Chapter 14
Finite Impulse Response (FIR) Filters

Learning Objectives

- **Introduction to the theory behind FIR filters:**
  - **Properties (including aliasing).**
  - **Coefficient calculation.**
  - **Structure selection.**
- **Implementation in Matlab, C, assembly and linear assembly.**

Introduction

- Amongst all the obvious advantages that digital filters offer, the FIR filter can guarantee linear phase characteristics.
- Neither analogue or IIR filters can achieve this.
- There are many commercially available software packages for filter design. However, without basic theoretical knowledge of the FIR filter, it will be difficult to use them.

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

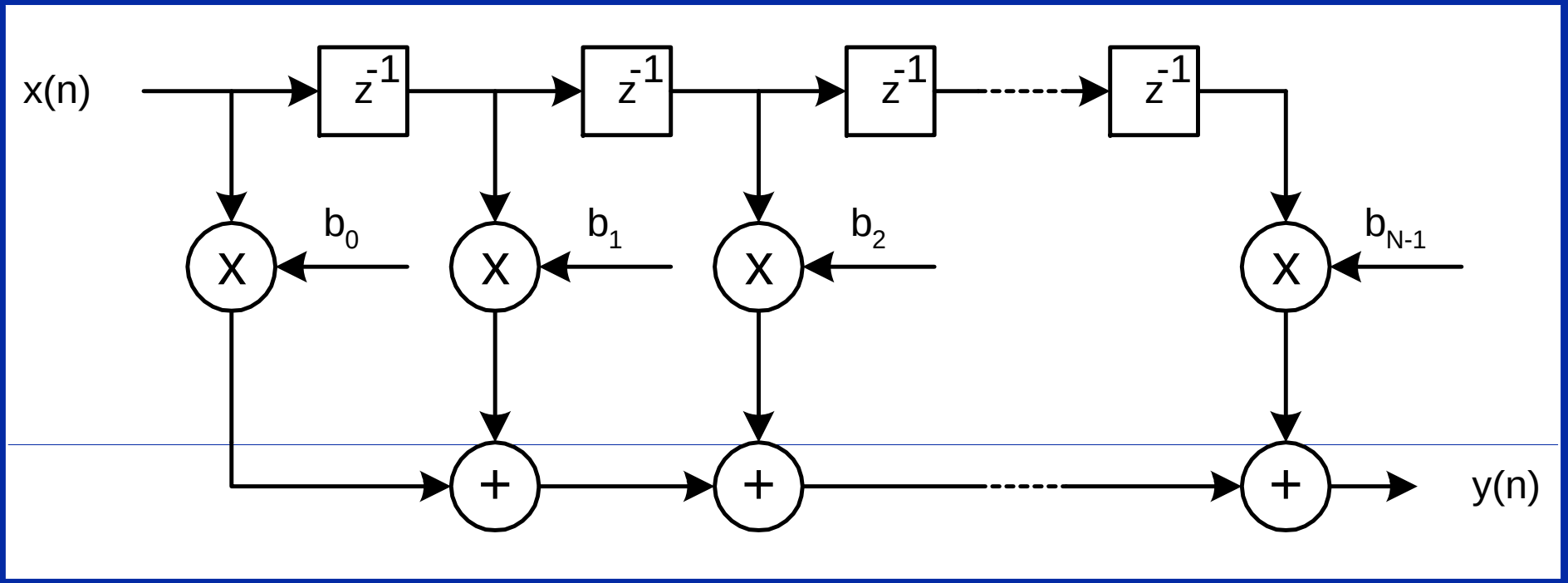
x[n]	**represents the filter input,**
b_k	**represents the filter coefficients,**
y[n]	**represents the filter output,**
N	**is the number of filter coefficients (order of the filter).**

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

FIR equation



Filter structure

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

- **If the signal x[n] is replaced by an impulse δ[n] then:**

$$y[n] = \sum_{k=0}^{N-1} b_k \delta[n-k]$$

$$y[0] = b_0\delta[0] + b_1\delta[-1] + \cdots + b_k\delta[-N]$$

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

- **If the signal x[n] is replaced by an impulse δ[n] then:**

$$y[n] = \sum_{k=0}^{N-1} b_k \delta[n-k]$$

$$y[n] = b_0\delta[n] + b_1\delta[n-1] + \cdots + b_k\delta[n-N]$$

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

- **If the signal x[n] is replaced by an impulse δ[n] then:**

$$y[n] = \sum_{k=0}^{N-1} b_k \delta[n-k]$$

$$\delta[n-k] = \begin{cases} 1 \text{ for } n = k \\ 0 \text{ for } n \neq k \end{cases}$$

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

- **Finally:**

$$b_0 = h[0]$$
$$b_1 = h[1]$$
$$\vdots$$
$$b_k = h[k]$$

Properties of an FIR Filter

- **Filter coefficients:**

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$

With: $b_k = h[k]$

- **The coefficients of a filter are the same as the impulse response samples of the filter.**

Frequency Response of an FIR Filter

- By taking the z-transform of h[n], H(z):

$$H(z) = \sum_{n=0}^{N-1} h[n] z^{-n}$$

- Replacing z by $e^{-j\omega}$ in order to find the frequency response leads to:

$$H(z)\big|_{z=e^{j\omega}} = H(\omega) = \sum_{n=0}^{N-1} h[n] e^{-jn\omega}$$

Frequency Response of an FIR Filter

- **Since $e^{-j2\pi k} = 1$ then:**

$$H(\omega + 2\pi) = \sum_{n=0}^{N-1} h[n] e^{-jn(\omega+2\pi)} = \sum_{n=0}^{N-1} h[n] e^{-jn\omega}$$

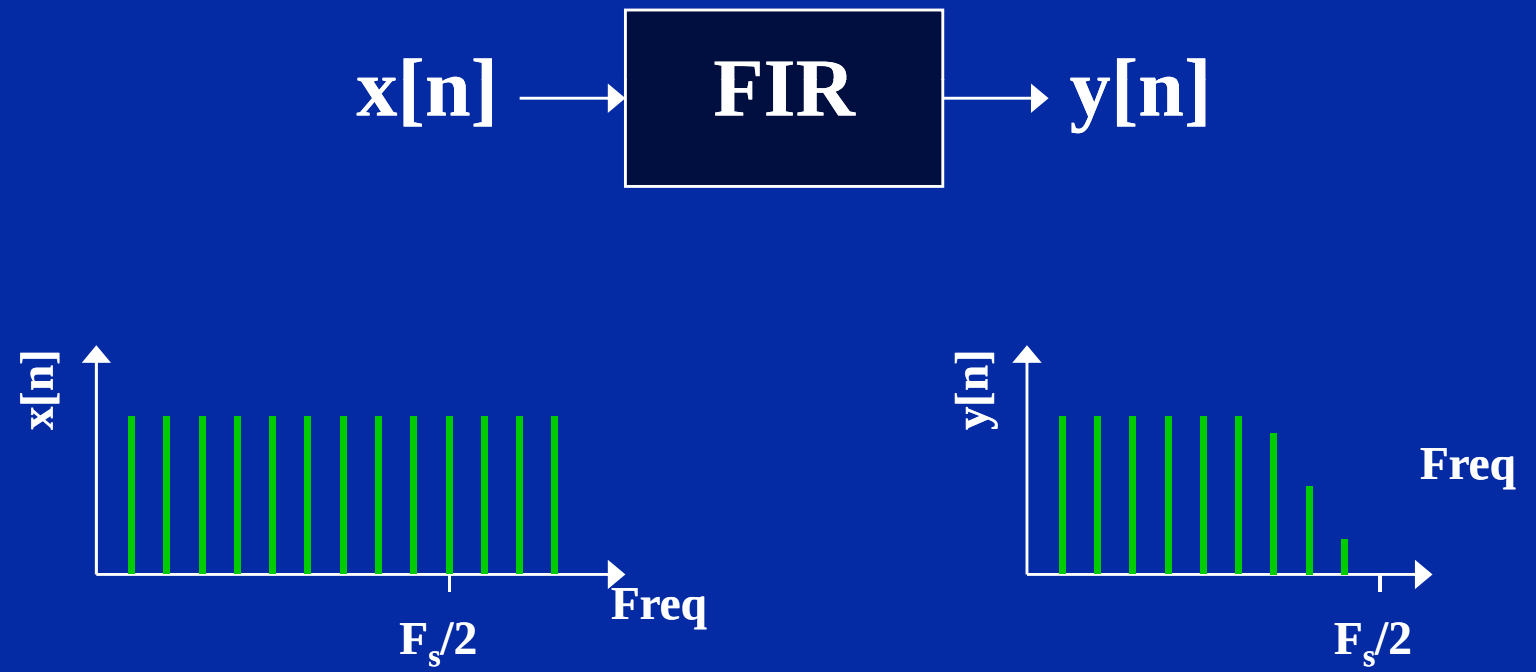
- **Therefore:**

$$H(\omega + 2k\pi) = H(\omega)$$

- **FIR filters have a periodic frequency response and the period is 2π.**

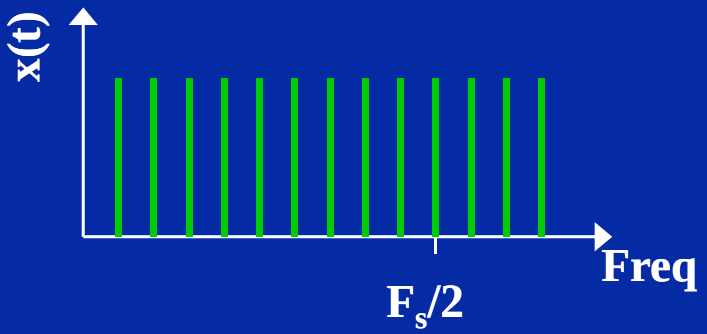
Frequency Response of an FIR Filter

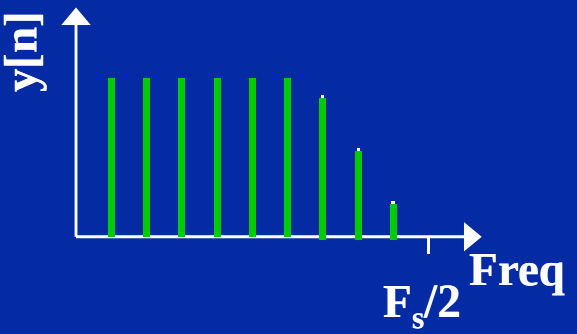
- **Frequency response:** $H(\omega + 2\pi) = H(\omega)$



Frequency Response of an FIR Filter

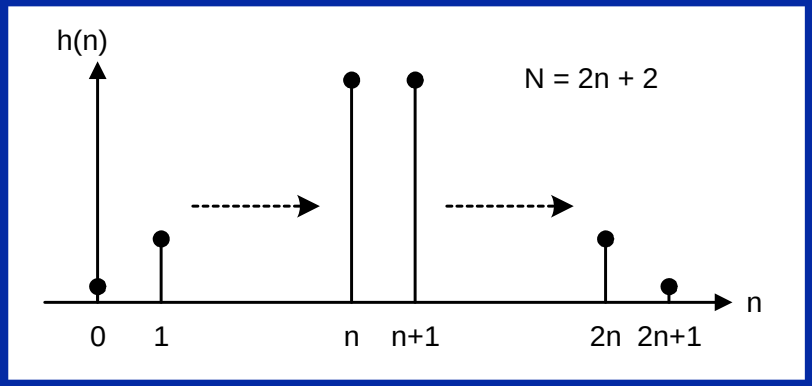
- **Solution: Use an anti-aliasing filter.**



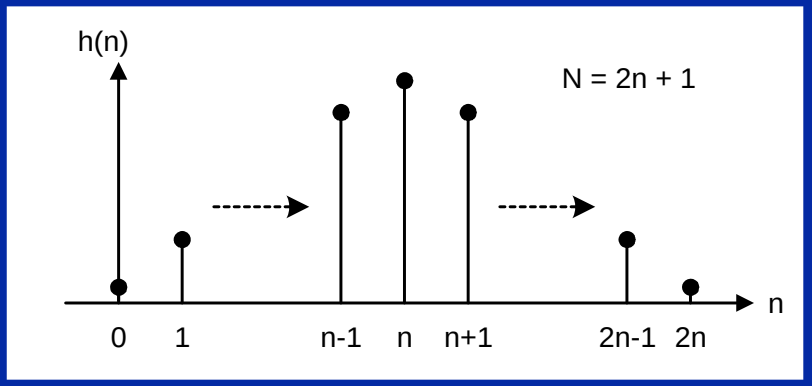


Phase Linearity of an FIR Filter

- **A causal FIR filter whose impulse response is symmetrical is guaranteed to have a linear phase response.**



Even symmetry



Odd symmetry

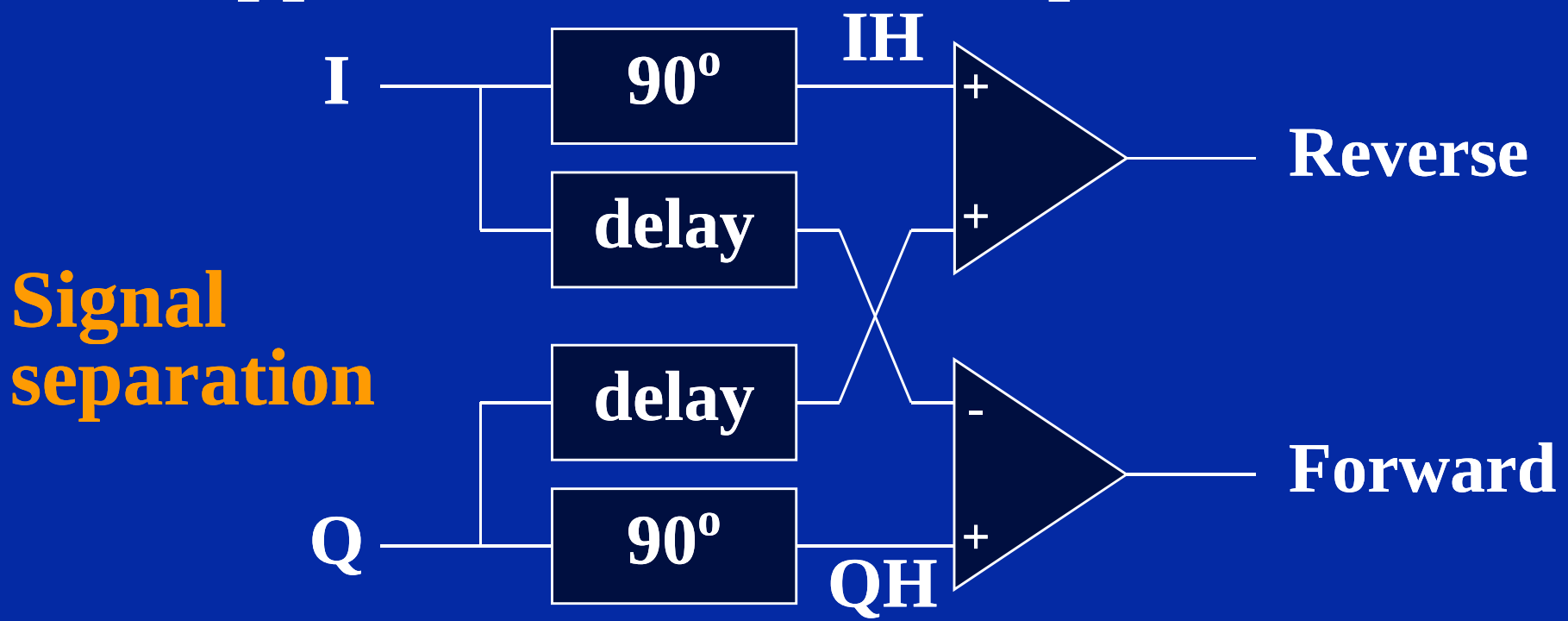
Phase Linearity of an FIR Filter

- **A causal FIR filter whose impulse response is symmetrical (ie h[n] = h[N-1-n] for n = 0, 1, …, N-1) is guaranteed to have a linear phase response.**

Condition	**Phase** $\left(k = -\frac{N-1}{2}\right)$	**Phase Property**	**Filter Type**
$h[n] = h[N-n-1]$ Positive Symmetry	$k\omega$	Linear phase	Odd Symmetry – Type 1 Even Symmetry – Type 2

Phase Linearity of an FIR Filter

- **Application of 90° linear phase shift:**



$$I = A\cos\omega_f t + B\sin\omega_r t$$

$$Q = A\sin\omega_f t + B\cos\omega_r t$$

$$IH = A\cos\left(\omega_f t + \frac{\pi}{2}\right) + B\sin\left(\omega_r t + \frac{\pi}{2}\right)$$
$$= -A\sin\omega_f t + B\cos\omega_r t$$

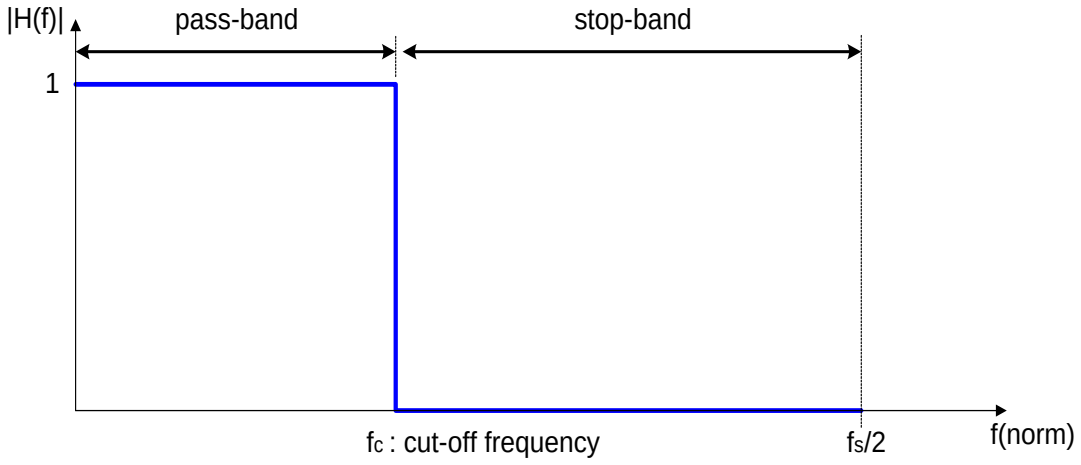
$$IH + Q = 2B\cos\omega_r t$$

$$QH - I = 2B\sin\omega_f t$$

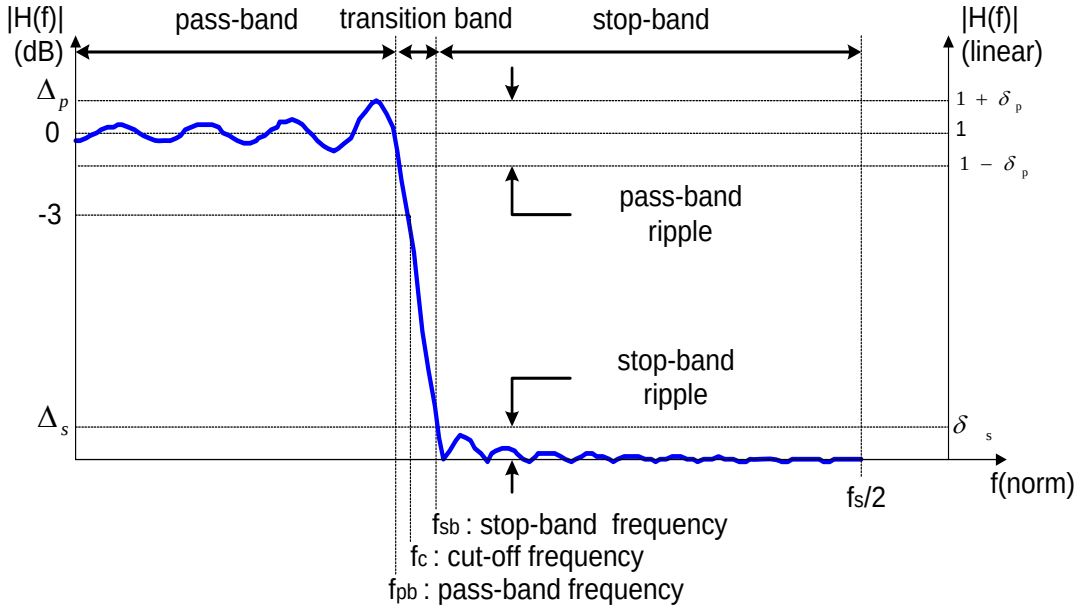
Design Procedure

- To fully design and implement a filter five steps are required:
  - (1) Filter specification.
  - (2) Coefficient calculation.
  - (3) Structure selection.
  - (4) Simulation (optional).
  - (5) Implementation.

Filter Specification - Step 1



(a)



(b)

Coefficient Calculation - Step 2

- There are several different methods available, the most popular are:
  - Window method.
  - Frequency sampling.
  - Parks-McClellan.
- We will just consider the window method.

Window Method

- First stage of this method is to calculate the coefficients of the ideal filter.
- This is calculated as follows:

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{j\omega n}d\omega$$

$$= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} 1\cdot e^{j\omega n}d\omega$$

$$= \begin{cases} \dfrac{2f_c \sin(n\omega_c)}{n\omega_c} & \text{for n} \neq 0 \\ 2f_c & \text{for n} = 0 \end{cases}$$

Window Method

- Second stage of this method is to select a window function based on the passband or attenuation specifications, then determine the filter length based on the required width of the transition band.

Window Type	Normalised Transition Width (Δf(Hz))	Passband Ripple(dB)	Stopband Attenuation (dB)
Rectangular	$\frac{0.9}{N}$	0.7416	21
Hanning	$\frac{3.1}{N}$	0.0546	44
Hamming	$\frac{3.3}{N}$	0.0194	53
Blackman	$\frac{5.5}{N}$	0.0017	74
Kaiser	$\frac{2.93}{N} \rightarrow \beta = 4.54$ $\frac{5.71}{N} \rightarrow \beta = 8.96$	0.0274 0.000275	50 90

Using the Hamming Window:

$$N = \frac{3.3}{\Delta f} = \frac{3.3}{(1.2-1.4)kHz} \cdot 8kHz = 132$$

Window Method

- **The third stage is to calculate the set of truncated or windowed impulse response coefficients, h[n]:**

$$h(n) = h_d(n) \cdot W(n) \quad \textbf{for} \quad \begin{cases} -\dfrac{N-1}{2} \le n \le \dfrac{N-1}{2} & \text{for N = odd} \\ -\dfrac{N}{2} \le n \le \dfrac{N}{2} & \text{for N = even} \end{cases}$$

Where:

$$W(n) = 0.54 + 0.46\cos\left(\frac{2\pi n}{N}\right)$$

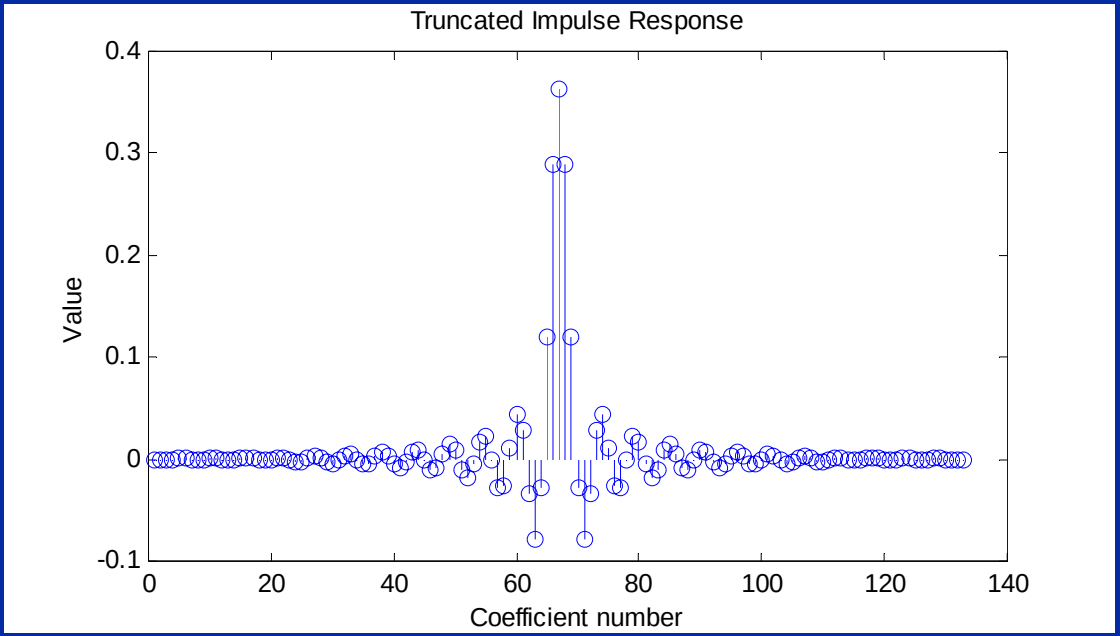
$$= 0.54 + 0.46\cos\left(\frac{2\pi n}{133}\right) \quad \textbf{for} \quad -66 \le n \le 66$$

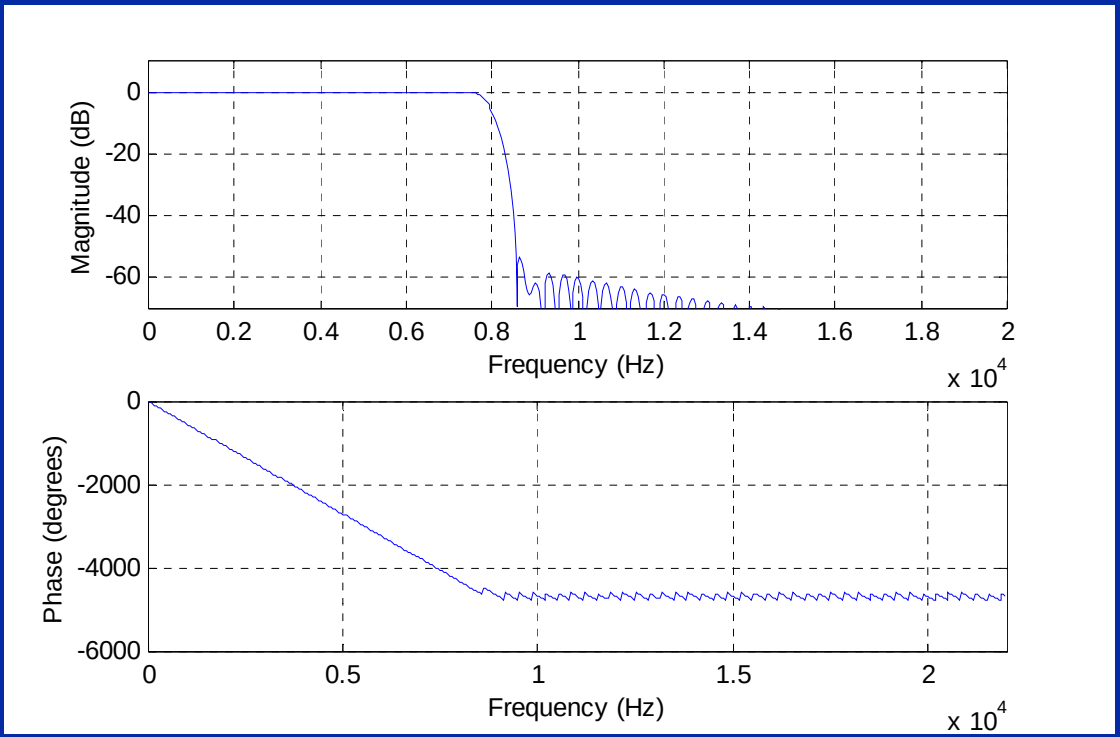
Window Method

- **Matlab code for calculating coefficients:**

```
close all;
clear all;

fc = 8000/44100;                        % cut-off frequency
N = 133;                                % number of taps
n = -((N-1)/2):((N-1)/2);
n = n+(n==0)*eps;                       % avoiding division by zero

[h] = sin(n*2*pi*fc)./(n*pi);           % generate sequence of ideal coefficients
[w] = 0.54 + 0.46*cos(2*pi*n/N);        % generate window function
d = h.*w;                               % window the ideal coefficients

[g,f] = freqz(d,1,512,44100);           % transform into frequency domain for plotting

figure(1)
plot(f,20*log10(abs(g)));               % plot transfer function
axis([0  2*10^4 -70 10]);

figure(2);
stem(d);                                % plot coefficient values
xlabel('Coefficient number');
ylabel ('Value');
title('Truncated Impulse Response');

figure(3)
freqz(d,1,512,44100);                   % use freqz to plot magnitude and phase response
axis([0  2*10^4 -70 10]);
```

Window Method





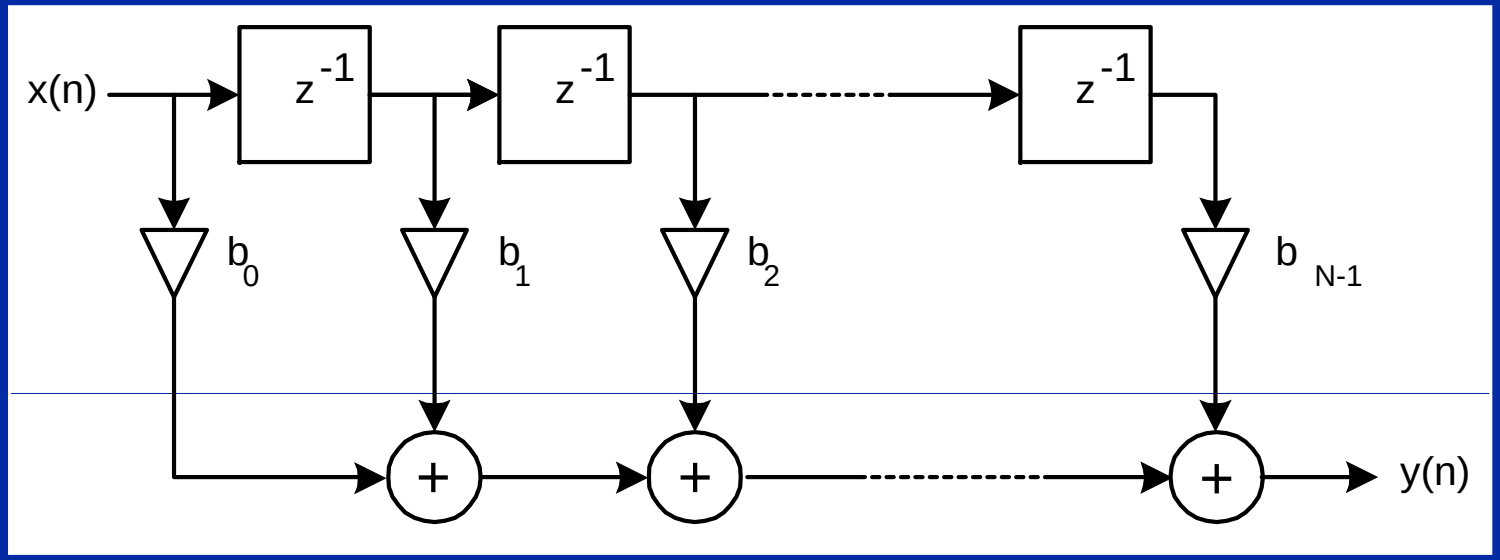
Realisation Structure Selection - Step 3

- **Direct form structure for an FIR filter:**

$$H(z) = \sum_{k=0}^{N-1} b_k z^{-k}$$

$$Y(z) = H(z) \cdot X(z)$$

$$y(n) = b_0 x(n) + b_1 x(n-1) + ... + b_{N-1} x(n-N+1)$$



Realisation Structure Selection - Step 3

- **Direct form structure for an FIR filter:**

$$H(z)=\sum_{k=0}^{N-1} b_k z^{-k}$$

- **Linear phase structures:**
  - **N even:**

$$H(z)=\sum_{k=0}^{\frac{N}{2}-1} b_k \left(z^{-k}+z^{N-k-1}\right)$$

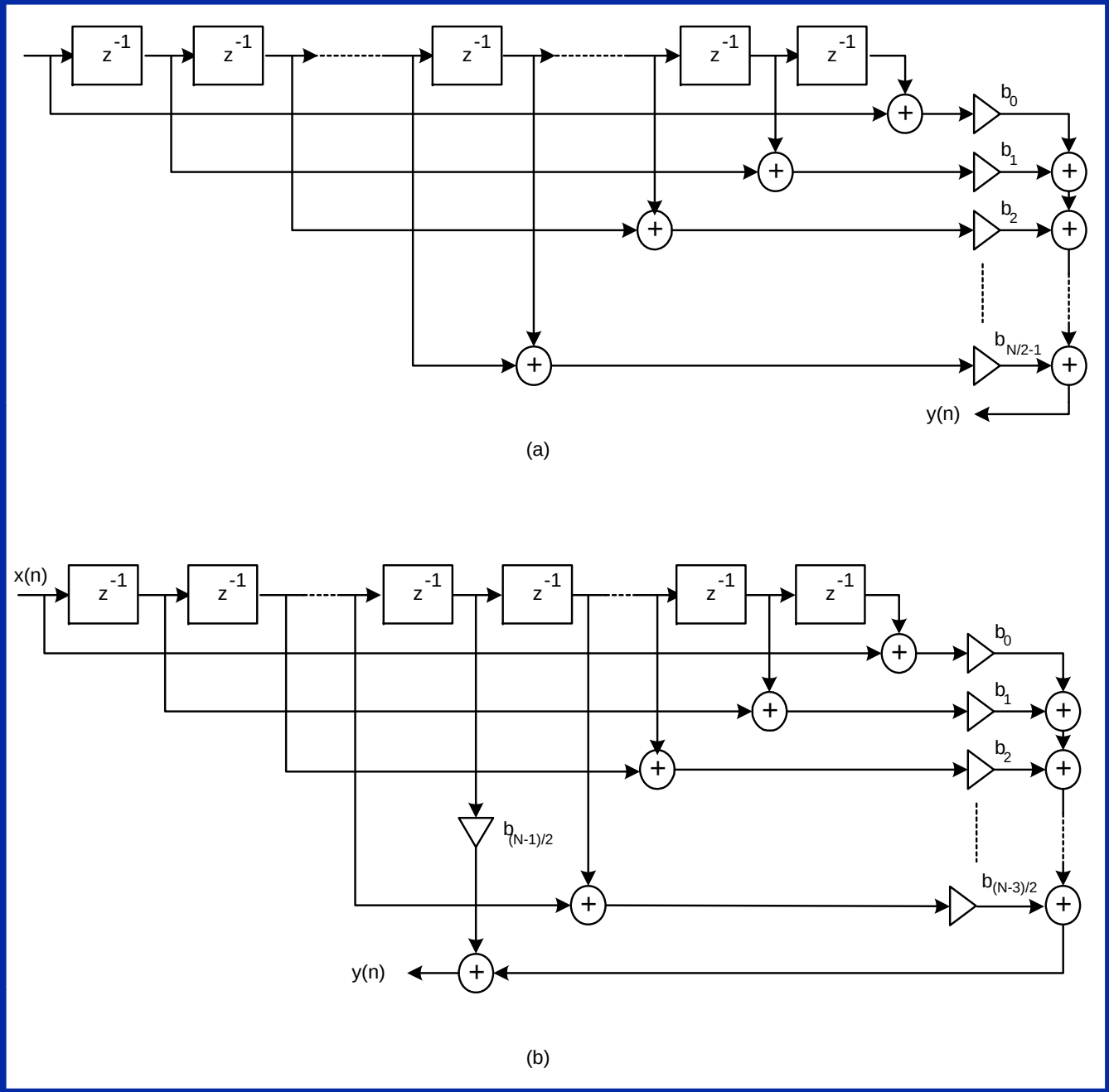
  - **N Odd:**

$$H(z)=\sum_{k=0}^{\frac{N-1}{2}} b_k \left(z^{-k}+z^{N-k-1}\right)+b_{\frac{N-1}{2}} z^{-\frac{N-1}{2}}$$

Realisation Structure Selection - Step 3

(a) N even.

(b) N odd.



Realisation Structure Selection - Step 3

- **Direct form structure for an FIR filter:**

$$H(z)=\sum_{k=0}^{N-1} b_k z^{-k}$$

- **Cascade structures:**

$$H(z)=\sum_{k=0}^{N-1} b_k z^{-k} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{N-1} z^{-(N-1)}$$

$$= b_0 \left[1 + \frac{b_1}{b_0} z^{-1} + \frac{b_2}{b_0} z^{-2} + \ldots + \frac{b_{N-1}}{b_0} z^{-(N-1)} \right]$$

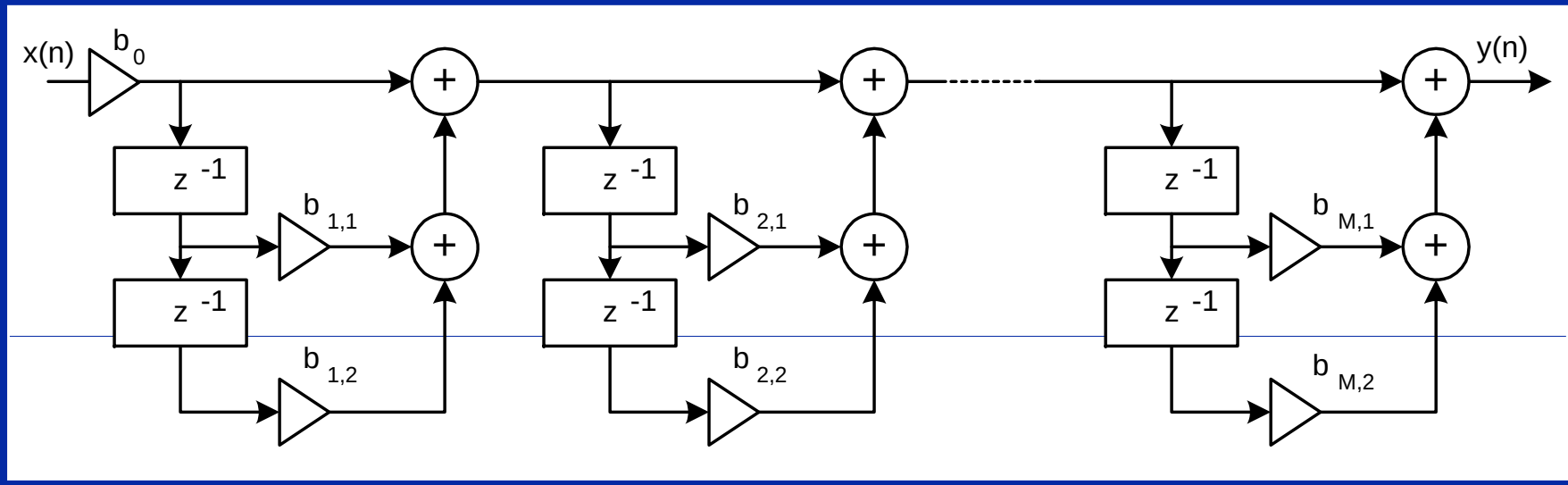
$$= b_0 \prod_{k=1}^{M} \left(1 + b_{k,1} z^{-1} + b_{k,2} z^{-2} \right)$$

Realisation Structure Selection - Step 3

- **Direct form structure for an FIR filter:**

$$H(z) = \sum_{k=0}^{N-1} b_k z^{-k}$$

- **Cascade structures:**



Implementation - Step 5

- **Implementation procedure in 'C' with fixed-point:**
  - **Set up the codec (\Links\CodecSetup.pdf).**
  - **Transform:** $$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$ **to 'C' code.**

    (\Links\FIRFixed.pdf)
  - **Configure timer 1 to generate an interrupt at 8000Hz (\Links\TimerSetup.pdf).**
  - **Set the interrupt generator to generate an interrupt to invoke the Interrupt Service Routine (ISR) (\Links\InterruptSetup.pdf).**

Implementation - Step 5

- **Implementation procedure in ‘C’ with floating-point:**

  Same set up as fixed-point plus:

  - **Convert the input signal to floating-point format.**
  - **Convert the coefficients to floating-point format.**
  - **With floating-point multiplications there is no need for the shift required when using Q15 format.**

- **See \Links\FIRFloat.pdf**

Implementation - Step 5

- **Implementation procedure in assembly:**

  Same set up as fixed-point, however:

  - $$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k]$$ **is written in assembly.**

    (\Links\FIRFixedAsm.pdf)

  - **The ISR is now declared as external.**

Implementation - Step 5

- **Implementation procedure in assembly:**

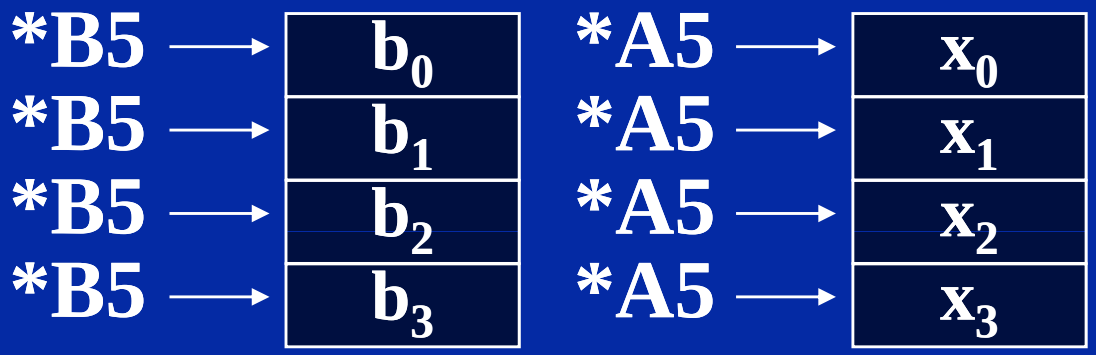
  The filter implementation in assembly is now using circular addressing and therefore:

  - **The circular pointers and block size register are selected and initialised by setting the appropriate values of the AMR bit fields.**
  - **The data is now aligned using:**

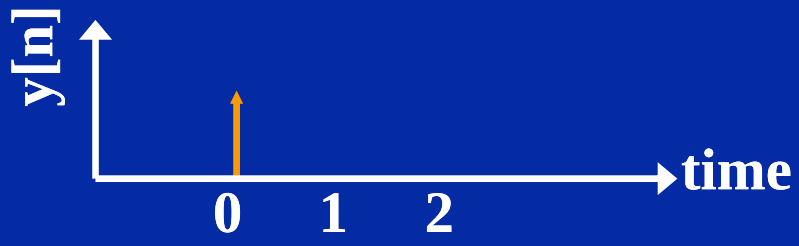
```
#pragma DATA_ALIGN (symbol, constant (bytes))
```

  - **Set the initial value of the circular pointers, see \Links\FIRFixedAsm.pdf.**

Implementation - Step 5

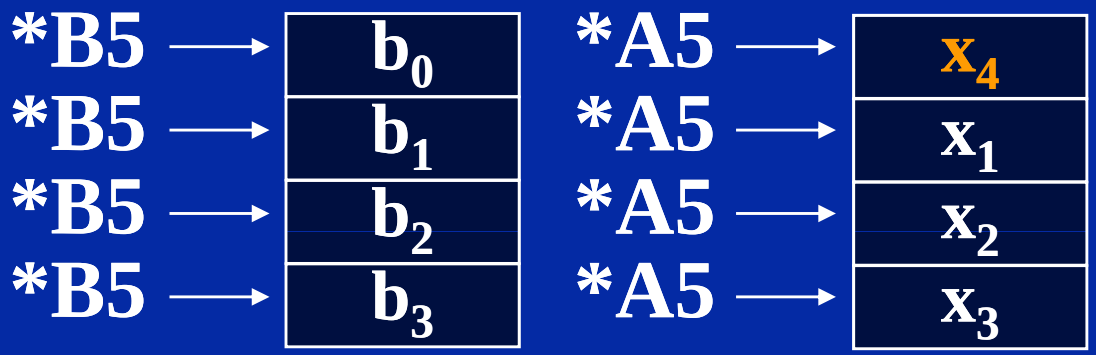


$y0 = b0*x0 + b1*x1 + b2*x2 + b3*x3$



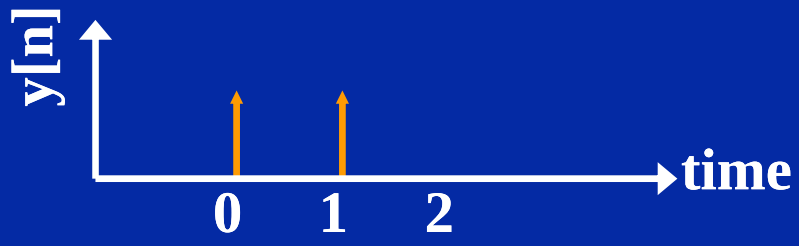
Circular addressing link slide.

Implementation - Step 5



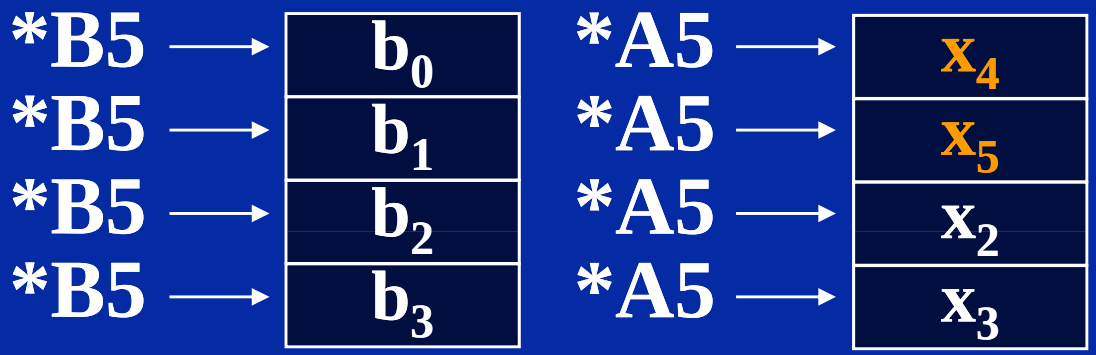
$y0 = b0*x0 + b1*x1 + b2*x2 + b3*x3$

$y1 = b0*x1 + b1*x2 + b2*x3 + b3*x4$



Circular addressing link slide.

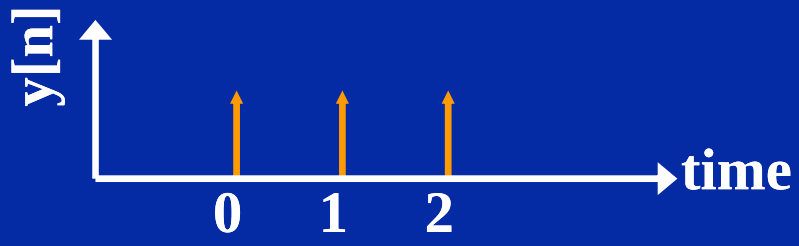
Implementation - Step 5



$$y0 = b0*x0 + b1*x1 + b2*x2 + b3*x3$$

$$y1 = b0*x1 + b1*x2 + b2*x3 + b3*x4$$

$$y2 = b0*x2 + b1*x3 + b2*x4 + b3*x5$$



Circular addressing link slide.

FIR Code

- **Code location:**
  - **Code\Chapter 14 - Finite Impulse Response Filters**
- **Projects:**
  - **Fixed Point in C: \FIR_C_Fixed**
  - **Floating Point in C: \FIR_C_Float**
  - **Fixed Point in Assembly: \FIR_Asm_Fixed**
  - **Floating Point in Assembly: \FIR_Asm_Float**

Chapter 14
Finite Impulse Response (FIR) Filters
- End -