Spiral: Program Generation for Linear Transforms and Beyond

Franz Franchetti

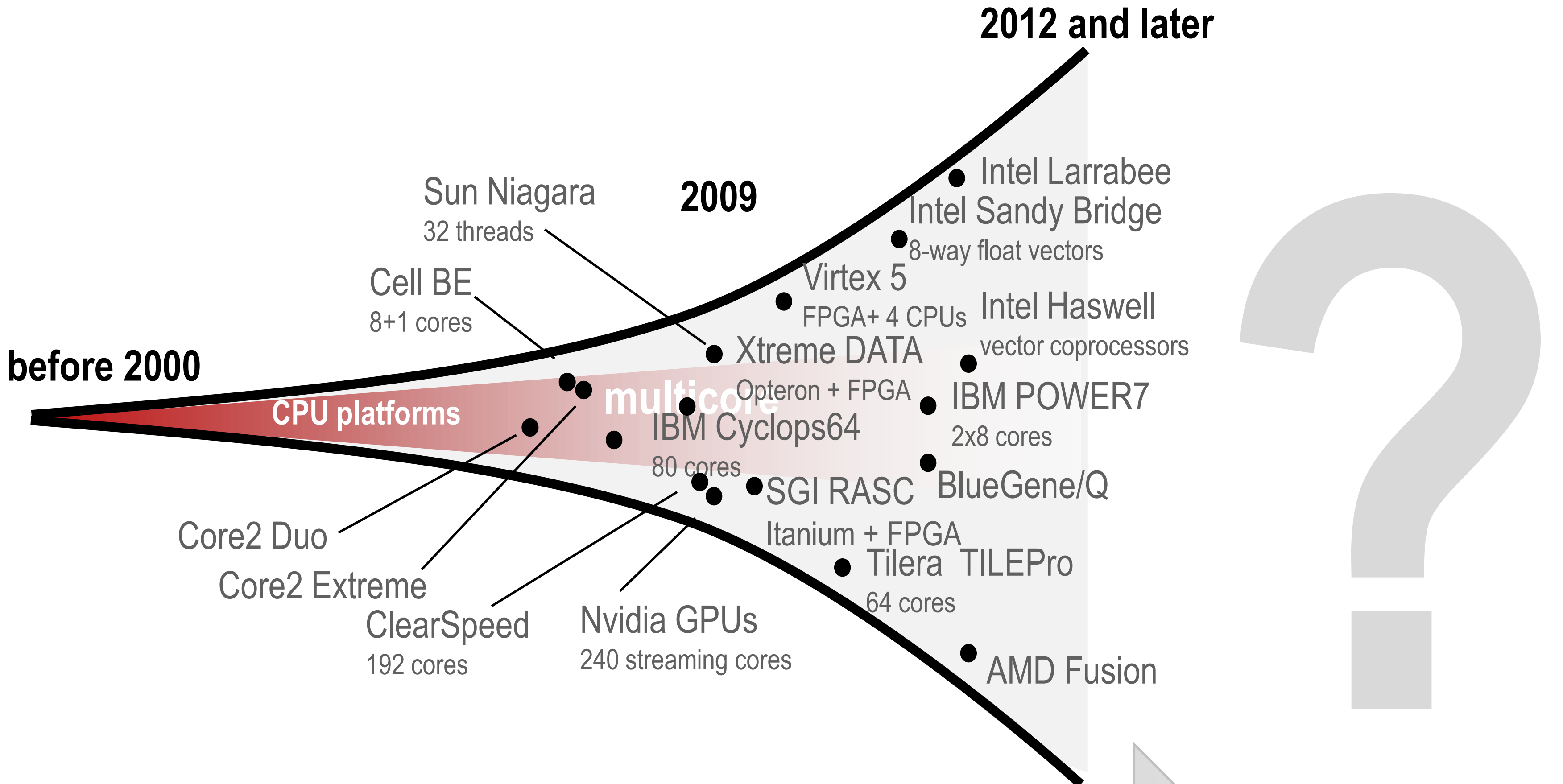
Electrical and
Computer Engineering
Carnegie Mellon University

Joint work with
Yevgen Voronenko
Frédéric de Mesmay
Daniel McFarlin
Markus Püschel

… and the Spiral team (only part shown)

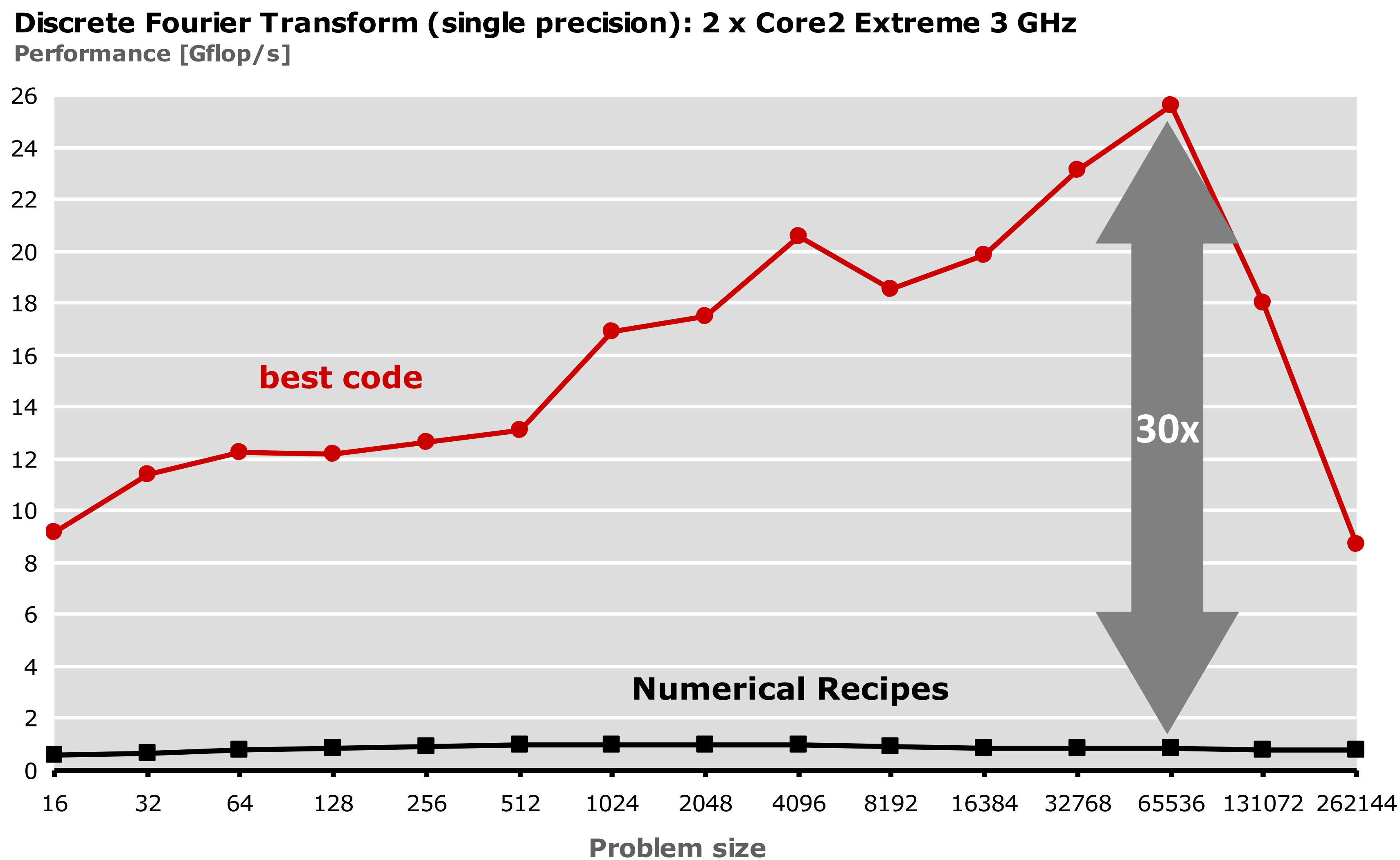
This work was supported by
DARPA DESA program, ONR, NSF-NGS/ITR, NSF-ACR, Mercury Inc., and Intel

The Future is Parallel and Heterogeneous



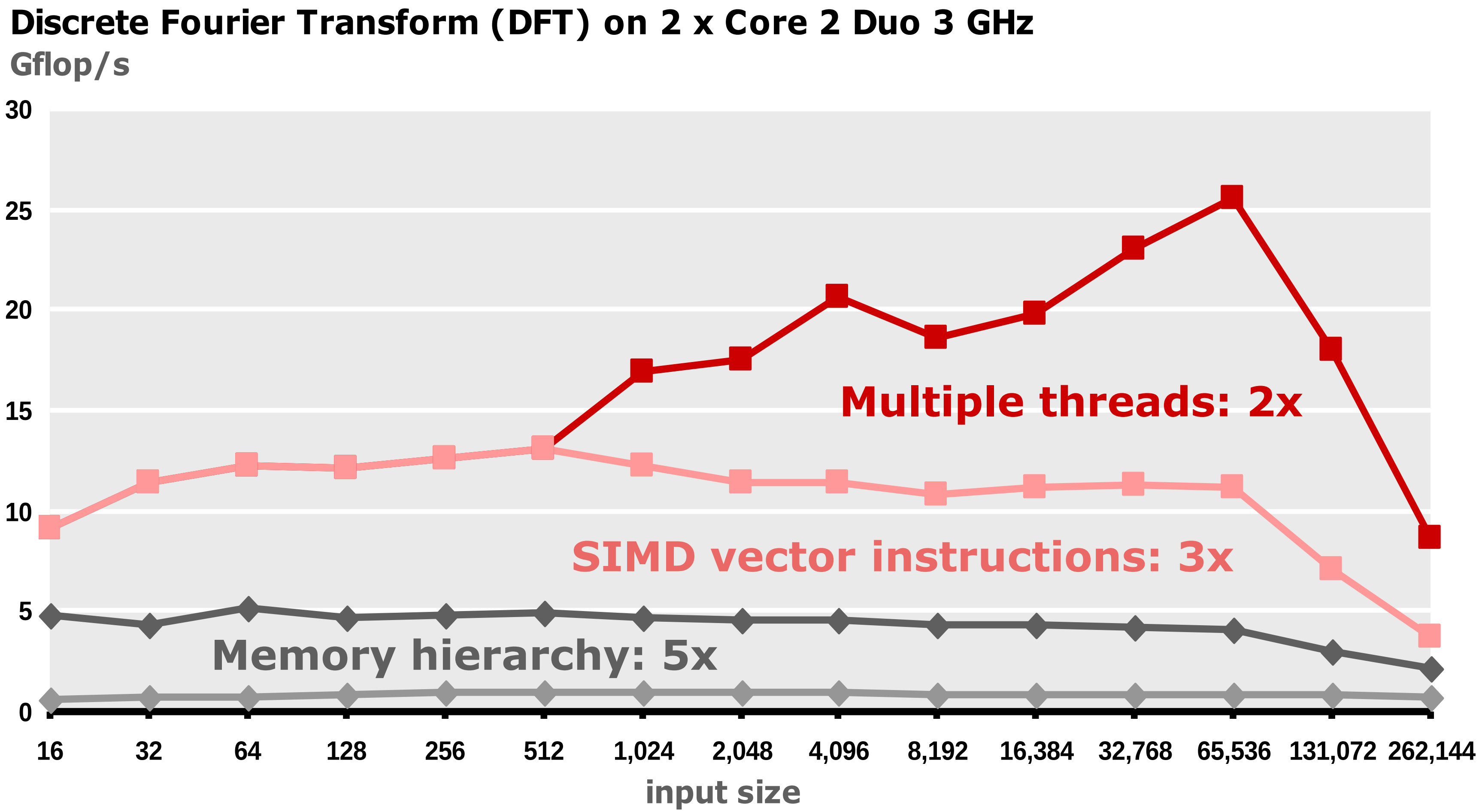
Programmability?
Performance portability?
Rapid prototyping?

The Problem



What's going on?

DFT Plot: Analysis



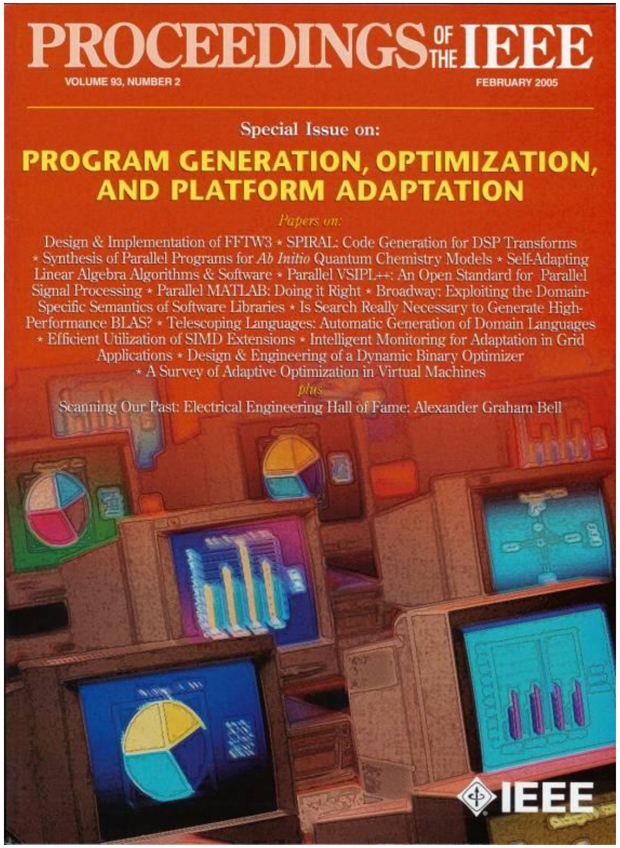
High performance library development has become a nightmare

Automatic Performance Tuning

- **Current vicious circle: Whenever a new platform comes out, the same functionality needs to be rewritten and reoptimized**

- **Automatic Performance Tuning**
  - BLAS: ATLAS, PHiPAC
  - Linear algebra: Sparsity/OSKI, Flame
  - Sorting
  - Fourier transform: FFTW
  - Linear transforms: Spiral
  - …others
  - New compiler techniques

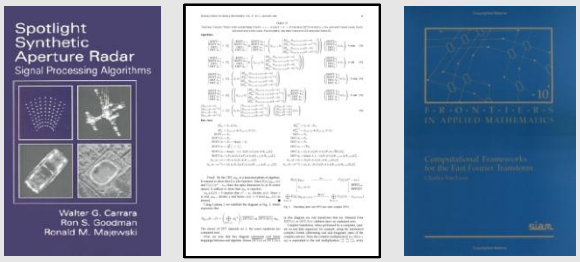
New challenge: ubiquitous parallelism



Proceedings of the IEEE special issue, Feb. 2005

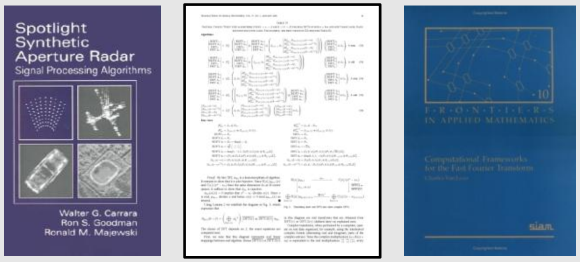
What is Spiral?

Traditionally



High performance library optimized for given platform

Spiral Approach



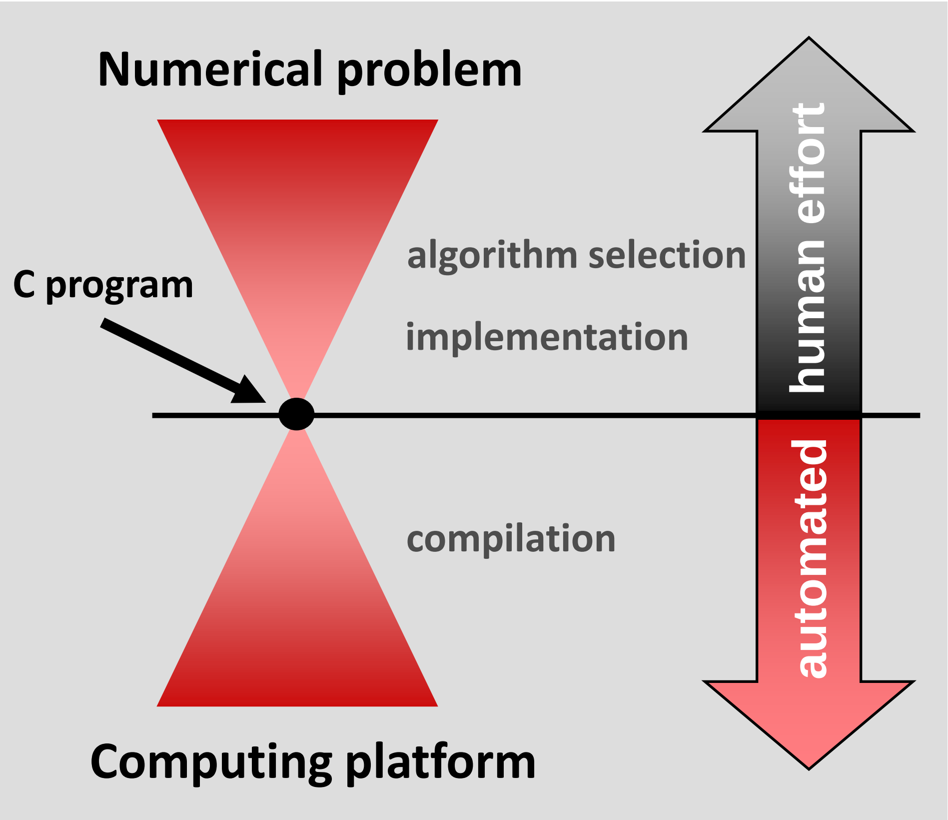
Spiral

High performance library optimized for given platform

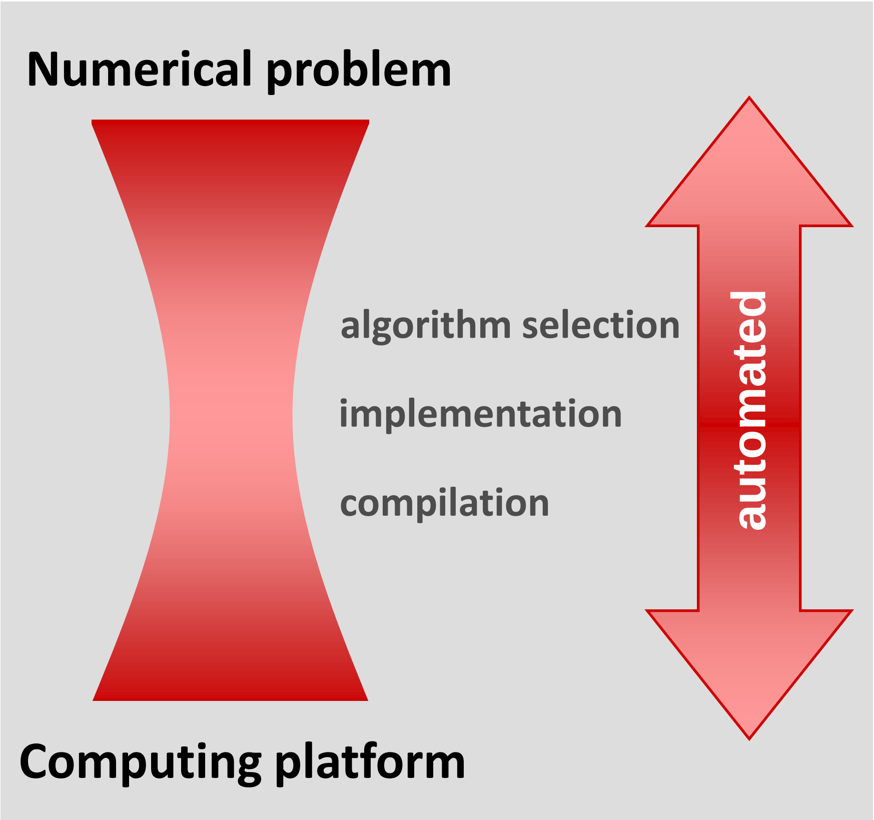
Comparable performance

Vision Behind Spiral

Current



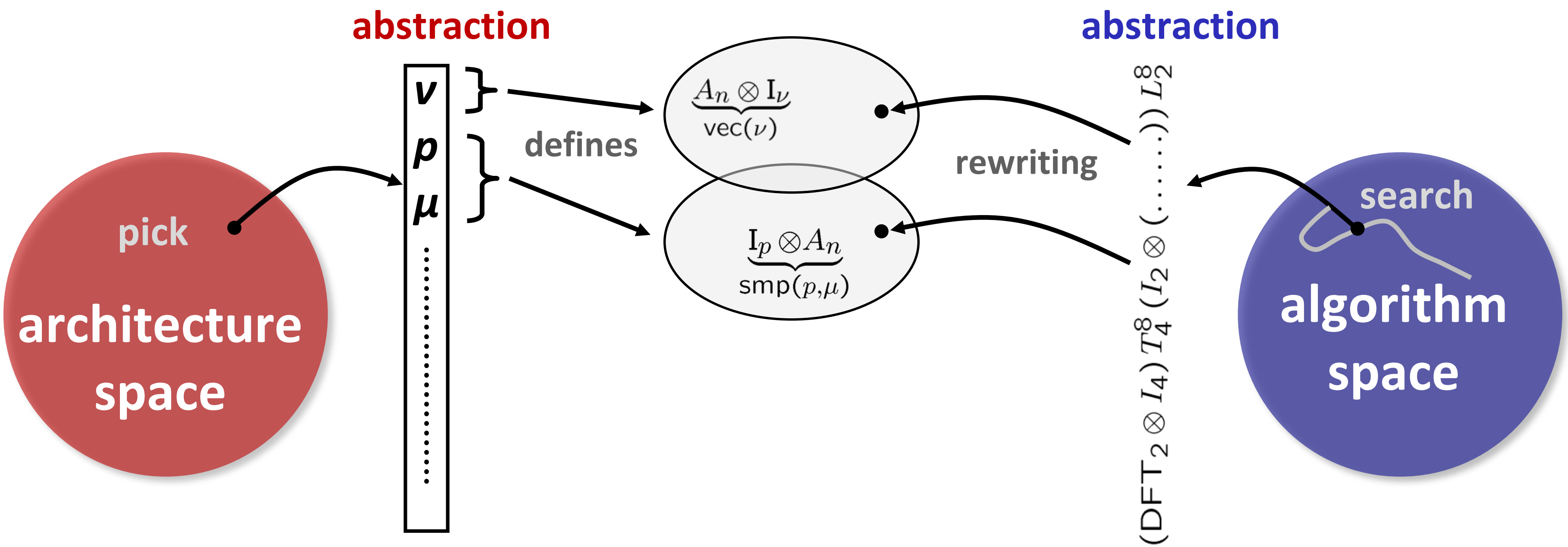
Future



- **C code a singularity: Compiler has no access to high level information**
- **Challenge: conquer the high abstraction level for complete automation**

Main Idea: Program Generation

Model: common abstraction
= spaces of matching formulas



Architectural parameter:
Vector length,
#processors, …

optimization

Kernel:
problem size,
algorithm choice

Organization

- **Spiral overview**
- Parallelization in Spiral
- Beyond Transforms
- Generating general-size libraries
- Results
- Concluding remarks

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.** Special issue, Proceedings of the IEEE 93(2), 2005

Franz Franchetti, Yevgen Voronenko, Markus Püschel: **Loop Merging for Signal Transforms.** In Proceedings of Programming Language Design and Implementation (PLDI) 2005.

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

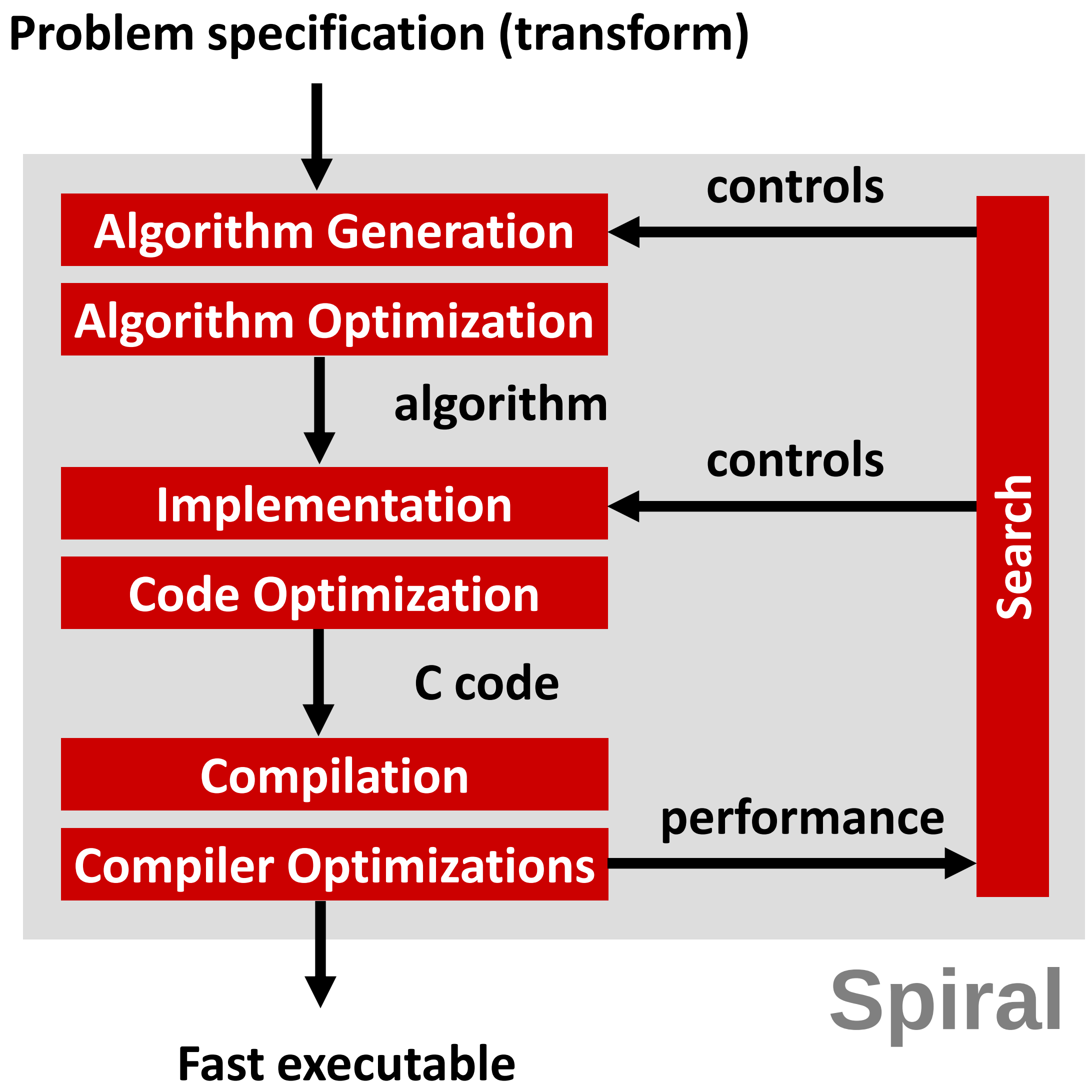
Intel uses Spiral to generate parts of their MKL and IPP libraries

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



What is a (Linear) Transform?

- **Mathematically: Matrix-vector multiplication**

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

input vector (signal)

output vector (signal)

transform = matrix

- **Example: Discrete Fourier transform (DFT)**

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

Transform Algorithms: Example 4-point FFT

Cooley/Tukey fast Fourier transform (FFT):

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 1 & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ 1 & \cdot & -1 & \cdot \\ \cdot & 1 & \cdot & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & j \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdot & \cdot \\ 1 & -1 & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

Fourier transform

Diagonal matrix (twiddles)

$$\mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathrm{I}_2)\, \mathsf{T}^4_2\, (\mathrm{I}_2 \otimes \mathrm{DFT}_2)\, \mathsf{L}^4_2$$

Kronecker product

Identity

Permutation

- Algorithms reduce arithmetic cost $O(n^2) \rightarrow O(n\log(n))$
- Product of structured sparse matrices
- Mathematical notation exhibits structure: SPL (signal processing language)

Examples: Transforms

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

Spiral currently contains 55 transforms

Examples: Breakdown Rules (currently ≈220)

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

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

DFT

DCT-3

DCT-4

IMDCT$_{2}$ … DCT-4$_{2m}$

- "Teaches" Spiral about existing algorithm knowledge
  (~200 journal papers)
- Includes many new ones
  (algebraic theory, Pueschel, Moura, Voronenko)

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

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

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

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

Base case rules

SPL to Sequential Code

SPL construct	code
$y = (A_n B_n)x$	`t[0:1:n-1] = B(x[0:1:n-1]);` `y[0:1:n-1] = A(t[0:1:n-1];)`

$y = (I_m \otimes A_n)x$	`for (i=0;i<m;i++)`
	`y[i*n:1:i*n+n-1] =`
	`A(x[i*n:1:i*n+n-1])`

$y = (A_m \otimes I_n)x$	`for (i=0;i<m;i++)`
	`y[i:n:i+m-1] =`
	`A(x[i:n:i+m-1]);`

$y = \left(\bigoplus_{i=0}^{m-1} A_n^i\right) x$	`for (i=0;i<m;i++)`
	`y[i*n:1:i*n+n-1] =`
	`A(i, x[i*n:1:i*n+n-1]);`

$y = D_{m,n}x$	`for (i=0;i<m*n;i++)` `y[i] = Dmn[i]*x[i];`

$y = L_m^{mn}x$	`for (i=0;i<m;i++)`
	`for (j=0;j<n;j++)`
	`y[i+m*j]=x[n*i+j];`

Example: tensor product

$$\mathrm{I}_m \otimes A_n = \begin{bmatrix} A_n & & \\ & \ddots & \\ & & A_n \end{bmatrix}$$

Program Generation in Spiral (Sketched)

Transform
user specified

$$\mathrm{DFT}_8$$

Fast algorithm in SPL
many choices

$$(\mathrm{DFT}_2 \otimes I_4)\, T^8_4\, (I_2 \otimes ((\mathrm{DFT}_2 \otimes I_2) \cdot T^4_2\, (I_2 \otimes \mathrm{DFT}_2)\, L^4_2))\, L^8_2$$

∑-SPL:

$$\sum \left(S_j\, \mathrm{DFT}_2\, G_j\right) \sum \left(\sum \left(S_{k,l}\, \mathrm{diag}\left(\mathrm{t}_{k,l}\right) \mathrm{DFT}_2\, G_l\right) \sum \left(S_m\, \mathrm{diag}\left(\mathrm{t}_m\right) \mathrm{DFT}_2\, G_{k,m}\right)\right)$$

C Code:

```
void sub(double *y, double *x) {
  double f0, f1, f2, f3, f4, f7, f8, f10, f11;
  f0 = x[0] - x[3];
  f1 = x[0] + x[3];
  f2 = x[1] - x[2];
  f3 = x[1] + x[2];
  f4 = f1 - f3;
  y[0] = f1 + f3;
  y[2] = 0.7071067811865476 * f4;
  f7 = 0.9238795325112867 * f0;
  f8 = 0.3826834323650898 * f2;
  y[1] = f7 + f8;
  f10 = 0.3826834323650898 * f0;
  f11 = (-0.9238795325112867) * f2;
  y[3] = f10 + f11;
}
```

Optimization at all abstraction levels

- parallelization
  vectorization
- loop
  optimizations
- constant folding
  scheduling
  ……

Spiral Web Interface @spiral.net

1. Select platform

2. Select functionality

3. Generate code "click"

Program Generation Interface collapse

Target platform for optimization: 2x Intel Xeon 3.6 GHz with 2048K L2 cache

parameter	value	explanation
Transform	DCT2 (Discrete Cosine Transform, type 2)	The transform for which you want to request C code
Data type	double	The data type of input and output vector
Transform size	6	The size of the transform = the length of the input vector
Optimize for	runtime	What you want to optimize the code for
Search method	Dynamic Programming	The search method SPIRAL uses (Dynamic Programming is a good choice)
Compiler profile	gcc -O3	Compiler and compiler options used for compilation

Generate code

Browse Archive expand

Filter by Platform: All Platforms Selected

Filter by Transform: All Transforms Selected

Filter by Size: All Sizes Selected

Query Database

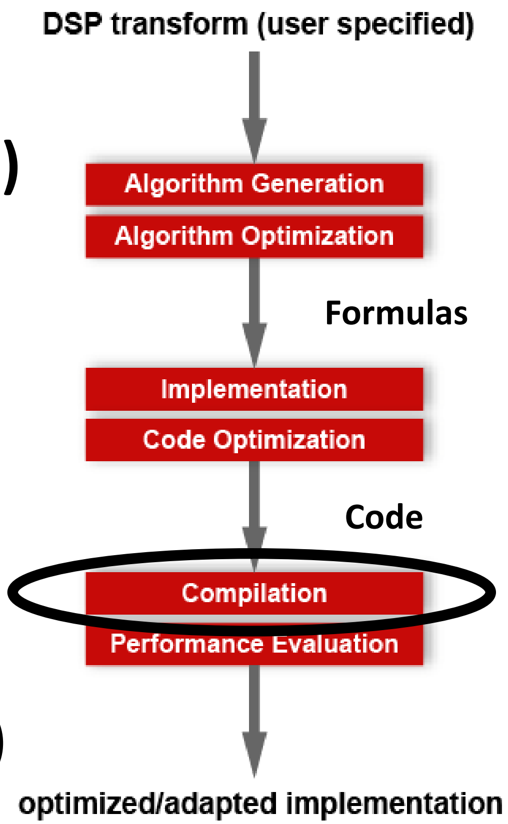
Organization

- Spiral overview
- **Parallelization in Spiral**
- Beyond Transforms
- Generating general-size libraries
- Results
- Concluding remarks

F. Franchetti, M. Püschel, Y. Voronenko, S. Chellappa, and J. M. F. Moura: **Discrete Fourier Transform Framework.** In IEEE Signal Processing Magazine, November 2009 (to appear).

Types of Parallelism

- Shared Memory (Multicore)
- Vector SIMD (SSE, VMX, Double FPU…)
- Message Passing (Clusters)
- Graphics Processors (GPUs)
- FPGA
- HW/SW partitioning
- Multiple Levels of Parallelism (Cell BE)

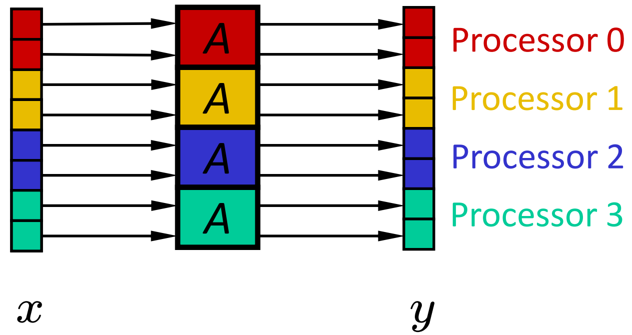


Spiral: One methodology optimizes for all types of parallelism

SPL to Shared Memory Code: Basic Idea

- Governing construct: tensor product

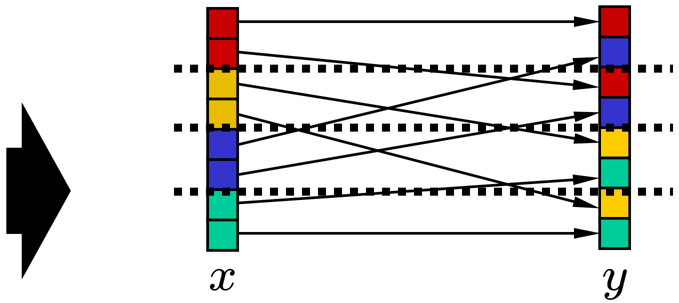
$$y = \left(\mathrm{I}_p \otimes A\right) x$$



Independent operation, load-balanced

- Problematic construct: permutations produce false sharing

$$y = \mathrm{L}^8_4\, x$$



Task: Rewrite formulas to
extract tensor product + keep contiguous blocks

Step 1: Shared Memory Tags

- **Identify crucial hardware parameters**
  - Number of processors: p
  - Cache line size: μ

- **Introduce them as tags in SPL**

$$\underbrace{A}_{\mathrm{smp}(p,\mu)}$$

This means: formula A is to be optimized for p processors and cache line size μ

Step 2: Identify "Good" Formulas

- **Load balanced, avoiding false sharing**

$$y = \left(\mathrm{I}_p \otimes A\right)x \quad \text{with} \quad A \in \mathbb{C}^{m\mu \times m\mu}$$

$$y = \left(\bigoplus_{i=0}^{p-1} A_i\right)x \quad \text{with} \quad A_i \in \mathbb{C}^{m\mu \times m\mu}$$

$$y = \left(P \otimes \mathrm{I}_\mu\right)x \quad \text{with } P \text{ a permutation matrix}$$

- **Tagged operators (no further rewriting necessary)**

$$\mathrm{I}_p \otimes_{\|} A, \quad \bigoplus_{i=0}^{p-1}{}_{\|} A_i, \quad P \bar{\otimes} \mathrm{I}_\mu$$

- **Definition: A formula is fully optimized if it is one of the above or of the form**

$$\mathrm{I}_m \otimes A \quad \text{or} \quad AB$$

where A and B are fully optimized.

SPIRAL www.spiral.net

Step 3: Identify Rewriting Rules

- **Goal: Transform formulas into fully optimized formulas**
  - Formulas rewritten, tags propagated
  - There may be choices

$$\underbrace{AB}_{\mathsf{smp}(p,\mu)} \rightarrow \underbrace{A}_{\mathsf{smp}(p,\mu)}\ \underbrace{B}_{\mathsf{smp}(p,\mu)}$$

$$\underbrace{A_m \otimes \mathrm{I}_n}_{\mathsf{smp}(p,\mu)} \rightarrow \underbrace{\left(\mathsf{L}_m^{mp} \otimes \mathrm{I}_{n/p}\right)\left(\mathrm{I}_p \otimes (A_m \otimes \mathrm{I}_{n/p})\right)\left(\mathsf{L}_p^{mp} \otimes \mathrm{I}_{n/p}\right)}_{\mathsf{smp}(p,\mu)}$$

$$\underbrace{\mathsf{L}_m^{mn}}_{\mathsf{smp}(p,\mu)} \rightarrow \begin{cases} \underbrace{\left(\mathrm{I}_p \otimes \mathsf{L}_{m/p}^{mn/p}\right)}_{\mathsf{smp}(p,\mu)} \underbrace{\left(\mathsf{L}_p^{pn} \otimes \mathrm{I}_{m/p}\right)}_{\mathsf{smp}(p,\mu)} \\ \underbrace{\left(\mathsf{L}_m^{pm} \otimes \mathrm{I}_{n/p}\right)}_{\mathsf{smp}(p,\mu)} \underbrace{\left(\mathrm{I}_p \otimes \mathsf{L}_m^{mn/p}\right)}_{\mathsf{smp}(p,\mu)} \end{cases}$$

$$\underbrace{\mathrm{I}_m \otimes A_n}_{\mathsf{smp}(p,\mu)} \rightarrow \mathrm{I}_p \otimes_{\parallel} \left(\mathrm{I}_{m/p} \otimes A_n\right)$$

$$\underbrace{(P \otimes \mathrm{I}_n)}_{\mathsf{smp}(p,\mu)} \rightarrow \left(P \otimes \mathrm{I}_{n/\mu}\right) \overline{\otimes}\, \mathrm{I}_\mu$$

Parallelization by Rewriting

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

$$\ldots$$

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

$$\ldots$$

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

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

**Fully optimized (load-balanced, no false sharing)
in the sense of our definition**

SPIRAL
www.spiral.net

Same Approach for Other Parallel Paradigms

Threading:

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

$$\dots$$

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

$$\dots$$

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

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

Vectorization:

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

$$\dots$$

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

$$\dots$$

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

$$\Big(\mathrm{I}_{m/\nu} \otimes (\overline{\mathrm{L}_\nu^n} \vec{\otimes} \mathrm{I}_\nu)(\mathrm{I}_{n/\nu} \otimes (\mathrm{L}_\nu^{2\nu} \vec{\otimes} \mathrm{I}_\nu)(\mathrm{I}_2 \otimes \underbrace{\mathrm{L}_\nu^{\nu^2}}_{\mathsf{sse}})(\mathrm{L}_2^{2\nu} \vec{\otimes} \mathrm{I}_\nu))(\overline{\mathrm{DFT}_n} \vec{\otimes} \mathrm{I}_\nu)\Big)$$

$$\Big((\mathrm{L}_m^{mn} \otimes \mathrm{I}_2) \vec{\otimes} \mathrm{I}_\nu\Big)(\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathrm{L}_2^{2\nu}}_{\mathsf{sse}})$$

GPUs:

$$\underbrace{\Big(\mathbf{DFT}_{r^k}\Big)}_{\mathsf{gpu}(t,c)} \rightarrow \underbrace{\left(\prod_{i=0}^{k-1} \mathrm{L}_r^{r^k}\Big(\mathrm{I}_{r^{k-1}} \otimes \mathbf{DFT}_r\Big)\Big(\mathrm{L}_{r^{k-i-1}}^{r^k}(\mathrm{I}_{r^i} \otimes \mathrm{T}_{r^{k-i-1}}^{r^{k-i}})\,\underbrace{\mathrm{L}_{r^{i+1}}^{r^k}}_{\mathsf{vec}(c)}\Big)\right)\mathrm{R}_r^{r^k}}_{\mathsf{gpu}(t,c)}$$

$$\dots$$

$$\rightarrow \left(\prod_{i=0}^{k-1} (\mathrm{L}_r^{r^n/2} \vec{\otimes} \mathrm{I}_2)\Big(\mathrm{I}_{r^{n-1}/2} \otimes_\times \underbrace{(\mathbf{DFT}_r \vec{\otimes} \mathrm{I}_2)\,\mathrm{L}_r^{2r}}_{\mathsf{shd}(t,c)}\Big)\mathrm{T}_i\right)$$

$$(\mathrm{L}_r^{r^n/2} \vec{\otimes} \mathrm{I}_2)(\mathrm{I}_{r^{n-1}/2} \otimes_\times \underbrace{\mathrm{L}_r^{2r}}_{\mathsf{shd}(t,c)})(\mathrm{R}_r^{r^{n-1}} \vec{\otimes} \mathrm{I}_r)$$

Verilog for FPGAs:

$$\underbrace{\Big(\mathbf{DFT}_{r^k}\Big)}_{\mathsf{stream}(r^s)} \rightarrow \underbrace{\left[\prod_{i=0}^{k-1} \mathrm{L}_r^{r^k}\Big(\mathrm{I}_{r^{k-1}} \otimes \mathbf{DFT}_r\Big)\Big(\mathrm{L}_{r^{k-i-1}}^{r^k}(\mathrm{I}_{r^i} \otimes \mathrm{T}_{r^{k-i-1}}^{r^{k-i}})\,\mathrm{L}_{r^{i+1}}^{r^k}\Big)\right]\mathrm{R}_r^{r^k}}_{\mathsf{stream}(r^s)}$$

$$\dots$$

$$\rightarrow \left[\prod_{i=0}^{k-1} \underbrace{\mathrm{L}_r^{r^k}}_{\mathsf{stream}(r^s)} \underbrace{\Big(\mathrm{I}_{r^{k-1}} \otimes \mathbf{DFT}_r\Big)}_{\mathsf{stream}(r^s)} \underbrace{\Big(\mathrm{L}_{r^{k-i-1}}^{r^k}(\mathrm{I}_{r^i} \otimes \mathrm{T}_{r^{k-i-1}}^{r^{k-i}})\,\mathrm{L}_{r^{i+1}}^{r^k}\Big)}_{\mathsf{stream}(r^s)}\right] \underbrace{\mathrm{R}_r^{r^k}}_{\mathsf{stream}(r^s)}$$

$$\dots$$

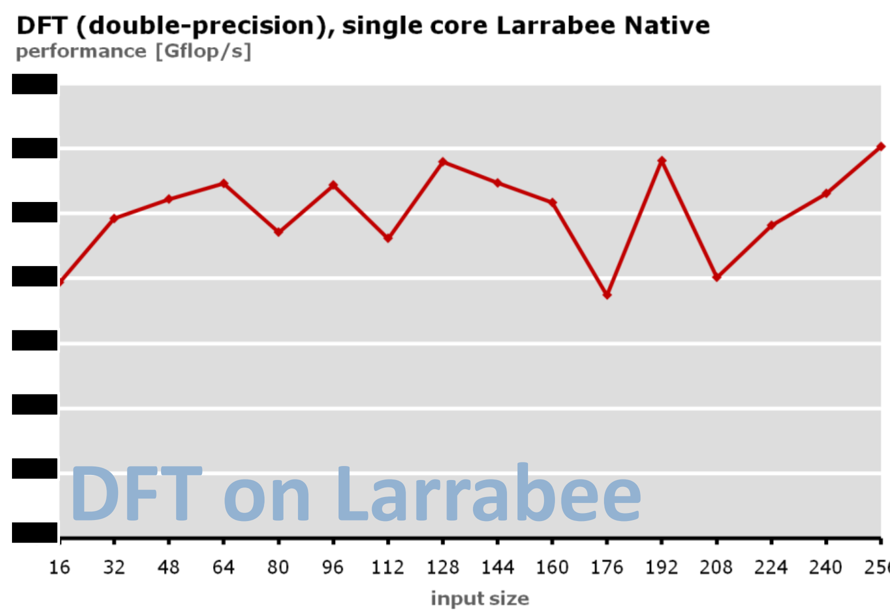
$$\rightarrow \left[\prod_{i=0}^{k-1} \underbrace{\mathrm{L}_r^{r^k}}_{\mathsf{stream}(r^s)} \Big(\mathrm{I}_{r^{k-s-1}} \otimes_s (\mathrm{I}_{r^{s-1}} \otimes \mathbf{DFT}_r)\Big) \underbrace{\mathrm{T}_i'}_{\mathsf{stream}(r^s)}\right] \underbrace{\mathrm{R}_r^{r^k}}_{\mathsf{stream}(r^s)}$$

- Rigorous, correct by construction
- Overcomes compiler limitations

SPIRAL www.spiral.net

Pre-Silicon Optimization: Larrabee and AVX

```
void dft64(float  *Y, float  *X) {
    __m512 U912, U913, U914, U915, U916, U917, U918, U919, U920, U921, U922, U923, U924, U925,...
 __m512  *a2153, *a2155;
    a2153 = ((__m512  *) X);  s1107 = *(a2153);
    s1108 = *((a2153 + 4));   t1323 = _mm512_add_ps(s1107,s1108);
    t1324 = _mm512_sub_ps(s1107,s1108);
    ...
    U926 = _mm512_swizupconv_r32(_mm512_set_1to16_ps(0.70710678118654757),_MM_SWIZ_REG_CDAB);
    s1121 = _mm512_madd231_ps(_mm512_mul_ps(_mm512_mask_or_pi(
        _mm512_set_1to16_ps(0.70710678118654757),0xAAAA,a2154,U926),t1341),
        _mm512_mask_sub_ps(_mm512_set_1to16_ps(0.70710678118654757),0x5555,a2154,U926),
        _mm512_swizupconv_r32(t1341,_MM_SWIZ_REG_CDAB));
    U927 = _mm512_swizupconv_r32(_mm512_set_16to16_ps(0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.7071067811865475
        0.70710678118654757, (-0.7071067811865475
    ...
    s1166 = _mm512_madd231_ps(_mm512_mul_ps(_mm51
        0.70710678118654757, (-0.7071067811865475
        0.70710678118654757, (-0.7071067811865475
        0.70710678118654757, (-0.7071067811865475
        0.70710678118654757, (-0.7071067811865475
        0xAAAA,a2154,U951),t1362),
        _mm512_mask_sub_ps(_mm512_set_16to16_ps(0
        (-0.70710678118654757), 0.707106781186547
        (-0.70710678118654757), 0.707106781186547
        (-0.70710678118654757), 0.707106781186547
        (-0.70710678118654757), 0.707106781186547
        _mm512_swizupconv_r32(t1362,_MM_SWIZ_REG_
    ...
}
```



Not actual data (NDA)

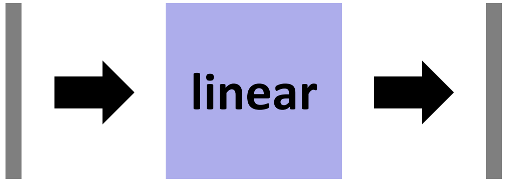
Organization

- Spiral overview
- Parallelization in Spiral
- **Beyond Transforms**
- Generating general-size libraries
- Results
- Concluding remarks

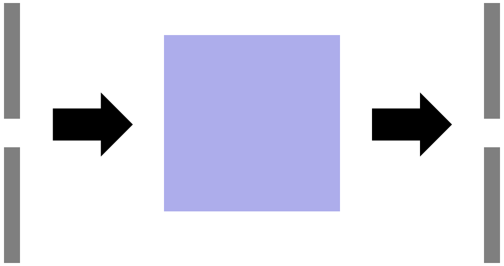
F. Franchetti, F. de Mesmay, D. McFarlin, and M. Püschel:
Operator Language: A Program Generation Framework for Fast Kernels. In Proceedings of DSL WC, 2009.

Going Beyond Transforms

- **Transform =**
  linear operator with one vector input and one vector output



- **Key ideas:**
  - Generalize to (possibly nonlinear) operators with several inputs and several outputs
  - Generalize SPL (including tensor product) to OL (operator language)
  - Generalize rewriting systems for parallelizations



Operator Language

name	definition
basic operators	
projection	$\pi_{\mathbf{x}} : \mathbb{C}^m \times \mathbb{C}^n \to \mathbb{C}^m;\ (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}$
linear transform	$\mathsf{M} : \mathbb{C}^n \to \mathbb{C}^m; \mathbf{x} \mapsto M\mathbf{x}$
stride	$\mathsf{L}^{mn}_m : \mathbb{C}^{mn} \to \mathbb{C}^{mn};\ \mathbf{x} \mapsto \mathsf{L}^{mn}_m \mathbf{x}$
vector sum	$\Sigma_n : \mathbb{C}^n \to \mathbb{C};\ \mathbf{x} \mapsto \sum_{i=0}^{n-1} x_i$
vector minimum	$\min_n : \mathbb{C}^n \to \mathbb{C};\ \mathbf{x} \mapsto \min(x_0, \ldots, x_{n-1})$
constant vector	$\mathsf{C}_{\mathbf{c}} : \varnothing \to \mathbb{C}^n;\ () \mapsto \mathbf{c}$
operations	
addition	$(\mathsf{M} + \mathsf{N})(\mathbf{x}, \mathbf{y}) = \mathsf{M}(\mathbf{x}, \mathbf{y}) + \mathsf{N}(\mathbf{x}, \mathbf{y})$
multiplication	$(\mathsf{M} \cdot \mathsf{N})(\mathbf{x}, \mathbf{y}) = \mathsf{M}(\mathbf{x}, \mathbf{y}) \cdot \mathsf{N}(\mathbf{x}, \mathbf{y})$
direct sum	$(\mathsf{M} \oplus \mathsf{N})(\mathbf{x} \oplus \mathbf{u}, \mathbf{y} \oplus \mathbf{v}) = \mathsf{M}(\mathbf{x}, \mathbf{y}) \oplus \mathsf{N}(\mathbf{u}, \mathbf{v})$
cartesian product	$(\mathsf{M} \times \mathsf{N})(\mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{v}) = \mathsf{M}(\mathbf{x}, \mathbf{y}) \times \mathsf{N}(\mathbf{u}, \mathbf{v})$
composition	$(\mathsf{M} \circ \mathsf{N})(\mathbf{x}, \mathbf{y}) = \mathsf{M}(\mathsf{N}(\mathbf{x}, \mathbf{y}))$
iterative composition	$\left(\prod_{i=0}^{n-1} \mathsf{M}_i\right)(\mathbf{x}, \mathbf{y}) = (\mathsf{M}_0 \circ \cdots \circ \mathsf{M}_{n-1})(\mathbf{x}, \mathbf{y})$
tensor product	$\mathrm{I} \otimes \mathsf{M},\ \mathsf{M} \otimes \mathrm{I}$

$$(\mathrm{I}_{m\times n\to mn} \otimes \mathsf{A}) \left(\sum_{i=0}^{m-1} \mathbf{e}_i^m \otimes \mathbf{x}, \sum_{i=0}^{n-1} \mathbf{e}_i^n \otimes \mathbf{y}\right) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \mathbf{e}_i^m \otimes \mathbf{e}_j^n \otimes \mathsf{A}(\mathbf{x}, \mathbf{y}),$$

$$(\mathsf{A} \otimes \mathrm{I}_{m\times n\to mn}) \left(\sum_{i=0}^{m-1} \mathbf{x} \otimes \mathbf{e}_i^m, \sum_{i=0}^{n-1} \mathbf{y} \otimes \mathbf{e}_i^n\right) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \mathsf{A}(\mathbf{x}, \mathbf{y}) \otimes \mathbf{e}_i^m \otimes \mathbf{e}_j^n.$$

Expressing Kernels as OL Formulas

Linear Transforms

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

Viterbi Decoding

$$\begin{aligned}
\underbrace{\mathbf{Vit}}_{\mathrm{vec}(v)} &\to \underbrace{\left(\prod (L \times I) \circ (I \otimes C)\right) \circ Id}_{\mathrm{vec}(v)} \\
&\to \left(\prod \underbrace{(L \times I) \circ (I \otimes C)}_{\mathrm{vec}(v)}\right) \circ Id \\
\times\quad &\to \left(\prod (L \otimes I_v \times I) \circ (I \otimes C \otimes I_v) \circ (\vec{L} \times I)\right) \circ Id \\
&\to \prod (L \otimes I_v \times I) \circ (I \otimes (B \otimes I_v)) \circ (\vec{L} \times I)
\end{aligned}$$

Matrix-Matrix Multiplication

$$\begin{aligned}
&\mathrm{MMM}_{1,1,1} \to (\cdot)_1 \\
&\mathrm{MMM}_{m,n,k} \to (\otimes)_{m/m_b \times 1} \otimes \mathrm{MMM}_{m_b,n,k} \\
&\mathrm{MMM}_{m,n,k} \to \mathrm{MMM}_{m,nb,k} \otimes (\otimes)_{1 \times n/nb} \\
&\mathrm{MMM}_{m,n,k} \to ((\Sigma_{k/k_b} \circ (\cdot)_{k/k_b}) \otimes \mathrm{MMM}_{m,n,k_b}) \circ \\
&\qquad\qquad ((L^{mk/k_b}_{k/k_b} \otimes I_{k_b}) \times I_{kn}) \\
&\mathrm{MMM}_{m,n,k} \to (L^{mn/n_b}_m \otimes I_{n_b}) \circ \\
&\qquad\qquad ((\otimes)_{1 \times n/n_b} \otimes \mathrm{MMM}_{m,n_b,k}) \circ \\
&\qquad\qquad (I_{km} \times (L^{kn/n_b}_{n/n_b} \otimes I_{n_b}))
\end{aligned}$$

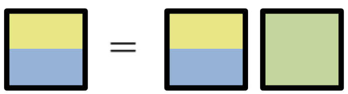
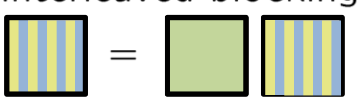
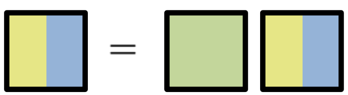
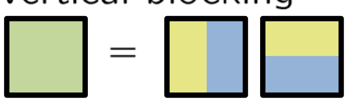
Synthetic Aperture Radar (SAR)

$$\begin{aligned}
\mathrm{SAR}_{k \times m \to n \times n} &\to \mathrm{DFT}_{n \times n} \circ \mathrm{Interp}_{k \times m \to n \times n} \\
\mathrm{DFT}_{n \times n} &\to (\mathrm{DFT}_n \otimes \mathrm{I}_n) \circ (\mathrm{I}_n \otimes \mathrm{DFT}_n) \\
\mathrm{Interp}_{k \times m \to n \times n} &\to (\mathrm{Interp}_{k \to n} \otimes_i \mathrm{I}_n) \circ (\mathrm{I}_k \otimes_i \mathrm{Interp}_{m \to n}) \\
\mathrm{Interp}_{r \to s} &\to \left(\bigoplus_{i=0}^{n-2} \mathrm{InterpSeg}_k\right) \oplus \mathrm{InterpSegPruned}_{k,\ell} \\
\mathrm{InterpSeg}_k &\to \mathrm{G}_f^{u \cdot n \to k} \circ \mathrm{iPrunedDFT}_{n \to u \cdot n} \circ \left(\frac{1}{n}\right) \circ \mathrm{DFT}_n
\end{aligned}$$

Example: Matrix Multiplication (MMM)

Breakdown rules:

capture various forms of blocking

breakdown rule	description
$\mathrm{MMM}_{1,1,1} \to (\cdot)_1$	base case
$\mathrm{MMM}_{m,n,k} \to (\otimes)_{m/m_b \times 1} \otimes \mathrm{MMM}_{m_b,n,k}$	horizontal blocking 
$\mathrm{MMM}_{m,n,k} \to \mathrm{MMM}_{m,nb,k} \otimes (\otimes)_{1 \times n/nb}$	interleaved blocking 
$\mathrm{MMM}_{m,n,k} \to ((\Sigma_{k/k_b} \circ (\cdot)_{k/k_b}) \otimes \mathrm{MMM}_{m,n,k_b}) \circ ((L^{mk/k_b}_{k/k_b} \otimes I_{k_b}) \times I_{kn})$	accumulative blocking 
$\mathrm{MMM}_{m,n,k} \to (L^{mn/n_b}_{m} \otimes I_{n_b}) \circ ((\otimes)_{1 \times n/n_b} \otimes \mathrm{MMM}_{m,n_b,k}) \circ (I_{km} \times (L^{kn/n_b}_{n/n_b} \otimes I_{n_b}))$	vertical blocking 

Parallelization: OL Rewriting Rules

- SPL rules and hardware model reused
- Few additional OL-specific rules required

$$\underbrace{\left(\mathrm{I}_k \otimes \mathsf{L}^{mn}_n\right)}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\mathsf{L}^{kmn}_{km}}_{\mathsf{smp}(p,\mu)} \rightarrow \left(\mathsf{L}^{kn}_k \otimes \mathrm{I}_{m/\mu}\right) \bar{\otimes} \mathrm{I}_\mu$$

$$\underbrace{\mathsf{L}^{kmn}_n}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\left(\mathrm{I}_k \otimes \mathsf{L}^{mn}_m\right)}_{\mathsf{smp}(p,\mu)} \rightarrow \left(\mathsf{L}^{kn}_n \otimes \mathrm{I}_{m/\mu}\right) \bar{\otimes} \mathrm{I}_\mu$$

$$\underbrace{\mathsf{A} \circ \mathsf{B}}_{\mathsf{smp}(p,\mu)} \rightarrow \underbrace{\mathsf{A}}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\mathsf{B}}_{\mathsf{smp}(p,\mu)}$$

$$\underbrace{\mathsf{A}^{k\times m\rightarrow n} \otimes \mathrm{I}^{1\times p\rightarrow p}}_{\mathsf{smp}(p,\mu)} \rightarrow \underbrace{\mathsf{L}^{pn}_n}_{\mathsf{smp}(p,\mu)} \circ \left(\mathrm{I}_{1\times p\rightarrow p} \otimes_{\|} \mathsf{A}^{k\times m\rightarrow n}\right) \circ \underbrace{\left(\mathrm{I}_k \times \mathsf{L}^{pm}_p\right)}_{\mathsf{smp}(p,\mu)}$$

$$\underbrace{\left(\mathsf{A} \times \mathsf{B}\right)}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\left(\mathsf{C} \times \mathsf{D}\right)}_{\mathsf{smp}(p,\mu)} \rightarrow \underbrace{\left(\mathsf{A} \circ \mathsf{C}\right)}_{\mathsf{smp}(p,\mu)} \times \underbrace{\left(\mathsf{B} \circ \mathsf{D}\right)}_{\mathsf{smp}(p,\mu)}$$

New OL rules

Parallelization Through Rewriting: MMM

$$\underbrace{\mathsf{MMM}_{m,n,k}}_{\mathsf{smp}(p,\mu)}$$

$$\to \underbrace{\left(\mathrm{I}_m \otimes \mathsf{L}^n_p\right) \circ \left(\mathsf{MMM}_{m,n/p,k} \otimes (\otimes)_{1\times p \to p}\right) \circ \left(\mathrm{I}_{km} \times (\mathrm{I}_k \otimes \mathsf{L}^n_{n/p})\right)}_{\mathsf{smp}(p,\mu)}$$

$$\to \underbrace{\left(\mathrm{I}_m \otimes \mathsf{L}^n_p\right)}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\left(\mathsf{MMM}_{m,n/p,k} \otimes (\otimes)_{1\times p \to p}\right)}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\left(\mathrm{I}_{km} \times (\mathrm{I}_k \otimes \mathsf{L}^n_{n/p})\right)}_{\mathsf{smp}(p,\mu)}$$

$$\to \underbrace{\left(\mathrm{I}_m \otimes \mathsf{L}^n_p\right)}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\mathsf{L}^{mn}_{m/pn}}_{\mathsf{smp}(p,\mu)} \circ \left((\otimes)_{1\times p \to p} \otimes_{\|} \mathsf{MMM}_{m/p,n,k}\right) \circ \underbrace{\left(\mathrm{I}_{km} \times \mathsf{L}^{kn}_p\right)}_{\mathsf{smp}(p,\mu)} \circ \underbrace{\left(\mathrm{I}_{km} \times (\mathrm{I}_k \otimes \mathsf{L}^n_{n/p})\right)}_{\mathsf{smp}(p,\mu)}$$

$$\to \left((\mathsf{L}^{mp}_m \otimes \mathrm{I}_{n/(p\mu)}) \bar{\otimes} \mathrm{I}_\mu\right) \circ \left((\otimes)_{1\times p \to p} \otimes_{\|} \mathsf{MMM}_{m,n/p,k}\right) \circ \left((\mathrm{I}_{km/\mu} \bar{\otimes} \mathrm{I}_\mu) \times ((\mathsf{L}^{kp}_p \otimes \mathrm{I}_{n/(p\mu)}) \bar{\otimes} \mathrm{I}_\mu)\right)$$

Load-balanced

No false sharing

Organization

- Spiral overview
- Parallelization in Spiral
- Beyond Transforms
- **Generating general-size libraries**
- Results
- Concluding remarks

Y. Voronenko, F. de Mesmay, and M. Püschel: **Computer generation of general size linear transform libraries** in Proceedings Code Generation and Optimization (CGO), 2009.

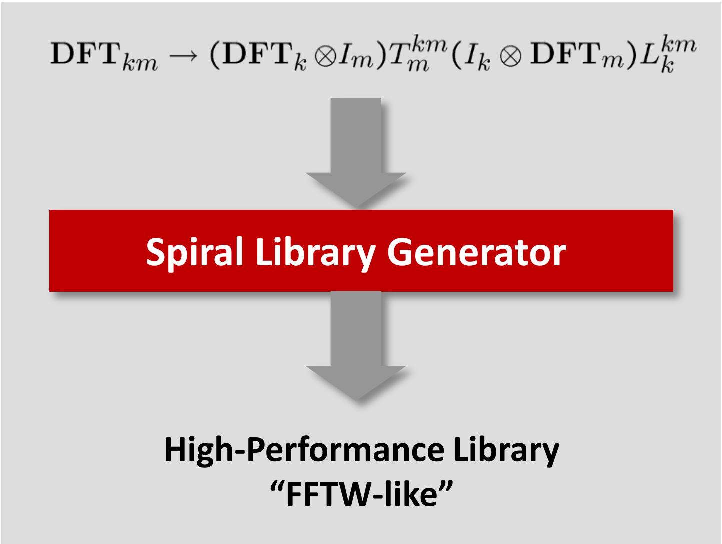
General-Size Library

Input:

- **Transform**: DFT_n
- **Algorithms**: $\mathrm{DFT}_{km} \rightarrow (\mathrm{DFT}_k \otimes I_m) T_m^{km} (I_k \otimes \mathrm{DFT}_m) L_k^{km}$
  $\mathrm{DFT}_2 \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- **Vectorization**: 2-way SSE
- **Threading**: Yes

Output:

- Optimized library (10,000 lines of C++)
- For general input size
  (**not** collection of fixed sizes)
- Vectorized
- Multithreaded
- With runtime adaptation mechanism
- Performance competitive with hand-written code



Beyond Fourier Transform and FFTW

Cooley-Tukey FFT → Spiral → "FFTW"

"Cooley-Tukey" DCT → Spiral → "FCTW"

Overlap-save/add FIR → Spiral → "FIRW"

Fast Walsh Transform → Spiral → "WHTW"

Fast Wavelet Transform → Spiral → "FWTW"

Fast Hartley Transform → Spiral → "FHTW"

Y. Voronenko's PhD Thesis (experimental results): 50+ "FFTW-like" libraries

How it Works: Recursion Step Closure

- **Input:** transform T and a breakdown rule
- **Output:** recursion steps + Σ-SPL implementation
- **Algorithm:**
  1. Apply the breakdown rule
  2. Convert to Σ-SPL
  3. Apply loop merging + index simplification rules.
  4. Extract recursion steps
  5. Repeat until closure is reached

$$\{\mathbf{DFT}_n\}$$

$$\downarrow$$

$$(\{\mathbf{DFT}_{n/k}\} \otimes I_k) T_k^n (I_{n/k} \otimes \{\mathbf{DFT}_k\}) L_{n/k}^n$$

$$\downarrow$$

$$\left(\sum_{i=0}^{k-1} \mathsf{S}_{h_{i,k}} \{\mathbf{DFT}_{n/k}\}\, \mathsf{G}_{h_{i,k}}\right) \mathsf{diag}(f) \left(\sum_{j=0}^{n/k-1} \mathsf{S}_{h_{jk,1}} \{\mathbf{DFT}_k\}\, \mathsf{G}_{h_{jk,1}}\right) \mathsf{perm}(\ell_{n/k}^n)$$

$$\downarrow$$

$$\sum_{i=0}^{k-1} \mathsf{S}_{h_{i,k}} \{\mathbf{DFT}_{n/k}\}\, \mathsf{diag}(f \circ h_{i,k})\, \mathsf{G}_{h_{i,k}} \sum_{j=0}^{n/k-1} \mathsf{S}_{h_{jk,1}} \{\mathbf{DFT}_k\}\, \mathsf{G}_{h_{j,n/k}}$$

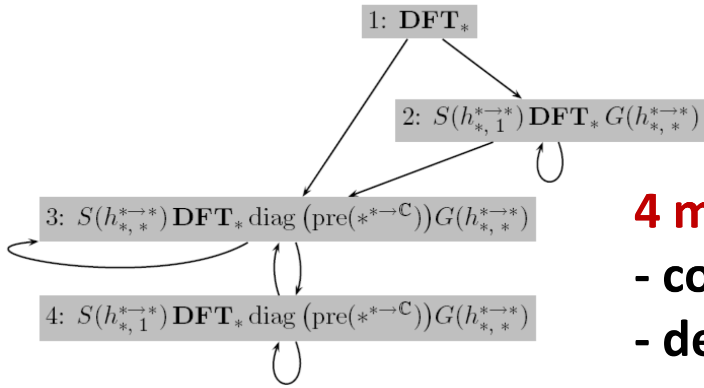
$$\downarrow$$

$$\sum_{i=0}^{k-1} \left\{ \mathsf{S}_{h_{i,k}}\, \mathbf{DFT}_{n/k}\, \mathsf{diag}(f \circ h_{i,k})\, \mathsf{G}_{h_{i,k}} \right\} \sum_{j=0}^{n/k-1} \left\{ \mathsf{S}_{h_{jk,1}}\, \mathbf{DFT}_k\, \mathsf{G}_{h_{j,n/k}} \right\}$$

Spiral translates recursion steps into adaptive library (codelets + recursion)

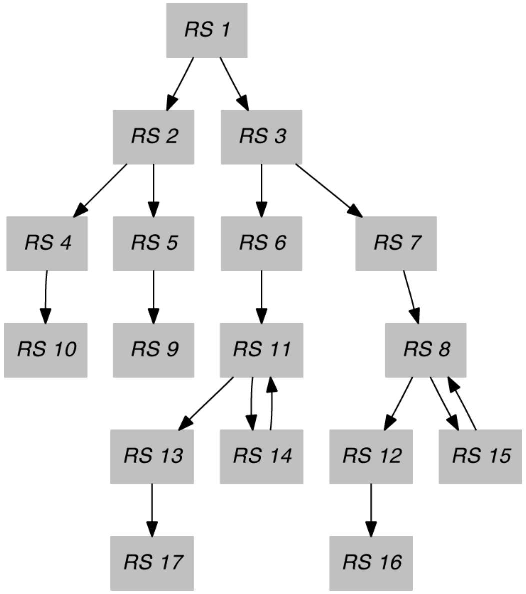
Recursion Step Closure: Examples

DFT (scalar)



4 mutually recursive functions
- computed automatically
- described using Σ-SPL formulas

DCT4 (vectorized)



$1:\ \mathrm{Vec}_2(\mathbf{DCT\text{-}4}_{u_1})$

$2:\ \mathrm{Vec}_2(\mathrm{GT}(\mathrm{diag}\,(N_{2u_8})\,\mathbf{RDFT\text{-}3}^{\top}_{2u_8}\,\mathrm{rcdiag}(\mathrm{pre}({u_4}^{\mathbb{Z}\times 2u_8\to\mathbb{R}})),\ h^{2u_8\to u_6}_{0,\,1,u_7}\circ\ell^{2u_8}_{u_8},\ r^{2u_8\to u_9}_{0,\,u_{11},1,\,u_{12}},\ \{u_{13}\}))$

$3:\ \mathrm{Vec}_2(\mathrm{GT}(\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1}),\ r^{u_1\to u_3}_{0,\,u_5,1,\,u_6},\ h^{u_1\to u_8}_{0,\,u_9,1},\ \{u_{10}\}))$

$4:\ \mathrm{VJam}_2(\mathrm{GT}(\mathrm{diag}\,(N_{2u_9})\,\mathbf{RDFT\text{-}3}^{\top}_{2u_9}\,\mathrm{rcdiag}(\mathrm{pre}({u_4}^{\mathbb{Z}\times\mathbb{Z}\times 2u_9\to\mathbb{R}})),\ h^{2u_9\to u_6}_{0,\,1,u_7,u_8}\circ\ell^{2u_9}_{u_9},\ r^{2u_9\to u_{10}}_{0,\,u_{12},1,2,\,u_{13}},\ \{2,u_{14}\}))$

$5:\ \mathrm{GT}(\mathrm{diag}\,(N_{2u_9})\,\mathbf{RDFT\text{-}3}^{\top}_{2u_9}\,\mathrm{rcdiag}(\mathrm{pre}({u_4}^{\mathbb{Z}\times 2u_9\to\mathbb{R}})),\ h^{2u_9\to u_6}_{u_7,\,1,u_8}\circ\ell^{2u_9}_{u_9},\ r^{2u_9\to u_{10}}_{u_{12},\,u_{13},1,\,u_{14}},\ \{u_{15}\})$

$6:\ \mathrm{VJam}_2(\mathrm{GT}(\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1}),\ r^{u_1\to u_3}_{0,\,u_5,1,2,\,u_6},\ h^{u_1\to u_8}_{0,\,u_9,1,2},\ \{2,u_{10}\}))$

$7:\ \mathrm{GT}(\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1}),\ r^{u_1\to u_3}_{u_5,\,u_6,1,\,u_7},\ h^{u_1\to u_9}_{u_{10},\,u_{11},1},\ \{u_{12}\})$

$8:\ \mathrm{S}(h^{u_1\to u_2}_{u_3,\,u_4})\,\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1})\mathrm{G}(r^{u_1\to u_7}_{u_9,\,u_{10},\,u_{11}})$

$9:\ \mathrm{S}(r^{2u_{13}\to u_1}_{u_3,\,u_4,\,u_5})\mathrm{diag}\,(N_{2u_{13}})\,\mathbf{RDFT\text{-}3}^{\top}_{2u_{13}}\,\mathrm{rcdiag}(\mathrm{pre}({u_9}^{2u_{13}\to\mathbb{R}}))\mathrm{G}(h^{2u_{13}\to u_{11}}_{u_{12},\,1}\circ\ell^{2u_{13}}_{u_{13}})$

$10:\ \mathrm{VJam}_2(\mathrm{GT}(\mathrm{diag}\,(N_{2u_9})\,\mathbf{RDFT\text{-}3}^{\top}_{2u_9}\,\mathrm{rcdiag}(\mathrm{pre}({u_4}^{\mathbb{Z}\times 2u_9\to\mathbb{R}})),\ h^{2u_9\to u_6}_{u_7,\,1,u_8}\circ\ell^{2u_9}_{u_9},\ r^{2u_9\to u_{10}}_{u_{12},\,u_{13},1,\,u_{14}},\ \{2\}))$

$11:\ \mathrm{VJam}_2(\mathrm{GT}(\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1}),\ r^{u_1\to u_3}_{u_5,\,u_6,1,\,u_7},\ h^{u_1\to u_9}_{u_{10},\,u_{11},1},\ \{2\}))$

$12:\ \mathrm{GT}(\mathrm{diag}\,(C_{u_1})\,\mathbf{rDFT}_{2u_1}(\lambda\text{-wrap}({\lambda_1}^{\mathbb{Z}\times\mathbb{Z}\to\mathbb{R}})),\ h^{2u_1\to u_4}_{0,\,1,u_5},\ h^{2u_{10}\to u_7}_{u_8,\,u_9}\circ(r^{u_1\to u_{10}}_{0,\,u_{12},1,\,u_{13}}\otimes\imath_2),\ \{u_{14}\})$

$13:\ \mathrm{VJam}_2(\mathrm{GT}(\mathrm{diag}\,(C_{u_1})\,\mathbf{rDFT}_{2u_1}(\lambda\text{-wrap}({\lambda_1}^{\mathbb{Z}\times\mathbb{Z}\times\mathbb{Z}\to\mathbb{R}})),\ h^{2u_1\to u_4}_{u_5,\,u_6,1,u_7},\ h^{2u_{12}\to u_9}_{u_{10},\,u_{11},1}\circ(r^{u_1\to u_{12}}_{0,\,u_{14},0,1,\,u_{15}}\otimes\imath_2),\ \{2,u_{16}\}))$

$14:\ \mathrm{VJam}_2(\mathrm{GT}(\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1}),\ r^{u_1\to u_3}_{u_5,\,u_6,1,u_7,\,u_8},\ h^{u_1\to u_{10}}_{u_{11},\,u_{12},1,u_{13}},\ \{2,u_{14}\}))$

$15:\ \mathrm{GT}(\mathbf{RDFT\text{-}3}_{u_1}\,\mathrm{diag}\,(N_{u_1}),\ r^{u_1\to u_3}_{u_5,\,u_6,u_7,\,u_8},\ h^{u_1\to u_{10}}_{0,\,u_{11},1},\ \{u_{12}\})$

$16:\ \mathrm{S}(h^{2u_5\to u_2}_{u_3,\,u_4}\circ(r^{u_6\to u_5}_{u_7,\,u_8,\,u_9}\otimes\imath_2))\mathrm{diag}\,(C_{u_6})\,\mathbf{rDFT}_{2u_6}(\lambda\text{-wrap}({\lambda_1}^{\mathbb{Z}\to\mathbb{R}}))\mathrm{G}(h^{2u_6\to u_{13}}_{u_{14},\,1})$

$17:\ \mathrm{VJam}_2(\mathrm{GT}(\mathrm{diag}\,(C_{u_1})\,\mathbf{rDFT}_{2u_1}(\lambda\text{-wrap}({\lambda_1}^{\mathbb{Z}\times\mathbb{Z}\to\mathbb{R}})),\ h^{2u_1\to u_4}_{u_5,\,u_6,1},\ h^{2u_{11}\to u_8}_{u_9,\,u_{10},1}\circ(r^{u_1\to u_{11}}_{u_{13},\,u_{14},\,u_{15}}\otimes\imath_2),\ \{2\}))$

17 mutually recursive functions

Organization

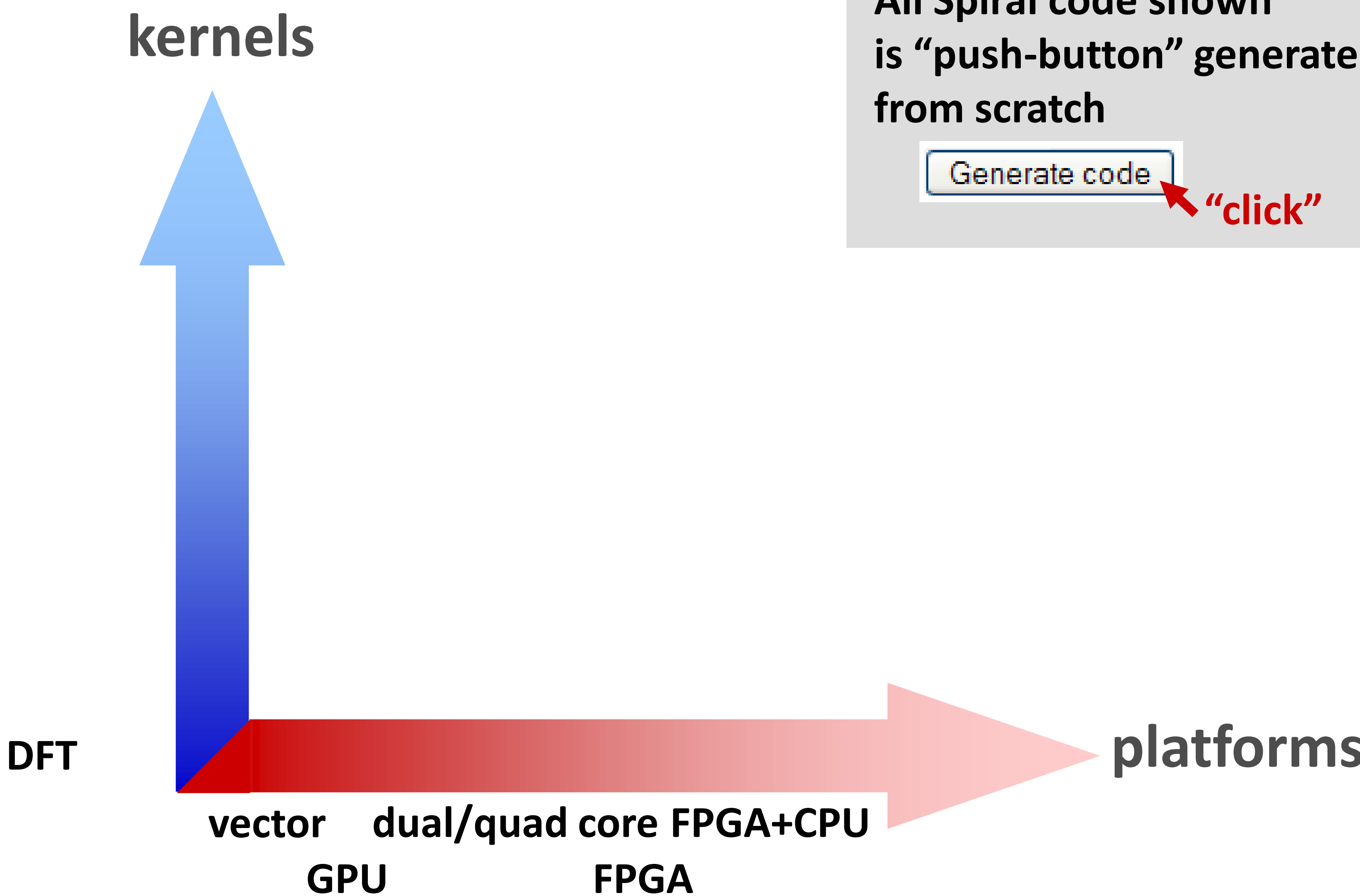
- Spiral overview
- Parallelization in Spiral
- Beyond Transforms
- Generating general-size libraries
- **Results**
- Concluding remarks

Benchmarks

All Spiral code shown
is “push-button” generated
from scratch

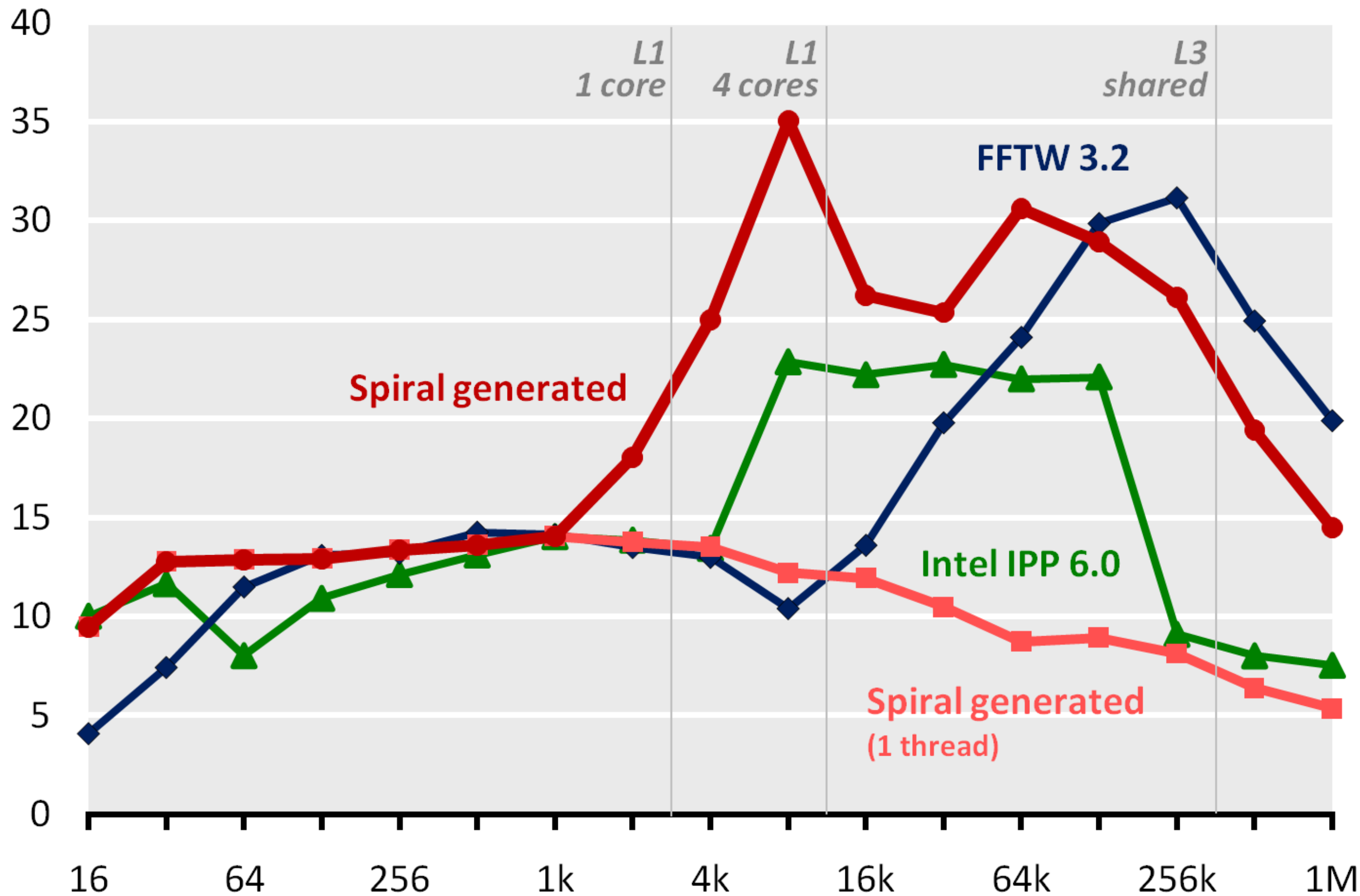
Generate code

“click”



Complex DFT (Intel Core i7, 2.66 GHz, 4 cores, single precision)

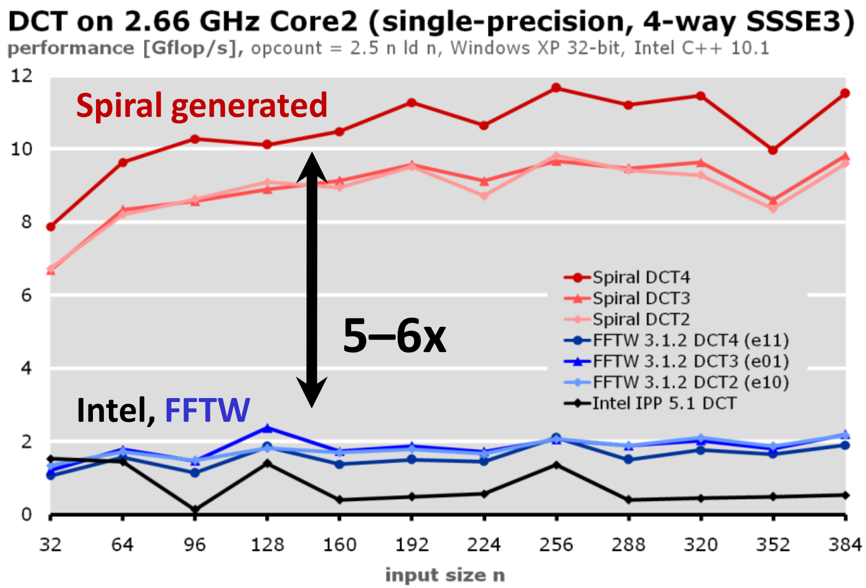
performance [Gflop/s] vs. input size

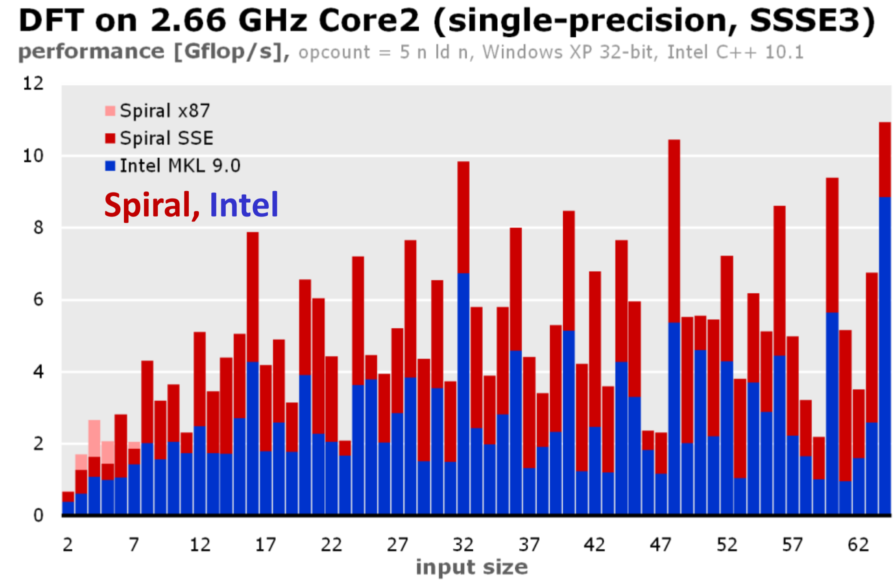


F. Franchetti, M. Püschel: **Short Vector Code Generation for the Discrete Fourier Transform.** In Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS '03).

F. Franchetti, Y. Voronenko, and M. Püschel: **FFT Program Generation for Shared Memory: SMP and Multicore.** In Proceedings of Supercomputing, 2006.

Intel Multicore: Off The Beaten Path



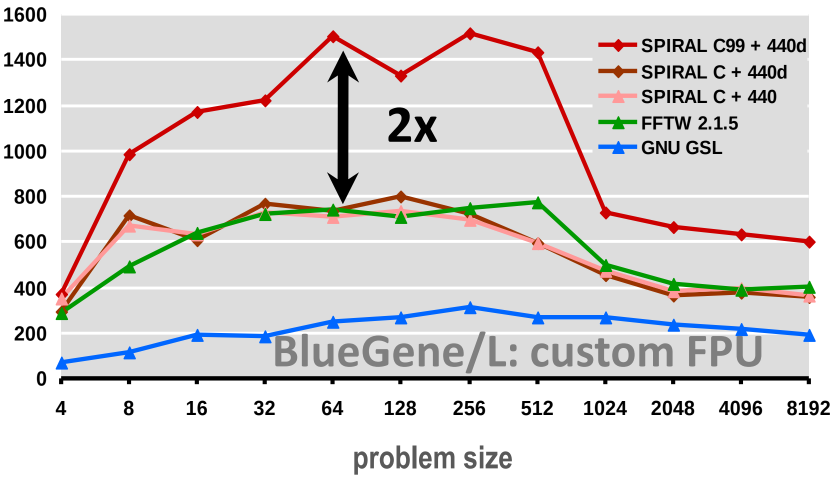


- **DCT: Native algorithm (Spiral) vs. FFT translation (FFTW, MKL)**
  Algorithms developed with the Algebraic Signal Processing theory
- **DFT: SIMD-specific aggressive data layout optimization**
  Included in IPP 6.0 (new domain: IPPGen)

F. Franchetti, M. Püschel:
SIMD Vectorization of Non-Two-Power Sized FFTs.
Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2007.

Single Node: BlueGene Supercomputers

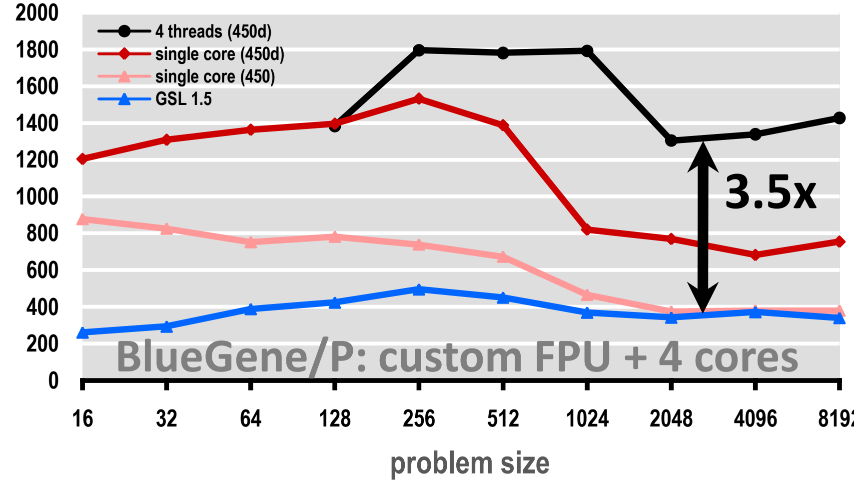
DFT, double precision, XL C compiler
performance [Mflop/s]



Single BlueGene/L CPU at 700 MHz
IBM T. J. Watson Research Center

SIMD vectorization

DFT, double precision, XL C compiler
performance [Mflop/s]



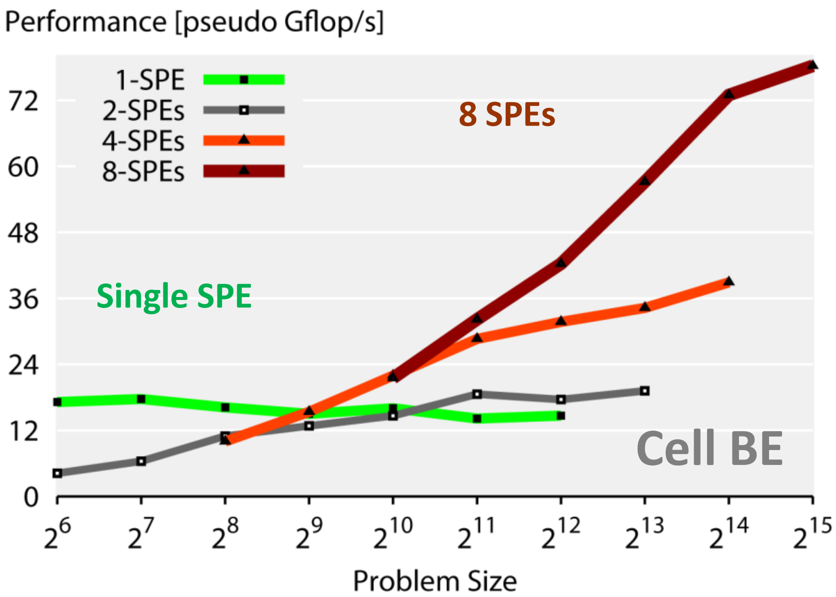
Single BlueGene/P node (4 CPUs) at 850 MHz
Argonne National Laboratory

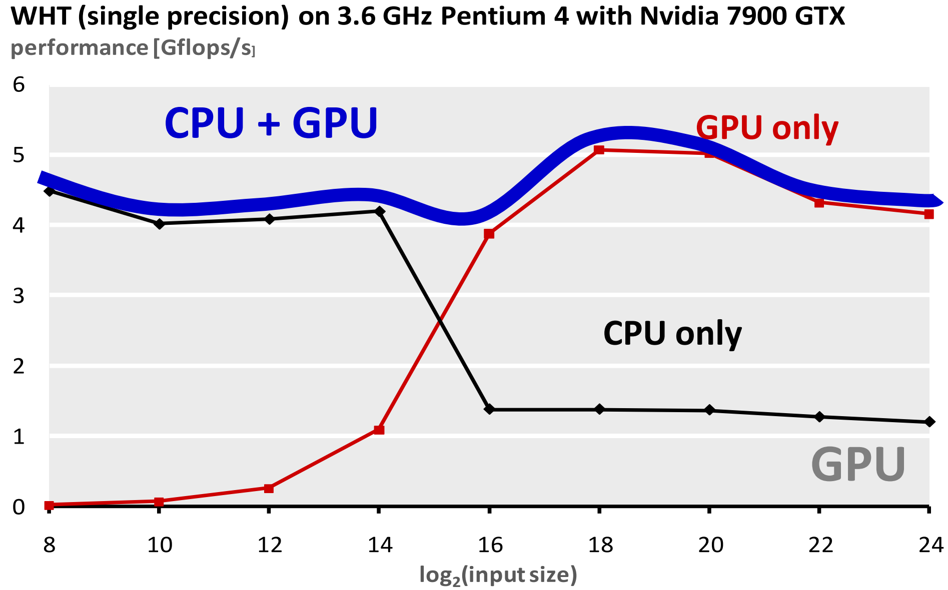
SIMD vectorization + multi-threading

F. Gygi, E. W. Draeger, M. Schulz, B. R. de Supinski, J. A. Gunnels, V. Austel, J. C. Sexton, F. Franchetti, S. Kral, C. W. Ueberhuber, J. Lorenz: **Large-Scale Electronic Structure Calculations of High-Z Metals on the BlueGene/L Platform.** In Proceedings of Supercomputing, 2006. **Winner of the 2006 Gordon Bell Prize (Peak Performance Award).**

J. Lorenz, S. Kral, F. Franchetti, C. W. Ueberhuber: **Vectorization Techniques for the Blue Gene/L double FPU.** IBM Journal of Research and Development, Vol. 49, No. 2/3, 2005.

New Multicore Architectures: Cell and GPU



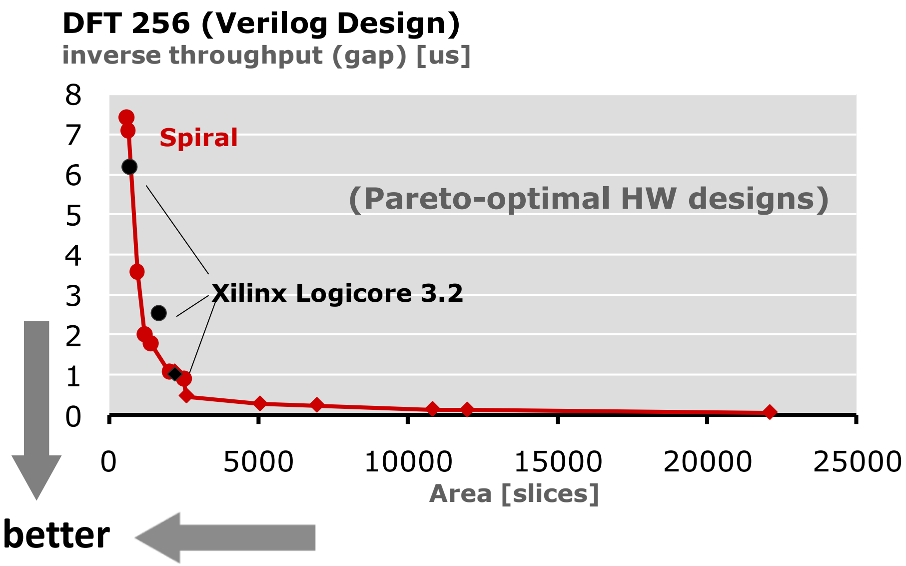


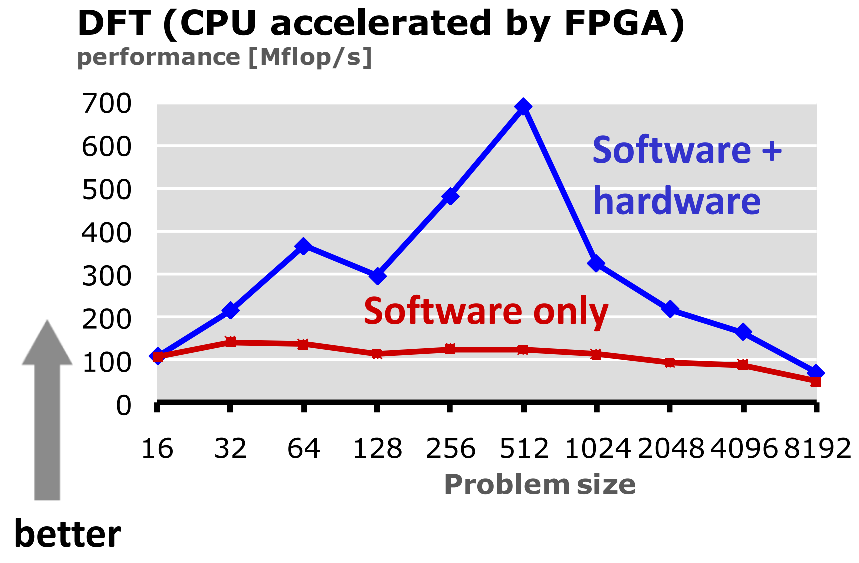
- **Cell BE:** SIMD vector + local stores + DMA
- **GPU:** SIMD vector + OpenGL + Cg + SMT

F. Franchetti, Y. Voronenko, P. A. Milder, S. Chellappa, M. Telgarsky, H. Shen, P. D'Alberto, F. de Mesmay, J. C. Hoe, J. M. F. Moura, M. Püschel: **Domain-Specific Library Generation for Parallel Software and Hardware Platforms.** In Proceedings of The NSF Next Generation Software (NGS) Workshop 2008.

S. Chellappa, F. Franchetti, and M. Püschel: **Computer Generation of Fast FFTs for the Cell Broadband Engine** In Proceedings of the International Conference on Supercomputing (ICS), 2009.

Hardware: FPGA, CPU + FPGA-Acceleration



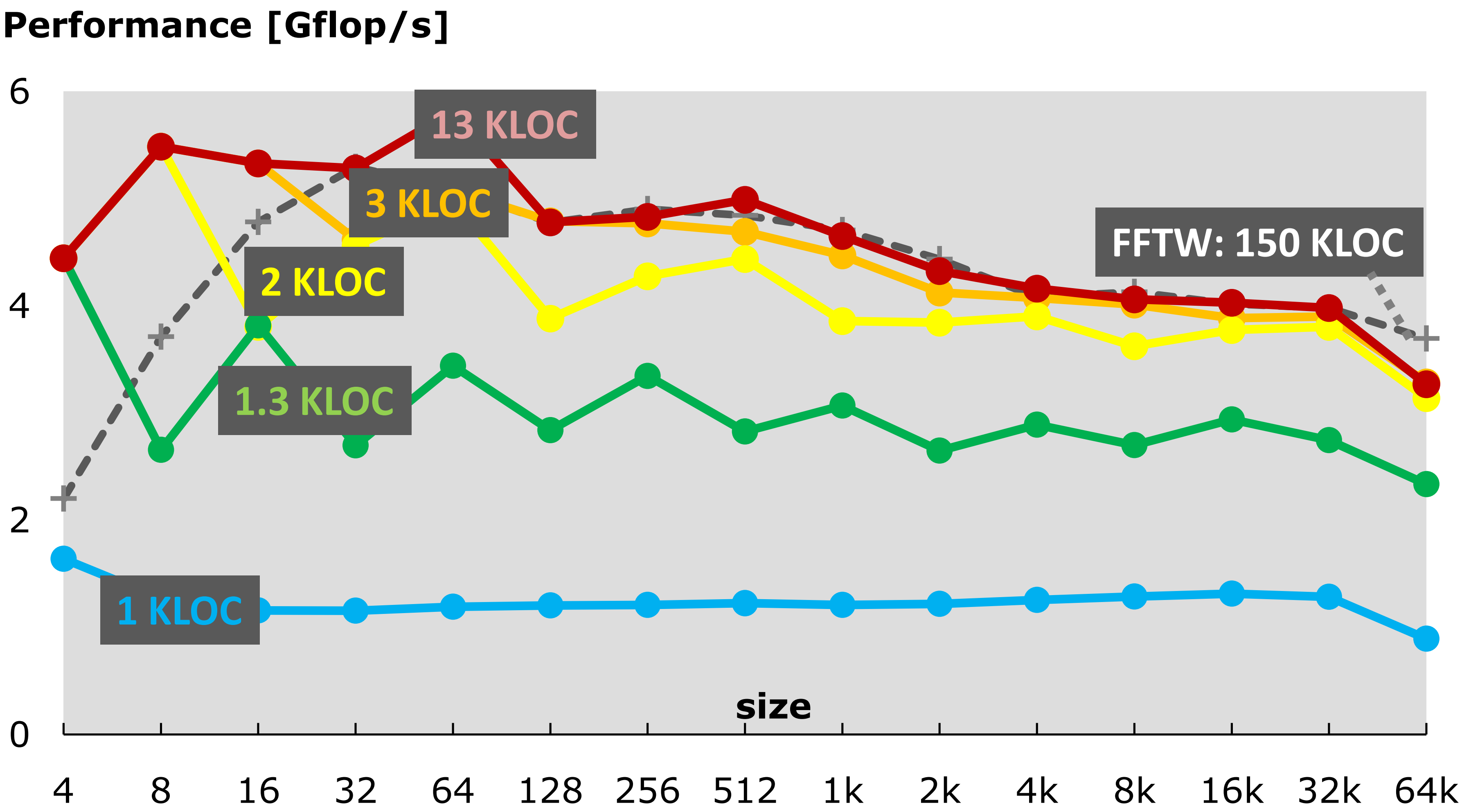


Xilinx Virtex 2 Pro FPGA: 1M gates @ 100 MHz + 2 PowerPC 405 @ 300 MHz

P. A. Milder, F. Franchetti, J. C. Hoe, and M. Püschel:
Formal Datapath Representation and Manipulation for Implementing DSP Transforms.
In Proceedings of Design Automation Conference (DAC), 2008.

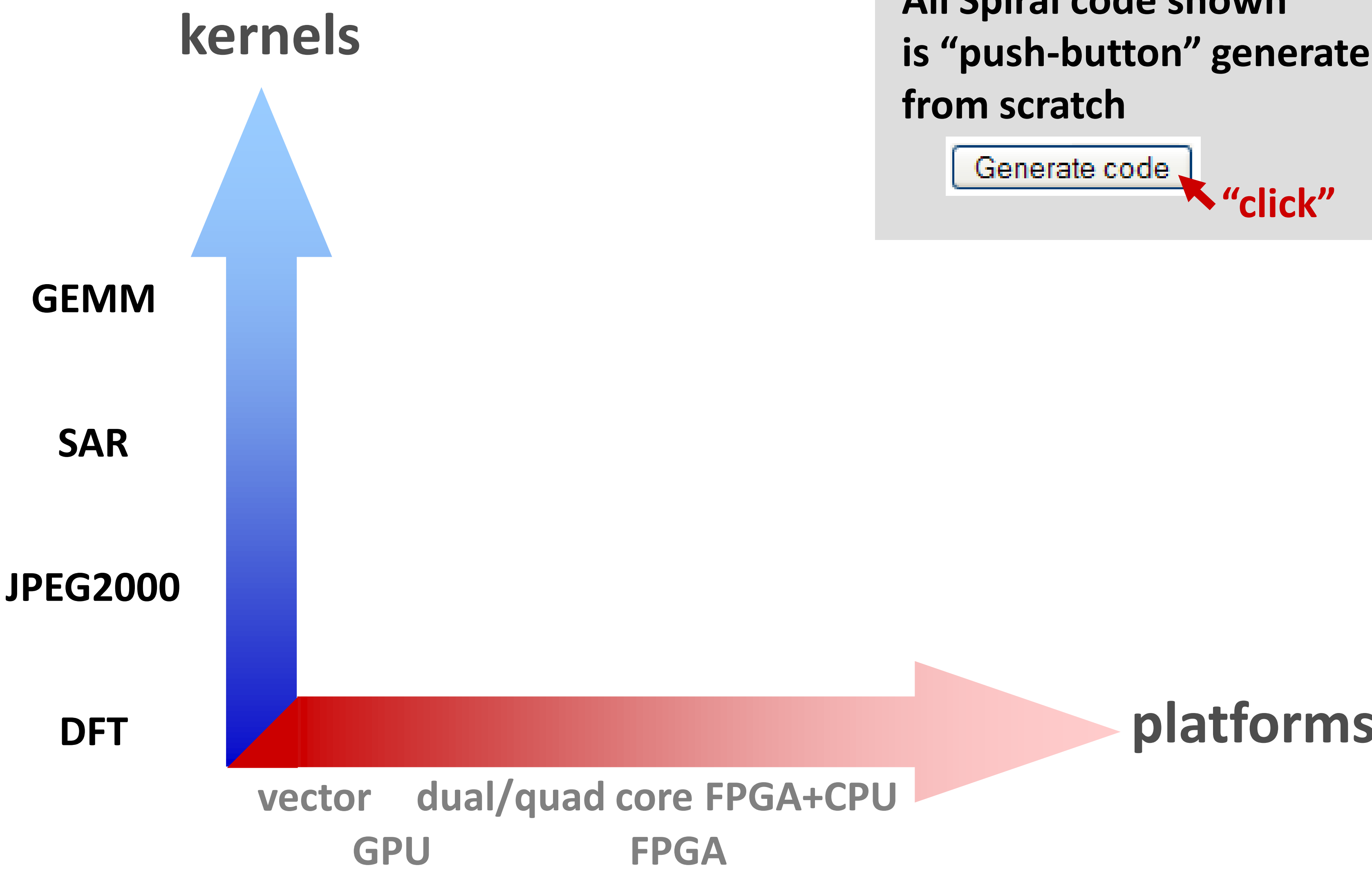
P. D'Alberto, F. Franchetti, P. A. Milder, A. Sandryhaila, J. C. Hoe, J. M. F. Moura, and M. Püschel:
Generating FPGA Accelerated DFT Libraries.
In Proceedings of Field-Programmable Custom Computing Machines (FCCM), 2007.

Qualitative Customization: Code Size



Y. Voronenko, F. de Mesmay, and M. Püschel: **Computer generation of general size linear transform libraries** in Proceedings Code Generation and Optimization (CGO), 2009.

Benchmarks

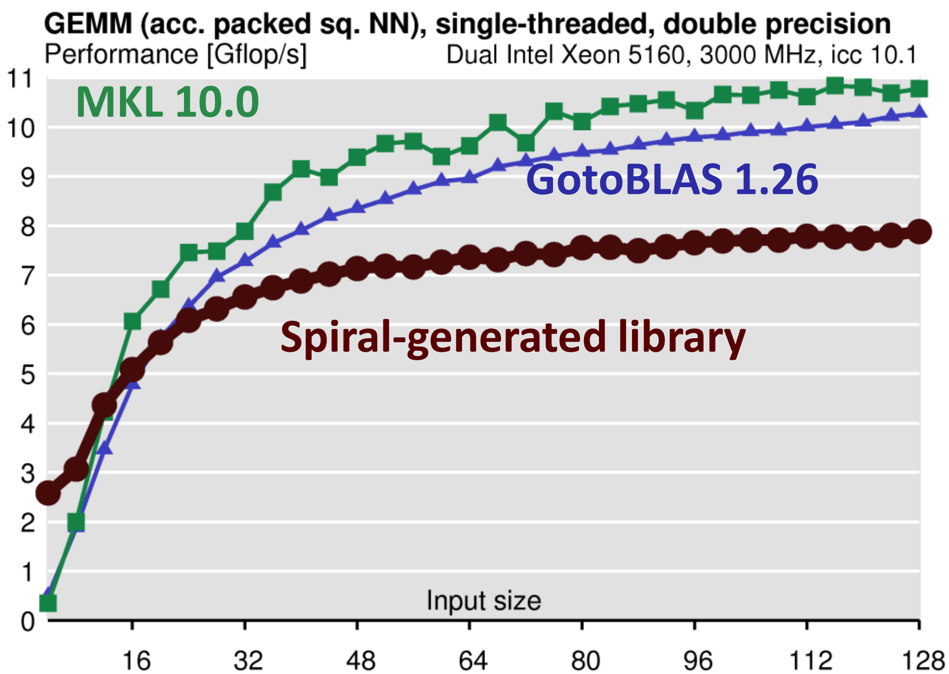


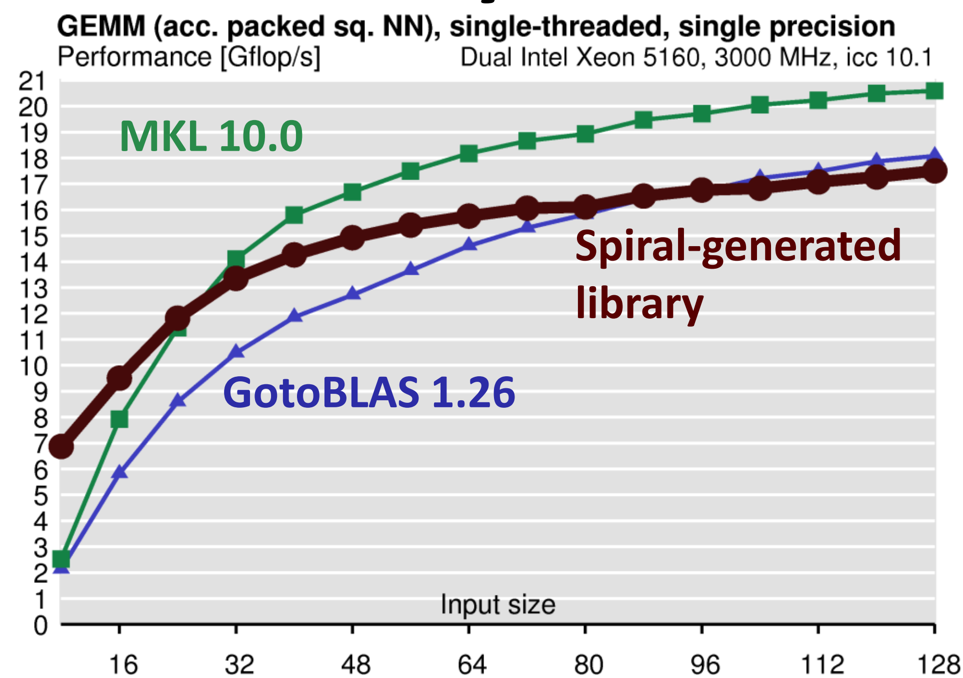
All Spiral code shown is "push-button" generated from scratch

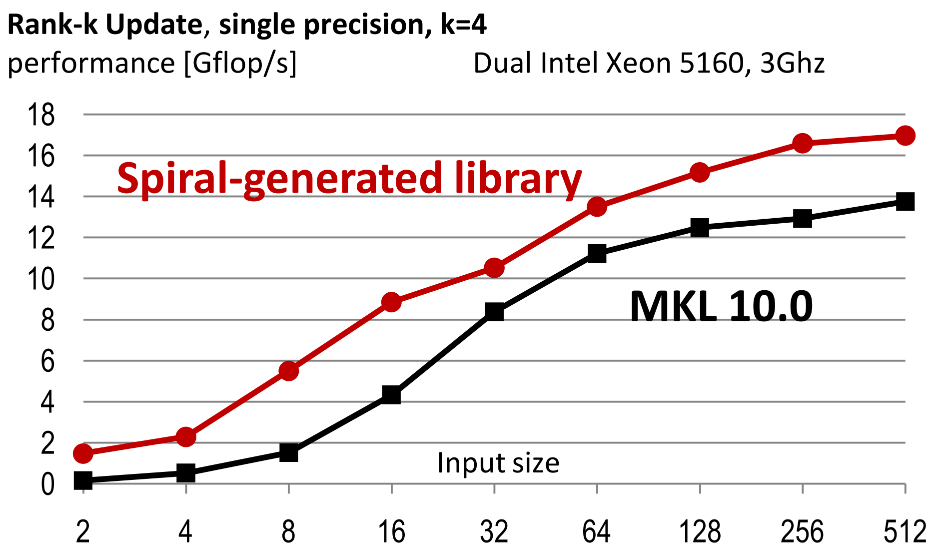
Generate code

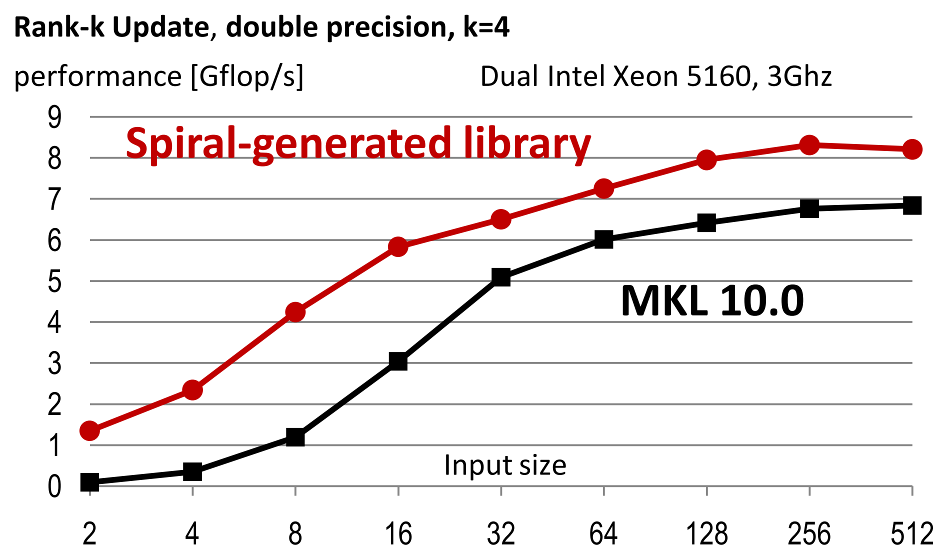
"click"

Result: Matrix Multiplication Library

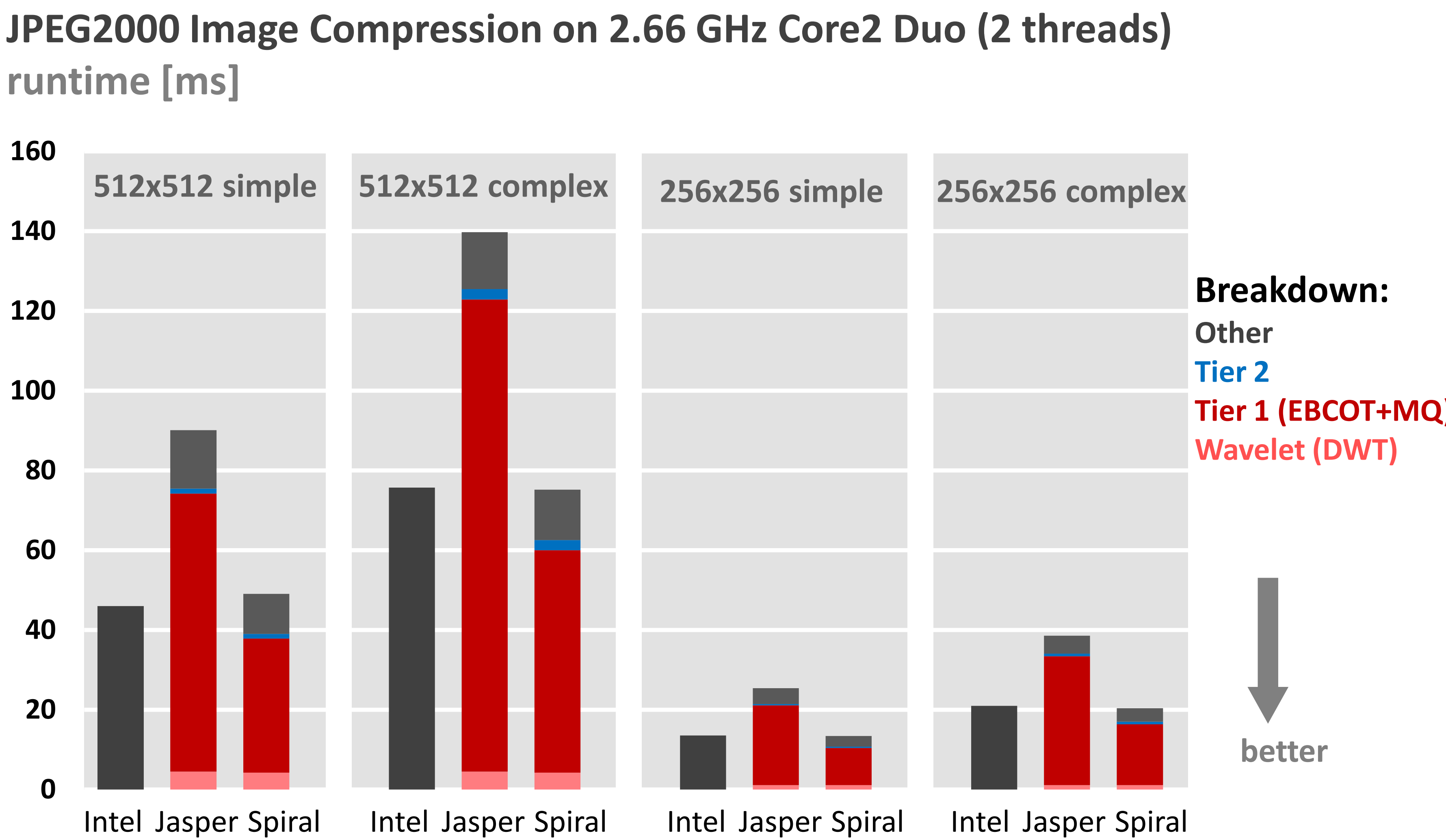








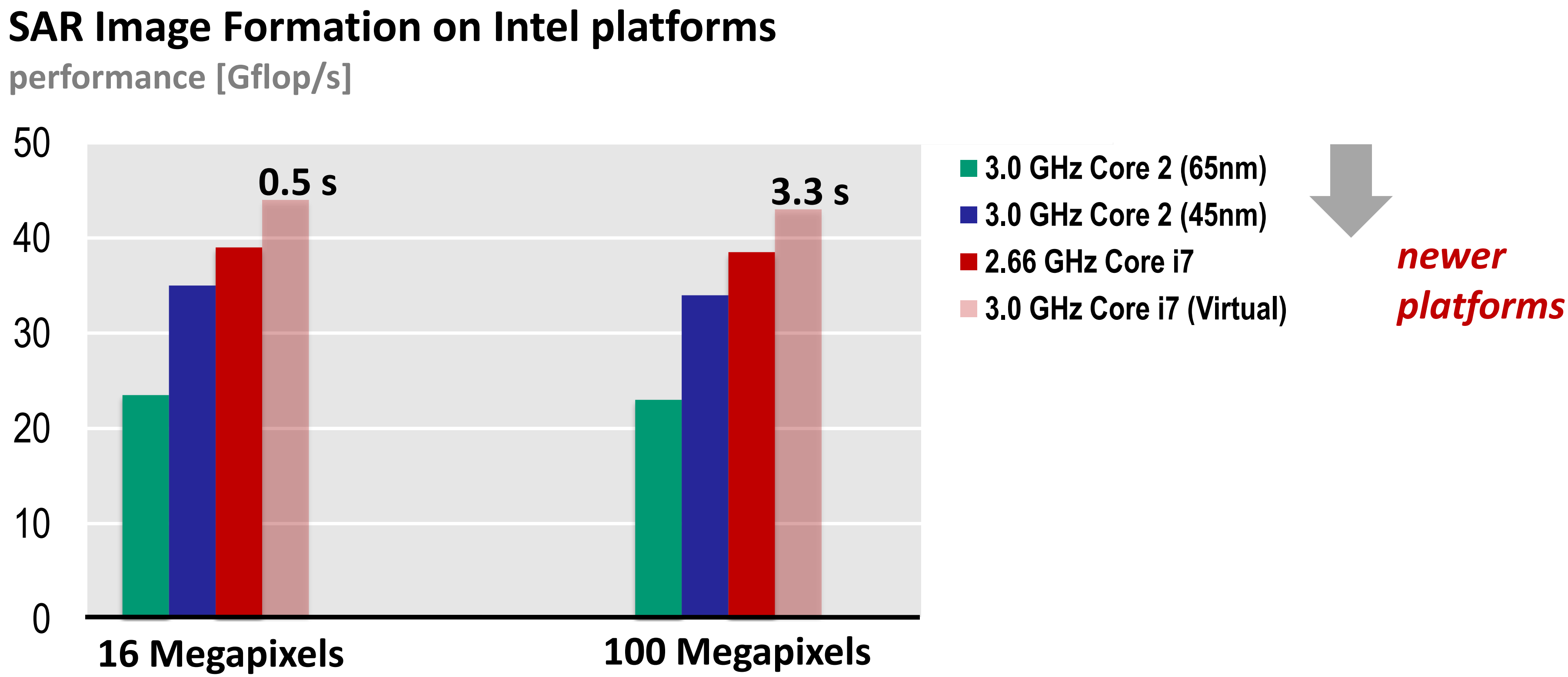
JPEG 2000: Wavelet + Entropy Coding



Tile size = image size, resolution level = 2, EBCOT size = 64x64

Joint work with S. Hao

Polar Format SAR on Intel Core2 Quad



- Algorithm by J. Rudin (best paper award, HPEC 2007): 30 Gflop/s on Cell
- Each implementation: vectorized, threaded, cache tuned, ~13 MB of code

D. McFarlin, F. Franchetti, M. Püschel, and J. M. F. Moura: **High Performance Synthetic Aperture Radar Image Formation On Commodity Multicore Architectures.** in Proceedings SPIE, 2009.

Organization

- Spiral overview
- Parallelization in Spiral
- Beyond Transforms
- Results
- **Concluding remarks**

Summary

- **Platforms are powerful yet complicated**
  optimization will stay a hard problem

Image: Intel

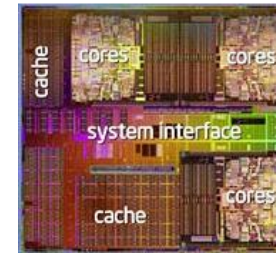
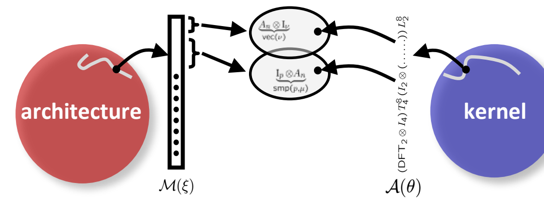


Image: Intel

- **Unified mathematical framework**
  captures platforms and algorithms

$$\underbrace{I_p \otimes A_n}_{\mathsf{smp}(p,\mu)}$$

- **Program generation and optimization**
  can provide full automation



- **Ongoing Research: Add more functionality**
  kernels, platforms,…