18-799 Algorithms and Computation in Signal Processing
Spring 2005
Assignment 4
Due Date: March 31st 1:30pm (before class)

General instructions:

- For all implementation exercises you have to submit the working code that goes along with the experiments.
- All other requested results (plots, tables, other results, text) submit in one word, ps, or pdf file. Make sure you answer all the questions posed.
- For plots/benchmarks **follow the guidelines presented in lecture 5** where applicable (no need for huge amounts of text, but provide necessary information and always briefly discuss the plot and draw a conclusion).
- When compiling use good optimization flags.

Exercises:

The Walsh-Hadamard transform (WHT) is a discrete signal transform related to the DFT but with simpler structure and simpler algorithms. It is used in signal processing and in communications.

The WHT is defined only for 2-power input sizes $n = 2^k$, and given by the matrix

$$\mathrm{WHT}_{2^k} = \underbrace{\mathrm{DFT}_2 \otimes \mathrm{DFT}_2 \otimes \ldots \otimes \mathrm{DFT}_2}_{k \text{ factors}}.$$

1. *(15 pts)* How many entries of the WHT are zeros and why? What is the cost $C_{2^k} = (A_{2^k}, M_{2^k})$ (number of additions, number of multiplications), when computing the WHT by definition?

2. *(40 pts)* The WHT can of an input vector can be computed iteratively or recursively using the following formulas:

$$\begin{aligned} \mathrm{WHT}_{2^k} &= \prod_{i=0}^{k-1} (\mathrm{I}_{2^{k-i-1}} \otimes \mathrm{DFT}_2 \otimes \mathrm{I}_{2^i}) \quad \text{(iterative)} \\ \mathrm{WHT}_{2^k} &= (\mathrm{DFT}_2 \otimes \mathrm{I}_{2^{k-1}})(\mathrm{I}_2 \otimes \mathrm{WHT}_{2^{k-1}}) \quad \text{(recursive)} \end{aligned}$$

   (a) Determine the exact arithmetic cost of both algorithms (again number of additions and multiplication separately). What is the peak performance you can theoretically achieve with the WHT on your computer and why?

   (b) Implement both algorithms. The iterative one as a triple loop; the recursive one uses recursive calls until the base case DFT_2 is reached, which is computed by definition. Create a performance plot (mflops versus size) for sizes 2^1–2^{20}. Discuss the plot.

3. *(45 pts)* Create unrolled "codelets" for the WHTs of sizes 4 and 8 based on the recursive algorithm above. (The number of operations in them should match the computed cost in 2(a).)

   Now implement recursive radix-4 and radix-8 implementations of the WHT based on the formulas

$$\begin{aligned} \mathrm{WHT}_{2^k} &= (\mathrm{WHT}_4 \otimes \mathrm{I}_{2^{k-2}})(\mathrm{I}_4 \otimes \mathrm{WHT}_{2^{k-2}}) \quad \text{(radix-4)} \\ \mathrm{WHT}_{2^k} &= (\mathrm{WHT}_8 \otimes \mathrm{I}_{2^{k-3}})(\mathrm{I}_8 \otimes \mathrm{WHT}_{2^{k-3}}) \quad \text{(radix-8)} \end{aligned}$$

   In these implementation, the left WHT (of size 4 or 8) should be your unrolled codelet (which then has to handle strided input data, the right one a recursive call. Both tensor products translate into loops. Further, in both implementations, you may need one step with a different radix to handle all input sizes.

   Measure the performance of both implementations, again for 2^1–2^{20} and add it to the previous plot (4 lines total). Discuss the plot.

4. *(up to 20 extra pts)* Try to further improve the code or do interesting experiments. For example, you can use a dynamic programming approach to find the best recursion (radix-2, -4, or -8) in each step independently. Or, what happens if one considers more general algorithms based on

$$\mathrm{WHT}_{2^k} = (\mathrm{WHT}_{2^i} \otimes \mathrm{I}_{2^{k-i}})(\mathrm{I}_{2^i} \otimes \mathrm{WHT}_{2^{k-i}}).$$