Spiral:
Automatic Generation of Industry Strength Performance Libraries

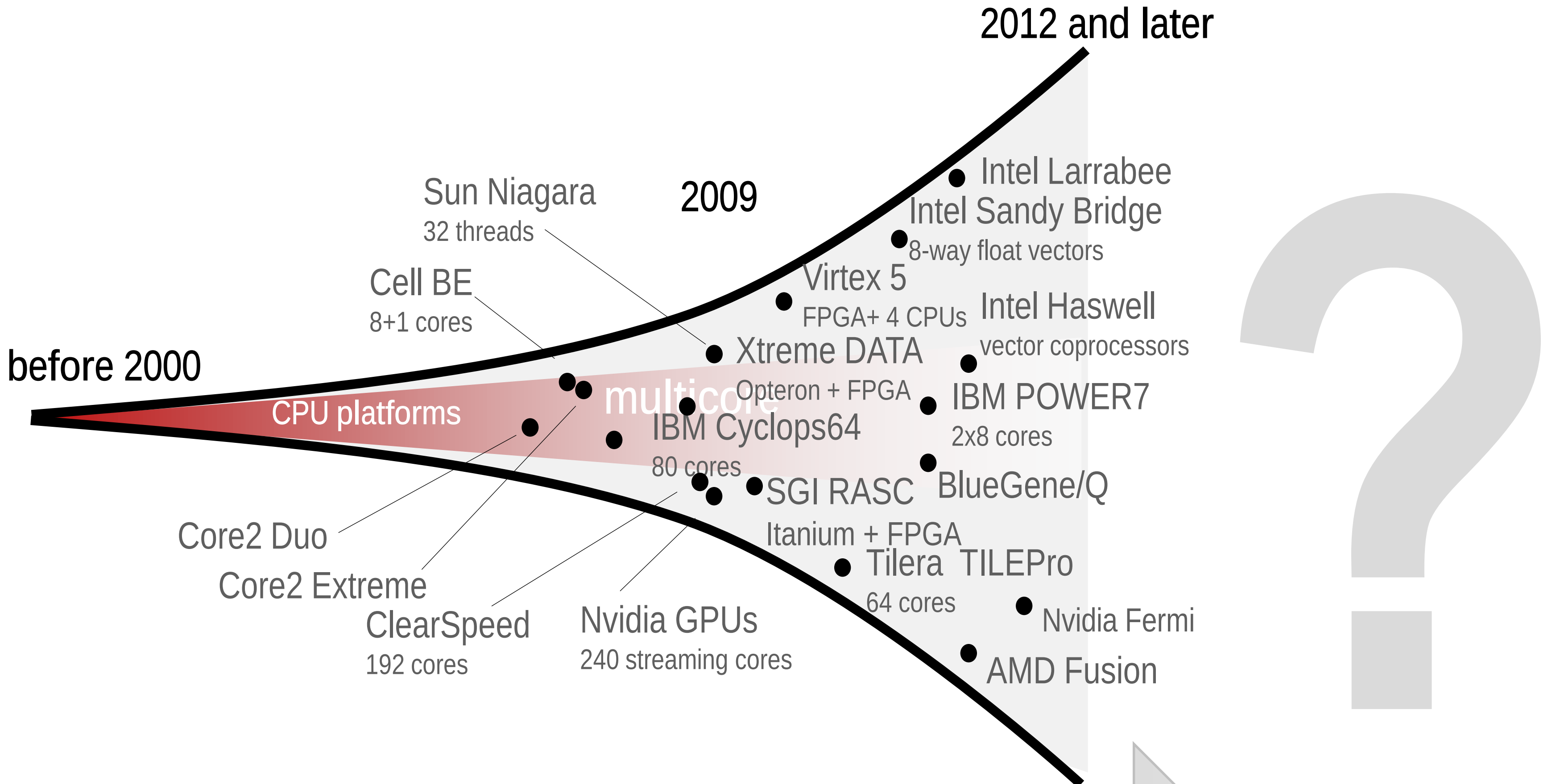
Franz Franchetti

Carnegie Mellon University
www.ece.cmu.edu/~franzf

CTO and Co-Founder, SpiralGen
www.spiralgen.com

This work was supported by
DARPA DESA program, NSF, ONR, Mercury Inc., Intel, and Nvidia

The Future is Parallel and Heterogeneous



Programmability?
Performance portability?
Rapid prototyping?

Spiral: Computer Writes Best SAR Code

Result

Same performance, 1/10th human effort, non-expert user

Key ideas

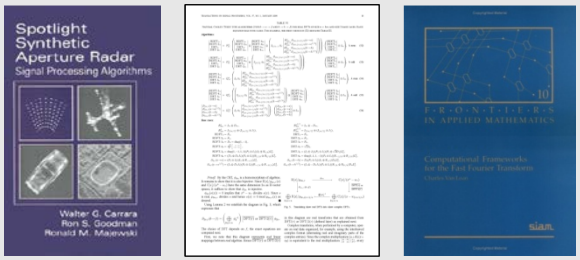
restrict domain, use mathematics, program synthesis

[1] Rudin, J., **Implementation of Polar Format SAR Image Formation on the IBM Cell Broadband Engine**, in Proceedings High Performance Embedded Computing (HPEC), 2007. ***Best Paper Award***.

[2] 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.

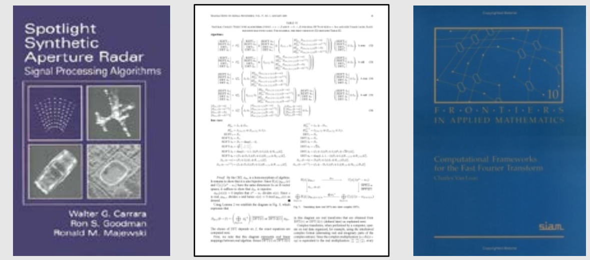
What is Spiral?

Traditionally



High performance library optimized for given platform

Spiral Approach



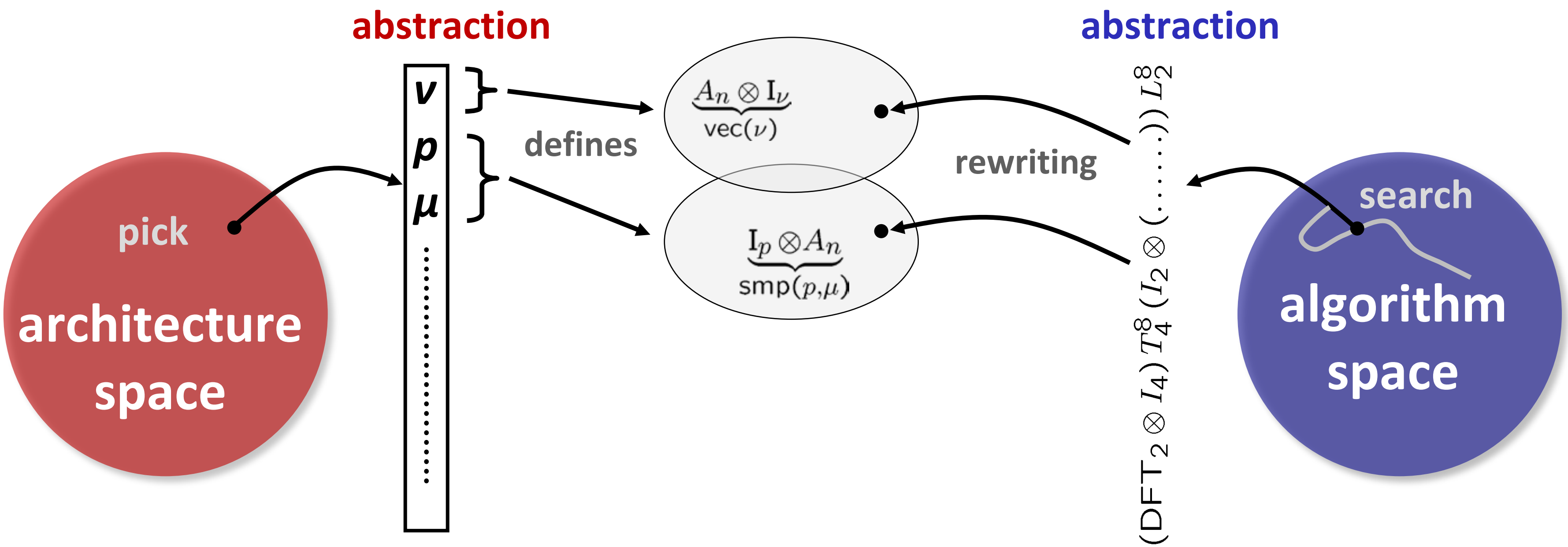
Spiral

High performance library optimized for given platform

Comparable performance

Spiral's Domain-Specific Program Synthesis

Model: common abstraction
= spaces of matching formulas



Architectural parameter:
Vector length,
#processors, …

optimization →

Kernel:
problem size,
algorithm choice

Related Work

Synthesis from Domain Math

- **Spiral**
  Signal and image processing, SDR
- **Tensor Contraction Engine**
  Quantum Chemistry Code Synthesizer
- **FLAME**
  Numerical linear algebra (LAPACK)

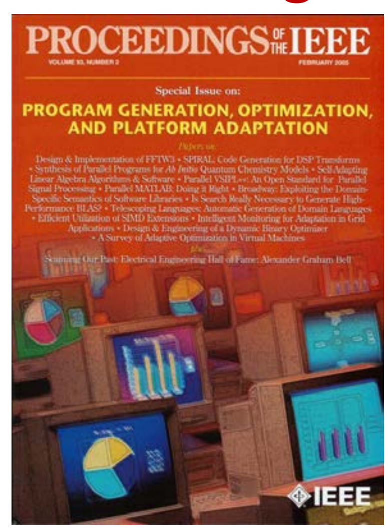
Compiler-Based Autotuning

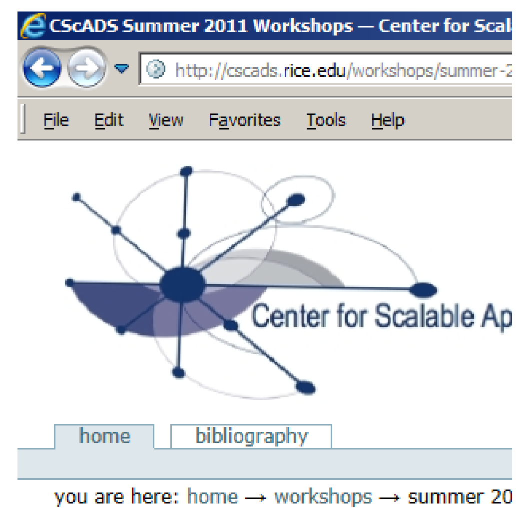
- **Polyhedral framework**
  IBM XL, Pluto, CHiLL
- **Transformation prescription**
  CHiLL, POET
- **Profile guided optimization**
  Intel C, IBM XL

Autotuning Numerical Libraries

- **ATLAS**
  BLAS generator
- **FFTW**
  kernel generator
- **Vendor math libraries**
  Code generation scripts

Autotuning Primer





Organization

- **Spiral overview**
- Validation and Verification
- Results
- Concluding remarks

M. Püschel, F. Franchetti, Y. Voronenko: **Spiral.** Encyclopedia of Parallel Computing, D. A. Padua (Editor), 2011.

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.

Spiral's Origin: Linear Transforms

- **Transform = Matrix-vector multiplication**
  Example: Discrete Fourier transform (DFT)

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

input vector (signal) → x

output vector (signal) → y

transform = matrix → T

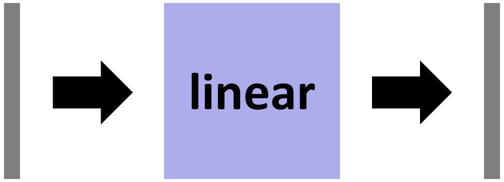
- **Fast algorithm = sparse matrix factorization = SPL formula**
  Example: Cooley-Tukey FFT algorithm

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

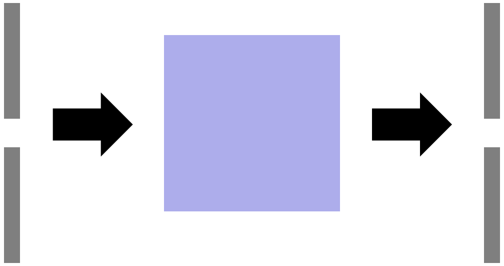
$$\mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathrm{I}_2)\,\mathsf{T}^4_2\,(\mathrm{I}_2 \otimes \mathrm{DFT}_2)\,\mathsf{L}^4_2$$

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



Expressing Kernels as Operator Formulas

Linear Transforms

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

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

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

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

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

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

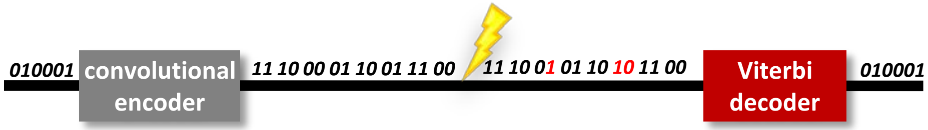
$$\mathbf{WHT}_{2^k} \rightarrow \prod_{i=1}^{t} (\mathrm{I}_{2^{k_1+\cdots+k_{i-1}}} \otimes \mathbf{WHT}_{2^{k_i}} \otimes \mathrm{I}_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t$$

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

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

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

Software Defined Radio



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

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

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

Matrix-Matrix Multiplication

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

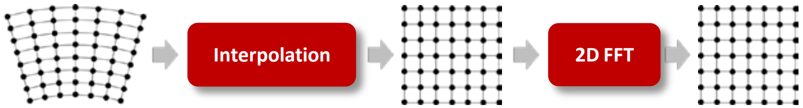
$$\mathrm{MMM}_{m,n,k} \rightarrow (\otimes)_{m/m_b \times 1} \otimes \mathrm{MMM}_{m_b,n,k}$$

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

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

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

Synthetic Aperture Radar (SAR)



$$\mathrm{SAR}_{k \times m \rightarrow n \times n} \rightarrow \mathrm{DFT}_{n \times n} \circ \mathrm{Interp}_{k \times m \rightarrow n \times n}$$

$$\mathrm{DFT}_{n \times n} \rightarrow (\mathrm{DFT}_n \otimes \mathrm{I}_n) \circ (\mathrm{I}_n \otimes \mathrm{DFT}_n)$$

$$\mathrm{Interp}_{k \times m \rightarrow n \times n} \rightarrow (\mathrm{Interp}_{k \rightarrow n} \otimes_i \mathrm{I}_n) \circ (\mathrm{I}_k \otimes_i \mathrm{Interp}_{m \rightarrow n})$$

$$\mathrm{Interp}_{r \rightarrow s} \rightarrow \left(\bigoplus_{i=0}^{n-2} \mathrm{InterpSeg}_k \right) \oplus \mathrm{InterpSegPruned}_{k,\ell}$$

$$\mathrm{InterpSeg}_k \rightarrow \mathrm{G}^{u \cdot n \rightarrow k}_f \circ \mathrm{iPrunedDFT}_{n \rightarrow u \cdot n} \circ \left(\frac{1}{n} \right) \circ \mathrm{DFT}_n$$

One Approach for all Types of Parallelism

- **Multithreading** (Multicore) $\quad \mathrm{I}_p \otimes_{\|} A_{\mu n}, \quad \mathbf{L}^{mn}_m \bar{\otimes} \mathrm{I}_\mu$
- **Vector SIMD** (SSE, VMX/Altivec,…) $\quad A \hat{\otimes} \mathrm{I}_\nu \quad \underbrace{\mathbf{L}^{2\nu}_2}_{\text{isa}}, \quad \underbrace{\mathbf{L}^{2\nu}_\nu}_{\text{isa}}, \quad \underbrace{\mathbf{L}^{\nu^2}_\nu}_{\text{isa}}$
- **Message Passing** (Clusters, MPP) $\quad \mathrm{I}_p \otimes_{\|} A_n, \quad \underbrace{\mathbf{L}^{p^2}_p \bar{\otimes} \mathrm{I}_{n/p^2}}_{\text{all-to-all}}$
- **Streaming/multibuffering** (Cell) $\quad \mathrm{I}_n \otimes_2 A_{\mu n}, \quad \mathbf{L}^{mn}_m \bar{\otimes} \mathrm{I}_\mu$
- **Graphics Processors** (GPUs) $\quad \prod_{i=0}^{n-1} A_i, \quad A_n \overset{\frown}{\otimes} \mathrm{I}_w, \quad P_n \otimes Q_w$
- **Gate-level parallelism** (FPGA) $\quad \prod_{i=0}^{n-1}{}^{\text{ir}} A, \quad \mathrm{I}_s \tilde{\otimes} A, \quad \underbrace{\mathbf{L}^m_n}_{\text{bram}}$
- **HW/SW partitioning** (CPU + FPGA) $\quad \underbrace{A_1}_{\text{fpga}}, \quad \underbrace{A_2}_{\text{fpga}}, \quad \underbrace{A_3}_{\text{fpga}}, \quad \underbrace{A_4}_{\text{fpga}}$

Autotuning in Constraint Solution Space

Intel MIC

DFT_{256}

Base cases

$I_4 \otimes_\parallel A_{16n}$

$L^{mn}_m \bar{\otimes} I_{16}$

$A \hat{\otimes} I_4$

$\underbrace{L^8_2}_{SSE}, \underbrace{L^8_4}_{SSE}, \underbrace{L^{16}_4}_{SSE}, \underbrace{L^4_2 \otimes I_2}_{SSE}$

Transformation rules

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

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

$$\underbrace{I_m \otimes A_n}_{smp(p,\mu)} \to I_p \otimes_\parallel \left(I_{m/p} \otimes A_n\right)$$

...

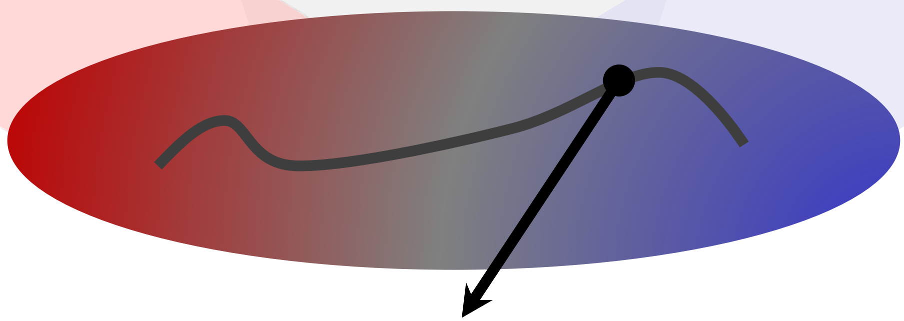
Breakdown rules

$$DFT_n \to (DFT_k \otimes I_m)\, T^n_m \cdot (I_k \otimes DFT_m)\, L^n_k$$

$$DFT_n \to P_n (DFT_k \otimes DFT_m) Q_n$$

$$DFT_p \to R^T_p (I_1 \oplus DFT_{p-1}) D_p \cdot (I_1 \oplus DFT_{p-1}) R_p$$

$$DFT_2 \to \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



$$\left((L^{mp}_m \otimes I_{n/p\mu}) \bar{\otimes} I_\mu\right)\left(I_p \otimes_\parallel (DFT_m \otimes I_{n/p})\right)\left((L^{mp}_p \otimes I_{n/p\mu}) \bar{\otimes} I_\mu\right)\left(\bigoplus_{i=0}^{p-1}{}_\parallel\, T^{mn,i}_n\right)\left(I_p \otimes_\parallel (I_{m/p} \otimes DFT_n)\right)\left(I_p \otimes_\parallel L^{mn/p}_{m/p}\right)\left((L^{pn}_p \otimes I_{m/p\mu}) \bar{\otimes} I_\mu\right)$$

Translating a Formula into Code

Constraint Solver Input: $\underbrace{\mathrm{DFT}_8}_{\text{double}}$

Output =

OL Formula: $(\mathrm{DFT}_2 \otimes I_4)\, T^8_4 \left(I_2 \otimes \left((\mathrm{DFT}_2 \otimes I_2)\, T^4_2\, (I_2 \otimes \mathrm{DFT}_2)\, L^4_2\right)\right) L^8_2$

Σ-OL: $\sum_{j=0}^{3} \left(S_j\, \mathrm{DFT}_2\, G_j\right) \sum_{k=0}^{1} \left(\sum_{l=0}^{1} \left(S_{k,l}\, \mathrm{diag}\left(\mathrm{t}_{k,l}\right) \mathrm{DFT}_2\, G_l\right) \sum_{m=0}^{1} \left(S_m\, \mathrm{diag}\left(\mathrm{t}_m\right) \mathrm{DFT}_2\, G_{k,m}\right)\right)$

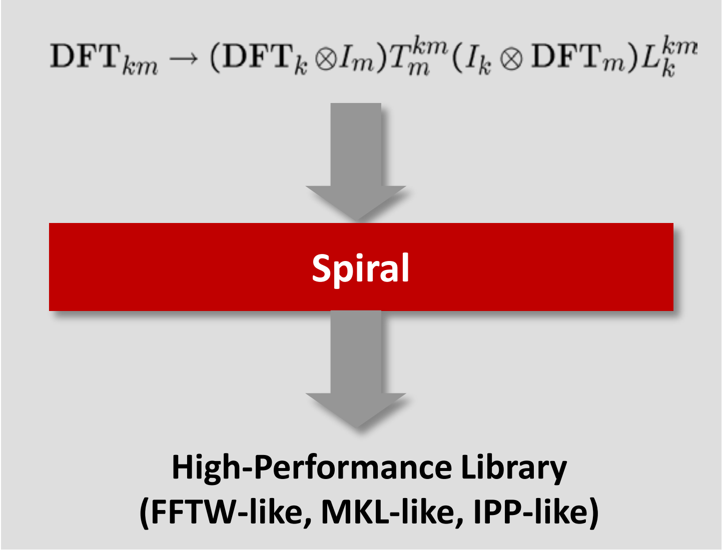
C Code:

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

Auto-Generation of Performance Library

Input:

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



Output:

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

Core Idea: Recursion Step Closure

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

- **Algorithm:**

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

1. Apply the breakdown rule

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

2. Convert to Σ-SPL

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

3. Apply loop merging + index simplification rules.

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

4. Extract recursion steps

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

5. Repeat until closure is reached

Spiral-Generated Code (Intel MIC/LRBni)

```
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_1to16_ps(0.70710678118654757),0x5555,a2154,U926),
        _mm512_swizupconv_r32(t1341,_MM_SWIZ_REG_CDAB));
    U927 = _mm512_swizupconv_r32(_mm512_set_16to16_ps(0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757)),_MM_SWIZ_REG_CDAB);
    ...
    s1166 = _mm512_madd231_ps(_mm512_mul_ps(_mm512_mask_or_pi(_mm512_set_16to16_ps(
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757),
        0.70710678118654757, (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757)),
        0xAAAA,a2154,U951),t1362),
        _mm512_mask_sub_ps(_mm512_set_16to16_ps(0.70710678118654757,
        (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757), 0.70710678118654757,
        (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757), 0.70710678118654757,
        (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757), 0.70710678118654757,
        (-0.70710678118654757), 0.70710678118654757, (-0.70710678118654757)),0x5555,a2154,U951),
        _mm512_swizupconv_r32(t1362,_MM_SWIZ_REG_CDAB));
    ...
}
```

Support For Library-Specific Interfaces

Complex FFT

name / data type / size / scaling

```
IPPAPI(IppStatus, ippgDFTFwd_CToC_32fc,
     (const Ipp32fc *pSrc, Ipp32fc *pDst, int length, int flag) )
```

Walsh-Hadamard Transform

log(size) / scaling / memory

```
IPPAPI(IppStatus, ippgWHT_32f,
     (const Ipp32f *pSrc, Ipp32f *pDst, int order, int flag, Ipp8u *pBuf)
```

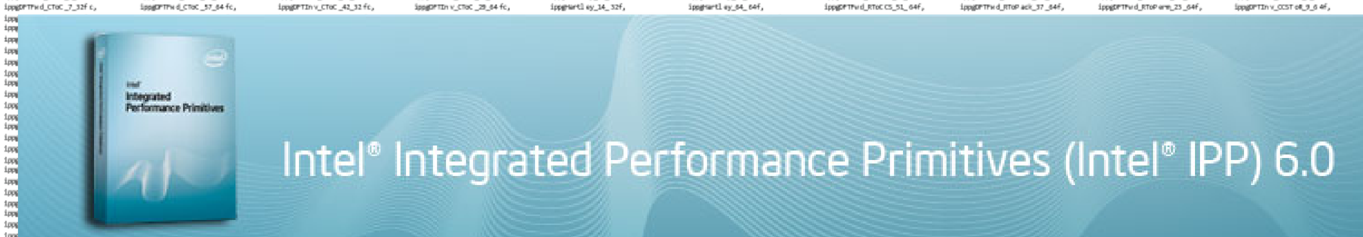
```
IPPAPI(IppStatus, ippgWHTGetBufferSize_32f,
     (int order, Ipp32u *pBufferSize) )
```

memory

Industry-Strength Code: Spiral and Intel IPP 6.0

Spiral-generated code in Intel's Library IPP

- IPP = Intel's performance primitives, used by 1000s of companies
- Generated: 3984 C functions (signal processing) = 1M lines of code
- Full parallelism support
- Computer-generated code: Faster than what was achievable by hand



Organization

- Spiral overview
- **Validation and Verification**
- Results
- Concluding remarks

Symbolic Verification

- **Transform = Matrix-vector multiplication**
  matrix fully defines the operation

$$\mathrm{DFT}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

= ?

- **Algorithm = Formula**
  represents a matrix expression, can be evaluated to a matrix

$$(\mathrm{DFT}_2 \otimes \mathrm{I}_2)\, \mathrm{T}^4_2 (\mathrm{I}_2 \otimes \mathrm{DFT}_2)\, \mathrm{L}^4_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Empirical Verification

- **Run program on all basis vectors, compare to columns of transform matrix**

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

= ?

```
DFT4([0,1,0,0])
```

- **Compare program output on random vectors to output of a random implementation of same kernel**

```
DFT4([0.1,1.77,2.28,-55.3])
```

= ?

```
DFT4_rnd([0.1,1.77,2.28,-55.3]))
```

Verification of the Generator

- **Rule replaces left-hand side by right-hand side when preconditions match**

$$\mathrm{I}_m \otimes A_n \rightarrow \mathrm{L}^{mn}_m (A_n \otimes \mathrm{I}_m)\, \mathrm{L}^{mn}_n$$

- **Test rule by evaluating expressions before and after rule application and compare result**

$$\mathrm{I}_2 \otimes \mathrm{DFT}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

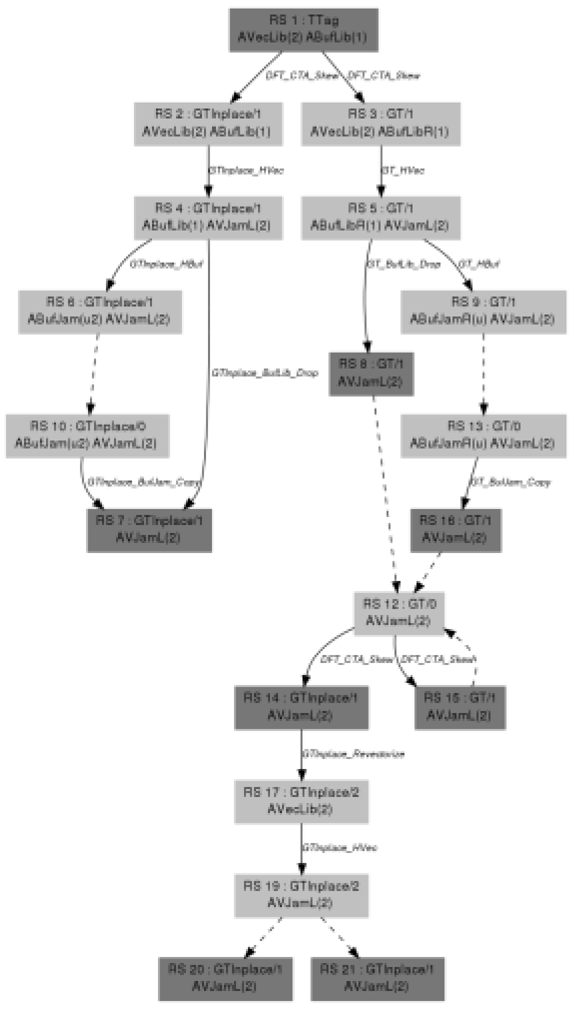
= ?

$$\mathrm{L}^4_2 (\mathrm{DFT}_2 \otimes \mathrm{I}_2)\, \mathrm{L}^4_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Verification of Autotuning Libraries

Auto-generated FFTW-like library

- Need verifier for each function
- Auto-generated from specification
- Auto-generate test harness
- Drop-in replacement into existing infrastructure

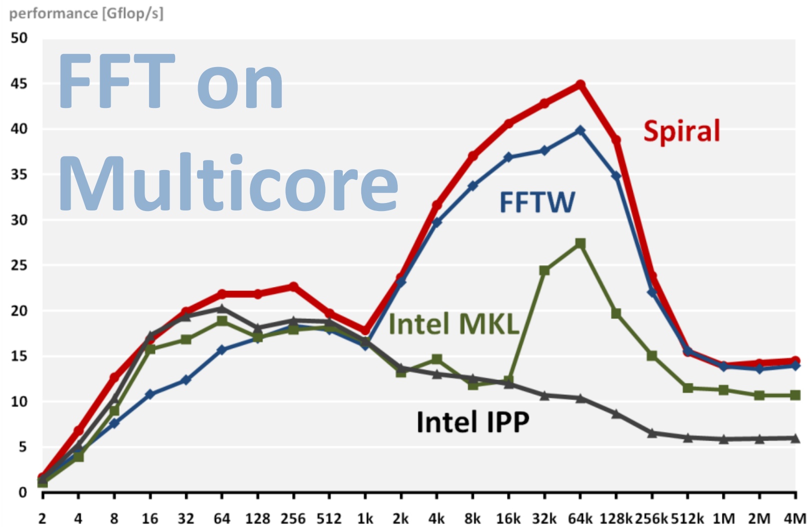


Organization

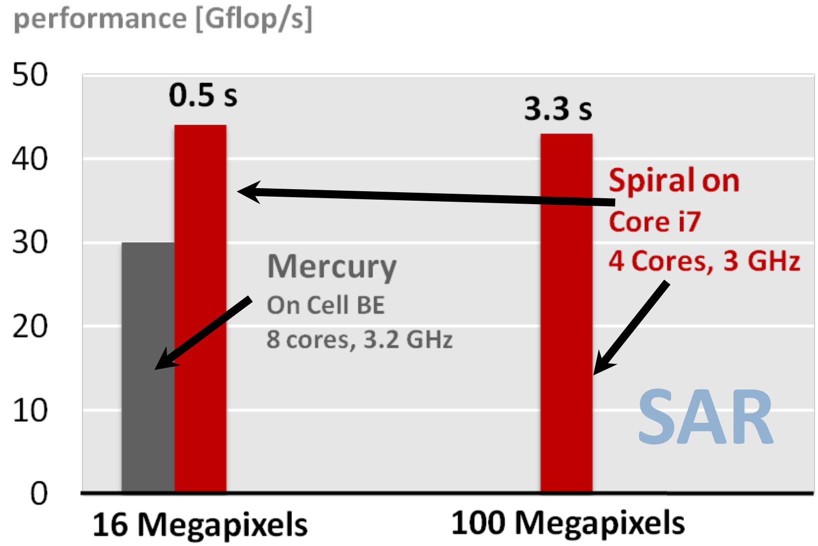
- Spiral overview
- Validation and Verification
- **Results**
- Concluding remarks

Results: Spiral Outperforms Humans

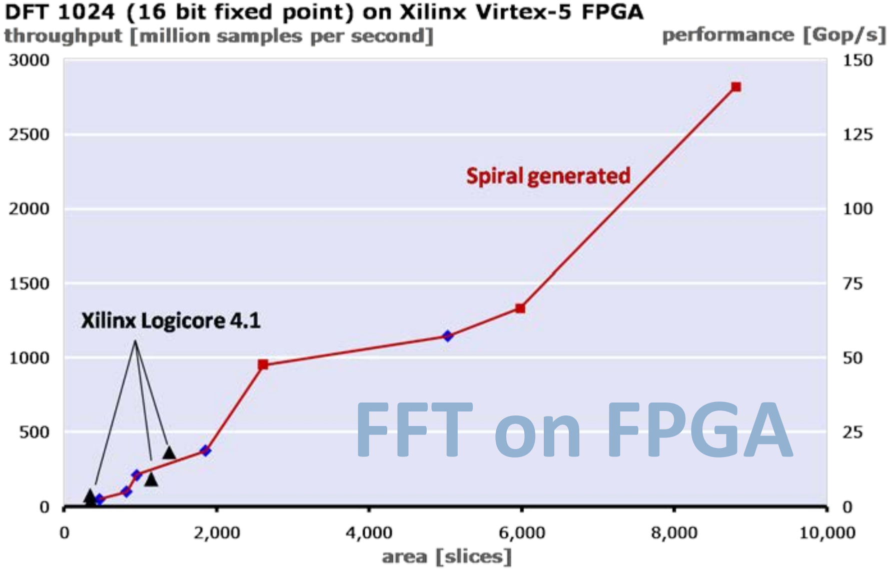
1D DFT on 3.3 GHz Sandy Bridge (4 Cores, AVX)



SAR Image Formation on Intel platforms

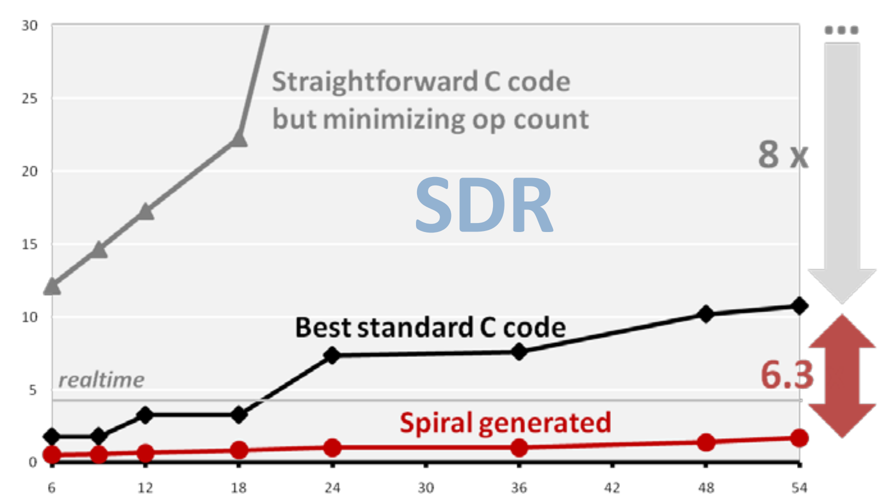


DFT 1024 (16 bit fixed point) on Xilinx Virtex-5 FPGA



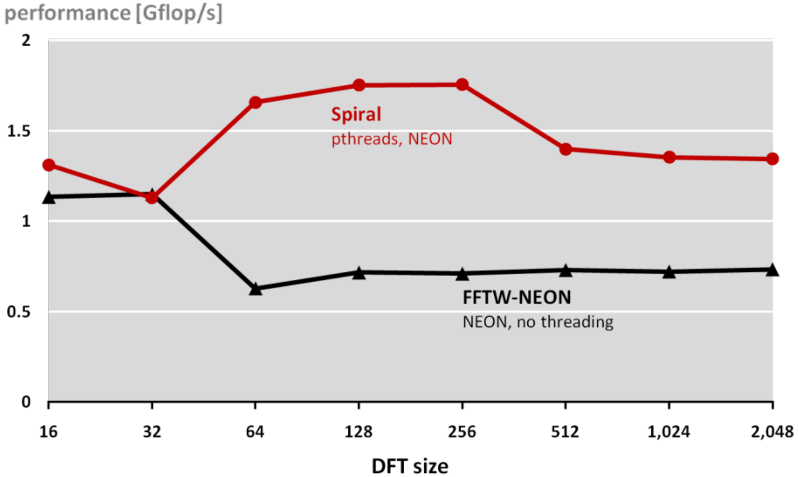
WiFi transmitter on Dualcore Intel Atom

Run time per OFDM symbol [micro seconds] vs. Data rate [Mbit/s]

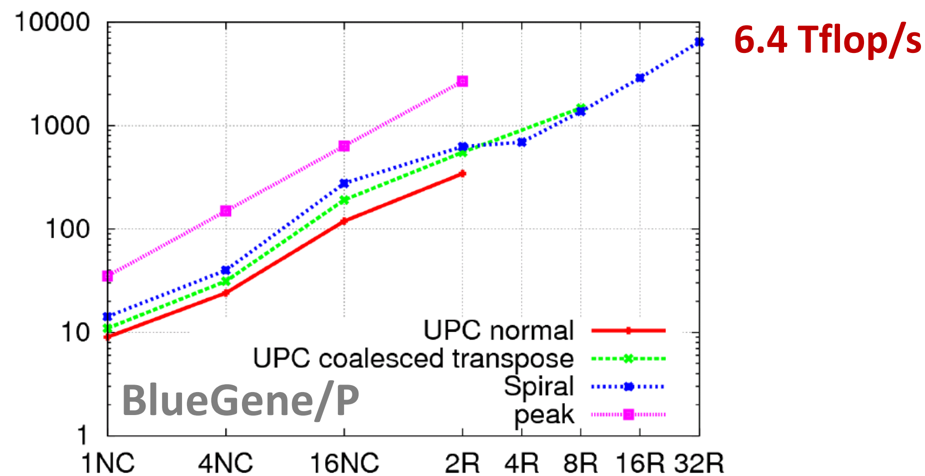


From Cell Phone To Supercomputer

DFT on Samsung Galaxy S II
Dual-core 1.2 GHz Cortex-A9 with NEON ISA



Global FFT (1D FFT, HPC Challenge)
performance [Gflop/s]



Samsung i9100 Galaxy S II
Dual-core ARM at 1.2GHz with NEON ISA

SIMD vectorization + multi-threading

BlueGene/P at Argonne National Laboratory
128k cores (quad-core CPUs) at 850 MHz

SIMD vectorization + multi-threading + MPI

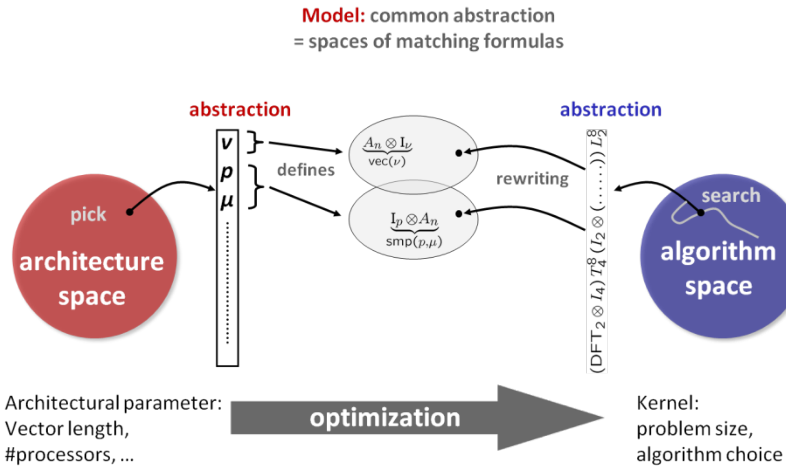
G. Almási, B. Dalton, L. L. Hu, F. Franchetti, Y. Liu, A. Sidelnik, T. Spelce, I. G. Tănase, E. Tiotto, Y. Voronenko, X. Xue:
2010 IBM HPC Challenge Class II Submission. Winner of the 2010 HPC Challenge Class II Award (Most Productive System).

Organization

- Spiral overview
- Validation and Verification
- Results
- **Concluding remarks**

Summary: Spiral in a Nutshell

Joint Abstraction



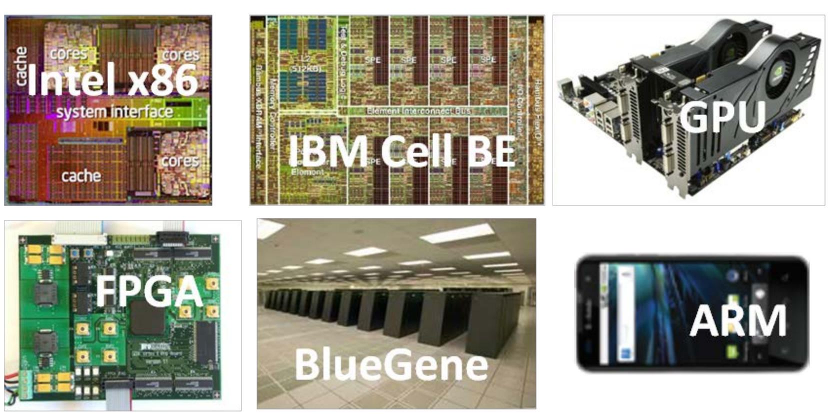
Verification

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

= ?

DFT4([0,1,0,0])

Target Machines



Academic @ CMU: www.spiral.net
Commercial: www.spiralgen.com

Application Domains

Signal Processing

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

Matrix Algorithms

Software Defined Radio

Image Formation (SAR)

Acknowledgement

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

Volodymyr Arbatov
Paolo D'Alberto
Peter A. Milder
Yevgen Voronenko
Qian Yu

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

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

More Information:

www.spiral.net

www.spiralgen.com