Please complete the ECE Mid-Semester Course Evaluation and Curriculum Review no later than midnight, Wednesday, March 16
FFT cont’d (First Blackboard)
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
FFT W Benchmarks, Pentium 4

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

Source: www.fftw.org/speed
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

$$WHT_{2^k} = (WHT_{2^{k_1}} \otimes I_{2^{k_2}})(I_{2^{k_1}} \otimes WHT_{2^{k_2}})$$

- 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