Algorithms and Computation in Signal Processing

special topic course 18-799B
spring 2005
17th Lecture Mar. 15, 2005

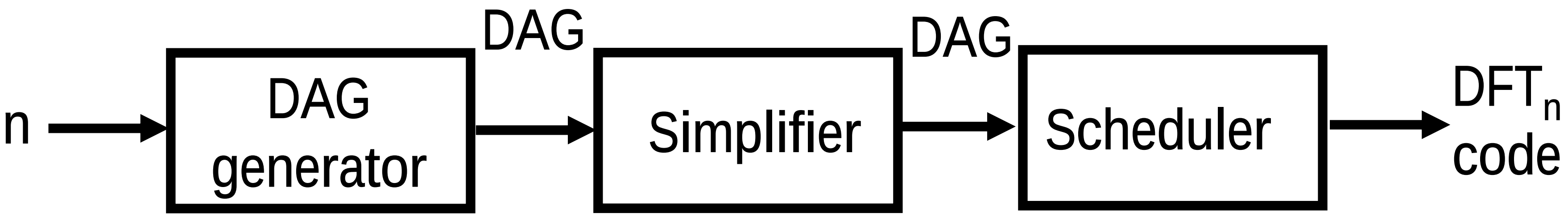
Instructor: Markus Pueschel

TA: Srinivas Chellappa

Please complete the ECE Mid-Semester Course Evaluation and Curriculum Review no later than midnight, Wednesday, March 16

FFT cont’d (First Blackboard)

FFTW Codelet Generator



- DAG generator
  - Generates (deterministically) DFT algorithm represented as DAG
- Simplifier
  - Removes trivial operations
  - Common subexpression elimination
  - Only positive constants
  - …
- Scheduler
  - Orders the DAG into sequential code to minimize register spills

Codelet Examples

- Notwiddle 2
- Notwiddle 3
- Twiddle 3
- Notwiddle 32

- Techniques not seen before:
  - Scoping (variables only defined where they occur)
    Purpose: simplifies dependency analysis
  - Single static assignment (SSA) style: Each variable has only one single definition in the code
    Purpose: no artificial dependencies

Dynamic Programming (DP)

- An algorithmic technique to solve optimization problems
- Definition: DP solves an optimization problem by caching subproblem solutions (memoization) rather than recomputing them
- Well-suited for divide-and-conquer algorithms with a degree of freedom in the divide step
- Inherent assumption: Best solution is independent of the context in which the problem has to be solved

DP for FFTs

- Goal: Find the best recursion strategy for a DFT of size 2^k, computed with the Cooley-Tukey FFT
- Assume the best recursions for sizes $2^1,\ldots,2^{k-1}$ are already computed
- Split DFT 2^k in all k-1 possible ways and use the best recursions for the smaller DFTs.
- The fastest of these k-1 algorithms is the solution for 2^k
- Cost: $(k-1)+(k-2)+\ldots+1 = O(k^2)$ for size 2^k

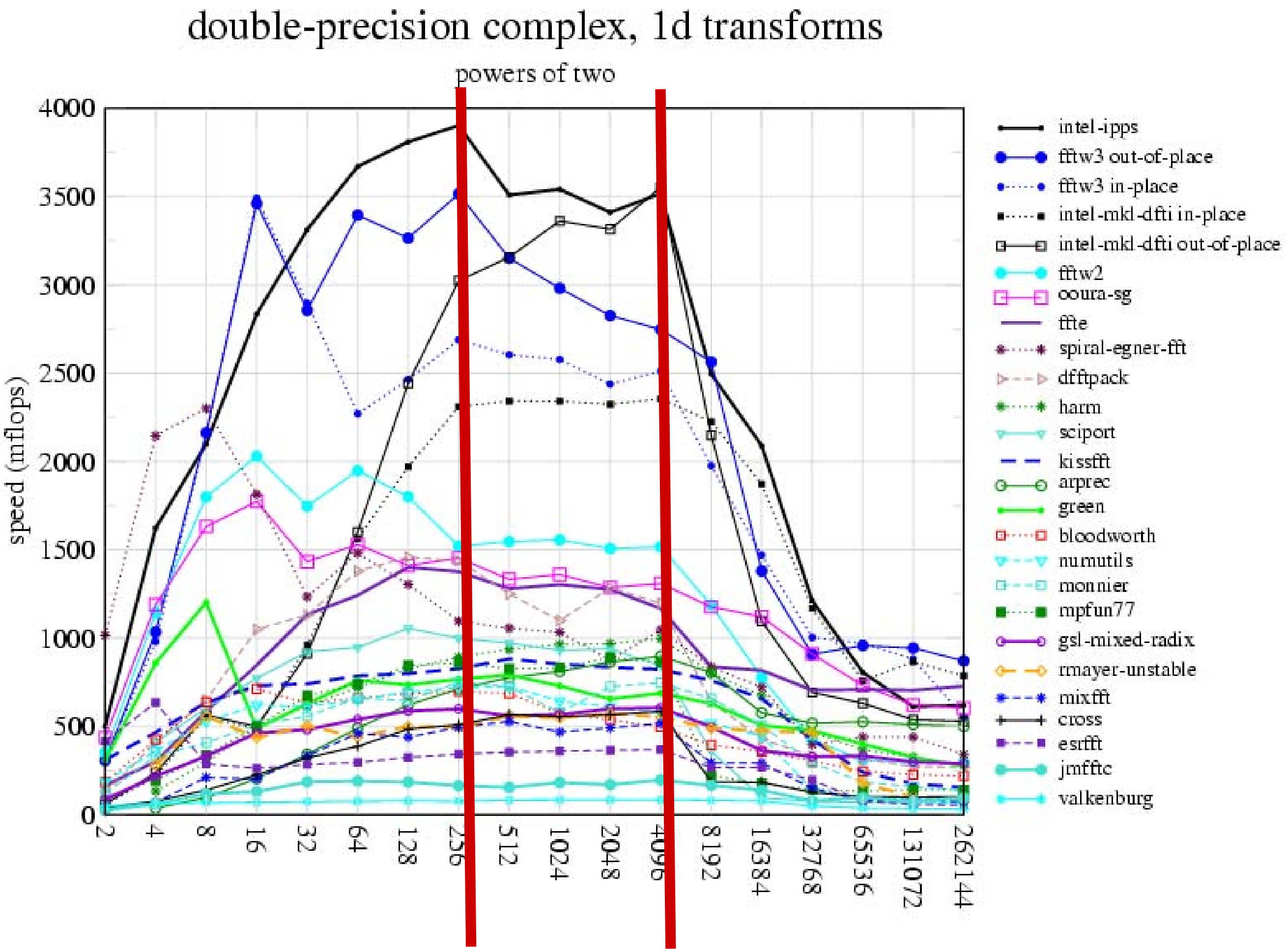
DP for FFTs (cont'd)

- In FFTW: Essentially as described on the previous slide, except left DFT is of size <= 64 (since twiddle codelet)

- Does DP assumption hold for FFTs?
  - Not clear. In particular the best FFT could depend on the stride.
  - But works well in practice and is fast

FFTW Benchmarks, Pentium 4

- Compute cache boundaries (8KB L1, 512KB L2)

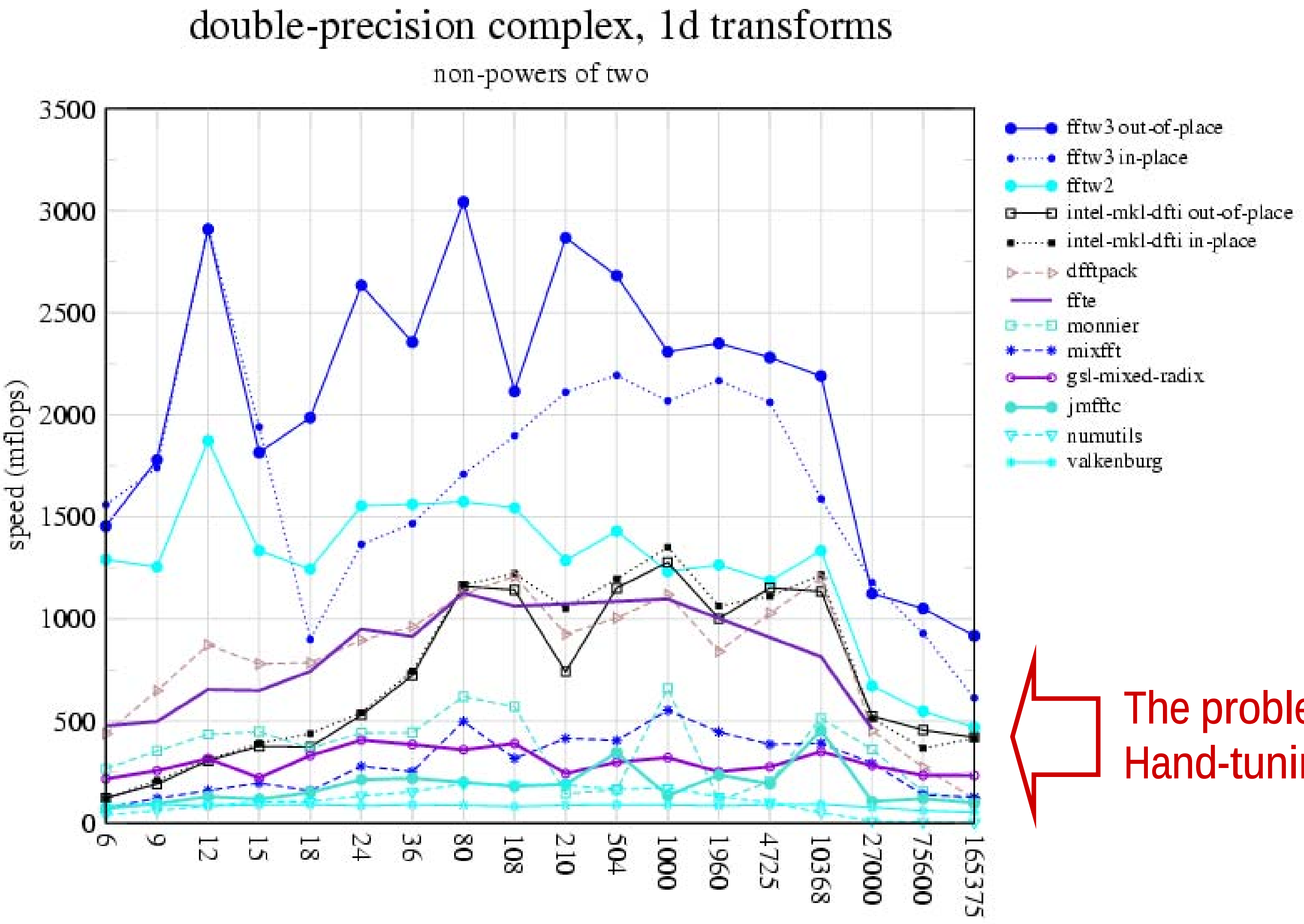


L1 Cache

L2 Cache

Source: www.fftw.org/speed

FFTW Benchmarks, Pentium 4



The problem with Hand-tuning

Source: www.fftw.org/speed

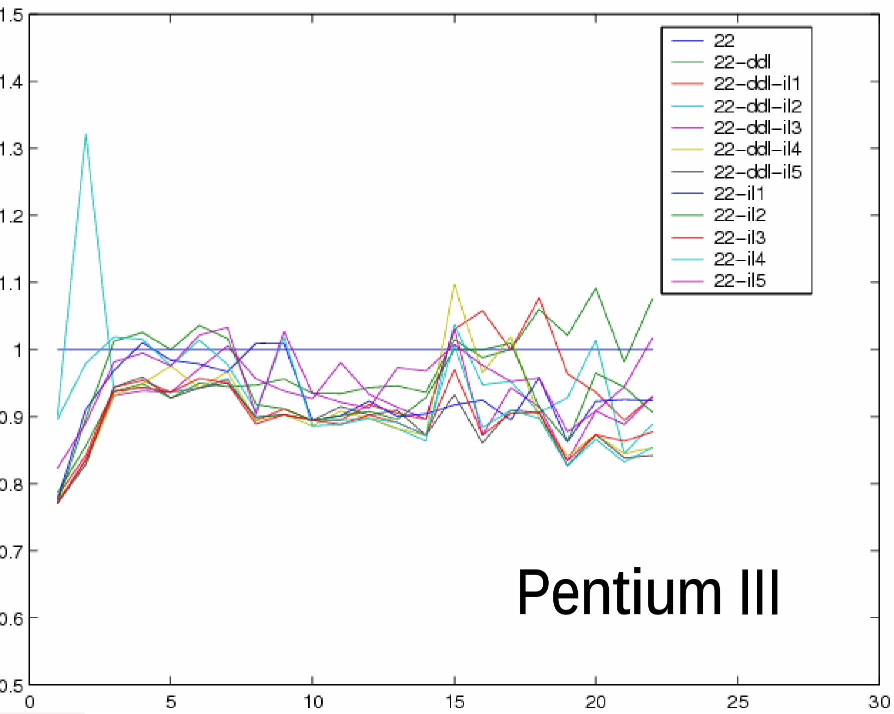
$A \otimes I$ Problem (Blackboard first)

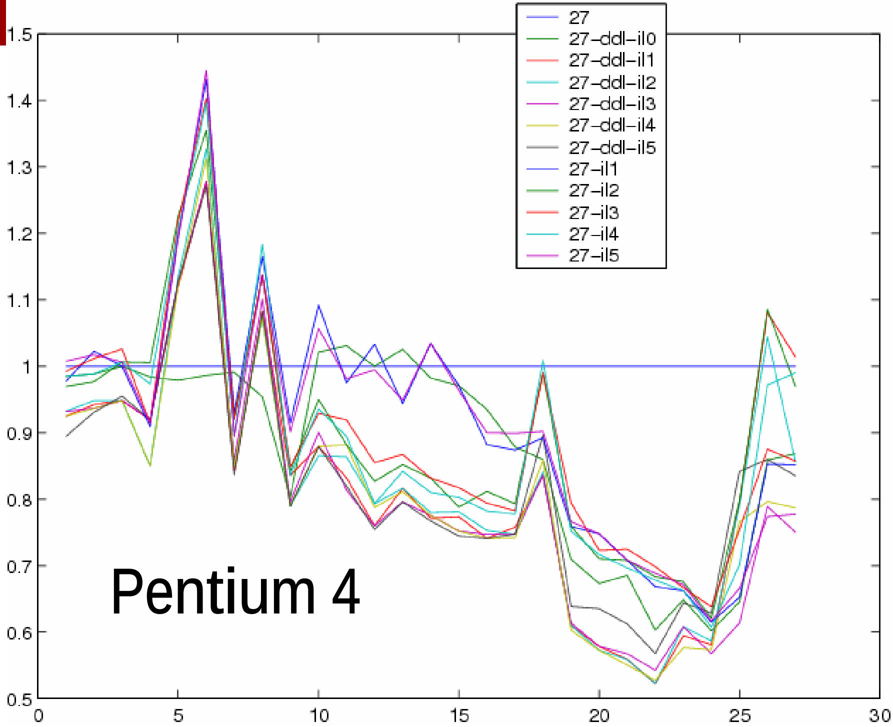
Experiments: $A \otimes I$ Problem

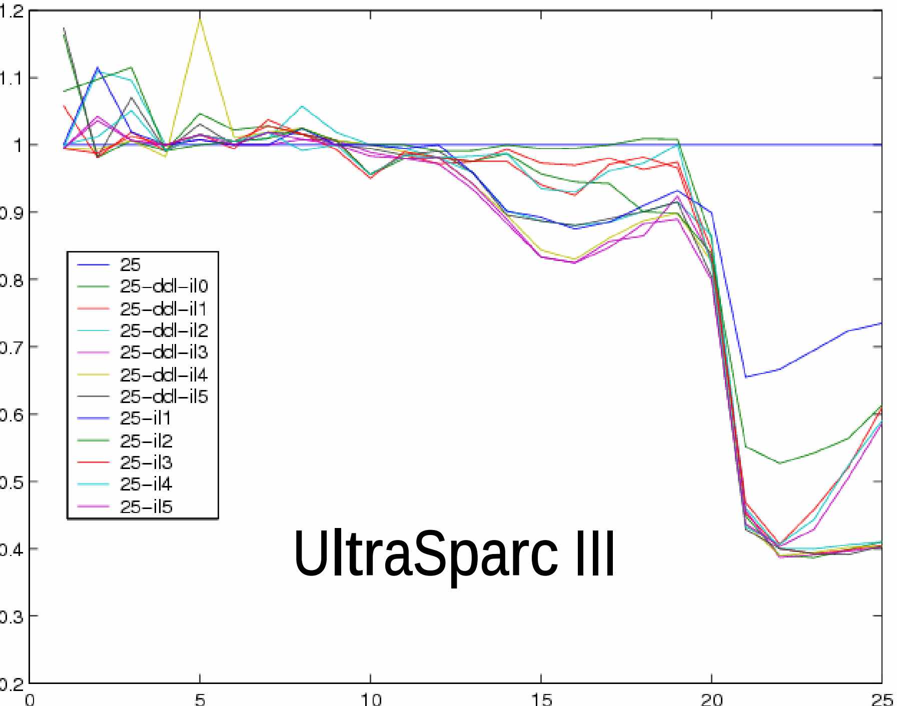
- Setup: WHT with recursion

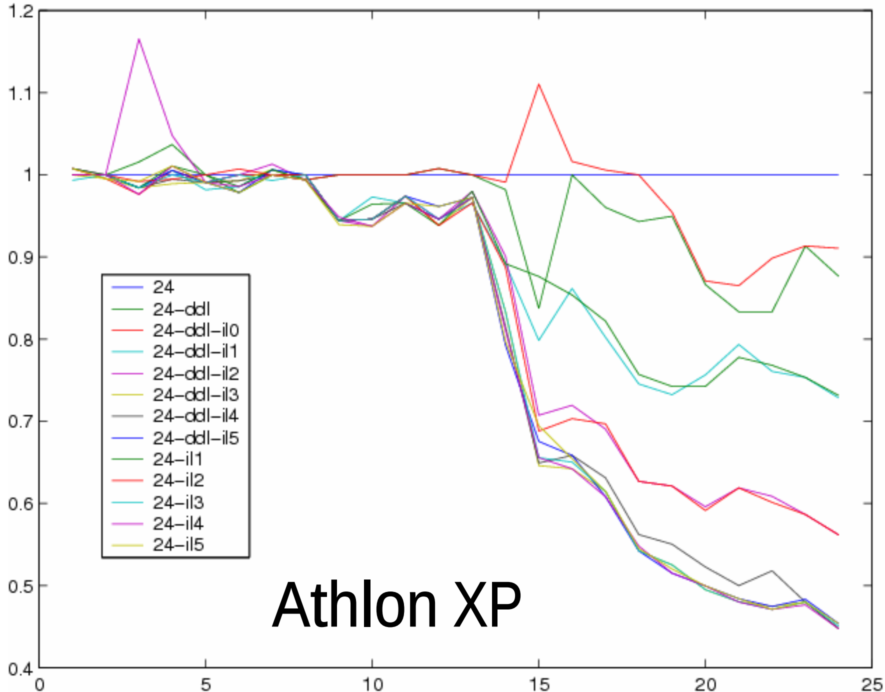
$$WHT_{2^k} = (WHT_{2^{k1}} \otimes I_{2^{k2}})(I_{2^{k1}} \otimes WHT_{2^{k2}})$$

- Find best recursion tree with DP (baseline)
- Find best recursion tree with left factor permuted (ddl)
- Find best recursion tree with left factor interleaved 2^1 – 2^5 times


1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
30
22
22–ddl
22–ddl–il1
22–ddl–il2
22–ddl–il3
22–ddl–il4
22–ddl–il5
22–il1
22–il2
22–il3
22–il4
22–il5
Pentium III


1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
30
27
27–ddl–il0
27–ddl–il1
27–ddl–il2
27–ddl–il3
27–ddl–il4
27–ddl–il5
27–il1
27–il2
27–il3
27–il4
27–il5
Pentium 4


1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
25
25–ddl–il0
25–ddl–il1
25–ddl–il2
25–ddl–il3
25–ddl–il4
25–ddl–il5
25–il1
25–il2
25–il3
25–il4
25–il5
UltraSparc III


1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
24
24–ddl
24–ddl–il0
24–ddl–il1
24–ddl–il2
24–ddl–il3
24–ddl–il4
24–ddl–il5
24–il1
24–il2
24–il3
24–il4
24–il5
Athlon XP