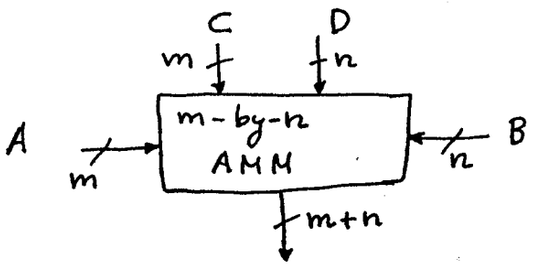
EE 7715

Designing unsigned multipliers using Additive Multiply Modules (AMMs).

An m-by-n Additive Multiply Module (AMM) is defined below:



$$P = A \times B + C + D$$

The numbers A, B, C, D are unsigned

The number P should be of size m+n bits.

Observe that

$$P_{max} = A_{max} \times B_{max} + C_{max} + D_{max}$$

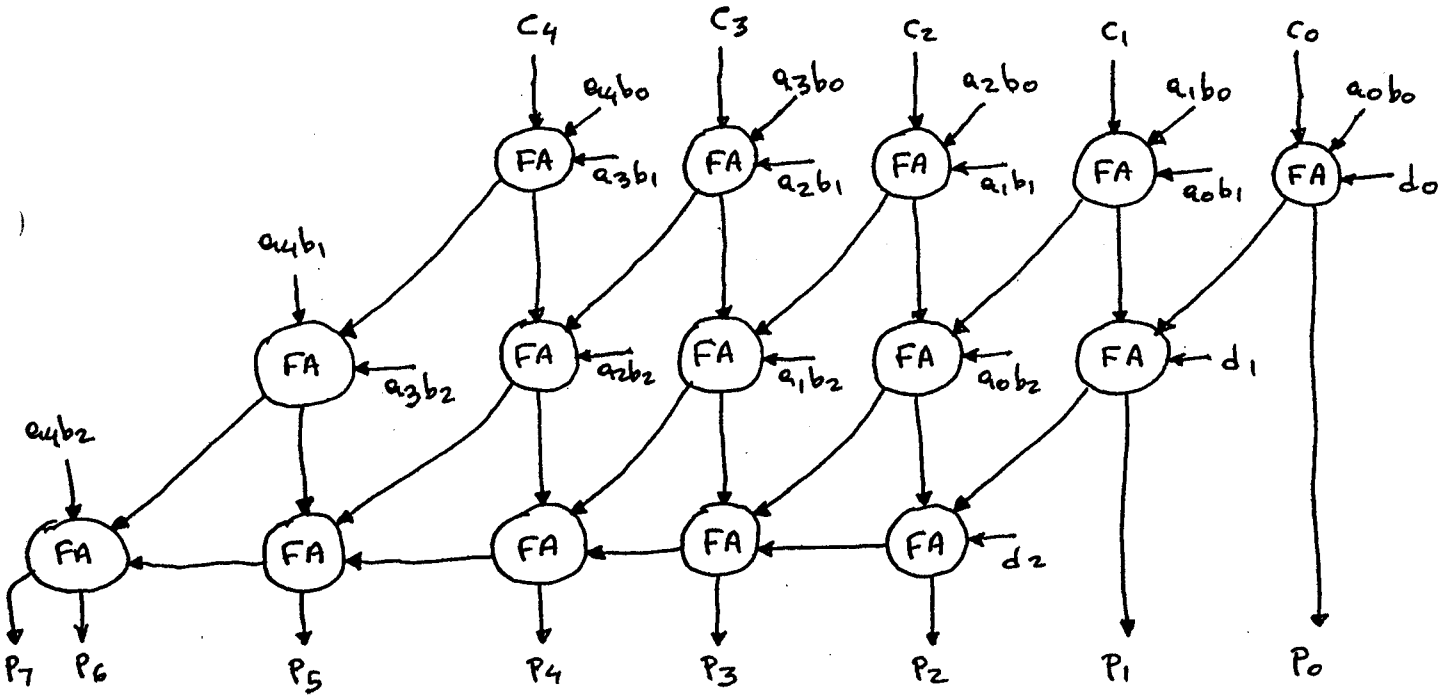
$$= (2^m - 1) \times (2^n - 1) + 2^m - 1 + 2^n - 1$$

$$= 2^{m+n} - 2^n - 2^m + 1 + 2^m - 1 + 2^n - 1$$

$$= 2^{m+n} - 1$$ which requires m+n bits for its representation.

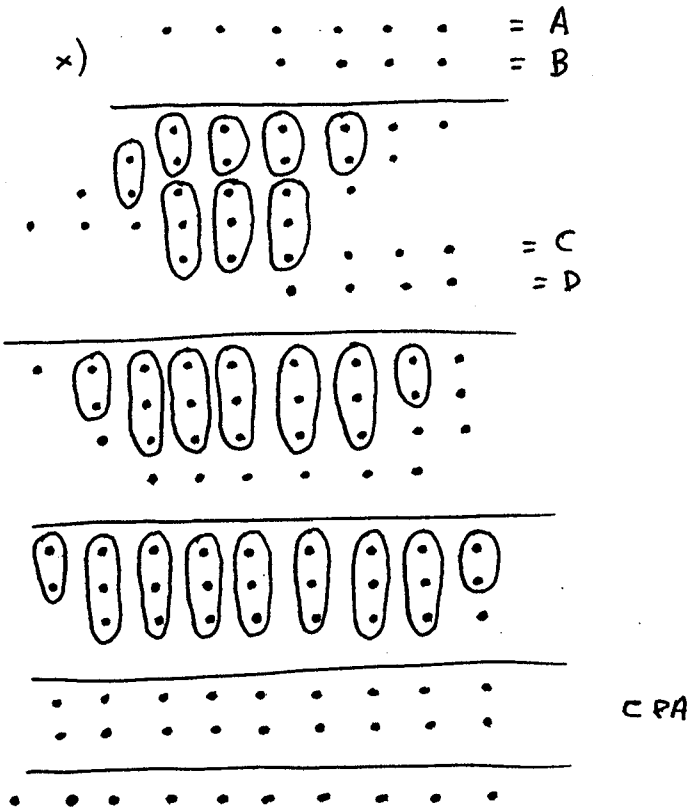
Problem : Construct a 5-by-3 AMM using a modification of the cellular array multiplier for unsigned numbers.

$$
\begin{array}{rccccc}
 & a_4 & a_3 & a_2 & a_1 & a_0 = A \\
\times) & & & b_2 & b_1 & b_0 = B \\
\hline
 & a_4b_0 & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & a_4b_1 & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 \\
+) & a_4b_2 & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 \\
\hline
 & c_4 & c_3 & c_2 & c_1 & c_0 = C \\
+) & & & d_2 & d_1 & d_0 = D \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 \quad p_1 \quad p_0 = P
\end{array}
$$

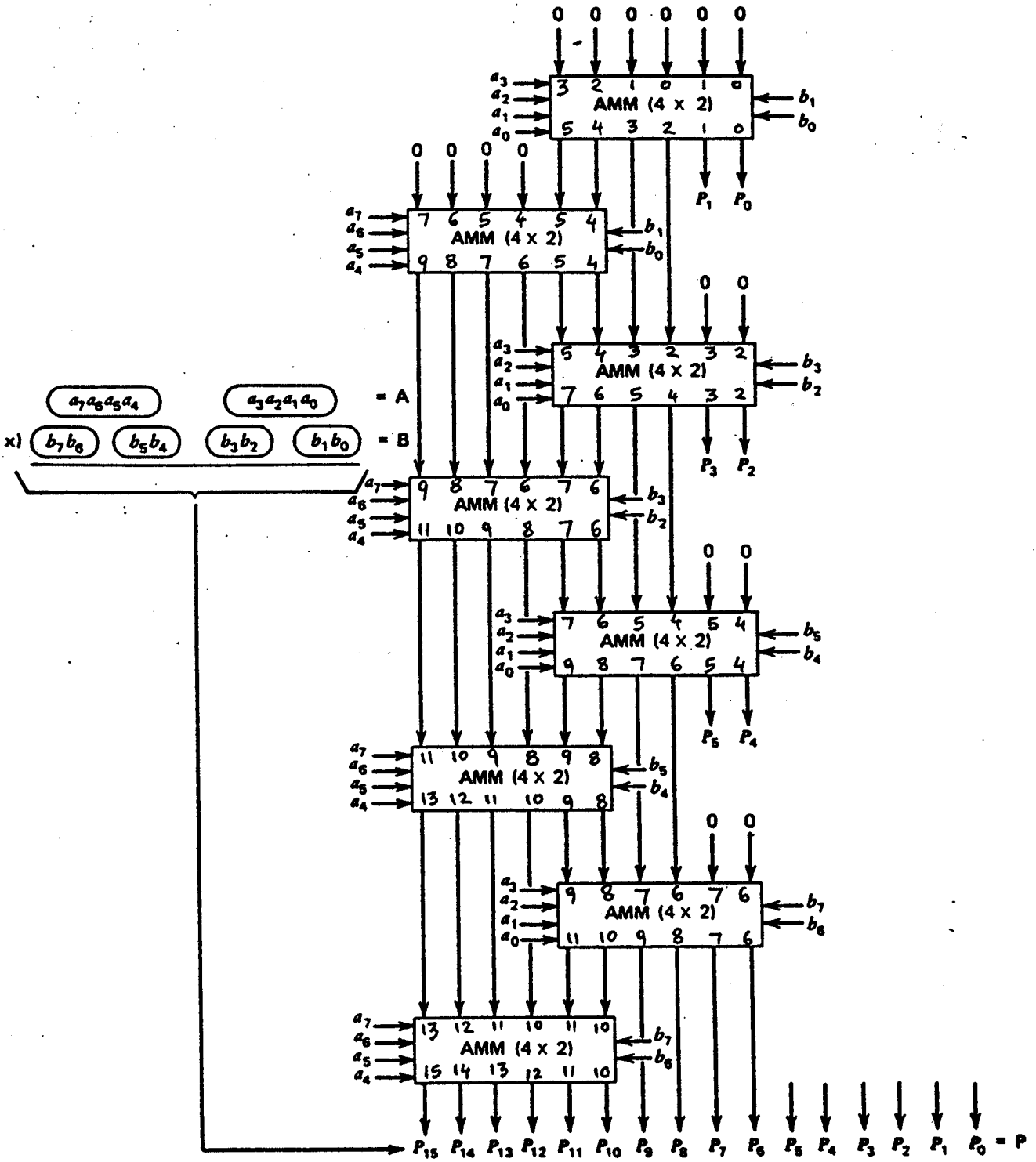


③ m

Problem : Construct a 6-by-4 AMM using (3,2) and (2,2) counters.



Problem: Construct an 8-by-8 unsigned multiplier using 4-by-2 AMMs. (4) m



Problem: Construct a 32-by-32 unsigned multiplier using 8-by-8 AMMs.

Solution

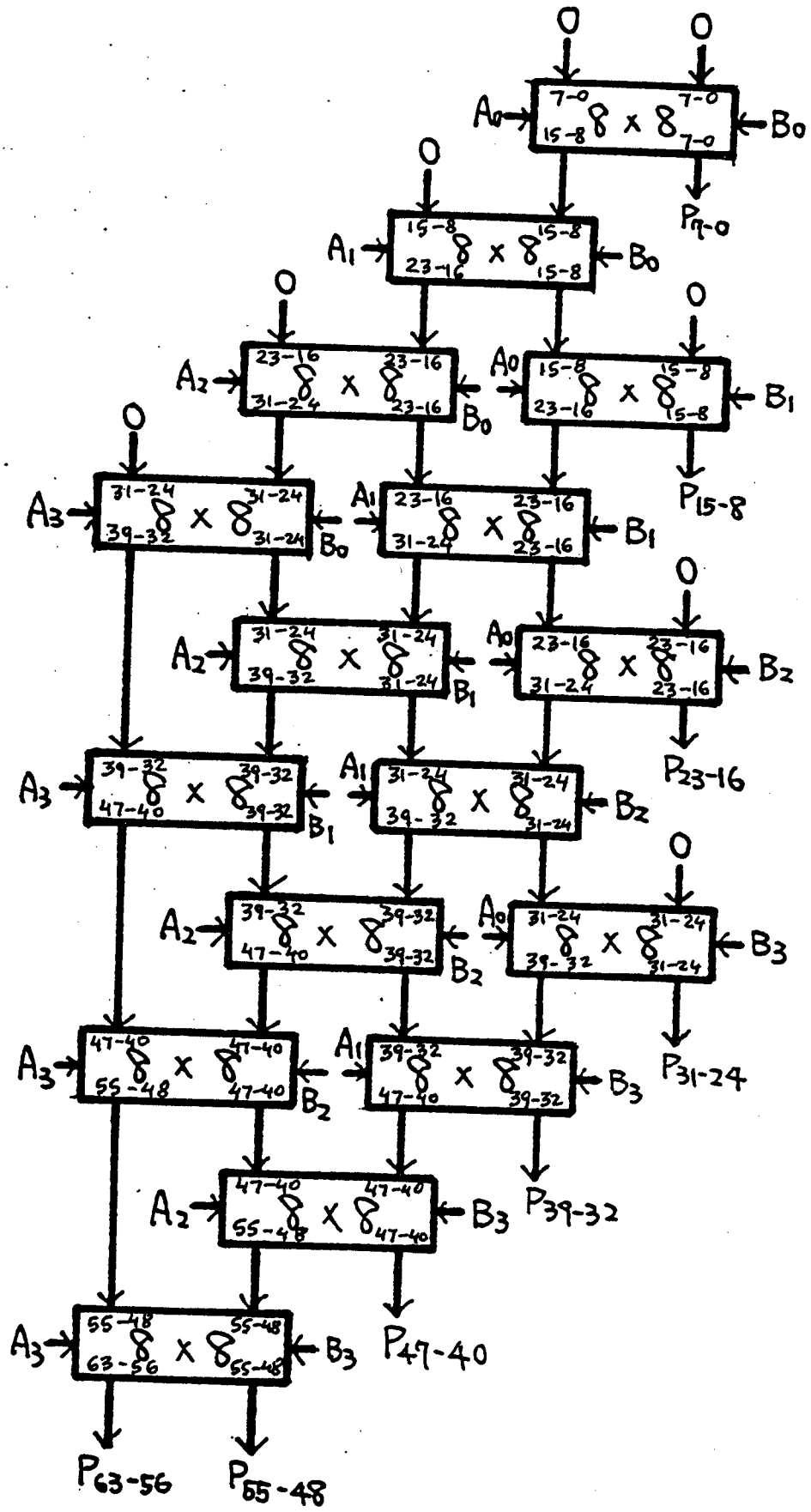
Let the two numbers to be multiplied be

$$A = \underbrace{a_{31}a_{30}\cdots a_{24}}_{A_3}\ \underbrace{a_{23}a_{22}\cdots a_{16}}_{A_2}\ \underbrace{a_{15}a_{14}\cdots a_{8}}_{A_1}\ \underbrace{a_{7}a_{6}\cdots a_{0}}_{A_0}$$

and

$$B = \underbrace{b_{31}b_{30}\cdots b_{24}}_{B_3}\ \underbrace{b_{23}b_{22}\cdots b_{16}}_{B_2}\ \underbrace{b_{15}b_{14}\cdots b_{8}}_{B_1}\ \underbrace{b_{7}b_{6}\cdots b_{0}}_{B_0}$$

The design is shown on next page.


0
0
A0
7-0 8 x 8 7-0
15-8
B0
P7-0
0
A1
15-8 8 x 8 15-8
23-16 15-8
B0
0
0
A2
23-16 8 x 8 23-16
31-24 23-16
B0
A0
15-8 8 x 8 15-8
23-16 15-8
B1
P15-8
0
A3
31-24 8 x 8 31-24
39-32 31-24
B0
A1
23-16 8 x 8 23-16
31-24 23-16
B1
0
A2
31-24 8 x 8 31-24
39-32 31-24
B1
A0
23-16 8 x 8 23-16
31-24 23-16
B2
P23-16
A3
39-32 8 x 8 39-32
47-40 39-32
B1
A1
31-24 8 x 8 31-24
39-32 31-24
B2
0
A2
39-32 8 x 8 39-32
47-40 39-32
B2
A0
31-24 8 x 8 31-24
39-32 31-24
B3
P31-24
A3
47-40 8 x 8 47-40
55-48 47-40
B2
A1
39-32 8 x 8 39-32
47-40 39-32
B3
P39-32
A2
47-40 8 x 8 47-40
55-48 47-40
B3
P47-40
A3
55-48 8 x 8 55-48
63-56 55-48
B3
P63-56
P55-48

Fixed Point Division

$2n$-bit integer dividend $= A = (a_{2n-1} a_{2n-2} \cdots a_0)_2$

n-bit integer divisor $= B = (b_{n-1} \cdots b_1 b_0)_2$

n-bit integer quotient $= Q = (q_{n-1} \cdots q_1 q_0)_2$

n-bit integer remainder $= R = (r_{n-1} \cdots r_1 r_0)_2$

$$A = B \times Q + R$$
$$|R| < |B|$$

- If the numbers are signed, then the remainder R has the same sign as the dividend A. The quotient Q is positive if the dividend A and divisor B are of the same sign. Otherwise Q is negative.

Division Overflow

Consider dividing a $2n$-bit integer dividend A by an n-bit integer divisor B in order to obtain an n-bit quotient Q and an n-bit remainder R.

A division overflow occurs if the resulting quotient Q needs more than n bits for its representation.

Statement: Consider the fixed point division of a $2n$-bit integer unsigned dividend A by an n-bit integer unsigned divisor B which should result in an n-bit quotient Q and an n-bit remainder R. Let A_1 and A_0 be respectively the n-bit leftmost and n-bit rightmost blocks of the dividend A. Then the necessary and sufficient condition for a division overflow to occur is $A_1 \geq B$.

Proof:

The following need to be proven:

(i) If $A_1 \geq B$ then a division overflow occurs.

(ii) If $A_1 < B$ then division overflow does not occur.

Proof of (i):

$$A = \underbrace{a_{2n-1}\, a_{2n-2} \cdots a_n}_{A_1}\ \underbrace{a_{n-1} \cdots a_1\, a_0}_{A_0}$$

Thus $A = A_1 \times 2^n + A_0$ (1)

Also $A = B \times Q + R$ (2).

(1), (2) ⟹ $\boxed{A_1 2^n + A_0 = B \times Q + R}$ (3).

Since $A_1 \geq B$, the minimum of the left side of (3) is $(A_1 2^n + A_0)_{min} = A_{1_{min}} 2^n + A_{0_{min}}$

$= B \times 2^n + 0 = B \times 2^n$ or

$\boxed{\text{min of left side of (3)} = B \times 2^n}$ (4).

Assume that division overflow does not occur. Then Q is an n-bit number or $Q_{maximum} = 2^n - 1$. Therefore, the maximum of the right side of (3) is (as a function of B) $(B \times Q + R)_{max} = B \times Q_{max} + R_{max} =$

$B \times (2^n - 1) + B - 1 = B \times 2^n - B + B - 1 = B \times 2^n - 1$

or

$\boxed{\text{max of right side of (3)} = B \times 2^n - 1}$ (5)

Equations (4), (5) imply

min. of left side of (3) > max. of right side of (3).

Thus, if Q is n-bit long, the max of the right side of (3) can't even reach the min of the left side of (3). Therefore, Q must be longer than n bits which implies a division overflow.

Proof of (ii):

Again (1), (2) and (3) hold true. Since $A_1 < B$, the maximum of the left side of (3) is

$$(A_1 2^n + A_0)_{max} = A_{1max} 2^n + A_{0max} = (B-1)2^n + 2^n - 1$$

$$= B2^n - 2^n + 2^n - 1 \quad \text{or}$$

$$\boxed{\text{max of left side of (3)} = B2^n - 1} \quad (6).$$

Let's find the size of Q that can achieve the same maximum of the right side of (3). Let Q be m-bit long. Then $Q_{max} = 2^m - 1$ and max. of right side of (3) (as a function of B) is

$$(B \times Q + R)_{max} = B \times Q_{max} + R_{max} = B \times (2^m - 1) + B - 1$$

$$= B \times 2^m - B + B - 1 = B \times 2^m - 1 \quad \text{or}$$

$$\boxed{\text{max of right side of (3)} = B2^m - 1} \quad (7)$$

Equations (6), (7) suggest that $m = n$. Therefore, a quotient-length of n bits is sufficient and division overflow does not occur.

Division Techniques

- Restoring division
- Non restoring division

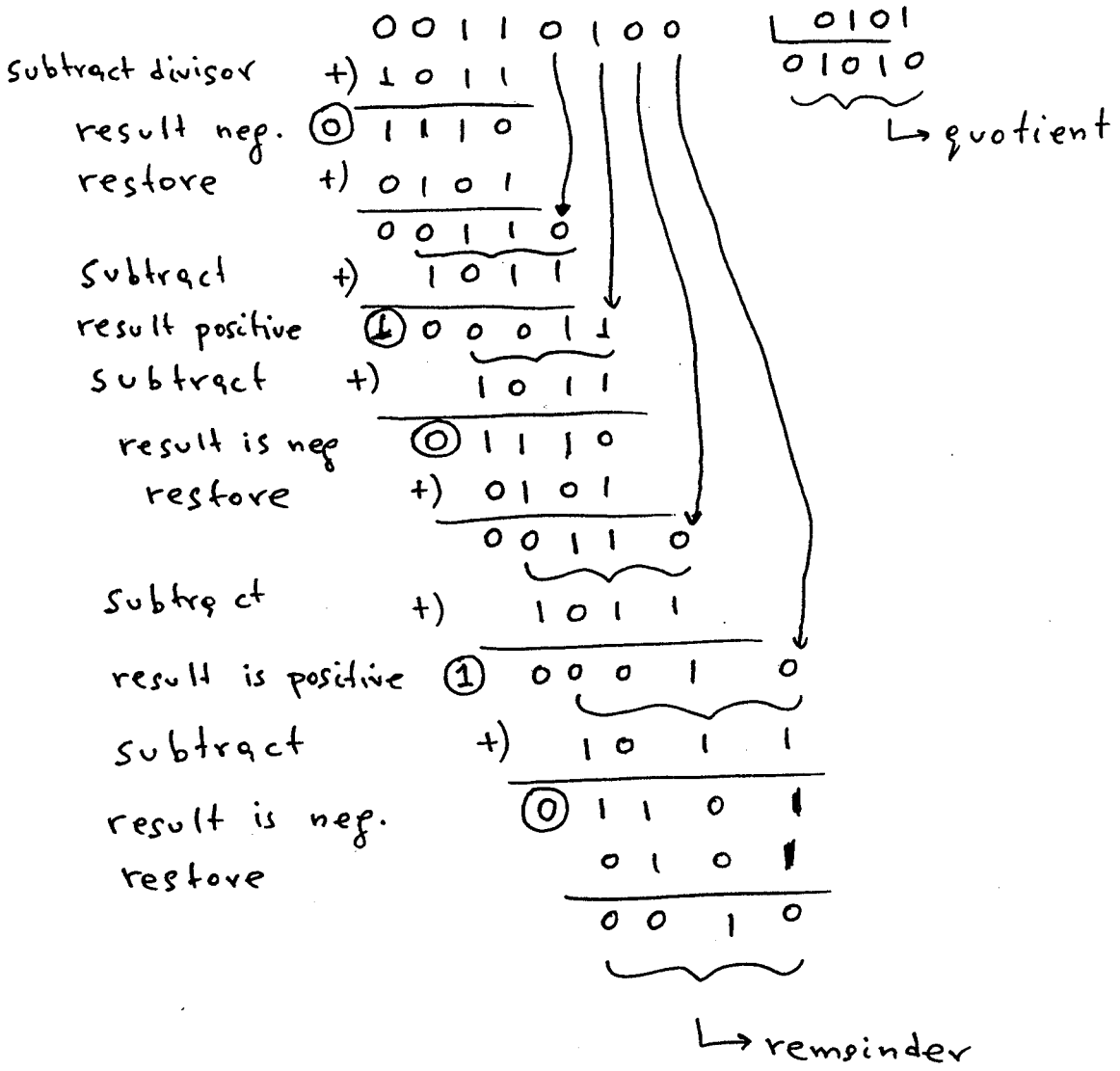
Implementations

- Sequential
- Combinational.

The next page shows the pencil and paper technique for performing the division which illustrates the main concept of the restoring division technique.

⑥ n

Perform the division with dividend $=(00110100)_2 =$ $= 52$ and divisor $=(0101)_2 = 5$.



Sequential Implementation of Restoring Division

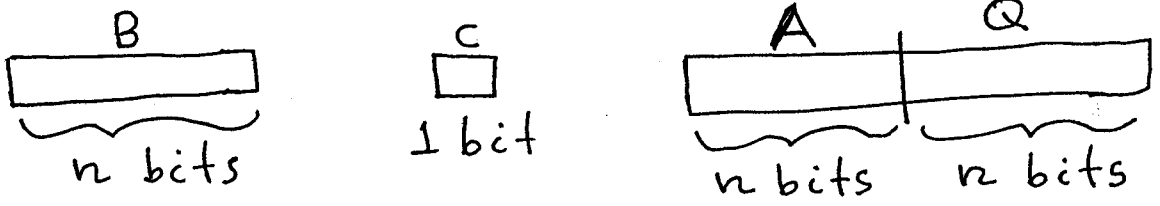
$2n$-bit dividend $= A = (a_{2n-1}\, a_{2n-2} \cdots a_0)_2$

n-bit divisor $= B = (b_{n-1}\, b_{n-2} \cdots b_0)_2$

Both A and B are positive numbers

or $a_{2n-1} = b_{n-1} = 0$

The fields involved are:



The sequential restoring division algorithm follows:

- Initialization: B ← divisor
  A, Q ← dividend
  C ← 0, or 1 (it doesn't matter).

- The algorithm:

Perform n cycles as follows:

(1) Shift A, Q one bit to the left.

(2) Subtract B from A. Place carry out in c. If the produced difference is negative (c=0) then restore it (by adding the divisor B), place the restored result in field A and make the quotient bit to be 0 (zero). The quotient bit is obviously the value of c and can be stored in the rightmost position of the field Q.

Else, if the produced difference is non-negative (c=1), do <u>not</u> restore, place the difference in field A and make the quotient bit equal to 1. Again, the quotient bit is the value of c which can be stored in the rightmost location of field Q.

After n such cycles, field A contains the remainder while Q contains the quotient.

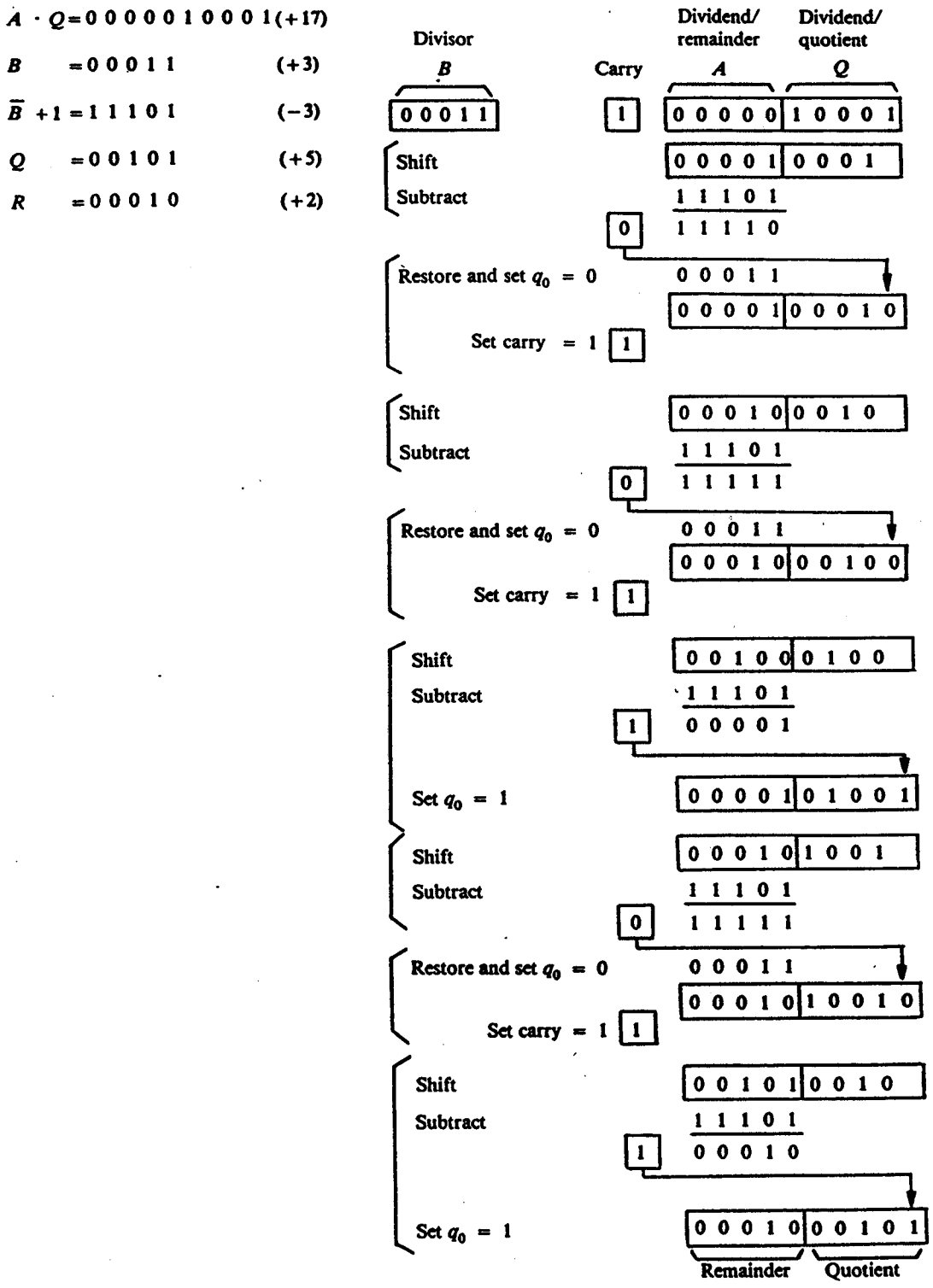


Figure 4.4 Hardware algorithm example with carry FF and separate restore cycle;

$A \cdot Q = 0\,0\,0\,0\,0\,1\,0\,0\,0\,1$ (+17)

$B = 0\,0\,0\,1\,1$ (+3)

$\bar{B} + 1 = 1\,1\,1\,0\,1$ (−3)

$Q = 0\,0\,1\,0\,1$ (+5)

$R = 0\,0\,0\,1\,0$ (+2)

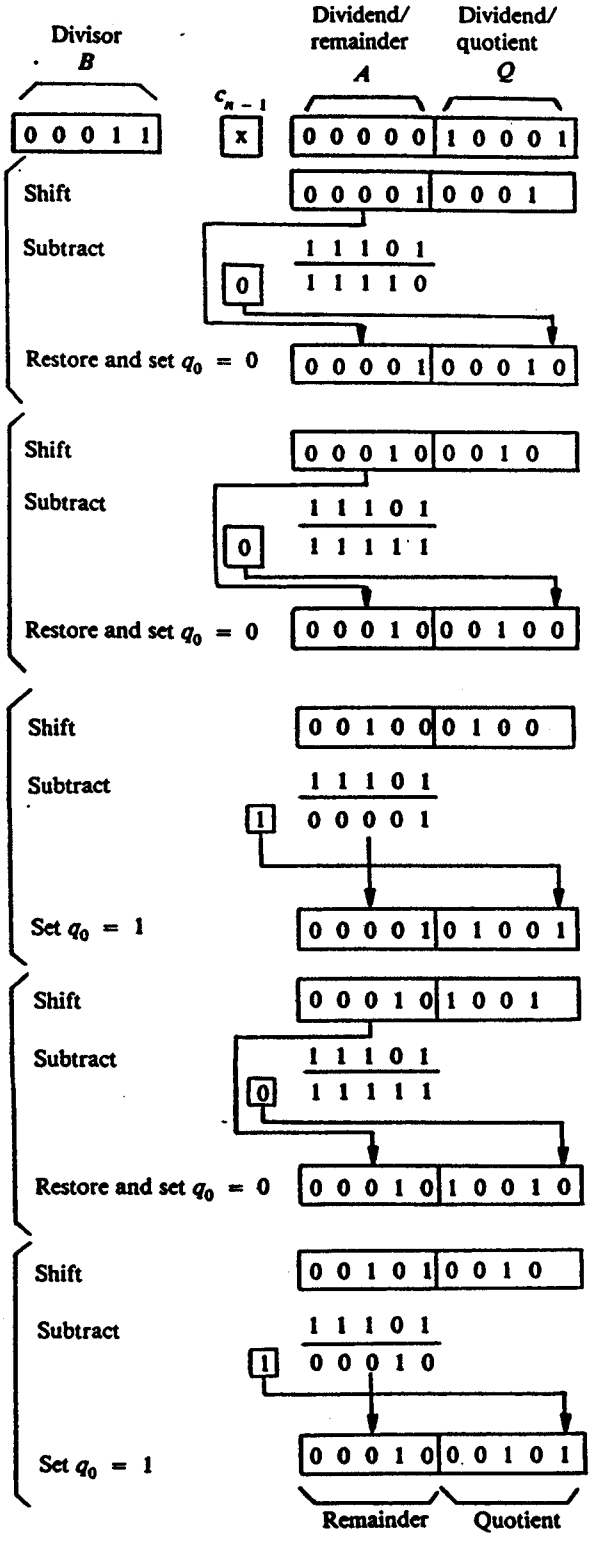
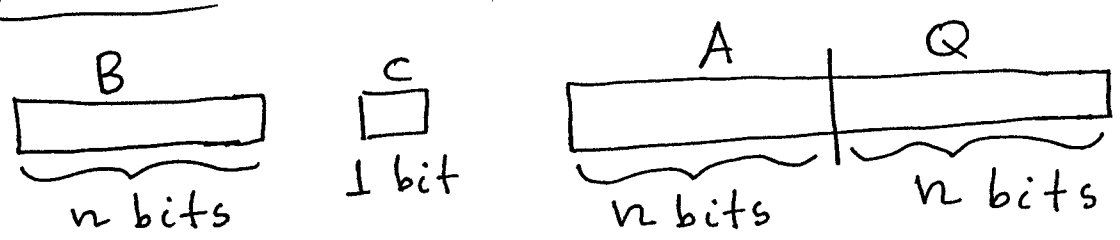


Figure 4.9 Example of restore division without the restoration addition cycle.

Sequential Nonrestoring Division

A = 2n-bit dividend; B = n-bit divisor;
A > 0, B > 0 (or $a_{2n-1} = b_{n-1} = 0$).

Fields:



The sequential nonrestoring division algorithm follows:

- Initialization: B ← divisor; A, Q ← dividend
  C ← 0 or 1 (doesn't matter).
- The algorithm: Perform n cycles as follows:

For 1st cycle

① Shift A, Q one bit to the left
② Subtract B from A. Place difference in field A and carry out in C and the rightmost position of field Q

For cycles 2, 3, 4, ..., n

(1) Shift A, Q one bit to the left.

(2) If previous cycle gave a negative result (c=0) then add B to the field A. Place the summation in field A and the carry out in c and the rightmost location of field Q. Else if previous cycle gave a non-negative result (c=1), then subtract B from A, place difference in field A and carry out in c and the rightmost position of field Q.

- After the completion of the n-th cycle, restore the remainder if the remainder is negative (else don't restore it). This means that if the carry out of the n-th cycle is c=0 then add B to field A and place result in A. Else if c=1 the remainder doesn't need to be restored.

The algorithm returns the remainder in field A and the quotient in field Q.

(13) n

$A \cdot Q = 0\,0\,0\,0\,0\,0\,1\,1\,0\,1$ (+13)

$B = 0\,0\,1\,0\,1$ (+5)

$\overline{B} + 1 = 1\,1\,0\,1\,1$ (−5)

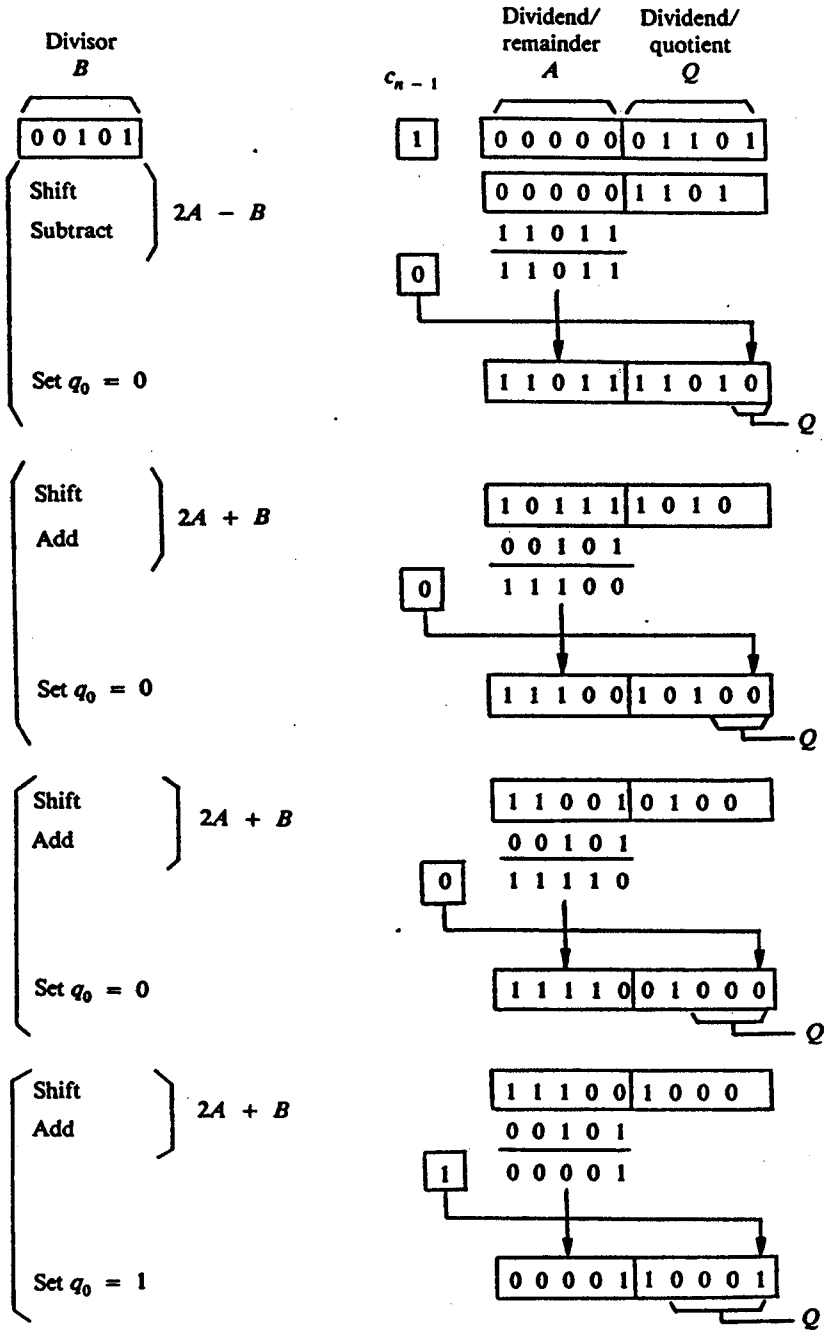


Figure 4.10 Example of nonrestoring division with a final restore cycle.

Continues on next page

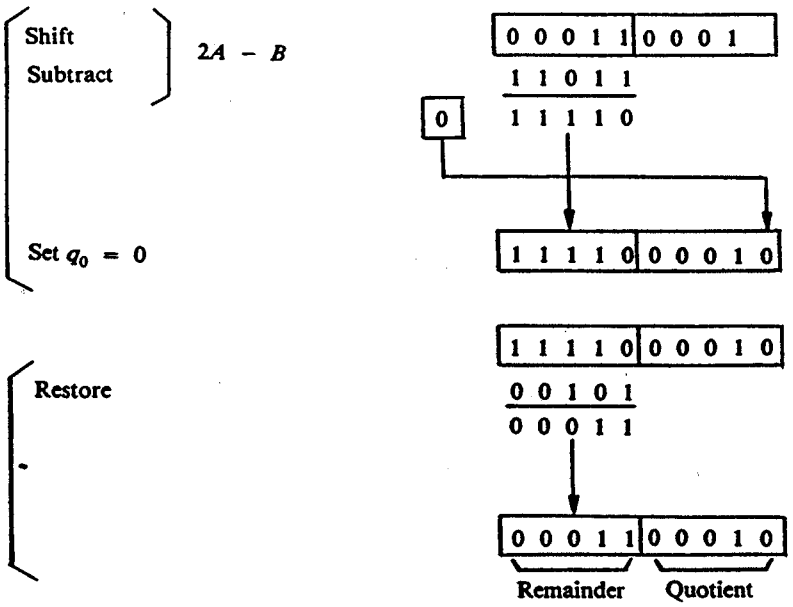


Figure 4.10 (Continued)

(15) n

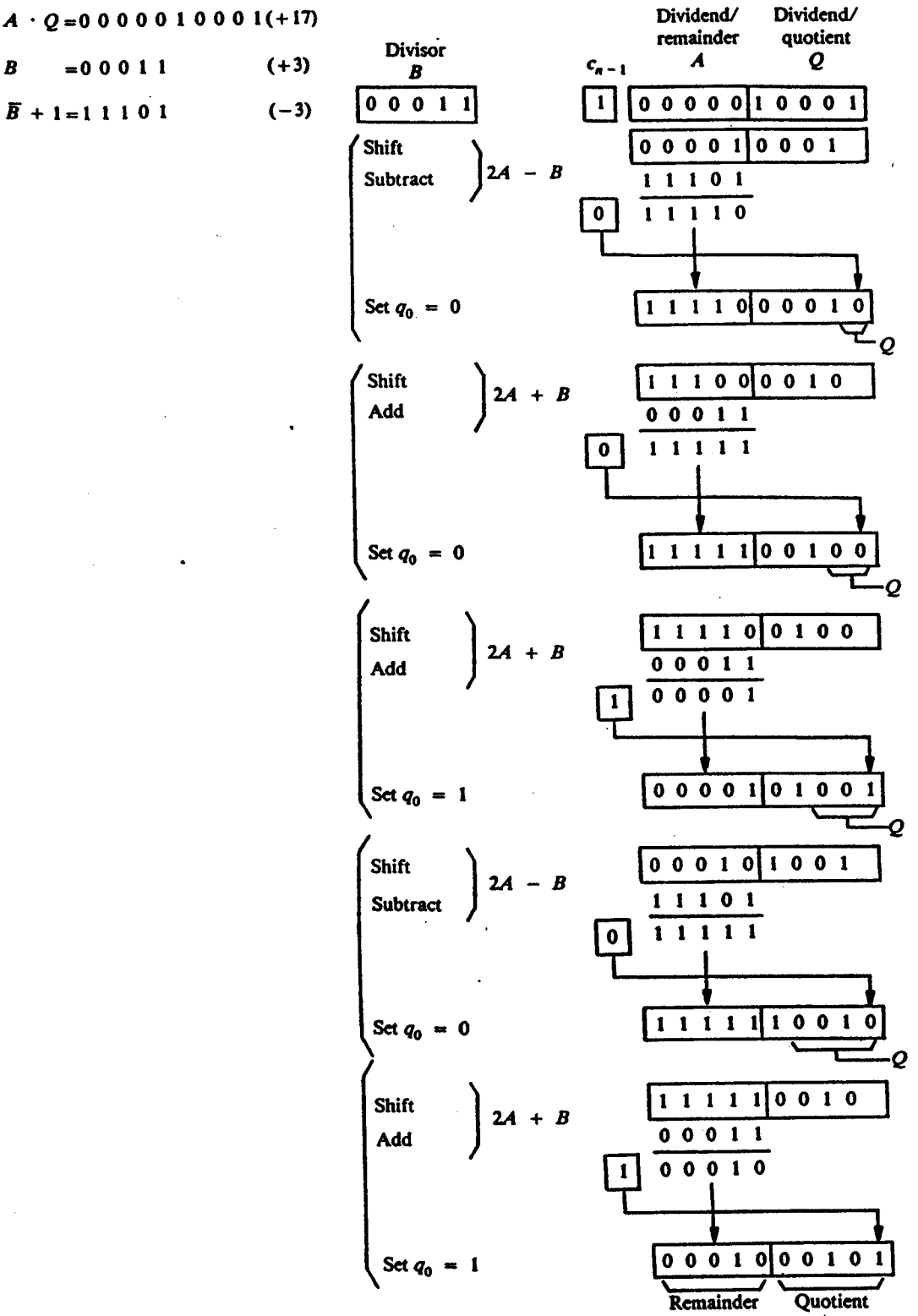
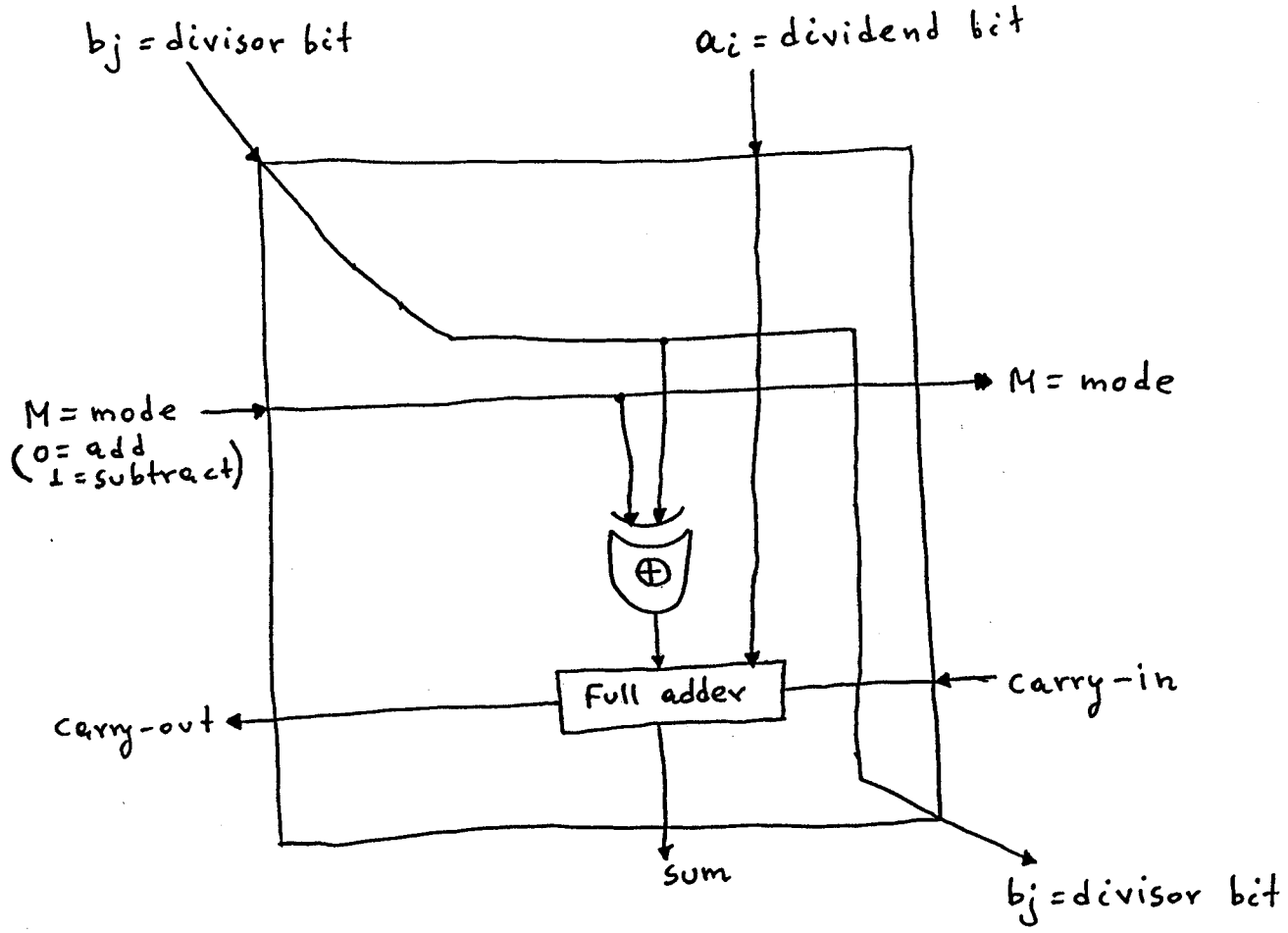


Figure 4.11 Example of nonrestoring division without a final restore cycle.

Nonrestoring array (combinational) dividers



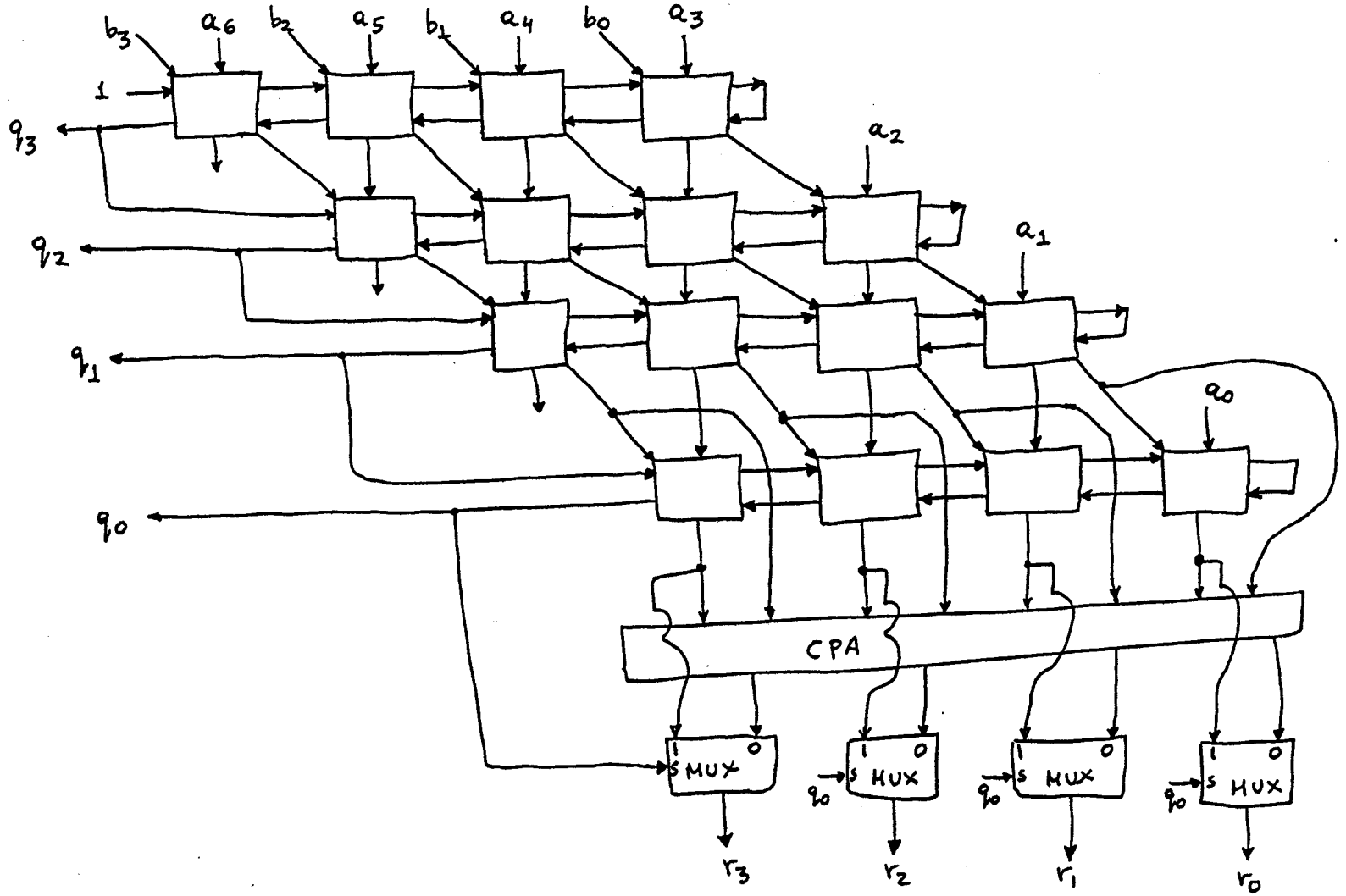
The cell for the nonrestoring array divider

Note: If M = 0 then $a_i + b_j +$ carry-in = (sum, carry out)

If M = 1 then $a_i + \overline{b_j} +$ carry in = (sum, carry out).

(17) 2

A 4-by-4 nonrestoring array divider. Here dividend $A = 0\,a_6\,a_5 \cdots a_0$, divisor $B = b_3\,b_2\,b_1\,b_0$, quotient $Q = q_3\,q_2\,q_1\,q_0$, remainder $R = r_3\,r_2\,r_1\,r_0$.



Multiplicative Division Technique

The Problem: A = dividend; B = divisor.

We want to get an approximation of the quotient Q where $Q \cong \frac{A}{B}$.

• Here A, B are n-bit positive fractions.

•• B = normalized fraction = .1xx…xx.

If B is not normalized then both A and B must be adjusted (ie; $\frac{A}{B} = \frac{.01010}{.01110} = \frac{.10100}{.11100}$).

The main concept: Find a factor F such that $B \times F \cong 1$. Then $Q = \frac{A}{B} = \frac{A \times F}{B \times F} \cong A \times F$

or $\boxed{Q \cong A \times F}$

The factor F is going to be the result of several iterations ($F = F_1 \times F_2 \times F_3 \times \cdots \times F_k$)

or

$$Q = \frac{A}{B} = \frac{A \times F_1}{B \times F_1} = \frac{(A \times F_1) \times F_2}{(B \times F_1) \times F_2} = \frac{(A \times F_1 \times F_2) \times F_3}{(B \times F_1 \times F_2) \times F_3} =$$

$$= \cdots = \frac{(A \times F_1 \times F_2 \times \cdots \times F_{k-1}) \times F_k}{(B \times F_1 \times F_2 \times \cdots \times F_{k-1}) \times F_k}.$$

For some value of k, $B \times \underbrace{F_1 \times F_2 \times \cdots \times F_k}_{F} = .\underbrace{11\ldots11}_{n \text{ ones}} \rightarrow 1$

and thus $A \times \underbrace{F_1 \times F_2 \times \cdots \times F_k}_{F} \rightarrow Q$. Therefore, the technique then stops.

The various iterations give:

Iteration 1: $Q_1 = \frac{A \times F_1}{B \times F_1} = \frac{A_1}{B_1}$ or $\boxed{\begin{aligned} B_1 &= B \times F_1 \\ A_1 &= A \times F_1 \end{aligned}}$

Iteration 2: $Q_2 = \frac{A_1 \times F_2}{B_1 \times F_2} = \frac{A_2}{B_2}$ or $\boxed{\begin{aligned} B_2 &= B_1 \times F_2 \\ A_2 &= A_1 \times F_2 \end{aligned}}$

Iteration 3: $Q_3 = \frac{A_2 \times F_3}{B_2 \times F_3} = \frac{A_3}{B_3}$ or $\boxed{\begin{aligned} B_3 &= B_2 \times F_3 \\ A_3 &= A_2 \times F_3 \end{aligned}}$

$\vdots \qquad \vdots \qquad \vdots$

Iteration k: $Q_k = \frac{A_{k-1} \times F_k}{B_{k-1} \times F_k} = \frac{A_k}{B_k}$ or $\boxed{\begin{aligned} B_k &= B_{k-1} \times F_k \\ A_k &= A_{k-1} \times F_k \end{aligned}}$

In the above, the sequence $B, B_1, B_2, \ldots, B_k$ must be such that $B < B_1 < B_2 < B_3 \cdots < B_k$, (or in general $B_{i-1} < B_i$). Also, $B_k \rightarrow 1$; (the system's precision dictates how close is B_k to 1).

Question: How do we choose the factors $F_1, F_2, F_3, \cdots, F_k$ so that $B \times F \to 1$; $(F = F_1 \times F_2 \times \cdots F_k)$?

Answer: B = normalized fraction $= 0.1xx\cdots xx$ or $\frac{1}{2} \le B < 1$. Thus $\boxed{B = 1-\delta \text{ where } 0 < \delta \le \frac{1}{2}}$

Iteration 1: Choose $F_1 = 1+\delta$. Then

$A_1 = A \times F_1 = A(1+\delta)$ and $\boxed{B_1 = B \times F_1 = (1-\delta)(1+\delta) = 1-\delta^2}$

Since $\delta^2 < \delta$ then $B < B_1$. Observe that $0 < \delta^2 \le \frac{1}{4}$ and thus $B_1 = 1-\delta^2 \ge \frac{3}{4}$ or

$\boxed{B_1 = 0.11xx\cdots xx}$; B_1 has at least two leading ones.

Iteration 2: Next choose $F_2 = 1+\delta^2$. Then

$A_2 = A_1 \times F_2 = A_1 \times (1+\delta^2) = A \times (1+\delta) \times (1+\delta^2)$ and

$\boxed{B_2 = B_1 \times F_2 = (1-\delta^2)(1+\delta^2) = 1-\delta^4}$.

Since $\delta^4 < \delta^2$, (or $1-\delta^2 < 1-\delta^4$), it is implied that $B_1 < B_2$. Observe that $0 < \delta^4 \le \frac{1}{2^4}$ and thus $B_2 = 1-\delta^4 \ge 1 - \frac{1}{2^4} = \frac{2^4-1}{2^4} = .1111$ or

$\boxed{B_2 = 0.1111xx\cdots xx}$; B_2 has at least four leading ones.

Iteration 3: Next choose $F_3 = 1+\delta^4$. Then

$A_3 = A_2 \times F_3 = A(1+\delta)(1+\delta^2)(1+\delta^4)$ and

$$B_3 = B_2 \times F_3 = (1-\delta^4)(1+\delta^4) = 1-\delta^8$$

Obsiously $\delta^8 < \delta^4$ and thus $1-\delta^4 < 1-\delta^8$ or $B_2 < B_3$. Observe that $0 < \delta^8 \leq \frac{1}{2^8}$ and thus

$$B_3 = 1-\delta^8 \geq 1-\frac{1}{2^8} = \frac{2^8-1}{2^8} = \cdot\underbrace{11111111}_{\text{eight ones}} \text{ or}$$

$B_3 = \cdot\underbrace{11111111}_{\text{eight ones}} xx\cdots xx$; B_3 has at least eight leading ones.

Iteration 4: The factor here is

$F_4 = 1+\delta^8$ etc, ...

* If the hardware allows precision of n fractional bits, (fractions A, B are n-bit long) the technique stops whenever

$$B_k = \cdot\underbrace{11\cdots11}_{n \text{ ones}}$$

Problem: The presented technique requires computing $1+\delta$ given that we have $B = 1-\delta$; computing $1+\delta^2$ after we get $B_1 = 1-\delta^2$; computing $1+\delta^4$ after we get $B_2 = 1-\delta^4$; computing $1+\delta^8$ " " " $B_3 = 1-\delta^8$; etc, etc ...

Question: How do we get these $1+\delta$, $1+\delta^2$, $1+\delta^4$, $1+\delta^8$, etc ... ?

Answer:

The problem here is: "given $1-x$ being a positive fraction, compute x or. $1+x$".

Let $1-x$ be $1-x = (.\, y_{n-1}\, y_{n-2} \cdots y_0)_2 \neq 0$

Then $x = (2\text{'s complement of } (1-x)) = (.\, z_{n-1} \cdots z_0)_2$

and (of course) $1+x = 1.\, z_{n-1} \cdots z_1\, z_0$

Proof :

$$\boxed{1 - x + x = 1} \quad (1)$$

$1 - x +$ 2's compl. of $(1-x)$ =

$$\begin{array}{rl} & \cdot\, y_{n-1}\, y_{n-2} \cdots y_1\, y_0 \\ & \cdot\, \overline{y_{n-1}}\, \overline{y_{n-2}} \cdots \overline{y_1}\, \overline{y_0} \\ +) & \qquad\qquad\qquad\qquad 1 \\ \hline & (1.\ 0\ 0 \cdots 0\ 0)_2 \end{array}$$

or

$$\boxed{1 - x + (\text{2's compl. of } (1-x)) = 1} \quad (2)$$

Eqs. (1), (2) imply

$$\boxed{x = \text{2's complement of } (1-x)}$$

Problem: Consider the multiplicative division technique for computing $Q = \frac{A}{B}$. Here A, B are n-bit positive fractions with B being normalized. If the hardware allows precision of 32 fractional bits, what is the maximum number of iterations that need to be performed before the technique stops? Justify your answer. What is the overall factor F in this case?

Solution: The maximum # of iterations to be performed (before the technique stops) is five. Since B is normalized fraction, then $\frac{1}{2} \le B < 1$ and therefore $B = 1-\delta$ where $0 < \delta \le 1/2$. The factors involved are $F_1 = 1+\delta$, $F_2 = 1+\delta^2$, $F_3 = 1+\delta^4$, $F_4 = 1+\delta^8$, $F_5 = 1+\delta^{16}$.

1st iteration gives $B_1 = B \times F_1 = (1-\delta)(1+\delta) = 1-\delta^2$.

2nd " " $B_2 = B_1 \times F_2 = (1-\delta^2)(1+\delta^2) = 1-\delta^4$.

3rd " " $B_3 = B_2 \times F_3 = (1-\delta^4)(1+\delta^4) = 1-\delta^8$.

4th " " $B_4 = B_3 \times F_4 = (1-\delta^8)(1+\delta^8) = 1-\delta^{16}$.

5th " " $B_5 = B_4 \times F_5 = (1-\delta^{16})(1+\delta^{16}) = 1-\delta^{32}$.

Since $0 < \delta \le \frac{1}{2}$ then $0 < \delta^{32} \le \frac{1}{2^{32}}$ and

thus $B_5 = 1-\delta^{32} \ge 1 - \frac{1}{2^{32}} = \frac{2^{32}-1}{2^{32}} = .\underbrace{111\cdots111}_{32 \text{ ones.}}$

Therefore, after at most five iterations the technique stops. The overall factor F is

$F = F_1 \times F_2 \times F_3 \times F_4 \times F_5 = (1+\delta)(1+\delta^2)(1+\delta^4)(1+\delta^8)(1+\delta^{16})$

and $Q \cong A \times F$.