

The City College of New York

Grove School of Engineering - Electrical Engineering Dept.

EE 42500 – Computer Engineering Lab

Professor Feng

Spring 2020

**Lab 6**

Report due: 5/12/2020

**Prepared By**: Ben Variano

**Introduction**

In this lab, we will be building off the previous lab to develop a moving average filter. In this lab we will be taking four different input signals, x1, x2, x3, x4 and convolving it with various moving average filters to get an output time series. This time series will be coded using a linear array buffer and performing basic arithmetic to get the output time series that is a result of the given filter and input signal.

**Task 1**

In Task 1, we were asked to compile and simulate the provide assembly file. While running the code, we examined the variable called value which would be our input signal x[n]. For each of the signals x1[n] to x4[n] we note that there are different periods pertaining to each. For x1[n] the period is 2, for x2[n] the period is 4, for x3[n] the period is 6, and for x4[n] the period is 8. For each input signal, we have to change the literal value that is moved to counter to pertain to the given period. In addition, we must comment out/uncomment the corresponding time series data for the input signal we wish to use at the bottom of the assembly code.

**Task 2**

In Task 2 our goal was to implement the following moving average filter for all four input signals:

A picture containing clock

Description automatically generated

We did so by implementing a subroutine containing a linear buffer 11 values long (we will see why we used 11 values as opposed to 4 in the following tasks) as well as a subroutine to compute the arithmetic. The linear buffer allowed us to access previous time data (ex. x[n-1], x[n-2], etc.) that would then be used in the arithmetic subroutine to compute y[n], before outputting it to the ‘output’ variable. By doing so for all four input signals, we were able to gather the following output time series where yn corresponds to the input xn, n= 1,2,3,4.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | K | K+1 | K+2 | K+3 | K+4 | K+5 | K+6 | K+7 | K+8 | K+9 | K+10 | K+11 |
| x1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| y1 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 |
| x2 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 |
| y2 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 |
| x3 | 180 | 240 | 200 | 244 | 216 | 236 | 180 | 240 | 200 | 244 | 216 | 236 |
| y3 | 225 | 224 | 219 | 218 | 214 | 216 | 225 | 224 | 219 | 218 | 214 | 126 |
| x4 | 180 | 240 | 200 | 244 | 216 | 236 | 160 | 176 | 180 | 240 | 200 | 244 |
| y4 | 225 | 224 | 214 | 197 | 188 | 189 | 199 | 216 | 225 | 224 | 214 | 197 |

*Table 1: Output of Moving Average Filter y[n] for an input signal x[n]*

**Task 3**

In Task 3, we had the same goal as in Task 2 but pertaining to the moving average filter:

A picture containing object, room

Description automatically generated

We were able to obtain the desired results by simply modifying the subroutine that calculated y[n] to correspond to the arithmetic for A[n] and using the same four input signals for x[n]. The following table shows the output time series data gathered.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | K | K+1 | K+2 | K+3 | K+4 | K+5 | K+6 | K+7 | K+8 | K+9 | K+10 | K+11 |
| x1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| A1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| x2 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 |
| A2 | 190 | 242 | 190 | 242 | 190 | 242 | 190 | 242 | 190 | 242 | 190 | 242 |
| x3 | 180 | 240 | 200 | 244 | 216 | 236 | 180 | 240 | 200 | 244 | 216 | 236 |
| A3 | 203 | 239 | 194 | 240 | 199 | 241 | 203 | 239 | 194 | 240 | 199 | 241 |
| x4 | 180 | 240 | 200 | 244 | 216 | 236 | 160 | 176 | 180 | 240 | 200 | 244 |
| A4 | 189 | 224 | 189 | 224 | 189 | 224 | 189 | 224 | 189 | 224 | 189 | 224 |

*Table 2: Output of Moving Average Filter A[n] for an input signal x[n]*

**Task 4**

For Task 4 we were asked to do the same procedure as in Task 2 and 3 but for the moving average filter:

A picture containing clock

Description automatically generated

We were able to obtain the desired results by simply modifying the subroutine that calculated y[n] to correspond to the arithmetic for B[n] and using the same four input signals for x[n]. The following table shows the output time series data gathered.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | K | K+1 | K+2 | K+3 | K+4 | K+5 | K+6 | K+7 | K+8 | K+9 | K+10 | K+11 |
| x1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| B1 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 |
| x2 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 |
| B2 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 |
| x3 | 180 | 240 | 200 | 244 | 216 | 236 | 180 | 240 | 200 | 244 | 216 | 236 |
| B3 | 228 | 218 | 212 | 228 | 218 | 212 | 228 | 218 | 212 | 228 | 218 | 212 |
| x4 | 180 | 240 | 200 | 244 | 216 | 236 | 160 | 176 | 180 | 240 | 200 | 244 |
| B4 | 210 | 215 | 215 | 207 | 205 | 198 | 198 | 206 | 208 | 215 | 215 | 207 |

*Table 3: Output of Moving Average Filter B[n] for an input signal x[n]*

**Task 5**

For Task 5, we repeated the same procedure as earlier for moving average filter C[n]:

A picture containing object, antenna

Description automatically generated

Where variables j, k, and l were of our choosing. I chose the values 4, 6, and 8 accordingly. We used the same four input signals x[n] as before and needed only to change our arithmetic subroutine to correspond to the moving average filter we applied for C[n]. The following table shows the output time series data gathered.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | K | K+1 | K+2 | K+3 | K+4 | K+5 | K+6 | K+7 | K+8 | K+9 | K+10 | K+11 |
| x1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| C1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| x2 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 |
| C2 | 195 | 243 | 185 | 241 | 195 | 243 | 185 | 241 | 195 | 243 | 185 | 241 |
| x3 | 180 | 240 | 200 | 244 | 216 | 236 | 180 | 240 | 200 | 244 | 216 | 236 |
| C3 | 203 | 239 | 194 | 240 | 199 | 241 | 203 | 239 | 194 | 240 | 199 | 241 |
| X4 | 180 | 240 | 200 | 244 | 216 | 236 | 160 | 176 | 180 | 240 | 200 | 244 |
| C4 | 194 | 225 | 193 | 222 | 175 | 209 | 194 | 240 | 194 | 225 | 193 | 222 |

*Table 4: Output of Moving Average Filter C[n] for an input signal x[n]*

**Tables**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | K | K+1 | K+2 | K+3 | K+4 | K+5 | K+6 | K+7 | K+8 | K+9 | K+10 | K+11 |
| X1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| y1 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 |
| A1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| B1 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 | 210 |
| C1 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 | 180 | 240 |
| X2 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 | 180 | 240 | 200 | 244 |
| y2 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 |
| A2 | 190 | 242 | 190 | 242 | 190 | 242 | 190 | 242 | 190 | 242 | 190 | 242 |
| B2 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 | 216 |
| C2 | 195 | 243 | 185 | 241 | 195 | 243 | 185 | 241 | 195 | 243 | 185 | 241 |
| X3 | 180 | 240 | 200 | 244 | 216 | 236 | 180 | 240 | 200 | 244 | 216 | 236 |
| y3 | 225 | 224 | 219 | 218 | 214 | 216 | 225 | 224 | 219 | 218 | 214 | 126 |
| A3 | 203 | 239 | 194 | 240 | 199 | 241 | 203 | 239 | 194 | 240 | 199 | 241 |
| B3 | 228 | 218 | 212 | 228 | 218 | 212 | 228 | 218 | 212 | 228 | 218 | 212 |
| C3 | 203 | 239 | 194 | 240 | 199 | 241 | 203 | 239 | 194 | 240 | 199 | 241 |
| X4 | 180 | 240 | 200 | 244 | 216 | 236 | 160 | 176 | 180 | 240 | 200 | 244 |
| y4 | 225 | 224 | 214 | 197 | 188 | 189 | 199 | 216 | 225 | 224 | 214 | 197 |
| A4 | 189 | 224 | 189 | 224 | 189 | 224 | 189 | 224 | 189 | 224 | 189 | 224 |
| B4 | 210 | 215 | 215 | 207 | 205 | 198 | 198 | 206 | 208 | 215 | 215 | 207 |
| C4 | 194 | 225 | 193 | 222 | 175 | 209 | 194 | 240 | 194 | 225 | 193 | 222 |

*Table 5: All Moving Average Filters for each input signal x[n]*

**Code**

The following assembly code is what I modified to get obtain each of the above outputs. The time series data shown is using the code for x4[n] but was switched out for the correct time signal data needed for each of the four runs in every moving average filter computed.

A screenshot of a social media post

Description automatically generated

*Image 1: Mainline Code*

*A screenshot of a social media post

Description automatically generated*

*Image 2: Linear Memory Buffer 10 Values in Length*

*A screenshot of text

Description automatically generated*

*Image 3: Moving Average Filter y[n] Subroutine*

*A screenshot of text

Description automatically generated*

*Image 4: Moving Average Filter A[n] Subroutine*

*A screenshot of text

Description automatically generated*

*Image 5: Moving Average Filter B[n] Subroutine*

*A screenshot of text

Description automatically generated*

*Image 6: Moving Average Filter C[n] Subroutine*

*A screenshot of a social media post

Description automatically generated*

*Image 7: Time Series Data for X1, X2, X3, X4*

**Conclusion**

In Conclusion, we were able to create various moving average filters within our assembly code and obtain data corresponding to the output time series. By creating a 10-value linear memory buffer, we were able to access past time series input data. Using this data, we could compute the output for a given moving average filter. Through simple manipulation of our computational subroutine, we could retrieve the output of any moving average filter that used data no more than 10 inputs behind the current input. To do anything greater that that we could simply extend the linear memory buffer as needed or normalize the offset from the current input to the period range since the inputs had a repeating pattern.