ECE4703 Midterm Exam

Your Name: SOLUTION Your box #: _____

November 17, 2011

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 midterm exam score
35 points	60 points	45 points	60 points	200 points

1. 35 points. Suppose you have a real-time DSP system running a floating point FIR filter on a 225MHz C6713 processor at a sampling frequency of 44.1kHz. You profile your ISR (on the cycle accurate simulator, of course) and find that it is taking 6000 cycles to perform the necessary processing. Show that your ISR is not running in real time and suggest at least two ways you might be able to get your ISR to run in real time.

6000 cycles = $\frac{6000}{225 \times 10^6} = 26.67\,\mu s$ } processing time

one sampling period = $\frac{1}{44100} = 22.68\,\mu s$ } allowed time

The processing time exceeds the allowed time, hence we are not running in real time.

Some ways to get this running in real time:

1. optimize the code, e.g. turn on optimization
2. Lower the sampling frequency to 32kHz
   ⇒ This gives an allowed time of 31.25 μs, which would be enough
3. Change to an IIR filter, since these usually require less computation
4. Switch to fixed-point processing since this is typically faster
5. Make sure code and data are in the internal SRAM, no the external SDRAM (which is much slower)

2. 60 points total. Consider the RT-DSP system shown in Figure 1.

Figure 1: RT-DSP system.

(a) 10 points. Suppose suppose $x(t) = \cos(2\pi \cdot 1500t)$, there is no processing, i.e. $y_q[k] = x_q[k]$, and that the ADC and DAC are ideal ($x_q[k] = x(kT_s)$ and the DAC performs ideal sinc reconstruction). Under what condition(s) will $y(t) = x(t)$?

Since the ADC and DAC are ideal, we just need to satisfy the Nyquist condition to get $x(t) = y(t)$.

The highest frequency of the signals in this system is 1500Hz, hence $f_s \geq 3000\text{Hz}$ is necessary and sufficient to ensure $y(t) = x(t)$.

(b) 20 points. Now suppose $y_q[k] = x_q^3[k]$, and the ADC and DAC are ideal as in part (a). Under what condition(s) will $y(t) = x^3(t)$? Hint: Recall $\cos(u)\cos(v) = \frac{1}{2}(\cos(u+v) + \cos(u-v))$.

$x(t) = \cos(\omega t)$ with $\omega = 2\pi \cdot 1000$

$x^3(t) = \frac{1}{2}(\cos(2\omega t) + 1)\cos(\omega t) = \frac{1}{4}(\cos(3\omega t) + \cos(\omega t)) + \frac{1}{2}\cos(\omega t)$

The highest frequency here is 4500Hz. This means we need a sampling rate of $f_s \geq 9000\text{Hz}$ to avoid aliasing. Note that $f_s \geq 3000\text{Hz}$ avoids aliasing at the input, but the subsequent processing will cause aliasing if f_s isn't at least 9000Hz.

(continued...)

(c) 30 points. Now suppose you perform this "signal cubing" processing in floating-point on the right channel of the TMS320C6713 DSK at a sampling rate of $f_s = 8000$Hz. Write the code for how you would do this processing in the ISR template below and discuss how the DSK output will differ from the ideal $y(t) = x^3(t)$ prediction.

```
interrupt void serialPortRcvISR()
{
    union {Uint32 combo; short channel[2];} temp;
    // Your local variable declarations below (if needed)
    float x, y;

    temp.combo = MCBSP_read(DSK6713_AIC23_DATAHANDLE);
    // Note that right channel is in temp.channel[0]
    // Note that left channel is in temp.channel[1]
    // Your signal cubing processing code below
    x = (float) temp.channel[0]; // grab right channel
    x = x / 32768; // scale to [-1, +1)
    y = x * x * x; // cube it; still in range [-1, +1)
    temp.channel[0] = (short) (y * 32768); //scale & cast

    MCBSP_write(DSK6713_AIC23_DATAHANDLE, temp.combo);
}
```

This output will differ from the ideal prediction in a few ways.

1. There will be aliasing if $x(t) = \cos(2\pi \cdot 1500t)$ since cubing the signal will generate a term at 4500Hz. This will alias back to 3500Hz and will appear in the output.
2. Quantization noise in the ADC will be present.
3. There will be some delay caused by the ADC → process → DAC.
4. There could be clipping if the input signal isn't scaled to fit in the full range of the ADC (5.9Vpp).

3. 45 points total.

(a) 15 points. Recall that FIR filters are guaranteed to be stable and can have nice properties like linear phase. Is there ever a reason to use an IIR filter in a RT-DSP system? Explain.

IIR filters usually have less coefficients to meet the same specs as an FIR filter. Hence, IIR filters are very useful when the corresponding FIR filter can't run in real-time.

Also, IIR filters are more appropriate for certain types of filtering, e.g. emulating an analog filter.

(b) 15 points. Recall that the output of an IIR filter can be computed as

$$y[n] = \sum_{k=0}^{N-1} b_k x[n-k] - \sum_{k=1}^{N-1} a_k y[n-k] \qquad \text{(direct form I)}. \tag{1}$$

We can also compute the output of the IIR filter in two steps as

$$u[n] = x[n] - \sum_{k=1}^{N-1} a_k y[n-k] \text{ and } y[n] = \sum_{k=0}^{N-1} b_k u[n-k] \qquad \text{(direct form II)}. \tag{2}$$

Is there any advantage to the second approach?

The main advantage to the second approach (DF-II) is that it only requires one buffer to store the intermediate results. DF-I requires two buffers.

The computational requirements are essentially identical, but the memory requirements of DF-II are approximately half of DF-I

(continued...)

(c) 15 points. Recall that we can factor any IIR filter into "second order sections" by factoring the transfer function $H(z) = H_1(z) \cdots H_s(z)$ where each $H_i(z)$ is a second order transfer function. When and why would we want to do this?

This doesn't reduce memory or computational requirements and typically has no measurable effect on the system if everything is floating point.

DF-II SOS becomes important when we switch to fixed-point processing. In this case, by factoring the transfer functions, we reduce the dynamic range of the coefficients in each $H_i(z)$, hence we can represent the coefficients with more fractional bits and get an overall better approximation of the ideal transfer function. In fact, we can sometimes "fix" an unstable filter by factoring it to second order sections before quantizing the coefficients.

4. 60 points total. Suppose all quantities in the IIR direct form II system shown in Figure 2 are 16-bit fixed point.

   (a) 40 points. Suppose the feedback coefficients a_1, a_2 are $Q-15$ shorts, the feedforward coefficients b_0, b_1, b_2 are $Q-14$ shorts, and the intermediate results $u[n], u[n-1], u[n-2]$ are all $Q-13$ shorts. Determine all of the shifts necessary to correctly perform the calculations and guarantee no overflow at any point in the processing. Clearly annotate the figure below with your "shift plan".

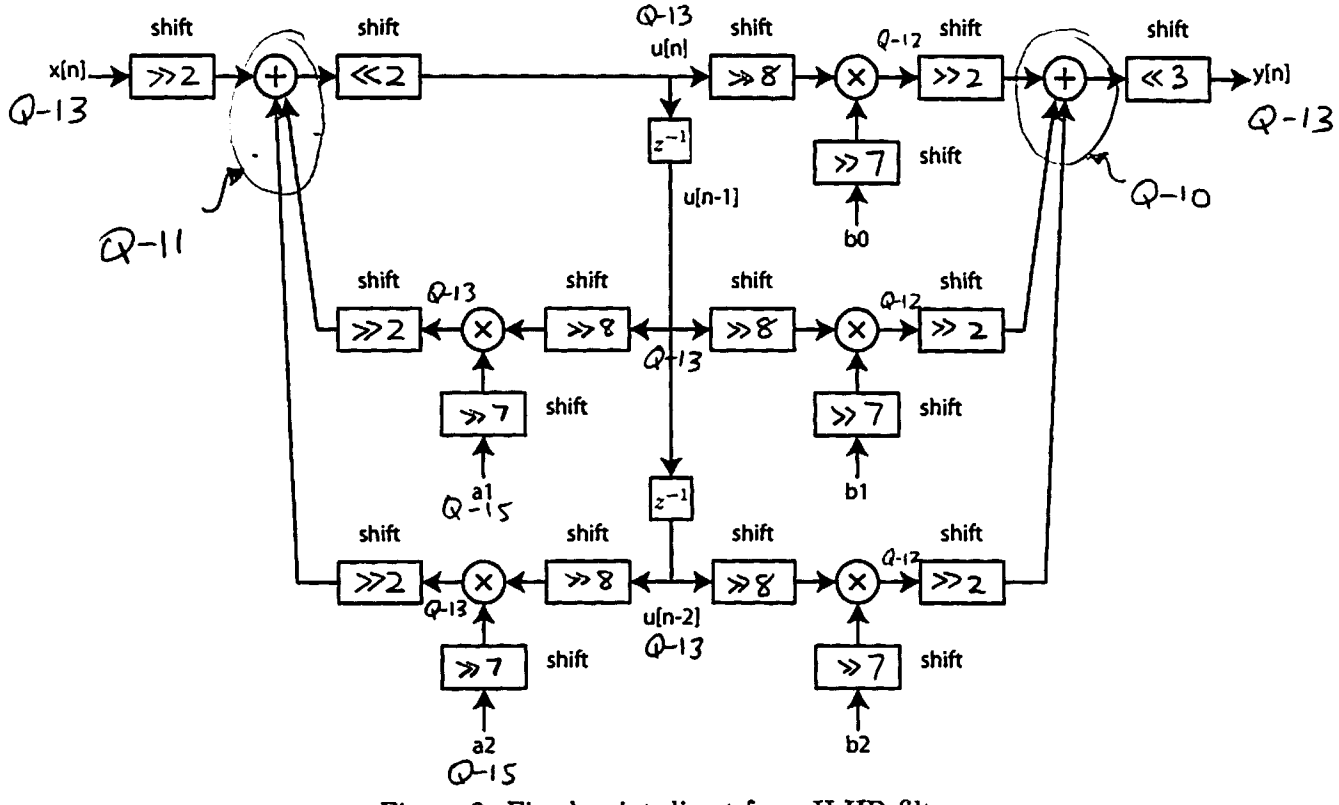

Figure 2: Fixed-point direct form II IIR filter.

The pre-product shifts require us to throw away 15 bits ~~total~~ to avoid overflow. We do that by shifting 8 bits from the intermediate values and 7 bits from the coefficients.

The sums require us to throw away $\lceil \log_2(3) \rceil = 2$ bits to avoid overflow.

The remaining shifts are necessary to align decimal points at the summers and ensure the Q-format of the input and output are identical.

(continued...)

(b) 20 points. Suppose you write the code to implement your fixed-point DF-II IIR filter with your shift plan from part (a) but forget to do any shifting on the input. Discuss how this will affect your filter.

By forgetting to shift the input, we effectively increase the input gain by a factor of four.

This doesn't mean the system will necessarily overflow, but it is possible that overflow could occur at the feedback summer.

If overflow doesn't occur, the magnitude response of the filter will be shifted up by 12dB.