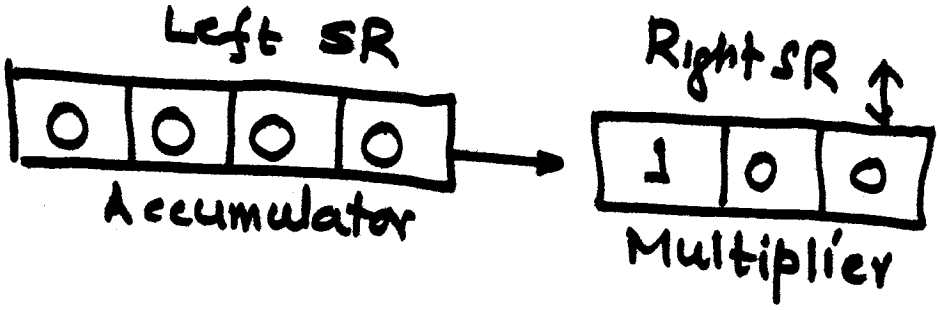
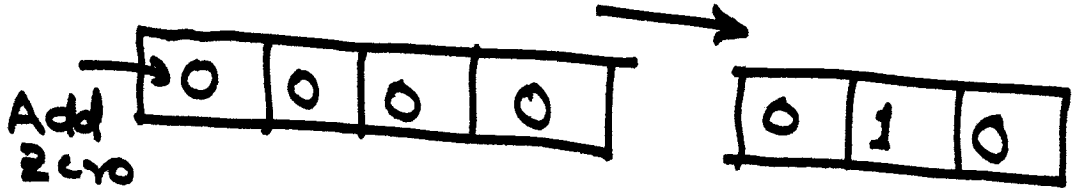
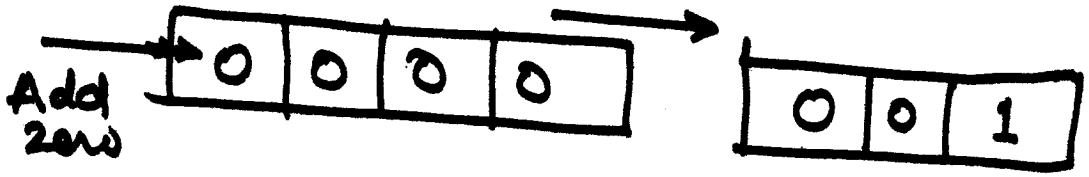


Mode = 0
No Addition is Desired
"NO ADD" &
Shift 1 bit on Right

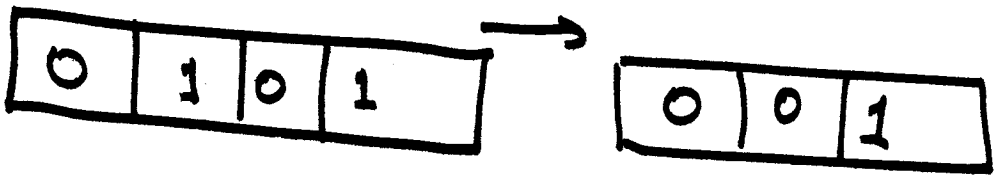


Mode = 0 NoAdd &
1 bit On Right



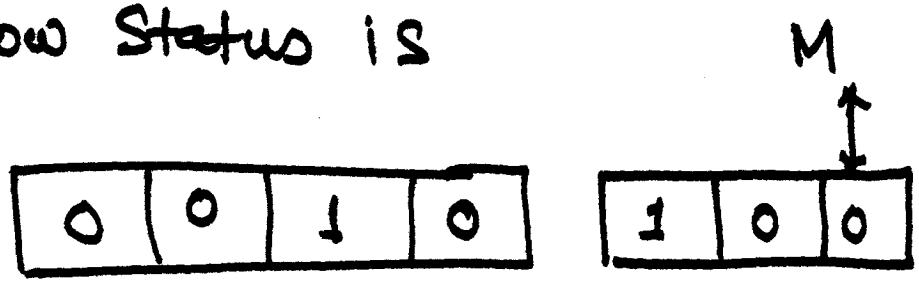
Mode=1 'Add operation of Multiplicand content with Left Shift Register

0 0 0 0
Multiplicand 0 1 0 1
0 1 0 1 → Transfer to Left Registrar &



As status of Registers
Now Shift bits to Right

Now Status is



Now M = 0

No 'Add', but Shift Right, Boot Operation. is not needed As 3 bit operation is completed

Final Status

∴ Final Number is 10100 <u>which is Ouput</u>

Decimal Numbers
- Multiplicand 101 → 5
- Multiplier 100 → 4
- Multiplication Output = 5 x 4 = 20
  = 10100

ALGORITHM of Baugh-Wooley

MULTIPLIER

X and Y are the Mutiplicand and Multiplier 2's Complemented Numbers. Then

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

$$Y = -y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} y_j 2^j$$

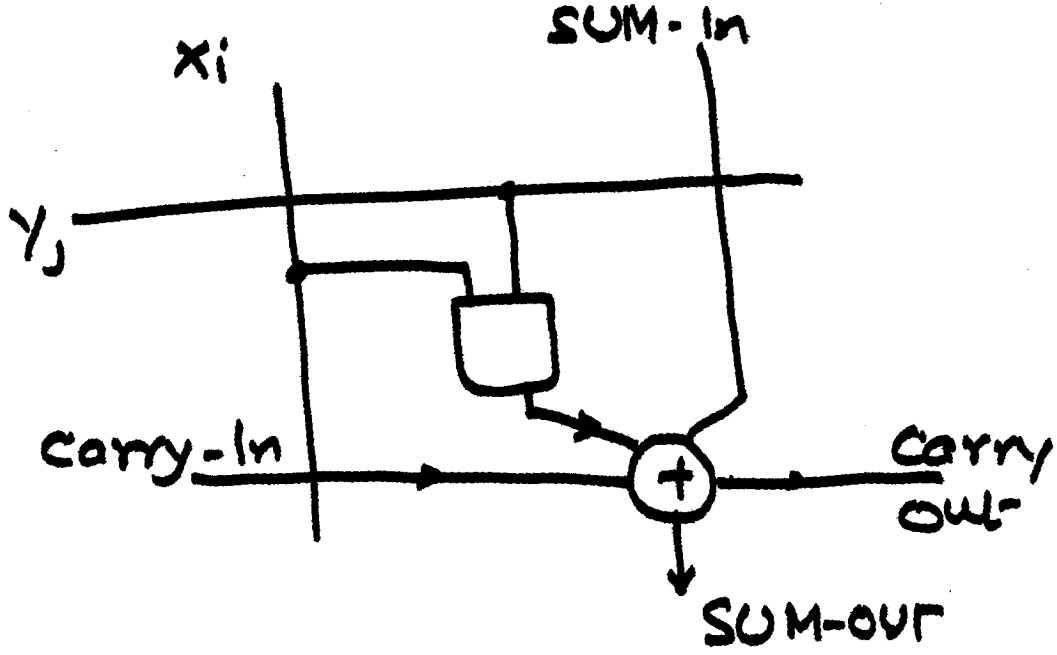
Product $P = X \cdot Y$

$$\therefore P = X_{n-1} Y_{n-1} 2^{2n-2} + \sum_{i=0}^{n-2} \sum_{j=0}^{n-2} X_i Y_j 2^{i+j}$$

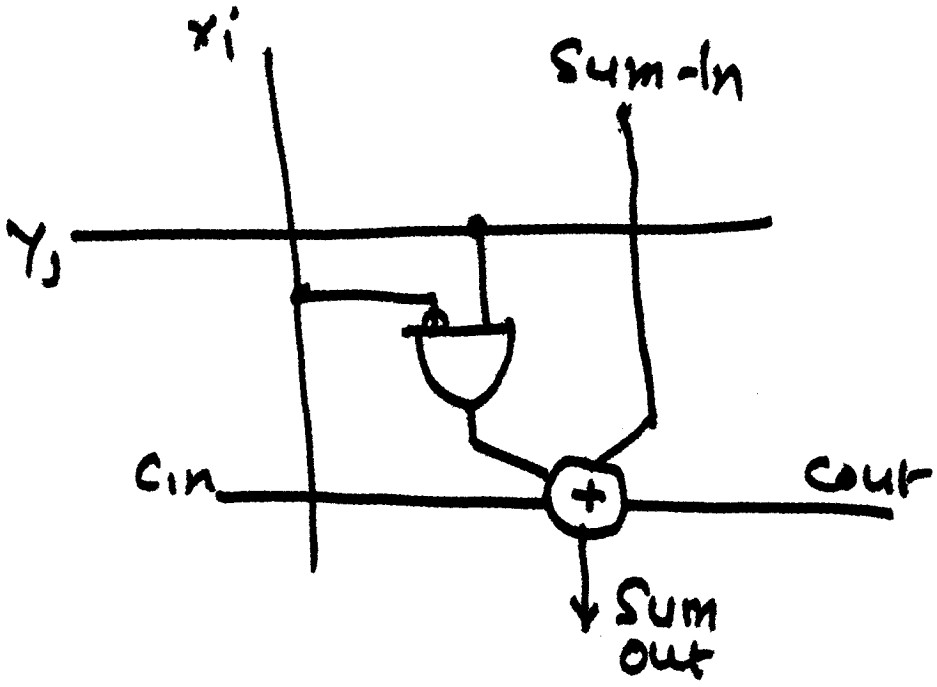
$$- X_{n-1} \sum_{j=0}^{n-2} Y_j 2^{n+j-1} - Y_{n-1} \sum_{i=0}^{n-2} X_i 2^{n+i-1}$$

To Avoid Subtractor Cell use, we represent -ve quantities as

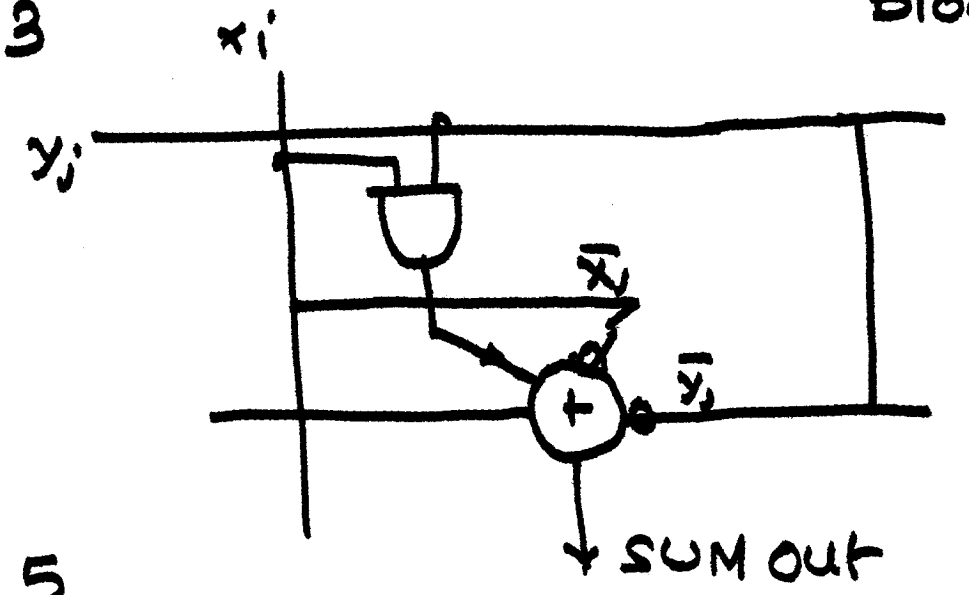
$$- X_{n-1} \sum Y_j 2^{n+j-1} = X_{n-1} \left[-2^{2n-2} + 2^{n-1} + \sum_{j=0}^{n-2} Y_j 2^{n+j-1} \right]$$



Block cell – 3

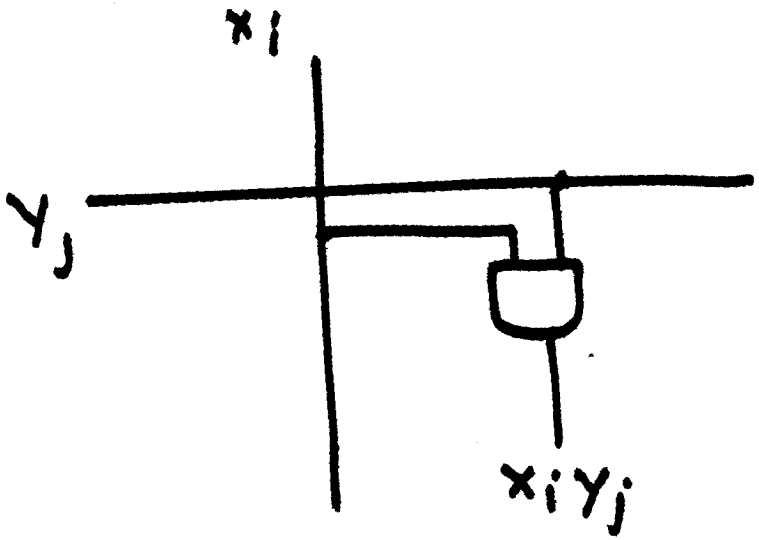


Block Cell – 4

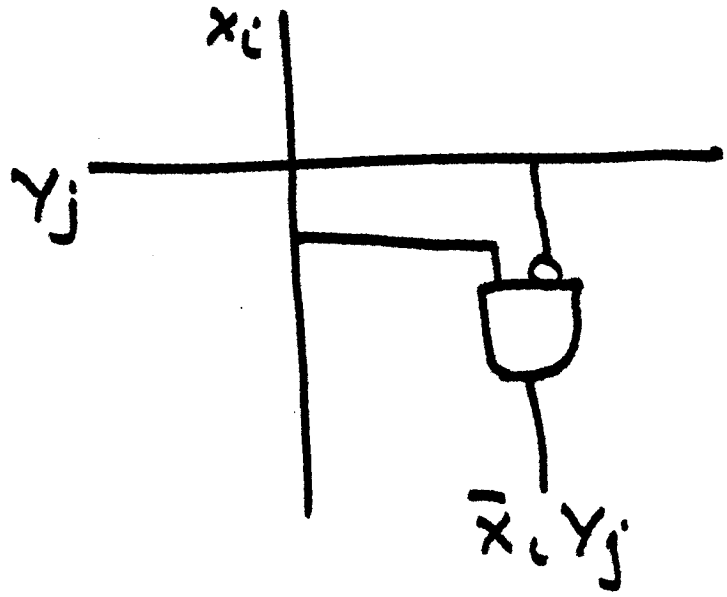


Block cell – 5

FIVE BASIC OPERATION BLOCKS
In
Baugh - Wooley Multiplier



Block Cell - 1



Block - cell - 2

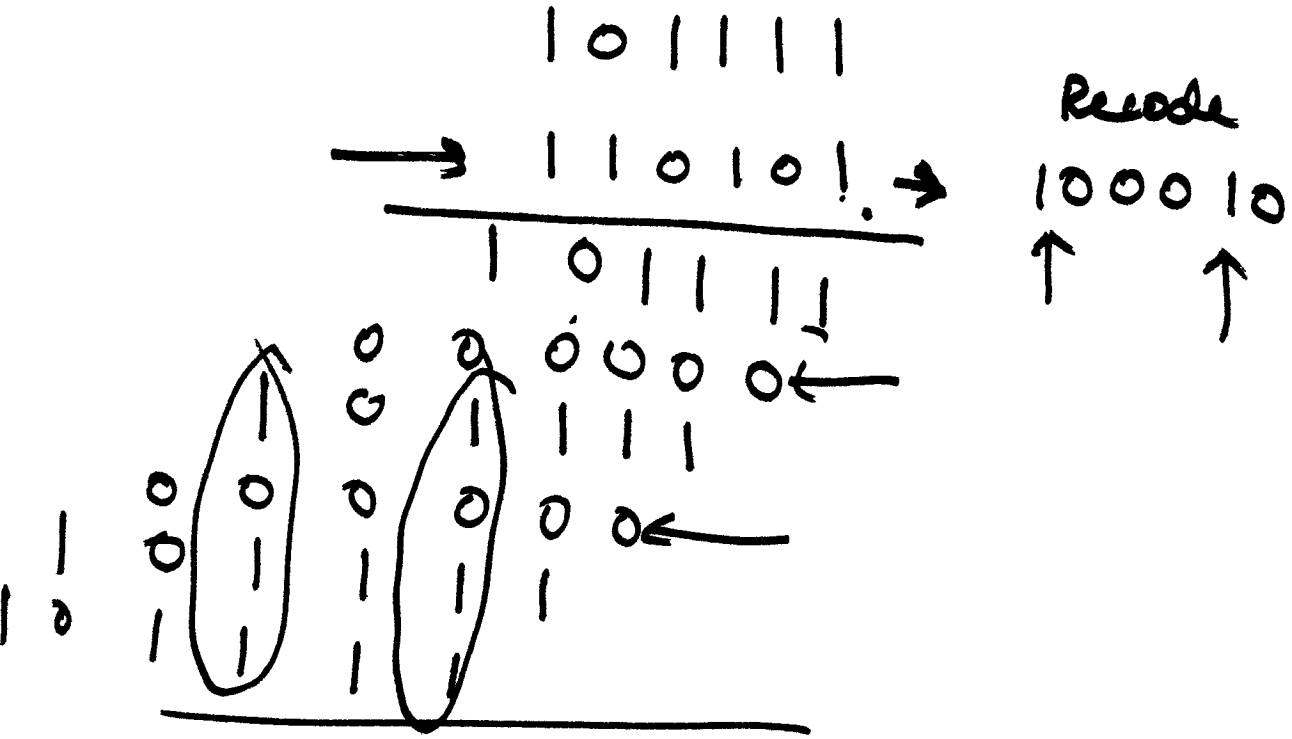
In this Multiplier we need

1. 2's Complement Generator

2 AND GATES to get Partial Products

3 Full Adders

To Save Area and also to Improve Speed nxm bit Multiplier is arranged in an ARRAY-FORM.


Recode
1000 10

In Booth's MULTIPLIER we 'recode'

2's Complement Nos.

Since we use Binary Nos, we observe that :

j-long Sequence of 1's is equivalent to

=> (j-1) long Sequence of 0's.

Replacement of 1's by 0's reduce Partial Product Terms.

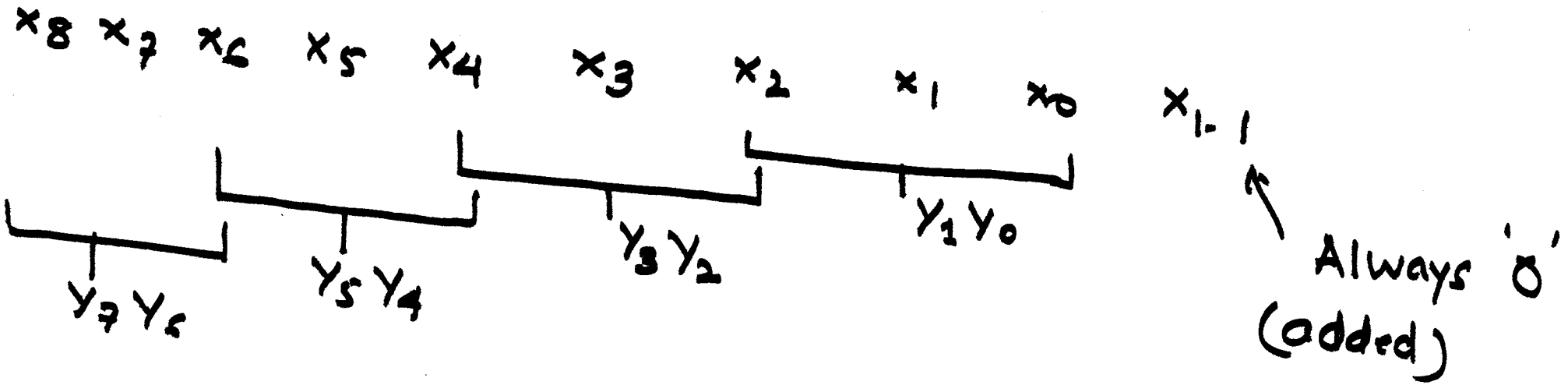
Weakness of Booth's Simple Recoding Techique:

It may creat more 1's then Initial number of 1's in the sequence after Recoding. This happens when in the number we have sparse 1's. Ex.

85 → 0 0 1 0 1 0 1 0 1 → Number

0 1, -1, 1, -1, 1, -1 → Booth Codrd

MODIFIED BOOTH RECODING



Here $x_0 - x_{n-1}$ are the Original Numbers

$y_0 - y_{n-2}$ are Modified Booth Recoded Numbers.

Example here uses Radix-4 scheme

What we learn :— 3 bits Inspected &
2 bits get eliminated

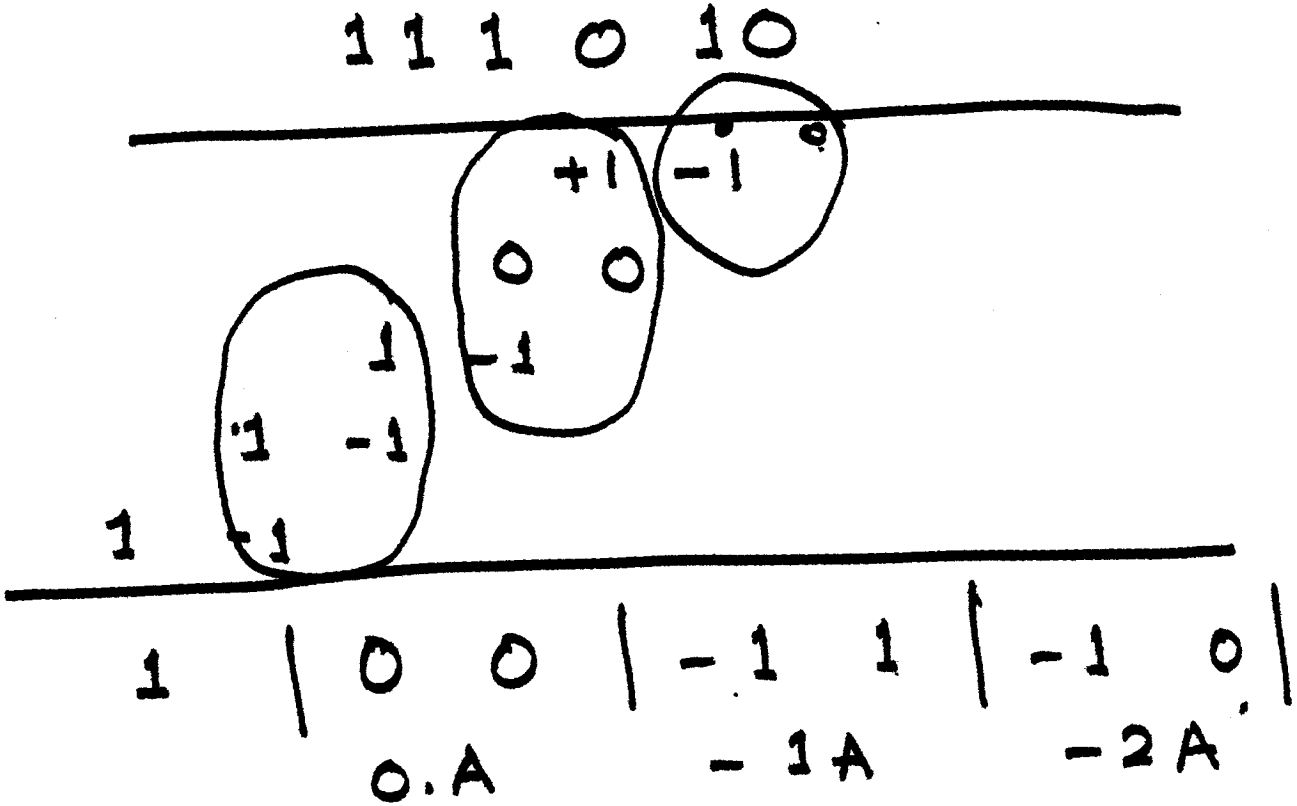
Inspected Bits			Recoded Bits		Recoded Digit times Multiplicand (A)	
x_i	x_{i-1}	x_{i-2}	y_i	y_{i-1}		
0	0	0	0	0	0	0
0	0	1	0	1	+1	1A
0	1	0	0	1	+1	1A
0	1	1	1	0	+2	2A
1	0	0	-1	0	-2	-2A
1	0	1	0	-1	-1	-1A
1	1	0	0	-1	-1	-1A
1	1	1	0	0	0	0

Example

Multiplicand A 0 0 1 1 0 1 (13)

Multiplier x 1 1 10 1 0 (-6)

Booth Recoding of x



Multiplicand A 0 0 1 1 0 1 Decimal 13

Multiplicy X 1 1 1 0 1 0 X -6

-78

OPERATION	0	-A	-2A
Initial 0	0 0 0 0 0 0		
Add (-2A)	+ 1 0 0 1 0 1		
	1 0 0 1 0 1		
Shift 2bits	1 1 0 1 0 1 0 1		
Add(-1A)	+ 1 1 0 0 1 1 0 0		
	Neglect 1 1 0 1 1 0 0 0 1		

overflow

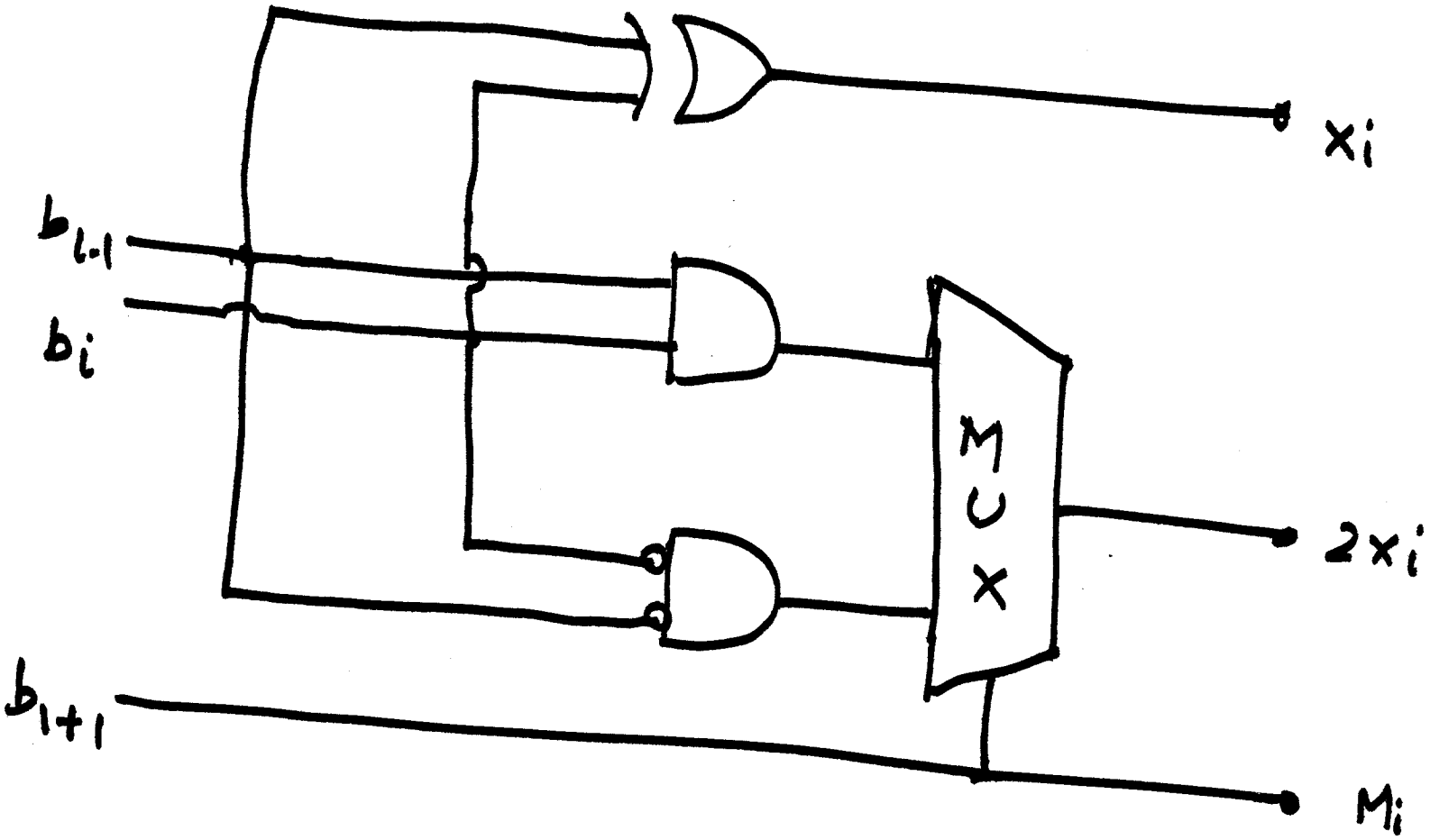
Last 1 0 1 1 0 0 0 1

Shift 2 bits 1 1 1 0 1 1 0 0 0 1

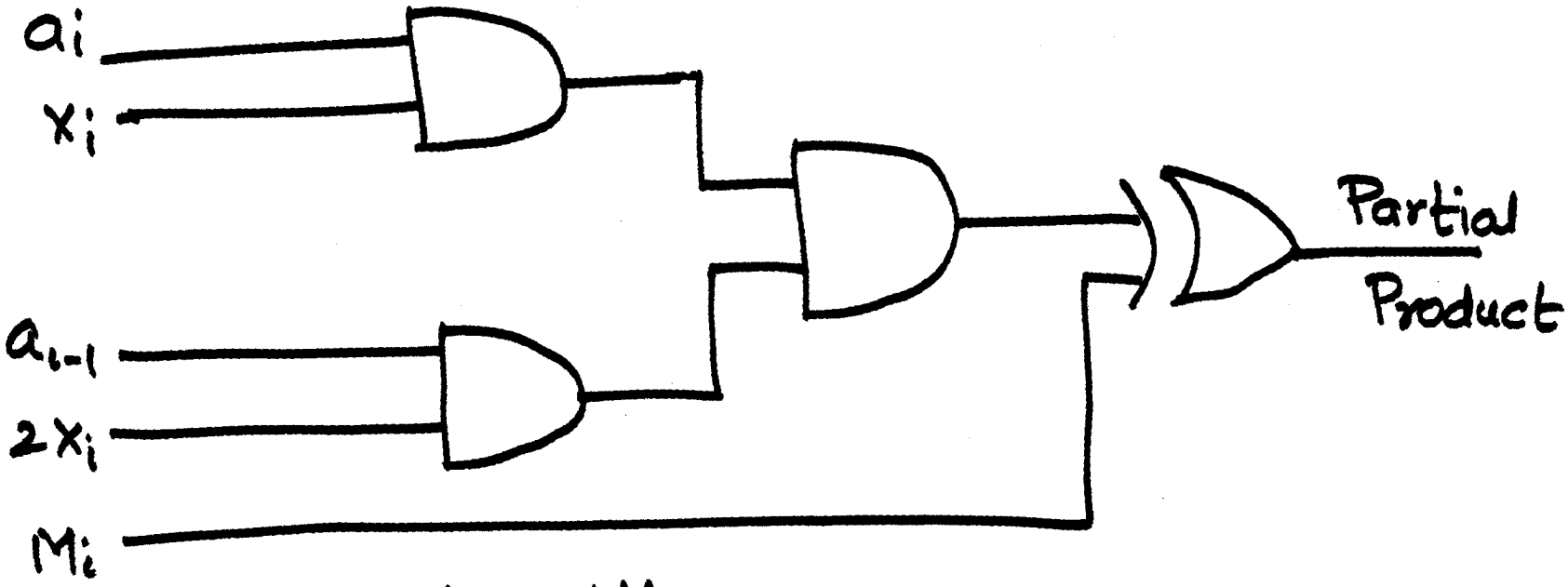
Add (0·A) + 0 0 0 0 0 0 0 0 0 0

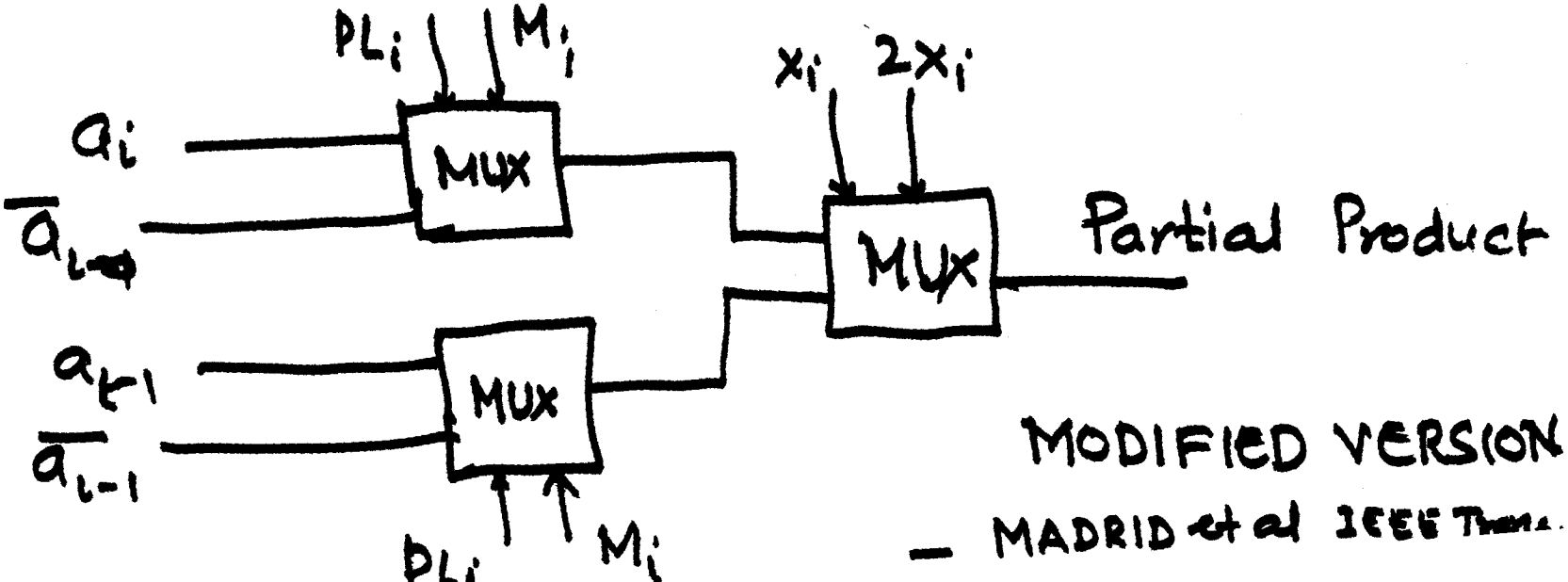
Sign Bit. 1 1 | 1 0 1 1 0 0 0 1 ⇒ −78

BOOTH ENCODER



Partial Product Generator





MODIFIED VERSION

– MADRID et al IEEE Trans. VLSI 1993

Shifter uses Pass Gates & Buffer.