VLSI Implementation of Digital Filters Using Distributed Arithmetic

by

Abhijit Karmakar
CEERI, Pilani

23RD Dec. 2009, TU Patiala

Contents

Filters

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range.

Analog and digital filters

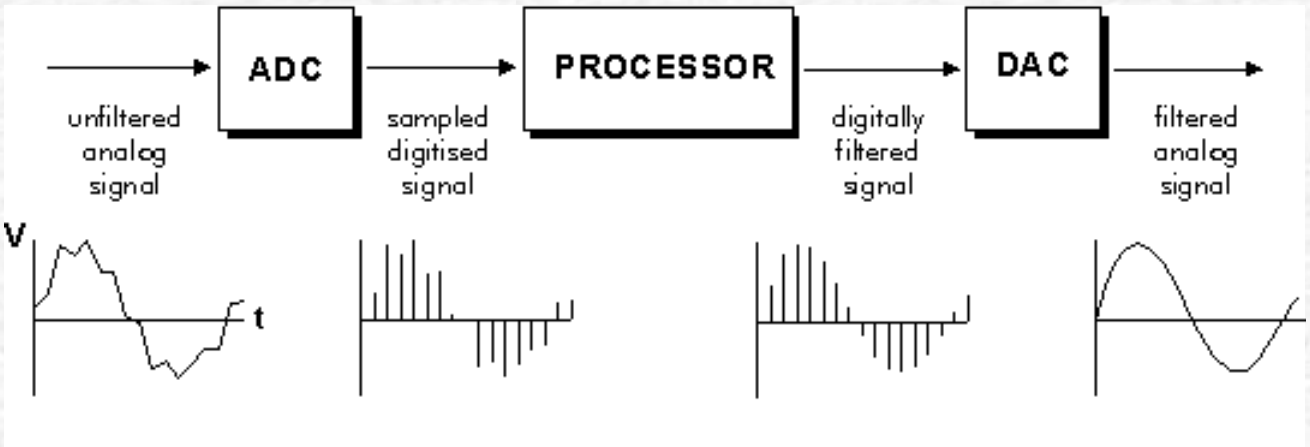
- Filters are two main kinds, *analog* and *digital*. They are quite different in their physical makeup and in how they work.
- An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op-amps to produce the required filtering effect.
- Such filter circuits are widely used in such applications as noise reduction, video signal enhancement, graphic equalisers in hi-fi systems, and many other areas.
- There are well-established standard techniques for designing an analog filter circuit for a given requirement. At all stages, the signal being filtered is an electrical voltage or current which is the direct analogue of the physical quantity (e.g. a sound or video signal or transducer output) involved.

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialised DSP (Digital Signal Processor) chip.

Analog and digital filters

■ The analog input signal must first be sampled and digitised using an ADC (analog to digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them.

■ These calculations typically involve multiplying the input values by constants and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital to analog converter) to convert the signal back to analog form.
Note that in a digital filter, the signal is represented by a sequence of numbers, rather than a voltage or current.
The following diagram shows the basic setup of such a system.



Different Types of Signals

A signal is any time-varying or spatial-varying quantity, containing information. In the physical world, any quantity measurable through time or over space can be taken as a signal.

Examples

- Motion: The motion of a particle through some space can be considered to be a signal, or can be represented by a signal.
- Sound : Sound is a vibration of a medium (such as air) and associates a pressure value to every value of time and three space coordinates.
- Picture: A picture assigns a color value to each of a set of points. Since the points lie on a plane, the domain is two-dimensional. If the picture is a physical object, such as a painting, it's a continuous signal. If the picture a digital, it's a discrete signal.
- Video: A video signal is a sequence of images. A point in a video is identified by its position (two-dimensional) and by the time at which it occurs, so a video signal has a three-dimensional domain.

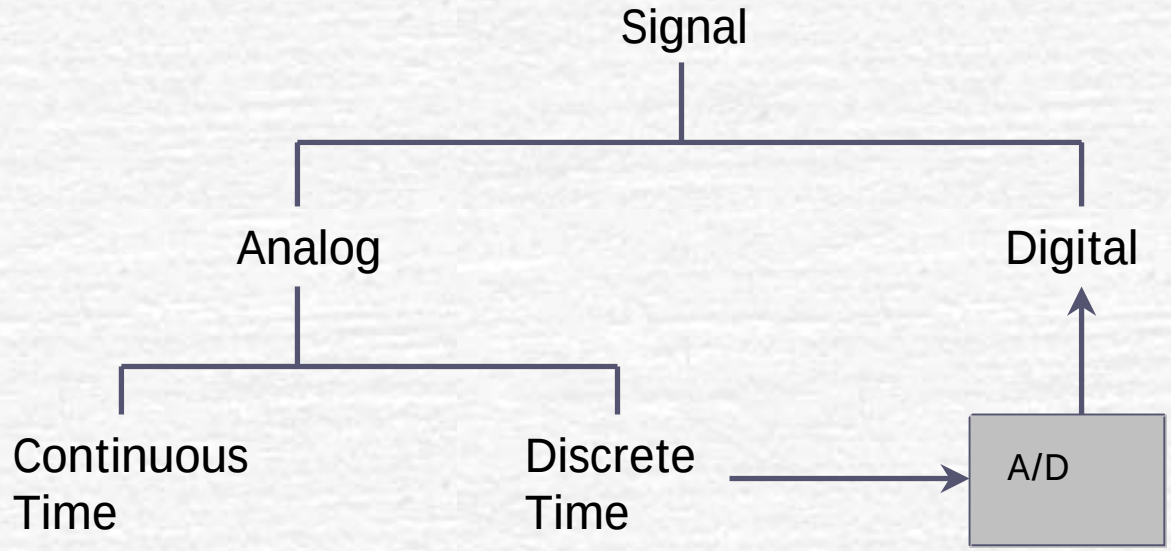
Characterization of Signals

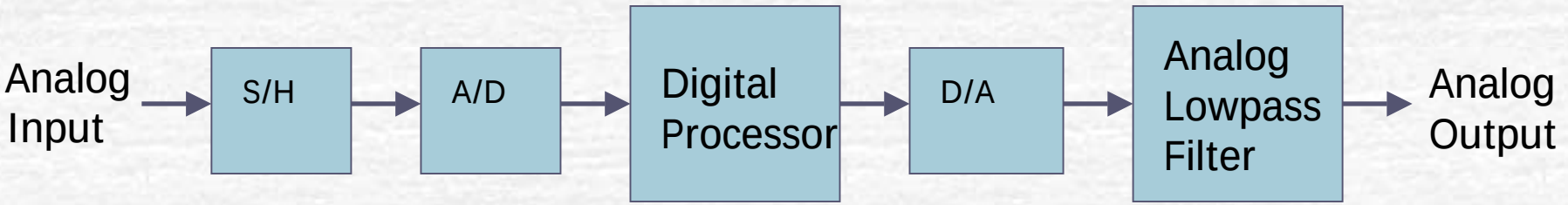
Signals can be categorized in various ways.

→ Based on the spaces (discrete / continuous), that the functions are defined over. Discrete-time signals are defined over discrete time domain, where Continuous-time signals are defined over continuous signals.

→ Based on whether the signal is discrete-valued and continuous-valued. Digital signals are discrete-valued, but are often derived from an underlying continuous-valued physical process.

Characterization of Signals





Scheme for the digital processing of an analog signal

Discrete Time Systems

Discrete-Time Systems

An important class of DT system is Linear Time Invariant (LTI) system.

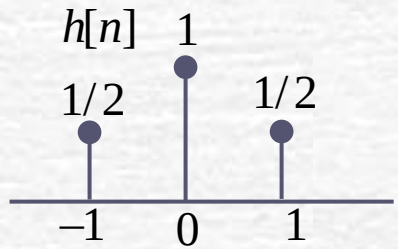
LTI DT obeys both Linearity and Time-Invariance.

LTI DT systems are uniquely characterized by the impulse response of the system.

$h[n]$ is the impulse response of the DT system $h[n]=0$ for $n<0 \Rightarrow$ causal

Ex. $y[n]=x[n]+(1/2)(x[n-1]+x[n+1])$

Impulse Response $h[n]=\delta[n]+(1/2)(\delta[n-1]+\delta[n+1])$



Discrete Time Systems

LTI System, Convolution

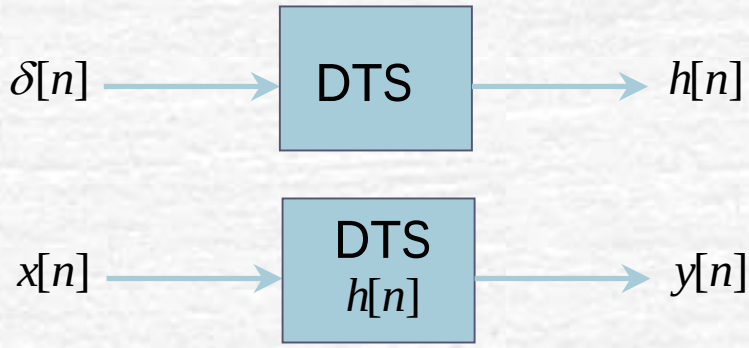
$$\delta[n] \rightarrow h[n]$$

$$\delta[n-k] \rightarrow h[n-k] \quad \text{Time-Invariance}$$

$$x_k \delta[n-k] \rightarrow x_k h[n-k] \quad \text{Homogenity}$$

$$\sum_{k=-\infty}^{\infty} x_k \delta[n-k] \rightarrow \sum_{k=-\infty}^{\infty} x_k h[n-k] \quad \text{Superposition}$$

$$\underbrace{\sum_{k=-\infty}^{\infty} x_k \delta[n-k]}_{x[n]} \rightarrow \underbrace{\sum_{k=-\infty}^{\infty} x_k h[n-k]}_{y[n]}$$



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

Output of the digital filter can be obtained by convolving its impulse response with the input.

Digital Filters are LTI Discrete Time Systems

FIR and IIR Filters

FIR & IIR

$$\left.\begin{array}{ll} h[n] \neq 0 & N_1 \leq n \leq N_2 \\ = 0 & \text{everywhere else} \end{array}\right\} \quad FIR$$

$$y[n] = \sum_{k=N1}^{N2} x[n-k]h[k] \qquad \text{If } N_1 \text{ or } N_2 \text{ or both } \rightarrow \infty \quad \text{IIR}$$

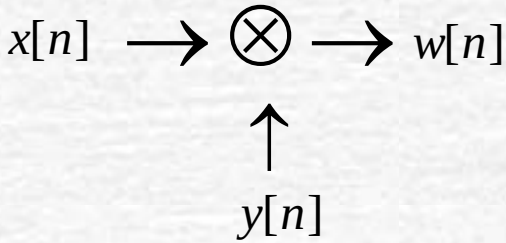
Ex: Moving Average (MA) System $y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$

$$\left\{ \underset{\substack{\uparrow \\ n=0}}{\frac{1}{M}}, \frac{1}{M}, \frac{1}{M}, \ldots, \underset{\substack{\uparrow \\ n=M-1}}{\frac{1}{M}} \right\} \quad \leftarrow \text{Impulse Response}$$

Operations on Sequences

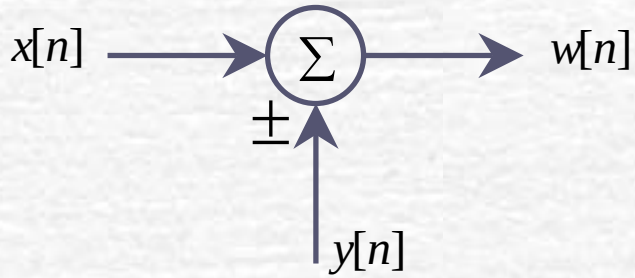
1. Multiplication

$$w[n] = x[n]y[n]$$



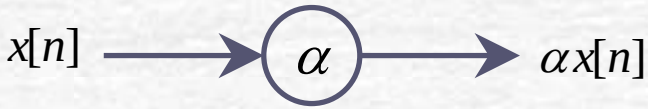
2. Addition/ Subtraction

$$w[n] = x[n] \pm y[n]$$

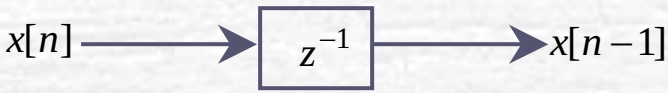


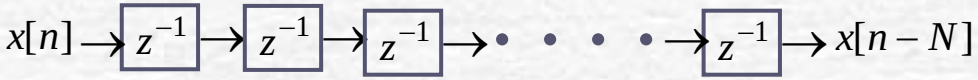
3. Scalar Multiplication

$$w[n] = \alpha x[n]$$



4. Delay

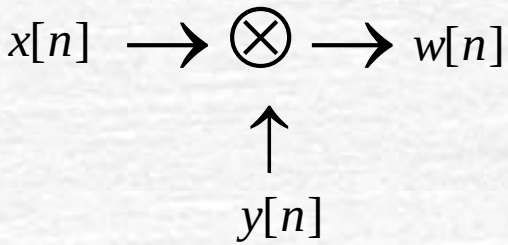




Operations on Sequences

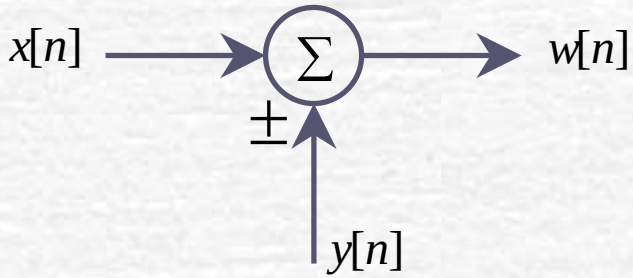
1. Multiplication

$$w[n] = x[n]y[n]$$



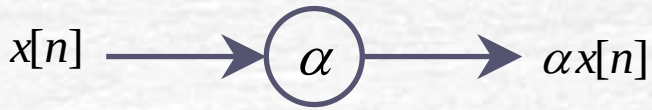
2. Addition/ Subtraction

$$w[n] = x[n] \pm y[n]$$

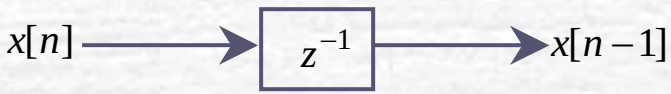


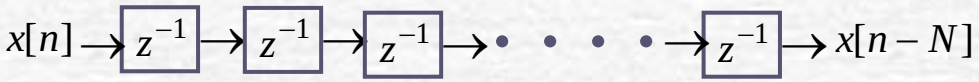
3. Scalar Multiplication

$$w[n] = \alpha x[n]$$



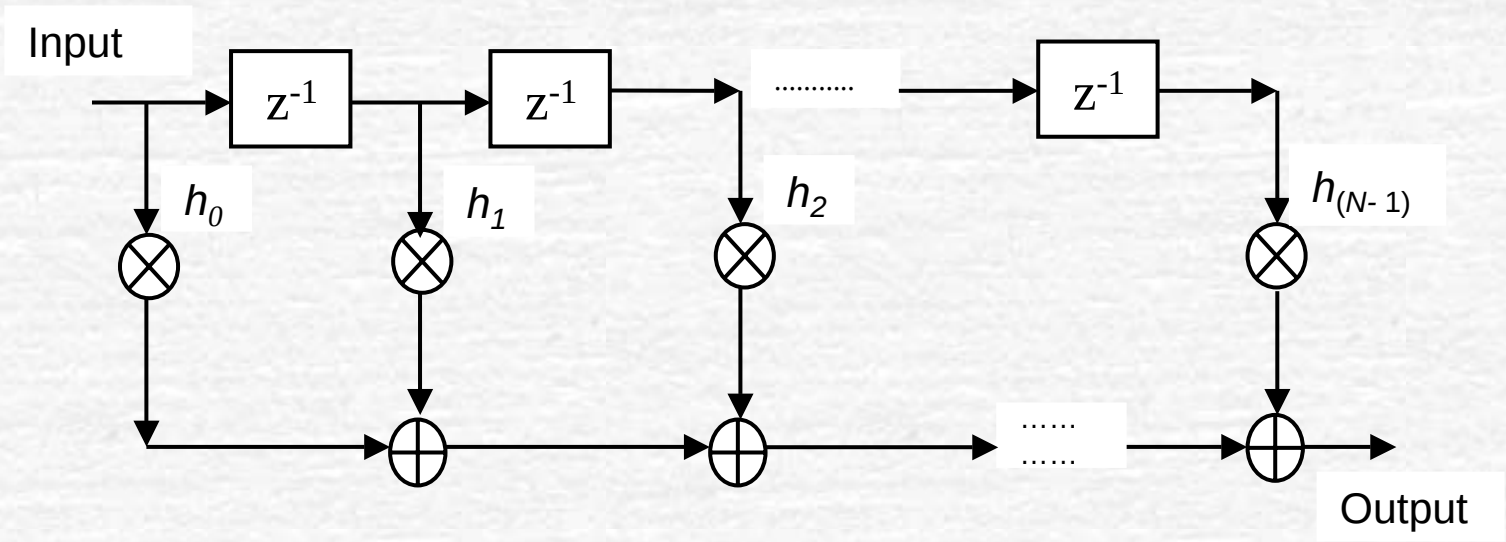
4.Delay





Digital Filter Structure

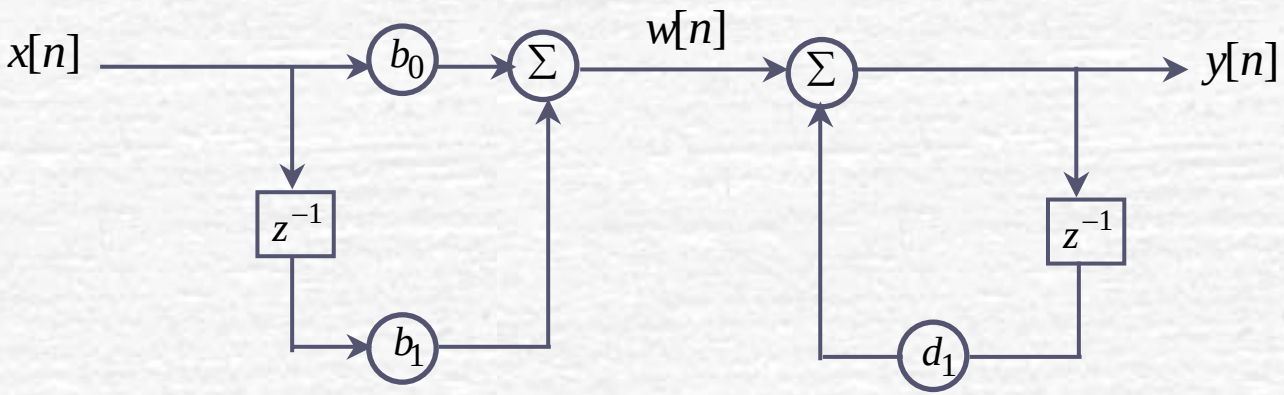
Finite Impulse Response Filter



$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k}$$

Digital Filter Structure

Infinite Impulse Response Filter



$$y[n]=b_0x[n]+b_1x[n-1]+d_1y[n-1]$$

Overview of Distributed Arithmetic

- Operations that occur in signal processing are not lumped
- They are distributed in an often unrecognized fashion
- Most-often encountered form of computation in DSP is dot-product (inner product generation)
- Extreme computational efficiency.
- Bit-serial computation.

Overview of Distributed Arithmetic

DA inner-product generation

$$y = \sum_{k=1}^{K} A_k x_k$$

A_k are fixed coefficients

x_k are fixed input data words

If each x_k is a $2^{'}$s complement binary number scaled such that $|x_k| < 1$,

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n}$$

b_{kn} are the bits, 0 or 1,

b_{k0} is the sign bit, and

$b_{k,N-1}$ is the least significant bit (LSB)

Overview of Distributed Arithmetic

Combining the above equations

$$y = \sum_{k=1}^{K} A_k \left[-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \right]$$

$$y = \sum_{n=1}^{N-1} \left[\sum_{k=1}^{K} A_k b_{kn} \right] 2^{-n} + \sum_{k=1}^{K} A_k \left(-b_{k0} \right)$$

Crucial step $\rightarrow$ Defines a distributed arithmetic operation

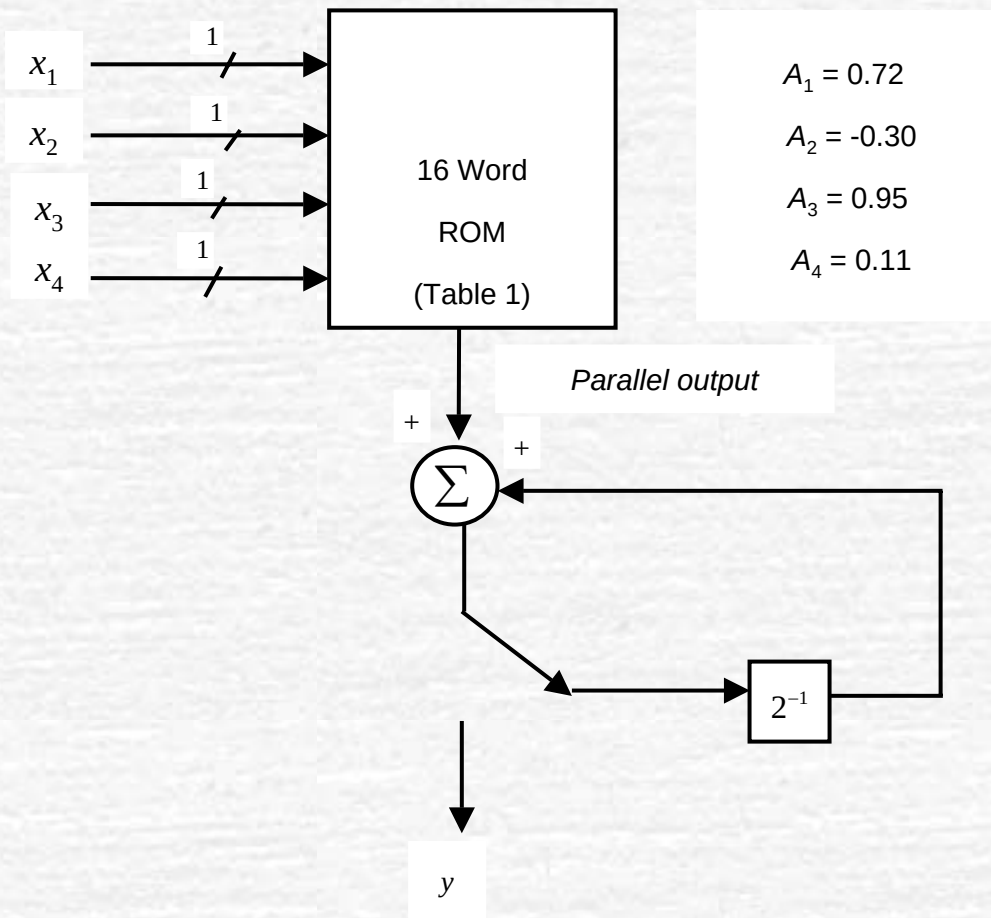
Because each b_{kn} may take on values of 0 and 1 only,

$\sum_{k=1}^{K} A_k b_{kn}$ may have only 2^K possible values.

Rather than computing these values on-line, we may precompute the values and store them in a ROM. The input data can be used to directly address the memory and the result, i.e., the $\sum_{k=1}^{K} A_k b_{kn}$ can be dropped into an accumulator

$\sum_{k=1}^{K} A_k b_{kn}$ may have only 2^K possible values.

Overview of Distributed Arithmetic



Inner-product computation using DA

Overview of Distributed Arithmetic

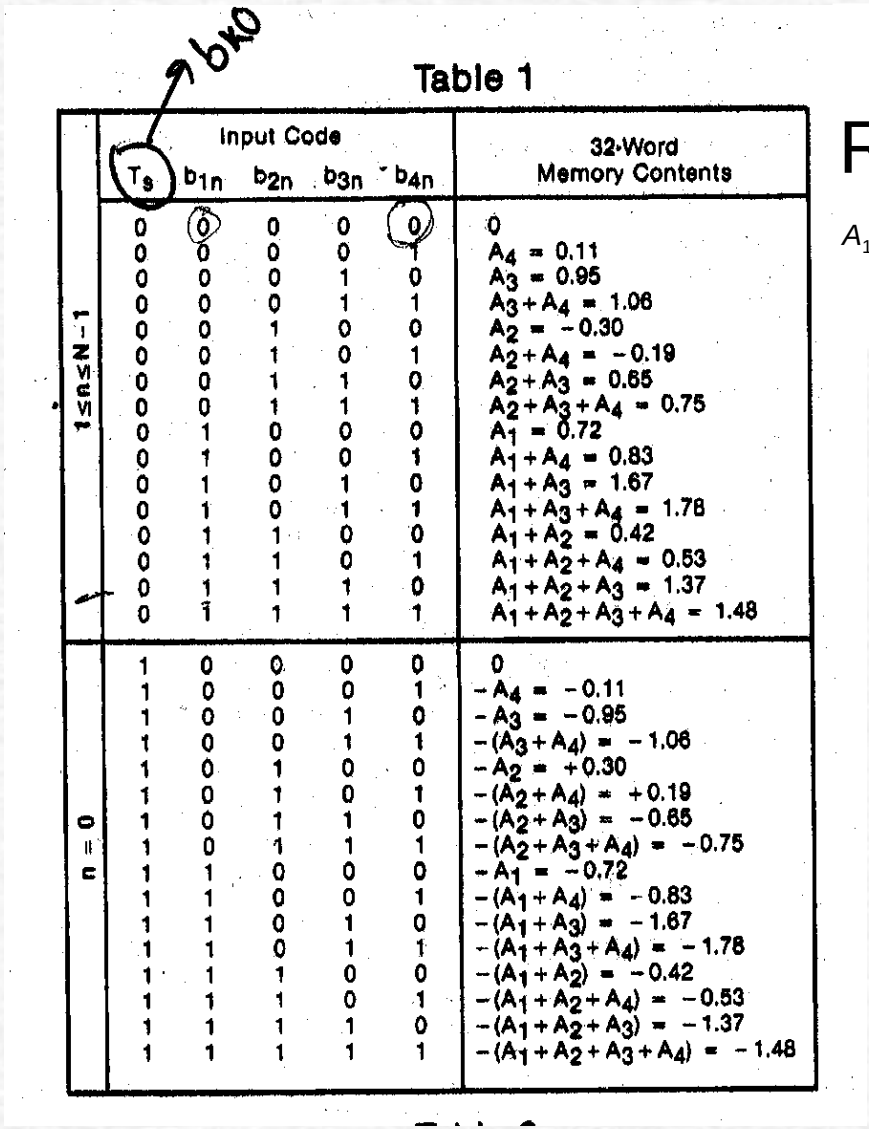


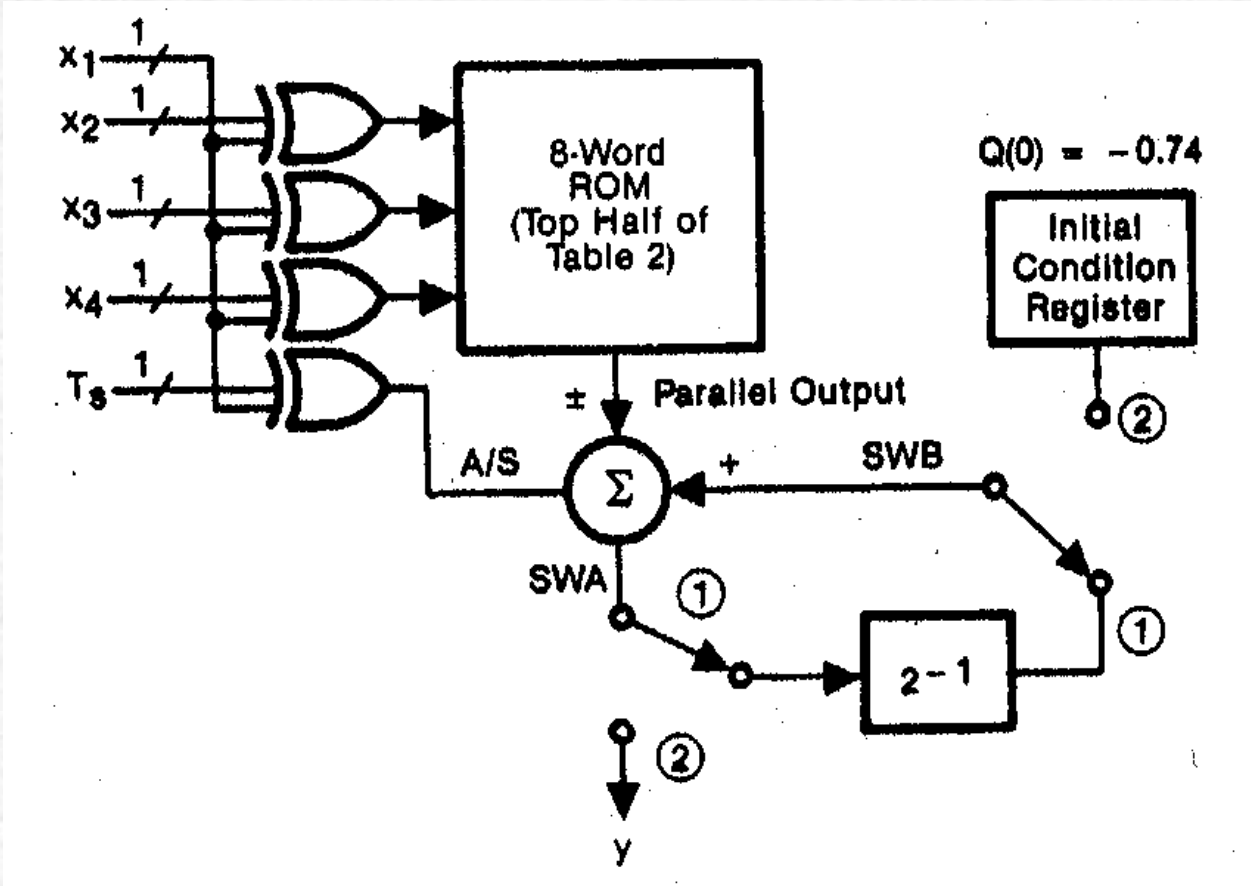
Table 1

	T_s	b_{1n}	b_{2n}	b_{3n}	b_{4n}	32-Word Memory Contents
$1 \le n \le N-1$	0	0	0	0	0	0
	0	0	0	0	1	$A_4 = 0.11$
	0	0	0	1	0	$A_3 = 0.95$
	0	0	0	1	1	$A_3 + A_4 = 1.06$
	0	0	1	0	0	$A_2 = -0.30$
	0	0	1	0	1	$A_2 + A_4 = -0.19$
	0	0	1	1	0	$A_2 + A_3 = 0.65$
	0	0	1	1	1	$A_2 + A_3 + A_4 = 0.75$
	0	1	0	0	0	$A_1 = 0.72$
	0	1	0	0	1	$A_1 + A_4 = 0.83$
	0	1	0	1	0	$A_1 + A_3 = 1.67$
	0	1	0	1	1	$A_1 + A_3 + A_4 = 1.78$
	0	1	1	0	0	$A_1 + A_2 = 0.42$
	0	1	1	0	1	$A_1 + A_2 + A_4 = 0.53$
	0	1	1	1	0	$A_1 + A_2 + A_3 = 1.37$
	0	1	1	1	1	$A_1 + A_2 + A_3 + A_4 = 1.48$
$n = 0$	1	0	0	0	0	0
	1	0	0	0	1	$-A_4 = -0.11$
	1	0	0	1	0	$-A_3 = -0.95$
	1	0	0	1	1	$-(A_3 + A_4) = -1.06$
	1	0	1	0	0	$-A_2 = +0.30$
	1	0	1	0	1	$-(A_2 + A_4) = +0.19$
	1	0	1	1	0	$-(A_2 + A_3) = -0.65$
	1	0	1	1	1	$-(A_2 + A_3 + A_4) = -0.75$
	1	1	0	0	0	$-A_1 = -0.72$
	1	1	0	0	1	$-(A_1 + A_4) = -0.83$
	1	1	0	1	0	$-(A_1 + A_3) = -1.67$
	1	1	0	1	1	$-(A_1 + A_3 + A_4) = -1.78$
	1	1	1	0	0	$-(A_1 + A_2) = -0.42$
	1	1	1	0	1	$-(A_1 + A_2 + A_4) = -0.53$
	1	1	1	1	0	$-(A_1 + A_2 + A_3) = -1.37$
	1	1	1	1	1	$-(A_1 + A_2 + A_3 + A_4) = -1.48$

ROM Table with

$A_1 = 0.72$, $A_2 = 0.32$, $A_3 = 0.95$, $A_4 = 0.11$

Overview of Distributed Arithmetic



Adder/ Subtractor and Reduced Memory

Overview of Distributed Arithmetic

Input Code b_{1n}	b_{2n}	b_{3n}	b_{4n}	8-Word Memory Contents, Q
0	0	0	0	$-1/2\,(A_1+A_2+A_3+A_4) = -0.74$
0	0	0	1	$-1/2\,(A_1+A_2+A_3-A_4) = -0.63$
0	0	1	0	$-1/2\,(A_1+A_2-A_3+A_4) = 0.21$
0	0	1	1	$-1/2\,(A_1+A_2-A_3-A_4) = 0.32$
0	1	0	0	$-1/2\,(A_1-A_2+A_3+A_4) = -1.04$
0	1	0	1	$-1/2\,(A_1-A_2+A_3-A_4) = -0.93$
0	1	1	0	$-1/2\,(A_1-A_2-A_3+A_4) = -0.09$
0	1	1	1	$-1/2\,(A_1-A_2-A_3-A_4) = 0.02$
1	0	0	0	$1/2\,(A_1-A_2-A_3-A_4) = -0.02$
1	0	0	1	$1/2\,(A_1-A_2-A_3+A_4) = 0.09$
1	0	1	0	$1/2\,(A_1-A_2+A_3-A_4) = 0.93$
1	0	1	1	$1/2\,(A_1-A_2+A_3+A_4) = 1.04$
1	1	0	0	$1/2\,(A_1+A_2-A_3-A_4) = -0.32$
1	1	0	1	$1/2\,(A_1+A_2-A_3+A_4) = -0.21$
1	1	1	0	$1/2\,(A_1+A_2+A_3-A_4) = 0.63$
1	1	1	1	$1/2\,(A_1+A_2+A_3+A_4) = 0.74$

ROM Table for Reduced Memory case

DA-Based FIR Implementation

A digital FIR filter is characterized by an input-output relationship of the form:

$$y_n = \sum_{k=0}^{N} h_k x_{n-k}$$

where $\{x_n\}$ is the input sequence, $\{y_n\}$is the output sequence, and $\{h_k\}$ are the filter coefficients.

Let us design a filter given by the relationship-

$$y_n = h_0 x_n + h_1 x_{n-1} + h_2 x_{n-2} + h_3 x_{n-3}$$

Let us assume that all signals are bounded by ±1 and that B binary bits including sign bit in 2's complement code are used to represent the data, that is-

$$x_k = -x_k^0 + \sum_{j=1}^{B-1} x_k^j 2^{-j} \qquad x_k^j = 0 \text{ or } 1$$

$$y_n = h_0\left(\sum_{j=1}^{B-1} x_n^j 2^{-j} - x_n^0\right) + h_1\left(\sum_{j=1}^{B-1} x_{n-1}^j 2^{-j} - x_{n-1}^0\right) + h_2\left(\sum_{j=1}^{B-1} x_{n-2}^j 2^{-j} - x_{n-2}^0\right) + h_3\left(\sum_{j=1}^{B-1} x_{n-3}^j 2^{-j} - x_{n-3}^0\right)$$

DA-Based FIR Implementation

Define a function ϕ with four binary arguments as follows:

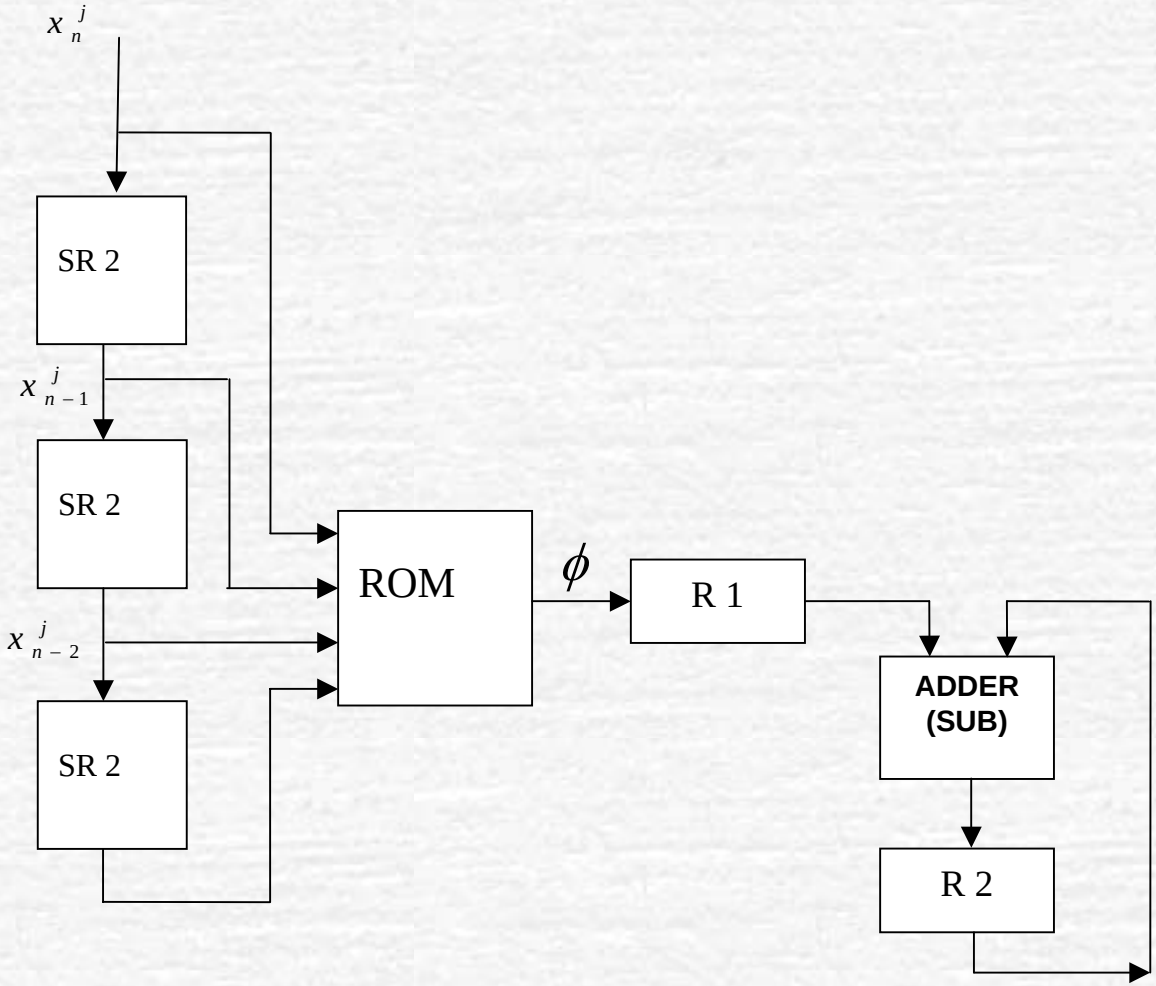
$$\Phi(x^1, x^2, x^3, x^4) = h_0 x^1 + h_1 x^2 + h_2 x^3 + h_3 x^4$$

Then we can rewrite y_n as

$$y_n = \sum_{j=1}^{B-1} 2^{-j} \Phi(x_n^j, x_{n-1}^j, x_{n-2}^j, x_{n-3}^j) - \Phi(x_n^0, x_{n-1}^0, x_{n-2}^0, x_{n-3}^0)$$

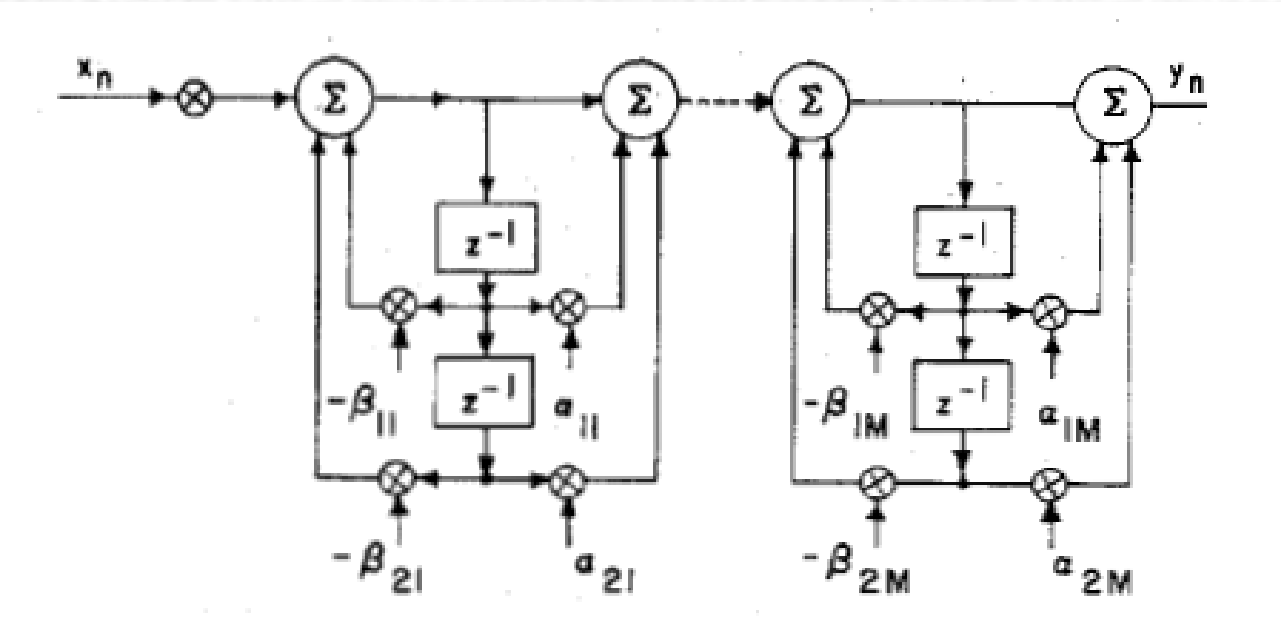
The function ϕ can take on only $2^4 = 16$ distinct values depending on the binary vector that forms its arguments. It can be realized by a combinatorial network or by a read only memory (ROM) addressed by the arguments. Thus (6) suggests a possible mechanization of (2) requiring only addition and shifting operations.

DA-Based FIR Implementation



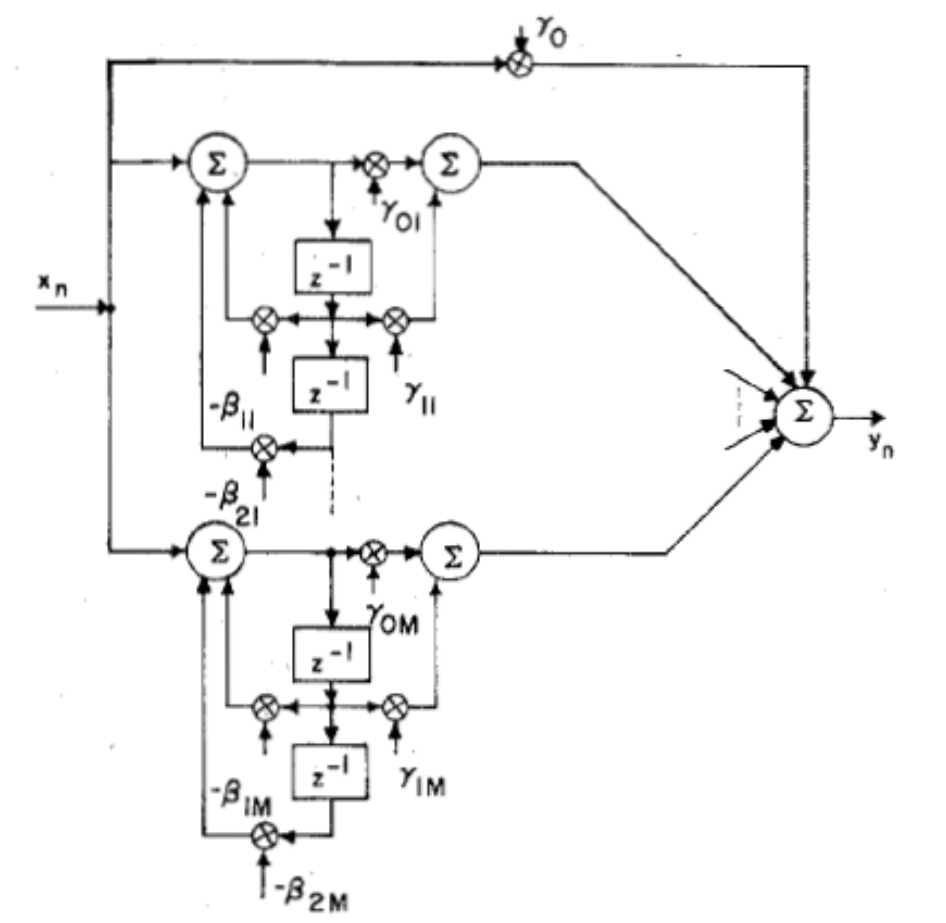
DA based hardware realization of a FIR Filter

DA-Based IIR Implementation



Cascade structure of a IIR Filter

DA-Based IIR Implementation



Parallel structure of a IIR Filter

DA-Based IIR Implementation

A digital IIR filter is characterized by an input-output relationship of the form-

$$y_n = \sum_{k=0}^{N} a_k x_{n-k} - \sum_{k=1}^{N} b_k y_{n-k}$$

where $\{x_n\}$ is the input sequence, $\{y_n\}$is the output sequence, and $\{a_k\},\{b_k\}$ are the filter coefficients.

Let us design a filter given by the relationship-

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} - b_1 y_{n-1} - b_2 y_{n-2}$$

$$x_k = -x_k^0 + \sum_{j=1}^{B-1} x_k^j 2^{-j} \qquad x_k^j = 0 \text{ or } 1$$

$$y_n = a_0\left(\sum_{j=1}^{B-1} x_n^j 2^{-j} - x_n^0\right) + a_1\left(\sum_{j=1}^{B-1} x_{n-1}^j 2^{-j} - x_{n-1}^0\right) + a_2\left(\sum_{j=1}^{B-1} x_{n-2}^j 2^{-j} - x_{n-2}^0\right)$$

$$-b_1\left(\sum_{j=1}^{B-1} y_{n-1}^j 2^{-j} - y_{n-1}^0\right) - b_2\left(\sum_{j=1}^{B-1} y_{n-2}^j 2^{-j} - y_{n-2}^0\right)$$

Define a function ϕ with five binary arguments as follows:

$$\Phi(x^1, x^2, x^3, x^4) = a_0 x^1 + a_1 x^2 + a_2 x^3 - b_1 x^4 - b_2 x^5$$

DA-Based IIR Implementation

Then we can rewrite y_n as : $y_n = \sum_{j=1}^{B-1} 2^{-j} \Phi(x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}^j, y_{n-2}^j) - \Phi(x_n^0, x_{n-1}^0, x_{n-2}^0, y_{n-1}^0, y_{n-2}^0)$

MEMORY ADDRESS	CONTENTS
00000	0 0 0 0 0 0 0
00001	1 1 0 0 0 1 1 0
00010	0 1 1 1 0 1 0 0
00011	0 0 1 1 1 0 0 1
00100	0 0 0 0 0 1 1 0
00101	1 1 0 0 1 1 0 0
00110	0 1 1 1 1 0 1 0
00111	0 0 1 1 1 1 1 1
01000	1 1 1 1 0 1 0 1
01001	1 0 1 1 1 0 1 1
01010	0 1 1 0 1 0 0 1
01011	0 0 1 0 1 1 1 1
01100	1 1 1 1 1 0 1 1
01101	1 1 0 0 0 0 0 1
01110	0 1 1 0 1 1 1 1
01111	0 0 1 1 0 1 0 1

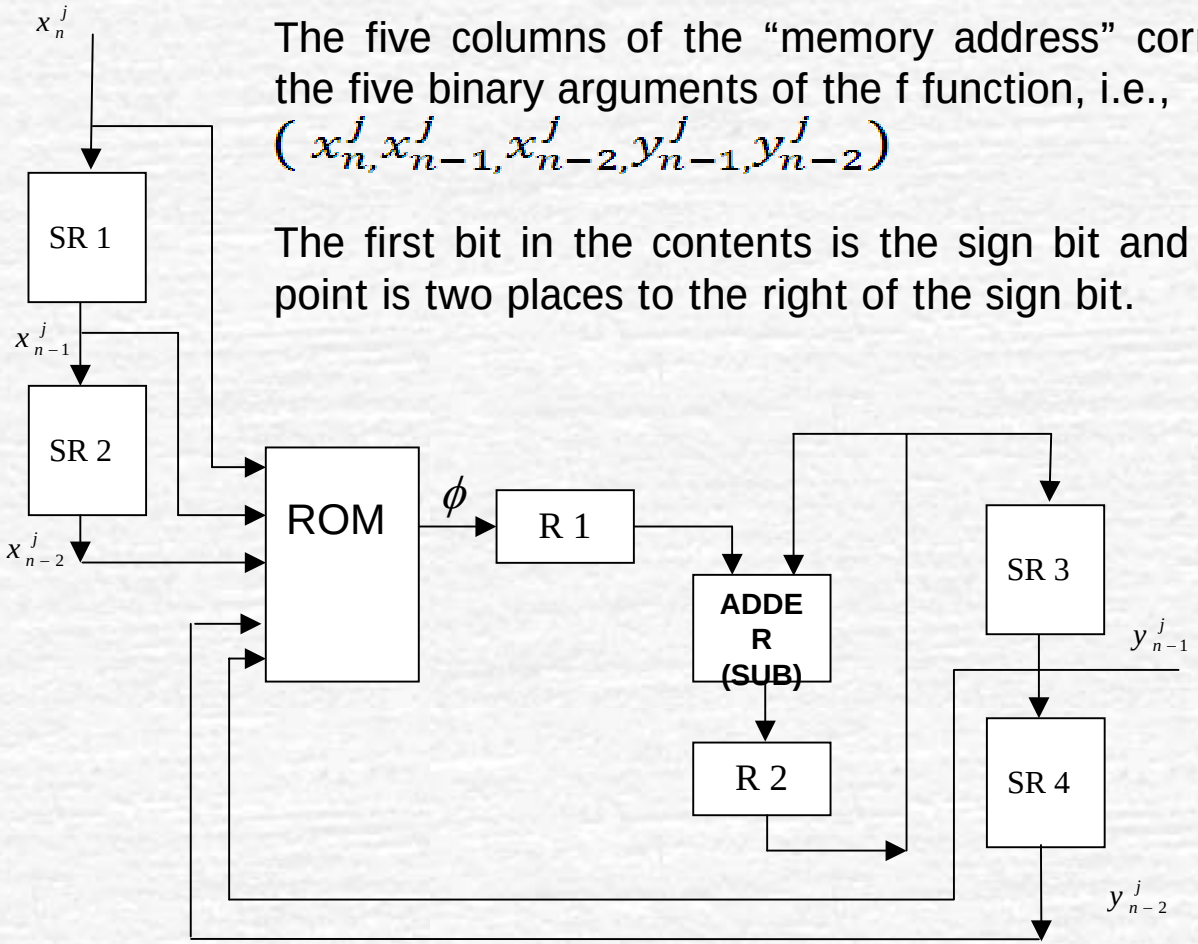
sign bit binary point

MEMORY ADDRESS	CONTENTS
10000	0 0 0 0 0 1 1 0
10001	1 1 0 0 1 1 0 0
10010	0 1 1 1 1 0 1 0
10011	0 0 1 1 1 1 1 1
10100	0 0 0 0 1 1 0 0
10101	1 1 0 1 0 0 1 0
10110	0 1 1 1 1 1 1 1
10111	0 1 0 0 0 1 0 1
11000	1 1 1 1 1 0 1 1
11001	1 1 0 0 0 0 0 1
11010	0 1 1 0 1 1 1 1
11011	0 0 1 1 0 1 0 1
11100	0 0 0 0 0 1 0 0
11101	1 1 0 0 0 1 1 1
11110	0 1 1 1 0 1 0 1
11111	0 0 1 1 1 1 0 1 1

sign bit binary point

In this example, $a_0 = 0.095$, $a_1 = -0.1665478$, $a_2 = 0.095$, $b_1 = -1.8080353$ and $b_2 = 0.9129197$.

DA-Based IIR Implementation

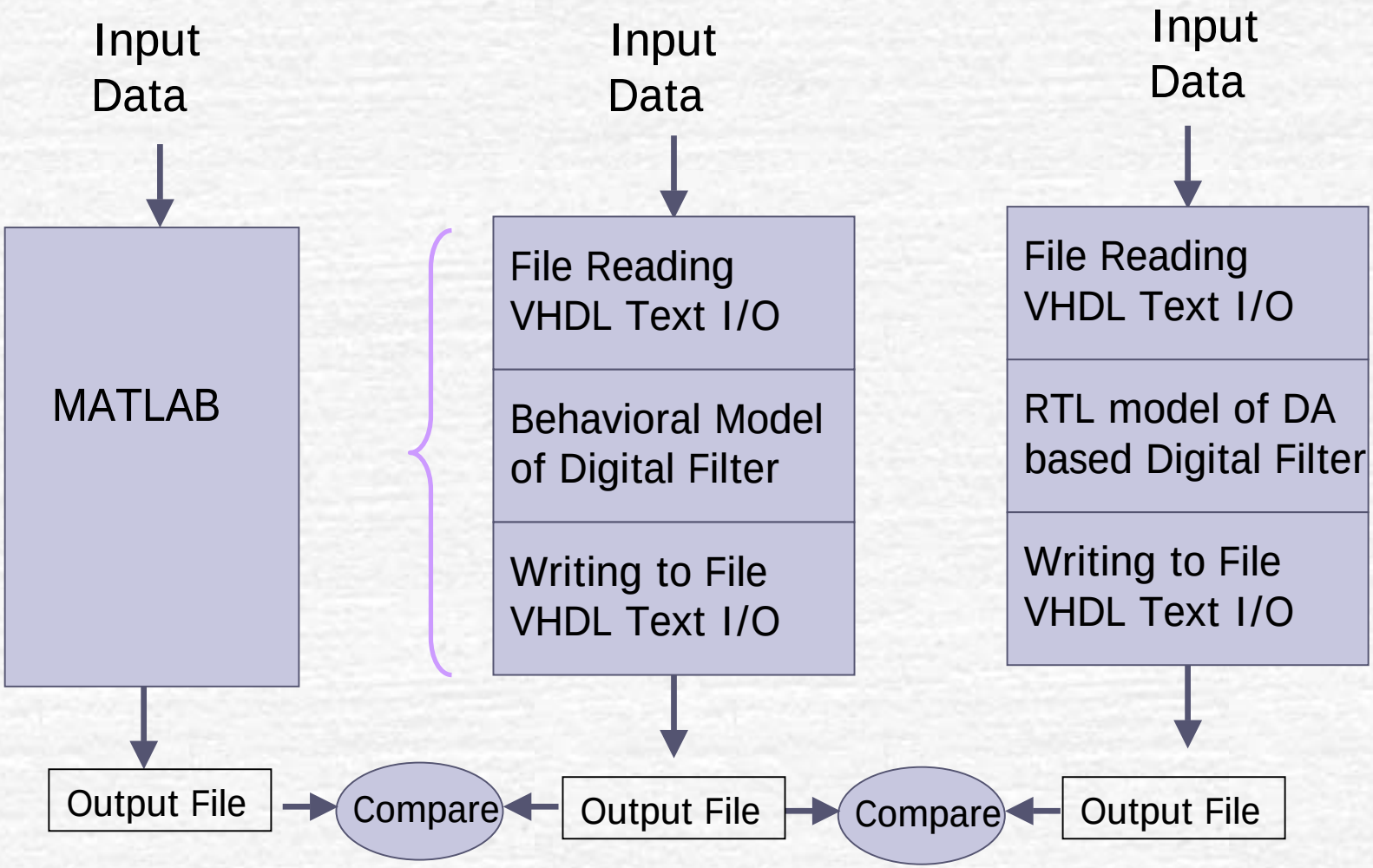


The five columns of the "memory address" correspond to the five binary arguments of the f function, i.e.,

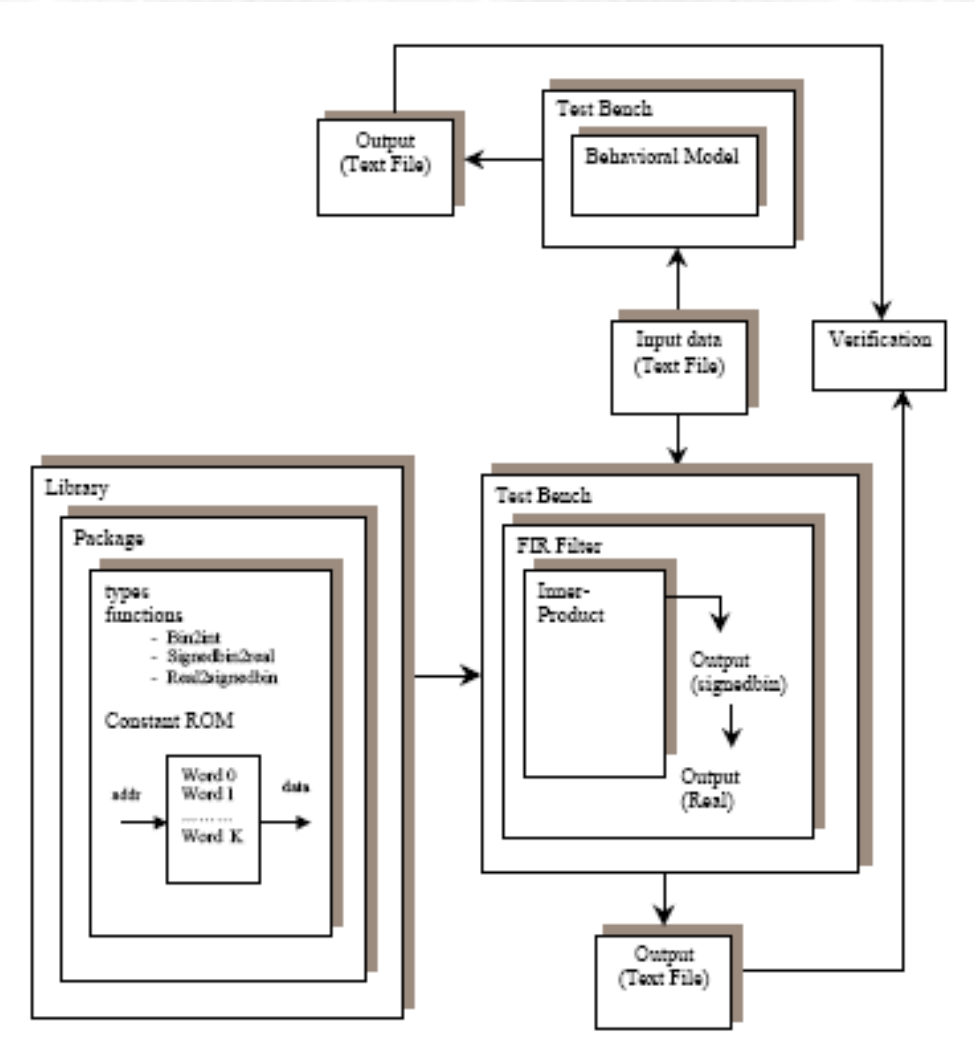
$$\left(x_n^j, x_{n-1}^j, x_{n-2}^j, y_{n-1}^j, y_{n-2}^j \right)$$

The first bit in the contents is the sign bit and the binary point is two places to the right of the sign bit.

Verification Strategy



Code Organization



Conclusion

1. DA is an efficient mean to mechanize computation that are dominated by inner products.
2. DA has been used for realizing other important DSP blocks such as DWT.
3. Compared to other computing methods, DA based methods have fared well.

References

[1] Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review, Stanley A. White, *IEEE Acoustics, Speech, and Signal Processing* (ASSP) Magazine, July 1989, Page 4-19.

[2] A New Hardware Realization of Digital Filters, *Abraham Peled and Bede Liu, IEEE Transactions on Acoustics, Speech, and Signal Processing*, Dec. 1974, Page 456-462.

THANK YOU