MPSoC '08

MPSoC Design Space Exploration Framework

Gerd Ascheid

RWTH Aachen University, Germany

Outline

- Motivation:
  MPSoC requirements in wireless and multimedia
- MPSoC design space exploration framework
- Summary

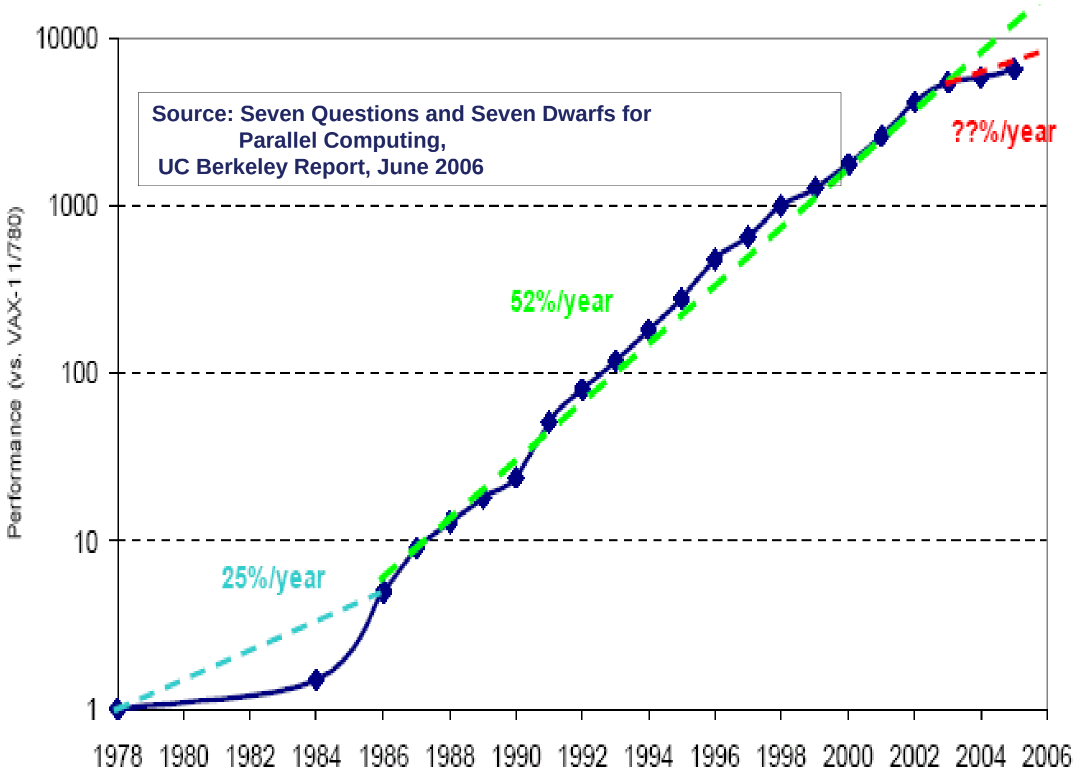
Parallel Computing in Mobiles

Massive parallelism required in the foreseeable future

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

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

GP - Processor Performance Improvement from 1978 to 2006



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

Embedded Applications Requirements

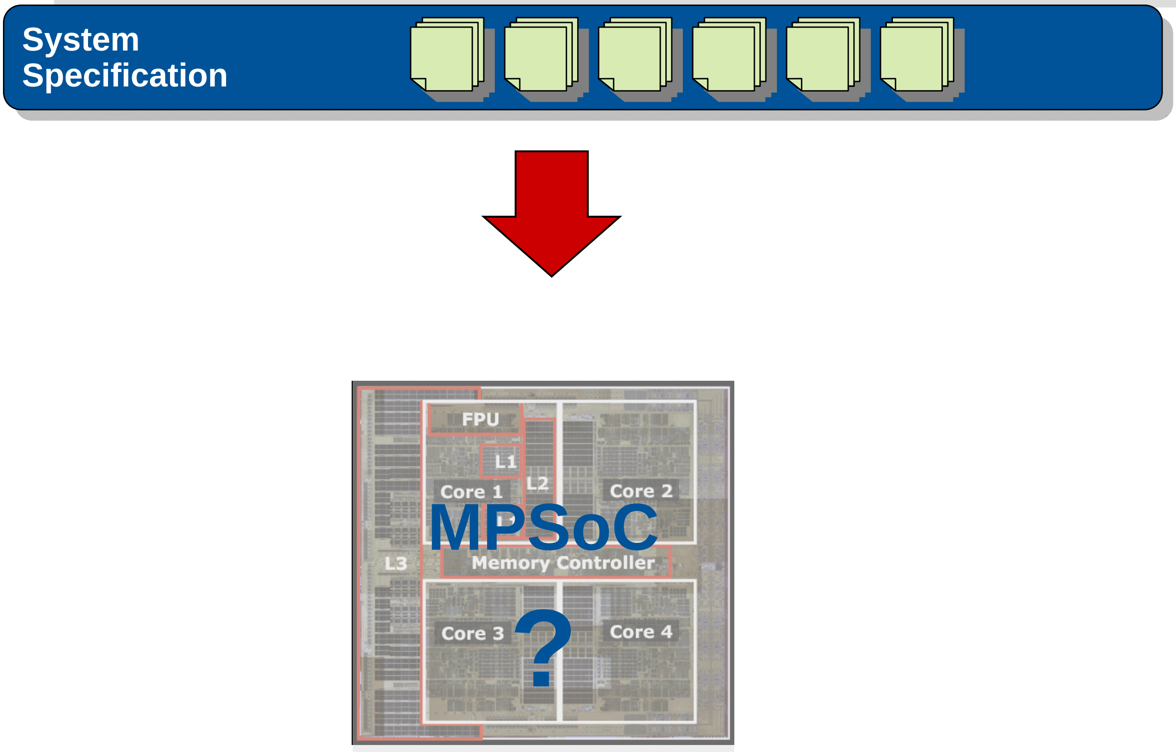
- **exponentially increasing performance**
  - feed demand for new features and value-added services
- **high flexibility**
  - complexity, multi-mode, multi-standard, time-to/in-market
- **power and energy efficiency**
  - for cost-sensitive and mobile consumer devices
- **heterogeneous processing requirements**

Heterogeneous Multi-Processor SoC (MPSoC) Platforms

Outline

- **Motivation:**
  MPSoC requirements in wireless and multimedia
- **MPSoC design space exploration framework**
- **Summary**

Multi-Processor System-on-Chip (MPSoC) Design Flow



System Specification

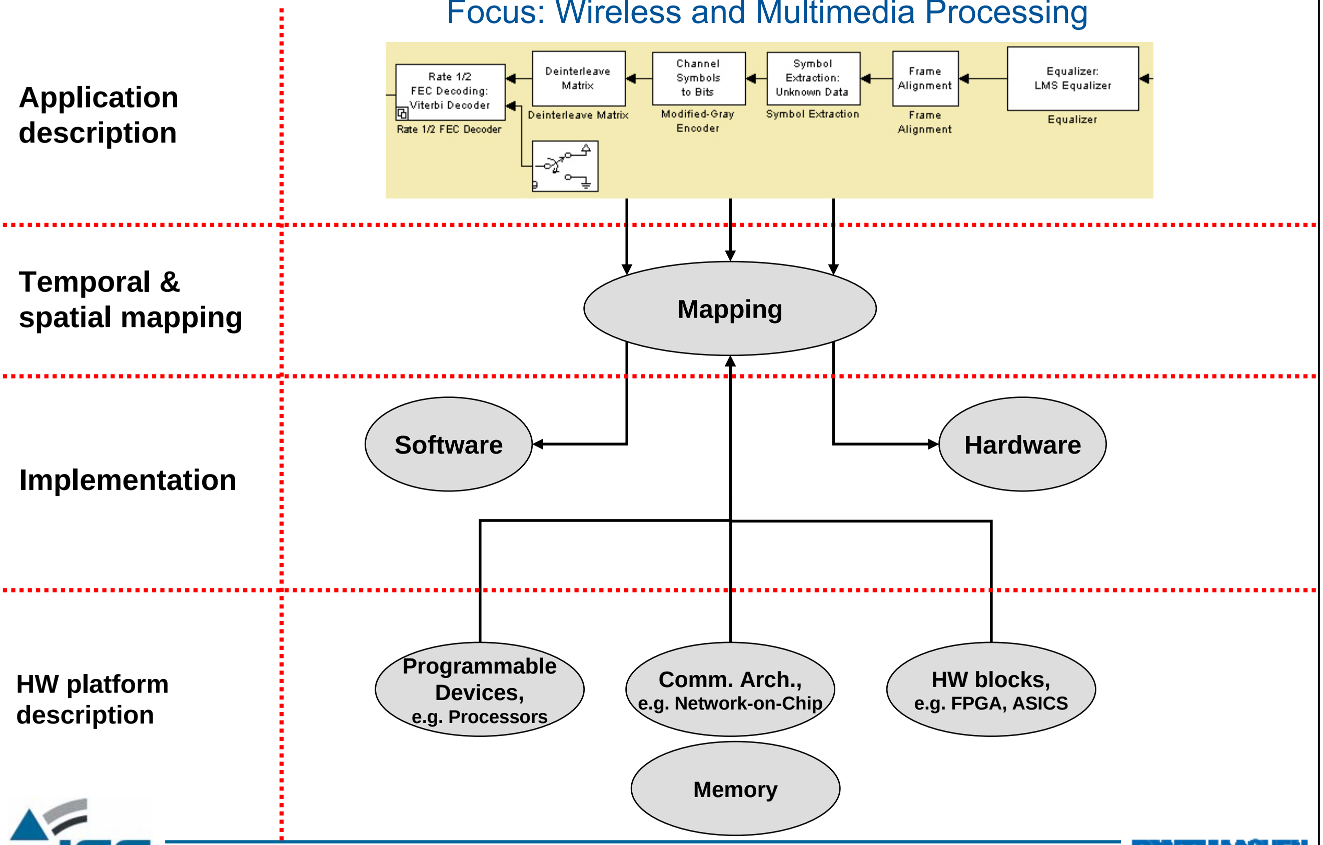
MPSoC

?

Layout: AMD Phenom

Constrained Optimization of Temporal and Spatial Mapping

Focus: Wireless and Multimedia Processing



Application description

Temporal & spatial mapping

Implementation

HW platform description

Mapping

Software

Hardware

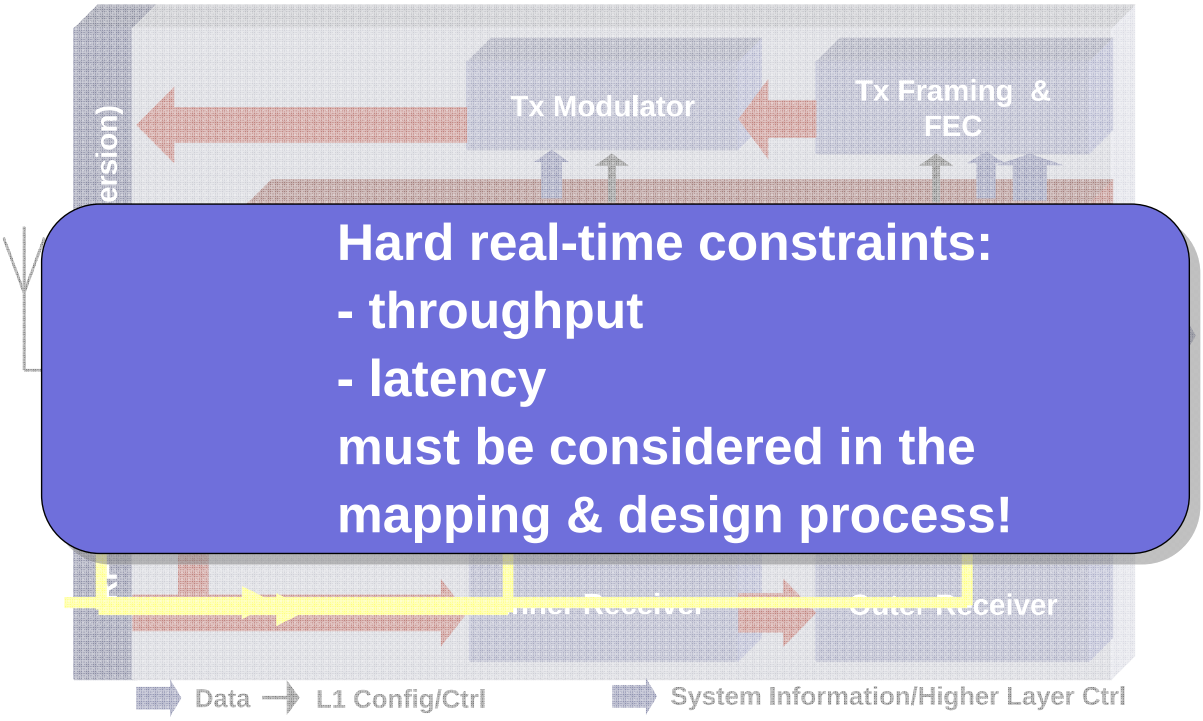
Programmable Devices, e.g. Processors

Comm. Arch., e.g. Network-on-Chip

HW blocks, e.g. FPGA, ASICS

Memory

Physical Layer Processing



Hard real-time constraints:
- throughput
- latency

must be considered in the mapping & design process!

Source: Dr. H. Dawid, Infineon

Mapping Issues

- **Design may start anywhere between**
  - a blank sheet of paper („from napkin to chip“)
  - a major redesign of an existing MPSoC platform
  - an improvement of an existing MPSoC platform
  - and the reuse of an existing MPSoC platform
- **To do and to evaluate a mapping in each design stage we need sufficiently precise characterization of processing and communication behavior to determine**
  - throughput
  - latency
  - critical paths
  - memory/buffering requirements

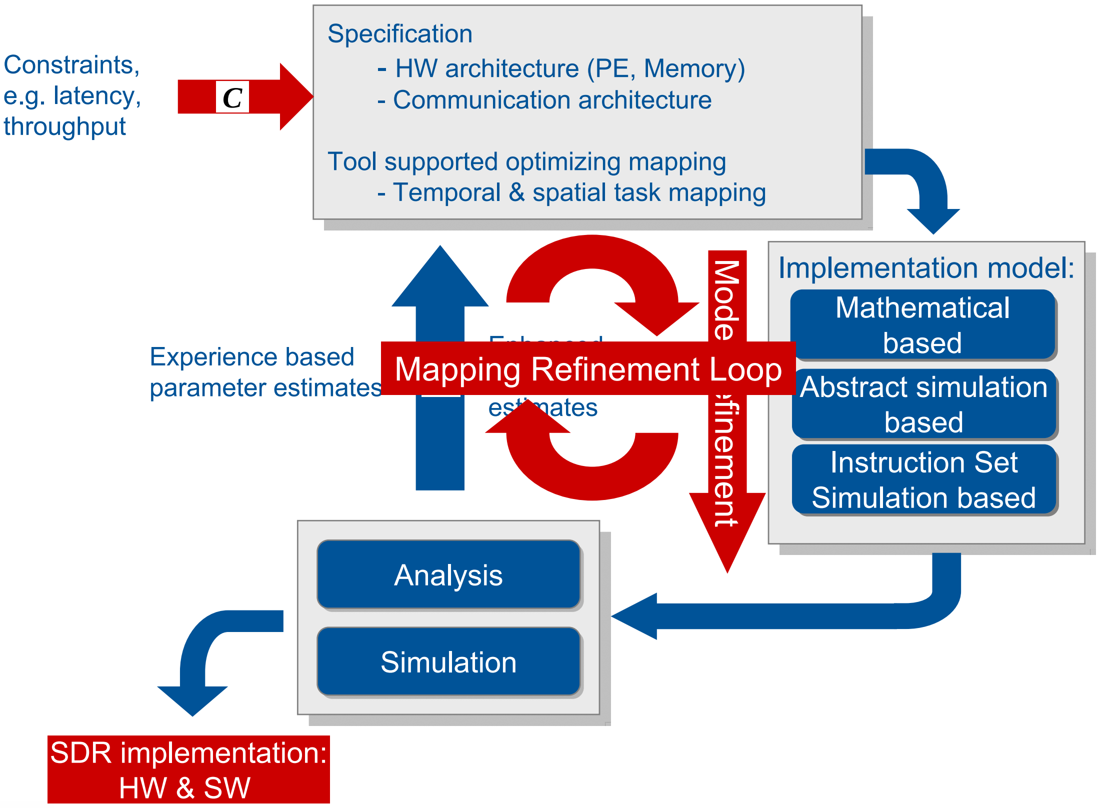
Performance Estimation

Performance data and/or estimates depend on characterization approach for processing and communication behaviour

- Coding style of software:
  from generic C model to assembly code optimized for architecture
- Modeling style for processing elements
- Communication and memory architecture
  ⇨ due to interaction of communication of parallel executed tasks actual performance can only be determined after mapping
  (⇨ simulation!?)

Suggested approach: Iterative mapping at each design stage

Iterative Mapping



Constraints, e.g. latency, throughput

C

Specification
- HW architecture (PE, Memory)
- Communication architecture

Tool supported optimizing mapping
- Temporal & spatial task mapping

Implementation model:

Mathematical based

Abstract simulation based

Instruction Set Simulation based

Experience based parameter estimates

Enhanced estimates

Mapping Refinement Loop

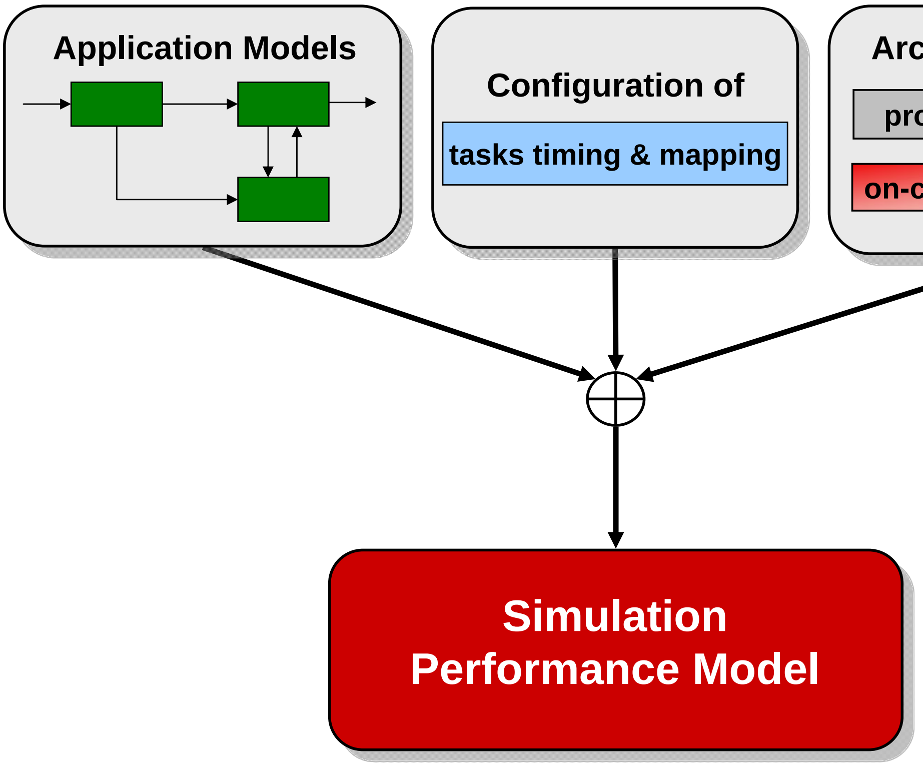
Mode finement

Analysis

Simulation

SDR implementation: HW & SW

Simulation Based on Virtual Architecture Mapping



Application Models

Configuration of

tasks timing & mapping

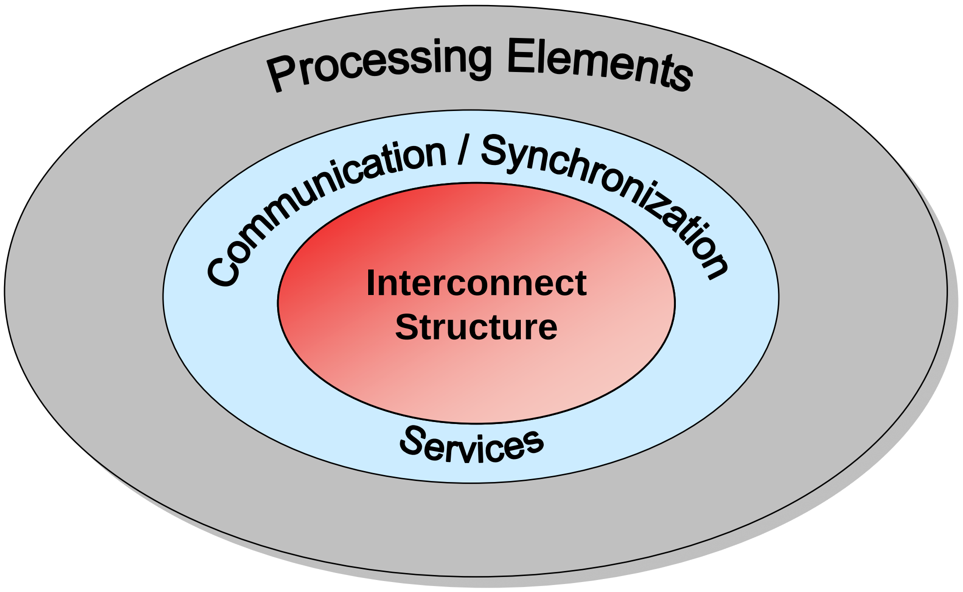
Architecture Models

processing elements

on-chip communication

Simulation
Performance Model

Orthogonalization of concerns



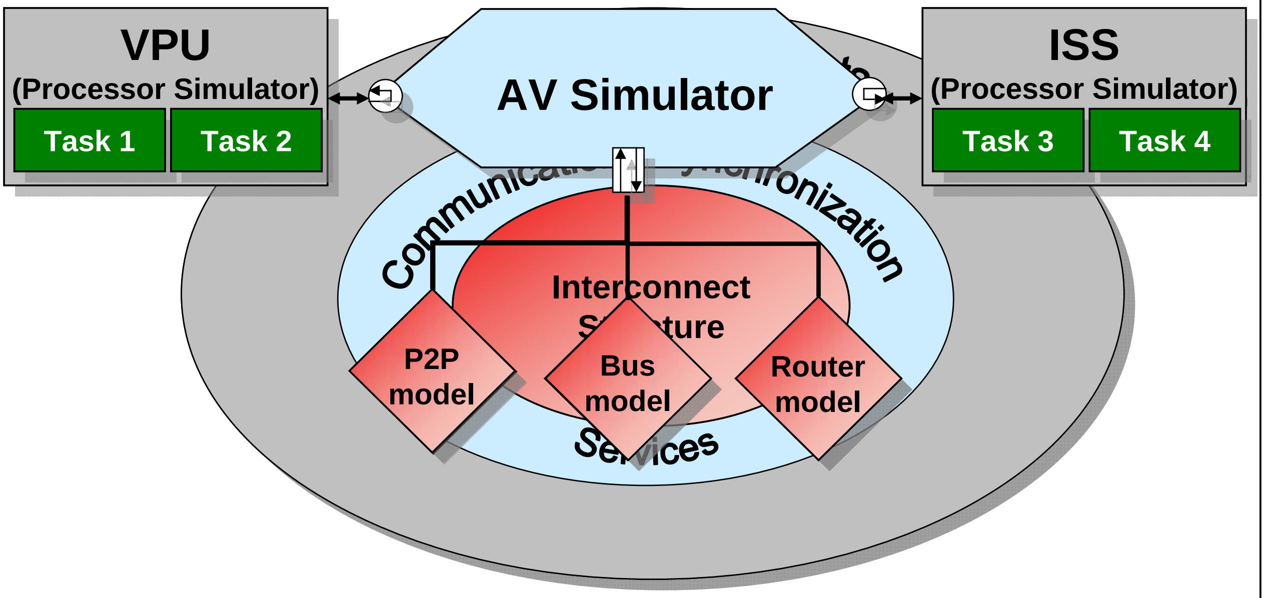
Processing Elements

Communication / Synchronization

Interconnect
Structure

Services

Orthogonalization of concerns



VPU

(Processor Simulator)

Task 1

Task 2

AV Simulator

ISS

(Processor Simulator)

Task 3

Task 4

Communication & Synchronization

Services

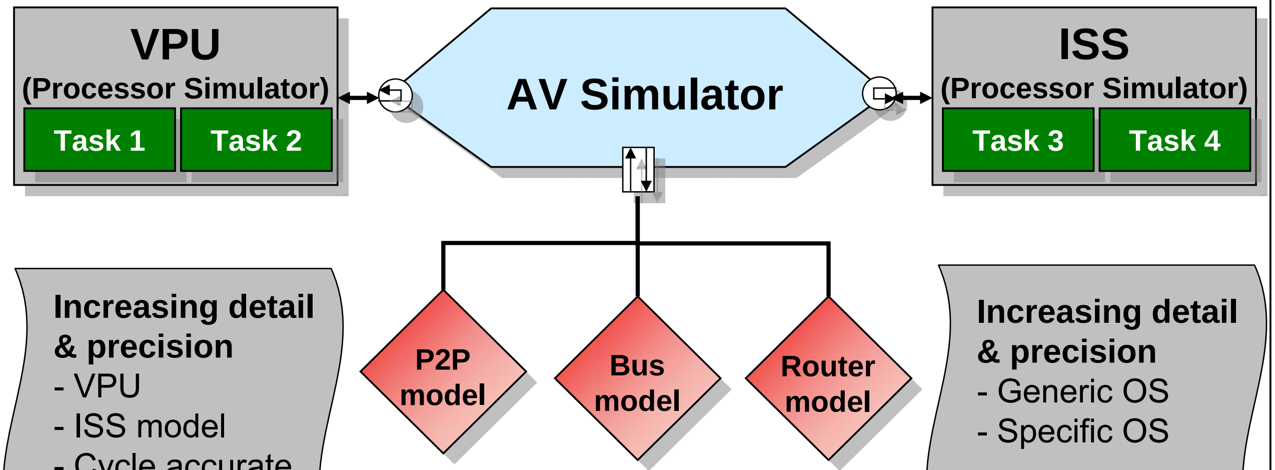
Interconnect Structure

P2P model

Bus model

Router model

Simulation Speed vs. Detail



VPU

(Processor Simulator)

Task 1

Task 2

AV Simulator

ISS

(Processor Simulator)

Task 3

Task 4

P2P model

Bus model

Router model

Increasing detail & precision
- VPU
- ISS model
- Cycle accurate model

Increasing detail & precision
- Transaction Level Model
- Cycle accurate model

Increasing detail & precision
- Generic OS
- Specific OS

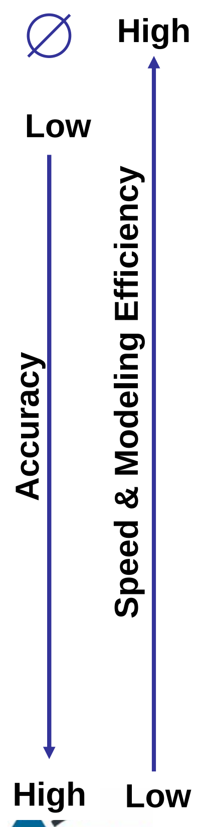
Design Stages

1. **„Napkin“**
   - Design evaluation (against criteria and constraints) based on
     - task graph, communication characteristic (deterministic)
     - initial „educated“ guesses for processing characteristics = timing, including timing uncertainty ranges, e.g. pdf
     - number of processing elements and interconnection
   - Temporal and spatial mapping

2. **Functional C-code**

3. **Hardware/software co-design**

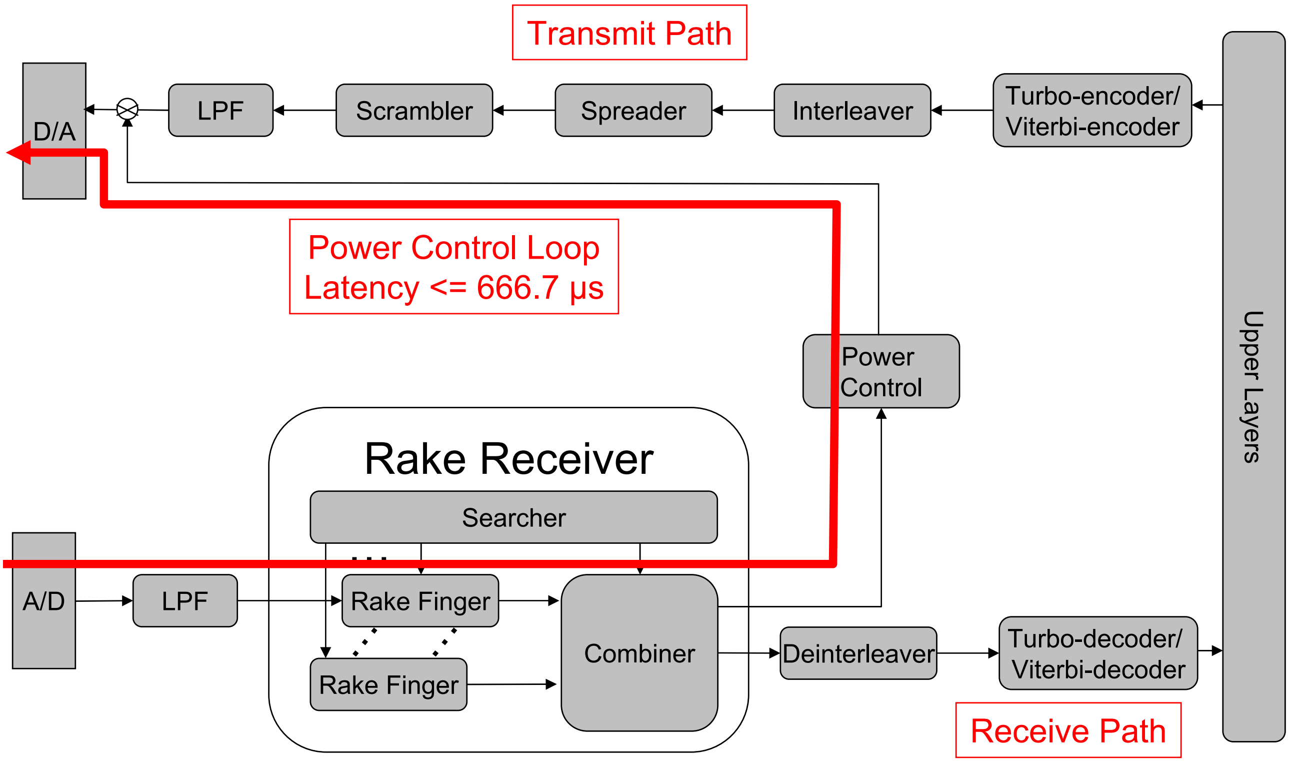
Software modeling abstraction levels



```
time = N(100,10);
```

Statistical analysis

Analytical Guidance: WCDMA Example



Transmit Path

D/A

LPF

Scrambler

Spreader

Interleaver

Turbo-encoder/ Viterbi-encoder

Upper Layers

Power Control Loop
Latency <= 666.7 µs

Power Control

Rake Receiver

Searcher

A/D

LPF

Rake Finger

Rake Finger

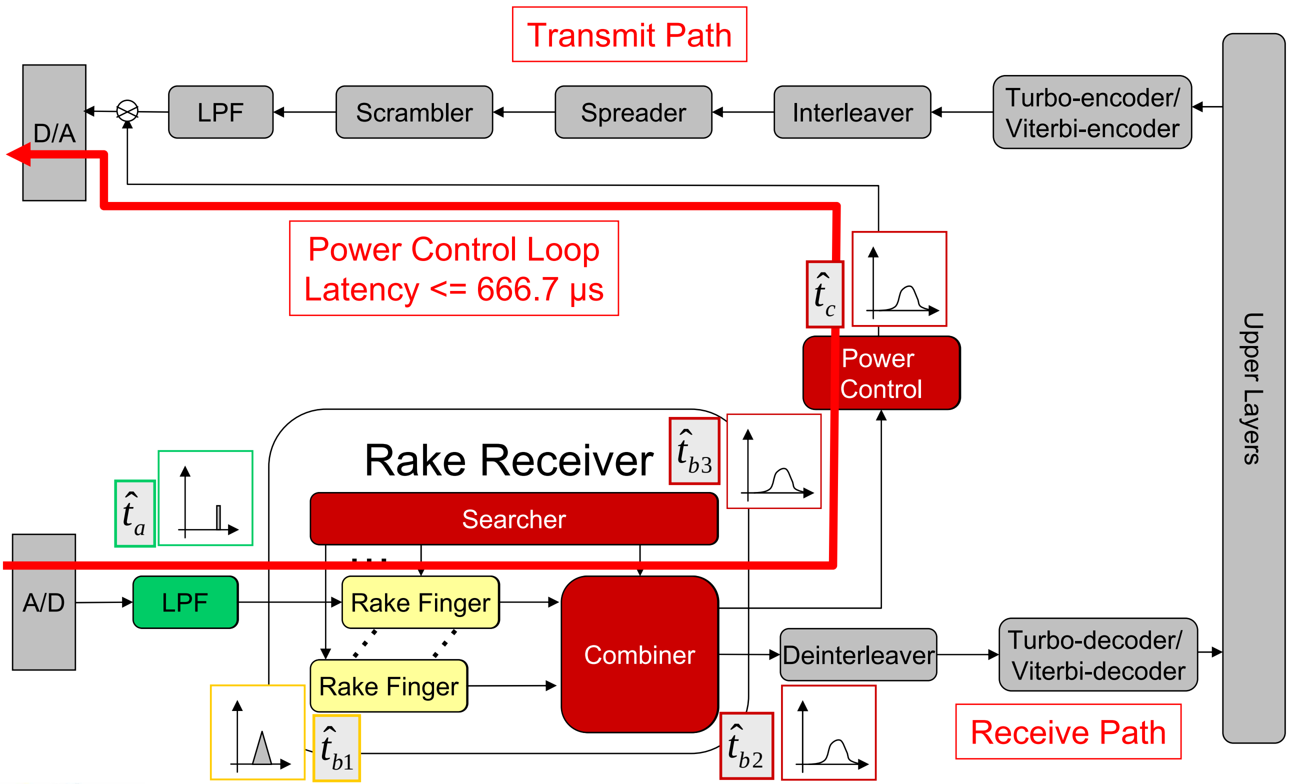
Combiner

Deinterleaver

Turbo-decoder/ Viterbi-decoder

Receive Path

Analytical Guidance: WCDMA Example



Transmit Path

D/A

LPF

Scrambler

Spreader

Interleaver

Turbo-encoder/ Viterbi-encoder

Upper Layers

Power Control Loop
Latency <= 666.7 µs

$\hat{t}_c$

Power Control

Rake Receiver

$\hat{t}_{b3}$

$\hat{t}_a$

Searcher

A/D

LPF

Rake Finger

Rake Finger

Combiner

Deinterleaver

Turbo-decoder/ Viterbi-decoder

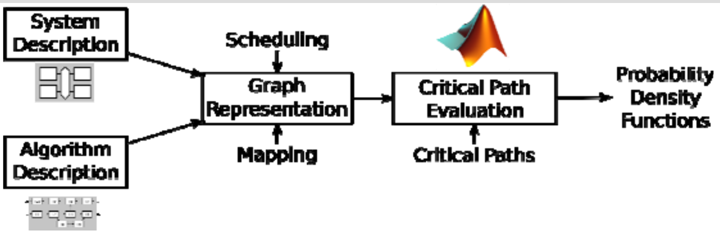
$\hat{t}_{b1}$

$\hat{t}_{b2}$

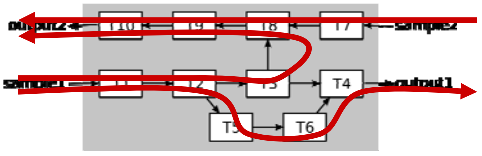
Receive Path

🟥 : algorithmic data 🟨 : implementation data 🟩 : known implementation reference

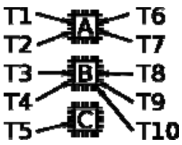
Analytical Guidance: General Tool Flow (I)



- **Task Graph:**



- **Spatial Mapping:**



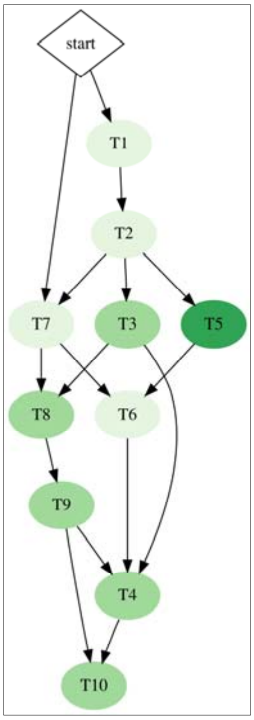
- **Temporal Mapping: (Schedules)**

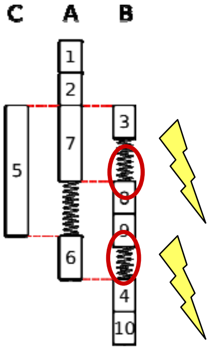
SC(A)=(T1,T2,T7,T6)
SC(B)=(T3,T8,T9,T4,T10)
SC(C)=(T5)

- **Need to find critical paths:**

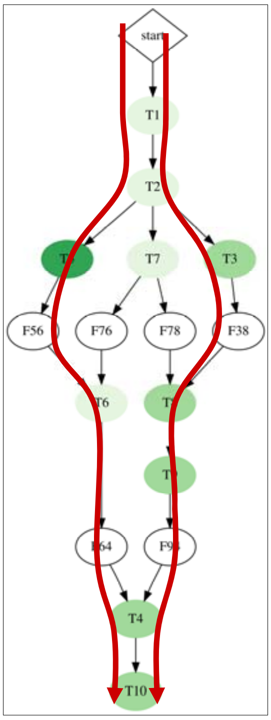
CP1=(T1,T2,T5,T6,T4)	CP4=SC(A)
CP2=(T7,T8,T9,T10)	CP5=SC(B)
CP3=(T1,T2,T3,T8,T9,T10)	CP6=SC(C)

Analytical Guidance: General Tool Flow (II)

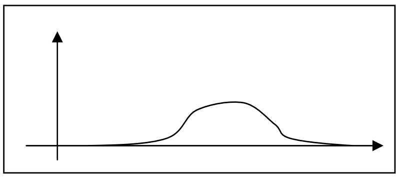


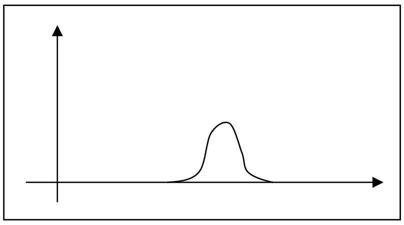


analyze data and control flow dependencies



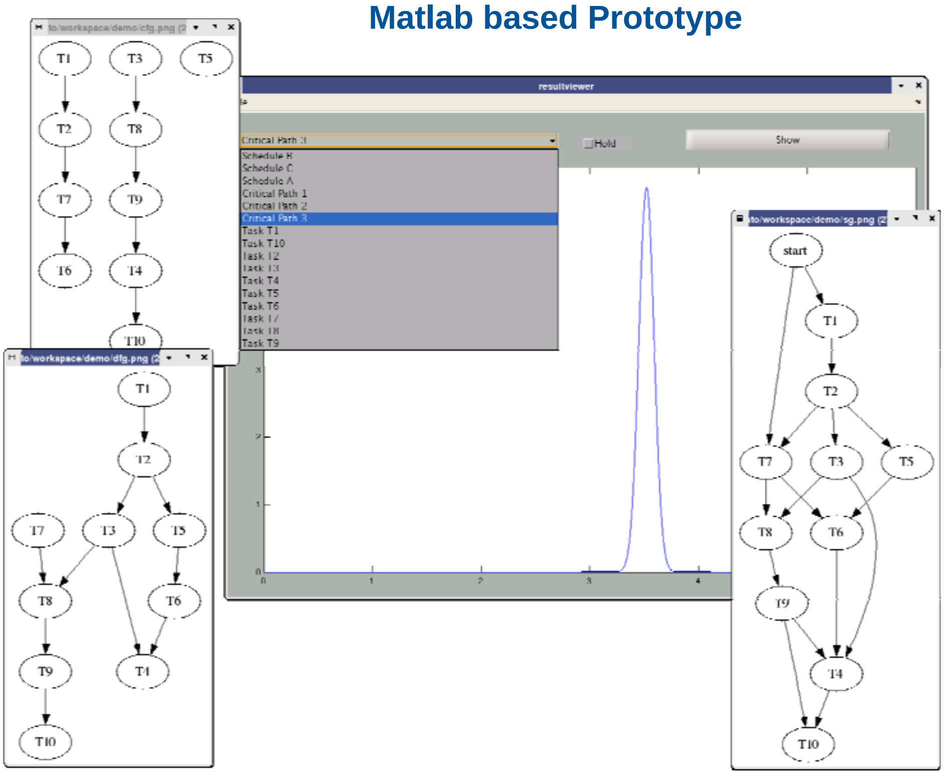
Probability density calculation for latency & throughput





Analytical Guidance: Tool Flow Realization

Matlab based Prototype

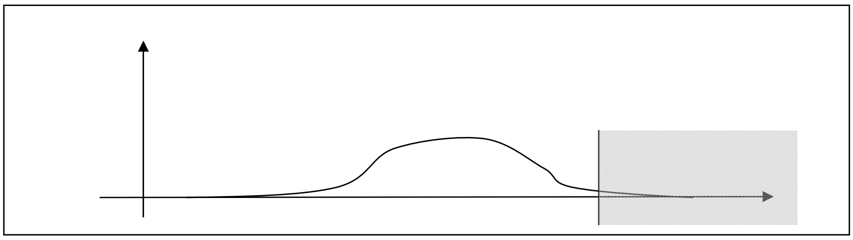


Analytical Guidance: Conclusions of Investigations (II)

Two extreme cases to illustrate occurrence of failure probability:

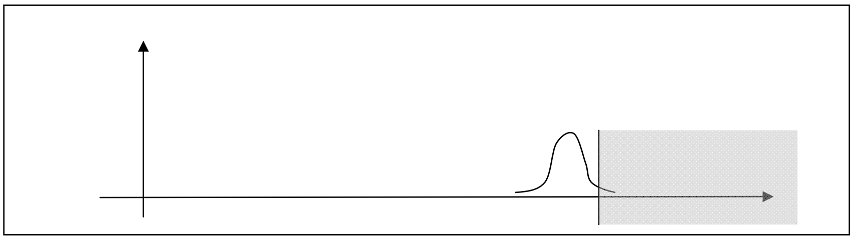
(1.) *Uncertainty* dominated

(Expected value $E(t_L)$ far from threshold, large standard deviation σ)



(2.) *Expected Value* dominated

(Expected value $E(t_L)$ near threshold, small standard deviation σ)

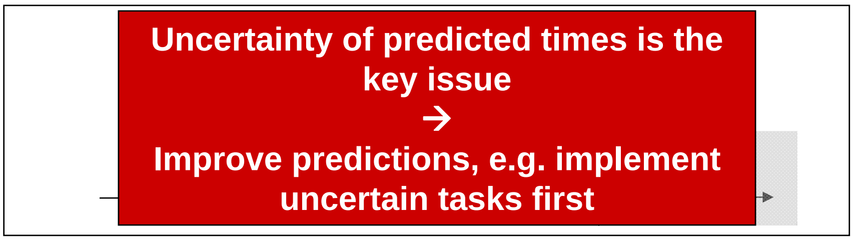


Analytical Guidance: Conclusions of Investigations (II)

Two extreme cases to illustrate occurrence of failure probability:

(1.) *Uncertainty* dominated

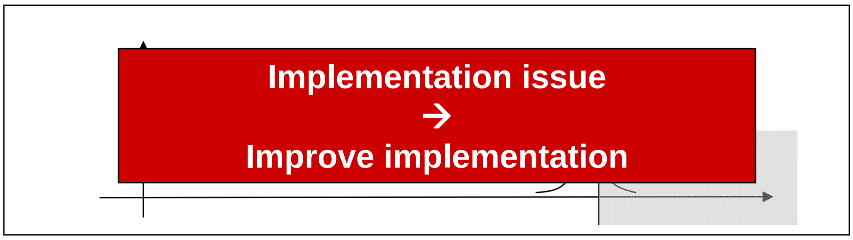
(Expected value $E(t_L)$ far from threshold, large standard deviation σ)



Uncertainty of predicted times is the key issue
→
Improve predictions, e.g. implement uncertain tasks first

(2.) *Expected Value* dominated

(Expected value $E(t_L)$ near threshold, small standard deviation σ)



Implementation issue
→
Improve implementation

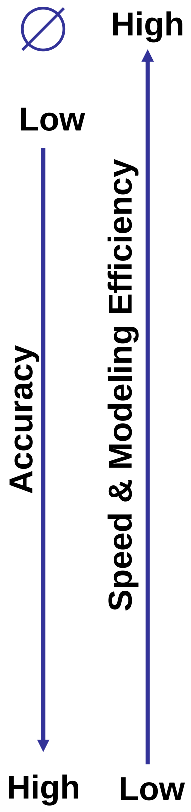
Design Stages

1. „Napkin“

2. **Functional C-code**
   - Performance evaluation (against criteria and constraints) based on
     - Functional C-code based execution timing estimates and/or experience based execution timing estimates
     - VPU model, generic OS
     - TLM-based communication architecture models (e.g. packet level)
   - Temporal and spatial mapping

3. Hardware/software co-design

Software modeling abstraction levels



```
time = N(100,10);
```

```
    …
    // functionality
    cycle_count += 100;
    consume(cycle_count);
    …
```

Statistical analysis

+ High simulation speed
- Low accuracy
- No functional verification

framework (VPU)

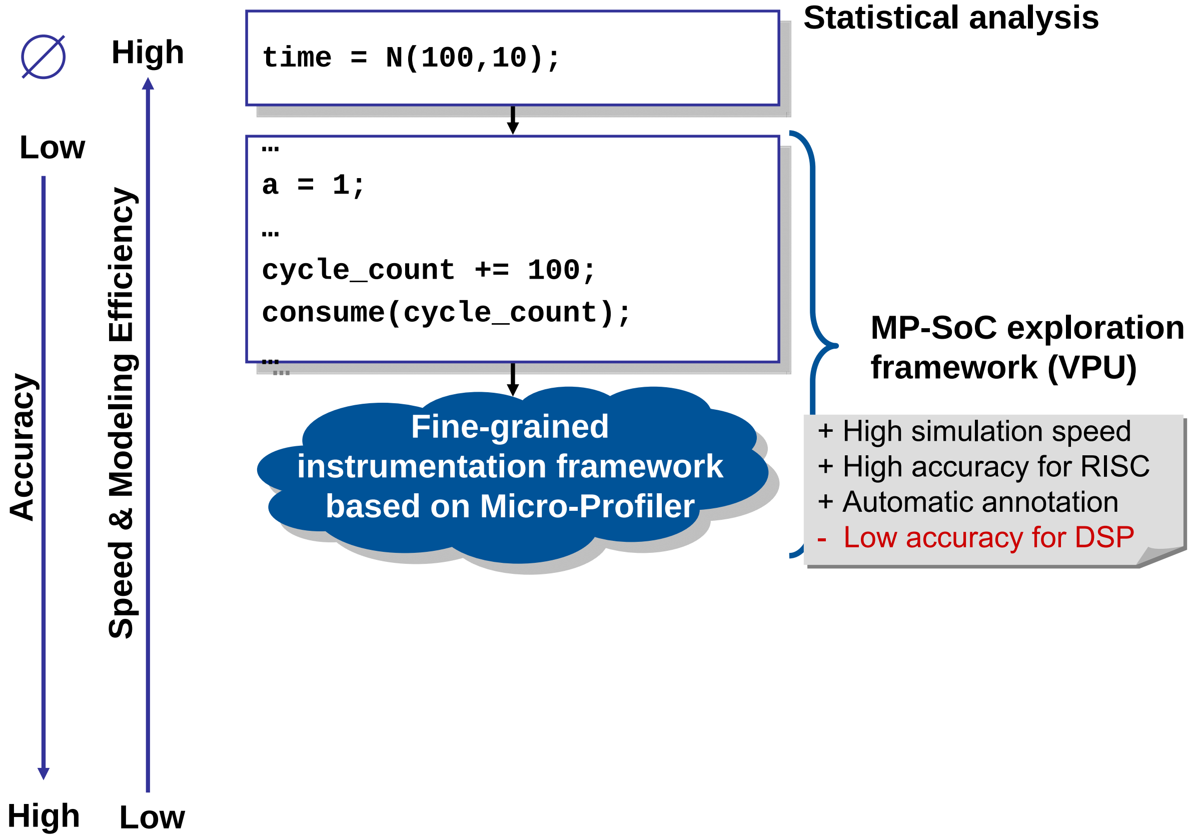
+ High simulation speed
+ Functional verification

Design Stages

1. „Napkin“
2. Functional C-code

3. **Hardware/software co-design**
   - from 3-address code (μ-profiler based)
     to optimized assembler code
     with communication made explicit
   - from VPU to
     to instruction set simulator (ISS)
     to cycle accurate model of actual processor
   - from generic OS
     to specific OS
   - from packet-level
     to cycle accurate TLM communication architecture model

Software modeling abstraction levels



Statistical analysis

```
time = N(100,10);
```

```
…
a = 1;
…
cycle_count += 100;
consume(cycle_count);
…
```

Fine-grained instrumentation framework based on Micro-Profiler

MP-SoC exploration framework (VPU)

- + High simulation speed
- + High accuracy for RISC
- + Automatic annotation
- - Low accuracy for DSP

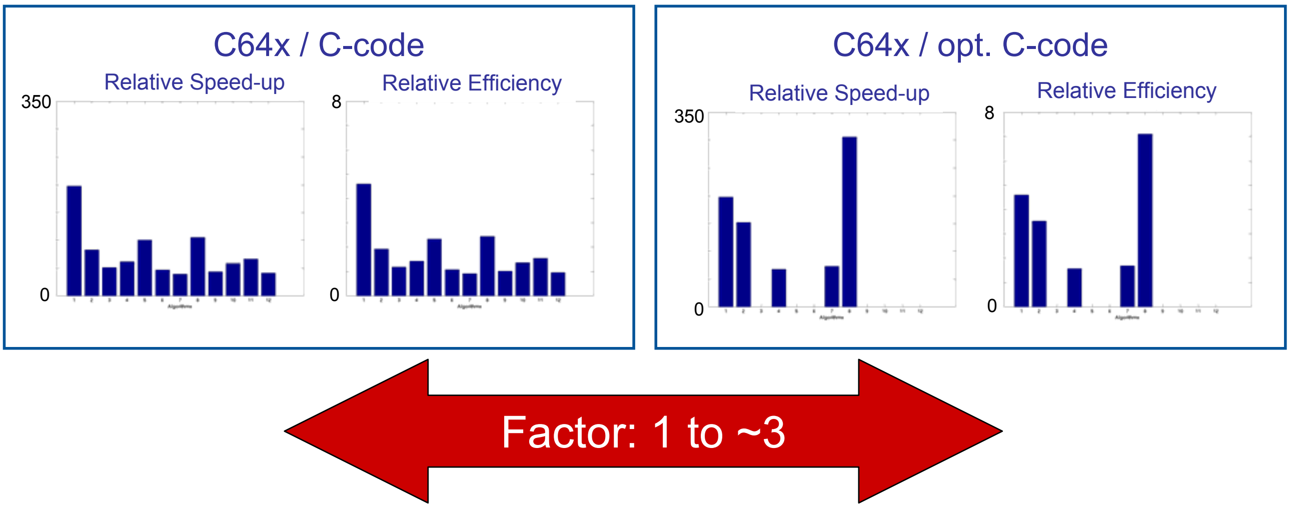
Mix of Estimation Methods

- **Why is it important to support seamless use of different timing estimation methods?**
- **Because**
  - of the imprecision of C-code based performance estimates
  - a designer‘s guess may be more precise than a functional C-code based estimate
  - there is efficient assembler code for key processing algorithms with known execution timing behavior

Measurement Results: optimized C-code investigations

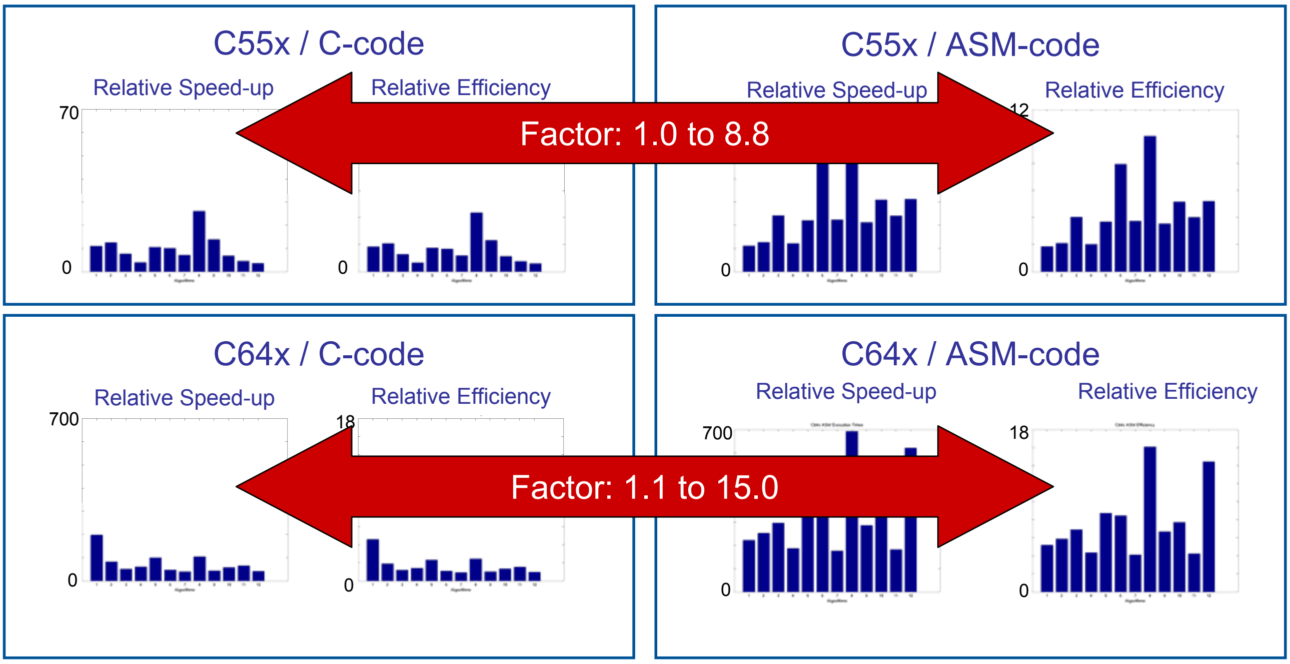
Algorithms:

1. Vector Addition
2. Vector Product
3. Vector Max Value
4. Vector Max Index
5. Vector Sum Square
6. Matrix Multiplication
7. Matrix Transpose
8. Autocorrelation
9. FIR filter (generic)
10. Complex FIR filter
11. Adaptive LMS FIR filter
12. FFT (Radix-2)



Note: Measurements are normalized to ARM720T / C-code implementation

Measurement Results: Assembly code investigations

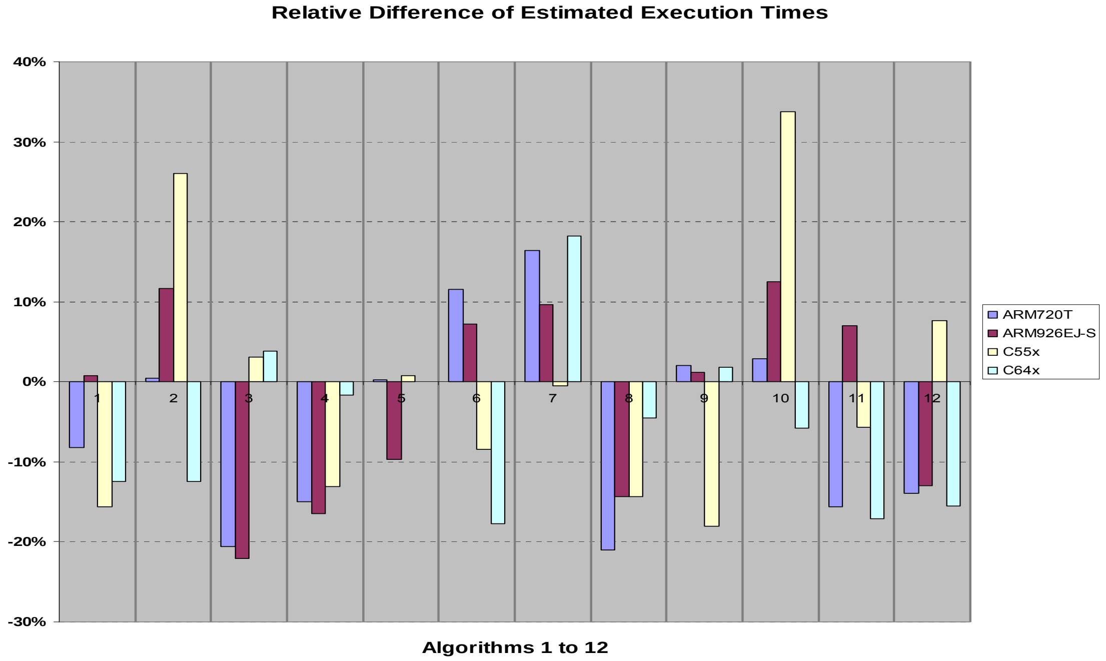


Algorithms:

1. Vector Addition
2. Vector Product
3. Vector Max Value
4. Vector Max Index
5. Vector Sum Square
6. Matrix Multiplication
7. Matrix Transpose
8. Autocorrelation
9. FIR filter (generic)
10. Complex FIR filter
11. Adaptive LMS FIR filter
12. FFT (Radix-2)

Note: Measurements are normalized to ARM720T / C-code implementation

Designer Estimate versus Cycle Accurate Simulation



Cycle annotation by designer: in C-code for Arm, in ASM for DSP

Algorithms:

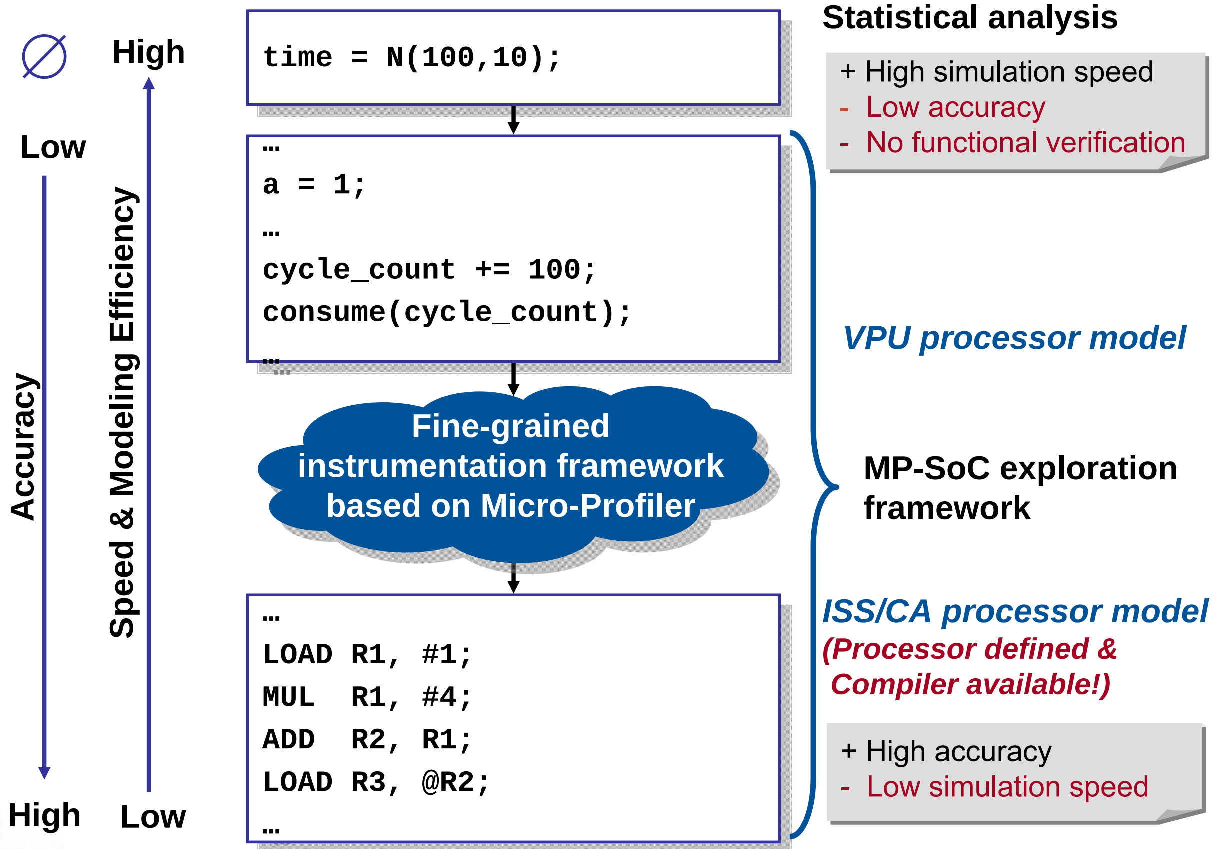
1. Vector Addition
2. Vector Product
3. Vector Max Value
4. Vector Max Index
5. Vector Sum Square
6. Matrix Multiplication
7. Matrix Transpose
8. Autocorrelation
9. FIR filter (generic)
10. Complex FIR filter
11. Adaptive LMS FIR filter
12. FFT (Radix-2)

Design Stages

1. „Napkin“
2. Functional C-code
3. Hardware/software co-design

4. **Final design**
   - Optimized C and assembler code
   - Cycle Accurate model of actual processor
   - Specific OS
   - Cycle accurate TLM communication architecture model

Software modeling abstraction levels



```
time = N(100,10);
```

```
…
a = 1;
…
cycle_count += 100;
consume(cycle_count);
…
```

Fine-grained instrumentation framework based on Micro-Profiler

```
…
LOAD R1, #1;
MUL  R1, #4;
ADD  R2, R1;
LOAD R3, @R2;
…
```

Statistical analysis

+ High simulation speed
- Low accuracy
- No functional verification

VPU processor model

MP-SoC exploration framework

ISS/CA processor model
(Processor defined & Compiler available!)

+ High accuracy
- Low simulation speed

Design Stages

1. „Napkin“
2. C-based simulation
3. Hardware/software co-design

4. **Final design verified:**

   Pass design to layout team and have a drink or two ...

Outline

- **Motivation:**
  MPSoC requirements in wireless and multimedia
- ➡ **MPSoC design space exploration framework**
- **Summary**

Summary & Outlook

- **Summary**
  - Analytically guided design space exploration and optimized design refinement
  - Seamless design flow from high level analysis to final implementation
  - Seamless mixing of different processing and communication characterization methods
- **Future work**
  - Define
    - performance requirement specification method for functional models and
    - feature description for processing elements and communication architectures

    to support tool based mapping
  - Mapping tool

Thank you for your attention !

Any questions?