Spiral
Program Synthesis for Performance

Joint work with …
Franz Franchetti
Yevgen Voronenko
Srinivas Chellappa
Frédéric de Mesmay
Daniel McFarlin
José Moura
James Hoe
…

…and the Spiral team (only part shown)

Markus Püschel

Computer Science

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SPIRAL
www.spiral.net

Scientific Computing
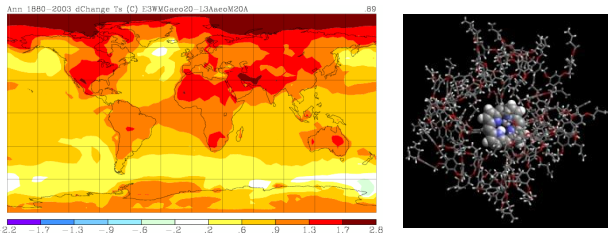
Physics/biology simulations

Consumer Computing

Audio/image/video processing

Embedded Computing
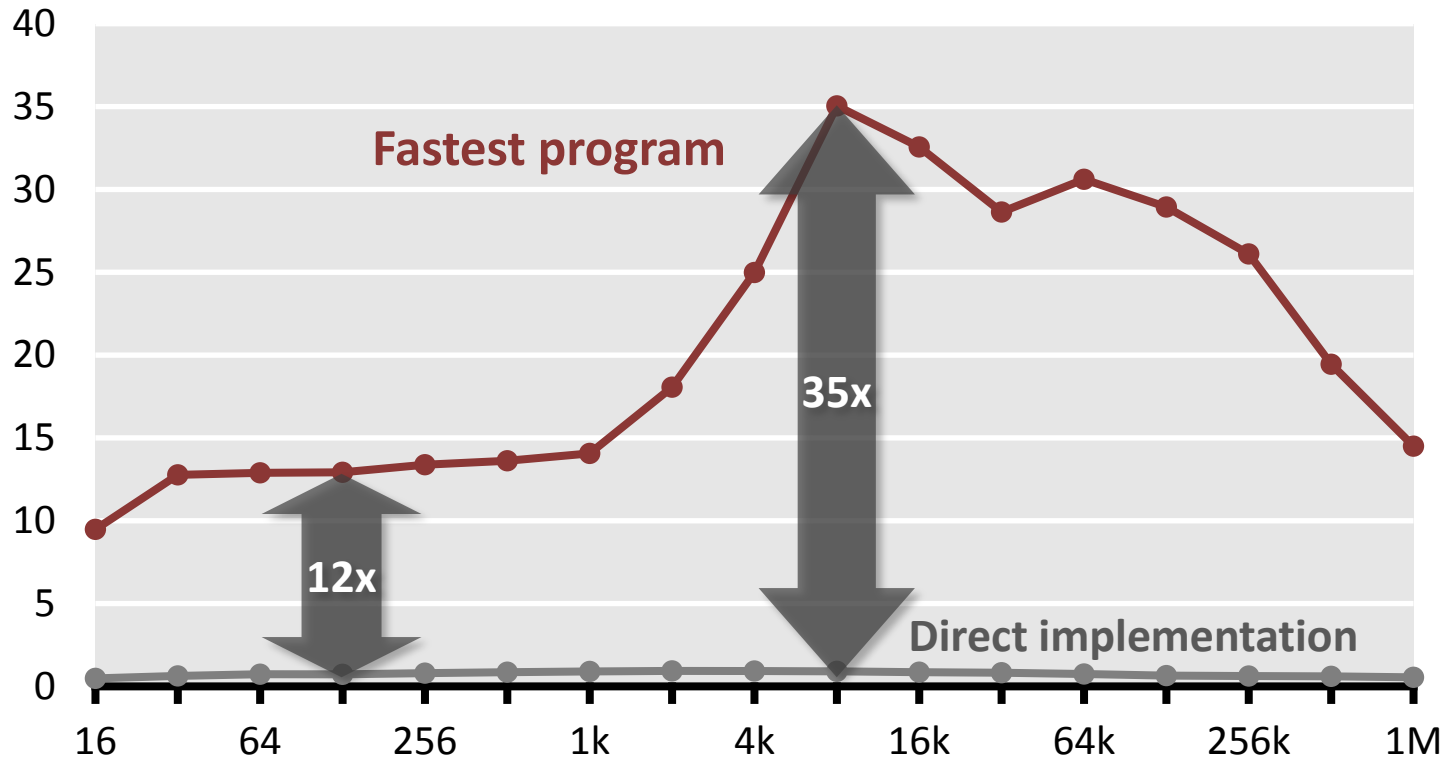

Signal processing, communication, control

Computing

- **Unlimited need for performance**

- **Large set of applications, but …**

- **Relatively small set of critical components (100s to 1000s)**
  - Matrix multiplication
  - Discrete Fourier transform
  - Viterbi decoder
  - Filter/stencil
  - ….

The Problem: Example DFT

DFT on Intel Core i7 (4 Cores, 2.66 GHz)
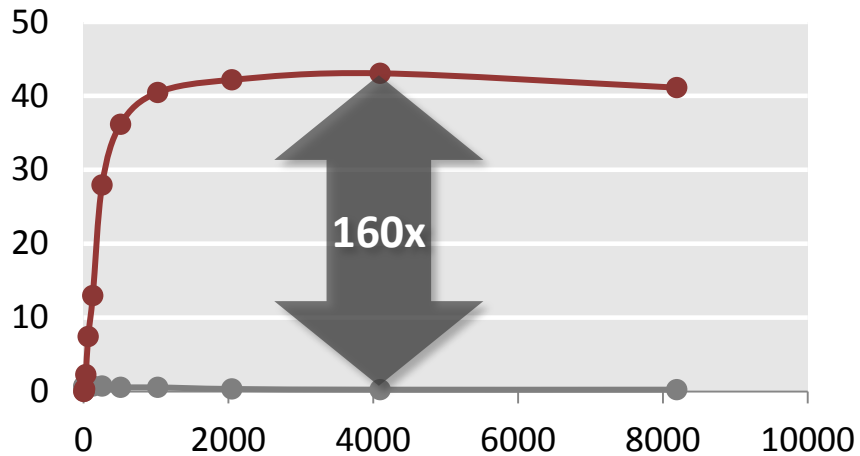
Performance [Gflop/s]

- **Same number of operations**
- **Best compiler**

The Problem Is Everywhere
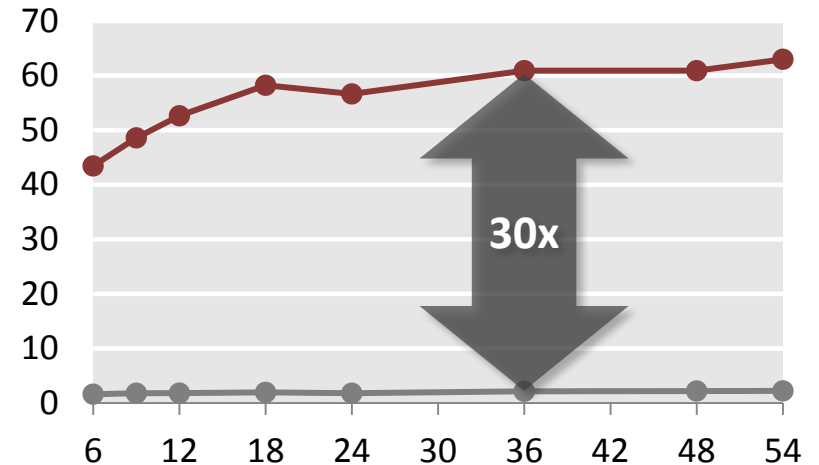
Matrix multiplication
Performance [Gflop/s]

160x

WiFi Receiver
Performance [Mbit/s]

30x

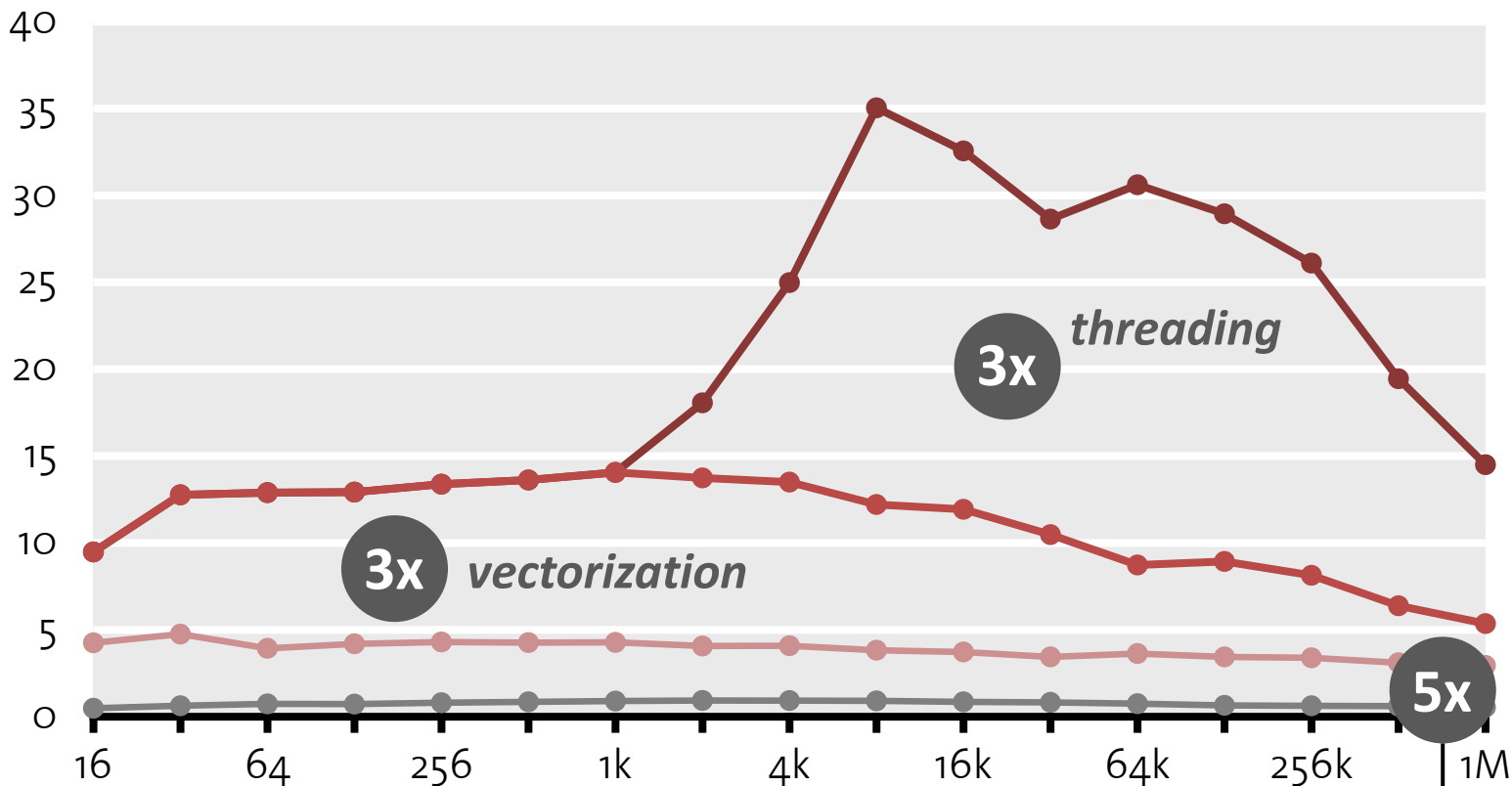## *Why is that?*

DFT: Analysis

DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)
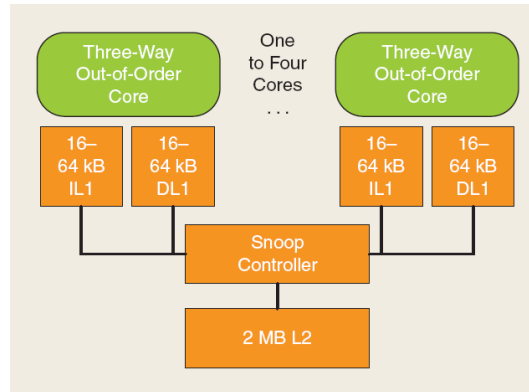Performance [Gflop/s]

- ■ **Compiler doesn't do it**

- ■ **Doing by hand: Very tough**

And There Is Not Only Intel …

Arm Cortex A9

Nvidia G200
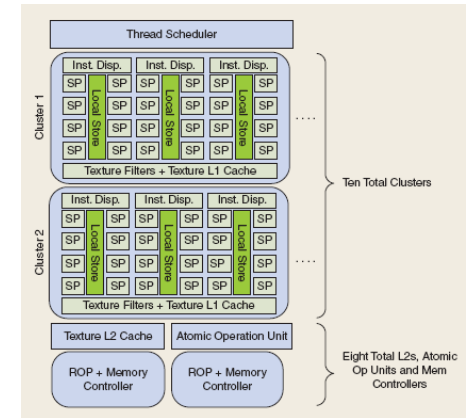
Core i7
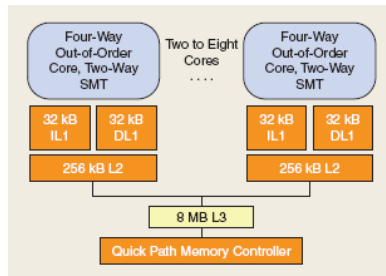
TI TNETV3020

Tilera Tile64

Current

Numerical problem

algorithm selection

implementation

C program →

compilation

human effort

automated

Computing platform
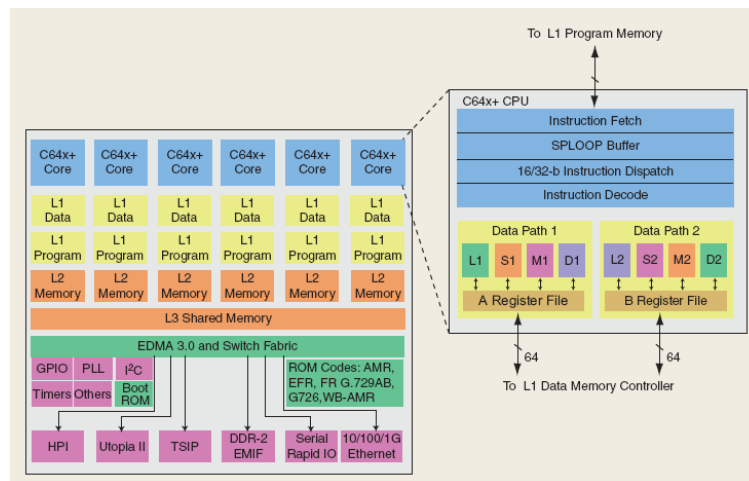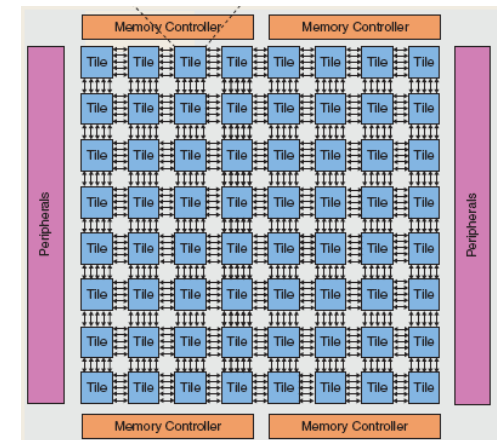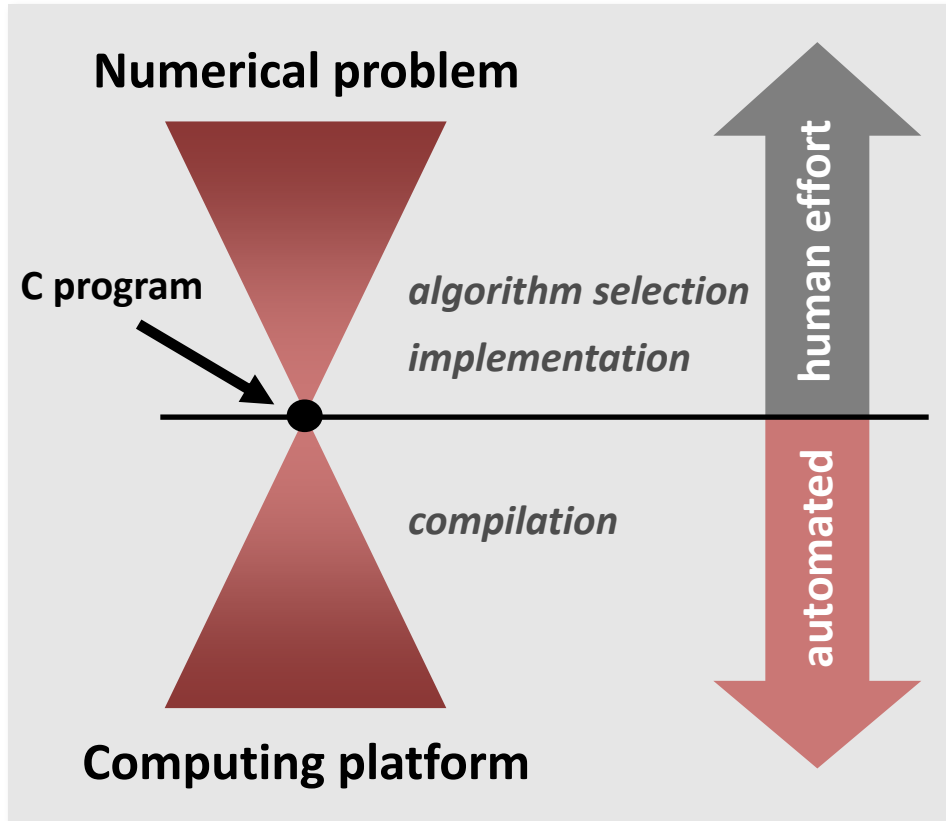
C code a singularity:

- **Compiler has no access to high level information**

- **No structural optimization**

- **No evaluation of choices**

Future

Numerical problem

algorithm selection

implementation

compilation

automated

Computing platform
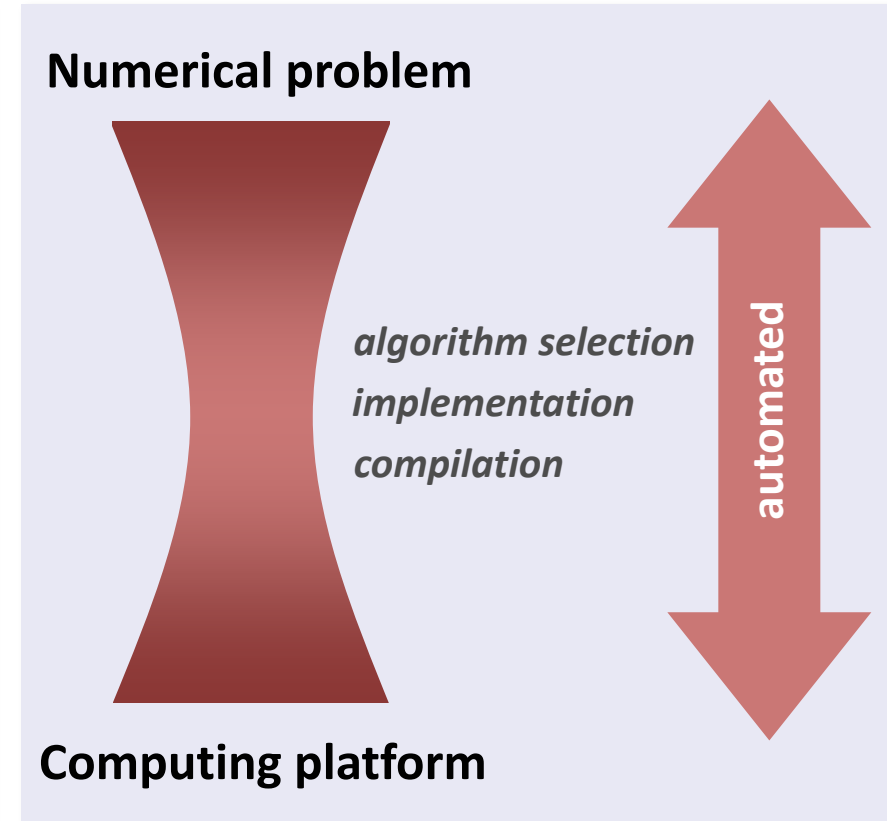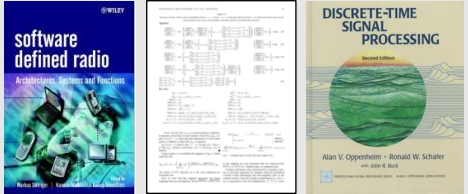
Challenge: conquer the high abstraction level for *complete automation*
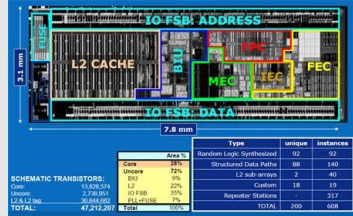
Current Solution

Algorithm knowledge

Processor knowledge
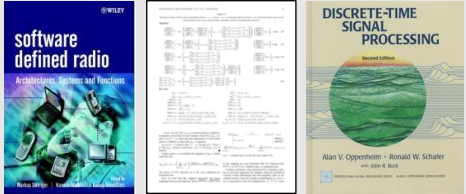
Optimal program
(Repeated for every processor)
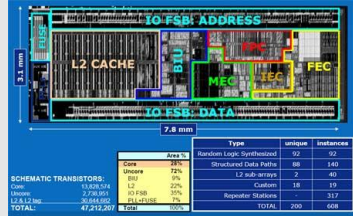
Our Research:
The Computer Writes the Program

Algorithm knowledge

Processor knowledge

Automation:
Spiral

Optimal program
(Regenerated for every processor)

"Computer Writes the Program"

Select transform

Transform	DCT2 ▾	transform
Size	2 ▾	number of samples

Pruning

Input	unpruned ▾	number of non-zero input samples
Output	unpruned ▾	number of non-zero output samples

Select implementation options

Data type	double precision ▾	data type
Scaling	-unscaled- ▾	output scaling

[Generate Code] [Reset]

"click"

- Computer writes close to optimal code

- Vectorized, parallelized, etc.

Organization

- Spiral: Basic system

- Parallelism

- General input size

- Results

- Final remarks

Linear Transforms

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = y = Tx \quad \longleftarrow \quad \boxed{T \cdot} \quad \longleftarrow \quad x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

Output   *Input*
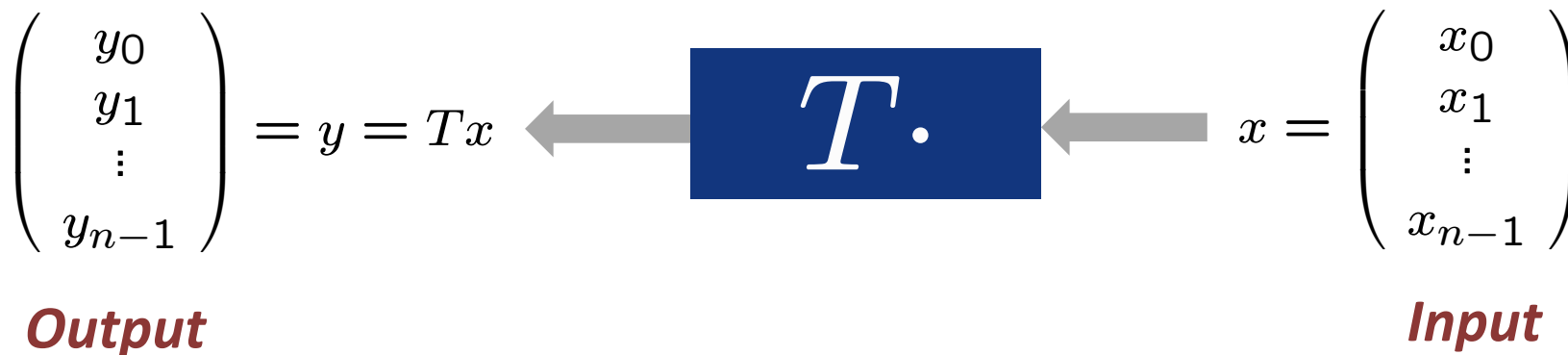
Example: $T = \mathbf{DFT}_n = [e^{-2k\ell\pi i/n}]_{0 \leq k,\ell < n}$

Algorithms: Example FFT, n = 4

Fast Fourier transform (FFT)

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

Representation using matrix algebra

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

- *SPL (Signal processing language):* **Mathematical, declarative, point-free**

- **Divide-and-conquer algorithms = breakdown rules in SPL**

Decomposition Rules (>200 for >40 Transforms)

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

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

$$\begin{vmatrix}\mathbf{rDFT}_{2n}(u)\\\mathbf{rDHT}_{2n}(u)\end{vmatrix} \to L_m^{2n}\left(I_k \; _i \begin{vmatrix}\mathbf{rDFT}_{2m}((i+u)/k)\\\mathbf{rDHT}_{2m}((i+u)/k)\end{vmatrix}\right)\left(\begin{vmatrix}\mathbf{rDFT}_{2k}(u)\\\mathbf{rDHT}_{2k}(u)\end{vmatrix} \quad I_m\right),$$

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

$$\mathbf{DCT\text{-}2}_n \to P_{k/2,2m}^\top \left(\mathbf{DCT\text{-}2}_{2m} K_2^{2m} \oplus \left(I_{k/2-1} \quad N_{2m} \mathbf{RDFT\text{-}3}_{2m}^\top\right)\right) B_n (L_{k/2}^{n/2} \quad I_2)(I_m \quad \mathbf{RDFT}_k')Q_{m/2,k},$$

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

<div style="background:#3a4a7a; color:white;">

Decomposition rules = Algorithm knowledge in Spiral

(from ≈100 publications)

</div>

$$\mathbf{DFT}_n \to (\mathbf{DFT}_k \quad I_m)\,T_m^n(I_k \quad \mathbf{DFT}_m)\,L_k^n, \quad n = km$$

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

$$\mathbf{DFT}_p \to R_p^\top (I_1 \oplus \mathbf{DFT}_{p-1})R_p, \quad p \text{ prime}$$

$$\mathbf{DCT\text{-}3}_n \to (I_m \oplus J_m)\, L_m^n (\mathbf{DCT\text{-}3}_m(1/4) \oplus \mathbf{DCT\text{-}3}_m(3/4))$$

$$\cdot (\mathsf{F}_2 \quad \mathsf{I}_m)\begin{bmatrix}\mathsf{I}_m & 0 \oplus -\mathsf{J}_{m-1}\\ & \frac{1}{\sqrt{2}}(\mathsf{I}_1 \oplus 2\,\mathsf{I}_m)\end{bmatrix}, \quad n = 2m$$

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

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

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

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

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

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

Combining these rules yields many algorithms for every given transform

SPL to Code

SPL S	Pseudo code for $y = Sx$
$A_n B_n$	```<code for: t = Bx>``` ```<code for: y = At>```

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

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

D_n	```for (i=0; i<n; i++)``` ``` y[i] = D[i]*x[i];```

L_k^{km}	```for (i=0; i<k; i++)```
	``` for (j=0; j<m; j++)```
	``` y[i*m+j] = x[j*k+i];```

F_2	```y[0] = x[0] + x[1];``` ```y[1] = x[0] - x[1];```

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

Correct code: easy   *fast code: very difficult*

Program Generation in Spiral

Transform	DFT_8

Decomposition rules

Algorithm *(SPL)*	$(\mathrm{DFT}_2 \otimes I_4)\, T_4^8\, (I_2 \otimes ((\mathrm{DFT}_2 \otimes I_2)$ $T_2^4\, (I_2 \otimes \mathrm{DFT}_2)\, L_2^4))\, L_2^8$

parallelization vectorization

Algorithm *(∑-SPL)*	$\sum \left(S_j\, \mathrm{DFT}_2\, G_j\right) \sum \left(\sum \left(S_{k,l}\, \mathrm{diag}\!\left(\mathsf{t}_{k,l}\right) \mathrm{DFT}_2\, G_l\right)\right.$ $\left.\sum \left(S_m\, \mathrm{diag}\!\left(\mathsf{t}_m\right) \mathrm{DFT}_2\, G_{k,m}\right)\right)$

locality optimization
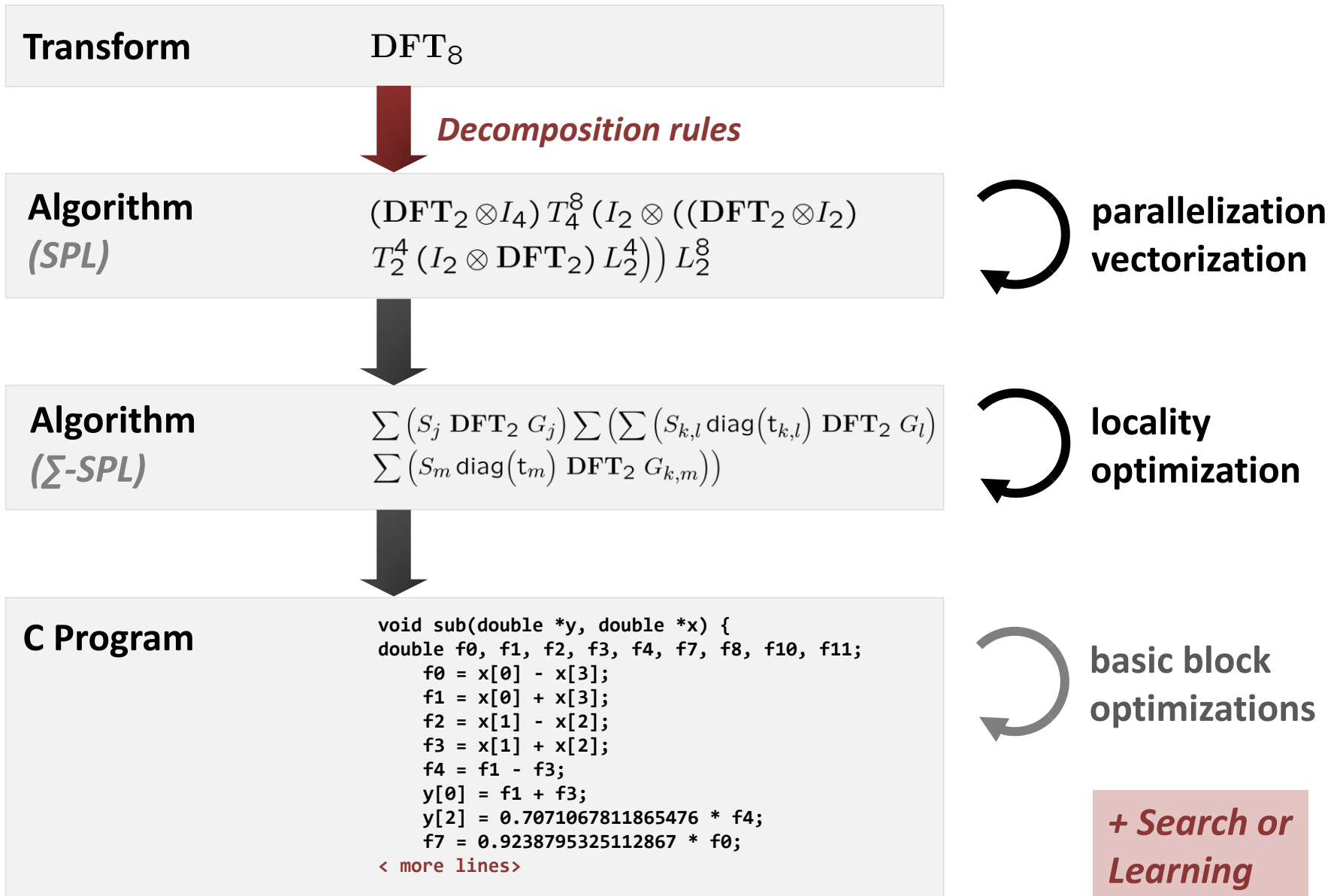
| **C Program** | ```
void sub(double *y, double *x) {
double f0, f1, f2, f3, f4, f7, f8, f10, f11;
    f0 = x[0] - x[3];
    f1 = x[0] + x[3];
    f2 = x[1] - x[2];
    f3 = x[1] + x[2];
    f4 = f1 - f3;
    y[0] = f1 + f3;
    y[2] = 0.7071067811865476 * f4;
    f7 = 0.9238795325112867 * f0;
< more lines>
``` |

basic block optimizations

+ Search or Learning

Organization

- Spiral: Basic system

- **Parallelism**

- General input size

- Results

- Final remarks

Example: Vectorization in Spiral

- **Relationship SPL expressions ↔ vectorization?**

$$y = \mathbf{DFT}_2 \, x$$

$$y = \begin{pmatrix} \mathbf{DFT}_2 & \mathrm{I}_4 \end{pmatrix} x$$

one addition
one subtraction

one (4-way) vector addition
one (4-way) vector subtraction

Step 1: Identify "Good" Vector Constructs

■ **Vector length:** ν

■ **Good (= easily vectorizable) SPL constructs:**

$$A \qquad \mathrm{I}_\nu$$

$$\mathsf{L}_\nu^{\nu^2}, \quad \mathsf{L}_2^{2\nu}, \mathsf{L}_\nu^{2\nu} \qquad \textit{base cases}$$

SPL expressions recursively built from those

■ ***Idea:*** **Convert a given SPL expression into a "good" SPL expression through rewriting (structural manipulation)**

Step 2: Find Manipulation Rules

$$\mathsf{L}_n^{n\nu} \rightarrow \left(\mathsf{I}_{n/\nu} \quad \mathsf{L}_\nu^{\nu^2}\right)\left(\mathsf{L}_{n/\nu}^n \quad \mathsf{I}_\nu\right)$$

$$\mathsf{L}_\nu^{n\nu} \rightarrow \left(\mathsf{L}_\nu^n \quad \mathsf{I}_\nu\right)\left(\mathsf{I}_{n/\nu} \quad \mathsf{L}_\nu^{\nu^2}\right)$$

$$\mathsf{L}_m^{mn} \rightarrow \left(\mathsf{L}_m^{mn/\nu} \quad \mathsf{I}_\nu\right)\left(\mathsf{I}_{mn/\nu^2} \quad \mathsf{L}_\nu^{\nu^2}\right)\left(\mathsf{I}_{n/\nu} \quad \mathsf{L}_{m/\nu}^m \quad \mathsf{I}_\nu\right)$$

$$\mathsf{I}_l \quad \mathsf{L}_n^{kmn} \quad \mathsf{I}_r \rightarrow \left(\mathsf{I}_l \quad \mathsf{L}_n^{kn} \quad \mathsf{I}_{mr}\right)\left(\mathsf{I}_{kl} \quad \mathsf{L}_n^{mn} \quad \mathsf{I}_r\right)$$

$$\mathsf{I}_l \quad \mathsf{L}_n^{kmn} \quad \mathsf{I}_r \rightarrow \left(\mathsf{I}_l \quad \mathsf{L}_{kn}^{kmn} \quad \mathsf{I}_r\right)\left(\mathsf{I}_l \quad \mathsf{L}_{mn}^{kmn} \quad \mathsf{I}_r\right)$$

$$\mathsf{I}_l \quad \mathsf{L}_{km}^{kmn} \quad \mathsf{I}_r \rightarrow \left(\mathsf{I}_{kl} \quad \mathsf{L}_m^{mn} \quad \mathsf{I}_r\right)\left(\mathsf{I}_l \quad \mathsf{L}_k^{kn} \quad \mathsf{I}_{mr}\right)$$

Manipulation rules = Processor knowledge in Spiral

$$\left(\mathsf{I}_m \quad A^{n\times n}\right)\mathsf{L}_m^{mn} \rightarrow \left(\mathsf{I}_{m/\nu} \quad \mathsf{L}_\nu^{n\nu}\left(A^{n\times n} \quad \mathsf{I}_\nu\right)\right)\left(\mathsf{L}_{m/\nu}^{mn/\nu} \quad \mathsf{I}_\nu\right)$$

$$\mathsf{L}_n^{mn}\left(\mathsf{I}_m \quad A^{n\times n}\right) \rightarrow \left(\mathsf{L}_n^{mn/\nu} \quad \mathsf{I}_\nu\right)\left(\mathsf{I}_{m/\nu} \quad \left(A^{n\times n} \quad \mathsf{I}_\nu\right)\mathsf{L}_n^{n\nu}\right)$$

$$\left(\mathsf{I}_k \quad \left(\mathsf{I}_m \quad A^{n\times n}\right)\mathsf{L}_m^{mn}\right)\mathsf{L}_k^{kmn} \rightarrow \left(\mathsf{L}_k^{km} \quad \mathsf{I}_n\right)\left(\mathsf{I}_m \quad \left(\mathsf{I}_k \quad A^{n\times n}\right)\mathsf{L}_k^{kn}\right)\left(\mathsf{L}_m^{mn} \quad \mathsf{I}_k\right)$$

$$\mathsf{L}_{mn}^{kmn}\left(\mathsf{I}_k \quad \mathsf{L}_n^{mn}\left(\mathsf{I}_m \quad A^{n\times n}\right)\right) \rightarrow \left(\mathsf{L}_n^{mn} \quad \mathsf{I}_k\right)\left(\mathsf{I}_m \quad \mathsf{L}_n^{kn}\left(\mathsf{I}_k \quad A^{n\times n}\right)\right)\left(\mathsf{L}_m^{km} \quad \mathsf{I}_n\right)$$

$$\overline{A\,B} \rightarrow \overline{A}\,\overline{B}$$

$$\overline{A^{m\times m} \quad \mathsf{I}_\nu} \rightarrow \left(\mathsf{I}_m \quad \mathsf{L}_\nu^{2\nu}\right)\left(\overline{A^{m\times m}} \quad \mathsf{I}_\nu\right)\left(\mathsf{I}_m \quad \mathsf{L}_2^{2\nu}\right)$$

$$\overline{\mathsf{I}_m \quad A^{n\times n}} \rightarrow \mathsf{I}_m \quad \overline{A^{n\times n}}$$

$$\overline{D} \rightarrow \left(\mathsf{I}_{n/\nu} \quad \mathsf{L}_\nu^{2\nu}\right)\vec{D}\left(\mathsf{I}_{n/\nu} \quad \mathsf{L}_2^{2\nu}\right)$$

$$\overline{P} \rightarrow P \quad \mathsf{I}_2$$

Example

$$\underbrace{\mathbf{DFT}_{mn}}_{\mathsf{vec}(\nu)} \ \rightarrow \ \underbrace{(\mathbf{DFT}_m \quad \mathrm{I}_n)\top_n^{mn}(\mathrm{I}_m \quad \mathbf{DFT}_n)\llcorner_m^{mn}}_{\mathsf{vec}(\nu)}$$

...

...

...

$$\rightarrow \ \left(\mathrm{I}_{\frac{mn}{\nu}} \quad \llcorner_\nu^{2\nu}\right)\left(\overline{\mathbf{DFT}_m} \quad \mathrm{I}_{\frac{n}{\nu}} \quad \mathrm{I}_\nu\right)\overline{\top}'^{mn}_n$$

$$\left(\mathrm{I}_{\frac{m}{\nu}} \quad \left(\llcorner_\nu^{2n} \quad \mathrm{I}_\nu\right)\left(\mathrm{I}_{\frac{2n}{\nu}} \quad \llcorner_\nu^{\nu^2}\right)\left(\mathrm{I}_{\frac{n}{\nu}} \quad \llcorner_2^{2\nu} \quad \mathrm{I}_\nu\right)\left(\overline{\mathbf{DFT}_n} \quad \mathrm{I}_\nu\right)\right)\left(\llcorner_{\frac{m}{\nu}}^{\frac{mn}{\nu}} \quad \llcorner_2^{2\nu}\right)$$

vectorized arithmetic
vectorized data accesses

Automatically Generate Base Case Library

- *Goal:* **Given instruction set, generate base cases**

$$\nu = 4: \quad \left\{ \mathrm{L}_2^4, \mathrm{I}_2 \quad \mathrm{L}_2^4, \mathrm{L}_2^4 \quad \mathrm{I}_2, \mathrm{L}_2^8, \mathrm{L}_4^8 \right\}$$

- *Idea:* **Instructions as matrices + search**

```
y = _mm_unpacklo_ps(x0, x1);
```

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{x}_0 \\ \vec{x}_1 \end{bmatrix}$$

```
y = _mm_shuffle_ps(x0, x1, _MM_SHUFFLE(1,2,1,2));
```

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{x}_0 \\ \vec{x}_1 \end{bmatrix}$$

```
y = _mm_shuffle_ps(x0, x1, _MM_SHUFFLE(3,4,3,4));
```

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x}_0 \\ \vec{x}_1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
y0 = _mm_shuffle_ps(x0, x1,
        _MM_SHUFFLE(1,2,1,2));   );
y1 = _mm_shuffle_ps(x0, x1,
        _MM_SHUFFLE(3,4,3,4));   ;
```


$$\mathrm{L}_2^4 \otimes \mathrm{I}_2$$
~~No base case~~
Base case

Same Approach for Different Paradigms

Threading:

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

$$\cdots$$

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

$$\cdots$$

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

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

Vectorization:

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

$$\cdots$$

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

$$\cdots$$

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

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

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

GPUs:

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

$$\cdots$$

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

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

Verilog for FPGAs:

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

$$\cdots$$

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

$$\cdots$$

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

- Rigorous, correct by construction
- Overcomes compiler limitations

Algorithm knowledge

Processor knowledge

Automation:
Spiral

Optimal program
(Regenerated for every processor)

Decomposition rules

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

$$\left| \begin{matrix} \mathbf{rDFT}_{2n}(u) \\ \mathbf{rDHT}_{2n}(u) \end{matrix} \right| \rightarrow L_m^{2n} \left(I_k \otimes_i \left| \begin{matrix} \mathbf{rDFT}_{2m}((i+u)/k) \\ \mathbf{rDHT}_{2m}((i+u)/k) \end{matrix} \right| \right) \left(\left| \begin{matrix} \mathbf{rDFT}_{2k}(u) \\ \mathbf{rDHT}_{2k}(u) \end{matrix} \right| \otimes I_m \right)$$

$$\mathbf{RDFT\text{-}3}_n \rightarrow (Q_{k/2,m}^\top \otimes I_2) \left(I_k \otimes_i \mathbf{rDFT}_{2m})(i+1/2)/k) \right) (\mathbf{RDFT\text{-}3}_k \otimes I_m)$$

Manipulation rules

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

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

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

Automation:

Spiral

Optimal program
(Regenerated for every processor)

Organization

- Spiral: Basic system

- Parallelism

- **General input size**

- Results

- Final remarks

Challenge: General Size Libraries

So far:
Code specialized to fixed input size

```
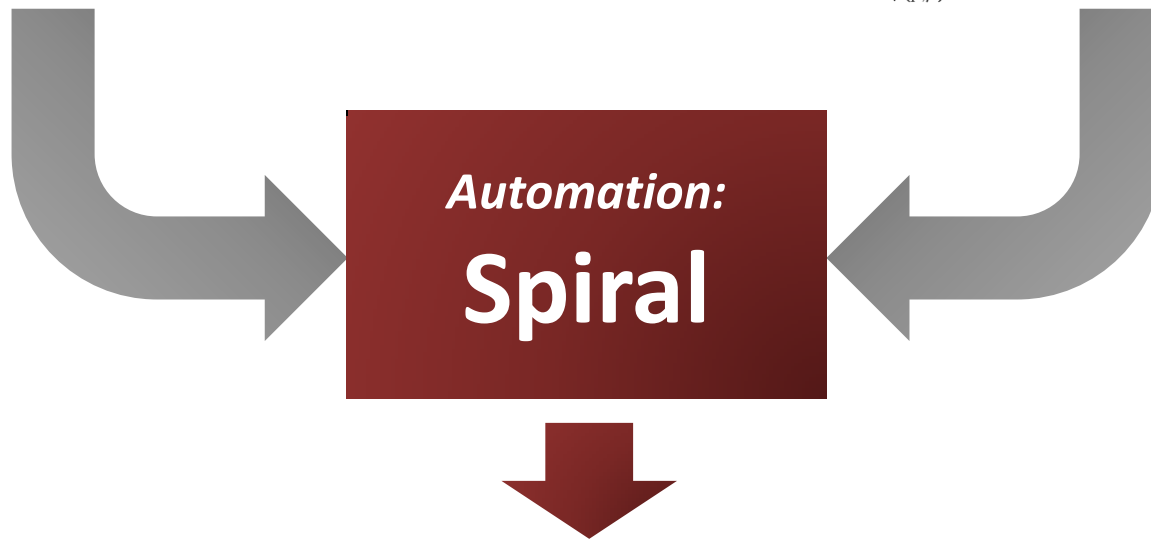DFT_384(x, y) {
  …
  for(i = …) {
   t[2i]   = x[2i] + x[2i+1]
   t[2i+1] = x[2i] - x[2i+1]
  }
  …
}
```

- **Algorithm fixed**
- **Nonrecursive code**

Challenge:
Library for general input size

```
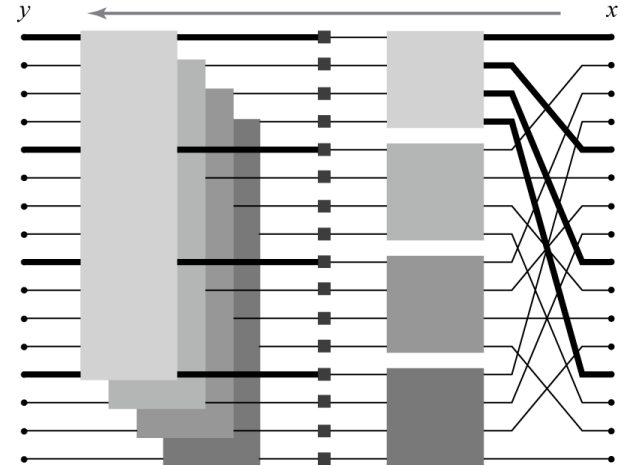DFT(n, x, y) {
  …
  for(i = …) {
    DFT_strided(m, x+mi, y+i, 1, k)
  }
  …
}
```

- **Algorithm cannot be fixed**
- **Recursive code**
- **Creates many challenges**

Challenge: Recursion Steps

- **Cooley-Tukey FFT**

$$y = (\mathbf{DFT}_k \otimes I_m)T_m^{km}(I_k \otimes \mathbf{DFT}_m)L_k^{km}x$$

- **Implementation that increases locality (e.g., FFTW 2.x)**

```
void DFT(int n, cpx *y, cpx *x) {
  int k = choose_dft_radix(n);
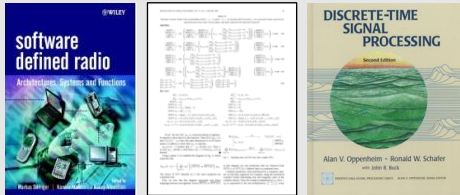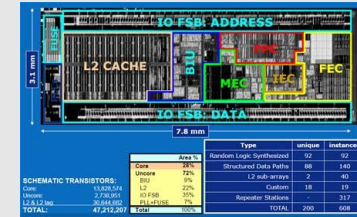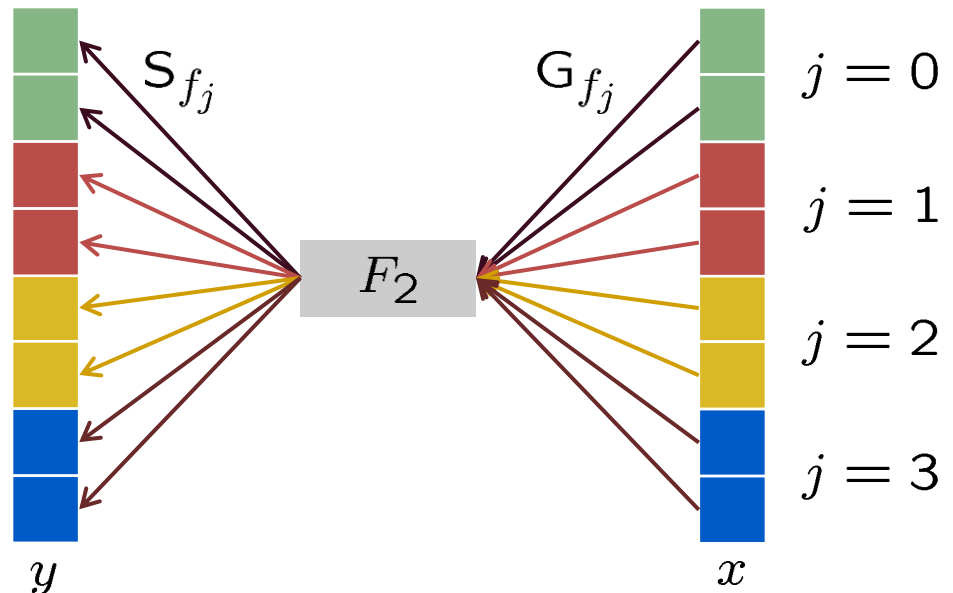
  for (int i=0; i < k; ++i)
    DFTrec(m, y + m*i, x + i, k, 1);
  for (int j=0; j < m; ++j)
    DFTscaled(k, y + j, t[j], m);
}
```

Σ–SPL : Basic Idea

- **Four additional matrix constructs: Σ, G, S, Perm**
  - **Σ (sum)**   explicit loop
  - **G_f (gather)**   load data with index mapping f
  - **S_f (scatter)**   store data with index mapping f
  - **Perm_f**   permute data with the index mapping f

- **Σ-SPL formulas = matrix factorizations**

  Example: $y = (I_4 \otimes F_2)x \;\rightarrow\; y = \sum_{j=0}^{3} \mathsf{S}_{f_j}\, F_2\, \mathsf{G}_{f_j}\, x$

$$y = \begin{bmatrix} F_2 & & & \\ & F_2 & & \\ & & F_2 & \\ & & & F_2 \end{bmatrix} x$$

Find Recursion Step Closure

Voronenko, 2008

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

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

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

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

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

Repeat until closure

Recursion Step Closure: Examples

DFT: scalar code

DFT: full-fledged (vectorized and parallel code)

Summary: Complete Automation for Transforms

- **Memory hierarchy optimization**
  Rewriting and search for algorithm selection
  Rewriting for loop optimizations

- **Vectorization**
  Rewriting

- **Parallelization**
  Rewriting   *fixed input size code*

- **Derivation of library structure**
  Rewriting
  Other methods   *general input size library*

Organization

- Spiral: Basic system

- Parallelism

- General input size

- **Results**

- Final remarks

DFT on Intel Multicore

Complex DFT (Intel Core i7, 2.66 GHz, 4 cores)
Performance [Gflop/s] vs. input size

$$\mathbf{DFT}_n \rightarrow (\mathbf{DFT}_k \otimes I_m) T_m^n (I_k \otimes \mathbf{DFT}_m) L_k^n$$
$$\mathbf{DFT}_n \rightarrow P_{k/2,2m}^\top \left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m} \mathbf{rDFT}_{2m}(i/k) \right) \right) (\mathbf{RDFT}_k \otimes I_m)$$
$$\mathbf{RDFT}_n \rightarrow (P_{k/2,m}^\top \otimes I_2) \left(\mathbf{RDFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i D_{2m} \mathbf{rDFT}_{2m}(i/k) \right) \right) (\mathbf{RDFT}_k \otimes I_m)$$
$$\mathbf{rDFT}_{2n}(u) \rightarrow L_m^{2n} \left(I_k \otimes_i \mathbf{rDFT}_{2m}((i+u)/k) \right) (\mathbf{rDFT}_{2k}(u) \otimes I_m)$$

Spiral

5MB vectorized, threaded, general-size, adaptive library

Often it Looks Like That

DCTs on 2.66 GHz Core2 (4-way SSSE3)
Performance [Gflop/s]

Spiral-generated

Available libraries

input size n

Computer generated Functions for Intel IPP 6.0

3984 C functions

1M lines of code

Transforms: DFT (fwd+inv), RDFT (fwd+inv), DCT2, DCT3, DCT4, DHT, WHT

Sizes: 2–64 (DFT, RDFT, DHT); 2-powers (DCTs, WHT)

Precision: single, double

Data type: scalar, SSE, AVX (DFT, DCT), LRB (DFT)

Computer generated

Results: SpiralGen Inc.

Mapping to FPGAs

DFT 1024 (16 bit fixed point) on Xilinx Virtex-5 FPGA

throughput [million samples per second]

performance [Gop/s]

best

Spiral generated

Xilinx Logicore 4.1

area [slices]

Organization

- Spiral: Basic system

- Parallelism

- General input size

- Results

- **Final remarks**

Beyond Transforms

Matrix-Matrix Multiplication

$$\text{MMM}_{1,1,1} \to (\cdot)_1$$

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

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

$$\text{MMM}_{m,n,k} \to ((\Sigma_{k/k_b} \circ (\cdot)_{k/k_b}) \otimes \text{MMM}_{m,n,k_b}) \circ$$
$$((L_{k/k_b}^{mk/k_b} \otimes I_{k_b}) \times I_{kn})$$

$$\text{MMM}_{m,n,k} \to (L_m^{mn/n_b} \otimes I_{n_b}) \circ$$
$$((\otimes)_{1 \times n/n_b} \otimes \text{MMM}_{m,n_b,k}) \circ$$
$$(I_{km} \times (L_{n/n_b}^{kn/n_b} \otimes I_{n_b}))$$

Viterbi Decoder

$$\mathbf{F}_{K,F} \to \prod_{i=1}^{F} \left(\left(I_{2^{K-2}} \otimes_j B_{F-i,j} \right) L_{2^{K-2}}^{2^{K-1}} \right)$$

JPEG 2000 (Wavelet, EBCOT)

$$\text{SC}(\chi_{m,n}, \sigma_{m,n}) : (\mathbb{Z}_2^9 \times \mathbb{Z}_2^9) \to (\mathbb{N}, \mathbb{Z}_2)$$

$$(I \times \text{xor}_2) \circ (T_{SC} \times I) \circ (H \times V \times I) \circ (\underline{L_4^2} \times G_4) \circ (\begin{pmatrix} 1 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix})$$

$$H, V : (\mathbb{Z}_2^9 \times \mathbb{Z}_2^9) \to \mathbb{N}$$
$$H: \quad h \circ (f \times f) \circ (G_1 \times C_{-2} \times G_1 \times G_7 \times C_{-2} \times G_7) \circ \underline{L_4^2} \circ (\begin{pmatrix} 1 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix})$$
$$V: \quad h \circ (f \times f) \circ (G_3 \times C_{-2} \times G_3 \times G_5 \times C_{-2} \times G_5) \circ \underline{L_4^2} \circ (\begin{pmatrix} 1 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix})$$
$$f : \text{mul}_2 \circ (I \times \text{sub}_2) \circ (I \times C_1 \times \text{mul}_2)$$
$$h : \text{min}_2 \circ (C_1 \times \text{max}_2) \circ (C_{-1} \times \text{sum}_2)$$

Synthetic Aperture Radar (SAR)

$$
\begin{aligned}
\text{SAR} &\to 2\text{D-iDFT} \circ \text{Interpl} \circ \text{MatchFilt} \circ \text{prep} \\
2\text{D-iDFT} &\to \text{iDFT} \otimes \text{iDFT} \\
\text{MatchFilt} &\to \text{Filt} \circ (I \times C_f) \\
\text{Filt} &\to (I \otimes (\cdot)) \\
\text{Interpl} &\to (\Sigma \otimes I) \circ (I \otimes_j S_{x_j \otimes y_j}) \circ \text{Filt} \circ ((I \otimes \mathbf{1} \otimes I) \times I) \circ (I \times C_{i \otimes_j g_j})
\end{aligned}
$$

Related Work: Autotuning

- *Goal:* **(Partial) automation of runtime optimization**

- **Projects**
  - ATLAS (U. Tennessee)
  - FFTW (MIT)
  - BeBop/OSKI (U. Berkeley)
  - Spiral (Carnegie Mellon)

  - Tensor contractions (Ohio State)
  - Fenics (some universities)
  - FLAME (UT Austin)
  - Adaptive sorting (UIUC)
  - **<many others>**

Proceedings of the IEEE special issue, Feb. 2005

Spiral: Summary

- **Spiral:**

  Successful approach to automating
  the development of computing software

  Commercial proof-of-concept

  Intel® Integrated Performance Primitives (Intel® IPP) 6.0

- **Key ideas:**

  Algorithm knowledge:
  Domain specific symbolic representation

  Platform knowledge:
  Tagged rewrite rules, SIMD specification

$$\mathbf{DFT}_{64}$$

```
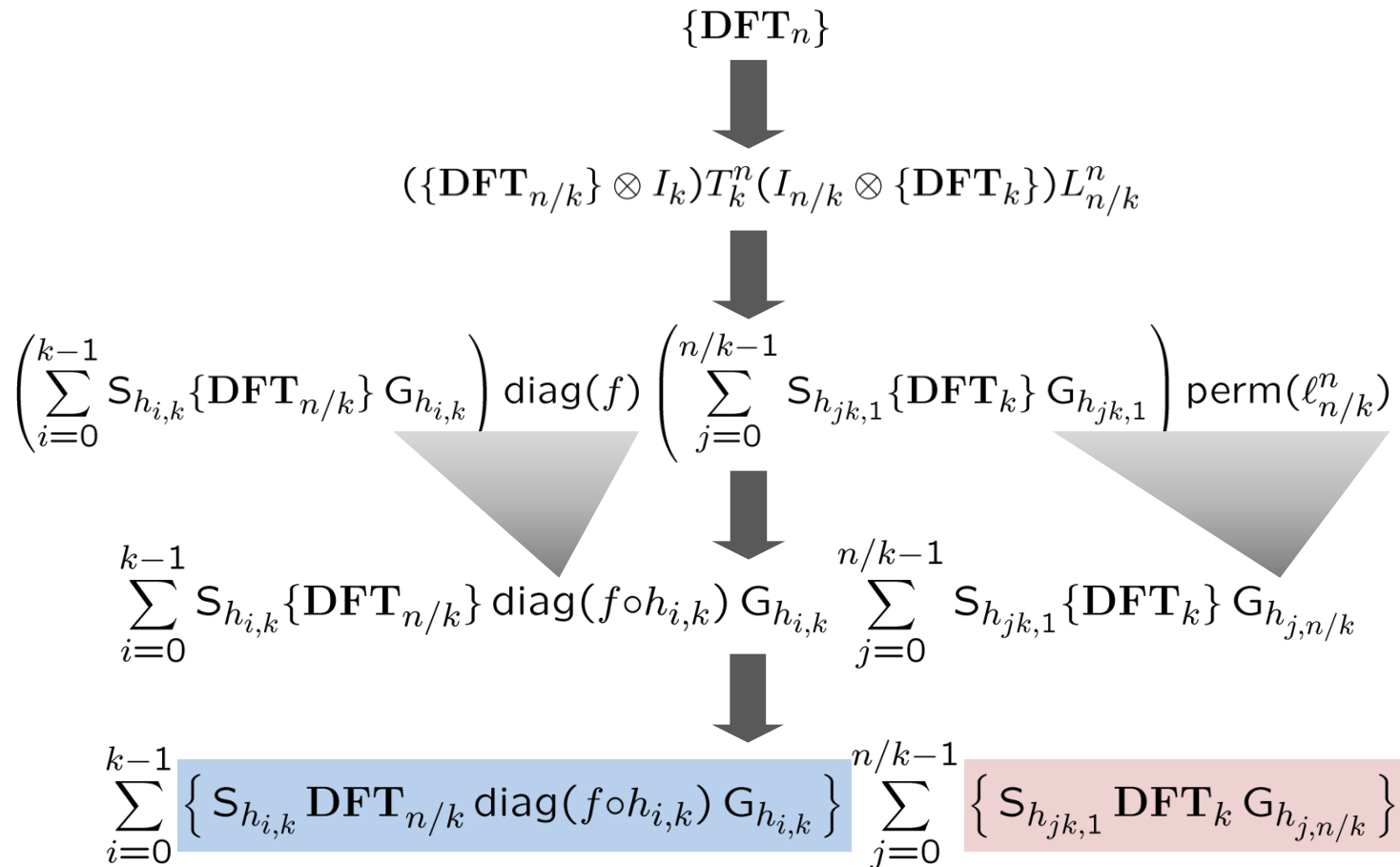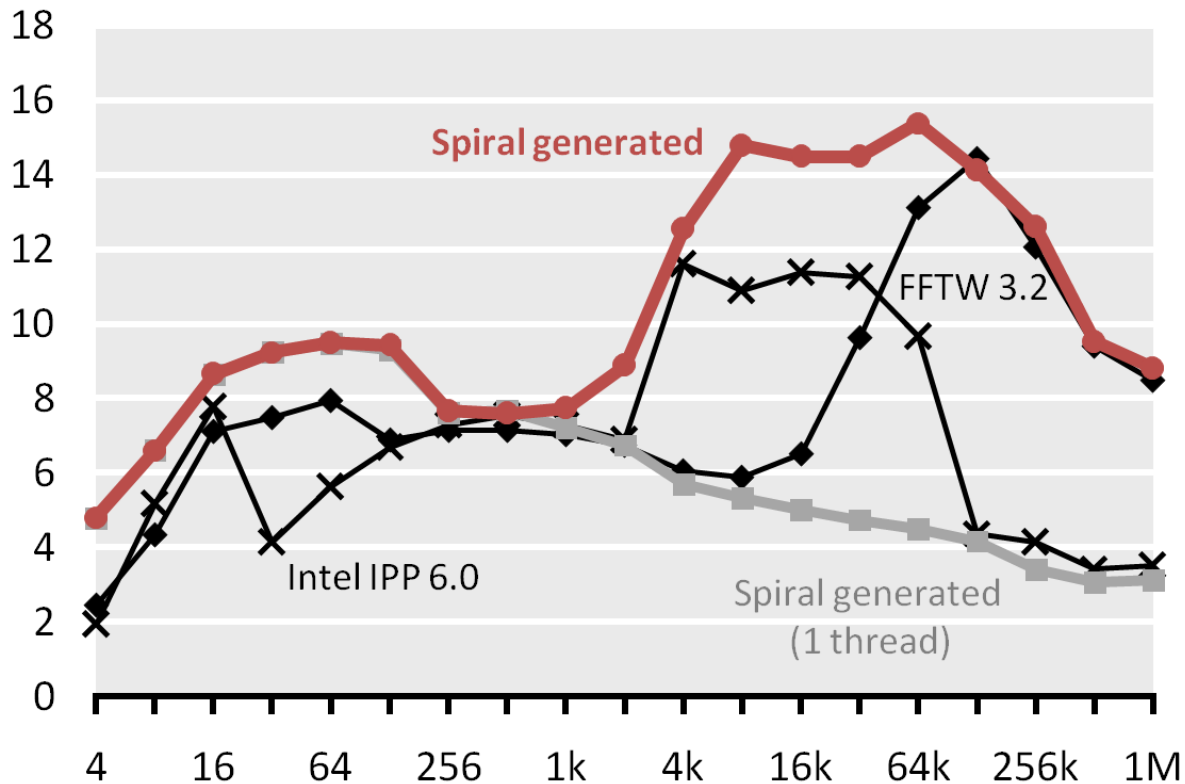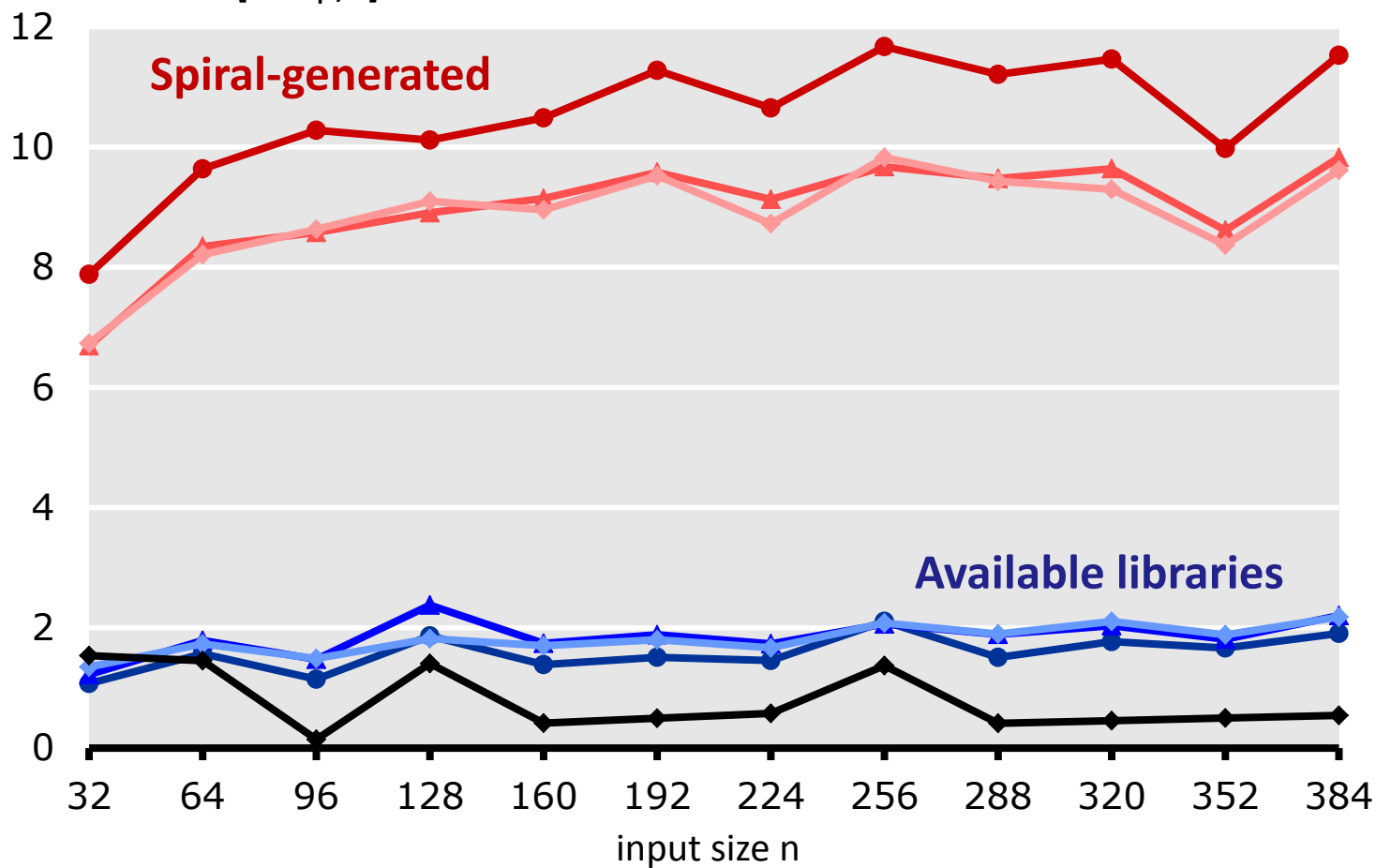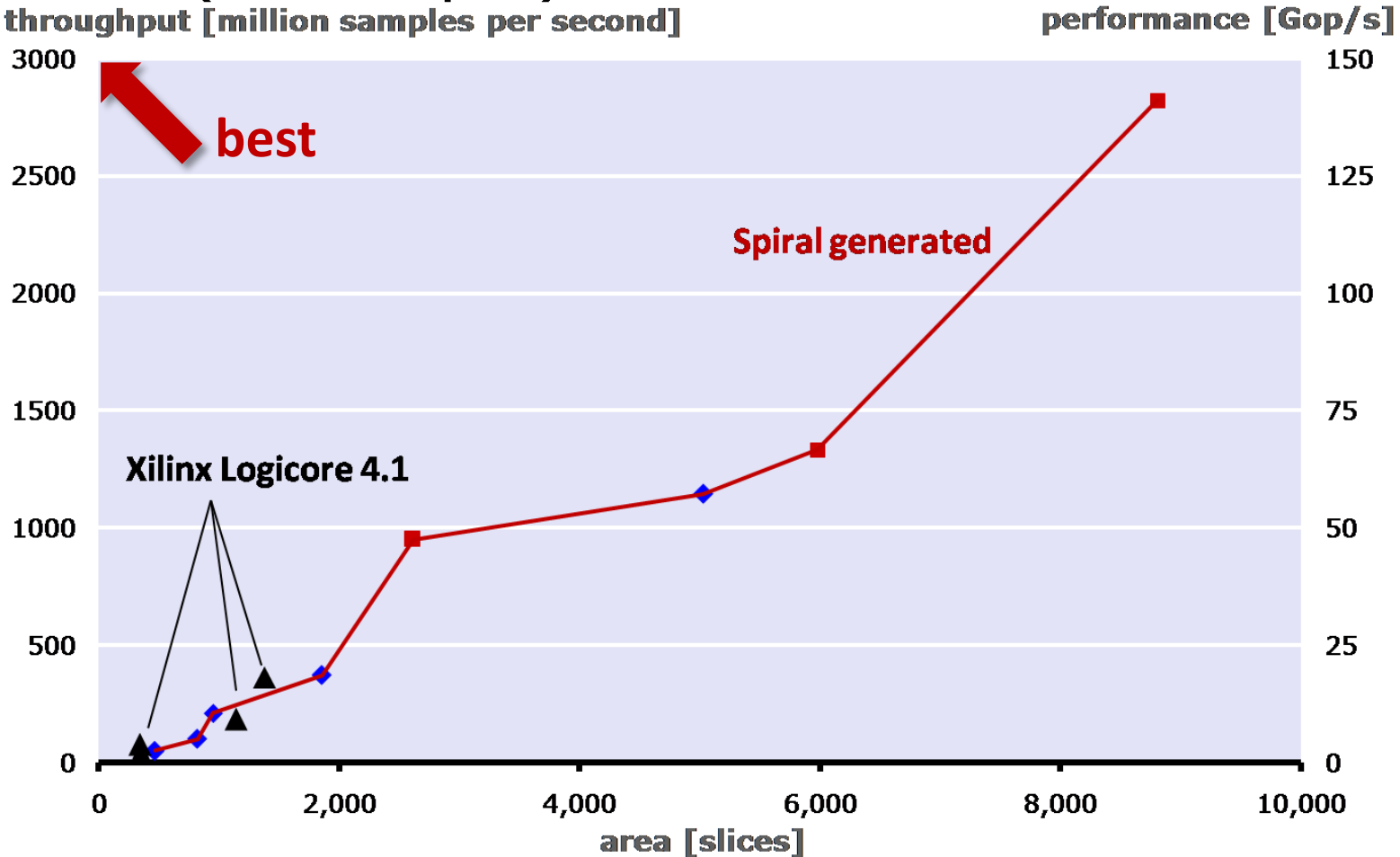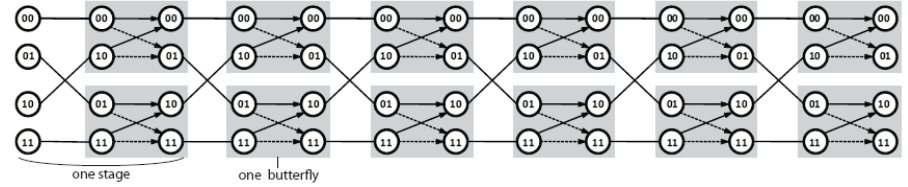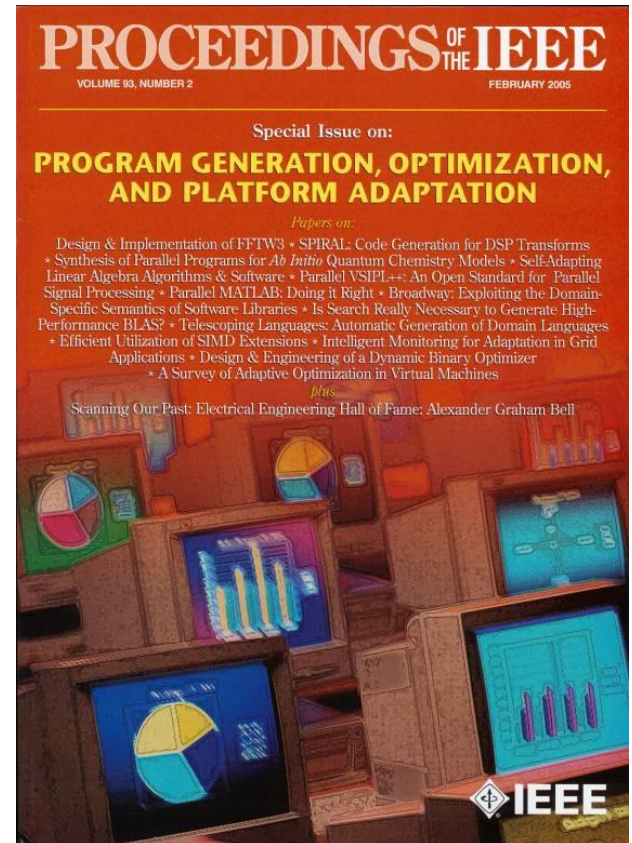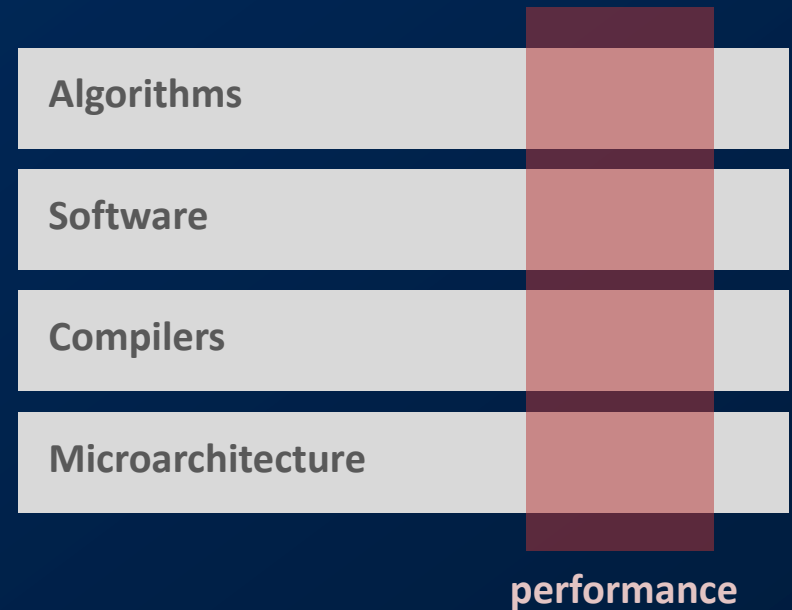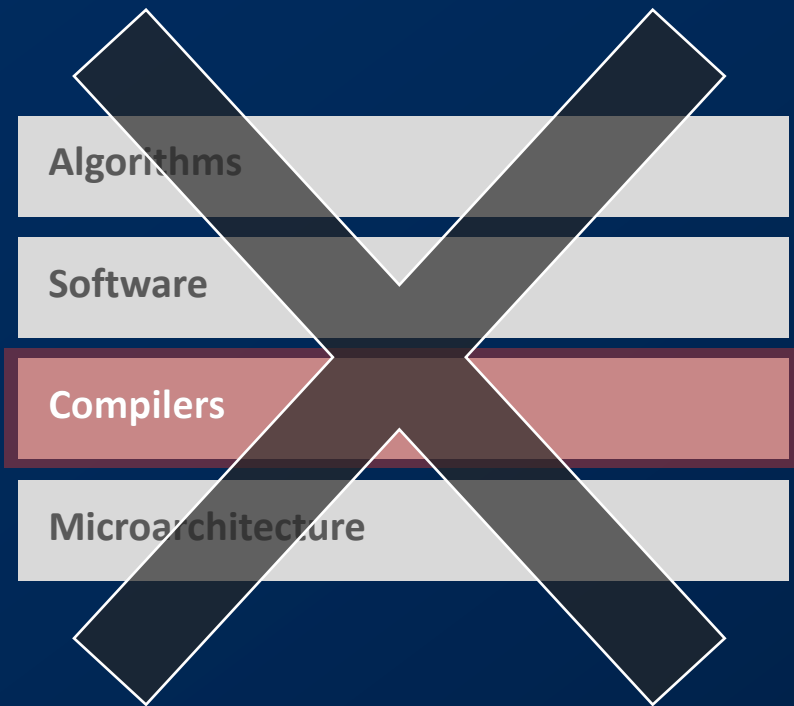void dft64(float  *Y, float  *X) {
  __m512 U912, U913, U914, U915,...
  __m512  *a2153, *a2155;
  a2153 = ((__m512  *) X);   s1107 = *(a2153);
  s1108 = *((a2153 + 4));    t1323 = _mm512_add_ps(s1107,s1108);
  t1324 = _mm512_sub_ps(s1107,s1108);
   <many more lines>
  U926 = _mm512_swizupconv_r32(…);
  s1121 = _mm512_madd231_ps(_mm512_mul_ps(_mm512_mask_or_pi(
   _mm512_set_1to16_ps(0.70710678118654757),0xAAAA,a2154,U926),t1341),
   _mm512_mask_sub_ps(_mm512_set_1to16_ps(0.70710678118654757),…),
   _mm512_swizupconv_r32(t1341,_MM_SWIZ_REG_CDAB));
  U927 = _mm512_swizupconv_r32
   <many more lines>
}
```

$$\mathbf{DFT}_4 \rightarrow (\mathbf{DFT}_2 \otimes \mathbf{I}_2)\mathsf{T}_2^4(\mathbf{I}_2 \otimes \mathbf{DFT}_2)\mathsf{L}_2^4$$

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

35x

Performance is different than other software quality features