RWTH AACHEN UNIVERSITY

MPSoC 2014

Efficiency, Flexibility and Ease of Programming

A comparison of two opposite architectural approaches

Gerd Ascheid, Xiaolin Chen, Daniel Günther

Institute for Communication Technologies and Embedded Systems

Agenda

- The two compared architectural approaches
- Efficiency, flexibility, and programmability
- Summary

Motivation

- **Flexible accelerator for high data rate wireless baseband processing**
  - Coarse grained reconfigurable array
    - In particular supporting matrix-vector ops (inversion!)
    - Basic processing elements are
      e.g. (complex) multipliers, square root, …
    - Fixed point arithmetic for efficiency
    - How close can you get to dedicated ASIC performance?

- **Challenge: How does the CGRA performance compare to an optimized processor?**
  - „Standard“ features, e.g. SIMD plus
  - Conversion of float to fixed not trivial
    (numerical stability e.g. of matrix inversion)
    -> Optimized floating point performance

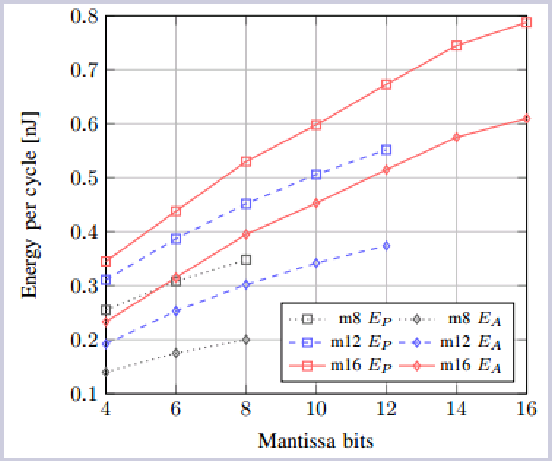
Lightweight SDR ASIP

Make the simplest architecture that can do the job as efficient as possible

- **Focus on programmability, flexibility and usability**
- **Slim, efficient core – Duplicate for high throughput**

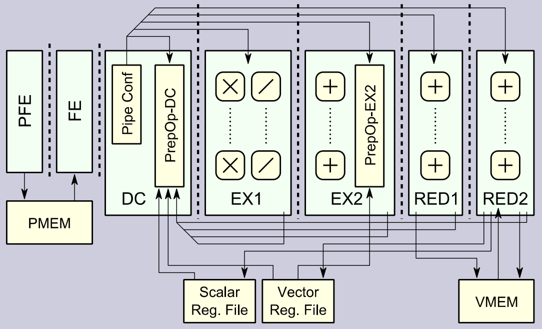
Arithmetic

- **Floating-point** arithmetic for programmability and numerical stability
- **Energy tuning** by bit-granular operand masking



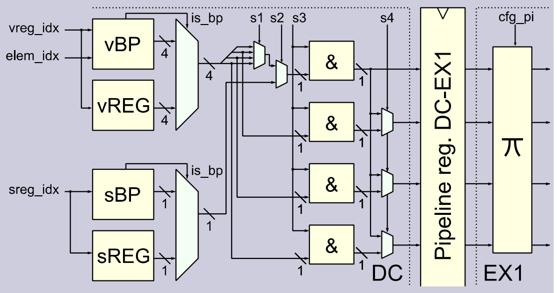
Structure

- **SIMD** for vector arithmetic
- **Pipelined ALU** (4-stage) due to floating-point HW complexity
- **Vector register file** partitioned for element-wise access (flexibility & energy optimization)



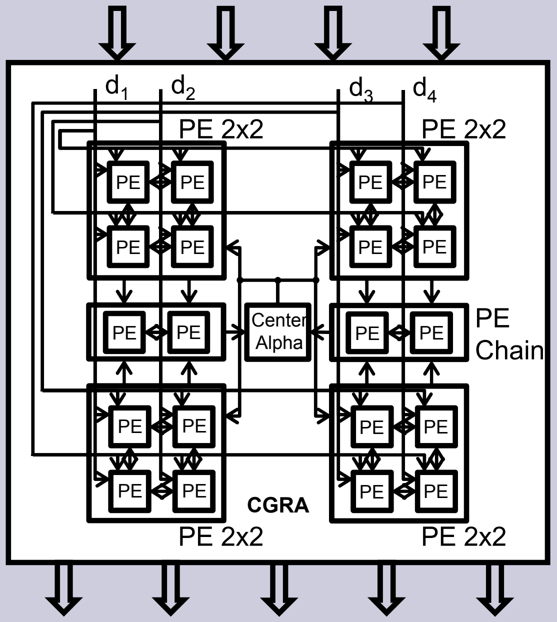
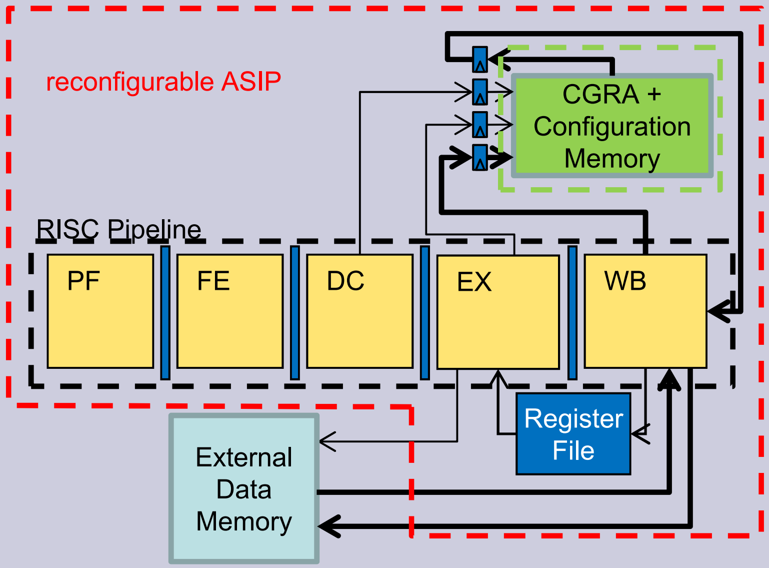
Data Path Routing

- **Access schemes** to support wide range of operations
- **Permutation units** for vector arithmetic
- **Intelligent bypassing** for multiple-length instructions



re-configurable ASIP with CGRA Accelerator

- **Accelerate data path through supporting complex matrix operations on CGRA**
  - Including matrix inversion
- **Special instructions in processor to implement control path**
  - Load/Store with wide bandwidth from/to external memory
  - Scheduling of matrix operation on CGRA by re-configuration

CGRA Accelerator	**re-configurable ASIP**
	

Agenda

- The two compared architectural approaches
- **Efficiency, flexibility, and programmability**
- Summary

Efficiency comparison: Linear MIMO detection

Technology scaling to 90nm → qualitative statement

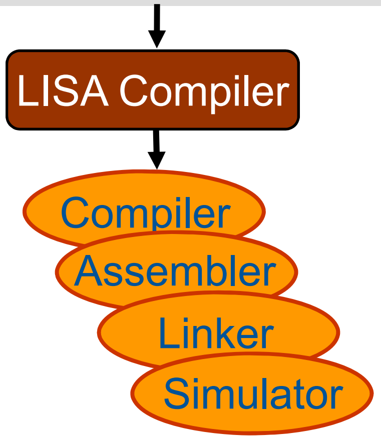
- **Flexible number format of napCore allows tuning of energy consumption**

	rASIP	napCore				MMSE PIC	Rank1 Inv
Process	65nm SP	90nm SP				90nm	250nm
Implementation	layout	layout				tapeout	synthesis
Architecture class	rASIP	flp. SIMD				ASIC	ASIC
Iterative	no	yes				yes	no
Clock [MHz]	400	400				568	167
Core voltage [V]	1.0	1.0				1.2	N/A
Scaled clock [MHz]	278	400				568	464
Scaled power [mW]	240		**4Q**	**16Q**	**64Q**	189.1	N/A
		it0	112	147	153		
		it1	98	111	124		
Area [kGE]	458	123				400	
Cycles per EQ	18	81		115		18	
Energy efficiency [vec/uJ]	**67**		**4Q**	**16Q**	**64Q**	**167**	N/A
		it0	**44**	**34**	**32**		
		it1	**36**	**30**	**28**		
Area efficiency [vec/s/GE]	**35**	**40**		**27**		**79**	**51**

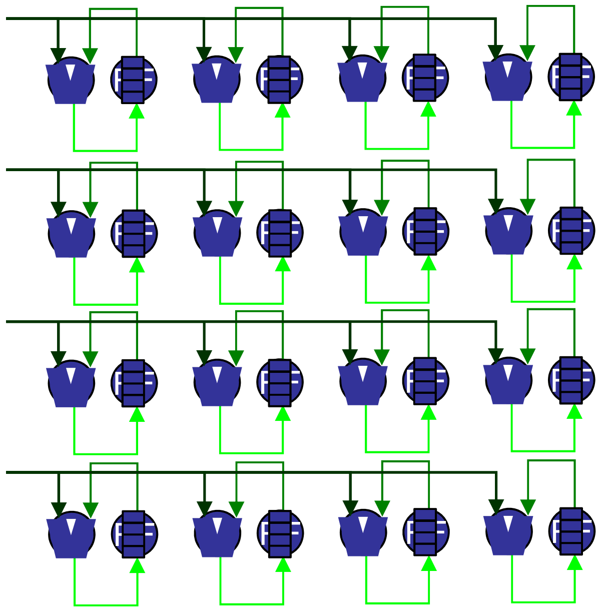
Flexibility

- **napCore**
  - Flexibility enablers
    - **Versatile instruction set** for complex vector arithmetic
    - **Wide dynamic range** due to floating-point arithmetic
  - Seamless implementation of different
    - **MIMO detection** algorithms: Open-loop & iterative …
    - **Baseband processing** tasks: Channel Est., SINR calc.
    - **Vector based applications**: Sensor applications
- **rASIP**
  - Fixed-point arithmetic with flexible scaling mechanism
  - Basic matrix operations to construct complex applications
  - Flexible implementation of control path on processor
  - Targets MIMO detection with different algorithm structures
    - e.g. MMSE-based, MCMC

Programmability



- **napCore**
  - Programmable via SIMD assembly or C
  - All tools (assembler, debugger, etc.) generated automatically by Synopsys Processor Designer
- **rASIP**
  - CGRA: Programming of matrix operation
    - Operation graph
      - Function patterns
      - Interconnect patterns, e.g. from/to local register file (RF)
    - Tools to map operation graph onto architecture generated from architecture description
      - Map: Instruction selection
      - Route: Interconnects inside/between PE's
  - Processor: Programming via assembly or C



Agenda

- The two compared architectural approaches
- Efficiency, flexibility, and programmability
- **Summary**

Summary

- **napCore**
  - Flexible and easy-to-use – focus on simplicity
  - All tools generated automatically
  - Programmable in SIMD assembly/C
  - Suitable for any complex vector arithmetic algorithm
  - Very competitive w.r.t. area efficiency
  - Minor drawbacks w.r.t. energy efficiency

- **rASIP with CGRA Accelerator**
  - Efficient, but CGRA is hard to program
    - Generated graph-based mapping & routing tools
  - Flexible control of reconfiguring CGRA by processor