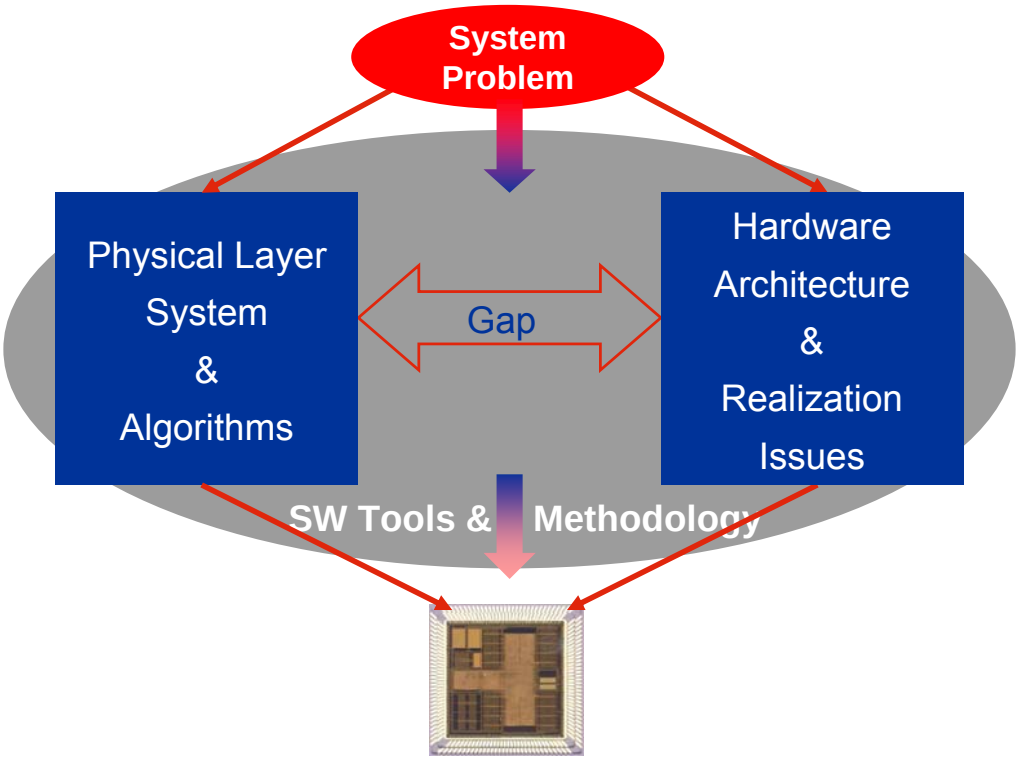
Challenges & Solutions for Building Future MPSoCs for Cellular Communications

Gerhard Fettweis

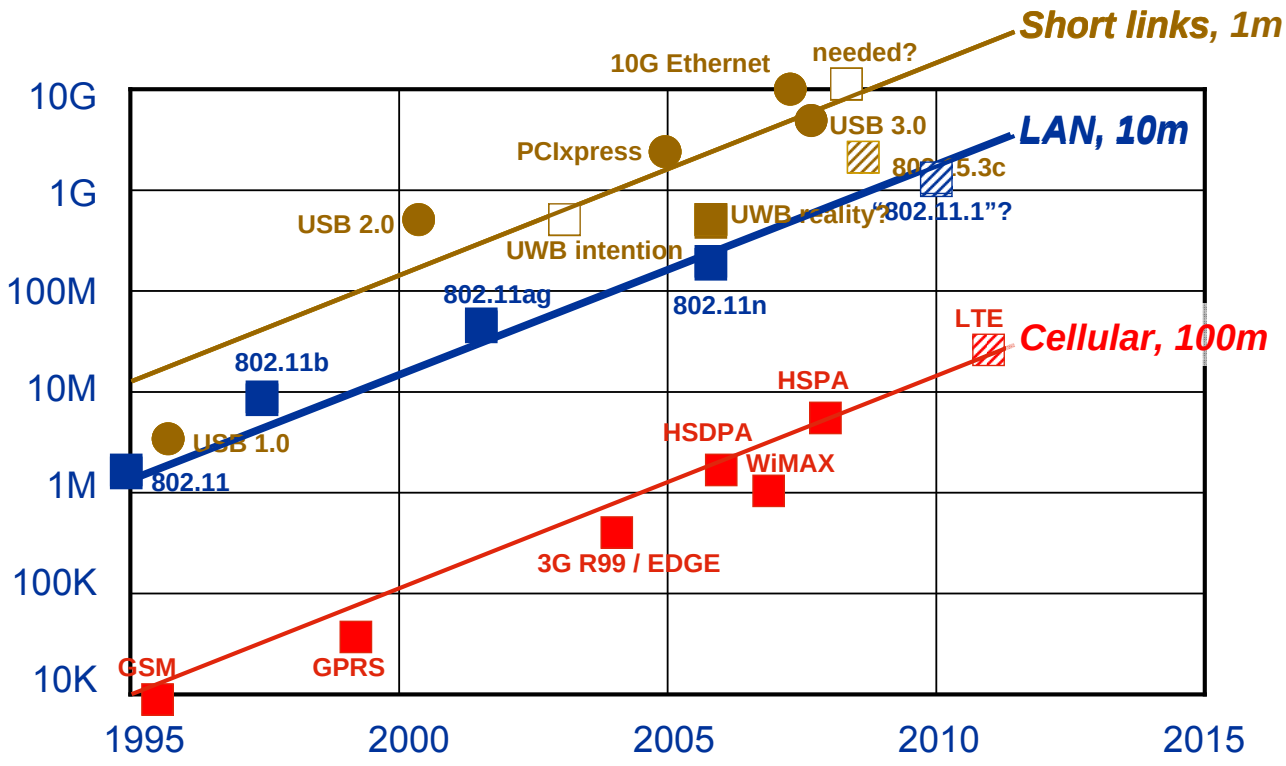
gerhard.fettweis@vodafone-chair.com

High-End Research System & Silicon

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair

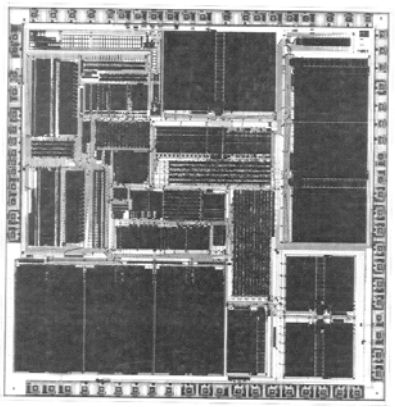


Data Rates Over Time



PDC ½-Rate Speech Coder DSP

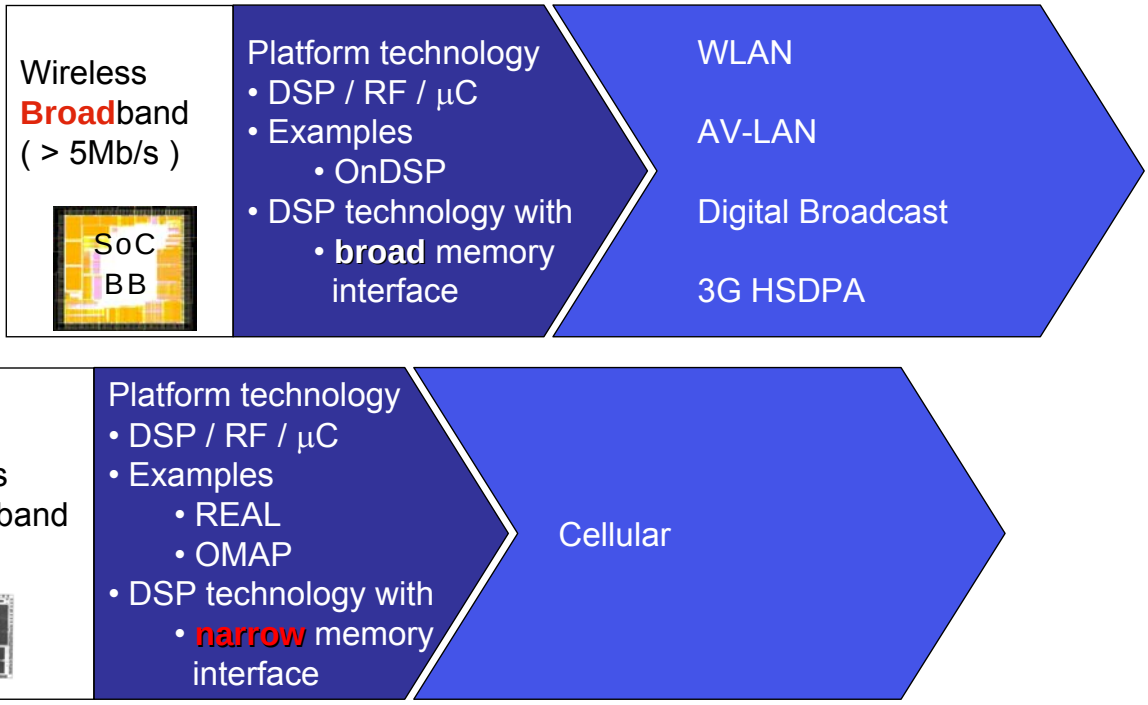
- Constraints of 1992 (0.8 µ CMOS):
  - 100 "MOPS"
  - 100 pins
  - 100 mm²
  - 100 mW
- Results
  - 100 "MOPS" in 20MHz
  - 80 pins
  - 70 mm²
  - 80 mW
- BUT
  - Proprietary tools & ISA: no open code available
  - 3 years development time



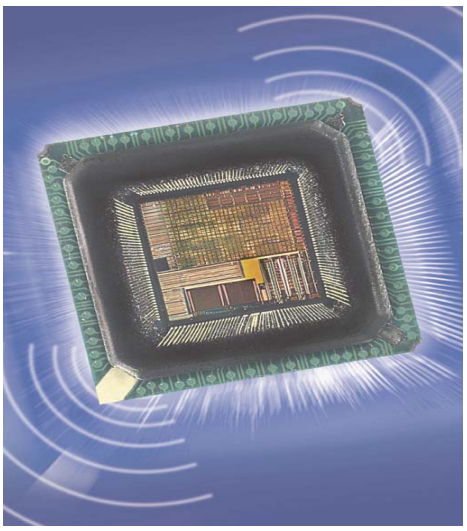
Teknekron Communications Systems

AsahiKASEI

Wireless Technology Drivers



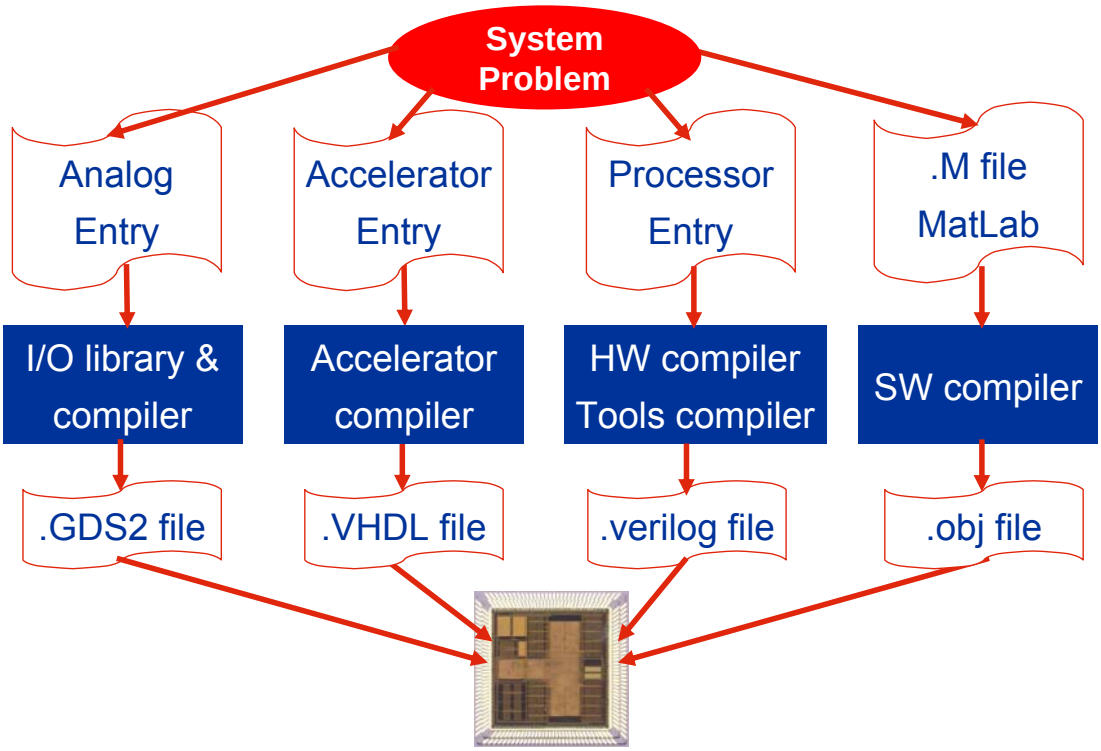
60 Mb/s Wireless LAN Chip (IEEE 802.11a/b)



- **True System on Chip**
  - **DSP Core**
  - **RAM**
  - **Encryption**
  - **Viterbi codec**
  - **Mixed Signal Block**
- **Process 0.18µ CMOS**
- **Supply Voltage**
  - **1.8 V core**
  - **3.3 V I/O**

www.systemonic.com

MPSoC: Hardware-Software Co-Design Path



MPSoC

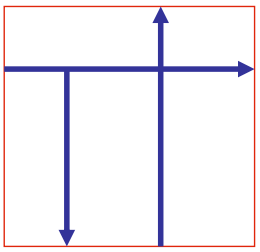
- Multi-core scalability ➔ Programming Wall
  - Software parallelization challenge
  - Hardware architecture challenge
- Memory architecture design challenge
- Design spec validation challenge ➔ Simulation Wall
  - Functional
  - Lantency
- Power consumption challenge ➔ Power Wall
- Design complexity challenge ➔ Design Wall

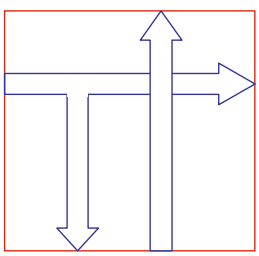
➔ One possible way out?

Algorithm Implementation ➔ Data Locality

Example: A Unit of Interconnect

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair





- Delay T
- Area A
- Complexity AT

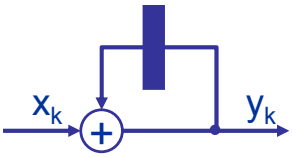
- Wordlength W
- Delay ~W
- Area ~W^2
- AT ~W^3

➔ Consider wordlength!

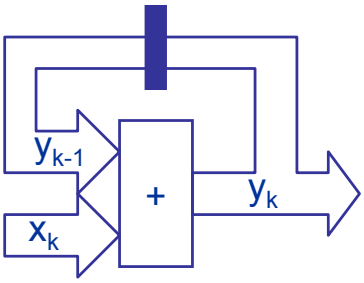
Example: Integrator/Accumulator

$$y_k = x_k + y_{k-1}$$

Algorithm expression level



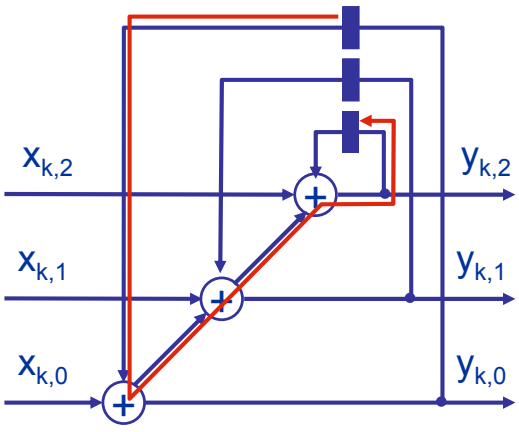
Algorithm signal level



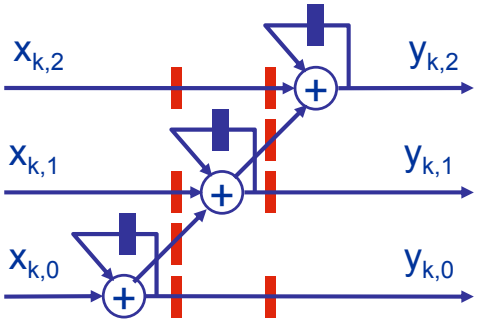
Algorithm word level

Example: Integrator/Accumulator

$$y_k = x_k + y_{k-1}$$



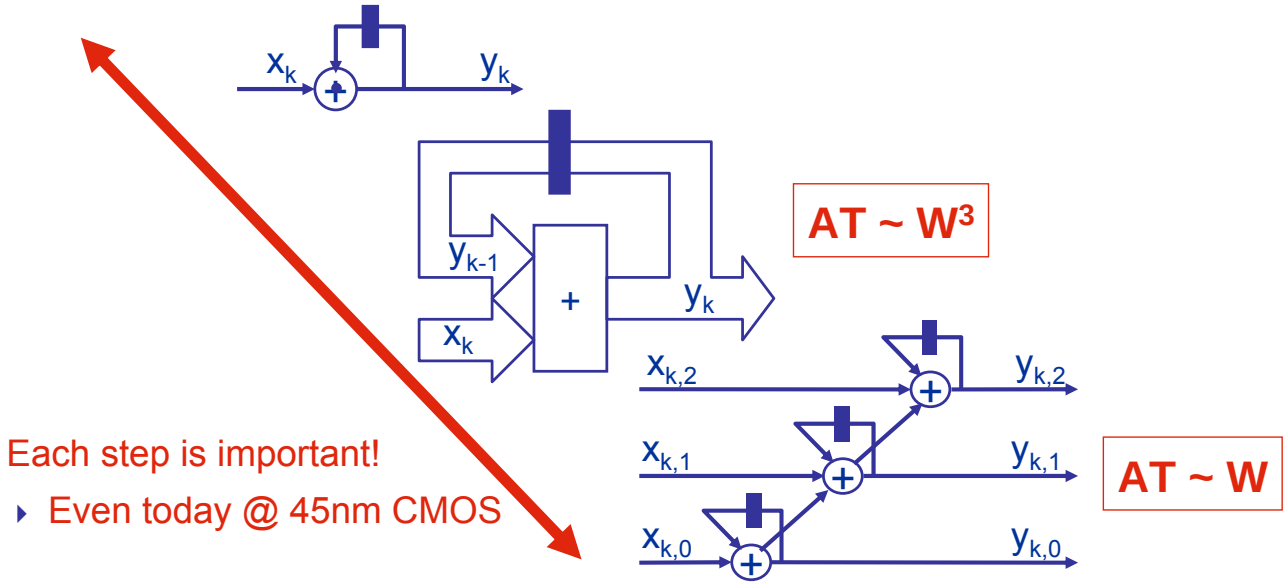
Algorithm bit level



Algorithm bit level

Example: Integrator/Accumulator

$$y_k = x_k + y_{k-1}$$

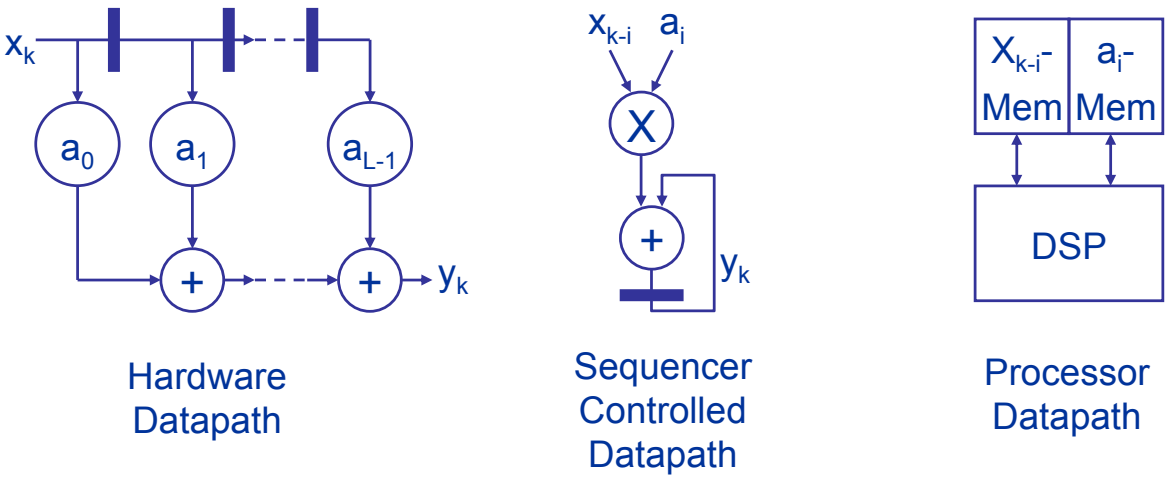


Each step is important!

- Even today @ 45nm CMOS

Mapping DSP Algorithms onto Hardware

$$y_k = \sum_{i=0}^{L-1} a_i x_{k-i}$$

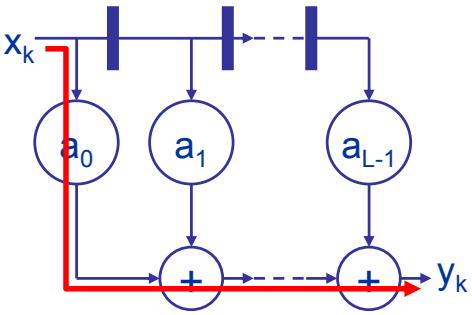


Hardware Datapath

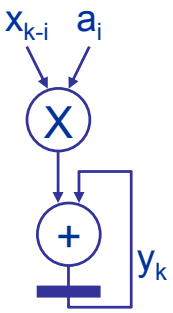
Sequencer Controlled Datapath

Processor Datapath

$$y_k = \sum_{i=0}^{L-1} a_i x_{k-i}$$



FIR Direct Form
algorithm graph



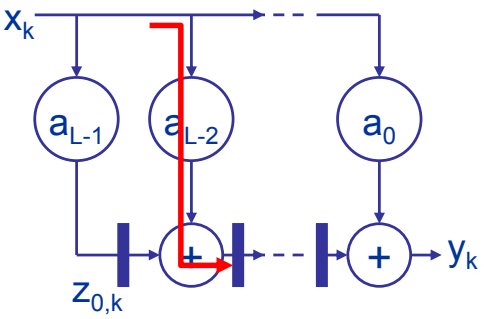
Sequencer Controlled Datapath

Memory I/O:

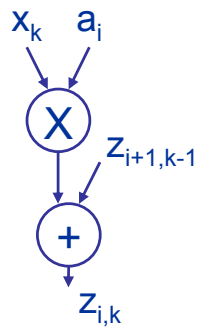
- N reads for x_{k-i}
- N reads for a_i
- Total: 2N reads
- Total: 0 writes

Sum: 2N R/W

$$y_k = \sum_{i=0}^{L-1} a_i x_{k-i}$$



FIR Transposed Form
algorithm graph



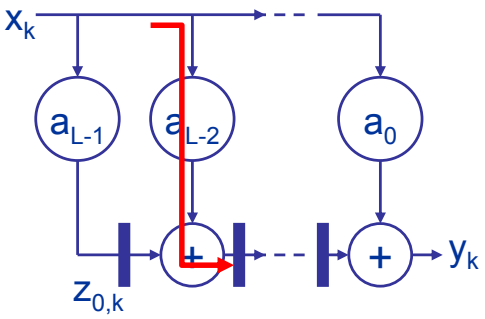
Sequencer Controlled Datapath

Memory I/O:

- N reads for a_i
- N reads for $z_{i+1,k-1}$
- N writes for zi,k
- Total: 2N reads
- Total: N writes

Sum: 3N R/W

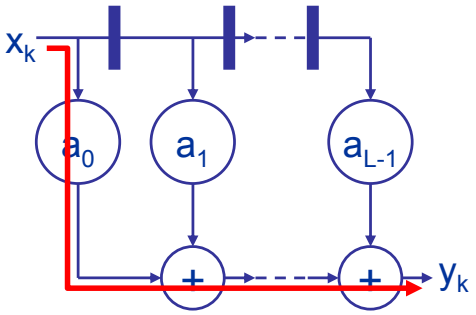
HW / SW Mapping: Conclusions



FIR Transposed Form

HW implementation

- Distribute algorithmic delays between logic
- Minimize critical path
- Minimize power consumption
- ***Maximize data locality***



FIR Direct Form

SW implementation

- Cluster registers to shift registers
- Maximize clustering of identical operations
- ***Maximize data locality***

Data Locality

Input x_k

$$\sum_i a_i x_{k-i} = y_k$$

Output y_k

control interference

Mapping into hardware
is fundamentally **different**
from mapping into software

In both cases:

Maximize data locality

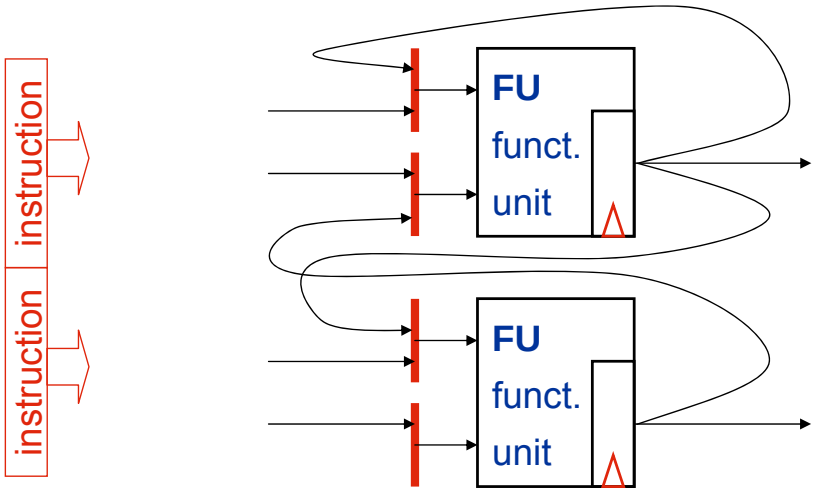
Minimize control interference

Parallelism

Instruction:
Synchronous Transfer Architecture

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair

Principle: Keep register files away – minimize data transfers to basic needs



FU Examples
- MAC
- ALU
- Shuffle
- Register File
- Address Gen.

Building on top of Jan Rabaey's silage idea of 1990

Data Locality

Vector:
Wireless Communications

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair

To complete task within given time constraint:
- Parallel processing of one task
  but how to achieve low-power (100-s mW range)?

Processing example: FIR filter

$$y_k = \sum_{i=0,N-1} a_i x_{k-i}$$

Task parallelism within equation

$$y_k = \sum_{i,even} a_i x_{k-i} + \sum_{i,odd} a_i x_{k-i}$$

4 memory reads per cycle

Parallel vector processing of task

$$y_k = \sum_i a_i x_{k-i}$$
$$y_{k+1} = \sum_i a_i x_{k-i+1}$$

z^{-1}

2 memory reads per cycle

Vector processing reduces memory I/O-bandwidth: Low-Power

3 equivalent equations/algorithms

$$y_k = \sum_{i=0,N-1} a_i x_{k-i}$$

$$y_k = \sum_{i=0,N-1} \left| a_i - x_{k-i} \right|$$

$$y_k = Max_{i=0,N-1} \left(a_i + x_{k-i} \right)$$

$$y_k = \bigoplus_{i=0,N-1} \left(a_i \otimes x_{k-i} \right)$$

Operations can be divided into

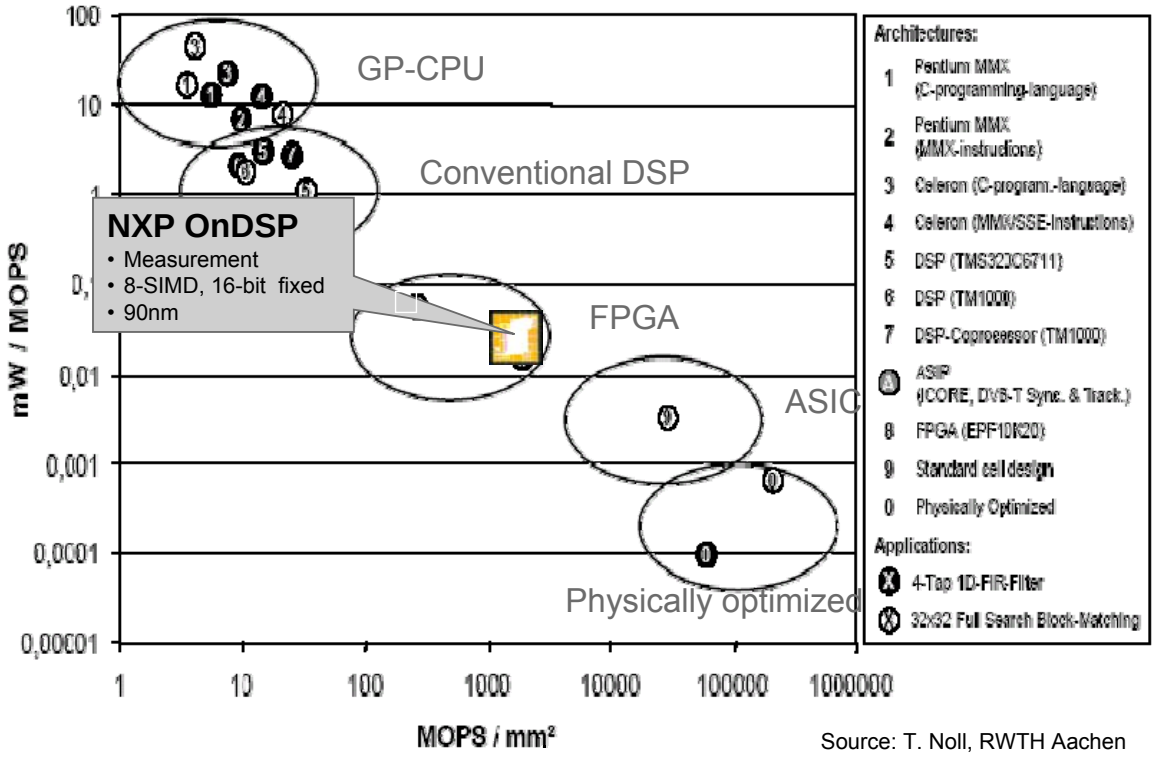
Data transfer (determining memory access)

Data manipulation (operation)

Data Locality

Demonstrating Low Power & Die-Size

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair



Source: T. Noll, RWTH Aachen

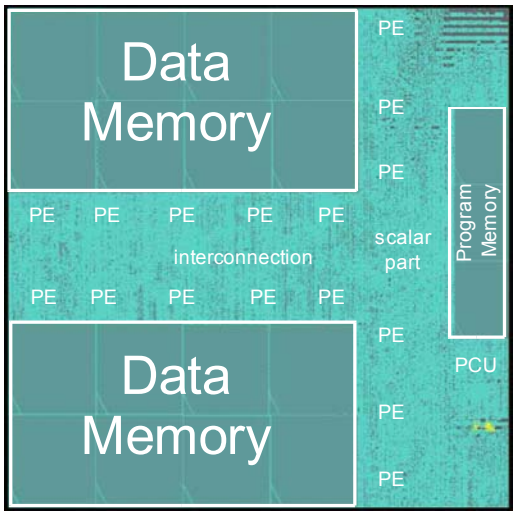
Case Studies

Application Specific DSPs

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair

- **SAMIRA:** 8-Slice FP-DSP Matlab Co-processor (tape out and silicon run)
- **M5**: 16 slice DSP for DVB-T
- **Minimal**: First STA-DSP FPGA Implementation
- **M4**: 2-Slice DSP with DVB-T tailored instructions
- **PUMA**: UMTS / WLAN Rake receiver combo
- **OFDM DSP**: 4-Slice OFDM transceiver demonstrator

M5: DVB-T Baseband Processor

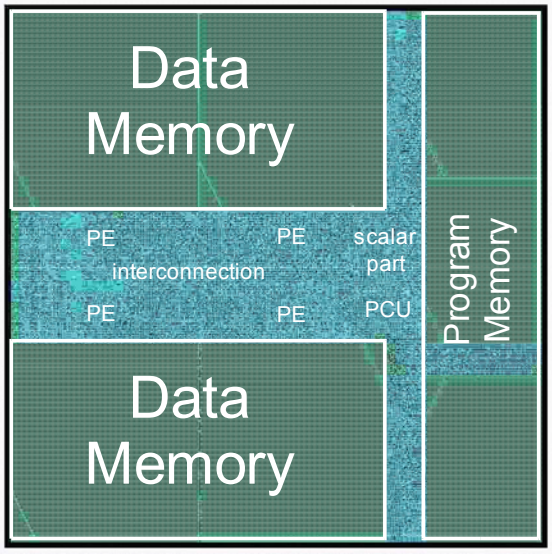


Component	*Area (mm²)*
1Mbit Dual-Port Data Memory	7.1
180kbit Single-Port Program Memory	0.7
Vector Part	1.6
Scalar Part	0.3
Sum	9.7

Clock rate	250MHz

In: “Implementing a Receiver for Terrestrial Digital Video Broadcasting in Software on an Application-Specific DSP”, in Proc. of SIPS 2004

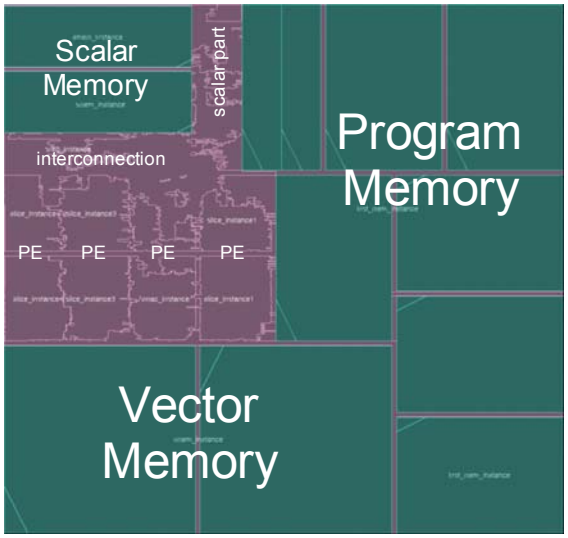
M4 – Case Study of a 4-Slice DSP



Component	*Area (mm²)*
256kbit Dual-Port Data Memory	1.8
192kbit Single-Port Program Memory	0.7
Vector Part	0.4
Scalar Part	0.3
Sum	3.2

In: “A HW/SW Design Methodology for Embedded SIMD Vector Signal Processors”, Int. Journal of Embedded Systems

OFDM-DSP



Component	*Area (um²)*
160 kbit Two-Port Data Memory	1,122,726
416 kbit Single-Port Program Memory	1,535,120
Vector Part	286,127
Scalar Part	31,786
Interconnectivity	63,660
Decoder	10,400
Sum	3,049,819

Power Consumption

- DVB-T Receiver Chipset

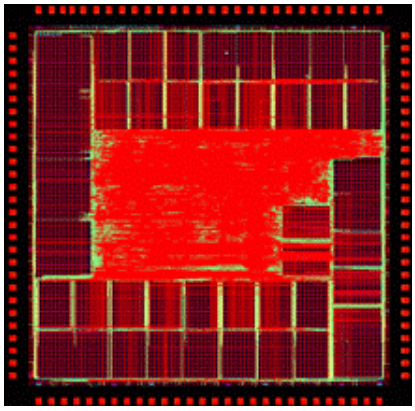
LSI-Logic COFDM DVB-T Receiver (L64782)	800mW

- Power Consumption of 2004 DSP/µP

Name	parallel data path	Power consumption
ARM 1022E	1	0,60mW/Mhz
Adelante Saturn DSP Core (VLIW)	2	0,25mW/Mhz
Renesas SH7750R (SH-4)	1	2,85mW/Mhz
TI TMS320C6713	2	6,85mW/Mhz
ADI ADSP-21161N	2	8,95mW/Mhz
LSI403LP	1	0,64mW/Mhz
Carmel	2	1,66mW/Mhz
16-Slice “M5 DSP”	16	0,1mW/Mhz

An Answer to Low-Cost/Power Flexibility: MathCope-DSP “Samira”

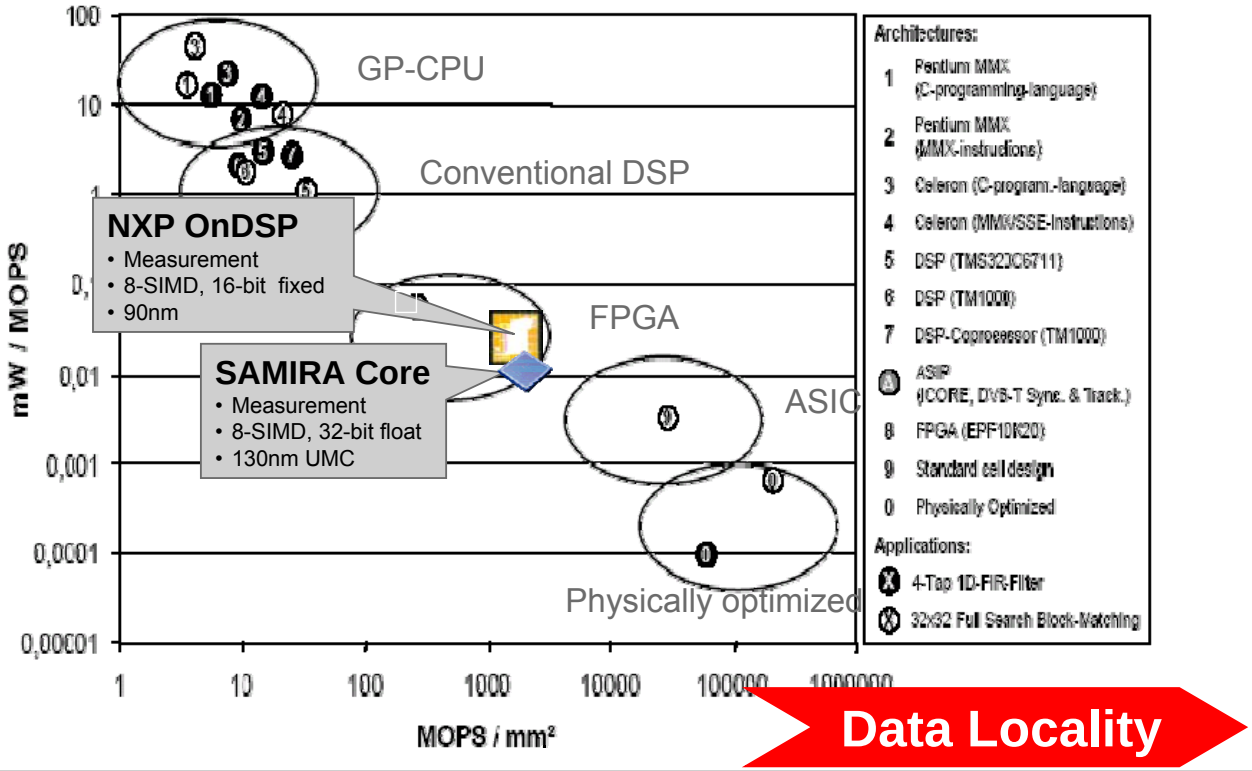
- SAMIRA – A floating point engine with 8 SIMD-parallel paths
- Tape-out on Feb 1, 2005



- 17 single-precision floating point units
- > 480k NAND gates
- 2.4mm² logic area on UMC 0.13µm
- 2.9 Mbit on-chip SRAM
- Multiple memory banks for synchronous access
- Compressed VLIW decoder
- Memory built-in self test (MBIST)
- Memory bandwidth:
  3.4 GBit/s extern – intern, 67 Gbit/s intern - intern
- 9 parallel scan paths
- JTAG interface
- 128 CQFP package
- 212 MHz measured
  by static timing analysis after place & route
- Performance: 3.6 GFLOP/s, 1.7 GMAC/s, 3.7 GOP/s

Demonstrating Low Power & Die-Size

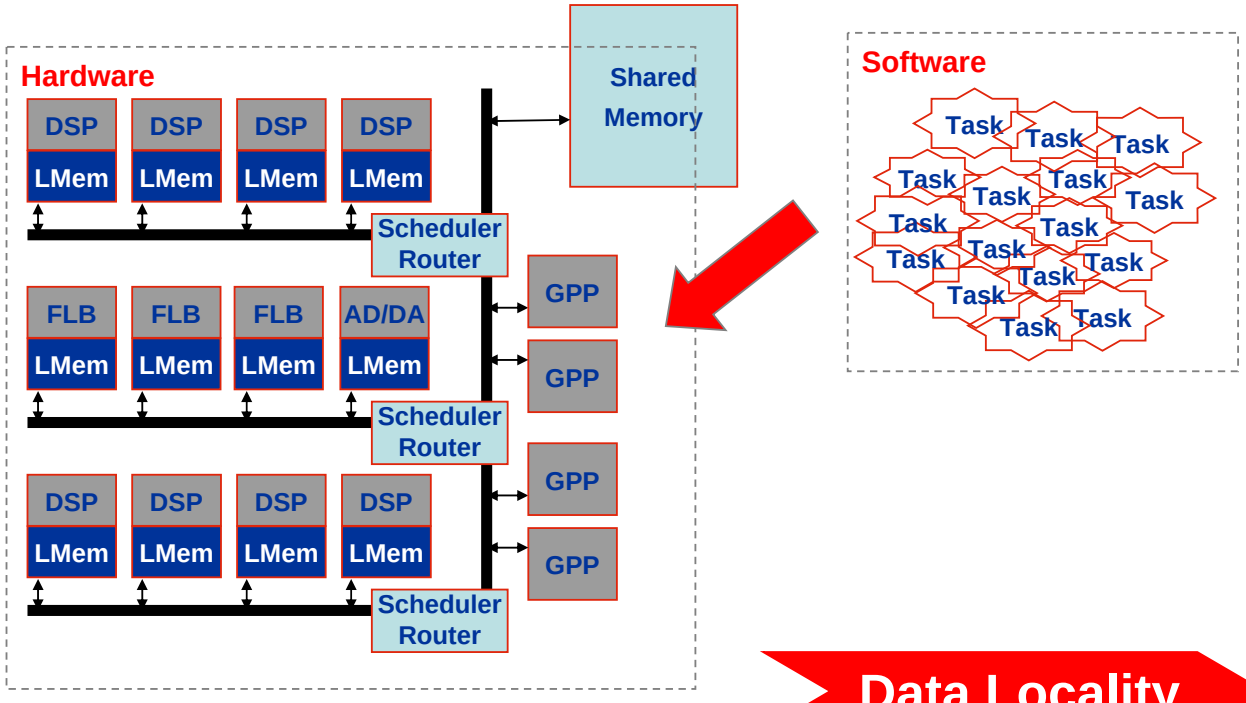
TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair



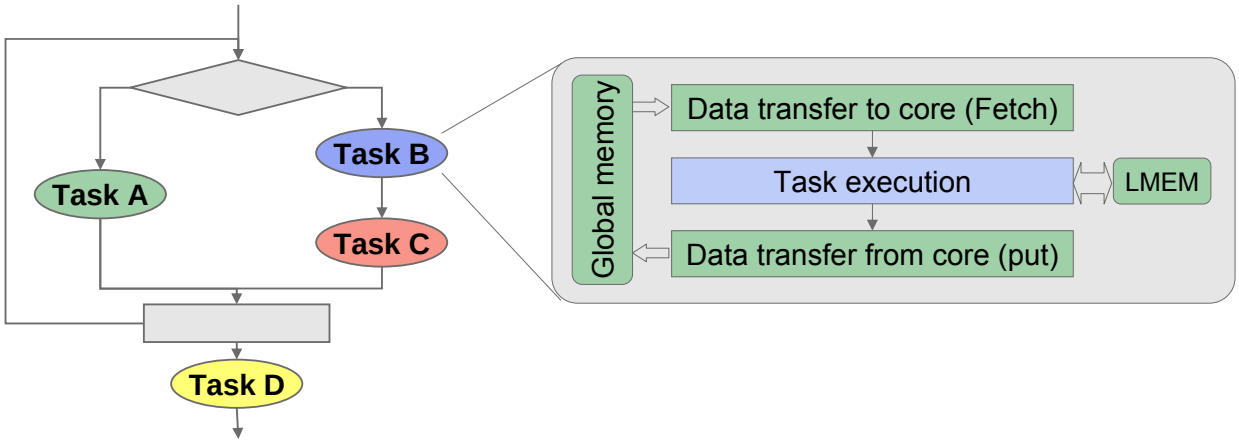
STLP: Super-Scalar Task Level Processing

Heterogeneous MP-SoC: Task Scheduling

TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair



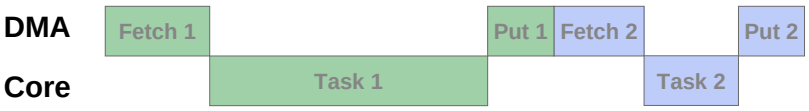
Task:
Composed of Atomic Tasks



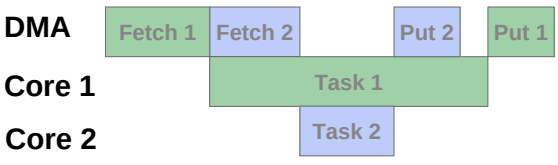
- Synchronous Data Flow Program
  - Contains control flow and “atomic” task calls
- Task
  - Consumes and produces chunks of data
  - Contains a terminating program that operates on input data
  - Data locality exploited by operating on local copy of data
  - Resulting data committed to global memory after execution of task

STLP: Super-scalar Processing

Conventional processing



STLP
- **Dynamic scheduling**
- **Super-scalar, out-of-order processing**

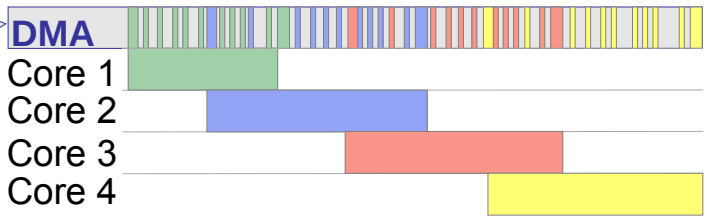


STLP Enables Efficient Memory Access

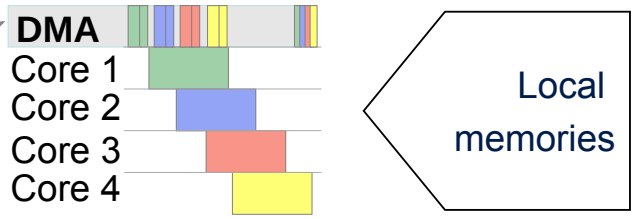
TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

DMA bottleneck: "random" memory access during task execution
(example: >12 Gbit/s SDRAM bandwidth needed for 1080i H.264 decoding)



Memory wall problem solved through scheduled memory access.



Data Locality

STLP Syntax

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

Sequentially coded application, contains task calls

```
void getQuarterpelBlock(int x, int y, int width, int height, Picture *pic, UInt8 *dest1, UInt8 *dest2)
{ // …
if (x<10) {
        task(task1, IN(pic,x,y,width,heigth), OUT(dest1,x,y,width,height));
        task(task2, IN(pic,x,y,width,height), OUT(dest2,x,y,width,height));
} else
        task(…);
}
```

- **Programmer writes a sequential program**
- **Automatic parallelization of tasks at run-time (dependency checking),**
- **Programmer has illusion of sequential execution**

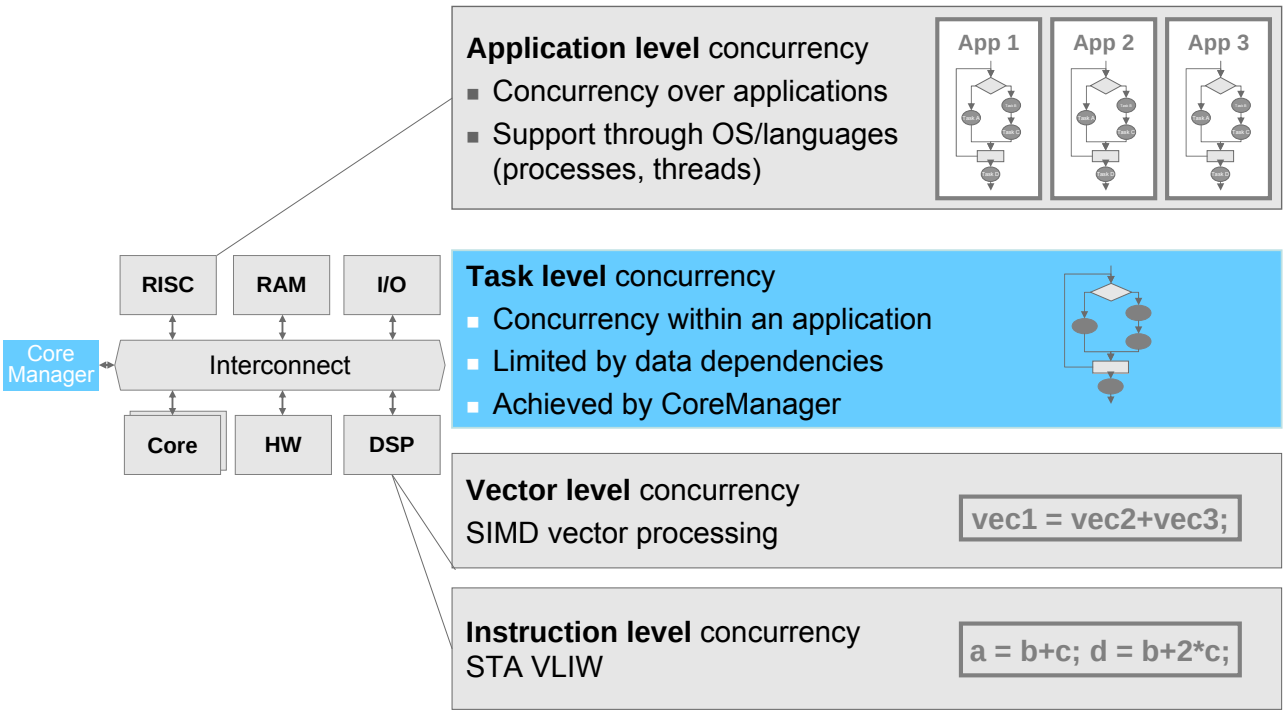
Tasks defined as C functions

```
#pragma TASK_BEGIN
void task2(char *picture_source, char *picture_target)
{ …
for (short i=0;i<4;i++)
   { …// some computations
    }
}
#pragma TASK_END
```

Levels of Concurrency

TECHNISCHE UNIVERSITÄT DRESDEN

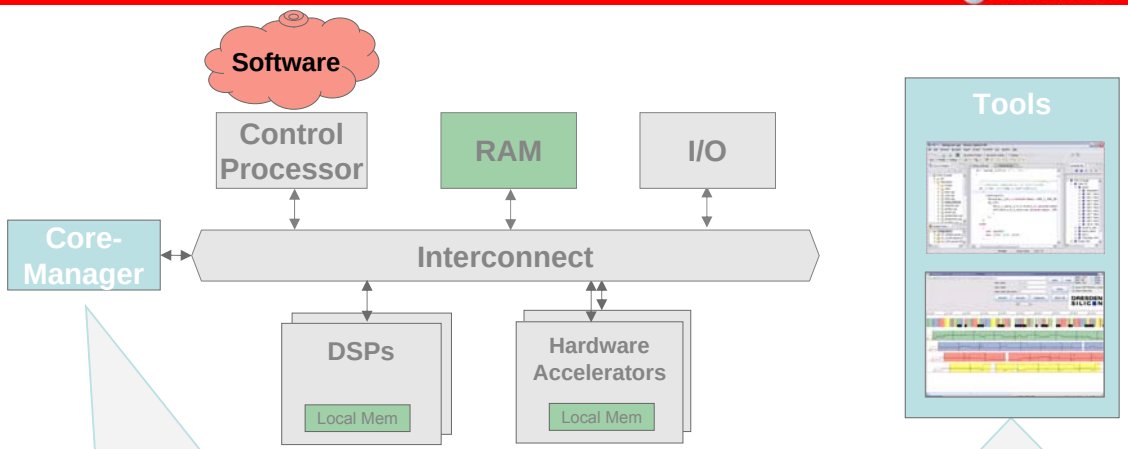
vodafone chair



A Platform Approach

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair



CoreManager
A run-time system enabling STLP and providing an abstraction of the hardware

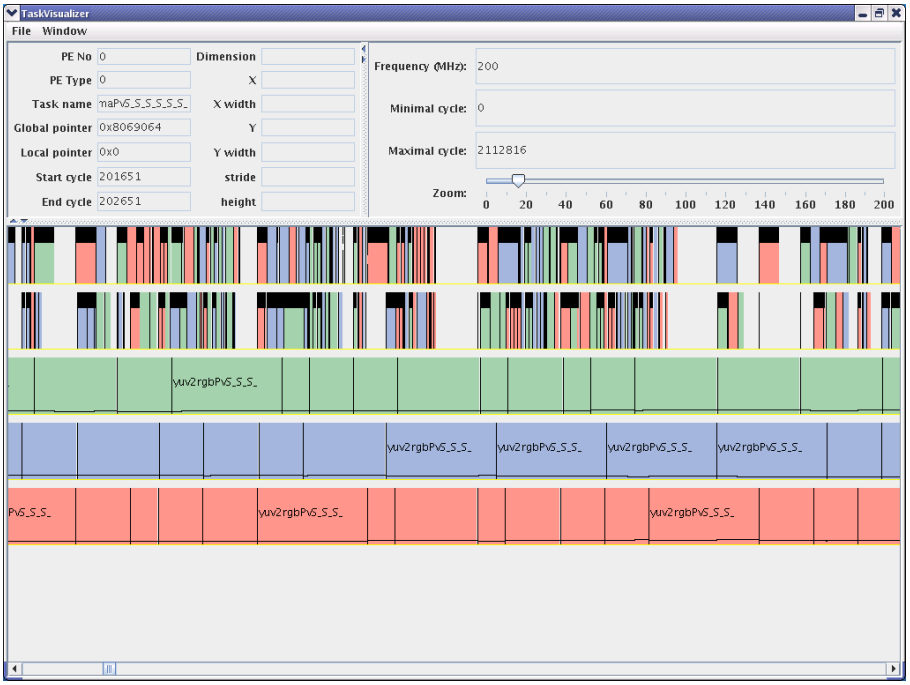
Tools
Complementary tooling for STLP software development
- Compile framework
- TaskVisualizer
- Eclipse IDE plugin

Tool Example: TaskVisualizer

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

- **H.264 decoding using 2 DMA channels and 3 processors**



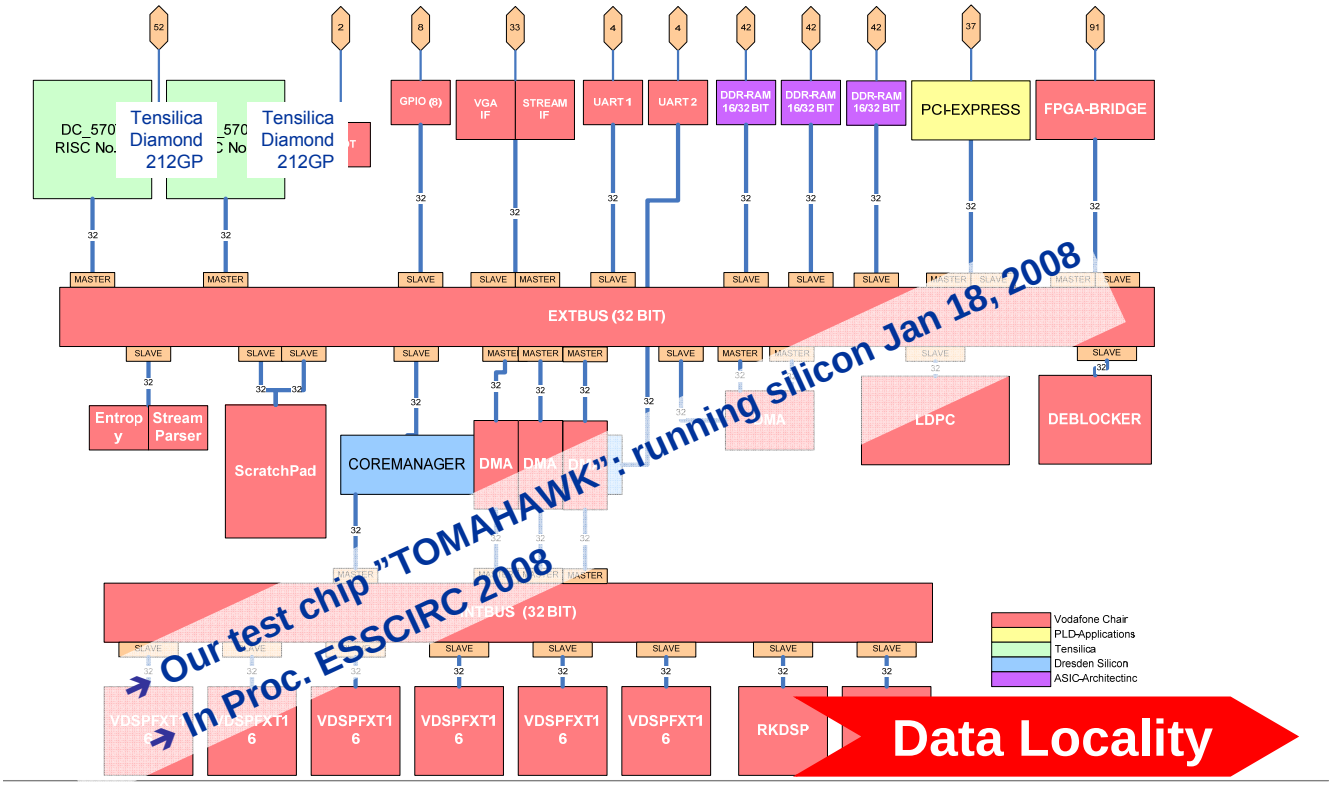
Running Demo

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

- CoreManager Reference FPGA Implementation
  - Full H.264 baseline video decoder based on CoreManager
  - Proves programming model for a complex application
  - Proves applicability for heterogeneous architectures

LTE/WiMAX Single-Chip SDR: Tomahawk – Block-Diagram

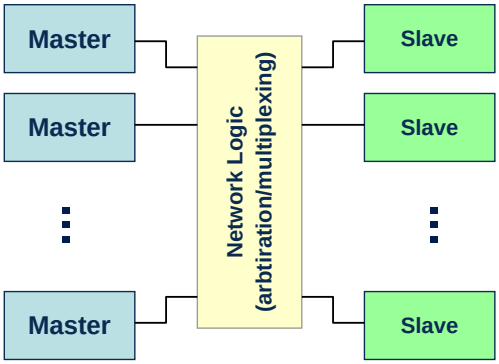


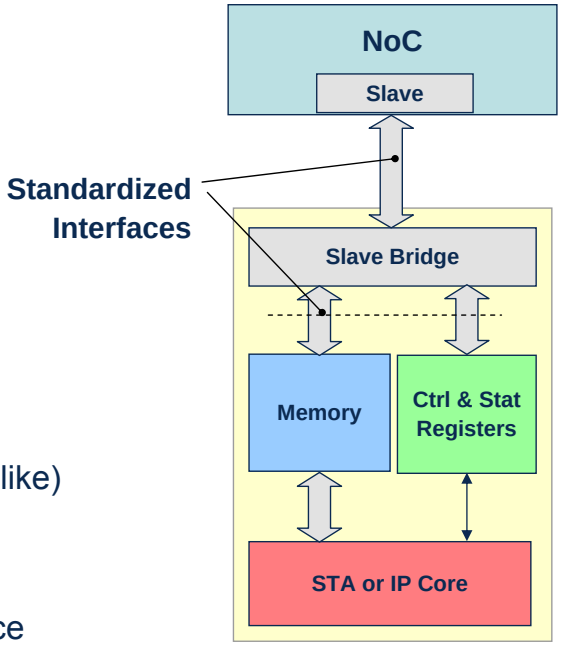
Design Flow

- Fully automated
  - Generation of STA hardware modules
    - Accelerators, e.g. LDPC
    - DSPs, e.g. vector DSP
  - Generation of ISS
  - Generation of test benches
  - NOC
- Regression test suite
- Backend by
  - Prof. René Schüffny & team

➔ Made it possible to design by university team

NoC – Topology and Integration





- Circuit-switched star topology (crossbar like)
- Working on positive/negative edge
  → reduced latency
- Easy integration due to standard interface components

Data Locality

Tomahawk - Characteristics

- 130 nm UMC technology
- 8 metal layers
- 175 MHz system clock
- 57 M Transistors
- 6.8 Mbit on-chip memory
- 7 clock domains
- 321 user IOs
- 154 power pins
- 476 CPGA package
- Area: 100 mm^2
- Total Power (estimated): 1500 mW
- Packaged at MPD Dresden
- ~5 MY of hardware development & verification

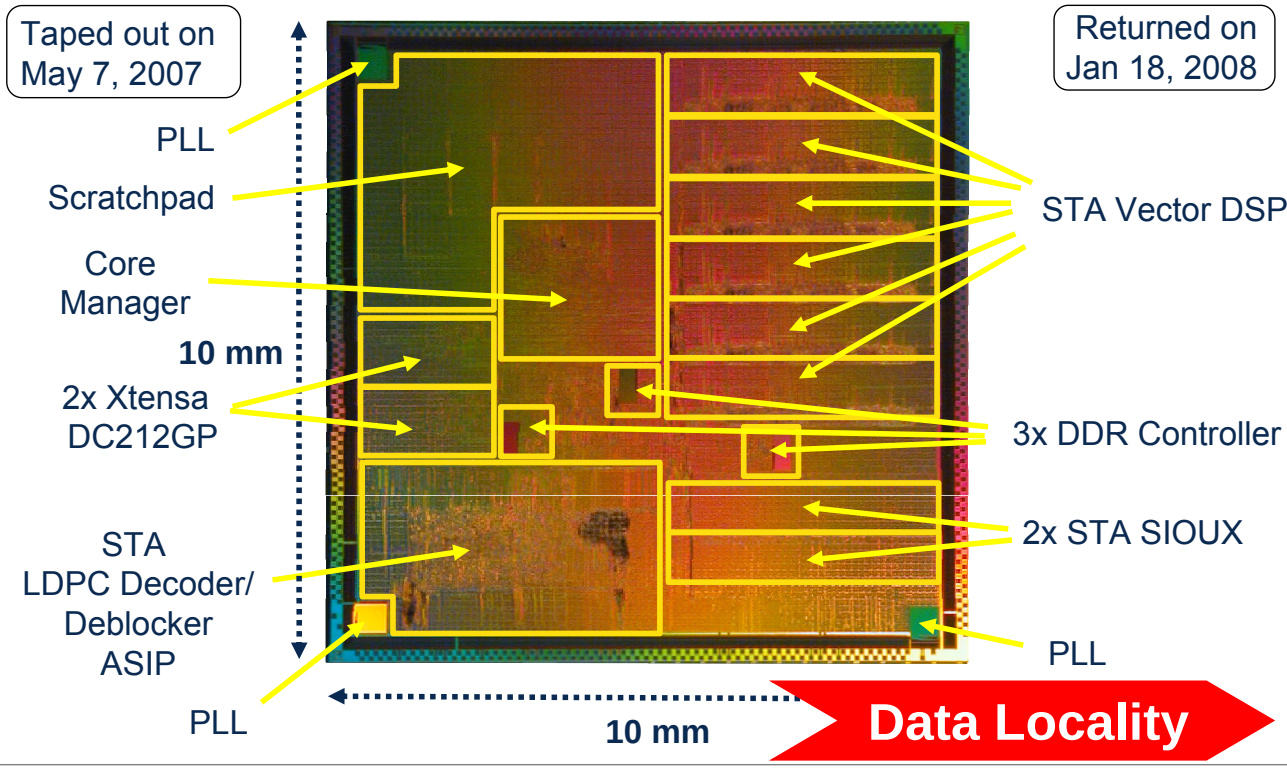
CoreManager:
- Area: 6 mm^2
- Power: ~282 mW

STA Fixed Point Vector DSP:
- Area: 3.8 mm^2
- Power: ~85 mW

STA Floating Point Scalar DSP:
- Area: 3.3 mm^2
- Power: ~26 mW

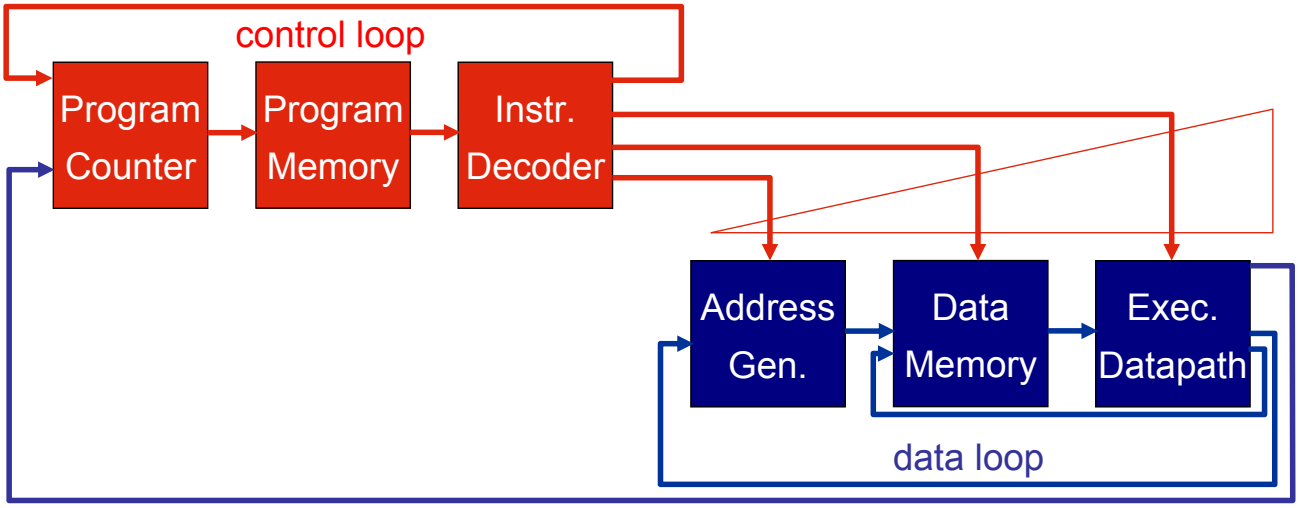
STA LDPC Decoder ASIP:
- Area: 7.8 mm^2
- Power: ~367 mW

Tomahawk - Die Photo



Outlook

Basic DSP Block Diagram



Control Loop ↔ Data Loop
Control/Program Memory ↔ Data memory
Program Scheduling ↔ Data Scheduling

DSP Algorithms: Common Features

- Computational complexity

+

- Dynamic adaptation to operational conditions
- Non-deterministic execution times
- Algorithm concurrency
- Wide range of applications (wireless baseband processing, HDTV/H.264, data streaming)

=

- Multitasking & Run-Time Management

Motivation

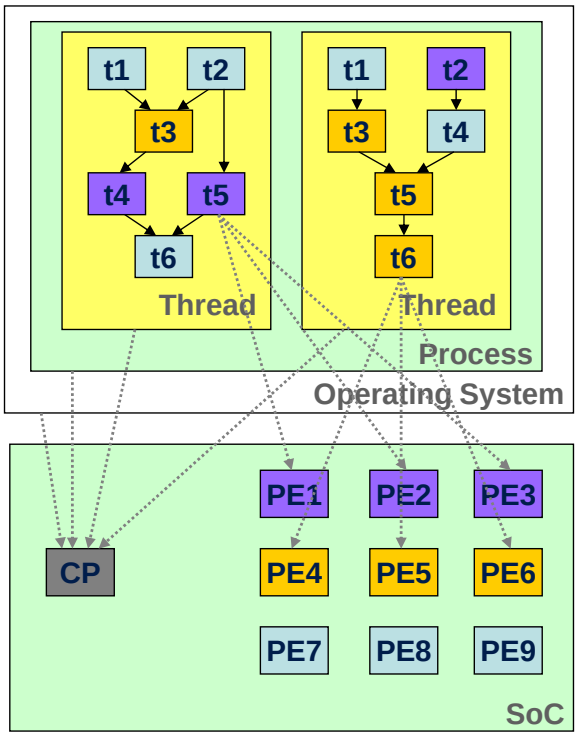
- Context Switches reduce Cache Performance
  - Hit rate 78% after CS, 90% after 1000, 96% after 10000 instructions
- Hardware-Software Interfaces
  - Interrupt based synchronization of control processor and accelerators creates huge scheduling overhead
- Processor and Application Incompatibility
  - e.g. control code intensive applications on DSPs
- Preemptive Real Time Operating Systems
  - RMS guarantees < 70% utilization for single processor

➔ Overall performance degradation by a factor of 3-5!

[Olli Silven and Karri Jyrkkä, Observations on Power-Efficiency Trends in Mobile Communication Devices EURASIP Journal on Embedded Systems Volume 2007]

Programming Model – Problem Statement

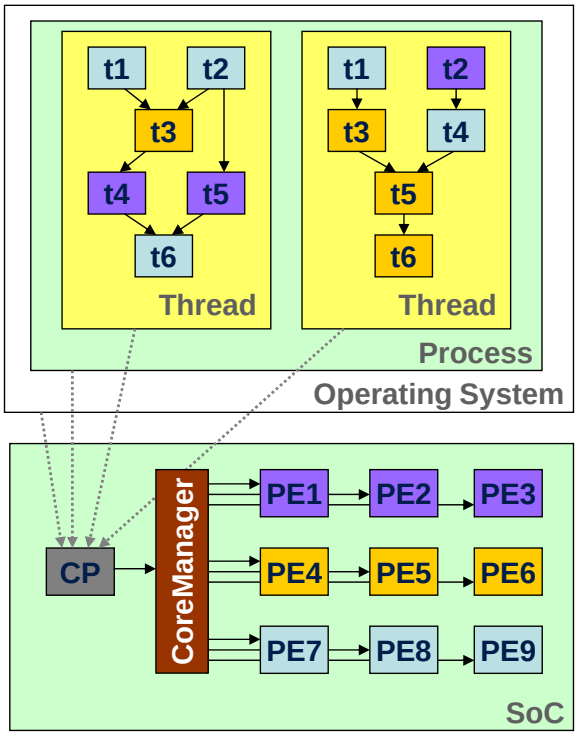
TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair



- Task is atomic execution unit
- RT-Thread: collection of data dependent tasks with execution time limit and control code
  - There are no data dependencies among RT-Threads
- Process: collection of RT-Threads and control code
- Tasks should be executed on Processing Elements (PE)
- Control code should be executed on Control Processor (CP)

Problem: How can efficient mapping of tasks and control code be realized?

Programming Model – Solution



Solution: "CoreManager" hardware unit which takes cares of:

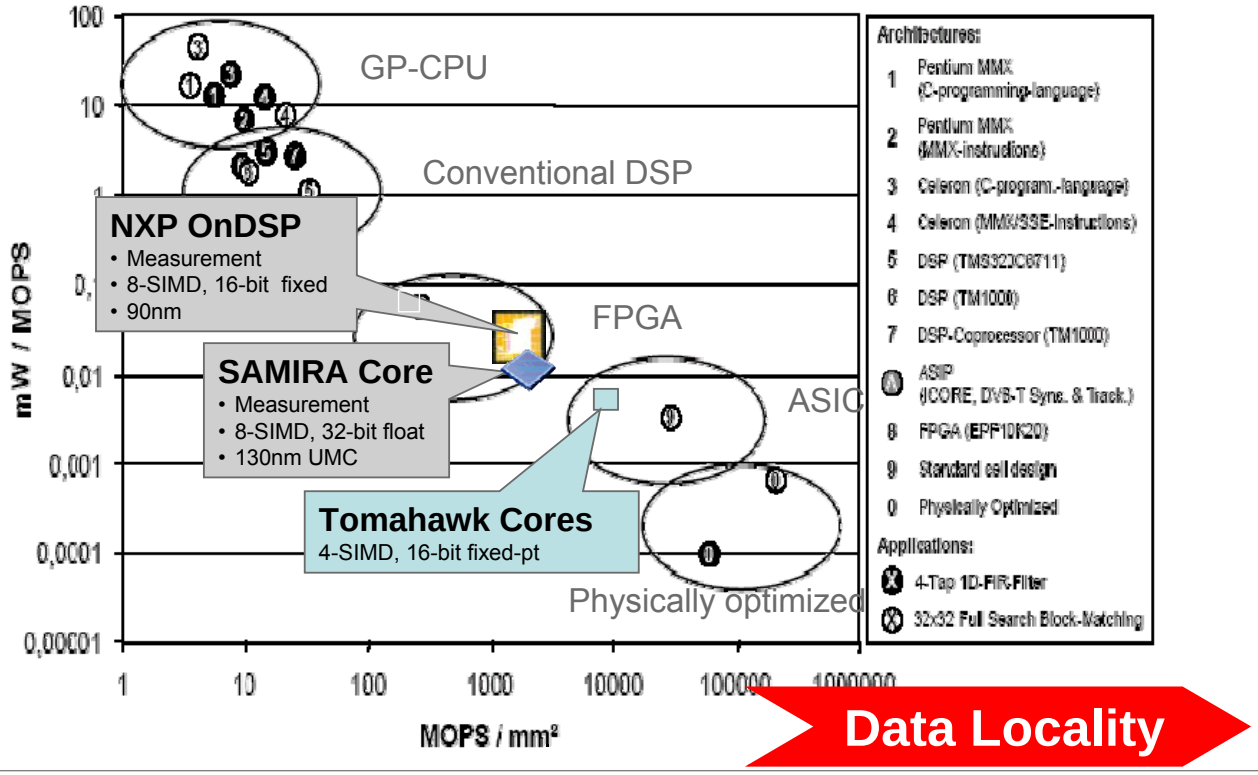
- Dependency checking
- Thread deadlines
- Priority scheduling
- Local memory management of PEs
- RT-OS scheduling support

- CP sends task descriptions to CoreManager
- Communication makes use of standard NoC interface

➔ No synchronization interrupts
➔ OS scheduling eased
➔ Predictable results due to explicit use of local memories

Conclusions

Demonstrating Low Power & Die-Size

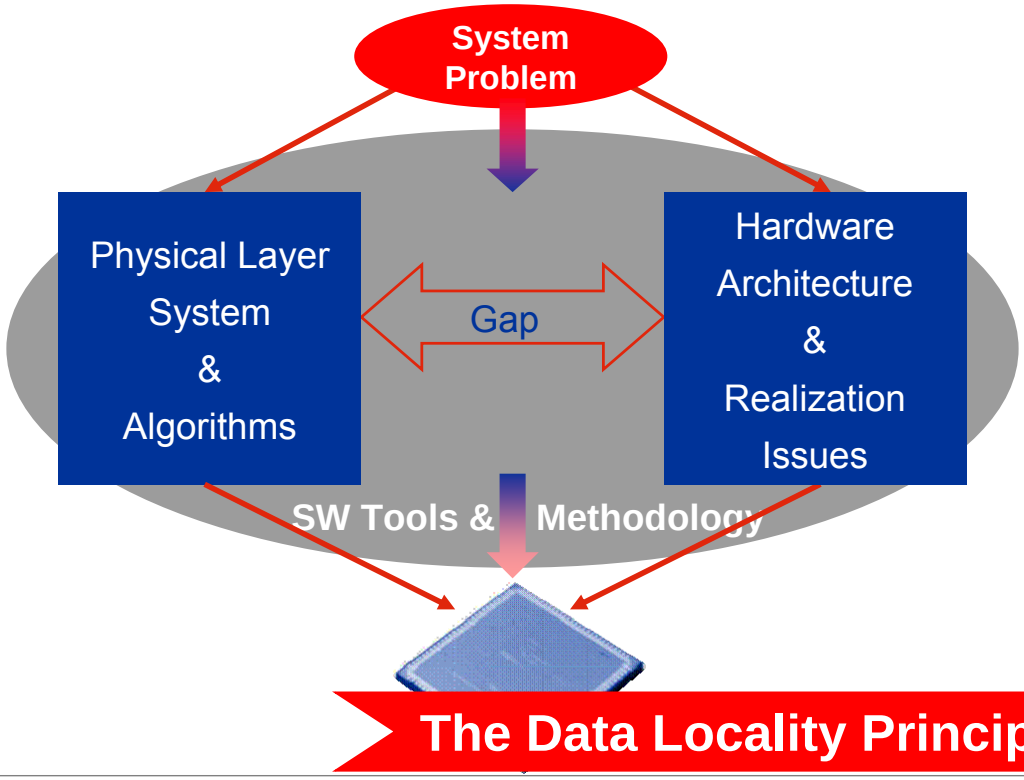


Embedded Systems: Conclusions

Hardware implementation systems challenges

are becoming

Software paradigm & implementation challenges as well *!!!*



The Data Locality Principle

Thanks *!*

www.vodafone-chair.com

Acknowledgement:
Emil Matus, Torsten Limberg, Markus Winter, Pablo Robelly, Hendrik Ahlendorf, Oliver Arnold, Jie Guo, Reimund Klemm, Björn Mennenga, Bastian Riestau, Mohammad Ali Shah, Marcos Tavares, Ulrich Walther, Frank Engel, Matthias Weiß, Shiro Kobayashi, Attila Römer, Thorsten Dräger, Wolfram Drescher, Gordon Cichon, Patrick Herhold, Michael Hosemann, Thomas Richter, Frank Schäfer, and many more