ECE4703 Final Exam

Your Name: SOLUTION Your box #: _____

December 17, 2009

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
30 points	60 points	60 points	50 points	200 points

1. 30 points. Determine the theoretical maximum number of single-precision floating point multiply accumulates that a TMS320C6713 running at 225MHz can compute per second. Explain your answer. Are there DSP applications in which it is possible to get close to this theoretical maximum?

We saw an example in class of a fully-pipelined single-precision-floating point dot product with two multiplies and two accumulates in each clock cycle. Hence, the theoretical maximum MACs for the C6713 is 450,000,000 MACs/sec.

Any DSP application that requires a long dot product will get close to this theoretical maximum since the prologue/epilogue overhead will be small.

2. 60 points total. Suppose you are given two N-point discrete-time sequences

$$x[0], \ldots, x[N-1] \text{ and } y[0], \ldots, y[N-1]$$

and you want to convolve these sequences to form the $2N-1$ sample output sequence $z[0], \ldots, z[2N-2]$.

(a) 30 points. What is the asymptotic complexity of *directly convolving* these two sequences together using the discrete-time convolution formula

$$z[n] = \sum_{\ell=0}^{N-1} x[\ell] y[n-\ell] \text{ for } n = 0, 1, \ldots, 2N-2$$

For each n, we have N multiplies and $N-1$ adds.
Hence there are $(2N-1)N = 2N^2 - N$ multiplies
and $(2N-1)(N-1) = 2N^2 - 3N + 1$ adds

$\Rightarrow$ asymptotic complexity is $\Theta(N^2)$

(b) 30 points. Suppose that, instead of directly convolving these sequences, you instead

- zero-pad the x and y sequences to make them each $2N-1$ samples long,
- compute the FFT of each sequence to get $X[0], \ldots, X[2N-2]$ and $Y[0], \ldots, Y[2N-2]$
- compute the product of these new sequences in the frequency domain such that $Z[k] = X[k]Y[k]$ for $k = 0, 1, \ldots, 2N-2$,
- and then compute the inverse FFT of the sequence $Z[0], \ldots, Z[2N-2]$ to get the time-domain output sequence $z[0], \ldots, z[2N-2]$.

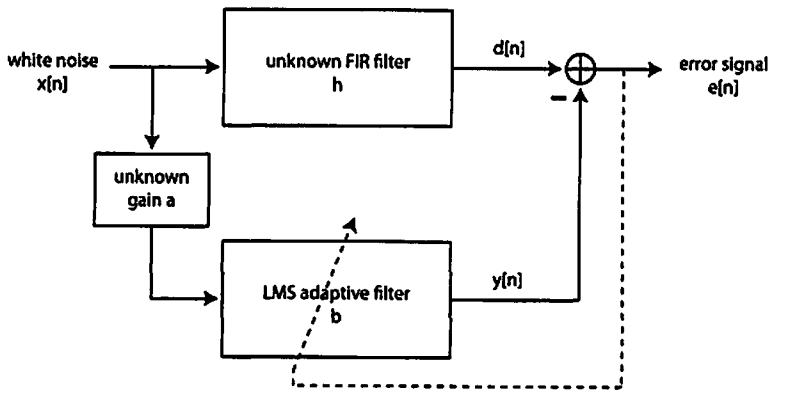
Assuming the zero-padding operation requires a constant number of operations (not dependent on N), what is the asymptotic complexity of performing the convolution using this approach? Hint: The inverse FFT has the same asymptotic complexity as the FFT.

- zero pad: $\Theta(1)$
- FFT(x): $\Theta(N \log_2 N)$
- FFT(y): $\Theta(N \log_2 N)$
- $Z = X .* Y$: $\Theta(N)$
- z = IFFT(Z): $\Theta(N \log_2 N)$

} worst complexity is $\Theta(N \log_2 N)$

$\Rightarrow$ overall asymptotic complexity is $\Theta(N \log_2 N)$

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 $\boldsymbol{h} = [0.2, 0.6]$,
- 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 the unknown gain $a = 1$ and that the adaptive FIR filter $\boldsymbol{b}$ has two coefficients. What will $\boldsymbol{b}$ be after convergence of the LMS algorithm? What will the MMSE be after convergence?

$b \rightarrow [0.2, 0.6]$

MMSE $\rightarrow 0$ because $b = h$

(b) 20 points. Now suppose $a = 2$. What will $\boldsymbol{b}$ be after convergence of the LMS algorithm? What will the MMSE be after convergence?

$b \rightarrow [0.1, 0.3]$ Note that $a \cdot b = [0.2, 0.6]$, which matches h.

MMSE $\rightarrow 0$

(c) 20 points. Now suppose $a = 1$ and that the adaptive FIR filter $\boldsymbol{b}$ has **three** coefficients. What will $\boldsymbol{b}$ be after convergence of the LMS algorithm? What will the MMSE be after convergence?

$b \rightarrow [0.2, 0.6, 0]$

MMSE $\rightarrow 0$ because b matches h in the first two coefficients and is zero in the third coefficient

4. 50 points total. Suppose your DSP is running the assembly code given on the last page of this exam (this is the fully-pipelined dot product discussed in lecture).

   (a) 10 pts. Draw a box around the instruction(s) in the fourth execute packet. Label it EP4.

   (b) 10 points. Suppose the LDDW instruction on line 28 is currently in pipeline stage E3. Put a pound sign (#) next to the instruction(s) currently in pipeline stage E1.

   (c) 10 points. Notice the two MPYSP instructions on lines 18-19. Put stars ($\star$) next to the instruction(s) in the execute packet that run immediately after the results from these MPYSP commands become valid.

   (d) 20 points. Suppose the register A6 contains the value 1000 at the start of this function. How many cycles does this function require to execute? You should count all cycles up to (but not including) the branch instruction at line 59.

Prologue = 9
Loop = 1 × 500
Epilogue = 9

518 total cycles

```
-3                              .def    _dotproduct4
-2
-1  _dotproduct4:
 0                              MV      .S1     A6, A1          ; copy count to register A1
 1  ||                          ZERO    .L1     A8              ; zero even accumulator
 2  ||                          ZERO    .L2     B8              ; zero odd accumulator
 3  ||                          LDDW    .D1     *A4++, A7:A6    ; get element from first array
 4  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
 5                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
 6  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
 7                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
 8  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
 9                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
10  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
11  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
12                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
13  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
14  ||      [A1]    B           .S2     LOOP            ; conditional branch
15  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
16                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
17  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
18  ||                          MPYSP   .M1x    A6, B6, A5      ; multiply
19  ||                          MPYSP   .M2x    A7, B7, B5      ; multiply
20  ||      [A1]    B           .S2     LOOP            ; conditional branch
21  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
22                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
23  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
24  ||                          MPYSP   .M1x    A6, B6, A5      ; multiply
25  ||                          MPYSP   .M2x    A7, B7, B5      ; multiply
26  ||      [A1]    B           .S2     LOOP            ; conditional branch
27  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
28                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
29  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
30  ||                          MPYSP   .M1x    A6, B6, A5      ; multiply
31  ||                          MPYSP   .M2x    A7, B7, B5      ; multiply
32  ||      [A1]    B           .S2     LOOP            ; conditional branch
33  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
34                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
35  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
36  ||                          MPYSP   .M1x    A6, B6, A5      ; multiply
37  ||                          MPYSP   .M2x    A7, B7, B5      ; multiply
38  ||      [A1]    B           .S2     LOOP            ; conditional branch
39  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
40
41  LOOP:
42                              LDDW    .D1     *A4++, A7:A6    ; get element from first array
43  ||                          LDDW    .D2     *B4++, B7:B6    ; get element from second array
44  ||                          MPYSP   .M1x    A6, B6, A5      ; multiply
45  ||                          MPYSP   .M2x    A7, B7, B5      ; multiply
46  ||                          ADDSP   .L1     A5, A8, A8      ; accumulate
47  ||                          ADDSP   .L2     B5, B8, B8      ; accumulate
48  ||      [A1]    B           .S2     LOOP            ; conditional branch
49  ||      [A1]    SUB         .S1     A1, 2, A1       ; decrement counter (NOTE DECREMENT BY 2)
50                              ADDSP   .L1x    A8, B8, A0
51                              ADDSP   .L2x    A8, B8, B0
52                              ADDSP   .L1x    A8, B8, A0
```

```
53                      ADDSP   .L2x    A8, B8, B0
54                      NOP
55                      ADDSP   .L1x    A0, B0, A5
56                      NOP
57                      ADDSP   .L2x    A0, B0, B5
58                      NOP
59                      B               B3              ; branch back to calling function
60                      NOP
61                      ADDSP   .L1x    A5, B5, A4      ; result returned in A4
62                      NOP             3
63                      .end
```