ECE4703 Final Exam

Your Name: SOLUTION

Your box #: _________

December 18, 2008

Tips:

• Look over all of the questions before starting.
• Budget your time to allow yourself enough time to work on each question.
• Write neatly and show your work!
• This exam is worth a total of 200 points.
• Attach your “cheat sheet” to the exam when you hand it in.

<table>
<thead>
<tr>
<th></th>
<th>problem 1</th>
<th>problem 2</th>
<th>problem 3</th>
<th>problem 4</th>
<th>total exam score</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>50</td>
<td>60</td>
<td>50</td>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>
1. 40 points. Suppose you write a frame-based DSP program that calculates a result on a buffer of \( N \) input samples. After profiling the code for various values of \( N \), you determine that the number of cycles to compute the matrix inverse follows the trend

\[
\text{cycles} = 100 + \frac{N^2}{2}.
\]

If your sampling rate is \( f_s = 8000 \text{Hz} \), your DSP clock rate is 225MHz, and all processing is performed on a frame-by-frame basis, how large can \( N \) be before your program will no longer run in real-time? Explain your answer.

\[
T_{\text{calc}}(N) = \frac{\text{cycles}}{225 \times 10^6} = \frac{100 + \frac{N^2}{2}}{225 \times 10^6} \text{ seconds}
\]

\[
T_{\text{avail}}(N) = \frac{N}{8 \times 10^2} \text{ seconds (frame-based processing)}
\]

We want to know when \( T_{\text{calc}}(N) = T_{\text{avail}}(N) \)

\[
\frac{100 + \frac{N^2}{2}}{225 \times 10^6} = \frac{N}{8 \times 10^2}
\]

\[
8 \times 10^5 + (4 \times 10^3)N^2 - (225 \times 10^6)N = 0
\]

divide through by \( 4 \times 10^3 \)

\[
N^2 - (56.25 \times 10^3)N + 200 = 0
\]

quadratic formula ... roots = \[
\frac{-b \pm \sqrt{b^2-4ac}}{2a}
\]

roots = \[
\frac{56.25 \times 10^3 \pm \sqrt{(56.25 \times 10^3)^2 - 800}}{2}
\]

We have one root very close to zero and the more interesting root is at \( N = 56250 \)

check: \( T_{\text{calc}}(56250) = 7.03 \text{ seconds} \)

\( T_{\text{avail}}(56250) = 7.03 \text{ seconds} \).
2. 50 points total. Suppose you wish to compute the 4-point FFT of the array

\[ x = \{x[0], x[1], x[2], x[3]\} . \]

The output of the FFT is denoted as

\[ X = \text{FFT}(x) = \{X[0], X[1], X[2], X[3]\} . \]

As shown in the figure below, you know that you need to call a 2-point FFT function twice in order to compute the 4-point FFT. Denote the output of the first 2-point FFT call as \( Y = \{Y[0], Y[1]\} \) and the output of the second 2-point FFT call as \( Z = \{Z[0], Z[1]\} \).

Given

\[ \begin{align*}
x[0] &= a \\
x[1] &= b \\
x[2] &= c \\
x[3] &= d,
\end{align*} \]

compute \( Y[0], Y[1], Z[0], Z[1], X[0], X[1], X[2], \) and \( X[3] \). Show your work and explain your reasoning.

**First two-point FFT:**

\[ \begin{align*}
x[0] &= a + c & Y[0] &= a + c & \leftarrow x_{\text{even}}[0], x_{\text{even}}[2] \\
x[2] &= a - c & Y[1] &= a - c & \leftarrow x_{\text{even}}[1], x_{\text{even}}[3]
\end{align*} \]

**Second two-point FFT**

\[ \begin{align*}
x[1] &= b + d & Z[0] &= b + d & \leftarrow x_{\text{odd}}[0], x_{\text{odd}}[2] \\
x[3] &= b - d & Z[1] &= b - d & \leftarrow x_{\text{odd}}[1], x_{\text{odd}}[3]
\end{align*} \]

**Now the 4-point FFT:**

\[ X[k] = \begin{cases} 
2\text{-pt FFT} & X_{\text{even}}[k] \\
X_{\text{odd}}[k] & \end{cases} + e^{-j \frac{2\pi k}{4}} X_{\text{odd}}[k] \]

check using DFT: \[ X[k] = \sum_{n=0}^{3} x[n] e^{-j \frac{2\pi nk}{4}} \]

\[ \begin{align*}
x[0] &= a + b + c + d \\
x[1] &= (a - c) - j(b - d) \\
x[2] &= (a + c) - (b + d) \\
x[3] &= (a - c) + j(b - d) \end{align*} \]
3. 60 points total. Consider the system identification adaptive filtering system shown below.

![Diagram of system identification adaptive filtering system]

For the following questions, assume that

- the mean squared value of the input noise \( x[n] \) is one, i.e. \( E[x^2[n]] = 1 \),
- the unknown filter coefficients are \( h = [0.1, 0.5, 0.3] \),
- the LMS adaptive filter \( b \) coefficients are initialized to zero prior to adaptation, and
- the LMS step-size is small enough to allow for convergence of the algorithm to the minimum mean squared error (MMSE) solution.

(a) 20 points. Suppose that \( D = 0 \) (no delay) and that \( b \) has three coefficients. What will \( b \) be after convergence of the LMS algorithm? What will the MMSE be after convergence?

After convergence, \( b = [0.1, 0.5, 0.3] \) and \( \text{MMSE} \to 0 \)

(b) 20 points. Now suppose \( D = 0 \) (no delay) and that \( b \) has two coefficients. What will \( b \) be after convergence of the LMS algorithm? What will the MMSE be after convergence?

After convergence, \( b = [0.1, 0.5] \). This is the best that \( b \) can approximate \( h \). The match isn't perfect, however, since

\[
\begin{align*}
e[n] &= d[n] - y[n] = (0.1x[n] + 0.5x[n-2] + 0.3x[n-3]) - (0.1x[n] + 0.5x[n-2]) \\
&= 0.3x[n-3]
\end{align*}
\]

The MMSE in this case is then \( E[e^2[n]] = 0.09 \) \( E[x^2[n]] = 0.09 \).

(c) 20 points. For the case when \( b \) has two coefficients, find the value of \( D \) that leads to the lowest possible MMSE. For this value of \( D \), what will \( b \) be after convergence of the LMS algorithm? For this value of \( D \), what will the MMSE be after convergence?

If we let \( D = 1 \), then \( b = [0.5, 0.3] \) after convergence.

The MMSE in this case can be calculated from the error

\[
e[n] = d[n] - y[n] = 0.1x[n]
\]

\( \Rightarrow \text{MMSE} = 0.01 \) (an improvement of 9x).
4. 50 points total. Suppose your DSP is running the assembly code given on the last page of this exam (taken from the Kehtarnavaz examples).

(a) 10 pts. Draw a box around the instruction(s) in the third fetch packet. Label it FP3.

(b) 10 pts. Draw a box around the instruction(s) in the third execute packet. Label it EP3.

(c) 10 points. Suppose the SHR instruction on line 17 is currently in pipeline stage E1. Put a pound sign (#) next to the instruction(s) currently in pipeline stage DP.

(d) 10 points. Including the 5 NOP cycles at the end of the listing, how many cycles does this code require to execute?

$$20 \text{ cycles}$$

(e) 10 points. Suppose the first LDW instruction results in A5=11, the second LDW instruction results in A5=12, and the third LDW instruction results in A5=13. Similarly, suppose the first LDH instruction results in B5=1, the second LDH instruction results in B5=2, and the third LDH instruction results in B5=3 (all values are decimal). Compute the results of the first and second MPY instructions (lines 13 and 14). Explain your answer.

First multiply = \( 1 \times 11 = 11 \) (5 cycles after first LDW/LDH)

Second multiply = \( 2 \times 12 = 24 \) (5 cycles after second LDW/LDH)

(LDW & LDH have 4 delay slots).
/* This code is written by N. Kehtarnavaz and N. Kim as part of the textbook "Real-Time Digital Signal Processing Based on TMS320C6000". */

.global _iir
.sect "iir"

.iir:
    .b 10
    .a 5
    .d 2
    .l 1
    .s 1
    .m 1
    .n 1
    .k 1
    .d 2
    .l 1
    .s 2
    .m 2
    .n 1
    .k 1

/* Simple iir filter implementation */

; EP1
    zero .s1 a10 ; BSUM
    ldw .d1 *a4++,a5 ; Load input sample (A5=11)
    ldh .d2 *b4++,b5 ; Load b coefficient (B5=1)
    ldw .d1 *a4++,a5 ; Load input sample (A5=12)
    ldh .d2 *b4++,b5 ; Load b coefficient (B5=2)

; EP2
    ldw .d1 *a4++,a5 ; Load input sample (A5=13)
    ldh .d2 *b4++,b5 ; Load b coefficient (B5=3)

; EP3
    ldw .d1 ***a6,a7 ; Load output sample
    ldh .d2 ***b6,b7 ; Load a coefficient
    ldw .d1 ***a6,a7 ; Load output sample
    ldh .d2 ***b6,b7 ; Load a coefficient

; EP4
    mpy .m1x a5, b5, a8 ; b * input
    mpy .m1x a5, b5, a8 ; b * input
    shr .s1 a8, 15, a9 ; Shift right
    mpy .m1x a5, b5, a8 ; b * input

; EP5
    shr .s1 a8, 15, a9 ; Shift right
    add .l1 a9, a10, a10 ; Add
    mpy .m2x a7, b7, b8 ; a * output
    shr .s1 a8, 15, a9 ; Shift right
    add .l1 a9, a10, a10 ; Add
    mpy .m2x a7, b7, b8 ; a * output

; DC
    add .l1 a9, a10, a10 ; Add

; DD
    add .l2 b2, b10, b10 ; Add
    sub .l1 a10, b10, a4 ; BSUM - ASUM