# University of Texas at Dallas

## Department of Electrical Engineering

## EEDG 6306 - Application Specific Integrated Circuit Design

## Homework #4

**Due on Mid-night 12:00, September 21th, 2016**

Submission for Homework #4:

**(a) Your C/C++ source code. (b) Your output file. (c) HW report**

Input files will be posted on: http://utdallas.edu/~zxb107020

Please submit this homework to **zxb107020@utdallas.edu**

Write a ***C/C++*** program to accomplish the computation presented in the paper. At this time, we introduce **virtual address coefficient** and **rj** to do the computation. Follow the tutorial below and section III of the paper.

(1.1) **Only shift and addition** operation are allowed for computation.

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

(1.3)The output data should be 40 bits and printed out as hexadecimal number.

(1.4) Assume $x(0-k_{max})=\cdots=x(-2)=x(-1)=0$. **Your input start from X(0).**

(1.5) Print your output data (for data2.in/Coeff2.in/Rj2.in) to the file named xxx_HW4.out, xxx is your **First Name** , if submit by team separate your first name by _ exp. xxx_xxx_HW4.out. Apply the same name style to your source code file.

(1.6) Follow the procedure in the example for computation.

(1.7) data1.in/Coeff1.in/Rj1.in and ouput1.out are for testing purpose

(1.8) Your report should include how to implement the algorithm.

## Computation transformation

Example:

Assume filter order N=3, POT digit limit to $2^{-4}$(in this HW, it can reach $2^{-16}$)

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

$$= h(0)x(n-0) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3)$$

Assume:

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

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

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

$$h(3) = -2^{-3}$$

=>

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

=>

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

Let:

$$u_4 = x(n-0) + x(n-2)$$

$$u_3 = x(n-2)$$

$$u_2 = -x(n-0) + x(n-1) - x(n-3)$$

$$u_1 = x(n-1) - x(n-2)$$

=>

$$y(n) = 2^{-1}u_4 + 2^{-2}u_3 + 2^{-3}u_2 + 2^{-4}u_1$$

=>

$$y(n) = 2^{-1}(u_4 + 2^{-1}(u_3 + 2^{-1}(u_2 + 2^{-1}u_1)))$$

***Use the highlight formulation for this homework.***

***Below is a general format for 16 bit shifting,***

$$y(n) = 2^{-1}(u_{16} + 2^{-1}(u_{15} + \ldots .\ 2^{-1}(u_2 + 2^{-1}u_1$$

**The format of input data:**

**C48B=**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

**MSB** **16 bits** **LSB**

**The format of coefficient data:**

**038=**

| s | Address bits | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

***s: sign bit*** ***Address bits:***

***0: addition*** ***The address of input data***

***1: subtraction***

**MSB** **9 bits** **LSB**

Coefficient data **"038" in hex number** is equivalent to **ADDING term x(n-56)**

**18C=**

| s | address bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

**MSB** **9 bits** **LSB**

Coefficient data **"18C" in hex number** is equivalent to **SUBTRACTING term x(n-140)**

**The format of rj data:**

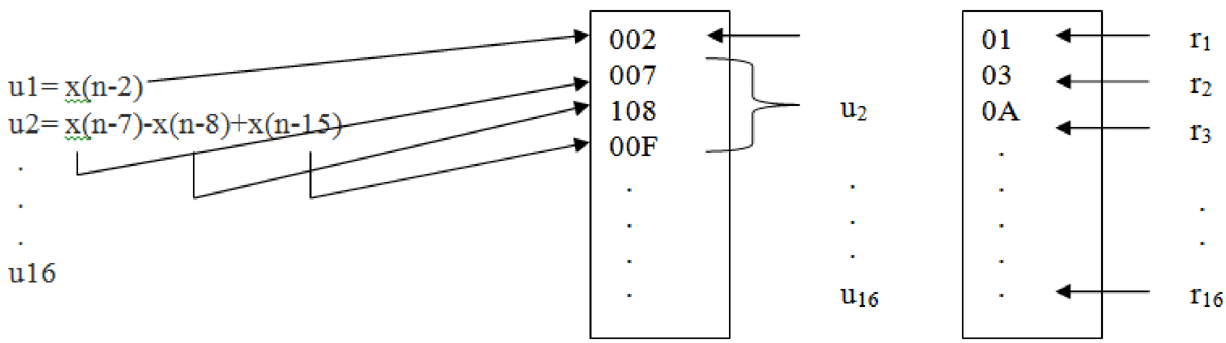
**OB=**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **1** |

**MSB** **8 bits** **LSB**

**The rj data shows the total number of addition/subtraction operation involved in computing each u term.**

For the rj data shows above, the specific u **($u_j$)** term have 11 addition/subtraction operations when doing the computation.




The above picture shows how to get specific u by coefficient and rj.

<table>
<tr><td>038</td><td>18C</td><td>00F</td><td>096</td><td>16A</td><td>130</td><td>110</td><td>013</td><td>14F</td><td>1F1</td><td>0FA</td><td>08E</td><td>1FC</td><td>1B0</td><td>13E</td><td>0CF</td></tr>
<tr><td colspan="11">$R_1$(0B)</td><td colspan="5">$R_2$(0F)</td></tr>
<tr><td>0EE</td><td>021</td><td>1AF</td><td>04C</td><td>0A5</td><td>076</td><td>0EB</td><td>13E</td><td>0A8</td><td>055</td><td>17A</td><td>078</td><td>0B4</td><td>03D</td><td>0B7</td><td>1DD</td></tr>
<tr><td colspan="10">$R_2$(0F)</td><td colspan="6">$R_3$(0E)</td></tr>
<tr><td>169</td><td>16C</td><td>1F3</td><td>0E1</td><td>0B2</td><td>14E</td><td>1D3</td><td>0F8</td><td>14D</td><td>1FF</td><td>170</td><td>002</td><td>04A</td><td>1B0</td><td>17E</td><td>115</td></tr>
<tr><td colspan="8">$R_3$(0E)</td><td colspan="8">$R_4$(0D)</td></tr>
<tr><td>132</td><td>1F9</td><td>15D</td><td>024</td><td>104</td><td>1A3</td><td>1BC</td><td>007</td><td>01A</td><td>1E4</td><td>0C3</td><td>095</td><td>0AF</td><td>1F7</td><td>08C</td><td>1C6</td></tr>
<tr><td colspan="5">$R_4$(0D)</td><td colspan="8">$R_5$(08)</td><td colspan="3">$R_6$(0D)</td></tr>
<tr><td>19C</td><td>0FC</td><td>00C</td><td>0FB</td><td>134</td><td>1E9</td><td>1AA</td><td>176</td><td>01C</td><td>075</td><td>0CC</td><td>14A</td><td>1B1</td><td>142</td><td>1B6</td><td>1B8</td></tr>
<tr><td colspan="10">$R_6$(0D)</td><td colspan="6">$R_7$(0D)</td></tr>
<tr><td>1BB</td><td>09F</td><td>1FC</td><td>027</td><td>134</td><td>0D1</td><td>00F</td><td>189</td><td>130</td><td>199</td><td>04B</td><td>017</td><td>081</td><td>0E1</td><td>09D</td><td>10C</td></tr>
<tr><td colspan="7">$R_7$(0D)</td><td colspan="6">$R_8$(06)</td><td colspan="3">$R_9$(05)</td></tr>
<tr><td>1F3</td><td>02B</td><td>1D3</td><td>09C</td><td>1D8</td><td>01D</td><td>0A6</td><td>02D</td><td>179</td><td>0AE</td><td>122</td><td>076</td><td>012</td><td>094</td><td>090</td><td>141</td></tr>
<tr><td colspan="2">$R_9$(05)</td><td colspan="6">$R_{10}$(06)</td><td colspan="4">$R_{11}$(04)</td><td colspan="4">$R_{12}$(06)</td></tr>
<tr><td>13D</td><td>104</td><td>062</td><td>0C1</td><td>093</td><td>068</td><td>132</td><td>183</td><td>0B6</td><td>0DD</td><td>0BD</td><td>1FE</td><td>0DD</td><td>07C</td><td>0E6</td><td>105</td></tr>
<tr><td colspan="2">$R_{12}$(06)</td><td colspan="9">$R_{13}$(09)</td><td colspan="5">$R_{14}$(0B)</td></tr>
<tr><td>1A2</td><td>0F1</td><td>00A</td><td>049</td><td>0AD</td><td>095</td><td>011</td><td>0E0</td><td>17C</td><td>1E3</td><td>1C2</td><td>0A7</td><td>1F8</td><td>12C</td><td>123</td><td>0C1</td></tr>
<tr><td colspan="6">$R_{14}$(0B)</td><td colspan="10">$R_{15}$(14)</td></tr>
<tr><td>050</td><td>176</td><td>057</td><td>102</td><td>155</td><td>13F</td><td>1FD</td><td>1E7</td><td>159</td><td>047</td><td>018</td><td>13D</td><td>1E6</td><td>053</td><td>192</td><td></td></tr>
<tr><td colspan="10">$R_{15}$(14)</td><td colspan="5">$R_{16}$(05)</td><td></td></tr>
</table>

Here is an example how to get y(2) (inputs are the same with your test set):

No x(n-0), x(n-1) terms

=>

Y(0)=0;

Y(1)=0;

C48B: 1100 0100 1000 1011 // x(0)
ext. to 24-bit: 1111 1111 1100 0100 1000 1011 // x(0)
two's complement: 0000 0000 0011 1011 0111 0101 // -x(0)

① u1 = 0000 0000 0000 0000 0000 0000

② $2^{-1}$u1 = 0000 0000 0000 0000 0000 0000 0

③ (②+u2) = 0000 0000 0000 0000 0000 0000 0

④ $2^{-1}$(②+u2) = 0000 0000 0000 0000 0000 0000 00

⑤ (④+u3) = 0000 0000 0000 0000 0000 0000 00

⑥ $2^{-1}$(④+u3) = 0000 0000 0000 0000 0000 0000 000

⑦ **(⑥+u4) = 0000 0000 0000 0000 0000 0000 000+**

**1111 1111 1100 0100 1000 1011 000**

**= 1111 1111 1100 0100 1000 1011 000**

⑧ $2^{-1}$(⑥+u4) = 1111 1111 1110 0010 0100 0101 1000

⑨ (⑧+u5) = 1111 1111 1110 0010 0100 0101 1000

⑩ $2^{-1}$(⑧+u5) = 1111 1111 1111 0001 0010 0010 1100 0

⑪ (⑩+u6) = 1111 1111 1111 0001 0010 0010 1100 0

⑫ $2^{-1}$(⑩+u6) = 1111 1111 1111 1000 1001 0001 0110 00

⑬ (⑫+u7) = 1111 1111 1111 1000 1001 0001 0110 00

⑭ $2^{-1}$(⑫+u7) = 1111 1111 1111 1100 0100 1000 1011 000

⑮ (⑭+u8) = 1111 1111 1111 1100 0100 1000 1011 000

⑯ $2^{-1}$(⑭+u8) = 1111 1111 1111 1110 0010 0100 0101 1000

⑰ (⑯+u9) = 1111 1111 1111 1110 0010 0100 0101 1000

⑱ $2^{-1}$(⑯+u9) = 1111 1111 1111 1111 0001 0010 0010 1100 0

⑲ (⑱+u10) = 1111 1111 1111 1111 0001 0010 0010 1100 0

⑳ $2^{-1}$(⑱+u10) = 1111 1111 1111 1111 1000 1001 0001 0110 00

㉑ (⑳+u11) = 1111 1111 1111 1111 1000 1001 0001 0110 00

㉒ $2^{-1}$(⑳+u11) = 1111 1111 1111 1111 1100 0100 1000 1011 000

㉓ (㉒+u12) = 1111 1111 1111 1111 1100 0100 1000 1011 000

㉔ $2^{-1}$(㉒+u12) = 1111 1111 1111 1111 1110 0010 0100 0101 1000

㉕ (㉔+u13) = 1111 1111 1111 1111 1110 0010 0100 0101 1000

㉖ $2^{-1}$(㉔+u13) = 1111 1111 1111 1111 1111 0001 0010 0010 1100 0

㉗ (㉖+u14) = 1111 1111 1111 1111 1111 0001 0010 0010 1100 0

㉘ $2^{-1}$(㉖+u14) = 1111 1111 1111 1111 1111 1000 1001 0001 0110 00

㉙ (㉘ *+u15) = 1111 1111 1111 1111 1111 1000 1001 0001 0110 00 +*

*0000 0000 0011 1011 0111 0101 0000 0000 0000 00*

*= 0000 0000 0011 1011 0110 1101 1001 0001 0110 00*

㉚ $2^{-1}$(㉘+u15) = 0000 0000 0001 1101 1011 0110 1100 1000 1011 000

㉛ (㉚+u16) = 0000 0000 0001 1101 1011 0110 1100 1000 1011 000

㉜ $2^{-1}$(㉚+u16) = 0000 0000 0000 1110 1101 1011 0110 0100 0101 1000

Hex= 0 0 0 E D B 6 4 5 8

y(2)= 000EDB6458