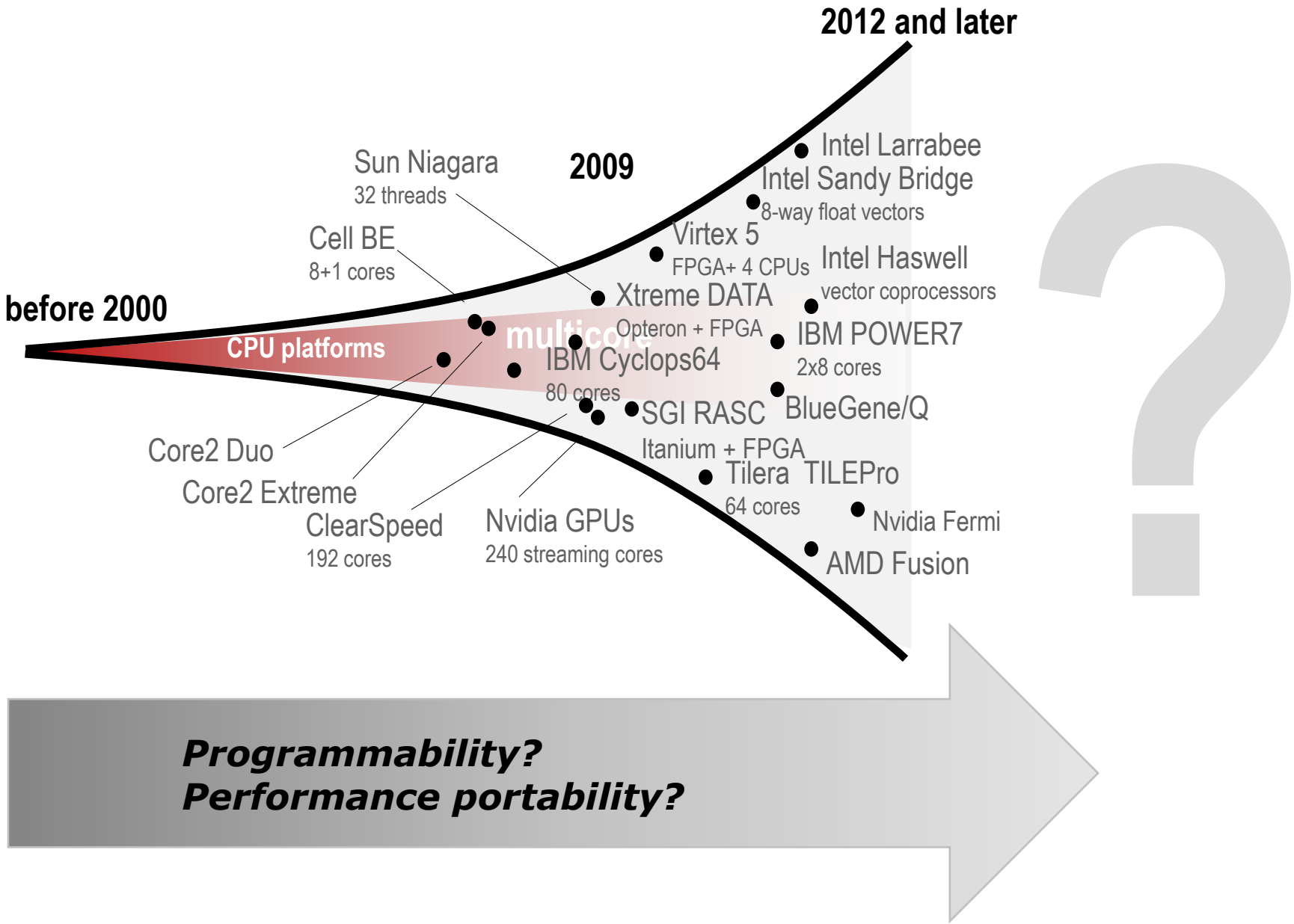
Spiral: Generating Efficient Programs for Emerging Parallel Platforms

Franz Franchetti

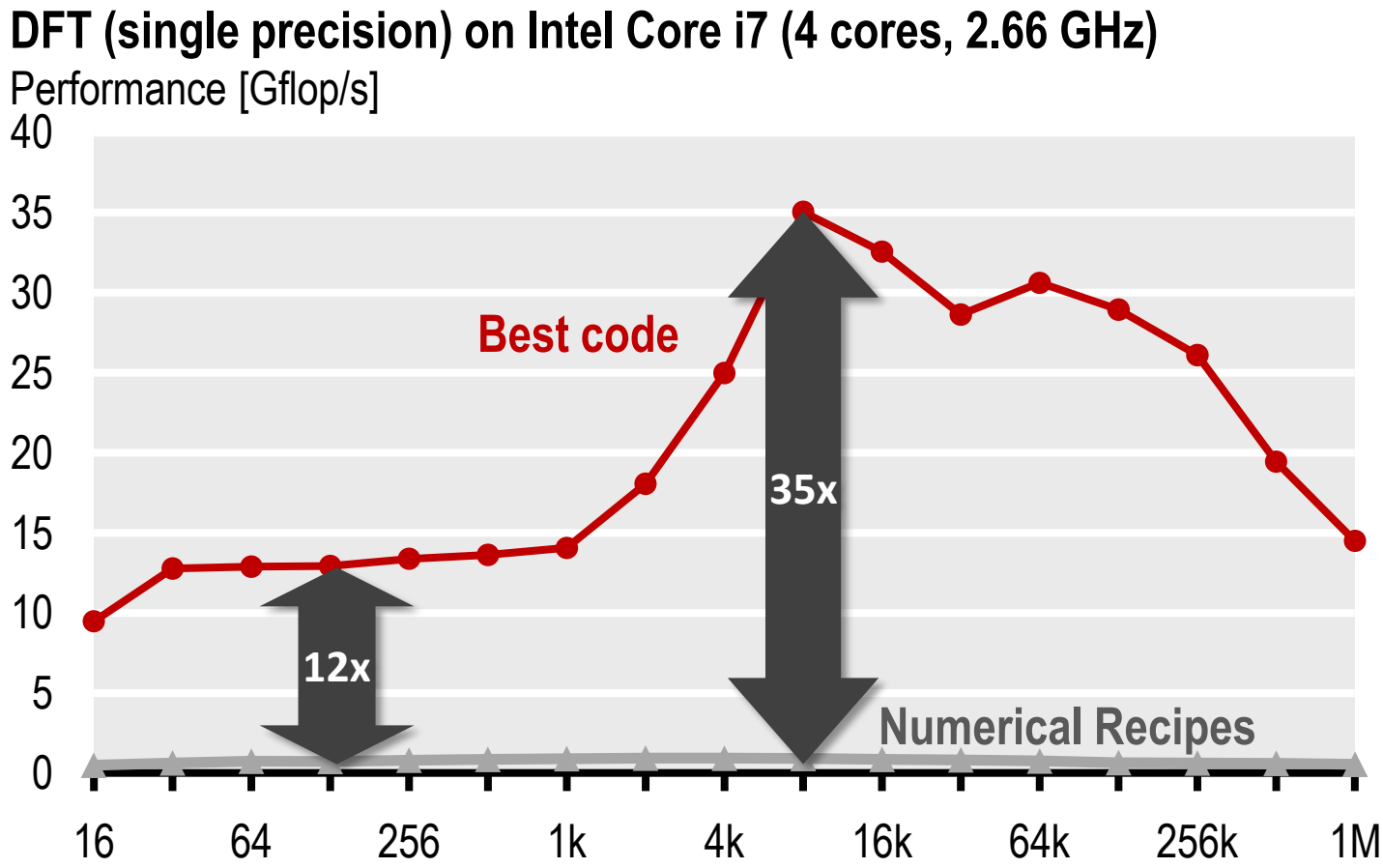
Electrical and Computer Engineering
Carnegie Mellon University

This work was supported by
DARPA DESA program, NSF, ONR, SRC (C2S2), Mercury Inc., Intel, Nvidia

The Future is Parallel and Heterogeneous

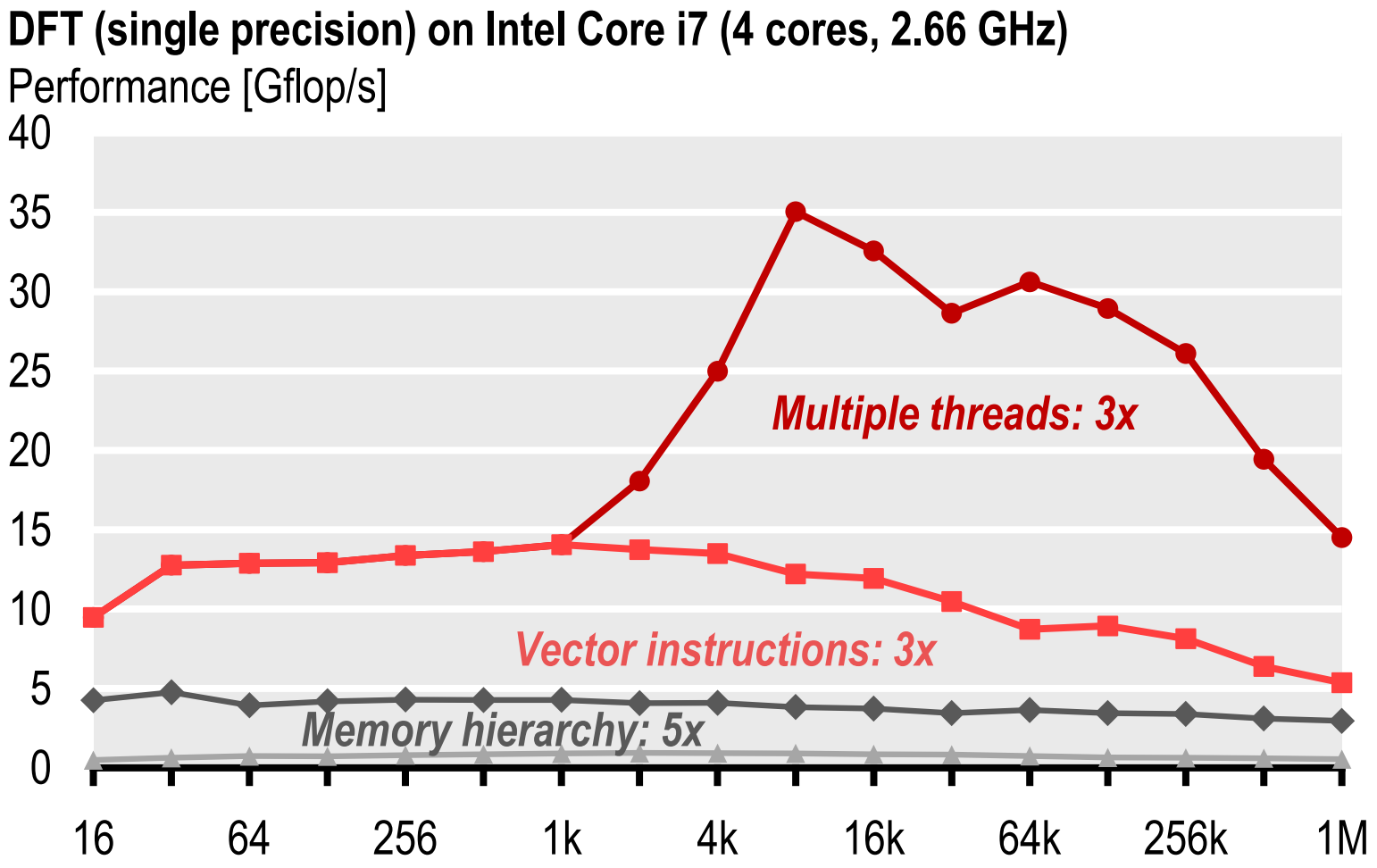


The Problem: Example DFT



- Standard desktop computer, cutting edge compiler, using optimization flags
- Implementations have same operations count: $\approx 4n\log_2(n)$
- ***Numerical Recipes is Code is actually good code (array-based ANSI C)***

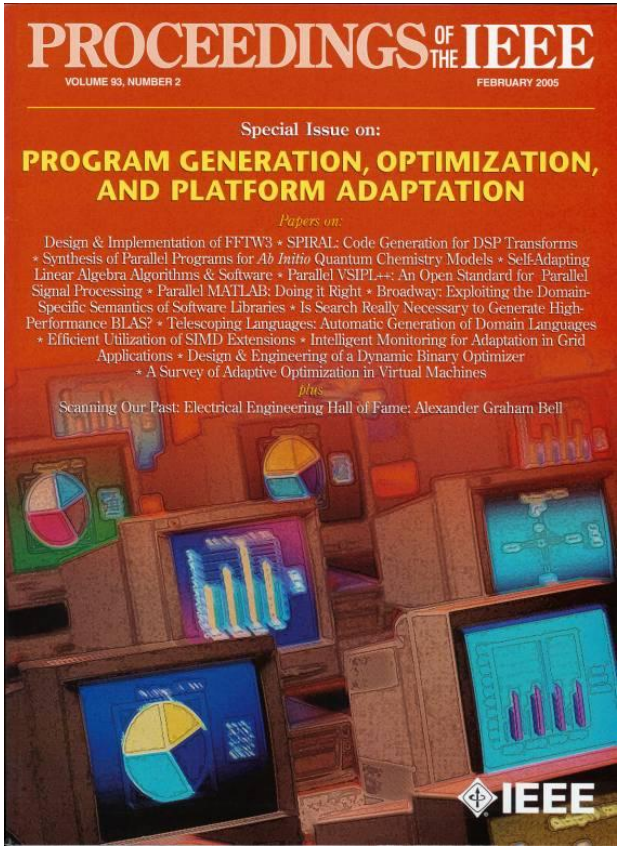
DFT Plot: Analysis



High performance library development has become a nightmare

Related Work

- **Automatic Performance Tuning**
  - Linear algebra: ATLAS, Bebop, Spike, Flame
  - Fourier transform: FFTW
  - Linear transforms: Spiral
- **Restructuring compilers**
  - Memory hierarchy: tiling,…
  - Vectorization, Parallelization
- **Algorithm design**
- **Computer algebra (rewriting)**
- **Computer language design**
- **Computer architecture**
- **High performance computing (HPC)**



Proceedings of the IEEE special issue, Feb. 2005

Organization

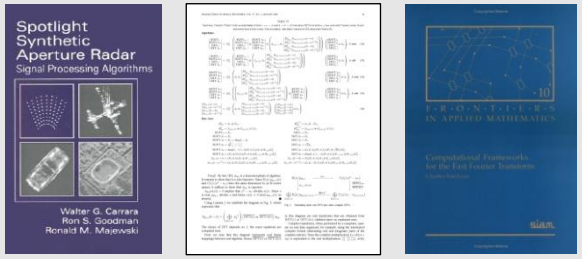
- **Spiral overview**
- Spiral's formal framework
- Parallelization in Spiral
- Algorithm/architecture co-design
- Results
- Concluding remarks

M. Püschel, F. Franchetti, Y. Voronenko: **Spiral.** Encyclopedia of Parallel Computing, D. A. Padua (Editor). *To appear.*

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.

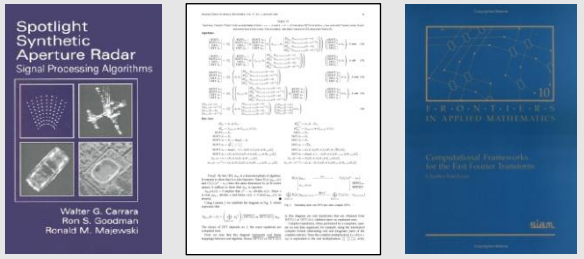
What is Spiral?

Traditionally



High performance library optimized for given platform

Spiral Approach



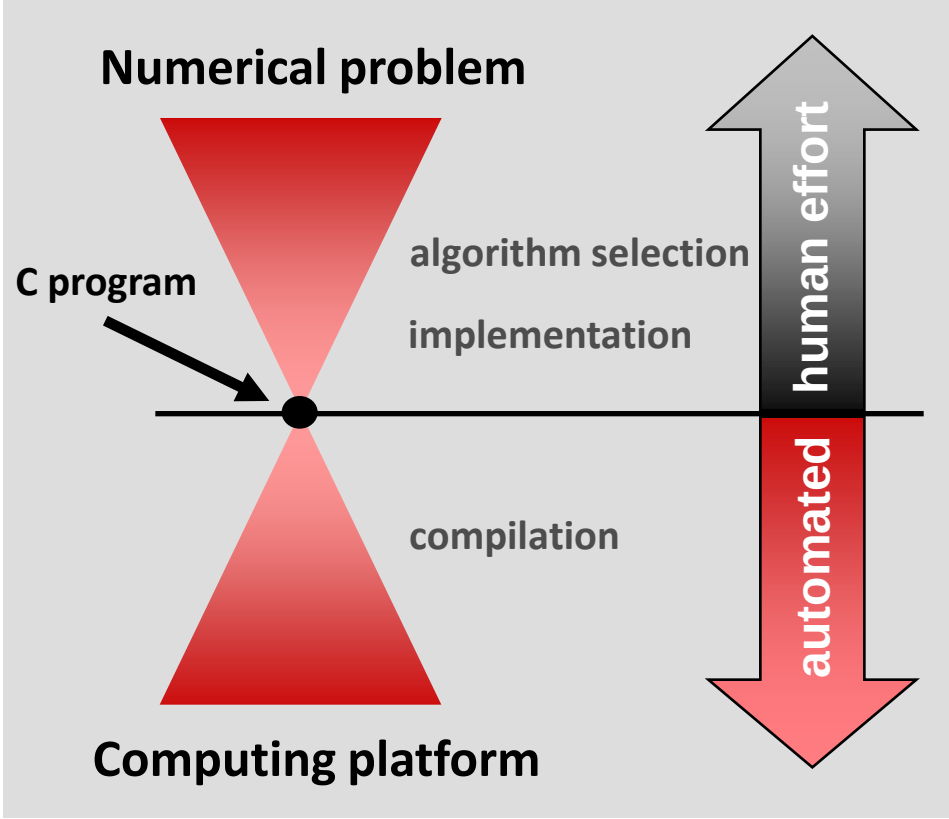
Spiral

High performance library optimized for given platform

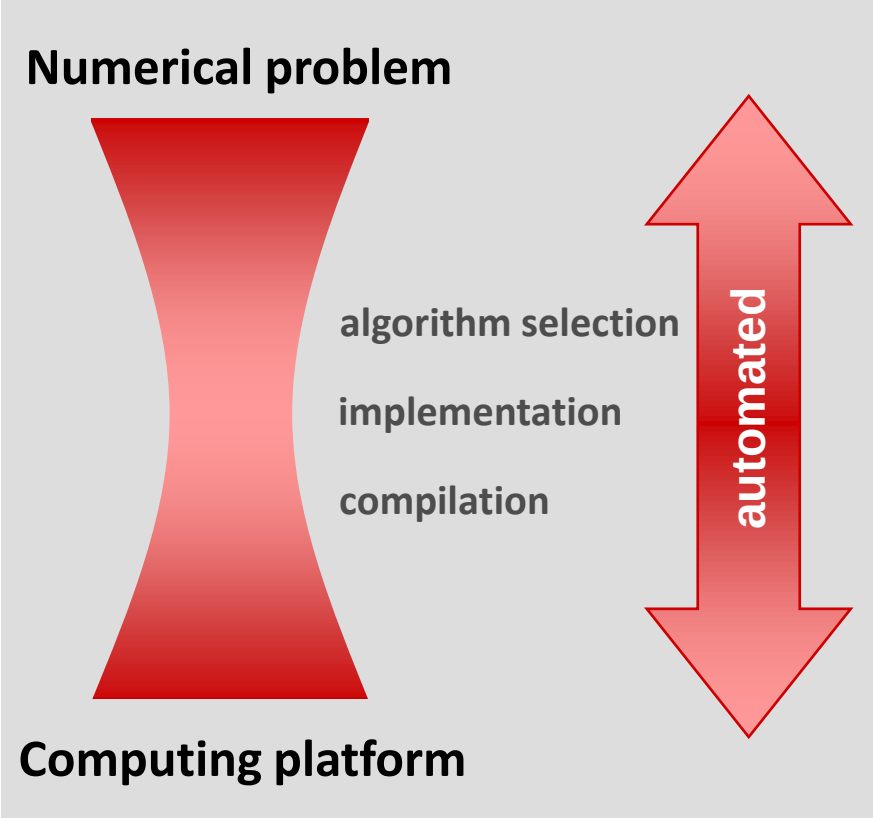
Comparable performance

Vision Behind Spiral

Current



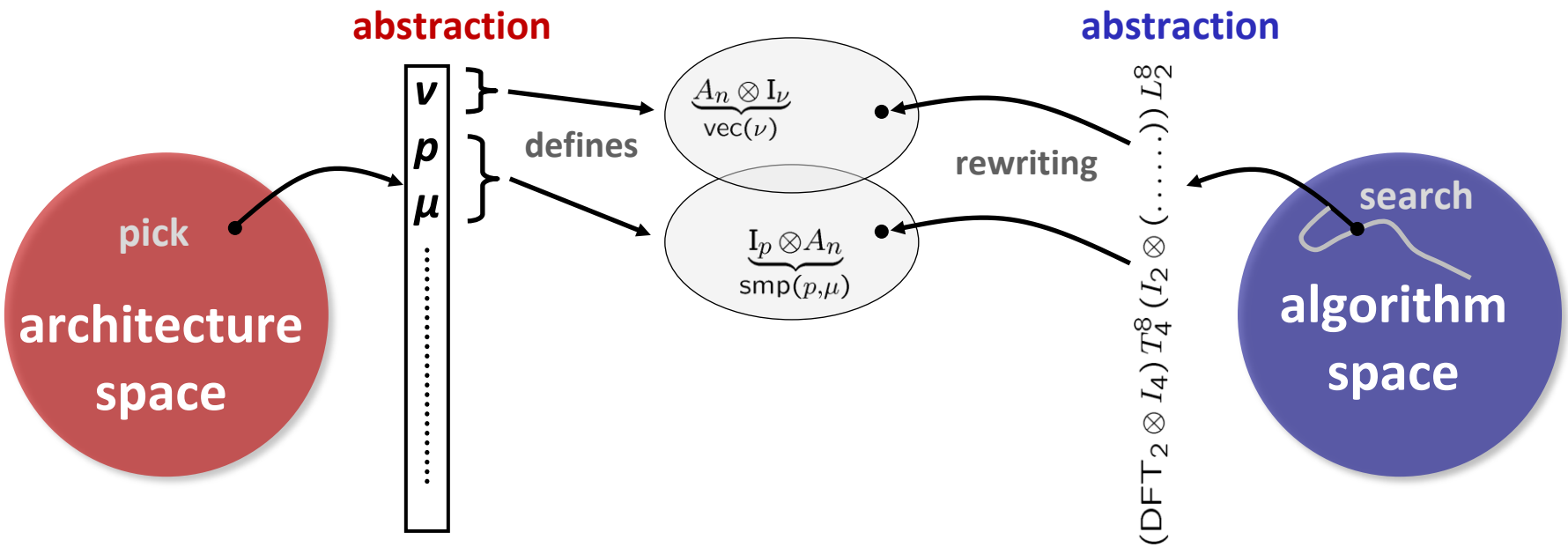
Future



- 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

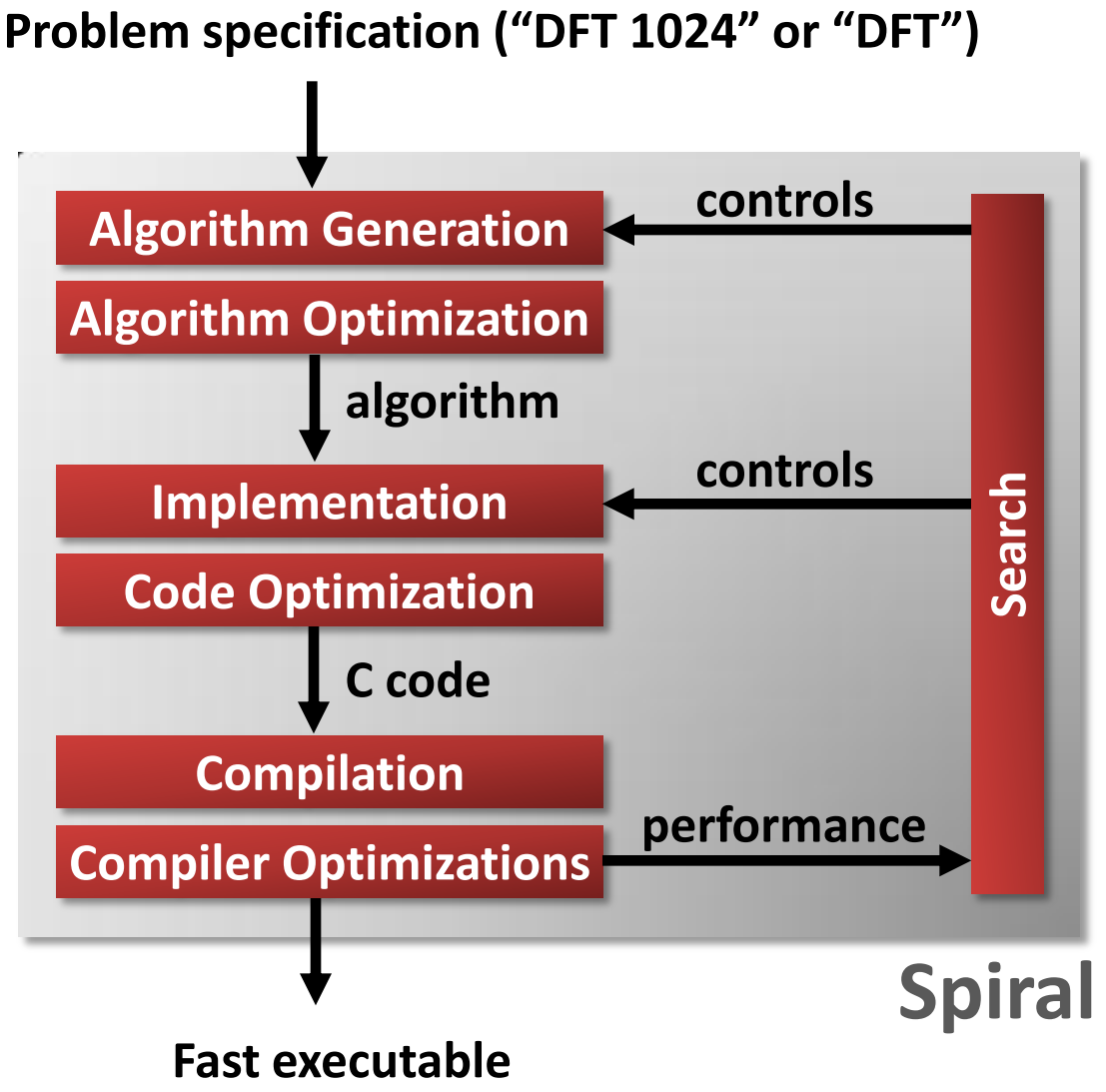
Kernel:
problem size,
algorithm choice

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



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 / 2 — code rate (?)

K 7 — constraint length (?)

polynomials 79, 91 — polynomials for the code in decimal notation (?)

Select implementation options

frame length 2048 — unpadded frame length in bits (?)

Vectorization level SSE 16-way — type of code (?)

[Generate Code] [Reset]

"Click": Push-button code generation

`http://www.spiral.net/software/viterbi.html`

Organization

- Spiral overview
- **Spiral's formal framework**
- Parallelization in Spiral
- Algorithm/architecture co-design
- 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)

output vector (signal)

transform = matrix

- **Fast algorithm = sparse matrix factorization = SPL formula**

  Example: Cooley-Tukey FFT algorithm

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

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

Breakdown Rules (>200 for >50 Transforms)

$$\mathbf{DFT}_n \to P^{\top}_{k/2,2m}\left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m}\,\mathbf{rDFT}_{2m}(i/k)\right)\right)\left(\mathbf{RDFT}'_k \otimes I_m\right), \quad k \text{ even},$$

$$\begin{vmatrix}\mathbf{RDFT}_n\\ \mathbf{RDFT}'_n\\ \mathbf{DHT}_n\\ \mathbf{DHT}'_n\end{vmatrix} \to (P^{\top}_{k/2,m} \otimes I_2)\left(\begin{vmatrix}\mathbf{RDFT}_{2m}\\ \mathbf{RDFT}'_{2m}\\ \mathbf{DHT}_{2m}\\ \mathbf{DHT}'_{2m}\end{vmatrix} \oplus \left(I_{k/2-1} \otimes_i D_{2m}\begin{vmatrix}\mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDFT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\\ \mathbf{rDHT}_{2m}(i/k)\end{vmatrix}\right)\right)\left(\begin{vmatrix}\mathbf{RDFT}'_k\\ \mathbf{RDFT}'_k\\ \mathbf{DHT}'_k\\ \mathbf{DHT}'_k\end{vmatrix} \otimes I_m\right), \quad k \text{ even},$$

$$\begin{vmatrix}\mathbf{rDFT}_{2n}(u)\\ \mathbf{rDHT}_{2n}(u)\end{vmatrix} \to L^{2n}_m\left(I_k \otimes_i \begin{vmatrix}\mathbf{rDFT}_{2m}((i+u)/k)\\ \mathbf{rDHT}_{2m}((i+u)/k)\end{vmatrix}\right)\left(\begin{vmatrix}\mathbf{rDFT}_{2k}(u)\\ \mathbf{rDHT}_{2k}(u)\end{vmatrix} \otimes I_m\right),$$

$$\mathbf{RDFT\text{-}3}_n \to (Q^{\top}_{k/2,m} \otimes I_2)\,(I_k \otimes_i \mathbf{rDFT}_{2m})(i+1/2)/k))\,(\mathbf{RDFT\text{-}3}_k \otimes I_m), \quad k \text{ even},$$

$$\mathbf{DCT\text{-}2}_n \to P^{\top}_{k/2,2m}\left(\mathbf{DCT\text{-}2}_{2m}\,K^{2m}_2 \oplus \left(I_{k/2-1} \otimes N_{2m}\,\mathbf{RDFT\text{-}3}^{\top}_{2m}\right)\right) B_n(L^{n/2}_{k/2} \otimes I_2)(I_m \otimes \mathbf{RDFT}'_k)Q_{m/2,k},$$

$$\mathbf{DCT\text{-}3}_n \to \mathbf{DCT\text{-}2}^{\top}_n,$$

$$\mathbf{DCT\text{-}4}_n \to Q^{\top}_{k/2,2m}\left(I_{k/2} \otimes N_{2m}\,\mathbf{RDFT\text{-}3}^{\top}_{2m}\right) B'_n(L^{n/2}_{k/2} \otimes I_2)(I_m \otimes \mathbf{RDFT\text{-}3}_k)Q_{m/2,k}.$$

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

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

$$\mathbf{DFT}_p \to R^T_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})D_p(\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1})R_p, \quad p \text{ prime}$$

$$\mathbf{DCT\text{-}3}_n \to (\mathrm{I}_m \oplus \mathrm{J}_m)\,\mathrm{L}^n_m(\mathbf{DCT\text{-}3}_m(1/4) \oplus \mathbf{DCT\text{-}3}_m(3/4)) \cdot(\mathrm{F}_2 \otimes \mathrm{I}_m)\begin{bmatrix}\mathrm{I}_m & 0 \oplus -\mathrm{J}_{m-1}\\ & \frac{1}{\sqrt{2}}(\mathrm{I}_1 \oplus 2\,\mathrm{I}_m)\end{bmatrix}, \quad n = 2m$$

$$\mathbf{DCT\text{-}4}_n \to S_n\mathbf{DCT\text{-}2}_n\,\mathrm{diag}_{0\le k<n}(1/(2\cos((2k+1)\pi/4n)))$$

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

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

$$\mathbf{DFT}_2 \to \mathrm{F}_2$$

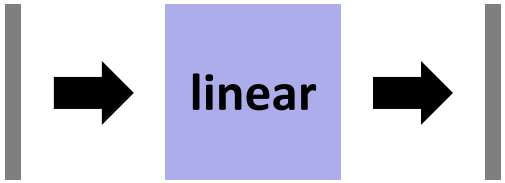
$$\mathbf{DCT\text{-}2}_2 \to \mathrm{diag}(1, 1/\sqrt{2})\,\mathrm{F}_2$$

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

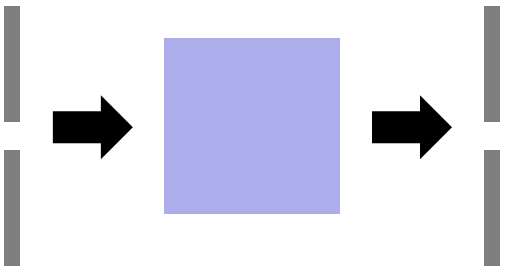
- *"Teaches" Spiral algorithm knowledge*
- *Combining these rules yields many algorithms for every given transform*

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
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	$\mathbf{I} \otimes \mathsf{M},\ \mathsf{M} \otimes \mathbf{I}$

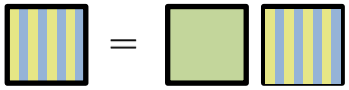
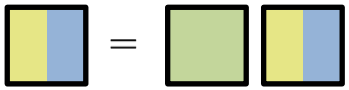
$$(\mathbf{I}_{m\times n\to mn} \otimes \mathsf{A}) \left(\sum_{i=0}^{m-1} \mathbf{e}_i^m \otimes \mathbf{x}, \sum_{i=0}^{n-1} \mathbf{e}_i^n \otimes \mathbf{y}\right) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \mathbf{e}_i^m \otimes \mathbf{e}_j^n \otimes \mathsf{A}(\mathbf{x}, \mathbf{y}),$$

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

Example: Matrix Multiplication (MMM)

Breakdown rules:

capture various forms of blocking

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

Expressing Kernels as OL Formulas

Linear Transforms

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

Viterbi Decoding

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

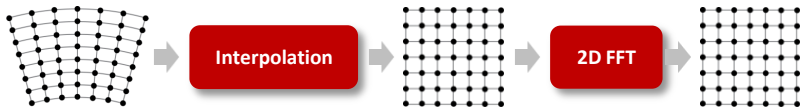
$$\underline{\mathbf{F}_{K,F}}\ \nu \rightarrow \prod_{i=1}^{F} \left(\left(\mathrm{I}_{2^{K-2}/\nu} \otimes_{j_1} \vec{\mathrm{L}}^{2\nu}_{\nu} \vec{B}^{\nu}_{F-i,j_1} \right) (\mathrm{L}^{2^{K-1}/\nu}_{2^{K-2}/\nu} \bar{\otimes} \mathrm{I}_\nu) \right)$$

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

Matrix-Matrix Multiplication

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

Synthetic Aperture Radar (SAR)



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

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])
```

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

$y = (A_m \otimes I_n)x$

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

General Operators

$\mathbf{r} = (\mathsf{A}_{K\times M\to N} \circ \mathsf{B}_{k\times m\to K\times M})(\mathbf{x},\mathbf{y})$

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

$(\mathbf{r},\mathbf{s}) = (\mathsf{A}_{m\to M} \times \mathsf{B}_{n\to N})(\mathbf{x},\mathbf{y})$

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

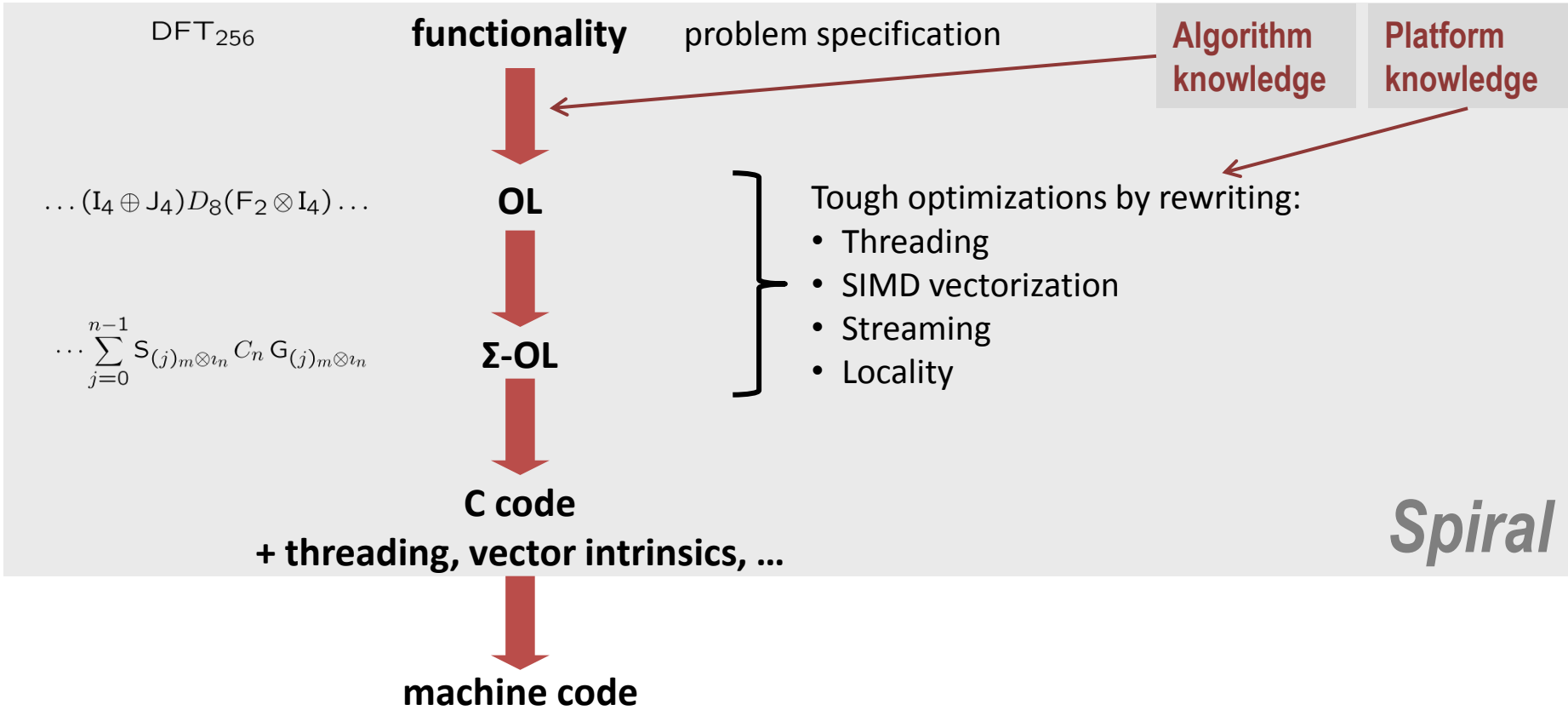
$\mathbf{r} = (\mathrm{I}_{m\times n\to mn} \otimes A_{M\times N\to K})(\mathbf{x},\mathbf{y})$

```
for (i=0;i<m;i++)
   for (j=0;j<m;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]);
```

$\mathbf{r} = (A_{M\times N\to K} \otimes \mathrm{I}_{m\times n\to mn})(\mathbf{x},\mathbf{y})$

```
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

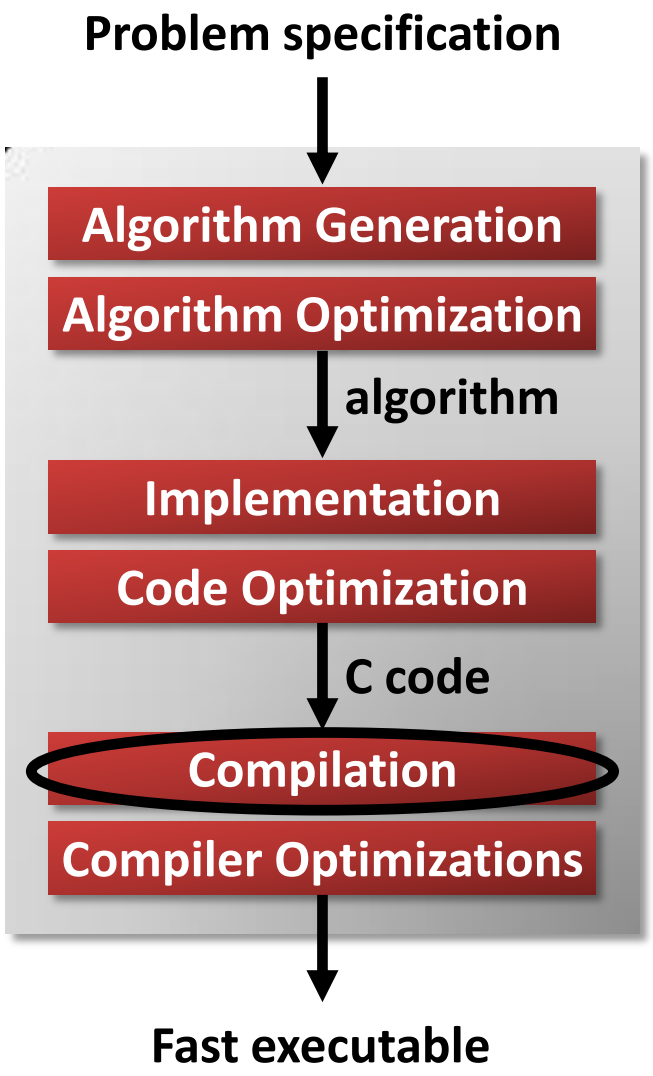
- Spiral overview
- Spiral's formal framework
- **Parallelization in Spiral**
- Generating general-size libraries
- Results
- Concluding remarks

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

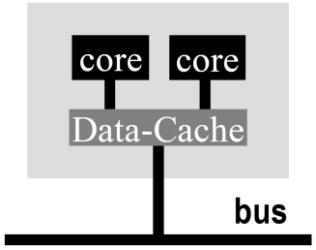
- **Multithreading** (Multicore)
- **Vector SIMD and SIMT** (SSE, GPUs)
- **Message Passing** (Clusters, MPP, SCC)
- **Software-managed cache** (Cell, GPUs)
- **Streaming** (Cell, GPUs)
- **Gate-level parallelism** (FPGA)
- **Heterogeneous** (CPU+FPGA or GPU)



Spiral: One methodology optimizes for all types of parallelism

Modeling Multicore: Base Cases

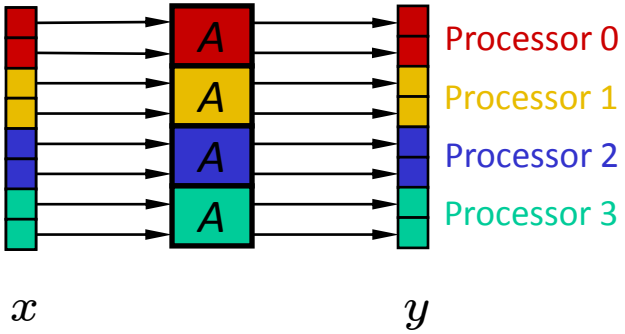
- **Hardware abstraction: shared cache with cache lines**



$$\underbrace{A_{k\times m\to n}}_{\mathrm{smp}(p,\mu)}$$

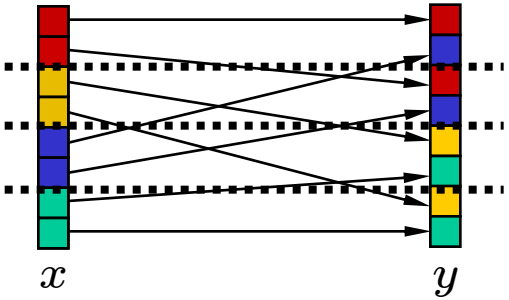
- **Tensor product: embarrassingly parallel operator**

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



- **Permutation: problematic; may produce false sharing**

$$y = \mathrm{L}^{8}_{4}(x)$$



Optimization Knowledge: Rewriting Rules

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

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

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

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

Arity (1,1) rules

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

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

Arity (2,1) rules

Rewriting: Map Algorithm to Platform

- **Algorithm rules: breakdown rules**

$$\mathrm{MMM}_{m,n,k} \rightarrow (\otimes)_{m/m_b \times 1} \otimes \mathrm{MMM}_{m_b,n,k}$$

$$\mathrm{MMM}_{m,n,k} \rightarrow \mathrm{MMM}_{m,nb,k} \otimes (\otimes)_{1 \times n/nb}$$

$$\mathrm{MMM}_{m,n,k} \rightarrow ((\Sigma_{k/k_b} \circ (\cdot)_{k/k_b}) \otimes \mathrm{MMM}_{m,n,k_b}) \circ ((L^{mk/k_b}_{k/k_b} \otimes I_{k_b}) \times I_{kn})$$

- **Hardware constraints: base cases**

$$\mathrm{I}_p \otimes_{||} A_{m\mu \rightarrow n\mu}, \quad \mathrm{P}_p \otimes_{||} A_{k\mu \times m\mu \rightarrow n\mu}, \quad \mathrm{K}_{q \times r} \otimes_{||} A_{k\mu \times m\mu \rightarrow n\mu},$$
$$M \bar{\otimes} \mu, \quad (M \bar{\otimes} \mathrm{I}_\mu) \times (N \bar{\otimes} \mathrm{I}_\mu)$$

- **Program transformations: manipulation rules**

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

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

Combined rule set spans search space for empirical optimization

MMM: Parallelization Through Rewriting

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

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

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

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

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

Fully optimized (load-balanced, no false sharing)

Same Approach for Other Parallel Paradigms

Message Passing:

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

$$\cdots$$

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

$$\cdots$$

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

Vectorization:

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

$$\cdots$$

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

$$\cdots$$

$$\rightarrow (\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathsf{L}^{2\nu}_\nu}_{\mathsf{sse}})(\overline{\mathbf{DFT}_m \otimes \mathrm{I}_{n/\nu}}\vec{\otimes}\mathrm{I}_\nu)\underbrace{(\mathsf{T}^{mn}_n)^{\nu}}_{\mathsf{sse}}$$
$$\left(\mathrm{I}_{m/\nu} \otimes (\overline{\mathsf{L}^{n}_{\nu}}\vec{\otimes}\mathrm{I}_\nu)(\mathrm{I}_{n/\nu} \otimes (\mathsf{L}^{2\nu}_\nu \vec{\otimes}\mathrm{I}_\nu)(\mathrm{I}_2 \otimes \underbrace{\mathsf{L}^{\nu^2}_\nu}_{\mathsf{sse}})(\mathsf{L}^{2\nu}_2 \vec{\otimes}\mathrm{I}_\nu))(\overline{\mathbf{DFT}_n}\vec{\otimes}\mathrm{I}_\nu)\right)$$
$$\left((\mathsf{L}^{mn}_m \otimes \mathrm{I}_2)\vec{\otimes}\mathrm{I}_\nu\right)(\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathsf{L}^{2\nu}_2}_{\mathsf{sse}})$$

GPUs:

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

$$\cdots$$

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

Verilog for FPGAs:

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

$$\cdots$$

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

$$\cdots$$

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

- **Rigorous, correct by construction**
- **Overcomes compiler limitations**

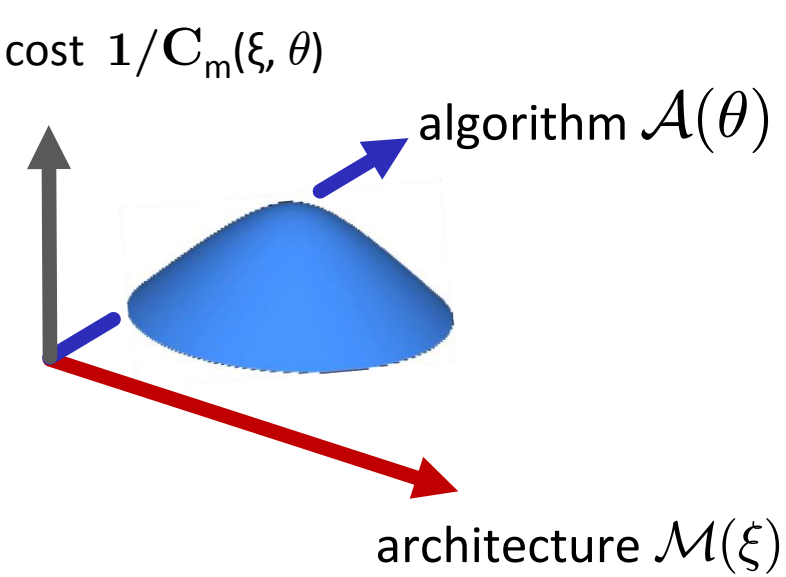
Organization

- Spiral overview
- Spiral's formal framework
- Parallelization in Spiral
- **Algorithm/architecture co-design**
- Results
- Concluding remarks

Q. Yu, D. F. Jones, N. Lawrence, D. McFarlin, P. A. Milder, B. Moore, A. Sidelnik, F. Franchetti, M. J. Garzaran, J. C. Hoe, J. Johnson, J. M. F. Moura, D. A. Padua, and M. Püschel: **The Data Pump Architecture for Algorithm-Specific DSP Processor/Program Co-Design.** Submitted for publication.

The "Inverse" Question

Design Space



Optimization Problem

$$(\hat{\mathcal{A}}, \hat{\mathcal{M}}) = \mathrm{argmin}_{\theta,\xi} \mathrm{C_m}(\mathcal{A}(\theta), \mathcal{M}(\xi))$$

- Algorithm $\mathcal{A}(\theta)$
- Architecture $\mathcal{M}(\xi)$
- Cost function $\mathrm{C_m}(\xi, \theta)$
- Parameters: ξ, θ
- Metric m: power, runtime,…

Task: Find ξ and θ s.t. $\mathrm{C_m}(\xi, \theta)$ is minimal

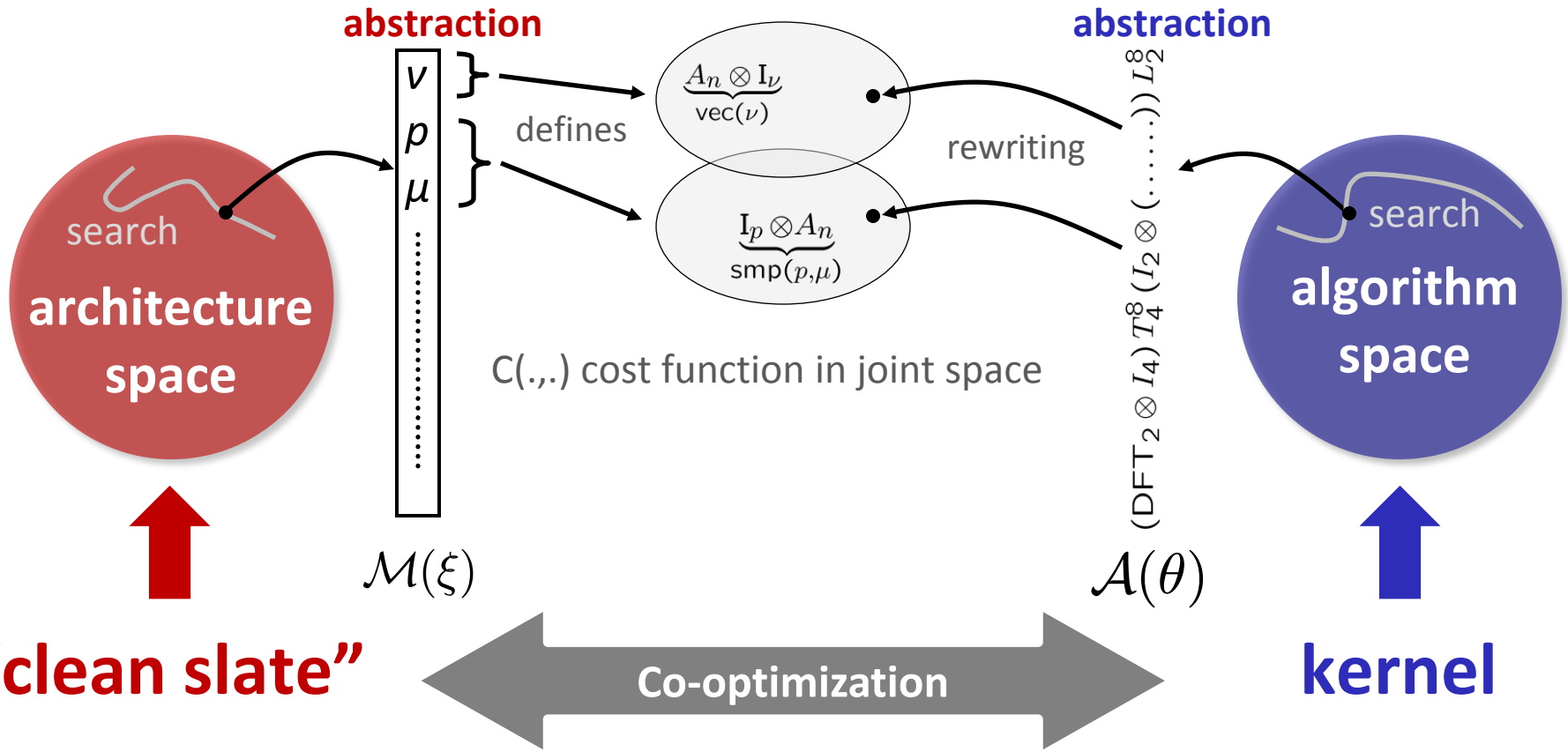
"What is the right architecture for my application?"
"What architecture features are good for my application?"

Co-Optimizing Architecture and Kernel

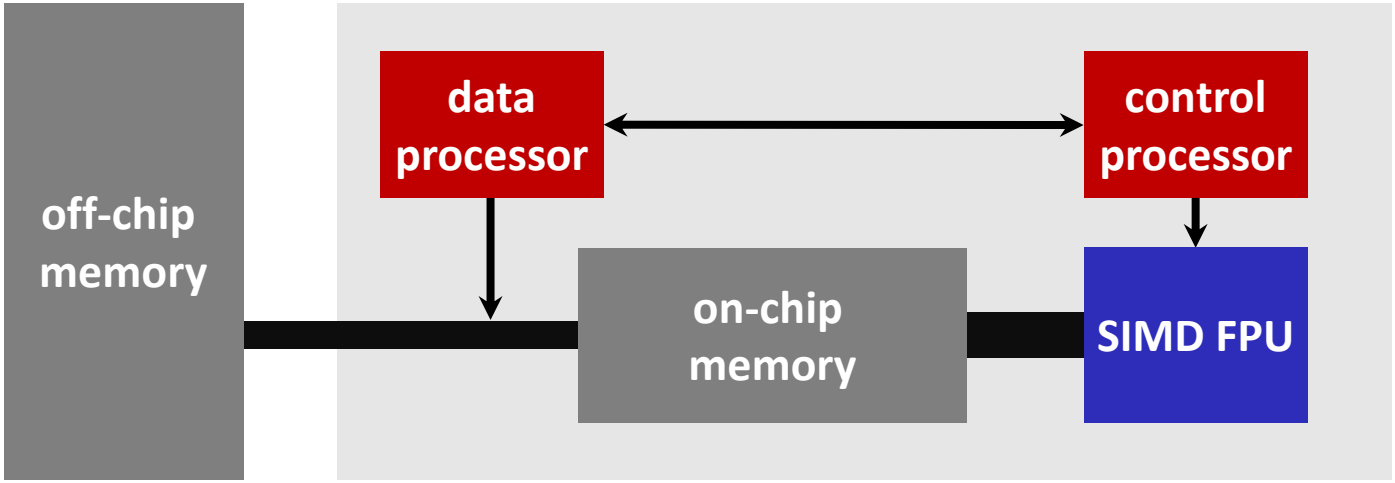
Architectural parameter:
Vector length,
#processors, …

Model: common abstraction
= spaces of matching formulas

Kernel:
problem size,
algorithm choice

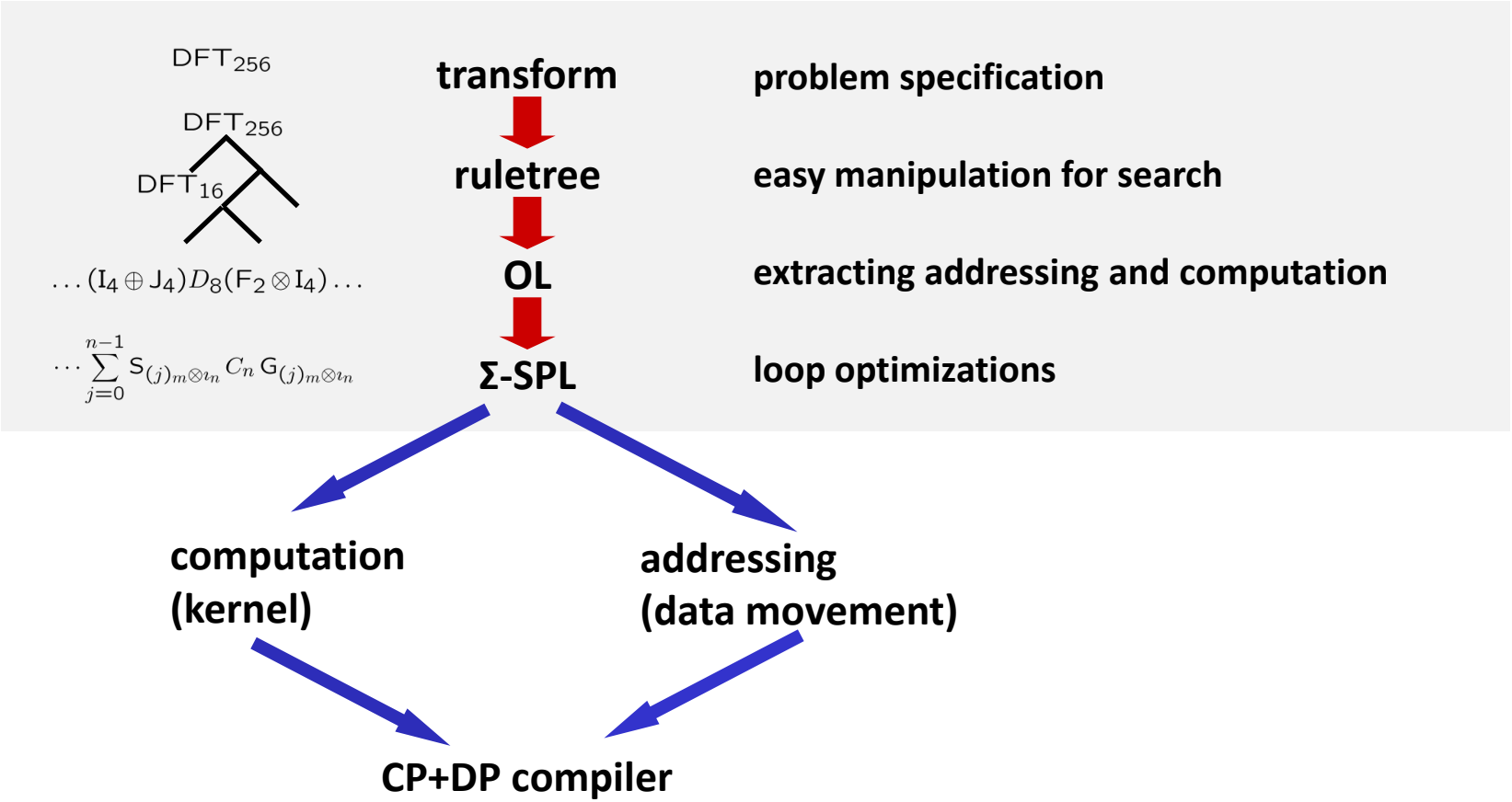


"Clean Slate:" Data Pump Architecture



- **Non-von Neumann paradigm**
  software-exposed on-chip memory hierarchy + concurrency
- **Flexible and parameterizable**
  parallel compute units + on-chip memory
- **Automatic co-synthesis (RTL + software)**
  processor instances and matched software executables

Program Generation for the Data Pump

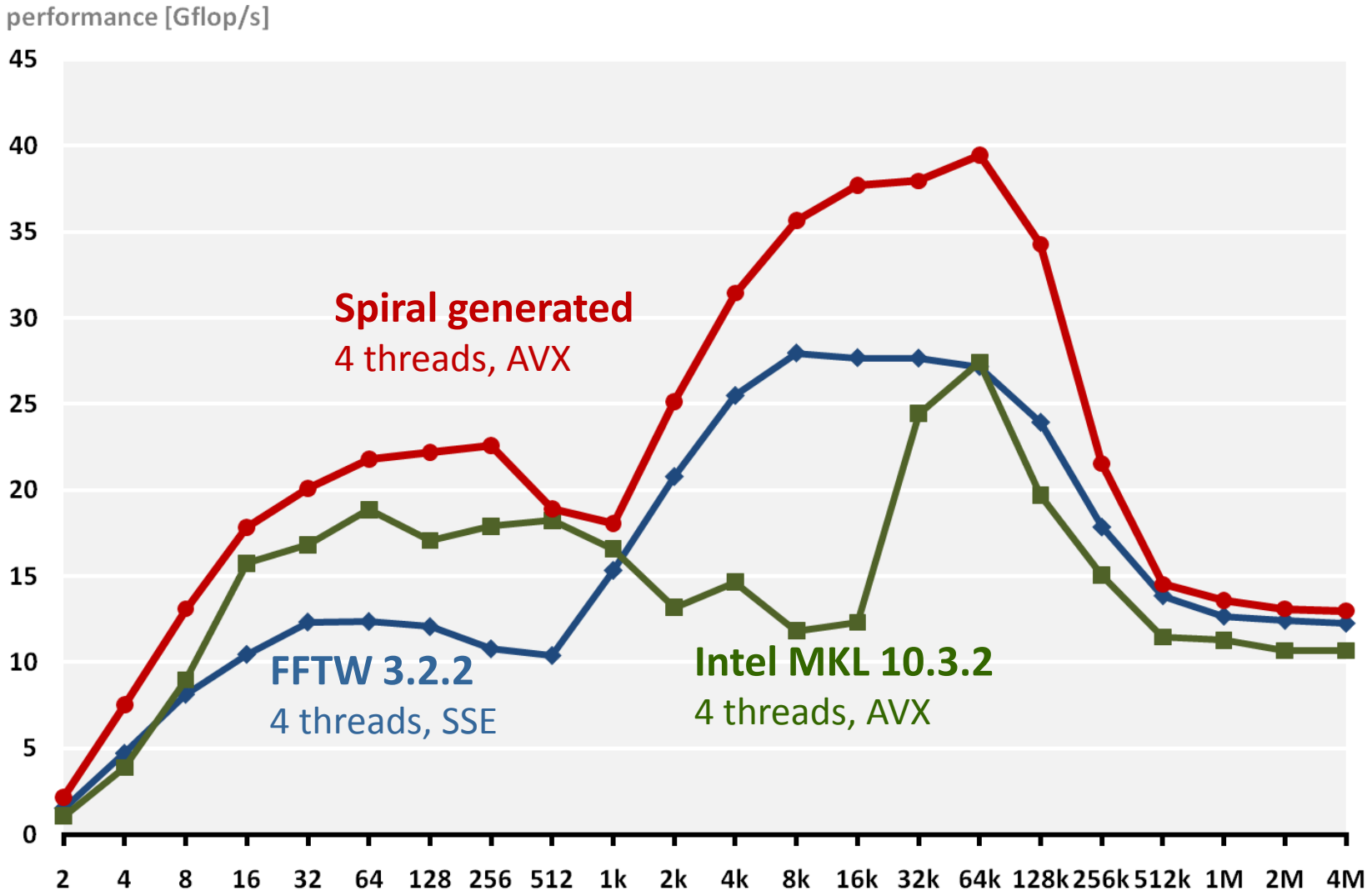


- Possible because of formal framework (OL, rewriting,…)
- Extract efficient mapping to DP and CP from formulas
- Language + compiler design to enable low-level optimization

Organization

- Spiral overview
- Parallelization in Spiral
- Beyond Transforms
- Algorithm/architecture co-design
- **Results**
- Concluding remarks

1D DFT on 3.3 GHz Sandy Bridge (4 Cores, AVX)

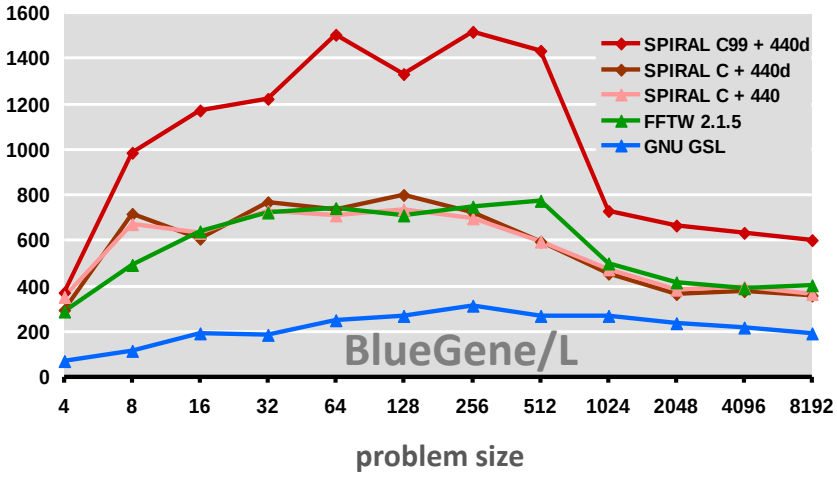


F. Franchetti, Y. Voronenko, S. Chellappa, J. M. F. Moura, and M. Püschel:
Discrete Fourier Transform on Multicores: Algorithms and Automatic Implementation
IEEE Signal Processing Magazine, special issue on "Signal Processing on Platforms with Multiple Cores", 2009.

BlueGene/L and P Supercomputers

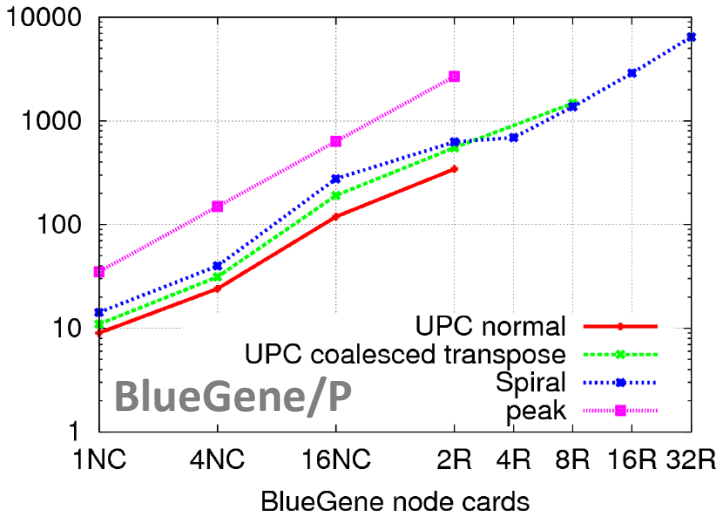
1 DFT, double precision

performance [Mflop/s]



Global FFT (1D FFT, HPC Challenge)

performance [Gflop/s]



6.4 Tflop/s

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

SIMD vectorization

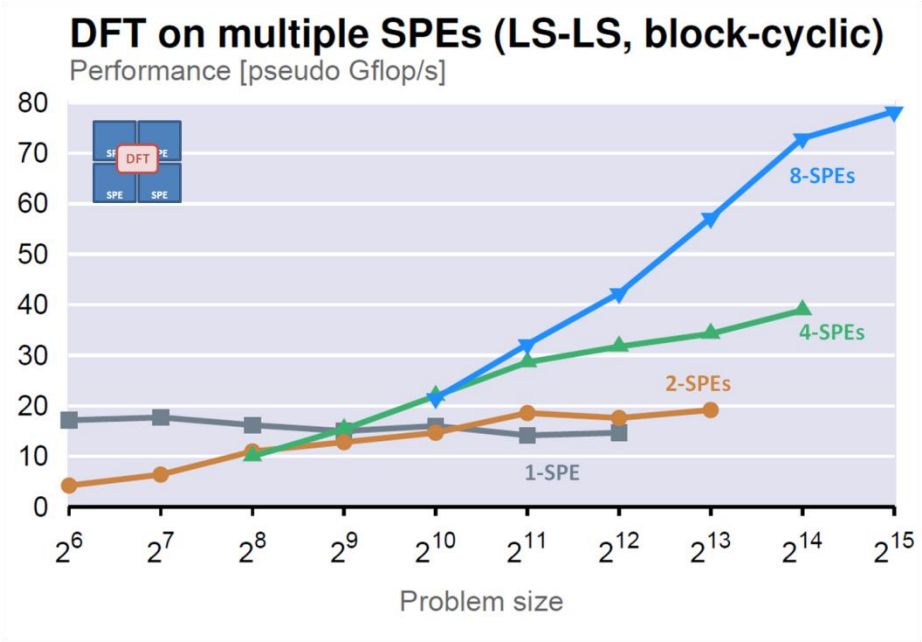
BlueGene/P (128k CPUs) at 850 MHz
Argonne National Laboratory

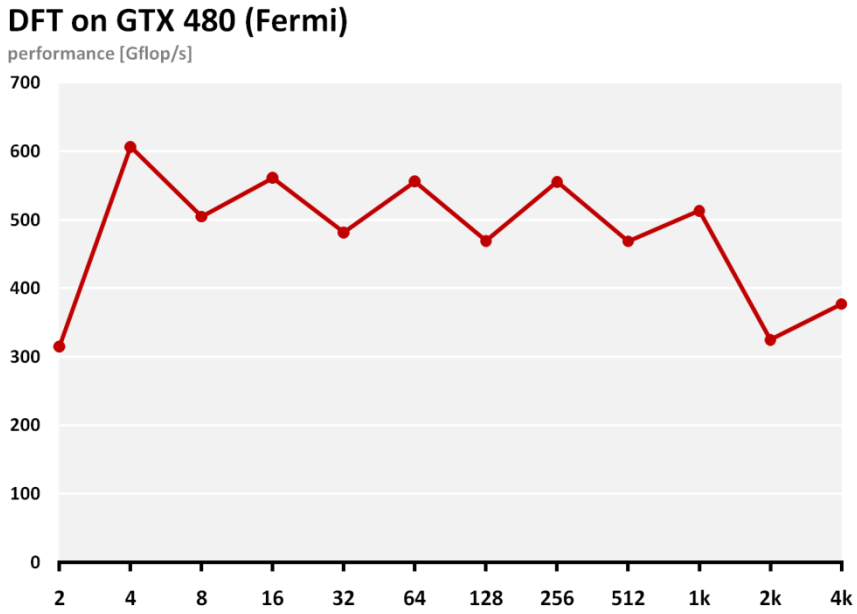
SIMD vectorization + multi-threading + MPI

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).**

G. Almási, B. Dalton, L. L. Hu, F. Franchetti, Y. Liu, A. Sidelnik, T. Spelce, I. G. Tānase, E. Tiotto, Y. Voronenko, X. Xue: **2010 IBM HPC Challenge Class II Submission. Winner of the 2010 HPC Challenge Class II Award (Most Productive System).**

Manycore Accelerators: Cell and Fermi





3.2 GHz Cell BE, latency performance
4-way SIMD unit, 8 SPEs, DMA

SIMD vectorization, threading

Nvidia GTX 480 Fermi, throughput performance
448 cores, 32-way warps, 15 multiprocessors

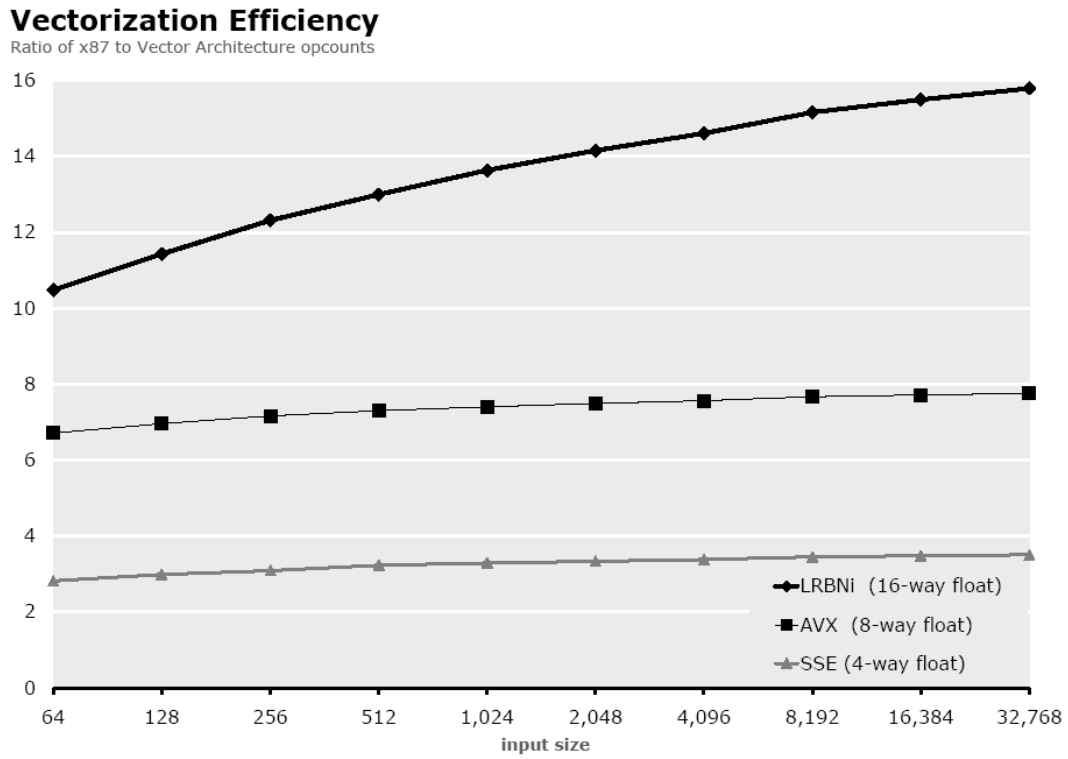
SIMT (vectorization+threading), memory layout

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.

C. Angelopoulos, F. Franchetti, and M. Püschel: **DFT Transform on the Fermi (GTX480): Automatic Program Generation**
NVIDIA Research Summit at the GPU Technology Conference, 2010.

Pre-Silicon Optimization: Larrabee/MIC

```
void dft64(float  *Y, float  *X) {
    __m512 U912, U913, U914, U915, U916, U917, U918, U919, U920, U921, U922, U923, U924, U925,...;
    a2153 = ((__m512  *) X);  s1107 = *(a2153);
    s1108 = *((a2153 + 4));   t1323 = _mm512_add_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_1tc
        _mm512_swizupconv_r32(t1341,_MM_S
    U927 = _mm512_swizupconv_r32(_mm512_s
        0.70710678118654757, (-0.7071067
        0.70710678118654757, (-0.7071067
        0.70710678118654757, (-0.7071067
        0.70710678118654757, (-0.7071067
    ...
    s1166 = _mm512_madd231_ps(_mm512_mul_
        0.70710678118654757, (-0.7071067
        0.70710678118654757, (-0.7071067
        0.70710678118654757, (-0.7071067
        0.70710678118654757, (-0.7071067
        0xAAAA,a2154,U951),t1362),
        _mm512_mask_sub_ps(_mm512_set_16t
        (-0.70710678118654757), 0.707106
        (-0.70710678118654757), 0.707106
        (-0.70710678118654757), 0.707106
        (-0.70710678118654757), 0.707106
        _mm512_swizupconv_r32(t1362,_MM_S
    ...
}
```

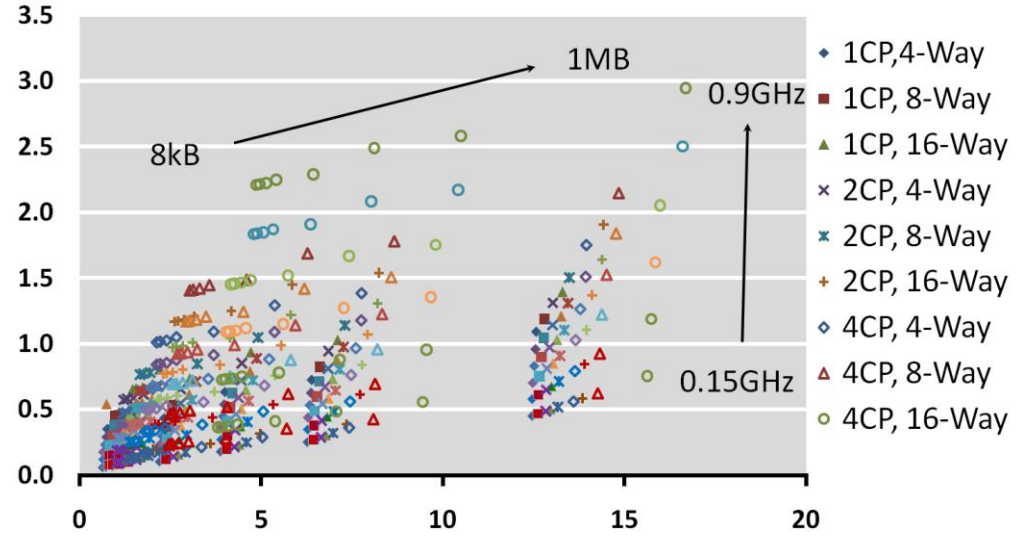


D. McFarlin, F. Franchetti, and M. Püschel: **Automatic Generation of Vectorized Fast Fourier Transform Libraries for the Larrabee and AVX Instruction Set Extension.** Proceedings of the 2009 High Performance Embedded Computing (HPEC), MIT Lincoln Laboratory. **Best paper award.**

Towards Architecture/Algorithm Co-Design

DPA Design Space

power [W] vs. area [mm²]

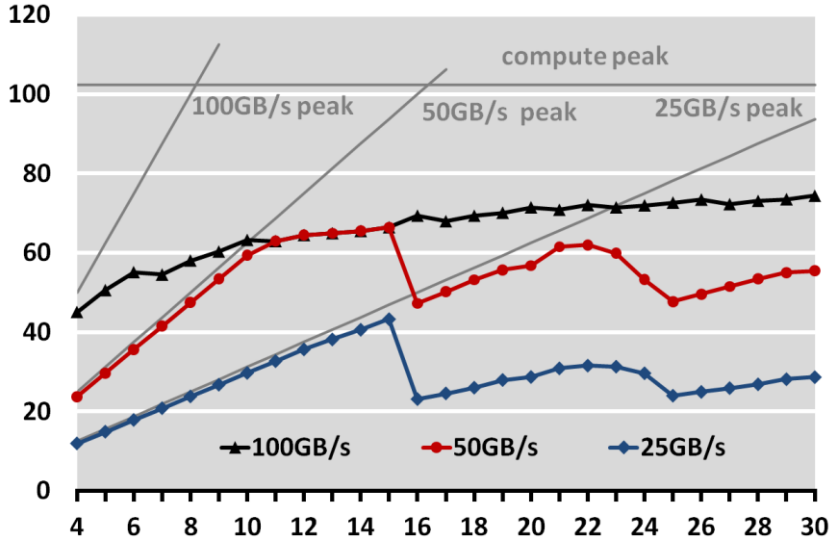


Area/power trade-off: 432 instances
local store: 8kB – 1 MB
CPU @ 0.15GHz – 0.9GHz

65nm process (simulation)

Effect of Off-Chip Bandwidth

performance [Gflop/s] vs.s log2(WHT size)

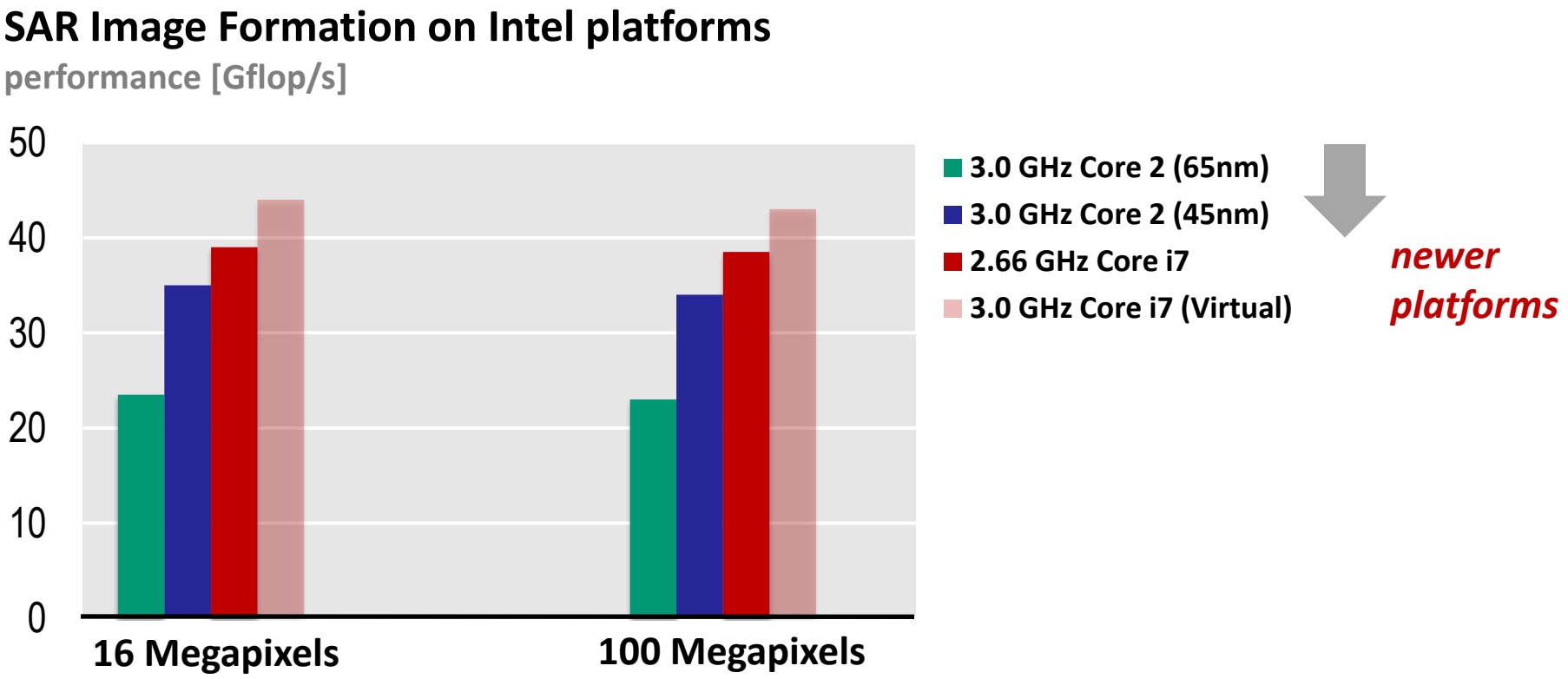


Memory bandwidth vs. CPU performance
25GB/s, 50GB/s, 100GB/s
1, 2, and 3-stage WHT algorithm

65nm process (simulation)

Q. Yu, D. F. Jones, N. Lawrence, D. McFarlin, P. A. Milder, B. Moore, A. Sidelnik, F. Franchetti, M. J. Garzaran, J. C. Hoe, J. Johnson, J. M. F. Moura, D. A. Padua, and M. Püschel:
The Data Pump Architecture for Algorithm-Specific DSP Processor/Program Co-Design. Submitted for publication.

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**

Direction 1: Emerging Multicore Platforms

Image: Intel

Intel Sandy Bridge
2 – 8 cores, SMT
AVX SIMD extension

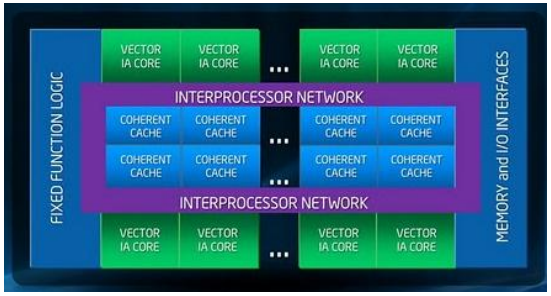


Image: Intel

Intel MIC
Up to 50 cores, SMT
Wide SIMD

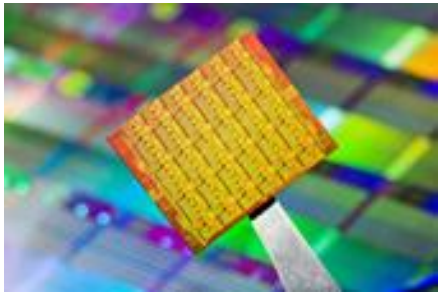


Image: Intel

Single Chip Cloud Computer
48 cores

Image: AMD

AMD Fusion APU
Multicore CPU + GPU

Image: Nvidia

Nvidia GTX480 Fermi
15 multiprocessors
448 cores

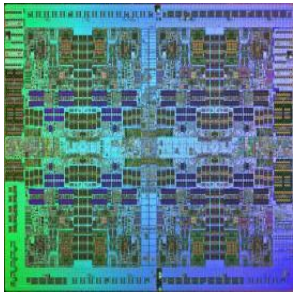


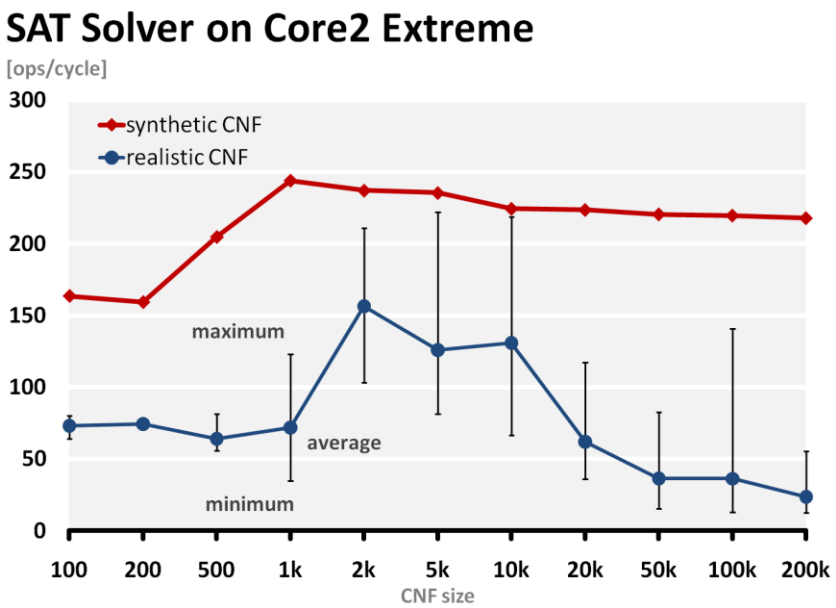
Image: IBM

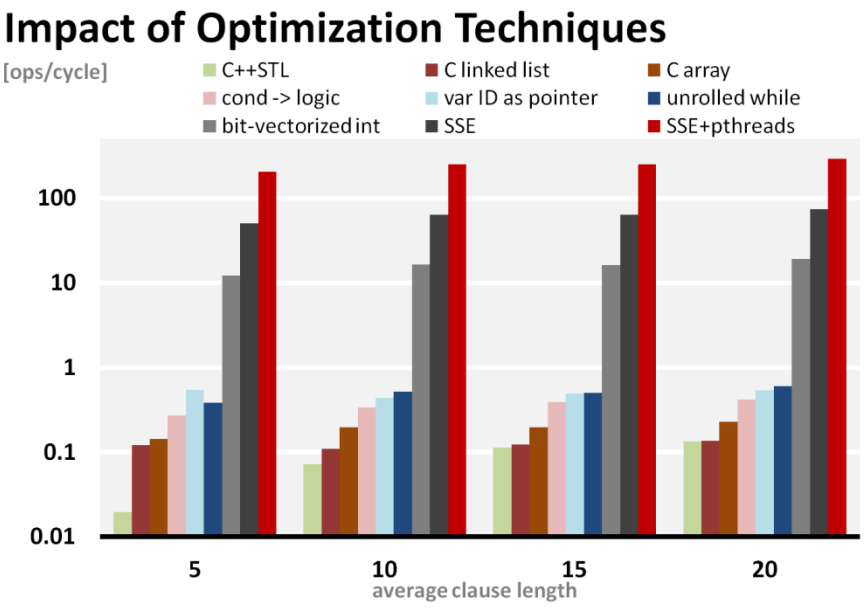
POWER7
8 cores, 4-way SMT
VSX SIMD extension

Direction 2: Beyond Regular Algorithms

Example: Random Walk SAT solver

find x_i s.t. $(x_0 + x_3' + x_5')\cdots(x_3' + x_4')(x_3 + x_5' + x_7 + x_8') = 1$





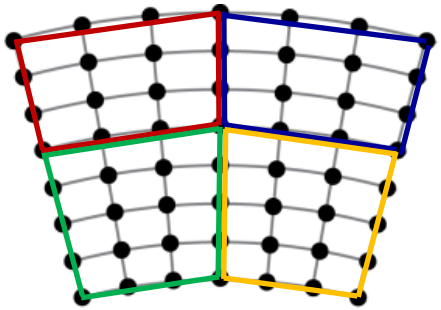
Catalogue optimization methods

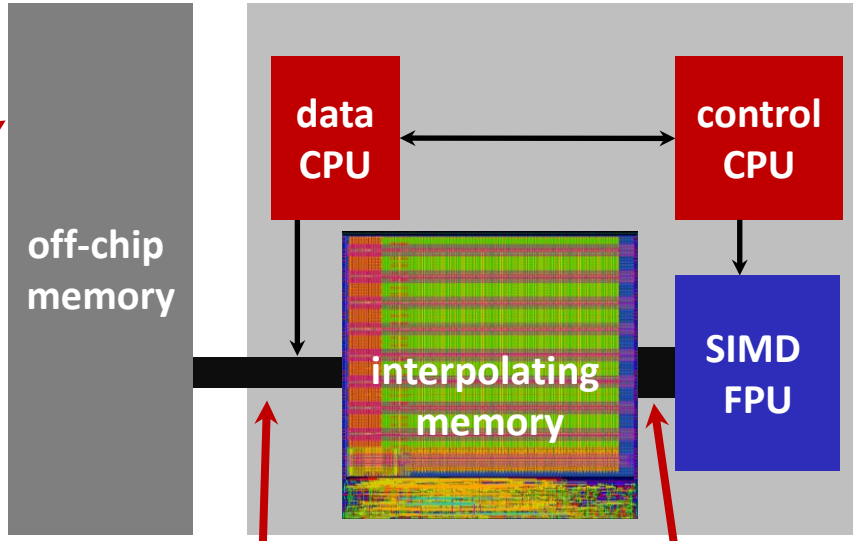
- Data layout transformations
- Control flow transformations
- Impact of C constructs
- Program Generation
- Autotuning
- Domain-specific optimization tricks

T. Cui and F. Franchetti: **Autotuning a Random Walk Boolean Satisfiability Solver.** To appear in Proceedings of iWAPT'11.

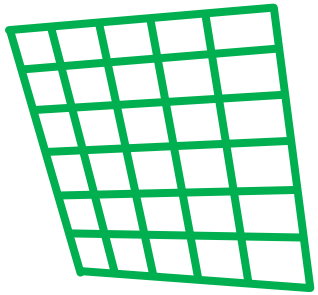
Direction 3: DPA and Logic-in-Memory

DRAM data
polar grid
tiled into trapezoids

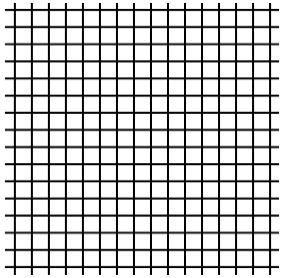




SRAM tile
low resolution
trapezoid



CPU view
high resolution
rectangular



Joint work with Zhu, Pileggi, Hoe, Horowitz in the context of the FCRP C2S2 center

Summary

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

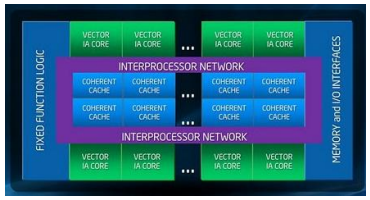
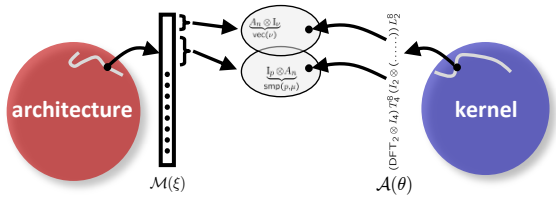
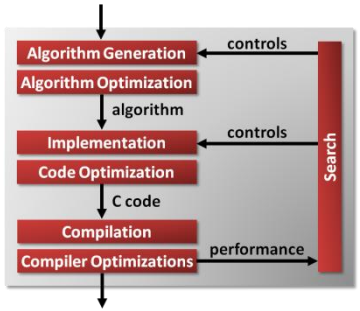


Image: Intel

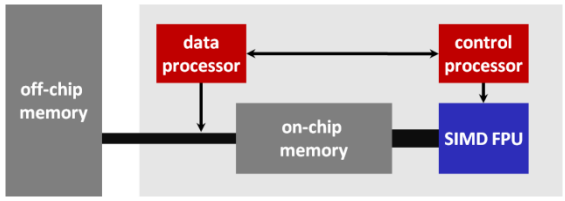
- **Unified mathematical framework**
  captures platforms and algorithms



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



- **Algorithm/architecture co-design**
  generate the architecture for a kernel

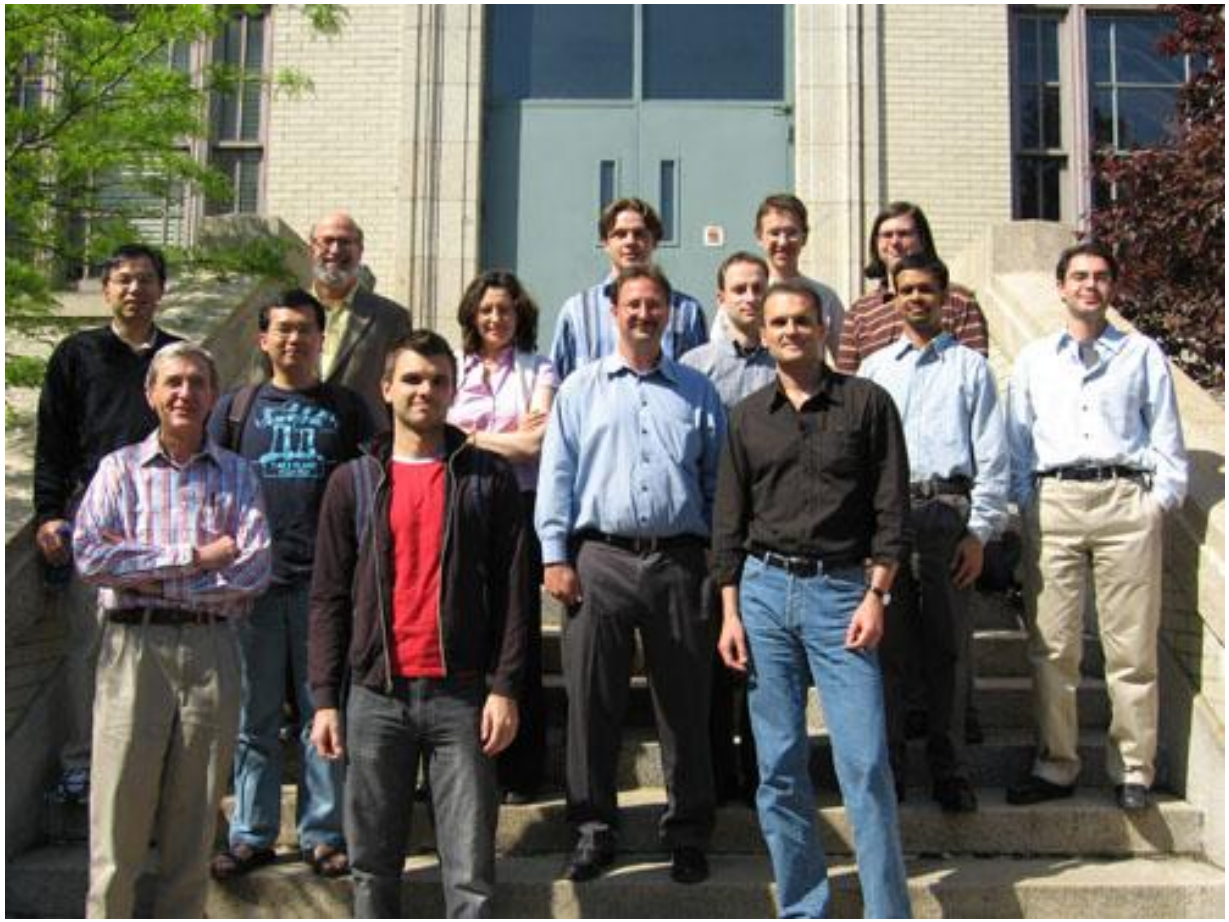


Acknowledgement

James C. Hoe
Jeremy Johnson
José M. F. Moura
David Padua
Larry Pileggi
Markus Püschel

Volodymyr Arbatov
Peter A. Milder
Yevgen Voronenko
Qian Yu

Berkin Akin
Christos Angelopoulos
Srinivas Chellappa
Frédéric de Mesmay
Daniel S. McFarlin
Marek R. Telgarsky
Qiuling Zhu



Special thanks to:
Randi Rost, Scott Buck (Intel), Jon Greene (Mercury Inc.), Yuanwei Jin (CMU)
Gheorghe Almasi, Jose E. Moreira, Jim Sexton (IBM)
Francois Gygi (LLNL, UC Davis), Kim Yates (LLNL), Kalyan Kumaran (ANL)

More Information:

www.spiral.net

www.spiralgen.com