Generating High Performance Pruned FFT Implementations

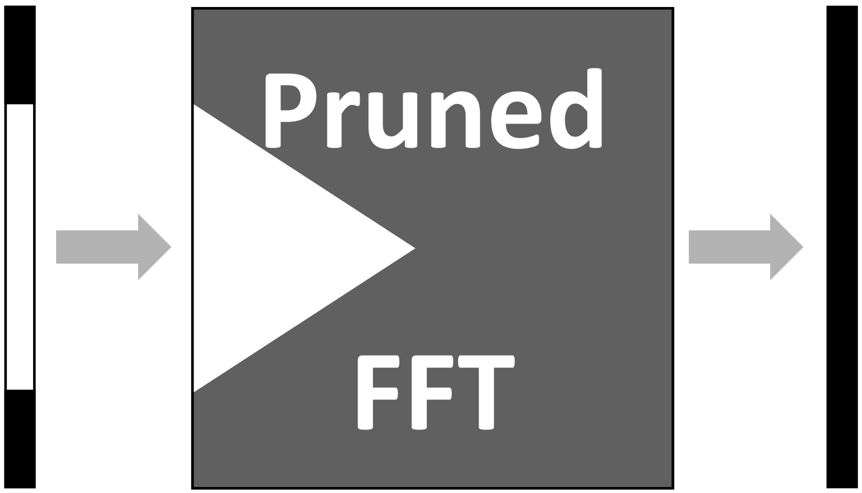
Franz Franchetti, Markus Püschel

Electrical and Computer Engineering
Carnegie Mellon University

Sponsors: DARPA DESA program, NSF-NGS/ITR, NSF-ACR, Mercury, and Intel

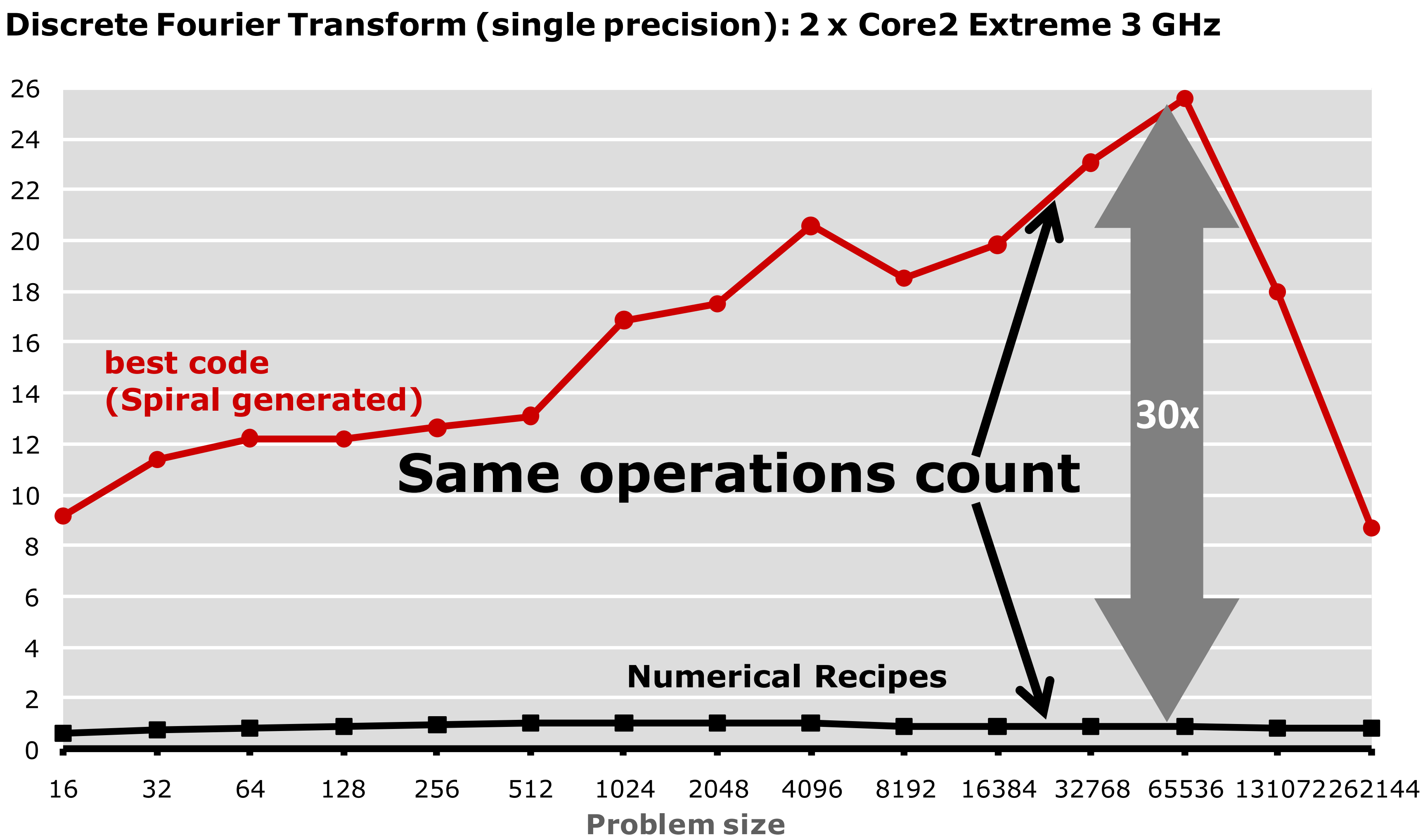
The Idea: Pruned FFT

- **Input pruning**
  E.g., center ¾ inputs are known to be zero
- **Output pruning**
  E.g., only the low ½ frequencies are used
- **Simultaneous input & output pruning**
  Some inputs known zeros and some outputs discarded



Pruned DFT: 5% – 30% operations reduction in application settings

The Problem



Can we turn 5% – 30% operations savings into *speed-up*?

Organization

- **Spiral overview**
- Pruned FFT
- Results
- Concluding remarks

Spiral

- **Library generator for linear transforms (DFT, DCT, DWT, filters, ….) *and recently more …***
- **Wide range of platforms supported: scalar, fixed point, vector, parallel, Verilog, GPU**
- **Research Goal: "Teach" computers to write fast libraries**
  - Complete automation of implementation and optimization
  - Conquer the "high" algorithm level for automation
- **When a new platform comes out: Regenerate a retuned library**
- **When a new platform paradigm comes out (e.g., CPU+GPU): Update the tool rather than rewriting the library**

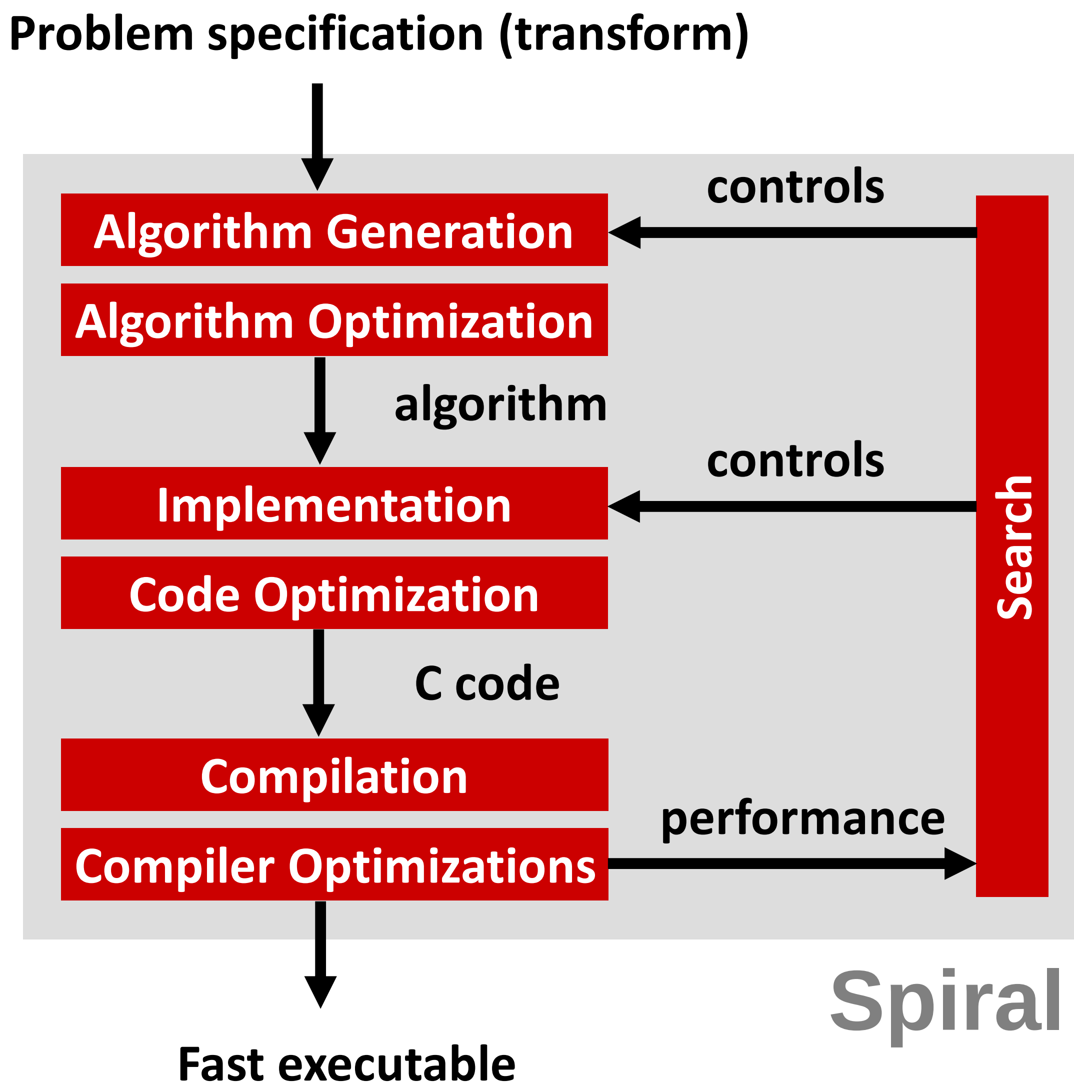
Intel uses Spiral to generate parts of their MKL and IPP libraries

How Spiral Works

Spiral:
Complete automation of the implementation and optimization task

Basic idea:
Declarative representation of algorithms

Rewriting systems to generate and optimize algorithms



Fast Algorithms, Example: 4-point FFT

- **Fast algorithms = matrix factorizations**

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix} x \rightarrow \begin{bmatrix}1 & \cdot & 1 & \cdot\\ \cdot & 1 & \cdot & 1\\ 1 & \cdot & -1 & \cdot\\ \cdot & 1 & \cdot & -1\end{bmatrix}\begin{bmatrix}1 & \cdot & \cdot & \cdot\\ \cdot & 1 & \cdot & \cdot\\ \cdot & \cdot & 1 & \cdot\\ \cdot & \cdot & \cdot & i\end{bmatrix}\begin{bmatrix}1 & 1 & \cdot & \cdot\\ 1 & -1 & \cdot & \cdot\\ \cdot & \cdot & 1 & 1\\ \cdot & \cdot & 1 & -1\end{bmatrix}\begin{bmatrix}1 & \cdot & \cdot & \cdot\\ \cdot & \cdot & 1 & \cdot\\ \cdot & 1 & \cdot & \cdot\\ \cdot & \cdot & \cdot & 1\end{bmatrix} x$$

12 adds
4 mults

4 adds **1 mult** **4 adds**

$$\mathrm{DFT}_4 \rightarrow (\mathrm{DFT}_2 \otimes \mathrm{I}_2) T_2^4 (\mathrm{I}_2 \otimes \mathrm{DFT}_2) L_2^4$$

Fourier transform **Kronecker product** **Identity** **Permutation**

- **SPL = mathematical, declarative specification**
- **SPL formula can be translated into program**

Transforms and Breakdown Rules

$$\mathrm{DFT}_n \rightarrow (\mathrm{DFT}_k \otimes \mathrm{I}_m)\, \mathrm{T}^n_m (\mathrm{I}_k \otimes \mathrm{DFT}_m)\, \mathrm{L}^n_k, \quad n = km$$

$$\mathrm{DFT}_n \rightarrow P_n(\mathrm{DFT}_k \otimes \mathrm{DFT}_m) Q_n, \quad n = km,\ \gcd(k, m) = 1$$

$$\mathrm{DFT}_p \rightarrow R_p^T (\mathrm{I}_1 \oplus \mathrm{DFT}_{p-1}) D_p (\mathrm{I}_1 \oplus \mathrm{DFT}_{p-1}) R_p, \quad p \text{ prime}$$

DCT-3 …

"Teaches" Spiral about existing algorithm knowledge (~200 journal papers)

DCT-4 …

$$\mathrm{IMDCT}_{2m} \rightarrow (\mathrm{J}_m \oplus \mathrm{I}_m \oplus \mathrm{I}_m \oplus \mathrm{J}_m)\left(\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes \mathrm{I}_m\right) \oplus \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} \otimes \mathrm{I}_m\right)\right) \mathrm{J}_{2m}\, \mathrm{DCT\text{-}4}_{2m}$$

$$\mathrm{WHT}_{2^k} \rightarrow \prod_{i=1}^{t} (\mathrm{I}_{2^{k_1+\cdots+k_{i-1}}} \otimes \mathrm{WHT}_{2^{k_i}} \otimes \mathrm{I}_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t$$

$$\mathrm{DFT}_2 \rightarrow \mathrm{F}_2$$

$$\mathrm{DCT\text{-}2}_2 \rightarrow \mathrm{diag}(1, 1/\sqrt{2})\, \mathrm{F}_2$$

$$\mathrm{DCT\text{-}4}_2 \rightarrow \mathrm{J}_2\, \mathrm{R}_{13\pi/8}$$

Base case rules

Goal: Derive Cooley-Tukey Pruned FFT rule

Organization

- Spiral overview
- **Pruned FFT**
- Results
- Concluding remarks

Data Sparseness: Block Sequences

- **Sequence**

  $$\sigma = \langle \sigma_i \rangle_{0 \le i < |\sigma|} \subset \{0, \ldots, N-1\}$$

- **Block sequence**

  $$\sigma \otimes k = \langle k\sigma_i, k\sigma_i + 1, \ldots, k\sigma_i + k - 1 \rangle_{0 \le i < |\sigma|}$$

- **Example**

  Let $\sigma = \langle 0, 1, 3 \rangle \subset \{0, 1, 2, 3\}$ and $k = 2$. Then
  $\sigma \otimes k = \langle 0, 1, 2, 3, 6, 7 \rangle \subset \{0, \ldots, 7\}$.

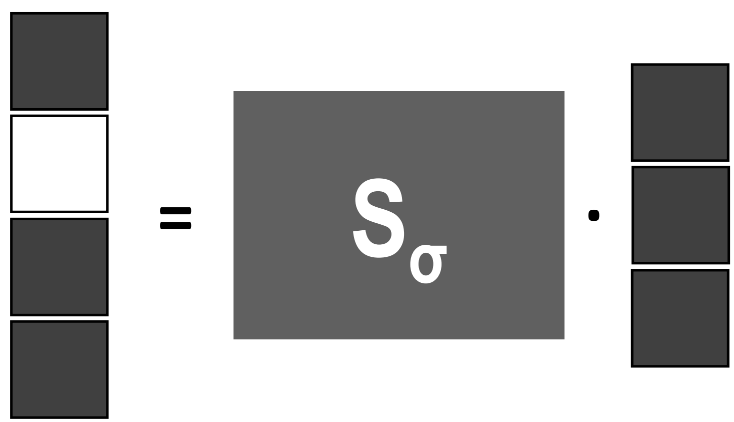
Zero-Padding: Scatter Matrix

- **Definition**

$$y = \mathrm{S}_\sigma x \quad \text{with} \quad \mathrm{S}_\sigma = \left[e^N_{\sigma_0} | e^N_{\sigma_1} | \ldots | e^N_{\sigma_{|\sigma|-1}} \right]$$

- **Example** $\sigma = \langle 0, 1, 3 \rangle \subset \{0, 1, 2, 3\}$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_0 \\ 0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$



Cooley-Tukey Pruned FFT Rule

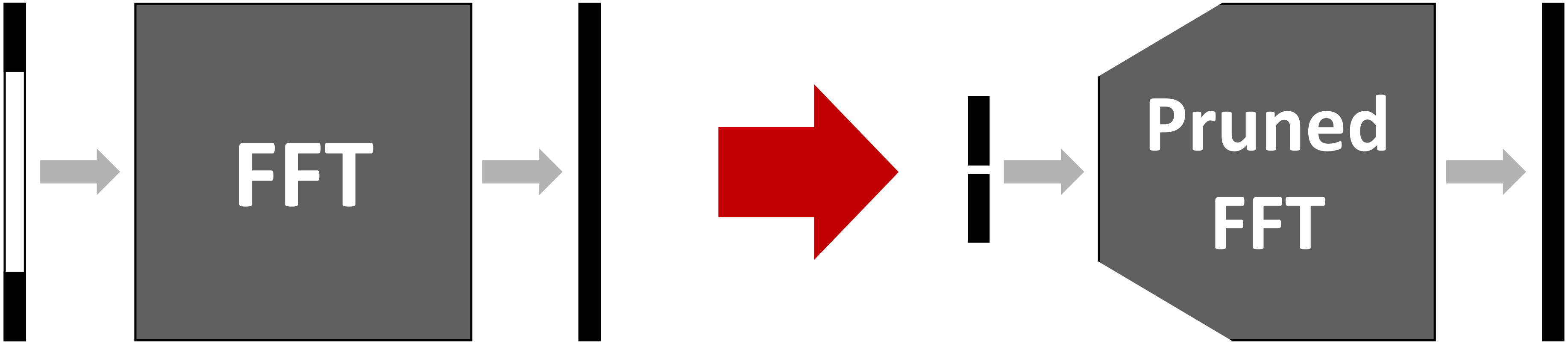
- **Recursive input pruning rule**

$$\mathrm{PDFT}_{kmn}^{\sigma\otimes km} \rightarrow (\mathrm{DFT}_m \otimes I_{kn}) T_{kn}^{kmn} L_m^{kmn} (\mathrm{PDFT}_{kn}^{\sigma\otimes k} \otimes I_m)$$

- **Base case**

$$\mathrm{PDFT}_n^{\sigma} \rightarrow \mathrm{DFT}_n \mathrm{S}_\sigma$$

- **Similar rule for output pruning and simultaneous pruning**



Derivation: Cooley-Tukey Pruned FFT Rule

$$\mathrm{PDFT}_{kmn}^{\sigma\otimes km} \rightarrow \underline{\mathrm{DFT}_{kmn}}\mathrm{S}_{\sigma\otimes km}$$

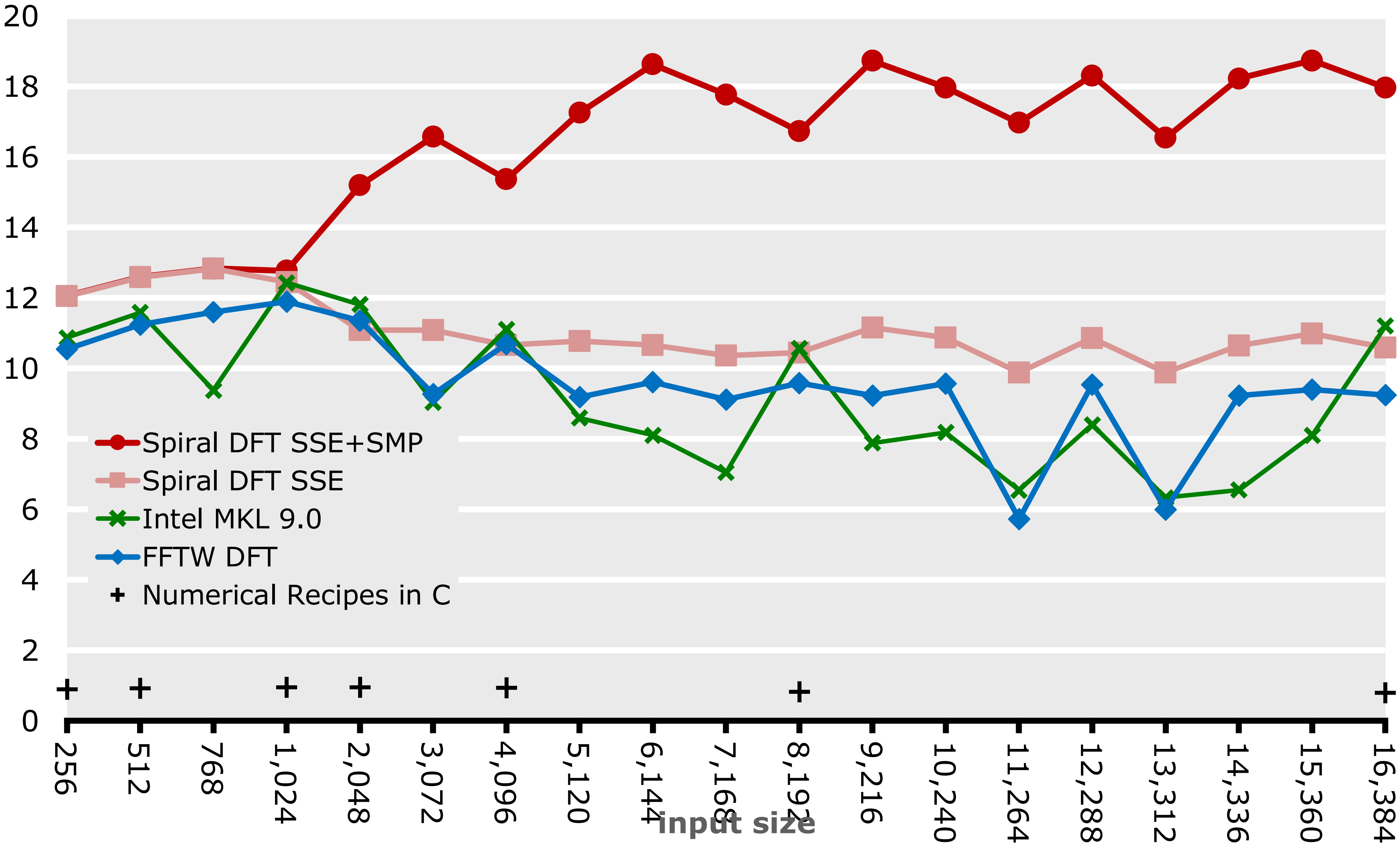
Cooley-Tukey FFT rule + Kronecker product identities

Organization

- Spiral overview
- Pruned FFT
- **Results**
- Concluding remarks

DFT: Spiral vs. FFTW and MKL (2 cores, 4-way SSE)

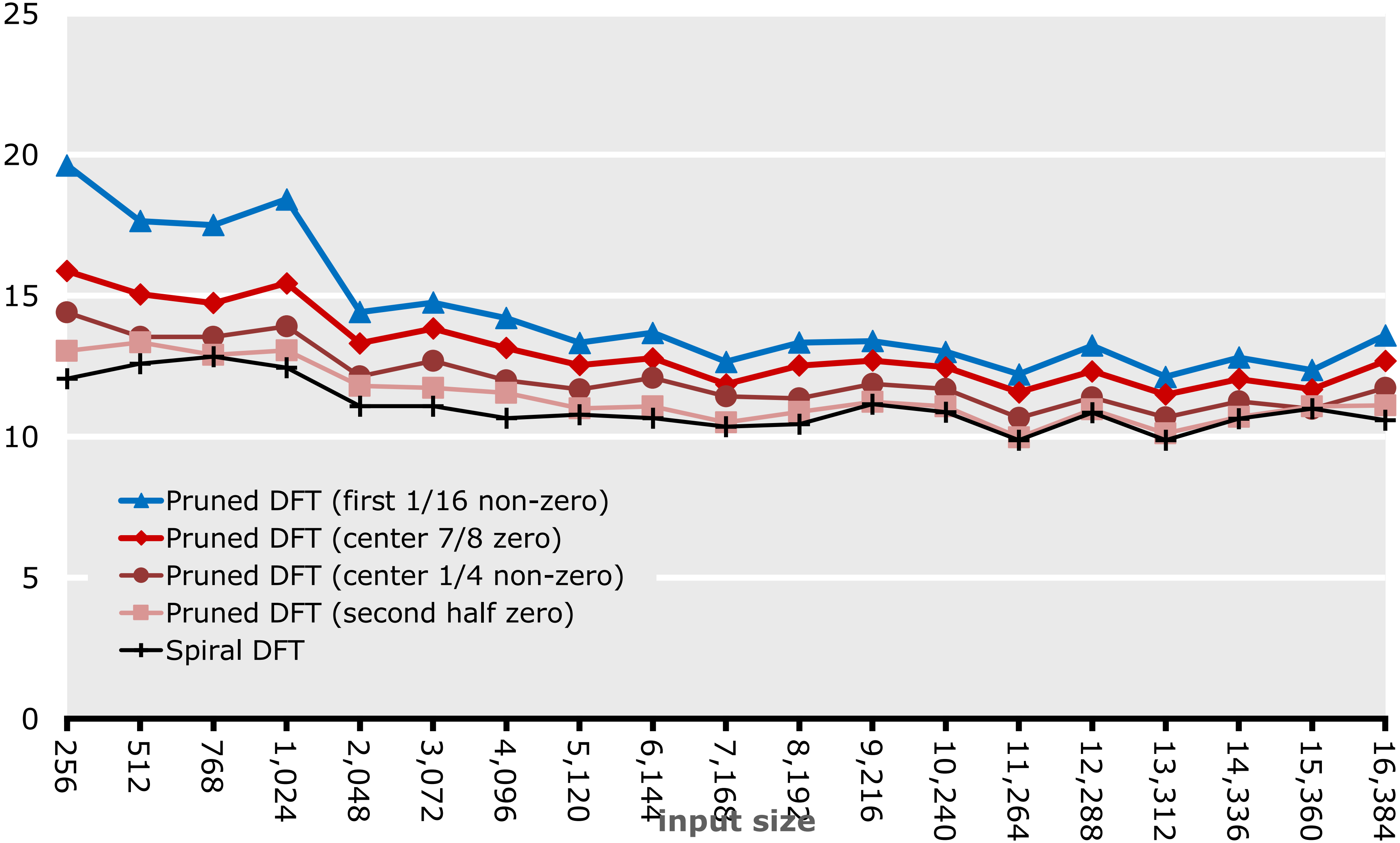
performance [Gflop/s], single-precision, Intel C++ 10.1, SSSE3, Windows XP 32-bit



Spiral-generated DFT is good *baseline*

Spiral: Pruned DFT vs. DFT (4-way SSE)

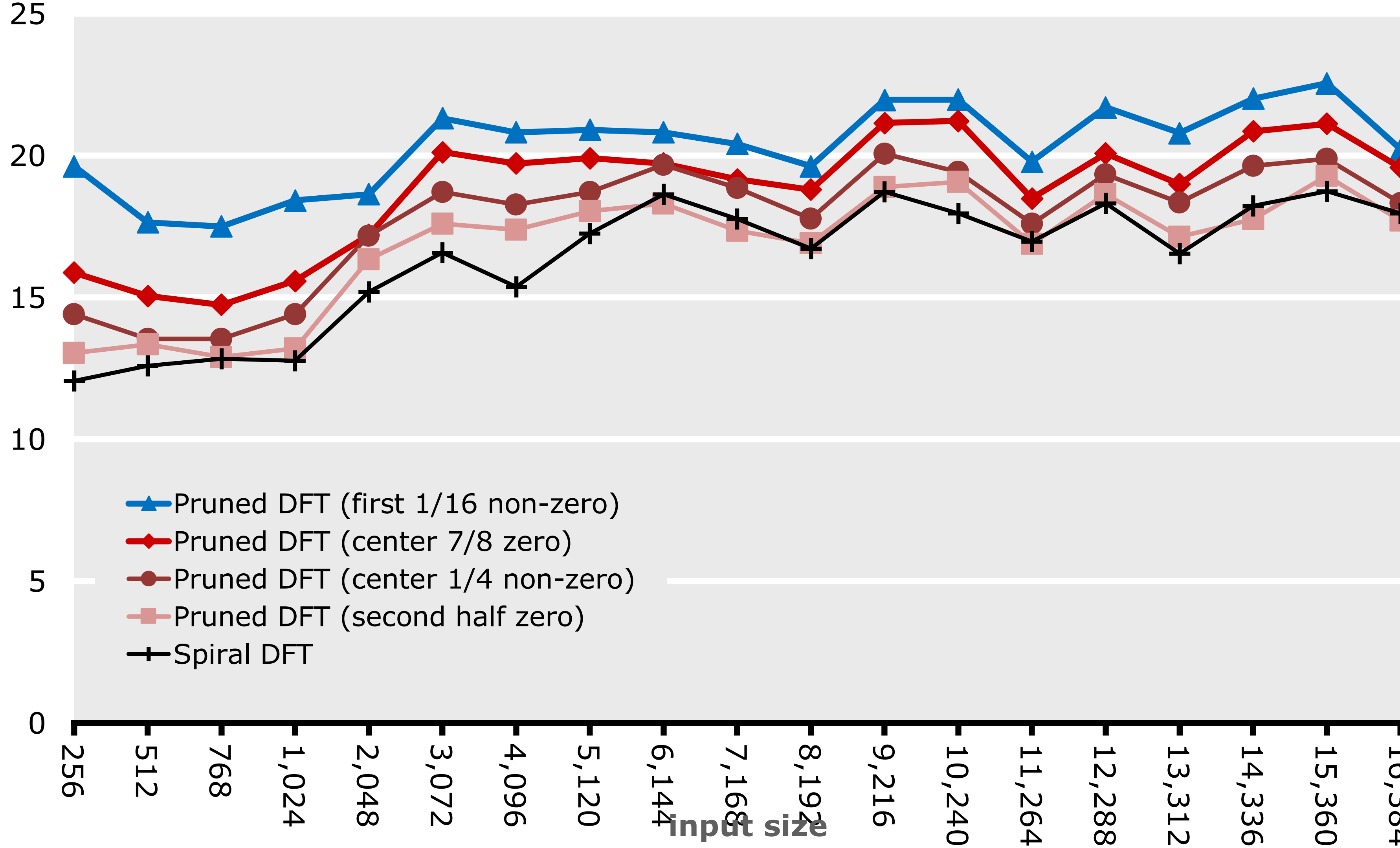
performance [Gflop/s], single-precision, Intel C++ 10.1, SSSE3, Windows XP 32-bit



FFT input pruning: speed-up for *sequential vector* DFT

Spiral: Pruned DFT vs. DFT (2 cores, 4-way SSE)

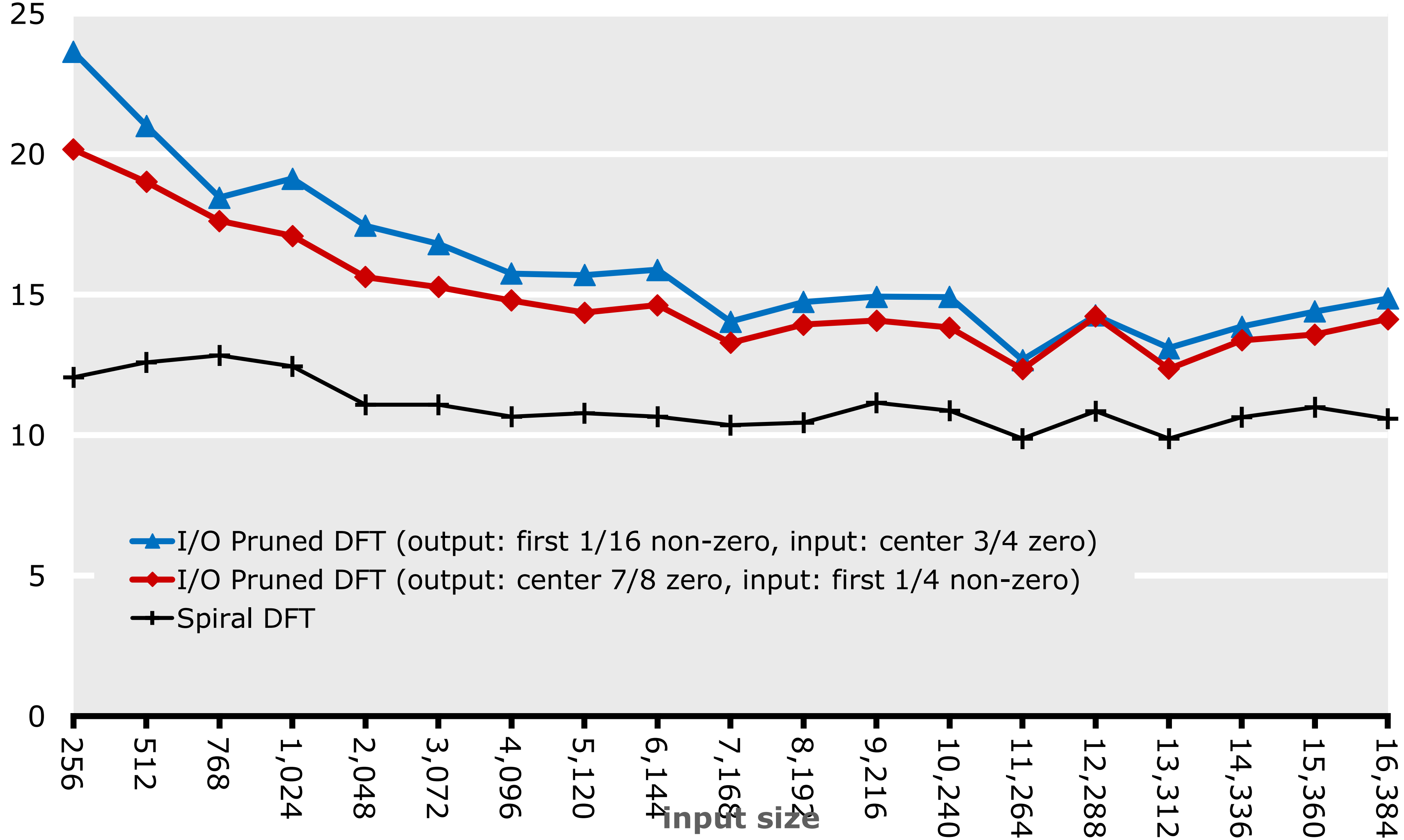
performance [Gflop/s], single-precision, Intel C++ 10.1, SSSE3, Windows XP 32-bit



FFT input pruning: speed-up for *parallel vector* DFT

Spiral: I/O Pruned DFT vs. DFT (4-way SSE)

performance [Gflop/s], single-precision, Intel C++ 10.1, SSSE3, Windows XP 32-bit



I/O pruning: better speed-up than input pruning only

Organization

- Spiral overview
- Pruned FFT
- Results
- **Concluding remarks**

Summary

- **Spiral's goal: "Teach" computers to write fast libraries**
  From problem specification to very fast code---automatically (click button)
- **Optimization at a high level of abstraction**
  Memory hierarchy, vector SIMD, multicore,…
- **The generated programs are very fast**
  Often better than human-written code
- **Pruned FFT: lower operations count translates into speed-up**
  up to 30% over best vector SIMD and multicore code for input pruning