Technische Universität Wien

A Naïve Timing Analyser's Wish List for MPSoC Architectures

Peter Puschner

ARTISTDesign TA Activity Meeting Brussels, Belgium January 2009

Remember …

we want to build safety-critical hard real-time systems

⇨ timing requirements have to be met!

⇨ adequate engineering process

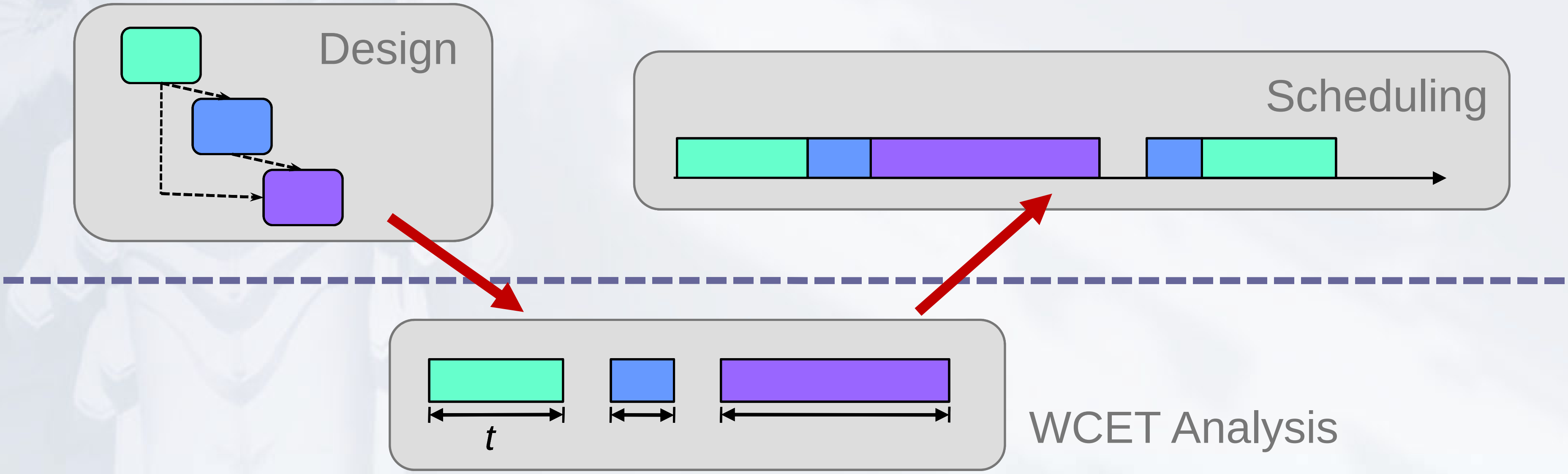
- straight-forward construction
- easy argumentation about properties – timing!

➡ **simple concepts!!!**

Hierarchical Design

Hierarchical design keeps complexity manageable

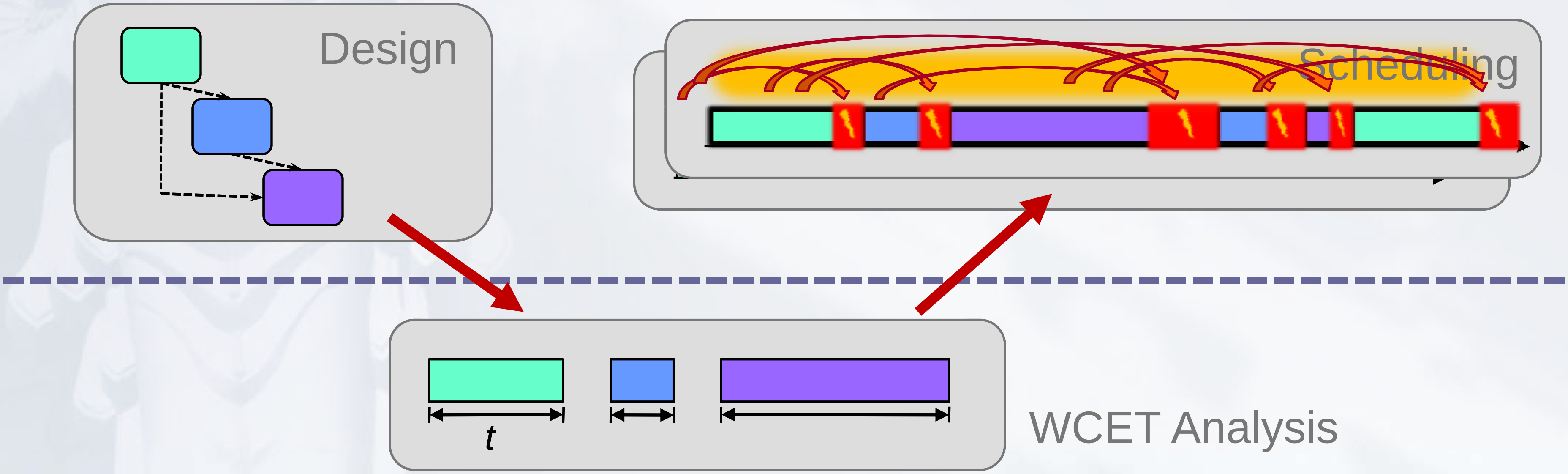
- subsystems need to be (de)composable:
  weak/no interactions among subsystems



Hierarchical Design

Hierarchical design keeps complexity manageable

- subsystems need to be (de)composable: weak/no interactions among subsystems



We need …

- simple, regular shape
  → dimensions are easy to assess, describe
- composability: it has the same dimensions under all circumstances (stand alone, when integrated, …)
- compositionality: combination function is simple

… for low-complexity integration of task timing

- composability: timing of task not affected by other tasks
- I/O compositionality of WCETs:

$$wcet_{A;B}(I) = wcet_A(I) + wcet_B(O_A(I))$$

⇨ no cross-border relationships between A and B, thus no costly over-estimation by hierarchical TA

- simple, regular shape: stable, invariable XTs

⇨ no undesirable effects of execution-time jitter, simple scheduling/schedulability tests

Realizing Composability

- Spatial isolation
  competing tasks use different resources (e.g., SMART cache)
  price: reduced resource bandwidth for each task
- Temporal isolation
  competing tasks use resources in different time windows
  price: save/restore state before/after each task activation
- Restrict state update strategy for shared resources
  prefetching into partitioned hierarchical memory:
  execute task from one region while loading other;
  prefetch controller interprets prefetch table (computed offline)

Realizing I/O Comp. and Stability

Stability implies I/O compositionality of WCETs

- Constant instruction-execution times
- Code alternatives take equal time
  - make timing of alternative traces equal; no cheap solution for complex architectures
  - eliminate alternatives:
    ⇨ single-path conversion of code

Obstacles in Chip-Multiprocessors

- Simultaneous multithreading
  - ⇨ strong coupling, e.g., due to use of same pipeline
  - ⇨ pessimism in static analysis
- Keeping caches coherent and consistent
  - ⇨ protocols: exchange of cache information causes variability of access time
- Shared caches and memory
  - ⇨ easy to use, but highly complex to analyze (non-local effects!)

Avoiding Unwanted Interactions

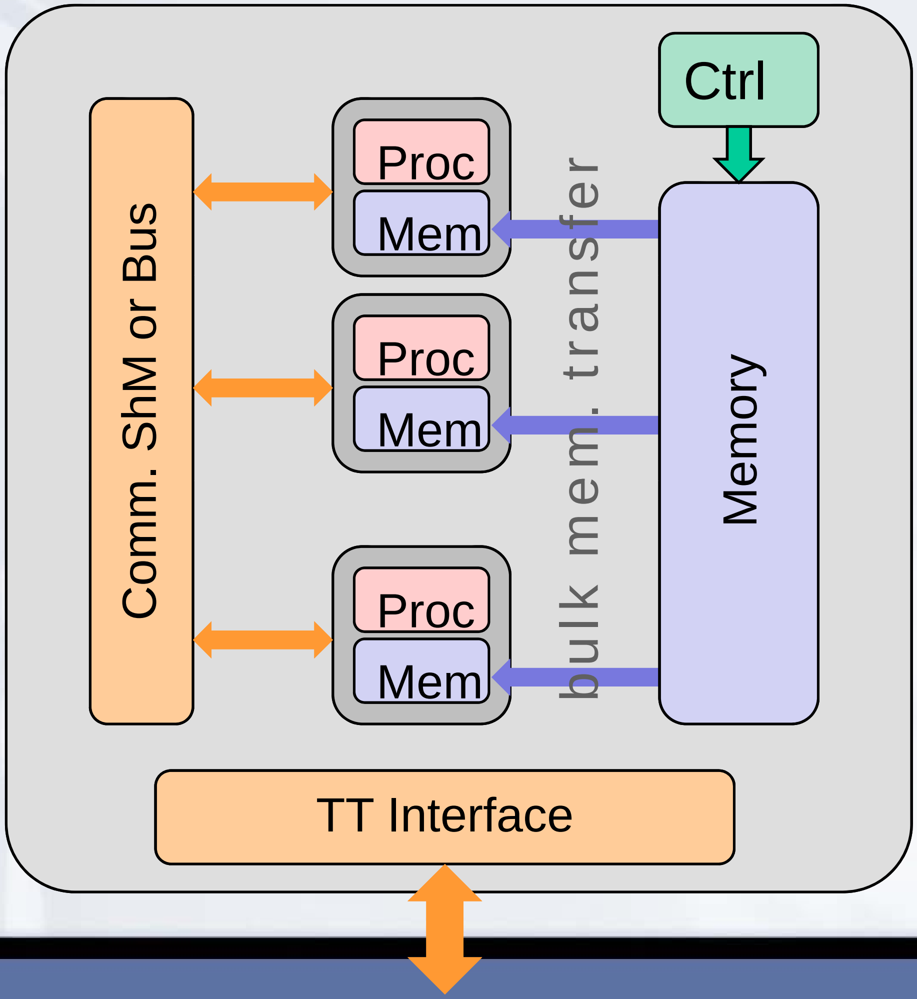
Protect time-relevant task state to make it predictable

- ⇨ spatial separation multiple cores; pre-fetching on cores
- ⇨ pre-planning instead of using dynamic run-time decisions

Mechanisms:

- ⇨ Use of single-path code (+ WCET-oriented programming)
- ⇨ No multi-threading on CMP cores (temp. isolation)
- ⇨ Use of simple, in-order pipelines
- ⇨ Strong use of local memory with prefetch update and statically scheduled access to shared memory/resources

MPSoC Structure



Summary

We aim at a clean and simple hierarchical design process

Properties: Composability, I/O compositionality, Stability of XT

Solution

- Task level: constant instruction XTs, in-order pipes, and single-path programming lead to invariable task XTs
  ⇨ simple WCET analysis, advantages for scheduling
- Application:
  Spatial isolation: multiple cores, local memory
  Temporal isolation: communication
  State update: pre-planned prefetching into local memory

… thank you!

http://ti.tuwien.ac.at/rts