3.5. Sequential Two's Complement Multiplication based on Booth's procedure.

Iteration step for S-M, Robertson methods: $\begin{cases} P_i = P_i + x_i \cdot Y & \text{ADD} \\ P_{i+1} = P_i \cdot 2^{-1} & \text{RShift} \end{cases}$

- each of the 2 operations requires 1 CLK cycle (C.C.)

! - if $x_i == 0$ → no ADD operation perform
  ⟹ iteration required only 1CC, not 2CC

! - the more bits of 1s in X ⟹ multiplication lasts longer
- the less bits of 1s in X ⟹ multiplication lasts less

Andrew Booth: inspect pairs of bits $x_i x_{i-1}$ (x_i → current bit, x_{i-1} → previous bit)

- analyse transitions in X

Without loss of generality, consider X, Y, - C_2, n-bits, integers

ADD Y ← → SUBTRACT Y

$$X = x_{n-1} x_{n-2} \dots x_{i+k+2}\, x_{i+k+1}\, x_{i+k} \dots x_{i+1}\, x_i\, x_{i-1}\, x_{i-2} \dots x_1 x_0$$

$$0 \; 1 \dots 1 \; 1 \; 0$$

(k+3 bits; k+1 bits of 1s)

$$X = X^* + X''$$

$$X^* = 0\,0 \dots 0 \;\; 0 \; 1 \dots 1 \; 1 \; 0 \; 0 \dots 0\,0$$

($x_{n-1} x_{n-2} \dots x_{i+k+2}\; x_{i+k+1} \dots x_{i-1}\, x_{i-2} \dots x_1 x_0$)

$$X'' = x_{n-1} x_{n-2} \dots x_{i+k+2}\; 0 \; 0 \dots 0\,0 \; 0 \; x_{i-2} \dots x_1 x_0$$

$$P = X * Y = (X^* + X'') * Y = X^* * Y + X'' * Y$$

$$P^* = X^* * Y$$

$$P^* = \sum_{j=i-1}^{i+k+1} x_j \cdot Y \cdot 2^j = Y(2^{i+k} + 2^{i+k-1} + \dots + 2^{i+1} + 2^i)$$

$$= Y(+2^{i+k+1} - 2^i)$$

! instead of k+1 ADD: 1: ADD, 1: SUB

- for pair $x_{i+k+1} x_{i+k} = 01$: ADD $Y \cdot 2^{i+k+1}$
- for pair $x_i x_{i-1} = 10$: SUBTRACT $Y \cdot 2^i$
- for pairs $x_j x_{j-1}$ = 00, = 11: no ADD / SUBTRACT.

Analysing X, requires the first pair $X_0 X_{-1}$ (for X integer X_{-1} - weight $\frac{1}{2}$)

- add X_{-1} to X: $\Rightarrow$ weight of $X_{-1} = \frac{1}{2} \cdot$ weight of X_0
  $\Rightarrow$ value of $X_{-1} = 0$ (not affect val of X)

Define Booth's recoding: uses signed digit
- not using bits anymore, but digit
- a Booth recoded digit X_{iB} can be $\begin{cases} 0: \text{weight} = 0 \cdot 2^i \\ 1: \text{weight} = 1 \cdot 2^i \\ \bar{1}: \text{weight} = -1 \cdot 2^i \end{cases}$
- adds X_{-1} to operand X.

Procedure: - scan from Right $\rightarrow$ to Left operand X
- replace each pair $X_i X_{i-1}$ accordingly

X_i	X_{i-1}	X_{iB}
0	0	0
0	1	1
1	0	$\bar{1}$
1	1	0

Example: Consider $X = -3 * 2^{-3}$, on 4 bits.

Ranks / Number	X_3	X_2	X_1	X_0	X_{-1}
	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
X_{SM}	1.	0	1	1	
X_{C2}	1.	1	0	1	0
X_B	0.	$\bar{1}$	1	$\bar{1}$	

$X_B = \bar{1} \cdot 2^{-1} + 1 \cdot 2^{-2} + \bar{1} \cdot 2^{-3}$
$= 2^{-3}(2^2 \cdot \bar{1} + 2^1 \cdot 1 + \bar{1})$
$= 2^{-3}(-4 + 2 - 1) = -3 \cdot 2^{-3}$

Since: $X_B == X_{C2} \Rightarrow X_{C2} * Y_{C2} \equiv X_B * Y_{C2}$

$X_B * Y_{C2}$: - at each iteration, depending on X_{iB}, perform the following operations:

$X_{iB} \begin{cases} = 0: \text{no ADD/SUBTRACT; Rightshift afterwards} \\ = 1: \text{ADD } Y_{C2} \text{ to partial prod; Rightshift afterw.} \\ = \bar{1}: \text{SUBTRACT } Y_{C2} \text{ from partial prod.; Rightshift afterw.} \end{cases}$

Accumulator (A): stores the most significant half of partial product.
- to A one ADDs Y or SUBTRACT Y.

$\Rightarrow$ A's sign can be different at different iterations
$\Rightarrow$ Rightshift must be ARITHMETIC.

multiplier 4

declare registers A[7:0], Q[7:-1], M[7:0], COUNT[2:0];

declare bus INBUS[7:0], OUTBUS[7:0];

BEGIN: A:=0 COUNT:=0, } ← {C0}

INPUT: M:=INBUS;

Q[7:0]:=INBUS[7:0], Q[-1]:=0; ← {C1}

TEST1: if Q[0]Q[-1]==01 then A:=A+M, goto TEST2; ← {C2}

if Q[0]Q[-1]==10 then A:=A-M; ← {C2,3}

TEST2: if COUNT2==1 then goto OUTPUT,

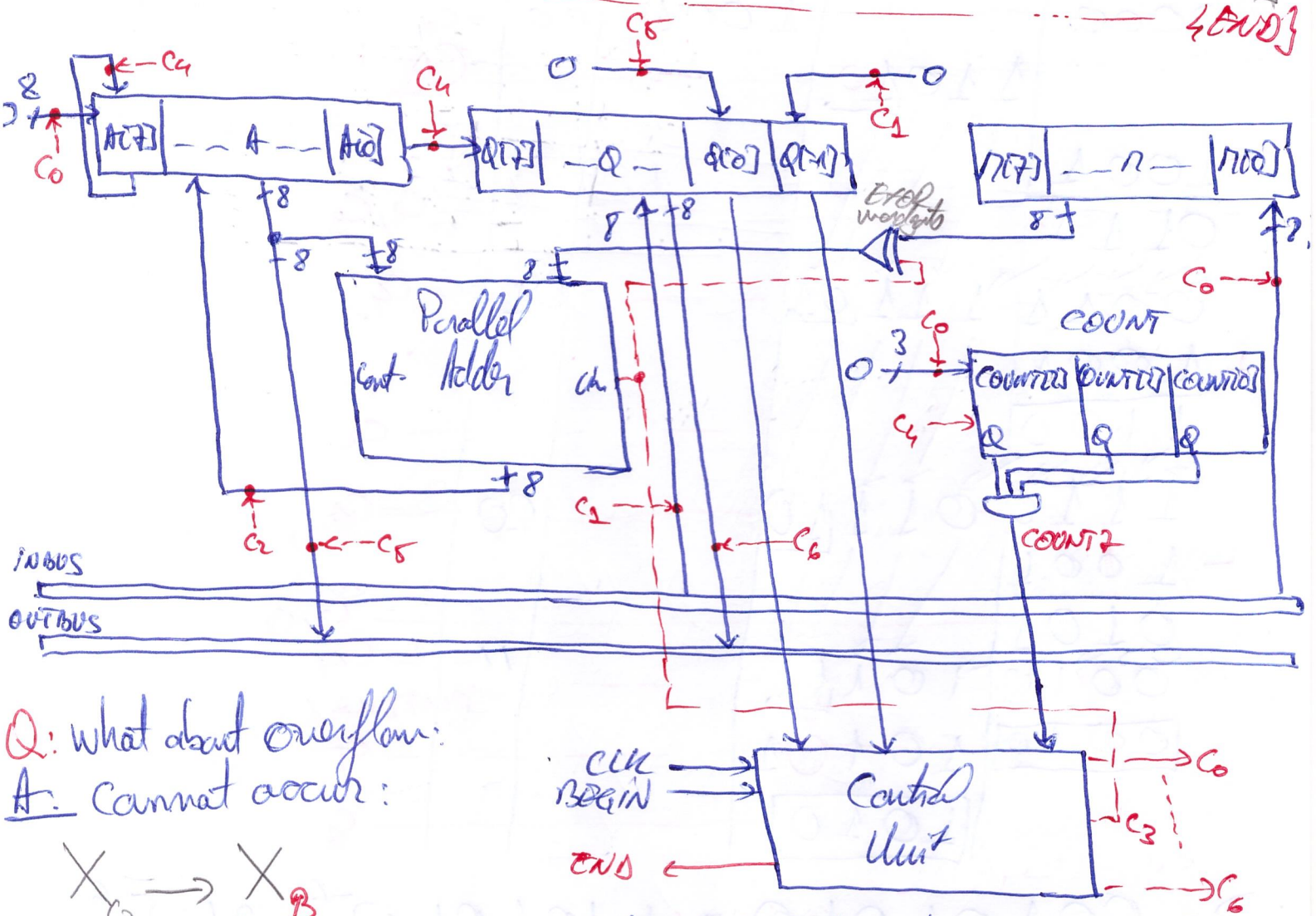
RShift: A[7]:=A[7], A[6:0],Q:=A,Q[7:0], } ← {C4}

INCREMENT: COUNT:=COUNT+1, goto TEST1;

OUTPUT: OUTBUS:=A, Q[0]:=0; ← {C5}

OUTBUS[7:0]:=Q[7:0]; ← {C6}

END: ... ← {END}



Q: What about overflow:

A: Cannot occur:

$X_{C2} \rightarrow X_B$

cannot have 2 consecutive ADDitions / SUBTRACTIONS

$X_{C2} = 01000101$

! No CORRECTION, + − + − + − + −

However: once the COUNT reached its final value
- perform one final ADD / SUBTRACT, if the next significant pair of bit of X requires it
- no subsequent Rightshift !

if $x_7 x_6 == 01$: one final ADD
if $x_7 x_6 == 10$: one final SUBTRACT
! no RShift.

Booth's method treats the sign bit of X, as a magnitude bit

Example: $X = -0.375 = -3 \cdot 2^{-3} = 1.101_{c2}$
$Y = -0.875 = -7 \cdot 2^{-3} = 1.001_{c2}$

A.	Q	M	COUNT	A.C.S.
0000		1001	00	C_0
	11010			C_1
− 1001				
0111				C_2, C_3
0011	11101		01	C_4
+ 1001				
1100				C_2
1110	01110		10	C_4
− 1001				
0101				C_2, C_3
0010	10111		11	C_4
			+COUNT3 == 1	
00010	10101			C_5
	1010			C_6

$P = 00101010 = +10101 \cdot 2^{-6} = 21 \cdot 2^{-6}$

$(-3) \cdot 2^{-3} \cdot (-7) \cdot 2^{-3} = +21 \cdot 2^{-6}$ ✓

$X = 5 = 0101_{c2}$

$Y = -2 = 1110_{c2}$

$[-8; +7]$ (3)

A	Q	M	COUNT
0000	0 1 0 1 0	1 1 1 0	00
− 1 1 1 0			
0 0 1 0			
0 0 0 1	0 0 1 0 1		01
+ 1 1 1 0			
1 1 1 1			
1 1 1 1	1 0 0 1 0		10
− 1 1 1 0			
0 0 0 1			
0 0 0 0	1 1 0 0 1		11
+ 1 1 1 0			
1 1 1 0			
1 1 1 1	0 1 1 0 0		

*Count[3] == 1

Integer RShift (Arithmetic)

$P = 1111011 0_c = -128 + 64 + 32 + 16 + 6$

$= -16 + 6 = -10$

$+5 * (-2) = -10$ ✓

3.6. Combinational array structures for binary multiplication.

X, Y – unsigned, 4 bits, integers;

$X = x_3 x_2 x_1 x_0$

$Y = y_3 y_2 y_1 y_0$

$$P = X * Y = \left(\sum_{i=0}^{3} x_i \cdot 2^i\right)\left(\sum_{j=0}^{3} y_j \cdot 2^j\right)_2$$

$$= \sum_{i=0}^{3} 2^i \left(\sum_{j=0}^{3} x_i y_j \cdot 2^j\right)$$

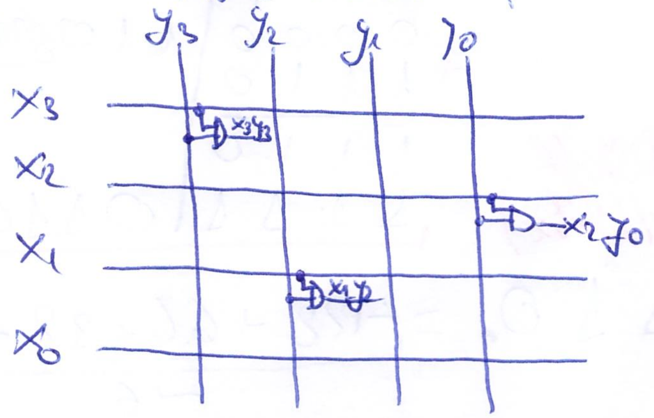
$$P = 2^0(x_0 y_0 2^0 + x_0 y_1 2^1 + x_0 y_2 2^2 + x_0 y_3 2^3) +$$
$$2^1(x_1 y_0 2^0 + x_1 y_1 2^1 + x_1 y_2 2^2 + x_1 y_3 2^3) + \ldots +$$
$$2^3(x_3 y_0 2^0 + x_3 y_1 2^1 + x_3 y_2 2^2 + x_3 y_3 2^3)$$

$P = 2^6 \cdot x_3 \cdot y_3 +$
$2^5 \cdot (x_3 y_2 + x_2 y_3) +$
$2^4 \cdot (x_3 y_1 + x_2 y_2 + x_1 y_3) +$
$2^3 \cdot (x_3 y_0 + x_2 y_1 + x_1 y_2 + x_0 y_3) +$
$2^2 \cdot (x_2 y_0 + x_1 y_1 + x_0 y_2) +$
$2^1 \cdot (x_1 y_0 + x_0 y_1) +$
$2^0 \cdot x_0 y_0$

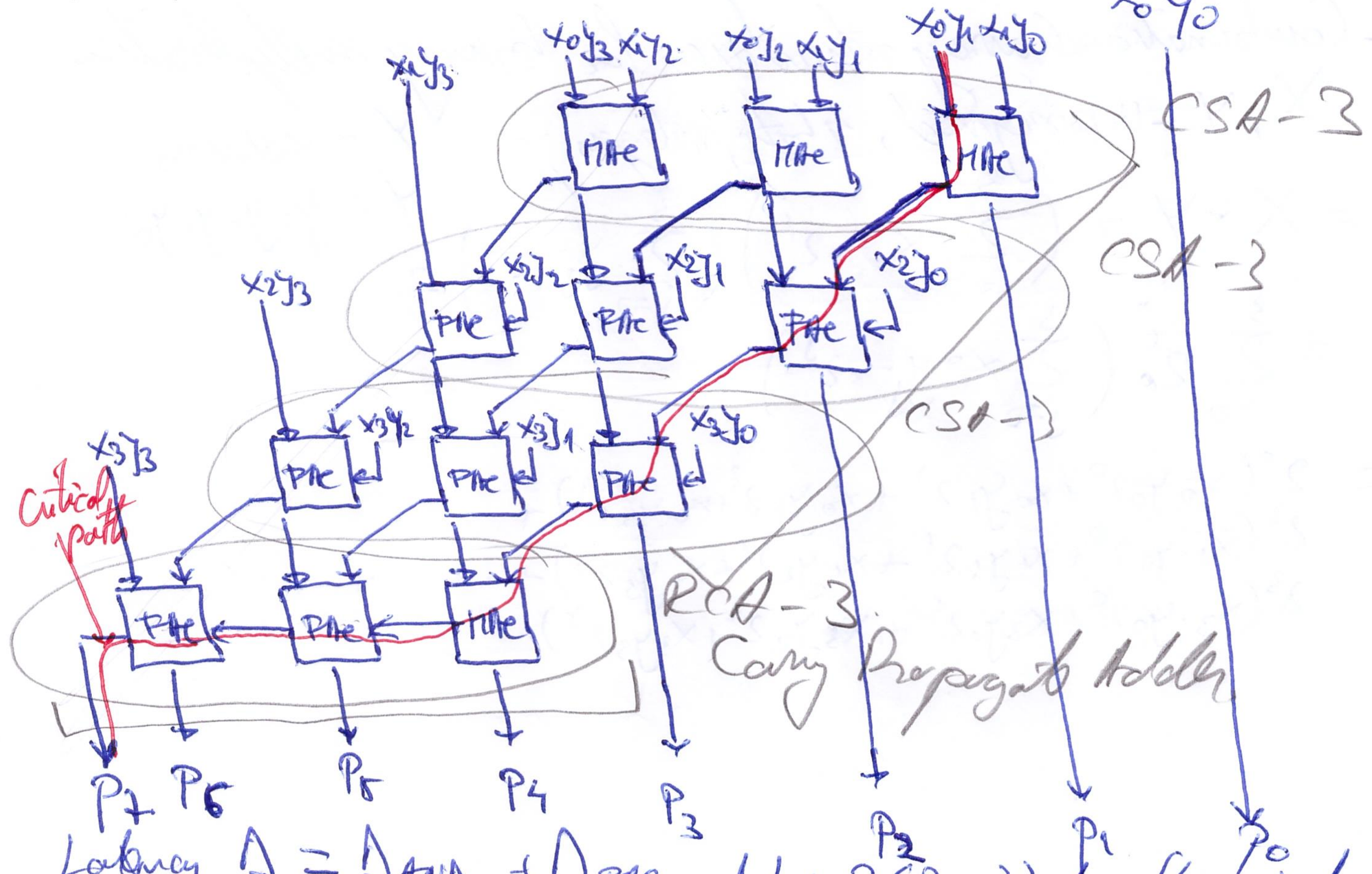
$x_i \cdot y_j \quad i+j =$ equal

2 matrices → AND matrix : $x_i \cdot y_j$ (logic AND)
→ FAC matrix :

AND matrix:



FAC matrix
FAC/HAC:



Latency $\Delta = \Delta_{AND\ matrix} + \Delta_{FAC\ matrix} = 1d + 2(2n-3)d = (4n-5)d.$