

# Spiral: Program Generation for Linear Transforms and Beyond

**Franz Franchetti**

ECE, Carnegie Mellon University  
[www.spiral.net](http://www.spiral.net)

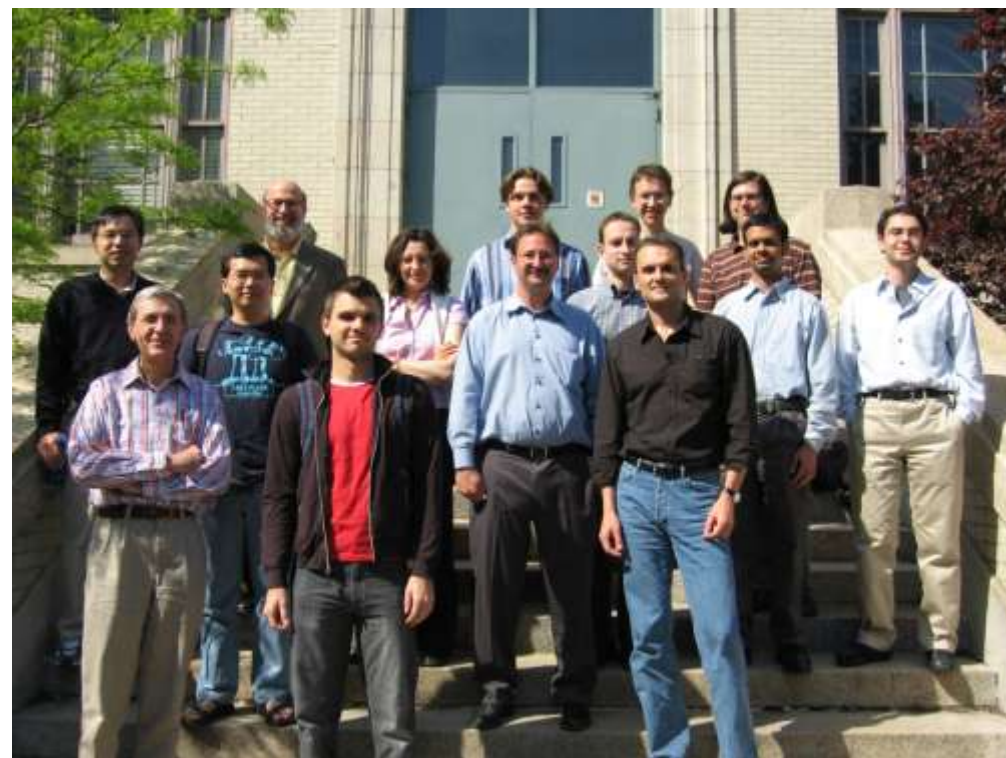
Co-Founder, SpiralGen  
[www.spiralgen.com](http://www.spiralgen.com)

**Joint work with**

**Yevgen Voronenko**  
**Frédéric de Mesmay**  
**Daniel McFarlin**  
**Markus Püschel**

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

**... and the Spiral team (only part shown)**

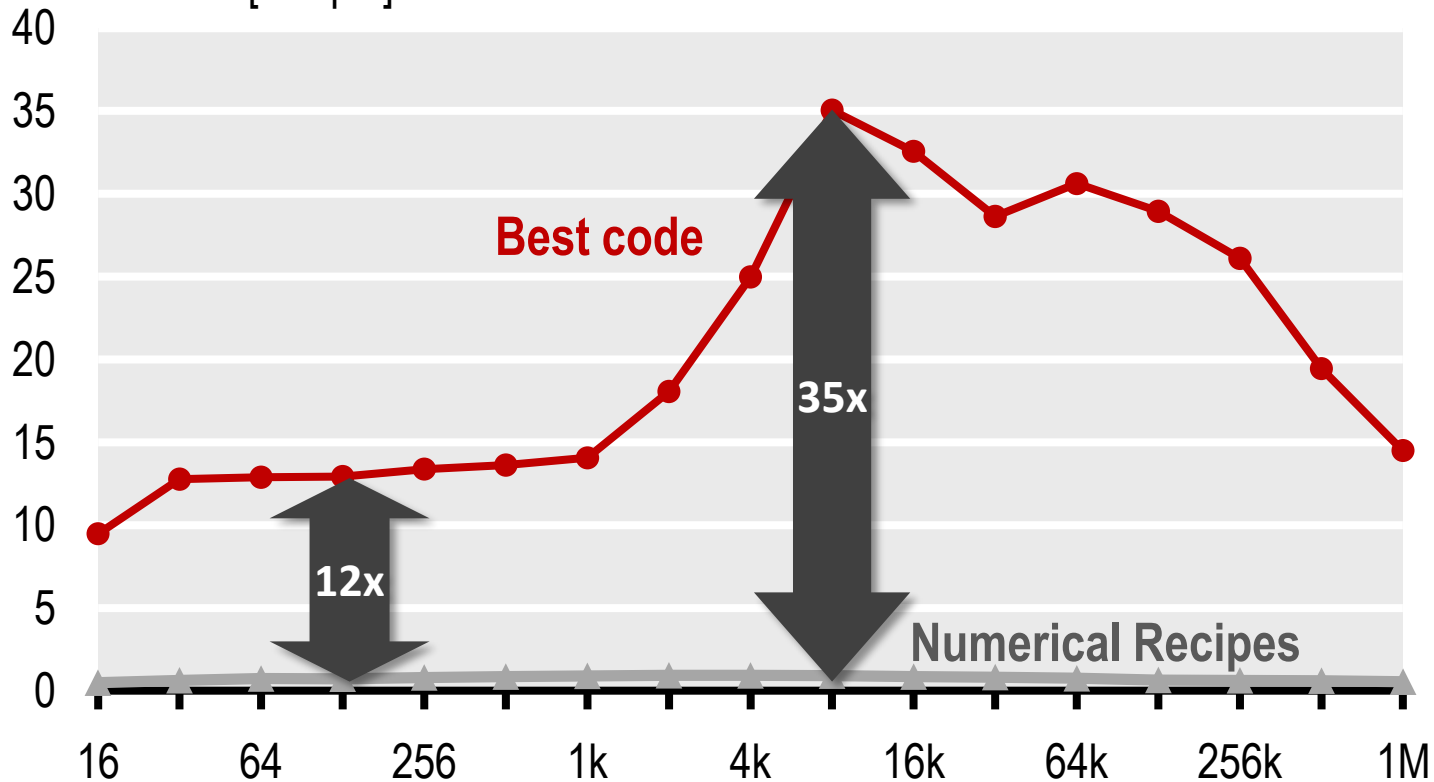




# The Problem: Example DFT

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

Performance [Gflop/s]

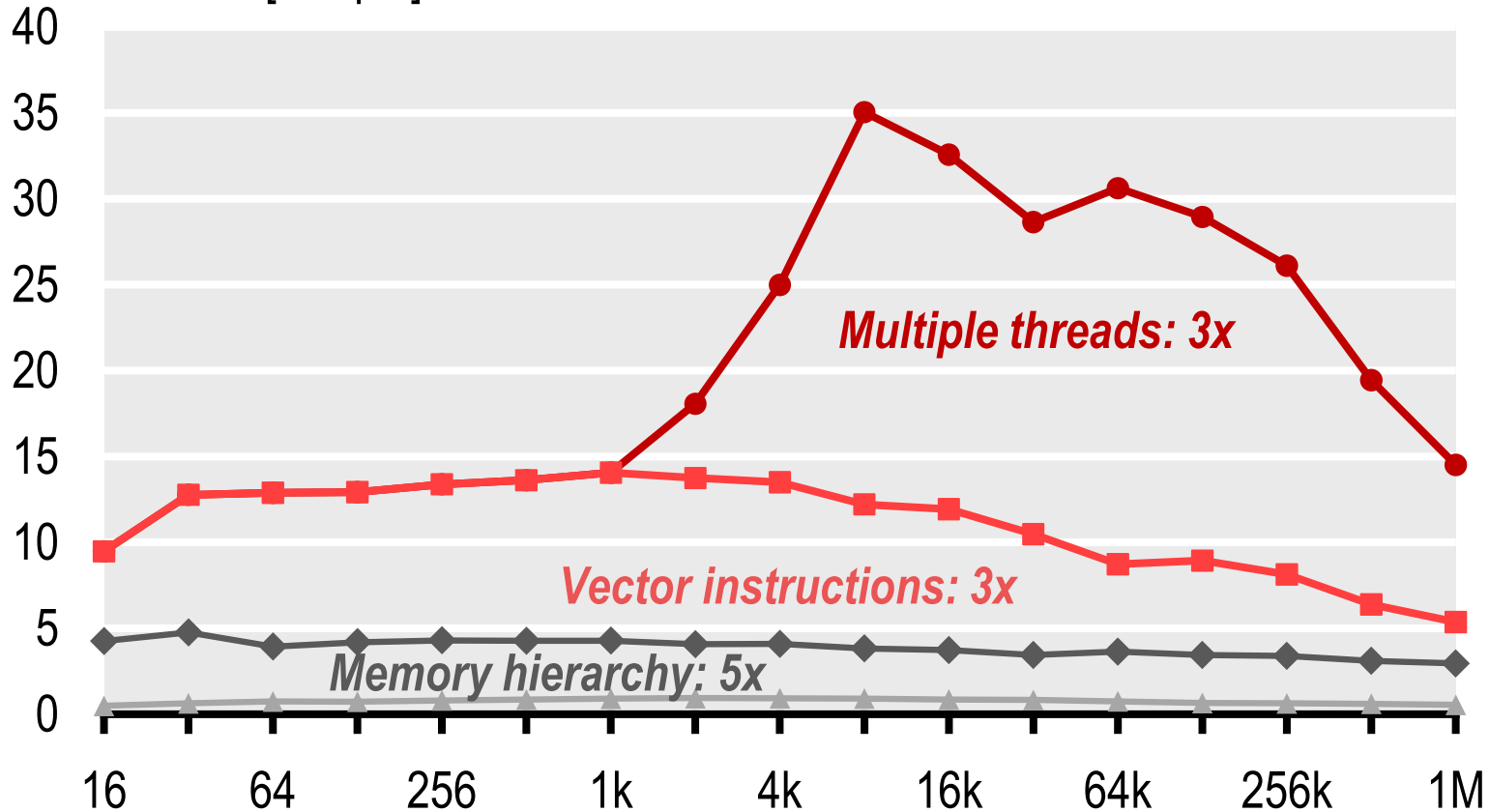


- Standard desktop computer, cutting edge compiler, using optimization flags
- Implementations have same operations count:  $\approx 4n \log_2(n)$
- ***Same plots can be shown for all mathematical functions***

# DFT Plot: Analysis

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

Performance [Gflop/s]



*High performance library development has become a nightmare*

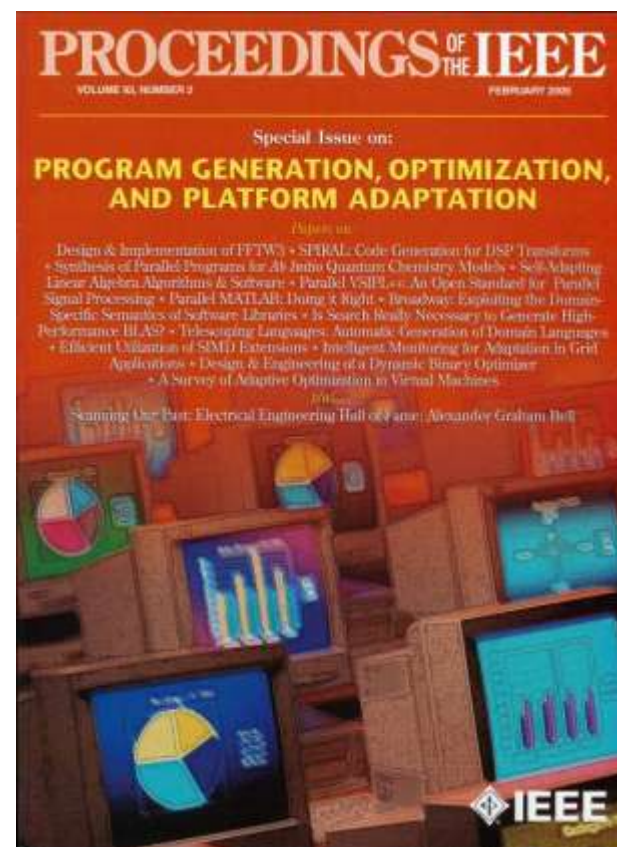
# Automatic Performance Tuning

- **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***



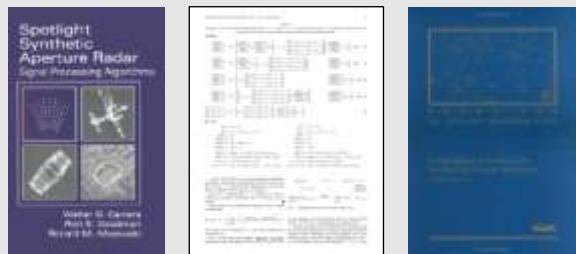
Proceedings of the IEEE special issue, Feb. 2005

# Organization

- **Spiral overview**
- **Spiral's formal framework**
- **Parallelization in Spiral**
- **Generating general-size libraries**
- **Results**
- **Concluding remarks**

# What is Spiral?

## *Traditionally*



High performance library  
optimized for given platform

## *Spiral Approach*



High performance library  
optimized for given platform

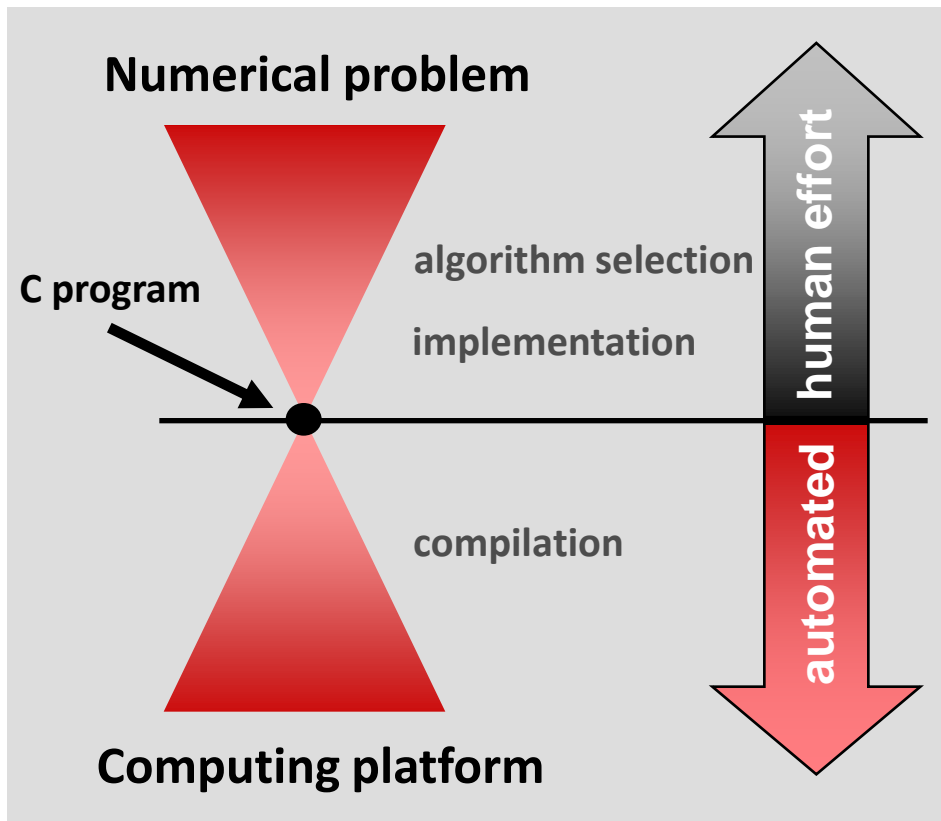
*Comparable  
performance*

# Spiral in a Nutshell

- **Library generator for computational kernels**  
focus on linear transforms; some support for other kernels
- **Wide range of parallel paradigms supported**  
SIMD vector, threading, messaging, streaming, gate level, offloading
- **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**  
Update the tool rather than rewriting the library
- **Commercial-grade software**  
Intel uses Spiral in MKL and IPP; SpiralGen commercializes the technology

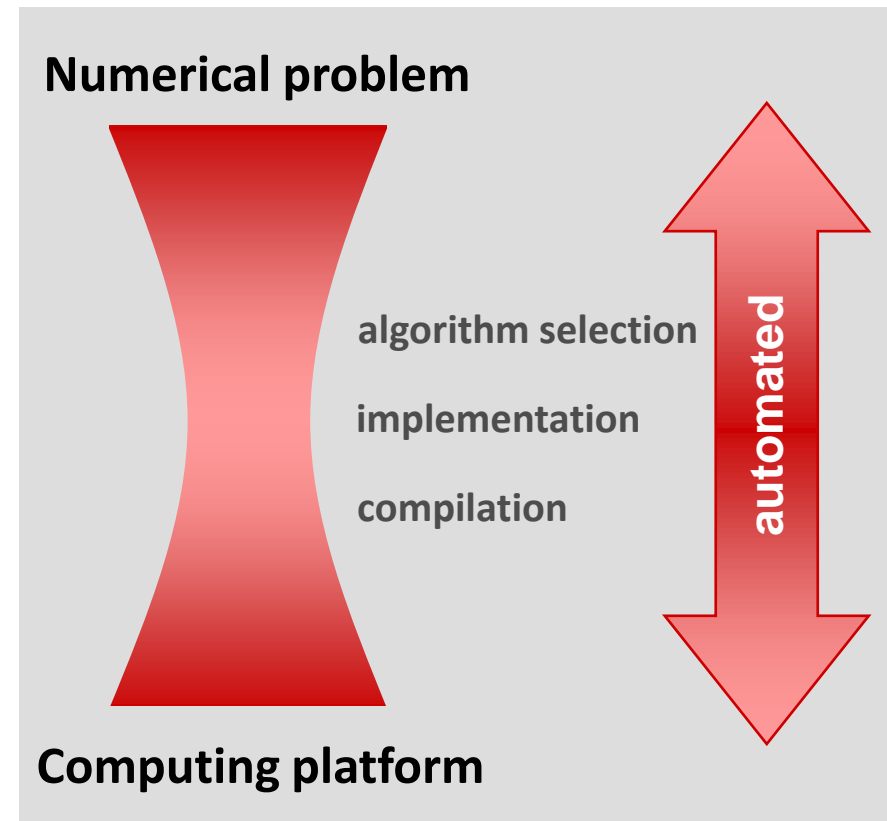
# Vision Behind Spiral

## Current



- C code a singularity: Compiler has no access to high level information

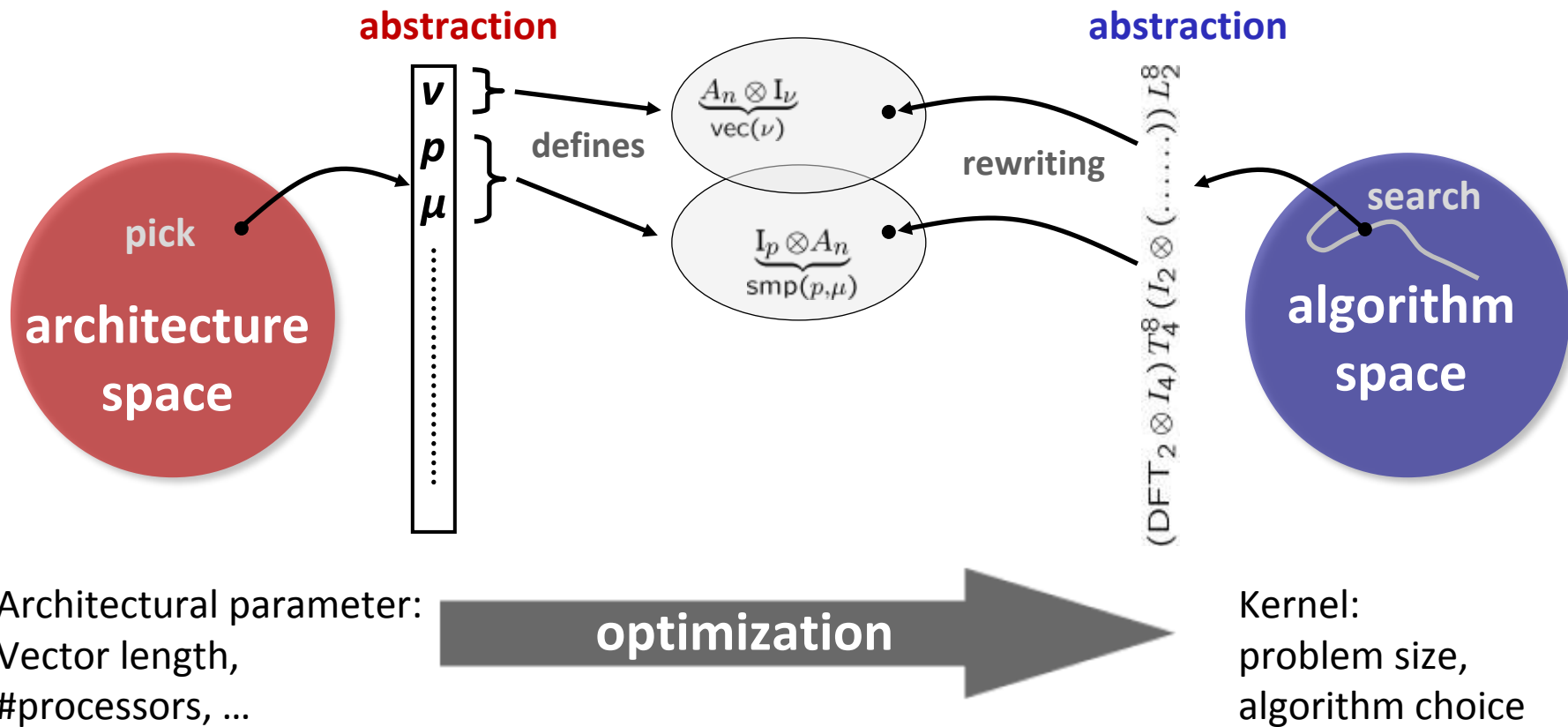
## Future



- Challenge: conquer the high abstraction level for **complete automation**

# Main Idea: Program Generation

**Model:** common abstraction  
= spaces of matching formulas



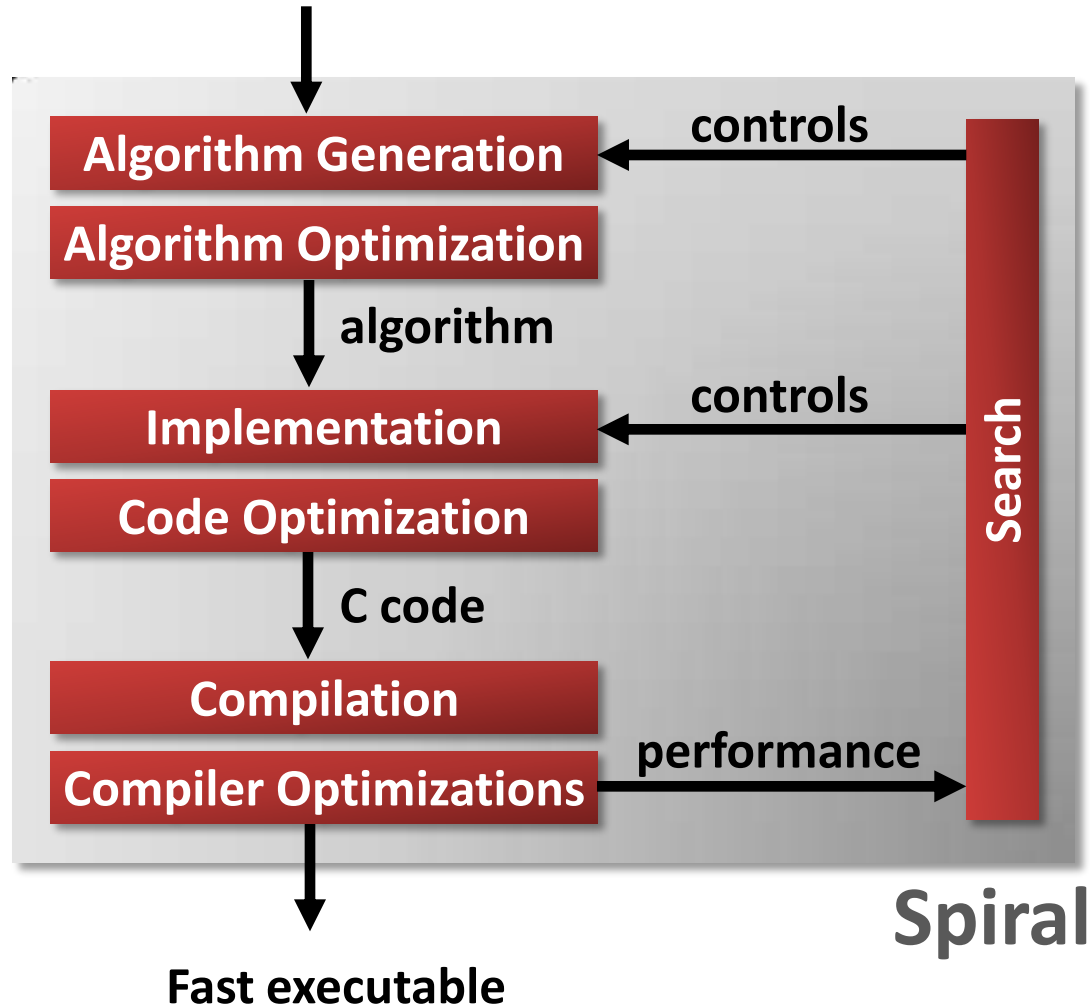
# How Spiral Works

**Complete automation** of the implementation and optimization task

**Basic ideas:**

- **Declarative representation** of algorithms
- **Rewriting systems** to generate and optimize algorithms at a high level of abstraction

Problem specification (“DFT 1024” or “DFT”)



# Spiral's Face: Web Interface @spiral.net

## Select convolutional code

Select a preset code or customize parameters

*custom*

Voyager

NASA-DSN

CCSDS/NASA-GSFC

WiMax

CDMA IS-95A

LTE (3GPP - Long Term Evolution)

UWB (802.15)

CDMA 2000

Cassini

Mars Pathfinder & Stereo

rate 1 /

code rate [\(?\)](#)

K

constraint length [\(?\)](#)

polynomials

polynomials for the  
code in decimal notation  
[\(?\)](#)

## Select implementation options

frame length

unpadded frame length in bits [\(?\)](#)

Vectorization level

type of code [\(?\)](#)



“Click”: Push-button code generation

<http://www.spiral.net/software/viterbi.html>

# Organization

- Spiral overview
- **Spiral's formal framework**
- Parallelization in Spiral
- 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

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.

# Spiral's Origin: Transforms and Algorithms

## ■ Transform = Matrix-vector multiplication

Example: Discrete Fourier transform (DFT)

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

input vector (signal)  $\curvearrowright$   $x$   $\curvearrowleft$  output vector (signal)  $y$

$T$   $\uparrow$  transform = matrix

## ■ Fast algorithm = sparse matrix factorization = SPL formula

Example: Cooley-Tukey FFT algorithm

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

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

# Breakdown Rules (>200 for >50 Transforms)

$$\begin{aligned}
 \text{DFT}_n &\rightarrow P_{k/2,2m}^\top \left( \text{DFT}_{2m} \oplus (I_{k/2-1} \otimes_i C_{2m} \text{rDFT}_{2m}(i/k)) \right) (\text{RDFT}'_k \otimes I_m), \quad k \text{ even,} \\
 \begin{pmatrix} \text{RDFT}'_n \\ \text{RDFT}_n \\ \text{DHT}'_n \\ \text{DHT}_n \end{pmatrix} &\rightarrow (P_{k/2,m}^\top \otimes I_2) \left( \begin{pmatrix} \text{RDFT}'_{2m} \\ \text{RDFT}_{2m} \\ \text{DHT}'_{2m} \\ \text{DHT}_{2m} \end{pmatrix} \oplus I_{k/2-1} \otimes_i D_{2m} \begin{pmatrix} \text{rDFT}'_{2m}(i/k) \\ \text{rDFT}_{2m}(i/k) \\ \text{rDHT}'_{2m}(i/k) \\ \text{rDHT}_{2m}(i/k) \end{pmatrix} \right) \left( \begin{pmatrix} \text{RDFT}'_k \\ \text{RDFT}_k \\ \text{DHT}'_k \\ \text{DHT}_k \end{pmatrix} \otimes I_m \right), \quad k \text{ even,} \\
 \begin{pmatrix} \text{rDFT}_{2n}(u) \\ \text{rDHT}_{2n}(u) \end{pmatrix} &\rightarrow L_m^{2n} \left( I_k \otimes_i \begin{pmatrix} \text{rDFT}_{2m}((i+u)/k) \\ \text{rDHT}_{2m}((i+u)/k) \end{pmatrix} \right) \left( \begin{pmatrix} \text{rDFT}_{2k}(u) \\ \text{rDHT}_{2k}(u) \end{pmatrix} \otimes I_m \right), \\
 \text{RDFT-3}_n &\rightarrow (Q_{k/2,m}^\top \otimes I_2) (I_k \otimes_i \text{rDFT}_{2m})(i+1/2/k) (\text{RDFT-3}_k \otimes I_m), \quad k \text{ even,} \\
 \text{DCT-2}_n &\rightarrow P_{k/2,2m}^\top \left( \text{DCT-2}_{2m} K_2^{2m} \oplus (I_{k/2-1} \otimes N_{2m} \text{RDFT-3}_{2m}^\top) \right) B_n(L_{k/2}^{n/2} \otimes I_2) (I_m \otimes \text{RDFT}'_k) Q_{m/2,k}, \\
 \text{DCT-3}_n &\rightarrow \text{DCT-2}_n^\top, \\
 \text{DCT-4}_n &\rightarrow Q_{k/2,2m}^\top (I_{k/2} \otimes N_{2m} \text{RDFT-3}_{2m}^\top) B'_n(L_{k/2}^{n/2} \otimes I_2) (I_m \otimes \text{RDFT-3}_k) Q_{m/2,k}. \\
 \text{DFT}_n &\rightarrow (\text{DFT}_k \otimes I_m) \Upsilon_m^n (I_k \otimes \text{DFT}_m) \mathcal{L}_k^n, \quad n = km \\
 \text{DFT}_n &\rightarrow P_n (\text{DFT}_k \otimes \text{DFT}_m) Q_n, \quad n = km, \text{ gcd}(k, m) = 1 \\
 \text{DFT}_p &\rightarrow R_p^\top (I_1 \oplus \text{DFT}_{p-1}) D_p (I_1 \oplus \text{DFT}_{p-1}) R_p, \quad p \text{ prime} \\
 \text{DCT-3}_n &\rightarrow (I_m \oplus J_m) \mathcal{L}_m^n (\text{DCT-3}_m(1/4) \oplus \text{DCT-3}_m(3/4)) \\
 &\quad \cdot (F_2 \otimes I_m) \begin{bmatrix} I_m & 0 \oplus -J_{m-1} \\ \frac{1}{\sqrt{2}}(I_1 \oplus 2I_m) & \end{bmatrix}, \quad n = 2m \\
 \text{DCT-4}_n &\rightarrow S_n \text{DCT-2}_n \text{diag}_{0 \leq k < n} (1/(2 \cos((2k+1)\pi/4n))) \\
 \text{IMDCT}_{2m} &\rightarrow (J_m \oplus I_m \oplus I_m \oplus J_m) \left( \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes I_m \right) \oplus \left( \begin{bmatrix} -1 \\ -1 \end{bmatrix} \otimes I_m \right) \right) J_{2m} \text{DCT-4}_{2m} \\
 \text{WHT}_{2^k} &\rightarrow \prod_{i=1}^t (I_2^{k_1+\dots+k_{i-1}} \otimes \text{WHT}_{2^{k_i}} \otimes I_2^{k_{i+1}+\dots+k_t}), \quad k = k_1 + \dots + k_t \\
 \text{DFT}_2 &\rightarrow F_2 \\
 \text{DCT-2}_2 &\rightarrow \text{diag}(1, 1/\sqrt{2}) F_2 \\
 \text{DCT-4}_2 &\rightarrow J_2 R_{13\pi/8}
 \end{aligned}$$

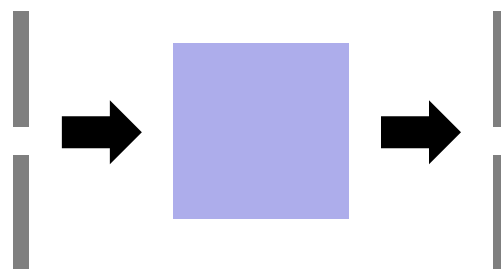
- *“Teaches” Spiral algorithm knowledge*
- *Combining these rules yields many algorithms for every given transform*

# Beyond Transforms: General Operators

- 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 (OL)

name	definition
<i>basic operators</i>	
projection	$\pi_{\mathbf{x}} : \mathbb{C}^m \times \mathbb{C}^n \rightarrow \mathbb{C}^m; (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}$
linear transform	$M : \mathbb{C}^n \rightarrow \mathbb{C}^m; \mathbf{x} \mapsto M\mathbf{x}$
stride	$L_m^{mn} : \mathbb{C}^{mn} \rightarrow \mathbb{C}^{mn}; \mathbf{x} \mapsto L_m^{mn} \mathbf{x}$
vector sum	$\Sigma_n : \mathbb{C}^n \rightarrow \mathbb{C}; \mathbf{x} \mapsto \sum_{i=0}^{n-1} x_i$
vector minimum	$\min_n : \mathbb{C}^n \rightarrow \mathbb{C}; \mathbf{x} \mapsto \min(x_0, \dots, x_{n-1})$
constant vector	$C_c : \emptyset \rightarrow \mathbb{C}^n; () \mapsto \mathbf{c}$
<i>operations</i>	
addition	$(M + N)(\mathbf{x}, \mathbf{y}) = M(\mathbf{x}, \mathbf{y}) + N(\mathbf{x}, \mathbf{y})$
multiplication	$(M \cdot N)(\mathbf{x}, \mathbf{y}) = M(\mathbf{x}, \mathbf{y}) \cdot N(\mathbf{x}, \mathbf{y})$
direct sum	$(M \oplus N)(\mathbf{x} \oplus \mathbf{u}, \mathbf{y} \oplus \mathbf{v}) = M(\mathbf{x}, \mathbf{y}) \oplus N(\mathbf{u}, \mathbf{v})$
cartesian product	$(M \times N)(\mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{v}) = M(\mathbf{x}, \mathbf{y}) \times N(\mathbf{u}, \mathbf{v})$
composition	$(M \circ N)(\mathbf{x}, \mathbf{y}) = M(N(\mathbf{x}, \mathbf{y}))$
iterative composition	$(\prod_{i=0}^{n-1} M_i)(\mathbf{x}, \mathbf{y}) = (M_0 \circ \dots \circ M_{n-1})(\mathbf{x}, \mathbf{y})$
tensor product	$I \otimes M, M \otimes I$

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

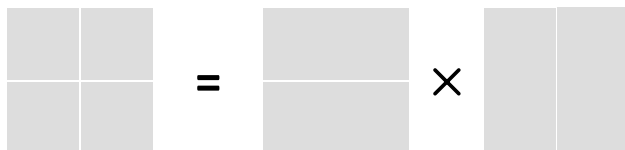
$$(A \otimes I_{m \times n \rightarrow mn}) \left( \sum_{i=0}^{m-1} \mathbf{x} \otimes \mathbf{e}_i^m, \sum_{i=0}^{n-1} \mathbf{y} \otimes \mathbf{e}_i^n \right) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A(\mathbf{x}, \mathbf{y}) \otimes \mathbf{e}_i^m \otimes \mathbf{e}_j^n.$$

# Expressing Kernels as OL Formulas

## Linear Transforms

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

## Matrix-Matrix Multiplication



$$\begin{aligned}
 \text{MMM}_{1,1,1} &\rightarrow (\cdot)_1 \\
 \text{MMM}_{m,n,k} &\rightarrow (\otimes)_{m/m_b \times 1} \otimes \text{MMM}_{m_b,n,k} \\
 \text{MMM}_{m,n,k} &\rightarrow \text{MMM}_{m,n_b,k} \otimes (\otimes)_{1 \times n/n_b} \\
 \text{MMM}_{m,n,k} &\rightarrow ((\Sigma_{k/k_b} \circ (\cdot)_{k/k_b}) \otimes \text{MMM}_{m,n,k_b}) \circ \\
 &\quad ((L_{k/k_b}^{m_k/k_b} \otimes \text{I}_{k_b}) \times \text{I}_{kn}) \\
 \text{MMM}_{m,n,k} &\rightarrow (L_m^{mn/n_b} \otimes \text{I}_{n_b}) \circ \\
 &\quad ((\otimes)_{1 \times n/n_b} \otimes \text{MMM}_{m,n_b,k}) \circ \\
 &\quad (\text{I}_{km} \times (L_{n/n_b}^{kn/n_b} \otimes \text{I}_{n_b}))
 \end{aligned}$$

## Viterbi Decoding



$$\mathbf{F}_{K,F} \rightarrow \prod_{i=1}^F \left( (\text{I}_{2^{K-2}} \otimes_j B_{F-i,j}) \text{L}_{2^{K-2}}^{2^{K-1}} \right)$$

$$\mathbf{F}_{K,F} \nu \rightarrow \prod_{i=1}^F \left( (\text{I}_{2^{K-2/\nu}} \otimes_{j_1} \text{L}_{\nu}^{-2\nu} \tilde{B}_{F-i,j_1}^{\nu}) (\text{L}_{2^{K-2/\nu}}^{2^{K-1/\nu}} \otimes \text{I}_{\nu}) \right)$$

$$B_{i,j} : \begin{cases} \pi_U = \min_{d_U} (\pi_A + \beta_{A \rightarrow U}, \pi_B + \beta_{B \rightarrow U}) \\ \pi_V = \min_{d_V} (\pi_A + \beta_{A \rightarrow V}, \pi_B + \beta_{B \rightarrow V}) \end{cases}$$

## Synthetic Aperture Radar (SAR)



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

# Translating OL Formulas into Programs

## Linear Operators

$y = (A_n B_n)x$ 

```

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])

```

$$I_m \otimes A_n = \begin{bmatrix} A_n & & \\ & \dots & \\ & & A_n \end{bmatrix}$$

$y = (A_m \otimes I_n)x$ 

```

for (i=0;i<m;i++)
  y[i:n:i+m-1] =
    A(x[i:n:i+m-1]);

```

## General Operators

$r = (A_{K \times M \rightarrow N} \circ B_{k \times m \rightarrow K \times M})(x, y)$ 

```

(t, u) = B(x, y);
r = A(t, u);

```

$(r, s) = (A_{m \rightarrow M} \times B_{n \rightarrow N})(x, y)$ 

```

r = A(x);
s = B(y);

```

$r = (I_{m \times n \rightarrow mn} \otimes A_{M \times N \rightarrow K})(x, y)$ 

```

for (i=0;i<m;i++)
  for (j=0;j<n;j++)
    r[(i*m+j)*K:1:(i*m+j+1)*K-1] =
      A(x[i*M:1:i*M+M-1], y[j*N:1:j*N+N-1]);

```

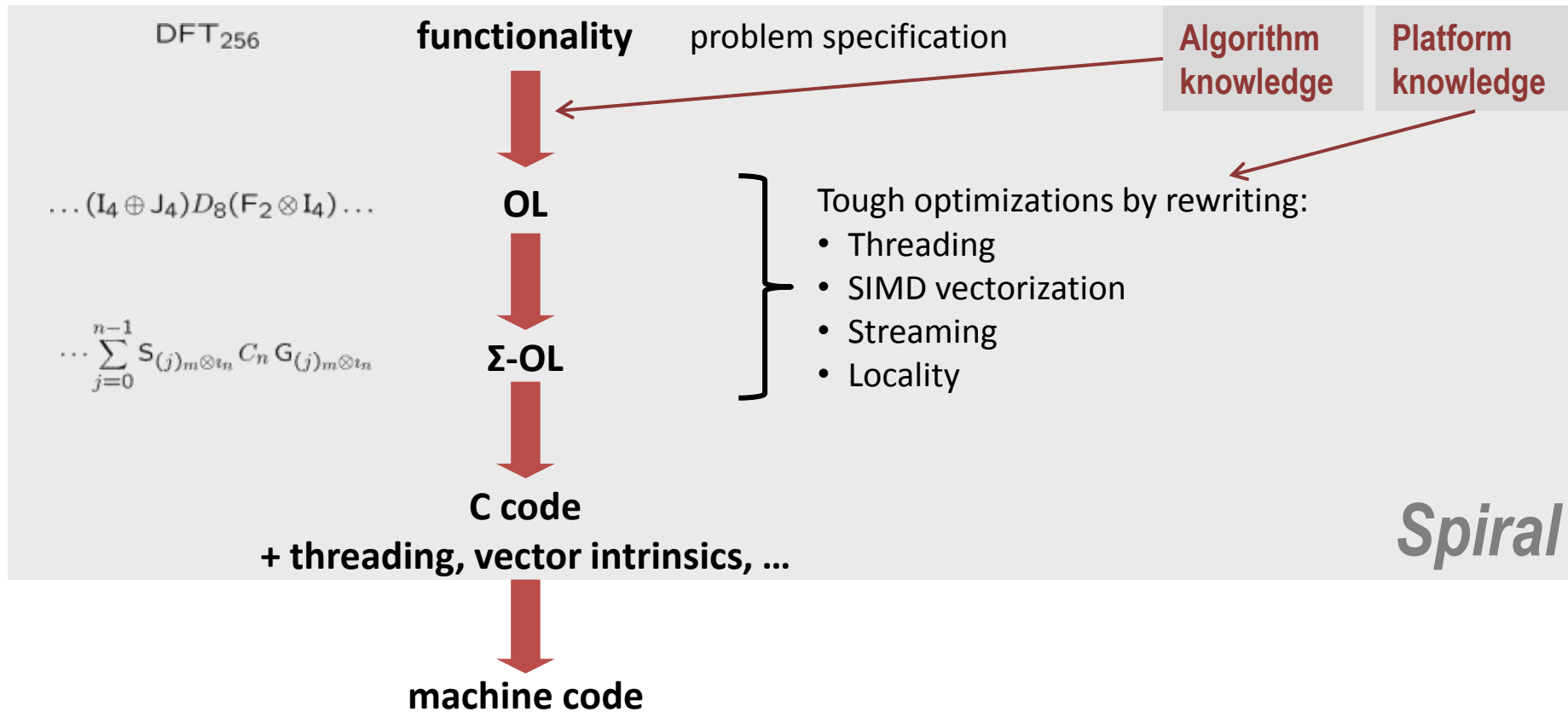
$r = (A_{M \times N \rightarrow K} \otimes I_{m \times n \rightarrow mn})(x, y)$ 

```

for (i=0;i<m;i++)
  for (j=0;j<n;j++)
    r[i*m+j*m*n:i*m+j*m*n*(K-1)] =
      A(x[i:m:i+m*(M-1)], y[j:n:j+n*(N-1)]);

```

# Program Generation in Spiral (Sketched)



- **Optimization at the high level of abstraction:**  
Overcomes compiler limitations
- **Complete automation**

# Organization

- Spiral overview
- Spiral's formal framework
- **Parallelization in Spiral**
- Generating general-size libraries
- Results
- Concluding remarks

F. Franchetti, M. Püschel, Y. Voronenko, S. Chellappa, and J. M. F. Moura:

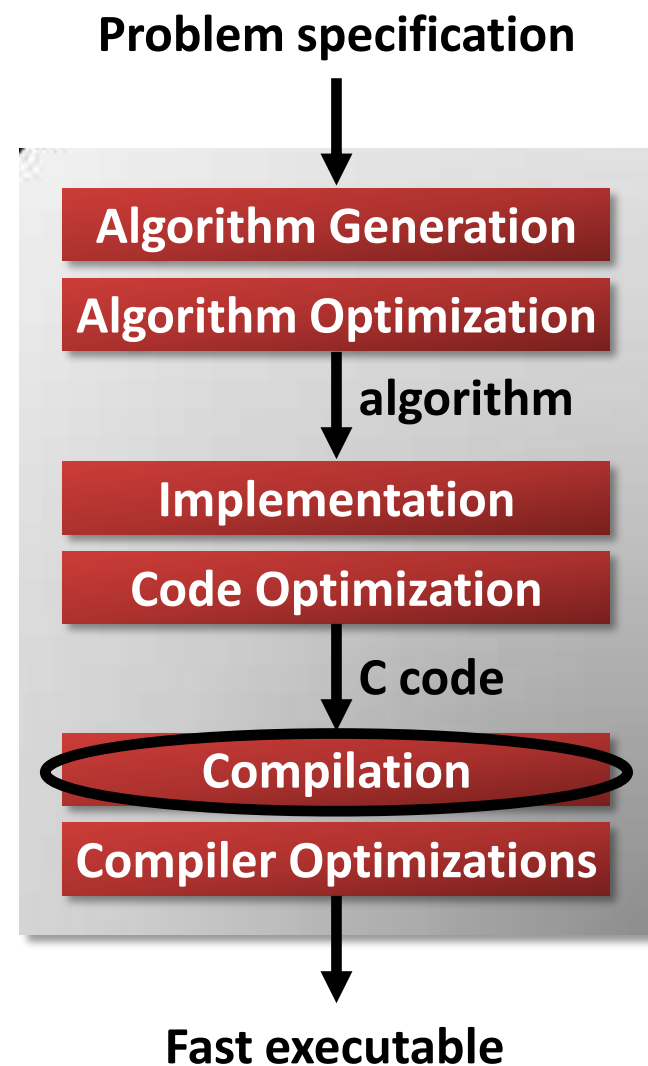
**Discrete Fourier Transform On Multicore.** In IEEE Signal Processing Magazine, November 2009.

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.

# Types of Parallelism

- Multithreading (Multicore)
- Vector SIMD (SSE, VMX/AltiVec,...)
- Message Passing (Clusters, MPP)
- Streaming/multibuffering (Cell)
- Graphics Processors (GPUs)
- Gate-level parallelism (FPGA)
- HW/SW partitioning (CPU + FPGA)

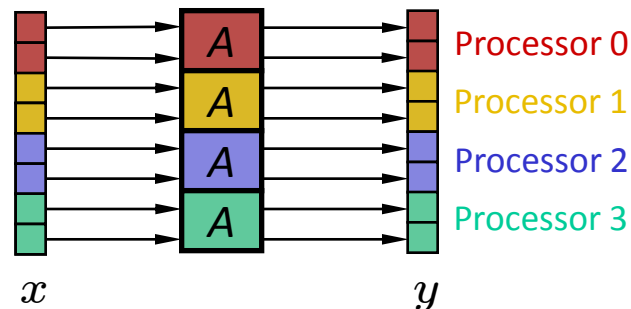


*Spiral: One methodology optimizes for all types of parallelism*

# SPL to Shared Memory Code: Basic Idea

- Key construct: Tensor product

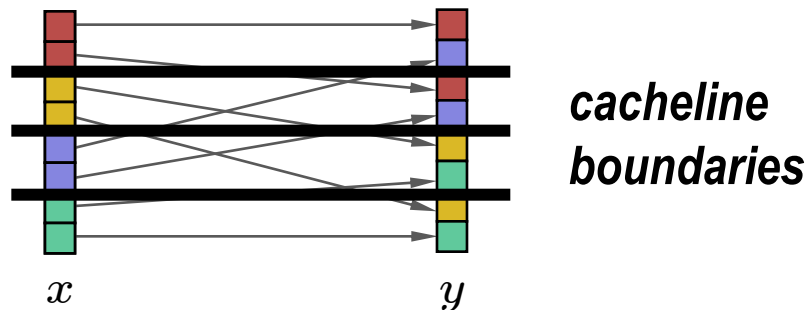
$$y = \left( I_p \otimes A \right) x$$



*p-way embarrassingly parallel, load-balanced*

- Problematic construct: Permutations produce false sharing

$$y = L_4^8 x$$



*cacheline boundaries*

*Task: Rewrite SPL formulas to extract tensor product + avoid false sharing*

# Optimization Knowledge: Rewriting Rules

## ■ Goal: Transform formulas into fully optimized formulas

- Formulas rewritten, tags propagated
- There may be choices

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

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

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

Arity (1,1) rules

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

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

Arity (2,1) rules

# DFT: Parallelization by Rewriting

$$\begin{aligned}
 \underbrace{\text{DFT}_{mn}}_{\text{smp}(p,\mu)} &\rightarrow \underbrace{\left( (\text{DFT}_m \otimes \text{I}_n) \text{T}_n^{mn} (\text{I}_m \otimes \text{DFT}_n) \text{L}_m^{mn} \right)}_{\text{smp}(p,\mu)} \\
 \dots & \\
 &\rightarrow \underbrace{\left( \text{DFT}_m \otimes \text{I}_n \right)}_{\text{smp}(p,\mu)} \underbrace{\text{T}_n^{mn}}_{\text{smp}(p,\mu)} \underbrace{\left( \text{I}_m \otimes \text{DFT}_n \right)}_{\text{smp}(p,\mu)} \underbrace{\text{L}_m^{nm}}_{\text{smp}(p,\mu)} \\
 \dots & \\
 &\rightarrow \underbrace{\left( (\text{L}_m^{mp} \otimes \text{I}_{n/p\mu}) \otimes_{\mu} \text{I}_{\mu} \right)}_{\text{red}} \underbrace{\left( \text{I}_p \otimes_{\parallel} (\text{DFT}_m \otimes \text{I}_{n/p}) \right)}_{\text{blue}} \underbrace{\left( (\text{L}_p^{mp} \otimes \text{I}_{n/p\mu}) \otimes_{\mu} \text{I}_{\mu} \right)}_{\text{red}} \\
 &\quad \underbrace{\left( \bigoplus_{i=0}^{p-1} \text{T}_n^{mn,i} \right)}_{\text{blue}} \underbrace{\left( \text{I}_p \otimes_{\parallel} (\text{I}_{m/p} \otimes \text{DFT}_n) \right)}_{\text{blue}} \underbrace{\left( \text{I}_p \otimes_{\parallel} \text{L}_{m/p}^{mn/p} \right)}_{\text{blue}} \underbrace{\left( (\text{L}_p^{pn} \otimes \text{I}_{m/p\mu}) \otimes_{\mu} \text{I}_{\mu} \right)}_{\text{red}}
 \end{aligned}$$

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

# MMM: Parallelization Through Rewriting

$$\begin{aligned}
 & \underbrace{\text{MMM}_{m,n,k}}_{\text{smp}(p,\mu)} \\
 \rightarrow & \underbrace{\left( \text{I}_m \otimes \text{L}_p^n \right) \circ \left( \text{MMM}_{m,n/p,k} \otimes (\otimes)_{1 \times p \rightarrow p} \right) \circ \left( \text{I}_{km} \times (\text{I}_k \otimes \text{L}_{n/p}^n) \right)}_{\text{smp}(p,\mu)} \\
 \rightarrow & \underbrace{\left( \text{I}_m \otimes \text{L}_p^n \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\left( \text{MMM}_{m,n/p,k} \otimes (\otimes)_{1 \times p \rightarrow p} \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\left( \text{I}_{km} \times (\text{I}_k \otimes \text{L}_{n/p}^n) \right)}_{\text{smp}(p,\mu)} \\
 \rightarrow & \underbrace{\left( \text{I}_m \otimes \text{L}_p^n \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\text{L}_{m/pn}^{mn}}_{\text{smp}(p,\mu)} \circ \underbrace{\left( (\otimes)_{1 \times p \rightarrow p} \otimes_{\parallel} \text{MMM}_{m/p,n,k} \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\left( \text{I}_{km} \times \text{L}_p^{kn} \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\left( \text{I}_{km} \times (\text{I}_k \otimes \text{L}_{n/p}^n) \right)}_{\text{smp}(p,\mu)} \\
 \rightarrow & \underbrace{\left( (\text{L}_m^{mp} \otimes \text{I}_{n/(p\mu)}) \bar{\otimes} \text{I}_\mu \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\left( (\otimes)_{1 \times p \rightarrow p} \otimes_{\parallel} \text{MMM}_{m,n/p,k} \right)}_{\text{smp}(p,\mu)} \circ \underbrace{\left( (\text{I}_{km/\mu} \bar{\otimes} \text{I}_\mu) \times \left( (\text{L}_p^{kp} \otimes \text{I}_{n/(p\mu)}) \bar{\otimes} \text{I}_\mu \right) \right)}_{\text{smp}(p,\mu)}
 \end{aligned}$$

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

# Same Approach for Other Parallel Paradigms

## Message Passing:

$$\begin{aligned}
 \underbrace{\text{DFT}_{mp}}_{\text{msg}(p,\mu)} &\rightarrow \underbrace{((\text{DFT}_m \otimes I_n) \tau_n^{mn} (I_m \otimes \text{DFT}_n) L_m^{mn})}_{\text{msg}(p,\mu)} \\
 &\dots \\
 &\rightarrow \underbrace{(\text{DFT}_m \otimes I_n)}_{\text{msg}(p,\mu)} \underbrace{\tau_n^{mn}}_{\text{msg}(p,\mu)} \underbrace{(I_m \otimes \text{DFT}_n)}_{\text{msg}(p,\mu)} \underbrace{L_m^{mn}}_{\text{msg}(p,\mu)} \\
 &\dots \\
 &\rightarrow ((L_m^{mp} \otimes I_{n/p\mu}) \otimes I_\mu) (I_p \otimes \| (\text{DFT}_m \otimes I_{n/p})) ((L_p^{mp} \otimes I_{n/p\mu}) \otimes I_\mu) \\
 &\quad \left( \bigoplus_{i=0}^{p-1} \tau_n^{mn,i} \right) (I_p \otimes \| (I_{m/p} \otimes \text{DFT}_n)) (I_p \otimes \| L_{m/p}^{mn/p}) ((L_p^{pn} \otimes I_{m/p\mu}) \otimes I_\mu)
 \end{aligned}$$

## Vectorization:

$$\begin{aligned}
 \underbrace{(\text{DFT}_{mn})}_{\text{vec}(\nu)} &\rightarrow \underbrace{((\text{DFT}_m \otimes I_n) \tau_n^{mn} (I_m \otimes \text{DFT}_n) L_m^{mn})}_{\text{vec}(\nu)} \\
 &\dots \\
 &\rightarrow \underbrace{(\text{DFT}_m \otimes I_n)^\nu}_{\text{vec}(\nu)} \underbrace{(\tau_n^{mn})^\nu}_{\text{vec}(\nu)} \underbrace{(I_m \otimes \text{DFT}_n) L_m^{mn}}_{\text{vec}(\nu)} \\
 &\dots \\
 &\rightarrow (I_{mn/\nu} \otimes \underbrace{L_{\nu}^{2\nu}}_{\text{sse}}) (\text{DFT}_m \otimes I_{n/\nu} \otimes I_\nu) \underbrace{(\tau_n^{mn})^\nu}_{\text{sse}} \\
 &\quad (I_{m/\nu} \otimes \underbrace{(L_{\nu}^{2\nu} \otimes I_\nu)}_{\text{sse}}) (I_{n/\nu} \otimes \underbrace{(L_{\nu}^{2\nu} \otimes I_\nu)}_{\text{sse}}) (I_2 \otimes \underbrace{L_{\nu}^{\nu^2}}_{\text{sse}}) (L_2^{\nu^2} \otimes I_\nu) (\text{DFT}_n \otimes I_\nu) \\
 &\quad ((L_m^{mn} \otimes I_2) \otimes I_\nu) (I_{mn/\nu} \otimes \underbrace{L_{\nu}^{2\nu}}_{\text{sse}})
 \end{aligned}$$

## GPUs:

$$\begin{aligned}
 \underbrace{(\text{DFT}_{r^k})}_{\text{gpu}(t,c)} &\rightarrow \underbrace{\left( \prod_{i=0}^{k-1} L_r^{r^k} (I_{r^{k-1}} \otimes \text{DFT}_r) (L_{r^{k-i-1}} (I_{r^i} \otimes \tau_{r^{k-i-1}}) L_{r^{i+1}}^{r^k}) \right)}_{\text{gpu}(t,c)} R_r^{r^k} \\
 &\dots \\
 &\rightarrow \left( \prod_{i=0}^{k-1} (L_r^{r^n/2} \otimes I_2) (I_{r^{n-1}/2} \otimes \underbrace{(\text{DFT}_r \otimes I_2) L_r^{2r}}_{\text{shd}(t,c)} \tau_i) \right) \\
 &\quad (L_r^{r^n/2} \otimes I_2) (I_{r^{n-1}/2} \otimes \underbrace{L_r^{2r}}_{\text{shd}(t,c)}) (R_r^{r^{n-1}} \otimes I_r)
 \end{aligned}$$

## Verilog for FPGAs:

$$\begin{aligned}
 \underbrace{(\text{DFT}_{r^k})}_{\text{stream}(r^k)} &\rightarrow \underbrace{\left[ \prod_{i=0}^{k-1} L_r^{r^k} (I_{r^{k-1}} \otimes \text{DFT}_r) (L_{r^{k-i-1}} (I_{r^i} \otimes \tau_{r^{k-i-1}}) L_{r^{i+1}}^{r^k}) \right]}_{\text{stream}(r^k)} R_r^{r^k} \\
 &\dots \\
 &\rightarrow \left[ \prod_{i=0}^{k-1} \underbrace{L_r^{r^k}}_{\text{stream}(r^k)} \underbrace{(I_{r^{k-1}} \otimes \text{DFT}_r)}_{\text{stream}(r^k)} \underbrace{(L_{r^{k-i-1}} (I_{r^i} \otimes \tau_{r^{k-i-1}}) L_{r^{i+1}}^{r^k})}_{\text{stream}(r^k)} \right]_{\text{stream}(r^k)} R_r^{r^k} \\
 &\dots \\
 &\rightarrow \left[ \prod_{i=0}^{k-1} \underbrace{L_r^{r^k}}_{\text{stream}(r^k)} (I_{r^{k-i-1}} \otimes \tau_{r^{k-i-1}} \otimes \text{DFT}_r) \right]_{\text{stream}(r^k)} \underbrace{\tau_i}_{\text{stream}(r^k)} \underbrace{R_r^{r^k}}_{\text{stream}(r^k)}
 \end{aligned}$$

- Rigorous, correct by construction
- Overcomes compiler limitations



# Organization

- Spiral overview
- Spiral's formal framework
- Parallelization in Spiral
- **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.

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

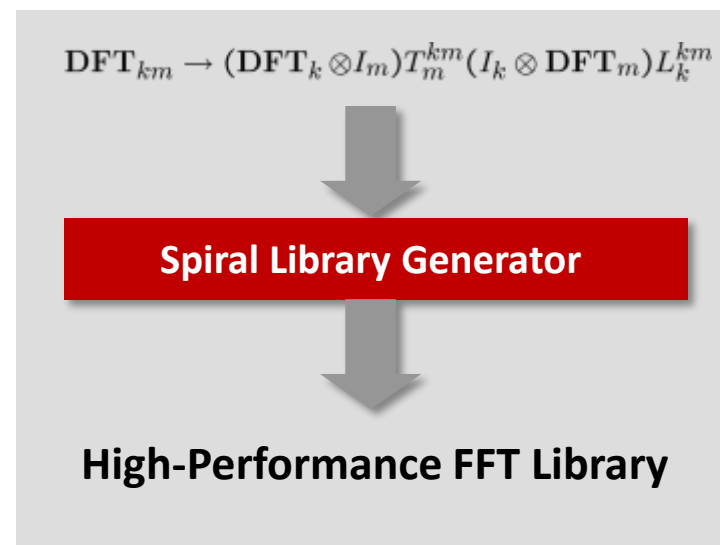
# General-Size Library

## Input:

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

## 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”, “MKL”,  
“FFTPACK”, “ESSL”

“Cooley-Tukey” DCT



**Spiral**



“FFTW”, “IPP”

Overlap-save/add FIR



**Spiral**



“FIRW”, “IPP”

MMM, blocking



**Spiral**



“ATLAS”, “MKL”

Fast Wavelet Transform



**Spiral**



“FWTW”

Fast Hartley Transform



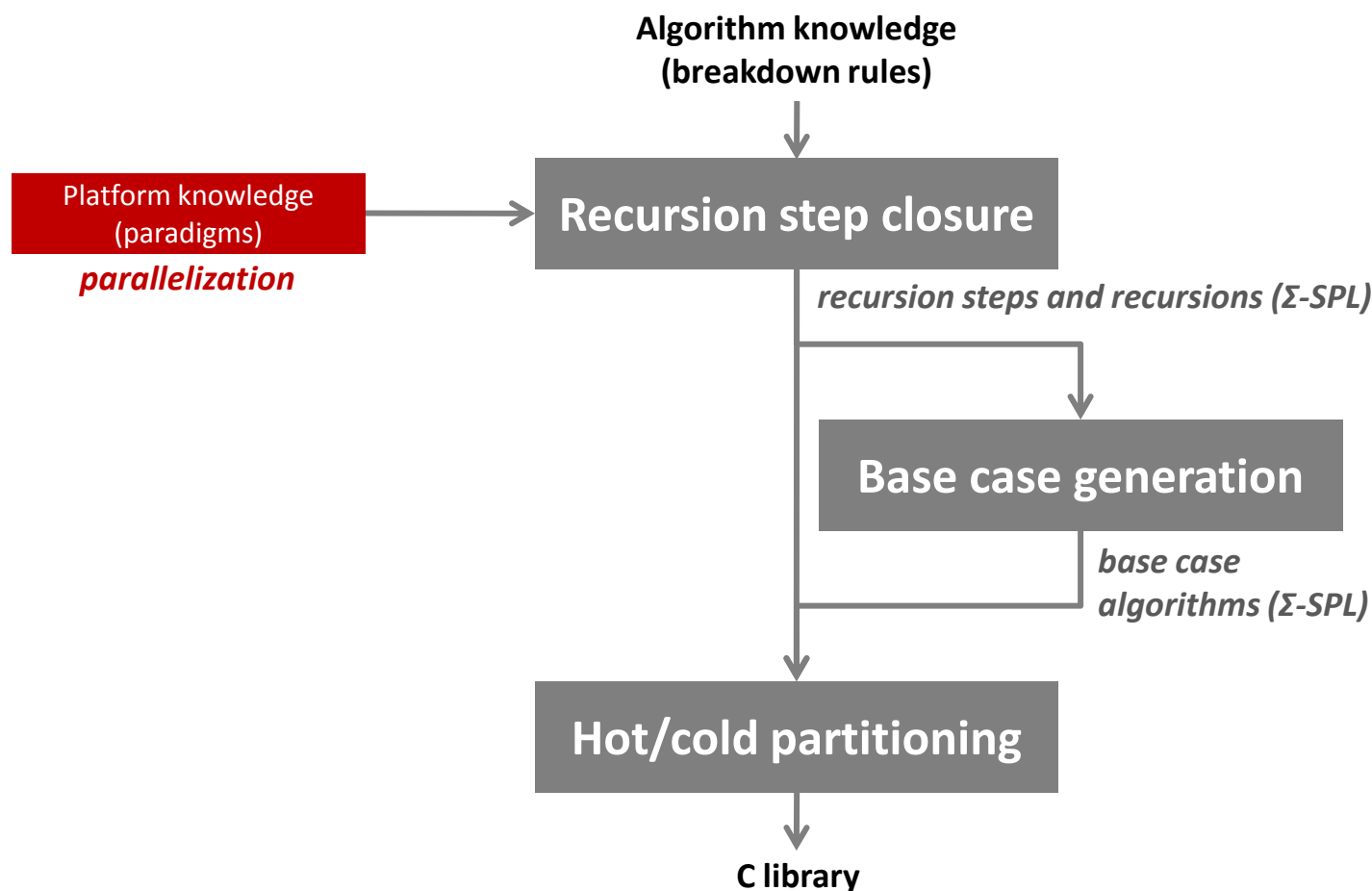
**Spiral**



“FHTW”

Y. Voronenko's PhD Thesis: 50+ “FFTW-like” libraries

# General Size Library Generator



- Same breakdown rules and paradigms as fixed-size Spiral
- Codelets are automatically discovered and built (fixed-size Spiral)
- Adaptive library infrastructure is automatically derived and built

# Core Idea: Recursion Step Closure

- **Input:** transform T and a breakdown rules
- **Output:** problem specifications for recursive function and codelets

- **Algorithm:**

1. Apply the breakdown rule

$$\{\text{DFT}_n\}$$

$$\downarrow$$

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

2. Convert to  $\Sigma$ -SPL

$$\left( \sum_{i=0}^{k-1} S_{h_{i,k}} \{\text{DFT}_{n/k}\} G_{h_{i,k}} \right) \text{diag}(f) \left( \sum_{j=0}^{n/k-1} S_{h_{j,k,1}} \{\text{DFT}_k\} G_{h_{j,k,1}} \right) \text{perm}(\ell_{n/k}^n)$$

3. Apply loop merging + index simplification rules.

$$\sum_{i=0}^{k-1} S_{h_{i,k}} \{\text{DFT}_{n/k}\} \text{diag}(f \circ h_{i,k}) G_{h_{i,k}} \quad \sum_{j=0}^{n/k-1} S_{h_{j,k,1}} \{\text{DFT}_k\} G_{h_{j,n/k}}$$

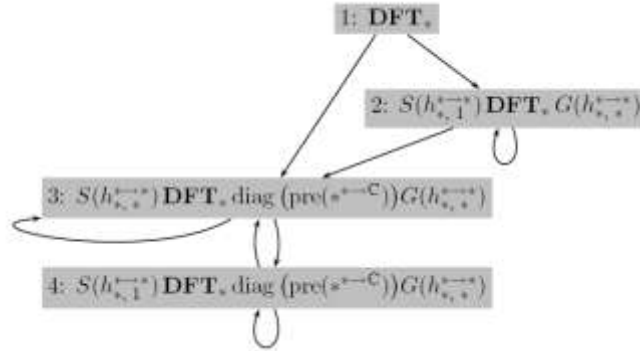
4. Extract recursion steps

$$\sum_{i=0}^{k-1} \left\{ S_{h_{i,k}} \text{DFT}_{n/k} \text{diag}(f \circ h_{i,k}) G_{h_{i,k}} \right\} \quad \sum_{j=0}^{n/k-1} \left\{ S_{h_{j,k,1}} \text{DFT}_k G_{h_{j,n/k}} \right\}$$

5. Repeat until closure is reached

# Recursion Step Closure: Examples

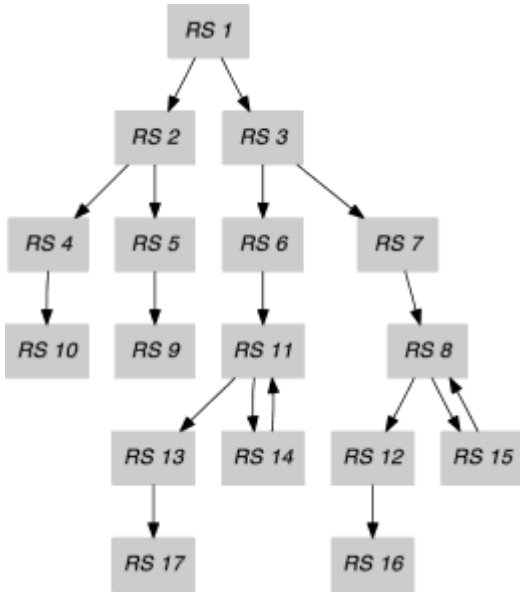
## DFT (scalar)



**Mutually recursive functions**

- computed automatically
- described using  $\Sigma$ -SPL formulas

## DCT4 (vectorized)



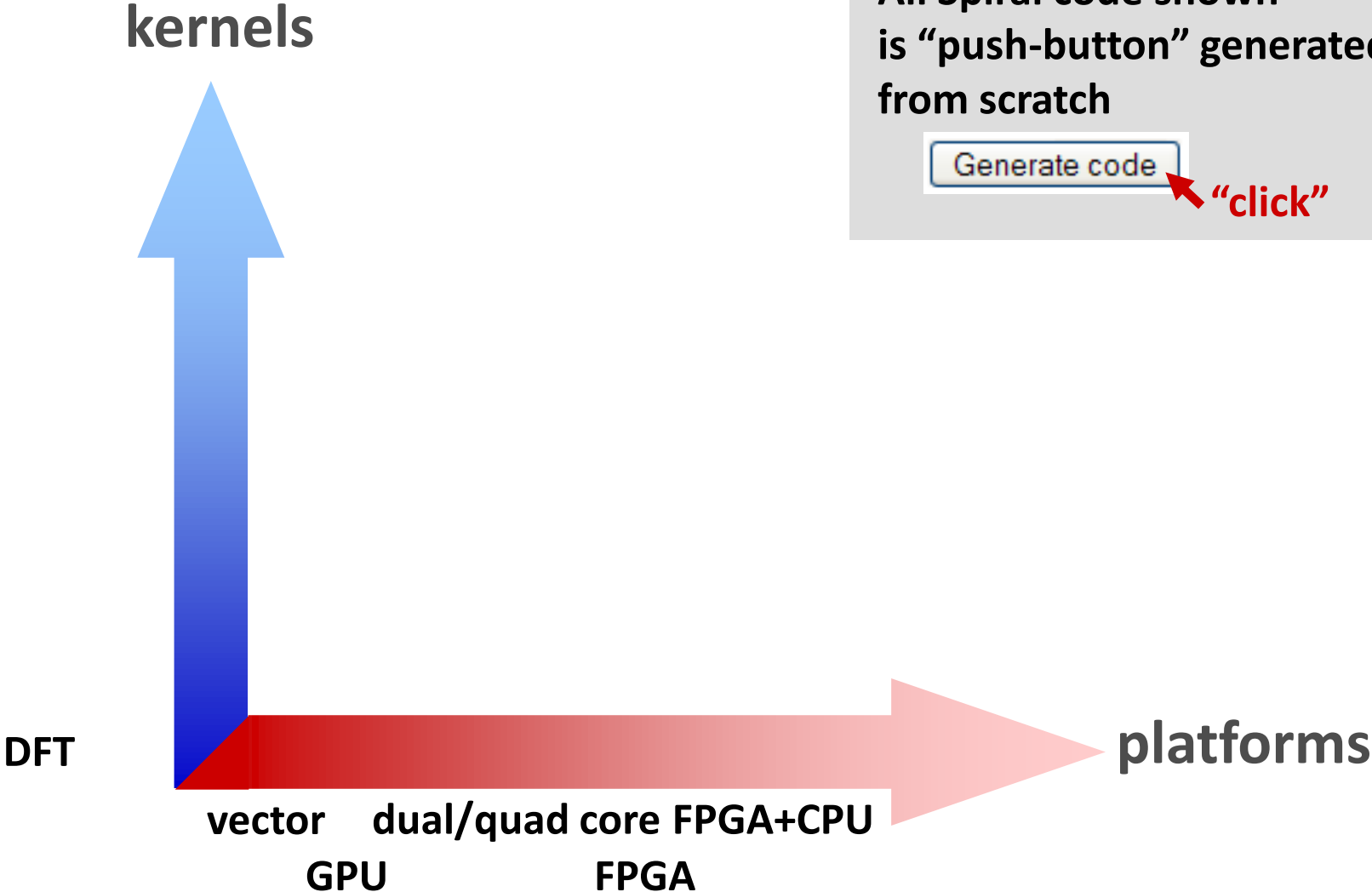
- 1:  $\text{Vec}_2(\text{DCT-4}_{u_1})$
- 2:  $\text{Vec}_2(\text{GT}(\text{diag}(N_{2u_8}) \text{RDFT-3}_{2u_8}^\top \text{reddiag}(\text{pre}(u_4^{\mathbb{Z} \times 2u_8 \rightarrow \mathbb{R}})), h_{0,1,u_7}^{2u_8 \rightarrow u_6} \circ \ell_{u_8}^{2u_8}, r_{0,u_{11},1,u_{12}}^{2u_8 \rightarrow u_9}, \{u_{13}\}))$
- 3:  $\text{Vec}_2(\text{GT}(\text{RDFT-3}_{u_1} \text{diag}(N_{u_1}), r_{0,u_5,1,u_6}^{u_1 \rightarrow u_3}, h_{0,u_9,1}^{u_1 \rightarrow u_8}, \{u_{10}\}))$
- 4:  $\text{VJam}_2(\text{GT}(\text{diag}(N_{2u_9}) \text{RDFT-3}_{2u_9}^\top \text{reddiag}(\text{pre}(u_4^{\mathbb{Z} \times 2u_9 \rightarrow \mathbb{R}})), h_{0,1,u_7,u_8}^{2u_9 \rightarrow u_6} \circ \ell_{u_9}^{2u_9}, r_{0,u_{12},1,2,u_{13}}^{2u_9 \rightarrow u_{10}}, \{2, u_{14}\}))$
- 5:  $\text{GT}(\text{diag}(N_{2u_9}) \text{RDFT-3}_{2u_9}^\top \text{reddiag}(\text{pre}(u_4^{\mathbb{Z} \times 2u_9 \rightarrow \mathbb{R}})), h_{u_7,1,u_8}^{2u_9 \rightarrow u_6} \circ \ell_{u_9}^{2u_9}, r_{u_{12},u_{13},1,u_{14}}^{2u_9 \rightarrow u_{10}}, \{u_{15}\}))$
- 6:  $\text{VJam}_2(\text{GT}(\text{RDFT-3}_{u_1} \text{diag}(N_{u_1}), r_{0,u_5,1,2,u_6}^{u_1 \rightarrow u_3}, h_{0,u_9,1,2}^{u_1 \rightarrow u_8}, \{2, u_{10}\}))$
- 7:  $\text{GT}(\text{RDFT-3}_{u_1} \text{diag}(N_{u_1}), r_{u_5,u_6,1,u_7}^{u_1 \rightarrow u_3}, h_{u_{10},u_{11},1}^{u_1 \rightarrow u_8}, \{u_{12}\}))$
- 8:  $S(h_{u_3,u_4}^{u_1 \rightarrow u_2}) \text{RDFT-3}_{u_1} \text{diag}(N_{u_1}) G(r_{u_9,u_{10},u_{11}}^{u_1 \rightarrow u_7})$
- 9:  $S(r_{u_3,u_4,u_5}^{2u_{13} \rightarrow u_1}) \text{diag}(N_{2u_{13}}) \text{RDFT-3}_{2u_{13}}^\top \text{reddiag}(\text{pre}(u_9^{2u_{13} \rightarrow \mathbb{R}})) G(h_{u_{12},1}^{2u_{13} \rightarrow u_{11}} \circ \ell_{u_{13}}^{2u_{13}})$
- 10:  $\text{VJam}_2(\text{GT}(\text{diag}(N_{2u_9}) \text{RDFT-3}_{2u_9}^\top \text{reddiag}(\text{pre}(u_4^{\mathbb{Z} \times 2u_9 \rightarrow \mathbb{R}})), h_{u_7,1,u_8}^{2u_9 \rightarrow u_6} \circ \ell_{u_9}^{2u_9}, r_{u_{12},u_{13},1,u_{14}}^{2u_9 \rightarrow u_{10}}, \{2\}))$
- 11:  $\text{VJam}_2(\text{GT}(\text{RDFT-3}_{u_1} \text{diag}(N_{u_1}), r_{u_5,u_6,1,u_7}^{u_1 \rightarrow u_3}, h_{u_{10},u_{11},1}^{u_1 \rightarrow u_8}, \{2\}))$
- 12:  $\text{GT}(\text{diag}(C_{u_1}) \text{rDFT}_{2u_1}(\lambda\text{-wrap}(\lambda_1^{\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}})), h_{0,1,u_5}^{2u_1 \rightarrow u_4}, h_{u_8,u_9}^{2u_{10} \rightarrow u_7} \circ (r_{0,u_{12},1,u_{13}}^{u_1 \rightarrow u_{10}} \otimes t_2), \{u_{14}\}))$
- 13:  $\text{VJam}_2(\text{GT}(\text{diag}(C_{u_1}) \text{rDFT}_{2u_1}(\lambda\text{-wrap}(\lambda_1^{\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}})), h_{u_5,u_6,1,u_7}^{2u_{11} \rightarrow u_4}, h_{u_{10},u_{11},1}^{2u_{12} \rightarrow u_9} \circ (r_{0,u_{14},0,1,u_{15}}^{u_1 \rightarrow u_{12}} \otimes t_2), \{2, u_{16}\}))$
- 14:  $\text{VJam}_2(\text{GT}(\text{RDFT-3}_{u_1} \text{diag}(N_{u_1}), r_{u_5,u_6,1,u_7,u_8}^{u_1 \rightarrow u_3}, h_{u_{11},u_{12},1,u_{13}}^{u_1 \rightarrow u_{10}}, \{2, u_{14}\}))$
- 15:  $\text{GT}(\text{RDFT-3}_{u_1} \text{diag}(N_{u_1}), r_{u_5,u_6,u_7,u_8}^{u_1 \rightarrow u_3}, h_{0,u_{11},1}^{u_1 \rightarrow u_{10}}, \{u_{12}\}))$
- 16:  $S(h_{u_3,u_4}^{2u_5 \rightarrow u_2} \circ (r_{u_7,u_8,u_9}^{u_6 \rightarrow u_5} \otimes t_2)) \text{diag}(C_{u_6}) \text{rDFT}_{2u_6}(\lambda\text{-wrap}(\lambda_1^{\mathbb{Z} \rightarrow \mathbb{R}})) G(h_{u_{14},1}^{2u_6 \rightarrow u_{13}})$
- 17:  $\text{VJam}_2(\text{GT}(\text{diag}(C_{u_1}) \text{rDFT}_{2u_1}(\lambda\text{-wrap}(\lambda_1^{\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}})), h_{u_5,u_6,1}^{2u_{11} \rightarrow u_4}, h_{u_9,u_{10},1}^{2u_{11} \rightarrow u_8} \circ (r_{u_{13},u_{14},u_{15}}^{u_1 \rightarrow u_{11}} \otimes t_2), \{2\}))$

**Closure: formal specification for codelet and infrastructure generation**

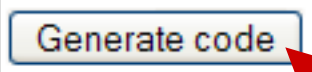
# Organization

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

# Benchmarks



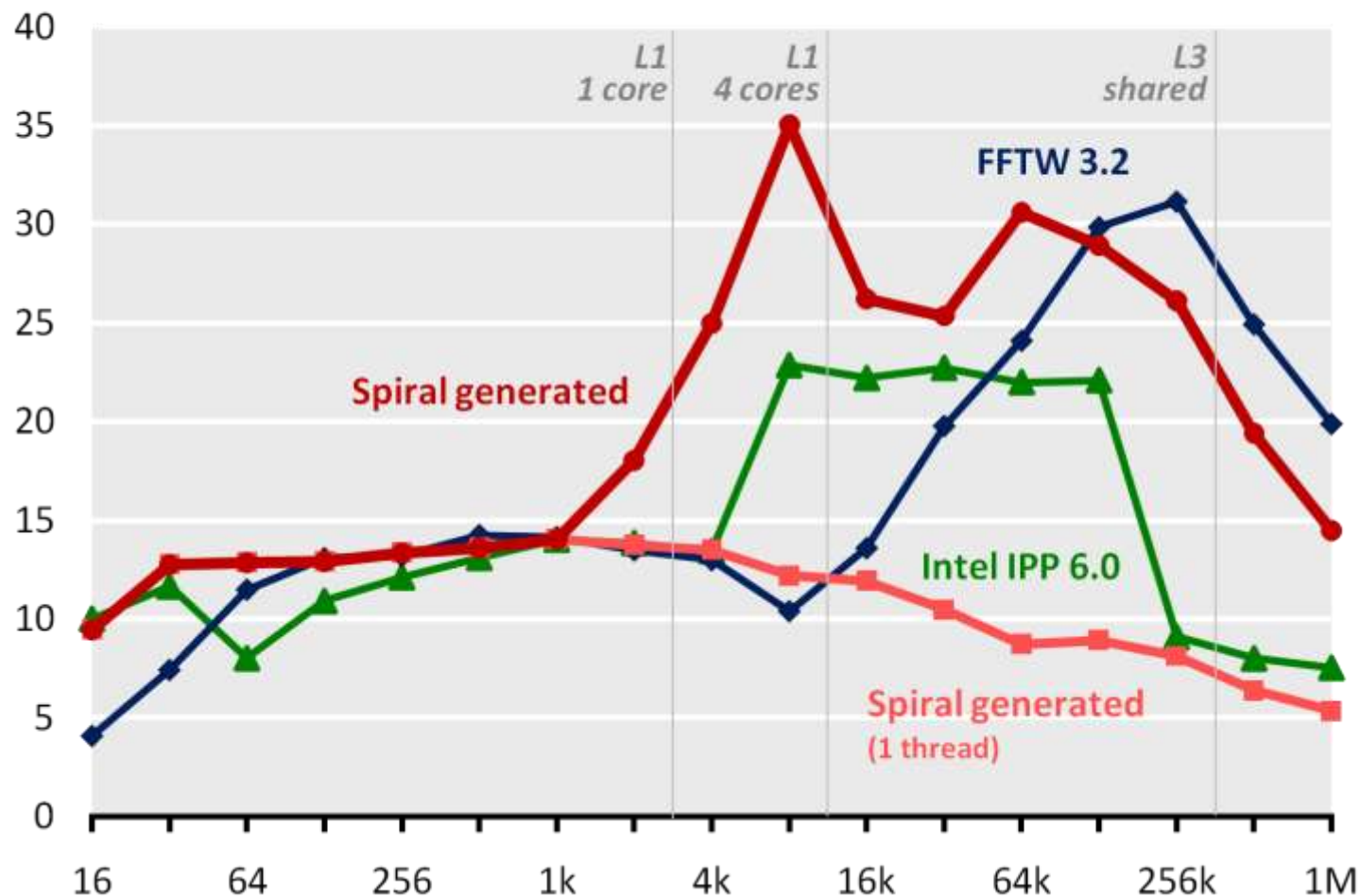
All Spiral code shown is “push-button” generated from scratch



“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).

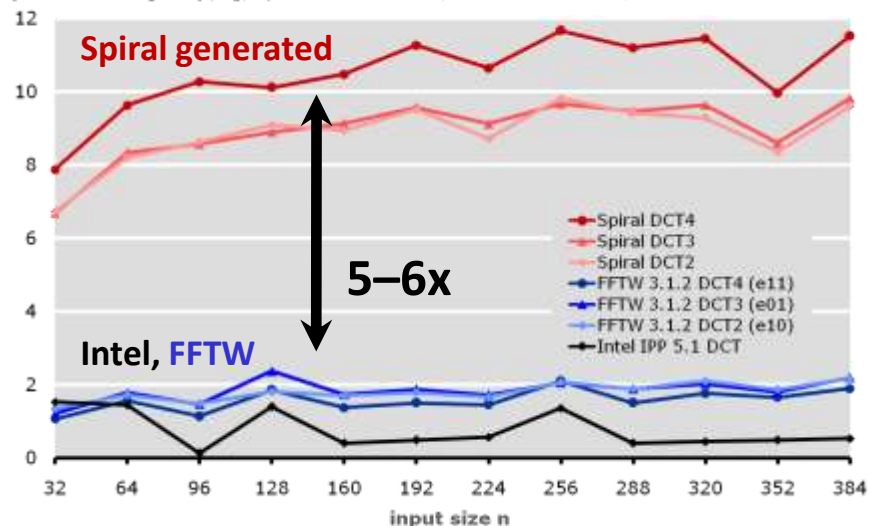
F. Franchetti, Y. Voronenko, and M. Püschel: **FFT Program Generation for Shared Memory: SMP and Multicore.**

In Proceedings of Supercomputing, 2006.

# Intel Multicore: Off The Beaten Path

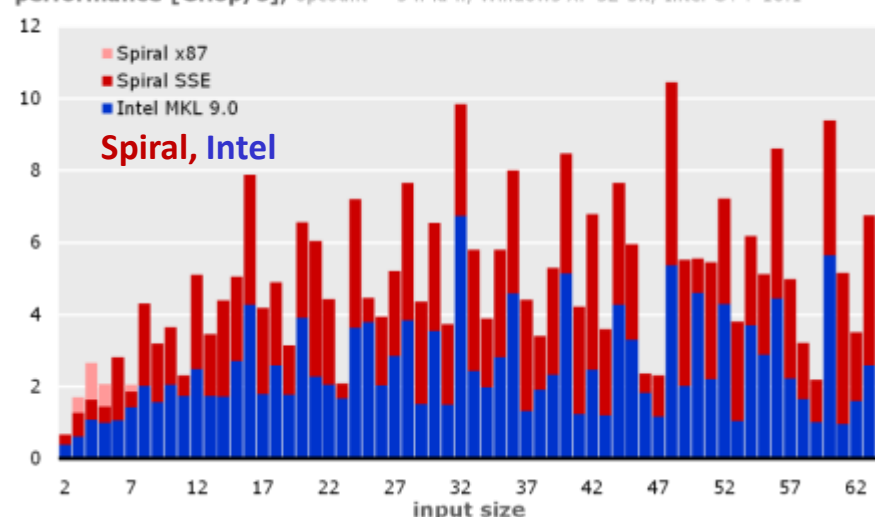
## DCT on 2.66 GHz Core2 (single-precision, 4-way SSSE3)

performance [Gflop/s], opcount =  $2.5 n \log n$ , Windows XP 32-bit, Intel C++ 10.1



## DFT on 2.66 GHz Core2 (single-precision, SSSE3)

performance [Gflop/s], opcount =  $5 n \log n$ , Windows XP 32-bit, Intel C++ 10.1



- 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)

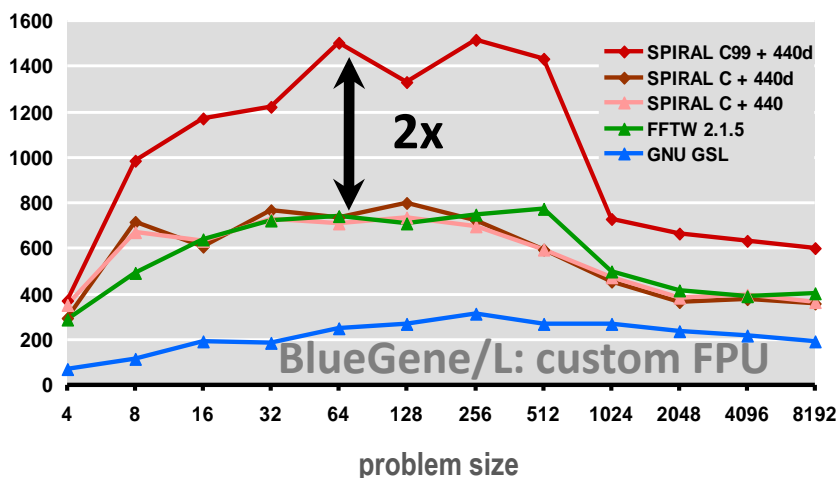
F. Franchetti, M. Püschel:

**SIMD Vectorization of Non-Two-Power Sized FFTs.**

Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2007.

# Single Node: BlueGene Supercomputers

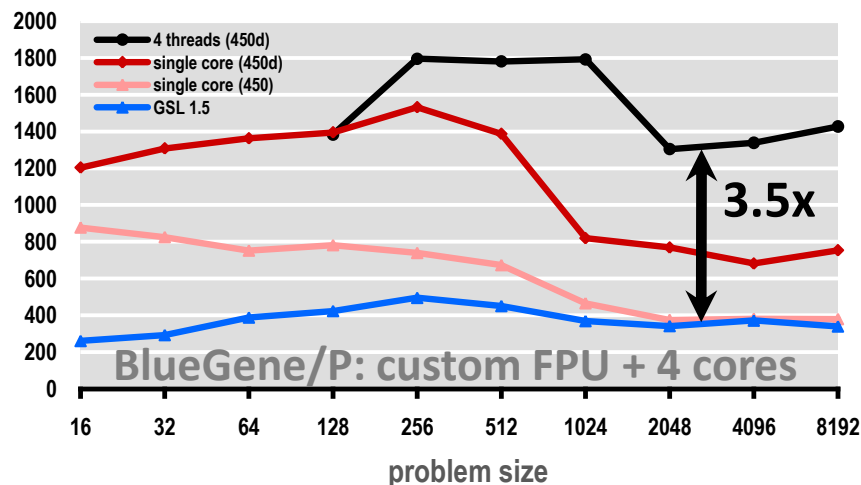
DFT, double precision, XL C compiler  
performance [Mflop/s]



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**

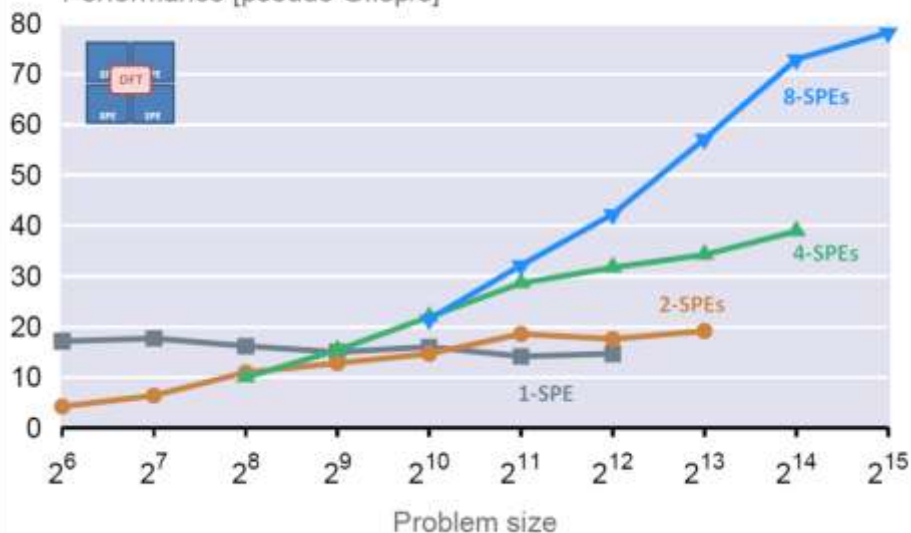
F. Gygi, E. W. Draeger, M. Schulz, B. R. de Supinski, J. A. Gunnels, V. Austel, J. C. Sexton, F. Franchetti, S. Kral, C. W. Ueberhuber, J. Lorenz: **Large-Scale Electronic Structure Calculations of High-Z Metals on the BlueGene/L Platform.** In Proceedings of Supercomputing, 2006. **Winner of the 2006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

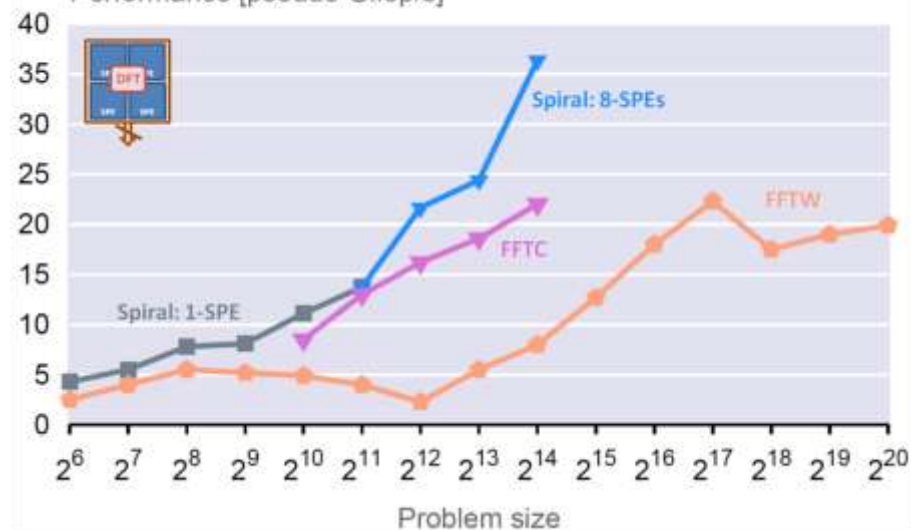
## DFT on multiple SPEs (LS-LS, block-cyclic)

Performance [pseudo Gflop/s]



## DFT on Multiple SPEs (Mem-mem)

Performance [pseudo Gflop/s]



### Single DFT, latency optimized

- Local store resident
- parallelized across SPEs
- Block-cyclic data format

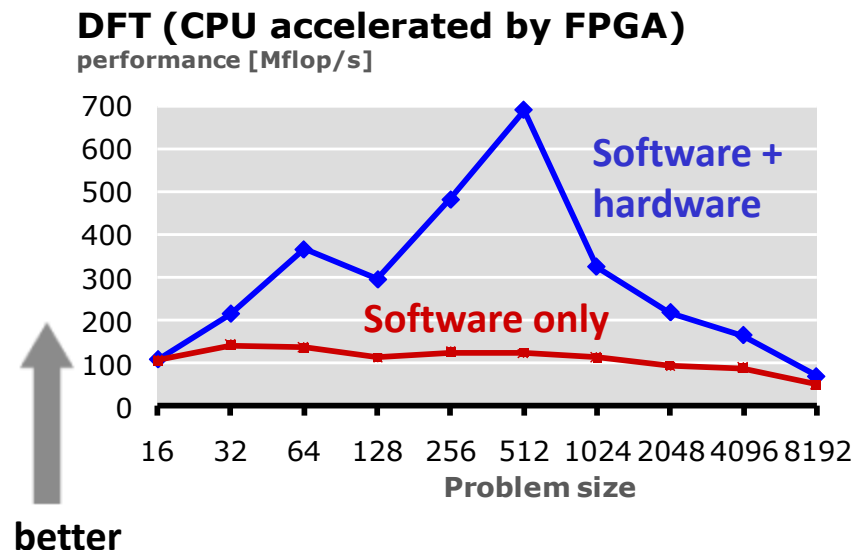
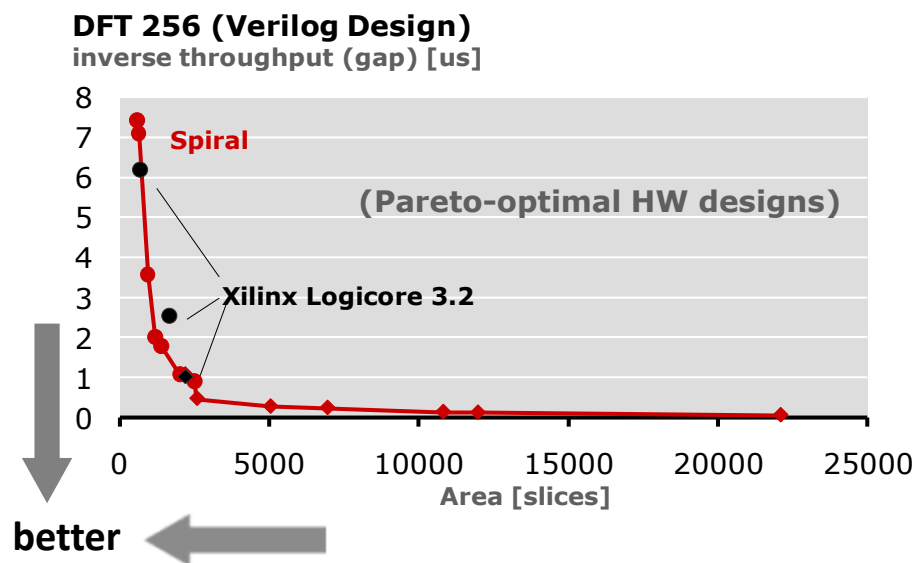
### Vectorization and parallelization

### Single DFT, latency optimized

- Data in XDRAM resident
- parallelized across SPEs
- standard data format

### Vectorization, parallelization, and streaming

# 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.

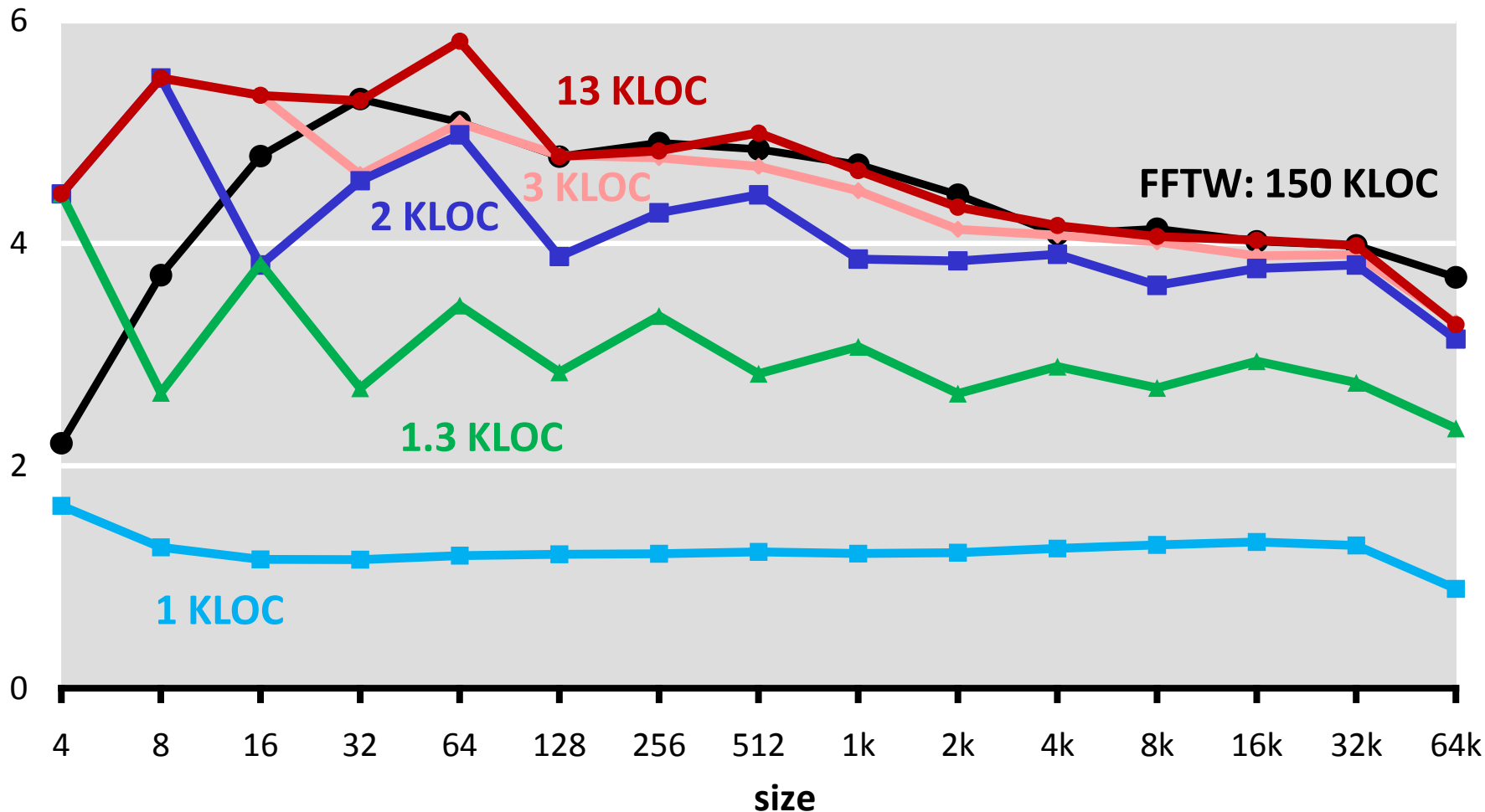
P. D'Alberto, F. Franchetti, P. A. Milder, A. Sandryhaila, J. C. Hoe, J. M. F. Moura, and M. Püschel:

**Generating FPGA Accelerated DFT Libraries.**

In Proceedings of Field-Programmable Custom Computing Machines (FCCM), 2007.

# General Size Library Customization: Code Size

Performance [Gflop/s]

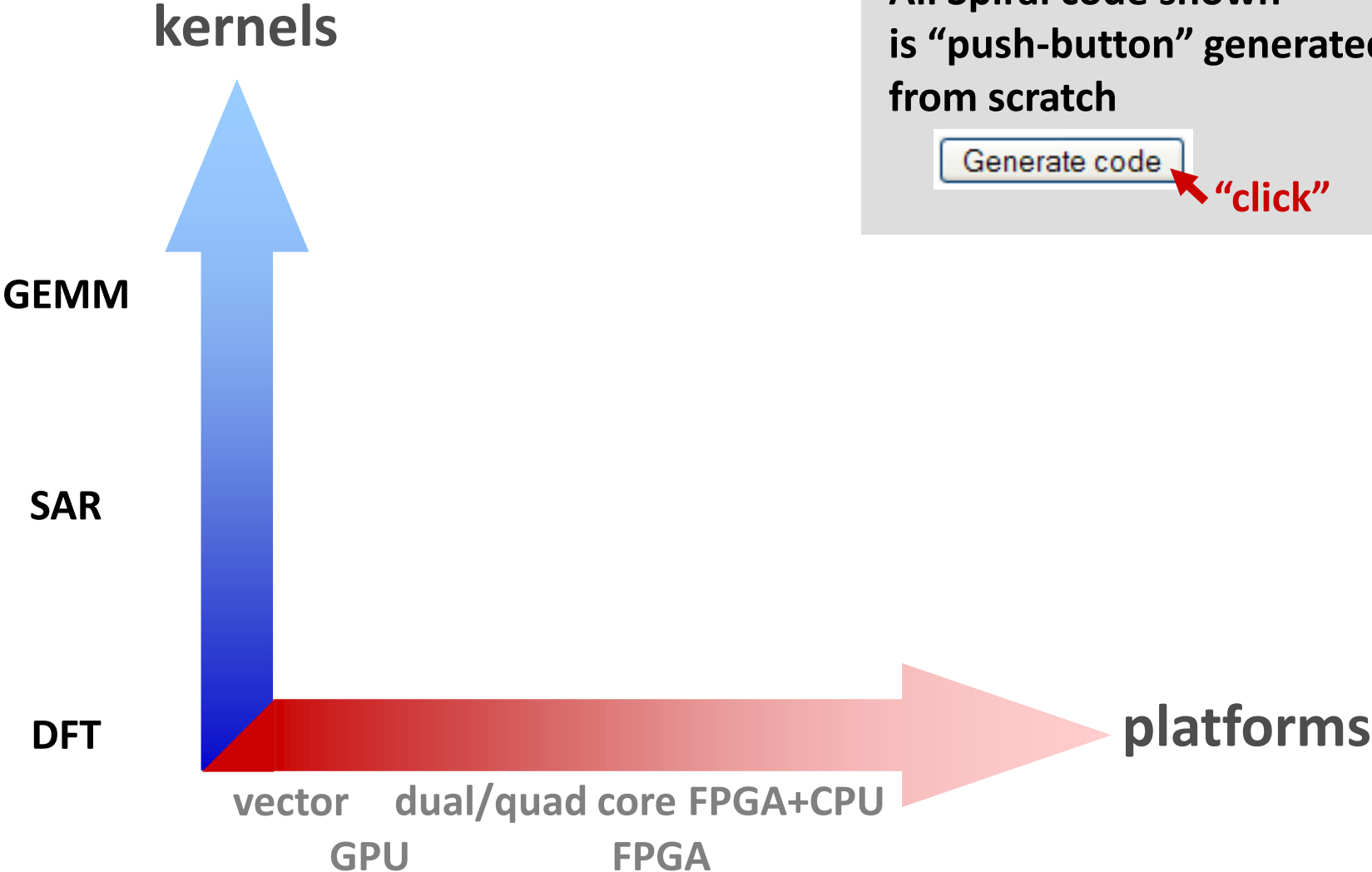


# Benchmarks

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

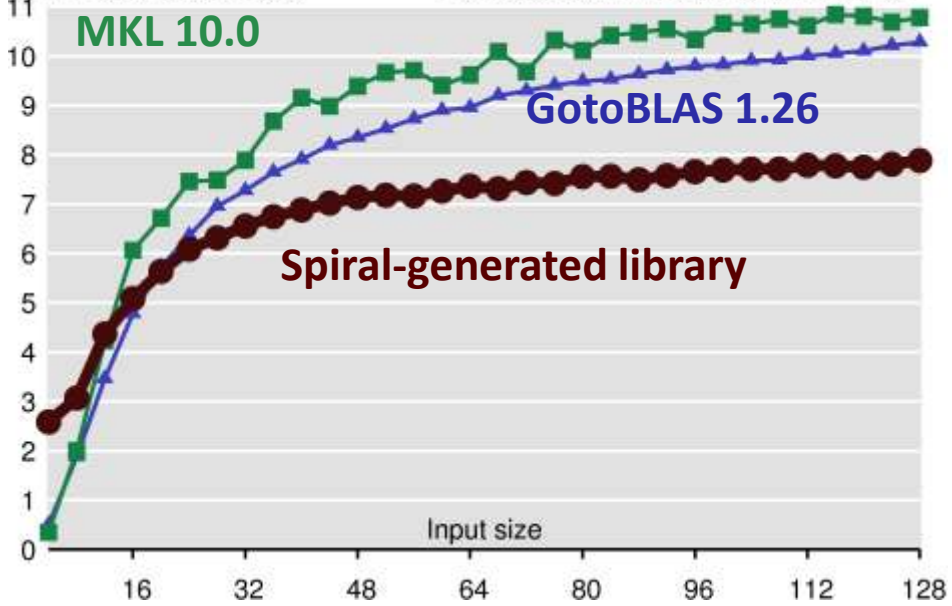
Generate code

“click”

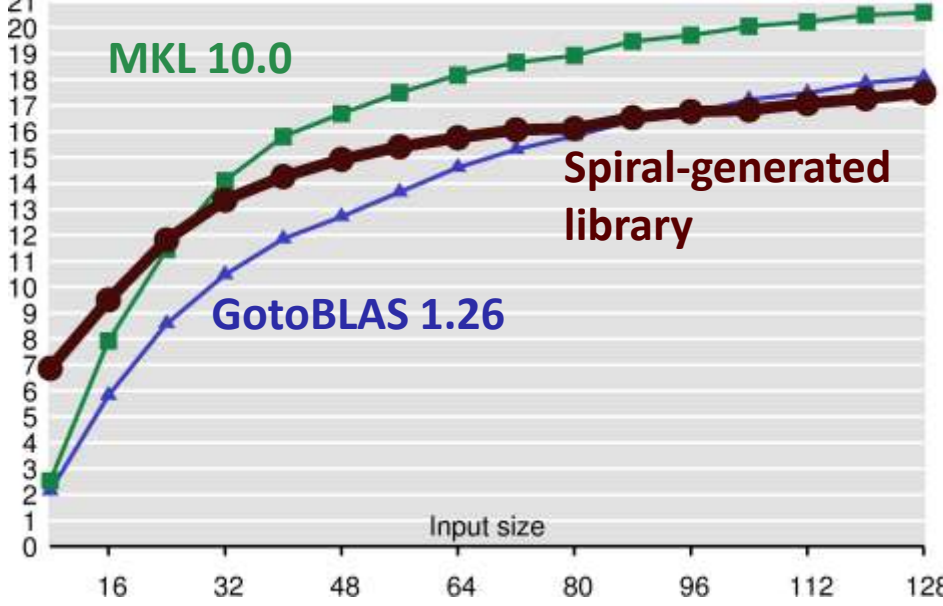


# Result: Matrix Multiplication Library

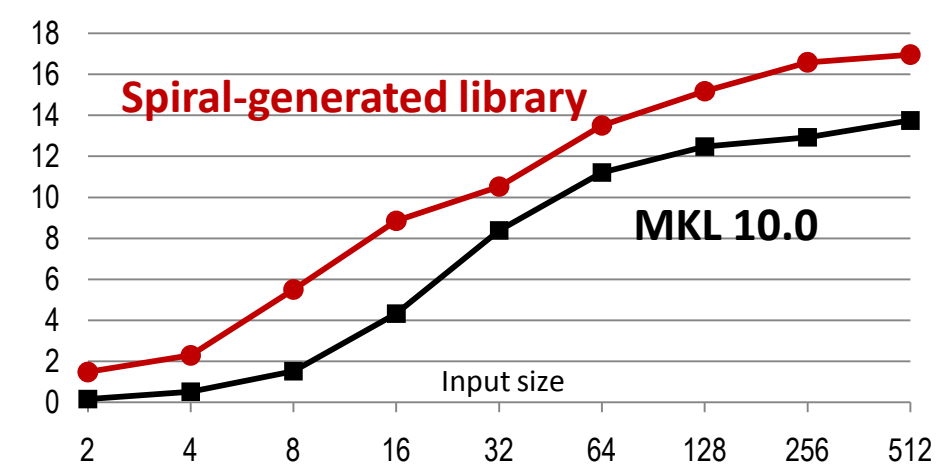
GEMM (acc. packed sq. NN), single-threaded, double precision  
 Performance [Gflop/s] Dual Intel Xeon 5160, 3000 MHz, icc 10.1



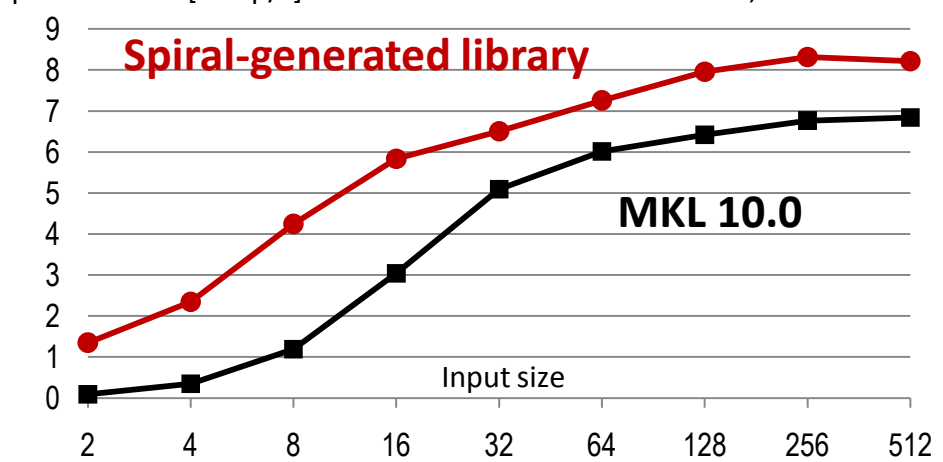
GEMM (acc. packed sq. NN), single-threaded, single precision  
 Performance [Gflop/s] Dual Intel Xeon 5160, 3000 MHz, icc 10.1



Rank-k Update, single precision, k=4  
 performance [Gflop/s] Dual Intel Xeon 5160, 3Ghz



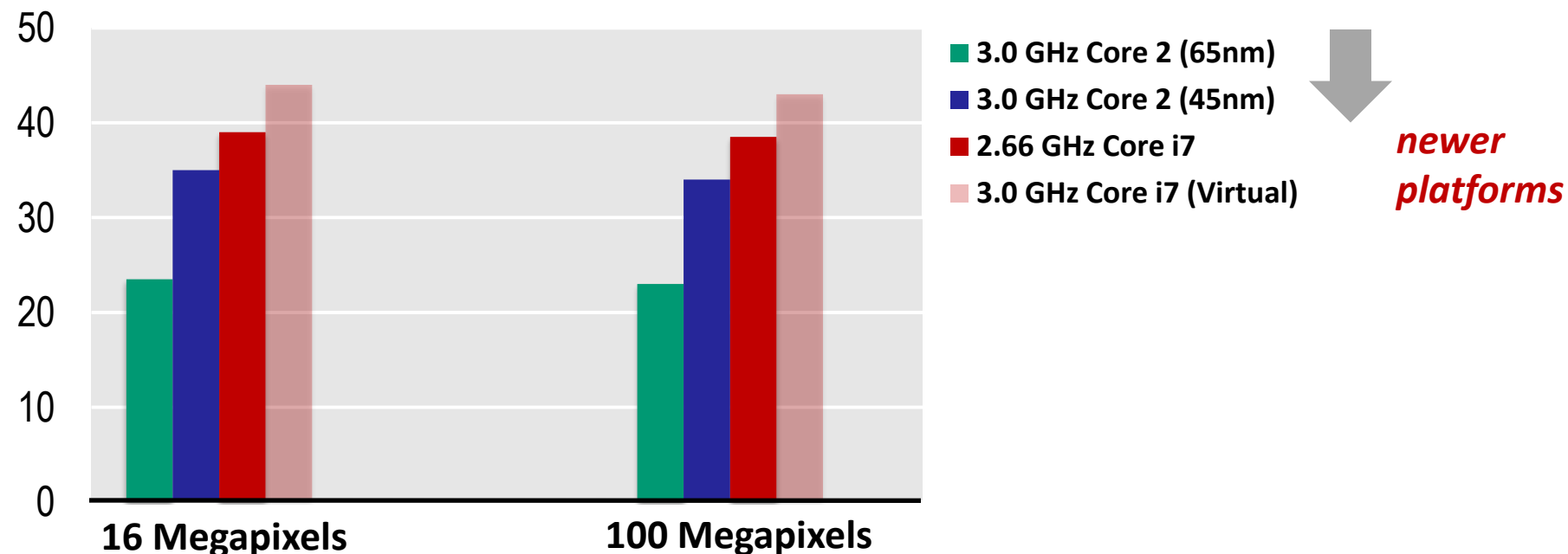
Rank-k Update, double precision, k=4  
 performance [Gflop/s] Dual Intel Xeon 5160, 3Ghz



# Polar Format SAR on Intel Core2 Quad

## SAR Image Formation on Intel platforms

performance [Gflop/s]



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

# Organization

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

# Current Directions

## Applications

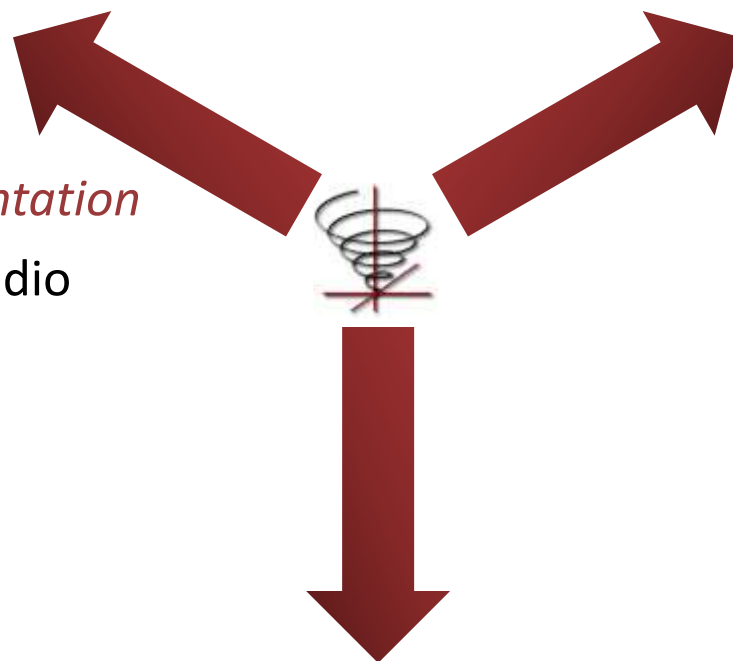
- Radar processing  
*SAR,...*
- Image processing  
*Correlation, segmentation*
- Software defined radio  
*Filters, encoders*
- Coding  
*Viterbi, JPEG2000*
- Linear algebra  
*Kalman filter, BLAS*

## Platforms

- Multicore CPUs  
*Core i7, POWER7*
- Next generation GPUs  
*Larrabee, Fermi*
- Accelerators  
*Virtex 5, SGI RASC*
- Homogeneous CMPs  
*TILEPro, Intel SCC*
- DSP multicores  
*TI DaVinci*

## Platform Design

- Application/architecture co-design
- Balanced architecture



# Summary

## ■ Spiral:

- Successful approach to automate the development of performance libraries
- Commercially used by Intel



- Commercialization: SpiralGen, Inc.

## ■ Key ideas:

- Domain specific symbolic algorithm representation
- Difficult optimizations through rewriting

DFT<sub>64</sub>



```
void dft64(float *Y, float *X) {
  __m512 U912, U913, U914, U915, ...
  __m512 *a2153, *a2155;
  a2153 = ((__m512 *) X);  s1107 = *(a2153);
  s1108 = *((a2153 + 4));  t1323 =
    __mm512_add_ps(s1107, s1108);
  t1324 = __mm512_sub_ps(s1107, s1108);
  <many more lines>
  U926 = __mm512_swizupconv_r32(...);
  s1121 = __mm512_madd231_ps(__mm512_mul_ps(
    __mm512_mask_or_pi(__mm512_set_lto16_ps(
      0.70710678118654757), 0xAAAA, a2154, U926), t1341),
    __mm512_mask_sub_ps(__mm512_set_lto16_ps(
      0.70710678118654757), ...),
    __mm512_swizupconv_r32(t1341, __MM_SWIZ_REG_CDAB));
  U927 = __mm512_swizupconv_r32
  <many more lines>
}
```

$$\text{DFT}_4 \rightarrow (\text{DFT}_2 \otimes I_2) T_2^4 (I_2 \otimes \text{DFT}_2) L_2^4$$

$$\underbrace{I_m \otimes A_n}_{\text{smp}(p, \mu)} \rightarrow I_p \otimes_{||} (I_{m/p} \otimes A_n)$$

**More Information:**

**[www.spiral.net](http://www.spiral.net)**

**[www.spiralgen.com](http://www.spiralgen.com)**