EE2016 Microprocessor: Theory & Lab. Fall 2020

EE Dept, IITM

Mid-Semester Exam - Part 3 (Take Home)

Instructions: 3 days are given to you to solve the following problem and upload the report, after tarring all the files to moodle. Name the files after your roll no. For eg. ee18b032partC.tar.gz. An assignment link would be opened in moodle, over which you could submit.

1 Implementation Problem

1.1 Problem Definition

Implement the 5-tap (case A) and 32-tap (case B) version of bandpass FIR filter in AVR Atmega8 microcontroller. The bandpass range is defined as frequency range from 800 Hz to 3400 Hz. Corresponding to both the above cases, the filter coefficients $\{b_i\}$ are given in a separate file. [You are required to ONLY implement FIR filter and NOT its design]. The filter coefficients are given in the file

1.2 Demonstration

You need to demonstrate its working by feeding in the following input

1. DC signal
2. Sinusoidal signal of frequency 2.125 kHz
3. White noise

Show the i/p and o/p spectrum in both the versions of the FIR filter and for all the above three inputs. Draw the output frequency response. You need to use the Atmel AVR Studio 7 emulation software for design, debugging and final demonstration.

1.2.1 Normalized Values of Inputs

There are many ways in which the inputs are normalised - so that comparisons makes sense (We are normalising so that the visual inspection of the output doesn't much depend on the shape, but on their frequency components).

Normalized Input Amplitude For DC it is 1 unit while for sinusoidal, maximum amplitude is set as 1. For the white noise the standard deviation is set as $\sigma = 0.33$. The marginal distribution is Gaussian and its value can ideally go from $-\infty$ to $+\infty$. Any value out of range of -3σ to $+3\sigma$, you can safely assume that they have not occurred since it has low probability (0.03%). Now, all these inputs (after sampling) lie in the range -1 to $+1$, which could be thought as lying between -128 to $+127$ (in 2's complement representation) after rounding (quantization).

[If you have used the following normalization, then also it is OK. Given that the DC signal value A, its power is equal to the normalized power of the sinusoidal signal of maximum amplitude A_{max} and the variance of white noise. $A^2 = \frac{A_{max}^2}{2} = \sigma_N^2$. Any other format which is reasonable is also acceptable].

1.3 General Guide Lines

1. Reading the data from the file (containing the above generated data), may not be easy. But, you copy the data and add it in the program memory by suitably defining the directive .db
2. Similarly the output of the filter you could store it in the SRAM. Problem is that again you need to display those values in the AVR studio 7, mouse copy and put them in a file. [Seems there is no way of writing them directly into the file or must be extremely difficult]. After this you may manipulate.
3. Note that both the input & output are HEX values and you need to some processing including the convertion into (and from) decimals.
4. The input before quantization could be taken the DTFT and output also, the DTFT (though they are quantized). Hence, Z-transform you need not use [We understand that you were not taught the Z-transform].
5. Use SUITABLE fixed point arithmetic: You may use Qm.n notation for fractional numbers. After quantization use 2's complement & use HEX representation. Take care of overflow etc. The number system you use should be consistent throughout.

These are only guidelines, you may choose to use other schemes /methods, but justify them in the report.

1.4 Generation of Test Signals

Use MATLAB to generate (a) sinusoidal signal at 1800 Hz (b) white noise. Sample them. Overall consider 1000 samples. Put these in a file and read it, into the AVR microcontroller.

1.5 Constraints

It is strongly recommended to use the Application Note AVR223 (Ref[1]) uploaded in moodle, as a reference file. In the following, the Equations refer to those in the above application note.

1. Avoid overflow (Eq 9, Ref[1]) including sub-results.
2. Adopt fixed point computation (Ref[4]).
3. Comply to conservation criteria for avoiding overflow (Eq 7, Ref[1]).
4. Accumulator resolution for both versions of the FIR filter.
5. Use AVR hardware multiplier: Use instructions MUL, MULS, MULSU (Ref[3]).
6. Implement the accumulator of right bit-length (given by equ 6 in [1]) using the 8-bit GPRs internal to Atmega8 microcontroller.
7. Use the circular register scheme (Page 4 of Ref[2]).

1.6 Guidelines for making report

It is expected that in the report following information is there:

1. A section indicating the theory of FIR, convolutions and DTFT etc
2. A section on Fixed point arithmetic used by you
3. A section on DTFT (or Z-transform) details
4. A section on actual implementation details
   (a) Input into the AVR and the address details of the register in which they are stored. And ditto for output values.

(b) Whether circular buffers are used. If it is so their details of the registers and pointers used if any.

(c) Where the coefficients are stored.

(d) Multiplication details

(e) Screen shot of AVR studio 7 ALL of the above register values, where the coefficients, input samples, outputs are shown.

5. Results

(a) Input and output in the same graph.

(b) Frequency domain representation of both input and output.

(c) Repeate (b) for ALL three inputs.

1.7 References

1. AVR223: Digital Filters with AVR, Atmel Corporation, 2008
2. AN852: Implementing FIR and IIR Digital Filters Using PIC18 Microcontrollers, Microchip Technology Inc, 2002
3. AVR201: Using the AVR hardware multiplier, Atmel Corporation, 2000.
4. AVR32765: AVR32 DSPLib Reference Manual, Atmel Corporation, 2009