A dynamically reconfigurable architecture concept for efficient wireless embedded systems

Heiko Hinkelmann, Peter Zipf, François Philipp, Manfred Glesner

Institute of Microelectronic Systems, Technische Universität Darmstadt
Karlstrasse 15, 64283 Darmstadt, Germany
{hinkelmann, zipf, francoisp, glesner}@mes.tu-darmstadt.de
Outline

• Project overview

• Architecture concept for energy-efficient dynamically reconfigurable systems: review and generalisation

• Latest results and proof of concept

• Conclusion and outlook
Project Overview: 1. Phase

- Reconfigurable hardware platforms for the MAC layer of WLAN systems
  - A dynamically reconfigurable function unit (RFU) for computation-intensive MAC layer tasks has been designed and evaluated
  - Efficient re-use of the RFU’s hardware resources by dynamic reconfiguration during runtime
    - High functional density
    - Very good area efficiency: Lower Area consumption than ASIC reference
    - High performance gains: 25x faster than processor architectures
Project Overview: 2. Phase

- Dynamically reconfigurable systems for wireless sensor networks
  - Key question: can a dynamically reconfigurable architecture fit the requirements of a small, highly energy-efficient and yet flexible system platform?

- A new type of RFU architecture for computation-intensive WSN tasks has been designed and evaluated
  - Approach: Frequent runtime reconfiguration of a small data path
    1) Coarse grain hardware structures provide near-ASIC efficiency
    2) Frequent runtime reconfiguration is exploited to keep the data path small and flexible

- Important results:
  - A novel reconfiguration mechanism provides very low reconfiguration costs
  - High energy and performance gains compared to processors
  - Programmability leads to lower costs and time to market than ASICs
Design Space: Energy vs. Performance

- RFU achieves 1 order of magnitude improvement over processors, bridging the gap to highly efficient but non-programmable ASICs.
Project Overview: 3. Phase

- **Generalisation of the results and design concepts**
  - A universal template for our runtime reconfiguration mechanism was derived and analysed in comparison to existing reconfiguration techniques
    - Superior efficiency for frequent runtime reconfiguration
    - Much lower reconfiguration costs than existing techniques
    - Wide applicability to other reconfigurable architectures, even to FPGAs*
  - Extension of the RFU test cases
    - Compatibility with different processor platforms shown
    - RFU data path extended for supporting common digital signal processing functions
      → Paving the way for smart WSNs
    - Proving the generality and portability of the RFU concept
      → The high energy efficiency is indeed an architecture-characteristic feature which can be obtained independent of functionality, application domains and processors used

RFU Architecture Concepts

• Objective for designing the RFU:
  – Combining small size, high energy efficiency, and flexibility

• Solution found with a new architecture concept:
  – Frequent runtime reconfiguration of a small coarse-grain data path

1) A small reconfigurable data path is designed
   • Coarse grain hardware structures provide near-ASIC efficiency

2) Frequent runtime reconfiguration is exploited to keep the data path small and flexible
   • Efficient reuse of sparse hardware resources
   • Intra-task resource sharing: reusing operators for different computation steps
   • Inter-task resource sharing: reusing the entire RFU for different tasks

The Coarse-grain Data Path of the RFU

All function blocks and interconnects are dynamically reconfigurable in every clock cycle.

Multiply-Accumulate Module:
- 16 multipliers and 16 adders
- 8-bit operations each

four special inversion operators

flexible bus system as top-level interconnect

256 byte local memory
LUT or FIFO

RFU Inputs
RFU outputs
3 global buses

Multiply-accumulate module

Inversion module
Register module
Memory module

Memory Access Unit

to main memory

to main memory

all interconnections on top-level are 32 bit wide (or 4x8 bit after splitting up within modules).
The RFU data path is controlled by a set of registers storing the actual configuration:

- Each register controls a single component of the data path.
- By changing the register contents, the components are reconfigured dynamically during runtime. Each component can be reconfigured in every clock cycle.

• **Dynamic reconfiguration:**
  – Each configuration register is associated with a *multi-context table* (MCT)
  – For each register, a new configuration can be selected from its MCT, possibly in every clock cycle
  – Context selection is done individually for each MCT!
  → *Fast, dynamic, partial, multi-context* reconfiguration
Multi-Context Configuration Table (MCT)

- **Solving the control problem:** How can the *individual context selection* in each MCT be realised without massive control overhead?
  - Each MCT entry is extended by a tag field, which is stored together with the configuration data.
  - A context is selected if a global marker matches a local tag entry.

![Diagram of Multi-Context Configuration Table (MCT)]

- Configuration Context #1
- Configuration Context #2
- Configuration Context #3
- Configuration Context #4

**Register**

- Configuration Data #2

- **01011** = global marker
The global marker is generated centralised from a reconfigurable LUT, which is controlled by a run control unit to autonomously define the reconfiguration schedule during RFU tasks.
Processor Integration

- Our RFU is integrated into the data path of a regular RISC processor
  - Used as reconfigurable hardware accelerator for data processing, for improved performance & efficiency

- Control via instruction set extension
  (a) Two execution instructions:
     - Single-cycle execution step
     - Multi-cycle operation sequence
     - (intra-task reconfiguration is handled automatically in the background, totally transparent to a programmer)
  (b) One configuration instruction
     - Initiates an inter-task reconfiguration, (which is completed autonomously by the RFU then)
     - Only parameter: the configuration memory start address for the desired task (like a unique task ID)
Our Reconfigurable Mote Architecture

- Reconfigurable architecture based on a hybrid computing core
  - A Reconfigurable Function Unit (RFU) is integrated into the microprocessor
  - The RFU is used as accelerator for data processing at low energy costs

![Diagram of LEON2 processor core](image)

- Config. Memory
- Instruction Memory
- Data Memory
- UART
- Bus Arbiter
- Sensor Interface
- Transceiver Interface
- FEC
- Timer
- Interrupt-Controller
- Wishbone Bus

can be realized as a single chip
Design Space: Energy vs. Performance

- RFU achieves 1 order of magnitude improvement over processors, bridging the gap to highly efficient but non-programmable ASICs.
Outline

• Project overview

• Architecture concept for energy-efficient dynamically reconfigurable systems: review and generalisation

• Latest results & proof of concept
  – Why does this concept work so well?
    → Detailed analyses of the energy costs for reconfiguration
  – Would it work also for other reconfigurable architectures?
    → Compatibility with different processor platforms?
    → Portability to other application domains?

• Conclusion
Power Consumption: An example

Power Consumption: An example

- **RFU data path**
- **RFU control & reconfiguration**
- **Processor**

*Dynamic Reconfiguration Costs*

**Power Consumption: An example**

- **Power Consumption**
  - **Time** [µs]
  - **Power** [mW]
  - RFU data path
  - RFU control & reconfiguration
  - Processor

- **Time** [µs]
  - 0
  - 5
  - 10
  - 15
  - 20
  - 25
  - 30

- **Power** [mW]
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5

- **RFU**
  - previous task
  - Conf.
  - Exec.
  - Conf.
  - Exec.

- **CPU**
  - previous task
  - AES keygen.
  - AES encryption
  - next task

**AES keygen.**

**AES encryption**
Energy Consumption Results

- Comparison of the LEON2 platforms:
  - The RFU reduces the energy consumption of the LEON2 by factor 4 to 6

Benchmark Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>ASIC version</th>
<th>RFU version</th>
<th>RFU incl. reconfiguration</th>
<th>Software version</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC128</td>
<td>13.8</td>
<td>2.3</td>
<td>8.6</td>
<td>13.3</td>
</tr>
<tr>
<td>BCH(15,10,3)</td>
<td>56.7</td>
<td>5.8</td>
<td>66.4</td>
<td>13.3</td>
</tr>
<tr>
<td>AES keygen.</td>
<td>8.6</td>
<td>5.8</td>
<td></td>
<td>21.0</td>
</tr>
<tr>
<td>AES encrypt.</td>
<td>8.6</td>
<td>5.8</td>
<td></td>
<td>52.5</td>
</tr>
</tbody>
</table>

Energy Consumption Results

- Comparison of the LEON2 platforms:
  - The RFU reduces the energy consumption of the LEON2 by factor 4 to 6
Similar characteristics are measured for the 8-bit ATmega AVR platforms: the architecture-specific differences are platform-independent.
RFU Extension

• Second case study: local signal processing for smart WSN applications
  – Requires the support of integer (fix-point) arithmetic in the RFU
• Three new reconfigurable modules added to the RFU data path
  – Multiply-Accumulate (MAC) unit
    • One 32-bit fix-point MAC-operation per cycle
    • Two 16-bit fix-point MAC-operations per cycle (in parallel)
    • One 16/16-bit MAC-operation of complex numbers per cycle
    • With variable fix-point each
  – 32-bit Division unit
  – 32-bit Square Root unit

• Benchmark extensions:
  – Previous set: {CRC-8, CRC-32, BCH-(15,5,3), BCH-(63,45,3), AES}
  – Additional benchmark tasks: {FFT-64, FIR-24, FIR-8, IIR-4, Localisation}
• The ASIC reference architecture is extended accordingly
Costs of the extended functionality

Cost advantage: The RFU requires much less area than the ASIC for adding new functionality!

Scalability advantage: Adding new functionality decreases the overall energy efficiency of the ASIC architecture, whereas the efficiency of the RFU remains stable!
Characteristic design space regions can be observed for each architecture. On average, the integration of a RFU into a processor platform leads to performance and energy efficiency gains of one order of magnitude each.
Conclusions

• The proposed RFU architecture concept is well suitable for designing area- and energy-efficient embedded systems
  – A programmable RFU platform is much more flexible and less costly than ASIC solutions, but still highly energy-efficient
  – High energy-efficiency and performance gains over processor platforms
  – Lower area consumption than ASIC reference

• These benefits are architecture-characteristic and more or less independent of specific functionality and applications

• The concepts developed and analysed in this project can serve as design guides for future embedded systems requiring the combination of high efficiency and programmability
  – Good portability and scalability of all parts
  – A novel reconfiguration mechanism enables efficient dynamic reconfiguration at very low overhead
Outlook

• Architectures like the RFU are promising solutions to enable new WSN applications with smart functionality in the future
  – by providing high processing power at low energy consumption,
  – thus reducing the costs for local data processing tremendously

• Future research on compilers for this type of architecture is required, particularly supporting its dynamic reconfiguration capabilities

• Next generation of dynamically reconfigurable multi-context FPGAs?
  

• Continuation of the WSN research topic: “Maintenance on Demand”
  – Research project funded by the EU
  – Consortium of ten partners from industry and academia
  – Taking the obtained knowledge into industry products
Acknowledgements

• Many thanks to the DFG, the organisers of the SPPRR, and all participants for supporting our research work!
  – 6 years of funding
  – ca. 30 publications in international conferences and journals
  – ca. 25 bachelor / master theses
  – 2 PhD theses
  – Many successful cooperations within the SPPRR and with international partners