A Rewriting System for the Vectorization of Signal Transforms

Franz Franchetti
Yevgen Voronenko
Markus Püschel

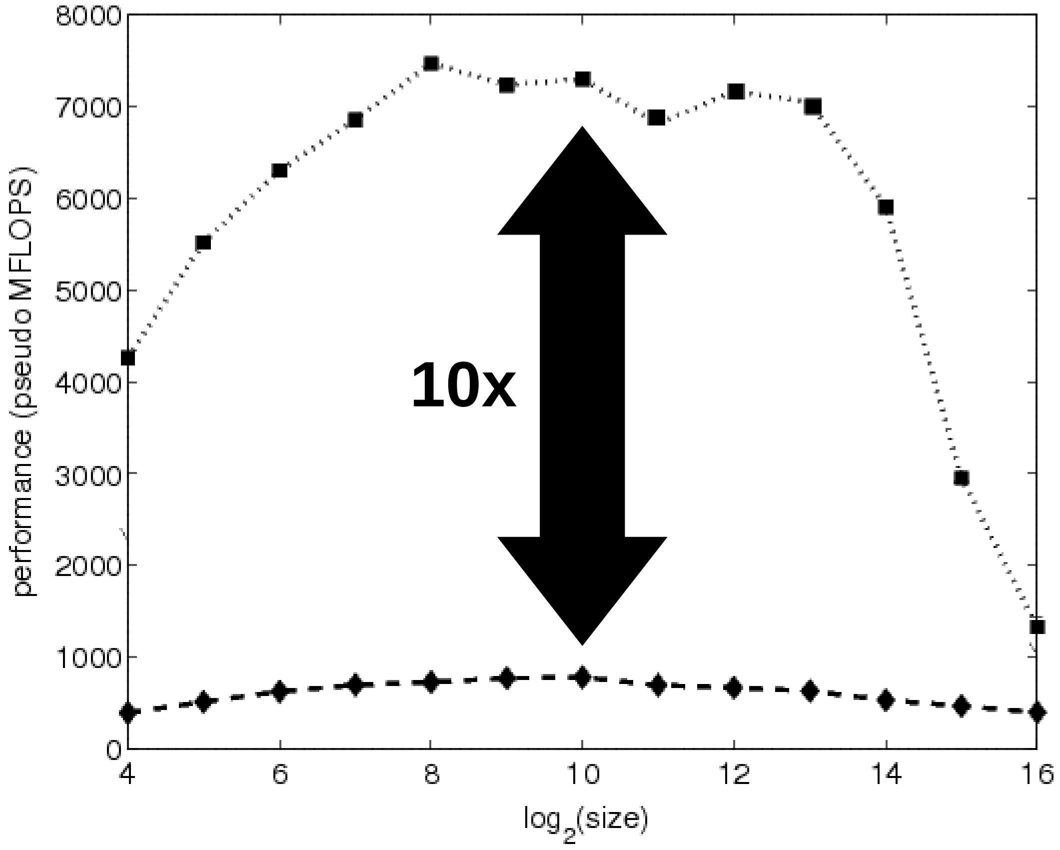
Department of Electrical & Computer Engineering
Carnegie Mellon University

`http://www.spiral.net`

Supported by: NSF ACR-0234293, ITR/NGS-0325687,
DARPA NBCH-105000, Intel, Austrian FWF

SPIRAL
www.spiral.net

The Problem (Example FFT Performance)



best available implementation
(FFTW, Intel IPP, Spiral)

roughly the same operations count

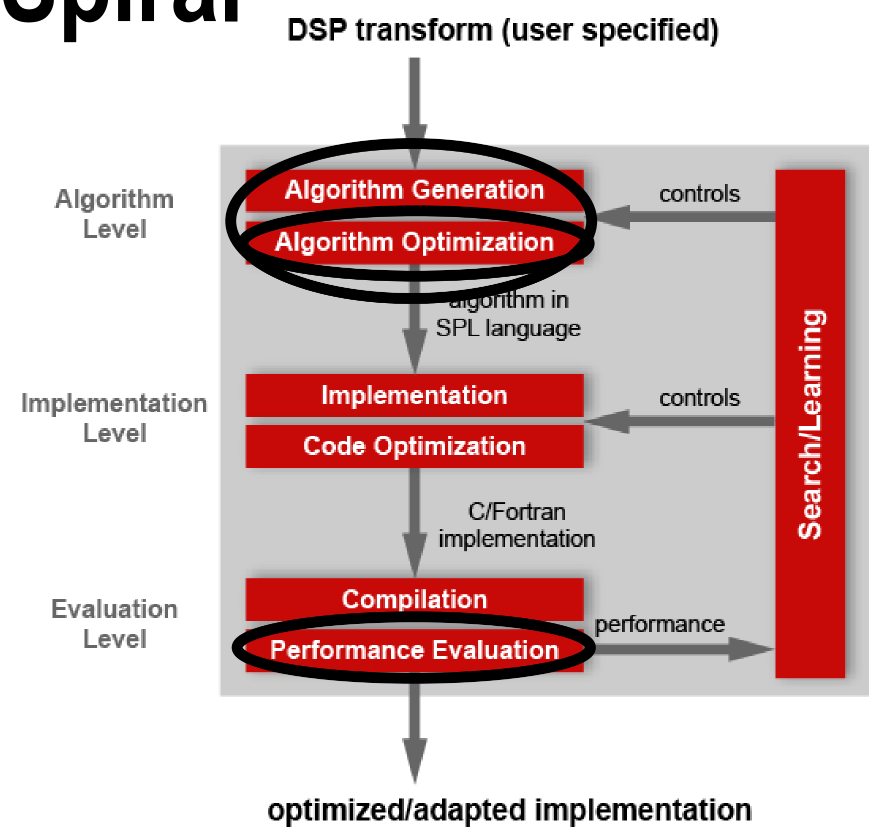
reasonable implementation
(Numerical recipes. GNU scientific library)

Solution: program generators like Atlas and Spiral, adaptive libraries like FFTW

Organization

- Spiral overview
- SIMD vector instructions
- Vectorization by rewriting
- Extension to SMP and Multicore
- Experimental results
- Summary

Spiral



- **Program generation from a problem specification** for linear digital signal processing (DSP) transforms (DFT, DCT, DWT, filters, ….)
- **Goal 1:** A flexible <u>push-button</u> program generation framework for an entire domain of algorithms
- **Goal 2:** With new architectures, update the tool rather than the individual programs in the library

Spiral: Principle 2: **Optimization at a high level of abstraction** ***ed memory,***

multicore, distributed memory, FPGAs, embedded CPUs

Markus Püschel, José M. F. Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo, **SPIRAL: Code Generation for DSP Transforms,** Proceedings of the IEEE 93(2), 2005

What is a DSP Transform?

- Mathematically: Matrix-vector multiplication

$$x \mapsto y = T \cdot x$$

input vector (signal)

output vector (signal)

transform = matrix

- Example: Discrete Fourier transform (DFT)

$$\mathrm{DFT}_n \;=\; [e^{-2k\ell\pi i/n}]_{0\le k,\ell<n}$$

DSP Algorithms: Example 4-point DFT

- Algorithm = sparse matrix factorization
- Reduce computation cost from $O(n^2)$ to $O(n \log n)$
- For every transform there are many fast algorithms

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} x \to \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} x$$

12 adds
4 mults

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

(when multiplied with input vector x)

$$\mathrm{DFT}_4 \to (\mathrm{DFT}_2 \otimes \mathrm{I}_2) D (\mathrm{I}_2 \otimes \mathrm{DFT}_2) P$$

- SPIRAL generates the space of algorithms using breakdown rules in the domain-specific Signal Processing Language (SPL)

$$\mathrm{DFT}_{mn} \to (\mathrm{DFT}_m \otimes \mathrm{I}_n) D (\mathrm{I}_m \otimes \mathrm{DFT}_n) P$$

Some Transforms

$$
\begin{aligned}
\mathbf{DFT}_n &= [e^{-2k\ell\pi i/n}]_{0\le k,\ell<n} \\
\mathbf{RDFT}_n &= [r_{k\ell}]_{0\le k,\ell<n}, \quad r_{k\ell} = \begin{cases} \cos\frac{2\pi k\ell}{n}, & k \le \lfloor\frac{n}{2}\rfloor \\ -\sin\frac{2\pi k\ell}{n}, & k > \lfloor\frac{n}{2}\rfloor \end{cases}, \\
\mathbf{DCT\text{-}2}_n &= \left[\cos(k(2\ell+1)\pi/2n)\right]_{0\le k,\ell<n}, \\
\mathbf{DCT\text{-}3}_n &= \mathbf{DCT\text{-}2}_n^T \quad \text{(transpose)}, \\
\mathbf{DCT\text{-}4}_n &= \left[\cos((2k+1)(2\ell+1)\pi/4n)\right]_{0\le k,\ell<n}, \\
\mathbf{IMDCT}_n &= \left[\cos((2k+1)(2\ell+1+n)\pi/4n)\right]_{0\le k<2n,0\le\ell<n}, \\
\mathbf{WHT}_n &= \begin{bmatrix} \mathbf{WHT}_{n/2} & \mathbf{WHT}_{n/2} \\ \mathbf{WHT}_{n/2} & -\mathbf{WHT}_{n/2} \end{bmatrix}, \quad \mathbf{WHT}_2 = \mathbf{DFT}_2, \\
\mathbf{DHT} &= \left[\cos(2k\ell\pi/n) + \sin(2k\ell\pi/n)\right]_{0\le k,\ell<n}.
\end{aligned}
$$

Spiral currently contains 45 transforms

Some Breakdown Rules

$$
\begin{array}{rcl}
\mathbf{DFT}_n & \rightarrow & (\mathbf{DFT}_k \otimes \mathrm{I}_m)\, \mathsf{T}^n_m (\mathrm{I}_k \otimes \mathbf{DFT}_m)\, \mathsf{L}^n_k, \quad n = km \\
\mathbf{DFT}_n & \rightarrow & P_n (\mathbf{DFT}_k \otimes \mathbf{DFT}_m) Q_n, \quad n = km,\ \mathsf{gcd}(k,m) = 1 \\
\mathbf{DFT}_p & \rightarrow & R_p^T (\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1}) D_p (\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1}) R_p, \quad p \text{ prime} \\
\mathbf{DCT\text{-}3}_n & \rightarrow & (\mathrm{I}_m \oplus \mathsf{J}_m)\, \mathsf{L}^n_m (\mathbf{DCT\text{-}3}_m(1/4) \oplus \mathbf{DCT\text{-}3}_m(3/4)) \\
 & & \cdot (\mathsf{F}_2 \otimes \mathrm{I}_m) \begin{bmatrix} \mathrm{I}_m & 0 \oplus -\mathsf{J}_{m-1} \\ & \frac{1}{\sqrt{2}}(\mathrm{I}_1 \oplus 2\,\mathrm{I}_m) \end{bmatrix}, \quad n = 2m \\
\mathbf{DCT\text{-}4}_n & \rightarrow & S_n \mathbf{DCT\text{-}2}_n\, \mathsf{diag}_{0 \le k < n}(1/(2\cos((2k+1)\pi/4n))) \\
\mathbf{IMDCT}_{2m} & \rightarrow & (\mathsf{J}_m \oplus \mathrm{I}_m \oplus \mathrm{I}_m \oplus \mathsf{J}_m) \left(\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes \mathrm{I}_m \right) \oplus \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} \otimes \mathrm{I}_m \right) \right) \mathsf{J}_{2m}\, \mathbf{DCT\text{-}4}_{2m} \\
\mathbf{WHT}_{2^k} & \rightarrow & \prod_{i=1}^{t} (\mathrm{I}_{2^{k_1+\cdots+k_{i-1}}} \otimes \mathbf{WHT}_{2^{k_i}} \otimes \mathrm{I}_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t \\
\mathbf{DFT}_2 & \rightarrow & \mathsf{F}_2 \\
\mathbf{DCT\text{-}2}_2 & \rightarrow & \mathsf{diag}(1, 1/\sqrt{2})\, \mathsf{F}_2 \\
\mathbf{DCT\text{-}4}_2 & \rightarrow & \mathsf{J}_2\, \mathsf{R}_{13\pi/8}
\end{array}
$$

Base case rules

Spiral currently contains 165 rules

SPL (Signal Processing Language)

- SPL expresses transform algorithms as structured sparse matrix factorization

- Examples:

$$\mathrm{F}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad A \otimes B = \left[a_{k,\ell} B\right]_{k,\ell}$$

$$A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix} \qquad \boxed{\mathrm{I}_n \otimes B = \begin{bmatrix} B & & \\ & \ddots & \\ & & B \end{bmatrix}}$$

- Kronecker product = loop (parallel, vector)

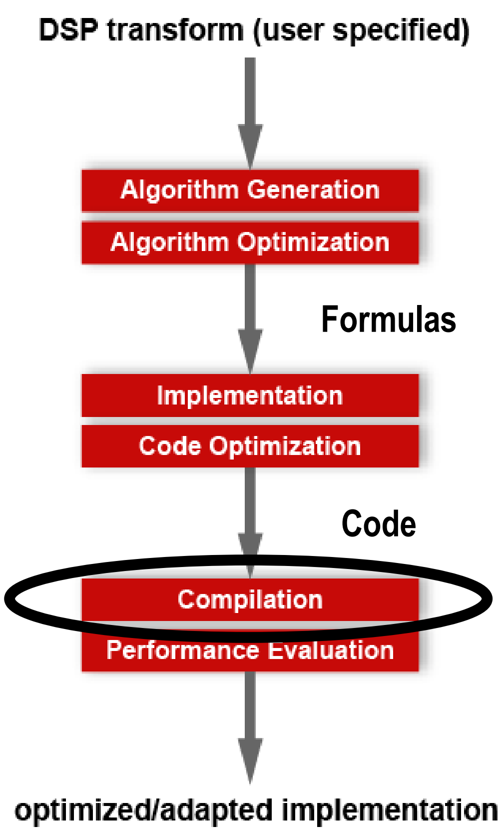
$$y = (\mathrm{I}_n \otimes B_{m\times m})x \quad \Longleftrightarrow$$

```
for i = 0:n-1
    y[im:im+m-1] = B·x[im:im+m-1]
endfor
```

Formula Level Optimization: Idea

Move optimizations to higher abstraction level:
Domain knowledge overcomes compiler limitations

Traditionally optimizations by C/Fortran compilers

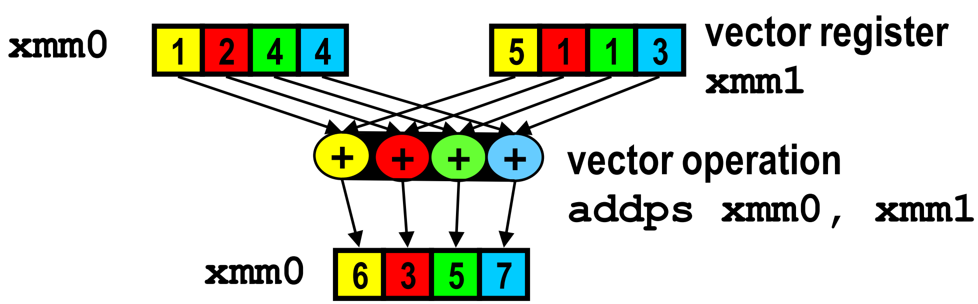


Formula level optimizations in Spiral:
Implemented through rewriting systems

- Loop merging
- **Vectorization**
- Parallelization

SIMD (Signal Instruction Multiple Data) Vector Instructions in a Nutshell

- **What are these instructions?**
  - Extension of the ISA. Data types and instructions for parallel computation on short **(2-way–16-way) vectors** of integers and floats



- Intel MMX
- AMD 3DNow!
- Intel SSE
- AMD Enhanced 3DNow!
- Motorola AltiVec
- AMD 3DNow! Professional
- Itanium
- Intel XScale
- Intel SSE2
- AMD-64
- IBM BlueGene/L PPC440FP2
- Intel Wireless MMX
- Intel SSE3
- ...

- **Problems:**
  - Not standardized
  - Compiler vectorization limited
  - Low-level issues (data alignment,…)
  - Reordering data kills runtime

One can easily slow down a program by vectorizing it

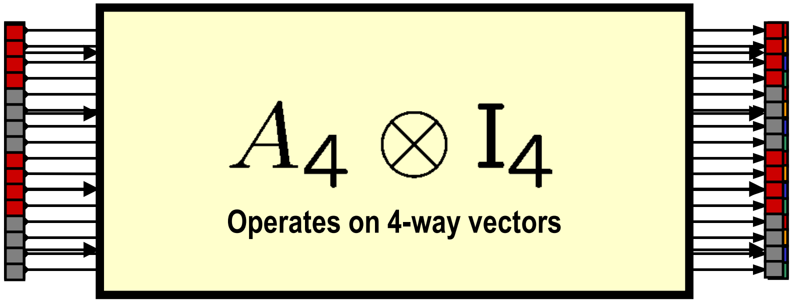
Vectorization of Formulas by Rewriting

- **Naturally vectorizable construct**

  Franchetti and Püschel (IPDPS 2002/2003)

$$y = (A \otimes \mathrm{I}_\nu)x$$

vector length (any two-power)



- **Rewriting rules to vectorize formulas**

  Introduces data reorganization (permutations)

$$\mathrm{I}_n \otimes A^{k\times m} \;\rightarrow\; \mathrm{I}_{n/\nu} \otimes \underline{\mathrm{L}^{k\nu}_{\nu}} \big(\underline{A^{k\times m} \otimes \mathrm{I}_\nu}\big) \underline{\mathrm{L}^{m\nu}_{m}}$$

$$\mathrm{L}^{m\nu}_{m} \;\rightarrow\; \big(\mathrm{I}_{m/\nu} \otimes \underline{\mathrm{L}^{\nu^2}_{\nu}}\big)\big(\underline{\mathrm{L}^{m}_{m/\nu} \otimes \mathrm{I}_\nu}\big)$$

vector construct

further rewriting

base case

Definition: ***Vectorized formula := vector constructs and base cases,***
A¢B*, and* IA *of vectorized formulas*

Example: DFT

$$\underbrace{\left(\overline{\mathbf{DFT}_{mn}}\right)}_{\mathrm{vec}(\nu)} \rightarrow \underbrace{\left(\overline{(\mathbf{DFT}_m \otimes \mathrm{I}_n)\,\mathrm{T}^{mn}_n\,(\mathrm{I}_m \otimes \mathbf{DFT}_n)\,\mathrm{L}^{mn}_m}\right)}_{\mathrm{vec}(\nu)}$$

$$\cdots$$

$$\rightarrow \overleftarrow{(\underbrace{\mathrm{DFT}_m \otimes \mathrm{I}_n}_{\mathrm{vec}(\nu)})}^{\nu}\;\overline{(\underbrace{\mathrm{T}^{mn}_n}_{\mathrm{vec}(\nu)})}^{\nu}\;\overrightarrow{\underbrace{(\mathrm{I}_m \otimes \mathrm{DFT}_n)\,\mathrm{L}^{mn}_m}_{\mathrm{vec}(\nu)}}^{\nu}$$

$$\cdots$$

$$\rightarrow (\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathrm{L}^{2\nu}_{\nu}}_{\mathrm{vec}(\nu)})(\overline{\mathrm{DFT}_m \otimes \mathrm{I}_{n/\nu}} \otimes_\nu \mathrm{I}_\nu)\overline{(\underbrace{\mathrm{T}^{mn}_n}_{\mathrm{vec}(\nu)})}^{\nu}$$

$$\left(\mathrm{I}_{m/\nu} \otimes (\overline{\mathrm{L}^{n}_{\nu}} \otimes_\nu \mathrm{I}_\nu)(\mathrm{I}_{n/\nu} \otimes (\mathrm{L}^{2\nu}_{\nu} \otimes_\nu \mathrm{I}_\nu)(\mathrm{I}_2 \otimes \underbrace{\mathrm{L}^{\nu^2}_{\nu}}_{\mathrm{vec}(\nu)})(\mathrm{L}^{2\nu}_{2} \otimes_\nu \mathrm{I}_\nu))(\overline{\mathrm{DFT}_n} \otimes_\nu \mathrm{I}_\nu)\right)$$

$$\left((\mathrm{L}^{mn}_{m} \otimes \mathrm{I}_2) \otimes_\nu \mathrm{I}_\nu\right)(\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathrm{L}^{2\nu}_{2}}_{\mathrm{vec}(\nu)})$$

vector constructs

base cases

Formula is vectorized w.r.t. Definition

Some Vectorization Rules

$$
\begin{aligned}
\underbrace{\left(\overline{A}\right)}_{\mathsf{vec}(\nu)} &\rightarrow \overleftrightarrow{\left(\underbrace{A}_{\mathsf{vec}(\nu)}\right)}^{\nu} \\
\overleftrightarrow{AB}^{\nu} &\rightarrow \overleftarrow{A}^{\nu}\,\overrightarrow{B}^{\nu} \\
\overleftarrow{AB}^{\nu} &\rightarrow \overleftarrow{A}^{\nu}\,\overline{B}^{\nu} \\
\underbrace{A \otimes \mathrm{I}_m}_{\mathsf{vec}(\nu)} &\rightarrow \left(A \otimes \mathrm{I}_{m/\nu}\right) \otimes_\nu \mathrm{I}_\nu \\
\underbrace{\left(\mathrm{I}_m \otimes A\right) \mathsf{L}^{mn}_{m}}_{\mathsf{vec}(\nu)} &\rightarrow \left(\mathrm{I}_{m/\nu} \otimes \underbrace{\mathsf{L}^{n\nu}_{\nu}}_{\mathsf{vec}(\nu)} \left(A \otimes_\nu \mathrm{I}_\nu\right)\right)\left(\mathsf{L}^{mn/\nu}_{m/\nu} \otimes_\nu \mathrm{I}_\nu\right) \\
\underbrace{\mathsf{L}^{n\nu}_{\nu}}_{\mathsf{vec}(\nu)} &\rightarrow \left(\mathsf{L}^{n}_{\nu} \otimes_\nu \mathrm{I}_\nu\right)\left(\mathrm{I}_{n/\nu} \otimes \underbrace{\mathsf{L}^{\nu^2}_{\nu}}_{\mathsf{vec}(\nu)}\right) \\
\overleftarrow{A \otimes_\nu \mathrm{I}_\nu}^{\nu} &\rightarrow (\mathrm{I}_{n/\nu} \otimes \underbrace{\mathsf{L}^{2\nu}_{\nu}}_{\mathsf{vec}(\nu)})(\overline{A} \otimes_\nu \mathrm{I}_\nu) \\
\overline{(\underbrace{\mathsf{L}^{\nu^2}_{\nu}}_{\mathsf{vec}(\nu)})}^{\nu} &\rightarrow (\mathsf{L}^{2\nu}_{\nu} \otimes_\nu \mathrm{I}_\nu)(\mathrm{I}_2 \otimes \underbrace{\mathsf{L}^{\nu^2}_{\nu}}_{\mathsf{vec}(\nu)})(\mathsf{L}^{2\nu}_{2} \otimes_\nu \mathrm{I}_\nu) \\
\overleftarrow{\mathrm{I}_m \otimes A}^{\nu} &\rightarrow \mathrm{I}_m \otimes \overleftarrow{A}^{\nu}
\end{aligned}
$$

Shared Memory Parallelization by Rewriting

$$\underbrace{\mathbf{DFT}_{mn}}_{\mathsf{smp}(p,\mu)} \to \underbrace{\Big((\mathbf{DFT}_m \otimes \mathrm{I}_n)\,\mathsf{T}^{mn}_n\,(\mathrm{I}_m \otimes \mathbf{DFT}_n)\,\mathsf{L}^{mn}_m\Big)}_{\mathsf{smp}(p,\mu)}$$

$$\dots$$

$$\to \underbrace{\Big(\mathbf{DFT}_m \otimes \mathrm{I}_n\Big)}_{\mathsf{smp}(p,\mu)}\ \underbrace{\mathsf{T}^{mn}_n}_{\mathsf{smp}(p,\mu)}\ \underbrace{\Big(\mathrm{I}_m \otimes \mathbf{DFT}_n\Big)}_{\mathsf{smp}(p,\mu)}\ \underbrace{\mathsf{L}^{nm}_m}_{\mathsf{smp}(p,\mu)}$$

$$\dots$$

$$\to \underline{\Big((\mathsf{L}^{mp}_m \otimes \mathrm{I}_{n/p\mu}) \otimes_\mu \mathrm{I}_\mu\Big)}\,\underline{\Big(\mathrm{I}_p \otimes_{\|} (\mathbf{DFT}_m \otimes \mathrm{I}_{n/p})\Big)}\,\underline{\Big((\mathsf{L}^{mp}_p \otimes \mathrm{I}_{n/p\mu}) \otimes_\mu \mathrm{I}_\mu\Big)}$$

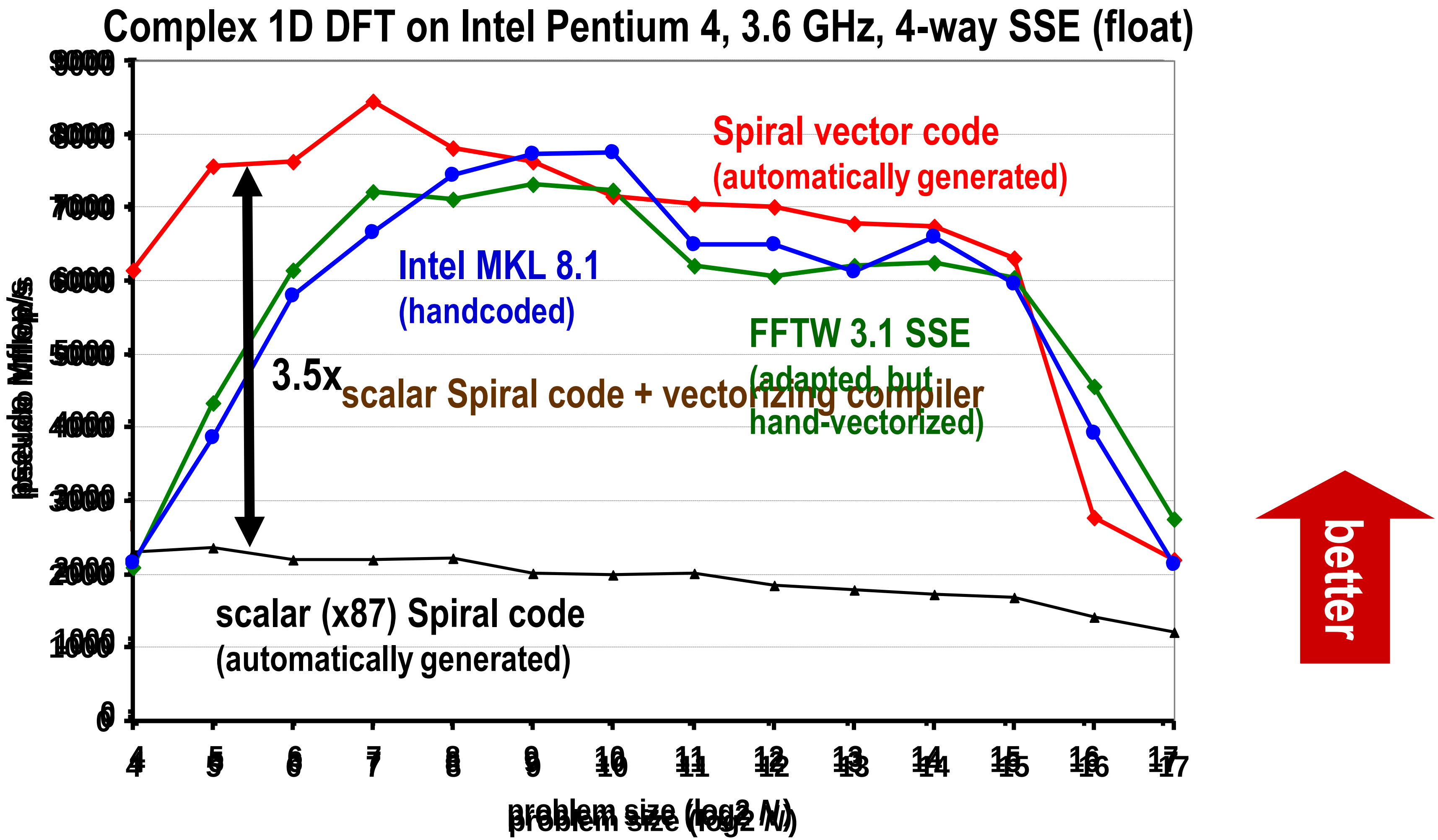
$$\underline{\Big(\bigoplus_{i=0}^{p-1}{}_{\|}\ \mathsf{T}^{mn,i}_n\Big)}\ \underline{\Big(\mathrm{I}_p \otimes_{\|} (\mathrm{I}_{m/p} \otimes \mathbf{DFT}_n)\Big)}\,\underline{\Big(\mathrm{I}_p \otimes_{\|} \mathsf{L}^{mn/p}_{m/p}\Big)}\,\underline{\Big((\mathsf{L}^{pn}_p \otimes \mathrm{I}_{m/p\mu}) \otimes_\mu \mathrm{I}_\mu\Big)}$$

Load balanced, contiguous blocks

No false sharing (entire cache lines are swapped)

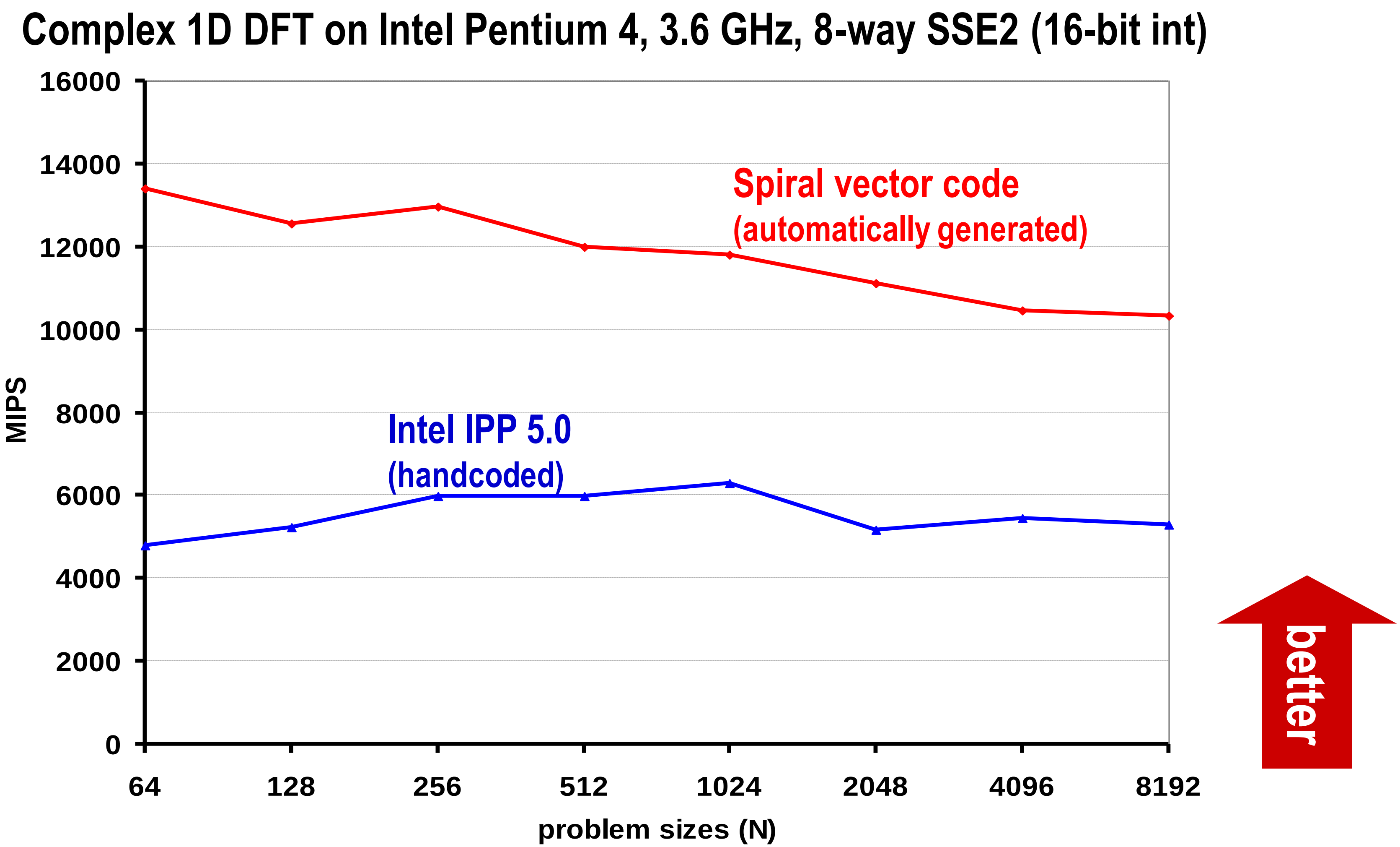
F. Franchetti, Y. Voronenko, and M. Püschel: "FFT Program Generation for Shared Memory: SMP and Multicore," to appear in SC|06

How Good is Our Generated Vector Code?



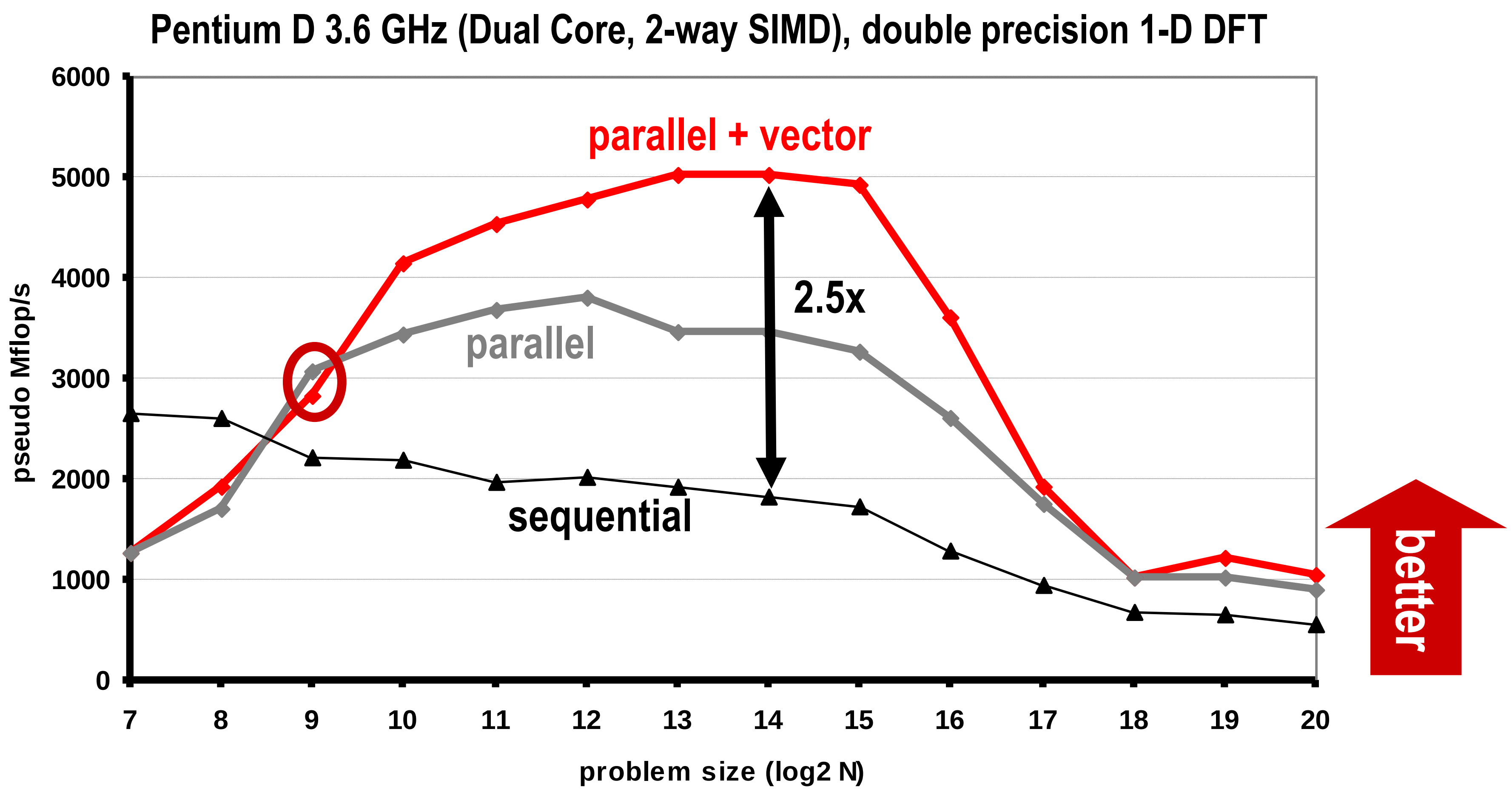
Spiral generated code performs comparable to expertly hand-tuned code

What About 8-way Vector Code?



Spiral generated code clearly outperforms expertly hand-tuned code

Combined Multicore and Vector Code



- *2.5x speed-up from parallel + vector*
- *Parallelization speed-up for small problems*

Summary

- **Parallelization and vectorization in Spiral**
  - **Entirely automatic**
  - **Principled approach**
  - **Rewriting system**
  - **Generated code is very fast**
- **Works for other hardware as well**
  - **Distributed memory: MPI**
    with C.W. Ueberhuber, A. Bonelli, and J. Lorenz, Vienna University of Technology
  - **Hardware: FPGAs**
    with J.C. Hoe and Peter Milder, Carnegie Mellon University

(Part of the) Spiral Team

www.spiral.net