EE 586 Assignment 2

Name: Shuo Wang

USC ID: 8749390300

Question 1:

For the question to deal with a small function, I decide to accomplish a 1-d convolution calculation since convolution is the basis of digital signal processing. According to the basic idea, I constructed basic algorithm of convolution, which can be shown as follow (for two input float vectors):

- Reverse one of the input vector, called filter.
- Padding zero at the two terminals of another vector (called input). The padding size for each terminal is the size of reversed vector minus 1.
- Build a new vector (called res) whose size is the sum of two vectors minus 1. Then, apply the formula to calculate result of each element in result vector:

$$\mathrm{res}(i) = \sum_{j=0}^{m} input(i+j) \times filter(j)$$

After applying such equation, I built the MATLAB version of this function and applied conv available in MATLAB to confirm the algorithm. In addition, I built the C-version convolution function based on the MATLAB version of this function.

Then, I burn the 6713DSK board by the C function. In the meantime, I repeated running this function for 10000 times to increase the accuracy of timing. The example input is shown as follow:

$$\text{Input signal: } [0.1, 0.3, 0.2, 0.4, 0.3, 0.5, 0.6, 0.4, 0.4, 0.3, 0.7, 0.1]$$

$$\text{Filter: } [-0.1, 0.0, 0.1, 0.2, 0.1, 0.0, -0.1]$$

The number of branches, which will be applied to calculate the running time of algorithm, is 279805892, meaning that average running time is 27981. To decrease the complexity of convolution on 6713, some modifications are applied to improve the calculation speed.

(1) Apply integers instead of float
As a matter of fact, applying float will not only consume a large amount of memory, but increase the calculation speed. Therefore, integers will be applied to increase the calculation speed. To preserve the accuracy, I will time 100 to the float and convert the float to be short integer, meaning that the accuracy is 10^{-2}, which can be shown as follow:

$$new\ input = 100 \times input$$

As a result, the total running time is 124921602 (average running time is 12492) when applying same input.

(2) Applying unroll loop on calculation of single $\mathrm{res}(i)$
Usually, loop will be executed one by one, meaning that there will be one more branch in each loop step. To decrease the time, we can execute such step every four step if there are four more steps available. The code for this step can be shown as follow:

```
while (i < an + bn - 1)

{
        short sum = 0;
        short j = 0;
        while (j < an)
        {
                if(j + 3 < an)
                {
                        sum += a[j] * bnew[j + i];
                        sum += a[j + 1] * bnew[j + i + 1];
                        sum += a[j + 2] * bnew[j + i + 2];
```

```
                        sum += a[j + 3] * bnew[j + i + 3];
                        j += 4;
                }
                else
                {
                        sum += a[j] * bnew[j + i];
                        j++;
                }
        }
        res[i] = sum;
        i++;
}
```

When we calculate the running time by same input, it can be found that the running time has been decreased to be 103471836, meaning that average time is 10347.

(3) Applying unroll loop on reverse step and res calculation

Similarly, in the reverse step, every three pairs in vectors can be swapped in each loop to decrease the time required to reverse. In addition, calculation of each $\mathrm{res}(i)$ requires the loop, meaning that the unroll loop can be applied to decrease the running time. In fact, it can be decreased into 101828177, meaning that the average time is 10183.

(4) Applying shift instead of multiplication

In the lecture course, shift calculation is applied to increase the calculation speed. However, there will be a significant restriction: the value of the number to multiply needs to be known and constant, or the complexity will increase since division will consume more complexity. As a result, when inputs are multiplied by 100, I made a modification as follow:

$$100 \times input = (1 \ll 7) \times 0.78125 \times input$$

Then, the time will be decreased to be 101227314, meaning that the average time is 10123. That is a slight change. In addition, the result value is slightly different from previous solutions since the $(1 \ll 7) \times 0.78125$ may not be exactly 100 for some input numbers.

In a word, there will be a series of methods to decrease the DSK 6713 running time, and the effect to decrease the complexity depends on specific occasions, meaning that