Chapter 13: Bit Level Arithmetic Architectures

Keshab K. Parhi

- A W-bit fixed point two's complement number A is represented as :

$$A=a_{w-1}.a_{w-2}...a_1.a_0$$

where the bits ai, $0 \leq i \leq W-1$, are either 0 or 1, and the msb is the sign bit.

- The value of this number is in the range of $[-1, 1 - 2^{-W+1}]$ and is given by :

$$A = - a_{w-1} + \Sigma a_{w-1-i}2^{-i}$$

- For bit-serial implementations, constant word length multipliers are considered. For a W×W bit multiplication the W most-significant bits of the (2W-1)-bit product are retained.

- Parallel Multipliers :

$$A = a_{w-1}.a_{w-2}\ldots a_1.a_0 = -a_{w-1} + \sum_{i=1}^{W-1} a_{w-1-i}2^{-i}$$

$$B = b_{w-1}.b_{w-2}\ldots b_1.b_0 = -b_{w-1} + \sum_{i=1}^{W-1} b_{w-1-i}2^{-i}$$

Their product is given by :

$$P = -p_{2W-2} + \sum_{i=1}^{2W-2} p_{2W-2-i}2^{-i}$$

In constant word length multiplication, W – 1 lower order bits in the product P are ignored and the Product is denoted as $X \Leftarrow P = A \times B$, where

$$X = -x_{w-1} + \sum_{i=1}^{W-1} x_{w-1-i}2^{-i}$$

- Parallel Multiplication with Sign Extension :

  Using Horner's rule, multiplication of A and B can be written as

$$P = A \times (-b_{W-1} + \Sigma\, b_{W-1-i} 2^{-i})$$

$$= -A.\, b_{W-1} + [A.\, b_{W-2} + [A.\, b_{W-3} + [\ldots +$$

$$[A.\ b_1 + A\, b_0\, 2^{-1}]\, 2^{-1}]\ldots]2^{-1}]\, 2^{-1}$$

  where 2^{-1} denotes scaling operation.

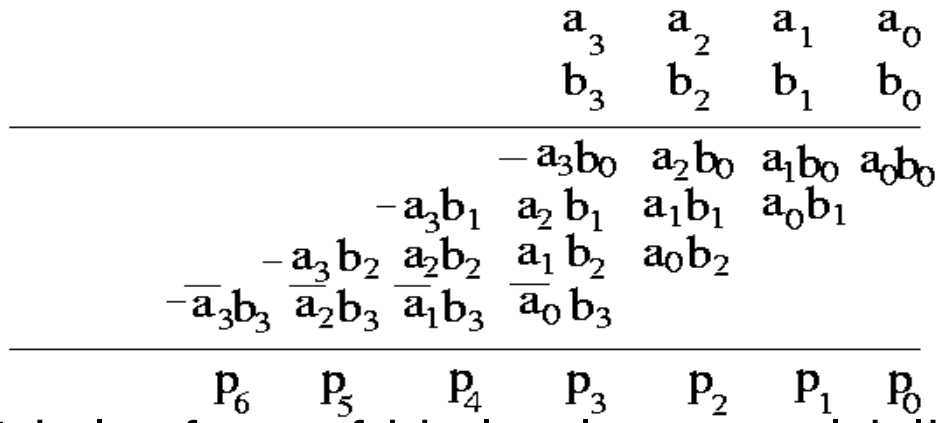
- In 2's complement, negating a number is equivalent to taking its 1's complement and adding 1 to lsb as shown below:

$$-A = a_{w-1} - \sum_{i=1}^{W-1} a_{w-1-i} 2^{-i}$$

$$= a_{w-1} + \sum_{i=1}^{W-1} (1 - a_{w-1-i}) 2^{-i} - \sum_{i=1}^{W-1} 2^{-i}$$

$$= a_{w-1} + \sum_{i=1}^{W-1} (1 - a_{w-1-i}) 2^{-i} - 1 + 2^{-W+1}$$

$$= -(1 - a_{w-1}) + \sum_{i=1}^{W-1} (1 - a_{w-1-i}) 2^{-i} + 2^{-W+1}$$



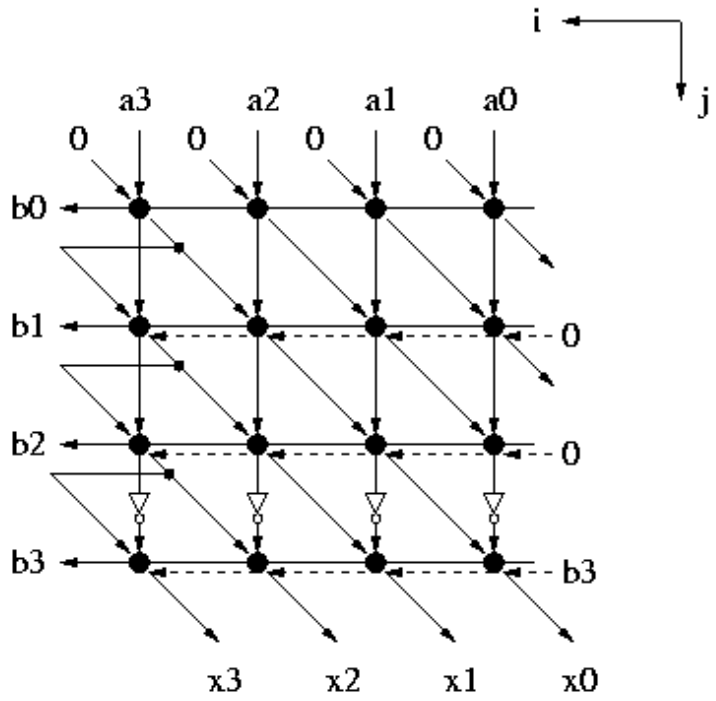
Tabular form of bit-level array multiplication

- The additions cannot be carried out directly due to terms having negative weight. Sign extension is used to solve this problem. For example,

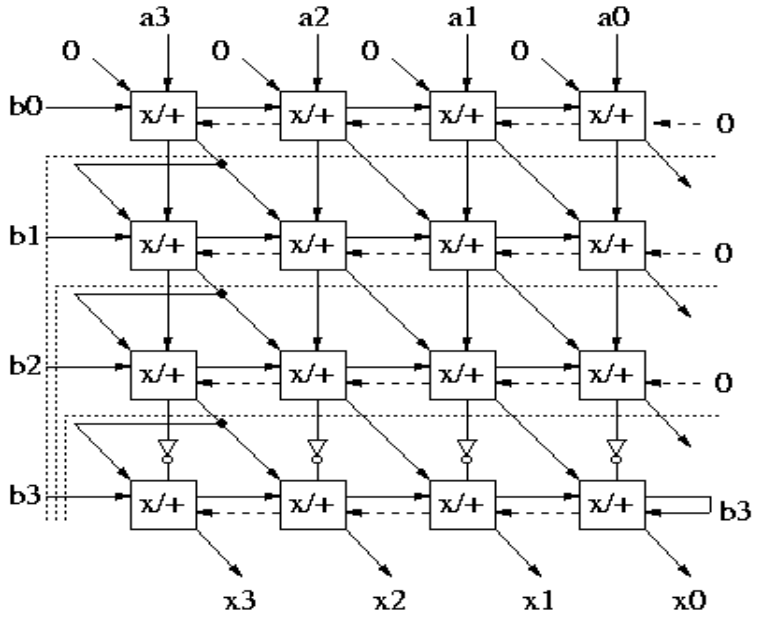
$$A = a_3 + a_2 2^{-1} + a_1 2^{-2} + a_0 2^{-3}$$
$$= -a_3 2 + a_3 + a_2 2^{-1} + a_1 2^{-2} + a_0 2^{-3}$$
$$= -a_3 2^2 + a_3 2 + a_3 + a_2 2^{-1} + a_1 2^{-2} + a_0 2^{-3}$$

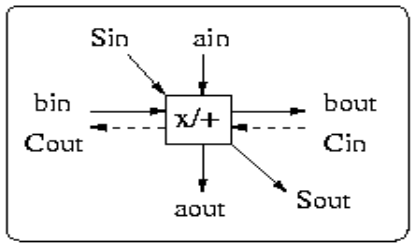
describes sign extension of A by 1 and 2 bits.

- Parallel Carry-Ripple Array Multipliers :



Bit level dependence Graph



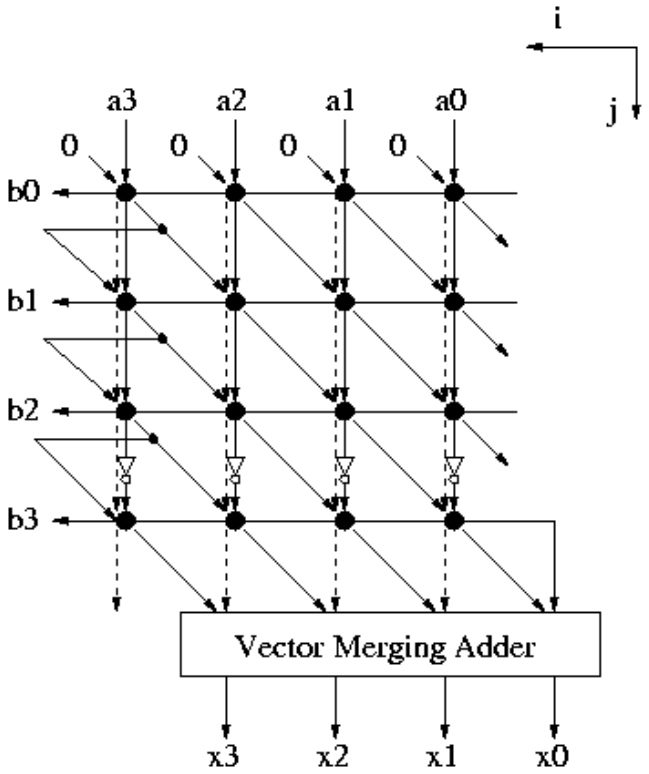


→ Broadcast signals:
bout=bin; aout=ain

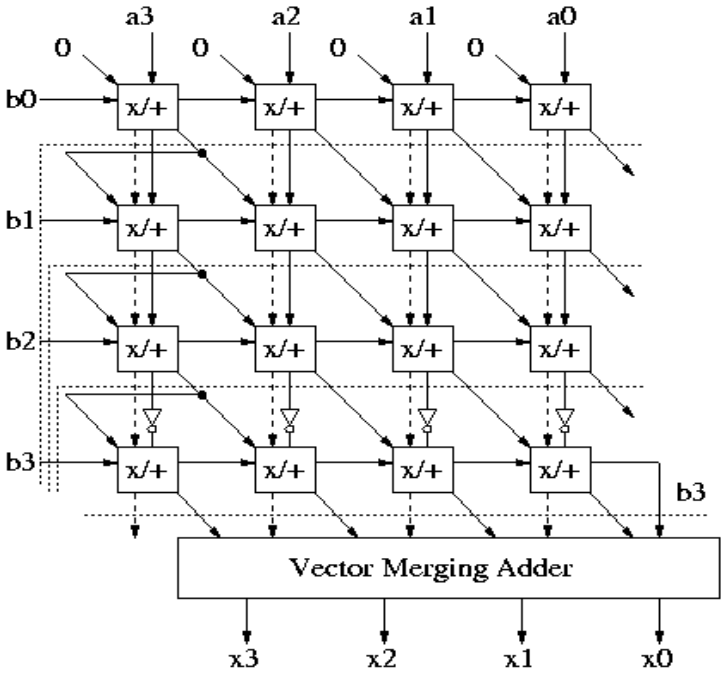
→ Single-bit Full-Adder:

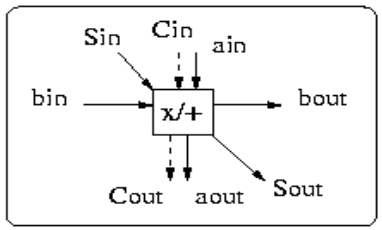
$2\,Cout + Sout = ain*bin + Sin + Cin$

Parallel Carry Ripple Multiplier



DG for 4×4-bit carry save array multiplication





→ Broadcast signals:
bout=bin; aout=ain

→ Single-bit Full-Adder:
2 Cout + Sout = ain*bin + Sin + Cin

Parallel carry-save array multiplier

- Baugh-Wooley Multipliers:
  - Handles the sign bits of the multiplicand and multiplier efficiently.

				a_3	a_2	a_1	a_0
				b_3	b_2	b_1	b_0
				$\overline{a_3b_0}$	a_2b_0	a_1b_0	a_0b_0
			$\overline{a_3b_1}$	a_2b_1	a_1b_1	a_0b_1	
		$\overline{a_3b_2}$	a_2b_2	a_1b_2	a_0b_2		
	a_3b_3	a_2b_3	a_1b_3	a_0b_3			
	x_3	x_2	x_1	x_0			

Tabular form of bit-level Baugh-Wooley multiplication

- Parallel Multipliers with Modified Booth Recoding :
  - Reduces the number of partial products to accelerate the multiplication process.
  - The algorithm is based on the fact that fewer partial products need to be generated for groups of consecutive zeros and ones. For a group of "m" consecutive ones in the multiplier, i.e.,

    $\ldots 0\{11\ldots 1\}0\ldots = \ldots 1\{00\ldots \bar{0}\}0\ldots - \ldots 0\{00\ldots 1\}0\ldots$

    $= \ldots 1\{00\ldots 1\}0\ldots$

    instead of "m" partial products, only 2 partial products need to be generated is signed digit representation is used.
  - Hence, in this multiplication scheme, the multiplier bits are first recoded into signed-digit representation with fewer number of nonzero digits; the partial products are then generated using the recoded multiplier digits and accumulated.

b_{2i+1}	b_{2i}	b_{2i-1}	b'_i	Operation	Comments
0	0	0	0	+0	string of 0's
0	0	1	1	+A	end of 1's
0	1	0	1	+A	a single 1
0	1	1	2	+2A	end of 1's
1	0	0	-2	-2A	beginning of 1's
1	0	1	-1	-A	A single 0
1	1	0	-1	-A	beginning of 1's
1	1	1	0	-0	string of 1's

Radix-4 Modified Booth Recoding Algorithm

Recoding operation can be described as:

$$b'_i = -2b_{2i+1} + b_{2i} + b_{2i-1}$$

Interleaved Floor-Plan and Bit-Plane-Based Digital Filters

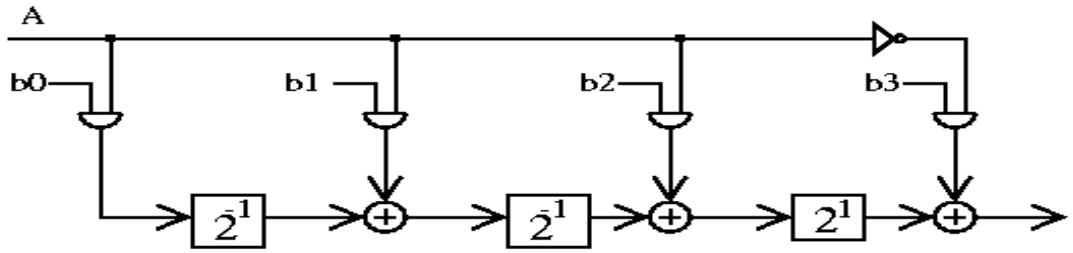
- A constant coefficient FIR filter is given by:

$$y(n) = x(n) + f \bullet x(n-1) + g \bullet x(n-2)$$

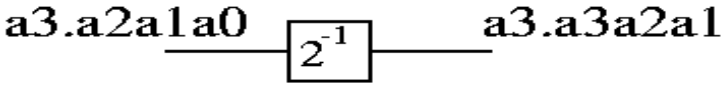
  where, x(n) is the input signal, and f and g are filter coefficients.
- The main idea behind the interleaved approach is to perform the computation and accumulation of partial products associated with f and g simultaneously thus increasing the speed.
- This increases the accuracy as truncation is done at the final step.
- If the coefficients are interleaved in such a way that their partial products are computed in different rows, the resulting architecture is called bit-plane architecture.

Bit-Serial Multipliers

- Lyon's Bit-Serial Multiplier using Horner's Rule :

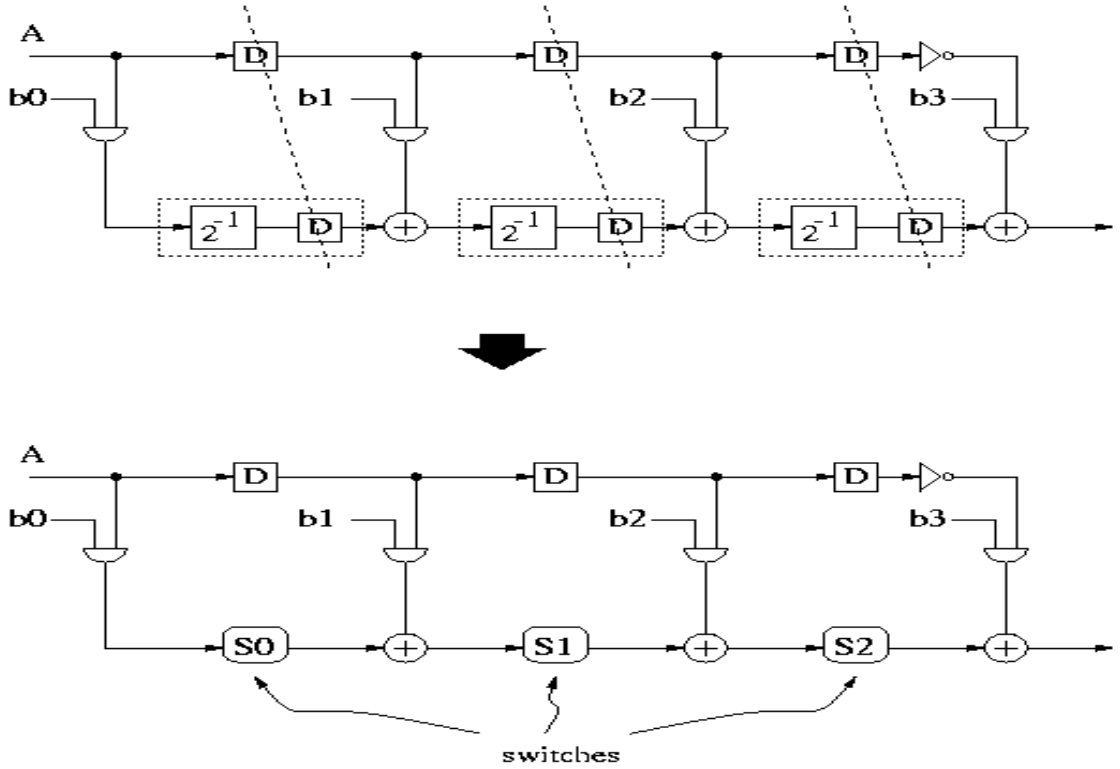


(a)

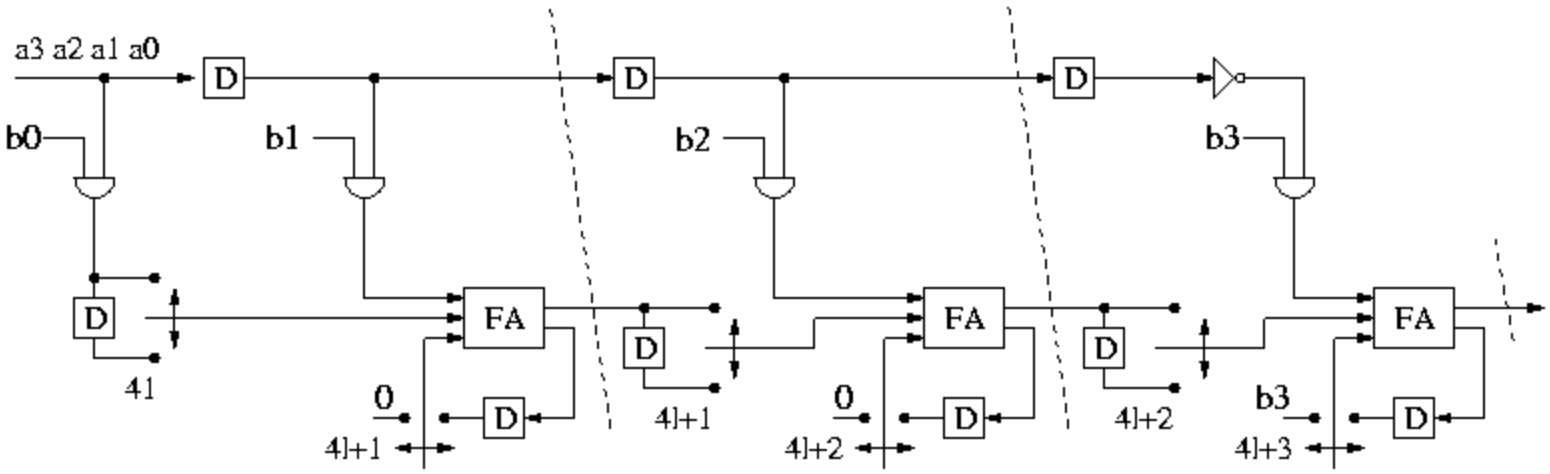


(b)

- For the scaling operator, the first output bit a_1 should be generated at the same time instance when the first input a_1 enters the operator. Since input a_1 has not entered the system yet, the scaling operator is non-causal and cannot be implemented in hardware.



Derivation of implementable bit-serial 2's complement multiplier



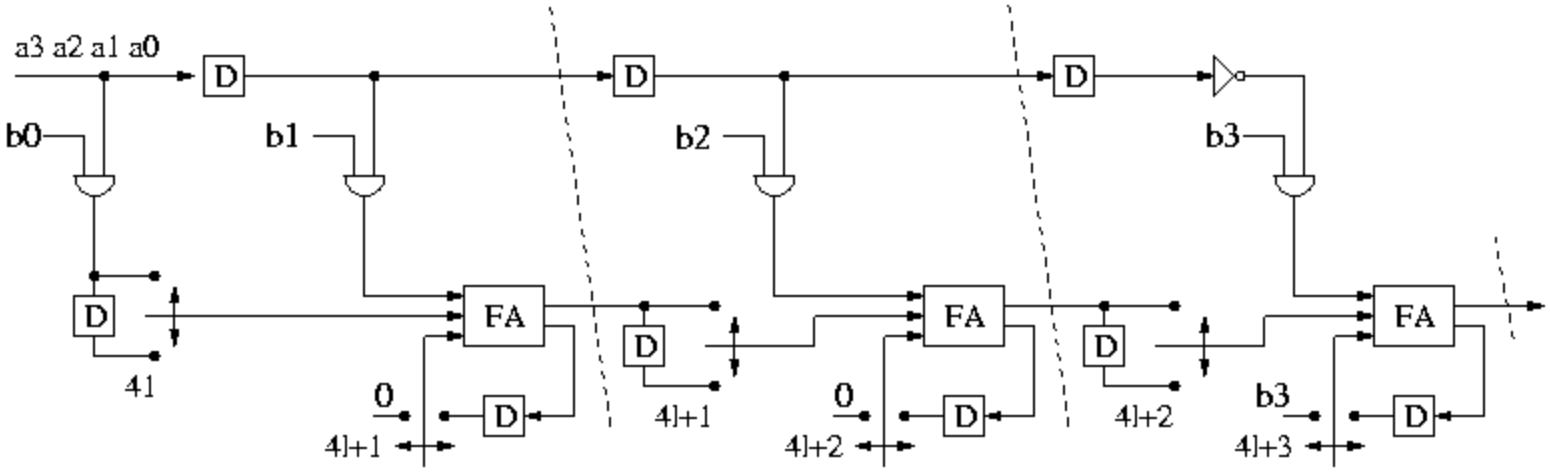
Lyon's bit-serial 2's complement multiplier

Design of Bit-Serial Multipliers Using Systolic Mappings

•Design of Lyon's bit-serial multiplier by systolic mapping Using DG of ripple carry multiplication.

e	$p^T e$	$s^T e$
a(0,1)	1	1
b(1,0)	0	1
carry(1,0)	0	1

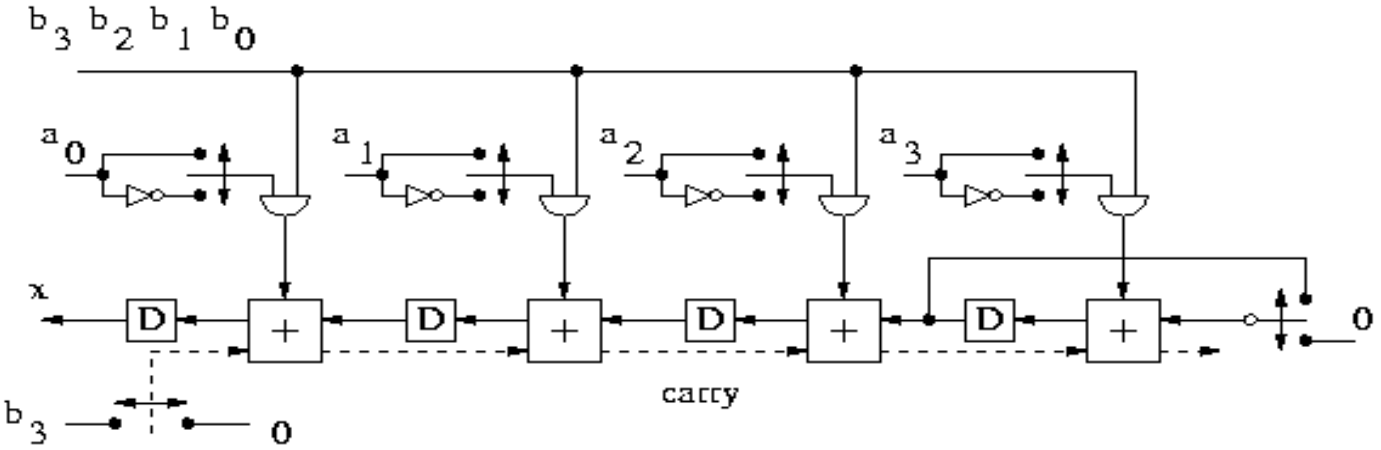
Here, $d^T = [1\ 0]$, $s^T = [1\ 1]$ and $p^T = [0\ 1]$



•Design of bit-serial multiplier by systolic mapping using DG of ripple carry multiplication and the following :

$d^T = [0\ 1]$, $s^T = [0\ 1]$ and $p^T = [1\ 0]$

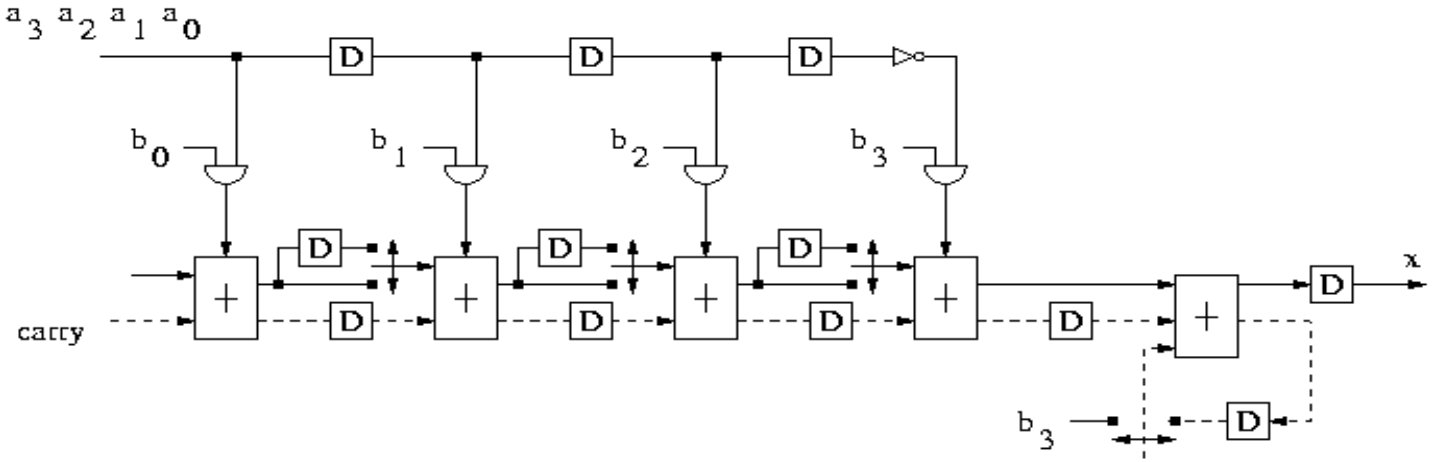
e	$p^T e$	$s^T e$
a(0,1)	0	1
b(1,0)	1	0
carry(1,0)	1	0
x(-1,1)	-1	1

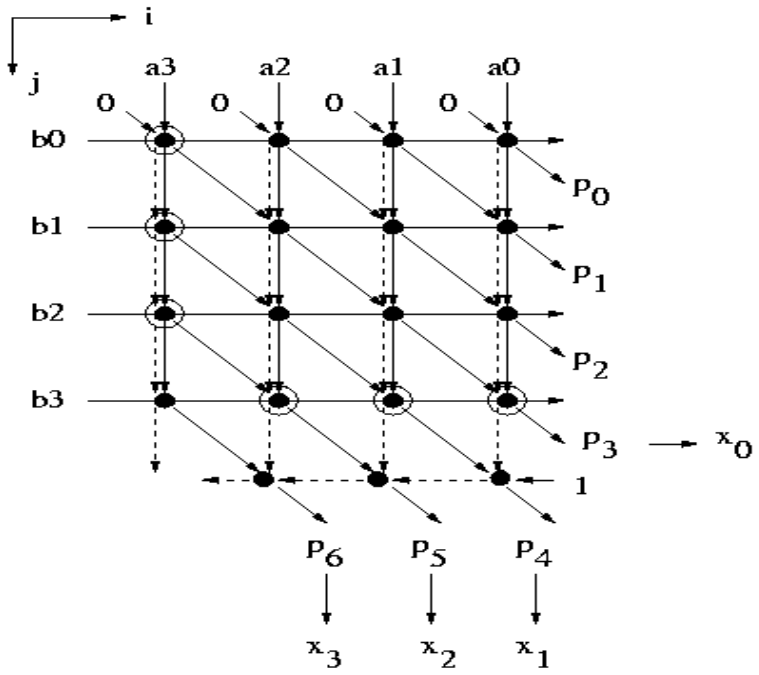


•Design of bit-serial multiplier by systolic mapping
using DG for carry-save array multiplication and the following :

$d^T = [1\ 0]$, $s^T = [1\ 1]$ and $p^T = [0\ 1]$

e	p^Te	s^Te
a(0,1)	1	1
b(1,0)	1	1
carry(1,0)	0	1
x(-1,1)	1	0





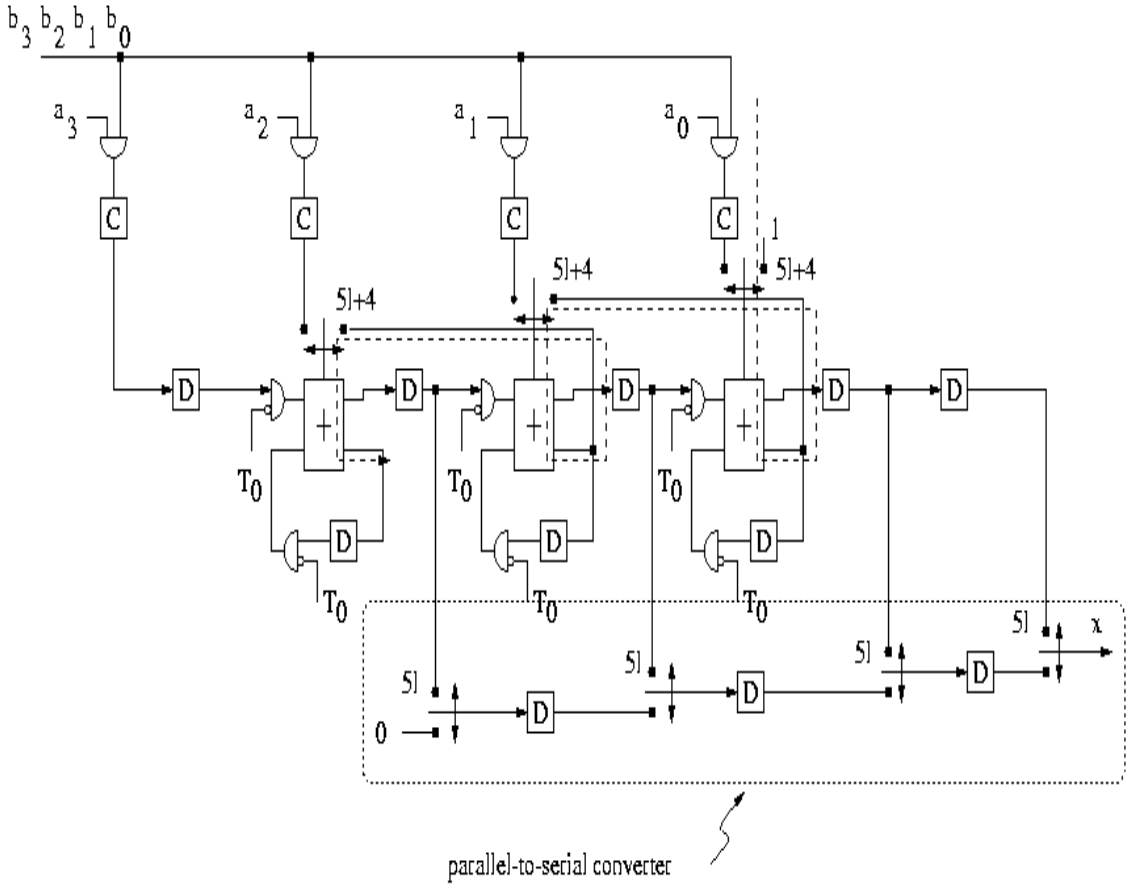
Dependence graph for carry save Baugh-Wooley multiplication with carry ripple vector merging

•Design of bit-serial Baugh-Wooley multiplier by systolic mapping using DG for Baugh-Wooley multiplication and the following :

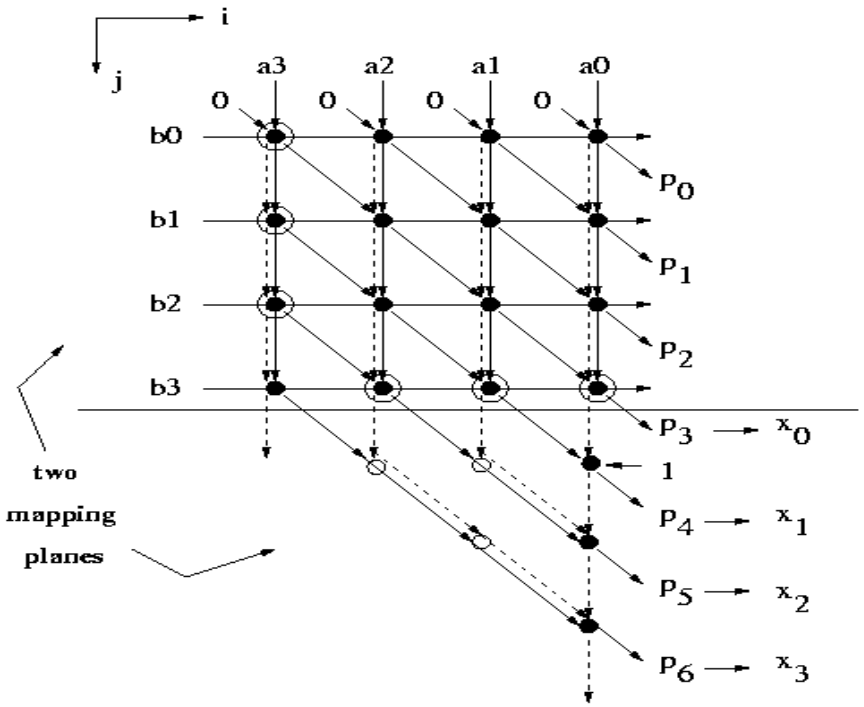
$d^T = [0\ 1]$, $s^T = [0\ 1]$ and $p^T = [1\ 0]$

e	$p^T e$	$s^T e$
a(0,1)	0	1
carry(0,1)	0	1
b(1,0)	1	0
x(1,1)	1	1
carry-vm(-1,0)	-1	0

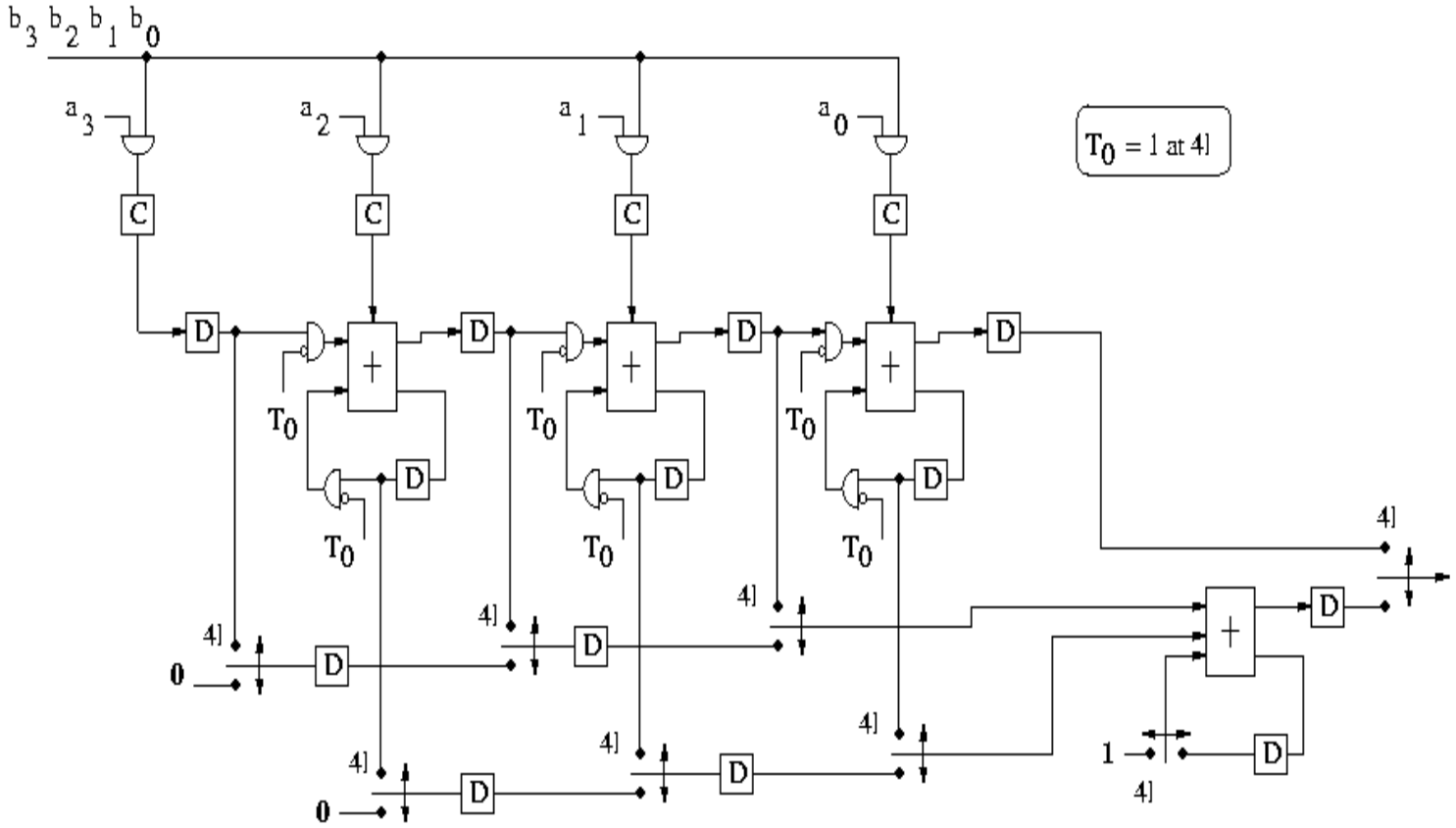
Here, carry-vm denotes the carry outputs in the vector merging portion.



Bit-Serial Baugh-Wooley Multiplier

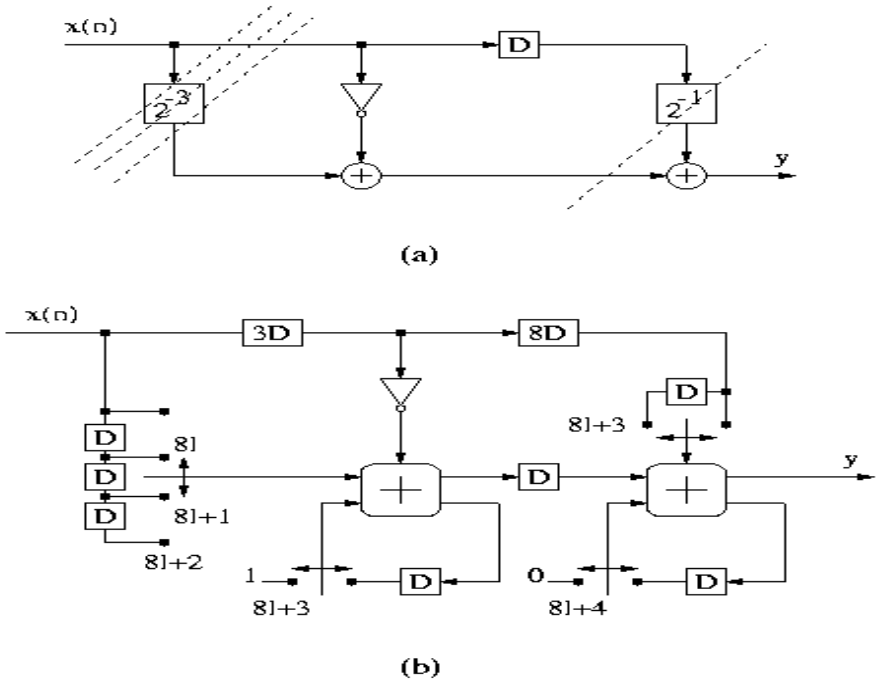


DG bit-serial Baugh-Wooley multiplier with carry-save array and vector merging portion treated as two separate planes



Bit-serial Baugh-Wooley multiplier
using the DG having two separate planes
for carry-save array and the vector merging portion

Bit-Serial FIR Filter



Bit-level pipelined bit-serial FIR filter, $y(n) = (-7/8)x(n) + (1/2)x(n-1)$, where constant coefficient multiplications are implemented as shifts and adds as $y(n) = -x(n) + x(n)2^{-3} + x(n-1)2^{-1}$.
(a)Filter architecture with scaling operators;
(b) feasible bit-level pipelined architecture

Bit-Serial IIR Filter

- Consider implementation of the IIR filter

$$Y(n) = (-7/8)y(n-1) + (1/2)y(n-2) + x(n)$$

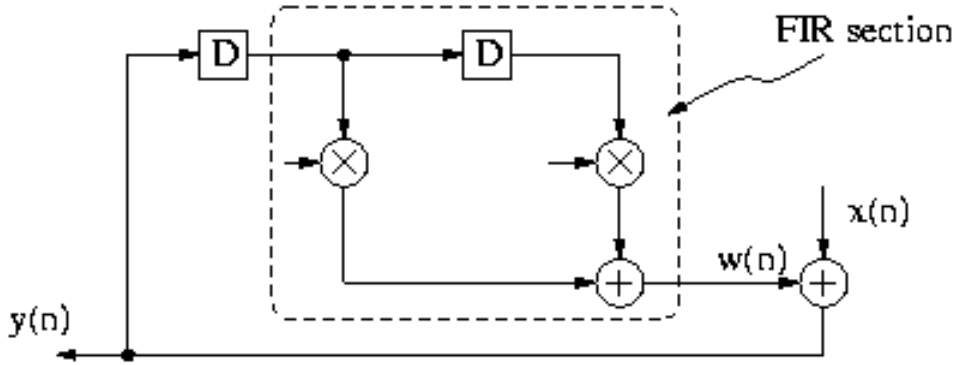
where, signal word-length is assumed to be 8.

- The filter equation can be re-written as follows:

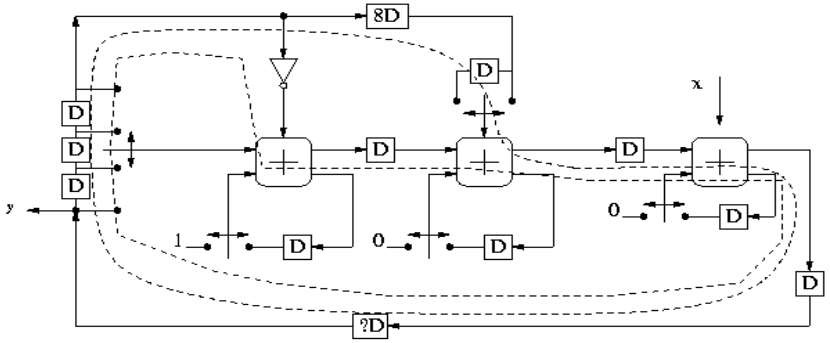
$$w(n) = (-7/8)y(n-1) + (1/2)y(n-2)$$

$$Y(n) = w(n) + x(n)$$

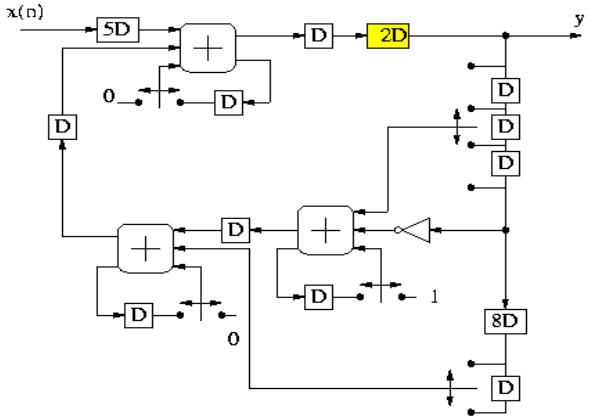
which can be implemented as an FIR section from y(n-1) with an addition and a feedback loop as shown below:



- Steps for deriving a bit-serial IIR filter architecture:
  - A bit-level pipelined bit-serial implementation of the FIR section needs to be derived.
  - The input signal x(n) is added to the output of the bit-serial FIR section w(n).
  - The resulting signal y(n) is connected to the signal y(n-1).
  - The number of delay elements in the edge marked ?D needs to be determined.(see figure in next page)
- For, systems containing loop, the total number of delay elements in the loops should be consistent with the original SFG, in order to maintain synchronization and correct functionality.
- **Loop delay synchronization** involves matching the number of word-level loop delay elements and that in the bit-serial architecture. The number of bit-level delay elements in the bit-serial loops should be $W \times N_D$, where W is signal word-length and N_D denotes the number of delay elements in the word-level SFG.



(a)



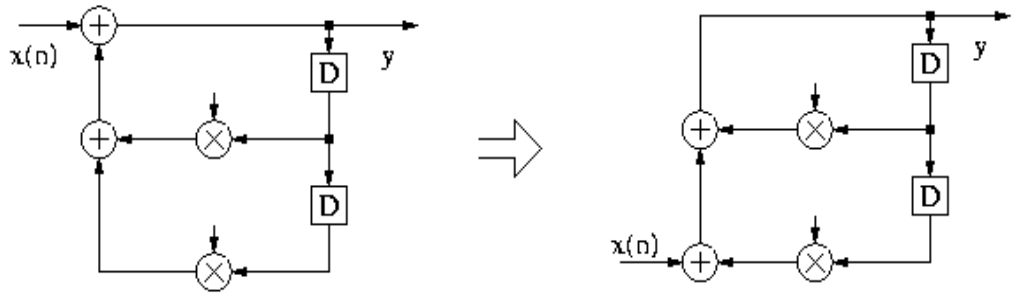
(b)

- Bit-level pipelined bit-serial architecture, without synchronization delay elements. (b) Bit-serial IIR filter. Note that this implementation requires a minimum feasible word-length of 6.

Note:

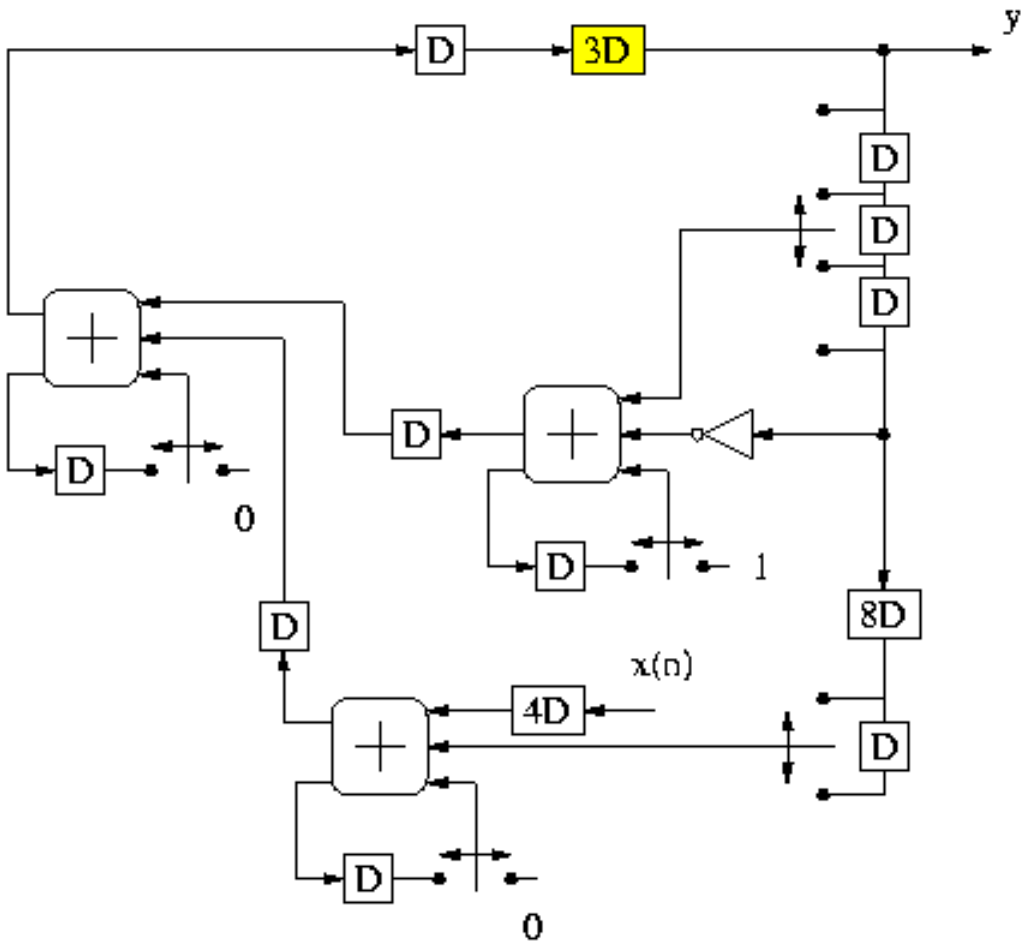
- To compute the total number of delays in the bit-level architecture, the paths with the **largest number of delay elements** in the switching elements should be counted.
- Input synchronizing delays (also referred as **shimming** delays or **skewing** delays).
- It is also possible that the loops in the intermediate bit-level pipelined architecture may contain more than $W \times N_D$ number of bit-level delay elements, in which case the word-length needs to be increased.
- The architecture without the two loop synchronizing delays can function correctly with a signal word-length of 6, which is the minimum word-length for the bit-level pipelined bit-serial architecture.

- **Associativity transformation** :



(a) (b)

Loop iteration bound of IIR filter can be reduced from one-multiply-two-add to one-multiply-add by associative transformation



Bit-serial IIR filter after associative transformation. This implementation requires a minimum feasible word-length of 5.

Canonic Signed Digit Arithmetic

- Encoding a binary number such that it contains the fewest number of non-zero bits is called **canonic signed digit**(CSD).
- The following are the properties of CSD numbers:
  - No 2 consecutive bits in a CSD number are non-zero.
  - The CSD representation of a number contains the minimum possible number of non-zero bits, thus the name canonic.
  - The CSD representation of a number is unique.
  - CSD numbers cover the range (-4/3,4/3), out of which the values in the range [-1,1) are of greatest interest.
  - Among the W-bit CSD numbers in the range [-1,1), the average number of non-zero bits is $W/3 + 1/9 + O(2^{-W})$. Hence, on average, CSD numbers contains about 33% fewer non-zero bits than two's complement numbers.

- Conversion of W-bit number to CSD format:
  - $A = a'_{W-1}.\ a'_{W-2} \ldots a'_1.\ a'_0$ = 2's complement number
  - Its CSD representation is $a_{W-1}.\ a_{W-2} \ldots a_1.\ a_0$
- Algorithm to obtain CSD representation:
  - $a'_{-1} = 0;$
  - $\gamma_{-1} = 0;$
  - $a'_W = a'_{W-1};$
  - for (i = 0 to W-1)

    {

    $\theta_i = a'_i \oplus a'_{i-1};$

    $\gamma_i = \overline{\gamma}_{i-1}\theta_i;$

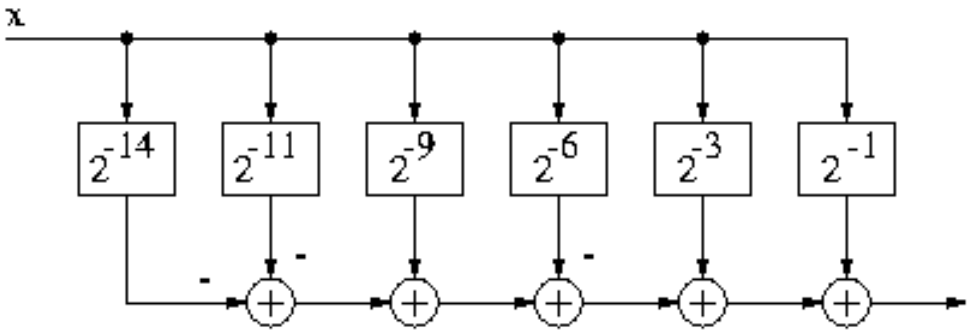
    $a_i = (1 - 2a'_{i+1})\gamma_i;$

    }

i	W	W-1								0	-1
a'_i	1	1	0	1	1	1	0	0	1	1	
θ_i		1	1	0	0	1	0	1	0	1	
γ_i		0	1	0	0	1	0	1	0	1	
$1 - 2a'_{i+1}$		-1	-1	1	-1	-1	-1	1	1	-1	
a_i		0	-1	0	0	-1	0	1	0	-1	

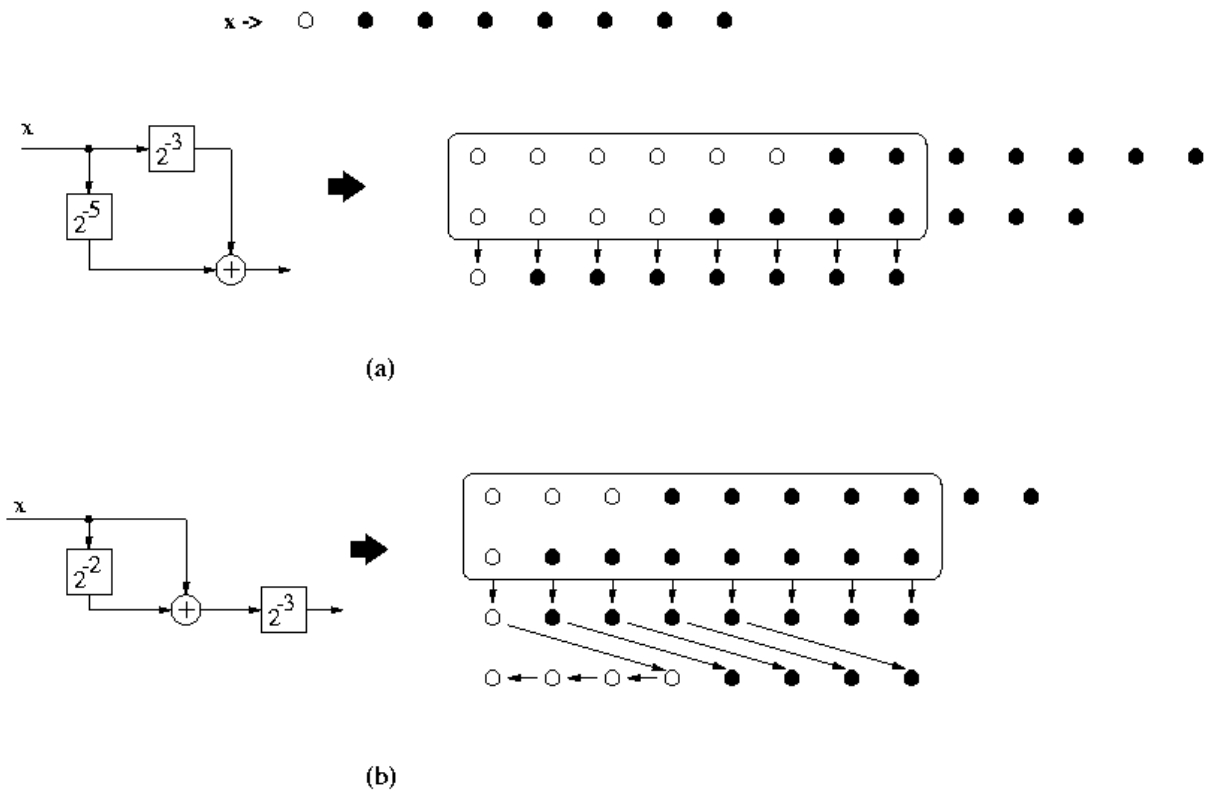
Table showing the computation of the CSD representation for the number 1.01110011.

CSD Multiplication

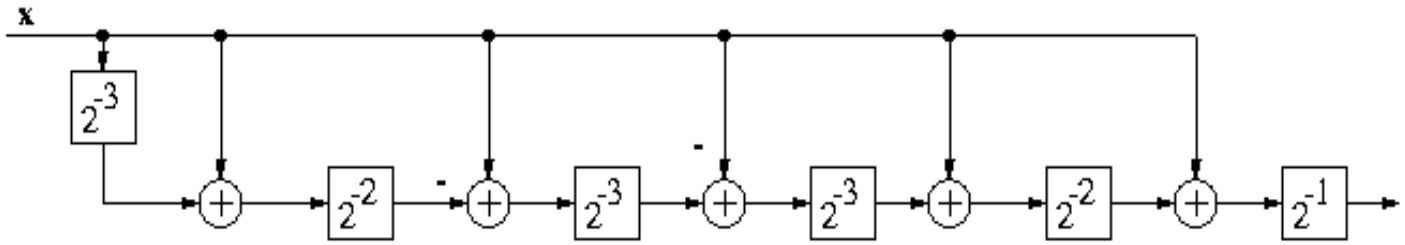


A CSD multiplier using linear arrangement of adders to compute $x \times 0.10100\bar{1}00\bar{1}0\bar{1}00\bar{1}$

- Horner's rule for precision improvement : This involves delaying the scaling operations common to the 2 partial products thus increasing accuracy.
- For example, $x \bullet 2^{-5} + x \bullet 2^{-3}$ can be implemented as $(x \bullet 2^{-2} + x)2^{-3}$ to increase the accuracy.

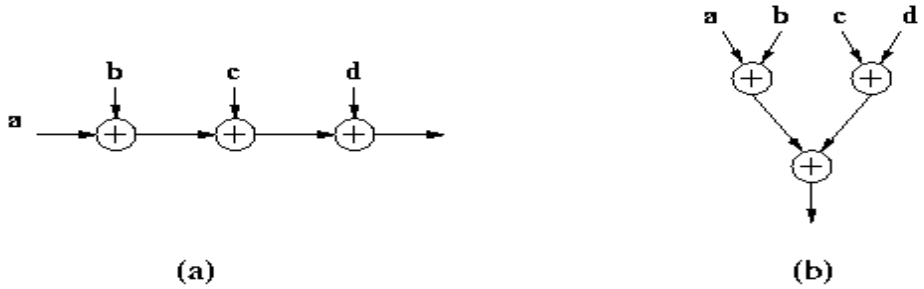


Using Horner's rule for partial product accumulation to reduce the truncation error.

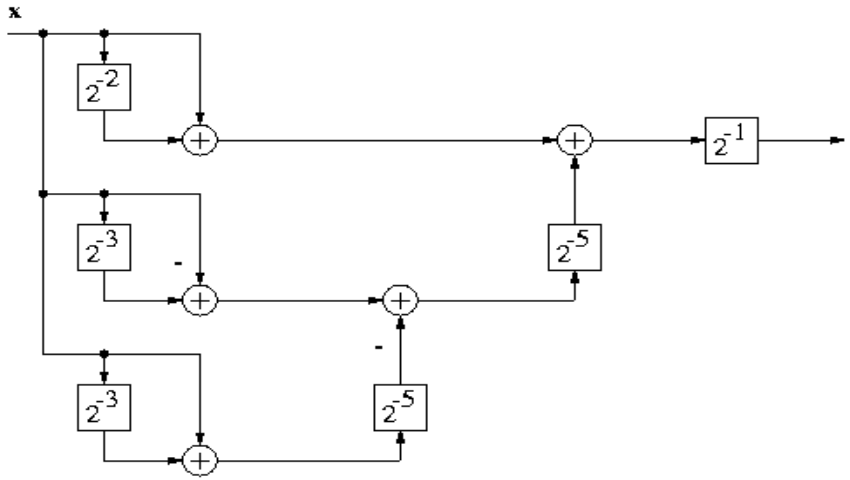


Rearrangement of the CSD multiplication of $x \times 0.1010010010101001$ using Horner's rule for partial product accumulation to reduce the truncation error.

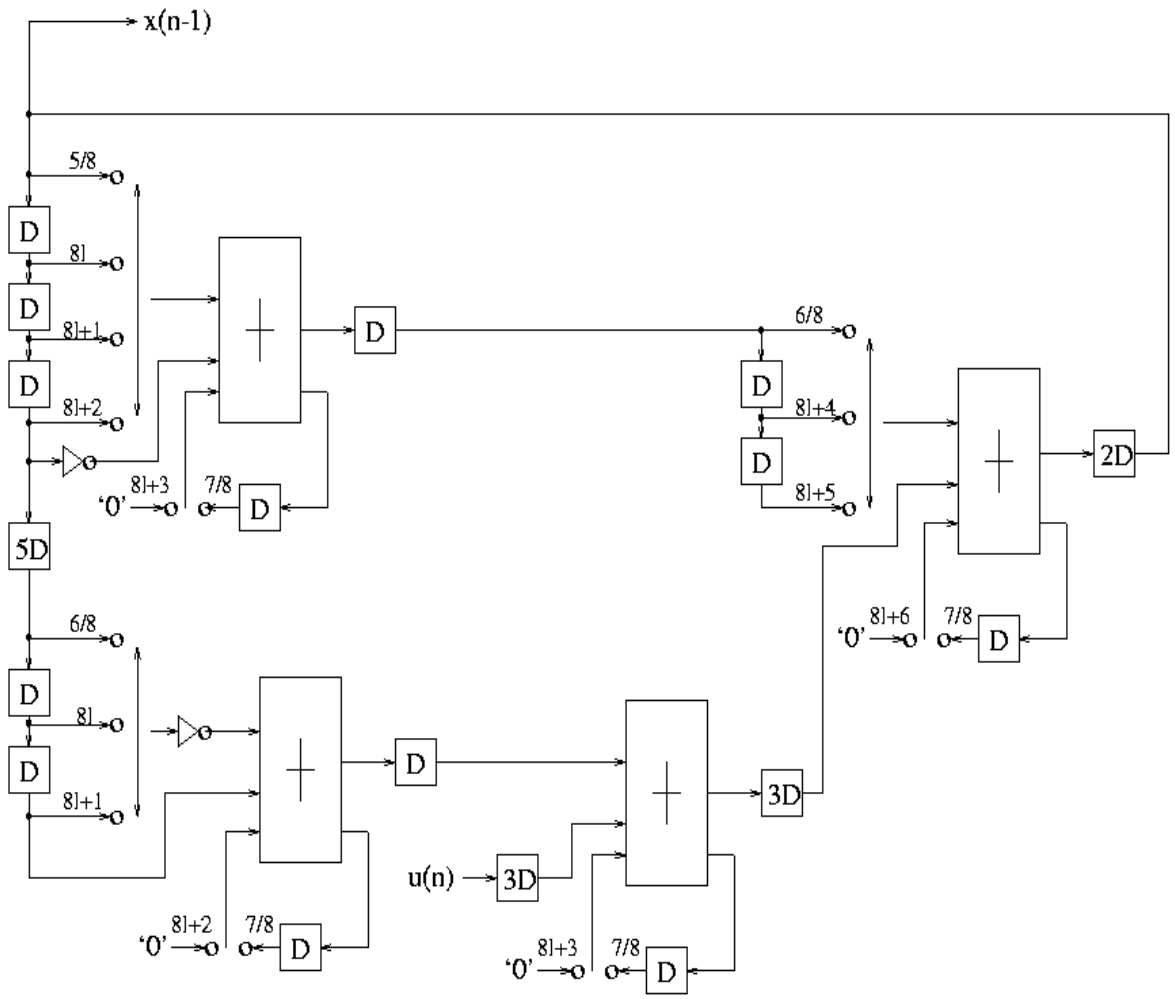
Use of Tree-Height Reduction for Latency Reduction



(a) linear arrangement (b) tree arrangement



Combination of tree-type arrangement and Horner's rule for the accumulation of partial products in CSD multiplication



Bit serial architecture using CSD. In this case the coefficients -7/32 = -1/4 + 1/32 is encoded as $0.0\bar{1}001$ and ¾ = 1 – ¼ is encoded as $1.0\bar{1}$.