No Coherence? No Problem!

Virtual Shared Memory for MPSoCs

Tobias Langer +, Jonas Rabenstein +, Timo Hönig *, Wolfgang Schröder-Preikschat +

\+ Friedrich-Alexander-University Erlangen Nuremberg

* Ruhr University Bochum

Increasing heterogeneity memory architecture

Great changes in computer architectures

- Increases in parallelism
- Increases in main memory size and performance
  - Non-volatile RAM with large, slow memory
  - High bandwidth memory fast memory local to the CPUs

Parallelism and new memory technologies require architectural changes

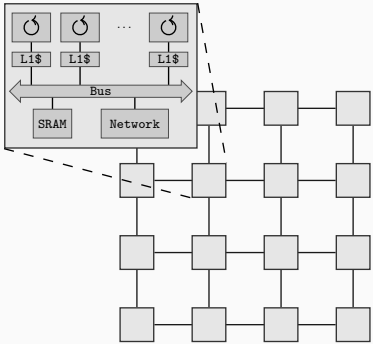
Tile-based systems

Organization in tiles

- Multiple cores
- Shared scratchpad memory
- Local bus
- Cache coherent
- Network adapter

System interconnect realized as NoC

- Communication traverses multiple hops
- Further away = higher latencies
- *No cache coherency*

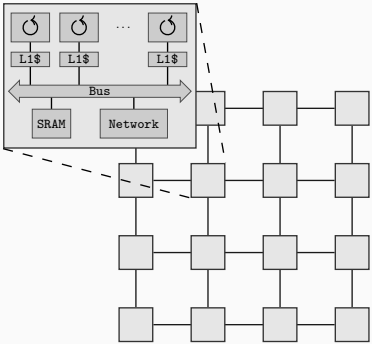


Tile-based systems

Organization in tiles
- Multiple cores
- Shared scratchpad memory
- Local bus
- Cache coherent
- Network adapter

System interconnect realized as NoC
- Communication traverses multiple hops
- Further away = higher latencies
- *No cache coherency*

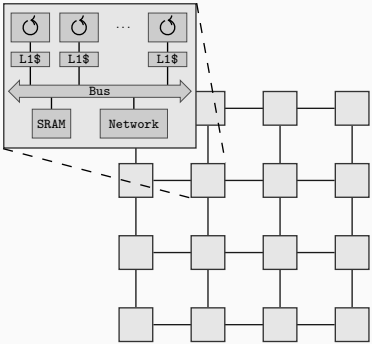
$\Rightarrow$ Effectively non-coherent NUMA



What is a suitable programming model?

Shared Memory?

- Requires implicit synchronization
- ✗ Requires coherency



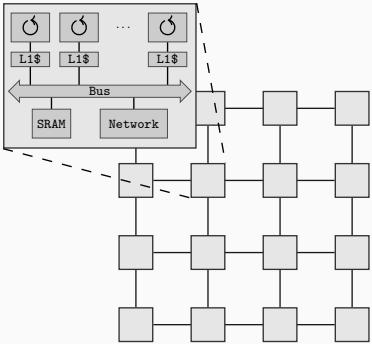
What is a suitable programming model?

Shared Memory?

- Requires implicit synchronization
- ✗ Requires coherency

Message Passing and PGAS?

- Requires explicit communication
- ✓ Require no coherency



What is a suitable programming model?

Shared Memory?

- Requires implicit synchronization
- ✗ Requires coherency

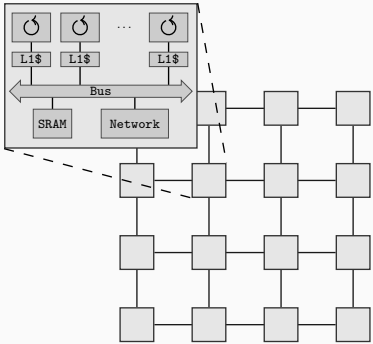
Message Passing and PGAS?

- Requires explicit communication
- ✓ Require no coherency

Message passing is no “silver bullet”

Explicit communication not always suitable

- Complicated for complex data structures
- High serialization/deserialization costs



Software-based coherency

Operating and runtime systems adapt to NUMA platforms

- Page placement and migration strategies
- First touch policies
- NUMA-awareness in allocators, schedulers, locking algorithms, ...

Software-based coherency

Operating and runtime systems adapt to NUMA platforms

- Page placement and migration strategies
- First touch policies
- NUMA-awareness in allocators, schedulers, locking algorithms, ...

⇒ Cross-domain coherency is not necessary most of the time

- Provide software component to implement coherency protocol if needed
- No hardware coherency mechanism required

Software-based coherency

Operating and runtime systems adapt to NUMA platforms

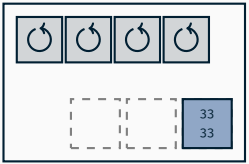
- Page placement and migration strategies
- First touch policies
- NUMA-awareness in allocators, schedulers, locking algorithms, ...

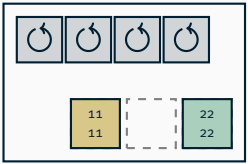
⇒ Cross-domain coherency is not necessary most of the time

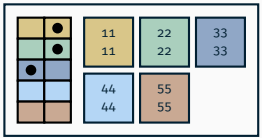
- Provide software component to implement coherency protocol if needed
- No hardware coherency mechanism required

⇒ Make use of techniques invented for Virtual Shared Memory

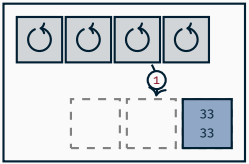
Memory page cache

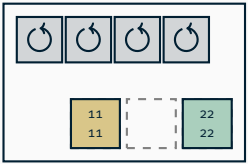


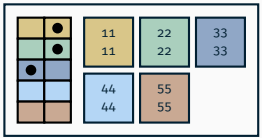




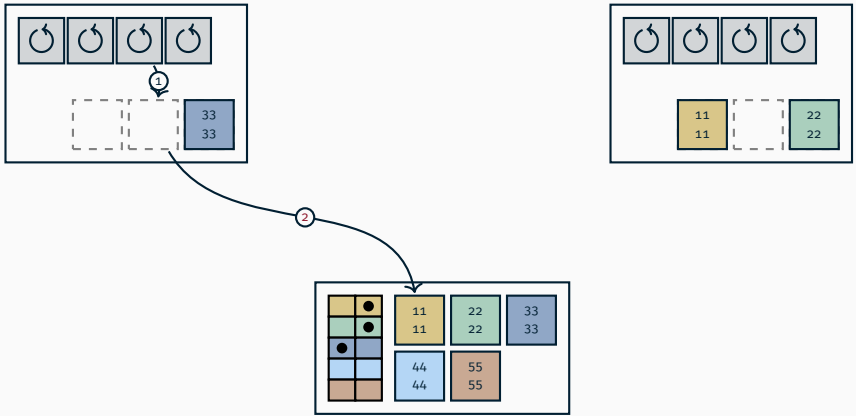
Memory page cache



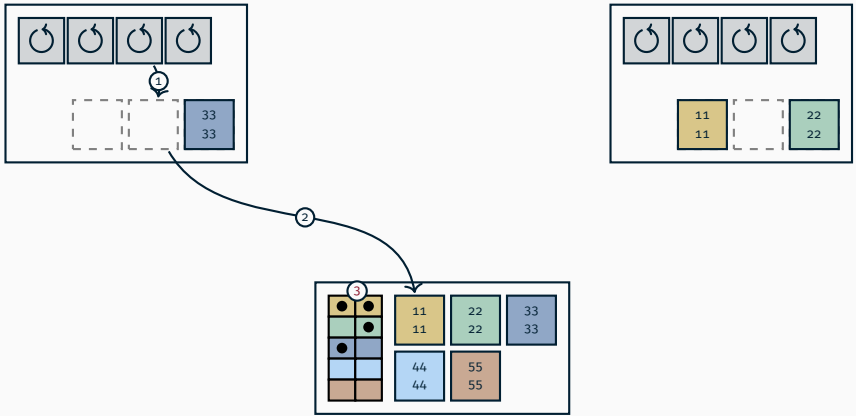




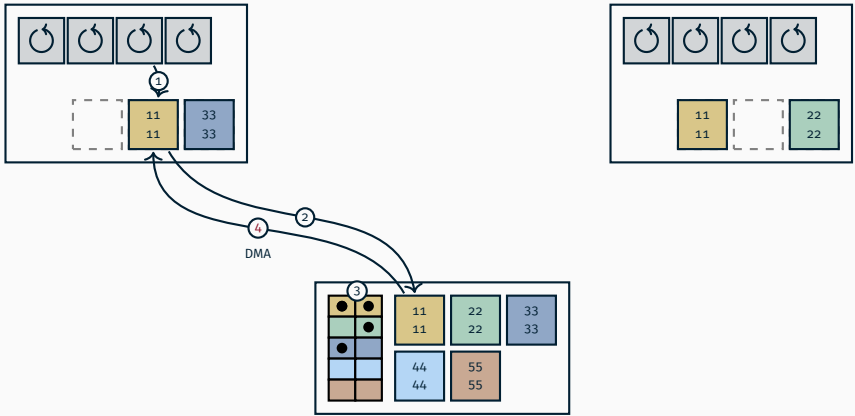
Memory page cache



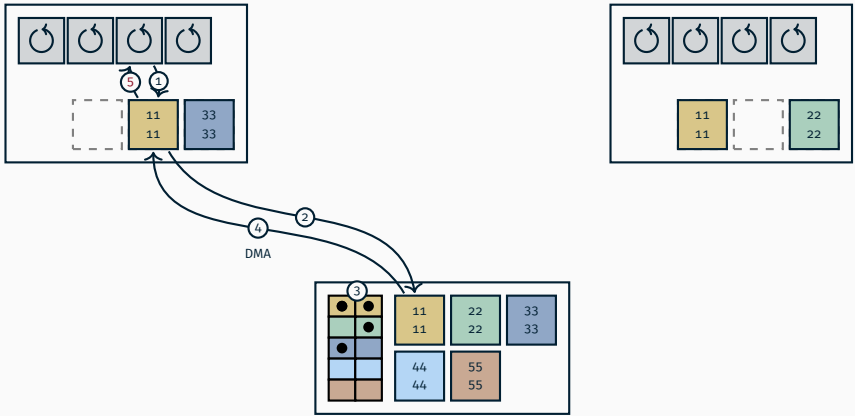
Memory page cache



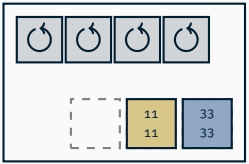
Memory page cache

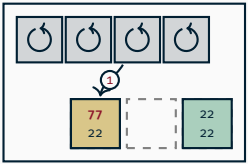


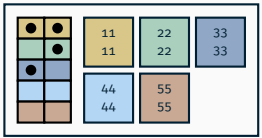
Memory page cache



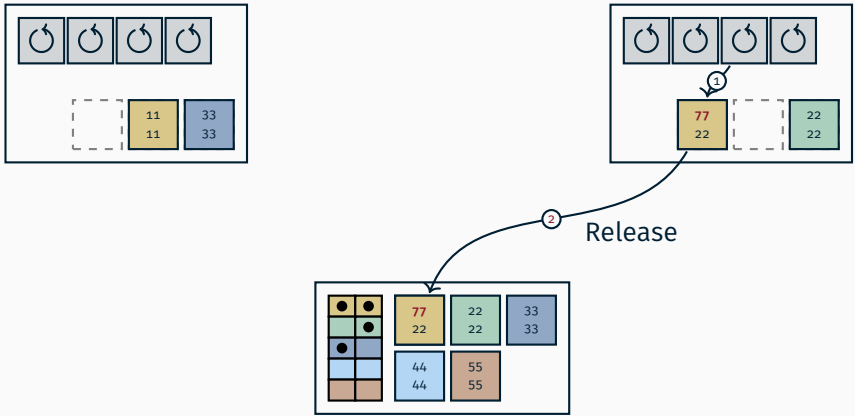
Coherency protocol



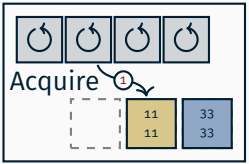


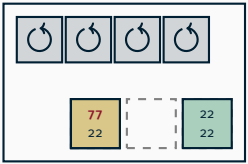


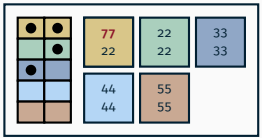
Coherency protocol



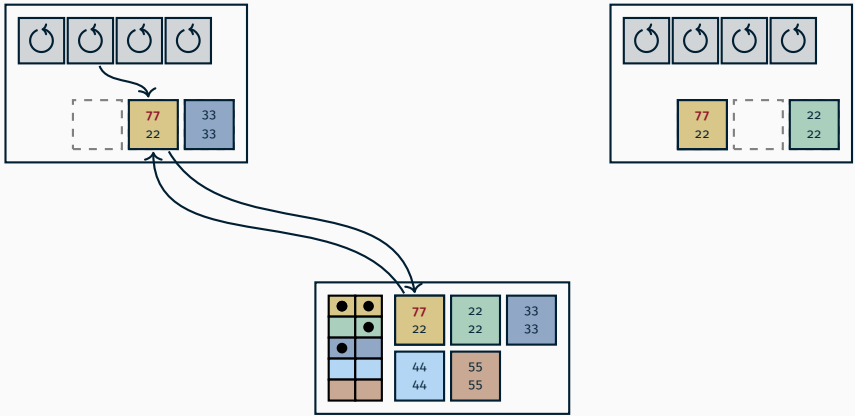
Coherency protocol







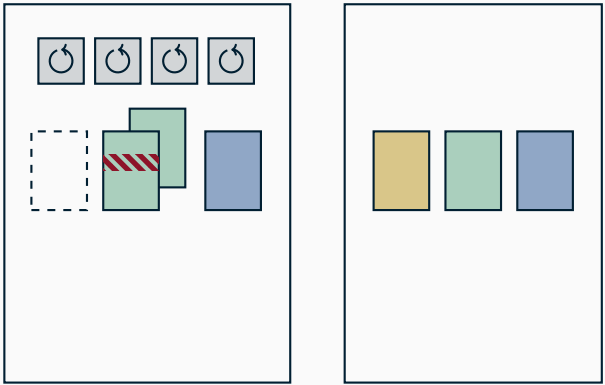
Coherency protocol



Synchronization of multiple writers

Multiple writers protocol

- Use *diff sets* to write back



Synchronization of multiple writers

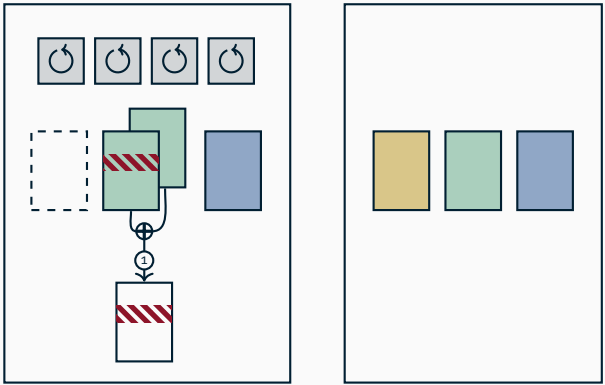
Multiple writers protocol

- Use *diff sets* to write back

⇒ Provide *golden copy* per page

⇒ Calculate diff against golden copy

⇒ Apply diff on original copy



Synchronization of multiple writers

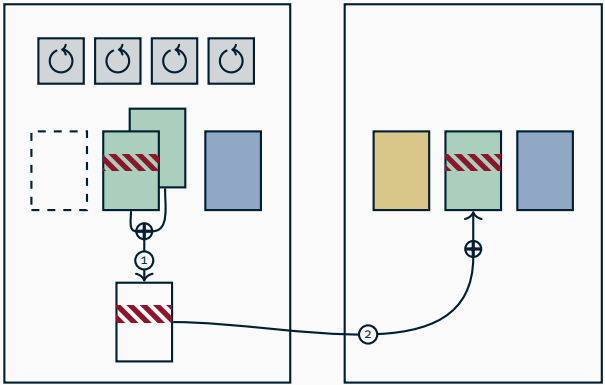
Multiple writers protocol

- Use *diff sets* to write back

⇒ Provide *golden copy* per page

⇒ Calculate diff against golden copy

⇒ Apply diff on original copy



Synchronization of multiple writers

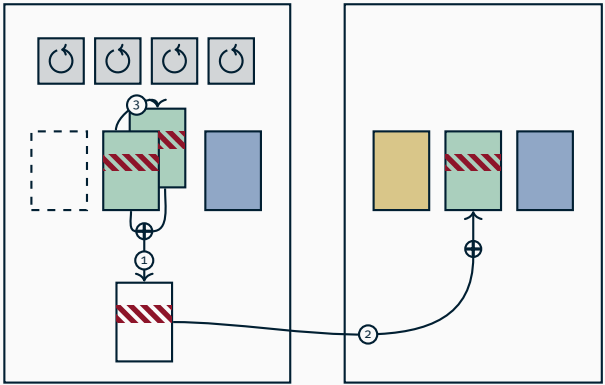
Multiple writers protocol

- Use *diff sets* to write back

⇒ Provide *golden copy* per page

⇒ Calculate diff against golden copy

⇒ Apply diff on original copy



Synchronization of multiple writers

Multiple writers protocol

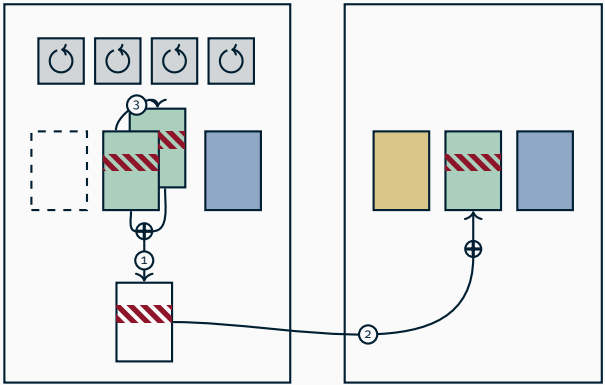
- Use *diff sets* to write back

⇒ Provide *golden copy* per page

⇒ Calculate diff against golden copy

⇒ Apply diff on original copy

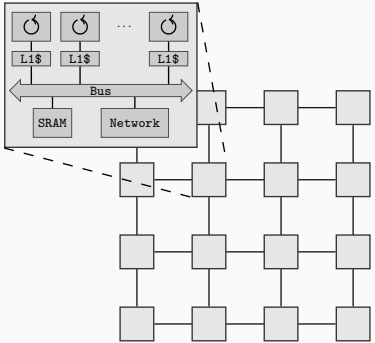
Requires properly synchronized data accesses



Evaluation Setup

FPGA prototype

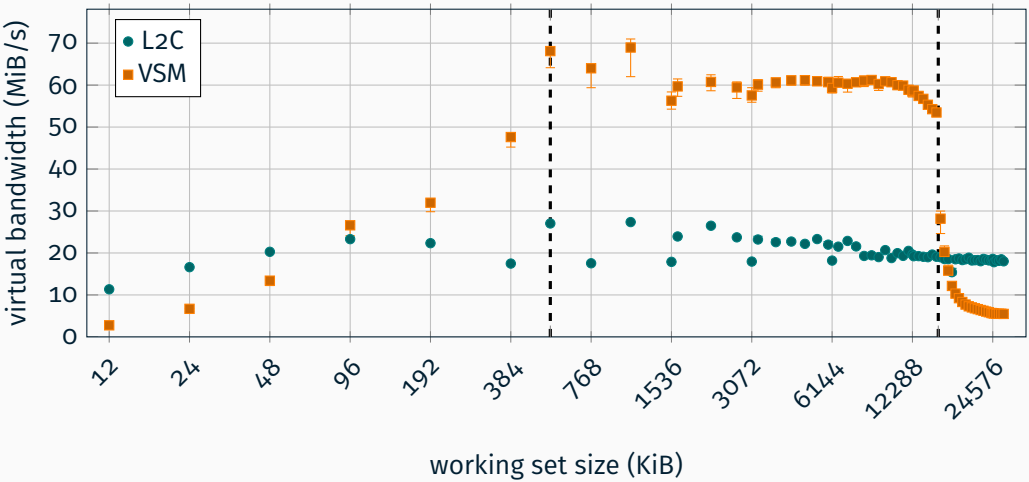
- 4x4 tile design
  - 1 tile used for memory only
  - 3 SPARC Leon 3 cores for applications
  - ⇒ total of 45 application cores
- 8 MB SRAM per tile
- 2 GB DRAM shared memory
- direct-mapped 130kB L2 cache



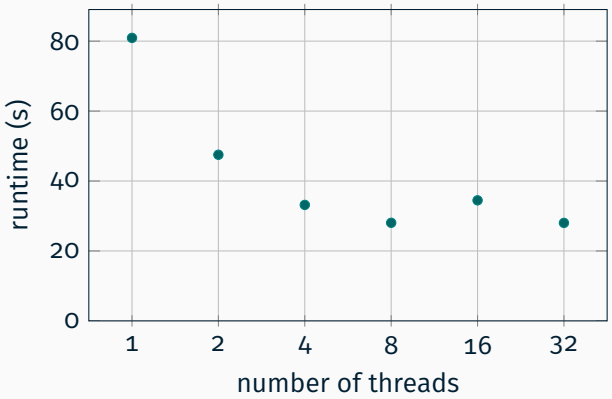
Coherency system

- Page cache with 256 entries
- Managed with LRU policy

STREAM Benchmark



Splash-3 Benchmark – LU Benchmark



Conclusion

Software-based coherency systems for tile-based MPSoCs

- On-demand coherency based on Virtual Shared Memory techniques
- Replacement for hardware-based coherency strategies

$\Rightarrow$ Allows for hardware designs without coherency protocols

Conclusion

Software-based coherency systems for tile-based MPSoCs

- On-demand coherency based on Virtual Shared Memory techniques
- Replacement for hardware-based coherency strategies

⇒ Allows for hardware designs without coherency protocols

Are we there yet?

- Yes
  - Software-based cache coherency is feasible
  - Restrictions: no atomic operations for shared data
- ... and No
  - Expensive due to frequent diff and copy operations

Future Work

- Reduce coherency system overheads
  - Write back changes ahead of time
- Application of additional hardware acceleration
  - Difference set calculation and transfer are ideal for near-memory computation
- Hybrid caching approach
  - Use currently unused L2 cache for unshared data
  - Use software-based mechanism for shared data
  - However, reduces benefits from exploiting the page cache

Thank you for your attention!

Conclusion

Software-based coherency systems for tile-based MPSoCs

- On-demand coherency based on Virtual Shared Memory techniques
- Replacement for hardware-based coherency strategies

⇒ Allows for hardware designs without coherency protocols

Are we there yet?

- Yes
  - Software-based cache coherency is feasible
  - Restrictions: no atomic operations for shared data
- ... and No
  - Expensive due to frequent diff and copy operations