DLTI Convolution (1B)

- Circular Convolution
- Numerical Convolution
- Moving Average Filter

Young Won Lim
11/18/19

Please send corrections (or suggestions) to youngwlim@hotmail.com.
This document was produced by using LibreOffice.

Based on

Introduction to Signal Processing

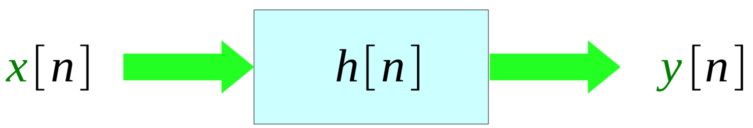
S. J. Ofranidis

The necessities in DSP C Programming

FIR Filter (A.pdf) 20191114

Linear Convolution using the DFT

$$y[n] = \sum_{k=-\infty}^{+\infty} h[k]\, x[n-k]$$

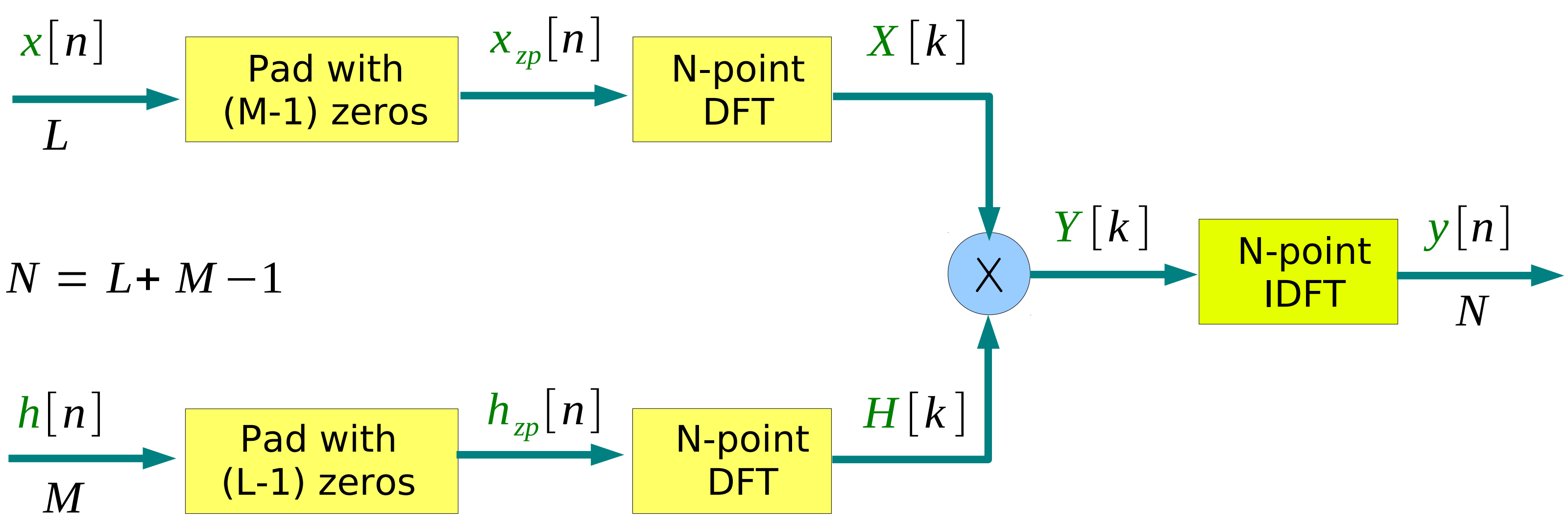


$x[n] \quad 0 \le n \le L-1$

$h[n] \quad 0 \le n \le M-1$

$y[n] \quad 0 \le n \le L+M-2$

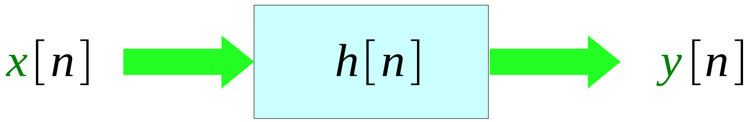
$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$$



Circular Convolution using the DFT

$$y[n] = \sum_{k=-\infty}^{+\infty} h[k]\, x[n-k]$$

Linear Convolution



$x[n] \qquad 0 \le n \le L-1$

$h[n] \qquad 0 \le n \le M-1$

$y[n] \qquad 0 \le n \le L+M-2$

$$Y(e^{j\hat{\omega}}) = H(e^{j\hat{\omega}})X(e^{j\hat{\omega}})$$

DTFT $\quad \omega \quad$ continuous frequency variable

$Y[k] = Y(e^{j2\pi k/N}) \qquad 0 \le k \le N-1$

$H[k] = H(e^{j2\pi k/N}) \qquad N-M \quad 0's$

$X[k] = X(e^{j2\pi k/N}) \qquad N-L \quad 0's$

N-point DFT $\quad k \quad$ sampled frequency variable

$$Y[k] = H[k]X[k]$$

$$y[n] = \sum_{m=0}^{N-1} h[m]\, x[\langle n-k \rangle_N] \qquad 0 \le n \le N-1$$

Circular Convolution

$$y[n] = h[n] \; \textcircled{N} \; x[n]$$

Circular Convolution using the DFT

Linear Convolution

$$y[n] = h[n] * x[n]$$

DTFT

$$Y(e^{j\hat{\omega}}) = H(e^{j\hat{\omega}})X(e^{j\hat{\omega}})$$

$$y[n] = \sum_{k=-\infty}^{+\infty} h[k]\, x[n-k]$$

ω continuous frequency variable

Circular Convolution

N-point DFT

$$y[n] = h[n] \circledN x[n]$$

$$Y[k] = H[k]X[k]$$

$$y[n] = \sum_{m=0}^{N-1} h[m]\, x[\langle n-k\rangle_N]$$

$$0 \le n \le N-1$$

k sampled frequency variable

Circular Convolution (1)

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} H[k]X[k]W_N^{-kn}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1}\left\{\sum_{m=0}^{N-1} h[m]W_N^{+km}\right\}\left\{\sum_{l=0}^{N-1} x[l]W_N^{+kl}\right\}W_N^{-kn}$$

$$= \sum_{m=0}^{N-1} h[m]\sum_{l=0}^{N-1} x[l]\left\{\frac{1}{N}\sum_{k=0}^{N-1} W_N^{+k(m+l-n)}\right\}$$

$$\left\{\frac{1}{N}\sum_{k=0}^{N-1} W_N^{+k(m+l-n)}\right\} = \begin{cases} 1, & m+l-n = rN \\ 0, & otherwise \end{cases}$$

$$l = n-m+rN = (n-m) \bmod N = \langle n-m \rangle_N$$

Circular Convolution (1)

$$X_1[k] = \sum_{n=0}^{N-1} x_1[n] e^{-j2\pi nk/N} \qquad k = 0, 1, \cdots, N-1$$

$$X_2[k] = \sum_{n=0}^{N-1} x_2[n] e^{-j2\pi nk/N} \qquad k = 0, 1, \cdots, N-1$$

$$X_3[k] = X_1[k] X_2[k] \qquad k = 0, 1, \cdots, N-1$$

$$x_3[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_3[k] e^{+j2\pi km/N}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X_1[k] X_2[k] e^{+j2\pi km/N}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \left\{ \sum_{n=0}^{N-1} x_1[n] e^{-j2\pi kn/N} \right\} \left\{ \sum_{l=0}^{N-1} x_2[l] e^{-j2\pi kl/N} \right\} e^{+j2\pi km/N}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} x_1[n] \sum_{l=0}^{N-1} x_2[l] \sum_{k=0}^{N-1} e^{-j2\pi kn/N} e^{-j2\pi kl/N} e^{+j2\pi km/N}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} x_1[n] \sum_{l=0}^{N-1} x_2[l] \sum_{k=0}^{N-1} e^{+j2\pi k(m-n-l)/N}$$

Circular Convolution (2)

$$X_1[k] = \sum_{n=0}^{N-1} x_1[n] e^{-j2\pi nk/N}$$

$$X_2[k] = \sum_{n=0}^{N-1} x_2[n] e^{-j2\pi nk/N} \qquad k = 0, 1, \cdots, N-1$$

$$X_3[k] = X_1[k] X_2[k] \qquad k = 0, 1, \cdots, N-1$$

$$x[k] = \frac{1}{N} \sum_{n=0}^{N-1} x_1[n] \sum_{l=0}^{N-1} x_2[l] \sum_{k=0}^{N-1} e^{+j2\pi k(m-n-l)/N}$$

$$\sum_{k=0}^{N-1} a^k = N \qquad (a = 1)$$

$$= \frac{1-a^N}{1-a} \qquad (a \neq 1)$$

$$a = e^{+j2\pi(m-n-l)/N}$$

Circular Convolution (2)

$$\frac{(m-n-l)}{N} \in \left\{ \begin{array}{llll} \frac{0}{N}, & \frac{1}{N}, & \cdots, & \frac{N-1}{N}, \\ \frac{0}{N}+1, & \frac{1}{N}+1, & \cdots, & \frac{N-1}{N}+1, \\ \frac{0}{N}+2, & \frac{1}{N}+2, & \cdots, & \frac{N-1}{N}+2, \\ \vdots & \vdots & & \vdots \end{array} \right\}$$

$$e^{j\frac{2\pi(m-n-l)}{N}} \in \left\{ e^{j2\pi\cdot 0/N}, \; e^{j2\pi\cdot 1/N}, \; \cdots, \; e^{+j2\pi(N-1)/N} \right\}$$

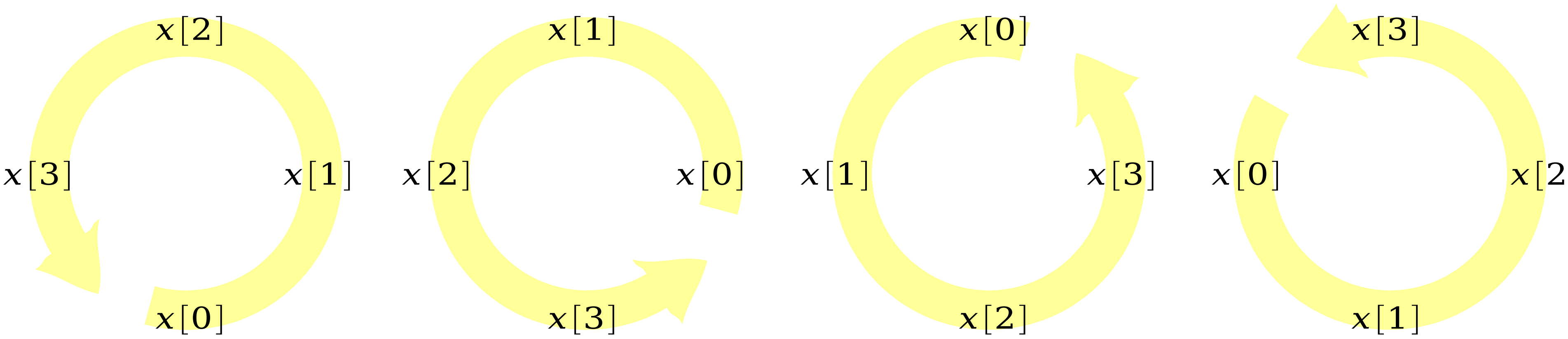
$$e^{j\frac{2\pi(m-n-l)}{N}} = e^{j\frac{2\pi((m-n-l))_N}{N}}$$

$$((m-n-l))_N = 0 \Rightarrow e^{j\frac{2\pi(m-n-l)}{N}} = e^{j\frac{2\pi((m-n-l))_N}{N}} = 1$$

$$\Rightarrow \sum_{k=0}^{N-1} e^{j\frac{2\pi k(m-n-l)}{N}} = \sum_{k=0}^{N-1} e^{j\frac{2\pi k((m-n-l))_N}{N}} = N$$

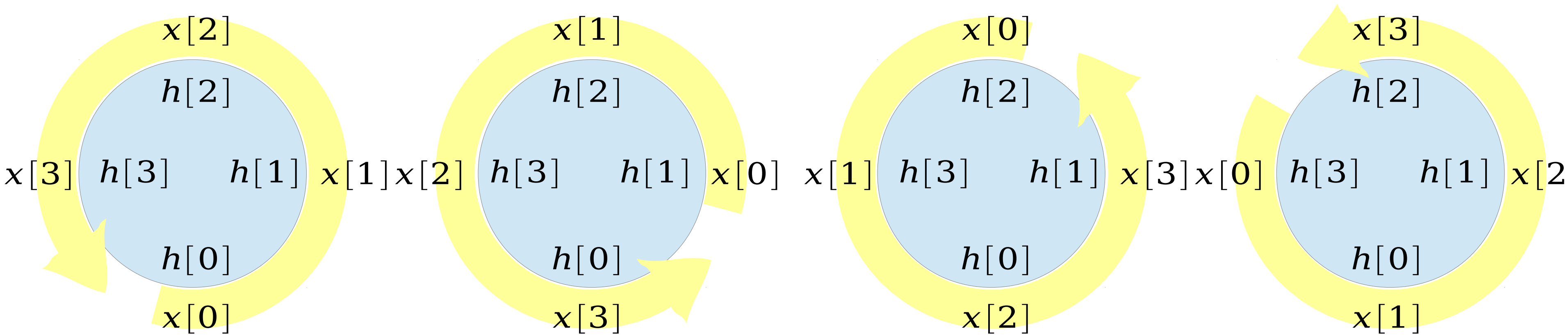
$$((m-n))_N = l$$

Circular Convolution

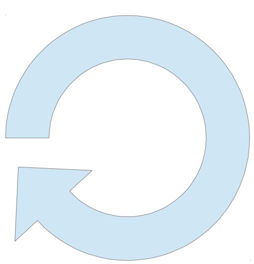


$x[\langle n-0\rangle_4]$ $x[\langle n-1\rangle_4]$ $x[\langle n-2\rangle_4]$ $x[\langle n-3\rangle_4]$

Circular Convolution



$x[\langle n-0\rangle_4]$ $x[\langle n-1\rangle_4]$ $x[\langle n-2\rangle_4]$ $x[\langle n-3\rangle_4]$



Circular Convolution

Octave conv()

```
conv (a, b)
conv (a, b, shape)
```

Convolve two vectors a and b.

The length of the output vector : length (a) + length (b) – 1

When a and b are the coefficient vectors of two polynomials,
the convolution represents the coefficient vector of the product polynomial.

The optional shape argument may be

- shape = "full"
  - Return the full convolution. (default)
- shape = "same"
  - Return the central part of the convolution with the same size as a.

See also: deconv conv2 convn fftconv

Octave fftconv()

fftconv (x, y)
fftconv (x, y, n)

Convolve two vectors using the FFT for computation.

c = fftconv (x, y) returns
a vector of length equal to length (x) + length (y) - 1.

If x and y are the coefficient vectors of two polynomials,
the returned value is the coefficient vector of the product polynomial.

The computation uses the FFT by calling the function fftfilt.
If the optional argument n is specified, an N-point FFT is used.

fftfilt (b, x, n)

With two arguments, fftfilt filters x with the FIR filter b using the FFT.

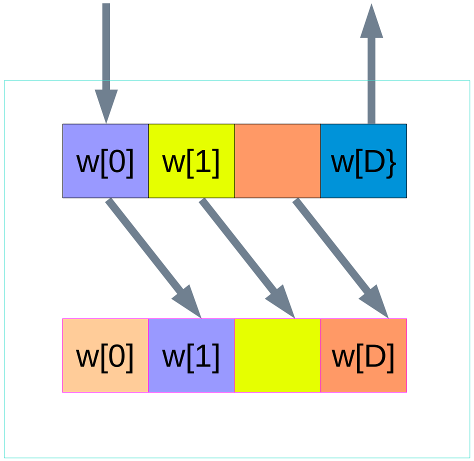
Given the optional third argument, n,
fftfilt uses the overlap-add method to filter x with b using an N-point FFT.

If x is a matrix, filter each column of the matrix.

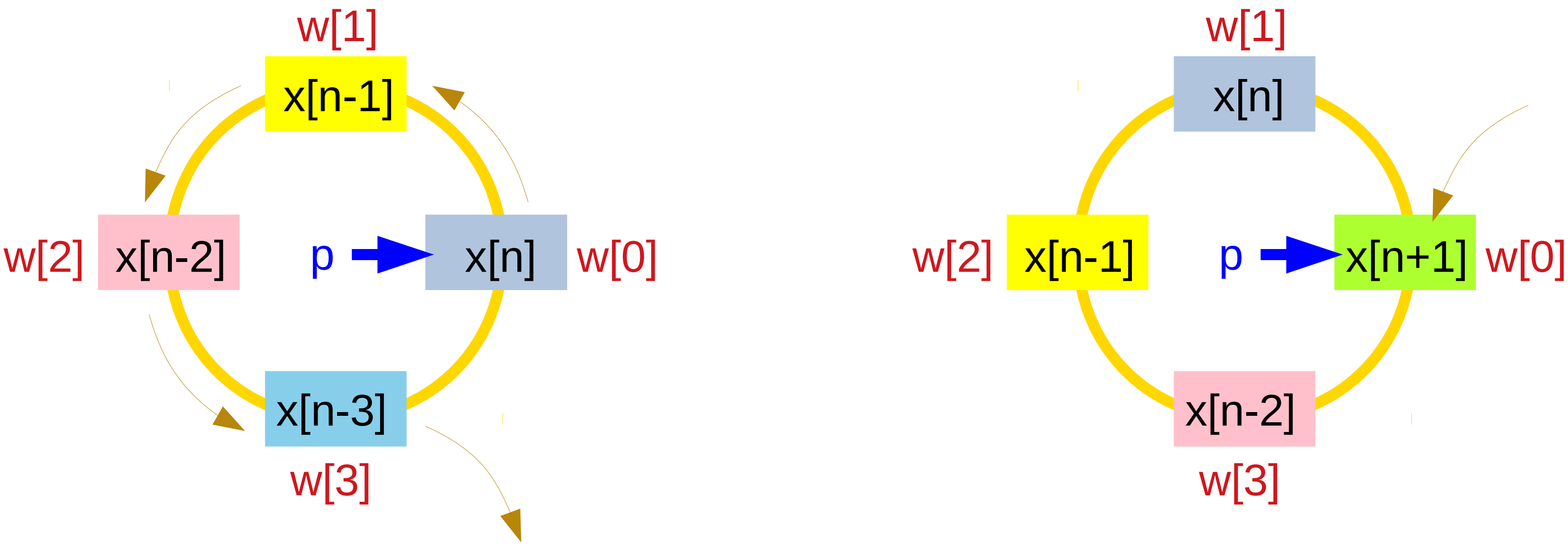
Linear delay-line buffer

At each time instant,
the <u>data</u> in the delay line are <u>shifted</u>
one memory location ahead.

<u>shifting</u> the data <u>forward</u>
while holding the <u>buffer</u> <u>addresses</u> <u>fixed</u>



Wrapped linear delay-line buffer

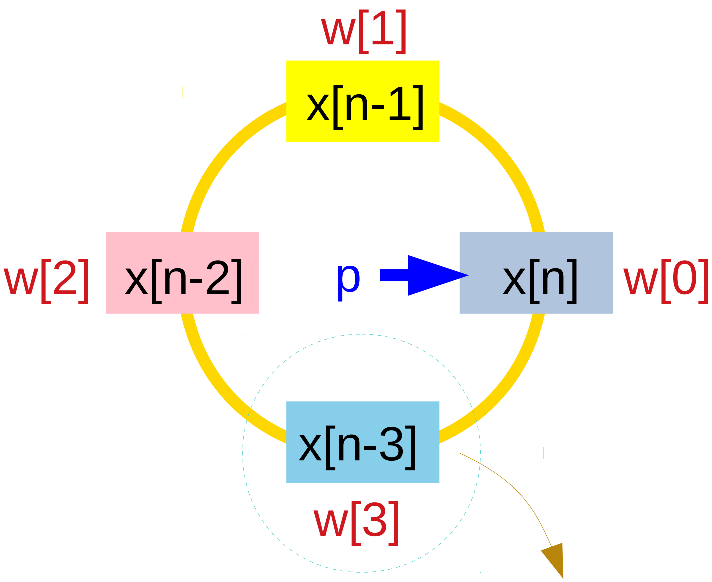


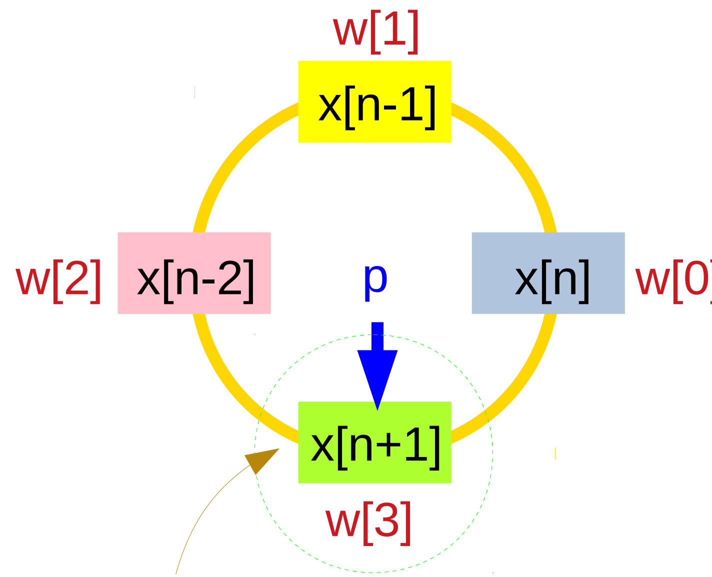
the **data** are shifted one location forward
the buffer **addresses** are fixed

the starting address p is
fixed

0	1	2	3	index
p	p+1	p+2	p+3	pointer at time **n**
p	p+1	p+2	p+3	pointer at time **n+1**
x[n]	x[n-1],	x[n-2],	x[n-3]	data at time **n**
x[n+1],	x[n],	x[n-1],	x[n-2]	data at time **n+1**

Circular delay-line buffer





the **data** are kept fixed
the **addresses** are shifted backwards

the starting address p is
shifted backward

0	1	2	3	location index
p	p+1	p+2	p+3	pointer at time **n**
p+1	p+2	p+3	p	pointer at time **n+1**
x[n]	x[n-1],	x[n-2],	x[n-3]	data at time **n**
x[n],	x[n-1],	x[n-2],	x[n+1]	data at time **n+1**

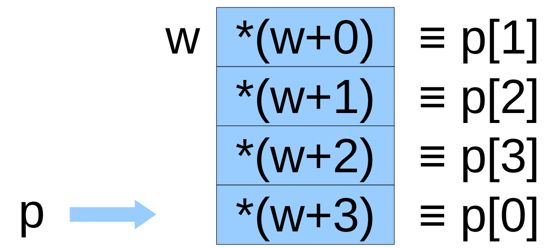
Pointer variable p

A pointer variable **p**
always points to the current input

A buffer address **w** (or w register)

At each time instant, the buffer (register)
pointed to by **p** gets loaded
with the current input sample

∗p = x   p[0] = x



Offset integer q

The pointer **p** is restricted to lie
within the pointer range of the linear buffer **w** ,

$$\mathbf{w \leq p \leq w + M}$$

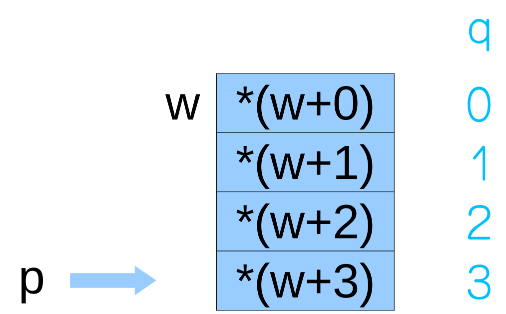
always point at somewhere in the **w** buffer, **w[q]**

$$\mathbf{p = w + q} \quad \Rightarrow$$

$$\mathbf{*p = p[0] = w[q]}$$

q is an integer that gives the offset of **p**
from the fixed beginning of the **w** buffer.

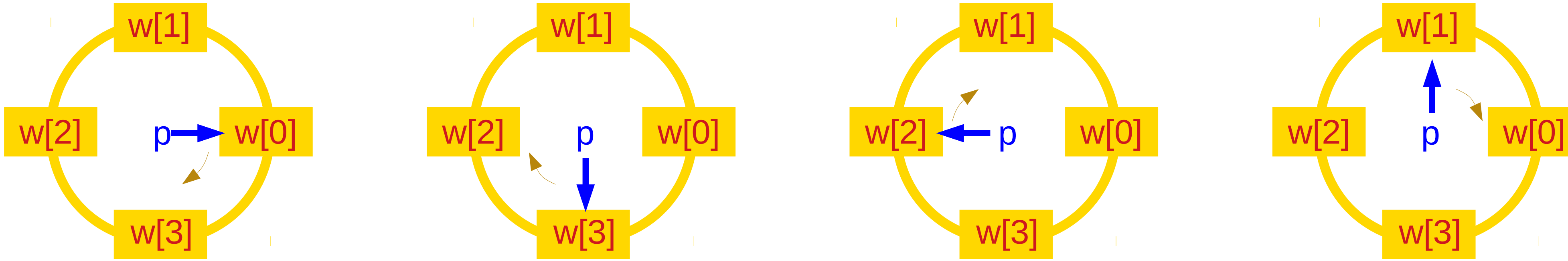
$$\mathbf{0 \leq q \leq M}$$



$p = w + 3$

$q = 3$

Address pointer p and index q



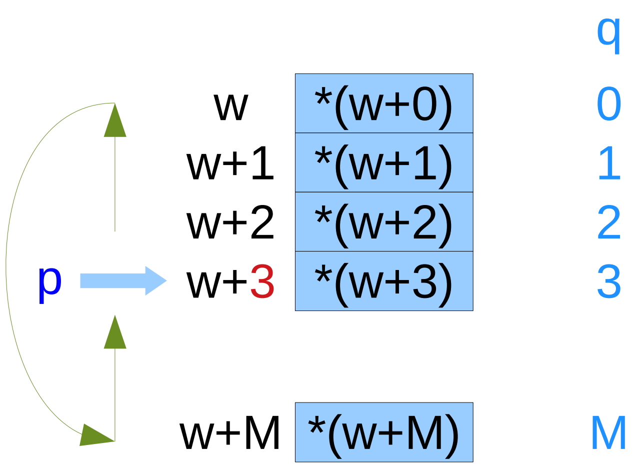
n =			
0, 4, 8, 12, …	1, 5, 9, 13,…	2, 6, 10, 14, …	3, 7, 11, 15, …
w[0] = *(w+0)	w[3] = *(w+3)	w[2] = *(w+2)	w[1] = *(w+1)
p when q = 0	p when q = 3	p when q = 2	p when q = 1

M = 3

modulo (M+1) = modulo 4

Address pointer p (moving) and w (fixed)

modulo (M+1)



$w \leq p \leq w + M$

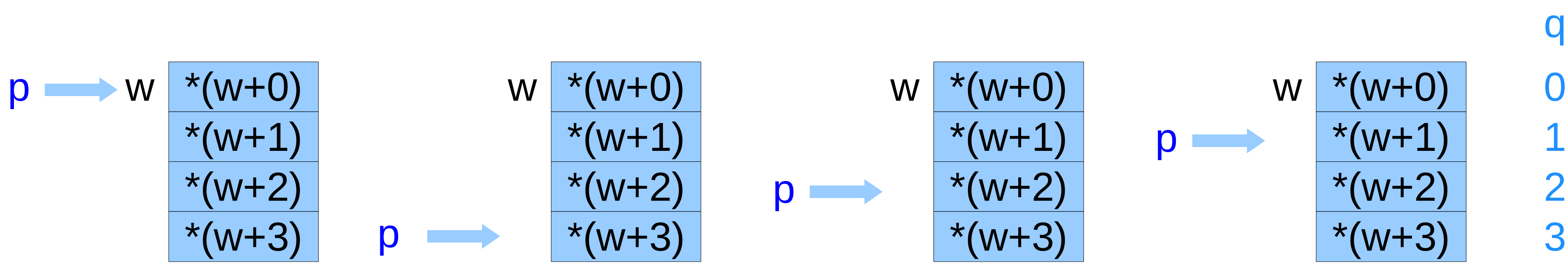
$p = w + q$ $\quad * p = * (w + q) = w[q]$

$0 \leq q \leq M$

$p[0]$	$*(p+0) = *(w+3+0) = w[3]$
$p[1]$	$*(p+1) = *(w+3+1) = w[0]$
$p[2]$	$*(p+2) = *(w+3+2) = w[1]$
$p[3]$	$*(p+3) = *(w+3+3) = w[2]$

Accessing the buffer w[k] by p[i]

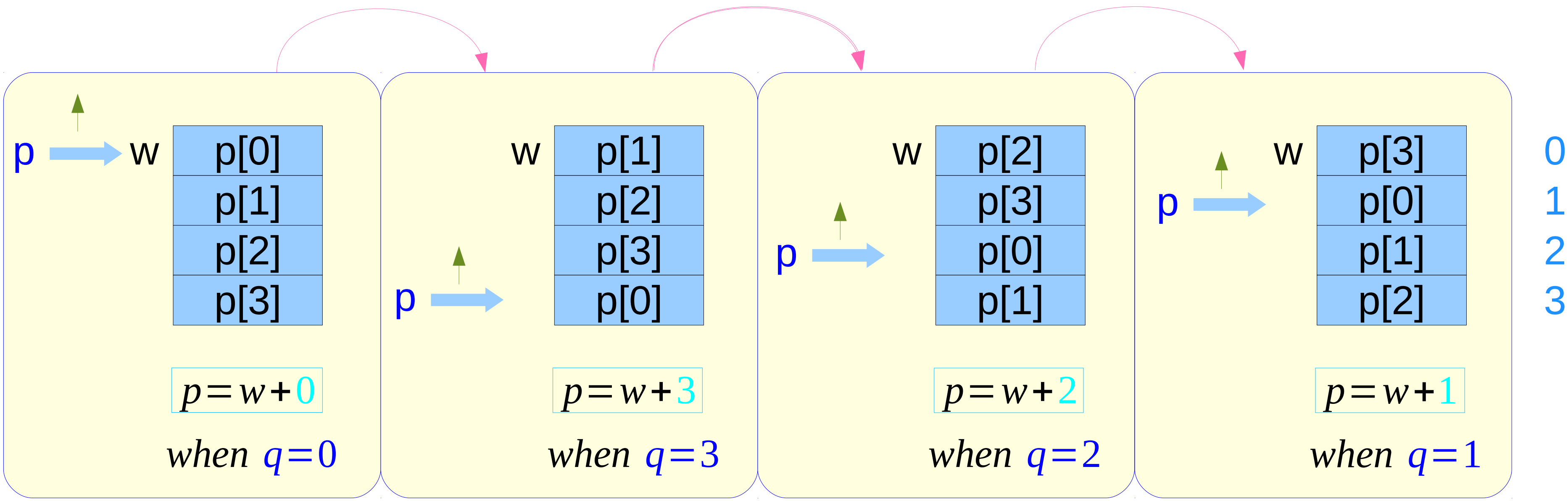
modulo (3+1)



$p[0]=w[0]$	$p[0]=w[3]$	$p[0]=w[2]$	$p[0]=w[1]$
$p[1]=w[1]$	$p[1]=w[0]$	$p[1]=w[3]$	$p[1]=w[2]$
$p[2]=w[2]$	$p[2]=w[1]$	$p[2]=w[0]$	$p[2]=w[3]$
$p[3]=w[3]$	$p[3]=w[2]$	$p[3]=w[1]$	$p[3]=w[0]$
when $q=0$	*when* $q=3$	*when* $q=2$	*when* $q=1$

Address pointer **p** moves backward

modulo (3+1)



$$p[0] \quad *(p+0) = *(w+q+0) = w[0],\ w[3],\ w[2],\ w[1]$$

$$p[1] \quad *(p+1) = *(w+q+1) = w[1],\ w[0],\ w[3],\ w[2]$$

$$p[2] \quad *(p+2) = *(w+q+2) = w[2],\ w[1],\ w[0],\ w[3]$$

$$p[3] \quad *(p+3) = *(w+q+3) = w[3],\ w[2],\ w[1],\ w[0]$$

modulo 4 $\quad q=0 \quad q=3 \quad q=2 \quad q=1$

Address pointer p

modulo (M+1)

$w \le p \le w+M$ $\qquad$ $0 \le q \le M$

$p = w + q$

$$*p = *(w+q) = w[q]$$

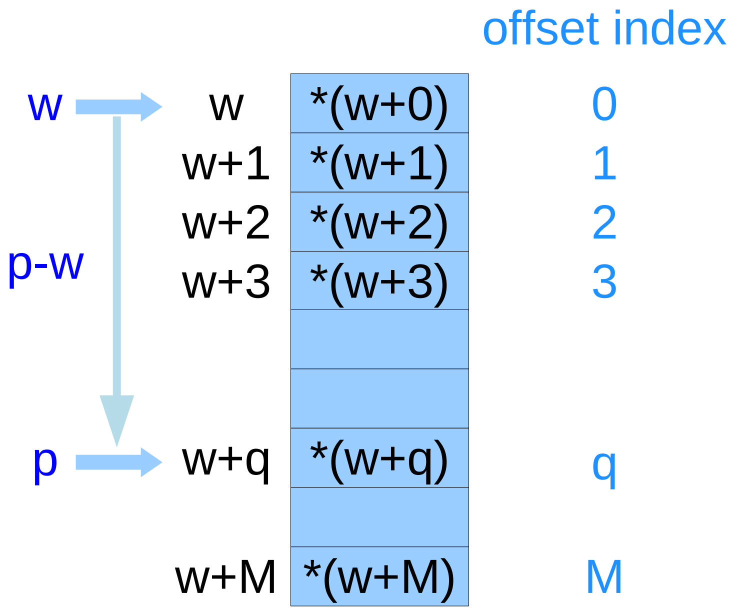
$$*(p+i) = *(w+q+i) = w[q+i]$$

$$p+i = w+(p-w+i)$$

$$*(p+i) = *(w+(p-w+i))$$

$$= *(w+(q+i))$$

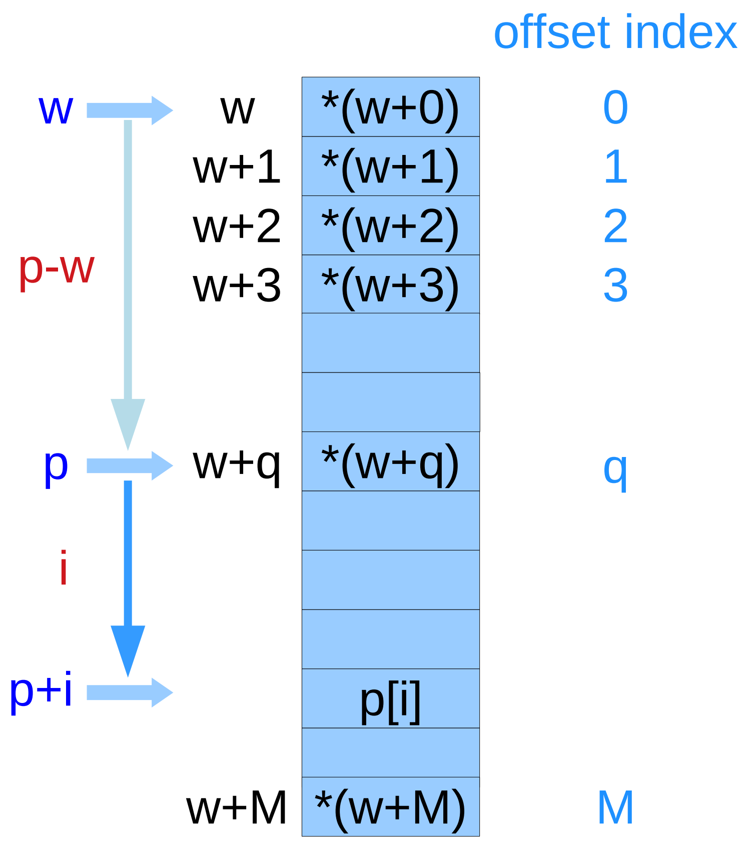
$$w[(p-w+i)\%(M+1)] = w[(q+i)\%(M+1)]$$



Accessing **w** buffer by the pointer **p**

modulo (M+1)

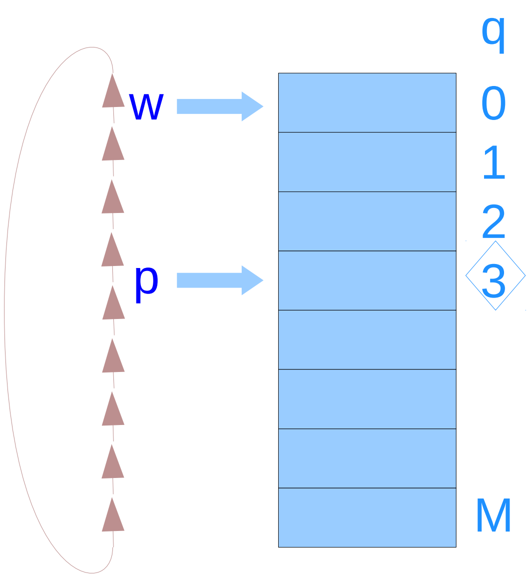
$$w[(p-w+i)\%(M+1)] = w[(q+i)\%(M+1)]$$



Address pointer **p** moves backward

modulo (M+1)

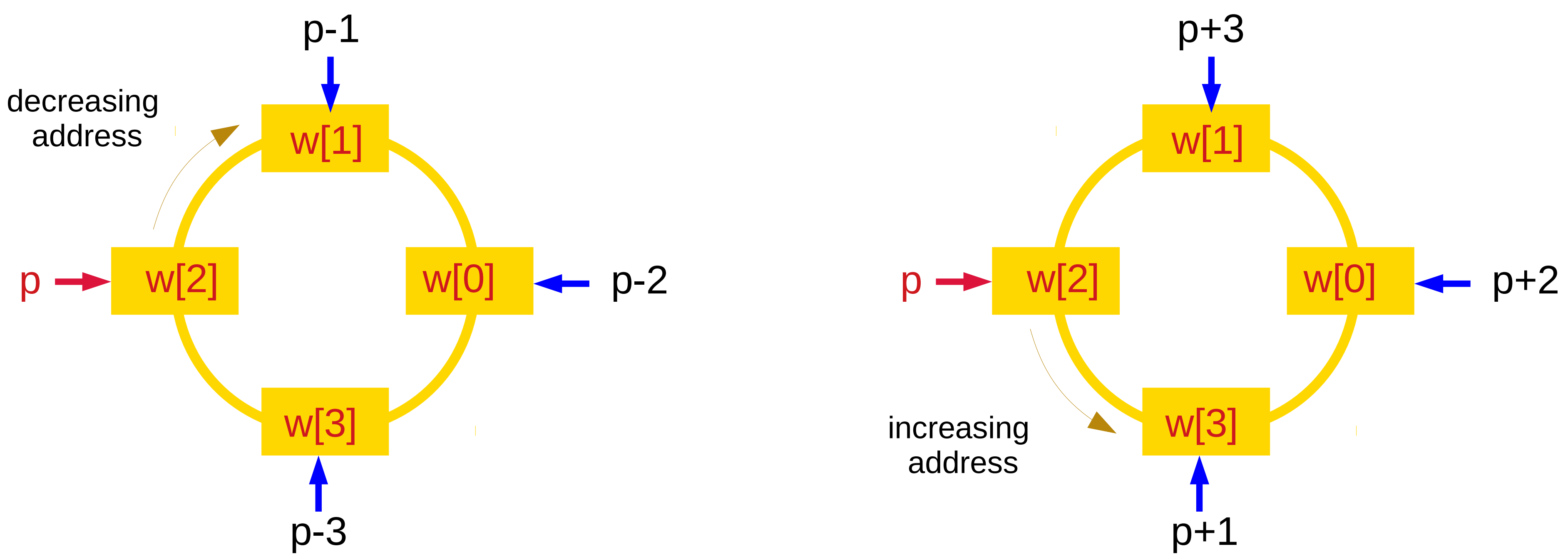
$$w[(p-w+i)\%(M+1)] = w[(q+i)\%(M+1)]$$



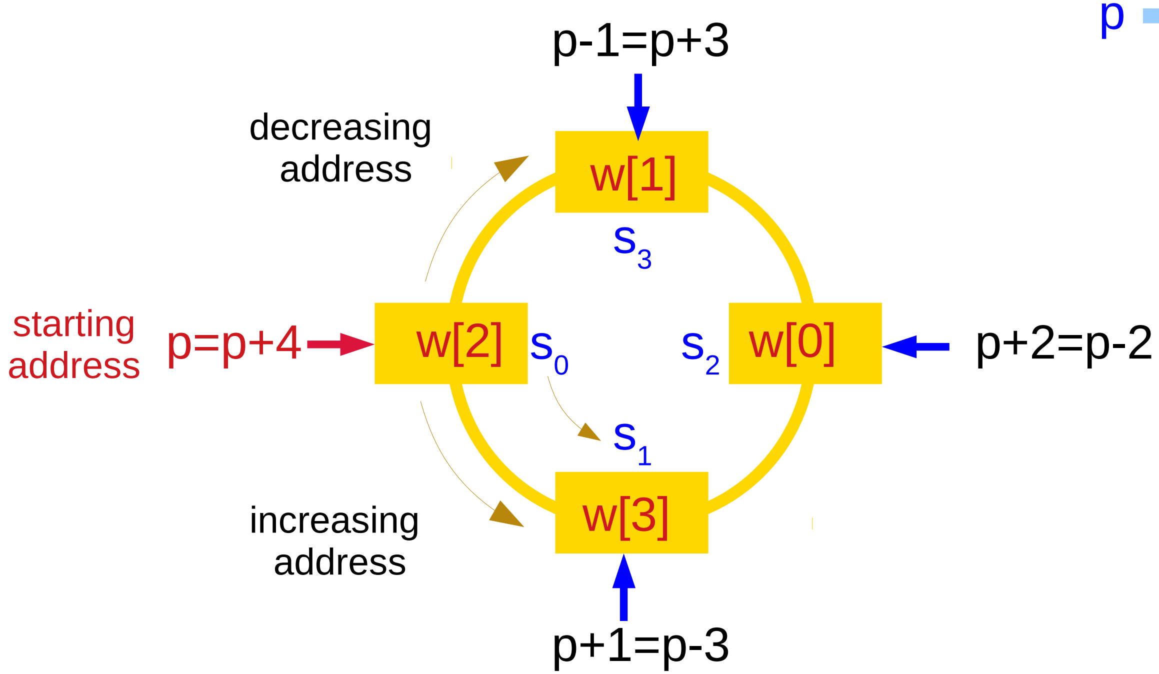
p moves backward

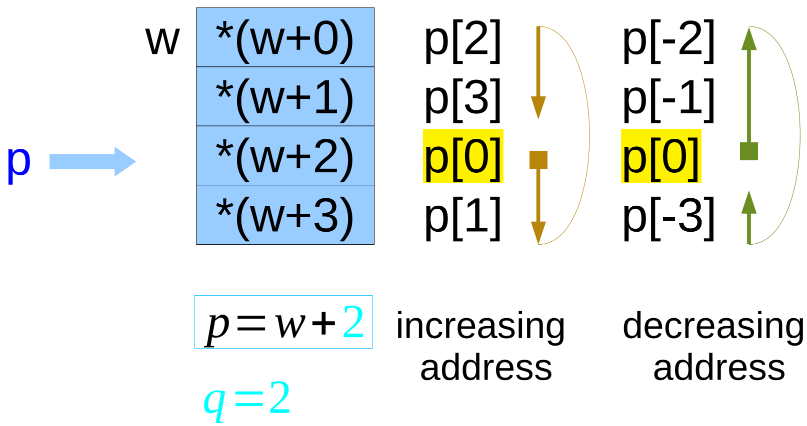
$$q = 0 \rightarrow M \rightarrow M-1 \rightarrow \cdots \rightarrow 2 \rightarrow 1 \rightarrow 0$$

Increasing and decreasing the pointer **p**



Internal States

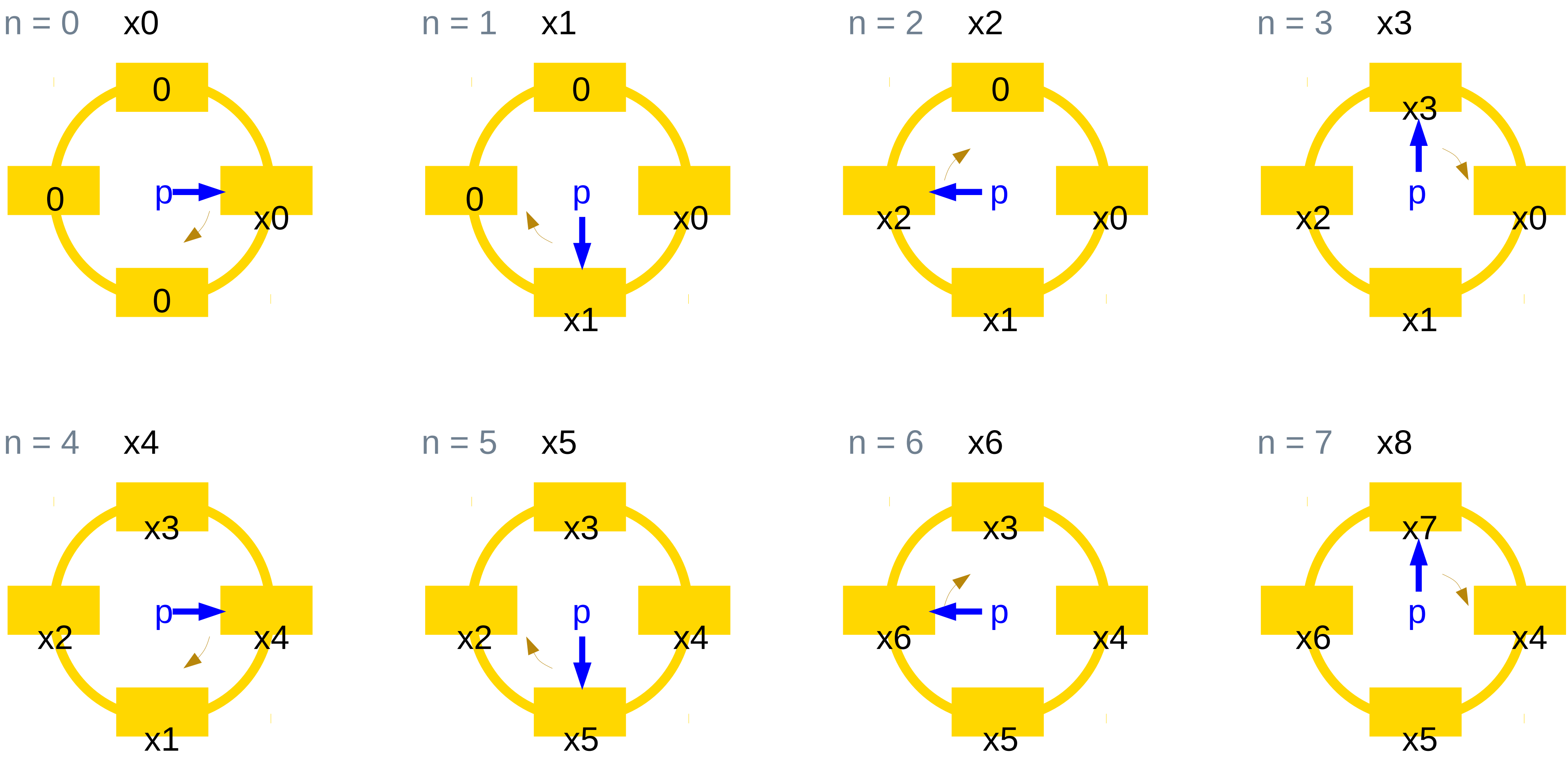




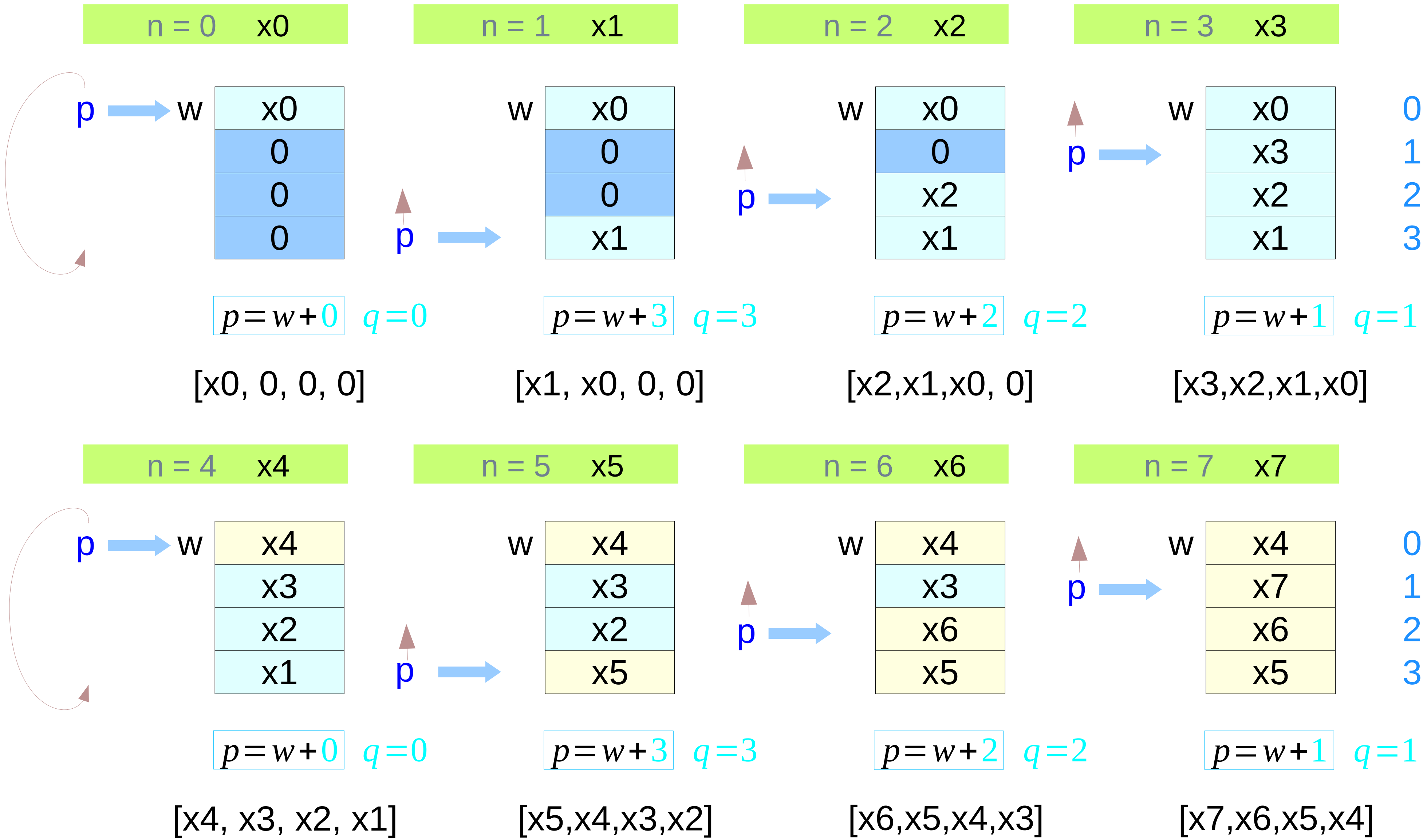
the state vector

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \quad \begin{pmatrix} p[0] \\ p[1] \\ p[2] \\ p[3] \end{pmatrix} \quad \begin{pmatrix} w[2] \\ w[3] \\ w[0] \\ w[1] \end{pmatrix}$$

Address pointer **p** and incoming inputs



Address pointer p and incoming inputs – linear array view



Internal states over the first 8 steps of n

n	q	[w0]	[w1]	[w2]	[w3]	s0	s1	s2	s3	s0	s1	s2	s3
0	0	x0	0	0	0	[w0]	[w1]	[w2]	[w3]	x0	0	0	0
1	3	x0	0	0	x1	[w3]	[w0]	[w1]	[w2]	x1	x0	0	0
2	2	x0	0	x2	x1	[w2]	[w3]	[w0]	[w1]	x2	x1	x0	0
3	1	x0	x3	x2	x1	[w1]	[w2]	[w3]	[w0]	x3	x2	x1	x0
4	0	x4	x3	x2	x1	[w0]	[w1]	[w2]	[w3]	x4	x3	x2	x1
5	3	x4	x3	x2	x5	[w3]	[w0]	[w1]	[w2]	x5	x4	x3	x2
6	2	x4	x3	x6	x5	[w2]	[w3]	[w0]	[w1]	x6	x5	x4	x3
7	1	x4	x7	x6	x5	[w1]	[w2]	[w3]	[w0]	x7	x6	x5	x4

[w0] value at the buffer w0
[w1] value at the buffer w1
[w2] value at the buffer w2
[w3] value at the buffer w3

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \quad \begin{pmatrix} p[0] \\ p[1] \\ p[2] \\ p[3] \end{pmatrix} \quad \begin{pmatrix} w[2] \\ w[3] \\ w[0] \\ w[1] \end{pmatrix}$$

References

[1] http://en.wikipedia.org/
[2] J.H. McClellan, et al., Signal Processing First, Pearson Prentice Hall, 2003
[3] M. J. Roberts, Fundamentals of Signals and Systems
[4] S. J. Orfanidis, Introduction to Signal Processing
[5] D. G. Manolakis, V. K. Ingle, Applied Digital Signal Processing