Karlsruhe Institute of Technology

Code Parallelization for Multi-Core Software Defined Radio Platforms with OpenMP

Dipl.-Ing. Michael Schwall

SDR'11 WInnComm Europe, Brussels, Belgium

Communications Engineering Lab
Prof. Dr.rer.nat. Friedrich K. Jondral

Overview

- Motivation
- Waveform design and OpenMP
- Case Studies and Results
- Conclusion

Motivation

Karlsruhe Institute of Technology

The **GPP** has become an **important** digital signal processing unit for **SDRs**

→ GPP aided platforms

Communication **speed** and **complexity** of modern waveforms **increases**

→ High requirements on signal processing

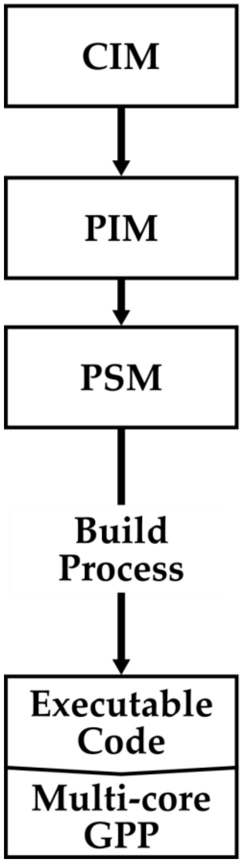
The **processing power** of GPPs increases through **parallelism** on processor level

→ Multi-core GPPs

Parallelization of algorithms to fully exploit the processing power of Multi-core GPPs

Waveform design and OpenMP

- Model-based waveform design



- Derived from the Model Driven Architecture (MDA) published by the Object Management Group (OMG)

- Computation Independent Model (CIM)

  Describes the waveform requirements independent of the implementation (specification of the radio standard)

- Platform Independent Model (PIM)

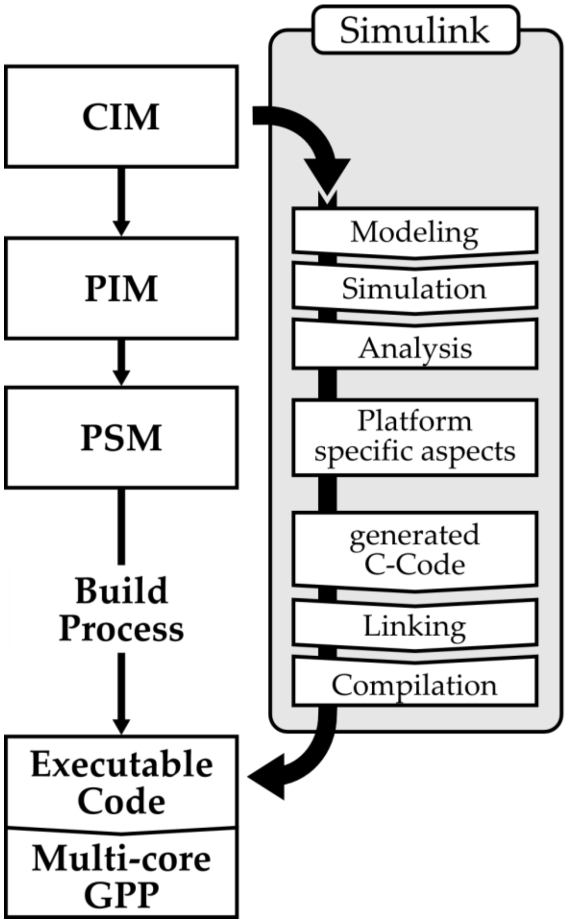
  Modeling the waveform's functionality without platform specific constraints

- Platform Specific Model (PSM)

  Extending the PIM with platform specific aspects

Waveform design and OpenMP

- Model-based waveform design with Simulink



- Simulink is used as an model-based design environment
- Modeling and simulation of dynamic systems

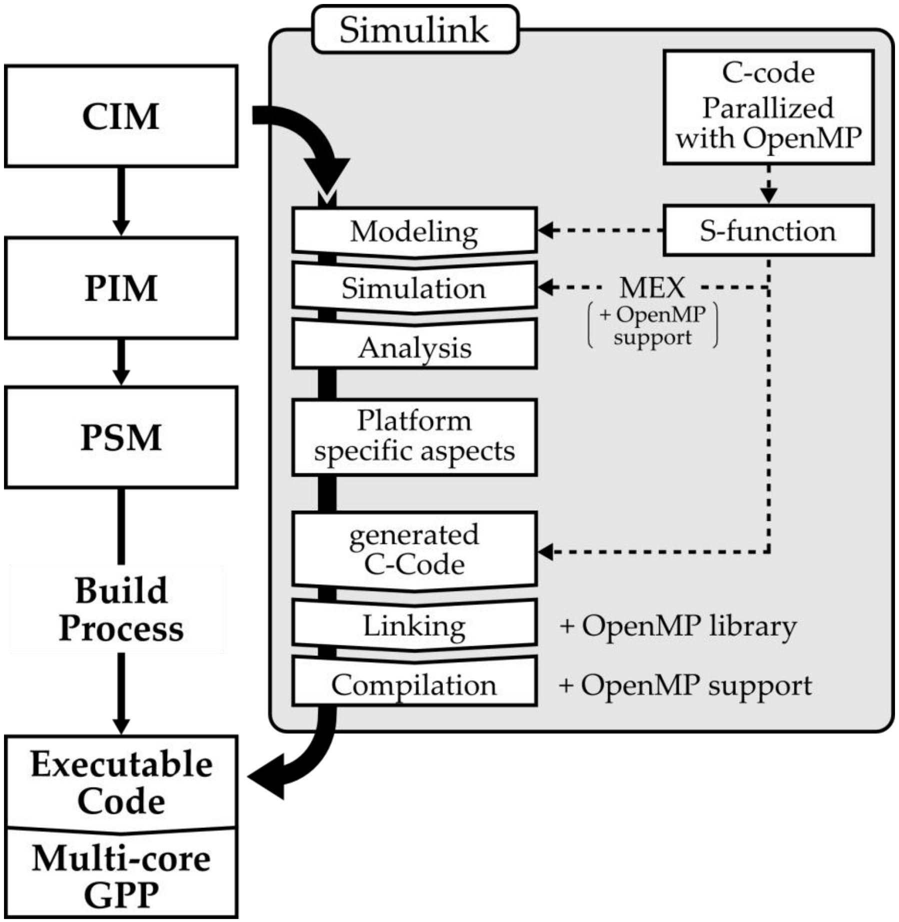
- ▶ Signal processing elements, e.g. a digital filter, are mapped to functional blocks
- ▶ An entire system is created by interlinking the blocks

Waveform design and OpenMP

- Open Multi-Processing (OpenMP)
  - Application Programming Interface (API) to parallelize C/C++ and Fortran code
  - Jointly developed by hardware vendors since 1997
  - Open standard for shared memory multiprocessing programming
  - Implemented in various compilers (e.g. GCC, Microsoft Visual Studio)
  - The programming language is extended with compiler directives, functions and environment variables
  - OpenMP's directives enable
    - … to initialize and start threads
    - … to terminate threads
    - and to share the work between them

Waveform design and OpenMP

- Model-based waveform design with Simulink supported by OpenMP

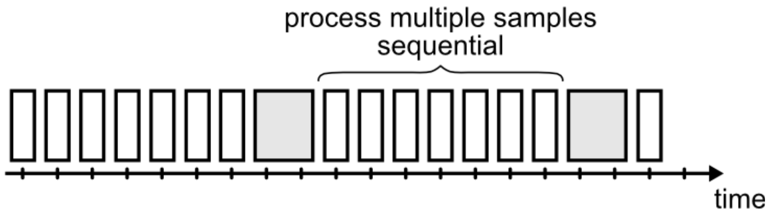


- Parallelized C-code is integrated into Simulink using S-functions
- The code can already be simulated in the PIM by generating an Matlab Executable (MEX) file
- The parallelized Code is embedded in the overall model C-Code and subsequently compiled with OpenMP support

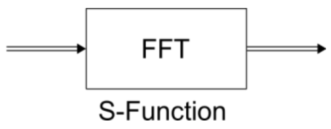
Waveform development and OpenMP

- What can be parallelized?

Frame-based



Simulink block

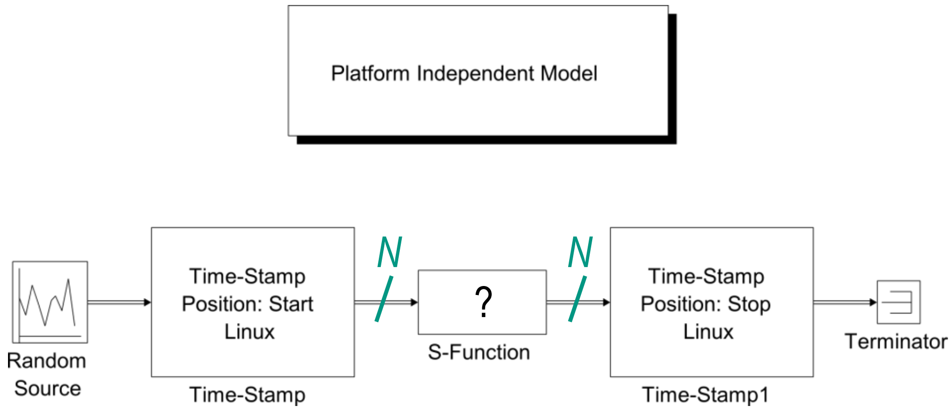


Frame-based processing

- Fixed process overhead is **distributed** across many samples
- **Common format** in real-time systems and communications

Simulink and OpenMP

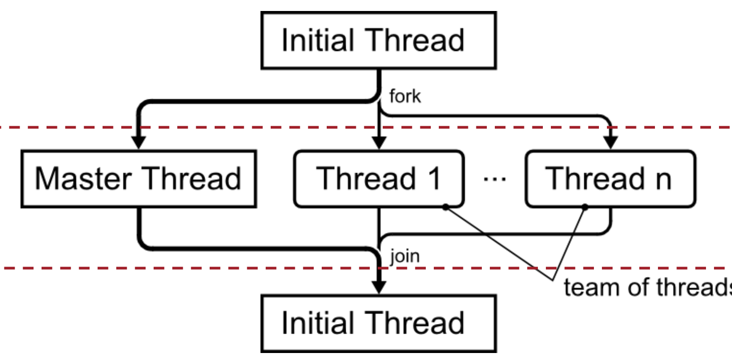
- Template Model:



Code snippet

```
pragma omp parallel for
for (i=0; i<N; i++) {

  output[i] = 2.0*input[i];
}
```



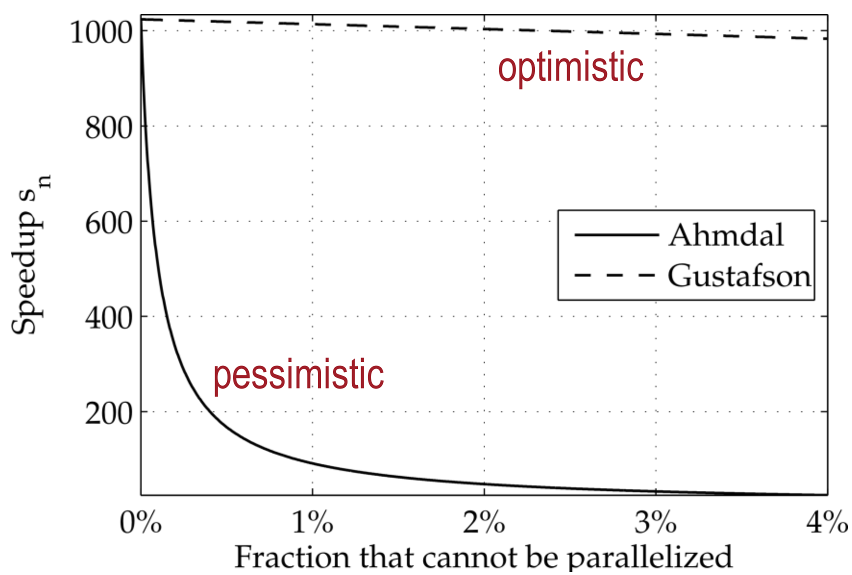
Case studies

Karlsruhe Institute of Technology

- Test environment:
  - AMD Phenom II X4 995 processor with **four cores**
  - Matlab Simulink R2010a
  - GCC compiler 4.4.3
  - OpenMP specification 3.0
- No SDR hardware considerations → only **simulation of the PIM**
- **Number of threads is independent of the number of processing cores**
  - … but, using more threads than available cores will dramatically reduce the performance
  - Number of threads will be varied from 1 to 4
- Simulink frame length will be $N=2^k$, whereas k=3,4,…,13

Case studies

- Figures of merit
  - Speedup: $$s_n = \frac{t_1}{t_n}$$ t_n: duration of the application using *n* threads
  - Efficiency: $$e_n = \frac{s_n}{n}$$



Case studies

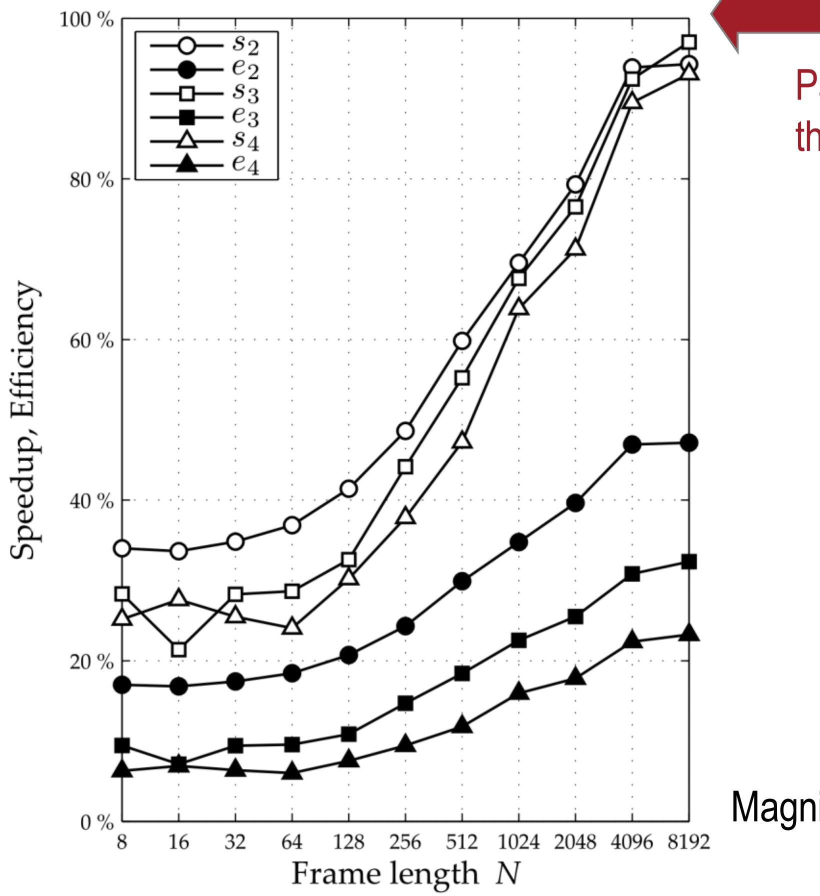
- CS1: Elementary DSP operations
  - Implementation

$$
\begin{aligned}
c(i) &= a(i) + b(i), \quad i = 0, \ldots, N-1 \\
c(i) &= a(i) \cdot b(i) \\
c(i) &= |z(i)|^2 \\
c(i) &= \angle z(i) \\
c &= \sum_{i=0}^{N-1} a(i) \cdot b(i)
\end{aligned}
$$

No data dependencies
within the equations!

Case studies

- CS1: Elementary DSP operations
  - Results



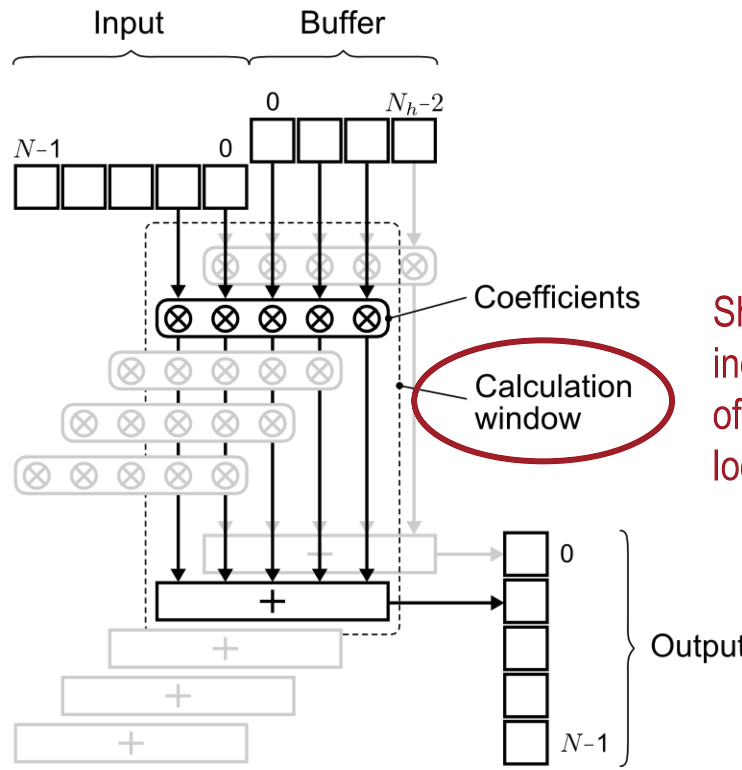
Parallelized code is **slower** than the serial one!

Magnitude-squared

Case studies

- CS2: FIR Filter
  - Implementation

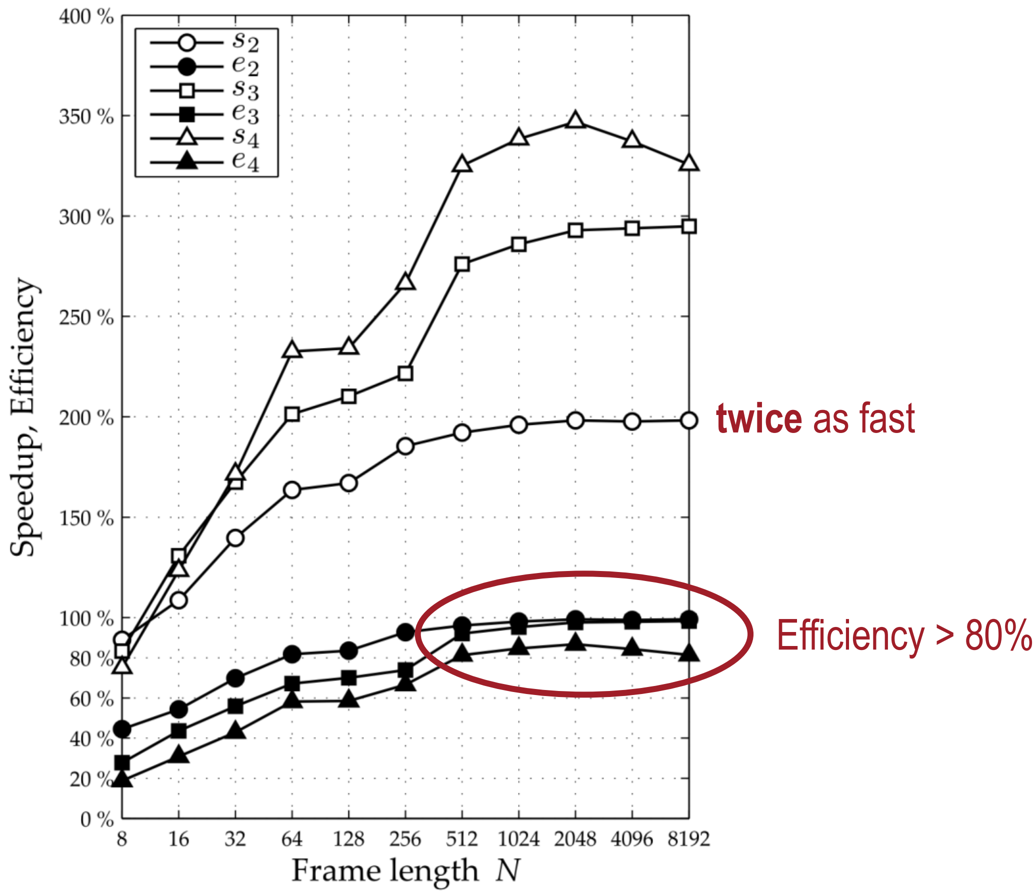
$$y(i) = \sum_{k=0}^{N_h-1} h(k)x(i-k)$$



Should be independent of the current loop index

Case studies

- CS2: FIR Filter
  - Results

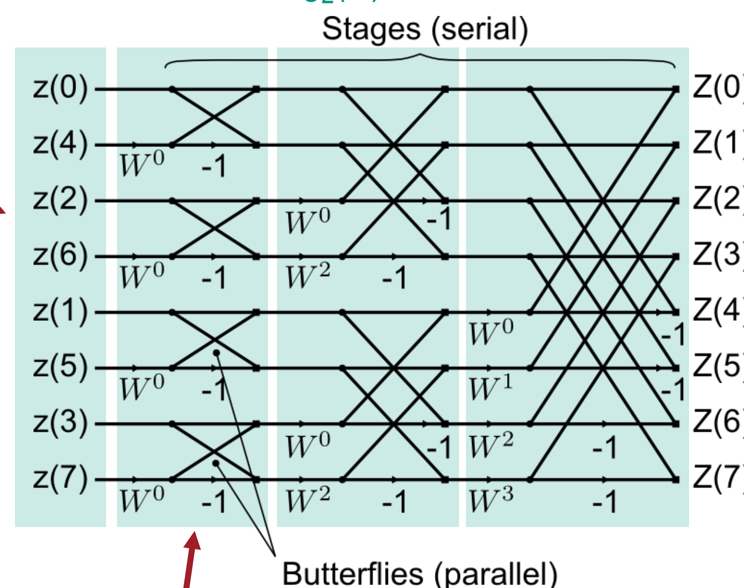


Case studies

- CS3: FFT
  - Implementation
    - Length N
    - Radix-2 method
    - Time domain reduction

$\text{Log}_2(\text{N})$ serial tasks

N parallel tasks



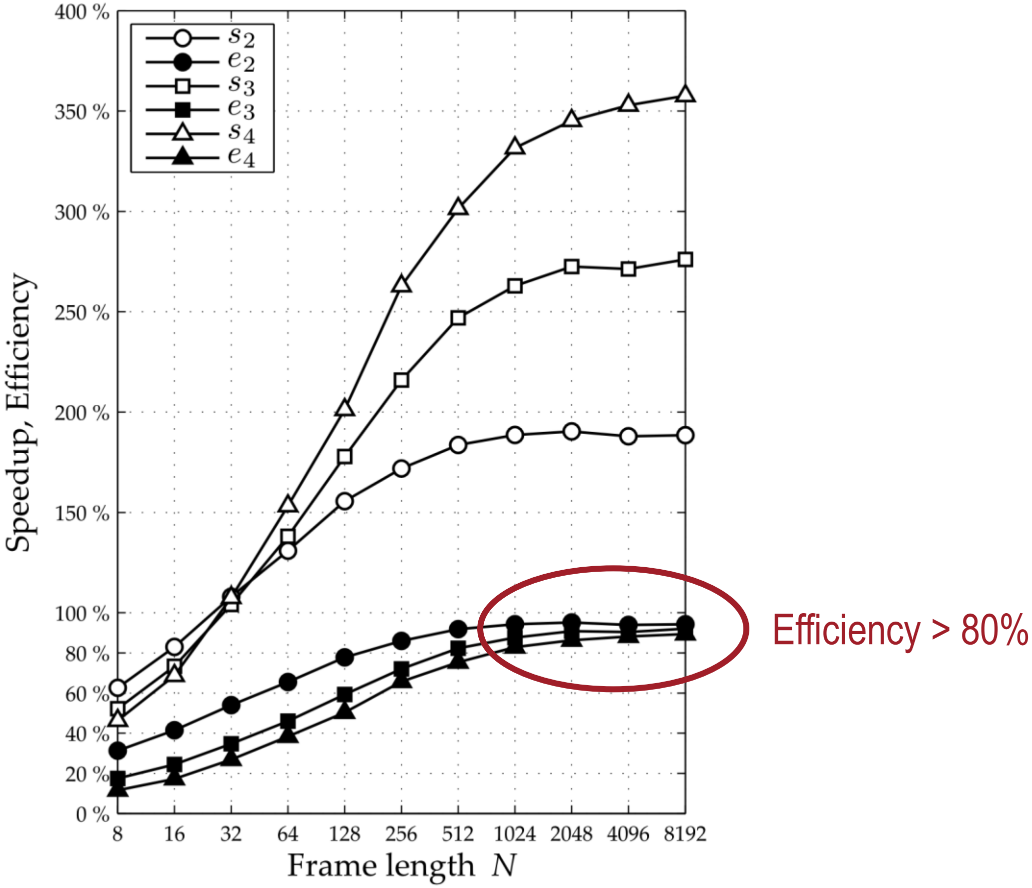
N/2 parallel tasks

twiddle factor:

$$W^n = e^{-j2\pi\frac{n}{N}}$$

Case studies

- CS3: FFT
  - Results



Dipl.-Ing. Michael Schwall
Code Parallelization for Multi-Core Software Defined Radio Platforms with OpenMP
Communications Engineering Lab
Prof. Dr.rer.nat. Friedrich K. Jondral

Case studies

Karlsruhe Institute of Technology

- Results
  - Including OpenMP (parallelized code) into an existing model-based waveform design environment is **possible**
  - Using OpenMP to parallelize code is **simple**

  - …but, **data dependencies** within an algorithm have to be identified an removed at first
  - In most cases, this means a complete re-structuring of the code
  - The **computational complexity** has to dominate the processing overhead, caused by the thread scheduling

  - …but, once computational complex algorithms are parallelized, their **speedup scales** with the number of threads (number of processing cores)

Thank you for your attention!

Q&A