

Memory-Based Computing for DSP Applications

Pramod Meher

**Institute for Infocomm Research
Singapore**

outline

- trends in memory technology
- memory-based computing: advantages and examples
- DA-based computation for DSP applications
- lookup-table design for constant multiplication
- DA-based vs LUT-multiplier-based implementations
- memory-based evaluation of non-linear functions
- conclusions

trends in memory technology

Application-specific memories [1-4]

- low power memories for mobile devices and consumer products
- high-speed memories for multimedia applications
- wide temperature memories for automotive
- high reliability memories for biomedical instruments
- radiation hardened memory for space applications

trends in memory technology

RAM-logic integration

- several nonvolatile RAM types are emerging: ferroelectric RAM (FeRAM), magneto-resistive RAM (MRAM), and varieties of phase change memory (PCM) [4-6]
- the upcoming/new memories provide faster access and consume less power [4-6]
- can be embedded directly into the structure of microprocessors or integrated in the functional elements of dedicated processors [7]

trends in memory technology

memory placement [7-11]

- traditional concept of memory as a stand alone subsystem is getting changed
- it is embedded within the logic components
- processor has been moved to memory or memory has been moved to processor
- the relocations result in higher bandwidth, lower power consumption and less access-delay

memory-based computing ?

- a class of dedicated systems, where the computational functions are performed by lookup tables (LUTs), instead of actual calculations
- close to human-like computing
- simple to design, and more regular compared with the multiply-accumulate structures
- have potential for high-throughput and reduced-latency implementation
- involves less dynamic power consumption due to minimization of switching activities

memory-based computations: examples

- inner-product computation using the distributed arithmetic (DA) [12]
- direct implementation of constant multiplications [13]
- well-suited for digital filtering and orthogonal transformations for digital signal processing
- implementation of fixed and adaptive FIR filters and transforms
- other applications: evaluation of trigonometric functions, sigmoid and other nonlinear function

DA to calculate inner-product : example

$\mathbf{X} = [X_0 \ X_1 \ X_2]^T$ and $\mathbf{A} = [A_0, A_1, A_2]^T$: 3-point vectors. \mathbf{A} is constant

X_0, X_1 and X_2 be 4-bit integers:

$$X_0 = [x_0(3) \ x_0(2) \ x_0(1) \ x_0(0)]$$
$$X_1 = [x_1(3) \ x_1(2) \ x_1(1) \ x_1(0)]$$
$$X_2 = [x_2(3) \ x_2(2) \ x_2(1) \ x_2(0)]$$

inner-product of \mathbf{X} and \mathbf{A} : $\mathbf{A} \cdot \mathbf{X} = A_0X_0 + A_1X_1 + A_2X_2$

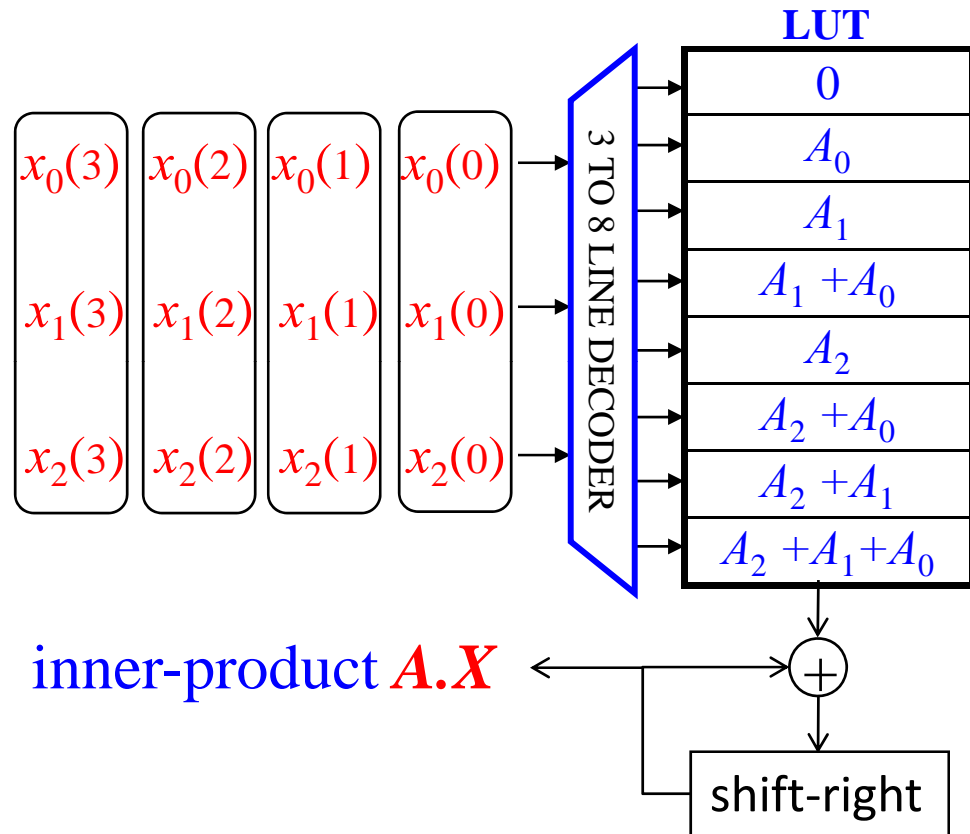
$$\begin{aligned} A_0X_0 &= \boxed{A_0 \cdot x_0(0)} + 2^1 \boxed{A_0 \cdot x_0(1)} + 2^2 \boxed{A_0 \cdot x_0(2)} + 2^3 \boxed{A_0 \cdot x_0(3)} \\ A_1X_1 &= \boxed{A_1 \cdot x_1(0)} + 2^1 \boxed{A_1 \cdot x_1(1)} + 2^2 \boxed{A_1 \cdot x_1(2)} + 2^3 \boxed{A_1 \cdot x_1(3)} \\ A_2X_2 &= \boxed{A_2 \cdot x_2(0)} + 2^1 \boxed{A_2 \cdot x_2(1)} + 2^2 \boxed{A_2 \cdot x_2(2)} + 2^3 \boxed{A_2 \cdot x_2(3)} \end{aligned}$$

$P_0 \qquad \qquad \qquad P_1 \qquad \qquad \qquad P_2 \qquad \qquad \qquad P_3$

inner-product of : $\mathbf{A} \cdot \mathbf{X} = P_0 + 2P_1 + 4P_2 + 8P_3$

LUT for inner-product using DA [12]

$x_2(i)$	$x_1(i)$	$x_0(i)$	partial sum
0	0	0	0
0	0	1	A_0
0	1	0	A_1
0	1	1	$A_1 + A_0$
1	0	0	A_2
1	0	1	$A_2 + A_0$
1	1	0	$A_2 + A_1$
1	1	1	$A_2 + A_1 + A_0$



2^N LUT words required for N -point inner-product. For $N=32$, it exceeds 10^9 words!!

For L -bit inputs, computation time = L cycles : Cycle time, $T = T_{MEM} + T_{ADD} + T_{FF}$

LUT compaction for DA [12]

$x_2(i)$	$x_1(i)$	$x_0(i)$	conventional	OBC LUT content
0	0	0	0	$-(A_2 + A_1 + A_0)$
0	0	1	A_0	$-(A_2 + A_1 - A_0)$
0	1	0	A_1	$-(A_2 - A_1 + A_0)$
0	1	1	$A_1 + A_0$	$-(A_2 - A_1 - A_0)$
1	0	0	A_2	$(A_2 - A_1 - A_0)$
1	0	1	$A_2 + A_0$	$(A_2 - A_1 + A_0)$
1	1	0	$A_2 + A_1$	$(A_2 + A_1 - A_0)$
1	1	1	$A_2 + A_1 + A_0$	$(A_2 + A_1 + A_0)$

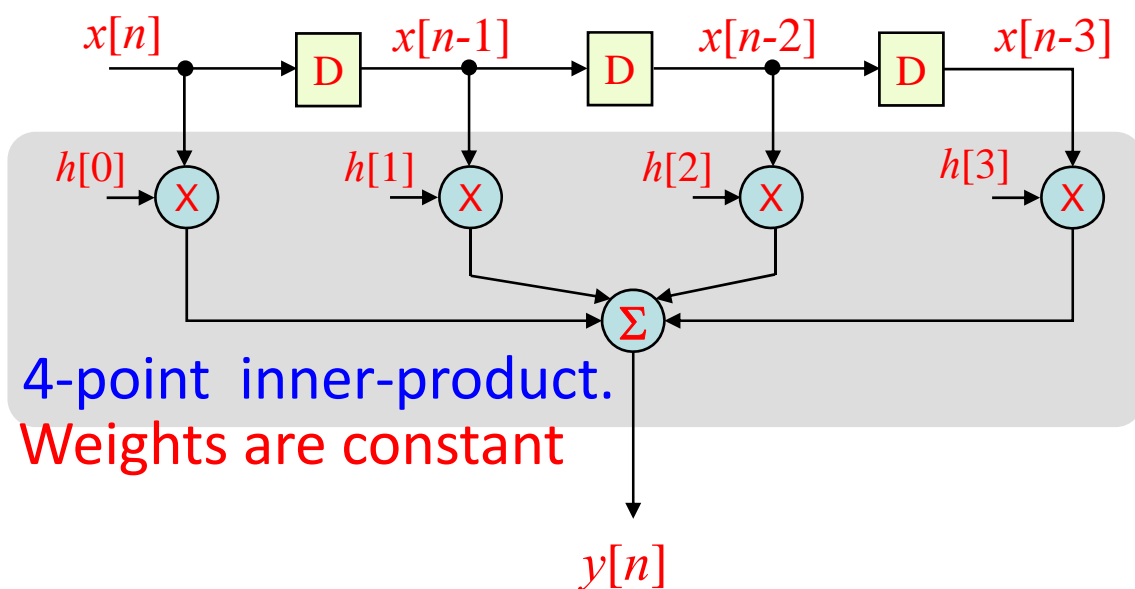
Desired partial sum of product = [OBC value + $(A_2 + A_1 + A_0)$]/2
 half the number of LUT words are saved if OBC is used

linear convolution/ FIR filtering [13]

N -tap FIR filter equation:

$$y[n] = h[0].x[n] + h[1].x[n-1] + \dots + \dots + h[N-1].x[n-N+1]$$

direct-form FIR filter for $N=4$.

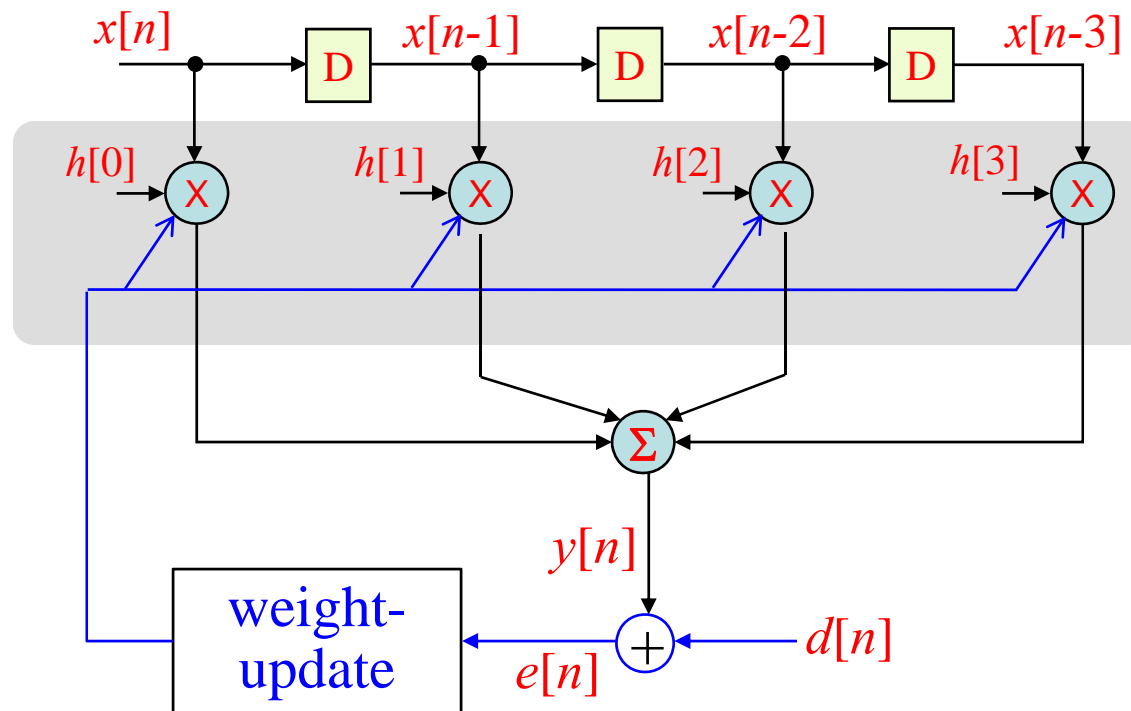


address	LUT content
0000	0
0001	$h[0]$
0010	$h[1]$
0011	$h[1]+h[0]$
0100	$h[2]$
0101	$h[2]+h[0]$
0110	$h[2]+h[1]$
0111	$h[2]+h[1]+h[0]$
1000	$h[3]$
1001	$h[3]+h[0]$
1010	$h[3]+h[1]$
1011	$h[3]+h[1]+h[0]$
1100	$h[3]+h[2]$
1101	$h[3]+h[2]+h[0]$
1110	$h[3]+h[2]+h[1]$
1111	$h[3]+h[2]+h[1]+h[0]$

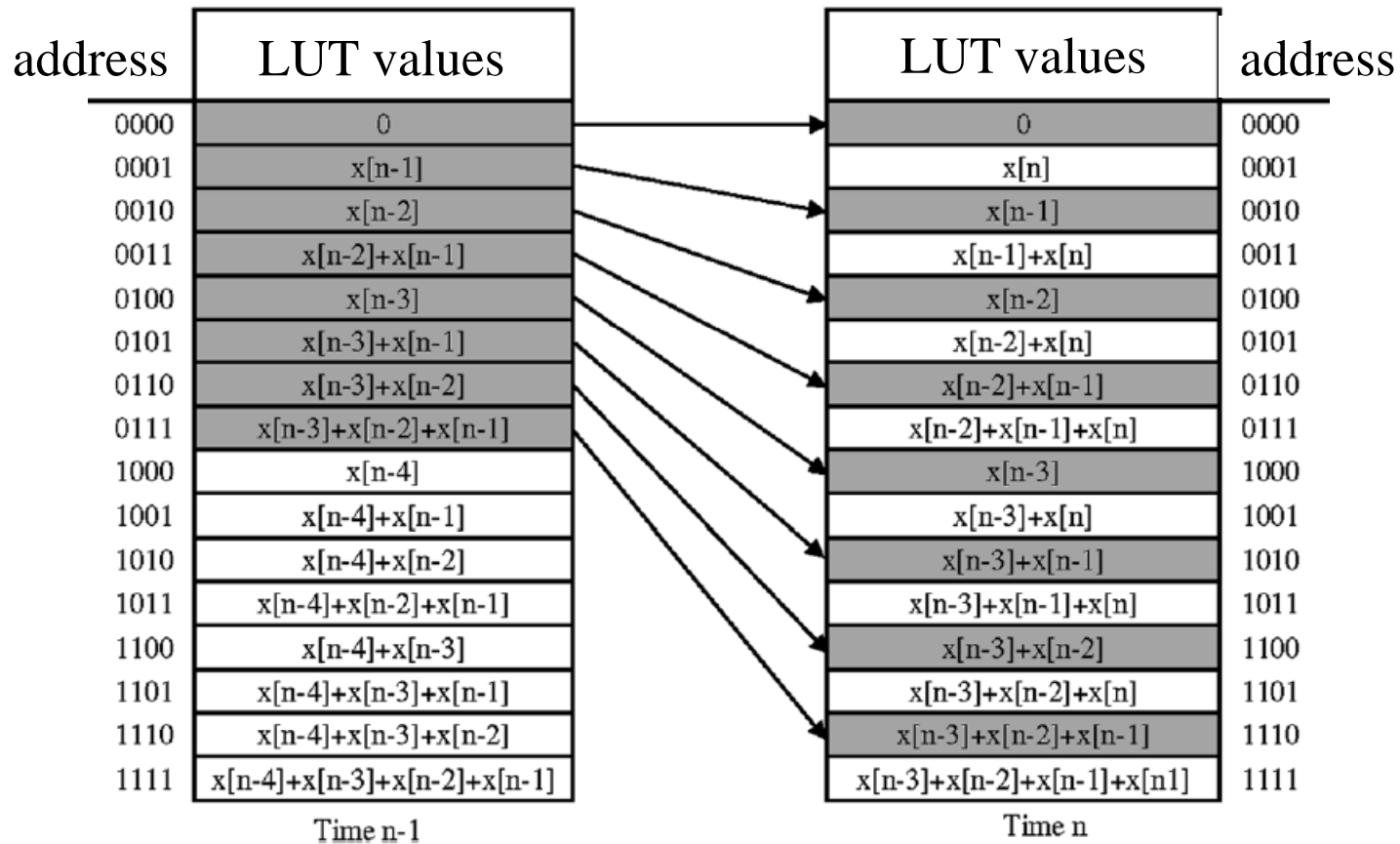
DA-based adaptive filtering [14]

example: 4-tap FIR adaptive filter

4-point
inner-product.
Weights are not
constant.



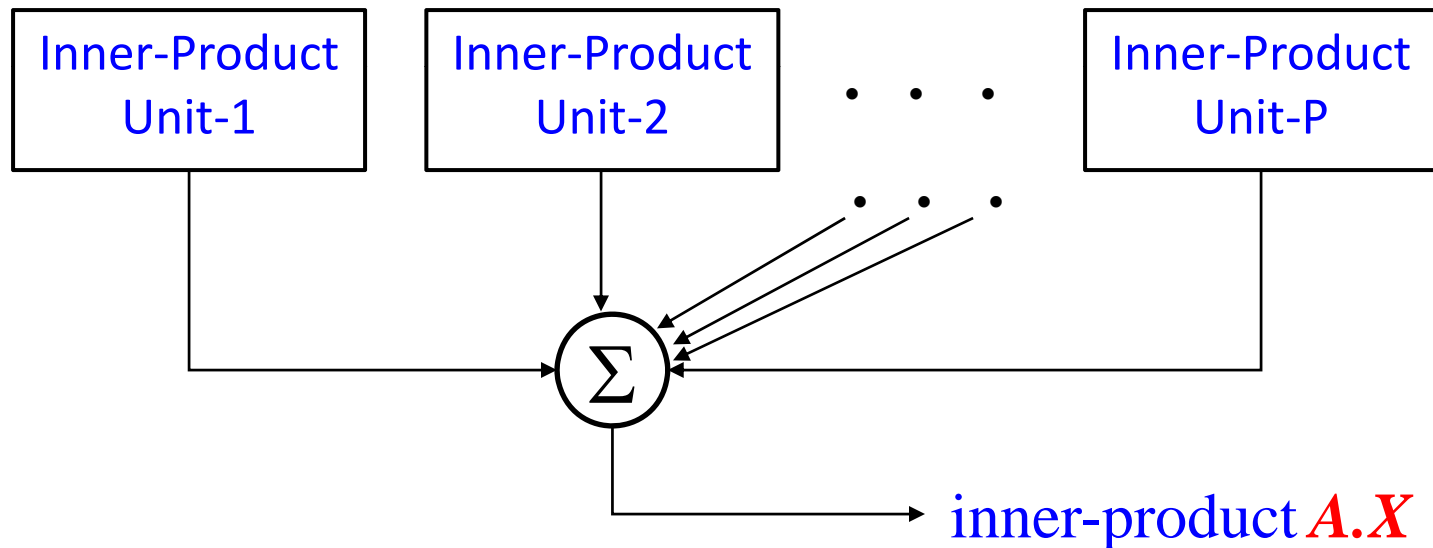
LUT for adaptive filter: example [14]



bits of the same place values of the filter coefficients are used as addresses

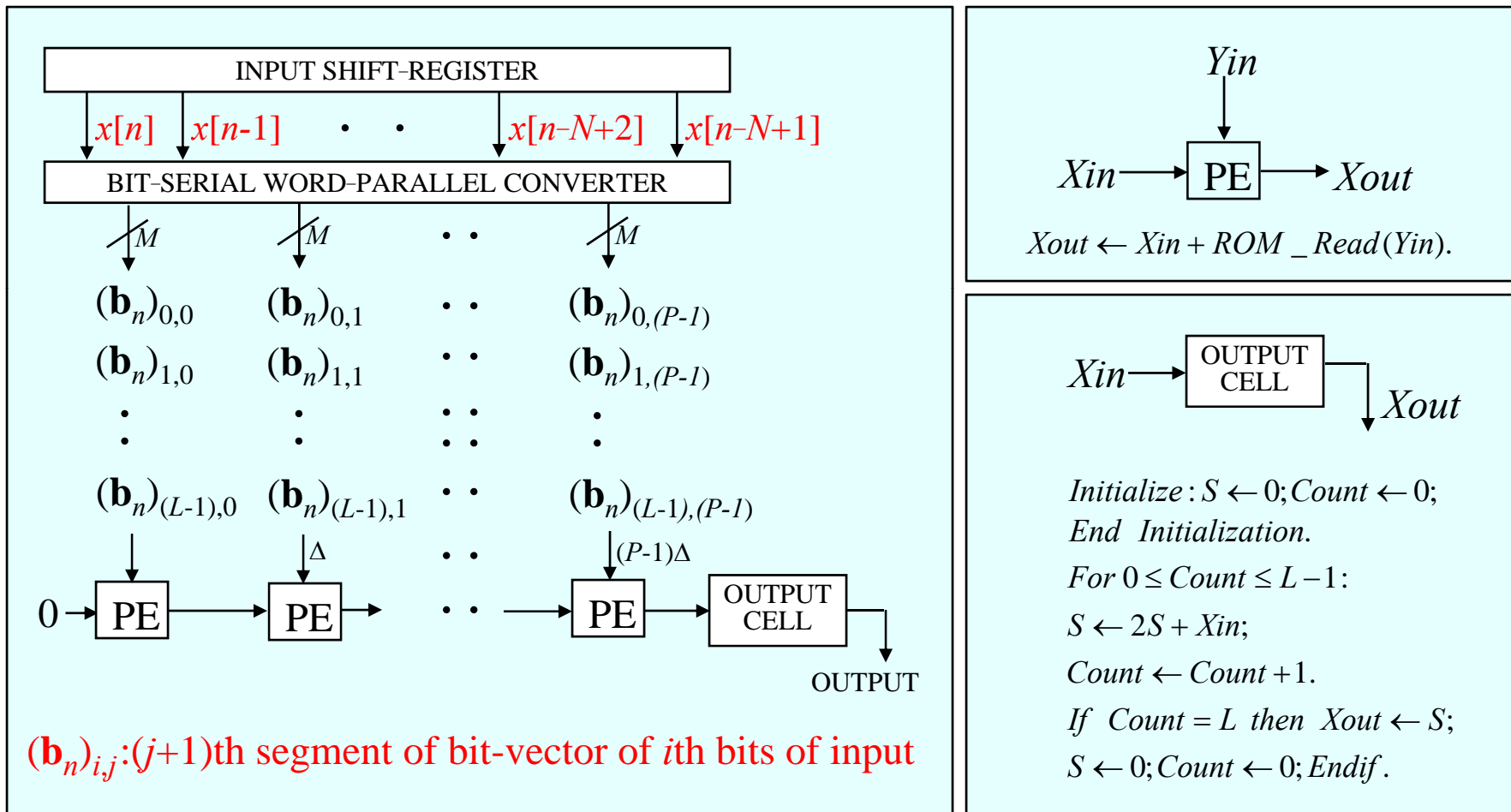
DA-based inner-product of long vectors

$$\mathbf{AX} = \sum_{n=0}^{N-1} A_n \cdot X_n = \sum_{n=0}^{P-1} A_n \cdot X_n + \sum_{n=P}^{2P-1} A_n \cdot X_n + \cdots + \sum_{n=P(M-1)}^{MP-1} A_n \cdot X_n \quad \text{for } N=MP$$

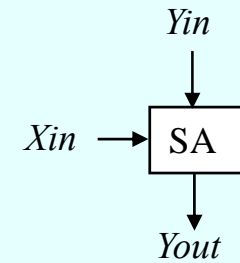
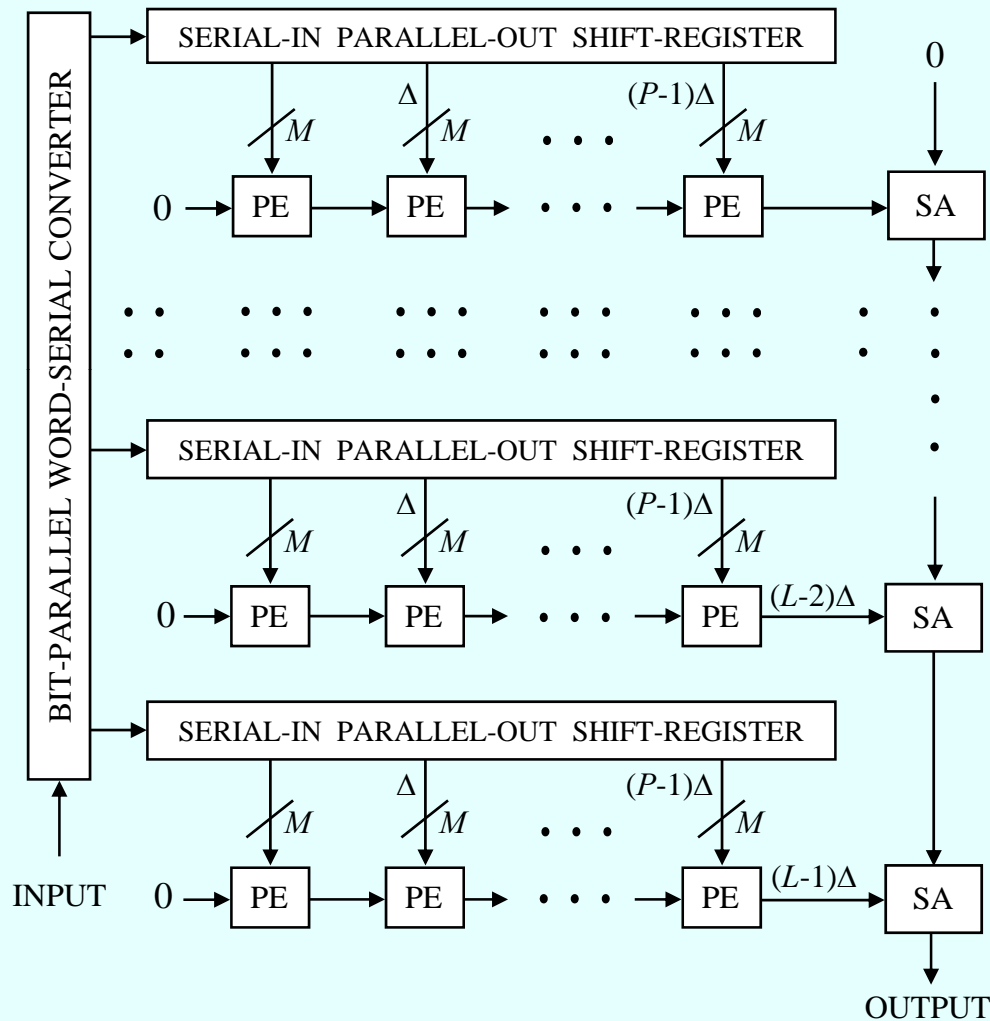


P LUTs of 2^M words and $(P-1)$ adders required for N -point inner-product.

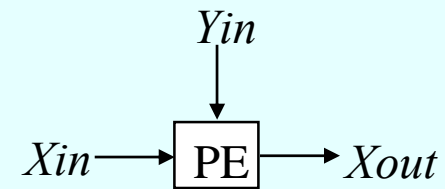
large order FIR filter using DA [15]



large order FIR filter: a 2-D design [15]



$$Y_{out} \leftarrow X_{in} + 2 \cdot Y_{in}$$



$$X_{out} \leftarrow X_{in} + ROM_Read(Y_{in}).$$

circular Convolution using DA [16]

circular convolution of two N -point sequences $\{x(n)\}$ and $\{h(n)\}$ is :

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot x((n-k) \bmod N) \quad \text{for } 0 \leq n \leq N-1$$

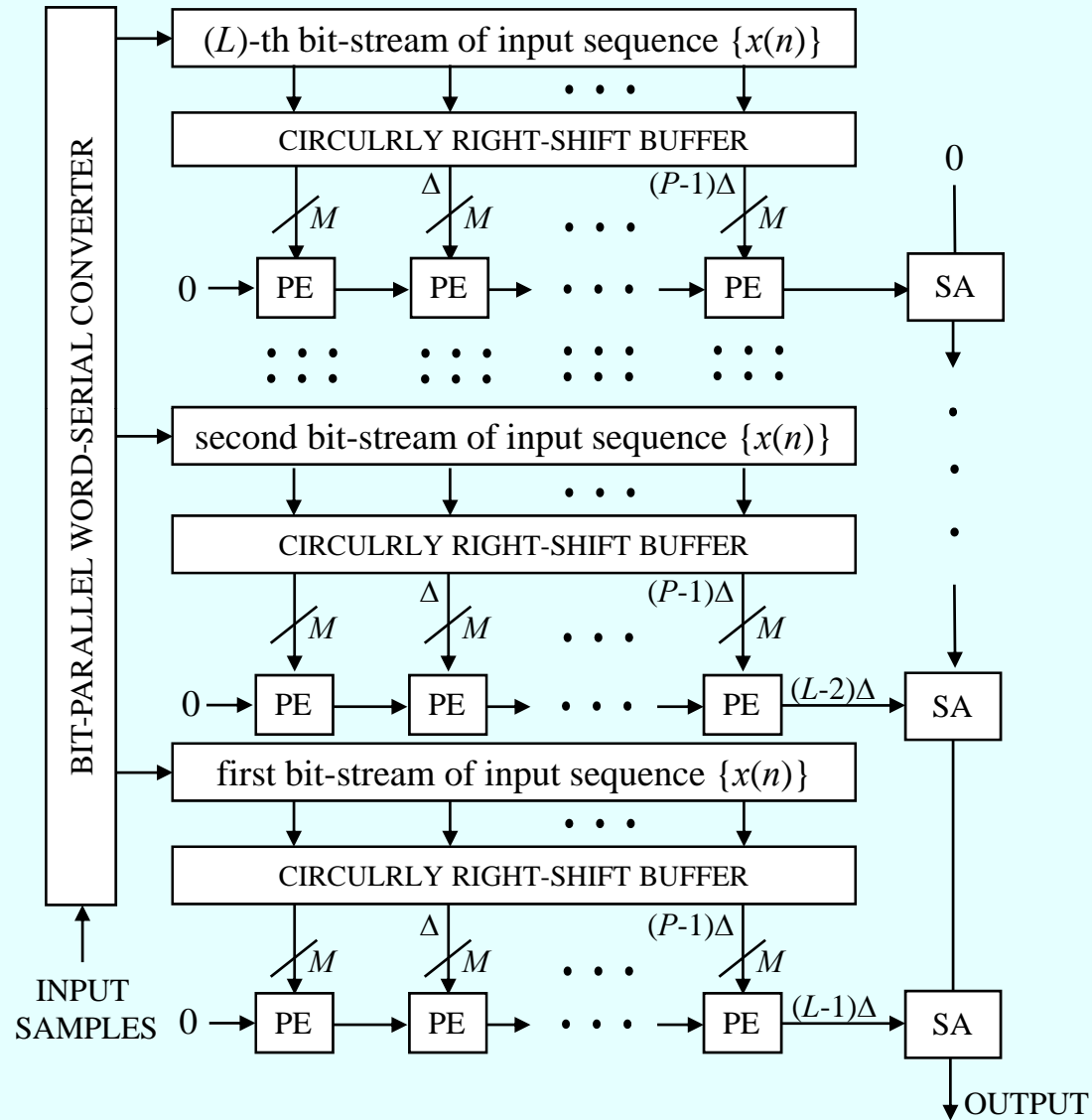
circular convolution for $N=4$:

$$\begin{aligned} y(0) &= h(0) \cdot x(0) + h(1) \cdot x(3) + h(2) \cdot x(2) + h(3) \cdot x(1) \\ y(1) &= h(0) \cdot x(1) + h(1) \cdot x(0) + h(2) \cdot x(3) + h(3) \cdot x(2) \\ y(2) &= h(0) \cdot x(2) + h(1) \cdot x(1) + h(2) \cdot x(0) + h(3) \cdot x(3) \\ y(3) &= h(0) \cdot x(3) + h(1) \cdot x(2) + h(2) \cdot x(1) + h(3) \cdot x(0) \end{aligned}$$



$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} x(0) & x(3) & x(2) & x(1) \\ x(1) & x(0) & x(3) & x(2) \\ x(2) & x(1) & x(0) & x(3) \\ x(3) & x(2) & x(1) & x(0) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ h(2) \\ h(3) \end{bmatrix}$$

cyclic convolution using DA: a 2-D design [16]



computation of sinusoidal transforms [17-20]

N -point sinusoidal transforms like the DFT, DCT and DHT are given by

$$X(k) = \sum_{n=0}^{N-1} C(k, n) \cdot x(n), \text{ for } k = 0, 1, \dots, N - 1$$

where the transform kernel is defined as

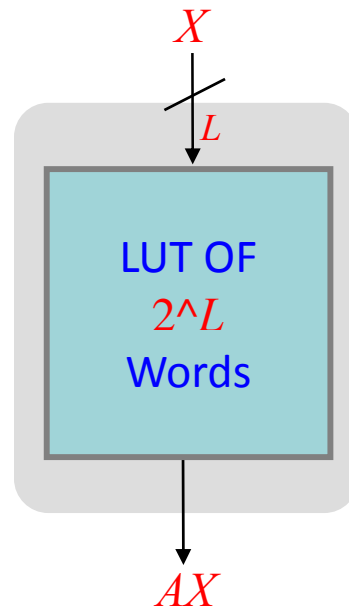
$$C(k, l) = \begin{cases} \cos(2\pi kn/N) - j \sin(2\pi kn/N), & \text{for DFT} \\ \cos(2\pi kn/N) + \sin(2\pi kn/N), & \text{for DHT} \\ \cos(\pi k(2n + 1)/2N), & \text{for DCT.} \end{cases}$$

- computation of N -point sinusoidal transforms involves multiplication of an $N \times N$ kernel matrix with N -point input vectors
- involves N number of inner-products of N -point input vector with the rows of kernel matrix
- the matrix-vector product requires N inner-product computation units by the DA approach
- for prime values of N , the $N \times N$ kernel matrix is transformed to an $(N-1)$ -point cyclic convolution.

multiplication using look-up-table

multiplication of an L -bit number X with constant A will require an LUT of 2^L words

multiplication time = memory latency



LUT to multiply a 4-bit word X with a constant A

address word, X	product word	address word, X	product word
0000	0	1000	8A
0001	A	1001	9A
0010	2A	1010	10A
0011	3A	1011	11A
0100	4A	1100	12A
0101	5A	1101	13A
0110	6A	1110	14A
0111	7A	1111	15A


LUT size increases exponentially with input size.

optimization for constant multiplications

- odd-multiple storage (OMS) scheme
- anti-symmetric product coding (APC) scheme
- input coding (IC) scheme
- combined techniques

odd-multiple storage scheme [21]

address word	product word	address word	product word
0000	0	1000	8A
0001	A	1001	9A
0010	2A	1010	10A
0011	3A	1011	11A
0100	4A	1100	12A
0101	5A	1101	13A
0110	6A	1110	14A
0111	7A	1111	15A



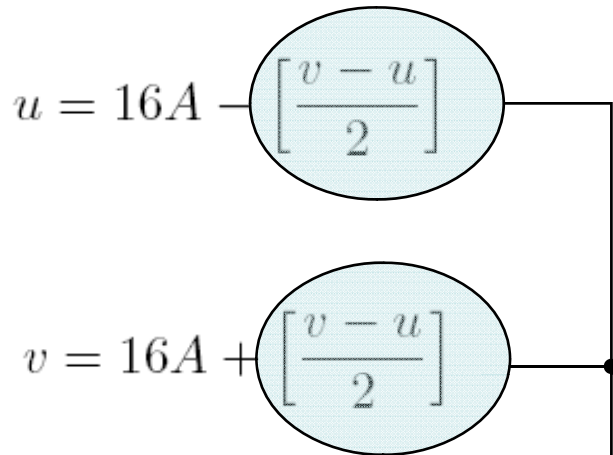
address word	product word
0001	A
0011	3A
0101	5A
0111	7A
1001	9A
1011	11A
1101	13A
1111	15A

- Only odd multiple of the constant are to be stored in the LUT.
- Even multiples could be derived from the stored words.
- Only half the number of product words are to be saved.

odd-multiple storage scheme [21]

- memory-unit of $(2^L)/2$ words of $(W+L)$ -bit width is used to store the odd multiples of constant A .
- a barrel-shifter for producing a maximum of $(L-1)$ left-shifts is used to derive all the even multiples of A .
- the L -bit input word is mapped to $(L-1)$ -bit address of the LUT by an encoder.
- the control-bits for barrel-shifter are derived by a control-circuit to perform the necessary shifts of the LUT output.
- RESET signal is generated by the same control circuit to reset the LUT output when the $X=0$.
- if only magnitude part could be used as address, LUT size is reduced to half.

anti-symmetric product coding [22]



instead of 32 words we need only 17 words to be stored in the LUT.

useful for high-precision multiplication and inner-product computation.

THE APC WORDS FOR DIFFERENT INPUT VALUES FOR $L = 5$

Input, X	product values	Input, X	product values	address $x'_3 x'_2 x'_1 x'_0$	APC words
0 0 0 0 1	A	1 1 1 1 1	31A	1 1 1 1	15A
0 0 0 1 0	2A	1 1 1 1 0	30A	1 1 1 0	14A
0 0 0 1 1	3A	1 1 1 0 1	29A	1 1 0 1	13A
0 0 1 0 0	4A	1 1 1 0 0	28A	1 1 0 0	12A
0 0 1 0 1	5A	1 1 0 1 1	27A	1 0 1 1	11A
0 0 1 1 0	6A	1 1 0 1 0	26A	1 0 1 0	10A
0 0 1 1 1	7A	1 1 0 0 1	25A	1 0 0 1	9A
0 1 0 0 0	8A	1 1 0 0 0	24A	1 0 0 0	8A
0 1 0 0 1	9A	1 0 1 1 1	23A	0 1 1 1	7A
0 1 0 1 0	10A	1 0 1 1 0	22A	0 1 1 0	6A
0 1 0 1 1	11A	1 0 1 0 1	21A	0 1 0 1	5A
0 1 1 0 0	12A	1 0 1 0 0	20A	0 1 0 0	4A
0 1 1 0 1	13A	1 0 0 1 1	19A	0 0 1 1	3A
0 1 1 1 0	14A	1 0 0 1 0	18A	0 0 1 0	2A
0 1 1 1 1	15A	1 0 0 0 1	17A	0 0 0 1	A
1 0 0 0 0	16A	1 0 0 0 0	16A	0 0 0 0	0

Arrows point from the pink box labeled u to the product values column (A-15A), from the orange box labeled v to the product values column (16A-31A), and from the blue box labeled 0 to the APC words column (0-15A).

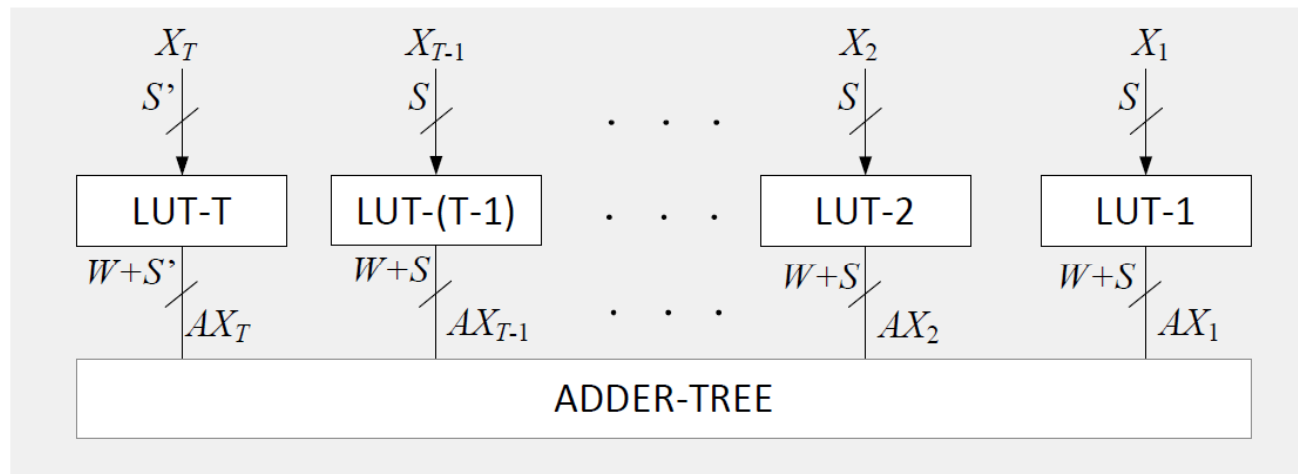
For $X = (0 0 0 0 0)$, the encoded word to be stored is 16A.

high-precision LUT-multiplier [22]

When the width of input multiplicand X is large, direct implementation of LUT-multiplier involves very large LUT.

But, the input word X could be decomposed into certain number of segments or sub-words $X=(X_1, X_2, \dots, X_T)$ and fed to separate LUTs.

The partial products pertaining to different sub-words could be read from the LUTs and shift-added to obtain the product values.



Generalized Architecture for High-Precision LUT-based Multiplier for $L = S(T - 1) + S'$.

input coding scheme: example [23]

$$X = (1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1).$$

We can decompose it to four words as

$$X = (1\ 0\ 1\ 1)\ (0\ 1\ 0\ 1)\ (1\ 1\ 0\ 0)\ (0\ 1\ 1\ 1).$$

i	X_i	D_i	C_i	D'_i	X'_i
1	0 1 1 1	7	0	7	0 1 1 1
2	1 1 0 0	12	1	4	0 1 0 0
3	0 1 0 1	5	0	6	0 1 1 0
4	1 0 1 1	11	1	5	0 1 0 1

$$C = AX = 7A + (12 \times 2^4)A + (5 \times 2^8)A + (11 \times 2^{12})A.$$

Using the encoded representation C can be expressed as

$$C = 7A - (4 \times 2^4)A + (6 \times 2^8)A - (5 \times 2^{12})A + 2^{16}A.$$

input coding scheme: basic concepts

Let the input word X consisting of T sub-words of S -bit each, be given by

$$X = \{X_T \cdots X_2 X_1\}$$

Then the encoded word X' consists of T sub-words, and a single-bit flag given by

$$X' = \{C_T X'_T \cdots X'_2 X'_1\}$$

where each X'_i is an $(S-1)$ -bit sub-word and C_T is an one-bit flag called as the overflow-flag.

Each sub-word X_i is associated with a carry-flag C_i such that for $N = 2^S$ it is given by

$$C_i = \begin{cases} 1, & \text{if } D_i \geq N/2 \\ 0, & \text{otherwise} \end{cases}$$

where X_i is the binary equivalent of integer D_i .

input coding scheme: a case for $L=5$

THE ENCODED WORDS AND STORED WORDS FOR DIFFERENT VALUES OF
 i -TH SUB-WORD X_i OF SIZE, $L = 5$

X_i	$C_i=0, C_{i-1}=1$		$C_i=1, C_{i-1}=0$		$C_i=1, C_{i-1}=1$	
	X'_i	AD'_i	X'_i	AD'_i	X'_i	AD'_i
x 0000	00001	A	10000	16A	01111	15A
x 0001	00010	2A	01111	15A	01110	14A
x 0010	00011	3A	01110	14A	01101	13A
x 0011	00100	4A	01101	13A	01100	12A
x 0100	00101	5A	01100	12A	01011	11A
x 0101	00110	6A	01011	11A	01010	10A
x 0110	00111	7A	01010	10A	01001	9A
x 0111	01000	8A	01001	9A	01000	8A
x 1000	01001	9A	01000	8A	00111	7A
x 1001	01010	10A	00111	7A	00110	6A
x 1010	01011	11A	00110	6A	00101	5A
x 1011	01100	12A	00101	5A	00100	4A
x 1100	01101	13A	00100	4A	00011	3A
x 1101	01110	14A	00011	3A	00010	2A
x 1110	01111	15A	00010	2A	00001	A
x 1111	10000	16A	00001	A	00000	0

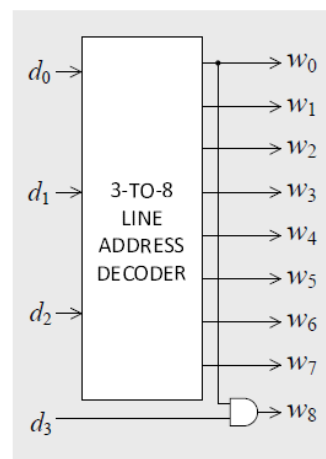
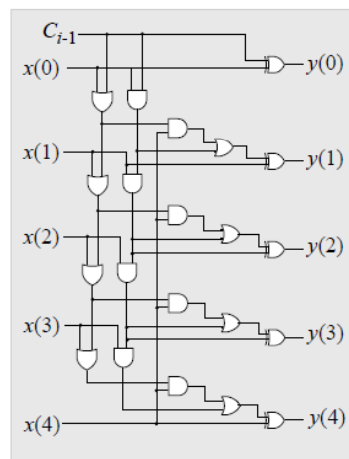
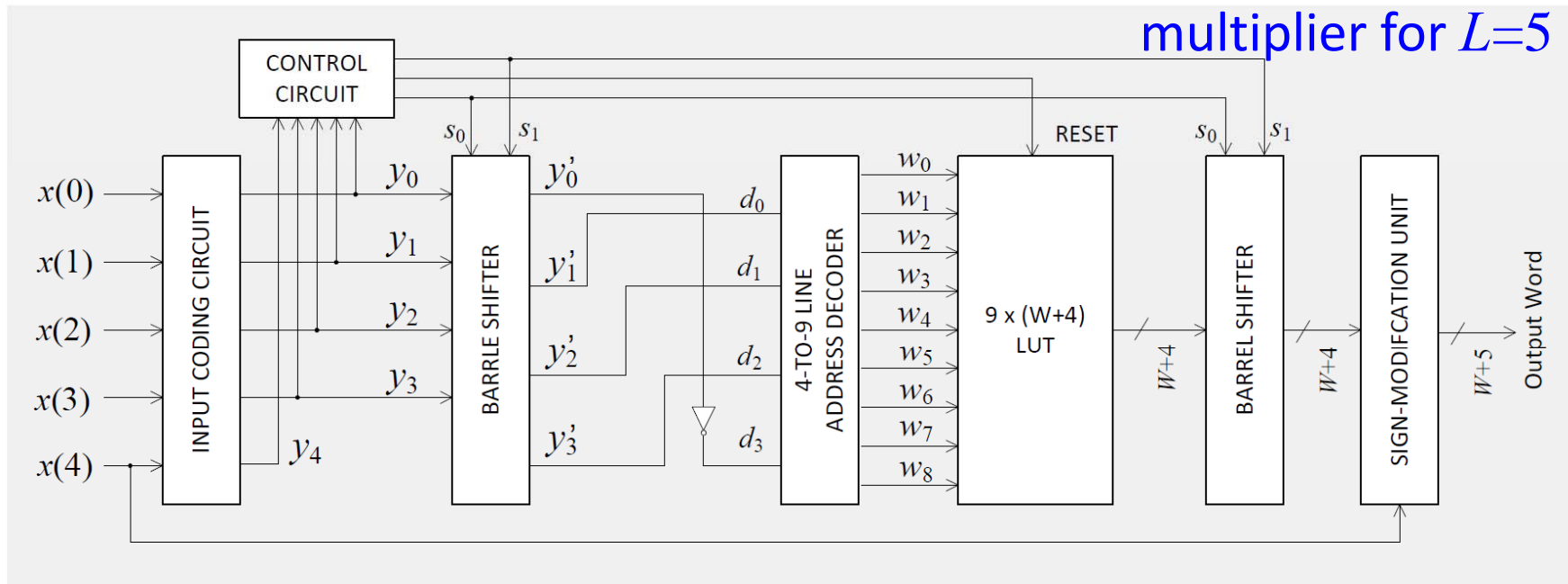
The MSB $x_i = C_i$, which may be 0 or 1. For $C_i = C_{i-1} = 0$ the input words are the same as the encoded words.

combining input coding with OMS

OMS-BASED DESIGN OF LUT OF INPUT CODED WORDS FOR $L = 5$

encoded word, Y	product value	# of shifts	shifted LSBs	LUT word	address $d_3d_2d_1d_0$
0 0 0 0 1	A	0	0 0 0 1	A	0 0 0 0
0 0 0 1 0	$2 \times A$	1			
0 0 1 0 0	$4 \times A$	2			
0 1 0 0 0	$8 \times A$	3			
0 0 0 1 1	$3A$	0	0 0 1 1	$3A$	0 0 0 1
0 0 1 1 0	$2 \times 3A$	1			
0 1 1 0 0	$4 \times 3A$	2			
0 0 1 0 1	$5A$	0	0 1 0 1	$5A$	0 0 1 0
0 1 0 1 0	$2 \times 5A$	1			
0 0 1 1 1	$7A$	0	0 1 1 1	$7A$	0 0 1 1
0 1 1 1 0	$2 \times 7A$	1			
0 1 0 0 1	$9A$	0	1 0 0 1	$9A$	0 1 0 0
0 1 0 1 1	$11A$	0	1 0 1 1	$11A$	0 1 0 1
0 1 1 0 1	$13A$	0	1 1 0 1	$13A$	0 1 1 0
0 1 1 1 1	$15A$	0	1 1 1 1	$15A$	0 1 1 1
1 0 0 0 0	$8 \times 2A$	3	0 0 0 0	$2A$	1 0 0 0

combining input coding with OMS



combining input coding with OMS

AREA AND TIME COMPLEXITIES OF LUT-MULTIPLIERS FOR DIFFERENT WORD-LENGTHS.

Word Size	Multiplier Design	Wallace-Tree Addition				Ripple-Carry-Chain Addition			
		Area	Delay	ADP	EADP	Area	Delay	ADP	EADP
8-bit	APC-OMS Based	---	---	---	---	654.09	2.97	1942.7	21.4%
	IC-OMS Based	---	---	---	---	625.2	2.56	1600.4	---
16-bit	APC-OMS Based	2679.16	5.13	11118.5	9.6%	2679.16	5.90	13181.5	14.9%
	IC-OMS Based	2195.8	4.62	10144.7	---	2169.0	5.29	11474.1	---
32-bit	APC-OMS Based	9171.39	8.99	73096.0	21.3%	9171.39	10.34	85477.3	19.4%
	IC-OMS Based	7103.3	8.48	60235.8	---	7060.2	10.14	71590.8	---

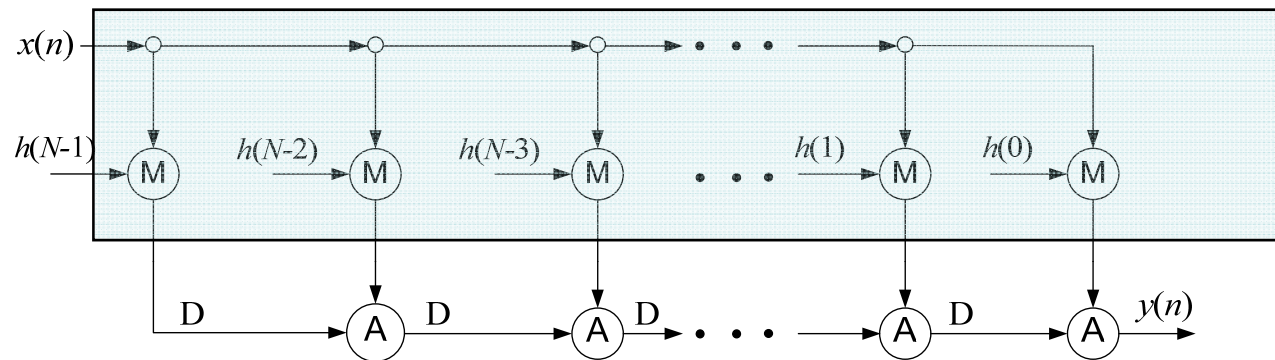
For word-size $L = 8$, the proposed LUT-based multiplier results in two LUT outputs and does not require Wallace-tree reduction. The pair of partial-products are added by a ripple-carry adder in this case. Areas are measured in μm^2 and delays are measured in nano-seconds. EADP stands for excess ADP over the corresponding proposed design.

DA-LUT vs LUT-multiplier-based designs

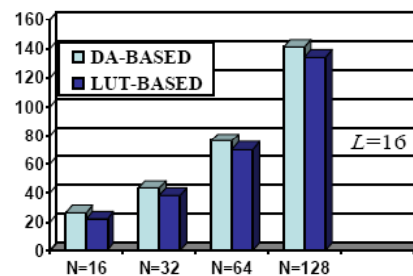
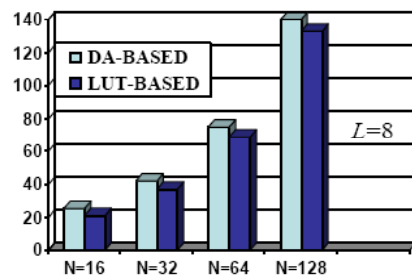
- each output of an N -tap FIR filter involves the computation of one N -point inner-product
- one sample could be processed by DA-approach in each cycle using L LUTs of (2^N) -words and $(L-1)$ adders
- LUT-multiplier-based approach to have the same throughput requires N LUTs of (2^L) -words each and $(N-1)$ adders.
- for $N=L$ and for the same throughput implementation, both the approaches have similar performances

LUT-multiplier-based FIR filter [21]

segmented memory core for N multiplications using OMS and APC [FIR 2010]



Latency chart of the DA-based and LUT-multiplier-based FIR filter.



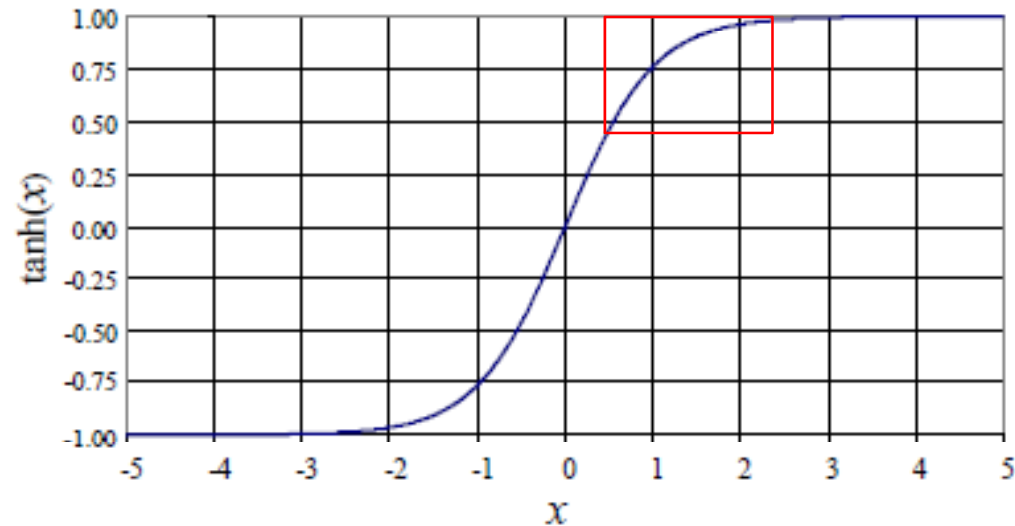
AREA COMPLEXITY OF DA-BASED AND LUT-MULTIPLIER-BASED FIR FILTERS FOR $N = 16$ AND $W = 8$.

FIR filter design	input operand width L	area (sq. μm)
DA-based	8	39029.56
	16	78895.96
conventional LUT-multiplier-based	8	40845.77
	16	82519.92
proposed LUT-multiplier-based	8	33880.80
	16	68558.92

15% less area than DA-based design for the same throughput rate.

LUT design for non-linear functions [24]

Example: sigmoid function



- For a range Δx of values of x one value of $\tanh(x)$ need to be stored.
- The range $\Delta x = 2\delta$, where $|\delta|$ is the maximum permissible value of error.

LUT design for non-linear functions

LUT FOR HYPERBOLIC TANGENT FUNCTION USING PROPOSED
COMPACTION TECHNIQUES

LUT Location	Limits of Positive Sub-domain		Stored Value	Maximum Error
	X1	X2		
1	0.390625	0.453125	0.4049	0.0196
2	0.453125	0.515625	0.4558	0.0186
3	0.515625	0.578125	0.5038	0.0175
4	0.578125	0.640625	0.5490	0.0164
5	0.640625	0.703125	0.5911	0.0152
6	0.703125	0.78125	0.6348	0.0186
7	0.78125	0.859375	0.6791	0.0168
8	0.859375	0.9375	0.7190	0.0151
9	0.9375	1.046875	0.7609	0.0197
10	1.046875	1.171875	0.8057	0.0191
11	1.171875	1.328125	0.8493	0.0195
12	1.328125	1.53125	0.8916	0.0190
13	1.53125	1.859375	0.9329	0.0197
14	1.859375	2.90625	0.9740	0.0200
15	2.90625	---	1	0.0041

Maximum allowable error = 0.02, and address-size = 8-bit. The input magnitude M satisfies the condition $I1 < I \leq I2$, where $I1$ and $I2$ are the binary equivalents of $X1$ and $X2$, respectively, in 8-bit representation.

conclusions

- memory-technology is growing quite fast and efficient memories for different applications are emerging over the years
- memory elements can be embedded directly into the structure of the microprocessor or integrated in the functional elements of dedicated processors.
- memory-based approach could be used for computation-intensive frequently used DSP tools.
- the DA-approach as well as the LUT-based multiplication could be used for memory-based implementation of digital filters

conclusions

- both the approaches could be used for the computation of discrete sinusoidal transforms by transforming the kernel matrix to cyclic convolution form.
- DA-approach could be used for reduced hardware realization
- when hardware is not a major constraint LUT-based multipliers could be used for a simple and straight-forward implementation of FIR filters
- a new approach to reduction of LUT-size for multiplication is proposed recently, where the memory-size is reduced significantly
- LUT could be designed for efficient evaluation of non-linear functions, like sinusoidal and hyperbolic functions, logarithms and multiple precision arithmetic.

references

- [1] K. Itoh, S. Kimura, and T. Sakata, “VLSI memory technology: Current status and future trends,” in *Proc. 25th European Solid-State Circuits Conference*, Sept. 1999, pp. 3–10.
- [2] B. Prince, “Trends in scaled and nanotechnology memories,” in *Proc. IEEE 2004 Conference on Custom Integrated Circuits*, Nov. 2005.
- [3] R. Barth, “ITRS commodity memory roadmap,” in *Proc. International Workshop on Memory Technology, Design and Testing*, July 2003 pp. 61-63.
- [4] Kinam Kim, “Memory Technologies for Mobile Era,” in *Proc. Asian Solid-State Circuits Conference*, Nov. 2005, pp. 7-11.
- [5] International Technology Roadmap for Semiconductors. [Online]. Available: <http://public.itrs.net/>
- [6] S. Lai, “Non-volatile memory technologies: The quest for ever lower cost”, in *Proc. IEEE International on Electron Devices Meeting*, Dec. 2008 pp.1 - 6

references

- [7] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. Mckenzie, “Computational RAM: implementing processors in memory,” *IEEE Trans. Design & Test of Computers*, vol. 16, no. 1, pp. 32–41, Jan-Mar 1999.
- [8] M. Wang, K. Suzuki, A. Sakai, W.Dai, “Memory and logic integration for System-in-a-Package,” *Proc. 4th International Conference on ASIC*, Oct. 2001, pp.843 - 847 .
- [9] T. Furuyama, “Trends and challenges of large scale embedded memories,” in *Proc. IEEE 2004 Conference on Custom Integrated Circuits*, Oct. 2004, pp. 449-456.
- [10] C. Trigas, S. Doll, J. Kruecken, “MRAM and Microprocessor System-In-Package: Technology Stepping Stone to Advanced Embedded Devices,” *IEEE Custom Integrated Circuits Conf*, 2004, pp.71-79.
- [11] US Patent 5790839 - System integration of DRAM macros and logic cores in a single chip architecture

references

- [12] S. A. White, “Applications of the distributed arithmetic to digital signal processing: A tutorial review,” *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 5–19, July 1989.
- [13] H.-R. Lee, C.-W. Jen, and C.-M. Liu, “On the design automation of the memory-based VLSI architectures for FIR filters,” *IEEE Trans. Consumer Electronics*, vol. 39, no. 3, pp. 619–629, Aug. 1993.
- [14] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, D. V. Anderson, LMS Adaptive Filters Using Distributed Arithmetic for High Throughput, *IEEE Trans Circuits & Systems-I*, vol. 52, no. 7, pp. 1327- 1337, July 2005.
- [15] P. K. Meher, S. Chandrasekaran, and A. Amira, ‘FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic,’ *IEEE Trans Signal Processing*, pp. 3009-3017, July 2008.
- [16] P. K. Meher, ‘Hardware-Efficient Systolization of DA-based Calculation of Finite Digital Convolution,’ *IEEE Trans Circuits & Systems-II*, pp.707-711, Aug 2006.

references

- [17] J.-I. Guo, C.-M. Liu, and C.-W. Jen, “The efficient memory-based VLSI array design for DFT and DCT,” *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, vol. 39, no. 10, pp. 723–733, Oct. 1992.
- [18] H.-C. Chen, J.-I. Guo, T.-S. Chang, and C.-W. Jen, “A memory-efficient realization of cyclic convolution and its application to discrete cosine transform,” *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 15, no. 3, pp. 445–453, Mar. 2005.
- [19] D. F. Chiper, M. N. S. Swamy, M. O. Ahmad, and T. Stouraitis, “Systolic algorithms and a memory-based design approach for a unified architecture for the computation of DCT/DST/IDCT/IDST,” *IEEE Trans. Circuits Syst.-I: Regular Papers*, vol. 52, no. 6, pp. 1125–1137, Jun. 2005.
- [20] P. K. Meher, J. C. Patra, and M. N. S. Swamy, “High-throughput memory-based architecture for DHT using a new convolutional formulation,” *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 54, no. 7, pp. 606–610, July 2007.

references

- [21] P. K. Meher, 'New Approach to Look-up-Table Design and Memory-Based Realization of FIR Digital Filter,' *IEEE Trans on Circuits & Systems-I*, pp.592-603, March 2010.
- [22] P. K. Meher, 'LUT Optimization for Memory-Based Computation,' *IEEE Trans on Circuits & Systems-II*, pp.285-289, April 2010.
- [23] P. K. Meher, 'Novel Input Coding Technique for High-Precision LUT-Based Multiplication for DSP Applications' *The 18th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC 2010)*, pp. 201-206, Madrid, Spain, September 2010.
- [24] P. K. Meher, 'An Optimized Lookup-Table for the Evaluation of Sigmoid Function for Artificial Neural Networks' *The 18th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC 2010)*, pp. 91-95, Madrid, Spain, September 2010.