Spiral

Computer Synthesis of Computational Programs

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

Markus Püschel

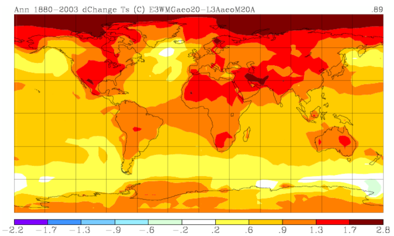
Computer Science

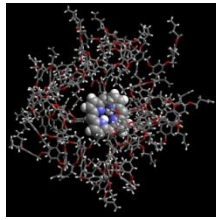
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SPIRAL
www.spiral.net

Supported by DARPA, ONR, NSF, Intel, Mercury

Scientific Computing



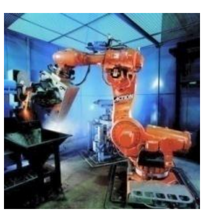


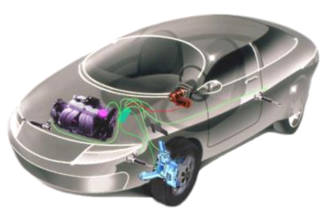
Physics/biology simulations

Consumer Computing

Audio/image/video processing

Embedded Computing



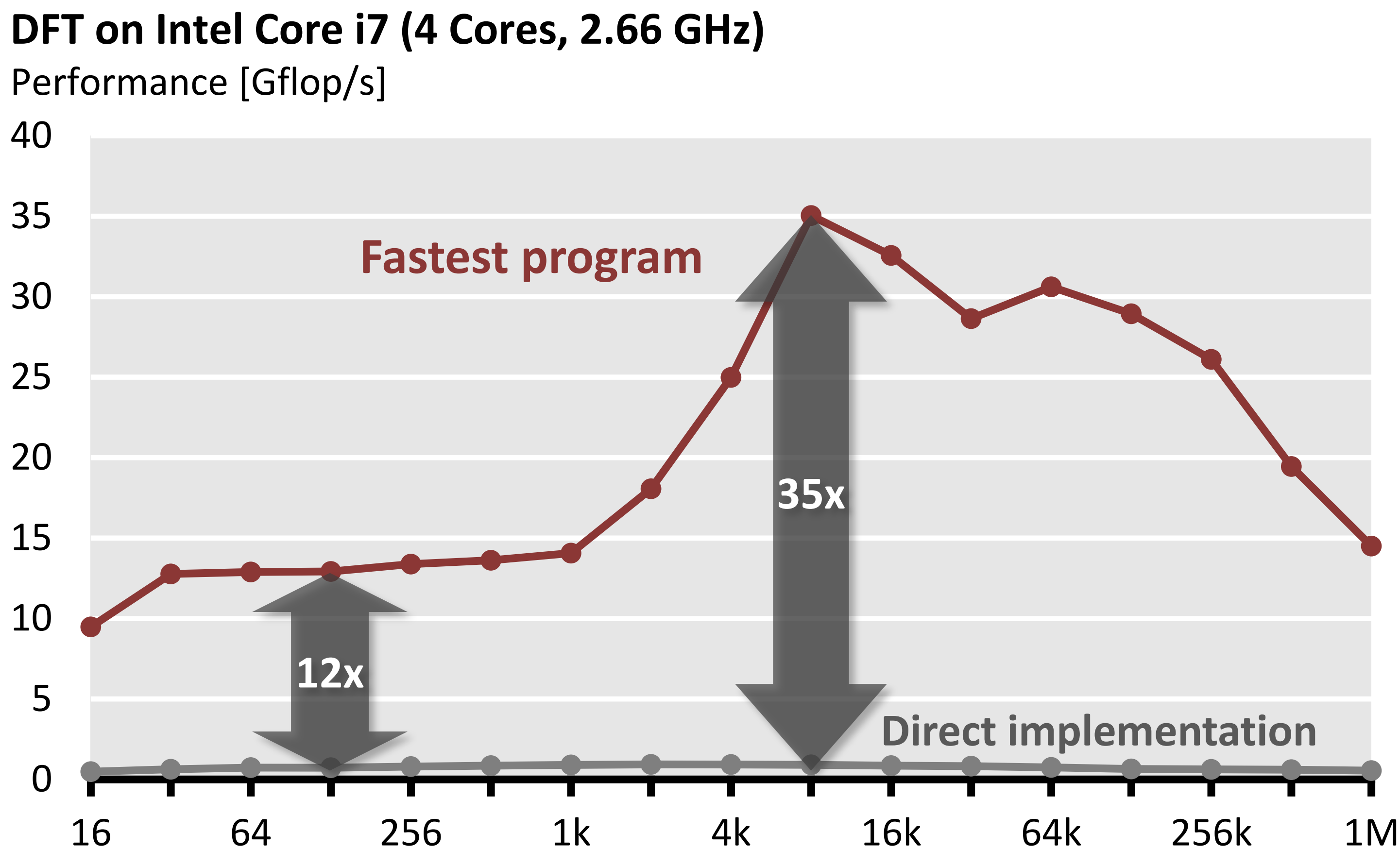


Signal processing, communication, control

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
  - ….

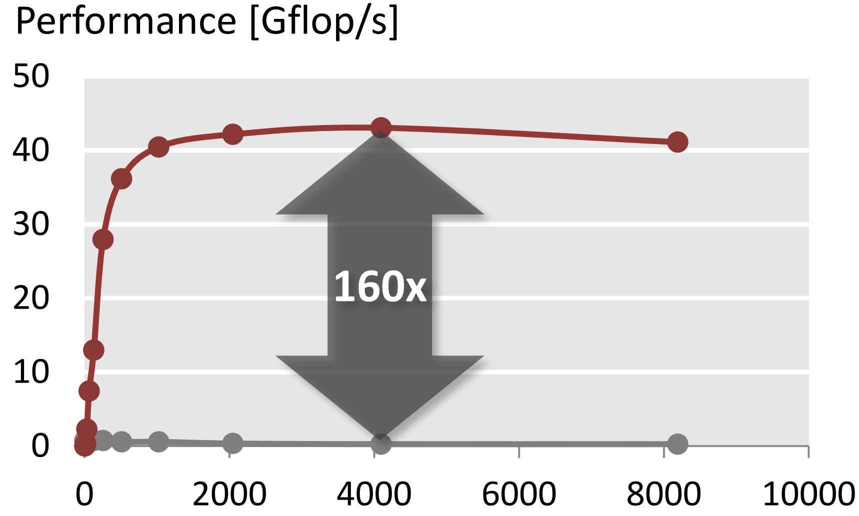
The Problem: Example DFT



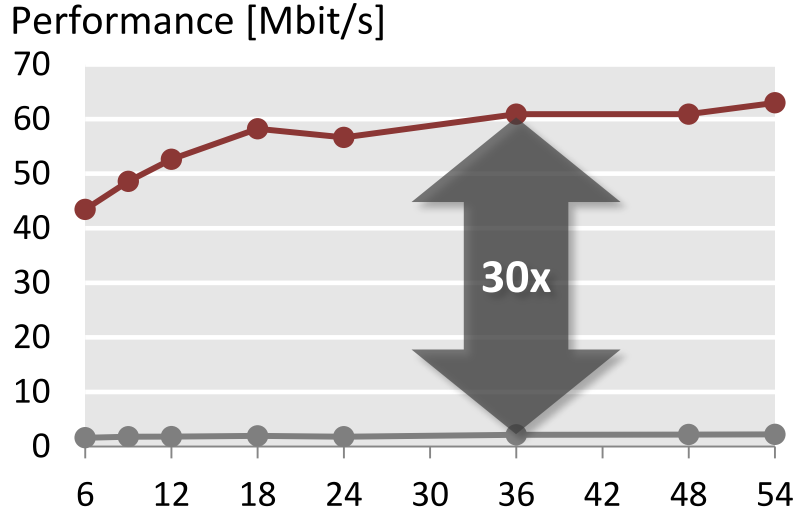
- Same number of operations
- Best compiler

The Problem Is Everywhere

Matrix multiplication



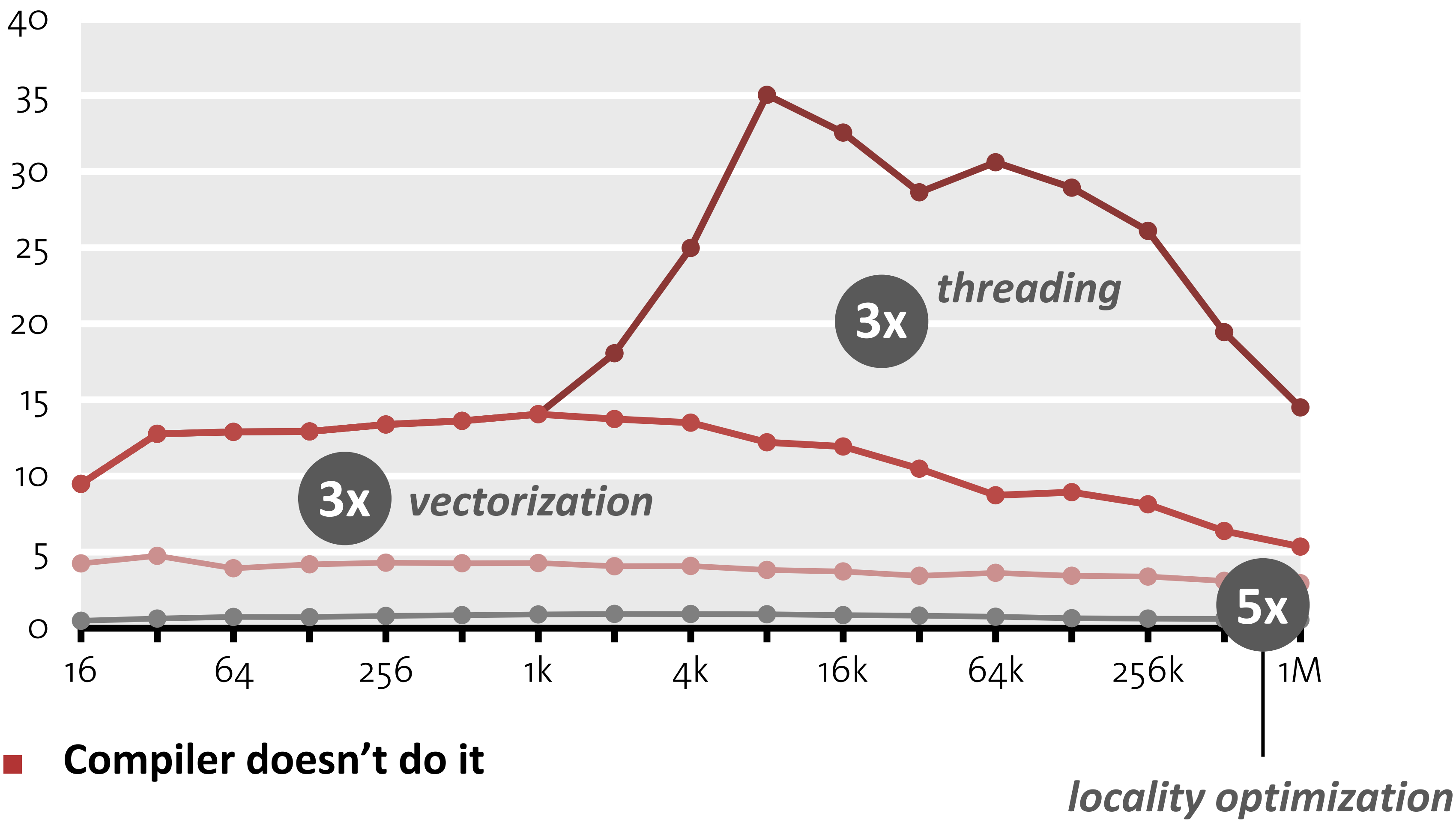
WiFi Receiver



Why is that?

DFT: Analysis

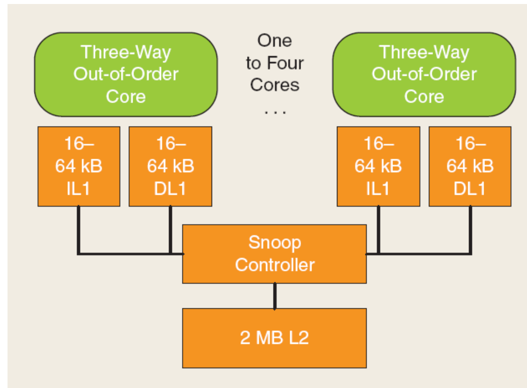
DFT (single precision) on Intel Core i7 (4 cores, 2.66 GHz)
Performance [Gflop/s]



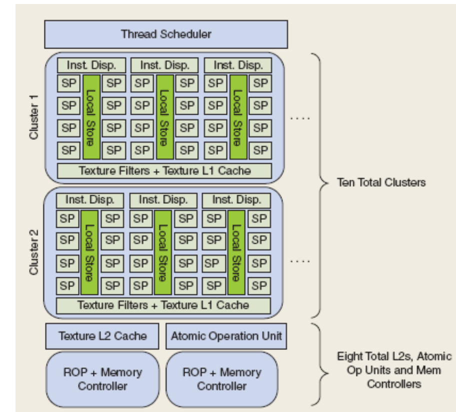
- **Compiler doesn't do it**
- **Doing by hand: Very tough**

And There Is Not Only Intel …

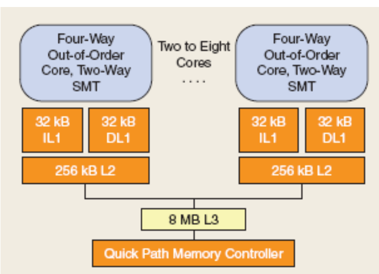
Arm Cortex A9



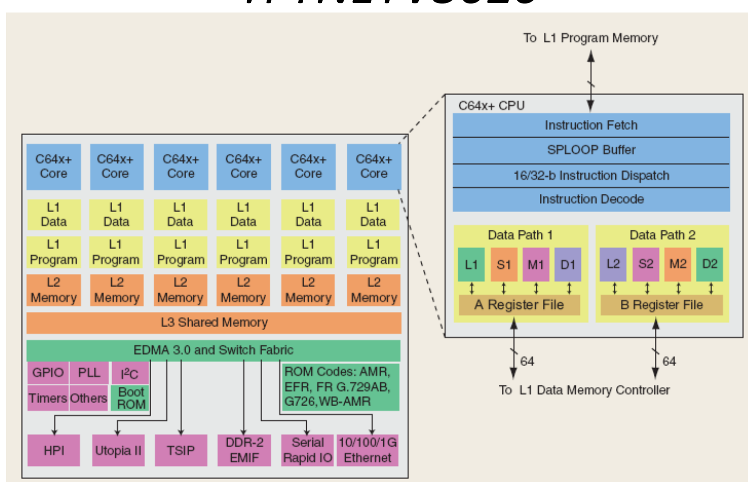
Nvidia G200



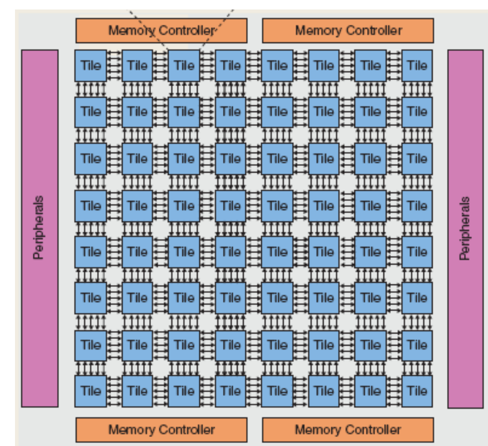
Core i7



TI TNETV3020

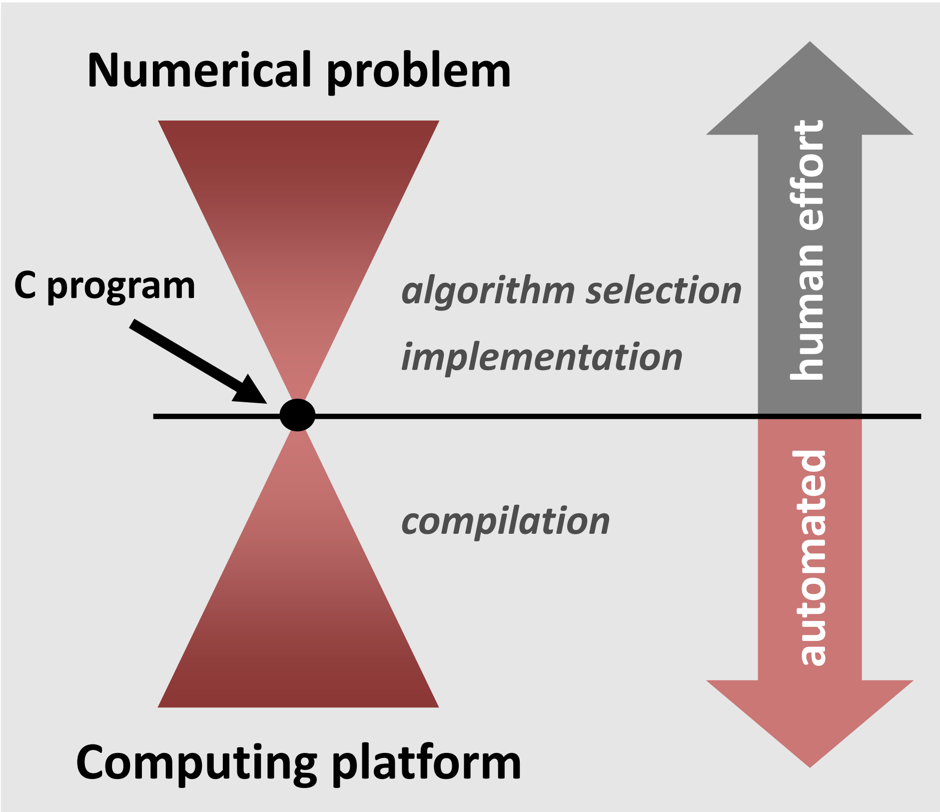


Tilera Tile64



Source: IEEE SP Magazine, Vol. 26, November 2009

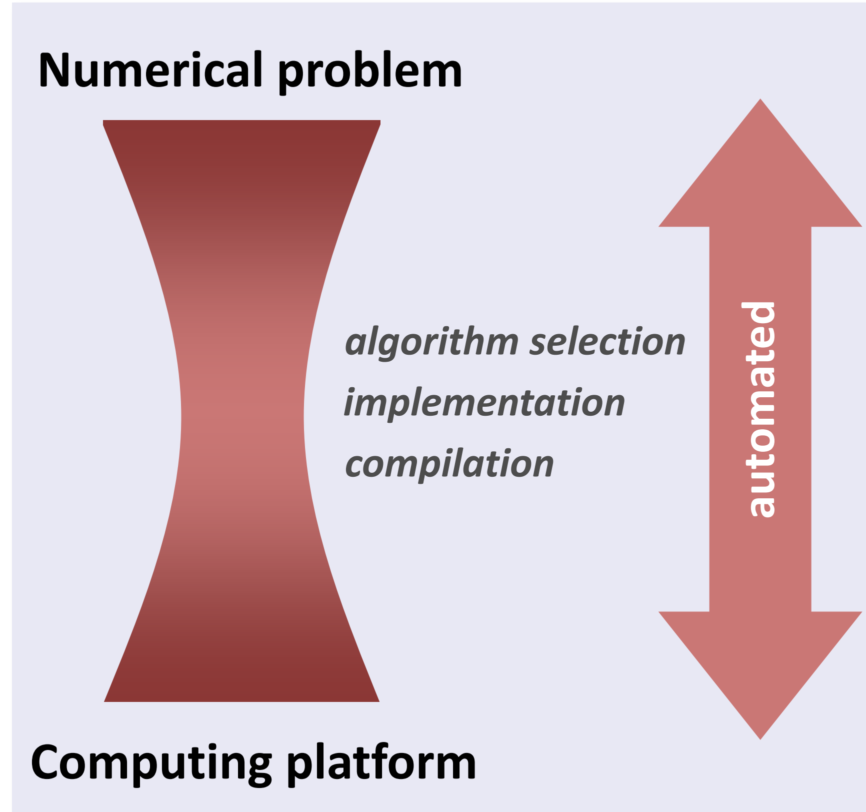
Current



C code a singularity:

- Compiler has no access to high level information
- No structural optimization
- No evaluation of choices

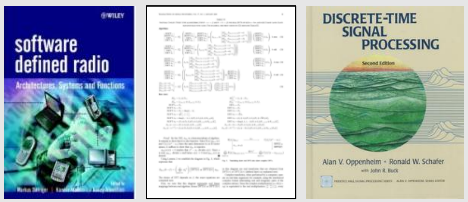
Future



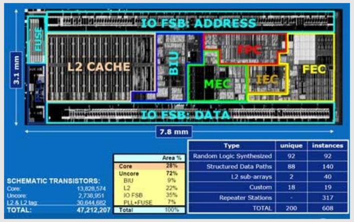
Challenge: conquer the high abstraction level for *complete automation*

Current Solution

Algorithm knowledge



Processor knowledge



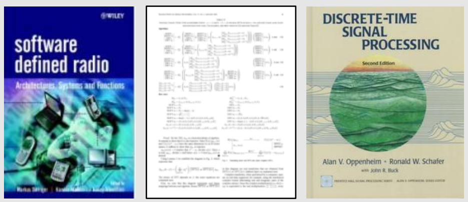
Optimal program

(Repeated for every processor)

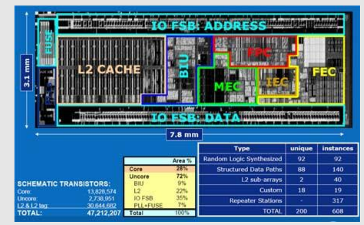
Our Research:

The Computer Writes the Program

Algorithm knowledge



Processor knowledge



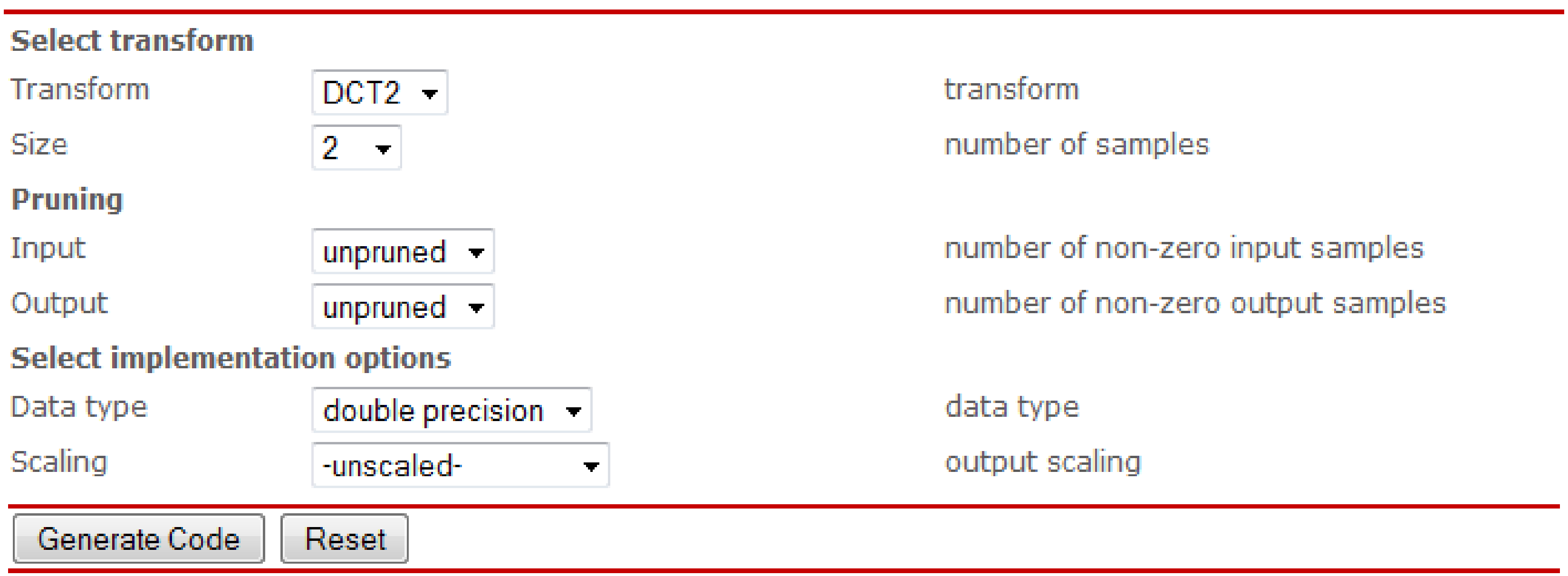
Automation:

Spiral

Optimal program

(Regenerated for every processor)

"Computer Writes the Program"



"click"

- **Computer writes close to optimal code**
- **Vectorized, parallelized, etc.**

Organization

- Spiral: Basic system
- Parallelism
- General input size
- Results
- Final remarks

Linear Transforms

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = y = Tx$$

$$x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

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

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

- *SPL (Signal processing language):* **Mathematical, declarative, point-free**
- **Divide-and-conquer algorithms = breakdown rules in SPL**

Decomposition Rules (>200 for >40 Transforms)

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

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

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

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

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

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

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

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

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

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

$$\mathbf{DCT\text{-}3}_n \to (\mathrm{I}_m \oplus \mathrm{J}_m)\, \mathrm{L}^n_m (\mathbf{DCT\text{-}3}_m(1/4) \oplus \mathbf{DCT\text{-}3}_m(3/4))$$

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

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

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

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

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

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

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

Decomposition rules = Algorithm knowledge in Spiral

(from ≈100 publications)

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];

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

Correct code: ***easy***   *fast code:* ***very difficult***

Program Generation in Spiral

Transform DFT_8

Decomposition rules

Algorithm *(SPL)*

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

parallelization
vectorization

Algorithm *(∑-SPL)*

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

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

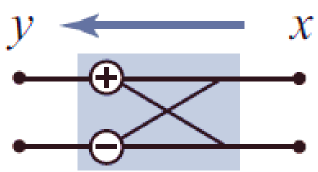
Organization

- Spiral: Basic system
- **Parallelism**
- General input size
- Results
- Final remarks

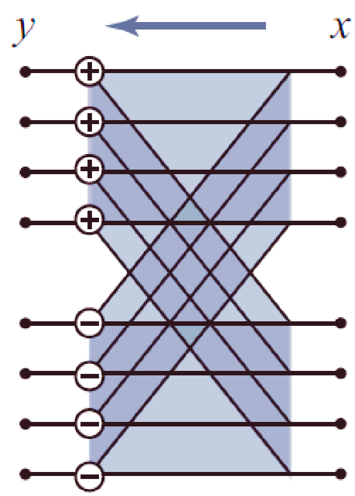
Example: Vectorization in Spiral

- **Relationship SPL expressions ↔ vectorization?**

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



$$y = \left(\mathbf{DFT}_2 \quad \mathrm{I}_4\right) x$$



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:**

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

  $\mathsf{L}_\nu^{\nu^2},\ \mathsf{L}_2^{2\nu}, \mathsf{L}_\nu^{2\nu}$ *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

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

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

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

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

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

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

$\mathrm{I}_l \quad \mathsf{L}^{kmn}$ [illegible] $\mathrm{I}_r \rightarrow \left(\mathrm{I}_l \quad \mathsf{L}^{kmn}\right.$ [illegible] $\left.\mathrm{I}_r\right)\left(\mathrm{I}_l \quad \mathsf{L}^{kmn}\right.$ [illegible] $\left.\mathrm{I}_r\right)$

Manipulation rules = Processor knowledge in Spiral

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

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

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

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

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

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

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

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

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

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

Example

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

$$\ldots$$

$$\ldots$$

$$\ldots$$

$$\rightarrow \left(\mathrm{I}_{\frac{mn}{\nu}} \quad \mathsf{L}^{2\nu}_{\nu}\right)\left(\overline{\mathbf{DFT}_m \quad \mathrm{I}_{\frac{n}{\nu}}} \quad \mathrm{I}_\nu\right)\overline{\mathsf{T}}'^{mn}_{n}$$

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

vectorized arithmetic

vectorized data accesses

Automatically Generate Base Case Library

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

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

- *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));
```

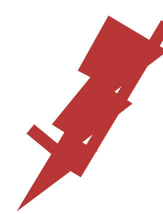
$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{x}_0 \\ \vec{x}_1 \end{bmatrix}$$

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

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

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

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



$\mathrm{L}^4_2 \otimes \mathrm{I}_2$

No base case

Base case

Same Approach for Different Paradigms

Threading:

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

$$\ldots$$

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

$$\ldots$$

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

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

Vectorization:

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

$$\ldots$$

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

$$\ldots$$

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

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

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

GPUs:

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

$$\ldots$$

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

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

Verilog for FPGAs:

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

$$\ldots$$

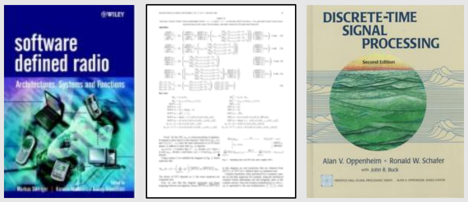
$$\rightarrow \left[\prod_{i=0}^{k-1} \underbrace{\mathrm{L}_r^{r^k}}_{\text{stream}(r^s)} \underbrace{\Big(\mathrm{I}_{r^{k-1}} \otimes \mathbf{DFT}_r\Big)}_{\text{stream}(r^s)} \underbrace{\Big(\mathrm{L}_{r^{k-i-1}}^{r^k}(\mathrm{I}_{r^i} \otimes \mathrm{T}_{r^{k-i-1}}^{r^{k-i}})\,\mathrm{L}_{r^{i+1}}^{r^k}\Big)}_{\text{stream}(r^s)}\right]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

$$\ldots$$

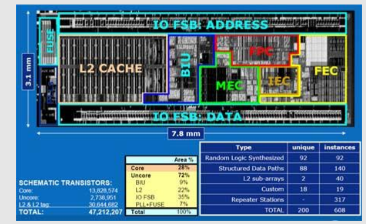
$$\rightarrow \left[\prod_{i=0}^{k-1} \underbrace{\mathrm{L}_r^{r^k}}_{\text{stream}(r^s)} \Big(\mathrm{I}_{r^{k-s-1}} \otimes_s (\mathrm{I}_{r^{s-1}} \otimes \mathbf{DFT}_r)\Big) \underbrace{\mathrm{T}_i'}_{\text{stream}(r^s)}\right] \underbrace{\mathrm{R}_r^{r^k}}_{\text{stream}(r^s)}$$

- Rigorous, correct by construction
- Overcomes compiler limitations

Algorithm knowledge



Processor knowledge



Automation:

Spiral

Optimal program

(Regenerated for every processor)

Decomposition rules

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

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

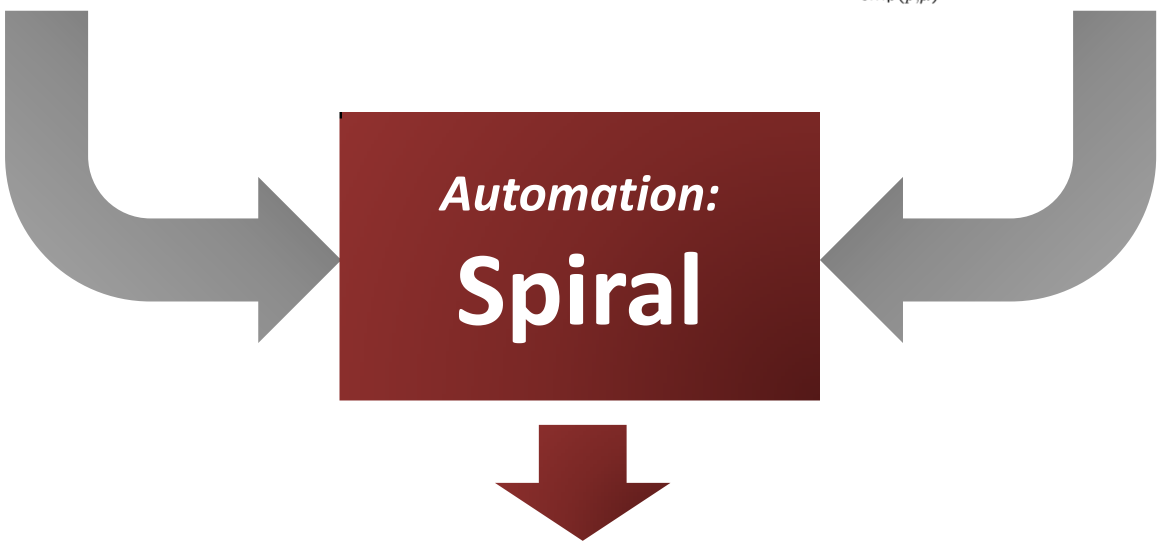
$$\mathbf{RDFT\text{-}3}_n \rightarrow (Q^{\top}_{k/2,m} \otimes I_2)\,(I_k \otimes_i \mathbf{rDFT}_{2m})(i+1/2)/k))\,(\mathbf{RDFT\text{-}3}_k \otimes I_m)$$

Manipulation rules

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

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

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



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

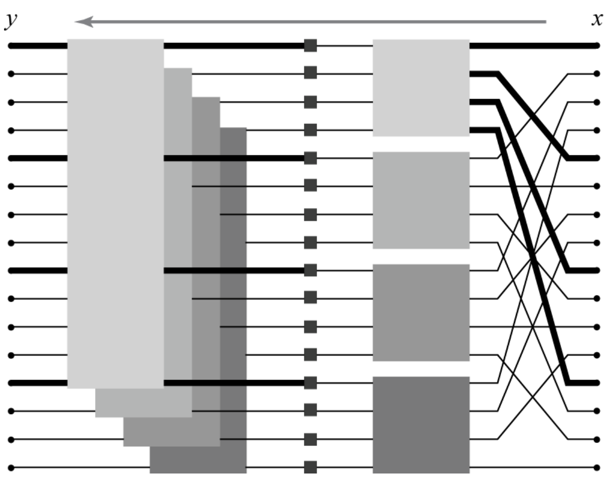
```
DFT(n, x, y) {
  …
  for(i = …) {
    DFT_strided(m, x+mi, y+i, 1, k)
  }
  …
}
```

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

Challenge: Recursion Steps

- **Cooley-Tukey FFT**

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



- **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);
}
```

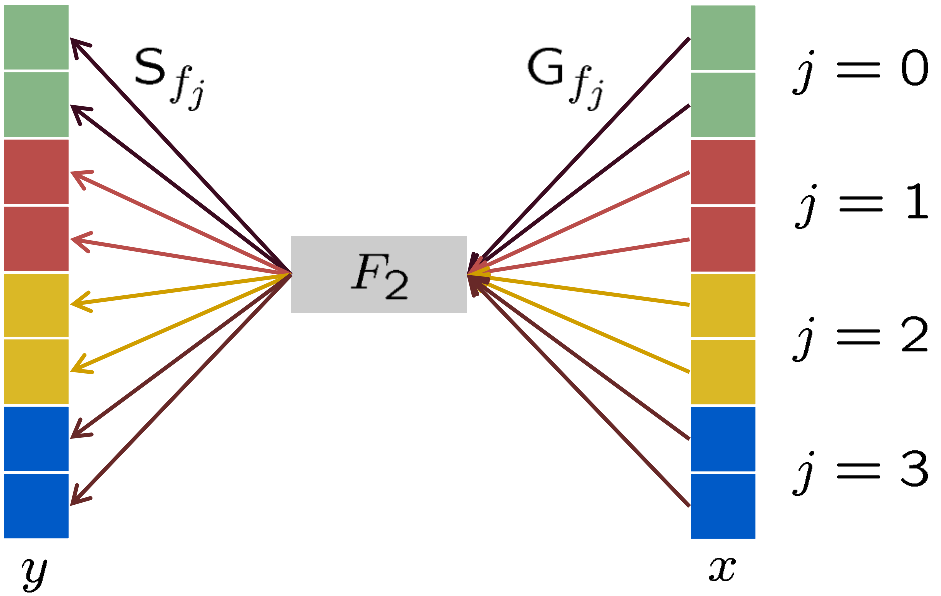
Σ–SPL : Basic Idea

- **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

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

$$\downarrow$$

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

$$\downarrow$$

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

$$\downarrow$$

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

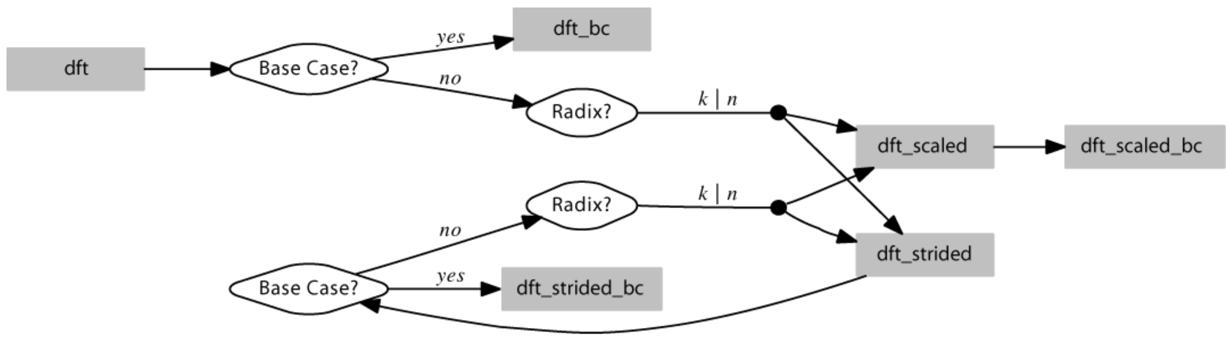
$$\downarrow$$

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

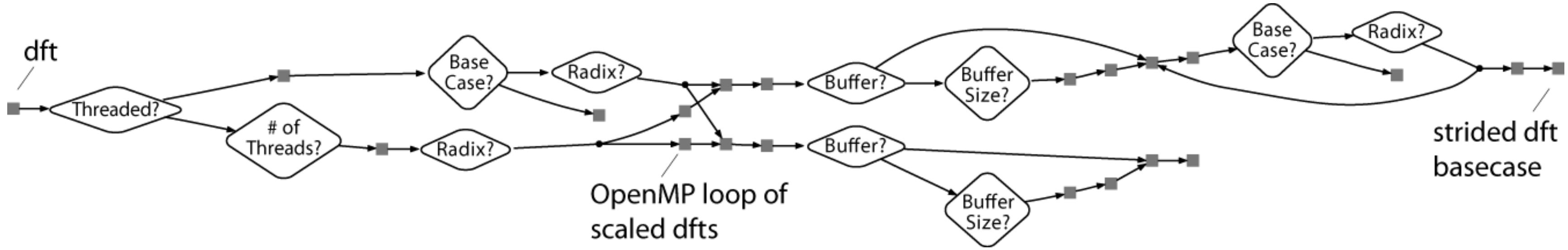
Repeat until closure

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

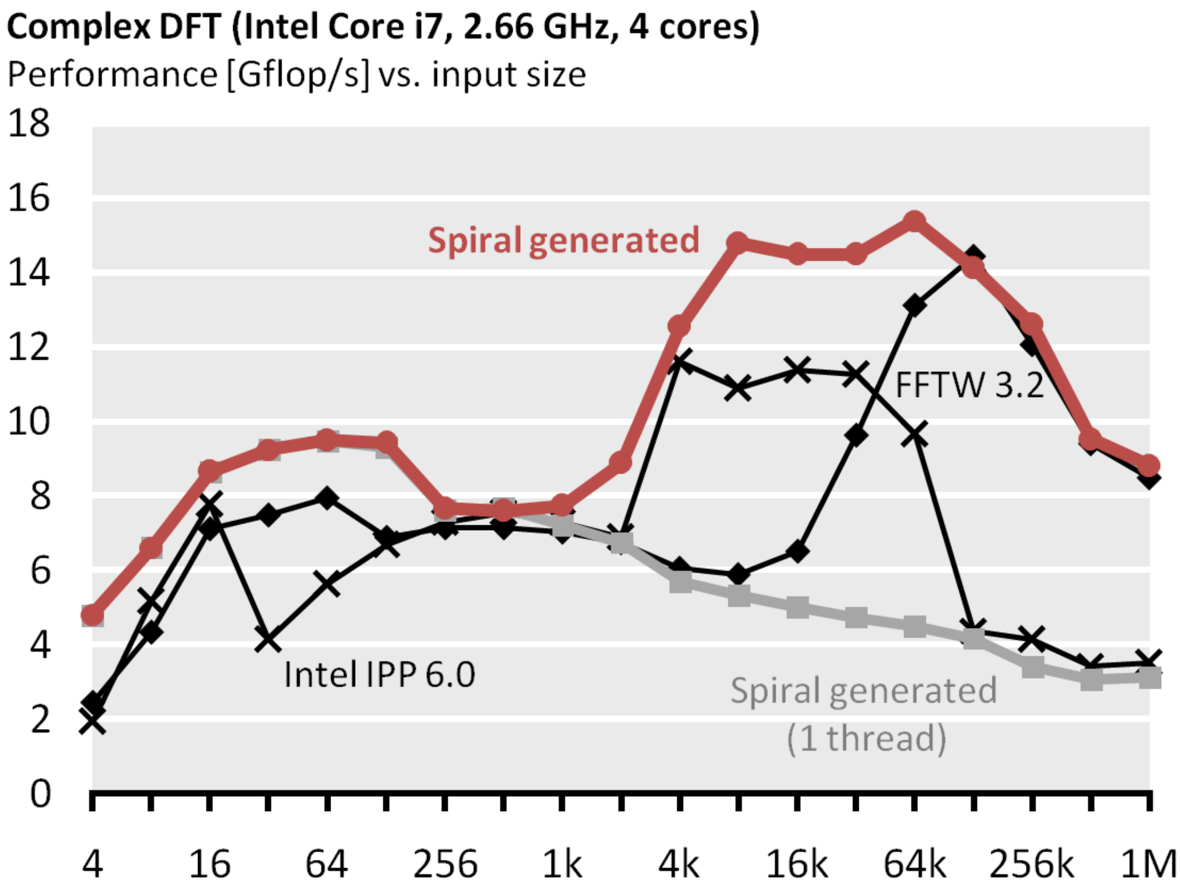
- **Derivation of library structure**
  Rewriting
  Other methods

general input size library

Organization

- Spiral: Basic system
- Parallelism
- General input size
- **Results**
- Final remarks

DFT on Intel Multicore



$$\mathbf{DFT}_n \rightarrow (\mathbf{DFT}_k \otimes I_m) T^n_m (I_k \otimes \mathbf{DFT}_m) L^n_k$$

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

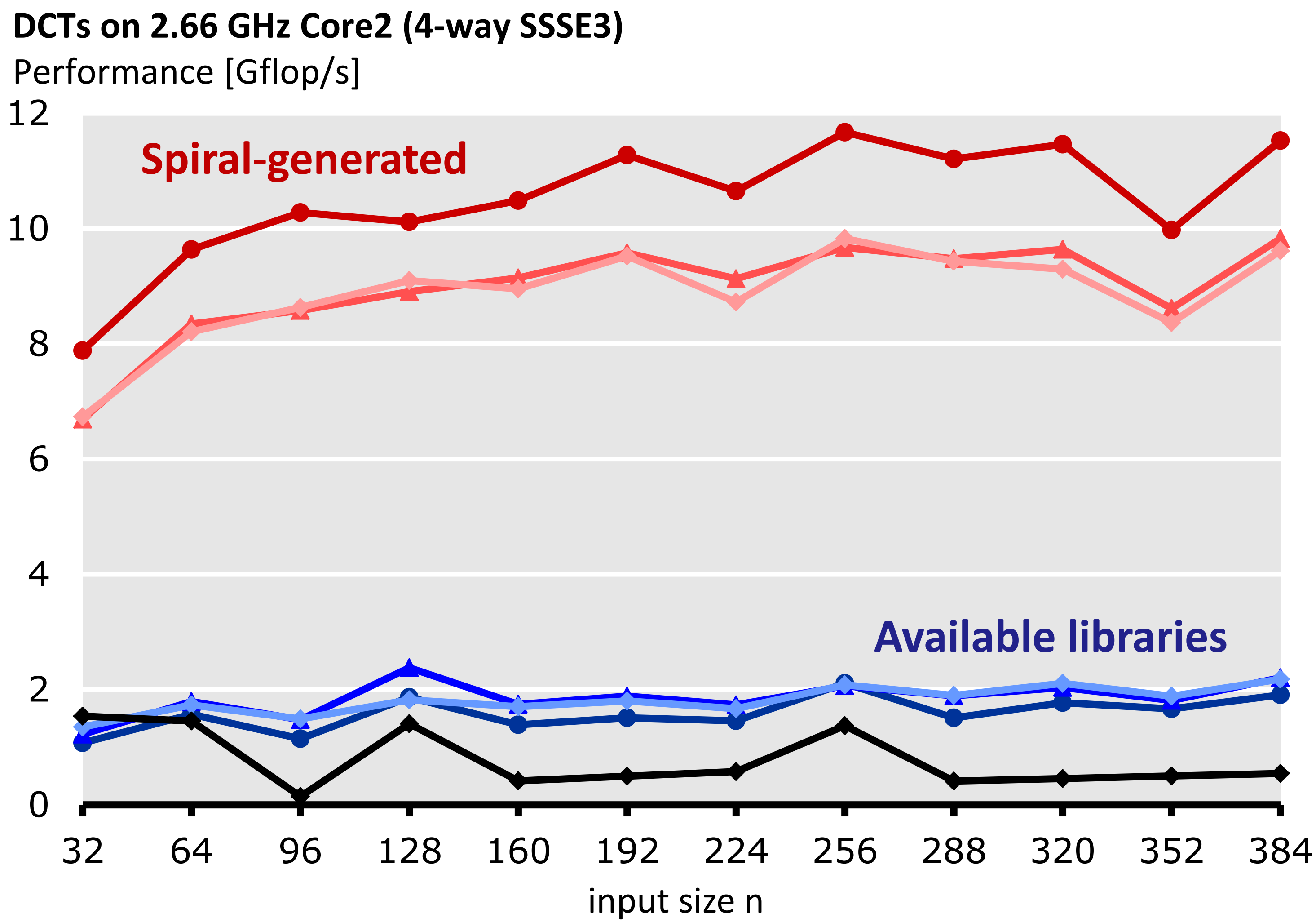
$$\mathbf{RDFT}_n \rightarrow (P^{\top}_{k/2,m} \otimes I_2) \left(\mathbf{RDFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i D_{2m}\, \mathbf{rDFT}_{2m}(i/k)\right)\right) (\mathbf{RDFT}_k \otimes I_m)$$

$$\mathbf{rDFT}_{2n}(u) \rightarrow L^{2n}_m \left(I_k \otimes_i \mathbf{rDFT}_{2m}((i+u)/k)\right) (\mathbf{rDFT}_{2k}(u) \otimes I_m)$$

Spiral

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

Often it Looks Like That



Computer generated Functions for Intel IPP 6.0

3984 C functions

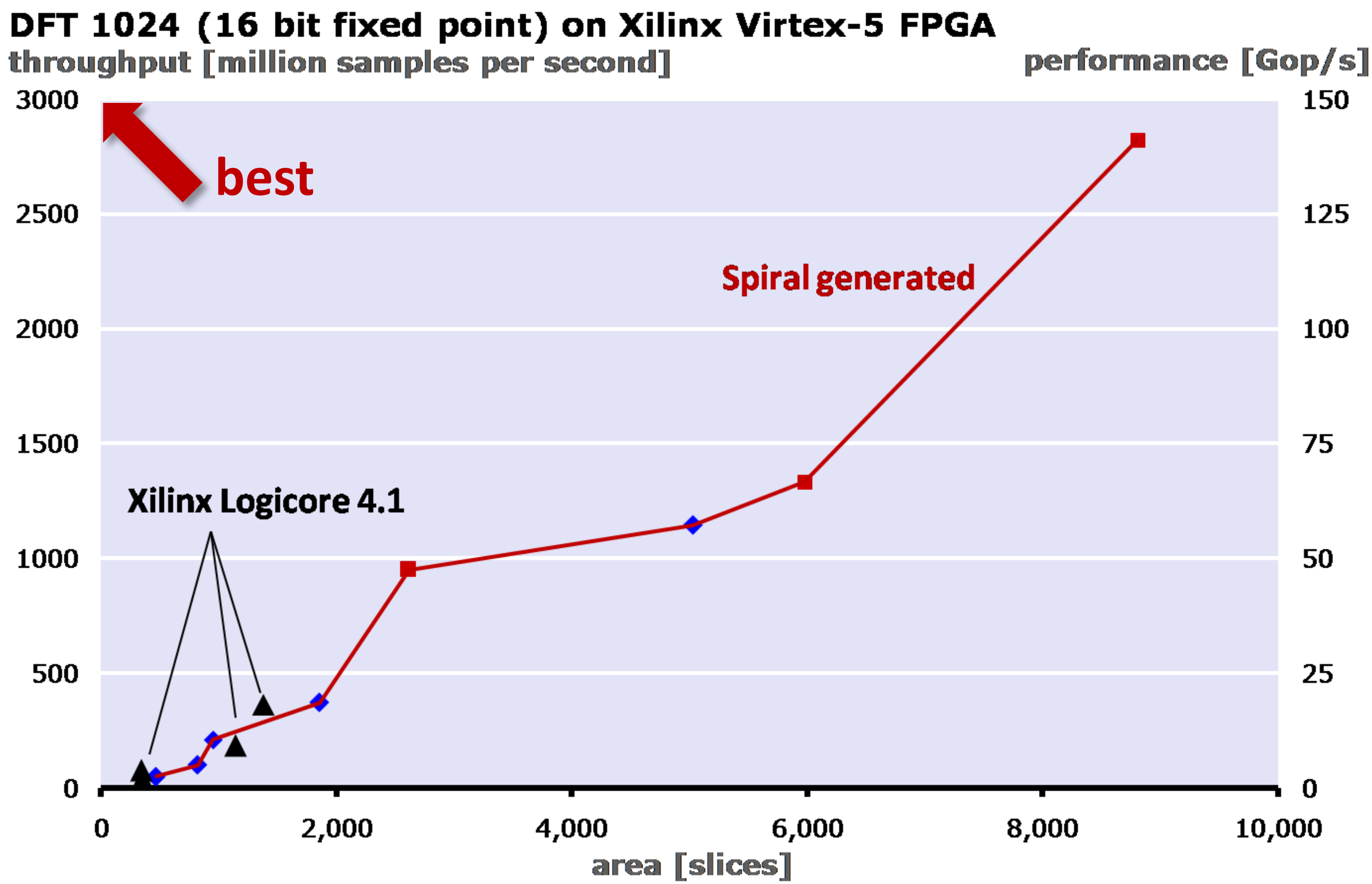
1M lines of code

- *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

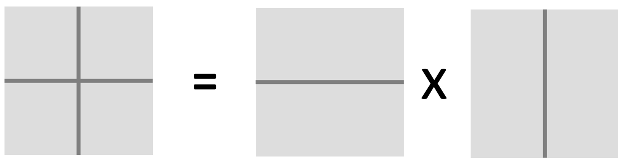


Organization

- Spiral: Basic system
- Parallelism
- General input size
- Results
- **Final remarks**

Beyond Transforms

Matrix-Matrix Multiplication



$$\mathsf{MMM}_{1,1,1} \rightarrow (\cdot)_1$$

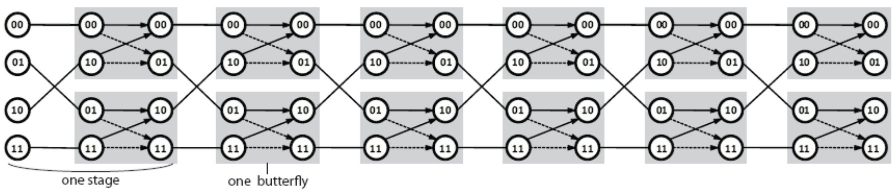
$$\mathsf{MMM}_{m,n,k} \rightarrow (\otimes)_{m/m_b \times 1} \otimes \mathsf{MMM}_{m_b,n,k}$$

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

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

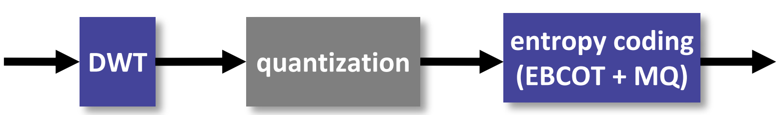
$$\mathsf{MMM}_{m,n,k} \rightarrow (L^{mn/n_b}_{m} \otimes I_{n_b}) \circ ((\otimes)_{1 \times n/n_b} \otimes \mathsf{MMM}_{m,n_b,k}) \circ (I_{km} \times (L^{kn/n_b}_{n/n_b} \otimes I_{n_b}))$$

Viterbi Decoder



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

JPEG 2000 (Wavelet, EBCOT)



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

$$(I \times \mathrm{xor}_2) \circ (T_{SC} \times I) \circ (H \times V \times I) \circ (\underline{L^2_4} \times G_4) \circ (\underline{\begin{pmatrix} 1 \\ 1 \end{pmatrix}} \times \underline{\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}})$$

$$\mathrm{H, V} : (\mathbb{Z}_2^9 \times \mathbb{Z}_2^9) \rightarrow \mathbb{N}$$

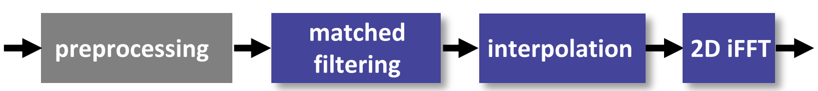
$$\mathrm{H:} \quad h \circ (f \times f) \circ (G_1 \times \mathrm{C}_{-2} \times G_1 \times G_7 \times \mathrm{C}_{-2} \times G_7) \circ \underline{L^2_4} \circ (\underline{\tbinom{1}{1}} \times \underline{\tbinom{1}{1}})$$

$$\mathrm{V:} \quad h \circ (f \times f) \circ (G_3 \times \mathrm{C}_{-2} \times G_3 \times G_5 \times \mathrm{C}_{-2} \times G_5) \circ \underline{L^2_4} \circ (\underline{\tbinom{1}{1}} \times \underline{\tbinom{1}{1}})$$

$$f : \mathrm{mul}_2 \circ (I \times \mathrm{sub}_2) \circ (I \times \mathrm{C}_1 \times \mathrm{mul}_2)$$

$$h : \mathrm{min}_2 \circ (\mathrm{C}_1 \times \mathrm{max}_2) \circ (\mathrm{C}_{-1} \times \mathrm{sum}_2)$$

Synthetic Aperture Radar (SAR)



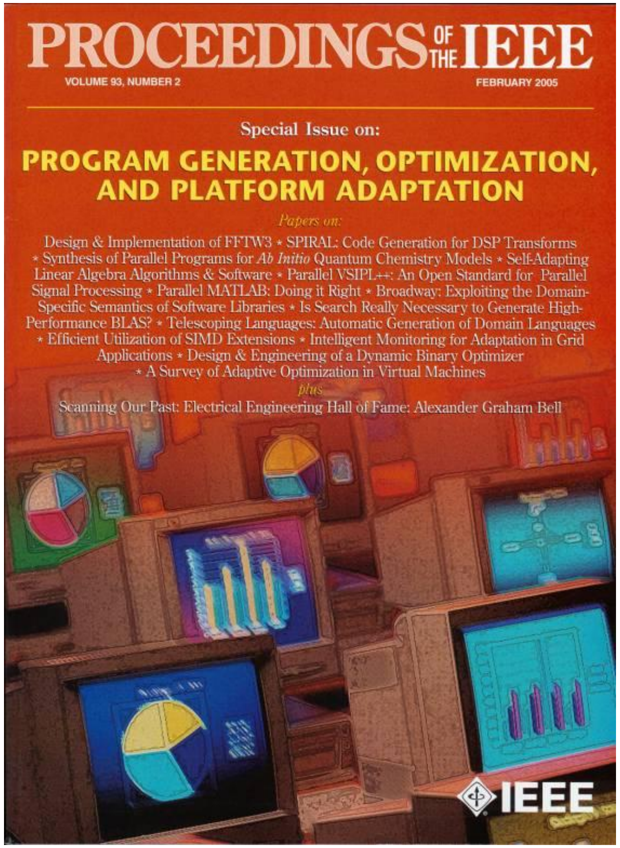
$$\begin{aligned}
\mathrm{SAR} &\rightarrow \mathrm{2D\text{-}iDFT} \circ \mathrm{Interpl} \circ \mathrm{MatchFilt} \circ \mathrm{prep} \\
\mathrm{2D\text{-}iDFT} &\rightarrow \mathrm{iDFT} \otimes \mathrm{iDFT} \\
\mathrm{MatchFilt} &\rightarrow \mathrm{Filt} \circ (\mathrm{I} \times \mathrm{C}_f) \\
\mathrm{Filt} &\rightarrow (\mathrm{I} \otimes (\cdot)) \\
\mathrm{Interpl} &\rightarrow (\Sigma \otimes \mathrm{I}) \circ (\mathrm{I} \otimes_j \mathrm{S}_{x_j \otimes y_j}) \circ \mathrm{Filt} \circ ((\mathrm{I} \otimes \mathbf{1} \otimes \mathrm{I}) \times \mathrm{I}) \circ (\mathrm{I} \times \mathrm{C}_{\imath \otimes_j g_j})
\end{aligned}$$

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

$$\mathrm{DFT}_{64}$$

```
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>
}
```

- **Key ideas:**

  Algorithm knowledge:

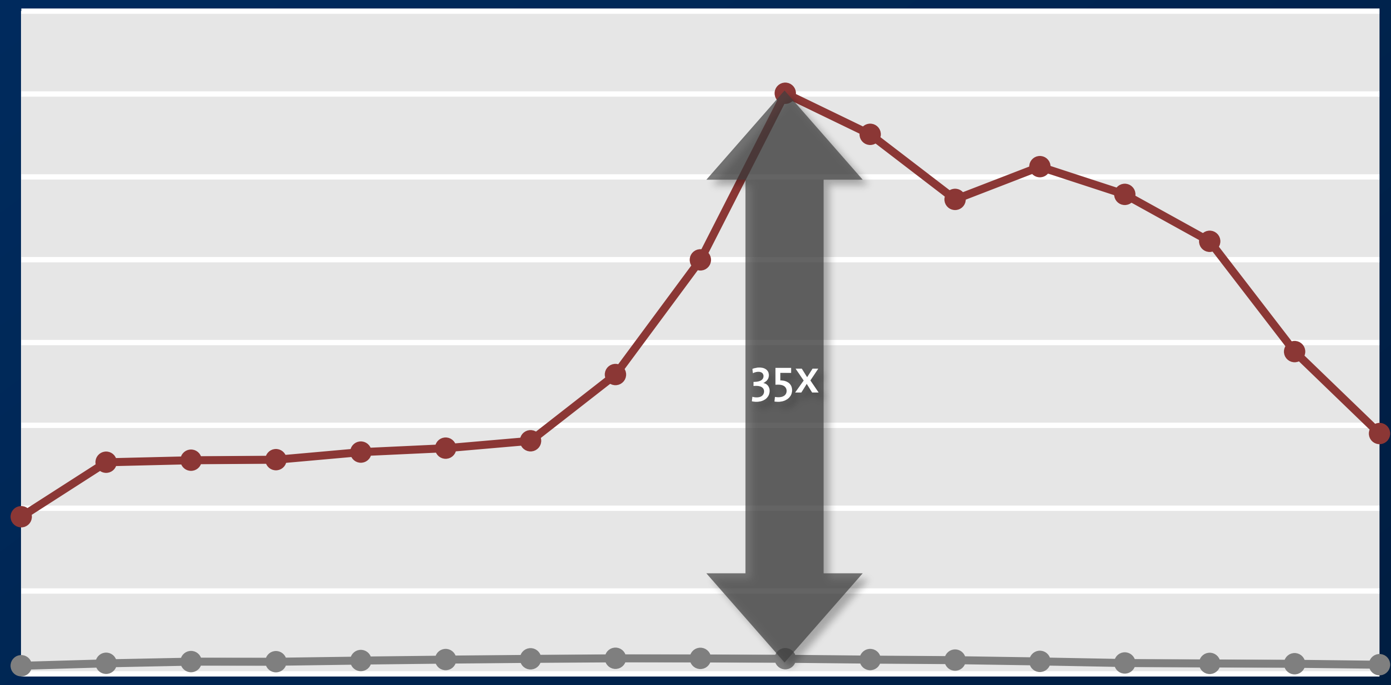
  Domain specific symbolic representation

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

  Platform knowledge:

  Tagged rewrite rules, SIMD specification

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


35x

Compilers

Programming languages
Program generation

Symbolic Computation
Rewriting

Automating High-Performance Numerical Software Development

Search
Learning

Algorithms
Mathematics

Software
Scientific Computing