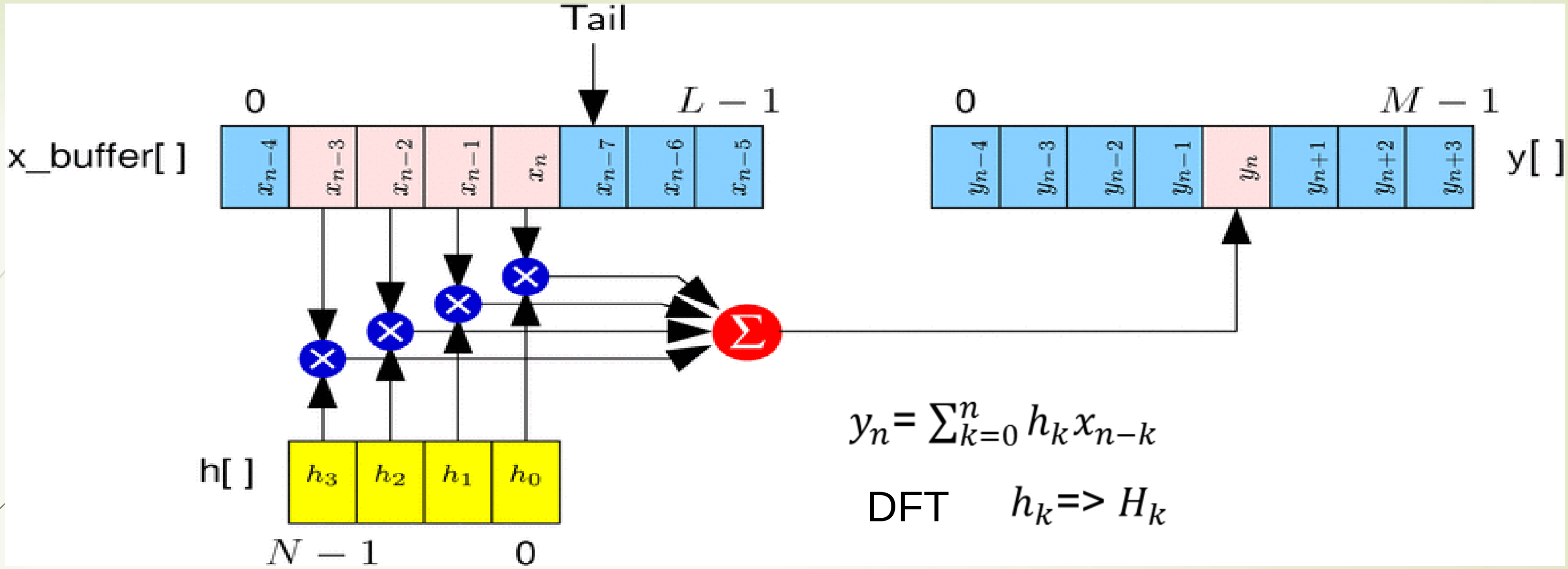
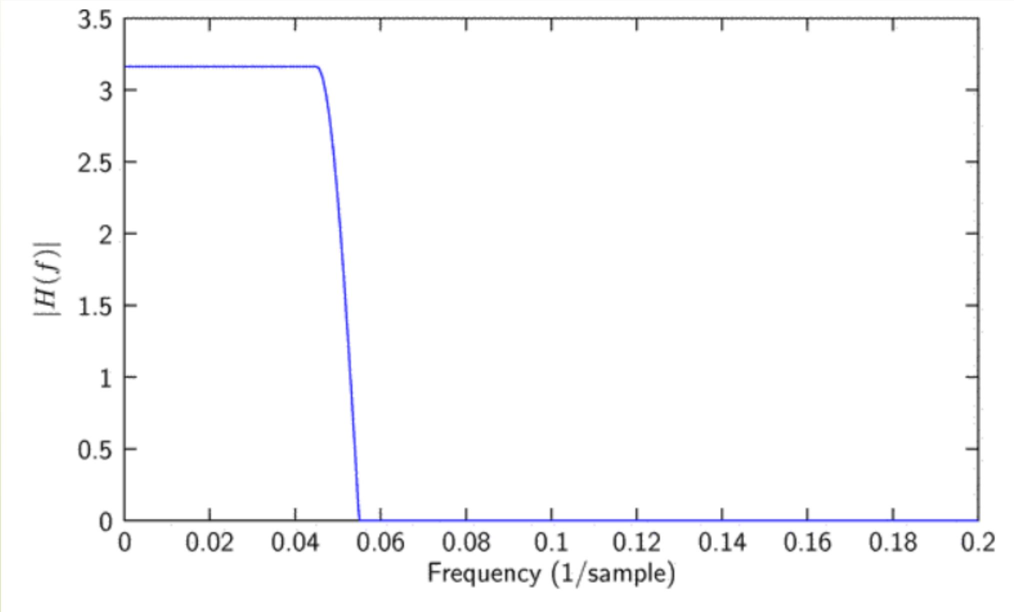
Implement FIR

Dr. Fangning Hu



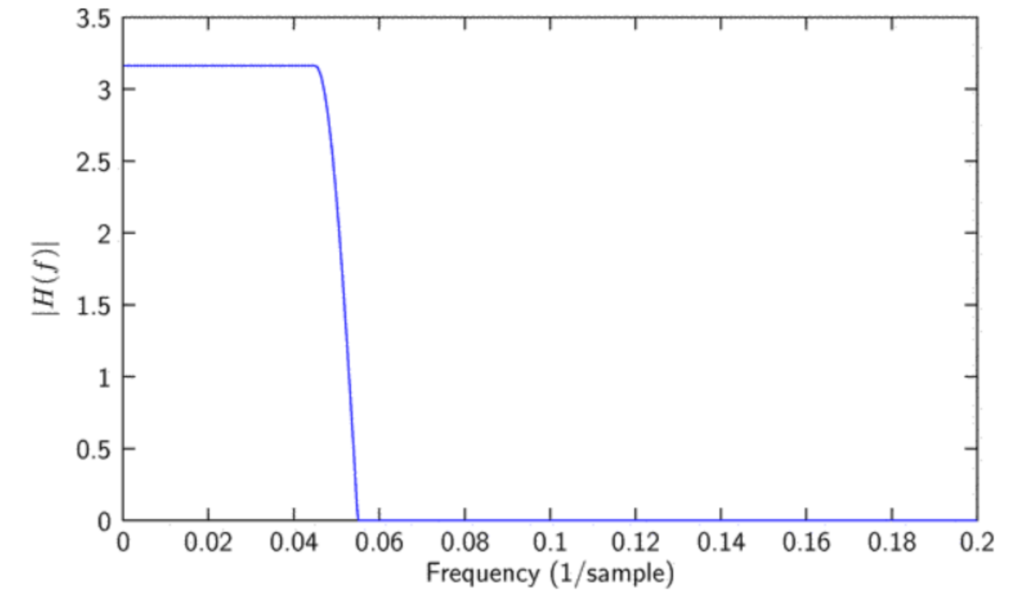


In the diagram, the frequency is in the discrete manner, i.e.,

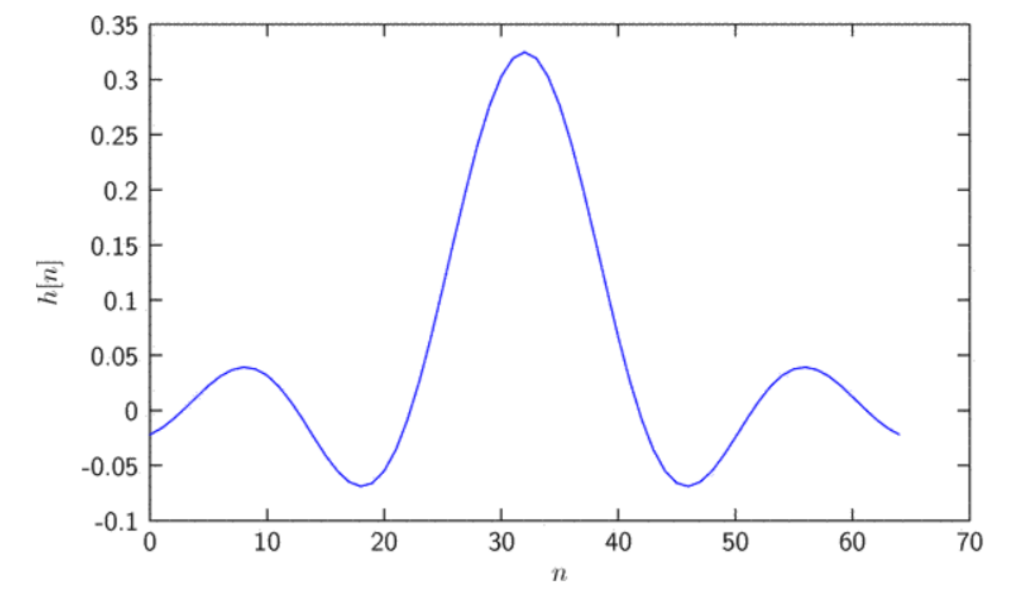
f = 0.05 => T = 1/0.05 = 20 samples per cycle

Design filter:

1. According to the requirement, design H_k
2. IDFT: H_k=> h_k
3. Apply (dot-wise multiplying) a window to h_k

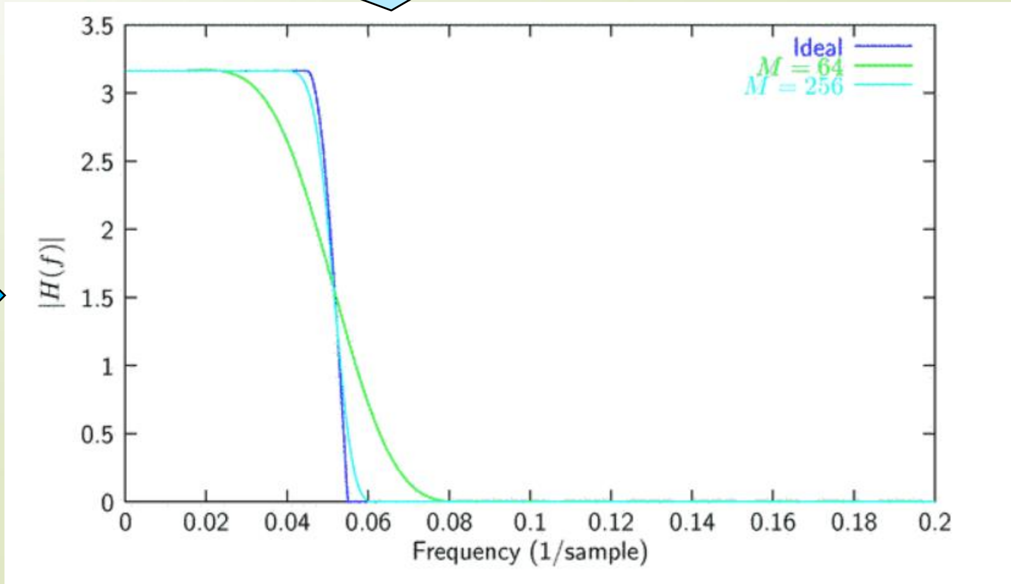


IDFT and applying a rectantular window on h_k

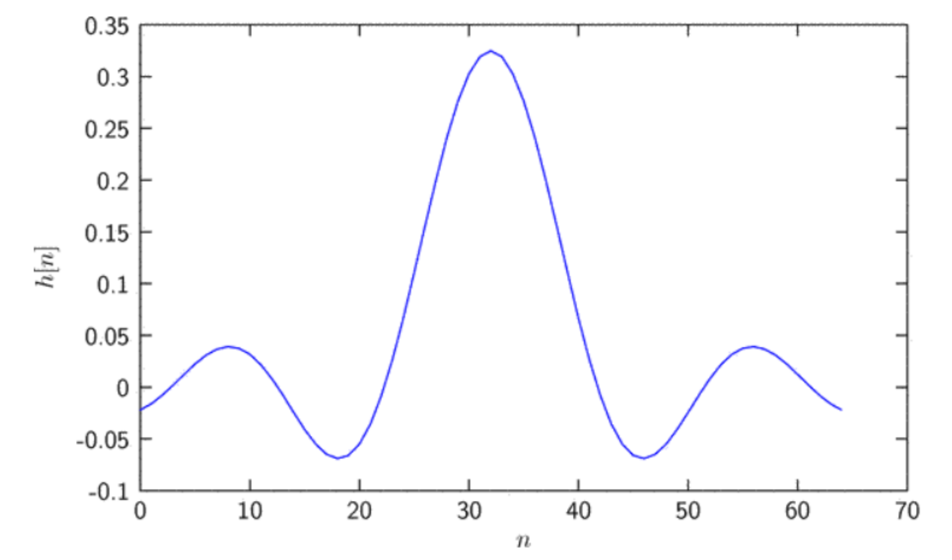


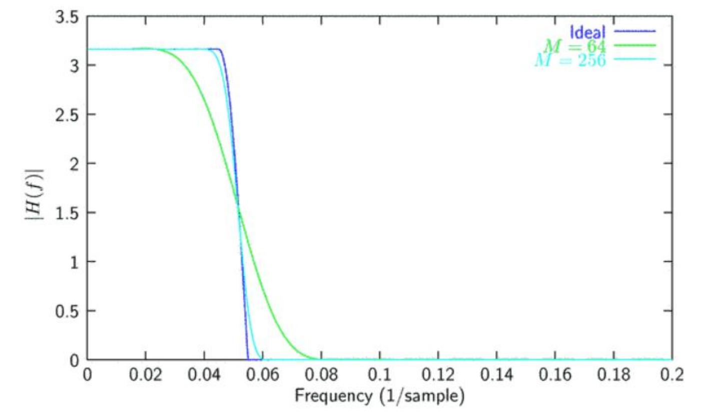
DFT

The frequency property of the filter after applying rectantular windows of different lenghth on h_k

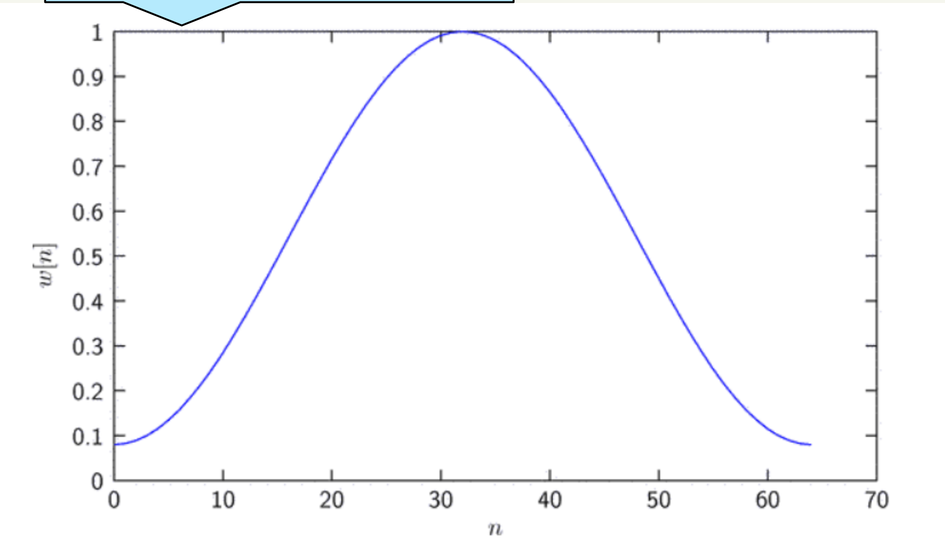


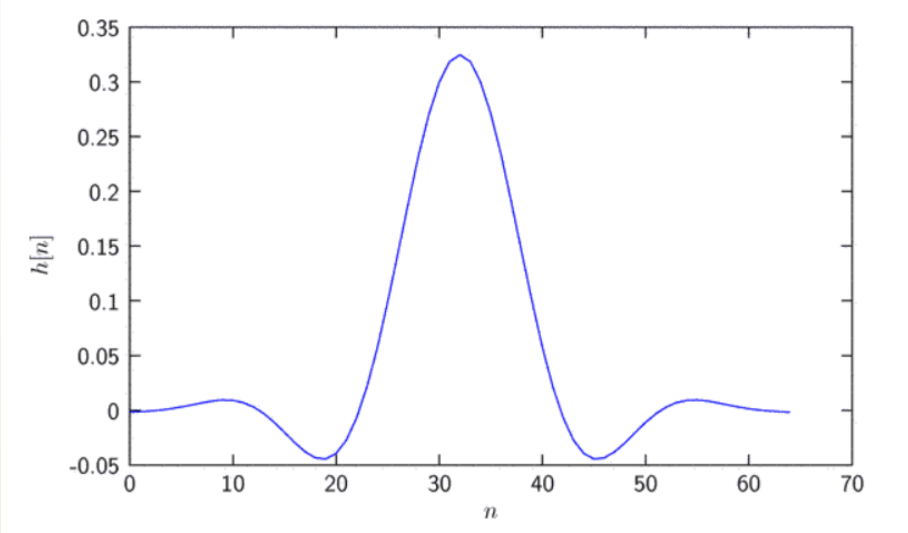
Disadvantage of a rectangular window: frequencies in the stop band may not be attenuated as much as we want





Hamming Window





Task1: Read the chapter **FIR Filter Design (Window Method)** at webpage http://dsp-fhu.user.jacobs-university.de/?page_id=142 (password:dsp2015) and understand how to design an FIR filter by window method. Describe the procedure to design an FIR filter in the lab report.

Task2: Down load dsp_lab.zip from http://dsp-fhu.user.jacobs-university.de/golden/dsp_lab.zip (password:dsp2015) Extract dsp_lab.zip and read the file „win_method.m“, explain what „Hp“ denotes in „win_method.m“ and which lines of the code try to calculate the time domain filter‘s coefficients h_k by IDFT .

Note that the formula for IDFT is:

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n \cdot e^{\frac{j2\pi kn}{N}}, \qquad n,k \in Z$$

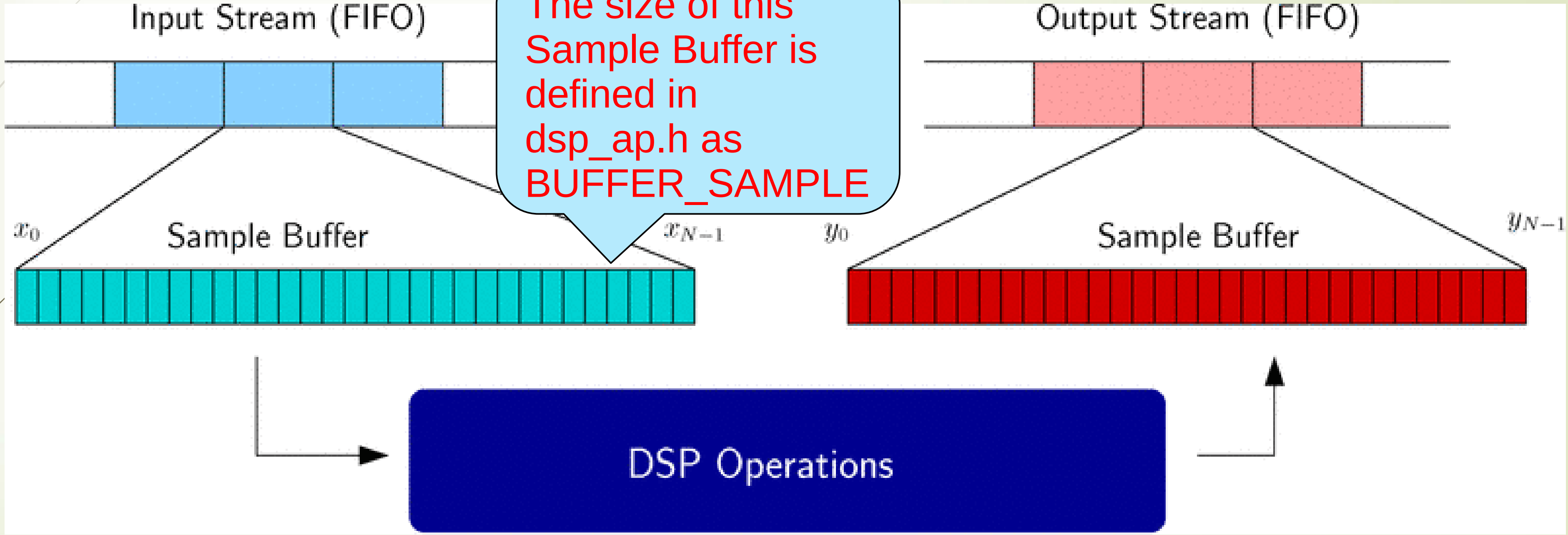
Use window method to design a Raised Cosine Filter

Task3: Read *Getting Filter Coefficients into C at* webpage
http://dsp-fhu.user.jacobs-university.de/?page_id=147 (password:dsp2015)
Read the file „make_fir.m“ and explain how the filter time domain coefficients to be produced in the file „make_fir.m“ and which file these coefficients will be stored after running the file.

Note that a fully understand of the file „win_method.m“ is not required. But if you want to fully understand how „win_method()“ can produce the coefficients for the raised cosine filter, you can try to understand the „eval()“ function in „win_method.c“ and try to understand the file „rc_filt.m“ (which produces frequency domain coefficients for raised cosine filter) in dsp_lab.zip.

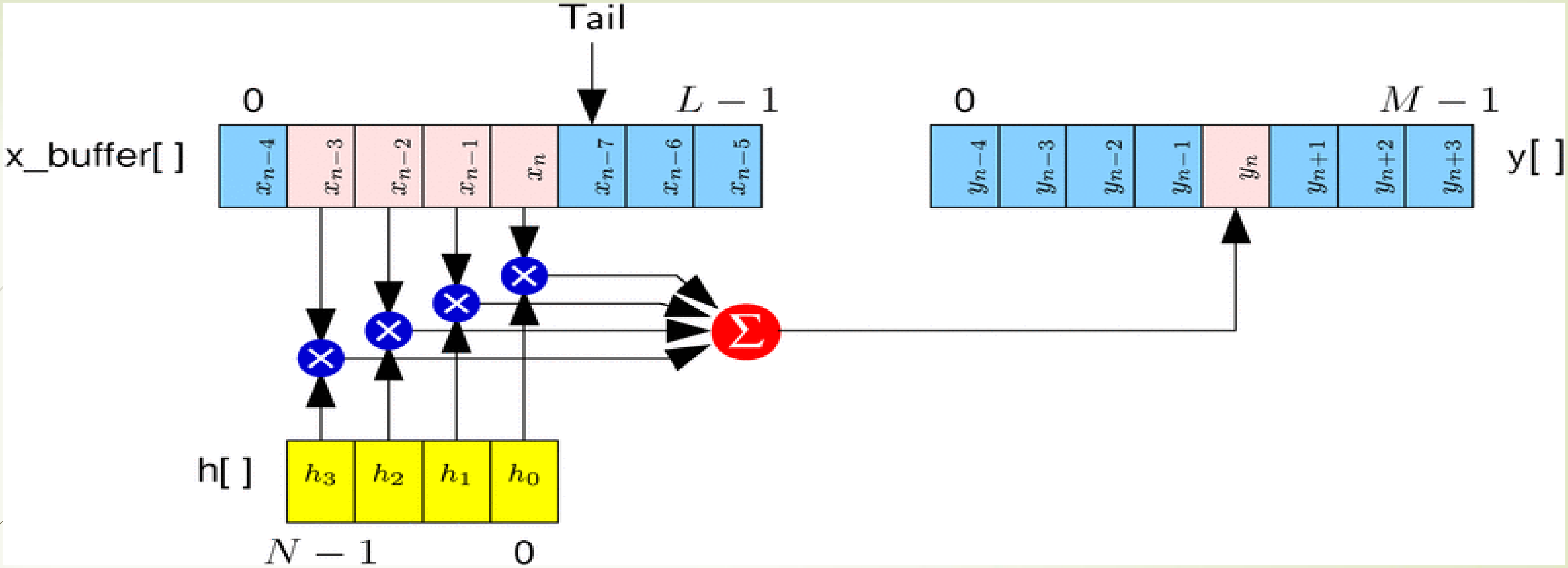
Task4: Copy the file „make_fir.m“ into the same directory where you extract dsp_lab.zip. Run the matlab file „make_fir.m“ and provide the resulting filter coefficients in your report or in a seperate file.

FIR Project



The size of this Sample Buffer is defined in dsp_ap.h as BUFFER_SAMPLE

1. Move input x[] into a circular buffer, the circular buffer size can be defined in <fir.h> as FIR_BUFFER_SIZE
2. Process the input x[], result in y = x convolve h
3. Move out the results to y[]



for n=0:M-1, (note M is size of input/output buffers)

1. Move the nth sample into the circular buffer (at tail)
2. Increment tail
3. ptr = tail-1
   sum = 0.0
4. for l=0:N-1,
   sum = sum + x_buffer[ptr]*h[l]
   ptr = ptr - 1
5. Move sum into the output position
   y[n] = sum;

end

Note that any increment/decrement operations on ptr and tail must be done in a circular fashion! Decrement can be done with the operation
ptr = (ptr + L-1) AND (L-1).
Task 5: Explain why we don't just use ptr-1 here?

Task6: Read the file „fir.h" at the webpage http://dsp-fhu.user.jacobs-university.de/?page_id=147 (password:dsp2015) and understand the code. Comment the following part:

```
typedef struct {
      float buffer[FIR_BUFFER_SIZE];
      float len;
      float *h;
      unsigned int t; }
fir_state_def;
```

Task7: Similiar to delay.c, write your own fir.c where you need write two functions:

```
fir_state_def *fir_init(int len, float *h);     // to allocate memory for necessary viarables
void fir(fir_state_def *s, const float x_in[], float y_out[]);   // implement FIR
```

Task8: Similiar to dsp_ap.c in your delay project, write your dsp_ap.c.
You need to initialize your filter and implement your filter there.