KAHRISMA: A Multi-grained Reconfigurable Multicore Architecture

Lars Bauer, Ralf König, Muhammad Shafique, Timo Stripf, Waheed Ahmed, Juergen Becker, Jörg Henkel

Talk Outline

- Motivation
- KAHRISMA Architecture Template
- Compile-Time Software Framework
- Run-Time System
- Case Study
- Summary & Conclusion
Motivation

- Increasing diversity of embedded applications
- More complex functionalities integrated in mobile devices
- Applications’ Characteristics
  - Hardly predictable
    - Characteristics of upcoming applications
    - Combinations of applications running in parallel
  - Diverse processing behavior
    - Control flow vs. ILP, DLP

- Crucial design decisions can no longer be determined or even fixed at design time
- Runtime adaptive architectures show to be good candidates to provide the required flexibility

Reconfigurable Architectures – Overview

- There is a large variety of different architectures
  - PRISC, DISC, OneChip, Chimaera, XiRISC, RISPP, …
  - ADRES, CCA, Montium, Chameleon, PACT XPP, PipeRench, …
- However, each has its unique strength but also weakness
  - A single one of these reconfigurable architectures is not necessarily sufficient to satisfy the various future requirements

- To compensate for the weakness, typically different reconfigurable architectures (IP Blocks) are combined to realize a heterogeneous System-on-Chip (SoC)
State of the Art SoC – Limitations

Software

Application I - Domain 1
- RISC₁
- CI₂₁
- CI₁₁

Application II - Domain 1
- RISC₂
- CI₁₂

Potential Problem:
- If scenario has not been considered at compile time, both threads could claim same resources (IP 1) due to static mapping
- either one of the threads might be executed

Hardware

Heterogeneous SoC

- IP 1
- IP 2
- IP n

State of the Art SoC – Limitations

Software

Application - Domain 2
- RISC
- CI m
- VLIW

Application - Domain 3
- RISC
- RISC

Problem:
- IP cores can not be reconfigured to realize additional RISC instances, VLIW instances, or ‘CI m’ resources
- Applications might either not be executed at all or might not be able to fulfill QoS requirements

Hardware

Heterogeneous MPSoC

- IP 1
- IP 2
- IP n
State of the Art - Summary

- Today's SoC characteristics
  - Domain specific selection of components
  - Less efficient for running applications of different computational domains
  - Loosely coupling of architecture components
    - Missing tight coupling of coarse- and fine-grained components
  - Cannot bridge the gap between extremes
    - ILP/DLP rich vs. sequential treading enabled applications
  - Realization of heterogeneous processor instances not possible on coarse-grained / mixed-grained architectures
  - Difficult to program
    - different tools, missing integrated tool chain, efficient programming requires expertise in various reconfigurable architectures

Talk Outline

- Motivation
- KAHRISMA Architecture Template
- Compile-Time Software Framework
- Run-Time System
- Case Study
- Summary & Conclusion
KAHRISMA – Novel Contributions

- Architectural concept
  - Variety of reconfigurable modules realizing basic functionalities that dynamically can be combined
  - Tightly integrated coarse- and fine-grained reconfigurable data processing fabrics
    - Coarse-Gained and Fine-Grained Encapsulated Datapath Element (CG-, FG-EDPE)
  - On demand realization of different Instructions Set Architectures (ISA) as well as hardware-accelerators

- Compile-time software framework
  - ADL-based, retargetable

- Run-Time System
  - exploiting the high degree of parallelism

The KAHRISMA Architecture - Overview

1. Instruction Fetch & Align: Cache access, extraction of the actual instruction packets
2. Instr. Analyze & Dispatch: Extraction of the individual Operations out of an instruction packet
   - Dispatching of instr. to EDPEs
   - Flow-Control, handling of Interrupts, Exceptions etc.
3. EDPE Array:
   - Realization of typical ALU operations with DSP optimizations
4. Memory Subsystem:
   - Load/Store Units
   - Multibanked cache
EDPE Array: Implementing ISAs and hardware accelerators

- FG-EDPEs are FPGA-like reconfigurable fabrics, optimized for bit/byte level operations, state machines etc.
- CG-EDPEs are ALU-like reconfigurable fabrics, optimized for word/sub-word level operations

The KAHRISMA Architecture – Appl. Scenario
Talk Outline

- Motivation
- KAHRISMA Architecture Template
- Compile-Time Software Framework
- Run-Time System
- Case Study
- Summary & Conclusion
Design/Compile-Time Tools

- Integrated tool chain
  - to simplify programming
  - to optimally exploit a flexible, heterogeneous architecture
- Substantial features
  - ADL based
    - Specification of Coarse-Grained (CG) & Fine-Grained (FG)-EDPEs and interconnect network
  - Automated application partitioning on code level: Processor Mode vs. Array-Mode
    - Processor Mode: selection of appropriate ISA
    - Array Mode: Custom Instruction (CI) detection and mapping
- Simulator
  - Supporting application development
  - Design-Space exploration

Compile-Time Toolflow

- Tool input
  - architecture description, code
- Machine independent optimizations
  - Dead code elimination, constant propagation, inlining etc.
- CI Identification
  - Array Mode implementation of computational hot-spots
- Compilation Backend
  - Code partitioning on ISA level
- Binary Utilities
  - ISA assembler, coarse-, fine- and mixed-grained linking
CI Identification, Selection and Mapping

- **Automatic CI Identification**
  - Identification of possible CIs for mapping to FG- and/or CG-EDPEs

- **Hierarchical Composition**
  - Decomposition of the CI graph
  - Selection of beneficial CIs
  - Creation of multiple implementation versions

- **Mapping of CIs**
  - Architecture dependent
  - Configuration emission

- **Library construction**
  - Faster identification and selection of created CIs for other applications

Compilation Backend

- **Extension of classical backend**
  - Each pass relies on information from ADL
  - Identification of control- vs. data-flow rich application sections
  - Selection of appropriate ISA
    - RISC for control-flow rich application sections
    - \( n \)-issue VLIW for data-flow rich application sections

- **Emission of implementation alternatives**
  - Fastest execution
  - Lowest hardware utilization
  - Variants in-between
Talk Outline

- Motivation
- KAHRISMA Architecture Template
- Compile-Time Software Framework
- Run-Time System
- Case Study
- Summary & Conclusion

Run-Time System Tasks

- Selection of most suited candidate from (compile-time prepared) CI implementation alternatives
  - Ranging from a software implementation up to a fast hardware implementation
  - Regarding the system status (battery level, user-defined performance constraints, current EDPE utilization, …)
- Binding to physical resources
  - Establish configurations on the architecture
Run-Time System – Overview

1. Request of new thread/application
2. Selection of implementation alternatives (ISAs and/or CIs)
3. Binding of required instances to physical resources
   - Steps 2 and 3 are assisted by online monitoring
   - Persistent online optimization

Online Monitoring
- CI execution frequency
- EDPE utilization

Multi-Grained CI Selection
- Determining specific EDPEs
- Connecting the EDPEs

Binding to EDPEs
- EDPE array
- Instruction decoding

Situation-Dependent Optimizations
- Array defragmentation
- Algorithm parametrization
- EDPE sharing

Legend:
- Data- and decision-flow to establish new EDPE configurations
- Trigger from thread
- Monitoring Data
- Optimizations: interacting with components and parametrizing the algorithms

Run-Time System – Overview (cont’d)

Online Monitoring
- Reconfiguration requires time
- Prediction of execution frequencies, threads deadlines, and system states
- Look ahead reconfiguration of processor resources to start computation as early as possible (prefetching)
- Persistent online optimization, e.g. array defragmentation

Selection
- Considering priorities and deadlines of the currently executing threads
- Considering communication requirements – spatially adjacent EDPEs, locality to memory
- Considering reconfiguration time of different modules

Binding
- Based on a hierarchical approach: determine a region before determining EDPEs within that region
Talk Outline

- Motivation
- KAHRISMA Architecture Template
- Compile-Time Software Framework
- Run-Time System
- Case Study
- Summary & Conclusion

Case Study

- Security domain: AES
- Multimedia domain: Deblocking Filter, CABAC
- Mapping of fine-/coarse-/multi-grained CIs has been done manually
  - Will serve as later comparison for the tool chain
- First implementation results for CG-EDPE
  - 90nm TSMC - tcbn90gwc
  - Speed: 500 MHz
  - Area: 536,007 μm²
- FG-EDPE clock derived from CG-EDPE clock
  - Clock ratio CG-EDPE/FG-EDPE: 5/1
  - FG-EDPE Frequency: 100 MHz
Adavanced Encryption Standard (AES)

- Symmetrical Block Cipher
- Fixed Data Block Size: 128 Bit
- Variable Key Block Size: 128, 192, 256 Bit
- Depending on the Key Block Size, the inner loop will be executed
  - 9 x – 128 Bit
  - 11 x – 192 Bit
  - 13 x – 256 Bit

AES - Single EDPE CI Implementations

- Processing time
  - 1 FG-EDPE: 550 Cycles
  - 1 CG-EDPE: 332 Cycles
  - Note: All execution times are given for the 500 MHz clock for the CG-EDPEs

Ressource Requirements:
1 FG-EDGE

- ARK 2 cycles
- SB 4 cycles
- SR 2 cycles
- MC 20 cycles
- KE 7 cycles
- ARK 2 cycles
- SB 4 cycles
- SR 2 cycles
- KE 7 cycles
- ARK 2 cycles

1 CG-EDGE

- SB
- SR
- MC
- ARK

Processing time

- 1 FG-EDPE: 550 Cycles
- 1 CG-EDPE: 332 Cycles
- Note: All execution times are given for the 500 MHz clock for the CG-EDPEs
AES - Multi EDPE CI Implementations

- Processing time
  - 2 CG-EDPEs: 208 Cycles
  - Multi-Grained (2 CG, 1 FG): 163 Cycles

<table>
<thead>
<tr>
<th>Ressource Requirements:</th>
<th>2 CG-EDPEs</th>
<th>2 CG- and 1 FG-EDPEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st CG-EDPE</td>
<td>ARK 2 cycles</td>
<td>ARK 2 cycles</td>
</tr>
<tr>
<td>2nd CG-EDPE</td>
<td>SB 4 cycles</td>
<td>SB 4 cycles</td>
</tr>
<tr>
<td></td>
<td>SR 2 cycles</td>
<td>SR 2 cycles</td>
</tr>
<tr>
<td>Communication 2 cycles</td>
<td>MC 10 cycles</td>
<td>Communication 2 cycles</td>
</tr>
<tr>
<td></td>
<td>ARK 2 cycles</td>
<td>ARK 2 cycles</td>
</tr>
<tr>
<td></td>
<td>SB 4 cycles</td>
<td>SB 4 cycles</td>
</tr>
<tr>
<td></td>
<td>SR 2 cycles</td>
<td>SR 2 cycles</td>
</tr>
<tr>
<td>Communication 2 cycles</td>
<td>MC 10 cycles</td>
<td>Communication 2 cycles</td>
</tr>
</tbody>
</table>

- RISC vs. multiple CI implementations

KAHRISMA Run-Time System selects most appropriate alternative from the Area-Performance Design Space at run-time.
Summary & Conclusion

- We invented a novel hardware architecture concept
  - Tightly integration of coarse- and fine-grained run-time reconfigurable fabrics
  - On-demand realization of different ISAs and/or CIs
  - Constituting KAHRISMA’s Hypermorphism
- Supporting Compile-Tools and Run-Time System
- Allowing to match the applications’ requirements much better
- Better hardware utilization compared to existing heterogeneous reconfigurable SoC approaches
- The approach allows for “soft” migration of applications
  - Hotspots can be realized as standard C(++) version running on a RISC core
  - Alternatively, they can be mapped to a VLIW processor
  - Afterwards, most critical parts can still be realized as sophisticated hardware accelerators (CIs)

KAHRISMA: A Multi-grained Reconfigurable Multicore Architecture

Lars Bauer, Ralf König, Muhammad Shafique, Timo Stripf, Waheed Ahmed, Juergen Becker, Jörg Henkel

Thank you for your attention!