Lecture – 10

$P = \sum_{i=0}^{n-1} 2^i x_i Y$

Combinational Array Multiplier

- Composed of arrays of simple combinational elements, each of which implements an add/sub and shift operation for small slices of the multiplicand operands.

- $X = x_{n-1}x_{n-2}.........x_1x_0$ and $Y = y_{n-1}y_{n-2}.........y_1y_0$ where both X and Y are unsigned integers. Now $P = X \times Y$ can be expressed as $P = \sum_{i=0}^{n-1} 2^i x_i Y$

  which can be rewritten as $P = \sum_{i=0}^{n-1} 2^i \left(\sum_{i=0}^{n-1} x_i y_j 2^j \right)$

- It requires $n \times n$ array of 2-input AND gate.

- The product terms are summed by an array of *n(n-1)* 1-bit full adders.

Multiplication

```
       1000   Multiplicand Y
X      1001   Multiplier X = x3 x2 x1 x0
      -----
       1000
      0000
     0000
    1000
   -------
   1001000   Product
```

Multiplicand Y

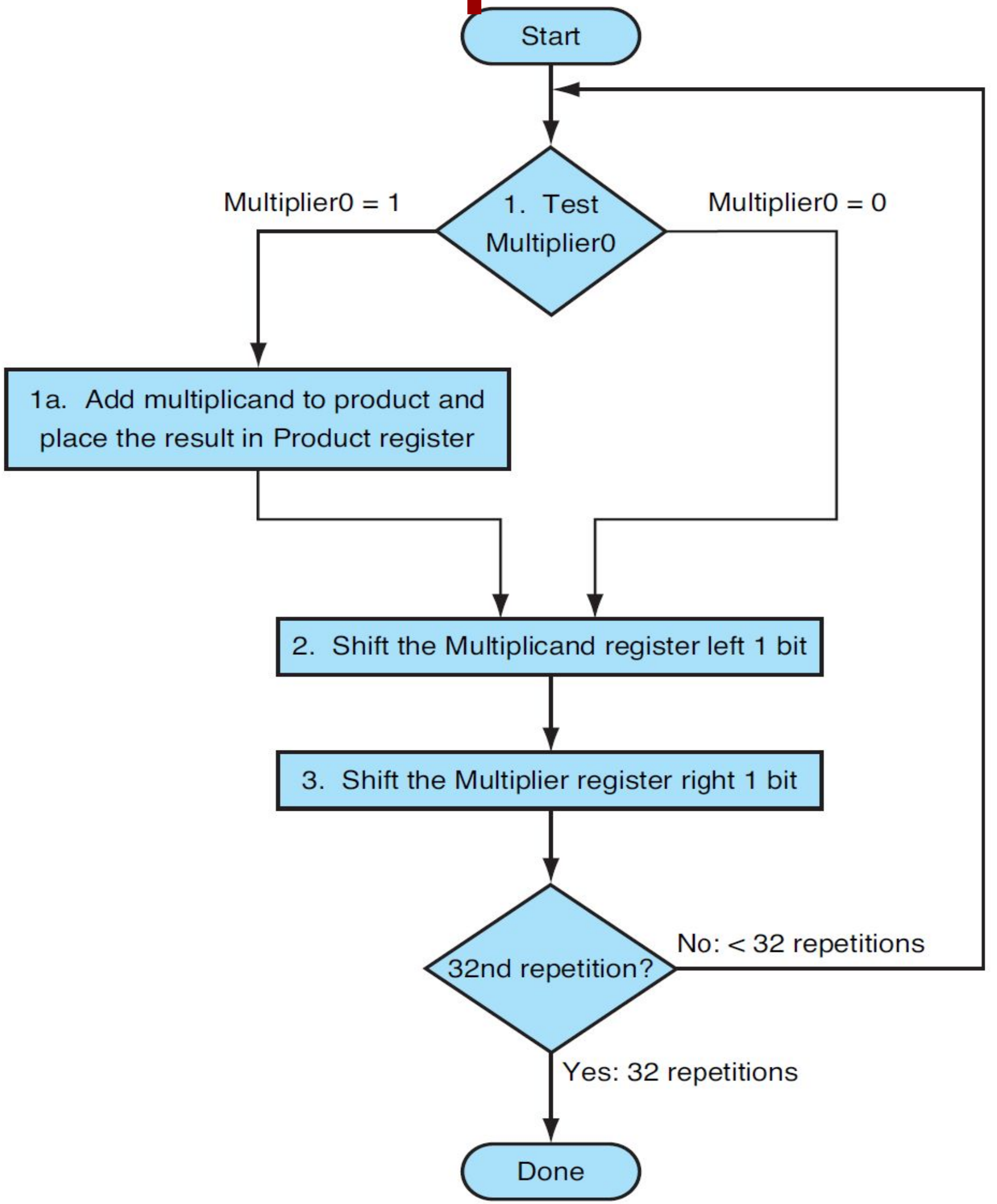
Multiplier $X = x_3 x_2 x_1 x_0$

Product $P = \sum_{j=0}^{3} x_j 2^j Y$

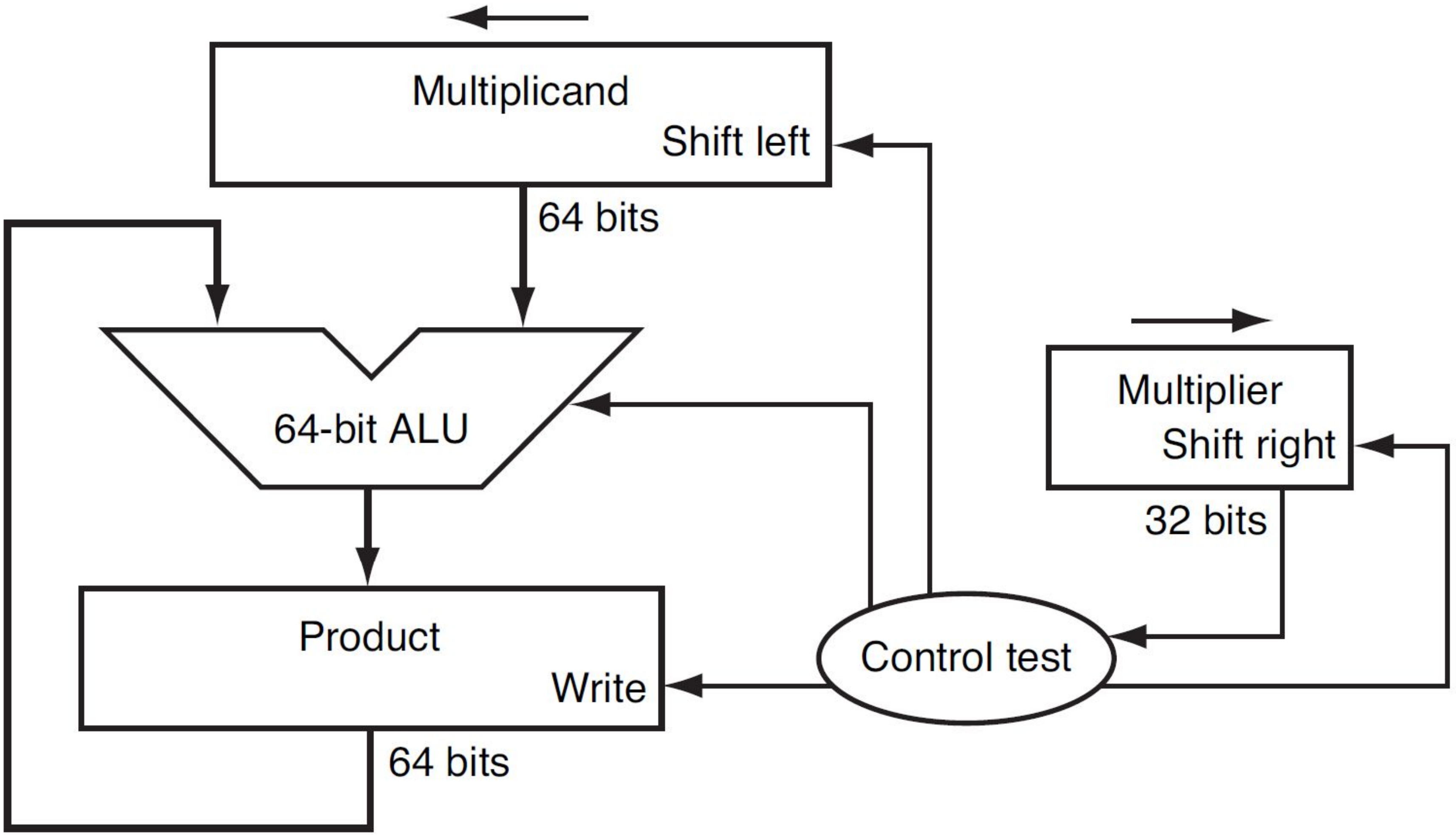
$$P_{i+1} = P_i + x_j 2^i Y$$

- If multiplicand = ***n*** bits and multiplier = ***m*** bits then product = ***n* + *m*** bits.
- Two rules:
  - Place a copy of multiplicand in the proper place if multiplier bit=1.
  - Place 0 in the proper place if multiplier bit = 0.

Sequential Multiplication Algorithm



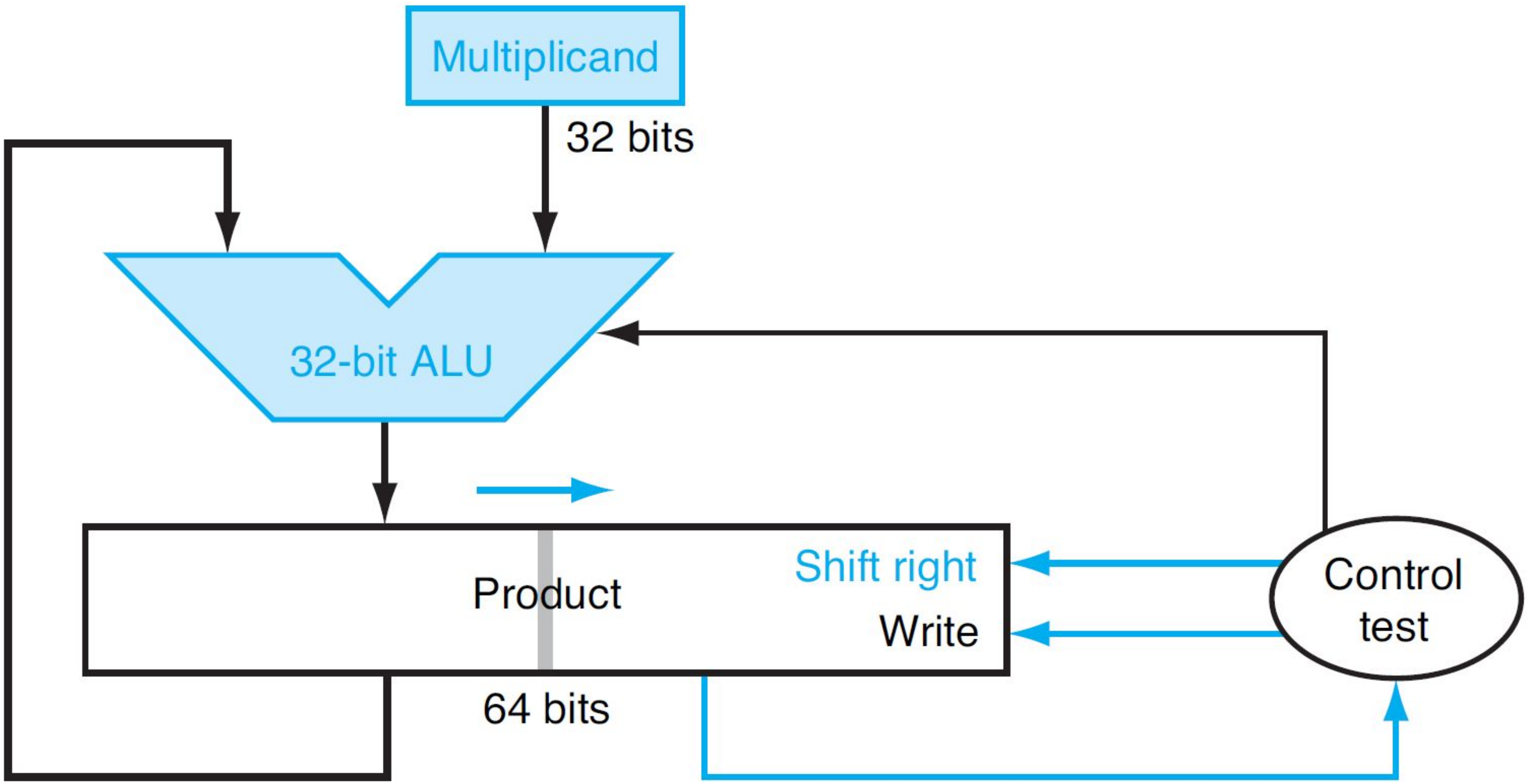
Sequential Multiplication Hardware



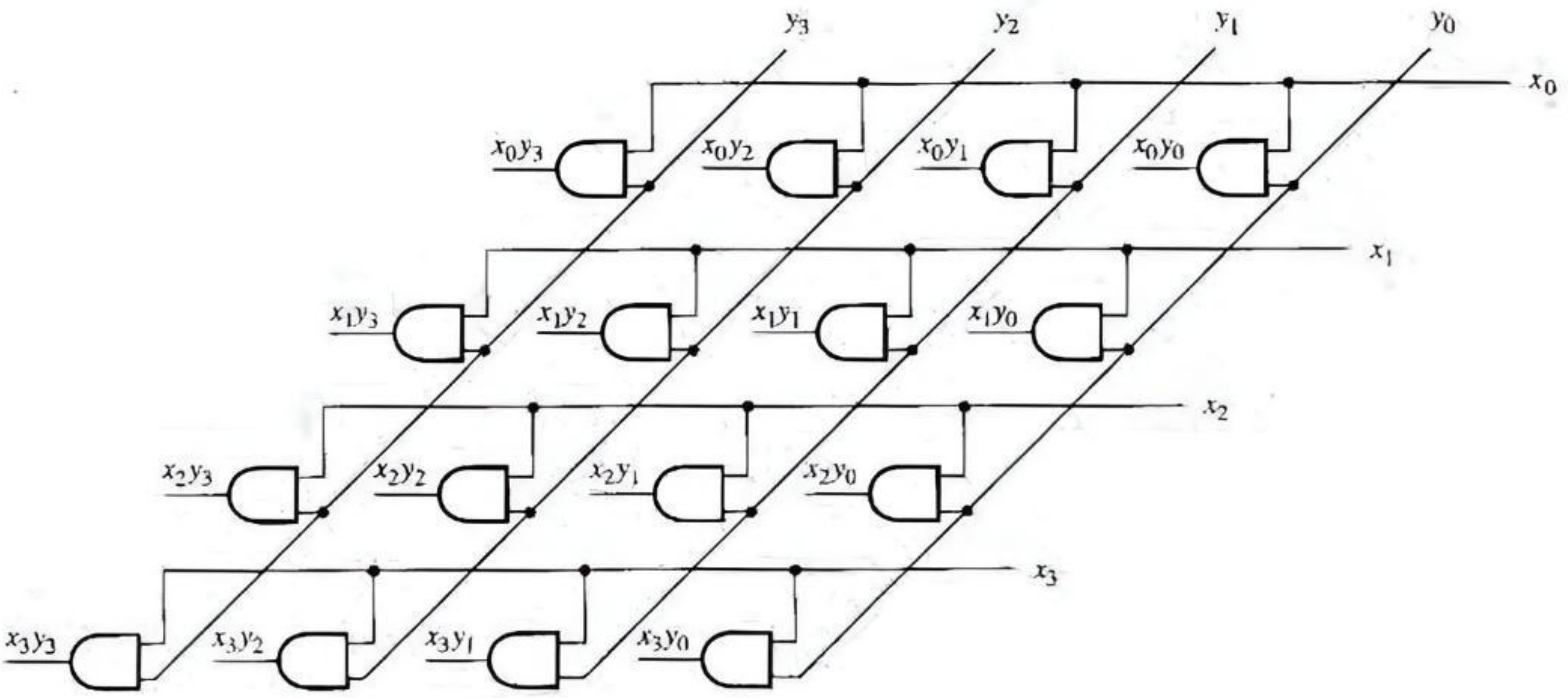
Sequential Multiplication Example

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	0011	0000 0010	0000 0000
1	1a: 1 ⇒ Prod = Prod + Mcand	0011	0000 0010	0000 0010
	2: Shift left Multiplicand	0011	0000 0100	0000 0010
	3: Shift right Multiplier	0001	0000 0100	0000 0010
2	1a: 1 ⇒ Prod = Prod + Mcand	0001	0000 0100	0000 0110
	2: Shift left Multiplicand	0001	0000 1000	0000 0110
	3: Shift right Multiplier	0000	0000 1000	0000 0110
3	1: 0 ⇒ No operation	0000	0000 1000	0000 0110
	2: Shift left Multiplicand	0000	0001 0000	0000 0110
	3: Shift right Multiplier	0000	0001 0000	0000 0110
4	1: 0 ⇒ No operation	0000	0001 0000	0000 0110
	2: Shift left Multiplicand	0000	0010 0000	0000 0110
	3: Shift right Multiplier	0000	0010 0000	0000 0110

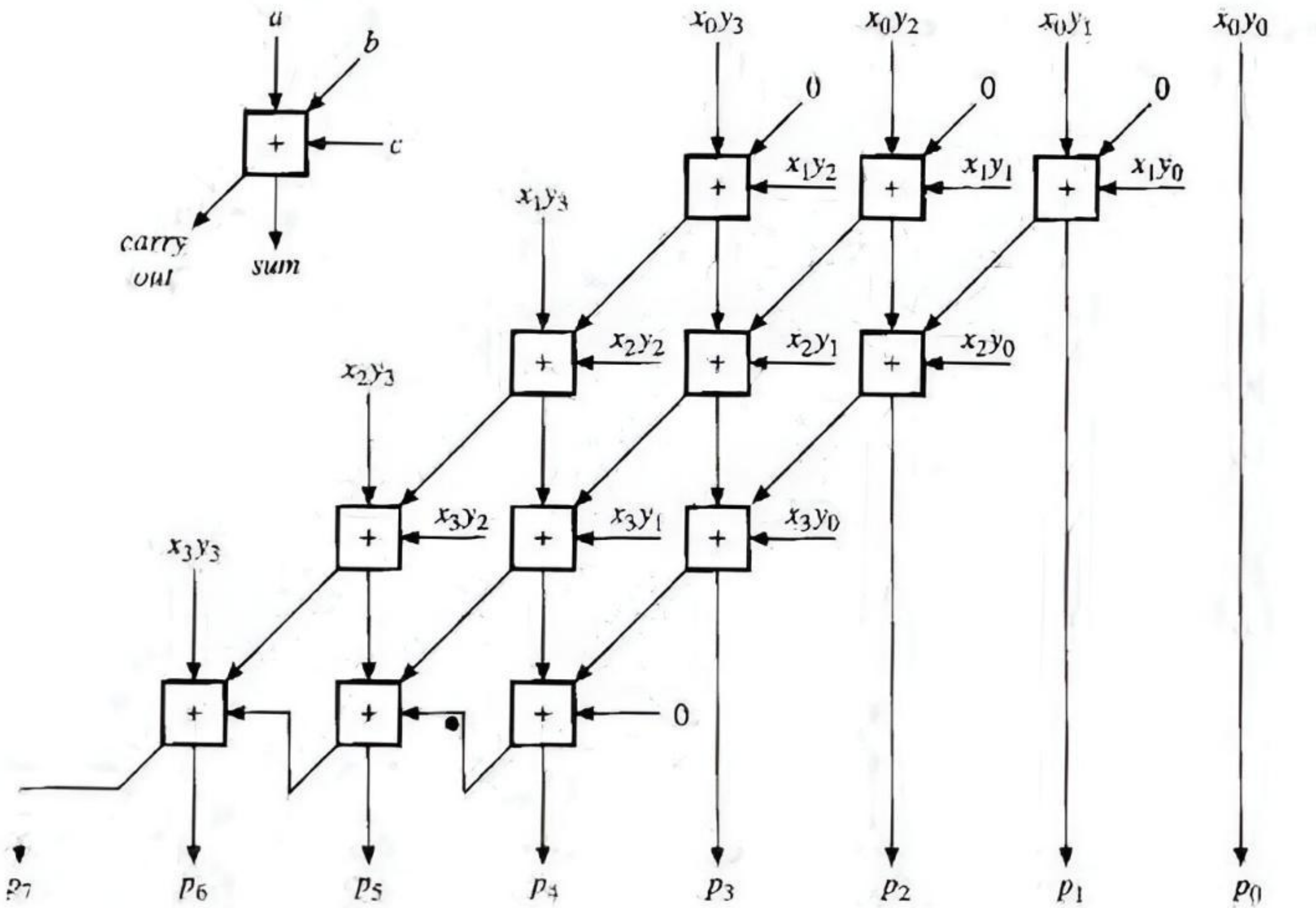
Refined Version of Multiplication Hardware



AND Array for 4 X 4 bit Unsigned Multiplication



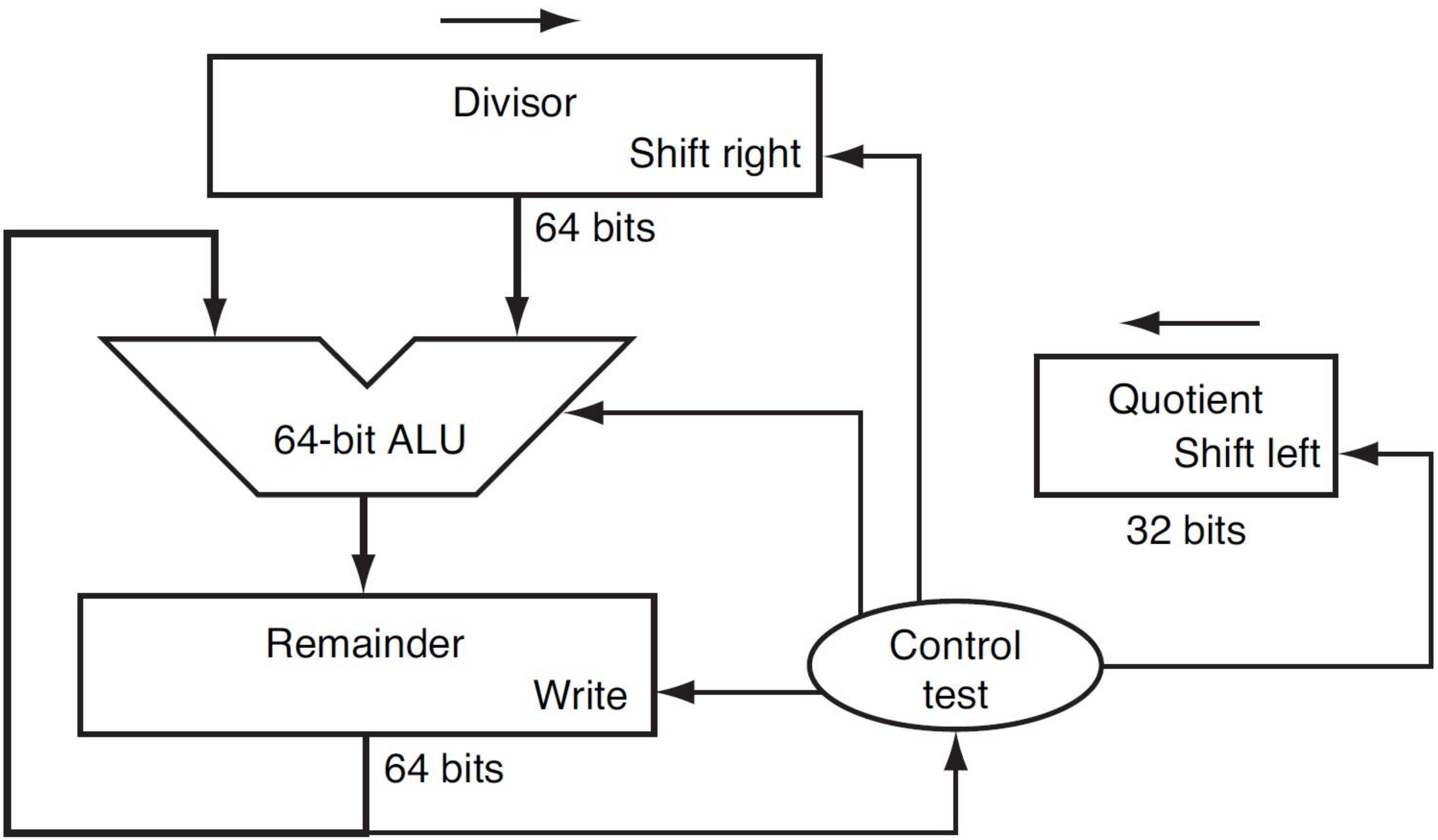
Full Adder Array for 4 X 4 bit Unsigned Multiplication



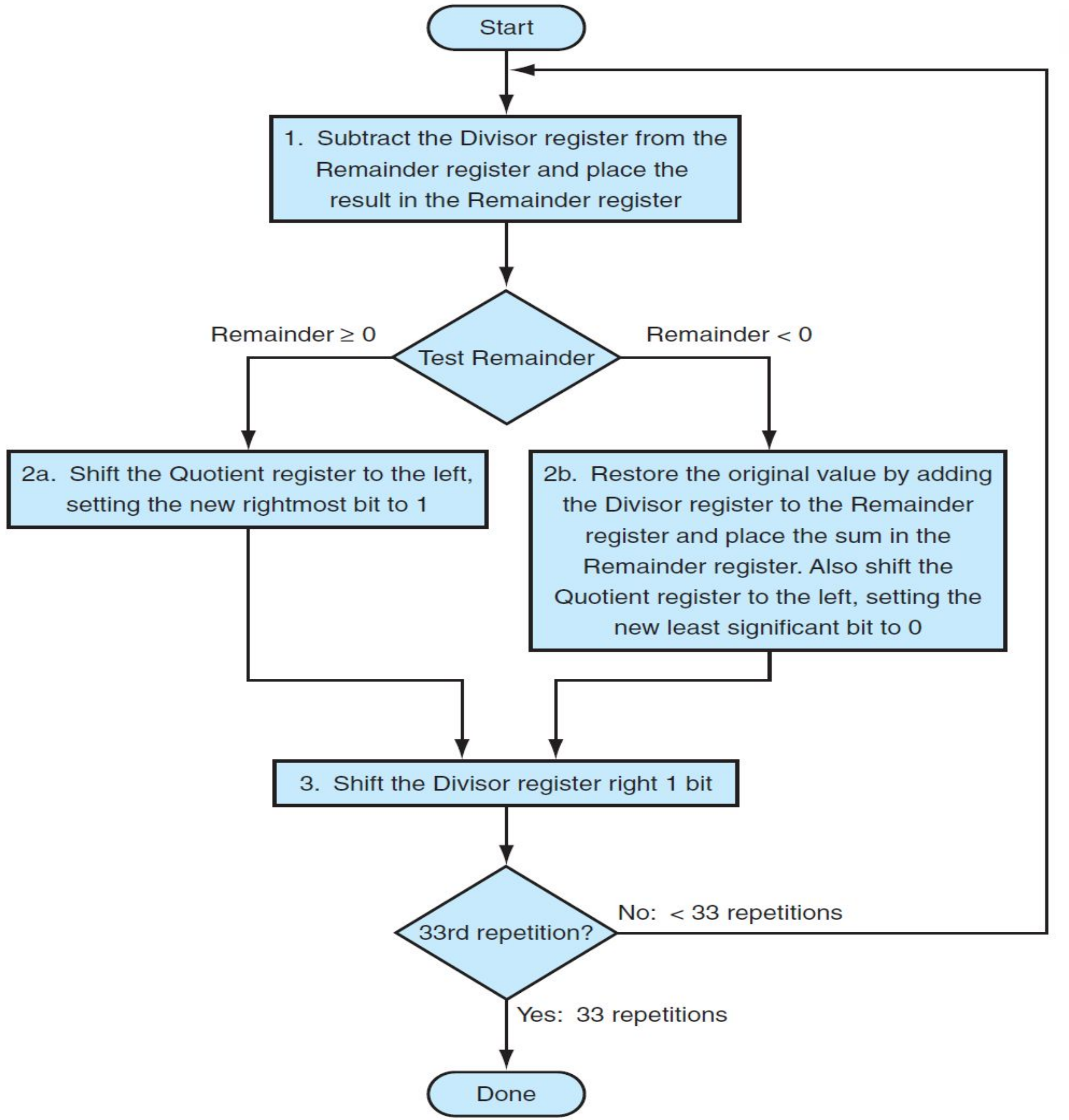
Division

$$
\begin{array}{rrl}
 & 1001_{ten} & \text{Quotient} \\
\text{Divisor } 1000_{ten} & \big)\overline{1001010_{ten}} & \text{Dividend} \\
 & \underline{-1000\phantom{000_{ten}}} & \\
 & 10\phantom{00_{ten}} & \\
 & 101\phantom{0_{ten}} & \\
 & 1010\phantom{_{ten}} & \\
 & \underline{-1000\phantom{_{ten}}} & \\
 & 10_{ten} & \text{Remainder}
\end{array}
$$

Division Hardware



Division Algorithm



Division Example

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem – Div	0000	0010 0000	1110 0111
	2b: Rem < 0 $\Longrightarrow$ +Div, sll Q, Q0 = 0	0000	0010 0000	0000 0111
	3: Shift Div right	0000	0001 0000	0000 0111
2	1: Rem = Rem – Div	0000	0001 0000	1111 0111
	2b: Rem < 0 $\Longrightarrow$ +Div, sll Q, Q0 = 0	0000	0001 0000	0000 0111
	3: Shift Div right	0000	0000 1000	0000 0111
3	1: Rem = Rem – Div	0000	0000 1000	1111 1111
	2b: Rem < 0 $\Longrightarrow$ +Div, sll Q, Q0 = 0	0000	0000 1000	0000 0111
	3: Shift Div right	0000	0000 0100	0000 0111
4	1: Rem = Rem – Div	0000	0000 0100	0000 0011
	2a: Rem $\geq$ 0 $\Longrightarrow$ sll Q, Q0 = 1	0001	0000 0100	0000 0011
	3: Shift Div right	0001	0000 0010	0000 0011
5	1: Rem = Rem – Div	0001	0000 0010	0000 0001
	2a: Rem $\geq$ 0 $\Longrightarrow$ sll Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001