4.4. Parallel Adder: 8-bit.

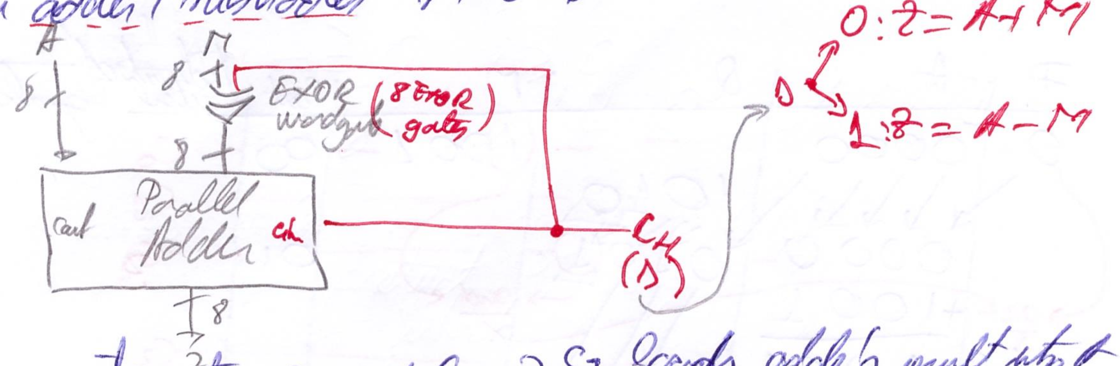
- cout is ignored (for C2 +/−, cout is ignored)

Q: Can overflow occur?!! A: Yes → Q: How is overflow treated?!?

A: RShift of A restores A's sign from flag F.

Flag F, stores the sign of the partial product at any given moment!

- is part of an adder/substracter structure




At the connection, activate $c_2$ and $c_4$ → $c_2$ loads adder's result into A
→ $c_4$ transform the adder into a substracter

Flag F: stores p. products sign; perform partial products Arithmetic RShift.

- iteration step: $\begin{cases} P_i := P_i + x_i \cdot Y & \text{(Add)} \\ P_{i+1} := P_i * 2^{-1} & \text{(RShift)} \end{cases}$ $i \geq 0$, $P_0 := 0$

- starting from $i=0$, as long as $x_i$ remains 0 (for all least significant bits of X having value 0) ⇒ no addition is performed for the least signif. bits $x_i == 0$. ($+x_i \cdot Y \equiv +0$)

⇒ $P_i$ remain 0, as long as lsb $x_i == 0$.

⇒ sign of those null $P_i$ must remain 0.
! not equal to M[7] (sign of Y)

this is why flag F is used (it would have been easier to restore A's sign directly from M[7])

after the first bit $x_i \neq 0$, flag F is set to value of M[7]

$F := F \text{ or } (\underbrace{Q[0]}_{\text{always stores current bit } x_i} \text{ and } M[7])$

if $M[7] == 0$ → F is always 0

Sign of the result is not anymore set (compare to S-M multiplication)

- operating C2's numbers generates the correct sign of result

Depending on the signs of X and Y, the algorithm performs the following operations

| $X_7$ | $Y_7$ | CORRECTION Step | Arithmetic RShift |
|---|---|---|---|
| 0 | 0 | NO | NO |
| 0 | 1 | NO | YES |
| 1 | 0 | YES | NO |
| 1 | 1 | YES | YES |

*Example:*

$X = -0.75 = -3 * 2^{-2} = 1.110_{SM} = 1.010_{C2}$

$Y = -0.875 = -7 * 2^{-3} = 1.111_{SM} = 1.001_{C2}$

$P = X * Y = (-3) * 2^{-2} * (-7) * 2^{-3} = 21 * 2^{-5}$

| F | A | Q | M | Count | Activated Control signals |
|---|---|---|---|---|---|
| 0 | 0000 | | 1001 | 00 | $C_0$ |
| | | 1010 | | | $C_1$ |
| | 0000 | 0101 | | 01 | $C_3$ |
| or F | +1001 | | and | | |
| 1 | 1001 | | | | $C_2$ |
| | 1100 | 1010 | | 10 | $C_3$ |
| | 1110 | 0101 | | 11 | $C_3$ |
| | | | | *COUNT3 == 1 | sign of X |
| | -1001 | | | | |
| | 0101 | 0100 | | | $C_2, C_4$ |
| | $\boxed{0101}$ | | | | $C_5$ |
| | | $\boxed{0100}$ | | | $C_6$ |

$P = 0.1010100 = +\overset{2^4}{1}0\underbrace{1 0 \overset{2^0}{1}} * 2^{-5} = 21 * 2^{-5}$

Example 2. consider X, Y - integers. one final Arithmetic RShift.

$X = +3 = 0011_{C2}$ $\quad Y = -5 = \underset{2^3}{1}011_{C2} = 0011 - 8 = 3 - 8$

$X * Y = +3 * (-5) = -15$

| F | A | Q | M | COUNT |
|---|---|---|---|---|
| 0 | 0000 | 0011 | 1011 | 00 |
| or F | +1011 | and | | |
| 1 | 1011 | | | |
| | 1101 | 1001 | | 01 |
| | +1011 | | | |
| | 1000 | | | |
| | 1100 | 0100 | | 10 |
| | 1110 | 0010 | | 11 |
| | 1111 | 0001 | | |
| | $\boxed{1111}$ | $\boxed{0001}$ | | |

*COUNT3 == 1 final Arithmetic RShift, requires an additional control signal

$16 \cdot 7 = 112$

$P = 1111\,0001_{C2} = -128 + \overbrace{0111}^{7*16}\,\overbrace{0001}^{1*16^0} = -128 + 113 = -15.$

4.5. Sequential Two's Complement Multiplication based on Booth's Procedure. ②

Iteration steps for SM, Robertson's methods: $\begin{cases} P_i := P_i + x_i \cdot Y & \text{(Addition)} \\ P_{i+1} := P_i * 2^{-1} \end{cases}$

! the more bits of 1's in multiplier X ⟹ the more additions performed,

! the less bits of 1's in X ⟹ the less additions performed

fewer additions ⟹ faster multiplication (fewer clk cycles)

Andrew Booth:

- inspect pairs of bits $x_i x_{i-1}$ instead of inspecting only $x_i$
  - analyse transitions into X: $x_i x_{i-1} = 01$ and $x_i x_{i-1} = 10$.

Suppose X, Y – C2, n bits, integers (without loosing generality)

k+3 bits

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

add Y to P ⟵ 0 1 ... 1 1 0 ⟶ subtract Y from P

continuous run of 1s (k+1)

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

$X^*$ captures the k+1 run of 1's inside Y.

$$X = X^* + \underbrace{X^{**}}_{\text{remaining bits in X after capturing the run of 1's into } X^*}$$

$$P = X * Y = (X^* + X^{**}) * Y =$$

$$= \underbrace{X^* * Y}_{P^* = X^* * Y} + X^{**} * Y$$

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

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

- instead of performing k+1 additions, perform 1 addition, 1 subtraction
  - for pair $x_{i+k+1} x_{i+k} = 01$: perform addition of $Y \cdot 2^{i+k+1}$
  - for pair $x_i x_{i-1} = 10$: perform subtraction of $Y \cdot 2^i$
  - for pair $x_j x_{j-1} = 11$: perform no-addition/subtraction
  - for pair $x_i x_{i-1} = 00$: perform no-addition/subtraction

Analyzing X, requires the first pair $x_0 \, x_{-1}$

- add $x_{-1}$ bit to X → weight of $x_{-1}$ is $\frac{1}{2}$ weight of $x_0$
  → value of 0 (not to influence value of X)

⇒ Booth's recoding

- uses signed digits: a bit $x_{i_B}$ can be
  → 0, weight $0 \cdot 2^i$
  → 1, weight $1 \cdot 2^i$
  → $\bar{1}$, weight $-1 \cdot 2^i$

  → not anymore bit, at least 2 bits

- extend operand with $x_{-1}$

Scan the operand from Right to Left and replace pairs $\boxed{x_i \, x_{i-1}}$ accordingly

| $x_i$ | $x_{i-1}$ | $x_{i_B}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | $\bar{1}$ |
| 1 | 1 | 0 |

Example: Consider $X = -1 * 2^{-3}$, on 4 bits

Booth's recoding:

| Number \ Ranks | $x_3$ $2^0$ | $x_2$ $2^{-1}$ | $x_1$ $2^{-2}$ | $x_0$ $2^{-3}$ | $x_{-1}$ $2^{-4}$ |
|---|---|---|---|---|---|
| $X_{SM}$ | 1. | 0 | 0 | 1 | |
| $X_{C2}$ | 1. | 1 | 1 | 1 | 0 |
| $X_B$ | 0. | 0 | 0 | $\bar{1}$ | |

← weight

$$X_B = \bar{1} \cdot 2^{-3} = -1 * 2^{-3}$$

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

- at each iteration depending on bit $x_{i_B}$ of $X_B$, the following operations will be performed:

$x_{i_B}$
→ 0: no addition/subtraction, perform RShift afterwards
→ 1: add $Y_{C2}$ to partial product, perform RShift afterwards
→ $\bar{1}$: subtract $Y_{C2}$ from partial product, perform RShift afterwards

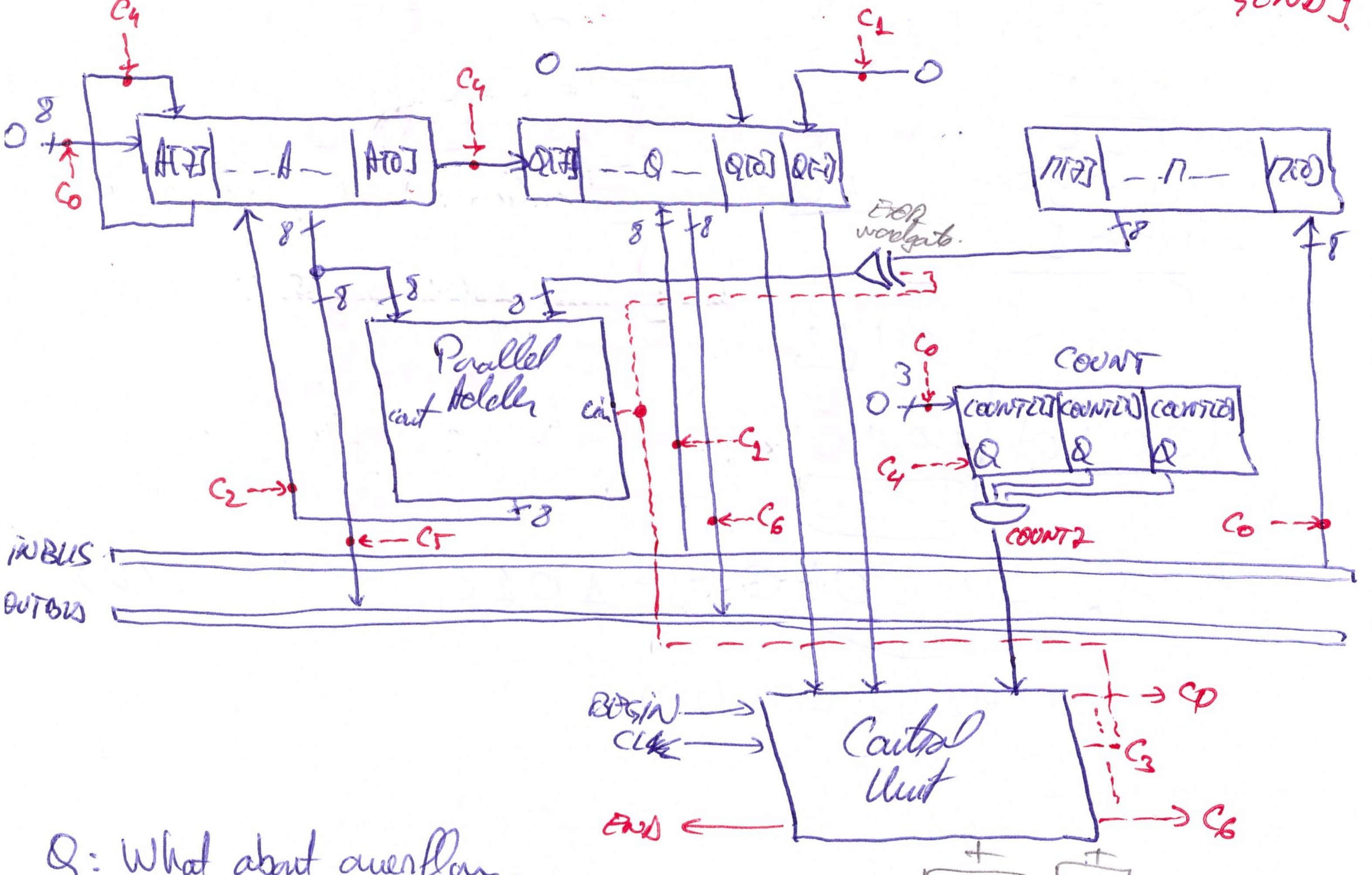
to A: one can add Y or subtract Y from.

⇒ A can have different signs at different moments

→ RShift needs to 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, go to TEST2; {c2}
if Q[0]Q[-1]==10 then A:=A-M; {c2, c3}
TEST2: if COUNT7==1 then go to OUTPUT,
RSHIFT: A[7]:=A[7], A[6:0].Q:=A.Q[7:0],
INCREMENT: COUNT:=COUNT+1, go to TEST1; {c4}
OUTPUT: OUTBUS:=A, Q[0]:=0; {c5}
END: OUTBUS[7:0]:=Q[7:0]; {c6}
{END}




Q: What about overflow
A: alternate + with −
- because no 2 consecutive ++ or −− can occur
=> no Overflow occurs.
No CORRECTION is required.
! However, a final arithmetic operation might be performed, when COUNT7 become 1

- if $x_7 x_6 == 01 \Rightarrow$ addition
- if $x_7 x_6 == 10 \Rightarrow$ subtraction

Conclusion: Booth's procedure treats the sign bit as any other magnitude bit.

Example: $X = -0.375 = -3 \times 2^{-3} = 1.101_{C2}$

$Y = -0.875 = -7 \times 2^{-3} = 1.001_{C2}$

| 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 |
| $\boxed{0010}$ | 10101 | | | $c_5$ |
| | $\boxed{1010}$ | | | $c_6$ |

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

$$(-3) \times (2^{-3}) \times (-7) \times (2^{-3}) = 21 \times 2^{-6}$$