University of Texas at Dallas

Department of Electrical Engineering

EEDG/CE 6306 - Application Specific Integrated Circuit Design

Homework #3

Submission: Due on 11:59 PM, Fri, Feb 9, 2024.

(a) Your *C/C++* source code. (.c .cpp) (90%)

(b) Your output file *output2.out* for *data2.in* (10%)

Related input files are posted on eLearning, which include

(a) *coeff.in*

(b) *data1.in, output1.out, data2.in*

Submit onto eLearning a zip file (team): <First name #1>_<Last name #1>_<FIrst name #2>_<Last name#2>_HW03.zip

* If you have any question about the homework, you may contact TA Tianning Guo (tianning.guo@utdallas.edu).

Write a **C/C++** program to implement an algorithm presented in the MSDAP paper (PART I). In this homework, we implement the general convolution procedure for fixed point data. Carefully read the following example and data format descriptions.

Convolution Example:

Digital FIR filter involves 1-D convolution of filter coefficients and input data. Assume filter order N=3, POT digit limit to 2^{-4}.

$$y(n) = \sum_{k=0}^{N=3} h(k)x(n-k)$$
$$= h(0)x(n-0) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3)$$

Assume:

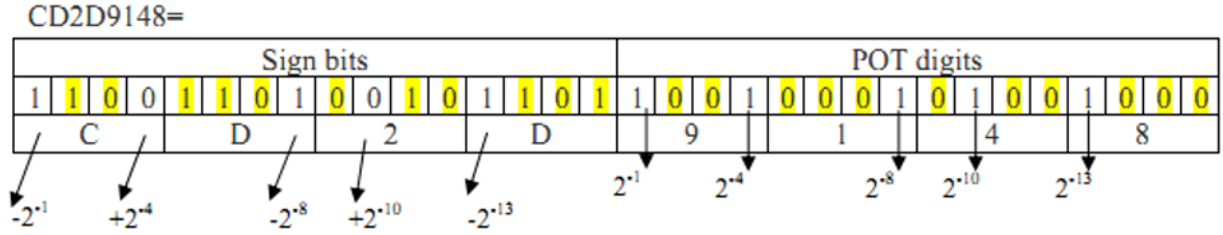
$$h(0) = 2^{-1} - 2^{-3}$$
$$h(1) = 2^{-3} + 2^{-4}$$
$$h(2) = 2^{-1} + 2^{-2} - 2^{-4}$$
$$h(3) = -2^{-3}$$

Then the equation you need to implement is

$$y(n) = (2^{-1} - 2^{-3})x(n-0) + (2^{-3} + 2^{-4})x(n-1) + (2^{-1} + 2^{-2} - 2^{-4})x(n-2) + (-2^{-3})x(n-3)$$

Coefficients format:

Coefficients contain sign bits and POT bits. Here is $h(0)$ and how to read it. Note in our homework, coefficients can reach 2^{-16}.



h(0)=$-2^{-1}+2^{-4}-2^{-8}+2^{-10}-2^{-13}$ (Bits marked as yellow are don't care value, simply discard)

Input data $x(n)$ format:

Input data are **signed 16-bit hex number**, **fixed-point, two's complement**, **Leftmost bit (MSB) is sign bit.**

C48B=

1	1	0	0	0	1	0	0	1	0	0	0	1	0	1	1

MSB **LSB**

Output data $y(n)$ format:

Given coefficients and $x(n)$ having 16 bits, we expect that output data have 40 bits.

Here is the detailed computation process of x(0)*h(0):

x(0) = 7138 : **0111 0001 0011 1000**

Firstly, sign-extend to 24-bit: **0000 0000 0111 0001 0011 1000**

(left 8 bits are the same as the MSB sign bit. Sign-extending by 8 gives enough room for carry in bits generated during the computation)

Next, ext. to 40-bit:

0000 0000 0111 0001 0011 1000 0000 0000 0000 0000

(right side 16 bits (0s) are for **SHIFT** operation (16 bits max))

C48B*2^{-1} (1st POT bit $h(0)$ in is 1, Sign bit is 1 so flip all bits and add 1 to LSB for -x(0), shift down by 1):

1111 1111 1100 0111 0110 0100 0000 0000 0000 0000

C48B*2^{-2} , *2^{-3} (2nd and 3rd POT bit in $h(0)$ are 0, so we add 0 to result)

C48B*2^{-4} (4th POT bit in $h(0)$ is 1, Sign bit is 0 so we shift x(0) by 4 and add):

```
 1111 1111 1100 0111 0110 0100 0000 0000 0000 0000
+1111 1111 1111 1000 1110 1100 1000 0000 0000 0000
-------------------------------------------------
 1111 1111 1100 0000 0101 0000 1000 0000 0000 0000
```

(next, we add $x(0)$ at 10th and minus $x(0)$ at 8th, 13th steps, why?)

C48B*2^{-16} (last bit of $h(0)$ is 0, add 0 to result then shift right by 1 bit)

Output is:

Homework requirements

(1.1) Implement above convolution procedure with only plus, minus and shift operations. (multiplication is not allowed). All coefficients are saved in coeff.in

(1.2) The output data should be 40 bits and saved as a hexadecimal number. Do not omit the leading 0s when saving the results. For example, please save “00FF” instead of “FF”.

(1.3) Filter’s order is 256 (N=255), total number of input data is 1000.

(1.4) Assume x(-255)=x(-254)= ⋯ =x(-2)=x(-1)=0

(1.5) You can use data1.in and output1.out to verify your program.

(1.6) Save your output data (for data2.in) in hex number to the file named as data2.out.

(1.7) Please use gcc/g++ to compile your code. You could test your code with gcc/g++ on server engnx.utdallas.edu. For any options used to compiling your code (e.g. “–std=c++14”), please attach a “readme.txt” in your submission. Otherwise, any compiling errors of your source code file (like copying code into .doc .docx or .pdf) will lead your grade to be 0.

(1.9) Your program should accept coeff.in, data.in, output.out as arguments. Half of the credits will be deducted if your program fails to accept the arguments (for example, you hardcoded the filename in your source code). For testing purpose, TA will call your program using the following format.

```
./filter  <path/to/coeff.in>  <path/to/data.in>  <path/to/output.out>
```

Appendix:

To use the gcc/g++ compiler, please refer to the following steps. In general, TA will follow the same process to test your program.

(1) Log on server engnx.utdallas.edu.

(2) Use following commands to compile your code:

If you program in C:

```
gcc your_source_file.c -o youroutput.o
```

If you program in C++:

```
g++ your_source_file.cpp -o youroutput.o
```

(3) Run your program:

```
./youroutput.o  <first number> <second number>
```

4th POT

1111	1111	1100	0111	0110	0100	0000	0000	0000	0000
+0000	0000	0000	0111	0001	0011	1000	0000	0000	0000
1111	1111	1100	1000	0111	0111	1000	0000	0000	0000

8th POT

1111	1111	1100	1000	0111	0111	1000	0000	0000	0000
+1111	1111	1111	1111	1100	0111	0110	0100	0000	0000
1111	1111	1100	1000	0011	1111	1110	0100	0000	0000

10th POT

1111	1111	1100	1000	0011	1111	1110	0100	0000	0000
+0000	0000	0000	0000	0001	1100	0100	1110	0000	0000
1111	1111	1100	1000	0100	1011	0011	0010	0000	0000