


A Practical View on Baseband Processing Portability

T. Kempf,
E. M. Witte,
V. Ramakrishnan,
G. Ascheid

M. Adrat,
M. Antweiler

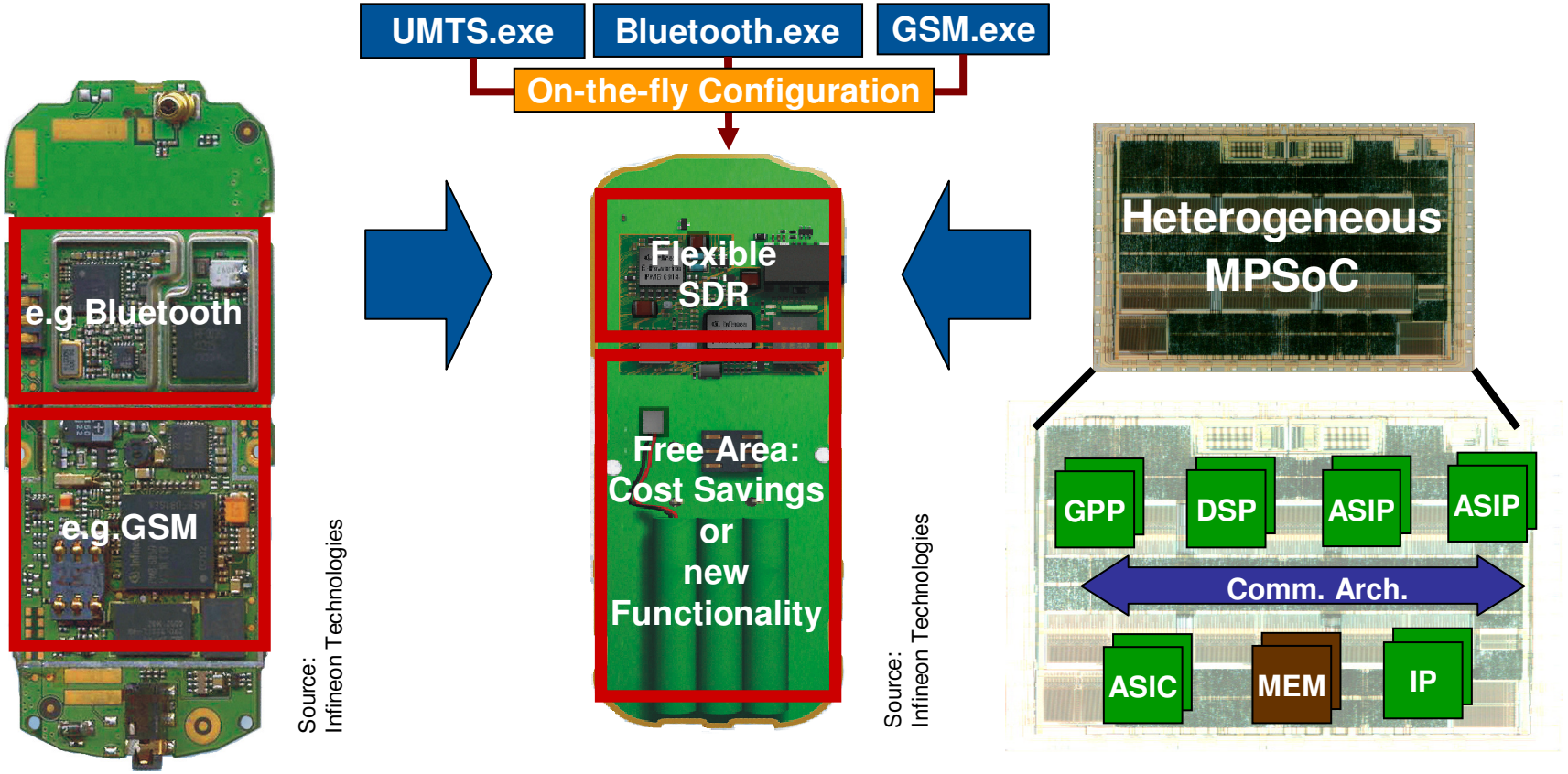
Institute for Integrated Signal Processing Systems

Agenda

- Introduction
- Portability versus Efficiency
- Case Study
  - Measurements
  - In-Depth Analysis: FFT kernel
- Conclusion/Outlook

Motivation

- **SDR Design**
- **Waveform Development**



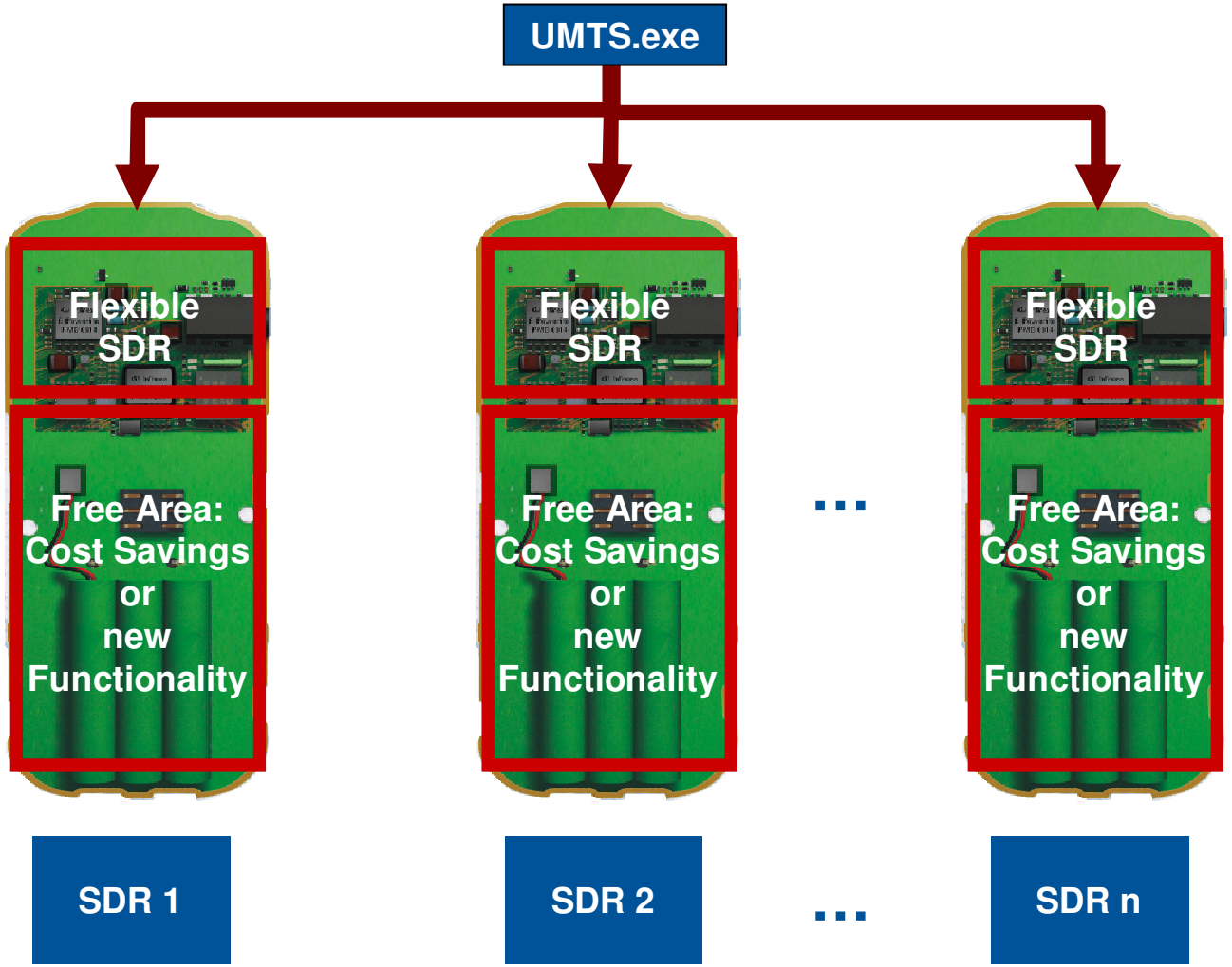
Today's Mobile Phone

Future SDR Mobile Phone

Heterogeneous Multi-Processor System-on-Chip

Motivation

- **SDR Design**
- **Waveform Development**



Agenda

- Introduction
- Portability versus Efficiency
- Case Study
  - Measurements
  - In-Depth Analysis: FFT kernel
- Conclusion/Outlook

Waveform Implementation Options

SDR Community View

"Portable" Design

- Pure Software Solutions
- High Level Language Implementations

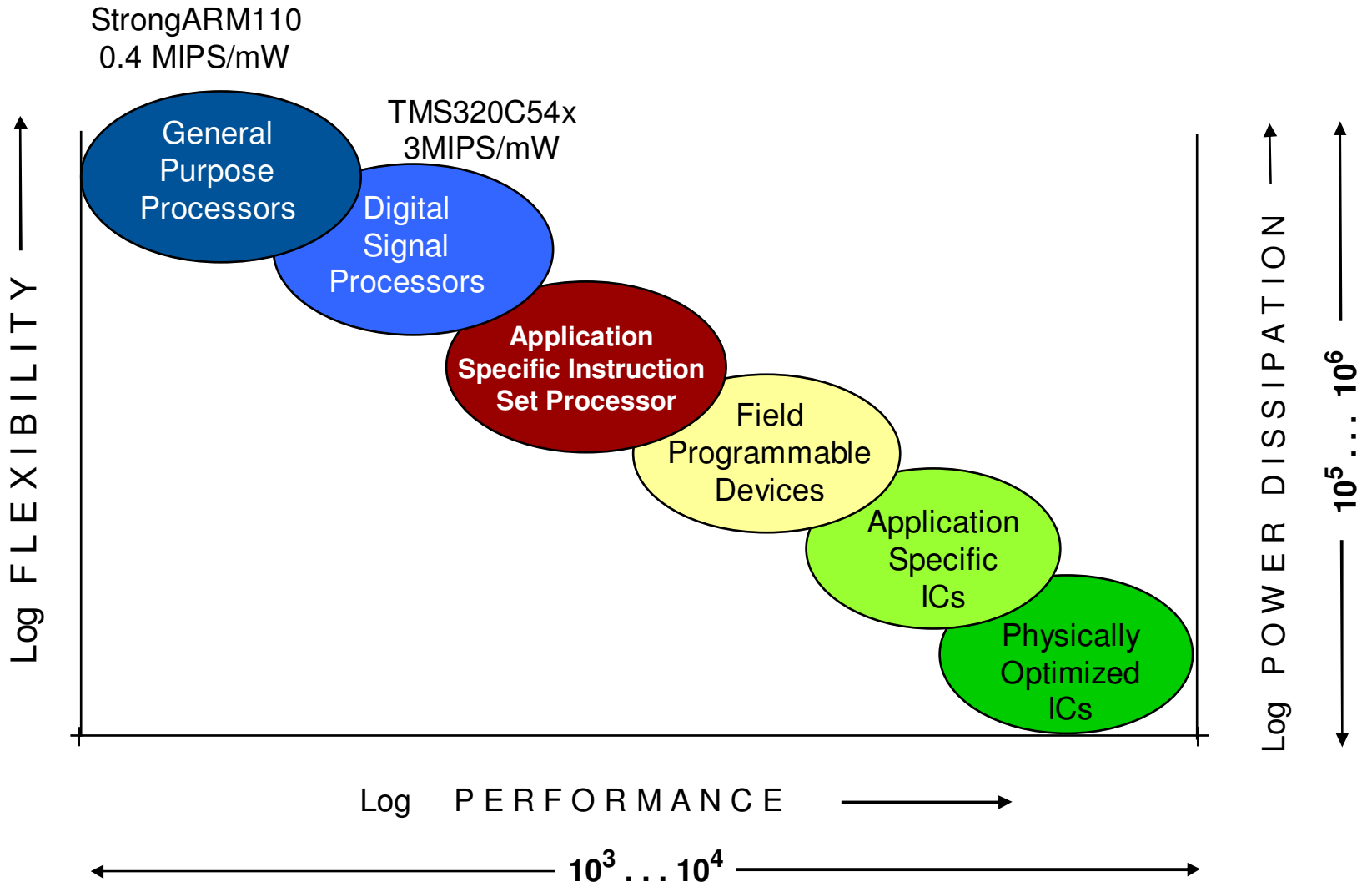
Hardware Designer's View

Maximum Efficiency

- Hardware Solution
- Add only minimum flexibility needed (e.g. ASIPs)

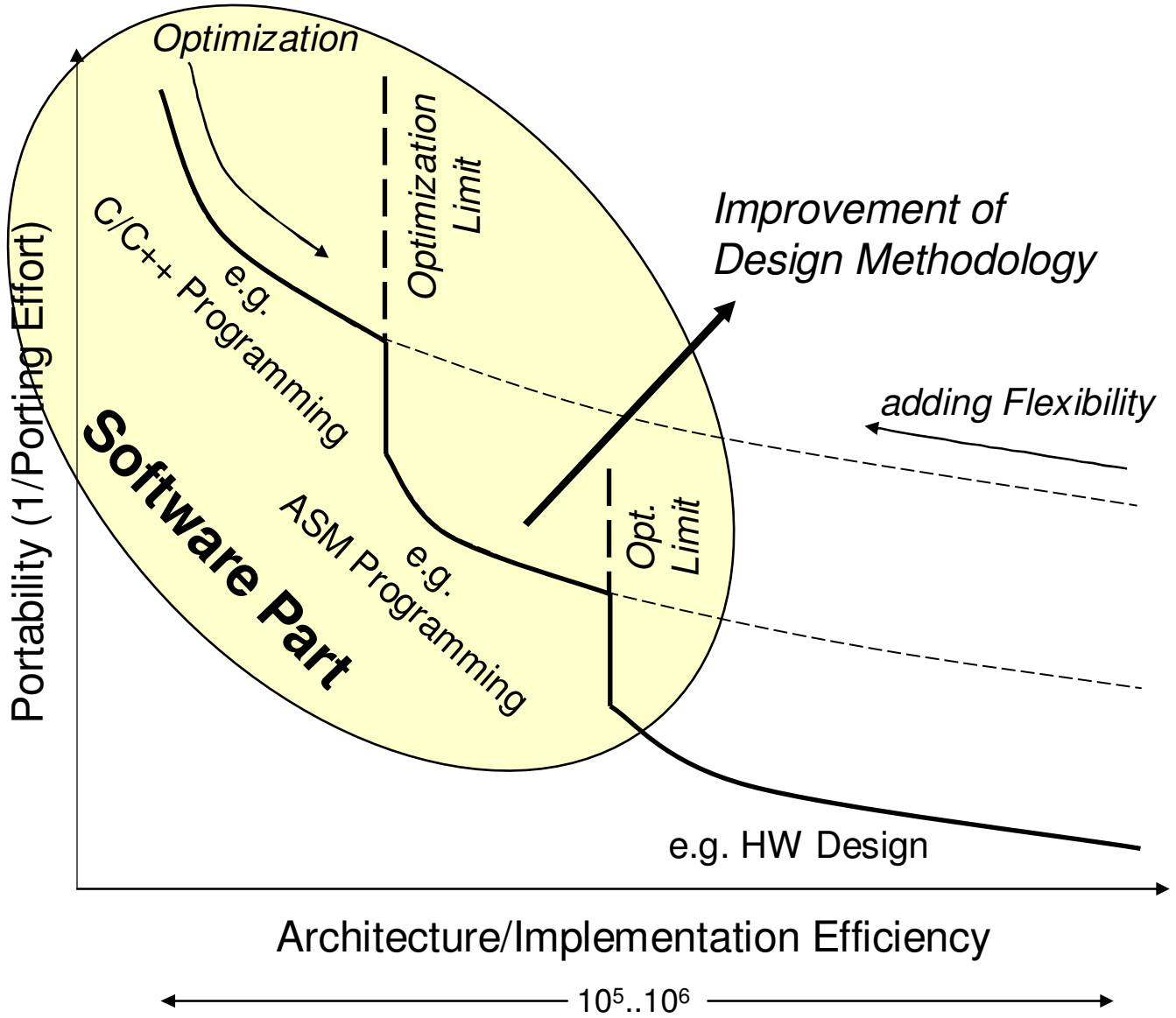
Cost Functions:

- Portability: e.g. Portability ~ 1 / Porting Effort
- Efficiency: e.g. Energy Efficiency ~ Bits / s / Watt



Source: T.Noll, RWTH Aachen University

The Efficiency vs. Portability Trade-Off



Agenda

- Introduction
- Portability versus Efficiency
- Case Study
  - Measurements
    - In-Depth Analysis: FFT kernel
- Conclusion/Outlook

Case Study: Portability vs. Efficiency

Algorithmic kernel:	X	**HW architecture:**	X	**Implementation Option:**

- *Vector Operations*, e.g. addition, product, etc.
- *Matrix operations*, e.g. transposition, etc.
- *Filter operations*, e.g. FIR, adaptive LMS filter, etc.
- *Correlation operations*, e.g. autocorrelation, etc.
- *FFT operations*, e.g. radix-2 FFT

- General Purpose Processors (GPPs):
  - *ARM720T*
  - *ARM926EJ-S*
- Digital Signal Processors (DSPs):
  - *TI C55x*
  - *TI C64x*

- *C-code*
- *Optimized C-code* (compiler directives)
- *Assembly code*

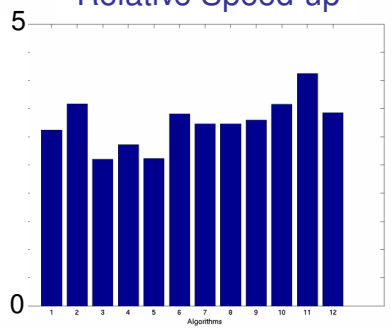
Measurement Results I: C-code implementations

Algorithms:

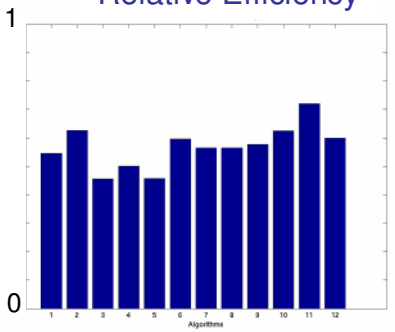
1. Vector Addition
2. Vector Product
3. Vector Max Value
4. Vector Max Index
5. Vector Sum Square
6. Matrix Multiplication
7. Matrix Transpose
8. Autocorrelation
9. FIR filter (generic)
10. Complex FIR filter
11. Adaptive LMS FIR filter
12. FFT (Radix-2)

ARM926EJ-S / C-code

Relative Speed-up

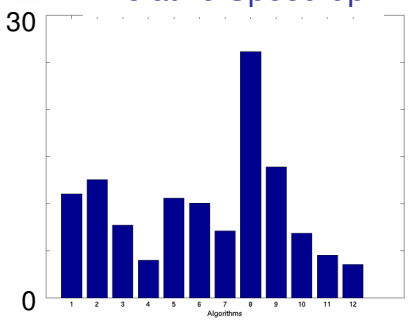


Relative Efficiency

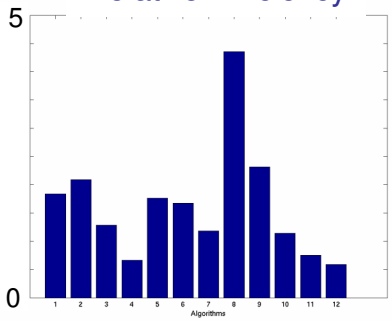


C55x / C-code

Relative Speed-up

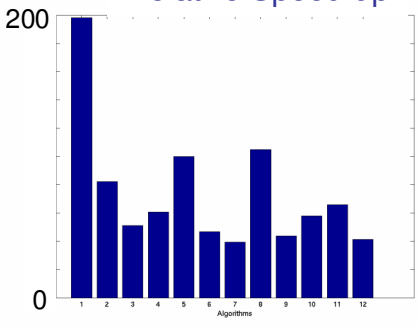


Relative Efficiency

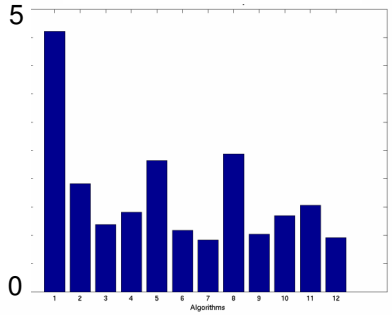


C64x / C-code

Relative Speed-up



Relative Efficiency



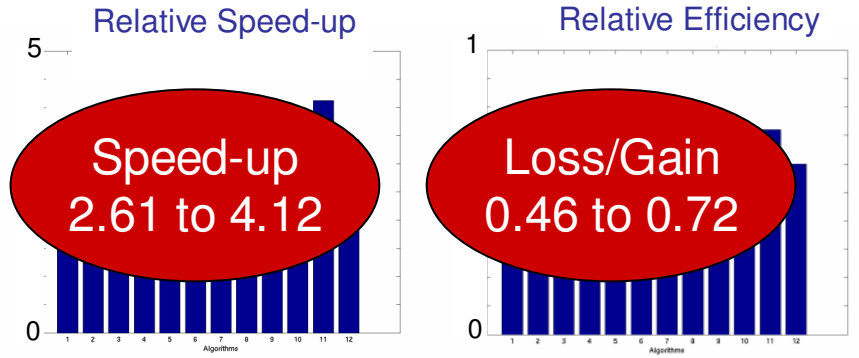
Note: Measurements are normed to ARM720T / C-code implementation

Measurement Results I: C-code implementations

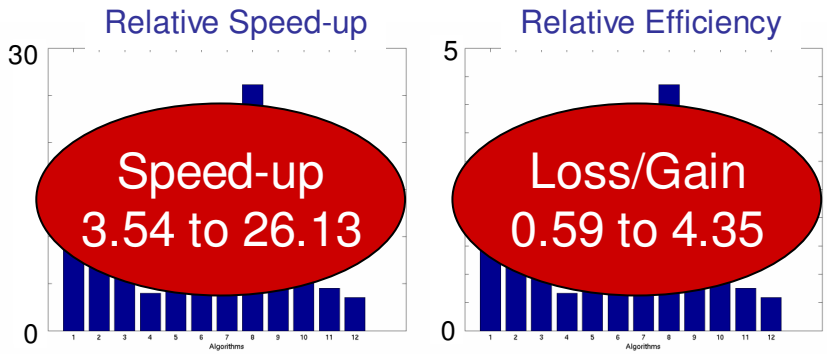
Algorithms:

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)

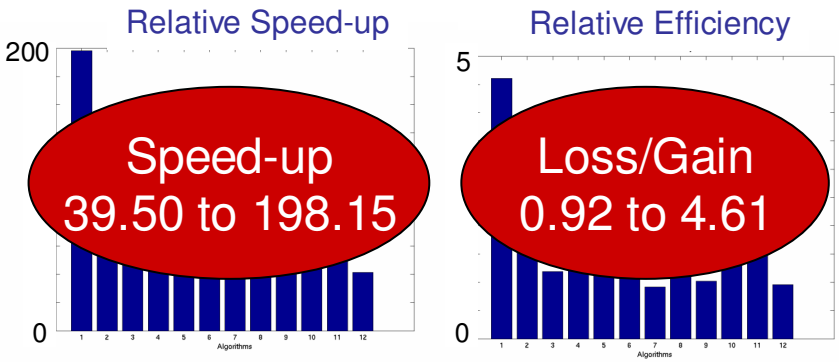
ARM926EJ-S / C-code



C55x / C-code



C64x / C-code



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

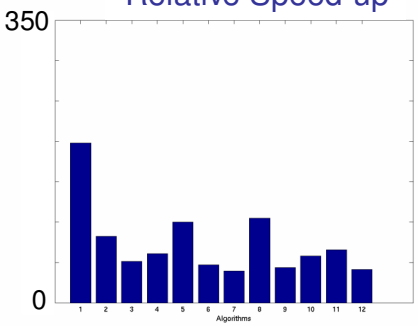
Measurement Results II: optimized C-code investigations

Algorithms:

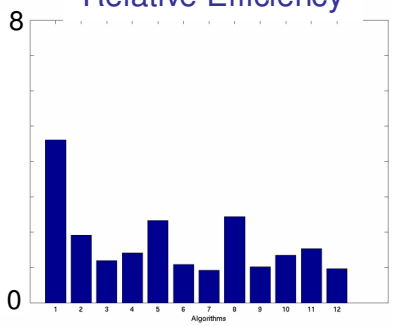
1.	Vector Addition	4.	Vector Max Index	7.	Matrix Transpose	10.	Complex FIR filter
2.	Vector Product	5.	Vector Sum Square	8.	Autocorrelation	11.	Adaptive LMS FIR filter
3.	Vector Max Value	6.	Matrix Multiplication	9.	FIR filter (generic)	12.	FFT (Radix-2)

C64x / C-code

Relative Speed-up

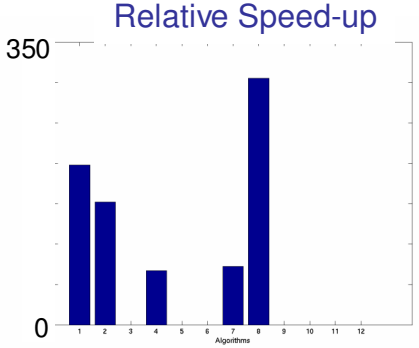


Relative Efficiency

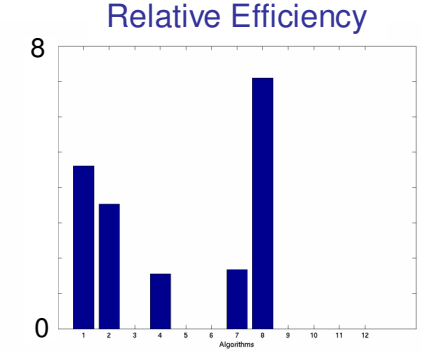


C64x / opt. C-code

Relative Speed-up



Relative Efficiency



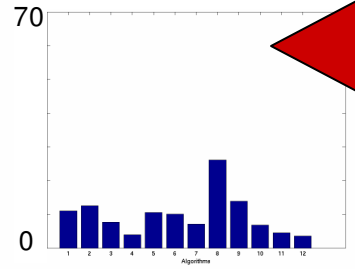
Factor: 1 to ~3

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

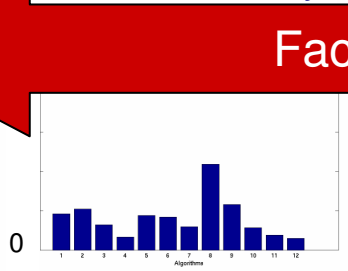
Measurement Results III: Assembly code investigations

C55x / C-code

Relative Speed-up

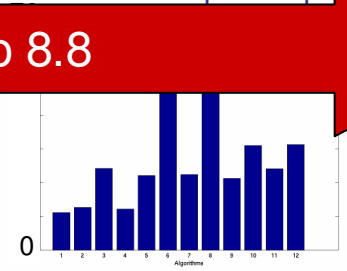


Relative Efficiency

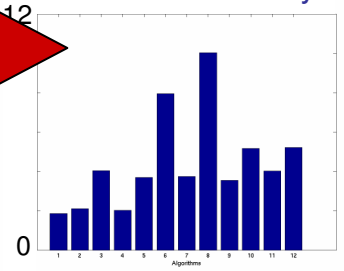


C55x / ASM-code

Relative Speed-up



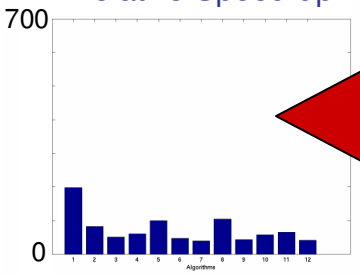
Relative Efficiency



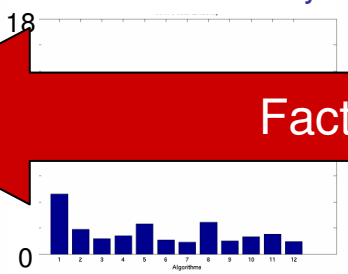
Factor: 1.0 to 8.8

C64x / C-code

Relative Speed-up

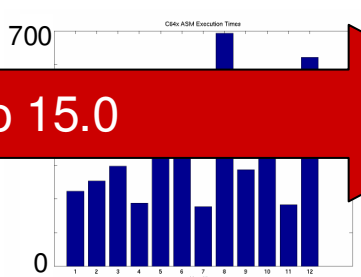


Relative Efficiency

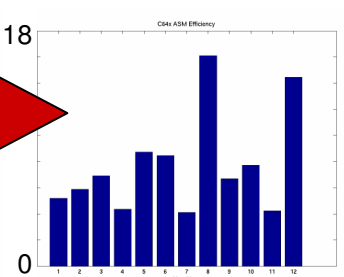


C64x / ASM-code

Relative Speed-up



Relative Efficiency



Factor: 1.1 to 15.0

Algorithms:

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

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

Summary: Measurement Results

1. **Hugh performance range exists from a "C-code on a DSP" to a "hand-optimized Assembly code on a DSP".**

2. **Minor C-code optimizations on basis of compiler directives can improve code generation by the compiler.**

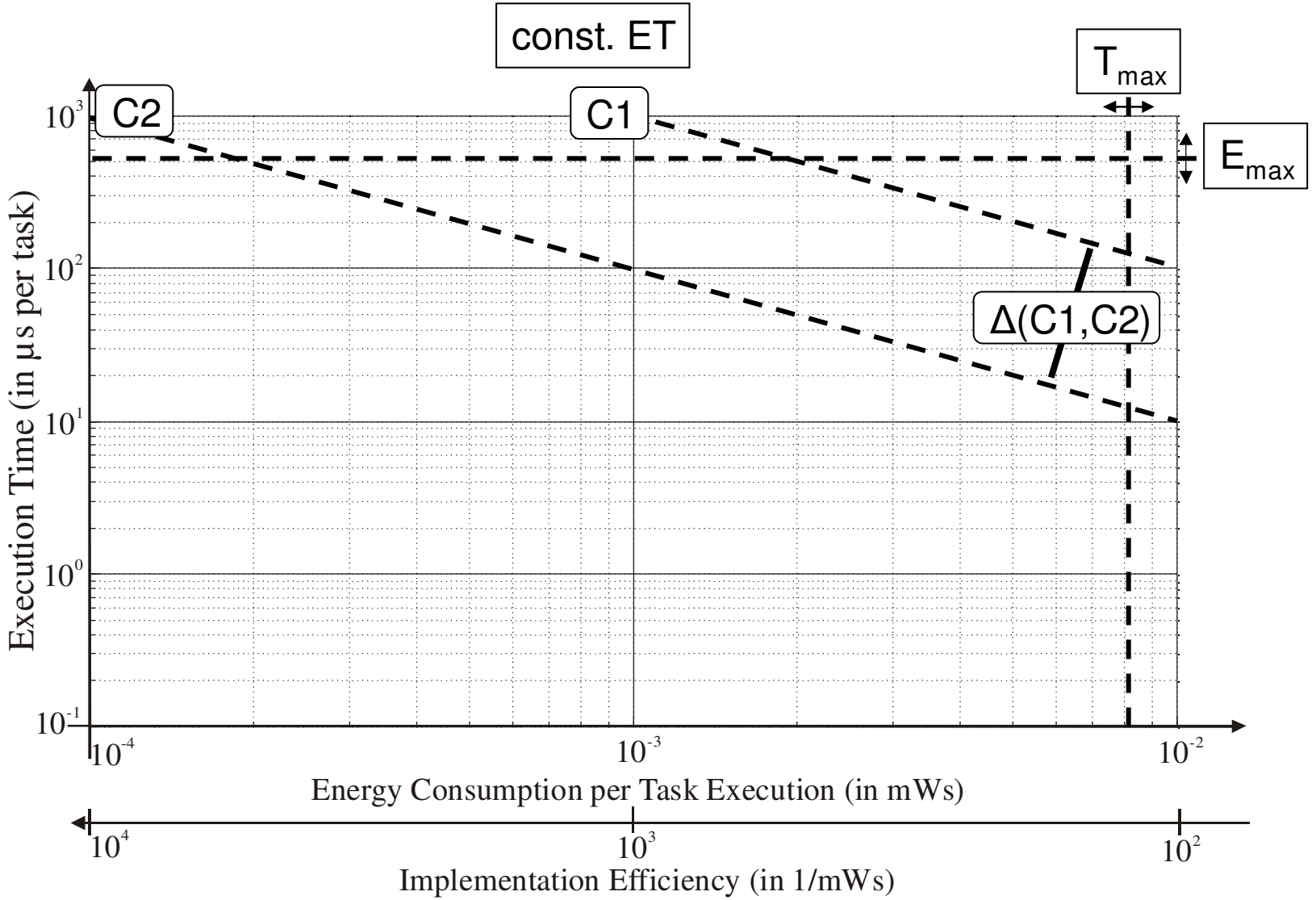
3. **Typical assumption that: "Assembly programming can be neglected" can not be supported at least not for baseband processing.**

Huge difference in execution time and efficiency forces developers to perform an in depth-analysis of this issue

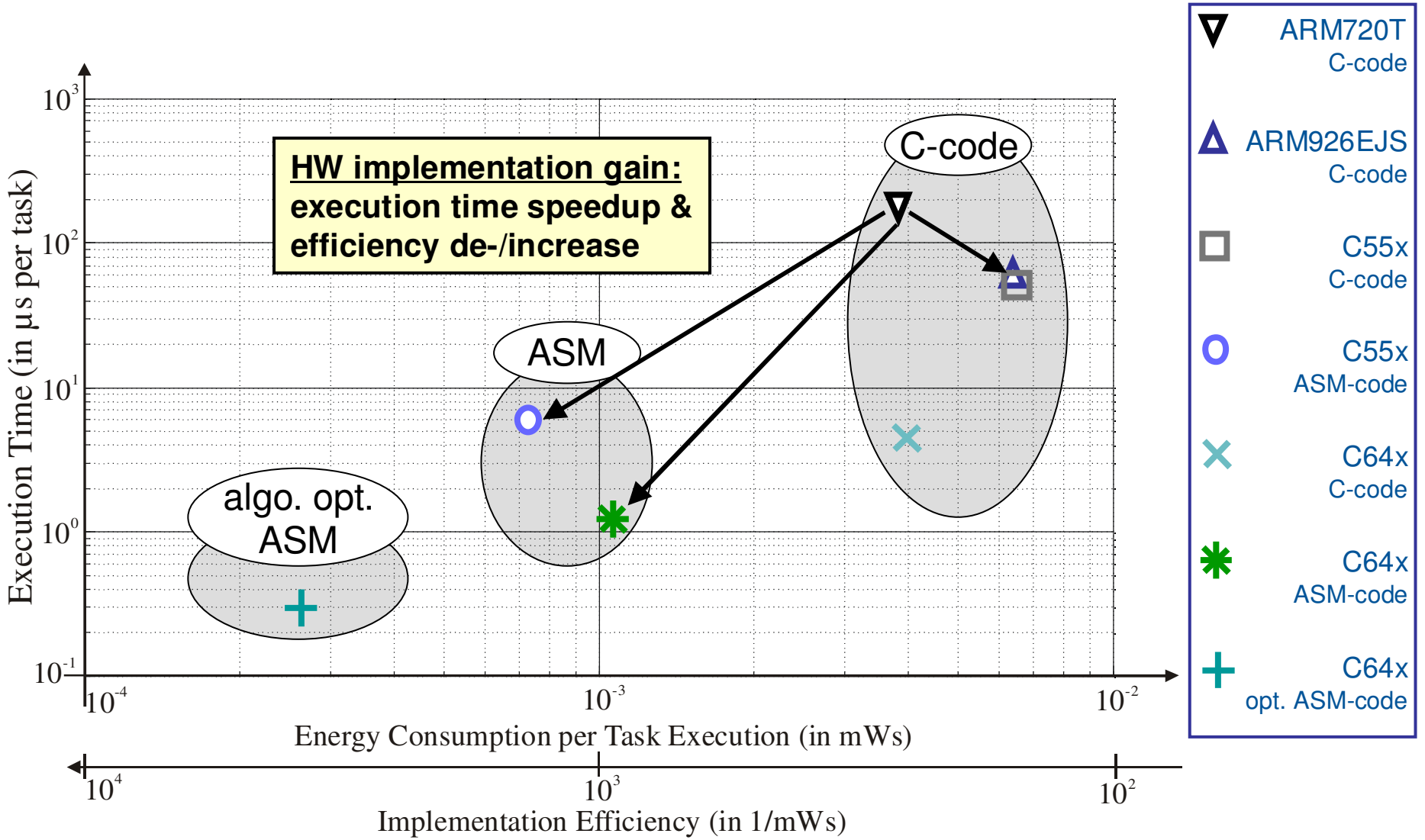
Agenda

- Introduction
- Portability versus Efficiency
- Case Study
  - Measurements
  - In-Depth Analysis: FFT kernel
- Conclusion/Outlook

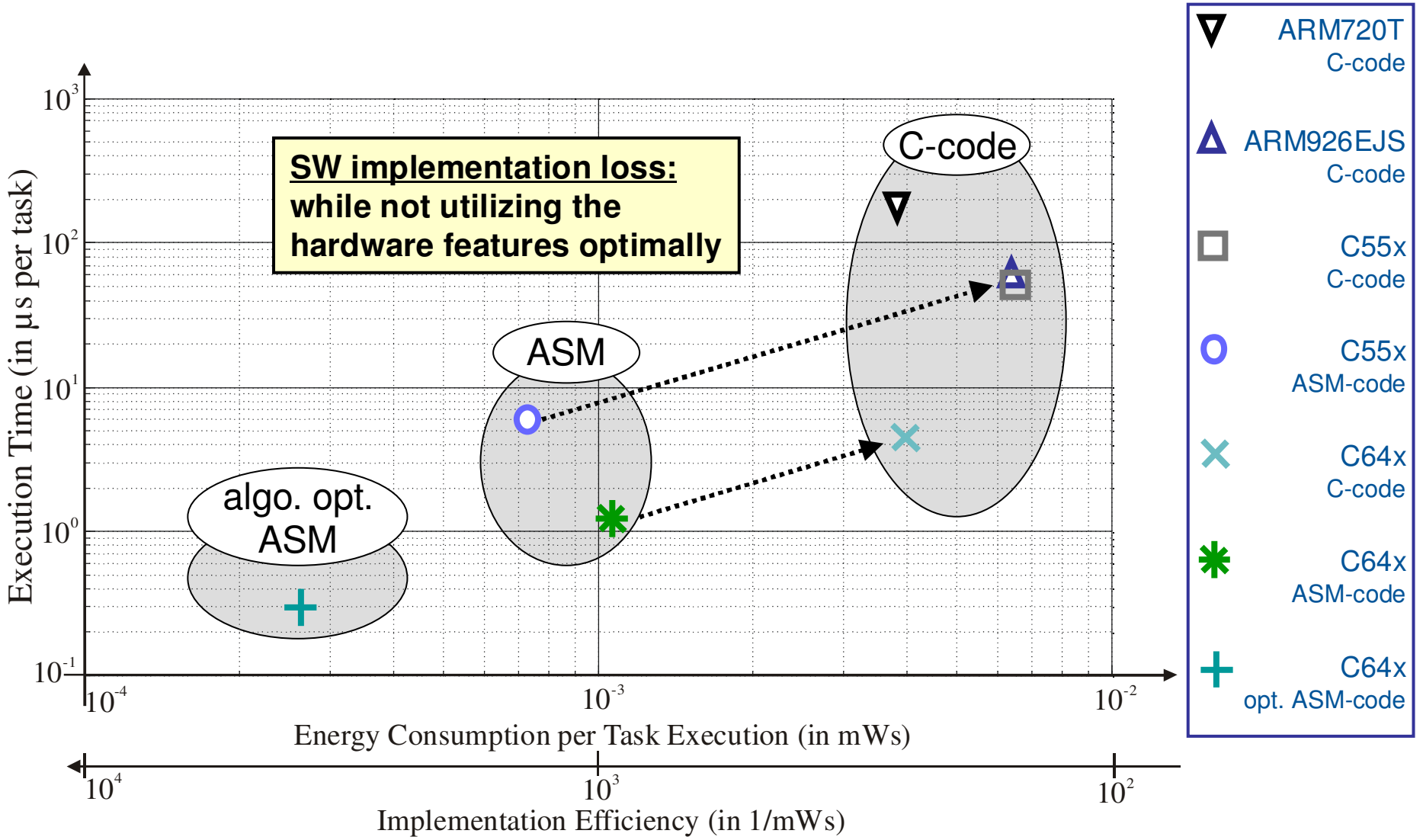
In-Depth Analysis: FFT results



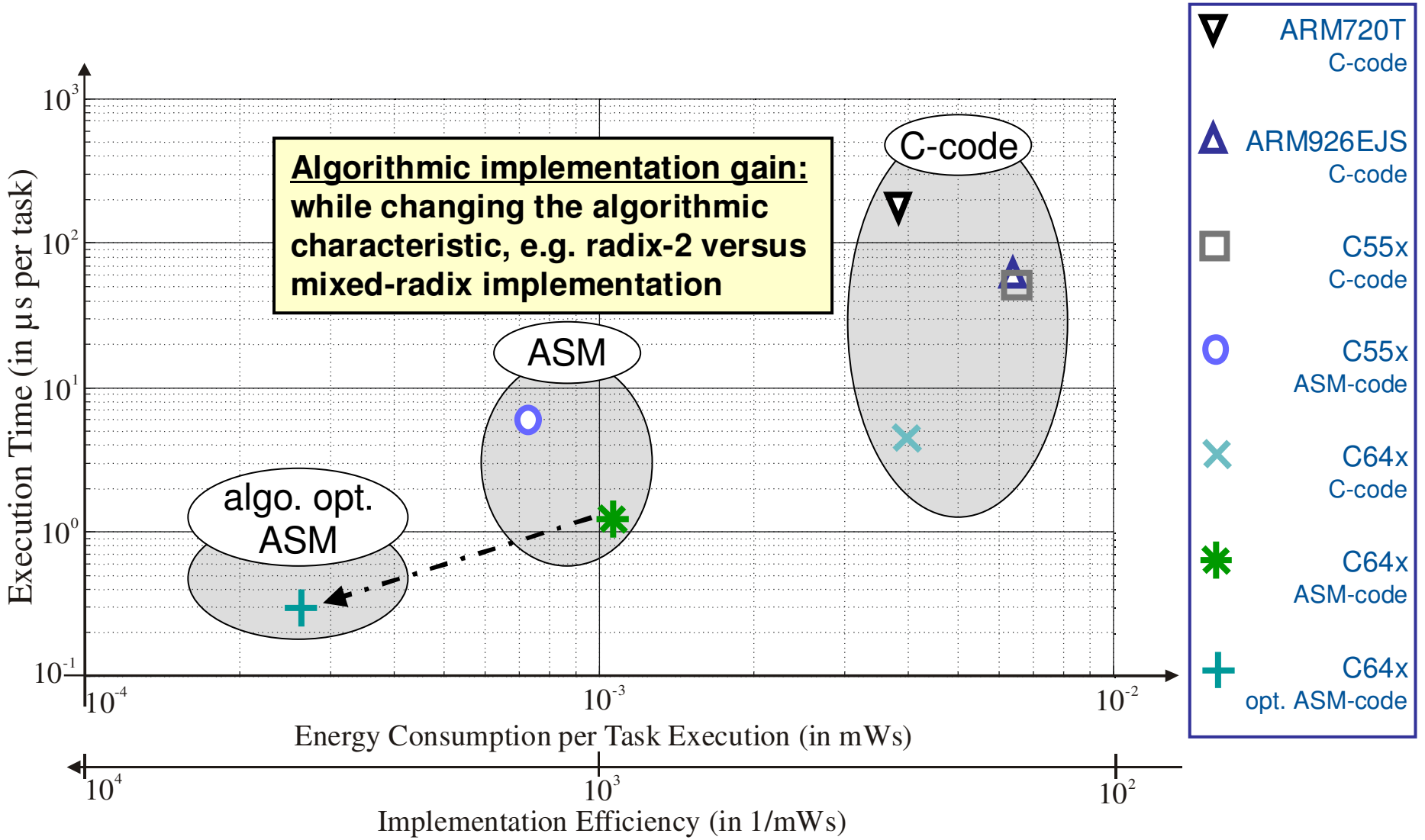
In-Depth Analysis: FFT results



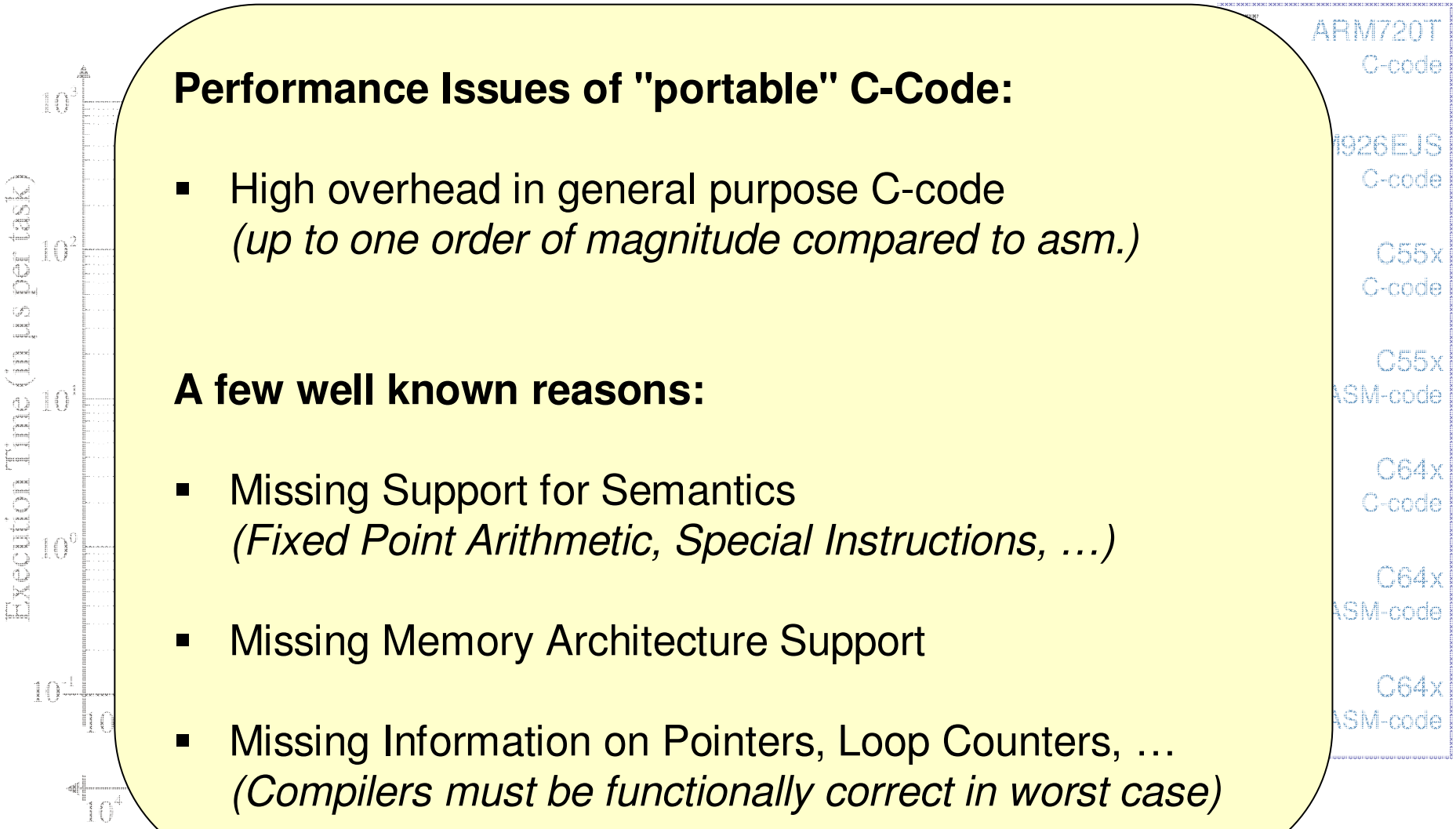
In-Depth Analysis: FFT results



In-Depth Analysis: FFT results



In-Depth Analysis: FFT results



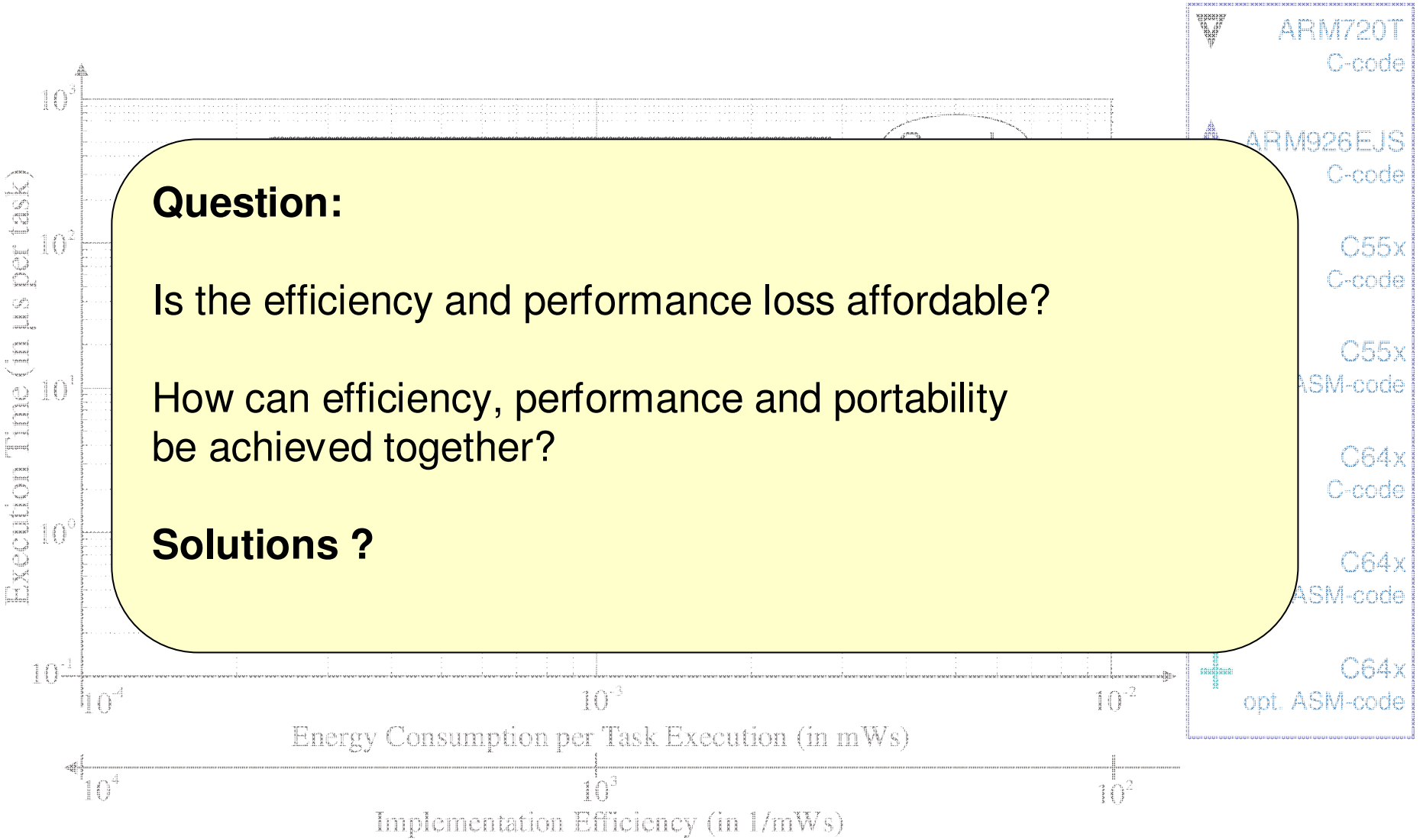
Performance Issues of "portable" C-Code:

- High overhead in general purpose C-code
  (up to one order of magnitude compared to asm.)

A few well known reasons:

- Missing Support for Semantics
  (Fixed Point Arithmetic, Special Instructions, …)
- Missing Memory Architecture Support
- Missing Information on Pointers, Loop Counters, …
  (Compilers must be functionally correct in worst case)

In-Depth Analysis: FFT results



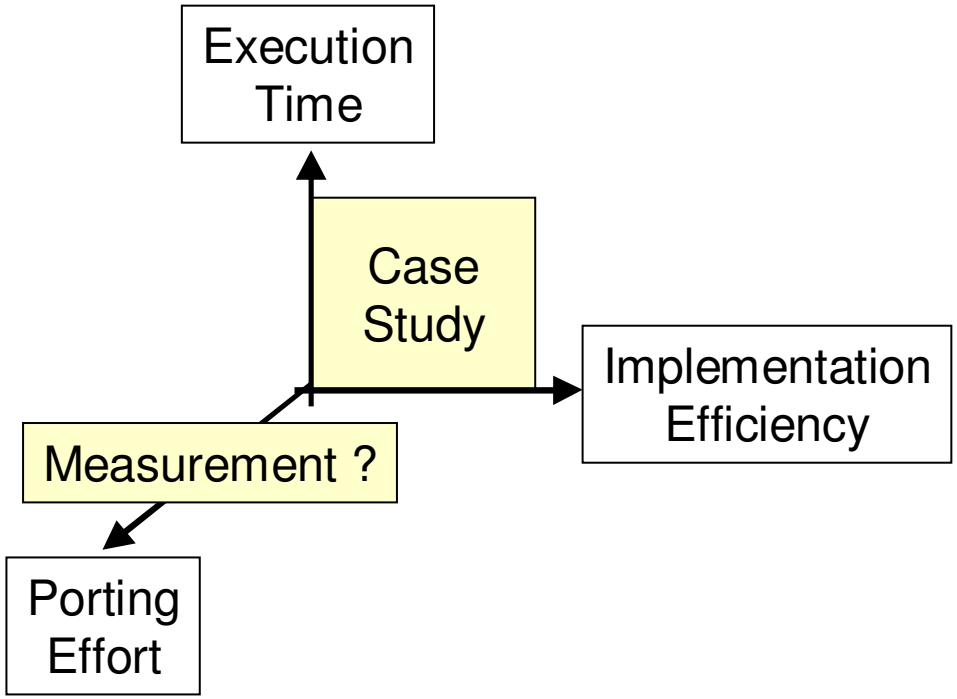
Question:

Is the efficiency and performance loss affordable?

How can efficiency, performance and portability be achieved together?

Solutions ?

Porting Effort



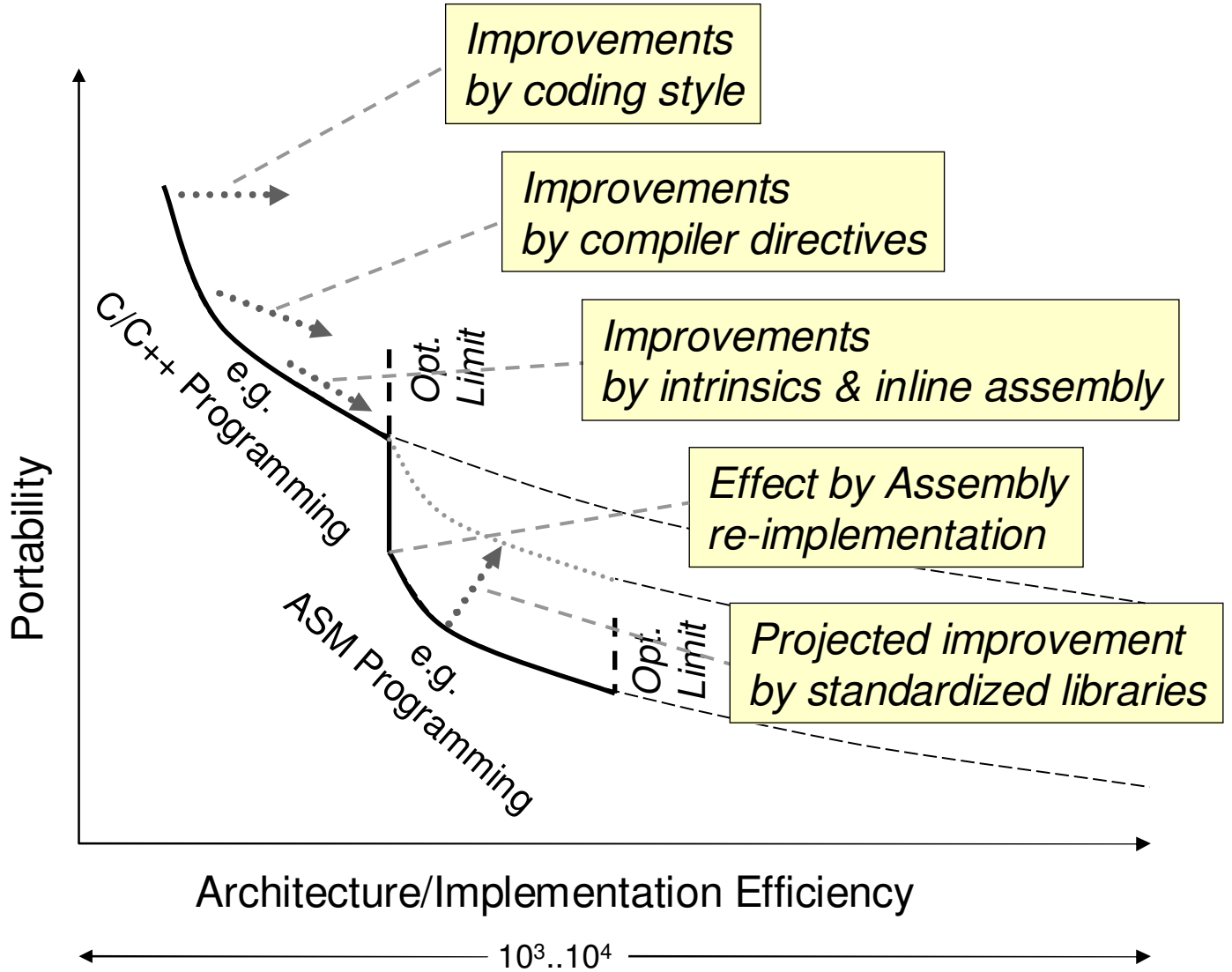
- **Measurement of Porting Effort is rather difficult, since development time depends heavily on developer (trainee vs. experienced developer)**
  - E.g. development of an optimized FIR filter in Assembly on the ARM Cortex-R4 is reported to be approx. 20 hours [1]

[1] BDTI Inc. “Evaluating the DSP Capabilities of the Cortex-R4”, InsideDSP

Agenda

- Introduction
- Portability versus Efficiency
- Case Study
  - Measurements
  - In-Depth Analysis: FFT kernel
- Conclusion/Outlook

Conclusions: Projected Effects of Optimizations



Summary & Outlook

Summary:

- Case study investigating portability vs. efficiency has been presented
- Three key effects have been discussed
  - HW implementation gain
  - SW implementation loss
  - Algorithmic implementation gain
- Possible optimizations to increase portability along with efficiency have been highlighted

Outlook:

- Further investigation of portability versus efficiency
- Waveform Description Languages (WDLs) on basis of such standardized libraries

Questions?

Thank you