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.

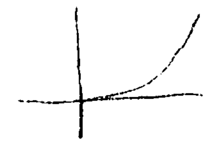
problem 1	problem 2	problem 3	problem 4	total nal exam score
40 points	50 points	60 points	50 points	200 points

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

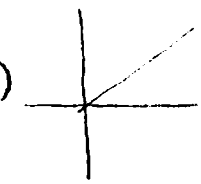
$$\text{cycles} = 100 + \frac{N^2}{2}.$$

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.

$$T_{calc}(N) = \frac{\text{cycles}}{225\times10^6} = \frac{100 + \frac{N^2}{2}}{225\times10^6} \text{ seconds}$$



$$T_{avail}(N) = \frac{N}{8\times10^3} \text{ seconds} \quad \text{(frame-based processing)}$$



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

$$\frac{100 + \frac{N^2}{2}}{225\times10^6} = \frac{N}{8\times10^3}$$

$$8\times10^5 + (4\times10^3)N^2 - (225\times10^6)N = 0$$

divide through by 4×10^3

$$N^2 - (56.25\times10^3)N + 200 = 0$$

quadratic formula ... roots $= \dfrac{-b \pm \sqrt{b^2-4ac}}{2a}$

$$\text{roots} = \frac{56.25\times10^3 \pm \sqrt{(56.25\times10^3)^2 - 800}}{2}$$

We have one root very close to zero and the more interesting root is at $\boxed{N = 56250}$

check: $T_{calc}(56250) = 7.03$ seconds ✓

$T_{avail}(56250) = 7.03$ seconds.

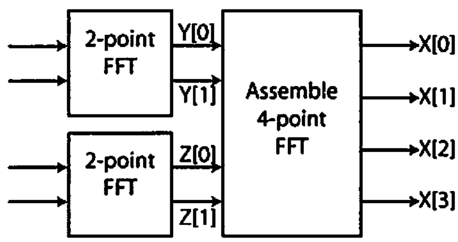
2. 50 points total. Suppose you wish to compute the 4-point FFT of the array

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

The output of the FFT is denoted as

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

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 $\boldsymbol{Y} = \{Y[0], Y[1]\}$ and the output of the second 2-point FFT call as $\boldsymbol{Z} = \{Z[0], Z[1]\}$.



Given

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

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.

$x[0]$ — $Y[0] = a+c \leftarrow X_{even}[0],\ X_{even}[2]$

$x[2]$ — $Y[1] = a-c \leftarrow X_{even}[1],\ X_{even}[3]$

Second two-point FFT

$x[1]$ — $Z[0] = b+d \leftarrow X_{odd}[0],\ X_{odd}[2]$

$x[3]$ — $Z[1] = b-d \leftarrow X_{odd}[1],\ X_{odd}[3]$

Now the 4-pt FFT:

$$X[k] = \underbrace{X_{even}[k]}_{\text{2 point FFT}} + e^{-j\frac{2\pi k}{4}} \underbrace{X_{odd}[k]}_{\text{2 point FFT}}$$

$$\boxed{\begin{aligned} X[0] &= a+b+c+d. \\ X[1] &= (a-c) - j(b-d) \\ X[2] &= (a+c) - (b+d) \\ X[3] &= (a-c) + j(b-d) \end{aligned}}$$

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

$X[0] = a+b+c+d.$

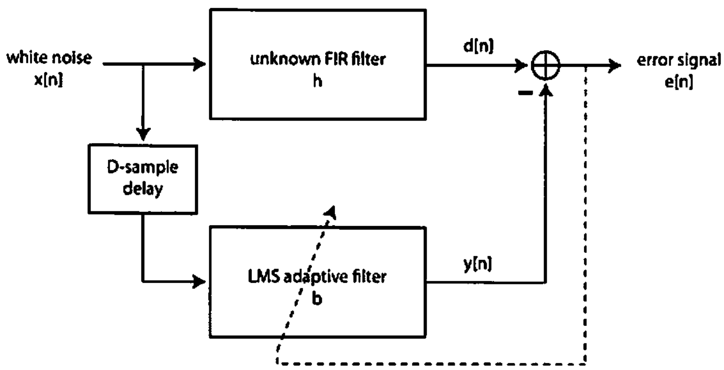
$X[1] = a - jb - c + jd = (a-c) - j(b-d)$

$X[2] = a - b + c - d = (a+c) - (b+d)$

$X[3] = a + jb - c - jd = (a-c) + j(b-d)$

✓

3. 60 points total. Consider the system identification adaptive filtering system shown below.



For the following questions, assume that

- the mean squared value of the input noise $x[n]$ is one, i.e. $\mathrm{E}\left[x^2[n]\right] = 1$,
- the unknown filter coefficients are $\boldsymbol{h} = [0.1, 0.5, 0.3]$,
- the LMS adaptive filter $\boldsymbol{b}$ 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 $\boldsymbol{b}$ has three coefficients. What will $\boldsymbol{b}$ be after convergence of the LMS algorithm? What will the MMSE be after convergence?

after convergence, $b = [0.1, 0.5, 0.3]$ and MMSE $\rightarrow 0$

(b) 20 points. Now suppose $D = 0$ (no delay) and that $\boldsymbol{b}$ has two coefficients. What will $\boldsymbol{b}$ 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

$e[n] = d[n] - y[n] = (0.1\,x[n] + 0.5\,x[n-2] + 0.3\,x[n-3]) - (0.1\,x[n] + 0.5\,x[n-2])$
$= 0.3\,x[n-3]$

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 $\boldsymbol{b}$ has two coefficients, find the value of D that leads to the lowest possible MMSE. For this value of D, what will $\boldsymbol{b}$ 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
$e[n] = d[n] - y[n] = 0.1\,x[n]$
$\Rightarrow$ MMSE $= 0.01$ (an impromant 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 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            ; Simple iir filter implementation
        .sect   ".iir"

_iir:
                ZERO    .S1     A10             ; BSUM
EP1          || ZERO    .S2     B10             ; ASUM
             || LDW     .D1     *A4++,A5        ; Load input sample (A5=11)
             || LDH     .D2     *B4++,B5        ; Load b coefficient (B5=1)        FP1
                LDW     .D1     *A4++,A5        ; Load input sample (A5=12)
EP2          || LDH     .D2     *B4++,B5        ; Load b coefficient (B5=2)
                LDW     .D1     *A4++,A5        ; Load input sample (A5=13)
EP3          || LDH     .D2     *B4++,B5        ; Load b coefficient (B5=3)
                LDW     .D1     *++A6,A7        ; Load output sample
EP4          || LDH     .D2     *++B6,B7        ; Load a coefficient
                LDW     .D1     *++A6,A7        ; Load output sample
EP5          || LDH     .D2     *++B6,B7        ; Load a coefficient               FP2
                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
                SHR     .S1     A8, 15, A9      ; Shift right
E1 ->        || ADD     .L1     A9, A10, A10    ; Add                              FP3
             || MPY     .M2x    A7, B7, B8      ; a * output
                SHR     .S1     A8, 15, A9      ; Shift right
DC->         || ADD     .L1     A9, A10, A10    ; Add
             || MPY     .M2x    A7, B7, B8      ; a * output
DP ->          #ADD     .L1     A9, A10, A10    ; Add
             ||#SHR     .S2     B8, 15, B9      ; Shift right
                SHR     .S2     B8, 15, B9      ; Shift right
             || ADD     .L2     B9, B10, B10    ; Add
                ADD     .L2     B9, B10, B10    ; Add
                SUB     .L1     A10, B10, A4    ; BSUM - ASUM
                B       .S2     B3
                NOP     5
```