Structure of Special Digital Signal Processors

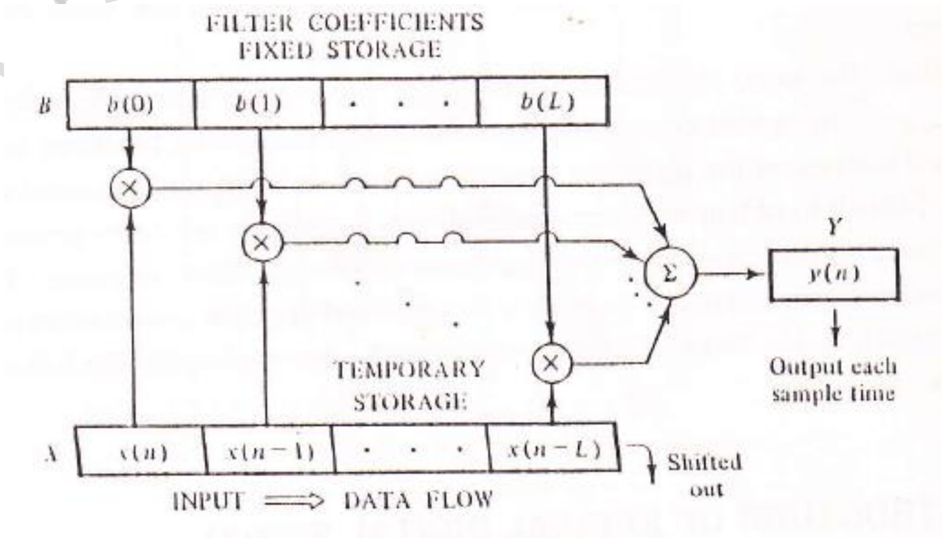
The simplicity of the digital signal processing structure is now illustrated by beginning with a single input-single filtering operation and evolving to a multiple input-multiple filter structure. For ease in presentation the structures are given for a particular type of filtering where the output is a weighted sum of a finite number of previous inputs. Such a filter is called a finite impulse response FIR filter, since it can be shown that its response to an impulse is of finite duration.

Single Input-Single Filter

If we let *y(n)* be the output and *x(n)* the input, a special class of *FIR* filters can be described by a difference equation given by:

$$y(n) = \sum_{k=0}^{L} b(k)\ x(n-k) \qquad \ldots\ldots\ldots(1)$$

In this way the output at time n is determined by a weighted sum of past and present inputs as shown in fig. 1 . for its operation, the B register is first fixed with the coefficient values *b(k), k=0,1, . . . , L* representing the filter coefficients. The operation can be explained as follows: At time *n*, *x(n)* is shifted into *X*, the input register , and *x(n – (L+1))* is pushed out of the *X* register. A term-by-term product of the contents of *X* and *B* is then completed and the sum of these products which gives the output *y(n)* is placed in the output register *Y*. After *y(n)* is calculated, another input value is shifted into the *X* register, with the last value on the right being shifted out and the multiplications and additions performed to obtain the next output value, etc. Thus, the determination of the output requires *L+1* multiplications and the addition of *L+1* terms. The *L+1* multiplications could be performed by a parallel bank of hardware multipliers or a sequential time-sharing operation on one multiplier. Hardware two-input adders could also be used in a sequential or parallel fashion. For the filter before the arrival of the next input sample. The time to perform these operations then places a constraint on the minimum time between samples and the corresponding maximum frequency of sampling.



***Single* Input-Multiple Filters**

In many cases we wish to process a single input signal to obtain different information about that signal. Such an operation may be realized as shown in fig.2.

The desired unit sample response vector $B_1, B_2, \dots, B_N$ for each of the N filtering operations on the input sequence are placed in the storage matrix as shown in fig. 2. At time n, $x(n)$ is shifted into X, the input register, the contents of B_1 are read from the storage matrix into the register B, and the switch SW is placed in position 1. This switching operation can easily be done digitally with a multiplex chip but is drawn as a switch for ease of representation and understanding. The output $y_1(1)$ from the first filter is then calculated by a term-by-term multiplication of the contents of B and X, with the results summed and placed in the first location in Y. the contents of B_2 are then read from the storage matrix into B, SW put in position 2, and $y_2(n)$, the output of filter 2 at time n, is placed in the proper location of Y. this process is continued until the N^{th} output $Y_N(n)$ is obtained. The previous operations are all repeated when another input value is shifted into X, and the outputs shifted from Y, to prepare for the new respective outputs to be calculated and placed in Y. for real-time operation, all N outputs must be calculated before a new input sample is obtained. Ignoring the reading and transfer times, the approximate minimum time T_{min} elapsed for these calculations with serial operations is:

$$T_{min} = N\,(\,(L+1)\,T_m + LT_a) \quad \dots\dots\dots(2)$$

Where T_m and T_a represent the corresponding two-input multiplication and addition times. For various degrees of parallelism the T_{min} would be less than the amount given in Eq. 2 and would depend on the specific structure. For a highly parallel structure using $N\ log_2(L+1)$ two-input adders and $N(L+1)$ two-input multipliers, the T_{min} can be reduced to :

$$T_{min} = T_m + T_a\, log_2(L+1) \quad \dots\dots\dots\dots(3)$$

Multiple Input-Single Filter Configuration

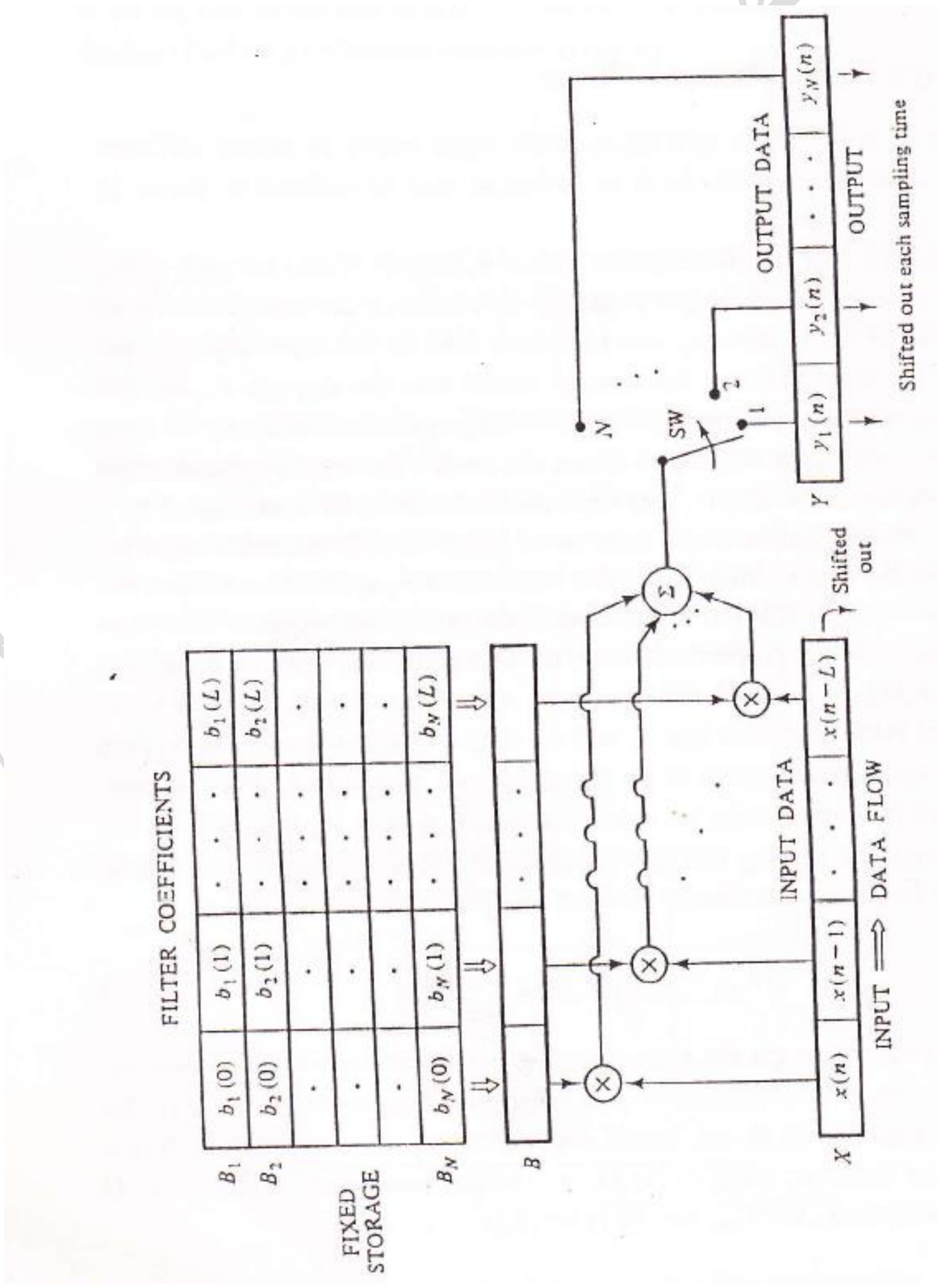
frequently we desire to process many signals with the same filter. So-called time sharing of single analog filters is impossible, thus requiring duplication of filters, whereas time sharing digitally is easily accomplished with a structure similar to that of the single input-multiple filter configuration described previously. A digital structure to handle P input signals is shown in fig. 3. the coefficients of the difference equation are stored in B. at time n, $x_1(n)$, $x_2(n)$, . . . , $x_N(n)$ are shifted into the first locations of the input registers $X_1, X_2, \dots, X_n$ and $x_1(n-(L+1))$, $x_2(n-(L+1))$, . . . , $x_N(n-(L+1))$ are pushed out the last locations, respectively. The $X_1, X_2, \dots, X_P$ form the input matrix X. the contents of X_1 are shifted into buffer register X, switch SW is placed in position 1, a term-by-term multiplication of X and B is obtained, followed by an addition to given $y_1(n)$ of the output register Y. Next, the contents of X_2 are shifted into register B, SW is placed in position 2, and term-by-term multiplication of X and B is obtained, added, and placed in $y_2(n)$. This process is continued until $y_P(n)$ has been calculated. At that time the filter is ready to accept another vector of input values. To operate in real time the T_{min} is seen to be identical to that of eq.2 and eq. 3 with N replaced by P, i.e:

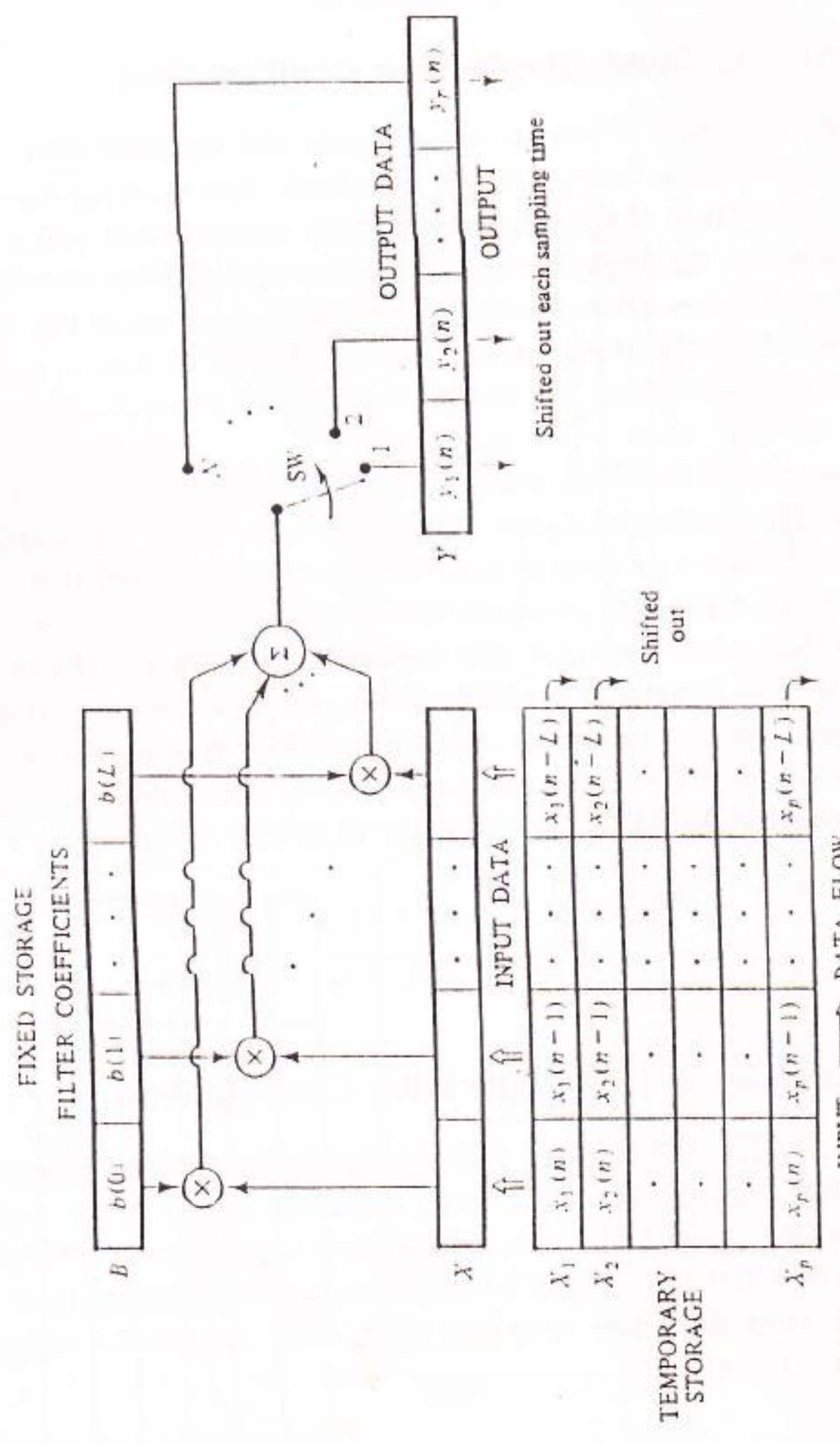
$$T_{min} = P\,((L+1)T_m + LT_a) \quad \text{......... (4)} \qquad \textit{serial}$$

$$T_{min} = T_m + T_a \log_2 (L+1) \quad \text{..........(5)} \qquad \textit{parallel}$$

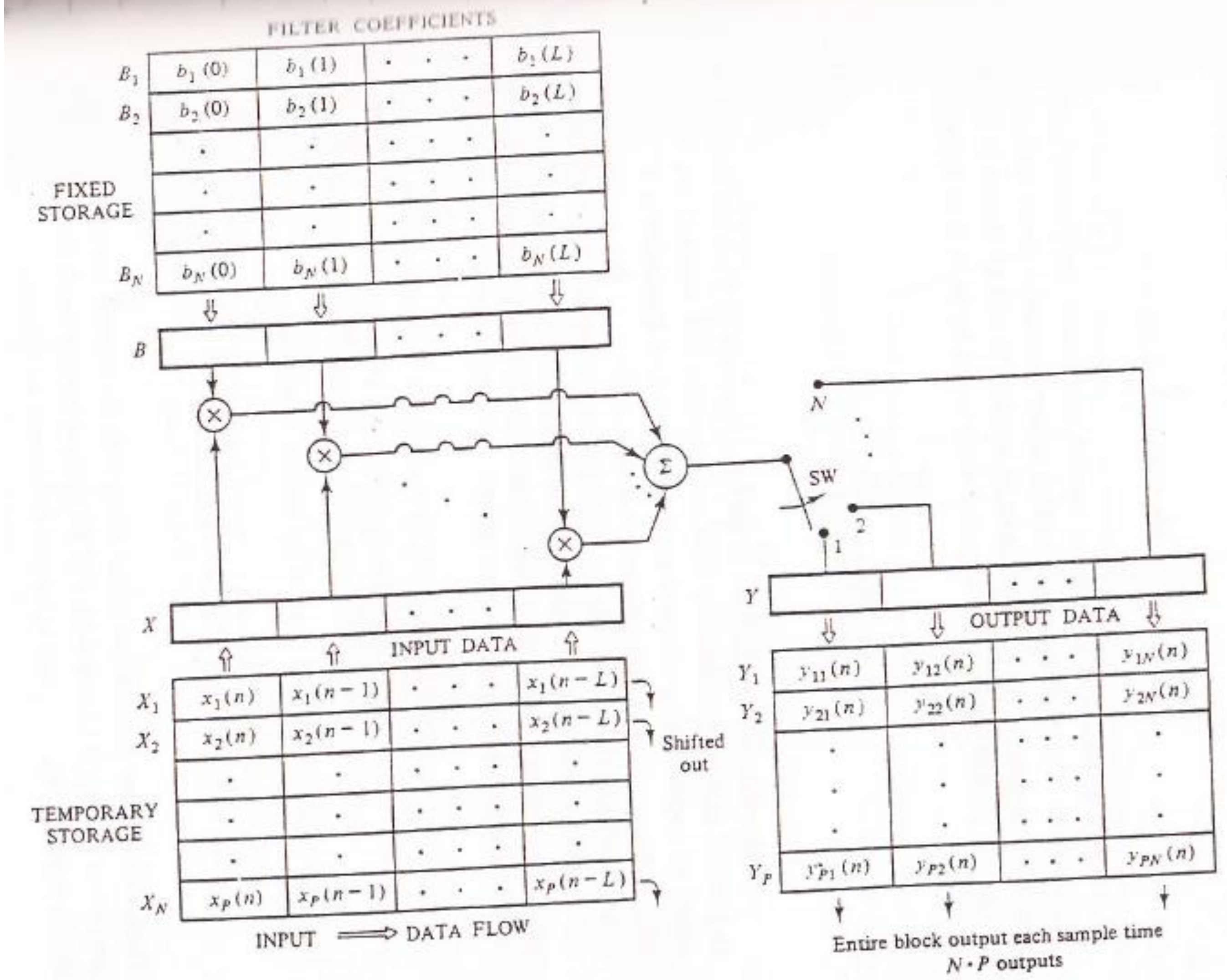
Multiple Input-Multiple Filter Configuration

By combining fig. 2 and fig. 3 we are able to obtain a P input N filter configuration shown in fig. 4 with operations similar to the multiple input-single filter for each filter, a matrix of $N.P$ outputs can be calculated. The minimum time between samples necessary to calculate all output values can be easily calculated depending upon how much serial and parallel multiplication and addition is performed.



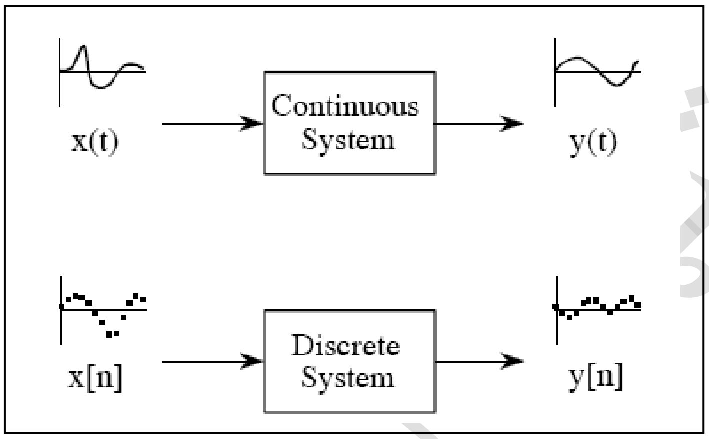


FIXED STORAGE
FILTER COEFFICIENTS
B
$b(0)$
$b(1)$
$b(L)$
Σ
SW
1
2
N
Y
$y_1(n)$
$y_2(n)$
$y_r(n)$
OUTPUT DATA
OUTPUT
Shifted out each sampling time
X
INPUT DATA
X_1
X_2
X_p
$x_1(n)$
$x_1(n-1)$
$x_1(n-L)$
$x_2(n)$
$x_2(n-1)$
$x_2(n-L)$
$x_p(n)$
$x_p(n-1)$
$x_p(n-L)$
Shifted out
TEMPORARY STORAGE
INPUT ⟹ DATA FLOW


FILTER COEFFICIENTS
B_1 $b_1(0)$ $b_1(1)$ . . . $b_1(L)$
B_2 $b_2(0)$ $b_2(1)$ . . . $b_2(L)$
FIXED STORAGE
B_N $b_N(0)$ $b_N(1)$ . . . $b_N(L)$
B
X
INPUT DATA
X_1 $x_1(n)$ $x_1(n-1)$ . . . $x_1(n-L)$
X_2 $x_2(n)$ $x_2(n-1)$ . . . $x_2(n-L)$
Shifted out
TEMPORARY STORAGE
X_N $x_P(n)$ $x_P(n-1)$ . . . $x_P(n-L)$
INPUT ⟹ DATA FLOW
Σ
SW
1
2
N
Y
OUTPUT DATA
Y_1 $y_{11}(n)$ $y_{12}(n)$ . . . $y_{1N}(n)$
Y_2 $y_{21}(n)$ $y_{22}(n)$ . . . $y_{2N}(n)$
Y_P $y_{P1}(n)$ $y_{P2}(n)$ . . . $y_{PN}(n)$
Entire block output each sample time
$N \cdot P$ outputs

Signals and Systems

A **signal** is a description of how one parameter varies with another parameter. For instance, voltage changing over time in an electronic circuit, or brightness varying with distance in an image. A **system** is any process that produces an *output signal* in response to an *input signal*. This is illustrated by the block diagram in the following Figure. Continuous systems input and output continuous signals, such as in analog electronics. Discrete systems input and output discrete signals, such as computer programs that manipulate the values stored in arrays.

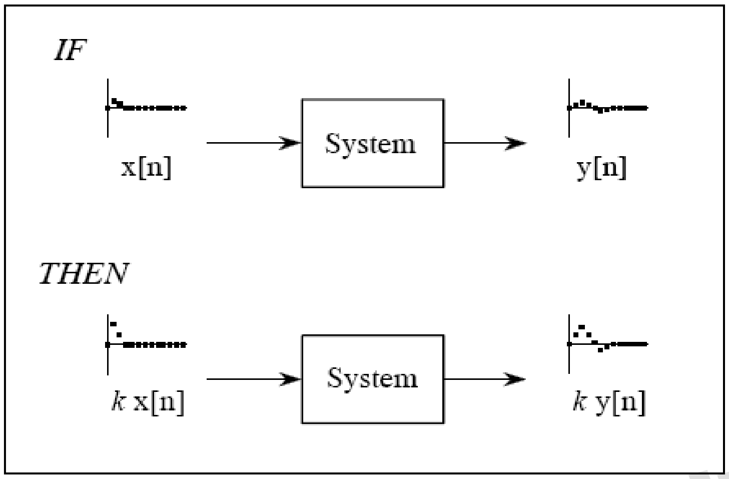


Terminology for signals and systems. A system is any process that generates an output signal in response to an input signal. Continuous signals are usually represented with parentheses, while discrete signals use brackets. All signals use lower case letters, reserving the upper case for the frequency domain. Unless there is a better name available, the input signal is called: *x*(*t*) or *x*[*n*], while the output is called: *y*(*t*) or *y*[*n*].

Requirements for Linearity

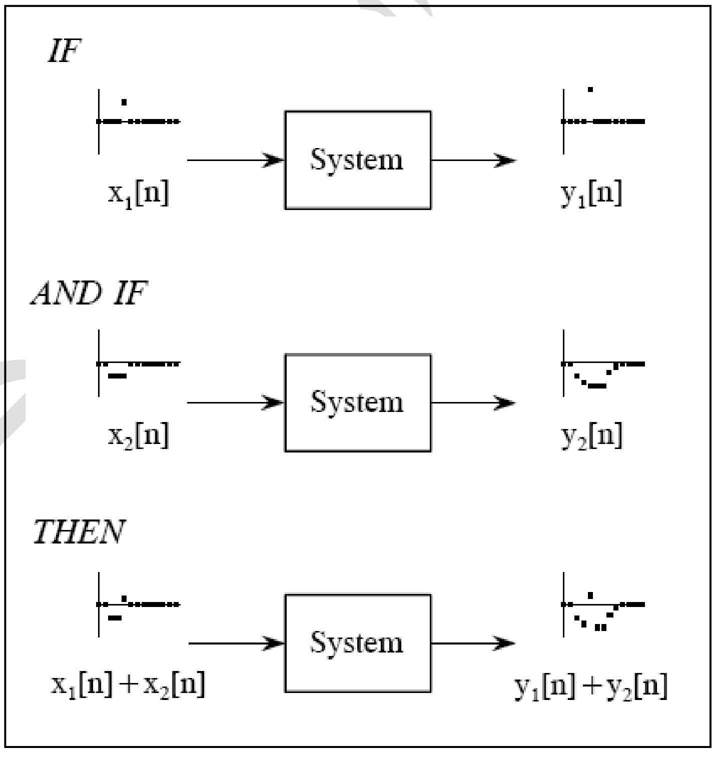
A system is called *linear* if it has two mathematical properties: **homogeneity** and **additivity**. If you can show that a system has both properties, then you have proven that the system is linear. Likewise, if you can show that a system doesn't have one or both properties, you have proven that it isn't linear. A third property, **shift invariance**, is not a strict requirement for linearity, but it is a mandatory property for most DSP techniques. When you see the term *linear system* used in DSP, you should assume it includes *shift invariance* unless you have reason to believe otherwise. These three properties form the mathematics of how linear system theory is defined and used.

For now, let's go through these formal mathematical properties. As illustrated in the following Figure, homogeneity means that a change in the input signal's amplitude results in a corresponding change in the output signal's amplitude. In mathematical terms, if an input signal of *x*[*n*] results in an output signal of *y*[*n*] , an input of *k x*[*n*] results in an output of *k y*[*n*], for any input signal and constant, *k*.



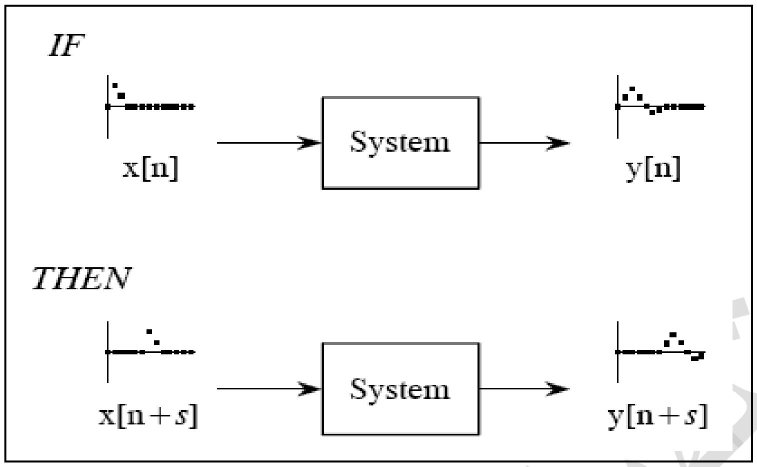
Definition of homogeneity. A system is said to be *homogeneous* if an amplitude change in the input results in an identical amplitude change in the output. That is, if $x[n]$ results in $y[n]$, then $kx[n]$ results in $ky[n]$, for any signal, $x[n]$, and any constant, k.

The property of additivity is illustrated in the following Fig. Consider a system where an input of $x_1[n]$ produces an output of $y_1[n]$. Further suppose that a different input, $x_2[n]$, produces another output, $y_2[n]$. The system is said to be *additive*, if an input of $x_1[n]+x_2[n]$ results in an output of $y_1[n] + y_2[n]$, for all possible input signals. In words, signals added at the input produce signals that are added at the output.



Definition of additivity. A system is said to be *additive* if added signals pass through it without interacting. Formally, if $x_1[n]$ results in $y_1[n]$, and if $x_2[n]$ results in $y_2[n]$, then $x_1[n]+x_2[n]$ results in $y_2[n]+y_2[n]$.

Definition of shift invariance. A system is said to be *shift invariant* if a shift in the input signal causes an identical shift in the output signal. In mathematical terms, if $x[n]$ produces $y[n]$, then $x[n+s]$ produces $y[n+s]$, for any signal, $x[n]$, and any constant, s.



Linear System

a discrete time system specified by $T[.]$ is linear if the response to a weighted sum of inputs $x_1(n)$ and $x_2(n)$ is a weighted sum (with the same weights) of the responses of the two inputs separately for all weights and all acceptable signals $x_1(n)$ and $x_2(n)$. Therefore, a system specified by $T[.]$ is linear if for all a_1 , a_2 , $x_1(n)$ and $x_2(n)$ we have:

$$T\,[a_1x_1(n) + a_2x_2(n)] = a_1\,T[x_1(n)] + a_2\,T[x_2(n)]$$

Example: $y(n) = n\,x(n) = T\,[x(n)]$

Let $x(n) = a_1x_1(n) + a_2\,x_2(n)$

Then the output : $y(n) = n\,[\,a_1x_1(n) + a_2\,x_2(n)]$

After multiplying and reorganizing y(n) can be written as:

$y(n) = a_1\,n\,x_1(n) + a_2\,n\,x_2(n)$

$= a_1\,T[x_1(n)] + a_2\,T[x_2(n)]$

$\therefore$ *Linear*

Example: $y(n) = x(n+1) + x(n-1)$

If $x(n) = a_1\,x_1(n) + a_2\,x_2(n)$

$y(n) = a_1[x_1(n+1) + x_1(n-1)] + a_2\,[\,x_2(n+1) + x_2(n-1)]$

$= a_1\,y_1(n) + a_2\,y_2(n)$

$\therefore$ *the system is linear*

Example: $y(n) = a\,x^2(n)$

if $x(n) = a_1\,x_1(n) + a_2\,x_2(n)$

$y(n)= a\,[\,a_1\,x_1(n) + a_2\,x_2(n)\,]^2$

$y(n) = a\,[\,a_1\,x_1^2\,(n) +2a_1x_1(n)\,a_2x_2(n) + a_2x_2^2(n)]$

$y(n\,) \neq a_1x_1^2(n) + a_2x_2^2(n)$

$\therefore$ *the system is non linear*