysis

- Simultaneous analysis of concurrently running tasks
- But: pipeline state graph of one basic block may contain already several thousand states when computing the singlecore WCET
- Computationally not feasible due to huge state space

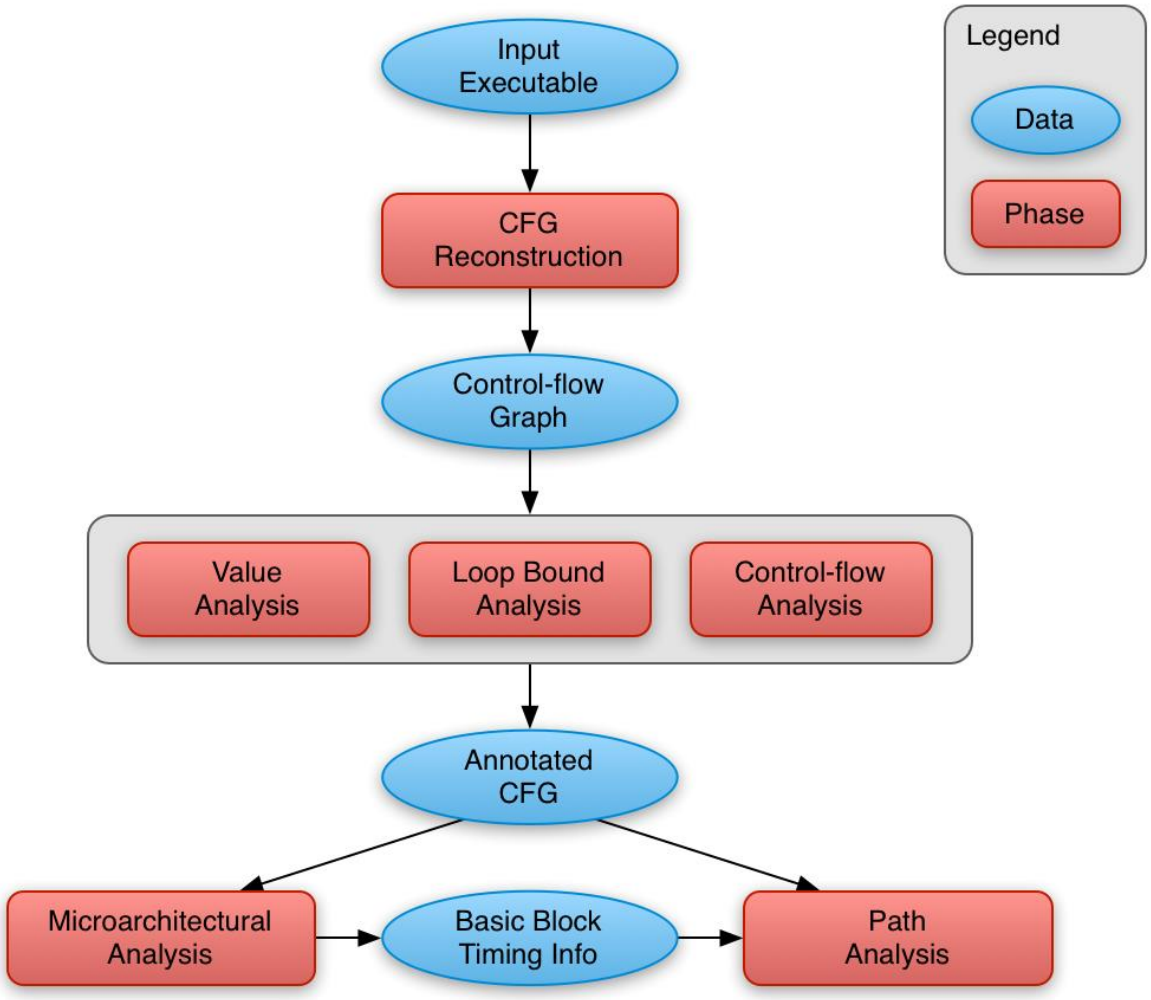


AbsInt

Separate Analysis

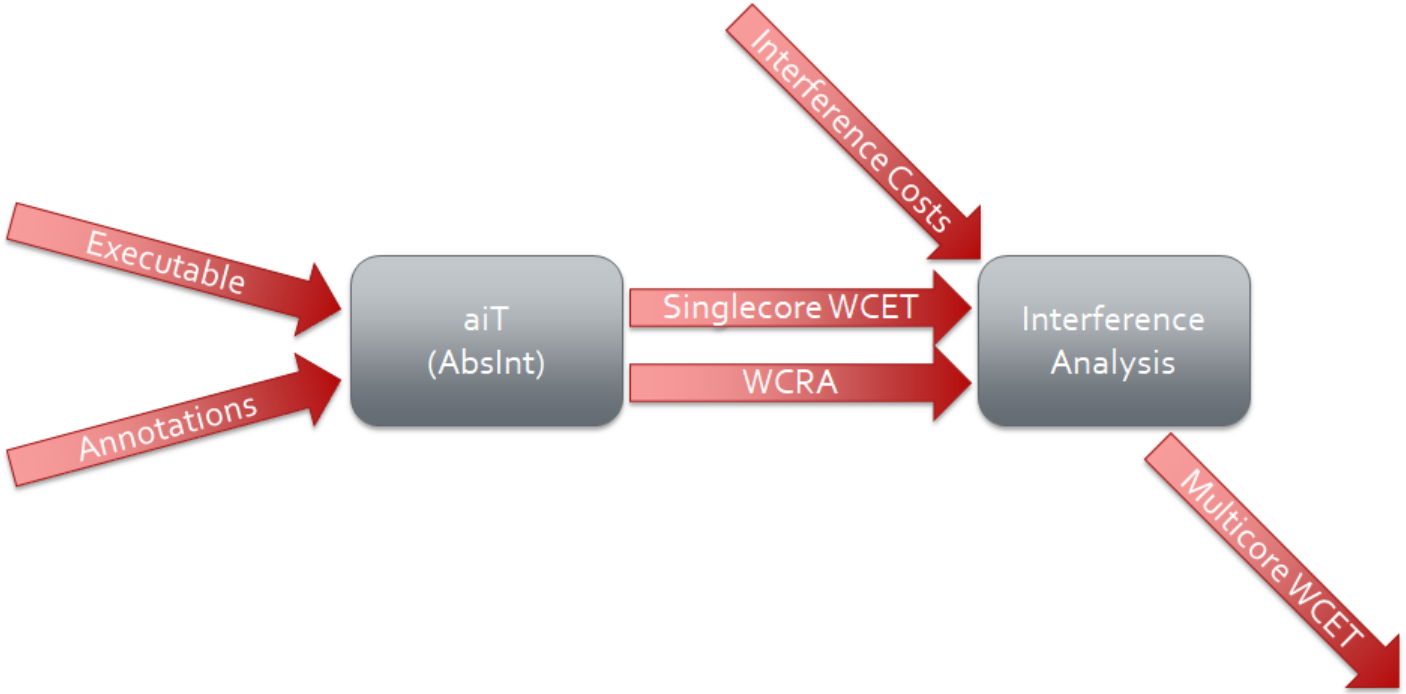
- Use singlecore WCET analysis, add interference costs in extra step
- Computationally easier
- But: we need to take care for non-timing-compositional architectures
- Idea for memory accesses:
  - Since memory accesses are orders of magnitude slower than normal instructions, one can argue that the pipeline will drain during the processing of memory accesses.
  - Thus, the interference delays imposed by resource conflicts do not cause timing anomalies and can be added later to the singlecore WCET bound.

aiT (Singlecore)



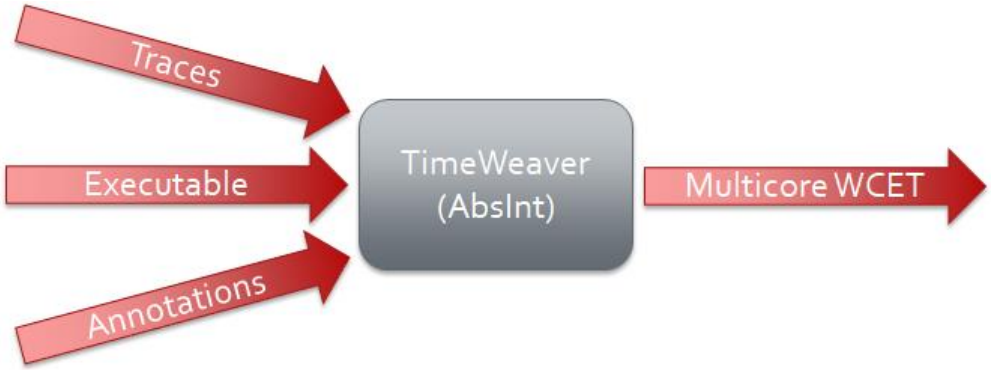
AbsInt

aiT (Multicore)



AbsInt

TimeWeaver (Multicore)



AbsInt

Hybrid WCET Analysis

- Combines static analysis and hardware measurements
- Computes WCET estimate based on
  - Execution times from traces and
  - Static value & worst-case path analysis
- Observed interferences are automatically taken into account
  $\Rightarrow$ by ist very nature a joint analysis

Probe Effect

- Caveat: probe effect!
  Measurements distorted by effects of code instrumentation

NOTE 3 If instrumented code is used to determine the degree of coverage, it can be necessary to show that the instrumentation has no effect on the test results. This can be done by repeating the tests with non-instrumented code.

9.4.6 The test environment for software unit testing shall correspond as closely as possible to the target environment. If the software unit testing is not carried out in the target environment, the differences in the source and object code, and the differences between the test environment and the target environment, shall be analysed in order to specify additional tests in the target environment during the subsequent test phases.

Excerpt from:
ISO 26262-6 Road vehicles - Functional safety – Part 6: Product development: Software Level, 2011.

⇒ Use non-intrusive hardware support of modern processors

AbsInt

Real-Time Trace Formats

- Provided many modern high-end processors
  - ✓ Nexus IEEE-ISTO 5001 program trace events (at least class 2)* in trace data, e.g. generated by
    - PowerPC NXP Qorivva, QorIQ P- and T-series, e.g. MPC55xx/MPC56xx/MPC57xx, P204x/P30xx/P40xx/P50xx
  - ✓ CoreSight - Embedded Trace Macrocell (ETM) instruction trace data, e.g. generated by
    - ARMv7/v8 AARch32, e.g., Cortex-A53, Cortex-R5F
  - ✓ Multi-Core Debug Solution (MCDS) Program Traces (at least Flow Traces) e.g. generated by
    - Infineon TriCore AURIX platform
    - Infineon C16x/XC2000 platform

(*) class 1 correspond to JTAG debugger -- class 4 to real-time instruction traces

Nexus Traces

- Trace segments, separated by trace events
- Contents of trace message for a trace event:
  - Time stamp + Address + Content of Branch-History-Buffer (BHB)

```
+056 TCODE =1D PT - IBHSM F- ADDR = F1F4 HIST =2 TS =8847
+064 TCODE =21 PT - PTCM EVCODE =A TS =88 F1
+072 TCODE =1C PT - IBHM U- ADDR =03 DC HIST =1 TS =8 D62
+080 TCODE =21 PT - PTCM EVCODE =A TS =8 E2F
+088 TCODE =21 PT - PTCM EVCODE =A TS =8 FBA
+096 TCODE =21 PT - PTCM EVCODE =A TS =9105
```

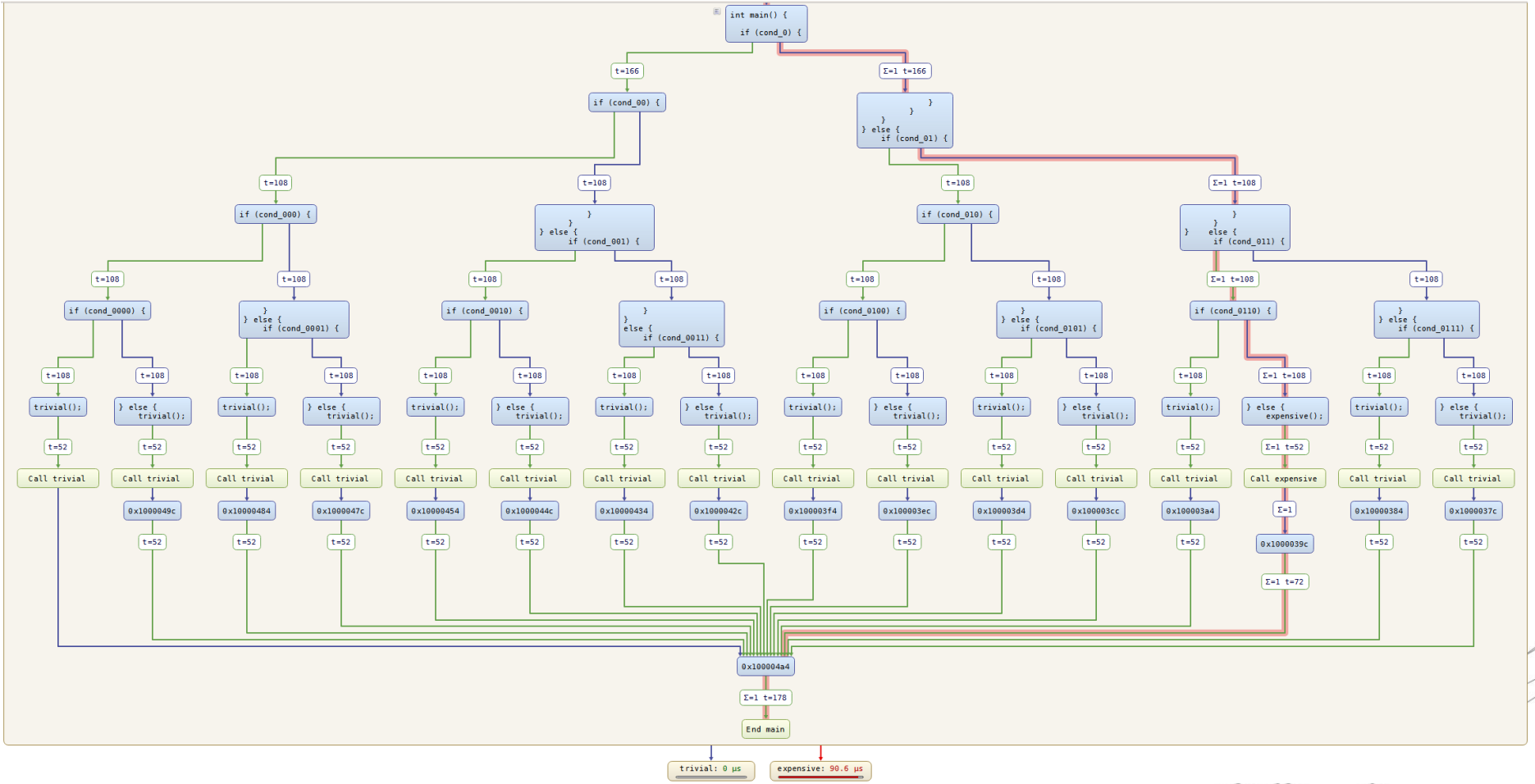
- One trace event
  - for each indirect branch
  - when the BHB is full

$\Rightarrow$ Not for every branch exists a timestamp

Trace Analysis

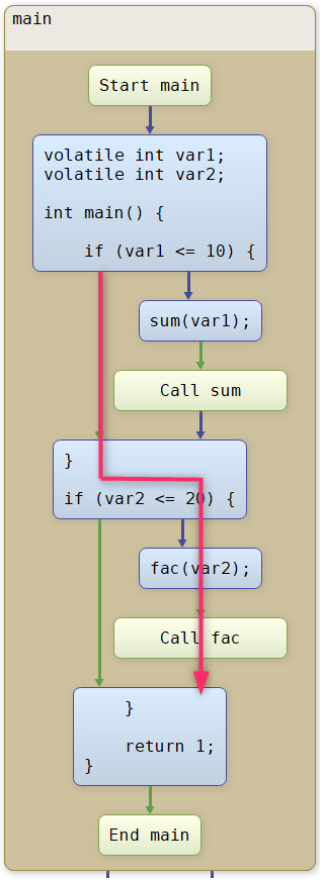
- Trace graph: super-graph over all input traces
  - Nodes: trace points (addresses of trace events)
  - Edges: trace segments; edge costs: execution time from traces
- Trace segments are context-sensitive
  - A trace segment represents (context-sensitive) CFG edges
  - Multiple trace graph edges between two trace points
- Connecting trace to input binary
  - Trace event/trace point ⟶ point in the control-flow graph (CFG)
  - Trace segment ⟶ program path between trace points, annotated with costs

The Path Coverage Aspect …

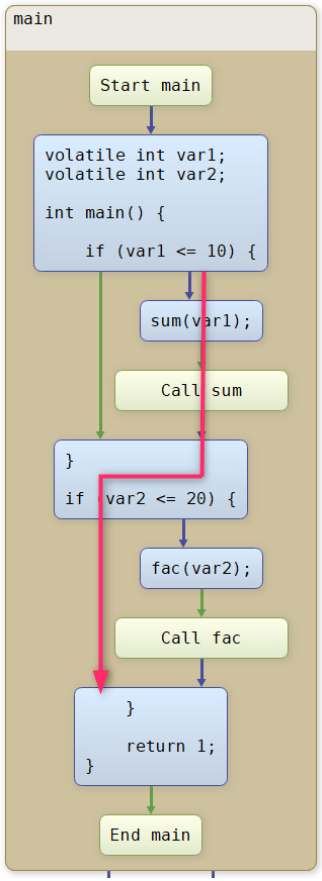


Extrapolated WCET

- TimeWeaver extrapolates the timing from concrete execution traces to a worst-case execution time (WCET) estimate. Example:

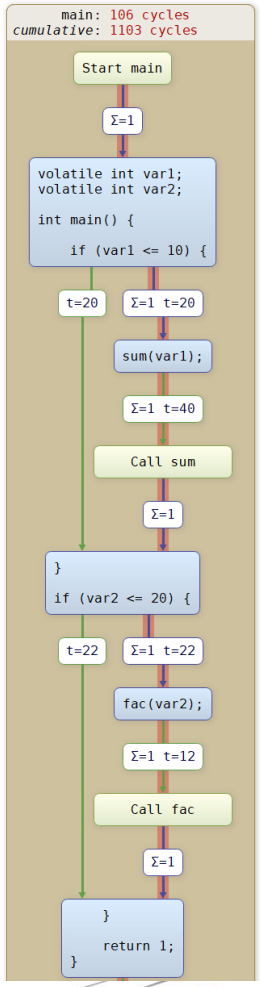


1st Execution Trace



2nd Execution Trace

TimeWeaver Analysis



Extrapolated **WCET** Path

AbsInt

PART I – Timing Analysis

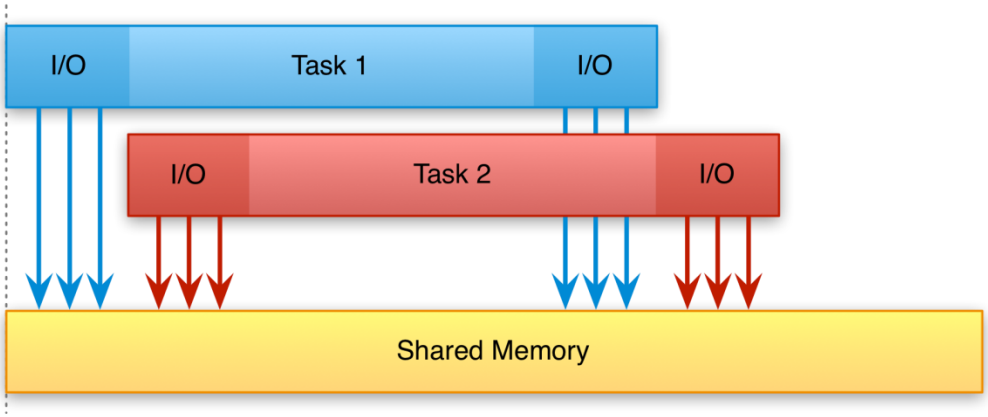
Reducing Resource Conflicts

Reducing Resource Conflicts

- Privatisation of shared resources
  - Temporal partioning
  - Spatial partioning
  - Budget-based partioning

Privatisation of Shared Resources

- TDMA-based resource scheduling (cf. Schranzhofer et al., „Timing predictability on multi-processor systems with shared resources“, RePP workshop 2009)

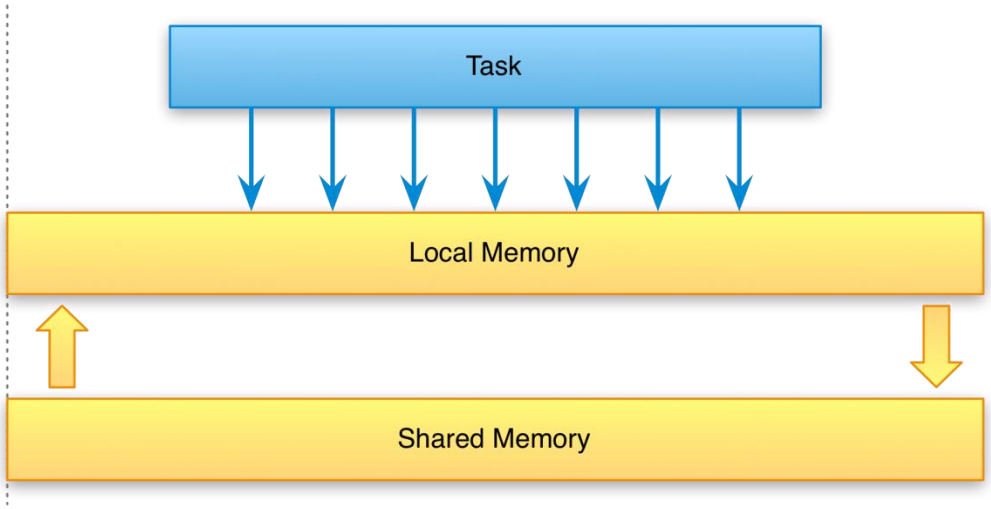


- Needs changes on existing code

AbsInt

Privatisation of Shared Resources

- Copy data in warm-up phase from shared memory to local memory, copy data in cool-down phase from local memory to shared memory



- Needs tool or operating system support
- May have severe performance impact

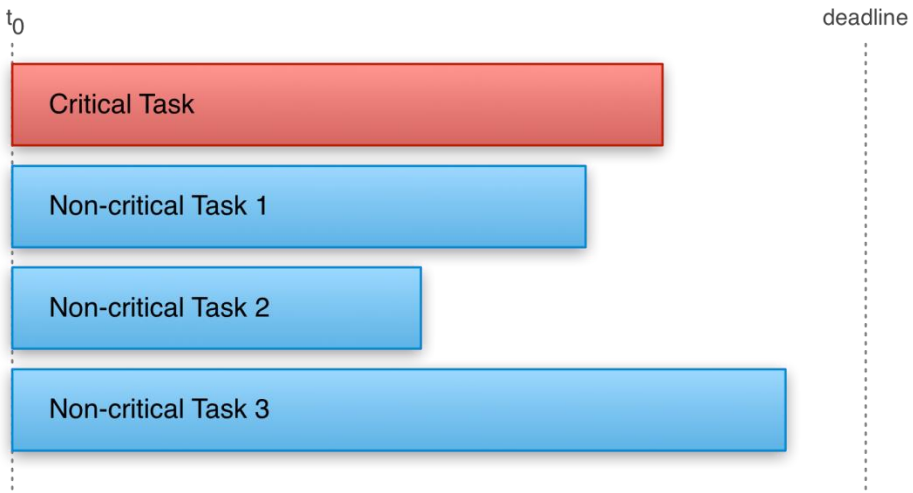
AbsInt

Runtime Resource Capacity Enforcement

- Uses three main concepts to reduce the interference delays
  - Limitation
  - Monitoring
  - Suspension
- Especially useful for mixed-critically systems
- Also provides a safety net against SEUs (single event upsets)

Example

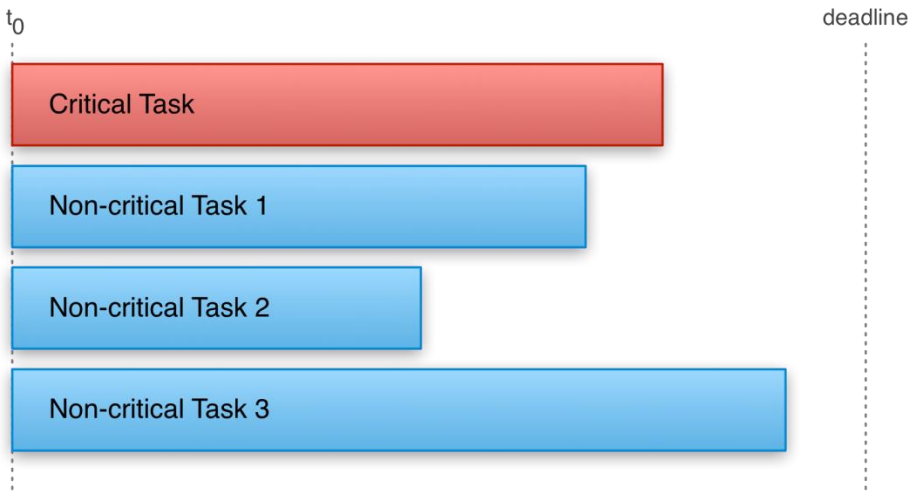
- Consider a system of four tasks, mapped to four cores
- One of the tasks has high criticality, the others have low criticality



AbsInt

Limitation of Critical Tasks

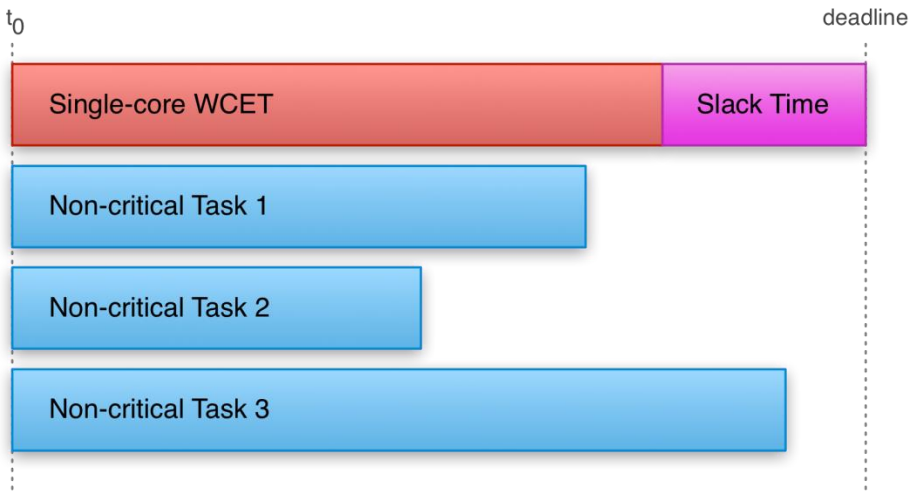
- The resource capacity is the worst-case number of resource accesses (WCRA)
- Can be computed statically, e.g. with a modified version of aiT



AbsInt

Limitation of Non-critical Tasks

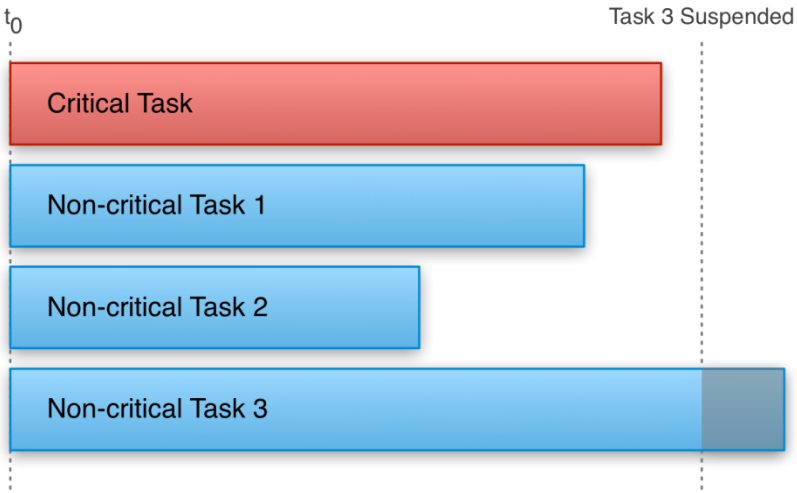
- Compute the single-core WCET of the critical task
- This gives the slack time of the critical task, which equals the amount of interference delay we can allow
- Divide the slack time by the interference delay of one access with maximal interference
- This gives the number of accesses that can be delayed in the critical task, and equals the capacity of the non-critical tasks



AbsInt

Monitoring & Suspension

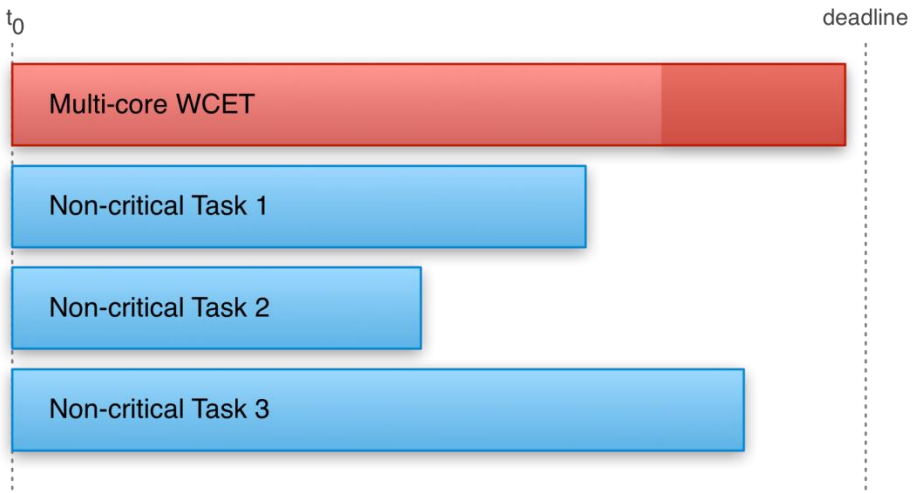
- Runtime monitoring via performance counters is used to observe the number of actual resource accesses
- Tasks exceeding their access capacity are suspended by the operating system



AbsInt

Multi-core WCET

- OS enforces interference delay which is smaller than the slack time ⇒ the deadline is never missed



AbsInt

PART I – Timing Analysis

Smart Hardware Configurations

AbsInt

Design Guidelines for Predictable Multicores

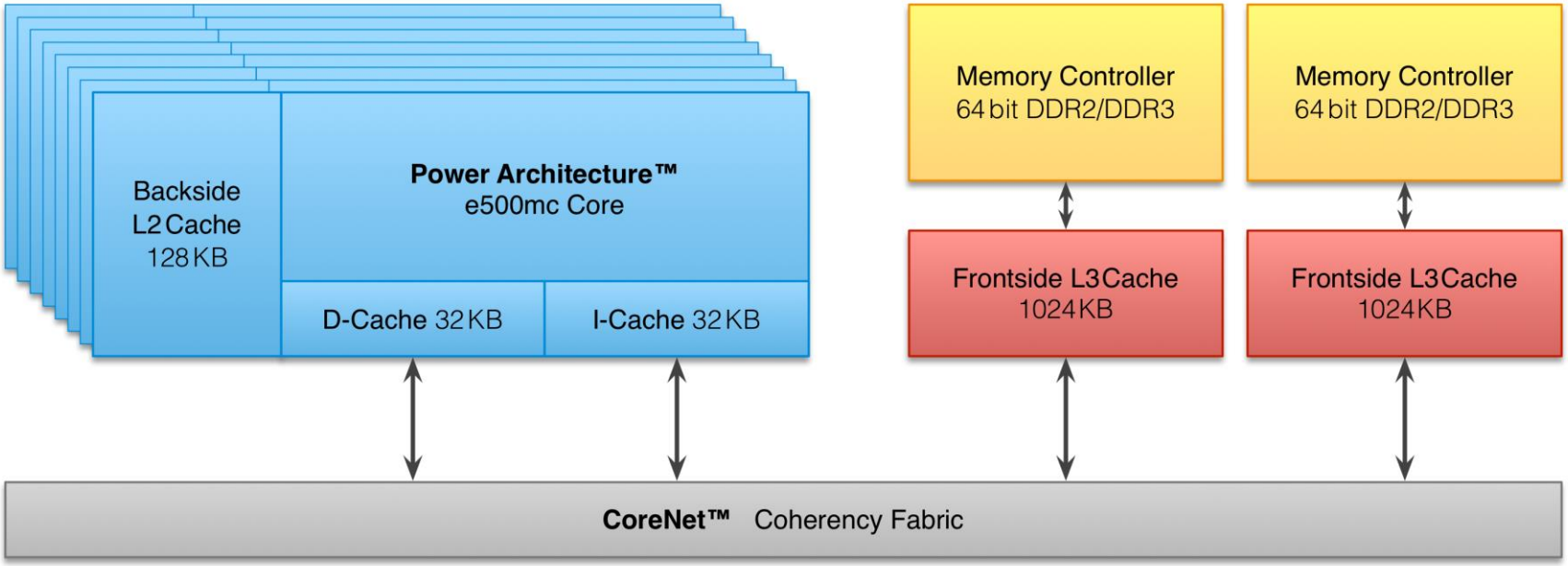
1. Fully timing compositional cores: no timing anomalies, no domino effects
2. Disjoint data and instruction caches
   - Unified caches cause uncertainties on data accesses to interfere with the instruction cache analysis
3. LRU replacement policies for caches
   - pLRU and FIFO replacement policies are not well predictable
4. Private caches
   - Shared caches induce uncertainty on their contents
5. Private memories, lonely sharing
   - Access latency to shared resources depends on utilization
6. Shared bus protocol with bounded access delay

PREDATOR

PREDATOR was an ICT project in the 7th Framework Program of the EU

AbsInt

Freescale QorIQ P4080

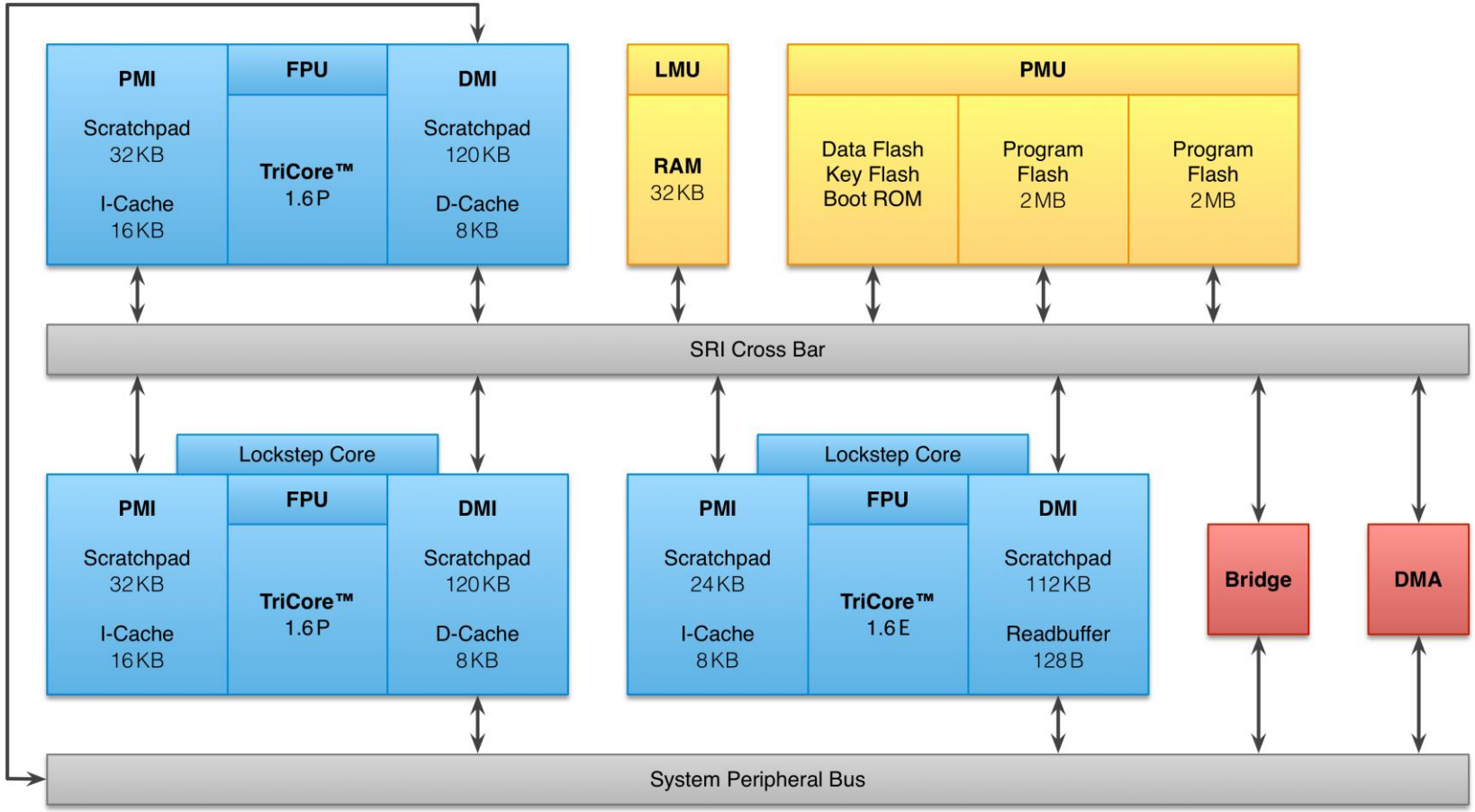


AbsInt

Freescale QorIQ P4080

1. Fully timing compositional cores ✗
2. Disjoint data and instruction caches (✓)
3. LRU replacement policies for caches (✓)
4. Private caches (✗)
5. Private memories, lonely sharing (✗)
6. Shared bus protocol with bounded access delay ✗

Infineon AURIX TC27x



AbsInt

Infineon AURIX TC27x

1. Fully timing compositional cores ✗
2. Disjoint data and instruction caches ✓
3. LRU replacement policies for caches ✓
4. Private caches ✓
5. Private memories, lonely sharing ✓
6. Shared bus protocol with bounded access delay (✓)

AbsInt

Infineon AURIX TC27x - Configuration

- Use one dedicated program flash memory for each of the performance cores to avoid conflicting accesses. Use the data flash for the efficiency core, if needed.
- Use the core-local data scratchpad whenever possible instead of the shared RAM to reduce conflicting accesses. If possible, preload data from the shared RAM and data flash to the local scratchpad memories to control when accesses to the shared memory happen.
- Place the stack in the core-local data scratchpad.
- I/O channels (CAN, FlexRay, . . .) should not be accessed by multiple cores. Assign each I/O channel in use to a specific core.
- Avoid accesses to core-local scratchpad memories from other cores.

PART II – Memory

Using Generic Software Components for Safety-critical Embedded Systems

This work was funded by the German Federal Ministry for Education and Research (BMBF) within the project ARAMiS II with the funding ID 01IS16025. The responsibility for the content remains with the authors.

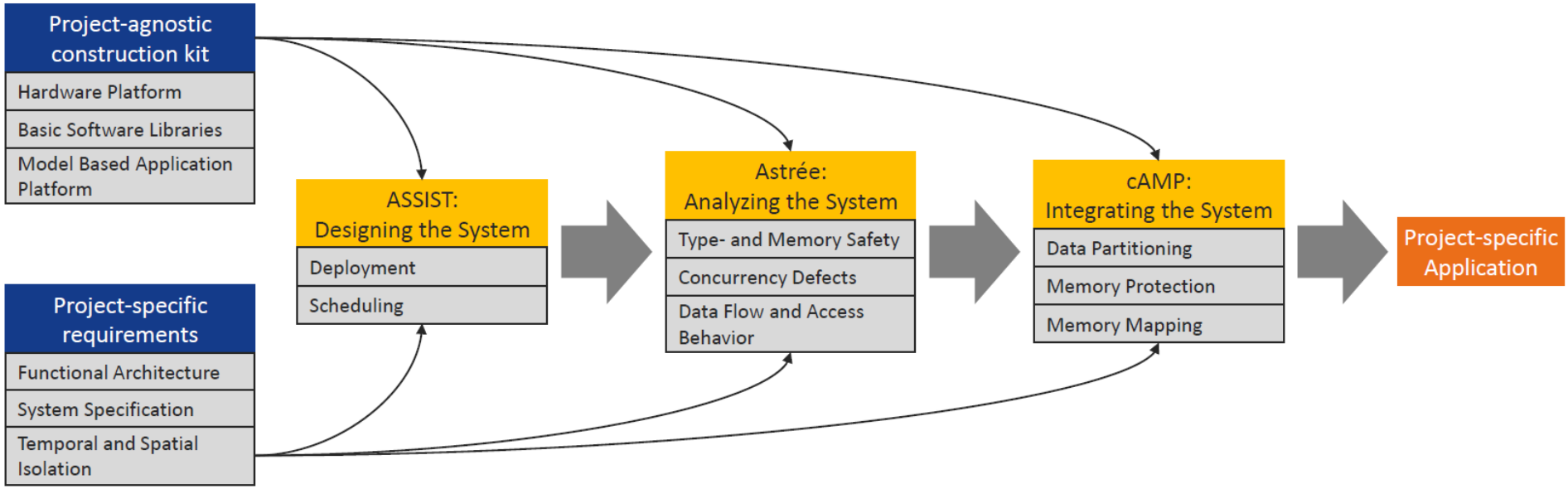
AbsInt

Using Generic Software Components for Safety-critical Embedded Systems

Goal

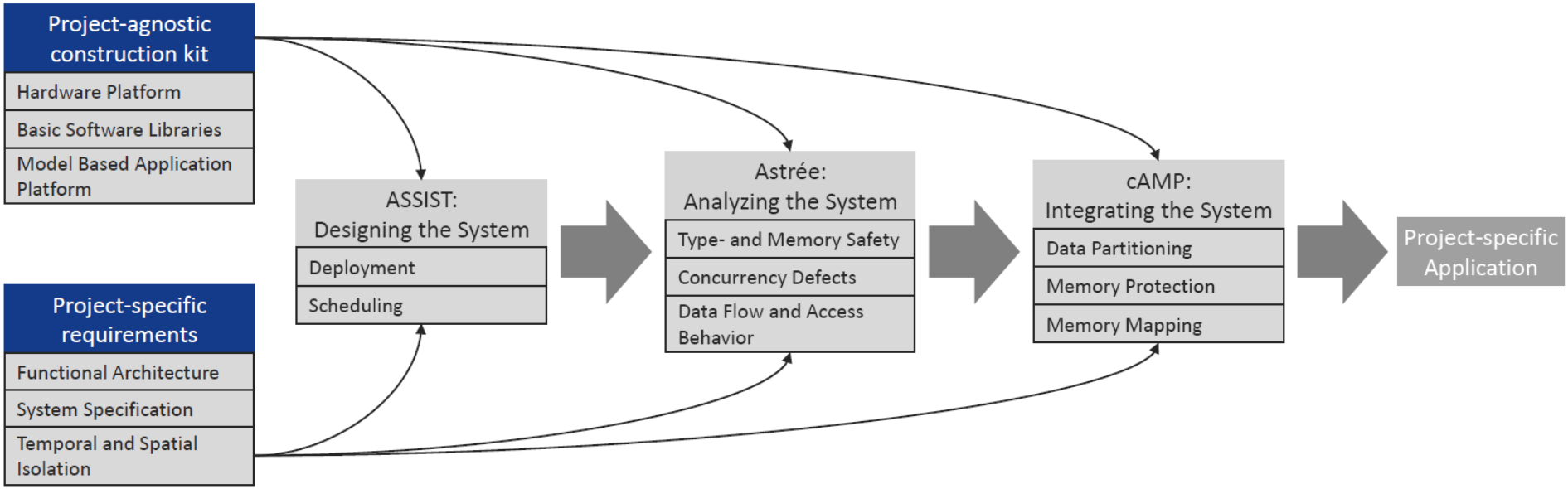
- Automate deployment and memory mapping
- Optimize mapping
- Enable the (re-)use of generic software components
- At the same time: ensure functional safety properties

Workflow



AbsInt

Input



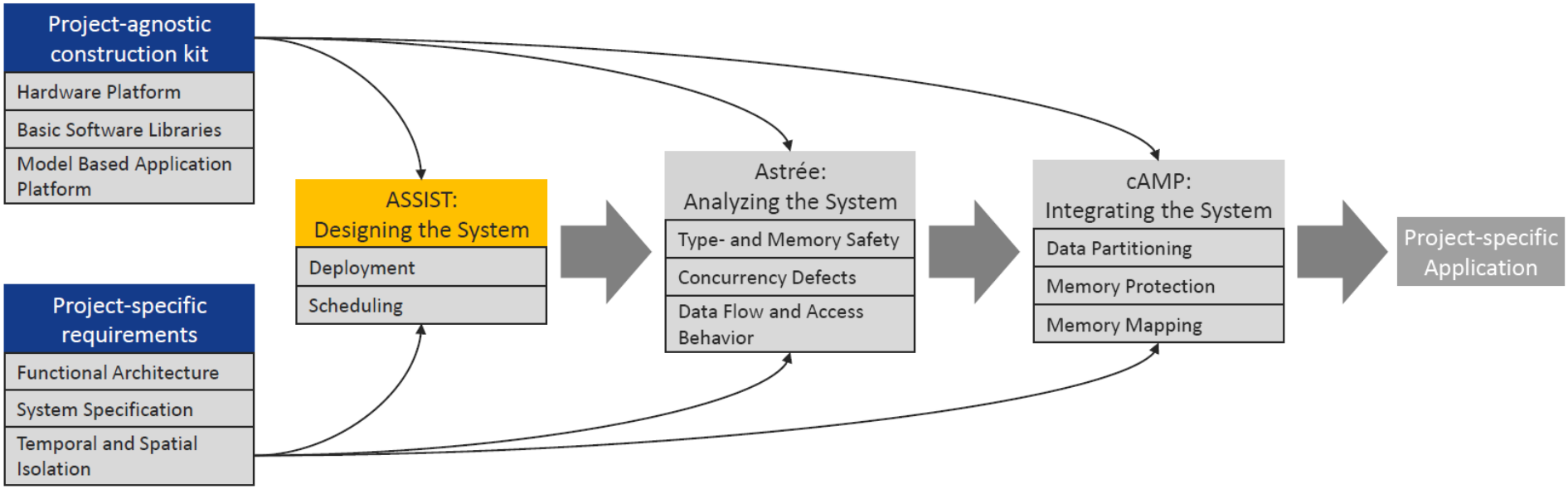
AbsInt

Construction Kit

- Hardware family is predetermined, for example Infineon AURIX
- … but different members of the family differ in cost and capabilities (e.g. memory size, number of cores)
- Control algorithms are available as a generic model-based component library
- … but need to be adapted to concrete hardware platform and software architecture

Project-specific Requirements

- Some tasks must be performed in a given timeframe, and with a given frequency ($\Rightarrow$ real-time requirements)
- Some tasks are not allowed to access data of other tasks ($\Rightarrow$ spatial isolation)
- Some components shall not run on the same core / processor ($\Rightarrow$ dislocality), for fail-safe monitoring or fail-operational systems
- …


Project-agnostic construction kit
Hardware Platform
Basic Software Libraries
Model Based Application Platform
Project-specific requirements
Functional Architecture
System Specification
Temporal and Spatial Isolation
ASSIST: Designing the System
Deployment
Scheduling
Astrée: Analyzing the System
Type- and Memory Safety
Concurrency Defects
Data Flow and Access Behavior
cAMP: Integrating the System
Data Partitioning
Memory Protection
Memory Mapping
Project-specific Application

AbsInt

ASSIST – Input

```
Hardware {
  /* ... */
  Processor Processor1 {
    Manufacturer = "Infineon";
    Type = "TC277";
    Provides 32768 of exclusive feature "LMU RAM";
    Provides 4194304 of exclusive feature "PMU Program Flash";
    Core Core0 {
        Capacity = 100;
        Architecture = "TriCore 1.6 P";
        Provides shared feature "Performance";
        Provides shared feature "FPU";
        Provides 16384 of exclusive feature "I-Cache";
        Provides 8192 of exclusive feature "D-Cache";
    }
    Core Core1 {
        /* identical to Core0 */
    }
    Core Core2 {
        Capacity = 60;
        Architecture = "TriCore 1.6 E";
        Provides shared feature "Efficiency";
        Provides shared feature "FPU";
        Provides shared feature "Lockstep";
        Provides 8192 of exclusive feature "I-Cache";
        Provides 128 of exclusive feature "DMI Readbuffer";
    }
  }
}
```

Figure 3: Hardware Specification in ASSIST

AbsInt

ASSIST – Input

```
Software {
  Application OS_Application_0 {
    Task T1_Controllers { CoreUtilization = 2; }
  }
  Application OS_Application_1 {
     Task T3_AttitudeObserver { CoreUtilization = 20; }
  }
  Application OS_Application_2 {
     Task T2_EngineController {
            CoreUtilization = 2;
            Requires shared Core feature "Lockstep";
      }
  }
  Application OS_Application_3 {
     Task T4_HeightObserver { CoreUtilization = 20; }
  }
  Application OS_Application_4 {
     Task T6_DSP { CoreUtilization = 3; }
  }
  Application OS_Application_5 {
     Task T5_AltitudeObserver { CoreUtilization = 20; }
  }
}
```

Figure 4: Applications and Tasks in ASSIST

AbsInt

ASSIST – Input

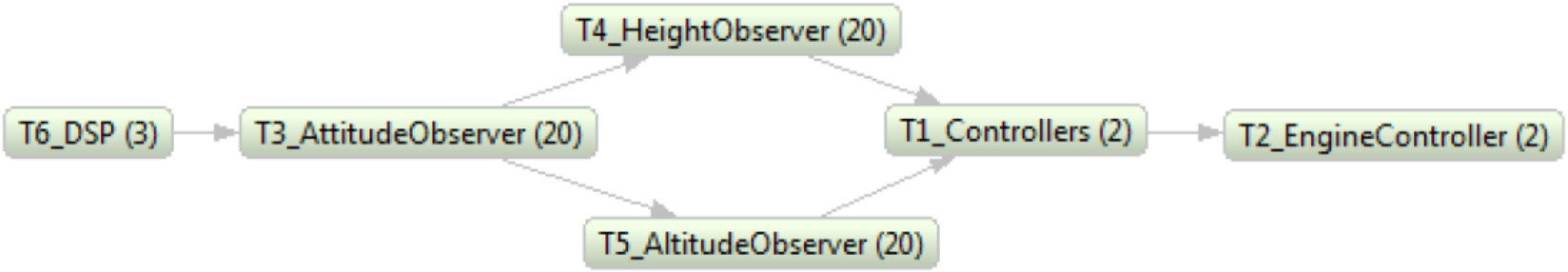


Figure 5: Task Dependency Graph

```
TaskGraph {
    T6_DSP                -> T3_AttitudeObserver;
    T3_AttitudeObserver -> T4_HeightObserver, T5_AltitudeObserver;
    T4_HeightObserver    -> T1_Controllers;
    T5_AltitudeObserver -> T1_Controllers;
    T1_Controllers        -> T2_EngineController;
}
```

Figure 6: Task Graph Specification in ASSIST

AbsInt

ASSIST – Input

```
Restrictions {
    T6_DSP, T3_AttitudeObserver dislocal up to Core;
    T6_DSP, T4_HeightObserver dislocal up to Core;
}
```

Figure 7: Mapping Constraint Specification in ASSIST

AbsInt

ASSIST – Output

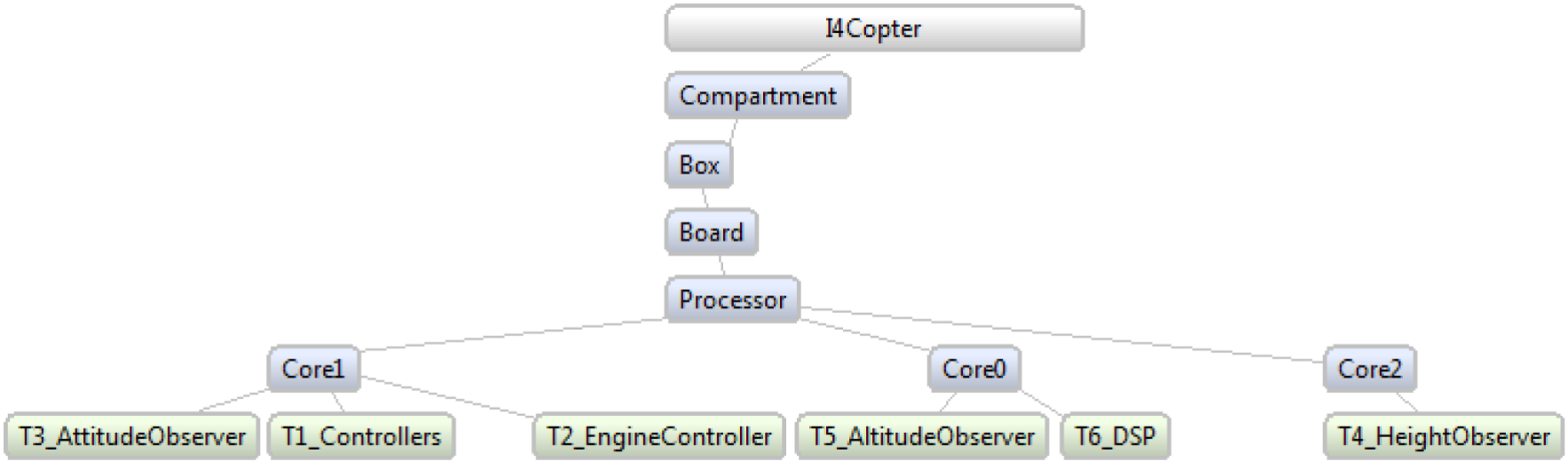


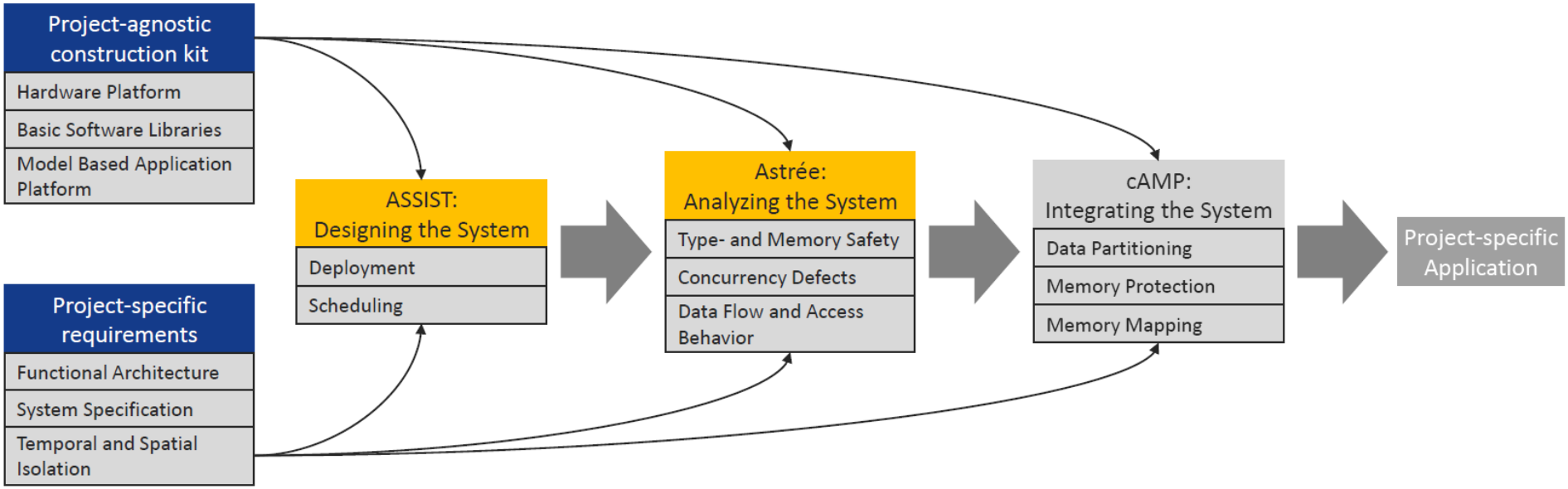
Figure 10: Deployment solution found by ASSIST

AbsInt

ASSIST – Output

```
3104            <ECUC-CONTAINER-VALUE UUID="4c1f47c8-5630-4b7f-836d-5c5618980ff2">
3105              <SHORT-NAME>T1_Controllers</SHORT-NAME>
3106              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/MICROSAR/Os/OsTask</DEFINITION-REF>
3107              <PARAMETER-VALUES>
3108                <ECUC-NUMERICAL-PARAM-VALUE>
3109                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskActivation</DEFINITION-REF>
3110                  <VALUE>1</VALUE>
3111                </ECUC-NUMERICAL-PARAM-VALUE>
3112                <ECUC-NUMERICAL-PARAM-VALUE>
3113                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskPriority</DEFINITION-REF>
3114                  <VALUE>6</VALUE>
3115                </ECUC-NUMERICAL-PARAM-VALUE>
3116                <ECUC-TEXTUAL-PARAM-VALUE>
3117                  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskSchedule</DEFINITION-REF>
3118                  <VALUE>FULL</VALUE>
3119                </ECUC-TEXTUAL-PARAM-VALUE>
3120                <ECUC-NUMERICAL-PARAM-VALUE>
3121                  <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskStackSharing</DEFINITION-REF>
3122                  <VALUE>false</VALUE>
3123                </ECUC-NUMERICAL-PARAM-VALUE>
3124                <ECUC-NUMERICAL-PARAM-VALUE>
3125                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskStackSize</DEFINITION-REF>
3126                  <VALUE>1024</VALUE>
3127                </ECUC-NUMERICAL-PARAM-VALUE>
3128                <ECUC-TEXTUAL-PARAM-VALUE>
3129                  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskType</DEFINITION-REF>
3130                  <VALUE>AUTO</VALUE>
3131                </ECUC-TEXTUAL-PARAM-VALUE>
3132              </PARAMETER-VALUES>
3133              <REFERENCE-VALUES>
```

OS Configuration (*.arxml)


Project-agnostic construction kit
Hardware Platform
Basic Software Libraries
Model Based Application Platform
Project-specific requirements
Functional Architecture
System Specification
Temporal and Spatial Isolation
ASSIST: Designing the System
Deployment
Scheduling
Astrée: Analyzing the System
Type- and Memory Safety
Concurrency Defects
Data Flow and Access Behavior
cAMP: Integrating the System
Data Partitioning
Memory Protection
Memory Mapping
Project-specific Application

AbsInt

Astrée – Input

```
3104            <ECUC-CONTAINER-VALUE UUID="4c1f47c8-5630-4b7f-836d-5c5618980ff2">
3105              <SHORT-NAME>T1_Controllers</SHORT-NAME>
3106              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/MICROSAR/Os/OsTask</DEFINITION-REF>
3107              <PARAMETER-VALUES>
3108                <ECUC-NUMERICAL-PARAM-VALUE>
3109                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskActivation</DEFINITION-REF>
3110                  <VALUE>1</VALUE>
3111                </ECUC-NUMERICAL-PARAM-VALUE>
3112                <ECUC-NUMERICAL-PARAM-VALUE>
3113                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskPriority</DEFINITION-REF>
3114                  <VALUE>6</VALUE>
3115                </ECUC-NUMERICAL-PARAM-VALUE>
3116                <ECUC-TEXTUAL-PARAM-VALUE>
3117                  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskSchedule</DEFINITION-REF>
3118                  <VALUE>FULL</VALUE>
3119                </ECUC-TEXTUAL-PARAM-VALUE>
3120                <ECUC-NUMERICAL-PARAM-VALUE>
3121                  <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskStackSharing</DEFINITION-REF>
3122                  <VALUE>false</VALUE>
3123                </ECUC-NUMERICAL-PARAM-VALUE>
3124                <ECUC-NUMERICAL-PARAM-VALUE>
3125                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskStackSize</DEFINITION-REF>
3126                  <VALUE>1024</VALUE>
3127                </ECUC-NUMERICAL-PARAM-VALUE>
3128                <ECUC-TEXTUAL-PARAM-VALUE>
3129                  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskType</DEFINITION-REF>
3130                  <VALUE>AUTO</VALUE>
3131                </ECUC-TEXTUAL-PARAM-VALUE>
3132              </PARAMETER-VALUES>
3133              <REFERENCE-VALUES>
```

OS Configuration (*.arxml)

AbsInt

Astrée – Input

2019-10-01 : vim — Konsole

File Edit View Bookmarks Settings Help

beebs : bash | swegener : bash | 2019-10-01 : vim

```
393      # combined # Sum: EngineController/thrust__torque-to-throttle conversion/Sum1 */
394    tor_to_thro_conv__Sum4 = Sb3_throttle_per_torqueZ + (Sb3_Lookup_Table +
395     Sb3_throttle_per_torqueY);
396
397    /* Saturation: EngineController/thrust__torque-to-throttle conversion/EngineCap2 */
398    if (tor_to_thro_conv__Sum4 > tor_to_thro_conv__EngineCap2_uplimit) {
399       /* # combined # TargetLink outport: EngineController/throttleFront_engCtr */
400       Out_throttleFront_engCtr = tor_to_thro_conv__EngineCap2_uplimit;
401    }
402    else {
403       /* CONST: global constants (ROM) | Width: 32 */
404       CONST Float32 tor_to_thro_conv__EngineCap2_lowlimit = -40.156864166259766F; /* MIN/MAX: -40.15
405       68641662598 .. -40.1568641662598 */
406
407       if (tor_to_thro_conv__Sum4 < tor_to_thro_conv__EngineCap2_lowlimit) {
408          /* # combined # TargetLink outport: EngineController/throttleFront_engCtr */
409          Out_throttleFront_engCtr = tor_to_thro_conv__EngineCap2_lowlimit;
410       }
411       else {
412          /* # combined # TargetLink outport: EngineController/throttleFront_engCtr */
413          Out_throttleFront_engCtr = tor_to_thro_conv__Sum4;
414       }
415    }
416
417    /* TableLookup: EngineController/throttle-to-thrust__torque conversion/Lookup Table2
418       # combined # TargetLink outport: EngineController/throttleFront_engCtr */
419    Sb2_Lookup_Table2 = Tab1DS3I2T3126_b(&Sb2_Lookup_Table2_map, Out_throttleFront_engCtr);
420
421    /* Sum: EngineController/thrust__torque-to-throttle conversion/Sum7
422       # combined # Sum: EngineController/thrust__torque-to-throttle conversion/Sum */
423    tor_to_thro_conv__Sum7 = (Sb3_Lookup_Table + Sb3_throttle_per_torqueX) -
424     Sb3_throttle_per_torqueZ;
425
426    /* Saturation: EngineController/thrust__torque-to-throttle conversion/EngineCap1 */
427    if (tor_to_thro_conv__Sum7 > tor_to_thro_conv__EngineCap1_uplimit) {
428       /* # combined # TargetLink outport: EngineController/throttleLeft_engCtr */
429       Out_throttleLeft_engCtr = tor_to_thro_conv__EngineCap1_uplimit;
430    }
431    else {
432       /* CONST: global constants (ROM) | Width: 32 */
433       CONST Float32 tor_to_thro_conv__EngineCap1_lowlimit = -40.156864166259766F; /* MIN/MAX: -40.15
434       68641662598 .. -40.1568641662598 */
435
436       if (tor_to_thro_conv__Sum7 < tor_to_thro_conv__EngineCap1_lowlimit) {
437          /* # combined # TargetLink outport: EngineController/throttleLeft_engCtr */
438          Out_throttleLeft_engCtr = tor_to_thro_conv__EngineCap1_lowlimit;
439       }
440       else {
441          /* # combined # TargetLink outport: EngineController/throttleLeft_engCtr */
442          Out_throttleLeft_engCtr = tor_to_thro_conv__Sum7;
443       }
444    }
445
446    /* TableLookup: EngineController/throttle-to-thrust__torque conversion/Lookup Table3
447       # combined # TargetLink outport: EngineController/throttleLeft_engCtr */
448    Sb2_Lookup_Table3 = Tab1DS3I2T3126_b(&Sb2_Lookup_Table3_map, Out_throttleLeft_engCtr);
449
450    /* TargetLink outport: EngineController/thrustZ_N_engCtr
```

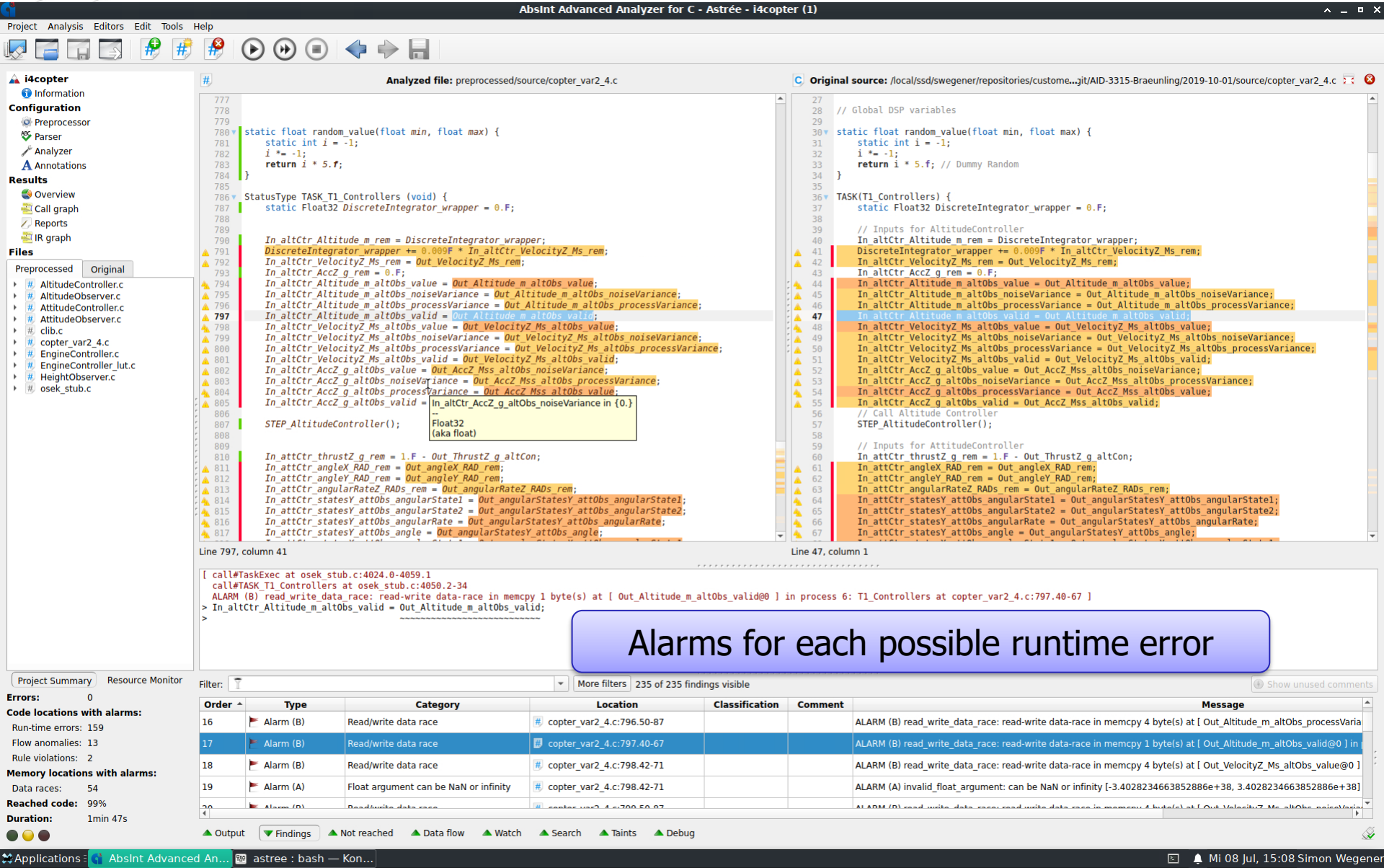
source/EngineController/EngineController.c [RO] 449,0-1 93%

```
98  /*--------------------------------------------------------------------------*\
99    TYPEDEFS
100 \*--------------------------------------------------------------------------*/
101 /*--------------------------------------------------------------------------*\
102   VARIABLES
103 \*--------------------------------------------------------------------------*/
104
105 /****************************************************************************\
106    SLGlobal: Default storage class for global variables | Width: 32
107 \****************************************************************************/
108 extern Float32 In_engCtr_takeoff_g_sw;
109 extern Float32 In_engCtr_thrustZ_N_attCtr;
110 extern Float32 In_engCtr_torqueX_NM_attCtr;
111 extern Float32 In_engCtr_torqueY_NM_attCtr;
112 extern Float32 In_engCtr_torqueZ_NM_attCtr;
113 extern Float32 Out_throttleFront_engCtr;
114 extern Float32 Out_throttleLeft_engCtr;
115 extern Float32 Out_throttleRear_engCtr;
116 extern Float32 Out_throttleRight_engCtr;
117 extern Float32 Out_thrustZ_N_engCtr;
118 extern Float32 Out_torqueX_NM_engCtr;
119 extern Float32 Out_torqueY_NM_engCtr;
120
121 /****************************************************************************\
122    SLGlobal: Default storage class for global variables | Width: 8
123 \****************************************************************************/
124 extern Bool Out_takeOff_engCtr;
125
126 /****************************************************************************\
127    SLGlobalConst: Default storage class for global const variables | Width: 32
128 \****************************************************************************/
129 extern const Float32 Sb2_Lookup_Table1_axis[10]; /* different constrained ranges */
130 extern const Float32 Sb2_Lookup_Table2_axis[10]; /* different constrained ranges */
131 extern const Float32 Sb2_Lookup_Table3_axis[10]; /* different constrained ranges */
132 extern const Float32 Sb2_Lookup_Table_axis[10]; /* different constrained ranges */
133 extern const Float32 Sb3_Lookup_Table_axis[10]; /* different constrained ranges */
134 extern const Float32 tor_conv__Lookup_Table1_table[10]; /* different constrained ranges */
135 extern const Float32 tor_conv__Lookup_Table2_table[10]; /* different constrained ranges */
136 extern const Float32 tor_conv__Lookup_Table3_table[10]; /* different constrained ranges */
137 extern const Float32 tor_conv__Lookup_Table_table[10]; /* different constrained ranges */
138 extern const Float32 tor_to_thro_conv__Lookup_Table_table[10]; /* different constrained ranges */
139
140 /*--------------------------------------------------------------------------*\
141   PARAMETERIZED MACROS
142 \*--------------------------------------------------------------------------*/
143 /*--------------------------------------------------------------------------*\
144   FUNCTION PROTOTYPES
145 \*--------------------------------------------------------------------------*/
146
147 /****************************************************************************\
148    GlobalFcnSpecStep: Default function class for not static model step functions
149 \****************************************************************************/
150 extern void STEP_EngineController(void);
151
152 #endif /* ENGINECONTROLLER_H */
153 /*--------------------------------------------------------------------------*\
154   END OF FILE
155 \*--------------------------------------------------------------------------*/
```

source/EngineController/EngineController.h [RO] 112,1 Bot

Applications | 2019-10-01 : vim —... | Mi 08 Jul, 14:53 Simon Wegener

Astrée – Output



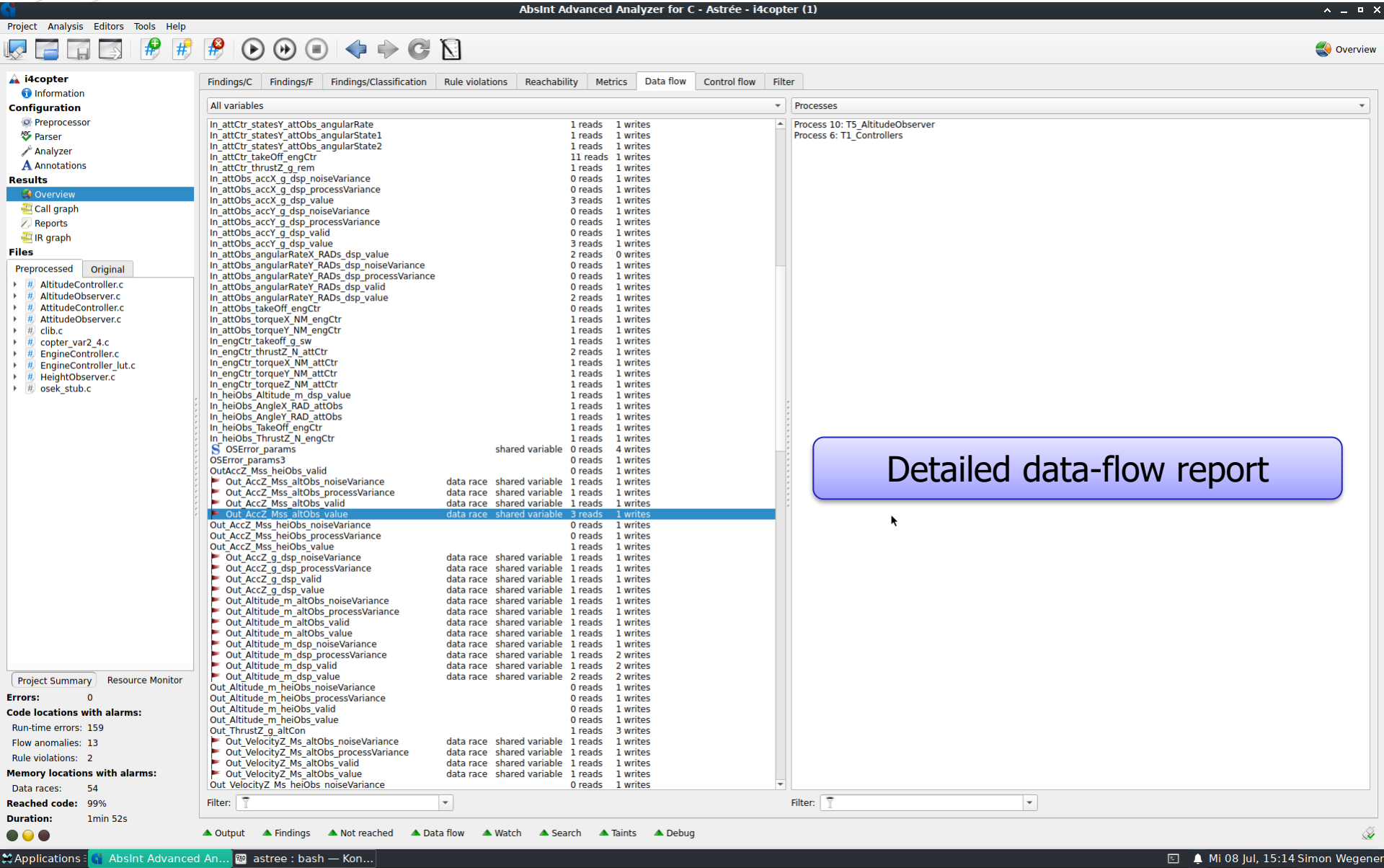
Astrée – Memory Safety

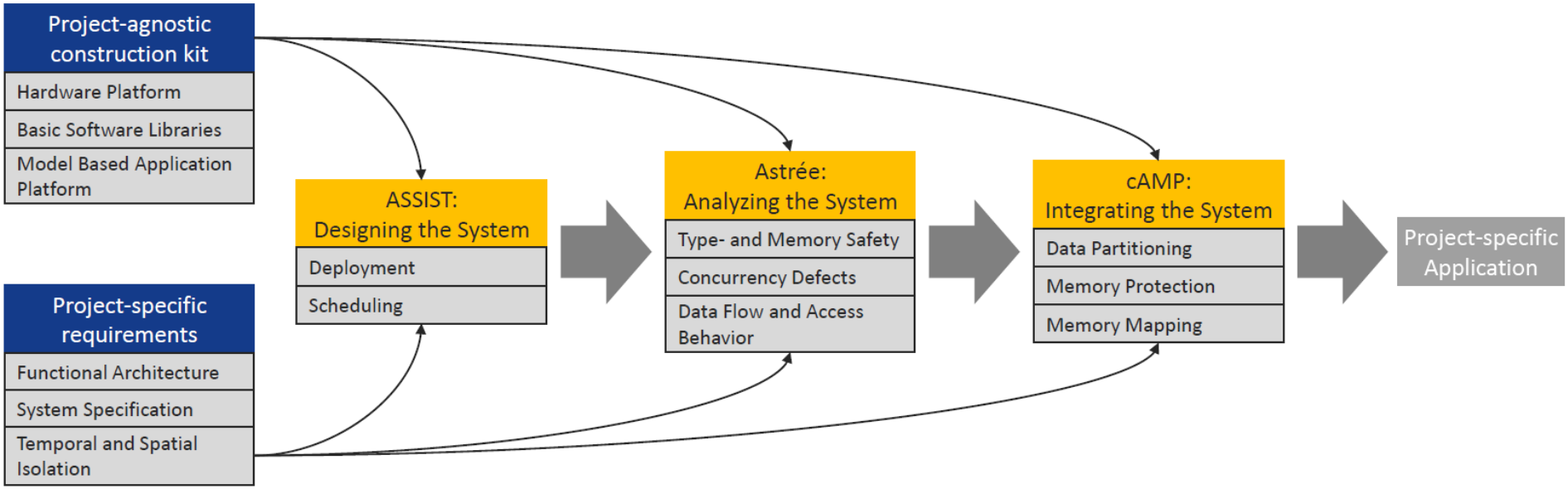
Table 1: Mapping of Requirements of Type- and Memory Safety [2] to Astrée Alarm Types [1].

Requirement	Alarm Type
Operations only applied for instances of correct type	Invalid pointer comparison Subtraction of invalid pointers Attempt to write to a constant Dereference of mis-aligned pointer Overflow of Integers or Float Invalid shift argument Use of uninitialized variables Division or modulo by zero Undefined integer modulo Invalid function calls Unsynchronized access to shared data
Access only existing objects	Dereference of null or invalid pointer Pointer to invalid or null function Use of dangling pointer Arithmetics on invalid pointers Possible overflow upon dereference
Access only inside object boundaries	Incorrect field dereference Out-of-bound array access Dereference of mis-aligned pointer Possible overflow upon dereference

AbsInt

Astrée – Output




Project-agnostic construction kit
Hardware Platform
Basic Software Libraries
Model Based Application Platform
Project-specific requirements
Functional Architecture
System Specification
Temporal and Spatial Isolation
ASSIST: Designing the System
Deployment
Scheduling
Astrée: Analyzing the System
Type- and Memory Safety
Concurrency Defects
Data Flow and Access Behavior
cAMP: Integrating the System
Data Partitioning
Memory Protection
Memory Mapping
Project-specific Application

AbsInt

cAMP – Input

```
3104            <ECUC-CONTAINER-VALUE UUID="4c1f47c8-5630-4b7f-836d-5c5618980ff2">
3105              <SHORT-NAME>T1_Controllers</SHORT-NAME>
3106              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/MICROSAR/Os/OsTask</DEFINITION-REF>
3107              <PARAMETER-VALUES>
3108                <ECUC-NUMERICAL-PARAM-VALUE>
3109                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskActivation</DEFINITION-REF>
3110                  <VALUE>1</VALUE>
3111                </ECUC-NUMERICAL-PARAM-VALUE>
3112                <ECUC-NUMERICAL-PARAM-VALUE>
3113                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskPriority</DEFINITION-REF>
3114                  <VALUE>6</VALUE>
3115                </ECUC-NUMERICAL-PARAM-VALUE>
3116                <ECUC-TEXTUAL-PARAM-VALUE>
3117                  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskSchedule</DEFINITION-REF>
3118                  <VALUE>FULL</VALUE>
3119                </ECUC-TEXTUAL-PARAM-VALUE>
3120                <ECUC-NUMERICAL-PARAM-VALUE>
3121                  <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskStackSharing</DEFINITION-REF>
3122                  <VALUE>false</VALUE>
3123                </ECUC-NUMERICAL-PARAM-VALUE>
3124                <ECUC-NUMERICAL-PARAM-VALUE>
3125                  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskStackSize</DEFINITION-REF>
3126                  <VALUE>1024</VALUE>
3127                </ECUC-NUMERICAL-PARAM-VALUE>
3128                <ECUC-TEXTUAL-PARAM-VALUE>
3129                  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/MICROSAR/Os/OsTask/OsTaskType</DEFINITION-REF>
3130                  <VALUE>AUTO</VALUE>
3131                </ECUC-TEXTUAL-PARAM-VALUE>
3132              </PARAMETER-VALUES>
3133              <REFERENCE-VALUES>
```

OS Configuration (*.arxml)

cAMP – Input

Table 2: Excerpt from Astrée Data Flow Report

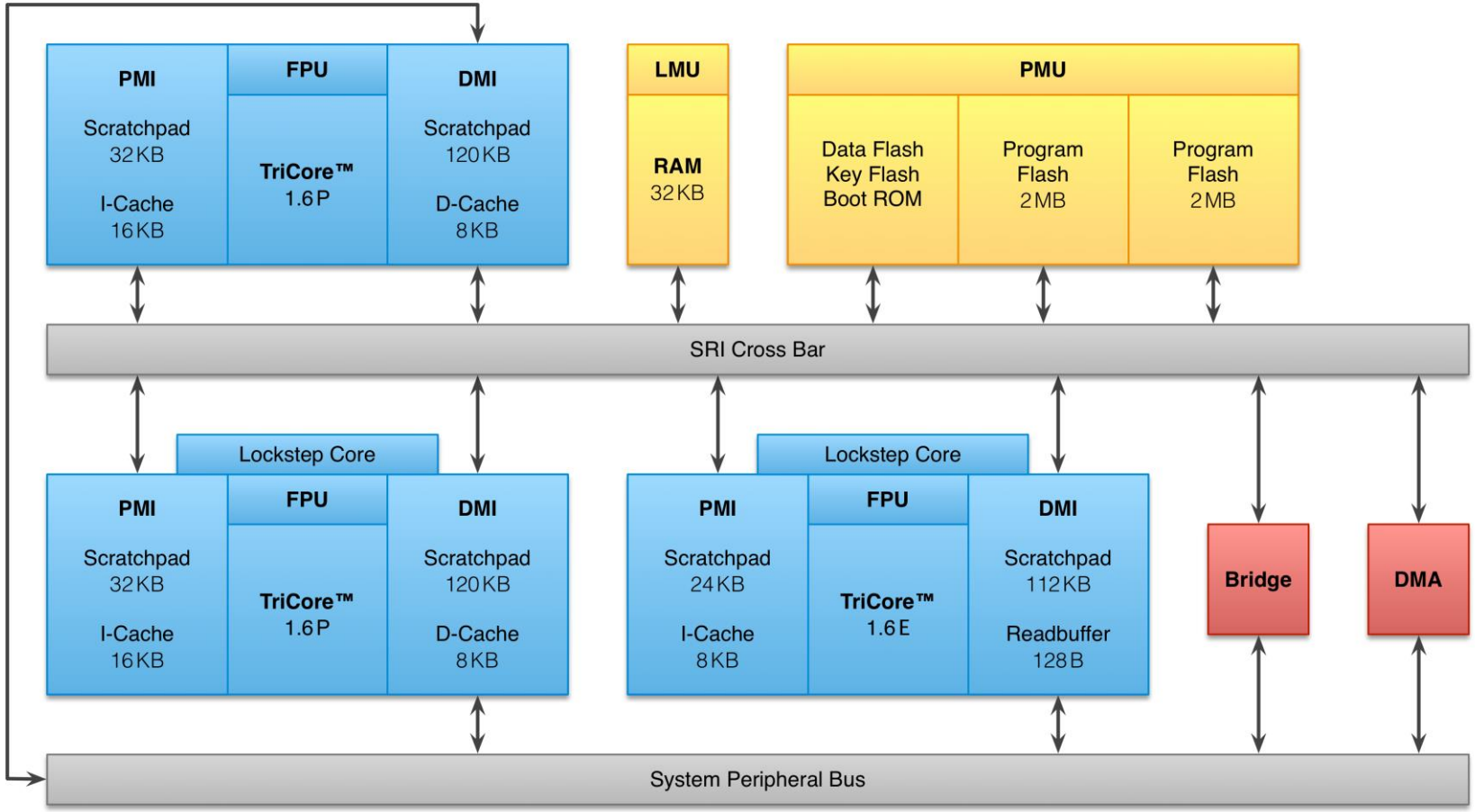
Variable	Function	Access	Process	Data Races	Shared	Class
In_altCtr_AccZ_g_altObs_noiseVariance	TASK_T1_Controllers	write	T1_Controllers	no	no	process local
In_altCtr_AccZ_g_altObs_processVariance	TASK_T1_Controllers	write	T1_Controllers	no	no	process local
Out_accX_g_dsp_noiseVariance	TASK_T3_AttitudeObserver	read	T3_AttitudeObserver	yes	yes	global
Out_accX_g_dsp_noiseVariance	TASK_T6_DSP	write	T6_DSP	yes	yes	global
Out_accX_g_dsp_processVariance	TASK_T3_AttitudeObserver	read	T3_AttitudeObserver	yes	yes	global
Out_accX_g_dsp_processVariance	TASK_T6_DSP	write	T6_DSP	yes	yes	global
Out_torqueX_NM_attCtr	TASK_T2_EngineController	read	T2_EngineController	yes	yes	core local
Out_torqueX_NM_attCtr	STEP_AttitudeController	write	T1_Controllers	yes	yes	core local
Out_torqueX_NM_engCtr	TASK_T3_AttitudeObserver	read	T3_AttitudeObserver	yes	yes	core local
...	...	...	...	...	...	...

AbsInt

cAMP – Rules

- The rule set used for the classification is heavily influenced by the capabilities of the target hardware and reflects which memories should be preferred during data binding.
- The classification takes into account:
  - access behavior (type, origin, frequency),
  - task system (tasks, periodicity, allocation),
  - available memory and
  - additional requirements (binding constraints, special data classes).

Infineon AURIX TC27x



AbsInt

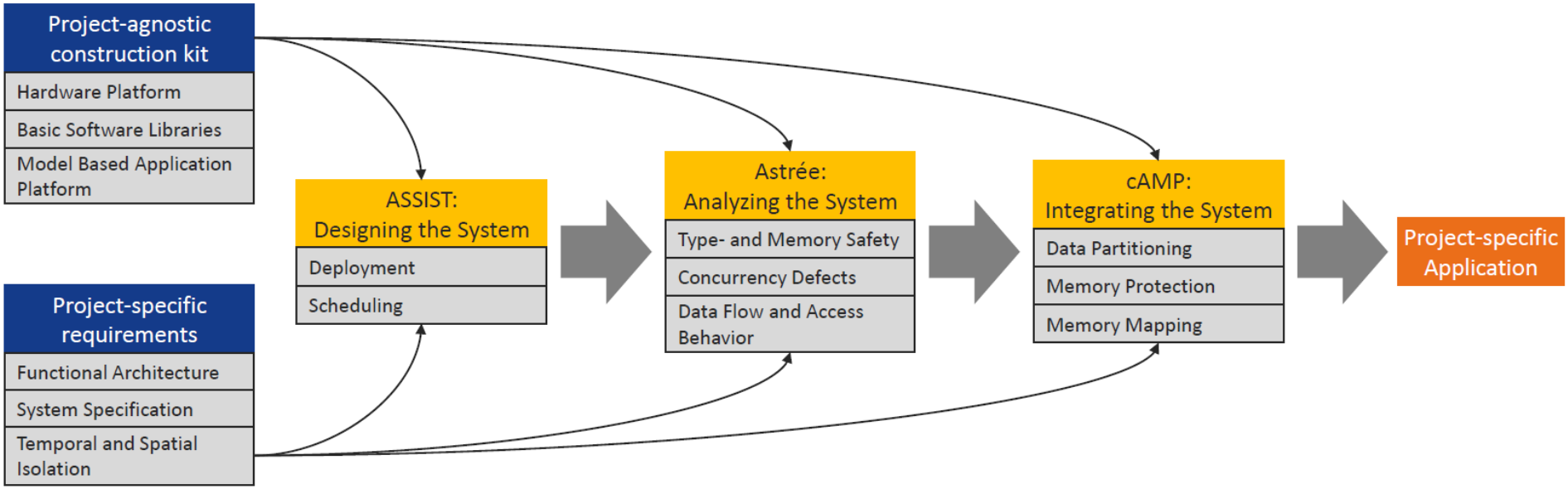
cAMP – Rules

- **Processor-n, Task-local, Memory-local** Data that is accessed by only one task, allocated on core *n* and placed in tightly-coupled memory
- **Processor-n, Task-global, Memory-local** Data that is accessed by more than one task, allocated on core *n* and placed in tightly-coupled memory
- **Processor-global, Task-global, Memory-global** Data that is accessed more than one task, allocated on one or more processing units and placed in global memory
- **Constant, Memory-global Data** that is only read by software and is placed in global memory.
- Some classes need special handling:
  - **Calibration Data** that is used for online calibration purposes.
  - **Measurement Data** that is used for online measurement purposes.

cAMP – Output

- Assignment of each variable to a memory section
- Adaption of code generation via TargetLink‘s variable classes, `__attribute__`, and `#pragma`
- Linker command file
- MPU config / AUTOSAR OS isolation regions

AbsInt


Project-agnostic construction kit
Hardware Platform
Basic Software Libraries
Model Based Application Platform
Project-specific requirements
Functional Architecture
System Specification
Temporal and Spatial Isolation
ASSIST: Designing the System
Deployment
Scheduling
Astrée: Analyzing the System
Type- and Memory Safety
Concurrency Defects
Data Flow and Access Behavior
cAMP: Integrating the System
Data Partitioning
Memory Protection
Memory Mapping
Project-specific Application

AbsInt

Result

- A project-specific application …
- … which satisfies important non-functional requirements (memory safety, spatial isolation, etc)
- … which is build from generic components
- ... and tailored to project-specific needs (e.g., hardware selection)
- … optimized for run-time and memory efficiency

Result

- Automation saves a lot of time
- Manual deployment and memory mapping of took about a week
- Automatic workflow does this in less than a hour

CONCLUSION

AbsInt

Conclusion

- Ensuring non-functional safety properties is possible for multicores – but needs some work!
- Reduce resource conflicts with smart software architecture – use sharing only where really needed.
- Reduce resource conflicts by using smart configurations of COTS multicores.
- Predictable multicores: less complexity and more precise results.

AbsInt

Conclusion

- The advanced tools of AbsInt help to …
- … verify the timing behaviour
- … ensure memory safety
- … prove the absence of runtime errors
- … optimize your code base

AbsInt

email: info@absint.com
https://www.absint.com