Is Chip-Multiprocessing the End of Real-Time Scheduling?

Martin Schoeberl
Peter Puschner

Time-Predictable CMPs?

- Future CMP will contain hundreds of cores
- Embedded systems are a good fit for CMPs
- One thread per core in reach

⇨ Can we build a time-predictable CMP system?

⇨ What are the cost?

Schedule Memory Access, not Tasks

What if #cores > #tasks?

- Traditional task scheduling disappears
- Memory is shared ⇨ schedule memory accesses
  - TDMA based memory arbitration
  - Adapt (static) schedule to application needs
  - Context switch overhead of arbiter is small

  ⇨ Fine-grained scheduling of memory accesses

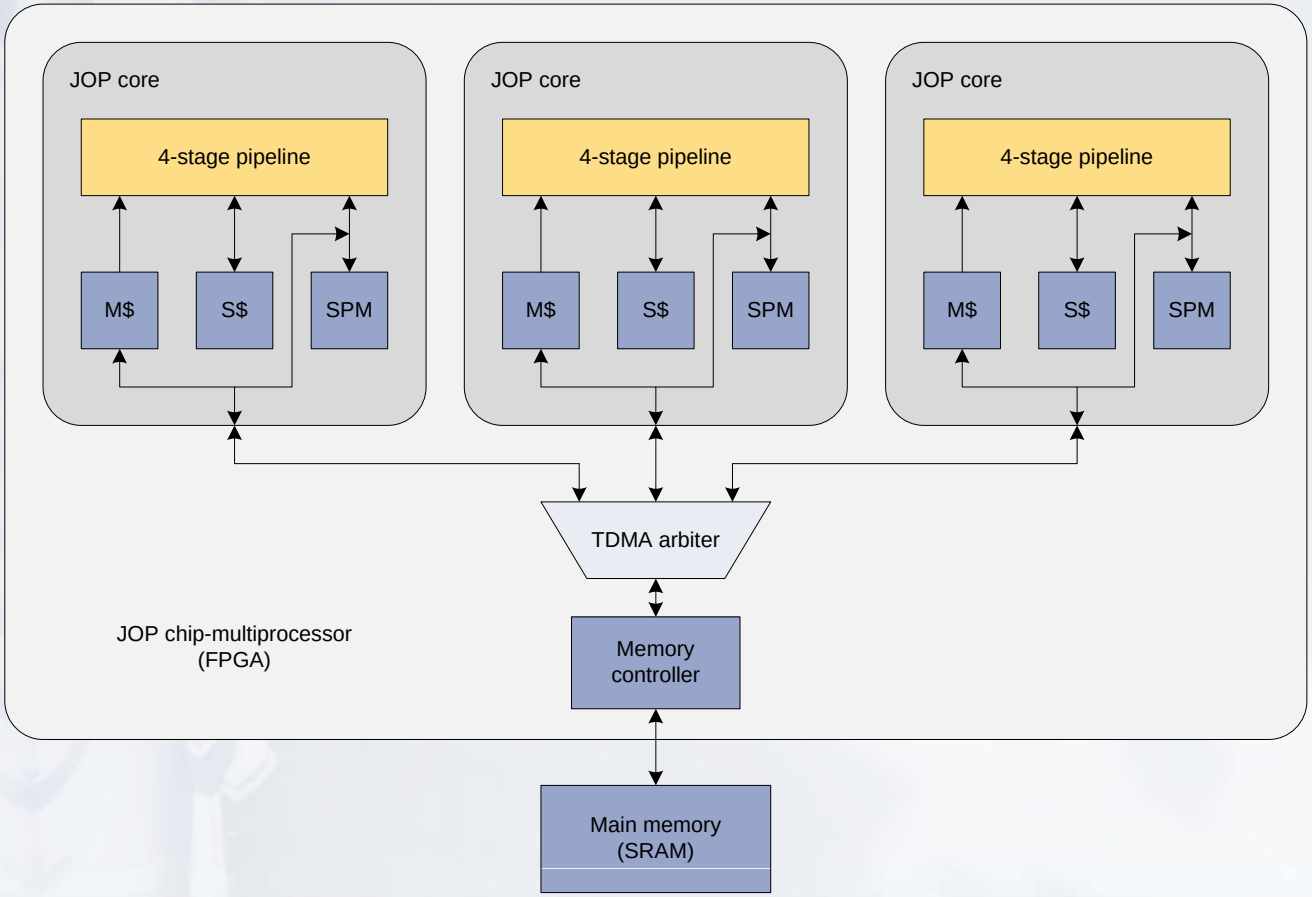
Memory-Access Schedules

- Tasks and memory-access schedules are not synchronized
- Tasks may have arbitrary periods
- One task per core ⇨ task can use 100% of its period
  - ⇨ Revision of memory-access schedule:
    - ⇨ Assign lower bandwidth to tasks with slack time
    - ⇨ Other tasks get additional bandwidth
- Use knowledge about memory-access schedule in WCET analysis
- Use results from WCET analysis to improve schedules

CMP System

- Multiple JOP cores (JOP … Java-optimized processor)
  - time-predictable JVM in hardware
  - local method cache (simplifies cache analysis)
  - local stack cache and scratch pad for local data
- Shared memory (4-cycle read, 6-cycle write)
  - TDMA memory arbitration
  - Lock-based synchronization (Java)

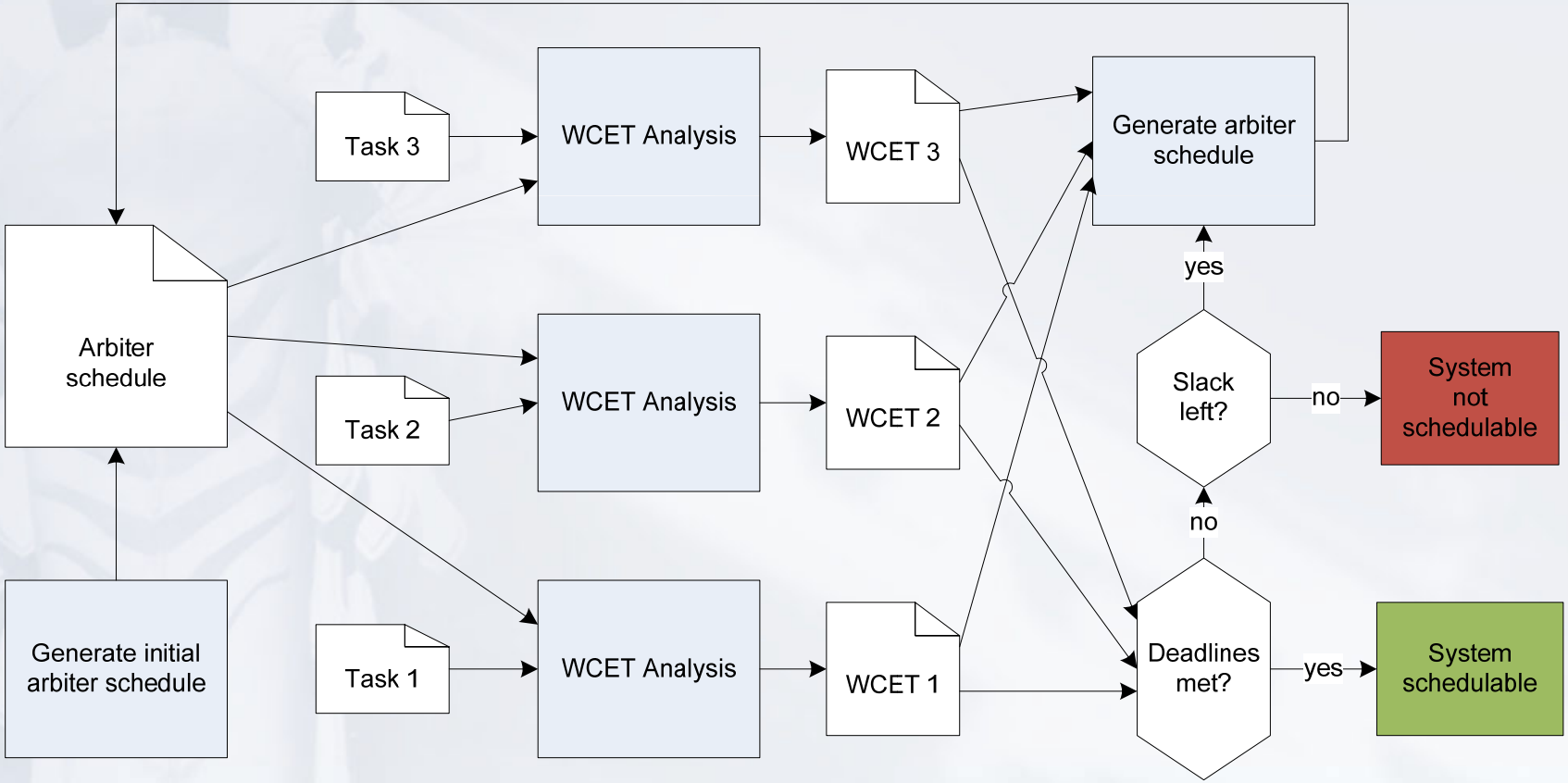
JOP CMP System



WCET Analysis

- At Java bytecode level
  - Standard IPET approach
- Simple bytecodes, constant execution time
- Memory access times
  - Depend on memory arbiter
  - TDMA schedule input for low-level analysis

TDMA Schedule Generation



Evaluation: Gain CMP vs. Single Core?

- 5 cores, one thread per core, same period
- Bus arbitration: RR, 6 cycles per thread

Function	Task	Single	5 cores	WCET increase
ipLink.run()	τ_1	1061	2525	2.4
resultServer()	τ_2	1526	3443	2.3
request()	τ_3	8825	23445	2.7
macServer()	τ_4	7004	17104	2.4
net.run()	τ_5	8188	18796	2.3
Sum		26604		

Variations of Arbiter Schedule

- Double-sized slots for long threads (1)
- Several (short) slots per thread per round (2, 3)
- Slot size variation: 6-/8-cycle slots (4, 5, 6)

Task	Equal	Schedule 1	Schedule 2	Schedule 3	Schedule 4	Schedule 5	Schedule 6
τ_1	2525	3605	3605	5045	4325	5285	5525
τ_2	3443	4865	4865	6761	5813	7077	7393
τ_3	23445	25313	20211	20201	20473	17955	18681
τ_4	17104	15284	14860	19468	14276	16756	17376
τ_5	18796	17856	16432	14832	16204	18924	18364

Speedup factor: 1.4 (26604 cycles for single core)

Conclusion

- Speedup of 1.4 on 5-core CMP is not impressive, but:
- Predictable task timing by TDMA mem. arbitration
- No scheduling and context switch overhead as in single-processor system (cost not included above)
- No cache thrashing (cost not included above)
- Task periods don't have to be harmonic
- Low complexity, low risk WCET & timing analysis!
  (JOP CMP and WCET tool available under GNU GPL)

… thank you!

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