Computer Generation of Platform-Adapted Physical Layer Software

Yevgen Voronenko (SpiralGen, CMU)
Christian Berger (CMU)
Volodymyr Arbatov (CMU)
Ronghui Peng (SpiralGen)
Franz Franchetti (CMU)
Markus Püschel (CMU)

SpiralGen at a Glance

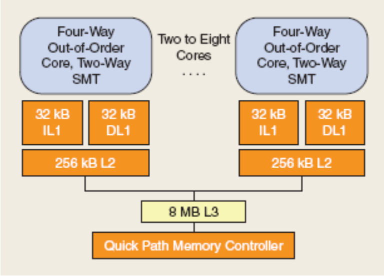
- **Software & Service**
  For computationally intensive apps
  Performance optimization service | Optimized components | Performance evaluation

- **Value proposition**
  - ***Faster time-to-market***
  - ***Quick customization***
  - ***Platform independence w/ no penalty***
- **Target applications:**
  - Core signal/image/video processing
  - Medical Imaging
- **Target platforms:**
  - Multi-core DSPs / GPPs

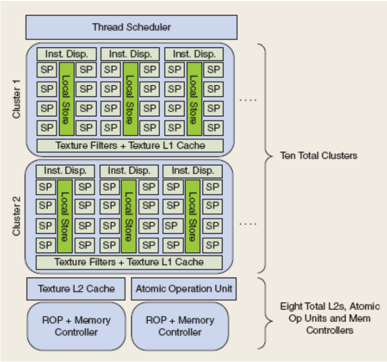
The Future is Parallel

- **General purpose/high performance platforms**
  - Intel Core i7 (4-8 cores, 128-B SIMD)
  - Intel Larrabee (16, 512-B)
  - ARM Cortex (1-4, -)
  - IBM Cell (9, 128-B)
  - Nvidia G200 (240)
- **DSP software platforms**
  - Tilera Tile64 (64, 3-way)
  - Sandbridge SB3500 (4, 128-B)
  - Freescale MSC8156 (6, ?)
  - ~~Clearspeed~~
  - ~~Stream Processors~~

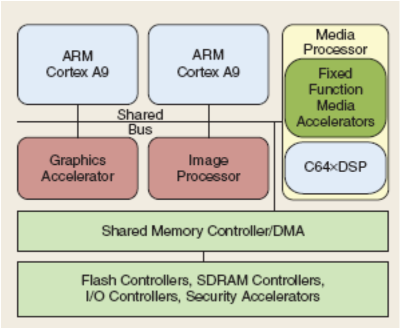
Core i7



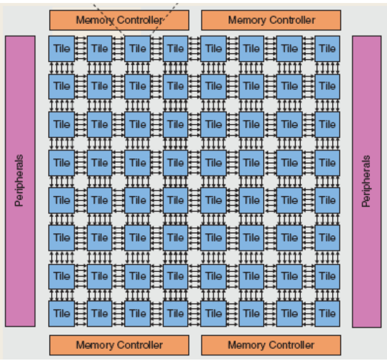
Nvidia G200



TI OMAP



Tilera Tile64

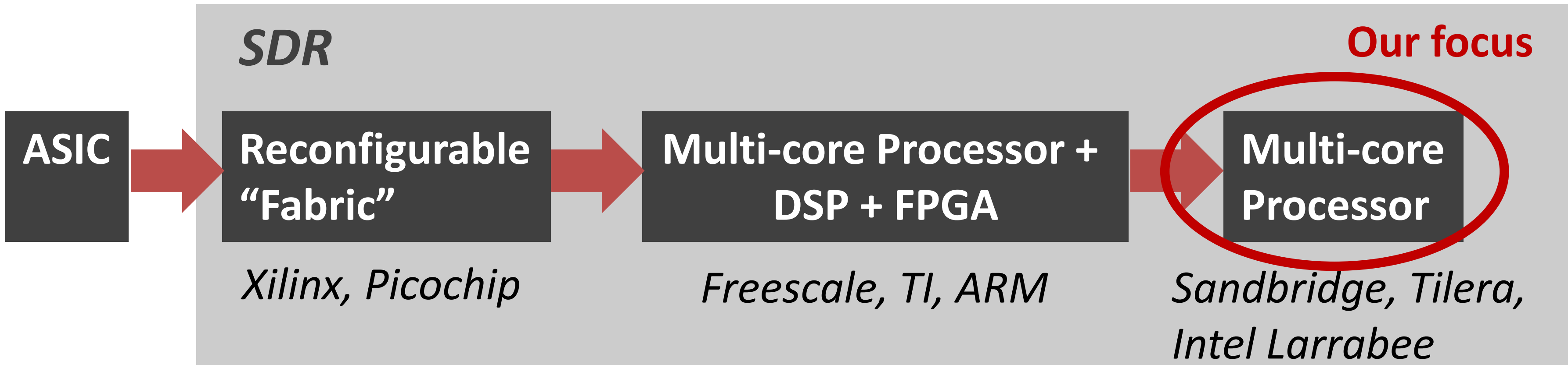


Source: IEEE SP Magazine, Vol. 26, November 2009

Can waveform porting be substantially accelerated?

Rethinking the Software Radio

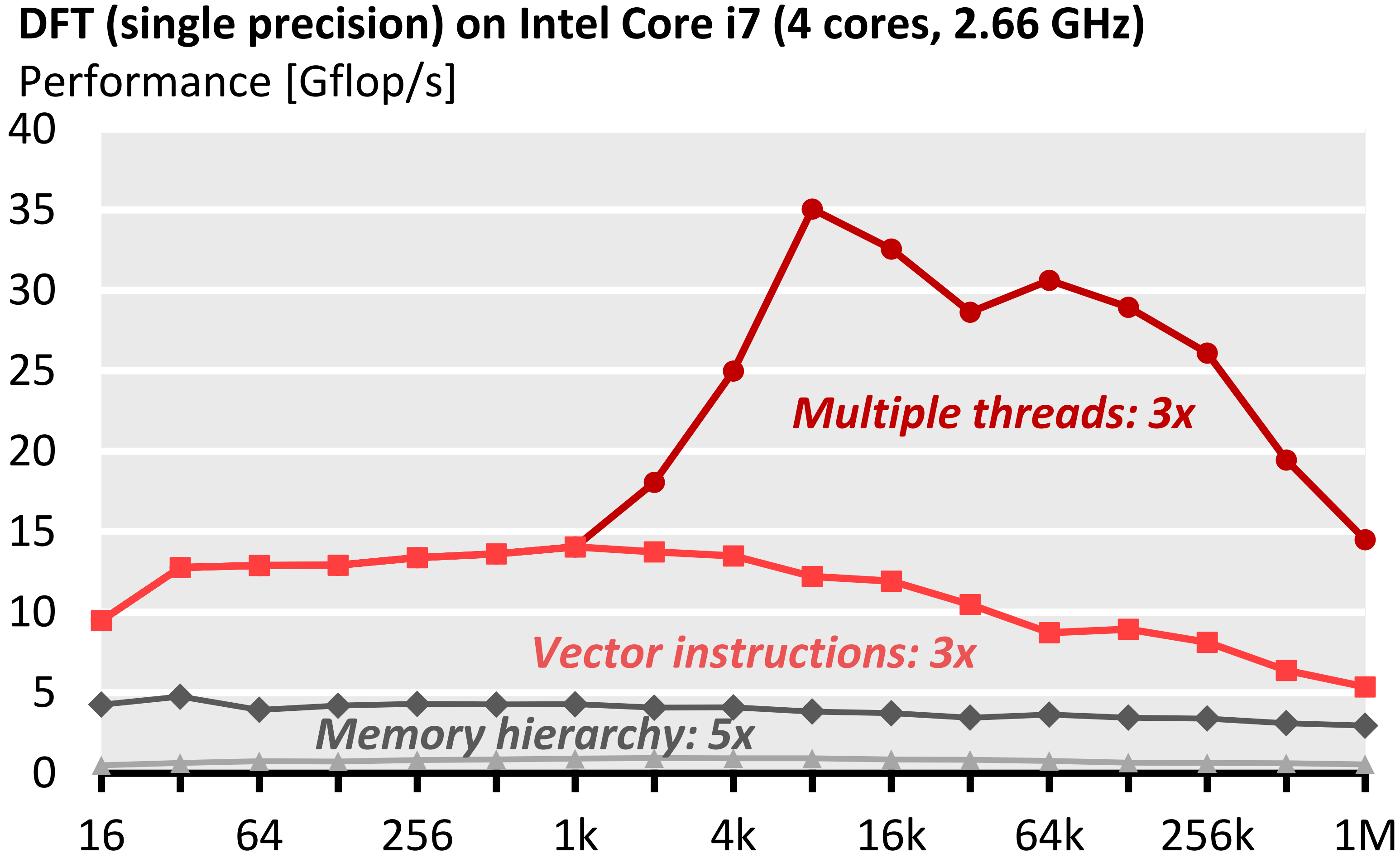
- **Trend for performance-driven apps?**



- **Key driver: reducing the chip count on board**
- **Advantages:**
  - Reduced software partitioning effort + easier porting
  - Lower power + longer battery life
- **Magic bullet: single-chip multi-layer SDR solution**
- **Example:** DSP/GPP/FPGA board – 25W, Intel board – 30W

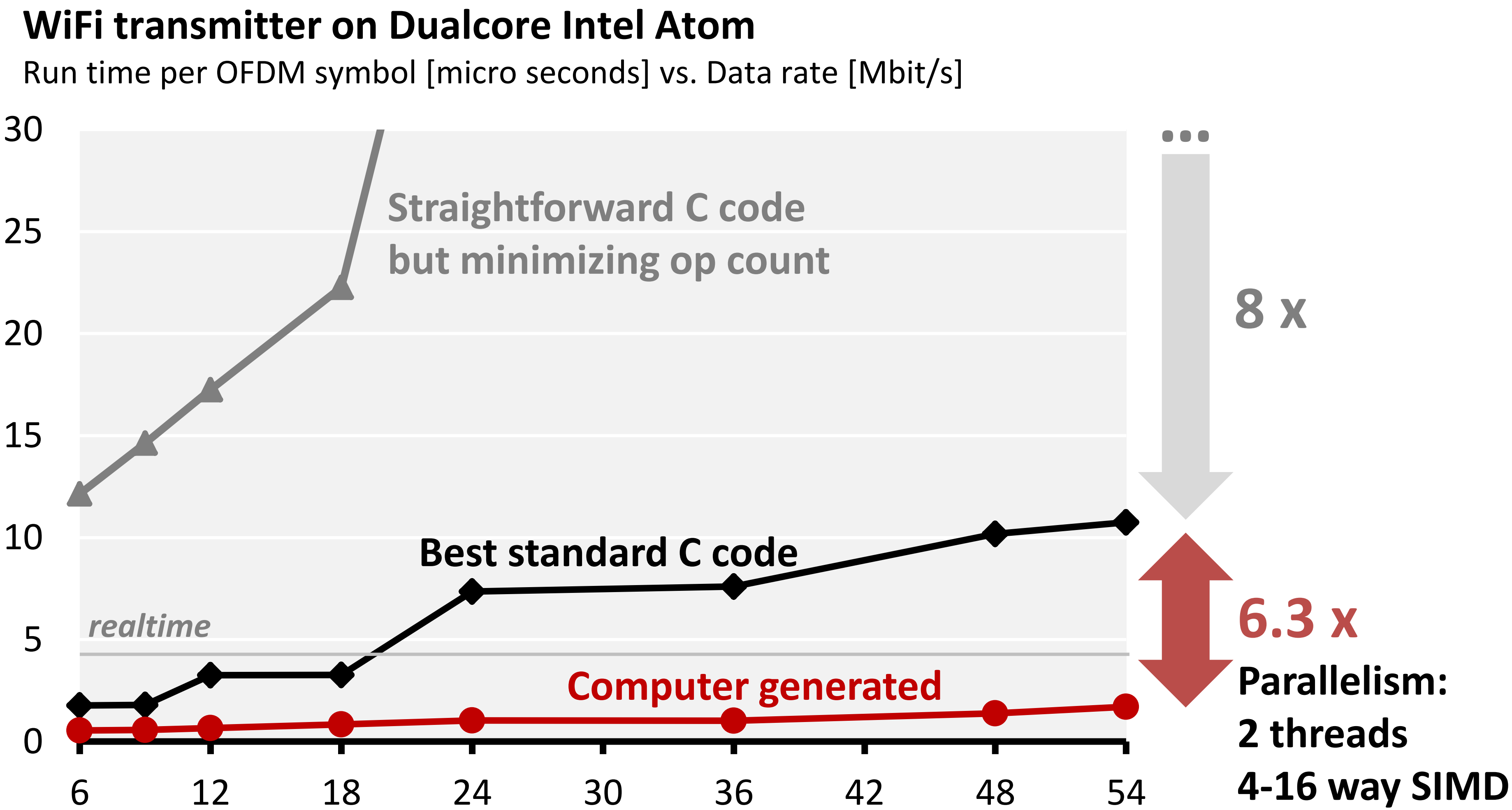
Multi-core GPPs are real. But can compilers retarget the code?

The Answer is No



Only guru programmers can do the job

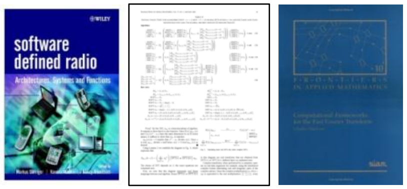
WiFi Transmitter Example



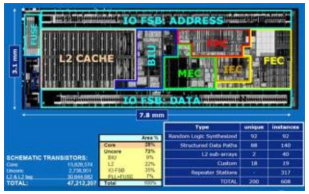
Compilers fail to optimize: 50x

SpiralGen's Core Technology:
Automation Tools

Application specification
Algorithm knowledge



Platform description
Instruction set architecture



_mm_set1_epi8(x) = …
_mm_xor_si128(x,y) = …
_mm_avg_epu8(x,y) = …
_mm_cmpeq_epi8(x,y) = …
_mm_unpacklo_epi8(x,y) = …
…

Spiral

Optimized implementation
(performance/power/energy)

Spiral frees up guru programmers

Proof-of-Concept: Spiral and Intel IPP 6.0

Spiral-generated code in IPP (since 6.0, fall 2008):

- **1M lines of code**
- **3984 C functions, 1203 functions are user visible**
- **Functionality: Cross product of**

  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

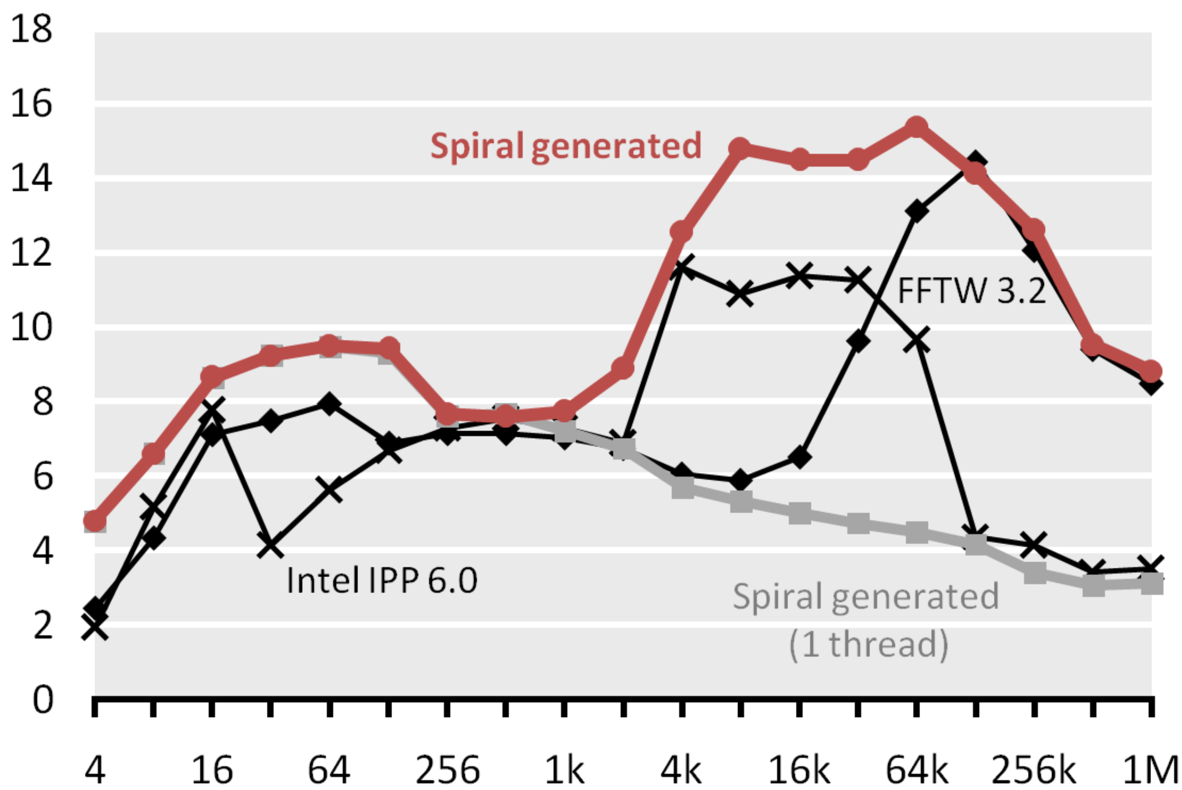
  Data type: scalar, SSE, **AVX** (DFT, DCT), **LRB** (DFT)

Written by a computer

Can we do the same for SDR?

DFT on Intel Multicore

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

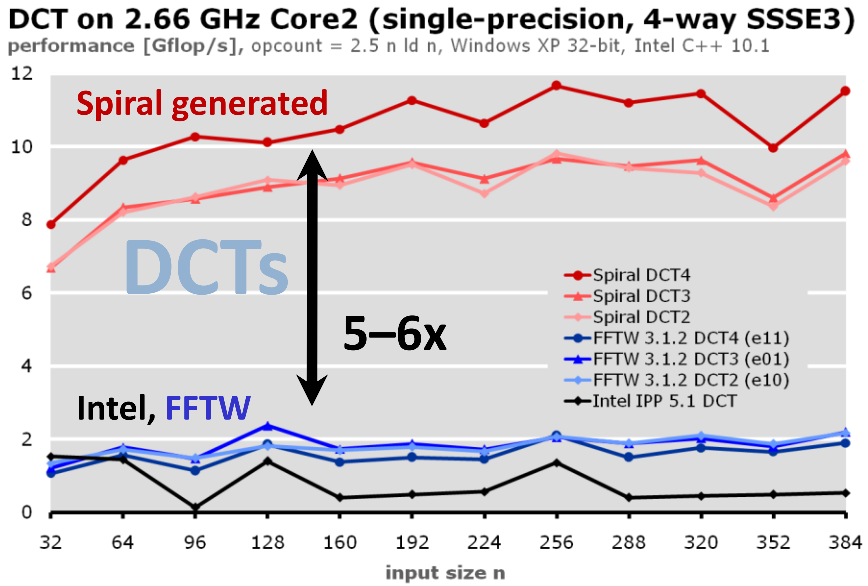


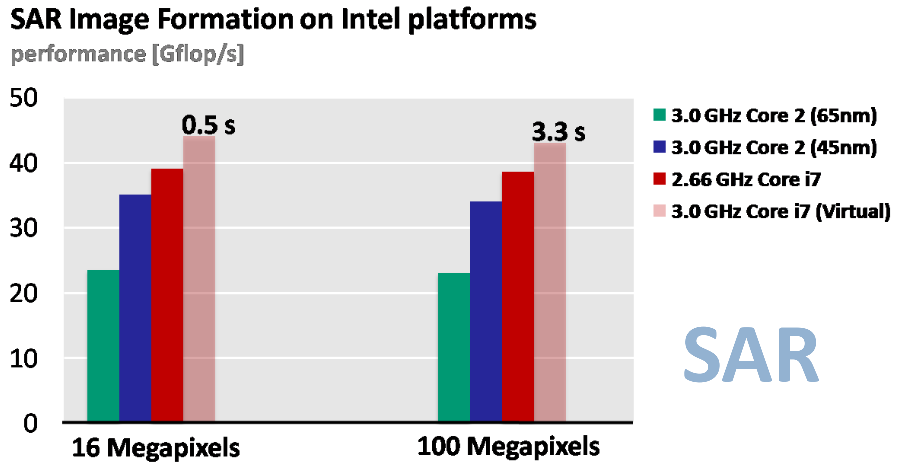
$$\mathbf{DFT}_n \to (\mathbf{DFT}_k \otimes I_m) T^n_m (I_k \otimes \mathbf{DFT}_m) L^n_k$$
$$\mathbf{DFT}_n \to P^{\top}_{k/2,2m} \left(\mathbf{DFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i C_{2m}\, \mathbf{rDFT}_{2m}(i/k)\right)\right) (\mathbf{RDFT}_k \otimes I_m)$$
$$\mathbf{RDFT}_n \to (P^{\top}_{k/2,m} \otimes I_2) \left(\mathbf{RDFT}_{2m} \oplus \left(I_{k/2-1} \otimes_i D_{2m}\, \mathbf{rDFT}_{2m}(i/k)\right)\right) (\mathbf{RDFT}_k \otimes I_m)$$
$$\mathbf{rDFT}_{2n}(u) \to L^{2n}_m \left(I_k \otimes_i \mathbf{rDFT}_{2m}((i+u)/k)\right) (\mathbf{rDFT}_{2k}(u) \otimes I_m)$$

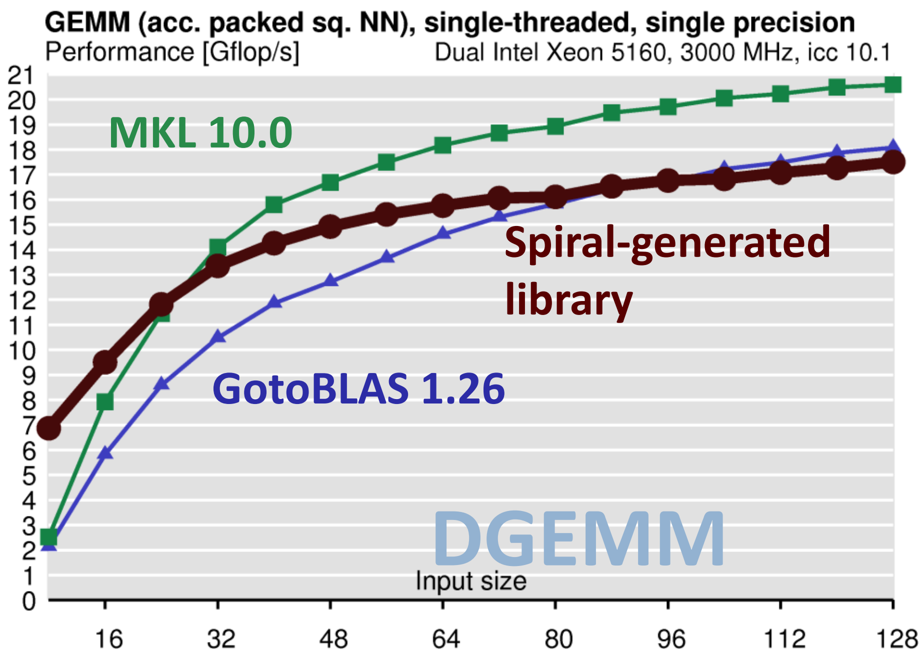
Spiral

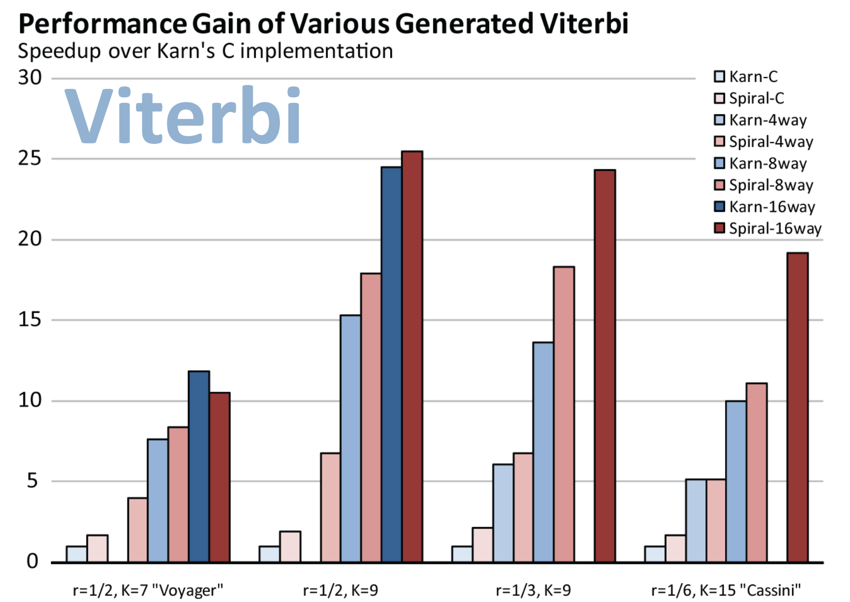
Vectorized, threaded, general-size, adaptive library

Spiral Outperforms Human Programmers

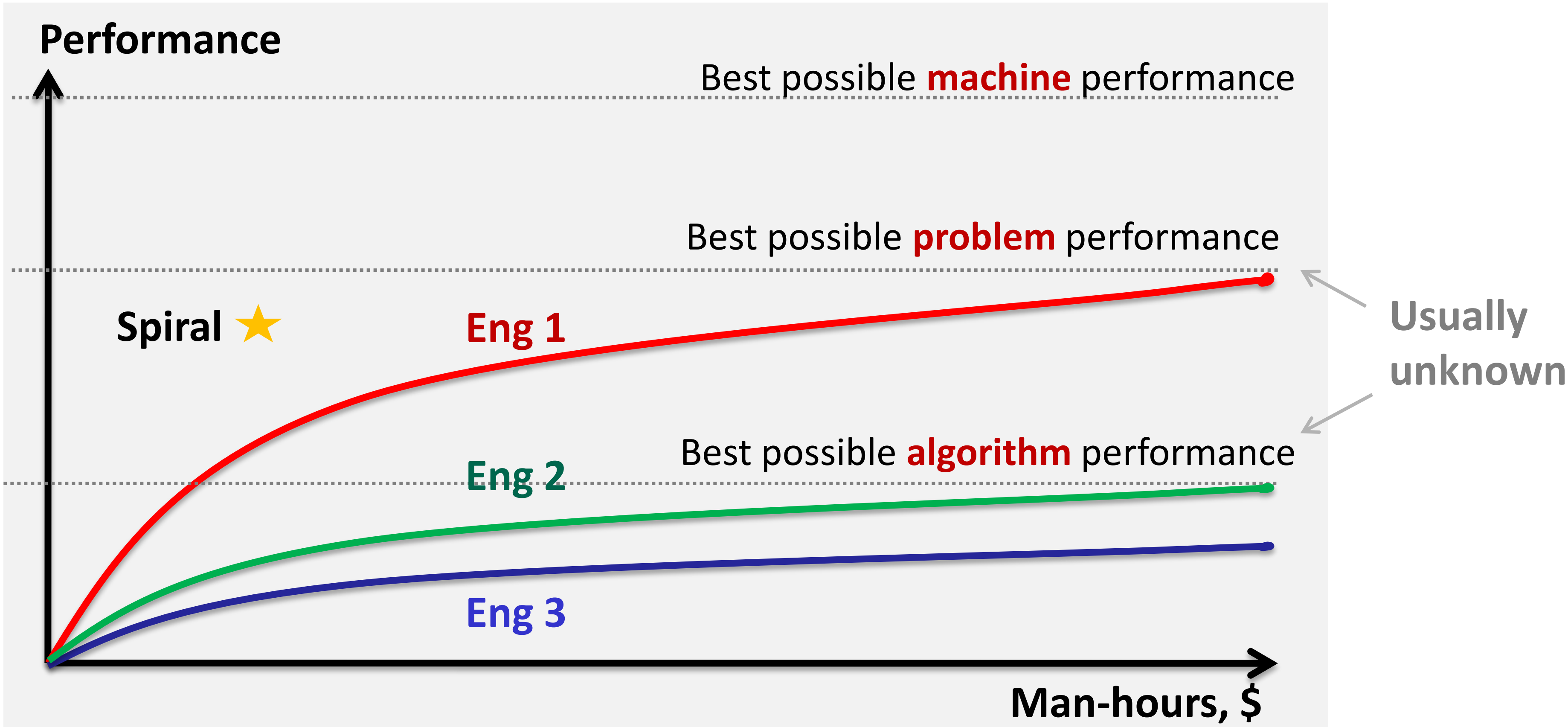








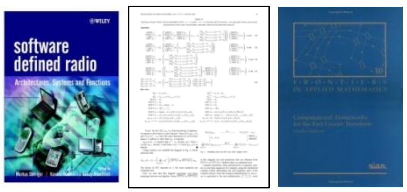
How Can Machine-Generated Code Outperform Hand-Optimized?



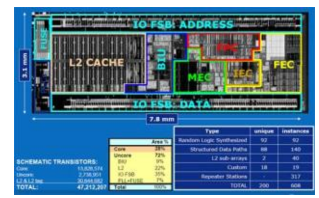
- **Spiral explores both**
  - Algorithms
  - Code restructurings

Spiral: Concept

Application specification
Algorithm knowledge



Platform description
Instruction set architecture



_mm_set1_epi8(x) = …
_mm_xor_si128(x,y) = …
_mm_avg_epu8(x,y) = …
_mm_cmpeq_epi8(x,y) = …
_mm_unpacklo_epi8(x,y) = …
…

Spiral

Optimized implementation
(performance/power/energy)

Spiral: How It Works I

Application specification
Algorithm knowledge

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

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

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

Platform description
Instruction set architecture

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

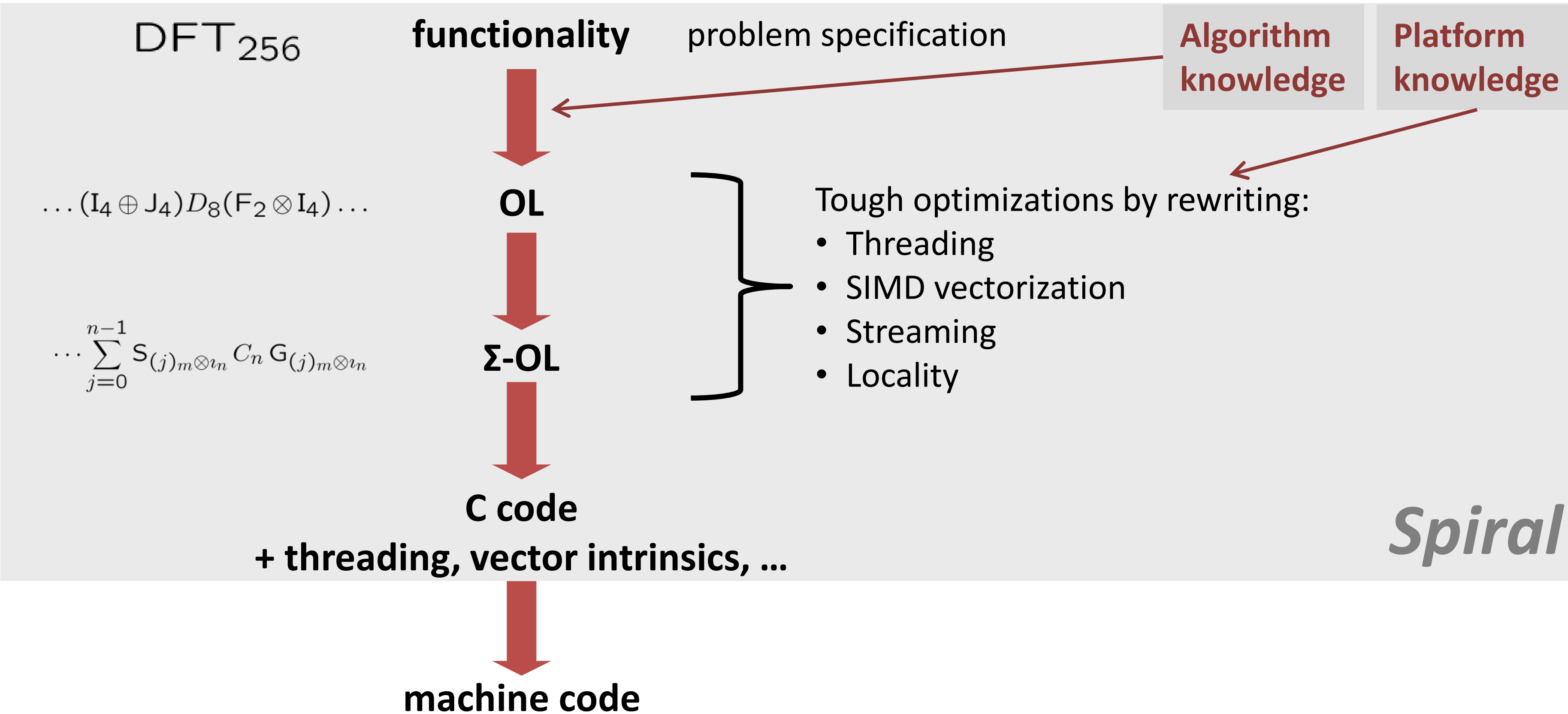
$$\underbrace{\mathrm{I}_m \otimes A_n}_{\mathsf{smp}(p,\mu)} \rightarrow \mathrm{I}_p \otimes_{\|}\left(\mathrm{I}_{m/p} \otimes A_n\right)$$

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

Spiral

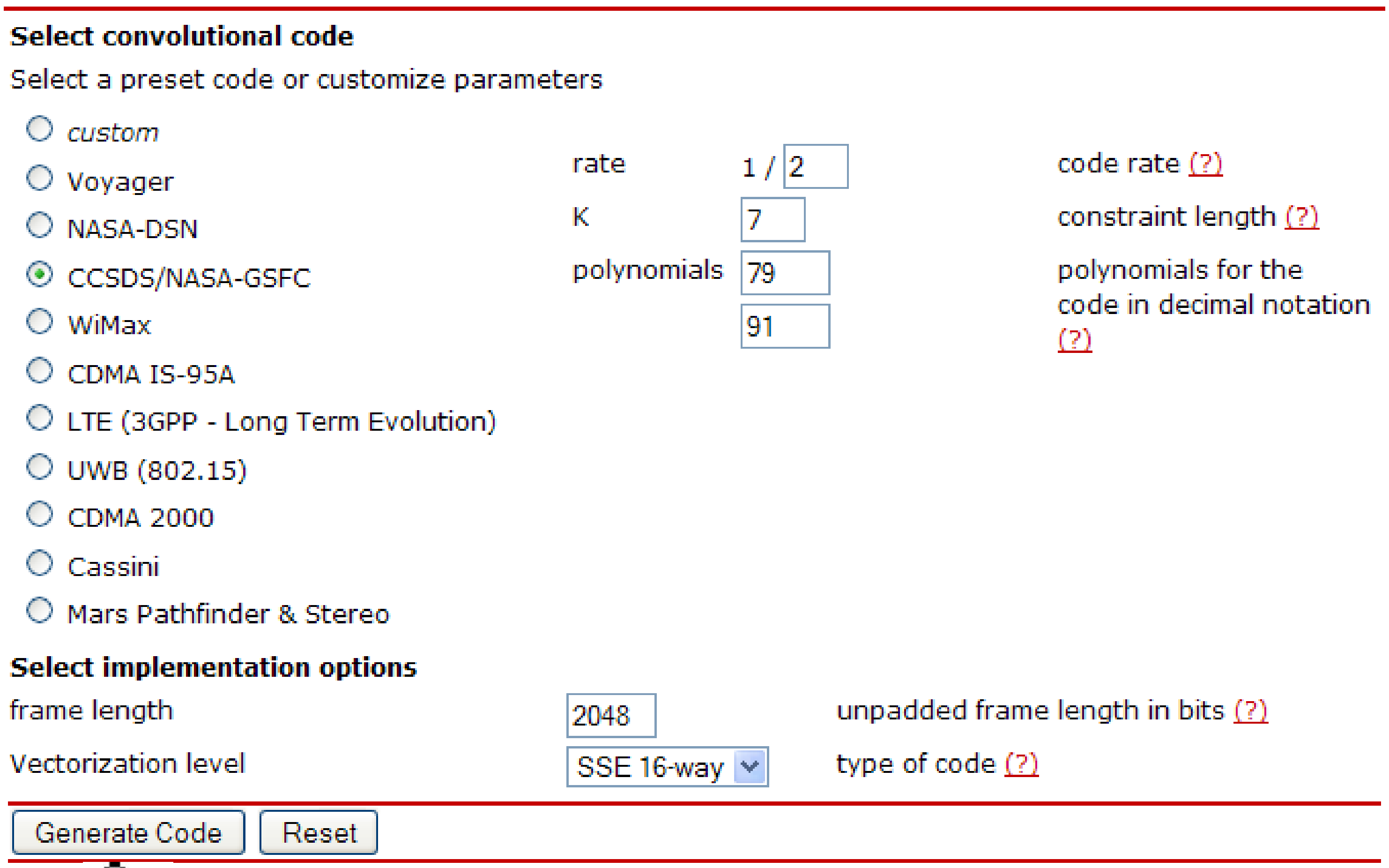
Optimized implementation
(performance/power/energy)

Spiral: How It Works II



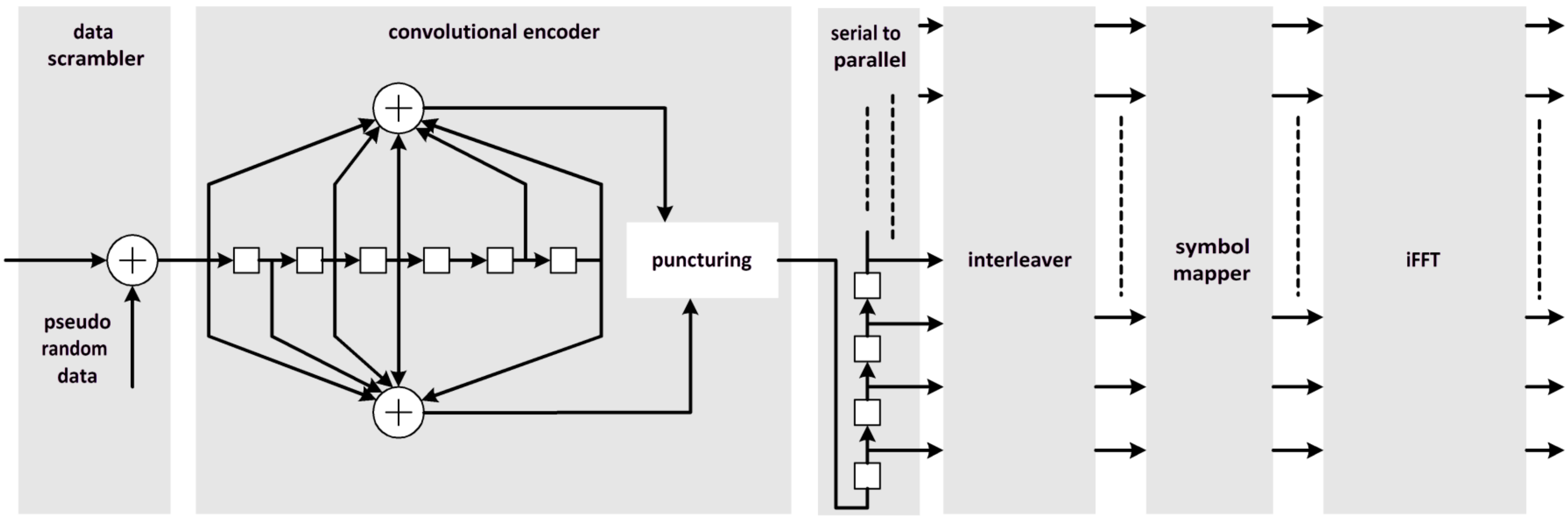
- **Optimization at the high level of abstraction:**
  Overcomes compiler limitations
- **Complete automation**

Example: Viterbi decoder



“Click”: Push-button code generation

“Teaching” Spiral SDR: Algorithm Knowledge



$$\mathbf{WiFiTX}_{k,r,m} \rightarrow \Big(\mathrm{I}_k \otimes (\mathbf{CP\text{-}iDFT}_{64} \circ \mathbf{SymbM}_m \circ \mathbf{Interl}_m)\Big) \circ \mathbf{ConvEnc}_{r,\ell} \circ \mathbf{Scramb}_{i_0,\ell}$$

$$\mathbf{Scramb}_{i_0,\ell} \rightarrow (\mathrm{I}_\ell \otimes_i \mathcal{C}_i) := \bigoplus_{i=i_0}^{i_0+\ell-1} \mathcal{C}_i$$

$$\mathbf{ConvEnc}_{r,\ell} \rightarrow \left(\mathrm{T}_{r,\ell+6} L_{\ell+6}^{2(\ell+6)} \begin{bmatrix} \mathrm{C}_{0,\ell} \\ \mathrm{C}_{1,\ell} \end{bmatrix}\right)$$

$$\mathbf{Interl}_m \rightarrow \left((\mathrm{I}_{16} \otimes_i (\mathrm{I}_3 \otimes \mathrm{W}_i^m))\, L_{16}^{48m}\right)$$

$$\mathbf{SymbM}_m \rightarrow G \circ [\mathrm{I}_{48} \otimes \mathrm{S}_m]$$

$$\mathbf{CP\text{-}iDFT}_N \rightarrow \left(\frac{1}{\sqrt{N}} P_N \mathbf{iDFT}_N\right)$$

plus
operator definitions and additional rules

“Teaching” Spiral SDR: Platform Knowledge

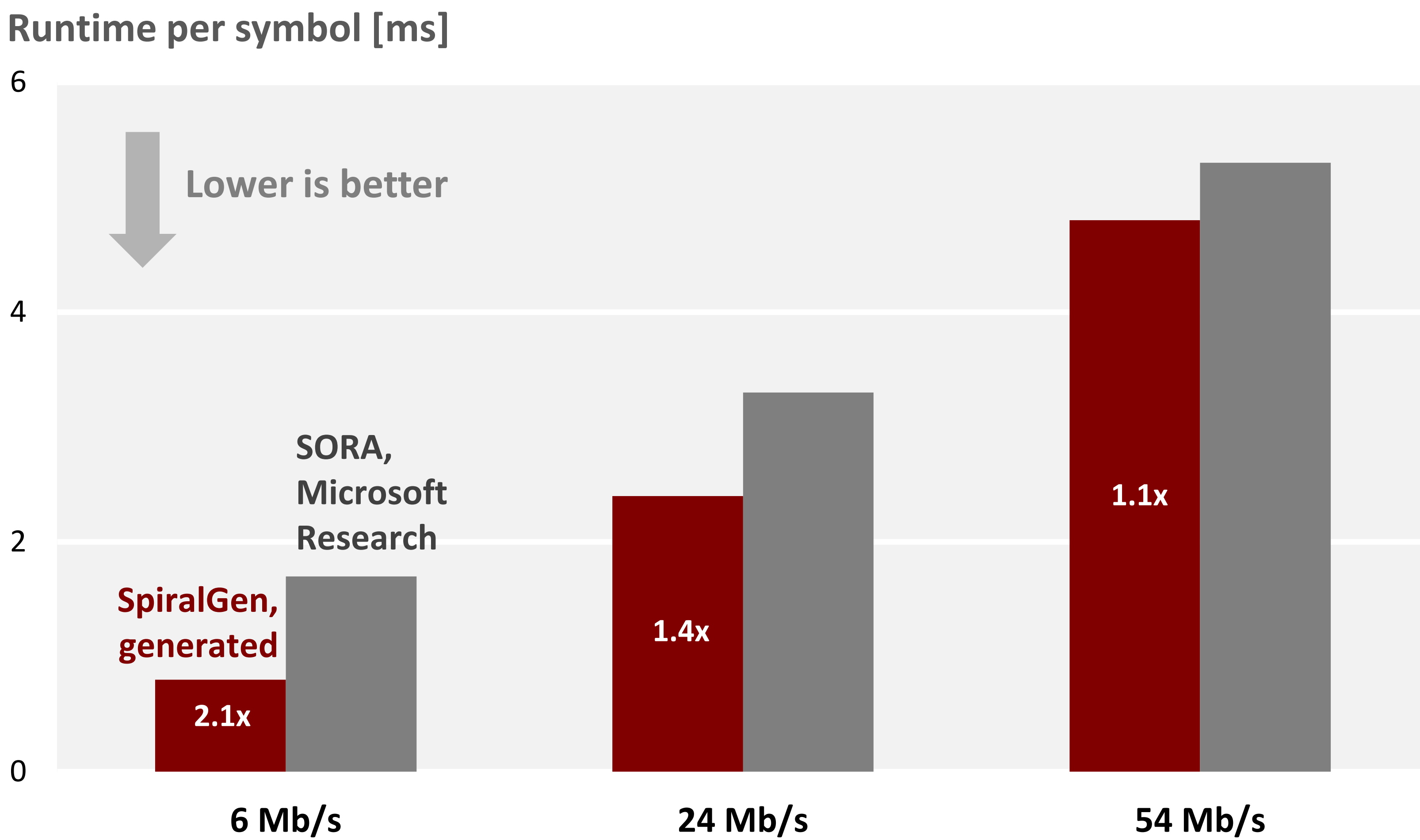
- **Rewriting rules**
  - Parallelization
  - SIMD vectorization
  - Cross block optimizations
  - SIMD ISA specific simplification

- **Backend extensions**
  - **New data types:** bit, integer (8, 16, 32)
  - **Vectorization modes:** 128-way, 16-way, 8-way, 4-way (ISAs)
  - **ISA bridges:** data types conversion in vector registers
  - Support for finite field arithmetic

What We Did in a Nutshell

- **WiFi 802.11.g (OFDM), 8 data rate modes**
  - Expressed in Spiral's OL
  - Extended OL for mixed data type formulas
  - Extended OL for mixed vector-length formulas
- **Created optimized implementations w/ Spiral**
  - Threading
  - SIMD vectorization, with **mixed data types / vector length**
  - Auto-tuning using best algorithm search
- **Benchmarked on 3 platforms**
  - Intel Atom
  - Intel Core 2
  - Intel Core i7

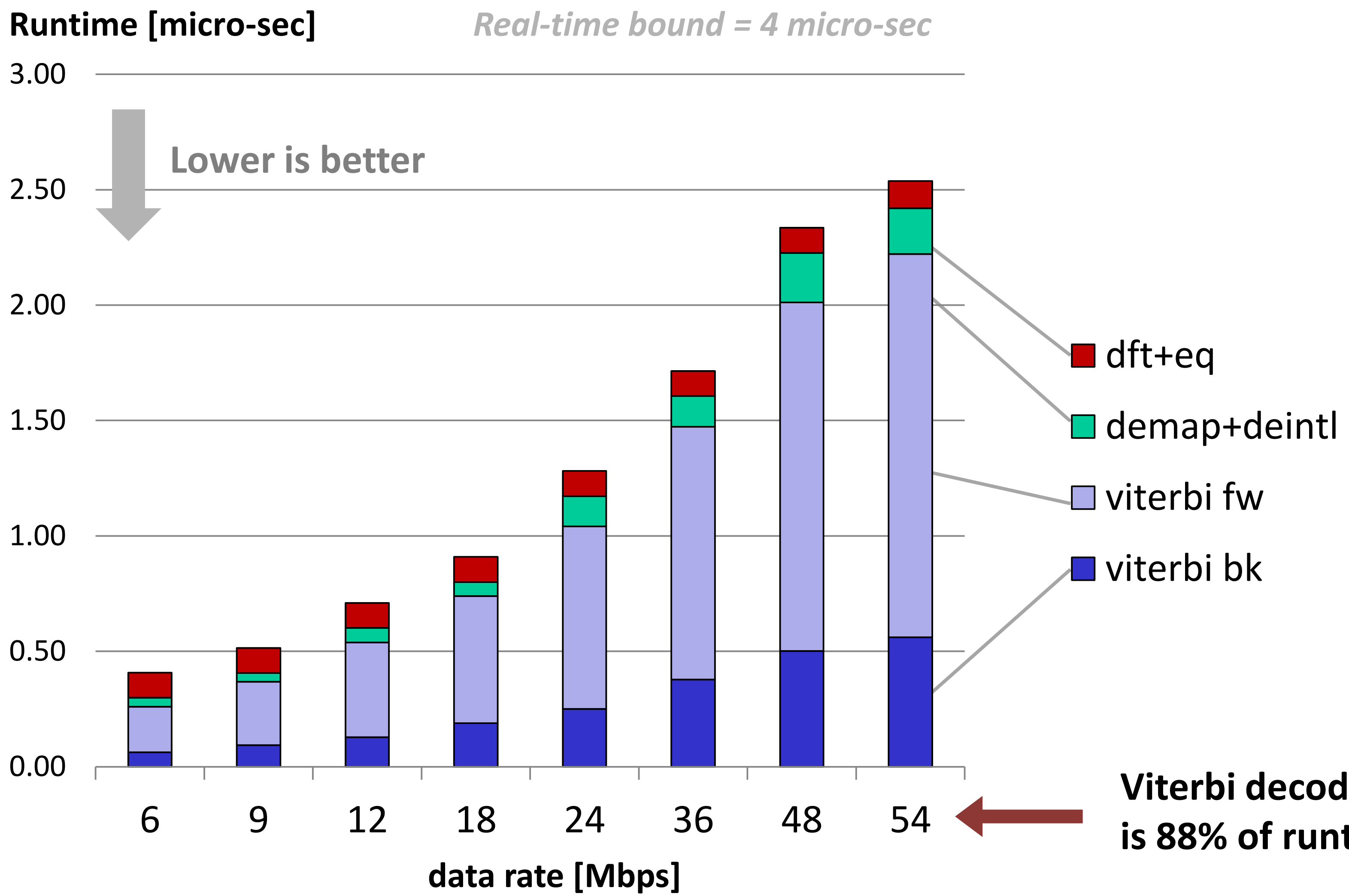
802.11b Receiver: Generated vs. Hand-tuned



2.1x more energy efficient
(Mflops/watt)

Compute Time Distribution

Auto-generated receiver on Core i7



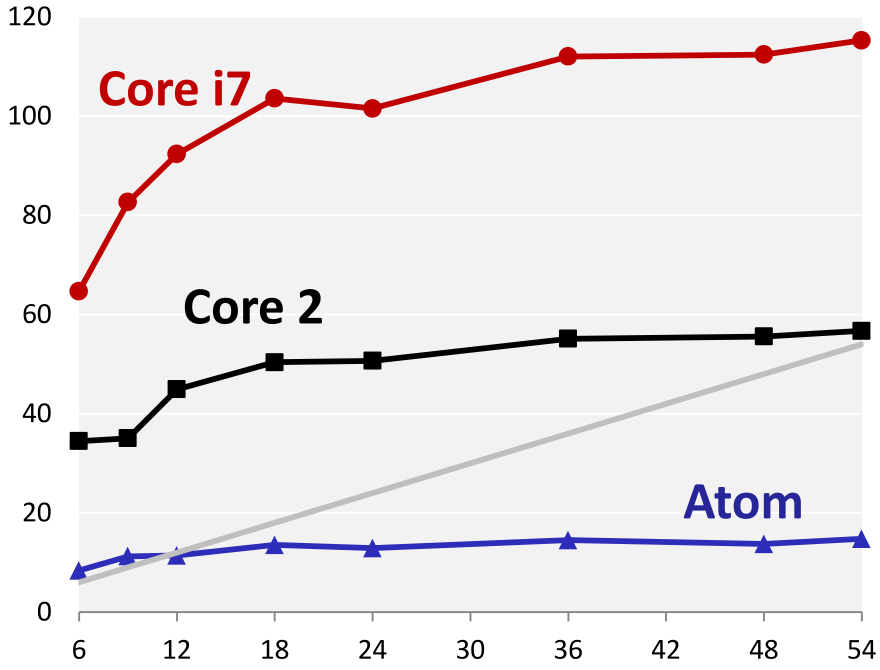
Viterbi decoding is 88% of runtime

Compute-bound Bandwidth

Auto-generated WiFi receiver/transmitter (parallel)

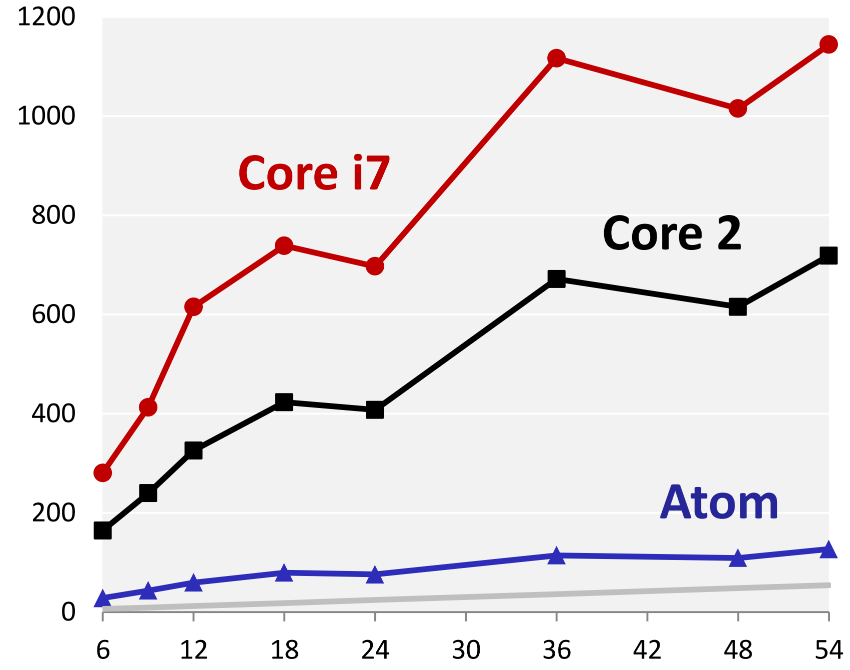
Receiver

Throughput [Mbit/s]

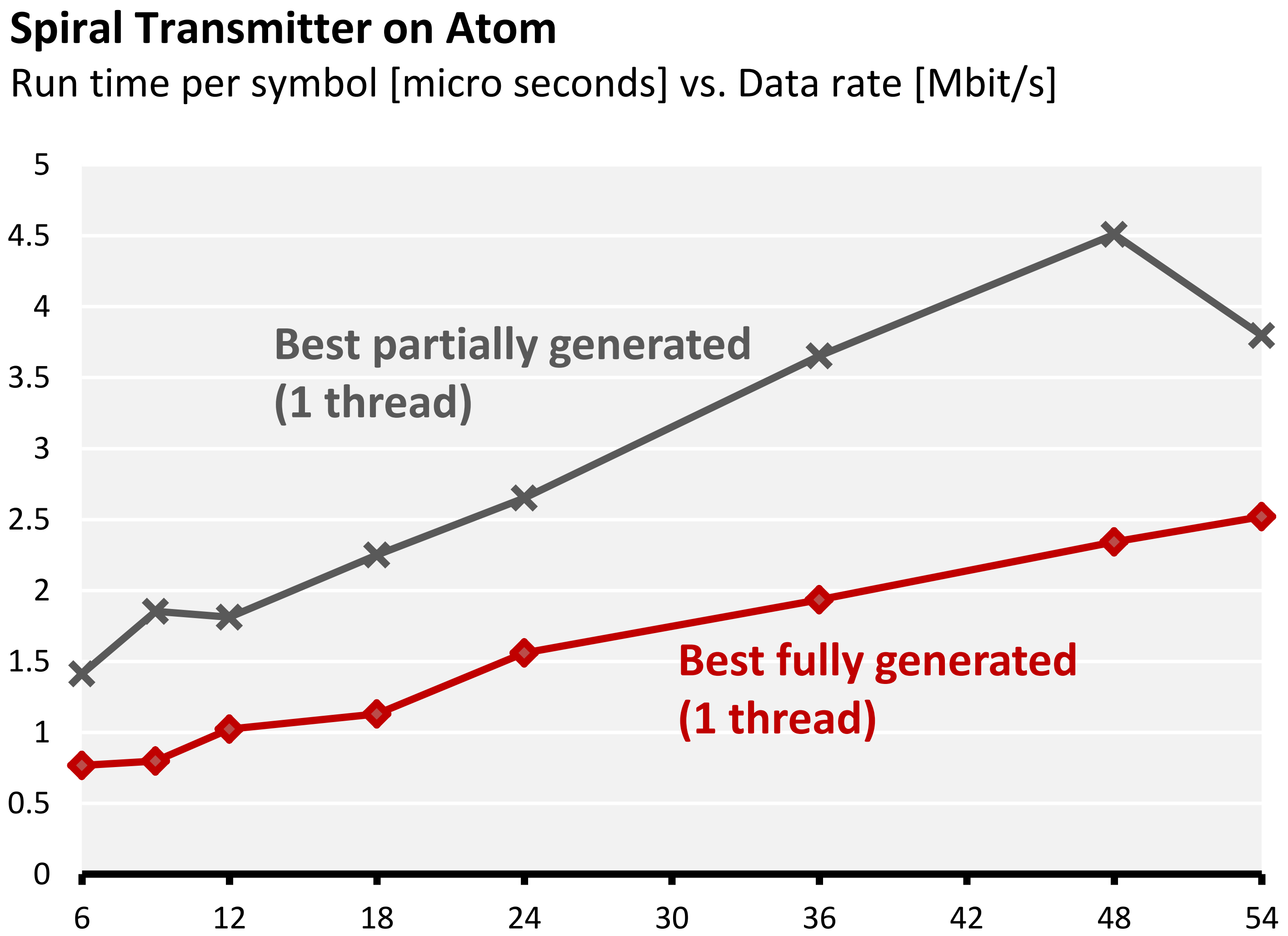


Transmitter

Throughput [Mbit/s]



Ability to Cross-Block Optimize



Conclusions

- The generated code is *very fast*
  (often faster than any human written code)

- Spiral supports *different types of architectures* including vector, multicore and distributed platforms

- Spiral enables *very fast transition to new architectures*

SpiralGen
Accelerating Innovation in Computing

Carnegie Mellon University

SPIRAL