EENG 3910: Project V – Digital Signal Processing System Design

Assignment 7

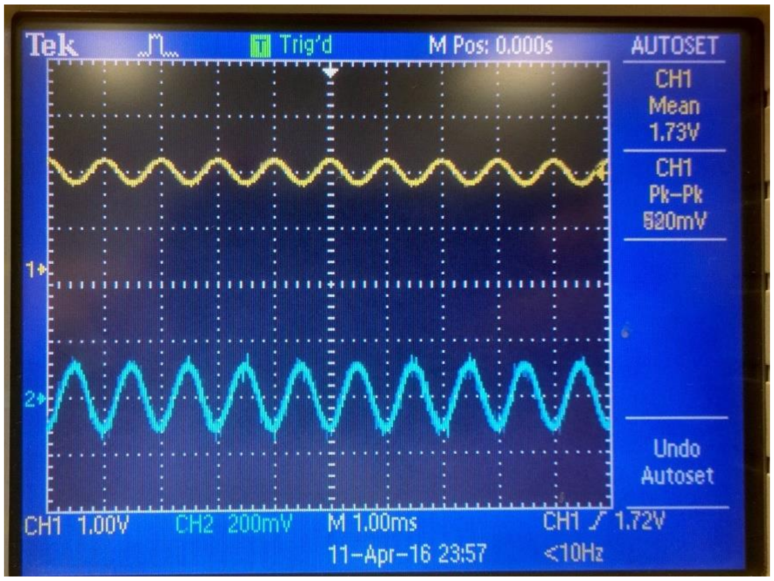
Due 11 April 2016

Nathan Ruprecht

UNT – Electrical Engineering

Introduction
The purpose of assignment 7 was to introduce us to Finite-duration Impulse Response (FIR) filtering as well as continue implementing triple buffering and ping-pong buffering. We still used the same circuit as from assignment 5 so this was a code heavy assignment.

Results and Discussion
Problem 1: Real-time FIR filtering with triple buffering. The code was given to us so we had a baseline to judge off of and change for the other 2 problems. We hooked up the oscilloscope probes to the input signal (matlab signal going into the circuit) and the final output.



Output of Lab7_1

The code was given and we've seen nearly everything before. Still initializing everything we need and implement a triple buffer. No problem. The hardest section for me to understand was the funcatin that processed the data. More spefically the section that included:

```
// FIR filtering: y[n] = sum_{i=0}^{M}(B[i]x[n-i]).
for(i=0; i<BUFFER_LEN; i++) {
        y = 0;
        for(j=0, k=i; j<=FILTER_M; j++, k++) {
                y += filter_B[j]*data[k];
        }
        p_buf[i] = (uint32_t) y;
}
```

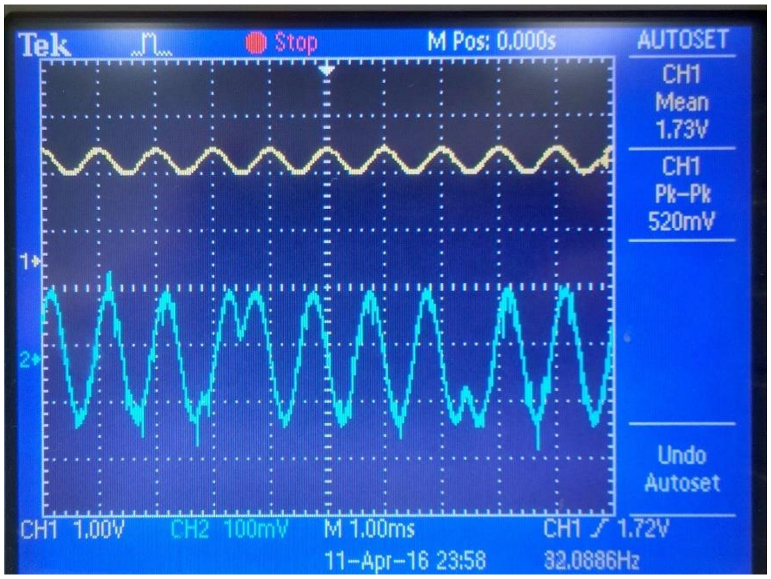
This is where is had to go back to the class notes to understand that this was how we used frame based filtering by multiplying frames then adding them all up.

$$y(n) = \sum_{k=0}^{M} b_k x(n-k)$$

Problem 2: FIR filter design with Matlab. For the second part of lab 7, we were designing our own filter with certain parameters. Fortunately, we were given step by step instructions on how to use Matlab to get the coefficients to plug into our code. Now, we have 51 variables instead of just 28.

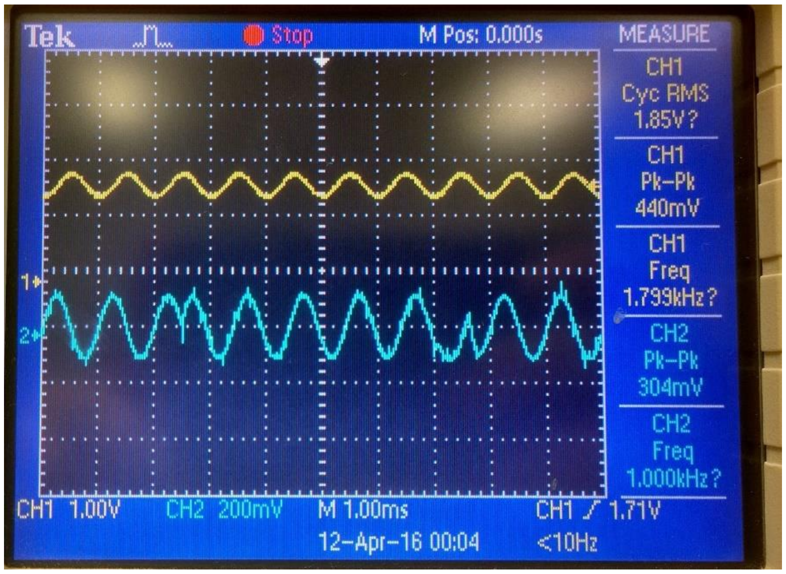
```
float filter_B[FILTER_M+1] = {1.0078983e-03, -1.9806310e-03, -2.0403235e-03,
1.7429967e-03,
             2.8160409e-03, -2.6722261e-03, -4.4943970e-03,  3.5317205e-03,
6.7031277e-03,
            -4.5027826e-03, -9.7065712e-03,  5.5132351e-03,  1.3749776e-02, -
6.5261811e-03,
            -1.9259214e-02,  7.4939108e-03,  2.7015470e-02, -8.3676315e-03, -
3.8691169e-02,
             9.1031735e-03,  5.8658164e-02, -9.6579211e-03, -1.0303509e-01,
1.0004935e-02,
             3.1727629e-01,  4.8987681e-01,  3.1727629e-01,  1.0004935e-02, -
1.0303509e-01,
            -9.6579211e-03,  5.8658164e-02,  9.1031735e-03, -3.8691169e-02, -
8.3676315e-03,
             2.7015470e-02,  7.4939108e-03, -1.9259214e-02, -6.5261811e-03,
1.3749776e-02,
             5.5132351e-03, -9.7065712e-03, -4.5027826e-03,  6.7031277e-03,
3.5317205e-03,
            -4.4943970e-03, -2.6722261e-03,  2.8160409e-03,  1.7429967e-03, -
2.0403235e-03,
            -1.9806310e-03,  1.0078983e-03};
```

The above list of numbers is the output of the app we used in Matlab for our filter design parameters. Putting this into our Lab7_2 code and changing FILTER_M to 51 gives us the below output on the oscilloscope.



Output of Lab7_2

Problem 3: Real-time FIR filtering with Ping-Pong buffering. The third questions was just taking problem two and implementing a ping-pong buffer instead of a triple buffer. Since we did this in lab 6, it was just a matter of changing variable names and using four buffers instead of three. Easy enough now that we've successfully did it before. The output on the oscilloscope should (and is) the same as from problem 2.



Output of Lab7_3

<u>Summary and Conclusion</u>

The same circuit from lab 5 was used so this assignment was just code. I have not used Matlab too much so I definitely do not consider myself that proficient with it. When it came to problem 2 and using Matlab, it was a matter of just following the given instructions. The end results, a list of coefficients to use for the filter variable in our C code. The output looks different in that, at a certain point, the signal switches direction but continues the signal, just swapped sign. If I remember right it's because of the phase shift in the signal. Whenever there is an impulse in the frequency domain, we see this result in the time domain of the signal seeming to switch signs and continue on. We've talked about it a little and we've definitely seen this result before.

Modified Source Code:

Below is the changes I made to Lab7_2 to go from the given triple buffer to the ping-pong buffer:

```
volatile uint32_t ping_pong_buffer[NUM_BUFFER][BUFFER_LEN];
volatile uint32_t in_buff = 0, in_data = 1, out_buff = 2, out_data = 3;

void init_buffer(void)
{
	uint32_t i, j;

	for(i=0; i<NUM_BUFFER; i++)
		for(j=0; j<BUFFER_LEN; j++) {
			ping_pong_buffer[i][j] = 0;
		}
}


void process_data(void)
{
	static float data[BUFFER_LEN + FILTER_M] = {0}; //augmented data frame,
initialized to 0
	float y;
	uint32_t *p_buf = (uint32_t *) ping_pong_buffer[in_data];
	uint32_t i, j, k;

	// Copy data from the data frame to the augmented data frame.
	// Start to fill ater FILTER_M elements.
	memcpy(data+FILTER_M, p_buf, BUFFER_LEN*sizeof(float));


	// FIR filtering: y[n] = sum_{i=0}^{M}(B[i]x[n-i]).
	for(i=0; i<BUFFER_LEN; i++) {
		y = 0;
		for(j=0, k=i; j<=FILTER_M; j++, k++) {
			y += filter_B[j]*data[k];
		}
		p_buf[i] = (uint32_t) y;
	}

	// Save the last FILTER_M number of samples to the beginning of the augmented
data frame.
	memcpy(data, data+BUFFER_LEN, FILTER_M*sizeof(float));
}


// Timer1 interrupt service routine
void Timer1_ISR(void)
{
	static uint32_t sample_index=0;
	uint32_t data, tmp_ui32;

	// Clear the timer interrupt.
	TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
```

```
        // Read data from ADC
        ADCProcessorTrigger(ADC0_BASE, ADC0_SEQ_NUM);
        while(!ADCIntStatus(ADC0_BASE, ADC0_SEQ_NUM, false)) {
        }
        ADCIntClear(ADC0_BASE, ADC0_SEQ_NUM);
        ADCSequenceDataGet(ADC0_BASE, ADC0_SEQ_NUM, &data);

        ping_pong_buffer[in_buff][sample_index] = data;

        // Send data to SPI DAC.
        SSIDataPut(SSI0_BASE,
SPI_CTRL_MASK|(SPI_DATA_MASK&ping_pong_buffer[out_data][sample_index]));

        // Update buffer indices and sample index.
        if(++sample_index >= BUFFER_LEN) {
                sample_index = 0;

                if(data_ready) {
                        buffer_overrun = true;
                }

                tmp_ui32 = in_buff;
                in_buff = in_data;
                in_data = tmp_ui32;

                tmp_ui32 = out_buff;
                out_buff = out_data;
                out_data = tmp_ui32;

                data_ready = true;
         }
}
```