Generating Platform-Adapted DSP Libraries Using SPIRAL

José Moura (CMU)

Jeremy Johnson (Drexel)
Robert Johnson (MathStar Inc.)
David Padua (UIUC)
Viktor Prasanna (USC)
Markus Püschel (CMU)
Bryan Singer (CMU)
Manuela Veloso (CMU)
Jianxin Xiong (UIUC)

www.ece.cmu.edu/~spiral

Electrical & Computer ENGINEERING

SPIRAL

Sponsor

Work supported by DARPA (DSO), Applied & Computational Mathematics Program, OPAL, through grant managed by research grant DABT63-98-1-0004 administered by the Army Directorate of Contracting.

Electrical & Computer ENGINEERING

SPIRAL

SPIRAL

Automates the

Implementation

- cuts development costs
- coding less error-prone

Optimization

- code manipulation techniques like, e.g., unrolling cannot be done by hand in reasonable time
- allows systematic exploration of alternatives both at algorithmic level and code optimizations

Platform-Adaptation

- takes advantage of architecture specific features
- porting without loss of performance

of DSP algorithms

- are performance critical

→ **A library generator for highly optimized, platform-adapted signal processing transforms**

Electrical & Computer ENGINEERING

SPIRAL

Organization

- **SPIRAL approach**
- SPIRAL system
- Some experimental results
- Recent work

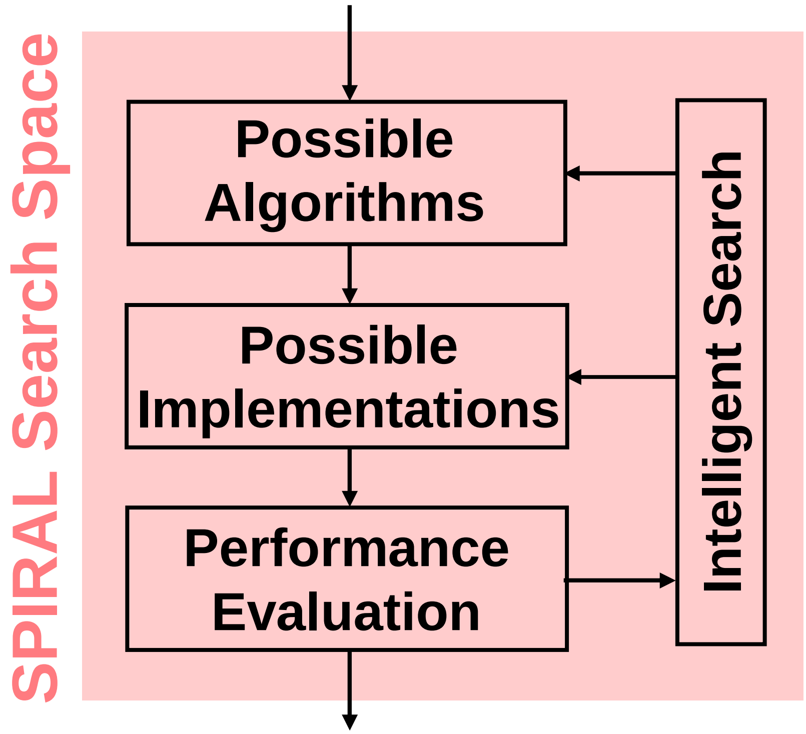
Key Observations

- For every DSP transform there are exponential many different algorithms, which do not differ in arithmetic cost
- The best algorithm is highly platform dependent
- The best algorithm is hard to determine

SPIRAL Methodology

given → **DSP Transform**

(DFT, DCT, Wavelets etc.)



→ **adapted implementation**

given → **Computer Architecture**

Uniprocessor:

- Pentium
- SUN
- Alpha

- Multiprocessor
- Hardware

Electrical & Computer ENGINEERING

SPIRAL

DSP Algorithms: Example 4-point DFT

Cooley/Tukey FFT (size 4):

Fourier transform | Diagonal matrix (twiddles)

$$DFT_4 = (DFT_2 \otimes I_2) \cdot T_2^4 \cdot (I_2 \otimes DFT_2) \cdot L_2^4$$

Kronecker product | Identity | Permutation

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & & 1 & \\ & 1 & & 1 \\ \hline 1 & & -1 & \\ & 1 & & -1 \end{array}\right] \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & j \end{bmatrix} \left[\begin{array}{cc|cc} 1 & 1 & & \\ 1 & -1 & & \\ \hline & & 1 & 1 \\ & & 1 & -1 \end{array}\right] \left[\begin{array}{cc|cc} 1 & & & \\ & 0 & 1 & \\ \hline & 1 & 0 & \\ & & & 1 \end{array}\right]$$

- **product of structured sparse matrices**
- **mathematical notation**

Electrical & Computer ENGINEERING

SPIRAL

Transforms, Rules, & Formulas

DSP transform DFT_{nm} **a matrix**

Rule

$$DFT_{nm} \rightarrow \left(DFT_n \otimes I_m\right) \cdot D \cdot \left(I_n \otimes DFT_m\right) \cdot P$$

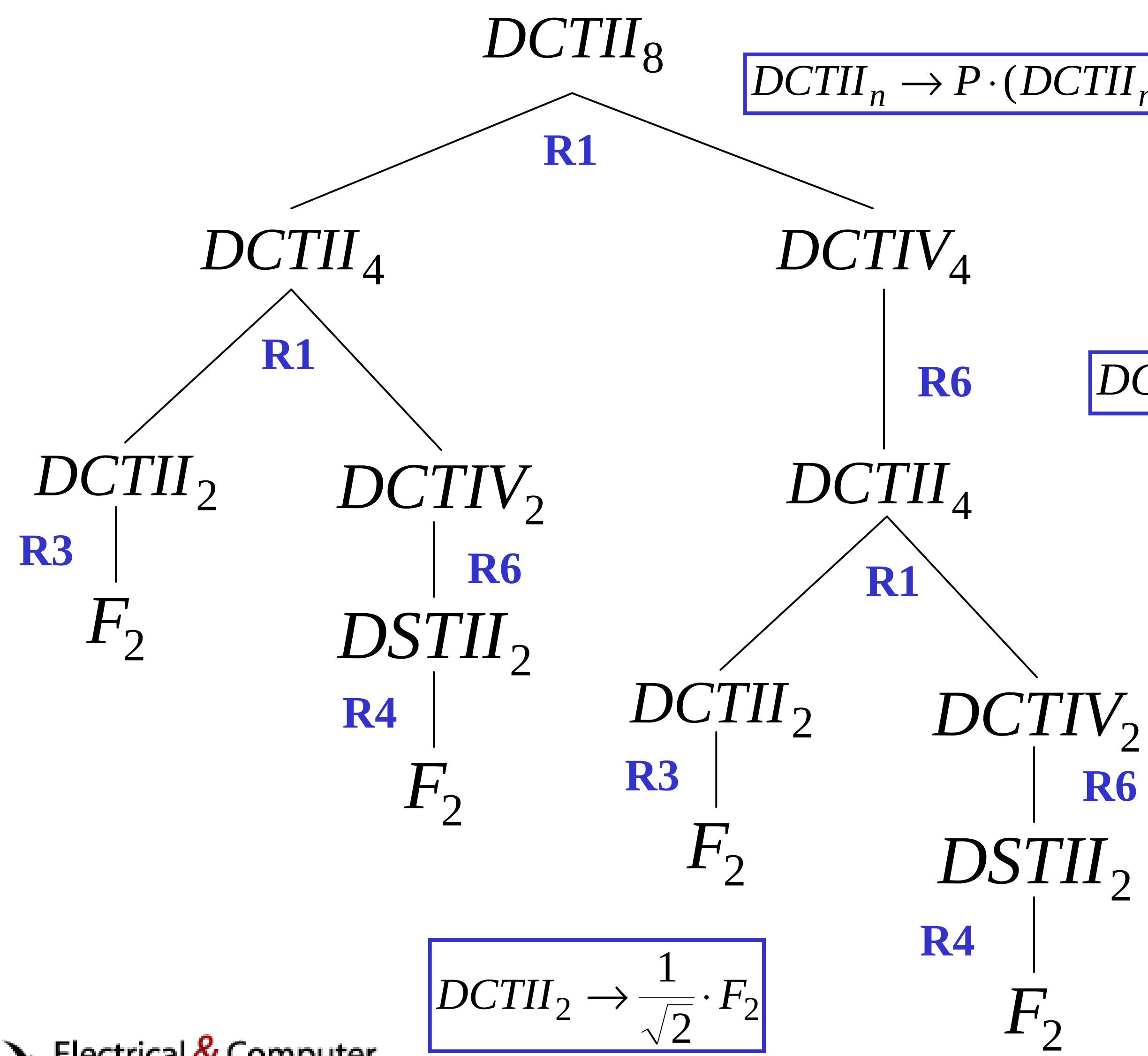
- a breakdown strategy
- product of sparse matrices

Formula

$$DFT_8 = \left(F_2 \otimes I_4\right) \cdot D \cdot \left(I_2 \otimes \left(I_2 \otimes F_2 \cdots\right)\right) \cdot P$$

- arises from recursive application of rules
- product of sparse matrices
- uniquely defines an algorithm

Algorithms = Ruletrees = Formulas



$$DCTII_n \rightarrow P \cdot (DCTII_{n/2} \oplus DCTIV_{n/2}) \cdot (F_2 \otimes I_{n/2})$$

$$DCTIV_n \rightarrow P \cdot DCTII_n \cdot S$$

$$DCTII_2 \rightarrow \frac{1}{\sqrt{2}} \cdot F_2$$

Electrical & Computer ENGINEERING

SPIRAL

Number of Formulas/Algorithms

Currently 12 transforms and 31 rules:

k	# DFTs, size 2^k	# DCTIVs, size 2^k
1	1	1
2	6	10
3	40	126
4	296	31242
5	27744	1924443362
6	162570361280	7343815121631354242
7	~1.01 • 10^27	~1.07 • 10^38
8	~2.31 • 10^61	~2.30 • 10^76
9	~2.86 • 10^133	~1.06 • 10^153

➡ **exponential search space**

Electrical & Computer ENGINEERING

SPIRAL

Formulas in SPL

• • • •

```
( compose
     ( diagonal ( 2*cos(1/16*pi) 2*cos(3/16*pi) 2*cos(5/16*pi) 2*cos(7/16*pi) ) )
     ( permutation ( 1 3 4 2 ) )
     ( tensor
       ( I 2 )
       ( F 2 )
     )
     ( permutation ( 1 4 2 3 ) )
     ( direct_sum
       ( compose
         ( F 2 )
         ( diagonal ( 1 sqrt(1/2) ) )
       )
       ( compose
         ( matrix
           ( 1 1 0 )
           ( 0 (-1) 1 )
         )
         ( diagonal ( cos(13/8*pi)-sin(13/8*pi) sin(13/8*pi) cos(13/8*pi)+sin(13/8*pi) ) )
         ( matrix
           ( 1 0 )
           ( 1 1 )
           ( 0 1 )
         )
         ( permutation ( 2 1 ) )
```

• • • •

SPL Compiler, 4-point FFT

fast algorithm as formula as SPL program

```
(compose (tensor (F 2) (I 2)) (T 4 2)
  (tensor (I 2) (F 2)) (L 4 2))
```

#codetype

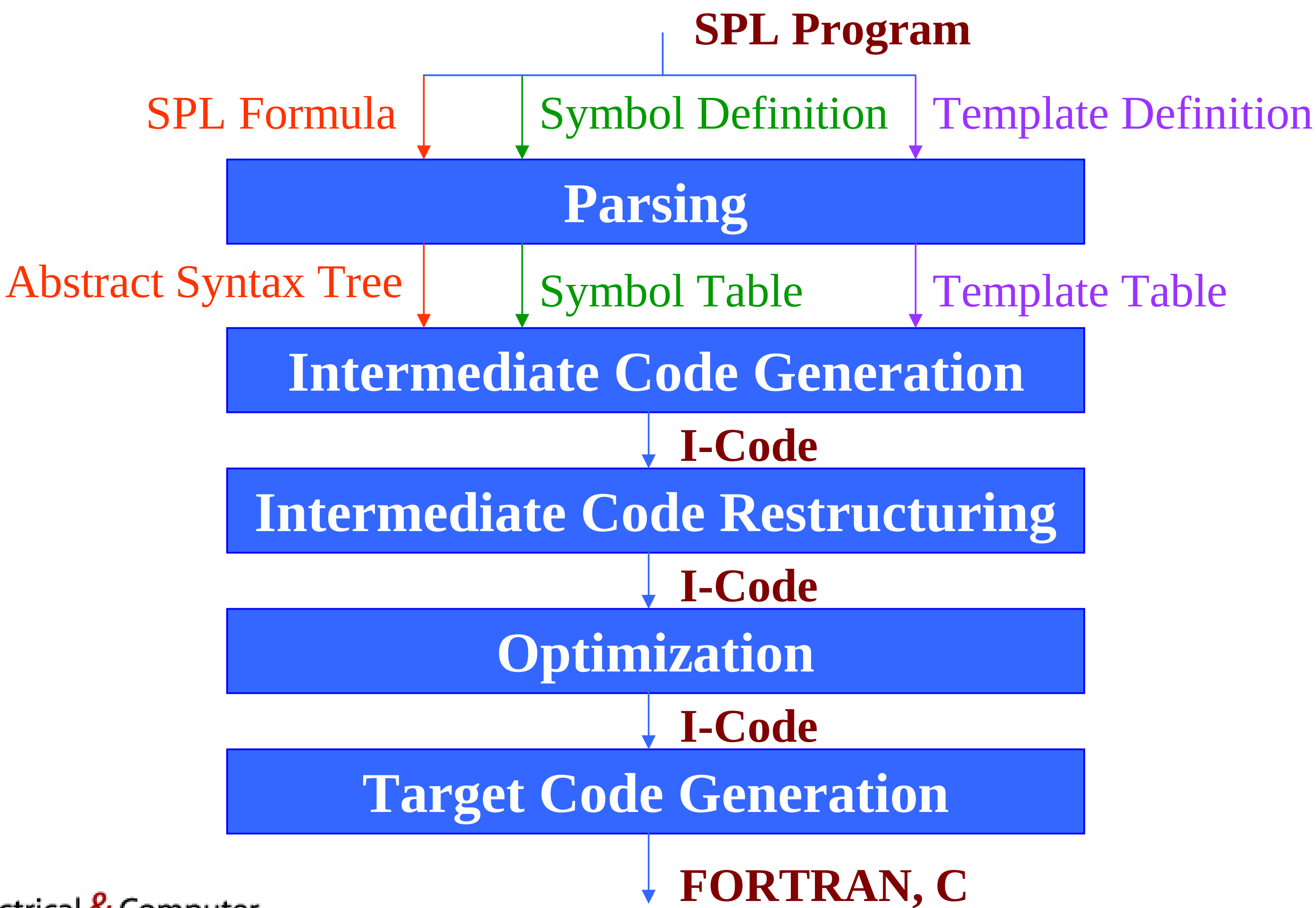
complex

```
f0   = x(1) + x(3)
f1   = x(1) - x(3)
f2   = x(2) + x(4)
f3   = x(2) - x(4)
f4   = (0.00d0,-1.00d0)*f(3)
y(1) = f0 + f2
y(2) = f0 - f2
y(3) = f1 + f4
y(4) = f1 - f4
```

real

```
r0   = x(1) + x(5)
r1   = x(1) - x(5)
r2   = x(2) + x(6)
r3   = x(2) - x(6)
r4   = x(3) + x(7)
r5   = x(3) - x(7)
r6   = x(4) + x(8)
r7   = x(4) - x(8)
y(1) = r0 + r4
y(2) = r1 + r5
y(3) = r0 - r4
y(4) = r1 - r5
y(5) = r2 + r7
y(6) = r3 - r6
y(7) = r2 - r7
y(8) = r3 + r6
```

The SPL Compiler



Search Methods Available in SPIRAL

- **Exhaustive Search**
- **Dynamic Programming (DP)**
- **Random Search**
- **STEER (similar to a genetic algorithm)**

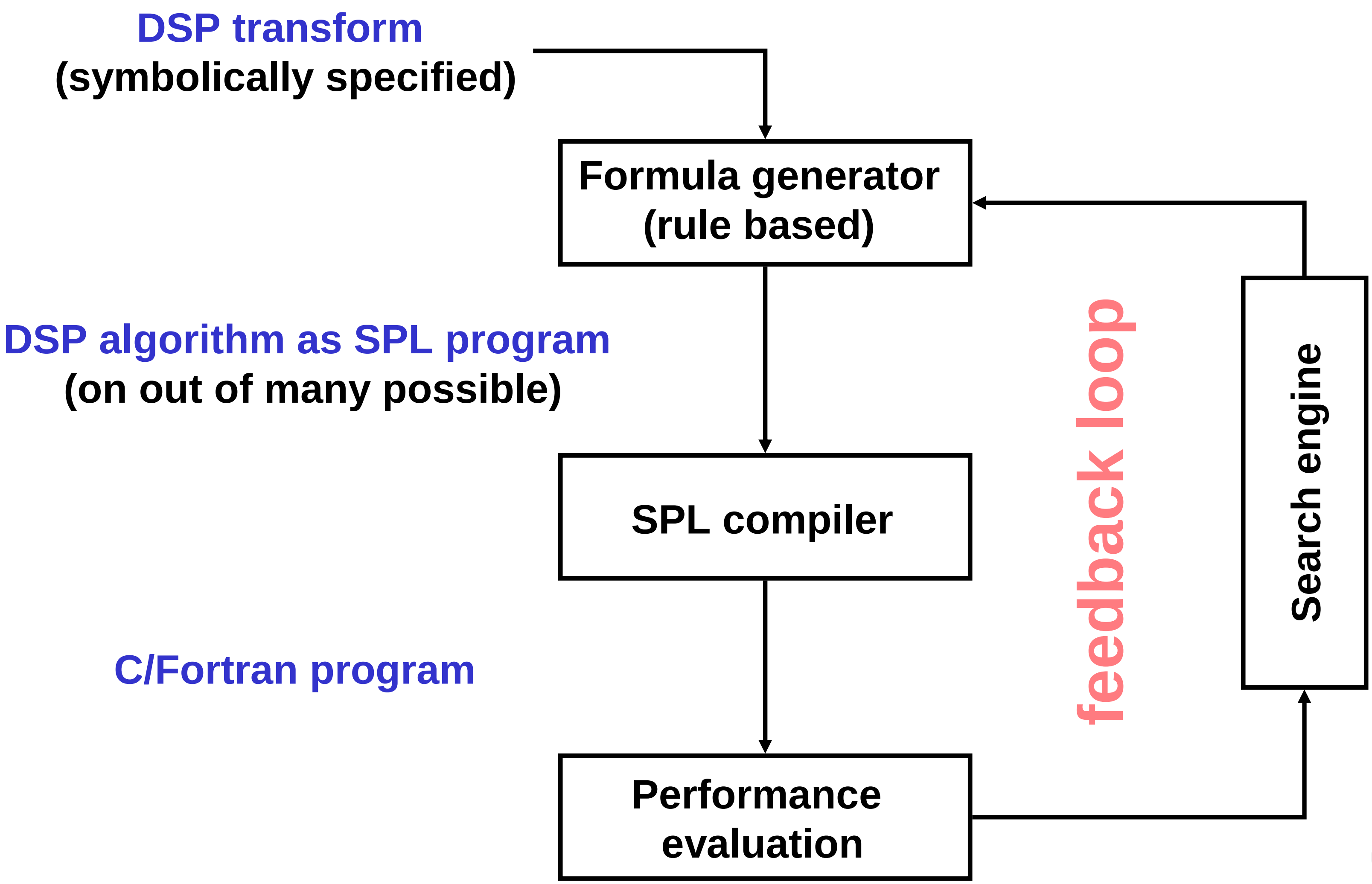
	Possible Sizes	Formulas Timed	Results
Exhaust	Very small	All	Best
DP	All	10s-100s	Good
Random	All	User decided	Poor to fair
STEER	All	100s-1000s	Very good

- **Search over new user-defined transforms and breakdown rules**
- **Search over formulas and options to SPL compiler**

Electrical & Computer ENGINEERING

SPIRAL

Summary: SPIRAL Architecture

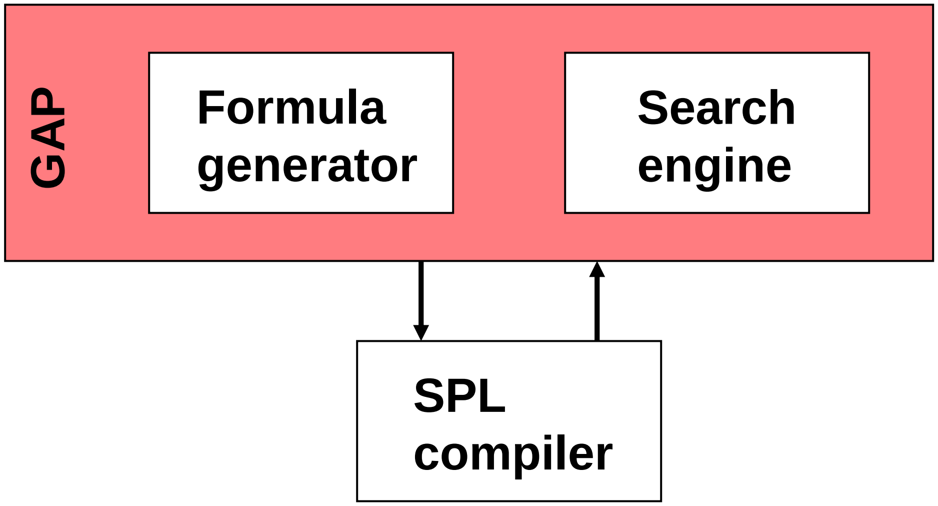


Organization

- SPIRAL approach
- **SPIRAL system**
- Some experimental results
- Recent work

The SPIRAL System: Implementation

- Infrastructure of SPIRAL is based on the computer algebra system and language GAP (`http://www-gap.dcs.st-and.ac.uk/~gap/`)
  - ➢ command line interface
  - ➢ symbolic (exact) computation with DSP formulas
  - ➢ full-fledged programming environment
- Formula generator and search engine implemented in GAP
- SPL compiler implemented in C



Electrical & Computer ENGINEERING

SPIRAL

The SPIRAL System: Main Features

- Easy installation from one source on
  - ➢ Unix based systems (configure – make)
  - ➢ native Windows systems (Visual C/Intel compiler make)
- DSP transforms: DFT, DCTs, DSTs, WHT, Haar transform, …
- new transforms can easily be included
- multi-dimensional transforms automatically supported
- composed DSP transforms supported
- verification of generated code
- programming environment included (GAP)
- online documentation

download at: `www.ece.cmu.edu/~spiral`

Electrical & Computer ENGINEERING

SPIRAL

SPIRAL System Examples I

Implementing a DFT of size 1024 in C:

SPIRAL command prompt

transform size

```
spiral> S := Transform("DFT", 1024);
spiral> Implement(S, rec(search := "DP",language := "c”));
```

search method: dynamic programming

target language

➡ C function in working directory

SPIRAL System Examples II

Implementing an 8 x 8 DCT of type 2 in Fortran:

```
spiral> S := Transform("DCT2", 8);
spiral> S1 := TensorSPL(S, S);
spiral> Implement(S1, rec(search := “STEER",
                          language := "f77”));
```

search method: STEER

SPIRAL System Examples III

Implementing a composed transform in C:

```
spiral> S1 := Transform("DFT", 8);
spiral> S2 := DiagSPL([1, 2, 4, 2, 3, 5, 1, -2]);
spiral> S3 := Transform(“DCT3”, 8);
spiral> S := S1 * S2 * S3;
spiral> Implement(S, rec(search := “TimedSearch",
                         timeLimit := 30,
                         language := “c”));
```

search method:
timed search 30 minutes

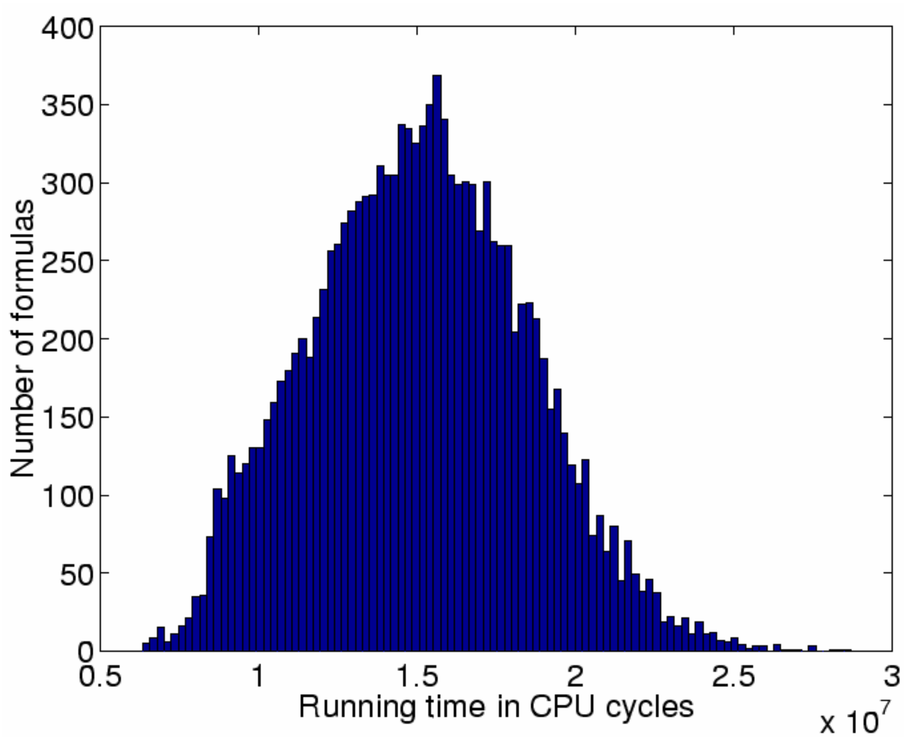
a DCT type 3 followed by scaling followed by a DFT

Organization

- SPIRAL approach
- SPIRAL system
- **Some experimental results**
- Recent work

Search Space and Varying Performance

WHT(2^{10}): 51,819 (binary) ruletrees = formulas



- **large spread in runtime**
- **not due to arithmetic cost**
- **good ones are rare**

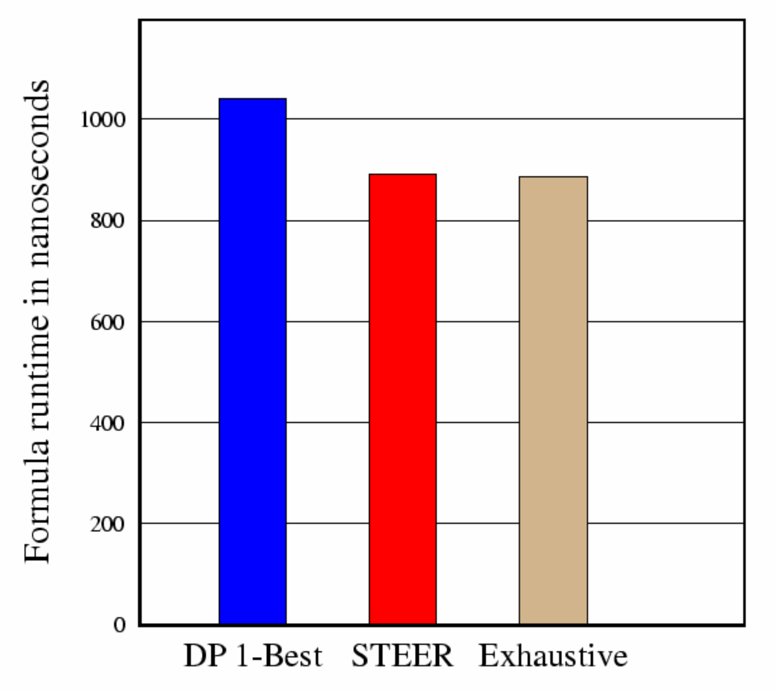
Electrical & Computer ENGINEERING

SPIRAL

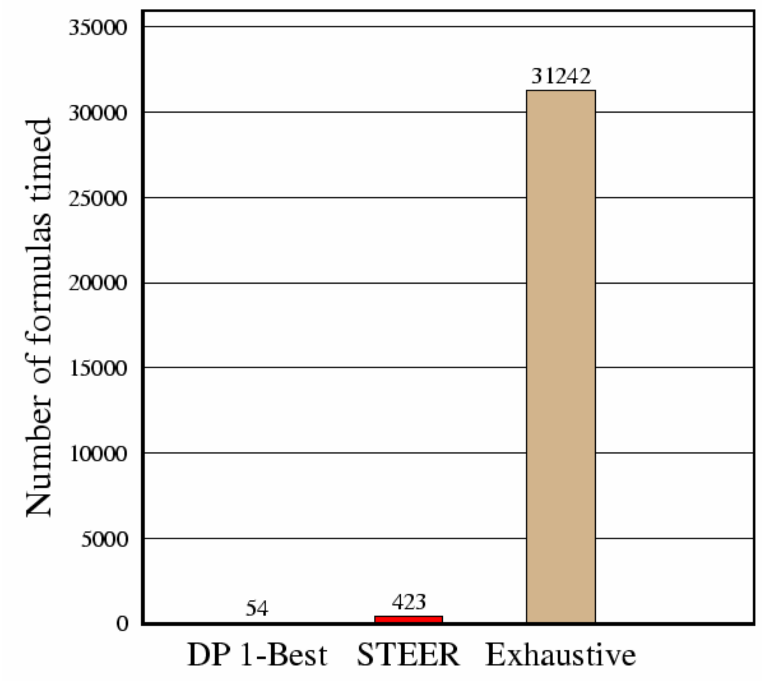
Comparison Search Methods I

DCT, type IV, size 16

Fastest Found Formulas



Number of Formulas Timed



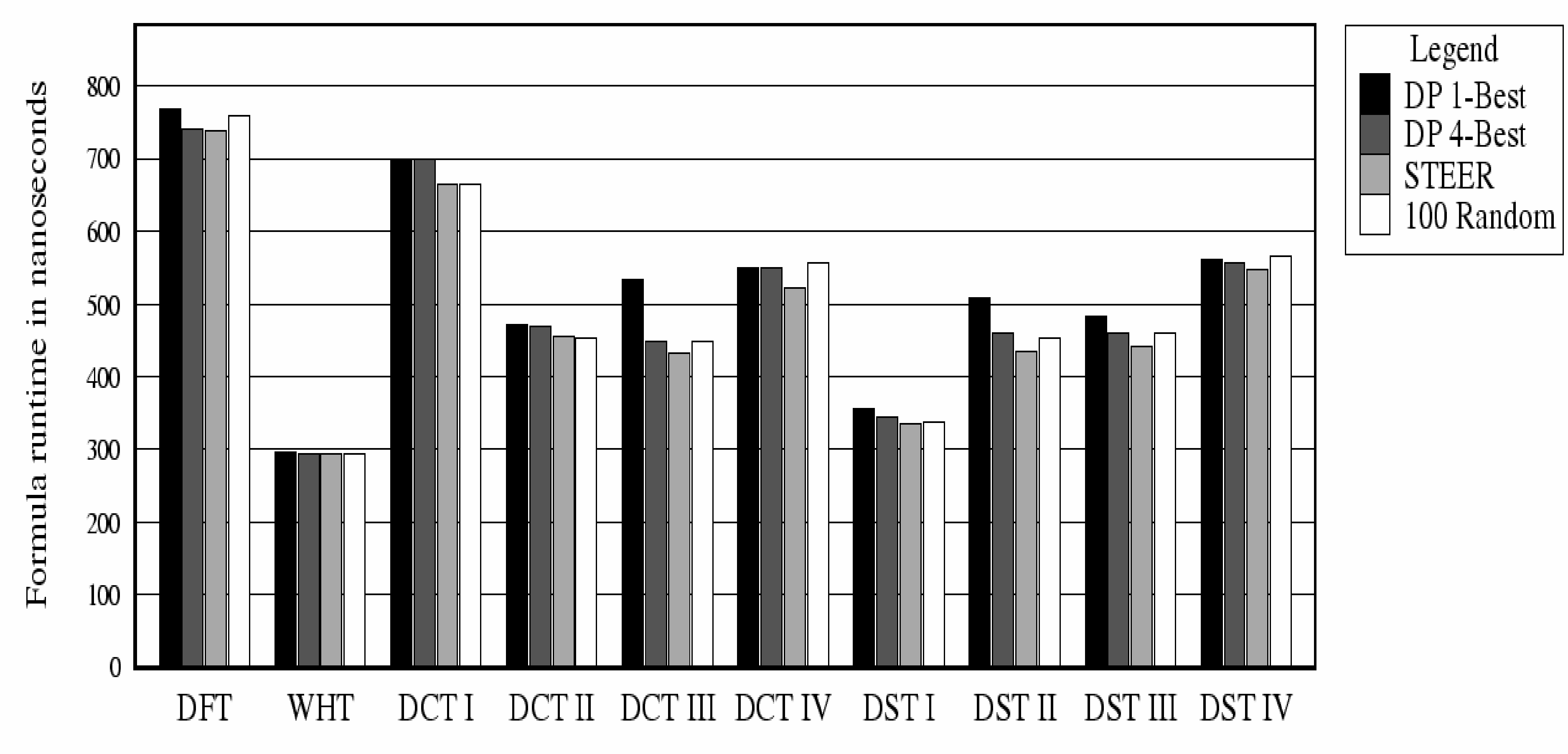
DP and STEER perform well

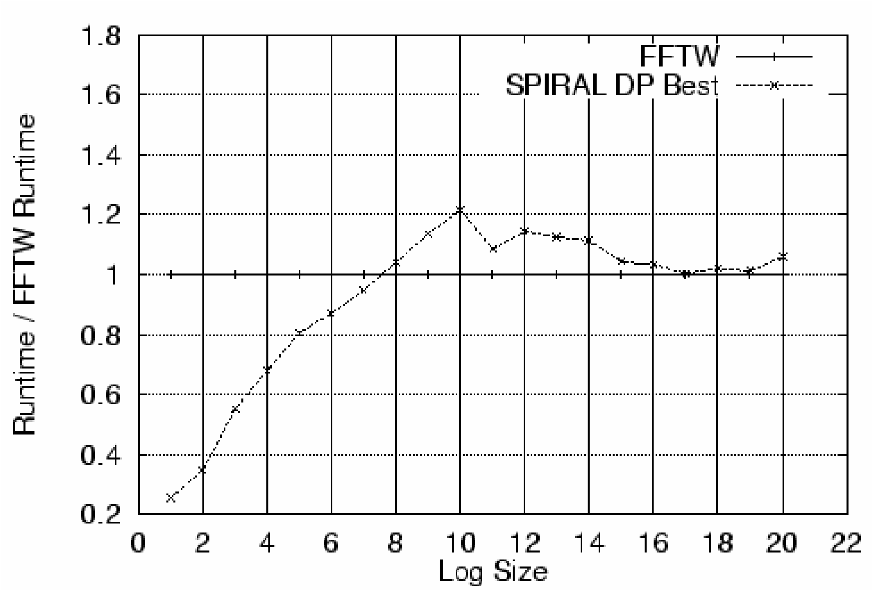
Electrical & Computer ENGINEERING

SPIRAL

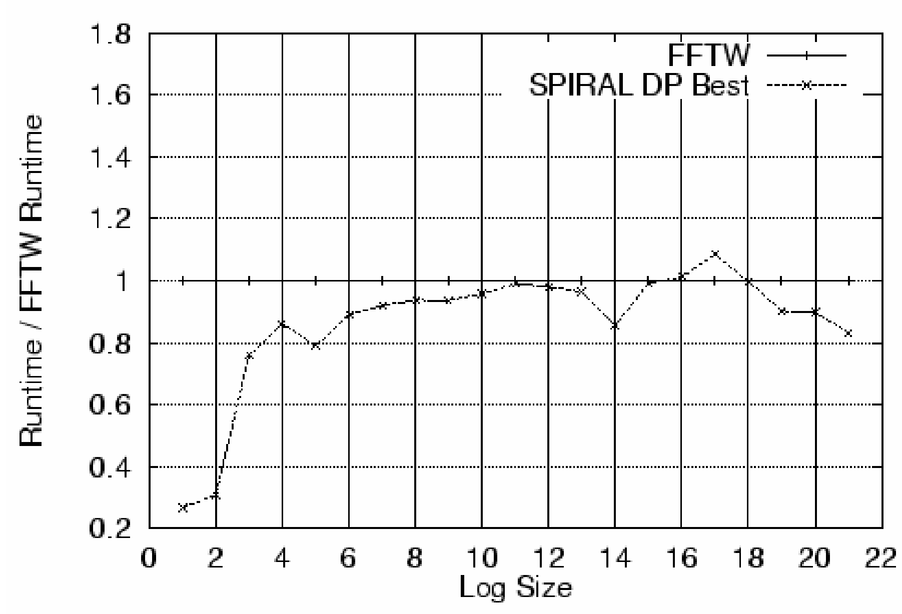
Comparison Search Methods II

across transforms of size 16

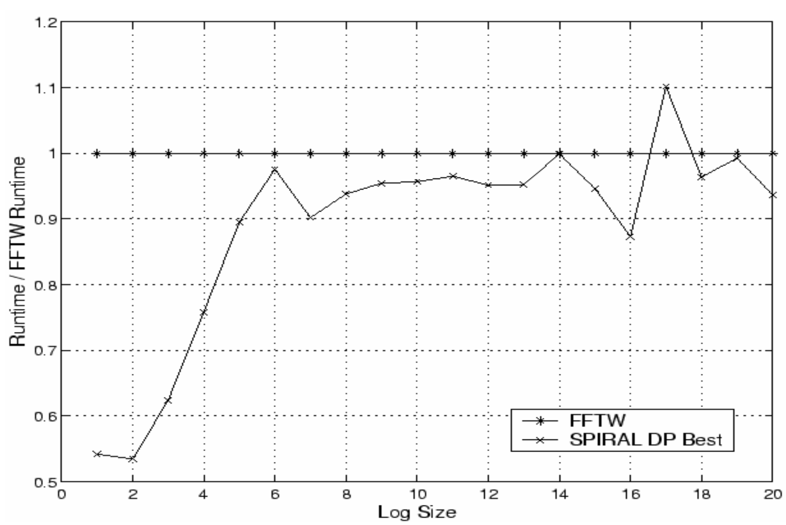




Pentium III/Linux/gcc



Athlon/Linux/gcc



Pentium III/Win2000/Intel compiler

SPIRAL vs. FFTW (lower = better)

comparable performance

SPIRAL

Organization

- SPIRAL approach
- SPIRAL system
- Some experimental results
- **Recent work**

Learning instead of Searching

- **Method:**
  - **Runs a number of formulas of one size**
  - **Analyzes the cache misses caused by different parts of the formulas**
  - **Then design fastest formulas of different sizes, even larger sizes!**
- **Designs fast formulas of sizes that it has never even timed before**
- **Designed fastest known formulas for WHT!**

Fast Formula Generation Results

Size	Number of Formulas Generated	Generated Included the Fastest Known	Top N Fastest Known Formulas in Generated
2^{12}	101	yes	77
2^{13}	86	yes	4
2^{14}	101	yes	70
2^{15}	86	yes	11
2^{16}	101	yes	68
2^{17}	86	yes	15
2^{18}	101	yes	25
2^{19}	86	yes	16
2^{20}	101	yes	16

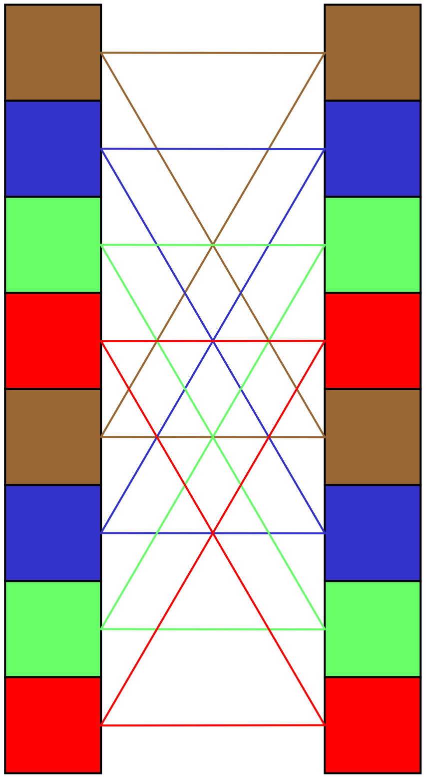
Electrical & Computer ENGINEERING

SPIRAL

SPIRAL SIMD

joint work with
Franz Franchetti, Christoph Űberhuber,
Technical University Vienna

$$DFT_2 \otimes I_4$$



- **Portable SIMD Support (SSE; planned: SSE2, AltiVec),**
  based on Compiler Support
- **Handle $A \otimes I_n$ and $I_n \otimes A$**
- **Support for Diagonals and Permutations**
- **Unrolled code and loop code**

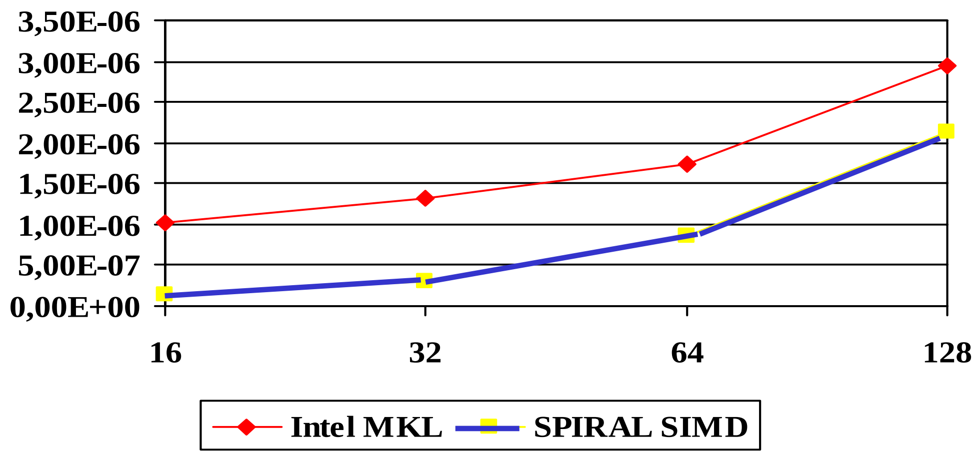
Electrical & Computer ENGINEERING

SPIRAL

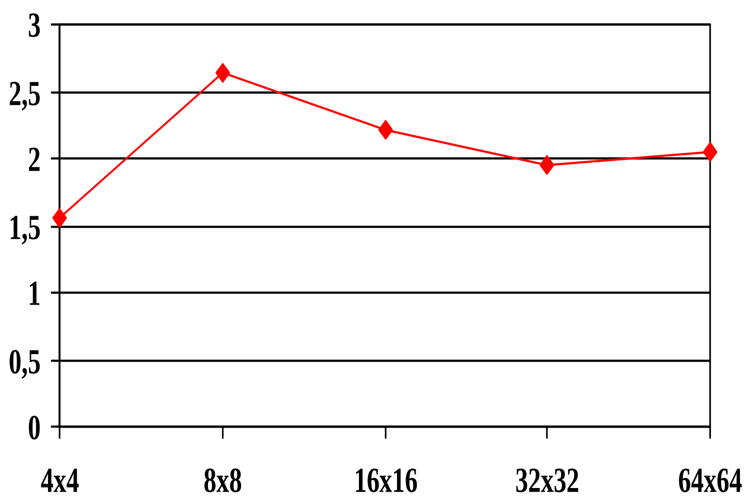
Experimental Results

Pentium4
SSE - float
Windows 2000
Intel C++ Compiler 5.0
Spiral 3.1

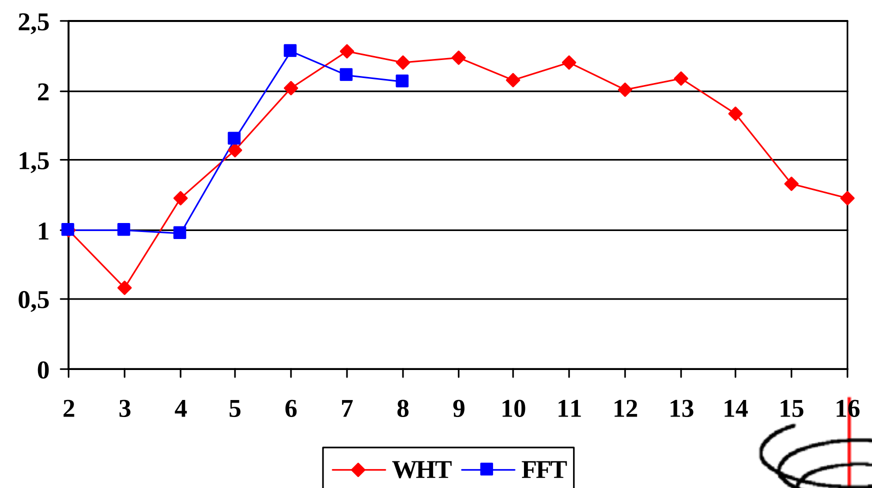
FFT: Benchmark



DCT2xDCT2: Speed-up



Speed-up



Electrical & Computer ENGINEERING

SPIRAL

Summary

- SPIRAL
  - ➢ generates platform-adapted code for linear DSP transforms
  - ➢ is extensible to include new transforms
  - ➢ easily installs on a variety of platforms
- The generated code is verified and very competitive

download at: `www.ece.cmu.edu/~spiral`

Electrical & Computer ENGINEERING

SPIRAL