# Signal Processing for Wireless Communications and Multimedia: Design, Tools, Architectures

## Advanced Digital System Design Course 2006, EPF-L

Prof. Heinrich Meyr

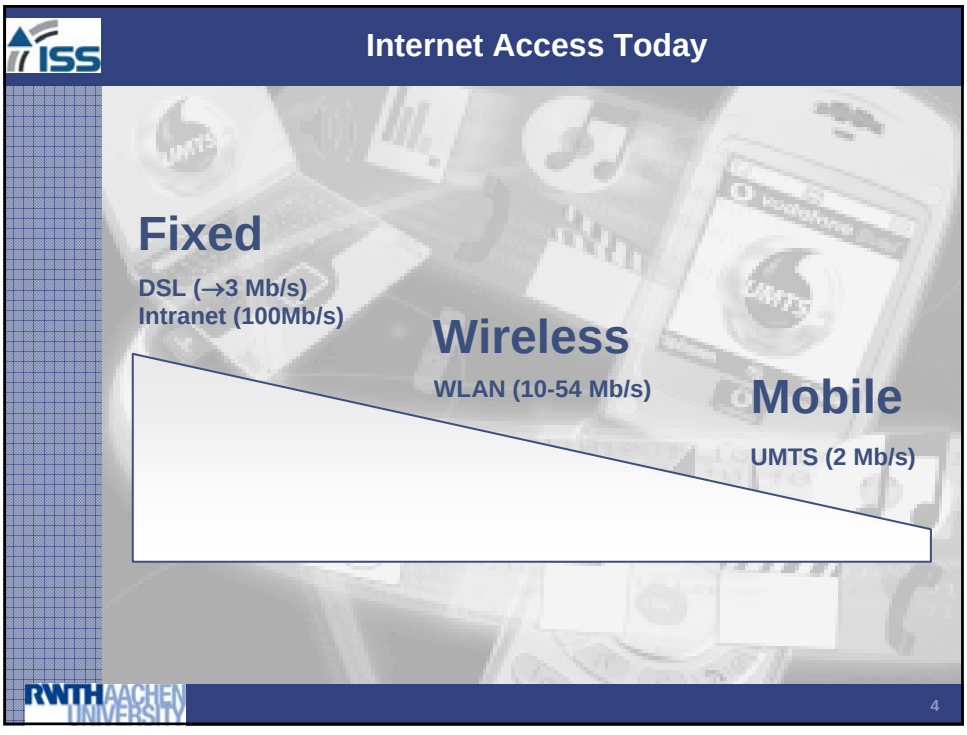
**RWTH Aachen University , Germany**
**and**
**Chief Scientific Officer, CoWare Inc**

## Agenda

- Future Wireless Communication System
- Future Wireless Communication Systems and ist Impact on ESL
- The End of Moore´s Law
- Receiver Structure, Models and Performance Metrics
- Massive Parallel Processing on heterogeneous MPSoC
- Application Specific Processors
- Summary and Conclusions

Future Wireless
Communication Systems

## Internet Access Today



## Mobile Internet Access

**The Vision**

**Ultra High-Speed Mobile Information and Communication *everywhere* at *low cost***

**UMTS Standard: 2 Mb/s**

**Reality today:**
**UMTS 0,1-0,3 Mb/s**
**GSM/GPRS 0,02 Mb/s**


- *I… located places*
- *F… rs*



## 4G and Beyond

- New concepts
  - Ultra high speed transmission
  - Mobile multimedia processing
  - Wearable and environmental information processing
  - Smart systems
  - Flexible, cognitive radio access
  - Multi-Processor Systems on Chip (MPSoC)
  - Digitized radio front end

## Mobile Applications and Services

- Future mobile wireless internet services:
  - **Information (web browsing, …)**
  - **Communication (VoIP, video, P2P, …)**
  - **Entertainment (distributed gaming, …)**
- Challenging mobile application classes
  - **Wearable and environmental information processing: work, sport, health care**
    - e.g. location aware services, seamless mobile working
  - **Mobile multimedia processing**
    - e.g. entertainment, information access, navigation,…



## Future Wireless Systems: In a nutshell

- **Will be**
  - cognitive
  - multifunctional
  - software definable
- **Will have**
  - multiple Antennas
- **They will make use of ultra-complex signal processing to optimally use the availabel bandwidth**
- **And process these algorithms on heterogeneous configurable computing engines**

Future Wireless Communication Systems and its Impact on ESL

## Impact of NGMN on Design Process: I

- To meet the schedule of NGMN it is imperative to have a concurrent and iterative development and validation process to design
  - Standard
  - Development and validation of algorithm and HW/SW of the digital receiver
  - Application (SW) development

**New approaches are needed !**

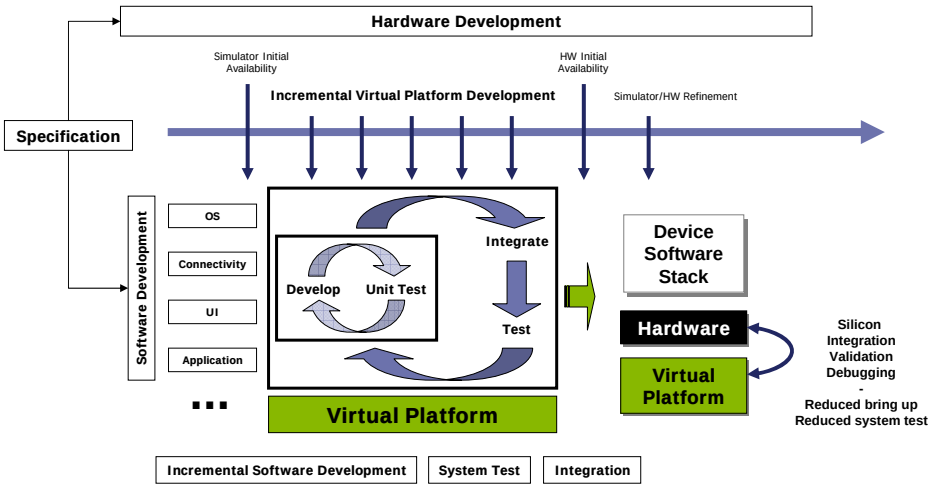
## Impact of NGMN on Design Process:II

- Development and integration issues need to be uncovered as early as possible
- Companies cannot wait for hardware to be available to start Software development
- Development costs need to be reduced and schedules accelerated

**New approaches are needed !**



## Virtual Platform Based Development



The End of Moore´s Law:

„Design Competence rules the World“

## Cross-disciplinary Task Management

### Analysis

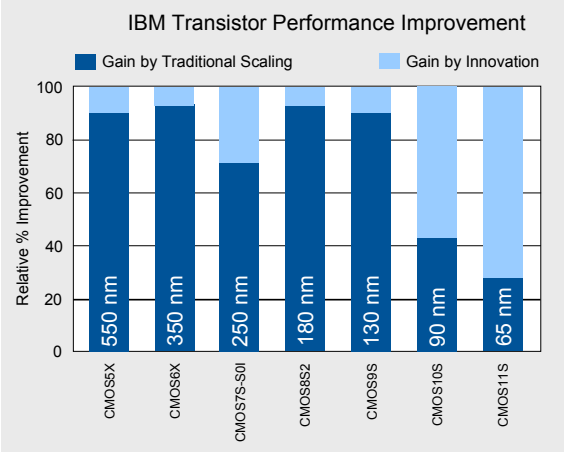
- The task comprises of many subtask in various disciplines
  - “The whole is more than the sum of the parts”

### Conclusion

- The solution requires the interaction of people in the various disciples

## The Paradigm Shift: Innovation Overtakes Scaling

- Innovation now dominates performance gains between generations
- **This means** that "***scheduled invention***" is now the majority component in all technology gains




Source:
Lisa Su /IBM:
MPSoC Conference 2005



## The Paradigm Shift: Integrated Design Approach

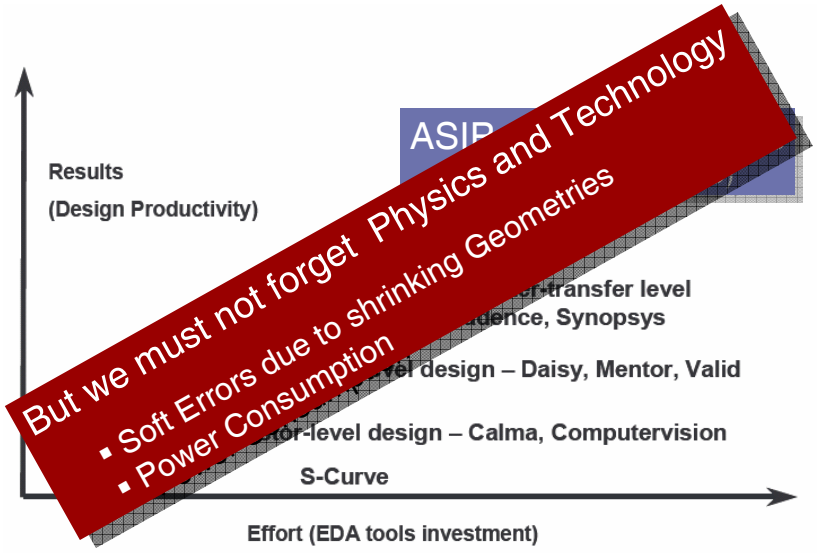
**Future improvement in systems performance will require an integrated design approach**

| | |
|---|---|
| Application | Languages<br>Software Tuning<br>Efficient Programming<br>Middleware |
| System Level | Dynamic Optimization<br>Assist Threads<br>Morphing Support<br>Fast Computation<br>Migration<br>Power Optimization<br>Compiler Support |
| Chip Level | Compiler Support<br>Morphing<br>Multiple Cores<br>SMT<br>Accelerators<br>Power Shifting<br>Interconect<br>Circuits |
| Technology | Silicon Innovation<br>Packaging<br>Efficient Cooling<br>Dense SRAM, embedded DRAM |

- Microprocessor frequency will no longer be the dominant driver of system level performance
- Integration over the entire stack, from semiconductor technology to end-user applications, will replace scaling as the major driver of increased system performance
- Systems will be designed with the ability to dynamically manage and optimize power
- Scale-out and small SMPs will continue to outpace scale-up growth
- Systems will increasingly rely on modular components for continued performance leadership

Source: Lisa Su /IBM: MPSoC 05 Conference 2005

## Core Proposition





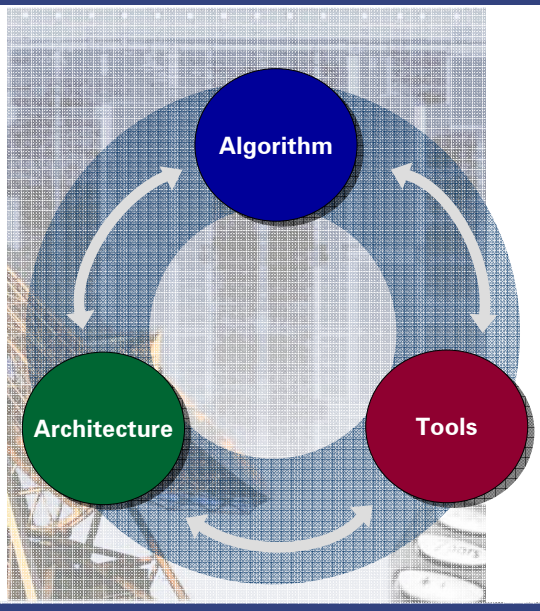

## The Human Element

**Building and managing an interdisciplinar engineering team of**

1. **Algorithm Designers**
2. **Computer/Compiler A**
3. **System Integr**
4. **RTL De**

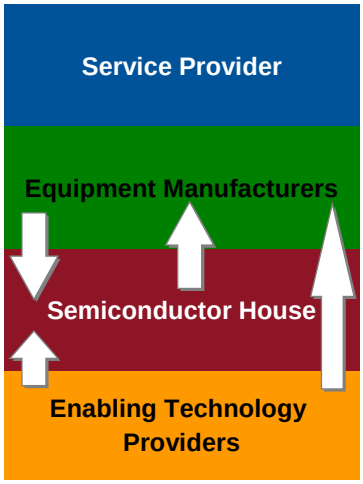
No psycho babble: It is the most critical element

## Cross-disciplinary Task






## Food Chain and Alliances

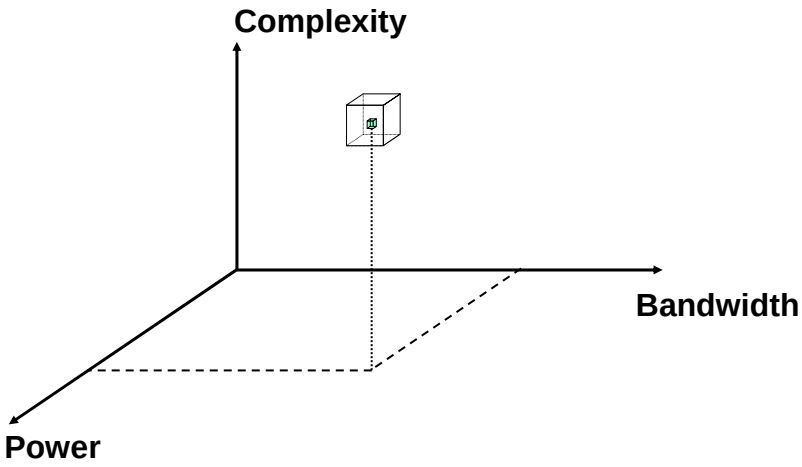
## Alliances and the Business Equation

- **Managing alliances is a key to success**
  - EDA
  - Mobile provider
  - Semiconductor company



# Receiver Structure , Models and Performance Metrics

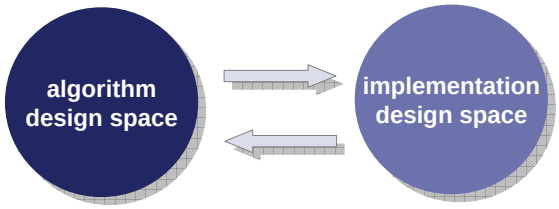
## Design - Space I: Physical Layer





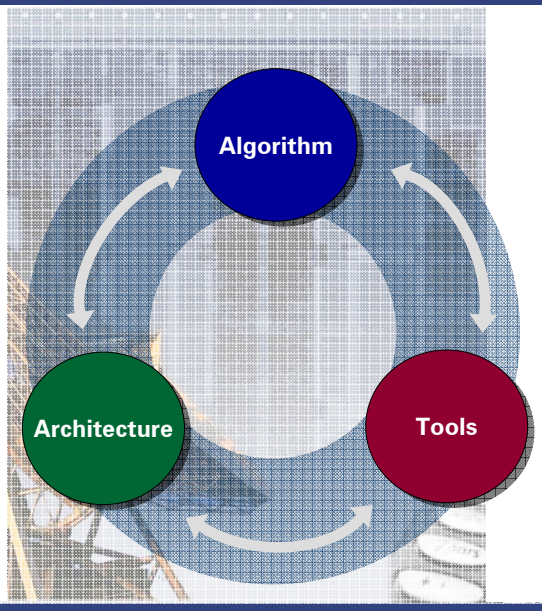

## System Design

**System design = algorithm design + implementation**




**JOINTLY** optimizing algorithm and architecture

## Center-of-Gravity Approach






# Methodology

## Design - Methodology: I

**Mathematical Theory and Experiment**

**are complementary**



## Design - Methodology: II

**Mathematical Theory provides Bounds**

1. Estimation and Detection Theory used to systematically derive (optimum) Receiver Structures

   ⇨ **Synthesis**

2. Mathematical Analysis used to compute Performance Bounds

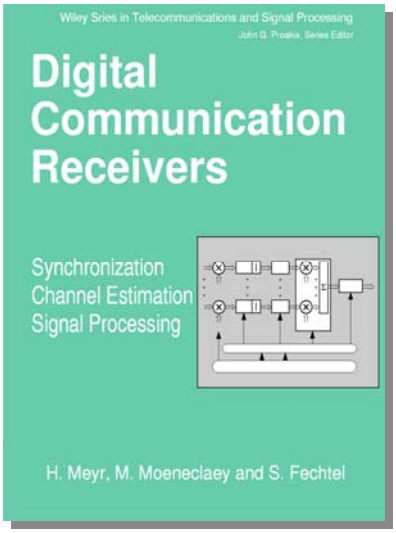
   ⇨ **Analysis**

## Design Methodology III

**Computer Simulation is used to**

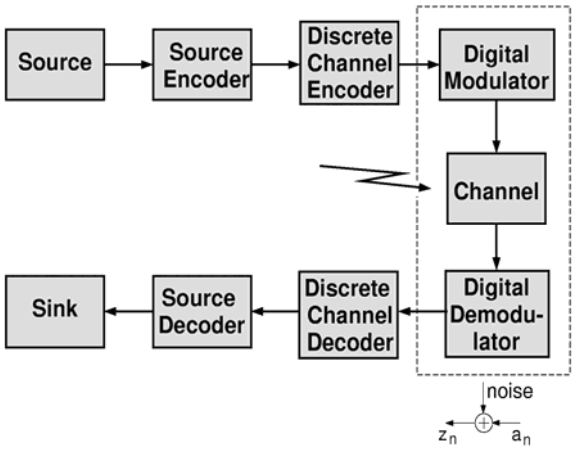
1. **Obtain numerical Performance Data**
   - Detection Loss
   - Implementation Loss
2. **Validate a Design**
   (Conformance to Standards)
3. **Verify Correctness of Implementation**
   (Verification) against Testpattern



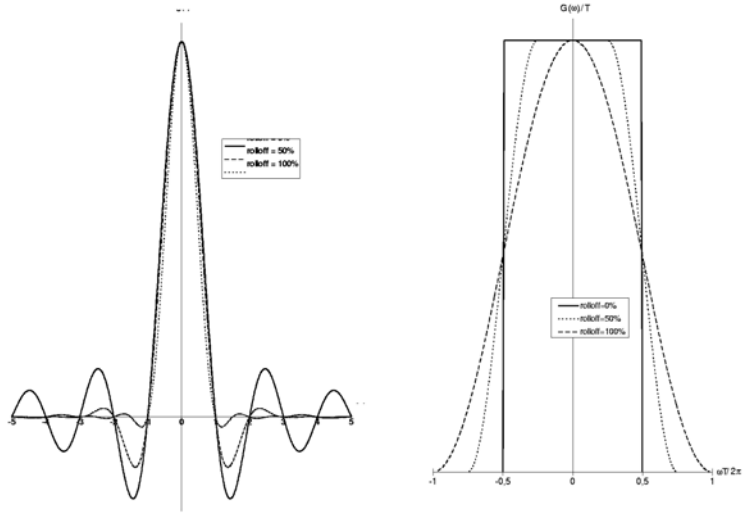


# Models

## Communication Model



# Signal Model




Source: CDMA, Principles of Spread Spectrum Communication, Andrew J. Viterbi

## Received Bandlimited Signals

$$r(t) = \sum a_n g(t - nT - \varepsilon T) e^{j(\theta + \Omega t)} + n(t)$$

$\varepsilon$, $\theta$, $\Omega$ } unknown Parameters

$\{r(kT_s)\}$ : contains all information (sufficient statistics)



## Approximation by BL Signals

$$E\left\{ \left| x(t) - x_{BL} \right|^2 \right\} = \int_{|\omega| \geq BL} S_x(\omega) d\omega$$

Approx. of non-bandlimited Signal x(t) by BL -Signal

$$\sum_{-k}^{k} x(kT_s)\varphi(k)$$

Truncation defines (2K+1) dim. Approx. In Vector space

## Equivalence of digital/analog Signal Processing

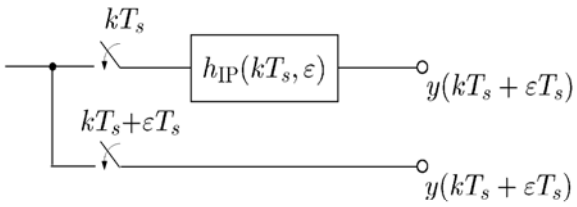
$$y(t)\Big|_{t=kT_s} = \int_{-\infty}^{\infty} h(kT_s - \nu)x(\nu)d\nu$$

$$= T_s \sum_{n=-\infty}^{\infty} h(kT_s - nT_s)x(nT_s)$$



## Properties

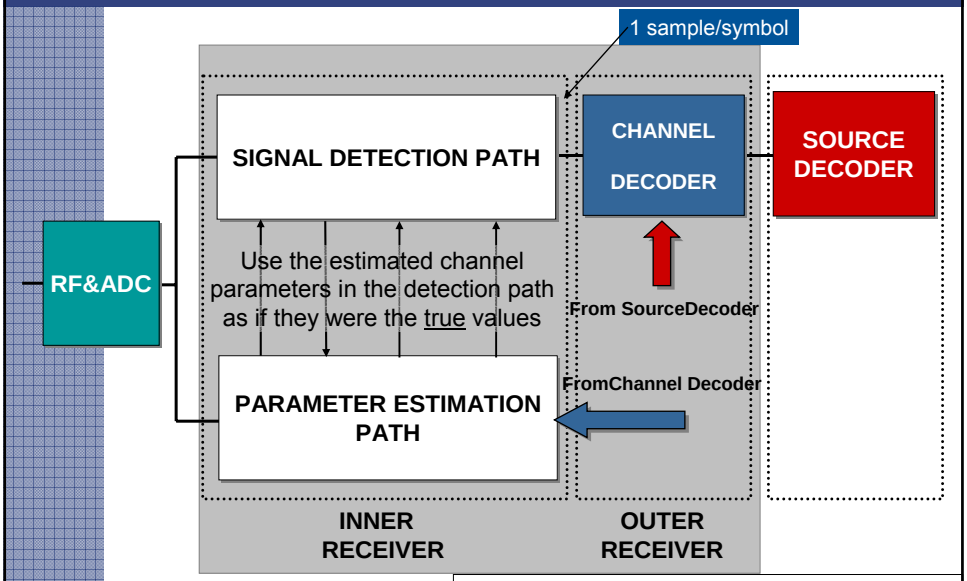
**Shift theorem**




$$y(t + \varepsilon T_s) = \sum_n y(nT_s + \varepsilon T_s)\mathrm{si}\left[\frac{\pi}{T_s}(t - nT_s)\right]$$

$$= \sum_n y(nT_s)\mathrm{si}\left[\frac{\pi}{T_s}(t + \varepsilon T_s - nT_s)\right]$$

## Canonical Receiver Model




H.Meyr et al., " Digital Communication Receiver", J.Wiley 1998



## Receiver Task

### Inner Receiver

To provide a "good" channel to the decoder based on the principle of synchronized Detection.

**NOTHING ELSE !**

### Outer Receiver

To decode the information

## Performance Measure

**Inner Receiver**

Properties of the estimator

- Variance
- Unbiased

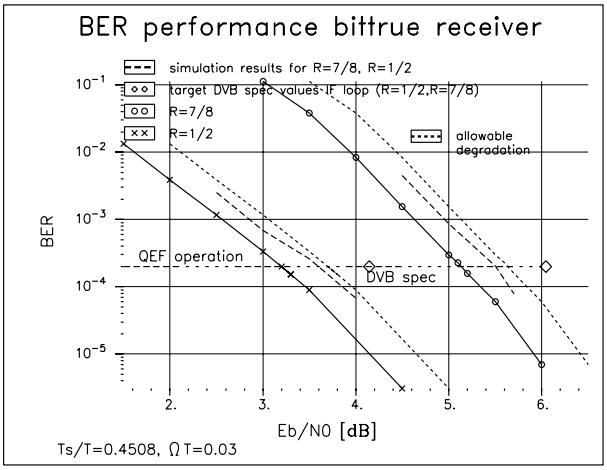
**Outer Receiver**

Bit-error-rate of the coded system



## Performance Loss

- **Detection Loss of synchronized Detection**
  - Δ SNR (dB) required to achieve the performance of perfect channel knowledge . (Infinite Precision arithmetic assumed)
- **Implementation Loss**
  - ΔSNR (dB) resulting from finite precision arithmetic and algorithmic approximations




Source: Digital Communication Receivers, H. Meyr, M. Moeneclaey, S.A. Fechtel

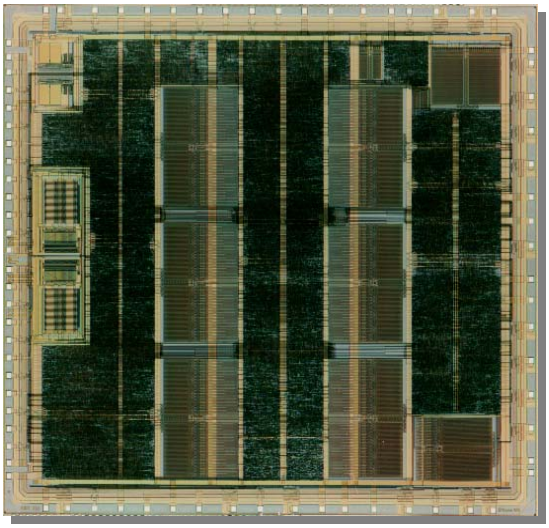


## Complexity DVB-S

| | Area (cell area without RAM) | Lines VHDL |
|---|---|---|
| Timing & Carrier Sync. | 32 % | 7000 (+1000 .dc) |
| Viterbi Dec. | 40 % (+RAM 15 mm^2) | 4000 (+ 340 .dc) |
| Frame Sync. | 1.5 % | 700 |
| Deinterleaver | 2 % (+RAM 1.5) | 640 |
| RS Decoder | 23 % (+RAM 1.4) | 5400 (+ 630 .dc) |
| Descrambler | 1 % | 360 |
| System | 100 % | 18100 |

Source: Digital Communication Receivers, H. Meyr, M. Moeneclaey, S.A. Fechtel

## DVB-S Chip



Siemens-RWTH Aachen
(ISS) Design 1997

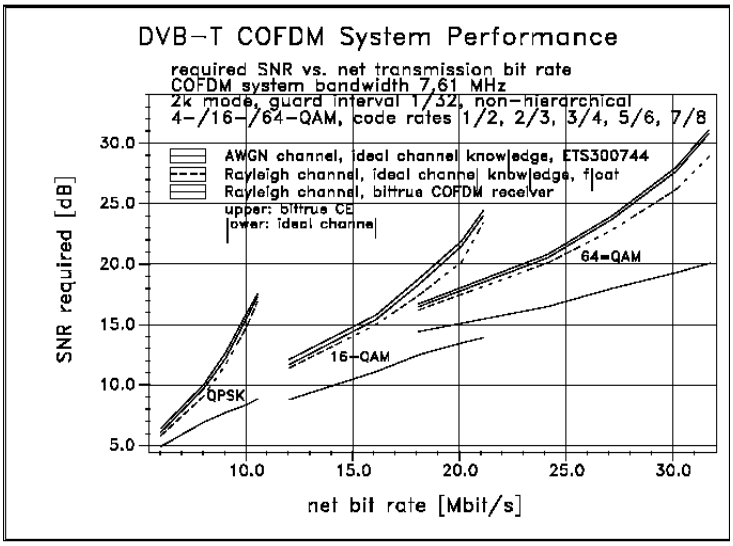
0.5 m technology
3 metal layer
1.5 W @ 88 MHz
> 500 k transistors
First silicon success



## DVB-T Specifications

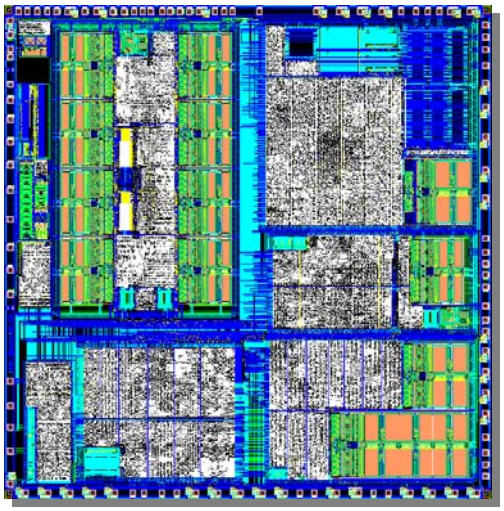
- **Digital terrestrial video broadcasting:**
  - high symbol rates: up to 7.4 Msym/s
  - sensitive modulation: 4 - 64 QAM
  - net bit rate up to 31.67 Mb/s
  - wide range of channels: (AWGN) 0 < Tau < 224 ❍s (SFN)
  - error correction:
    - outer coder: Reed Solomon (204,188)
    - inner code: punctured convolutional
  - BER < 10e-9 (after RS)
  - 3dB < Es/No < 40 dB
- **Challenges: > 200 transmission modes**
  - algorithms
  - design methodology

## System Performance: DVB-T






## DVB-T Chip: First single Chip Solution



Joint Infineon-Nokia-ISS Design 1999

AGC: Automatic Gain Control
IQ: IQ-Mixer and Resampling
PPU: Postprocessing Unit
FFT: Fast Fourier Transform (2k,8k)
DTO: Digital Timing Oscillator
RAM: OFDM Symbol Memory
CHE: Channel Estimation
IFFT: Inverse FFT and Fine Timing
ESG: Equalization and Softbit Generation
FEC: Forward Error Correction (Viterbi, Reed-Solomon)

## DVB-T Complexity

- **Analog part : 10%**
  - Input interfaces
  - DC removal
  - anti-aliasing filter
  - ADC,AGC
- **Digital demodulator: 60 %**
  - Channel estimation and equalization
  - synchronization
  - control flow implementation
  - FFT (alone 30%)
- **Channel decoder : 20 %**
  - Viterbi and RS decoder
- **Miscellaneous : 10%**
  - IIC bus controller, DAC



## Design Space : Architecture and Algorithm

### Inner Receiver

The algorithms of the inner receiver are never specified by the standard

⇨ **BOTH algorithm** and **architecture** space exploration

### Outer Receiver

The decoder is **exaclty** specified in the standard

⇨ **ONLY architecture** space exploration

# Massive Parallel Processing on Heterogeneous MPSoC

## Parallel Computing in Mobiles

**Massive Parallelism required in the foreseeable future**

| | 2003 | 2009 | 2013 |
|---|---|---|---|
| Frequency (MHz) | 300 | 600 | 1500 |
| Giga Operations | 0,3 | 14 | 2458 |
| Operations per Cycle | 1 | 23 | 1638 |

Source: International Technology Roadmap for Semiconductors (ITRS, TX 2003)

## Why Many-Processors Architectures today?

- **Not because of a fundamental breakthough in novel software and parallel architecture**
- **…..simply because the problems with tradtional architectures pose an even greater challenge**



## Guding Principles for Manycore SoC I

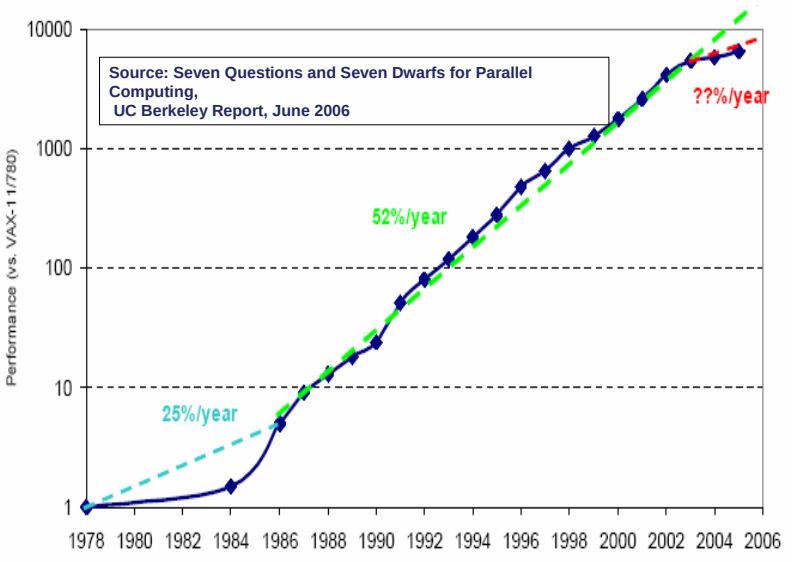
- **Energy Efficiency and Power are the dominating issues**
  - ⇨ There exists a fundamental trade-off between energy efficiency and flexibility
- **Below 65nm high soft and hard error rates occur**
- **Bandwidth improves by at least the square of the latency**
- **Memory wall: Load and stores are slow ( up to 200 cyles to access DRAM)**

## Guding Principles for Manycore SoC :II

- **Multiplies are fast**
- **Instruction Level Parallelism (ILP) wall: Dimishing return on finding new ILP**
  - ⇨ Brick wall:Power Wall+Memory Wall+ILPWall
  - ⇨ Increasing parallelism and decreasing clock frequency is the primary source of improving processor performance



## GP -Processor Performance Improvement between 1978 and 2006



## Parallel Computing

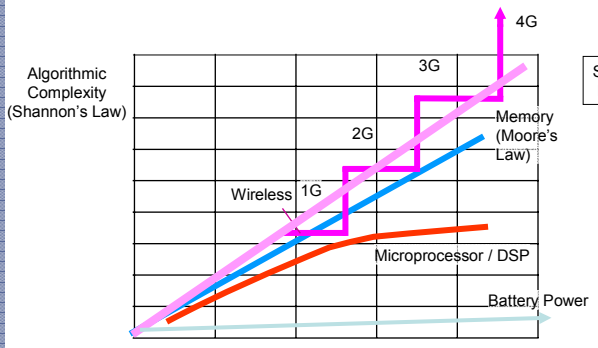
*"Switching from sequential to modestly parallel computing will make programming much more difficult…….without a dramatic improvement in performance"*

Source: Seven Questions and Seven Dwarfs for Parallel Computing, UC Berkeley Report, June 2006

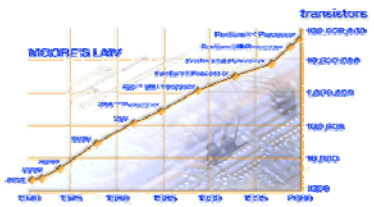
⇨ **We need to go to from multiple processors to many cores**

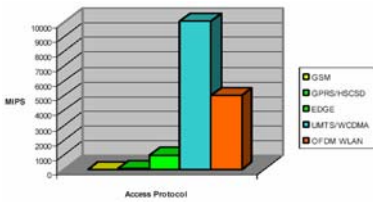


## The Need for New Architectures

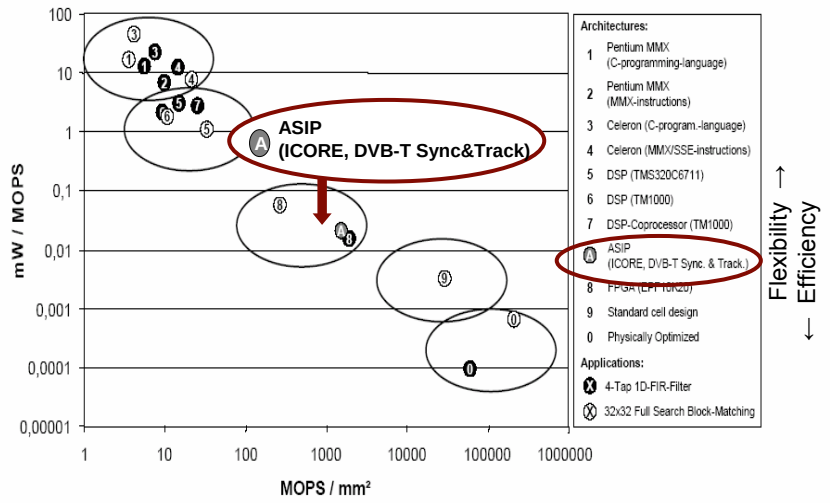



Source: R.Subramanian. Berkeley Design Automation Inc



## Computational Efficiency vs. Flexibility





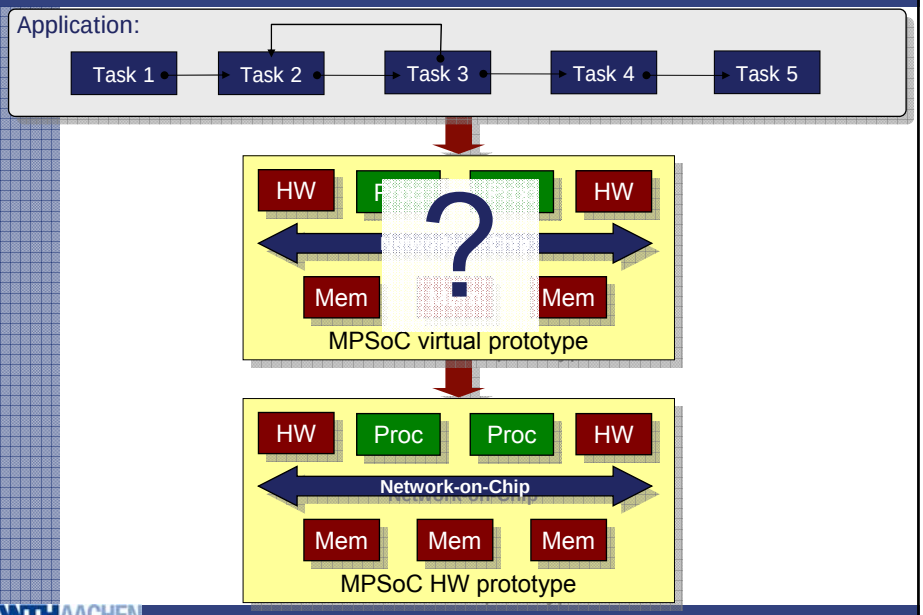

How to Exploit the Design Space and Design MPSoC´s?

## Design Principles

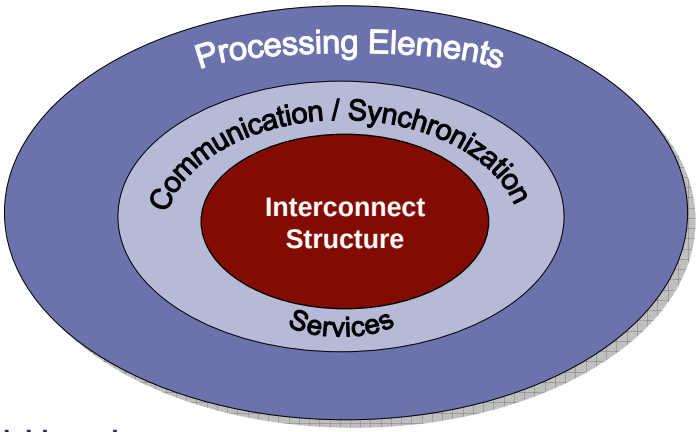
- **Focus ….** first on applications and constituent algorithms, not the silicon architecture
  - Identify key attributes of the application
    - Identify periodicity of signal processing taks (cyclostationarity)
    - Block processing
    - Identify loose coupling of tasks
- Use…. extensive profiling to find the spatial and temporal mapping with the following goal
  - **Minimize** the processor flexibility to a constrained set to optimize the energy efficiency
  - **Maximize** the software parameterizability and ease of use of the programmer's model for flexibility



## MPSoC design flow: Temporal and Spatial Mapping



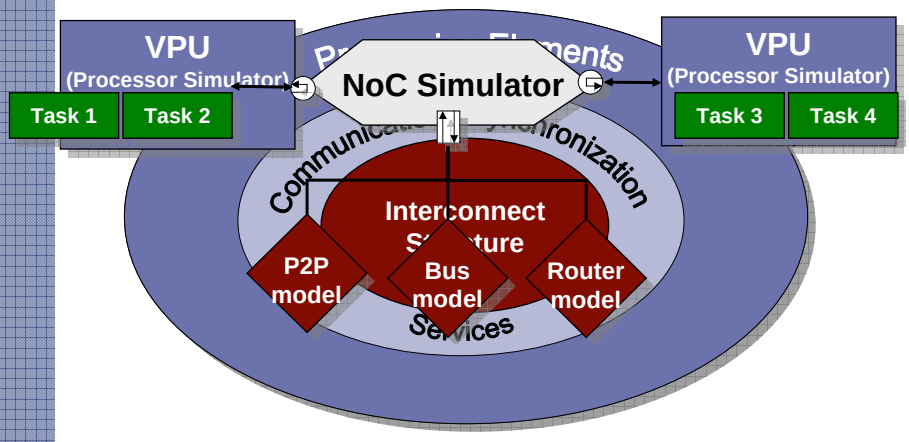
## MPSoC exploration principles




- Divide and conquer
- Separate processing elements from communication
- Early SW performance estimation



## MPSoC virtual prototyping platform



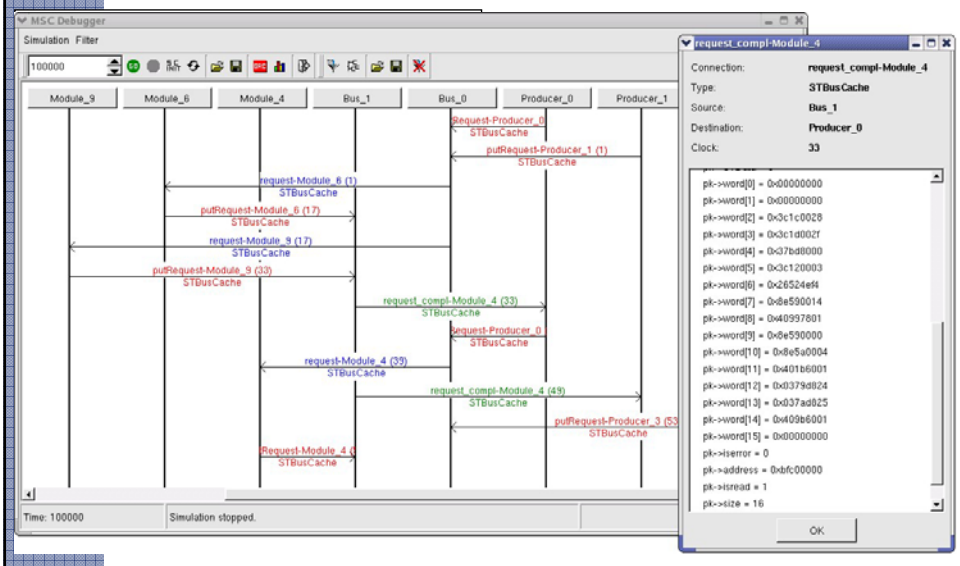

- Communication: CoWare Architect´s View Framework (AVF)
- VPU: virtual processing unit
- Enables modeling spatial and temporal task-to-PE mapping

## MPSoC exploration Results

- An MPSoC is defined by its processing elements (PE) and their interconnect (NoC)
  - Interconnect is defined by its topology. Communication performance is measured for a given topology
  - PE performance is determined by a set of numbers

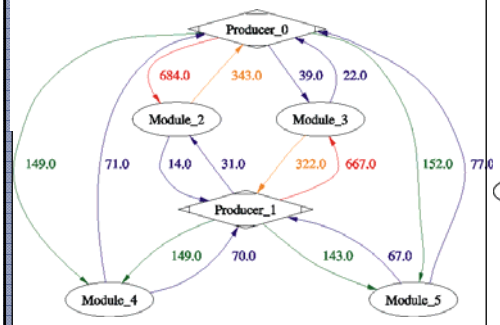


## Message Sequence Chart (MSC) Trace

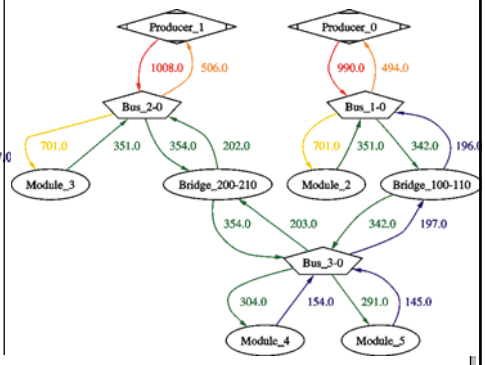


## Aggregated Communication Graph

**Interacting Partner View**

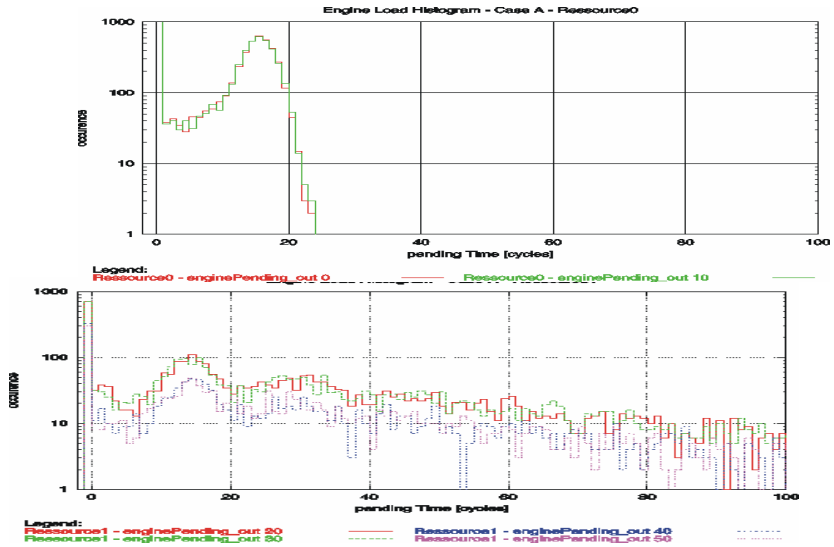



**Topology View**

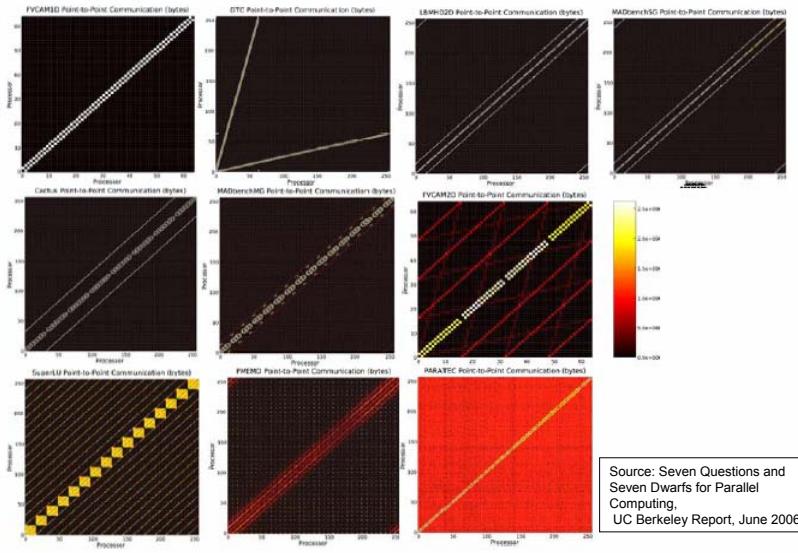





## Histogram Views



## MPSoC Exploration Results: Communication



Source: Seven Questions and Seven Dwarfs for Parallel Computing,
UC Berkeley Report, June 2006



## The „Key Algorithm“ Propostion

**Each application is composed of a small number of fundamental algorithms ( „Nuclei“) that represent a significant amount of the computation.**

**⇨ Focus on an efficient composition („design of an MPSoc) or mapping („programming of the MPSoC“)**

## Composition of Nuclei

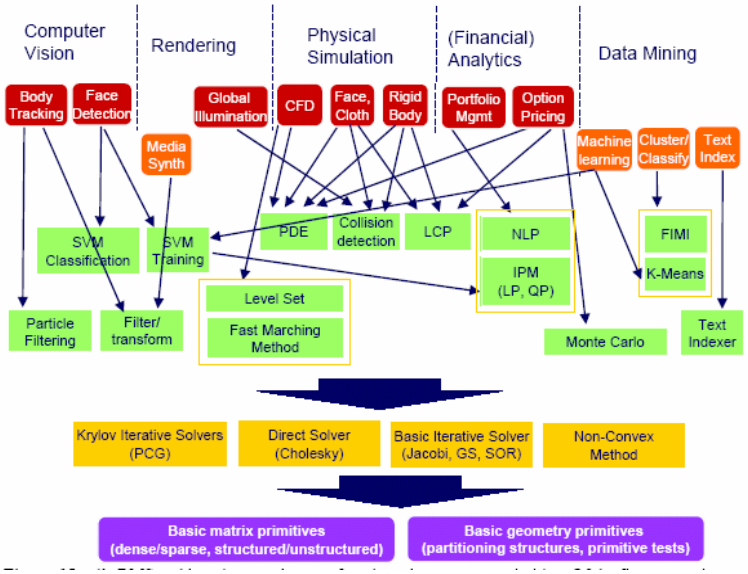
Nuclei can be ***composed/mapped*** on a multiprocessor in three different ways

- ***Temporally distributed*** or time-shared on a common processor
- ***Spatially distributed*** with each Nucleus occupying one or more processors
- ***Pipelined:*** A single nucleus is distributed in time and space
  - In a given time slot a nucleus is running on a group of processors
  - On a given processor a group of nucleus computation run over time

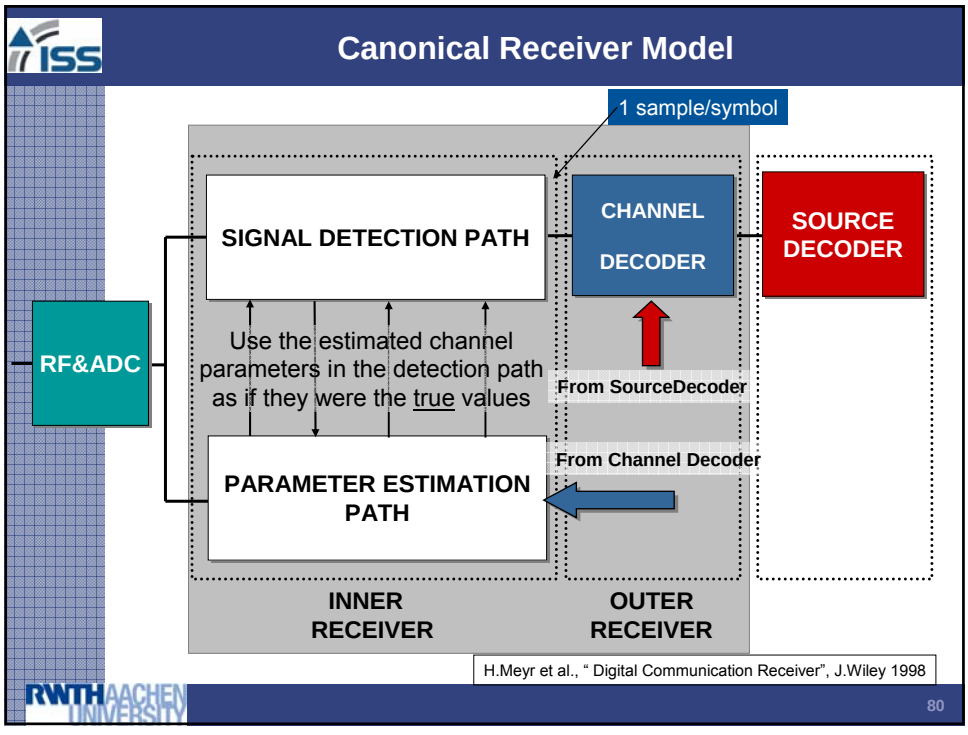
Source: Schaumont et. al.2001



## Intel RMS View (Recognition, Mining,Synthesis)



# Example: Baseband Processing for 4G



## Lessons Learned from Design Reviews 2005

- **Virtual Prototype (Product) of utmost importance**
  - Early customer interaction
  - Debugging
  - Verification&Validation
- **Product Differentiator**
  - 80% of Area and Power Consumption in the inner receiver (Algorithm and Architecture Design)
  - 10-15% of Area and Power Consumption in Decoder (Architecture Design)
  - 5% of Area and Power Consumption in the ARM (But major portion of cost is SW/Protocol implementation)



## Properties of the Task

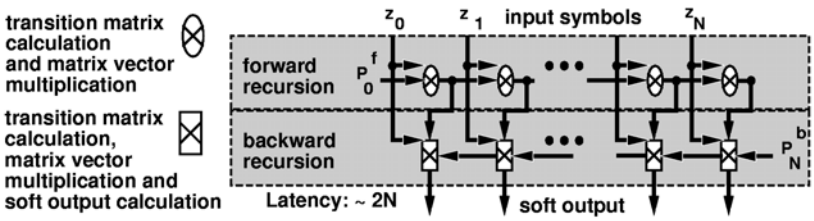
- The signal/information processing task can be naturally **partitioned**
  - Decoders
  - Filters
  - Channel estimator
- The building blocks are **loosely coupled**
- The signal processing task is (mostly) **cyclostationary**

## From Function to Algorithm Classes

- Butterfly unit
  - Viterbi & MAP decoder
  - MLSE equalizer
- Eigenvalue decomposition (EVD)
  - Delay acquisition (CDMA)
  - MIMO Tx processing
- Matrix-Matrix & Matrix-Vector Multiplication
  - MIMO processing (Rx & Tx)
  - LMMSE channel estimation (OFDM & MIMO)
  - Iterative (Turbo) Decoding
  - Message Passing Algorithm , LDPC Decoding
- CORDIC
  - Frequency offset estimation (e.g. AFC)
  - OFDM post-FFT synchronization (sampling clock, fine frequency)
- FFT & IFFT (spectral processing)
  - OFDM
  - Speech post processing (noise suppression)
  - Image processing (not FFT but DCT)



## Decoder for Convolutional Codes




$$\underline{x}_{k+1} = \begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} a_{11,k} & a_{12,k} \\ a_{21,k} & a_{22,k} \end{bmatrix} \otimes \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} = \begin{bmatrix} a_{11,k} \otimes x_{1,k} \oplus a_{12,k} \otimes x_{2,k} \\ a_{21,k} \otimes x_{1,k} \oplus a_{22,k} \otimes x_{2,k} \end{bmatrix}$$

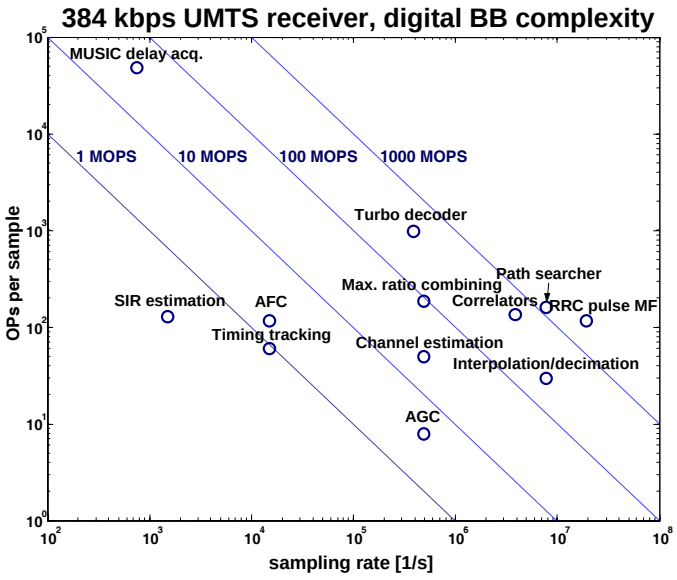
| OPERATIONS | MAP | LOGMAP | VITERBI |
|---|---|---|---|
| $x \oplus y$ | $x + y$ | $\log_e[e^x + e^y]$ | $\max(x,y)$ |
| $x \otimes y$ | $x \cdot y$ | $x + y$ | $x + y$ |

## Algorithmic Descriptors

- **Clock rate of processing elements (1/Tc)**
- **Sampling rate of the signal (1/Ts)**
- **Algorithm characteristic**
  - Complexity (MOPS/sample)
  - Computational characteristic
    - Data flow
      - Data locality
      - Data storage
      - Parallelism
    - Control flow
- **Connectivity of algorithms**
  - Spatial
  - Temporal



## 384 kbps UMTS Receiver BB Complexity



# Hardware



## Employ all forms of Parallelism

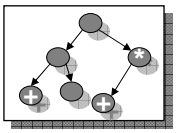
## Potential Processor Parallelism

- **Three form of instruction-set parallelism**
- Instruction parallelism
- Data parallelism
- Pipeline parallelism

**Multi-issue instructions (VLIW)**
- Instruction size L
- Number of operation slot per instruction
- Operation mix in each slot

**Maximum parallism: LxMxN**

**SIMD Instructions**
- Types of vector operations M
- Number of vector elements
- Number and size of vector register files



**Fusion of Operation**
- Number and type of composing operations N
- Number of inputs and outputs
- Latency

Source: C.Rowen, Tensilica

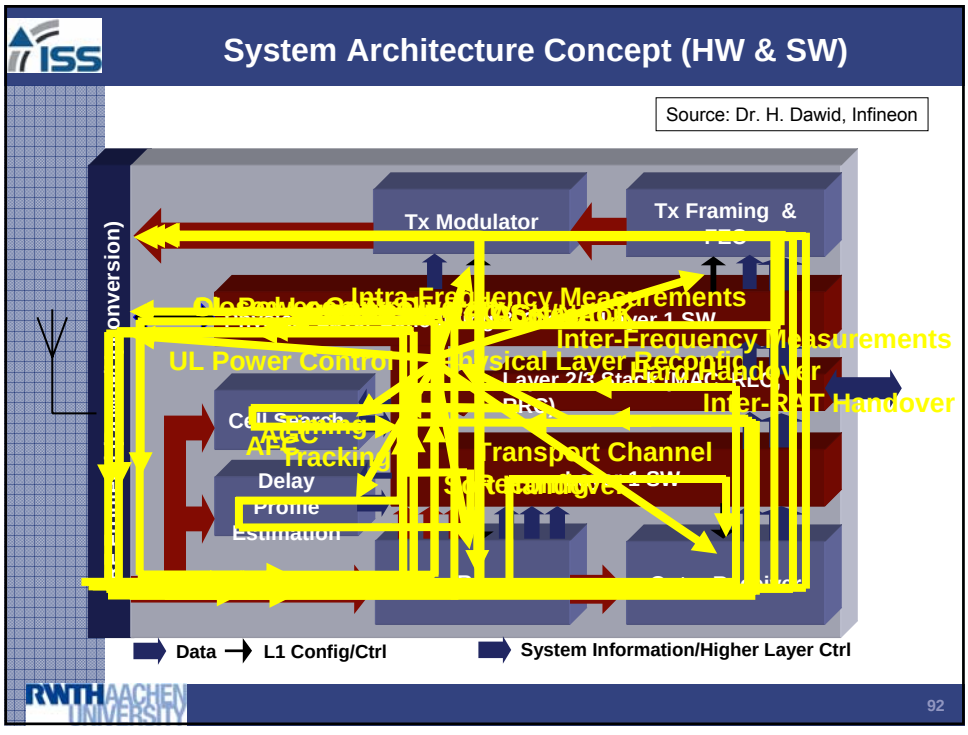


## Memory Architecture

- DRAM prices have draramatically decreased
  - From $ 10,000,000 for1 Gigabyte in 1980
  - To $ 100 in 2006
- Memory wall is the major obstacle to good performance of many applications

⇨ Novel memory architecture are a key component of ASIP

# Programming Models and Design Methodology



## Programming Models

- **Goal: To maximize programmers productivity**
- **Requirement**
  - **Independent of number of processors**
  - **Allow to describe concurrency naturally**
  - **Support rich set of data types**
  - **Support parallel models**
    - Data level parallelism
    - Instruction level parallelism
    - Independent task paralleism
- **Autotuners should take on a complementary role to compilers**
- **Far more formal methods must be developed to guarantee correctness ( e.g. avoid dead locks using threads )**

Source: Seven Questions and Seven Dwarfs for Parallel Computing, UC Berkeley Report, June 2006



## Software Synthesis and Autotuners

**Principle of Autotunners:**

- Optimize a set of library kernels by generating many variants of a given kernel
- Benchmark each variant on a given platform

Source: Bilmes et al. 1997; Frigo and Johnson 1998; Whaleyand Dongarra 1998, IM et al. 2005

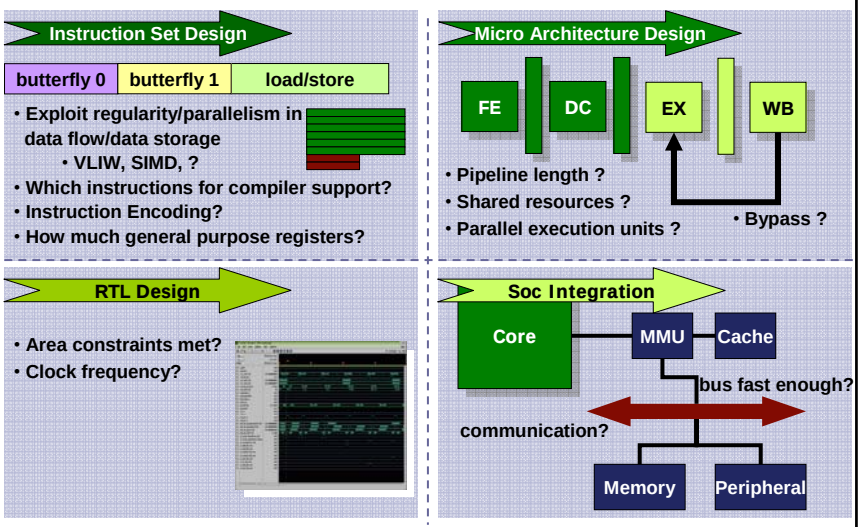
## Conclusion

- We are presently at a juncture of the semiconductor industry as it seldom occurs
  - The existing ( RTL) design paradigm has reached its end-of-life ⇨ we need to move to a higher level of abstraction (ESL) to keep the cost within resonable bounds
  - The existing processor multiprocessor architectures and the programing tools do not scale
    ⇨ we need much innovation in these areas to make economic use of scaling



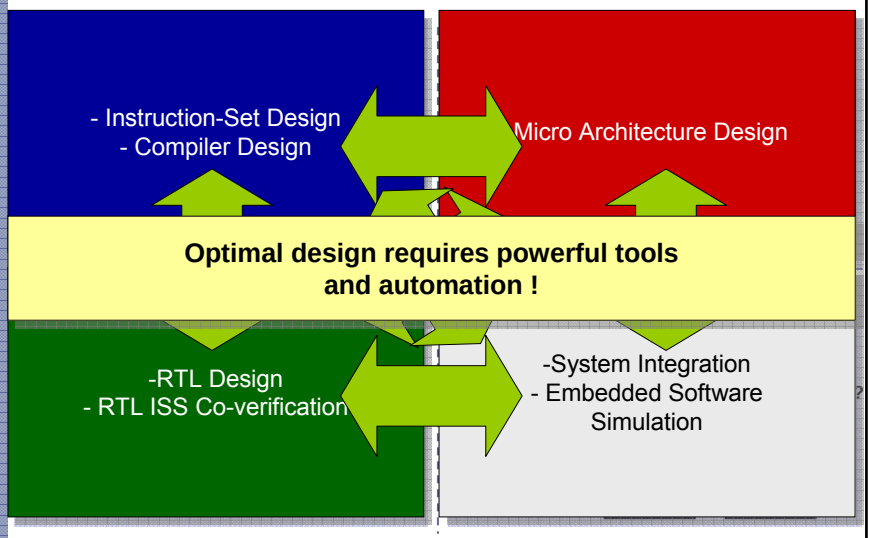
# Application Specific Processors Design

## Processor Design Space



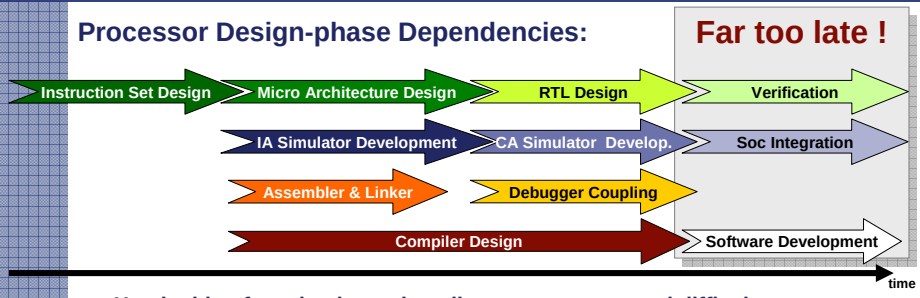



## Processor Design Space



## Traditional Processor Design

**Processor Design-phase Dependencies:**




- Handwriting fast simulators is tedious, error-prone and difficult
- Compiler cannot be considered in the architecture definition cycle
  - Risk of compiler un-friendly instruction-set
- Inconsistencies between tools and models
- Traditional design methodology does not allow for efficient processor design
- Verification, Software Development and SoC integration too late
  - Real-world stimuli and SoC interaction might reveal bottlenecks

**Design phases need to be parallelized!**



## Today: ADL based Processor design

**OBJECTIVE**

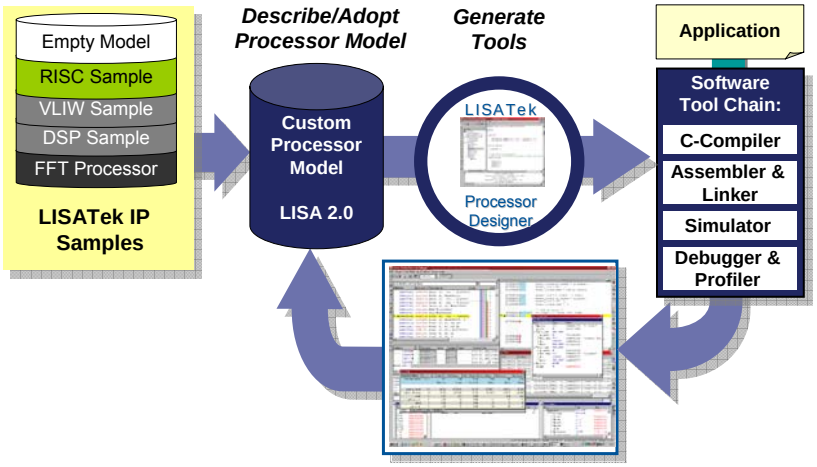
**Improve Design- and Implementation Efficiency**

**…..at the same time**

## Architecture Description Language based Processor Design

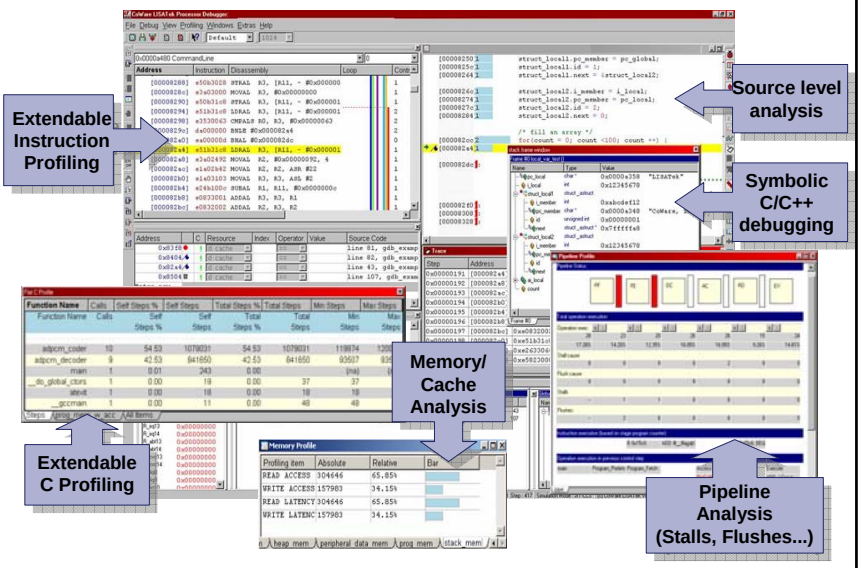
- The purpose of an **architecture description language** (e.g LISA) is:
  - To allow for an iterative design to efficiently explore architecture alternatives
  - To jointly design “Architecture –Compiler” and on chip communication
  - To automatically generate hardware (path to implementation)
  - To automatically generate tools
    - Assembler ,Linker, Compiler, Simulator, co-simulation interfaces
- From a **single** model at various level of temporal and spatial abstraction



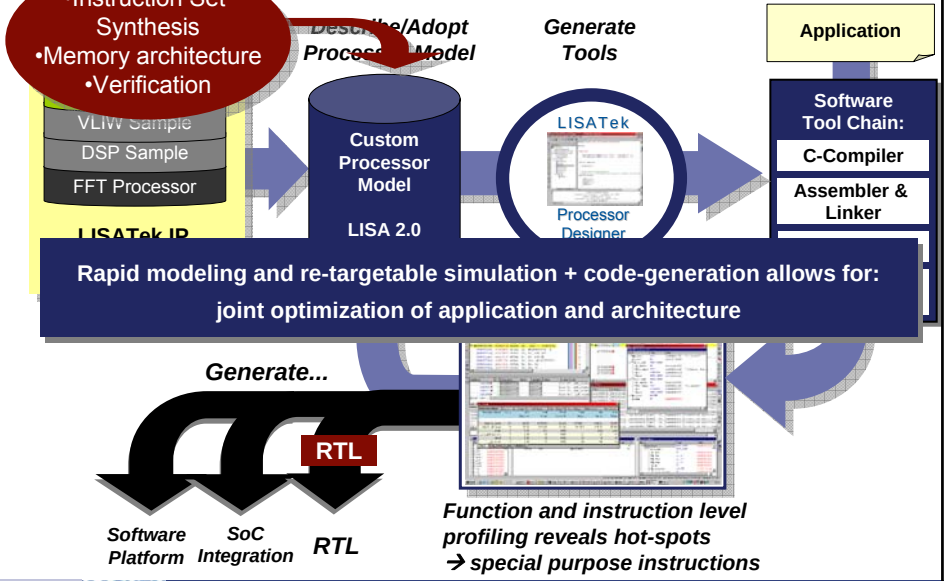



***Function and instruction level profiling reveals hot-spots***
***→ special purpose instructions***

## LISATek (Multi Core) Analyzer





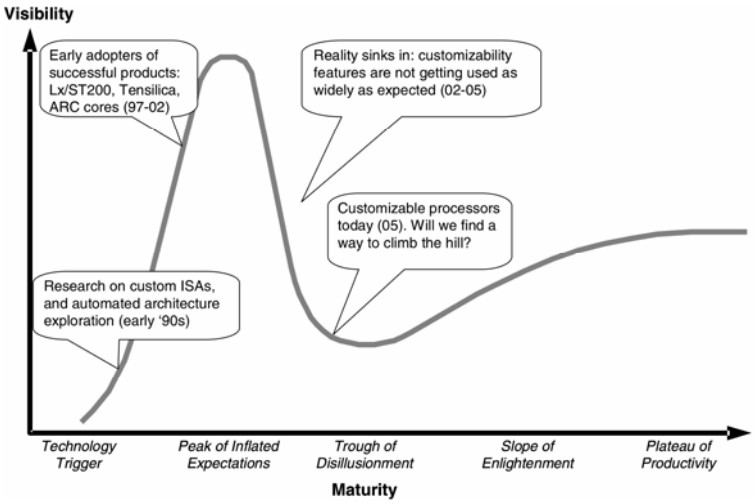



# Orthogonalize „Workbench“ and Optimization Tools

- R.Leupers.et al „Fine Grained Application Source Code Profiling for ASIP“, DAC 2005
- R.Leupers et al., “A Design Flow for Configurable Embedded Processors based on Optimized Instruction Set Extension Synthesis”, DATE 2006
- P.Ienne,R.Leupers (Editors), "Customizable Embedded Processors”, Morgan Kaufmann (Elsevier), 2006

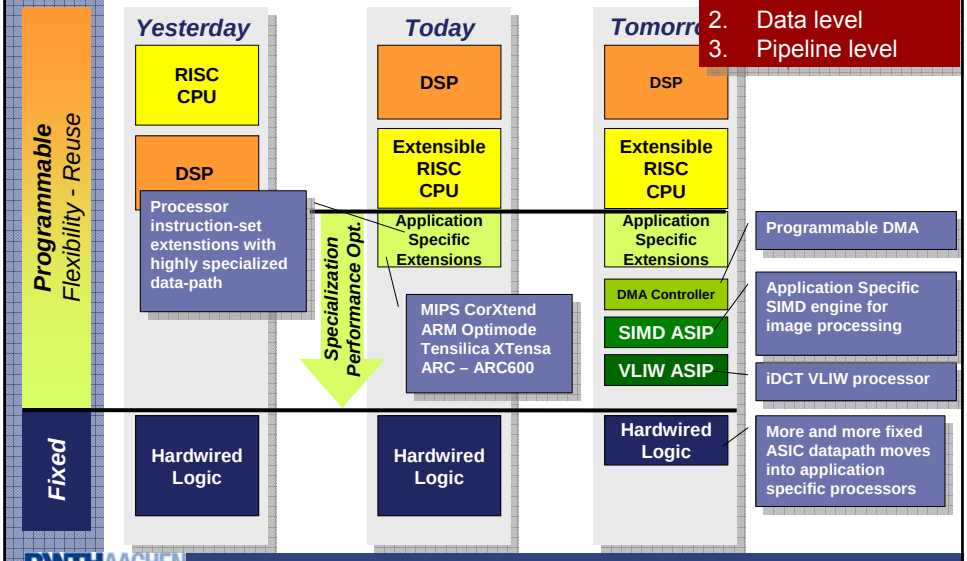







J. Fisher, “Customizing Processors :Lofty Ambitions, Stark Realities, Chapter 2 in: Customizable Embedded Processors, ed. By L.Leupers, Paolo Ienne, to be published by Morgan Kaufmann July 2006

## Mapping Application to Architecture



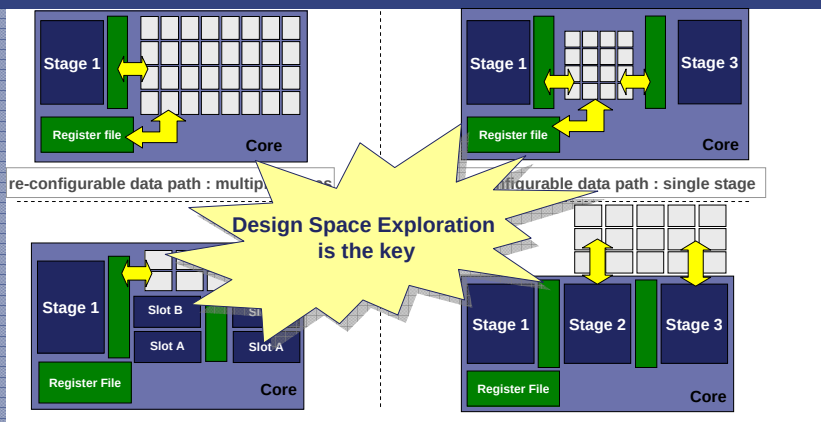



## From C-to Complex Instructions

# Mapping Algorithm to Architecture

### Fixed and Re-Configurable ASIP

## rASIP : A Huge and Complex Design Space




re-configurable data path in VLIW slots

re-configurable data path in loosely coupled rASIP

**Pre-fabrication**
- ASIP architecture
- FPGA architecture
- ASIP-FPGA Interface
- Static / Dynamic re-configurability
- ..

**Post-fabrication**
- Instruction-Set Extension
- Configuration code generation
- Scheduling and Code-Generation
- FPGA-targeted optimization
- ..



# Case Studies

**References:**


- Tilman Glöckler,H. Meyr, Design of Energy efficient Application-Specific Instruction Set Processors, Kluwer Academic Publisher,2004
- Oliver Wahlen, C Compiler Aided Design of Application Specific Instruction-Set Processors Using the Machine Description Language LISA, Ph.D thesis submitted to Aachen University of Technology (RWTH), 2004

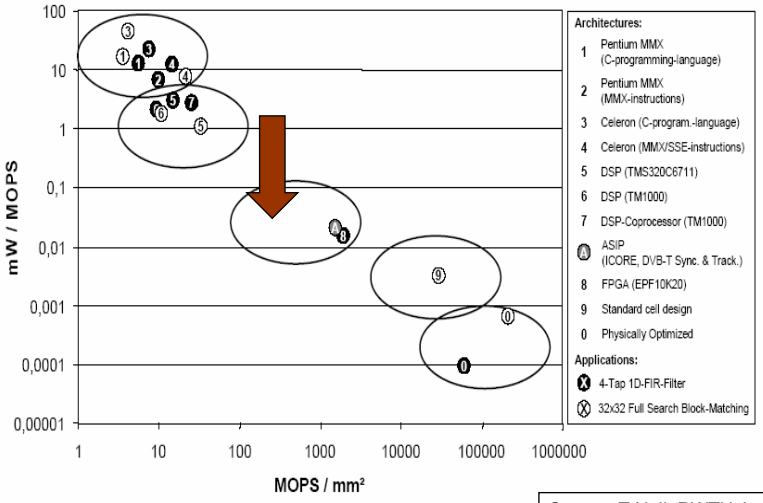
## The ICORE Example

### A low-power ASIP for Infineon DVB-T 2nd generation single-chip receiver:

- ASIP for DVB-T acquisition and tracking algorithms (sampling-clock-synchronization, interpolation / decimation, carrier frequency offset estimation)
- Harvard architecture
- 60 mostly RISC-like instructions & special instructions for CORDIC-algorithm
- 8x32-Bit general purpose registers, 4x9-Bit address registers
- 2048x20-Bit instruction ROM, 512x32-Bit data memory
- I2C registers and dedicated interfaces for external communication



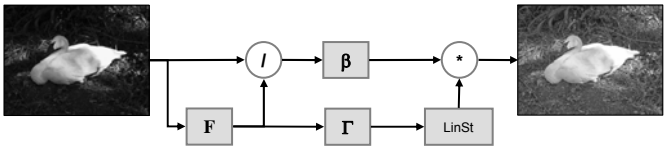
## Computational Effiency vs. Flexibility




Source: T.Noll, RWTH Aachen

## The Retinex Project

**Application:** Retinex-like Algorithms




**Knowledge:** Application Knowledge,
VLSI and ***Basic*** Processor Design Knowledge

**Outline:** From Specification to FPGA Prototyping

**Duration:** 7,5 Weeks

**A cooperation between Pisa University and RWTH Aachen University**

Newcom ISS VLSI Systems



## Retinex Architecture Reference

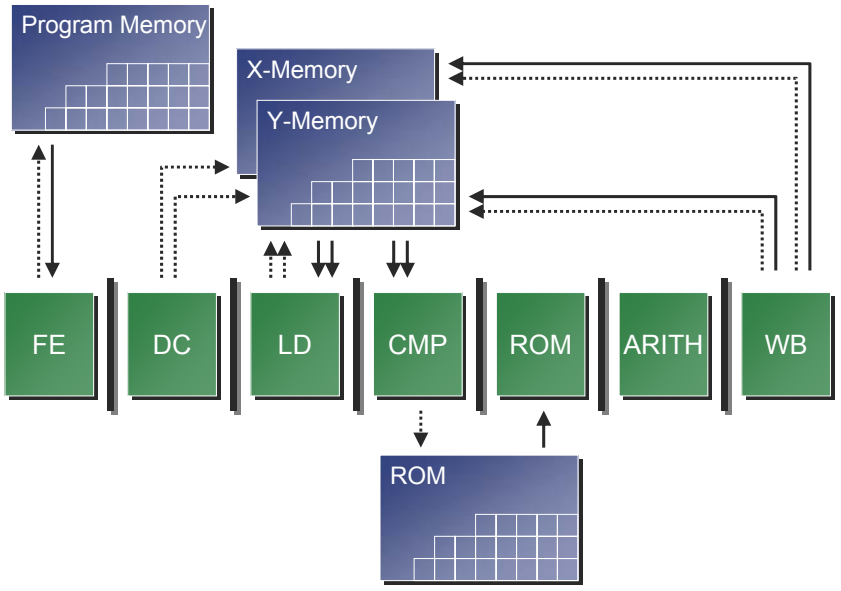
Paper presentation at DATE 2006

**ASIP DESIGN AND SYNTHESIS FOR NON LINEAR FILTERING IN IMAGE PROCESSING**

L. Fanucci, M. Cassiano and S. Saponara,
DIIEIT-Pisa University, Italy

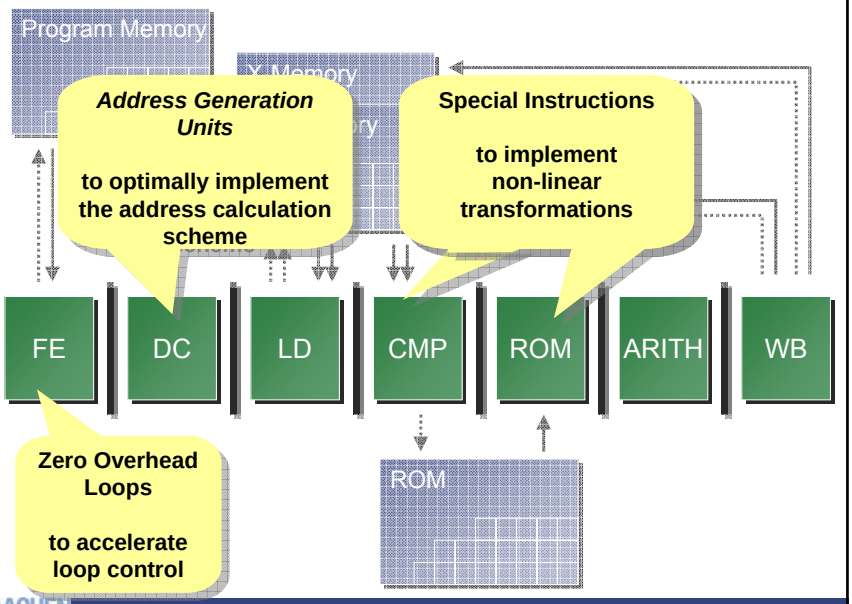
D. Kammler, E. M. Witte, O. Schleibusch, G. Ascheid,
R. Leupers and H. Meyr,
RWTH Aachen University, Germany

## The Retinex ASIP






## The Retinex ASIP



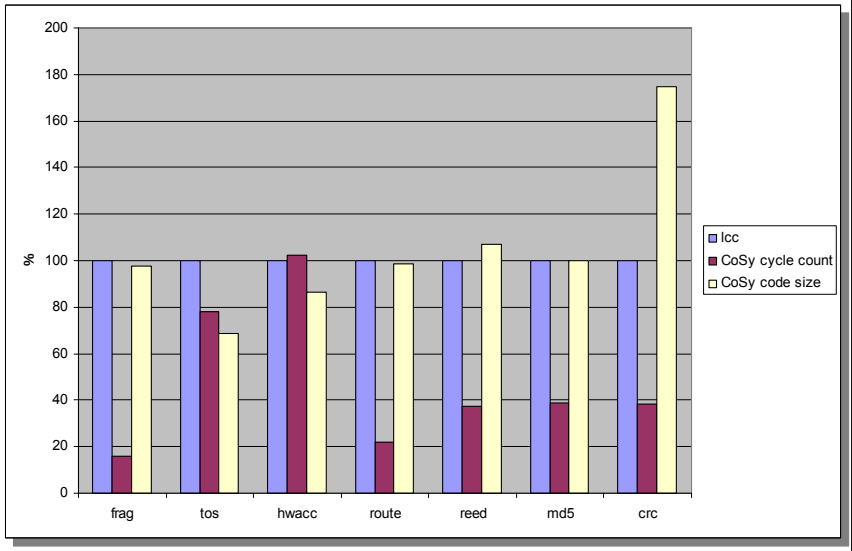
## Performance Comparison

| System | Athlon XP 3000+ | Retinex ASIP mapped on FPGA |
|---|---|---|
| Design Flow | plain C-application, compiled with gcc, executed on AMD Athlon | Optimized ASIP and handwritten assembly program (~100 lines of code) |
| Frequency | 2100 MHz | 16 MHz |
| Computation time (Picture 513x385) | ~ 3000 ms | 593 ms<br>~ 20 % of Athlon run-time |



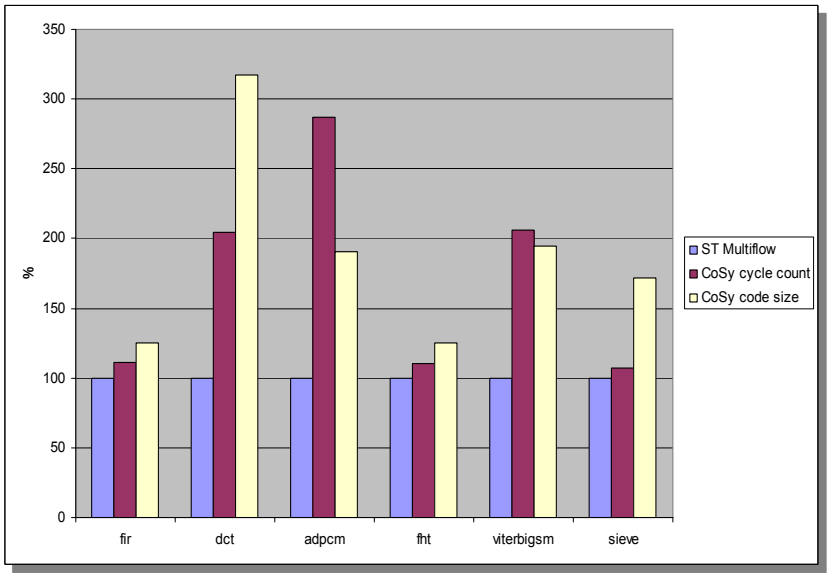
# Retargetable Compiler

## Infineon PP32 Network Processor






## ST200 VLIW Multimedia Processor



# Low Cost Commercial ASIP



## Project Goals

**Initial goal:**

- **Custom processor design to save royalties**
  LISA processor design
- **development of an ASIP with superior architectural efficiency**
  General purpose register file
- **support a smooth legacy code migration**
  Perl - translation script
- **an architecture which is smaller than the existing architecture**
  LISA !!!

## Development Time Sheet

Phase I

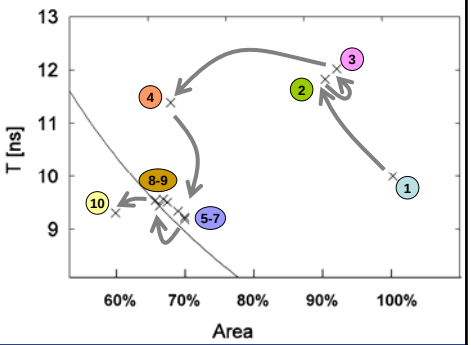
Phase II

| | |
|---|---|
| Initial Model | 4 weeks |
| Design Space Analysis | 3 weeks |
| Design Space Exploration | 4 weeks |
| - Address Calculation | 1 week |
| - Non-delayed Branches | 1 week |
| - Timing Improvement | ½ week |
| - Others | 1½ weeks |
| Translation Script | 5 weeks |
| Move Elimination | 2 week |
| Verification Script | 5 weeks |
| Synthesis & FPGA Mapping | 1 day |
| FPGA System ( one time effort) | 10 weeks |



## Moving through the Design Space

1. First synthesis of verified RTL code, no port constraints, no optimizations
2. Memory **port constraints**, autom. optimization: path sharing
3. Grouping in functional units for more detailed analysis
4. Change in **address calculation** enabling resource sharing
5. Critical path analysis → modification of **fetch mechanism**, optimization: decision minimization
6. Pin for FPGA prototype added
7. Implementation of non-delayed branches → prog-mem size reduction
8. Changed multiplier implementation from 32bit to 17bit
9. Removed functional unit grouping from 3
10. Final Synthesis: timing constraint adapted to synthesis results



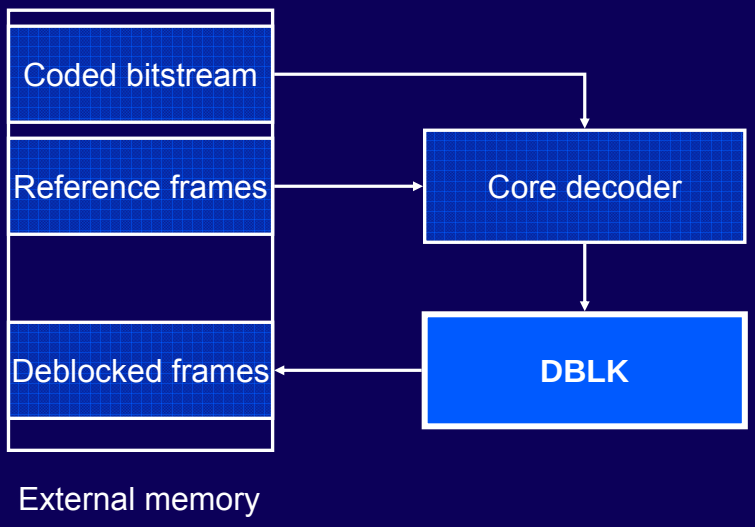
Multimedia Processor

PHILIPS

Processor Designer in a video deblocking unit

PHILIPS

# Multi standard video decoder IP





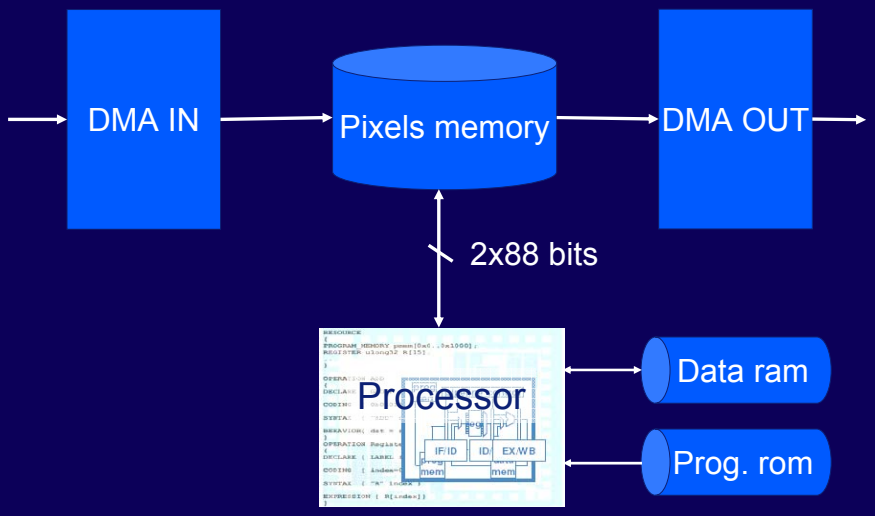

PHILIPS

# Why Processor Designer ?

- Until now : a RTL block for each standard. => Make a generic block for all (changing !) video standards.
- A programmable architecture brings flexibility (C compilation).
- 288 conditionals filters (4 and 8 taps) to be done in 600 cycles.
- High throughput needed : custom operations and special memory addressing scheme are required.

PHILIPS

# DBLK architecture

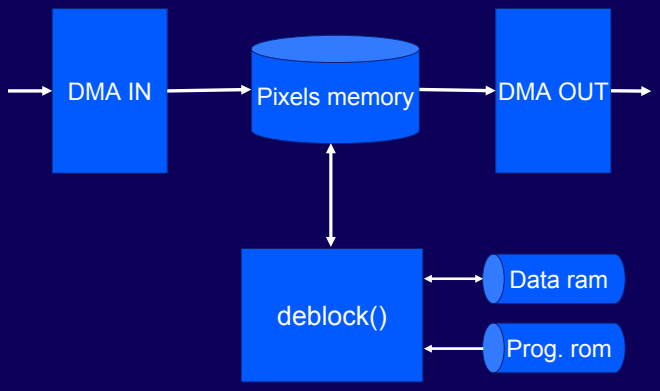





PHILIPS

# Step 1 : function call

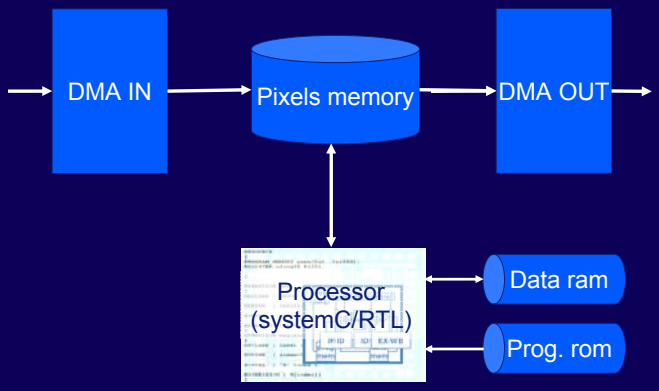
Application development :
- Get quickly a C model for the system
- Debug the application in a SystemC environment



PHILIPS

# Step 2 : integration of lt_risc_32p5

- Provided model of RISC used
- Compilation of application on the Lisa Processor
- Memories latency are modelled in the pipeline






PHILIPS

# Step 3 : RTL generation and performance improvement

- RTL generation
- C optimization
- Asm. optimization
- Use of specialized asm. instruction
- Remove unnecessary asm. Instructions
- Improve model for RTL generation (clock speed, area)

PHILIPS

# Results

- Architecture far from the initial RISC
- Target of 166 MHz easily reached
- Size comparable to a all RTL design (processor = 50 kgates)
- Performances reached
- IP taped out in a Set Top Box chip

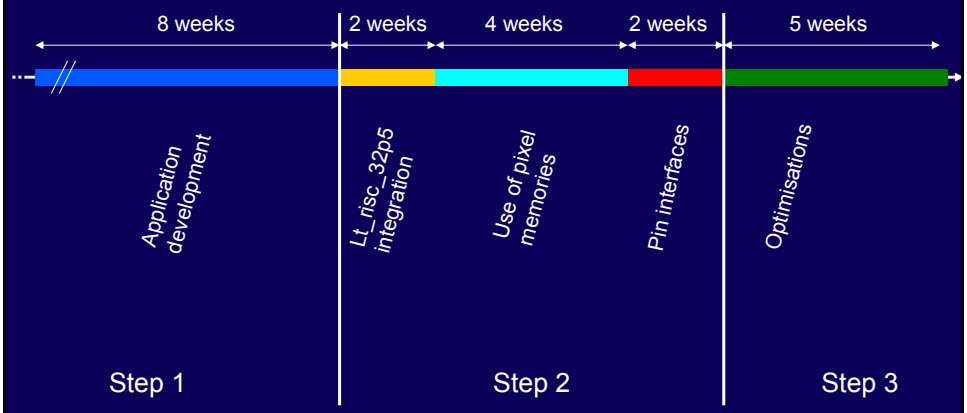
# Next steps

- No problem met yet on prototype
- Make the block more generic to handle others standards



PHILIPS

# Planning



PHILIPS

# Conclusion

**- Con**

- Long learning
- First use -> rough estimate of time needed

**+ Pro**

- RTL and SystemC always consistent (=> most of the validation can be run on SC)
- Faster than writing independent SC and RTL models
- Fast exploration of architecture choices
- Use of firmware :
  - can be generic
  - C debug
  - If program ram : fixes and feature changes can be downloaded
- No royalties



# Thank You