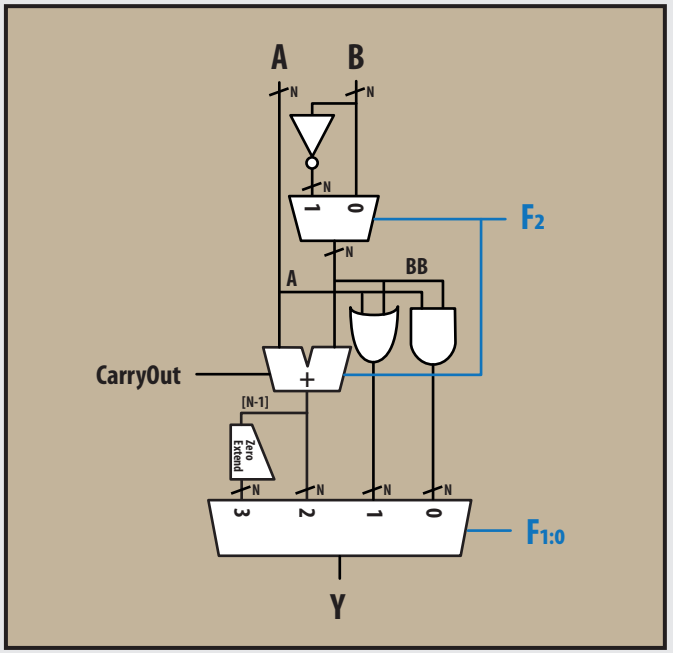
Computer Architecture

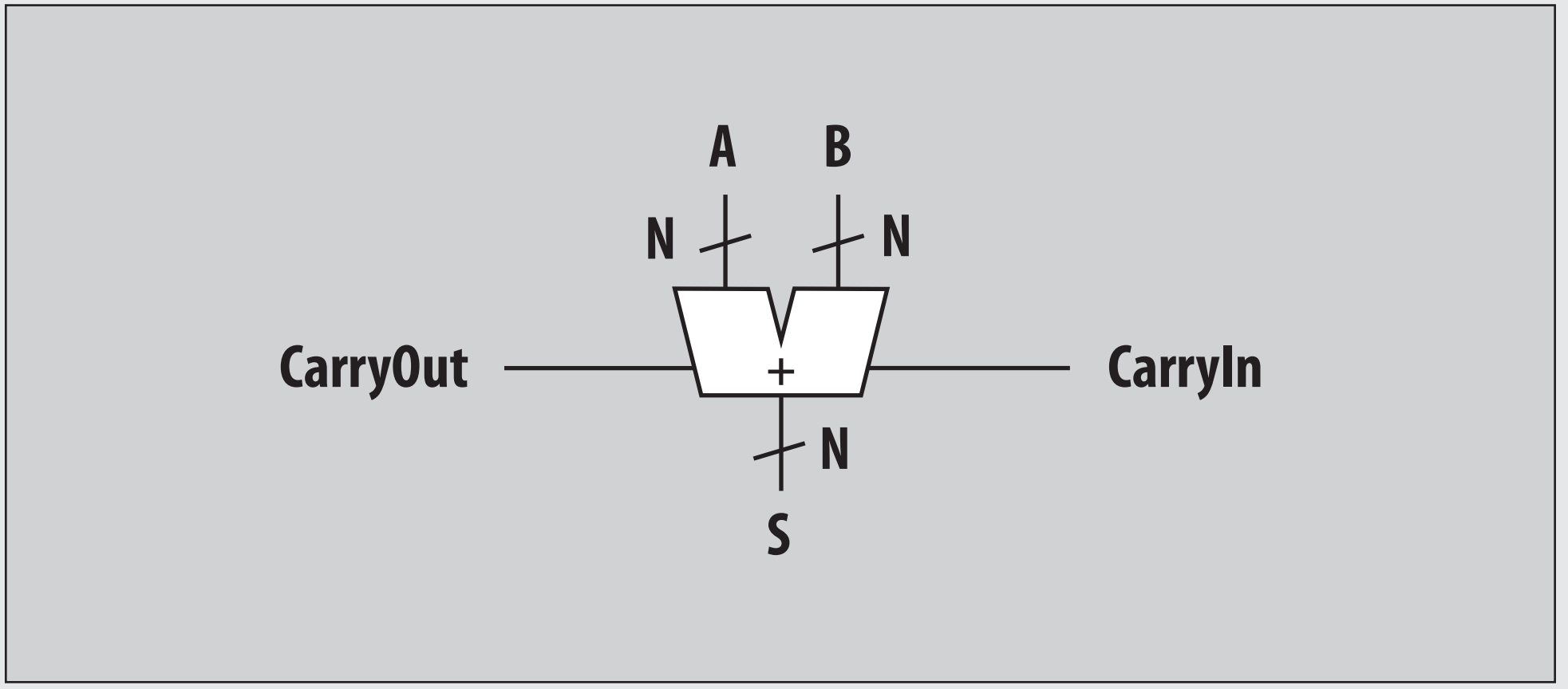
Paul Mellies

Lecture 12 : Digital Building Blocks



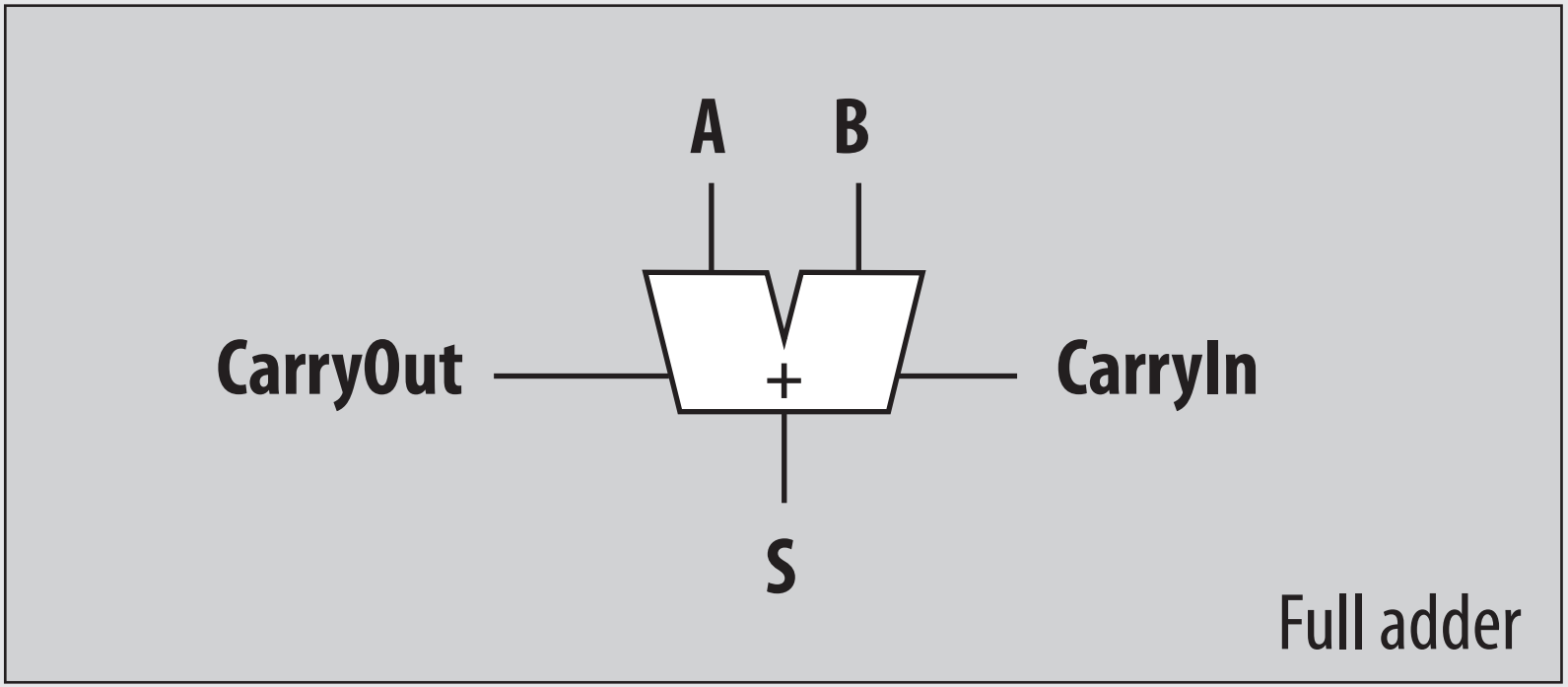
Arithmetic Circuits

Carry Propagate Adder



Carry Propagate Adder

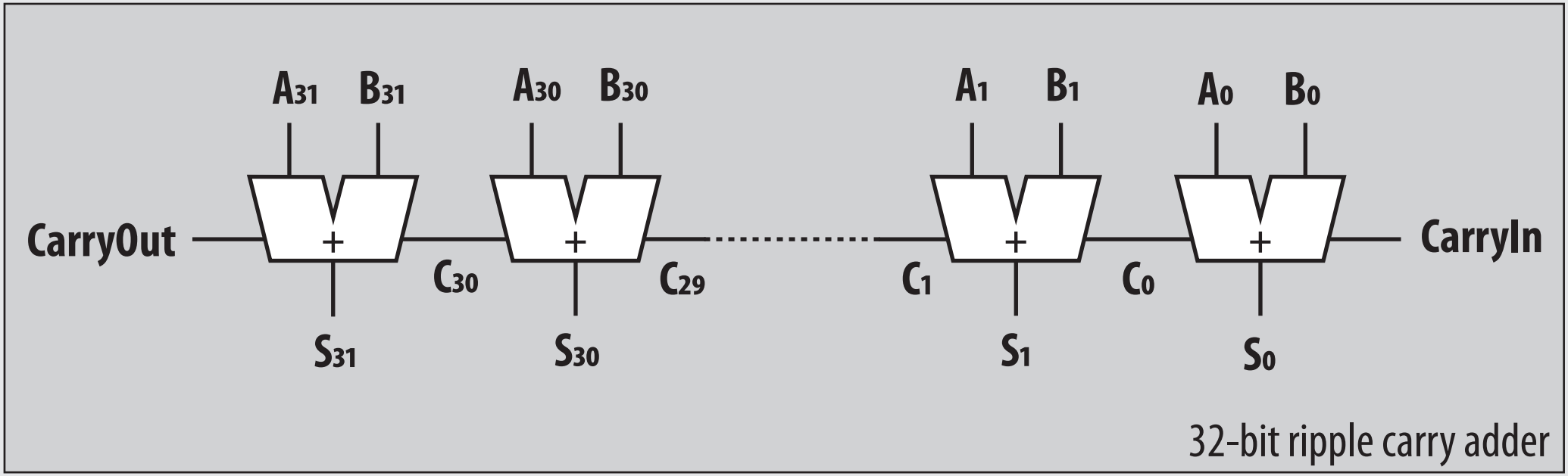
Full Adder



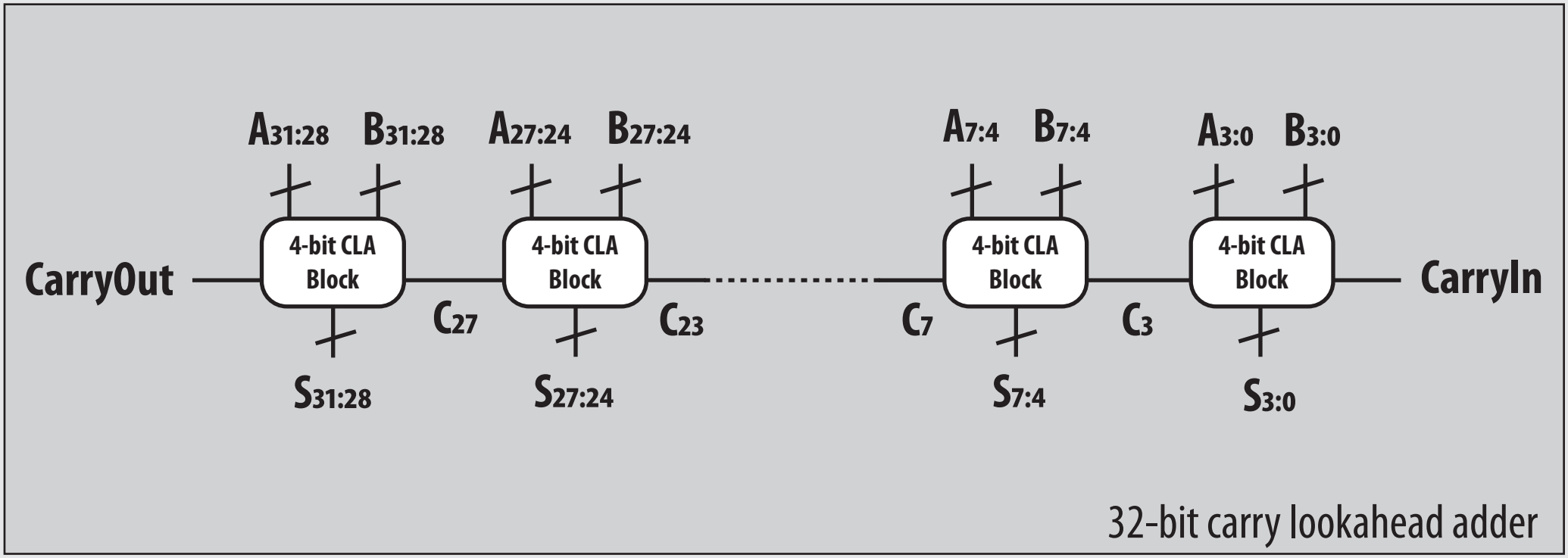
$$S = A \oplus B \oplus CarryIn$$

$$CarryOut = A.B + A.CarryIn + B.CarryIn$$

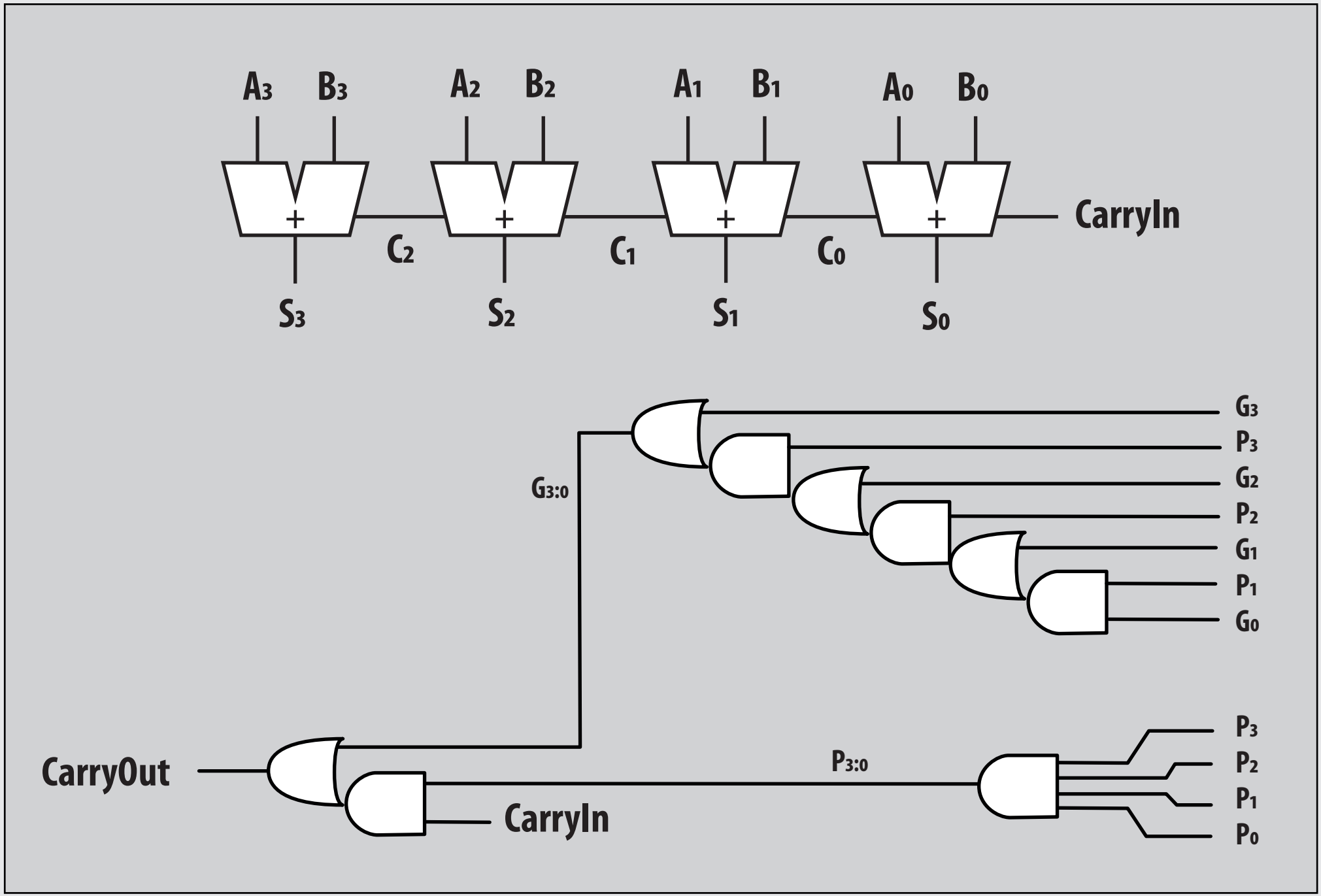
Ripple Carry Adder



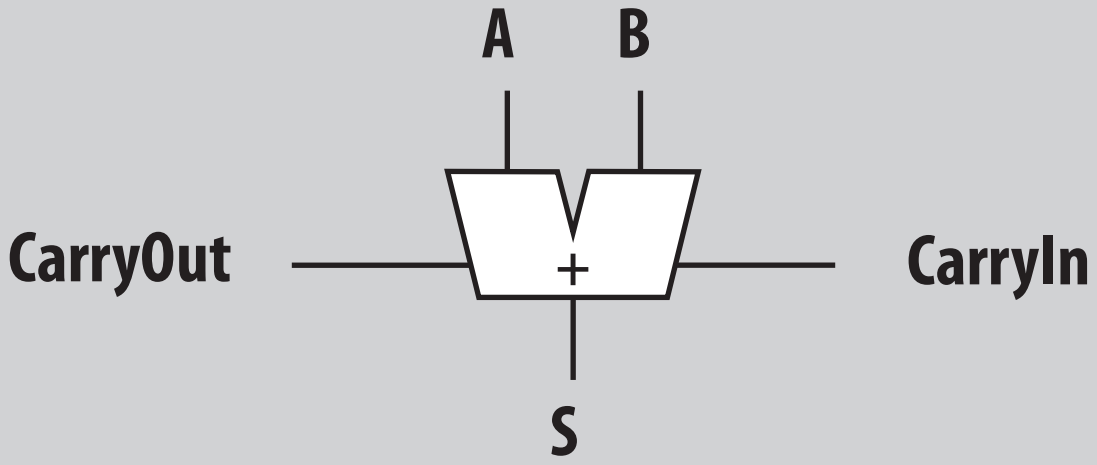
Carry Lookahead Adder



4-bit CLA Block



Generate (G) and Propagate (P) signals



$$G = A \cdot B$$

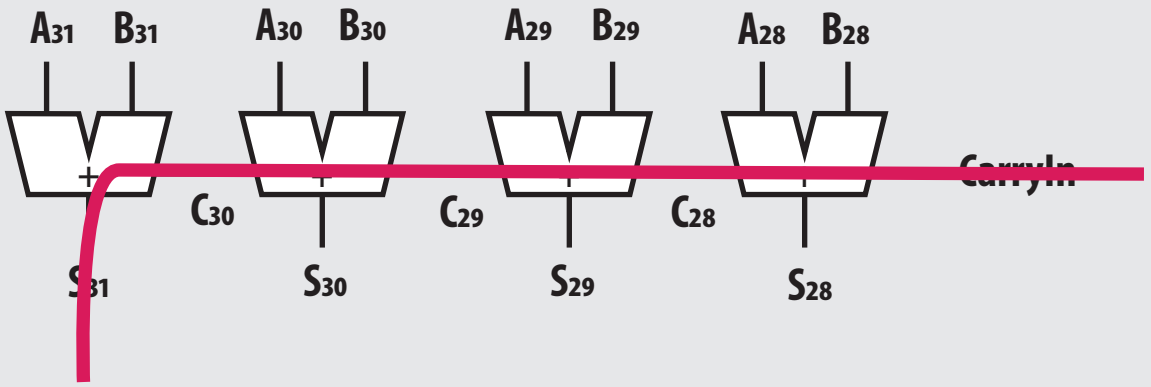
$$P = A + B$$

G = 1 when the adder ***generates*** a CarryOut whatever the value of CarryIn

P = 1 when the adder ***propagates*** the CarryIn to the CarryOut

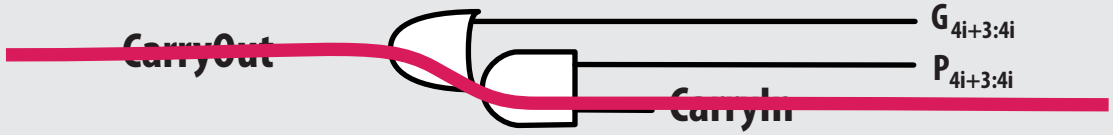
Critical path of the Carry Lookahead Adder

On the lefthand side 4-bit CLA block :
(most significant bits from S_{31} to S_{28})

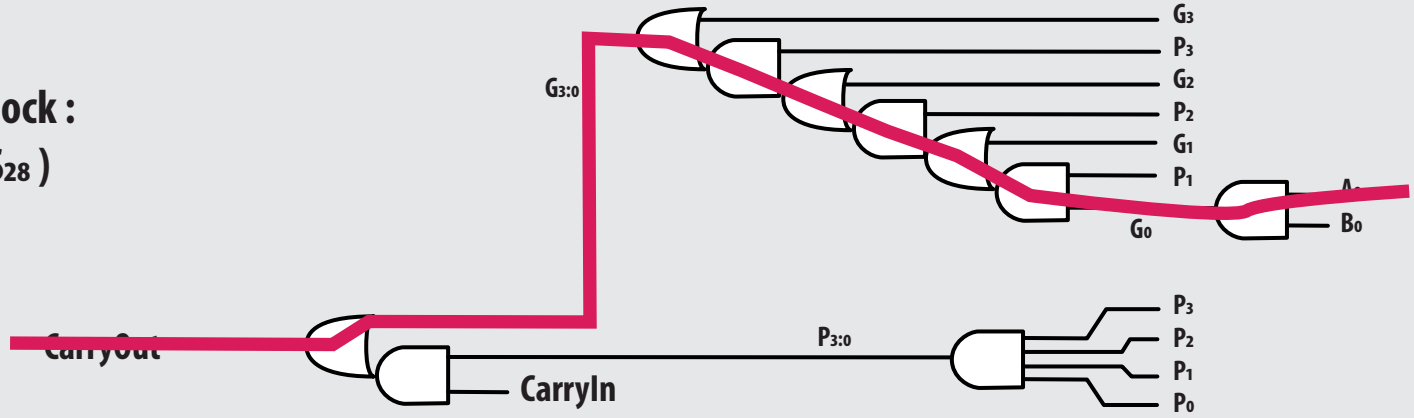


On the six internal 4-bit CLA blocks :

i = 6 , 5 , 4 , 3 , 2 , 1



On the righthand side 4-bit CLA block :
(less significant bits from S_{31} to S_{28})



Propagation Delay

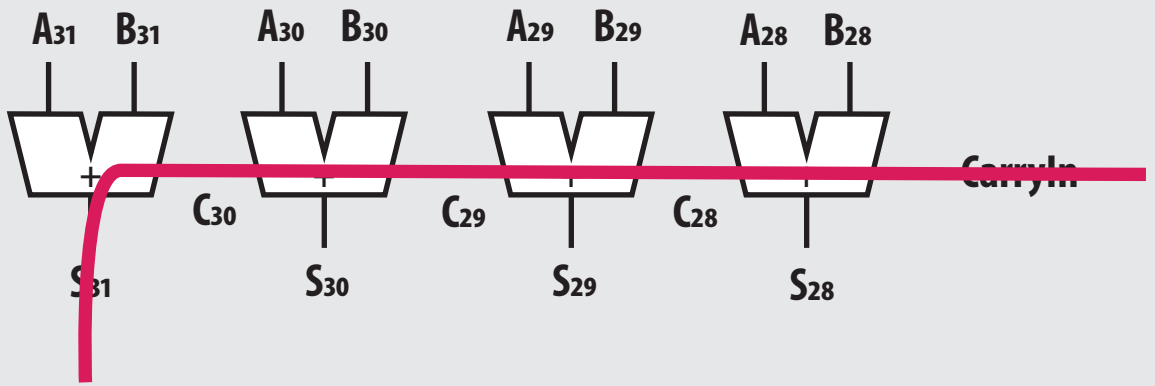
$4\,t_{FULL\_ADDER}$

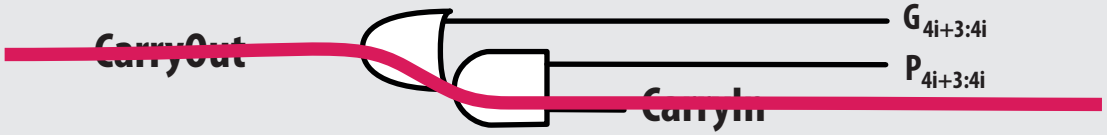
+

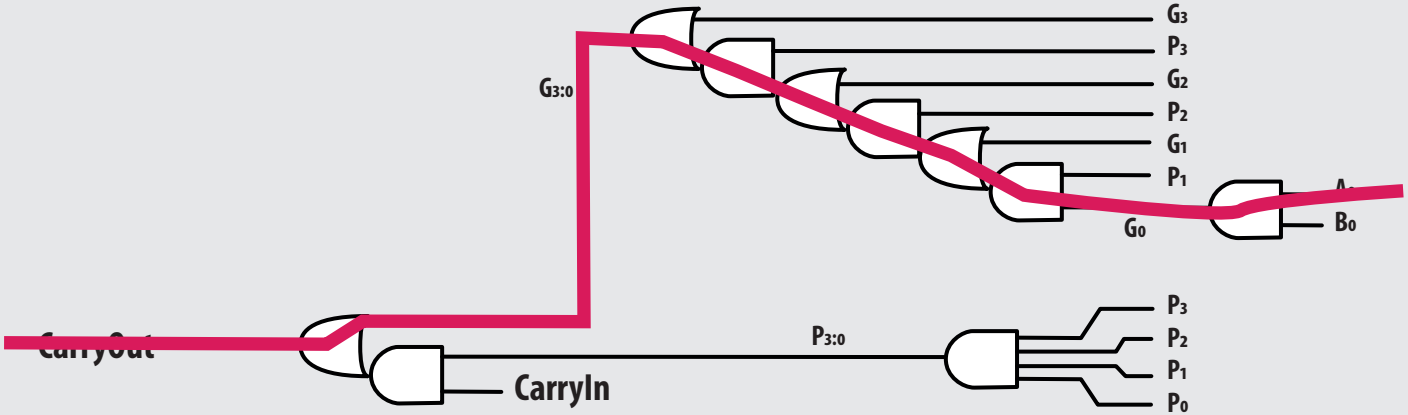
$6 \times (t_{AND} + t_{OR})$

+

$4\,t_{AND} + 4\,t_{OR}$







$$t_{CLA} = 10\,t_{AND} + 10\,t_{OR} + 4\,t_{FULL\_ADDER}$$

Exercise (H&H Example 5.1)

Compare the delays of a 32-bit ripple-carry adder and a 32-bit carry-lookahead adder with 4-bit blocks.

Assume that each two-input gate delay is 100 ps and that a full adder delay is 300 ps.

Solution (H&H Example 5.1)

Compare the delays of a 32-bit ripple-carry adder and a 32-bit carry-lookahead adder with 4-bit blocks.

Assume that each two-input gate delay is 100 ps and that a full adder delay is 300 ps.

32-bit ripple-carry adder:
32 x 300 ps = 9 600 ps = 9.6 ns

32-bit carry-lookahead adder:
10 x 100 ps + 10 x 100 ps + 4 x 300 ps = 3200 ps = 3.2 ns

Carry Lookahead Adder
(summary)

The 32-bit Carry Lookahead Adder is faster
than the 32-bit Ripple Carry Adder because :

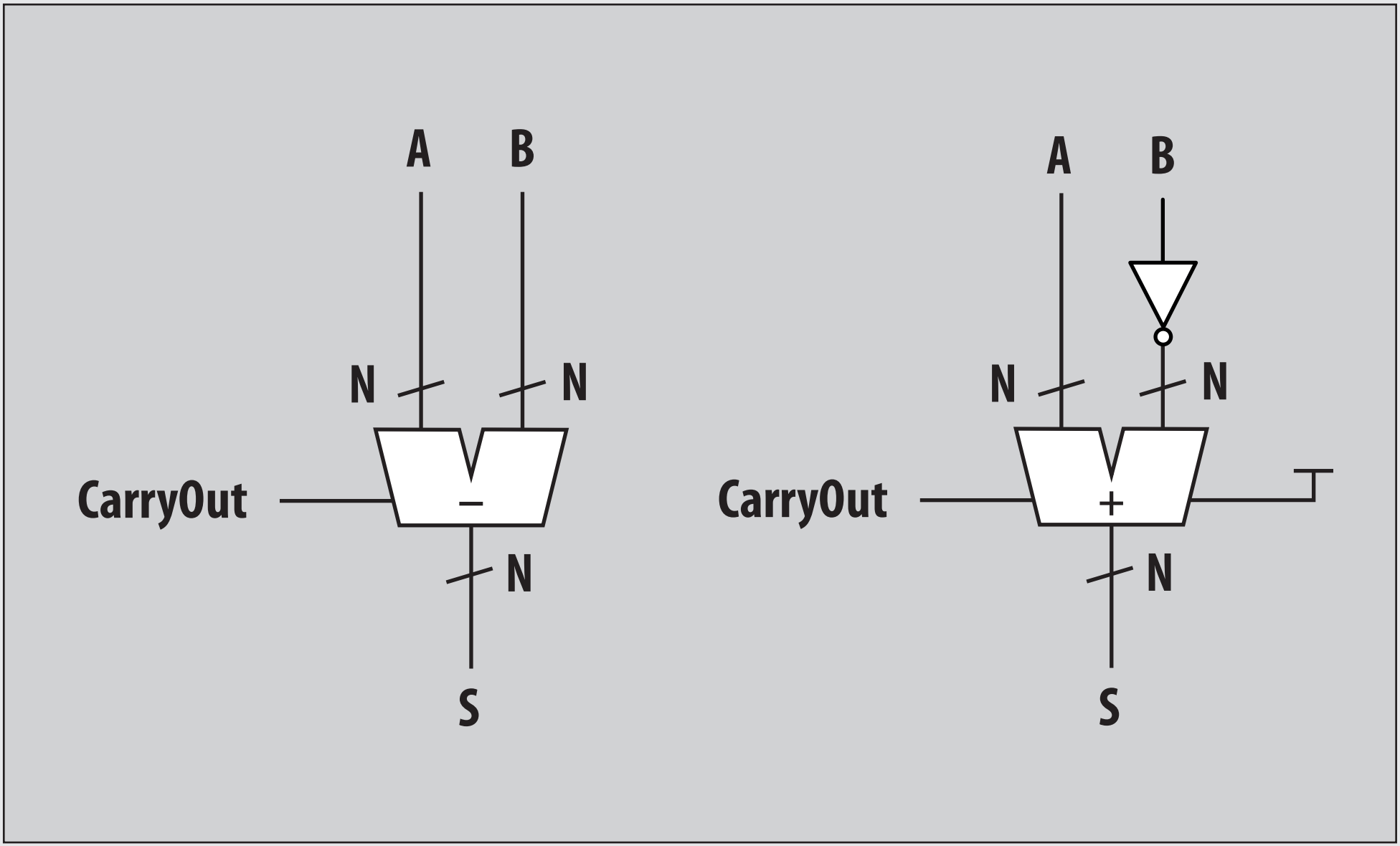
1. In each 4-bit block, the generate and propagate signals
   do not depend on the input signal **CarryIn**.

2. for that reason, the two signals can be computed
   independently and ***in parallel*** by each 4-bit block.

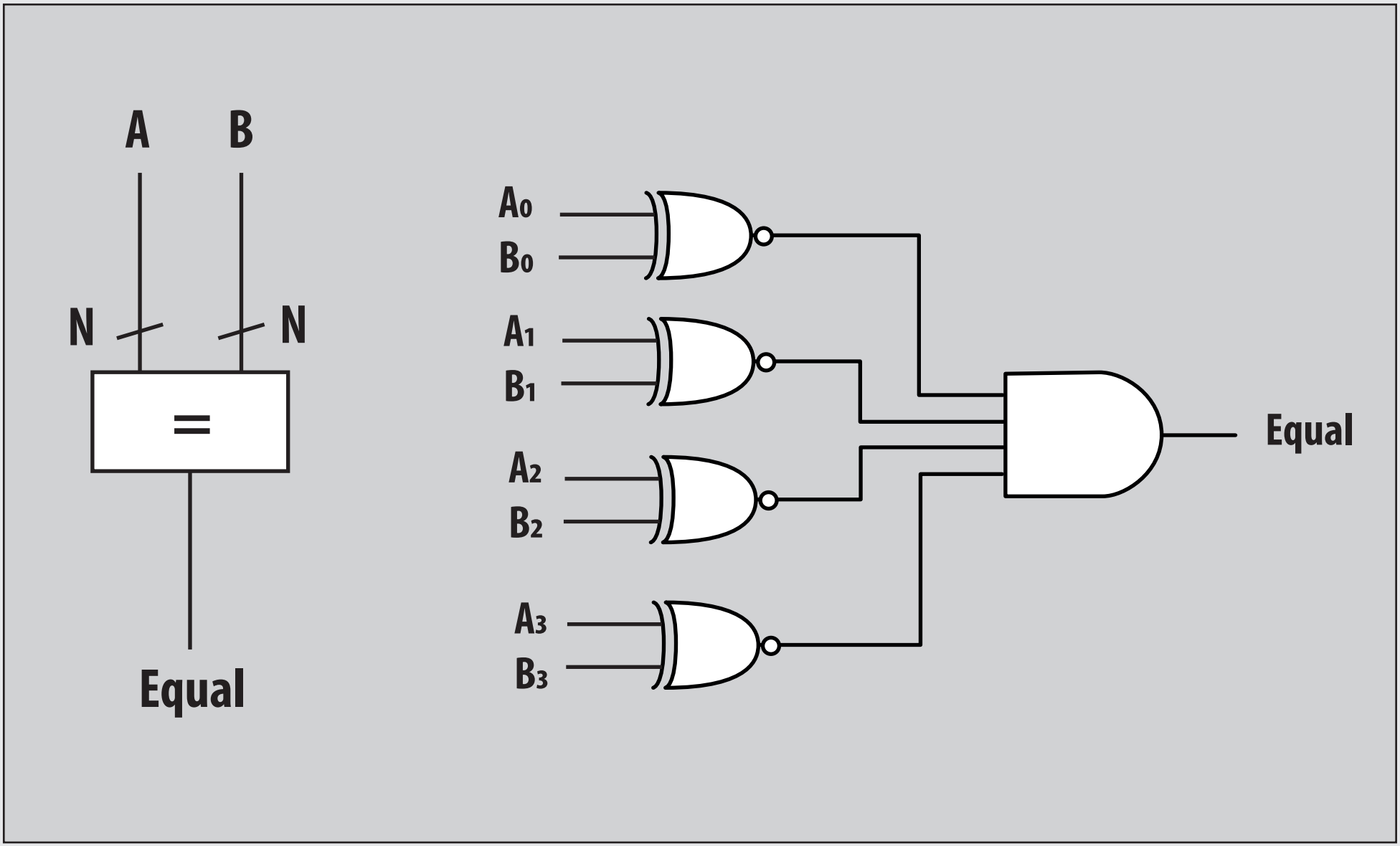
3. once a 4-bit block « knows » the values of the two signals **P** and **G**,
   it can generate and/or transmit carries much faster.

4. for that reason, each 4-bit block « receives » the signal **CarryIn**
   faster than in the Ripple Carry Adder circuit.

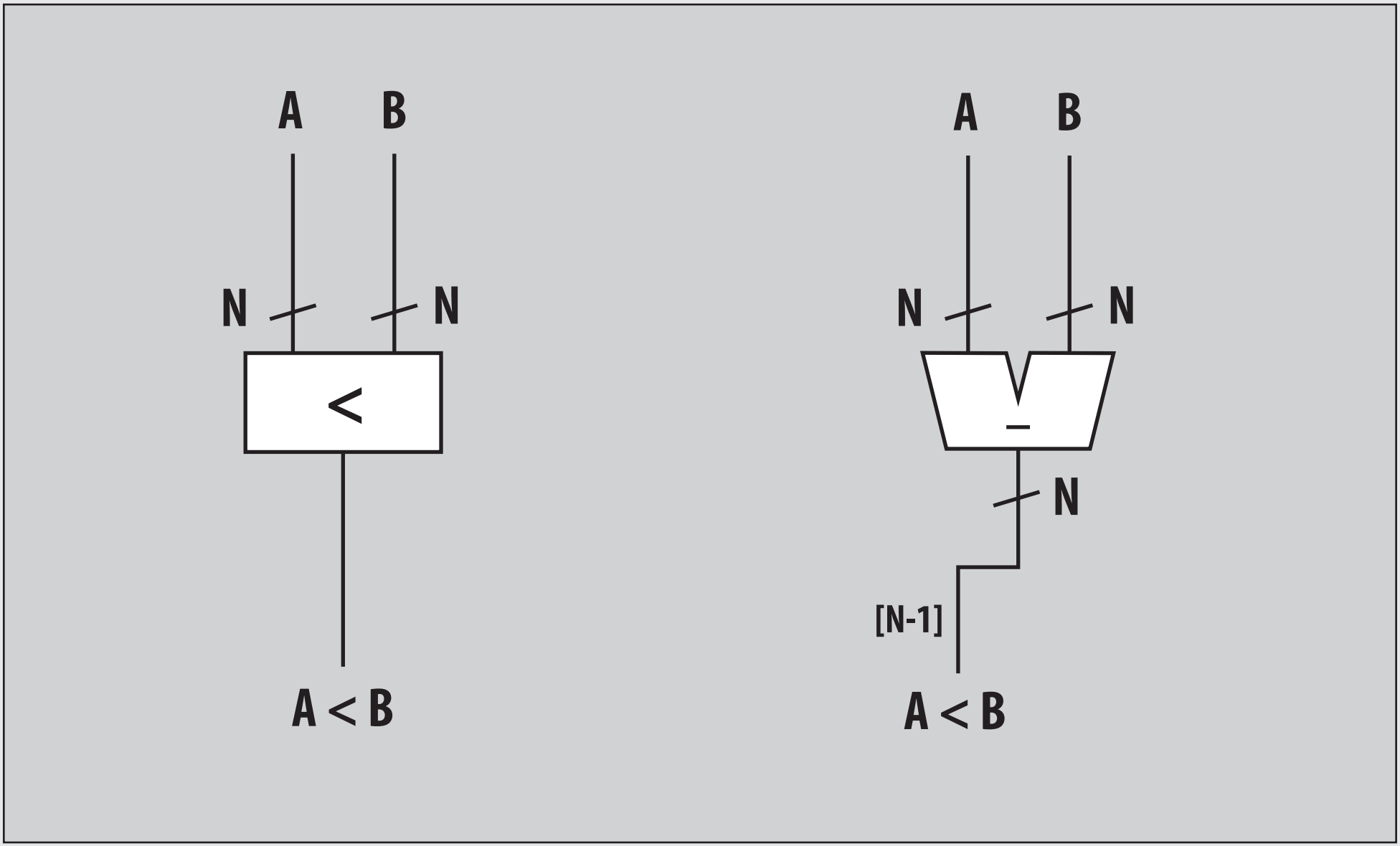
Subtraction



Equality Operator

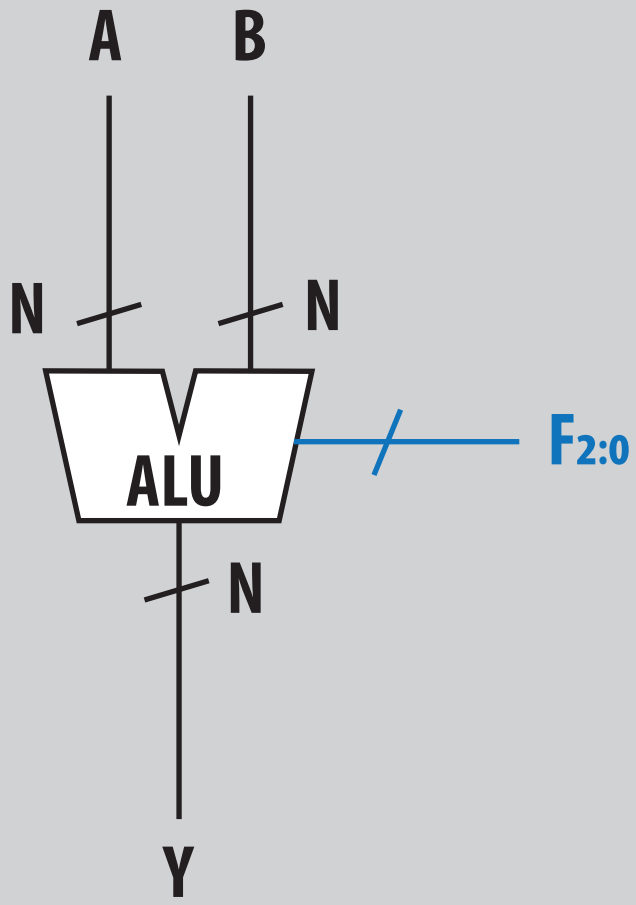


Magnitude Operator



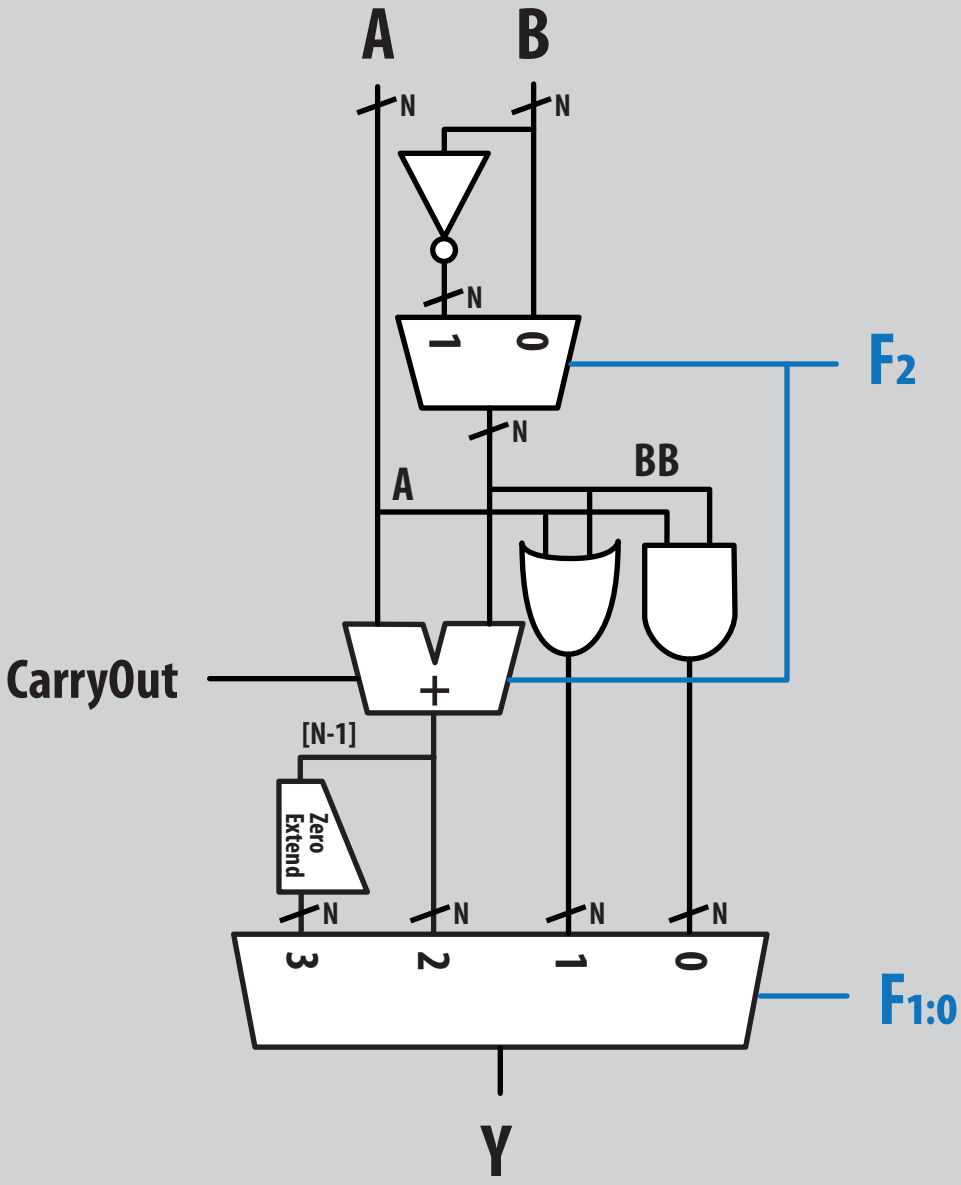
Arithmetic Logical Unit (ALU)

$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A - B
111	SLT

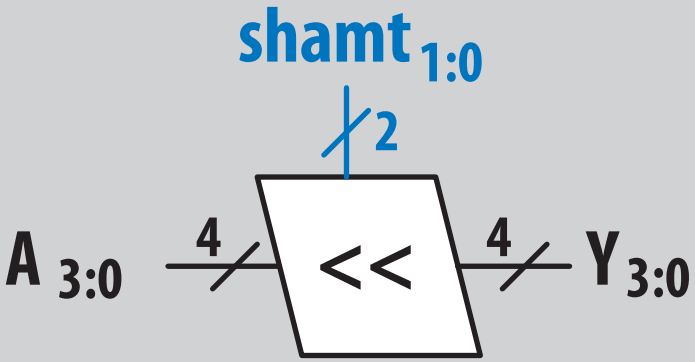


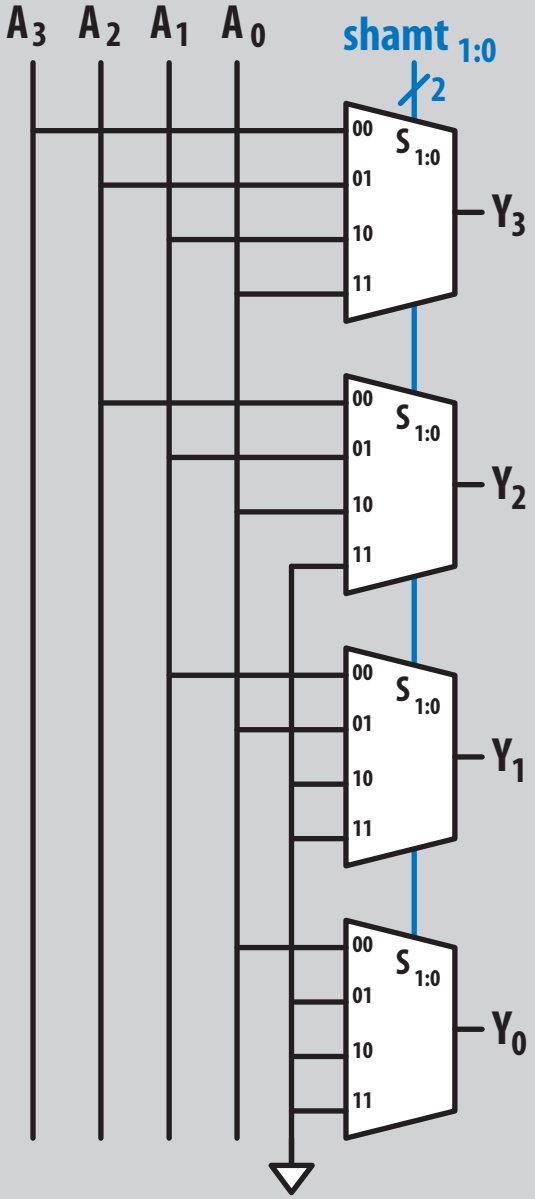
Arithmetic Logical Unit (ALU)

$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A - B
111	SLT

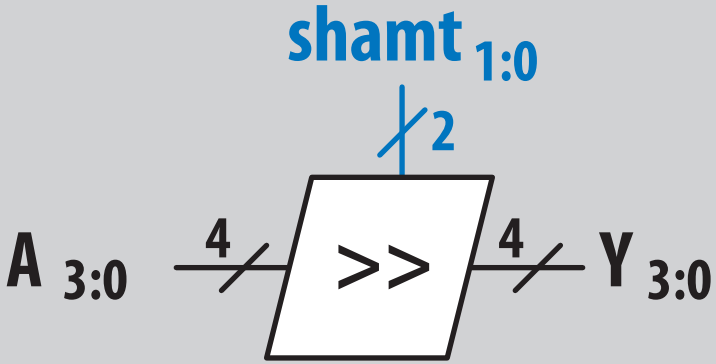


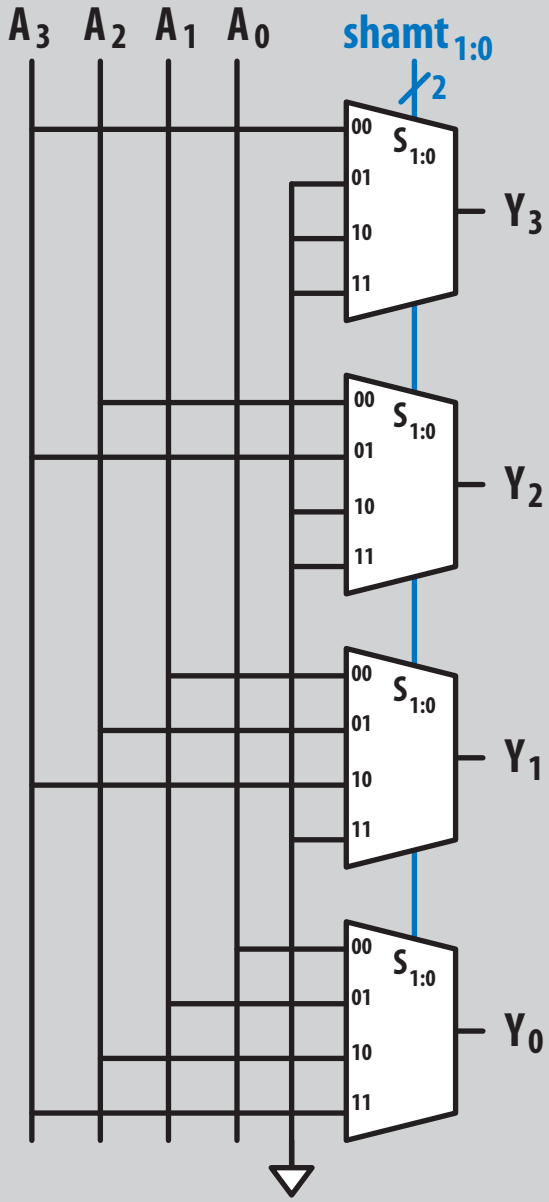
Shift Left


shamt 1:0
2
A 3:0
4
<<
4
Y 3:0

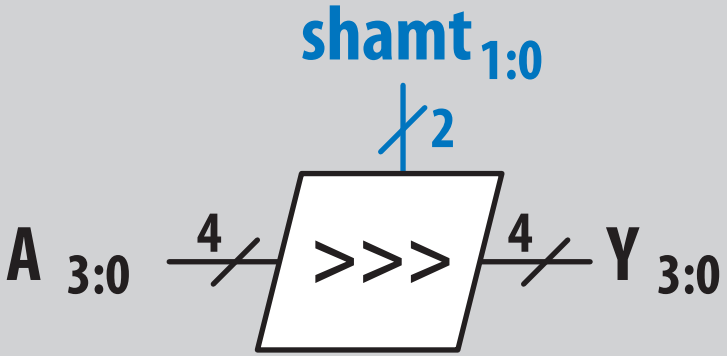

A3 A2 A1 A0
shamt 1:0
2
00
01
10
11
S 1:0
Y3
Y2
Y1
Y0

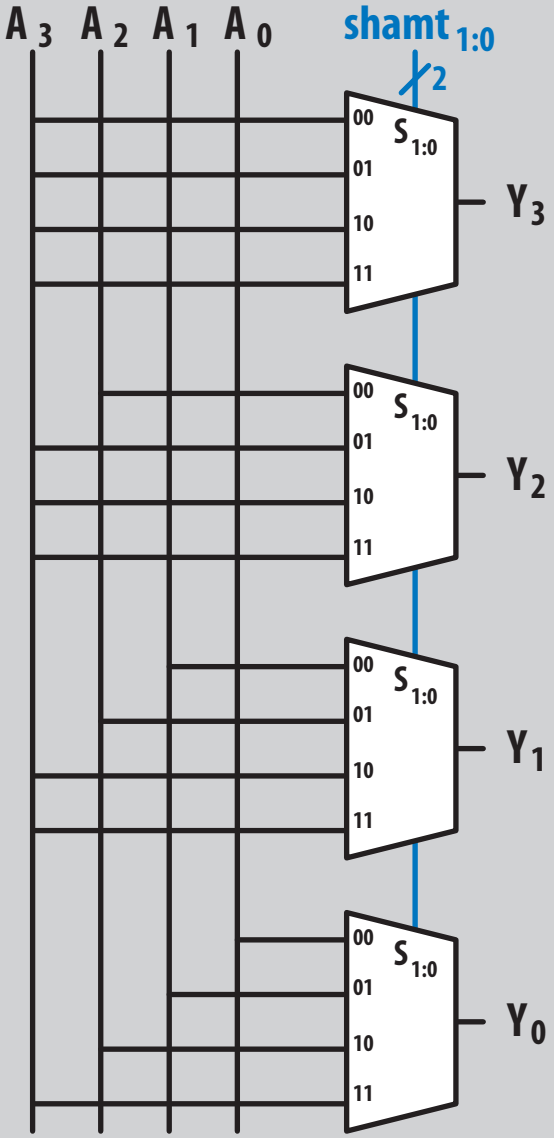
Shift Right


shamt 1:0
2
A 3:0
4
>>
4
Y 3:0


A3
A2
A1
A0
shamt 1:0
2
00
01
10
11
S1:0
Y3
Y2
Y1
Y0

Arithmetic Shift Right





Multiplication

Multiplication

					A_3	A_2	A_1	A_0
				$\times$	B_3	B_2	B_1	B_0
					A_3B_0	A_2B_0	A_1B_0	A_0B_0
				A_3B_1	A_2B_1	A_1B_1	A_0B_1	
			A_3B_2	A_2B_2	A_1B_2	A_0B_2		
+		A_3B_3	A_2B_3	A_1B_3	A_0B_3			
	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

The multiplication algorithm performed sequentially

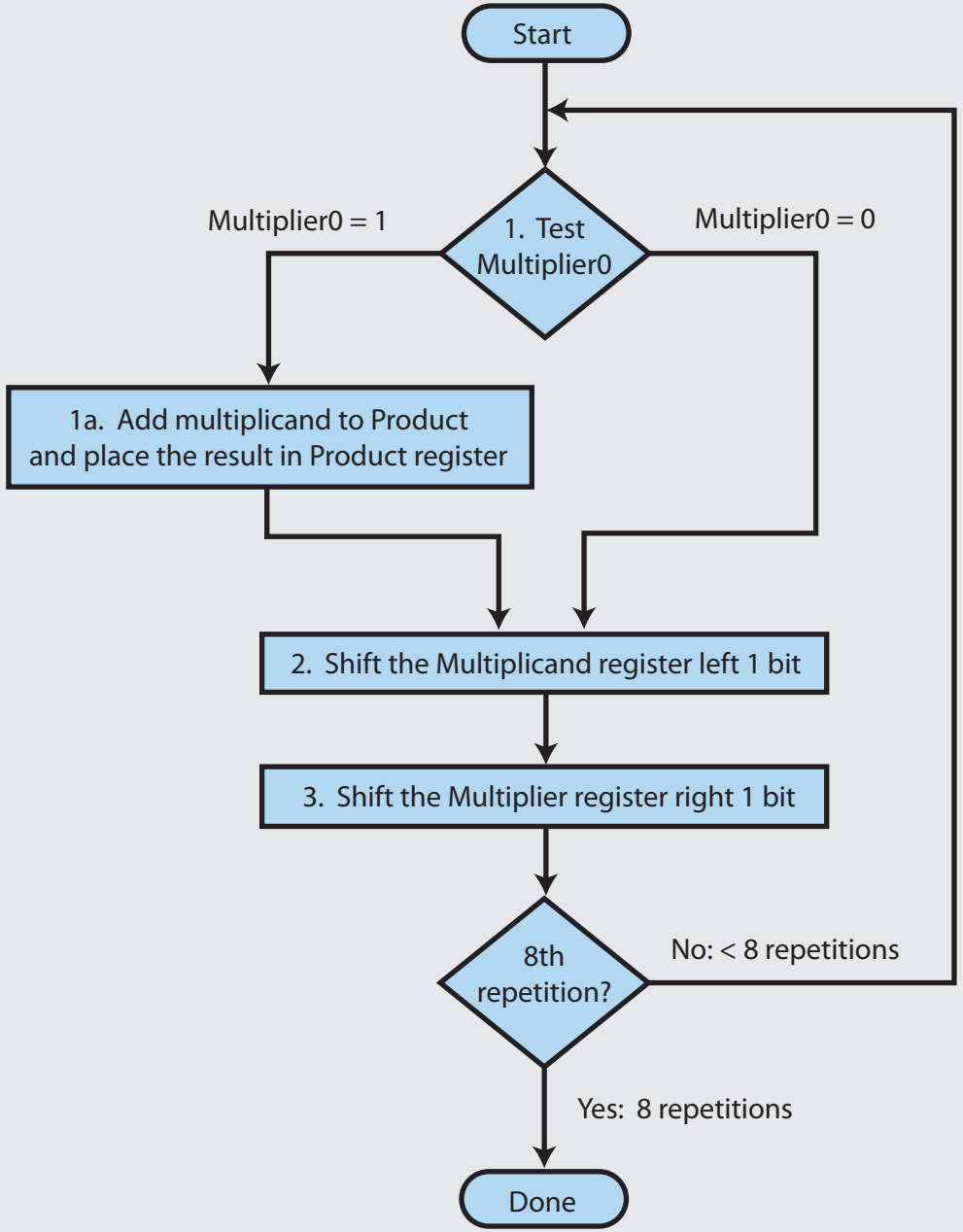
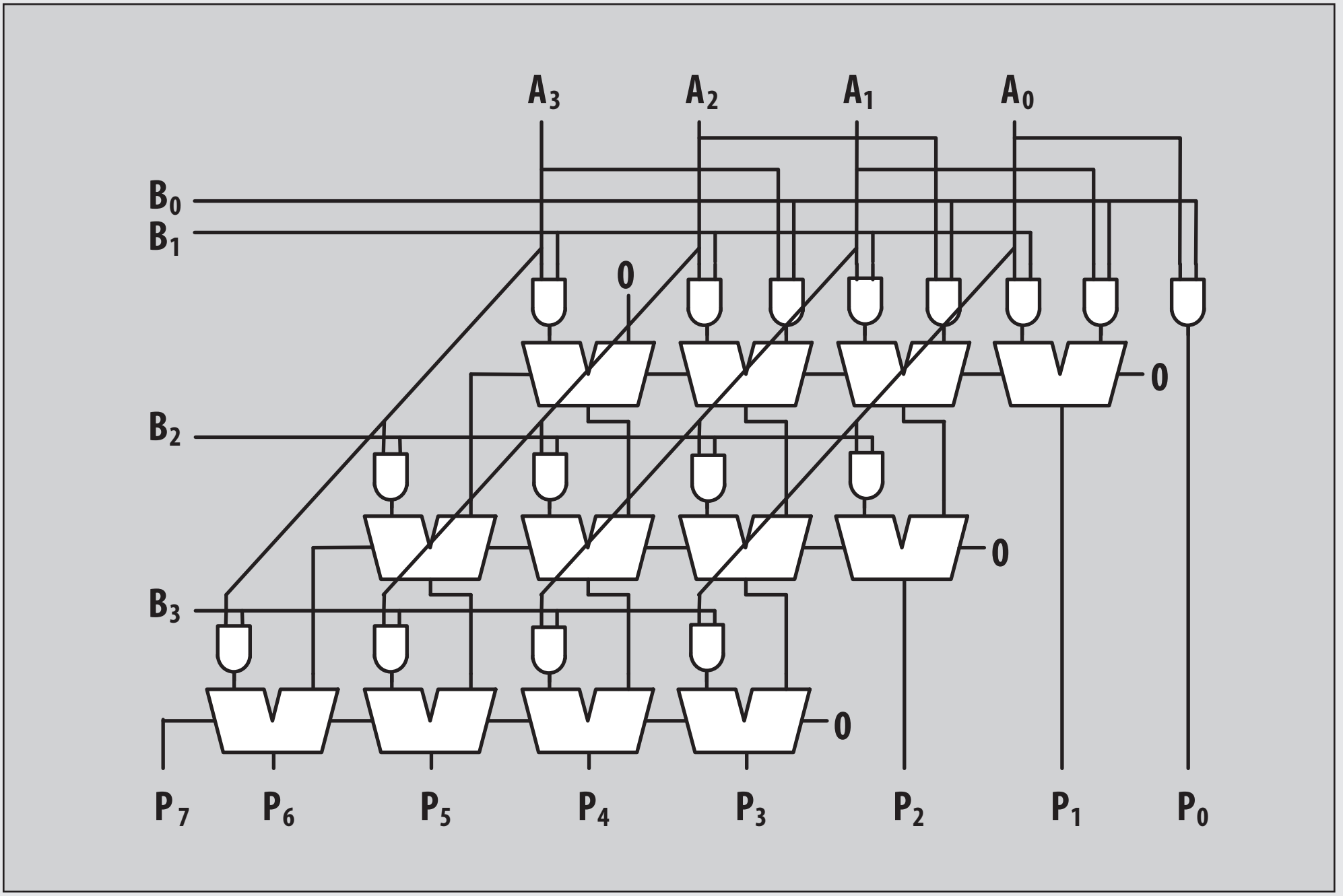


Figure 3.4 in Patterson and Hennessy

Multiplication



Division

Division Algorithm A ÷ B for N-bit unsigned numbers

```
R' = 0
for i = N-1 to 0
  R = { R' << 1 , A_i }     //  shift  R'  to the left
                            //  with  A_i  as least significant bit
  D = R - B
  if D < 0 then   Q_i = 0 , R' = R     //  R < B
  else            Q_i = 1 , R' = D     //  R >= B
R = R'
```

The partial remainder R is initialized to 0
The most significant bit of the dividend A becomes the least significant bit of R
The divisor B is repeatedly subtracted from this partial remainder to determine whether it fits :

- if the difference R - B is negative, then the quotient bit Q_i is 0 and the difference is discarded.
- if the difference R - B is positive, then the quotient bit Q_i is 1 and the partial remainder is updated to be the difference R - B.

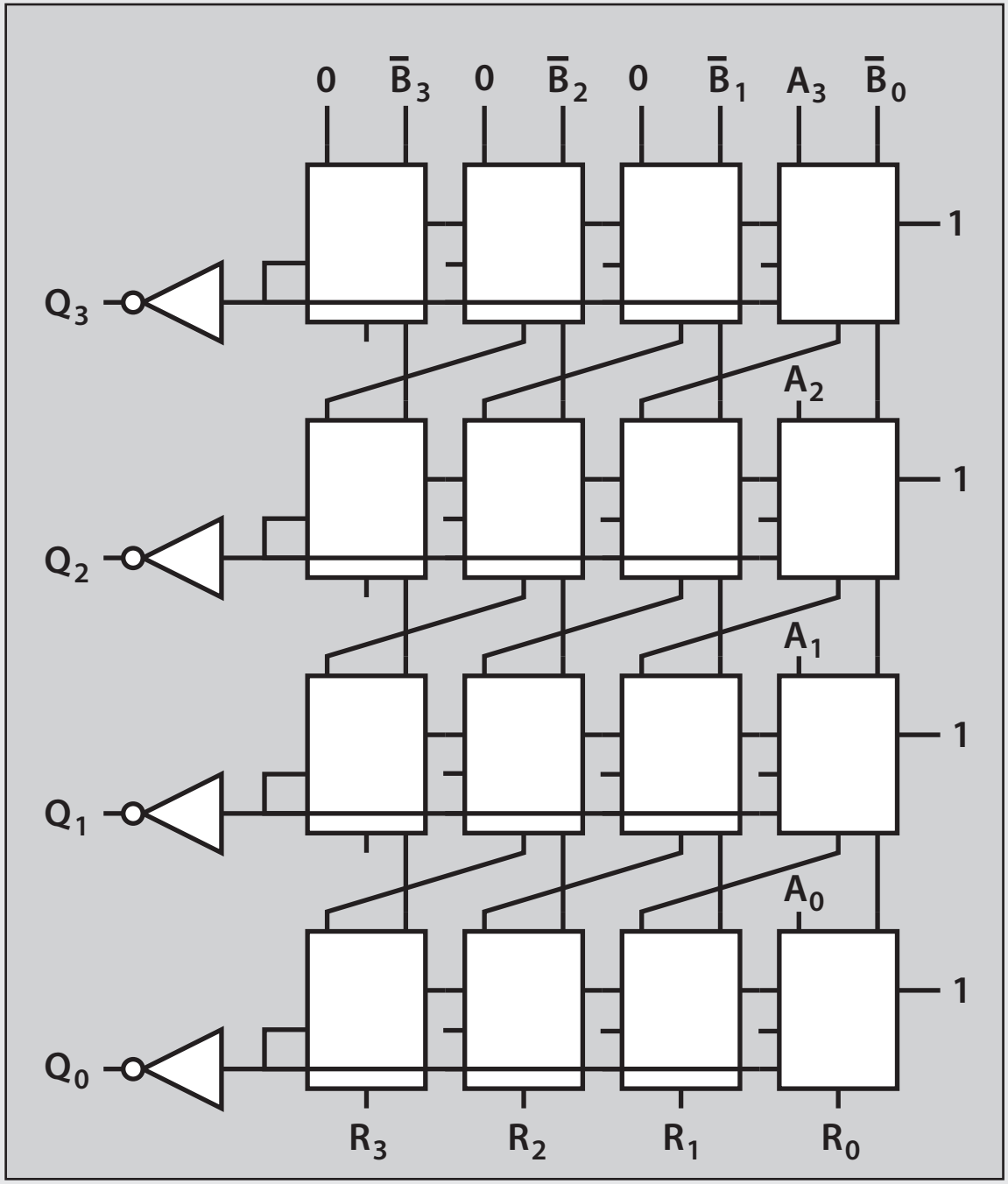
Exercise :

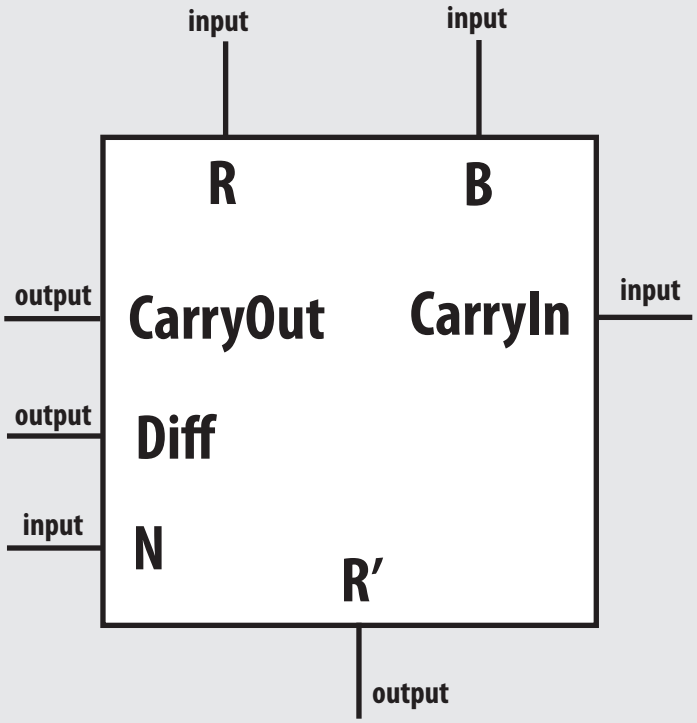
Use the algorithm to compute $A \div B$ for :

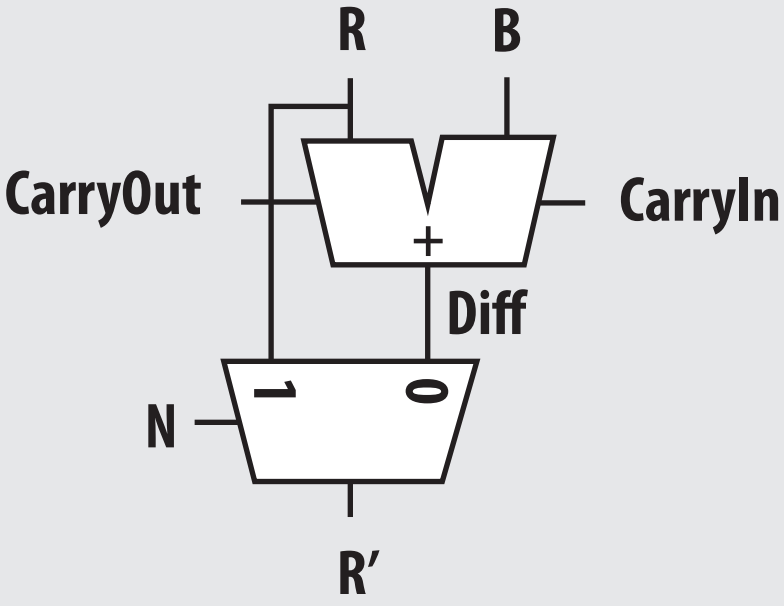
1. $A = 83$ and $B = 5$
2. $A = 97$ and $B = 23$

```
R' = 0
for i = N-1 to 0
  R = { R' << 1 , A_i }     //  shift R' to the left
                            //  with A_i as least significant bit
  D = R - B
  if D < 0 then   Q_i = 0 , R' = R    //  R < B
  else            Q_i = 1 , R' = D    //  R >= B
R = R'
```

Array Divider

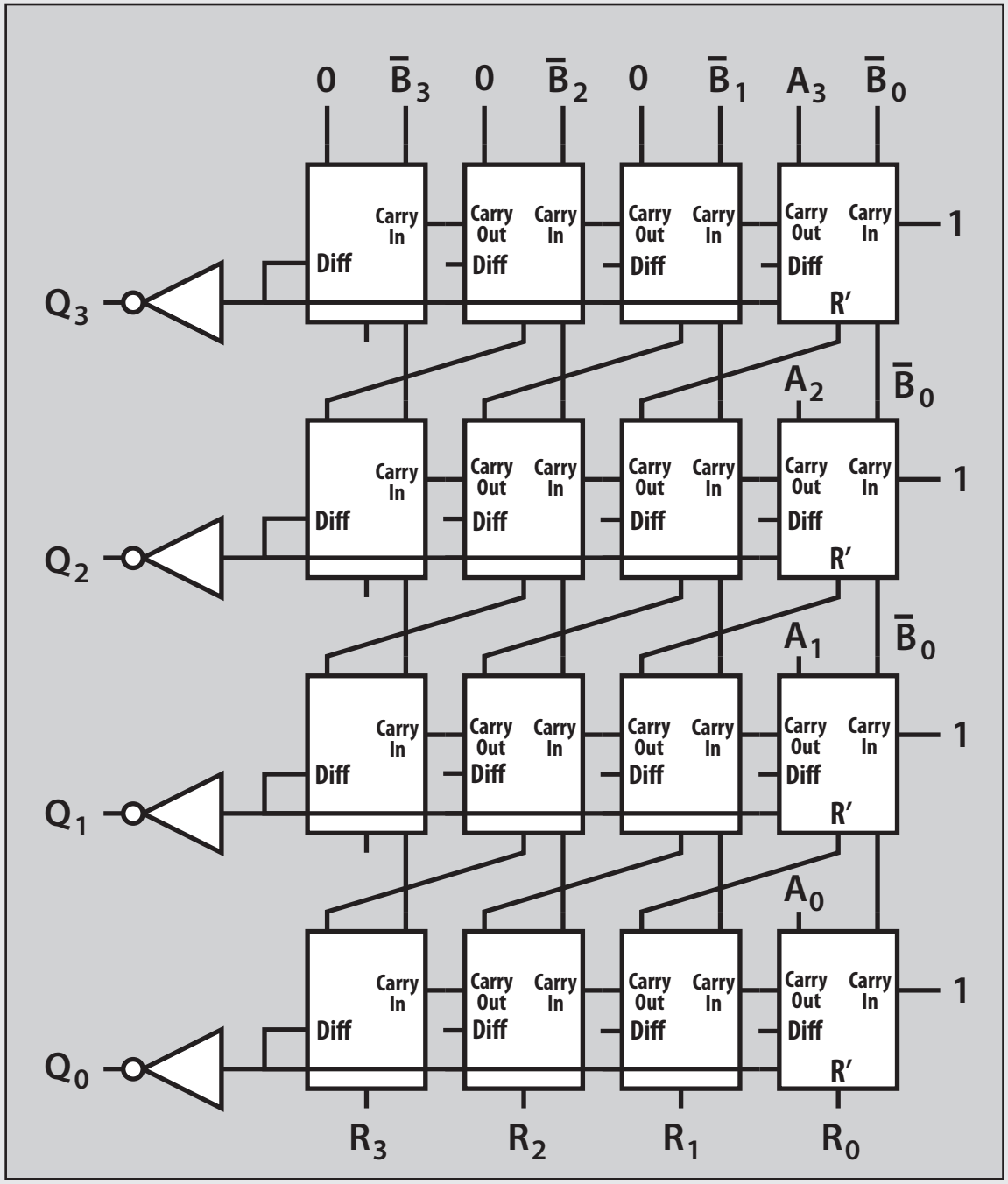


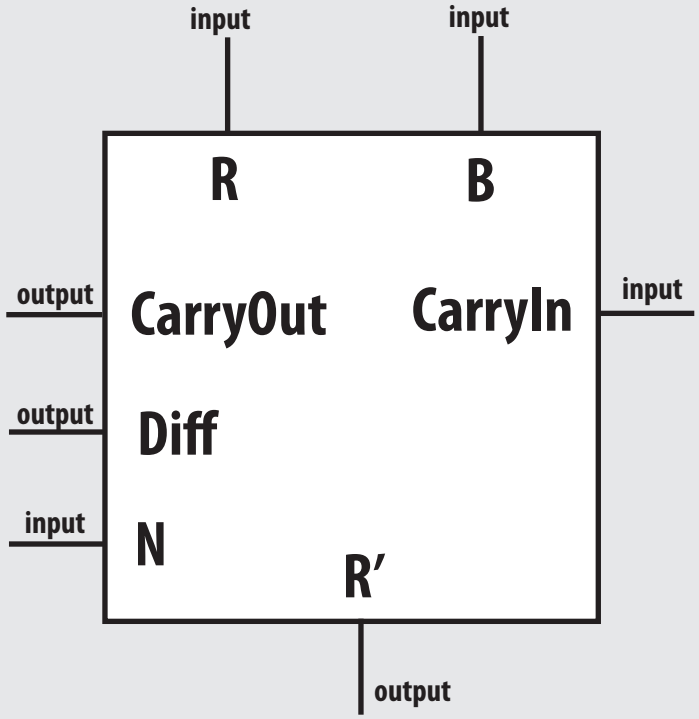


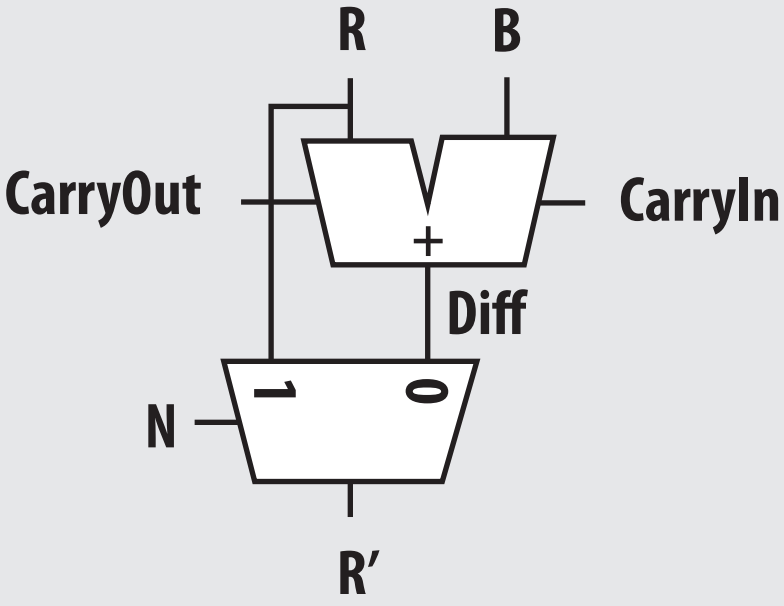


Exercise :

Explain how the circuit works !

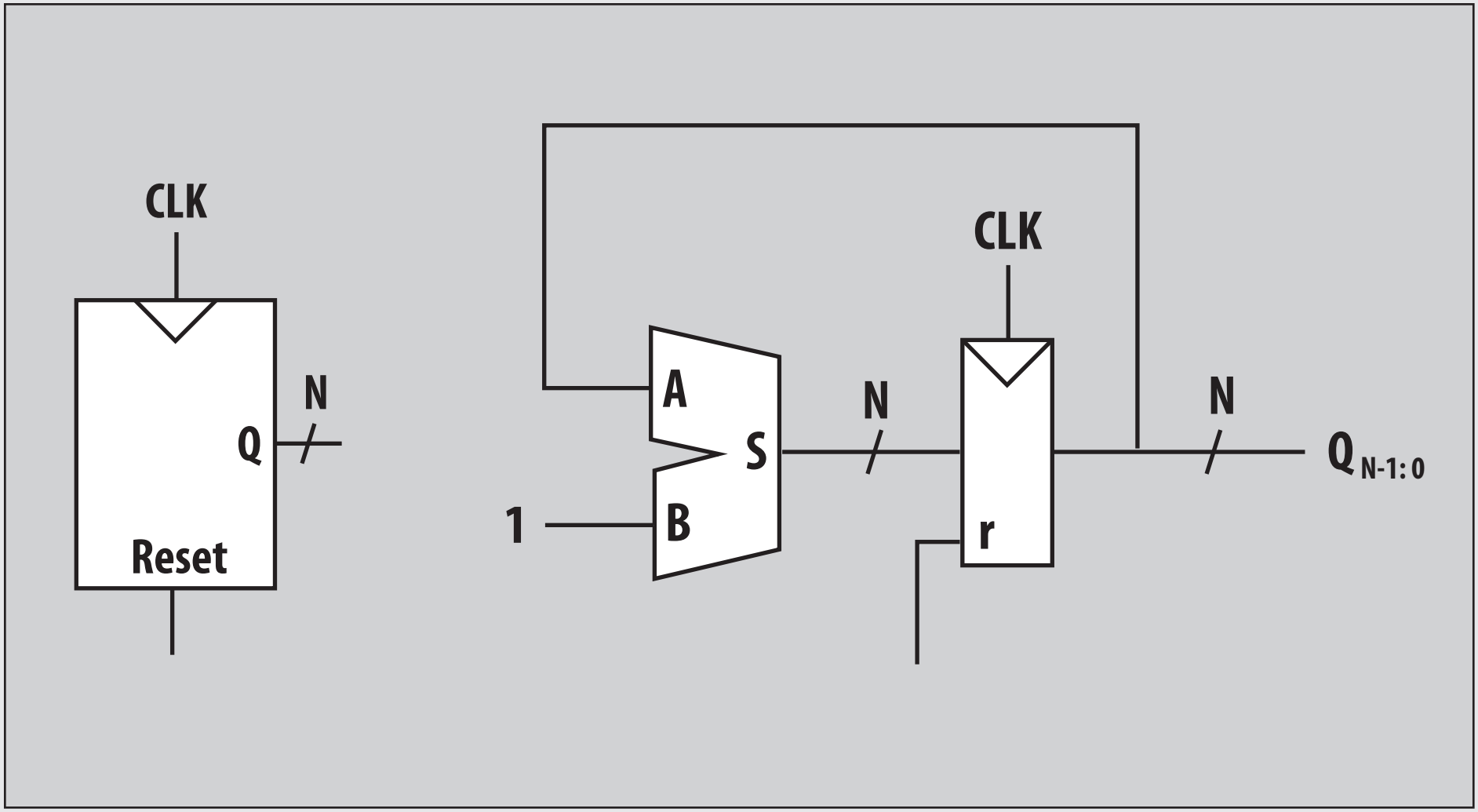




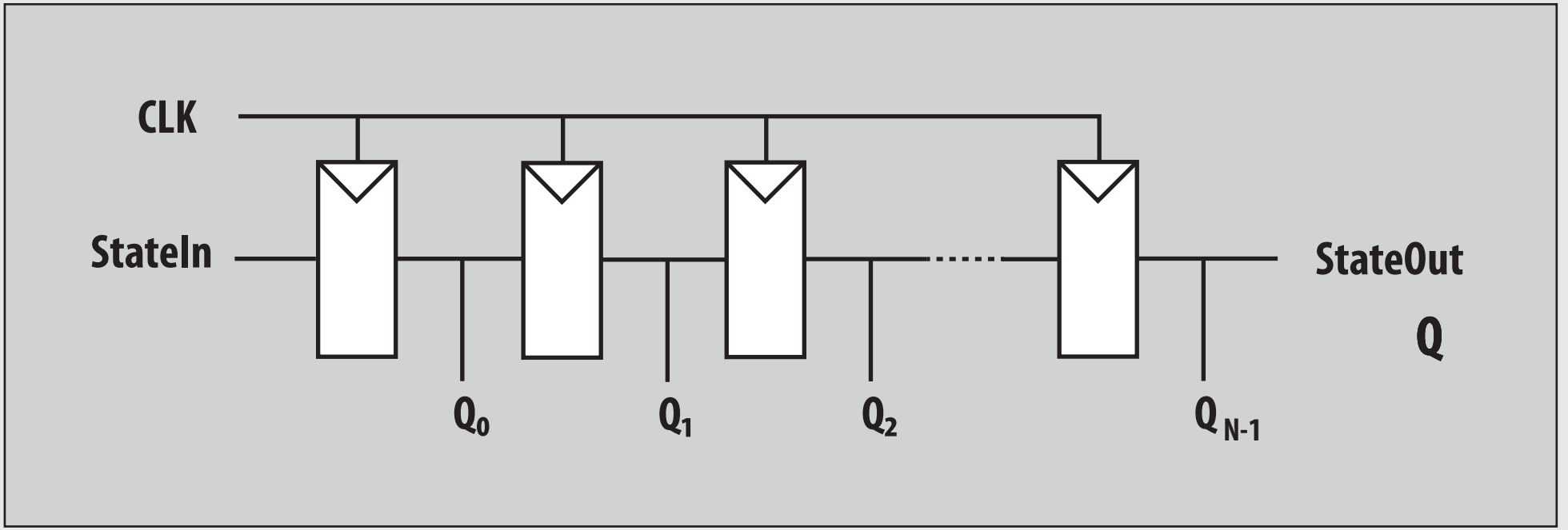


Sequential Building Blocks

Counters

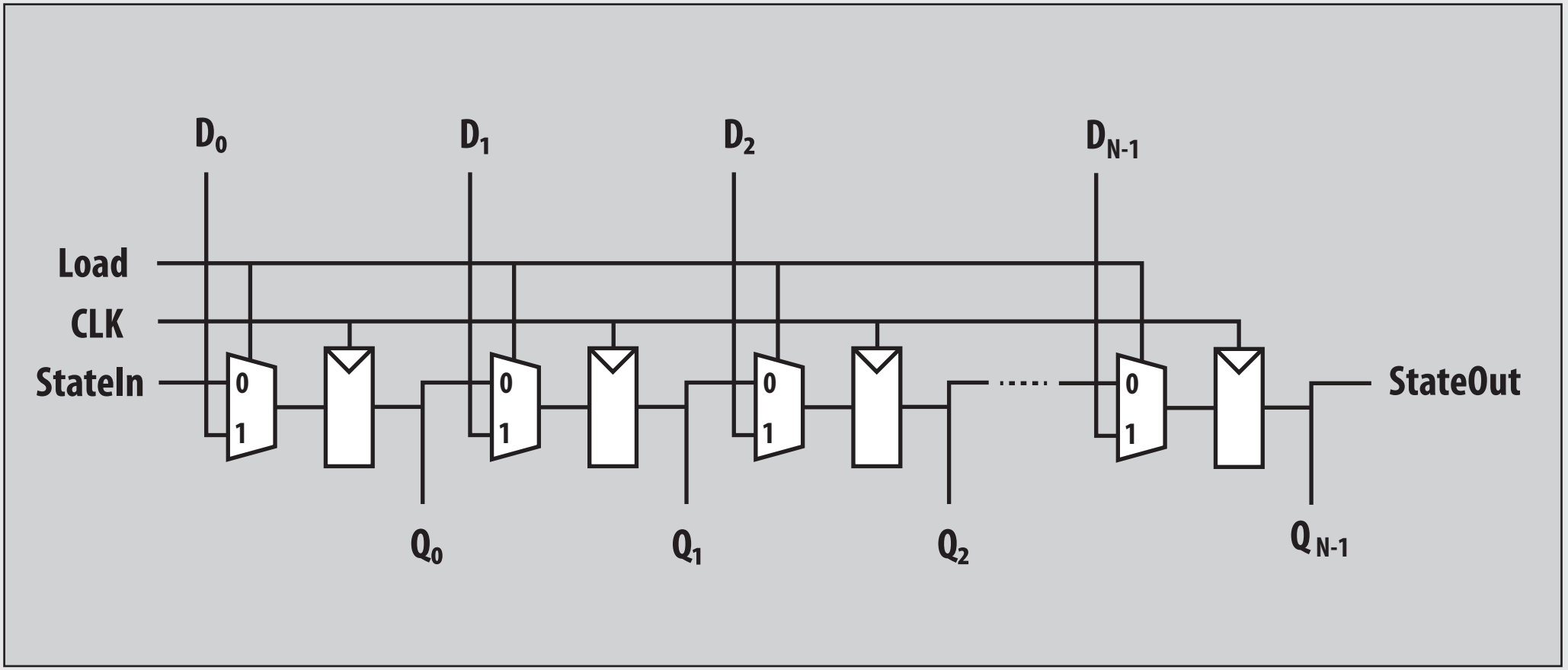


Shift Register



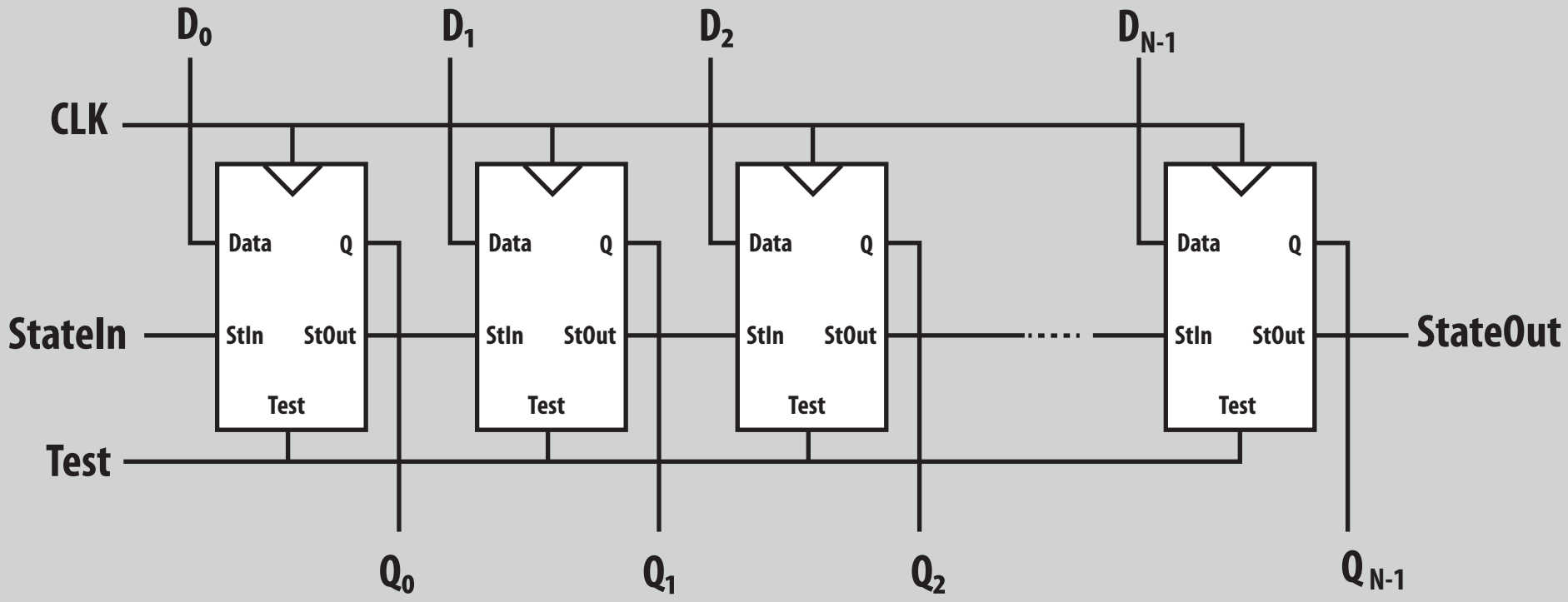
After N cycles, the past N values of the input **StateIn** are available in parallel at the output **Q**.

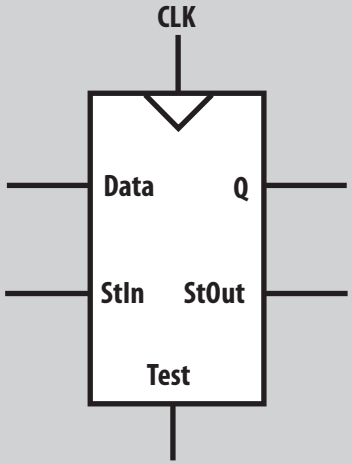
Shift Register with Parallel Load

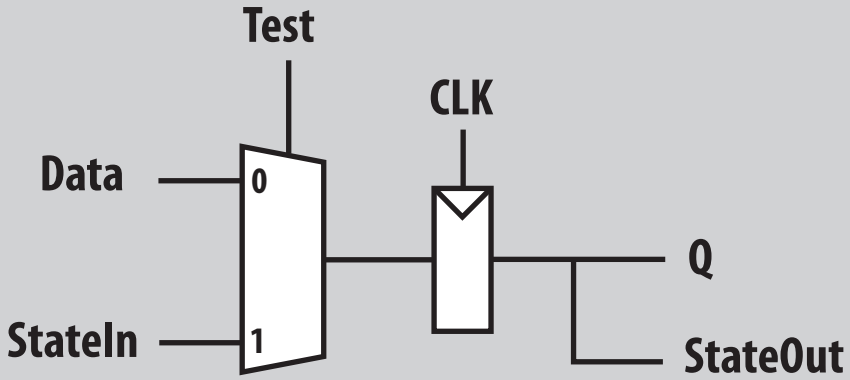


Here, the data can be loaded in parallel using the input D

Scannable flip-flop







Scan Chains