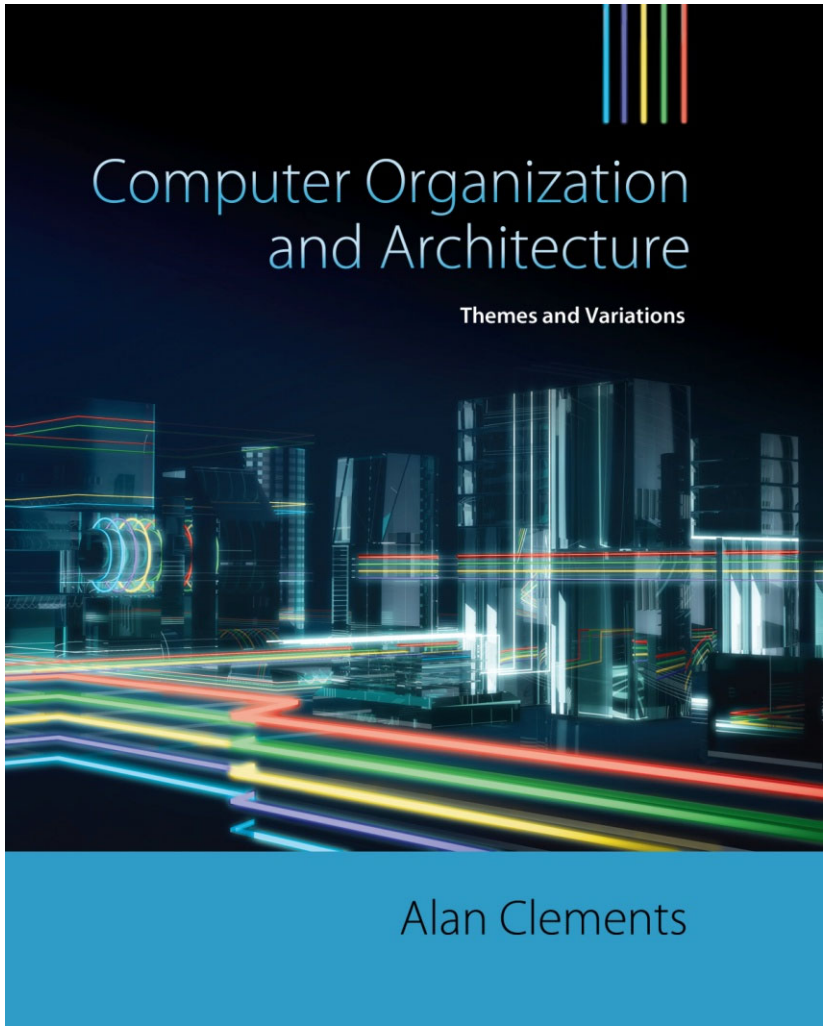
Part 5

Chapter 2

Computer Arithmetic and Digital Logic

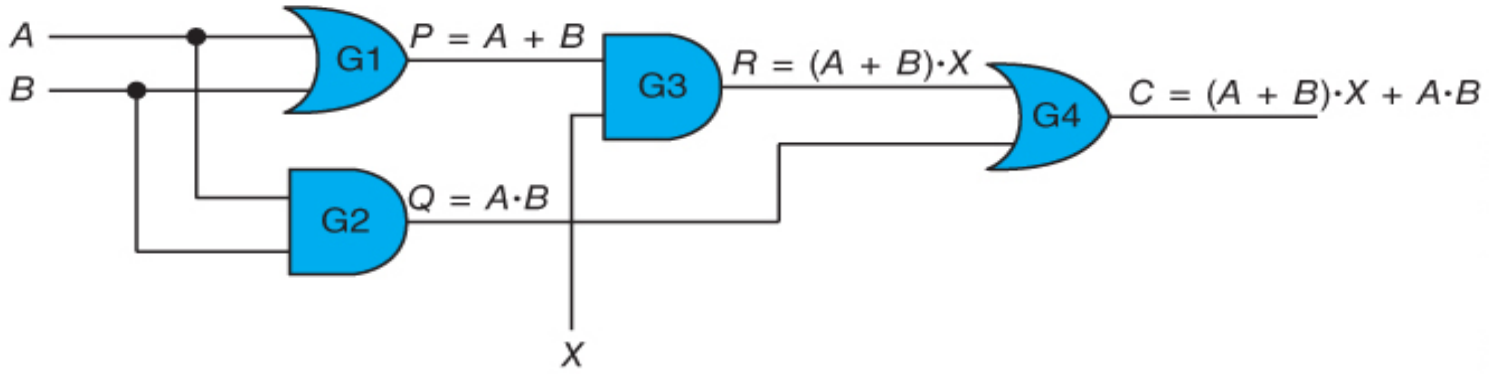


CENGAGE Learning

More Example of a Digital Circuit

- Figure 2.21 describes a circuit with
  - four gates, labeled **G1**, **G2**, **G3** and **G4**.
  - three inputs A, B, and X, and
  - an output C.
  - It also has three intermediate logical values labeled P, Q, and R.
- We can treat a gate as a *processor* that operates on its inputs according to its logical function;
  - For example, the inputs to gate **G3** are P and X, and its output is P · X.
  - Because P = A + B, the output of **G3** is (A + B) · X.
  - Similarly, the output of gate **G4** is R + Q,
  - Because R = (A + B) · X and Q = A · B, the output of gate **G4** is (A + B) · X + A · B.

FIGURE 2.21 A circuit with four gates



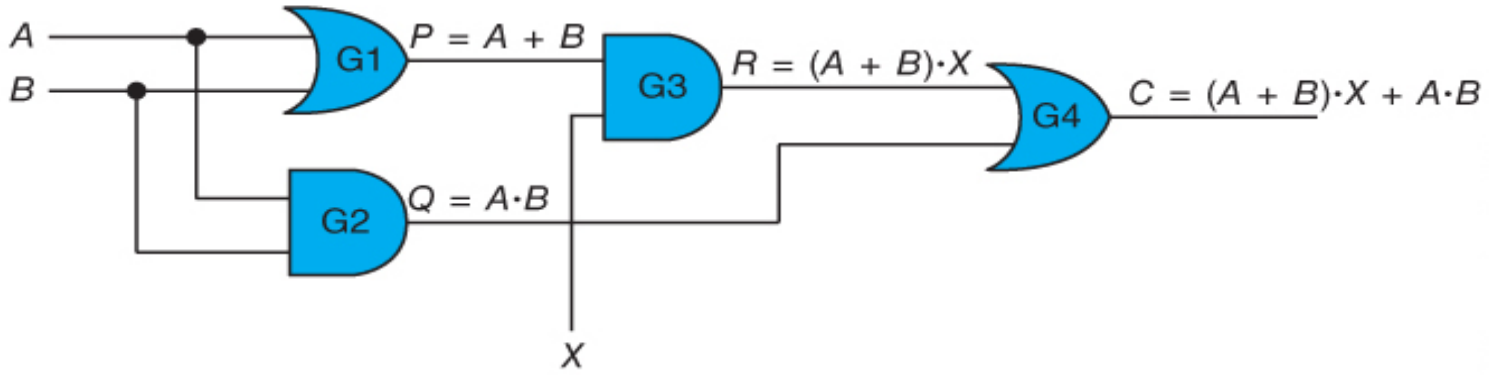
More Example of a Digital Circuit

- Table 2.12 gives the truth table for Figure 2.21.
- Note that the ***output corresponds to the carry out of a 3-bit adder***.

TABLE 2.12 Truth Table for Figure 2.21

Inputs			Intermediate Values			Output
X	A	B	$P = A + B$	$Q = A \cdot B$	$R = (A + B) \cdot X$	$C = Q + R$
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	1	0	1	1
1	1	1	1	1	1	1

FIGURE 2.21 A circuit with four gates



The Half-Adder and Full-Adder

- Table 2.13 gives the truth table of a *half-adder* that adds bit A to bit B to get a sum and a carry.

A single-bit full-adder is a logical circuit that performs an addition operation on three one-bit binary digits

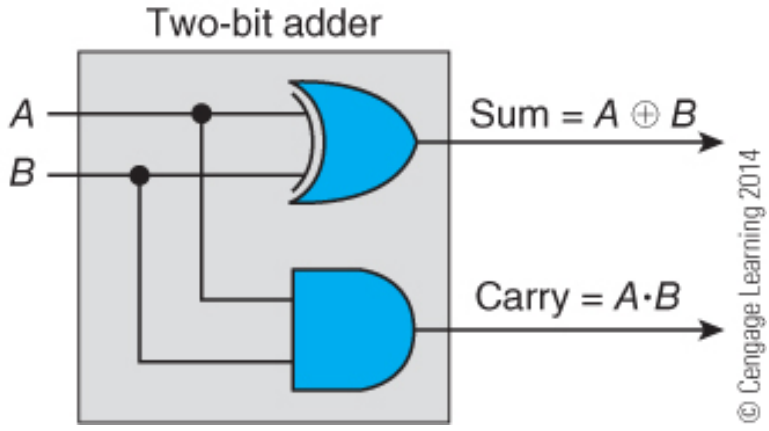
- Figure 2.22 shows the possible structure of a two-bit adder.
  - The sum bit is generated by **XOR**ing the two inputs.
  - The carry bit is generated by **AND**ing the two inputs.

TABLE 2.13

Truth Table of a Half Adder

A	B	Sum	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FIGURE 2.22 The two-bit adder (the half adder)



Full-Adder Circuit

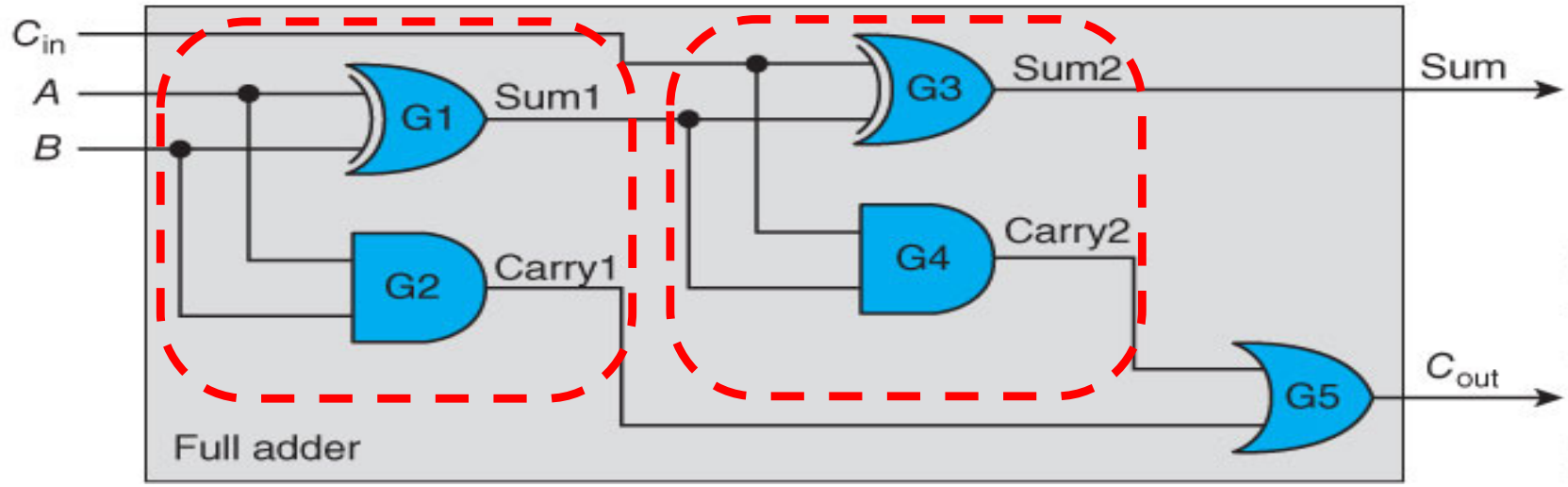
- Figure 2.3 gives the possible circuit of a *one-bit full-adder*.
  - Consists of *two half-adder* and a *one* ***OR*** gate

$$\text{Sum} = (A \oplus B) \oplus C_{in} \quad \leftarrow \text{Carry}$$
$$= (A \,.\, B + \overline{A} \,.\, B) \,.\, \overline{C_{in}} + \overline{(A \,.\, B + \overline{A} \,.\, B)} \,.\, C_{in}$$

$$C_{out} = A \,.\, B + (A \oplus B) \,.\, C_{in}$$

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

FIGURE 2.23 The full adder



Full-Adder Circuit

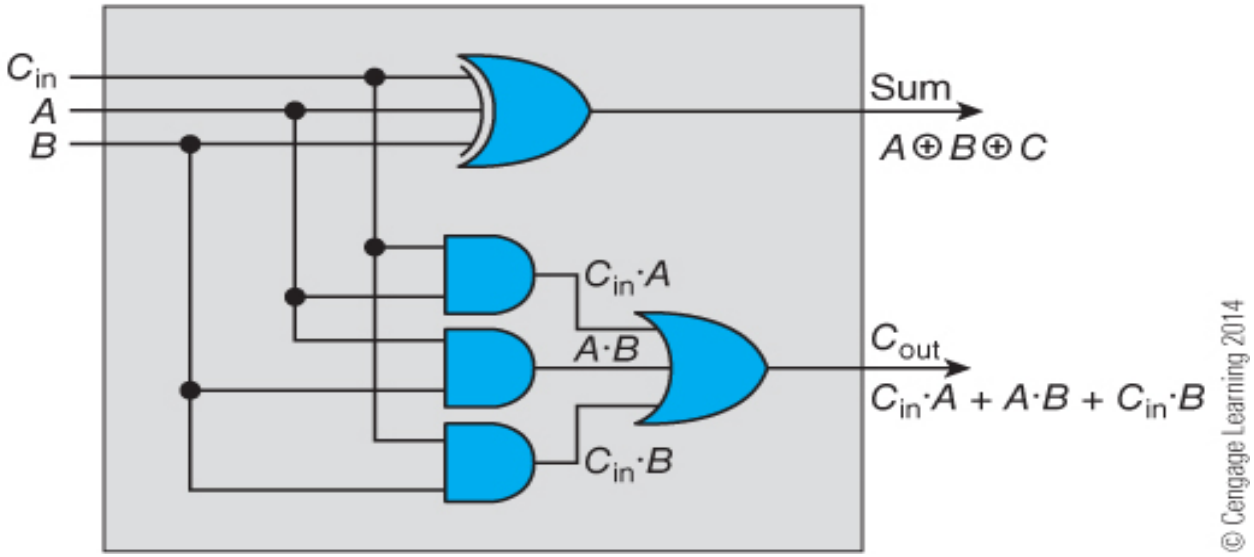
- Figure 2.3 gives an alternative circuit of a *one-bit full-adder*.

$$\text{Sum} = (A \oplus B) \oplus C_{in}$$
$$= (A \cdot \overline{B} + \overline{A} \cdot B) \cdot \overline{C_{in}} + \overline{(A \cdot \overline{B} + \overline{A} \cdot B)} \cdot C_{in}$$

$$C_{out} = C_{in} \cdot A + A \cdot B + C_{in} \cdot B$$

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

FIGURE 2.24 Alternative full adder circuit



Full-Adder Circuit

$\text{Sum} = (A \oplus B) \oplus C$

$= (A \,.\, \bar{B} + \bar{A} \,.\, B) \,.\, \bar{C} + \overline{(A \,.\, \bar{B} + \bar{A} \,.\, B)} \,.\, C$

Using De Morgan's law: $\overline{X + Y} = \bar{X} \,.\, \bar{Y}$

$= (A \,.\, \bar{B} + \bar{A} \,.\, B) \,.\, \bar{C} + (\overline{(A \,.\, \bar{B})} \,.\, \overline{(\bar{A} \,.\, B)}) \,.\, C$

Using De Morgan's law: $\overline{X \,.\, Y} = \bar{X} + \bar{Y}$

$= (A \,.\, \bar{B} + \bar{A} \,.\, B) \,.\, \bar{C} + ((\bar{A} + \bar{\bar{B}}) \,.\, (\bar{\bar{A}} + \bar{B})) \,.\, C$

Using property $\bar{\bar{A}} = A$

$= (A \,.\, \bar{B} + \bar{A} \,.\, B) \,.\, \bar{C} + ((\bar{A} + B) \,.\, (A + \bar{B})) \,.\, C$

Using Distributive law: $X \,.\, (Y + Z) = X \,.\, Y + X \,.\, Z$

$= (A \,.\, \bar{B} + \bar{A} \,.\, B) \,.\, \bar{C} + ((\bar{A} + B) \,.\, A + (\bar{A} + B) \,.\, \bar{B})) \,.\, C$

Using Commutative law: $X \,.\, Y = Y \,.\, X$

$= \bar{C} \,.\, (A \,.\, \bar{B} + \bar{A} \,.\, B) + (A \,.\, (\bar{A} + B) + \bar{B} \,.\, (\bar{A} + B)) \,.\, C$

Using Distributive law: $X \,.\, (Y + Z) = X \,.\, Y + X \,.\, Z$

$= (\bar{C} \,.\, A \,.\, \bar{B} + \bar{C} \,.\, \bar{A} \,.\, B) + ((A \,.\, \bar{A} + A \,.\, B) + (\bar{B} \,.\, \bar{A} + \bar{B} \,.\, B)) \,.\, C$

Using property $\bar{X} \,.\, X = 0$

$= (\bar{C} \,.\, A \,.\, \bar{B} + \bar{C} \,.\, \bar{A} \,.\, B) + ((0 + A \,.\, B) + (\bar{B} \,.\, \bar{A} + 0)) \,.\, C$

Using inversion property: $X + 0 = X$

$= (\bar{C} \,.\, A \,.\, \bar{B} + \bar{C} \,.\, \bar{A} \,.\, B) + (A \,.\, B + \bar{B} \,.\, \bar{A}) \,.\, C$

Using Commutative law: $X \,.\, Y = Y \,.\, X$

$= (\bar{C} \,.\, A \,.\, \bar{B} + \bar{C} \,.\, \bar{A} \,.\, B) + C \,.\, (A \,.\, B + \bar{B} \,.\, \bar{A})$

Using Distributive law: $X \,.\, (Y + Z) = X \,.\, Y + X \,.\, Z$

$= \bar{C} \,.\, A \,.\, \bar{B} + \bar{C} \,.\, \bar{A} \,.\, B + C \,.\, A \,.\, B + C \,.\, \bar{B} \,.\, \bar{A}$

Using Commutative law: $X \,.\, Y = Y \,.\, X$

$= A \,.\, \bar{B} \,.\, \bar{C} + \bar{A} \,.\, B \,.\, \bar{C} + A \,.\, B \,.\, C + \bar{A} \,.\, \bar{B} \,.\, C$

A	B	C	Sum
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Full-Adder Circuit

$C_{out} = (A + B)\cdot C + A\cdot B$ *From Figure 2.21*

$C_{out} = C\cdot A + A\cdot B + C\cdot B$ *From Figure 2.24*

$C_{out} = A\,.B + (A \oplus B).C$ *From Figure 2.23*

$C_{out} = A\,.B + (A.\overline{B} + \overline{A}.B).C$

Using Distributive law

$C_{out} = A\,.B + A.\overline{B}.C + \overline{A}.B.C$

Using Distributive law

$C_{out} = A\,.(B + \overline{B}.C) + \overline{A}.B.C$

Using $X + \overline{X}.Y = X + Y$

$C_{out} = A\,.(B + C) + \overline{A}.B.C$

Using Distributive law

$C_{out} = A.B + A.C + \overline{A}.B.C$

Using Distributive law

$C_{out} = A.B + (A+\overline{A}.B).C$

Using $X + \overline{X}.Y = X + Y$

$C_{out} = A.B + (A+ B).C$

Using Distributive law

$C_{out} = A.B + A.C+ B.C$

Using Commutative law

$C_{out} = C\cdot A + A\cdot B + C\cdot B$ *As in Figure 2.24*

Using Commutative law:

$C_{out} = A\cdot C + B\cdot C + A\cdot B$

Using Distributive law

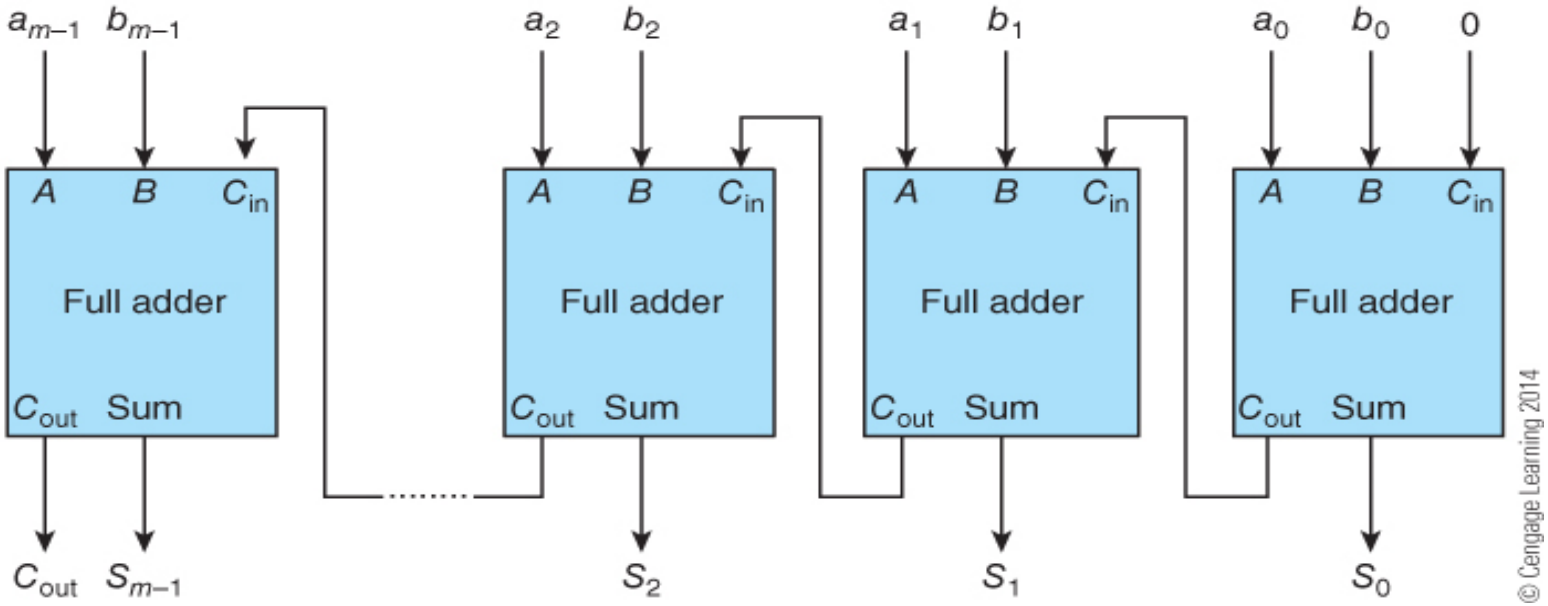
$C_{out} = (A + B)\cdot C + A\cdot B$ *As in Figure 2.21*

A	B	C	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Full-Adder

- We need *m full-adder* circuits to add two *m*-bit words *in parallel* as Figure 2.25 demonstrates.
- The m_i *full-adder* adds bit a_i to bit b_i, together with a *carry-in* from the stage on its right, to produce a sum_i and a *carry-out* to the stage on its left.

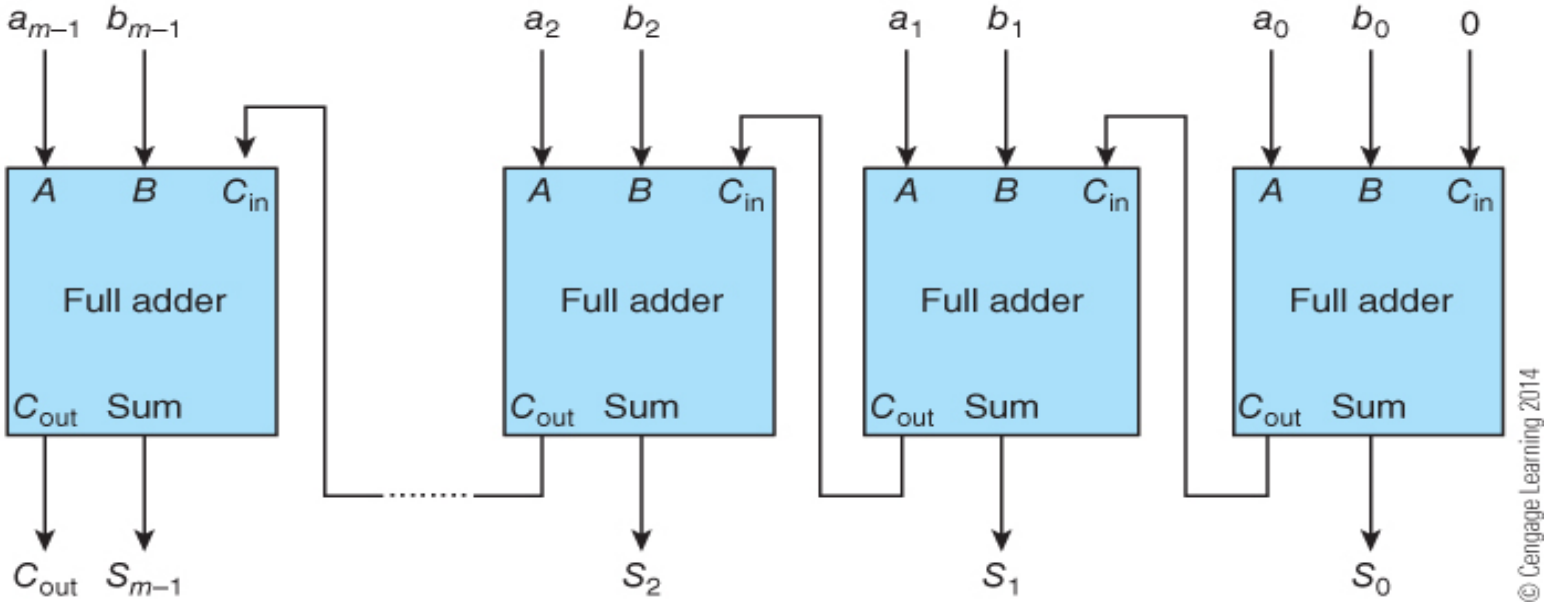
FIGURE 2.25 The parallel adder



Full-Adder

- This circuit is called a parallel-adder because all the bits of the two words to be added are presented to it at the same time.
- The circuit is *not truly parallel* because bit s_i cannot be correctly produced until the *carry-in*$_i$ bit has been calculated by the *previous stage*.
- This is a *ripple through* adder because addition is not complete until the carry bit has ***rippled*** through the circuit.
- *True parallel-adders* use high-speed *look-ahead carry* circuits to produce all carry bits at once, hence speeding up the addition operation.

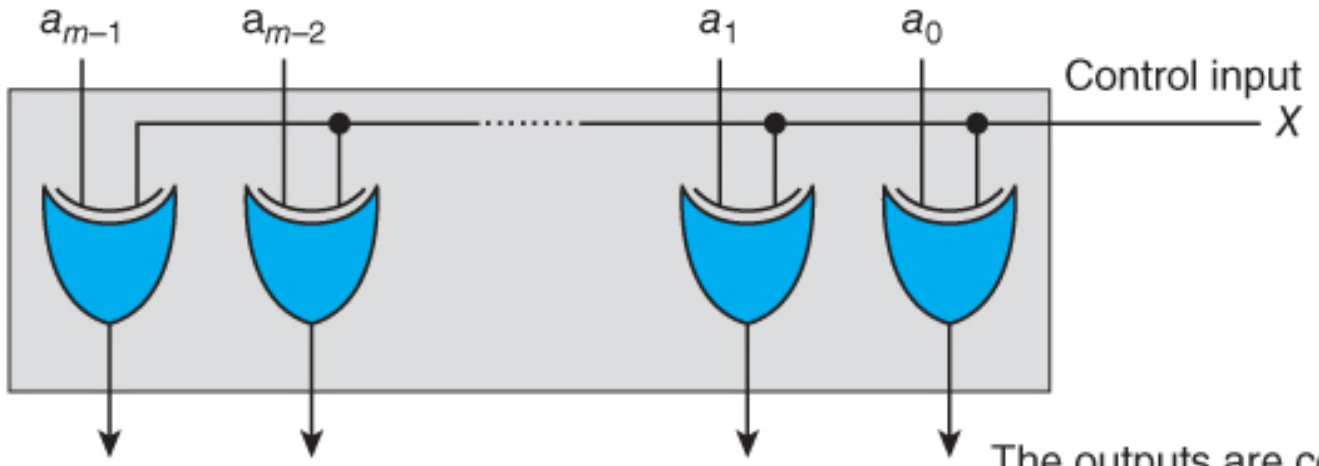
FIGURE 2.25 The parallel adder



Programmable Inverter

a	X	a ⊕ X
0	0	0
0	1	1
1	0	1
1	1	0

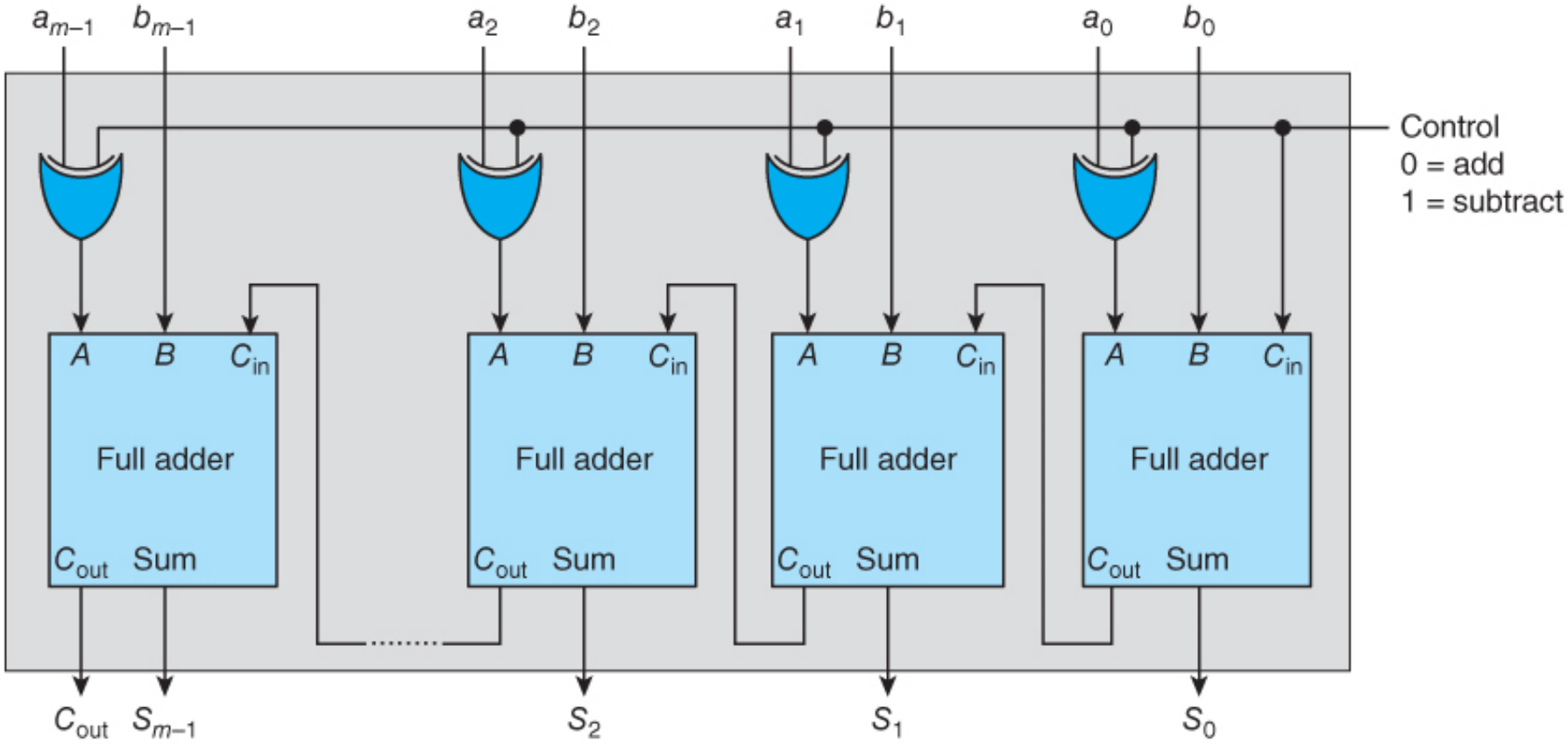
FIGURE 2.26 The programmable inverter



The outputs are copies of the respective inputs if $X = 0$, and the complements of the inputs if $X = 1$

Full-Adder/Subtractor

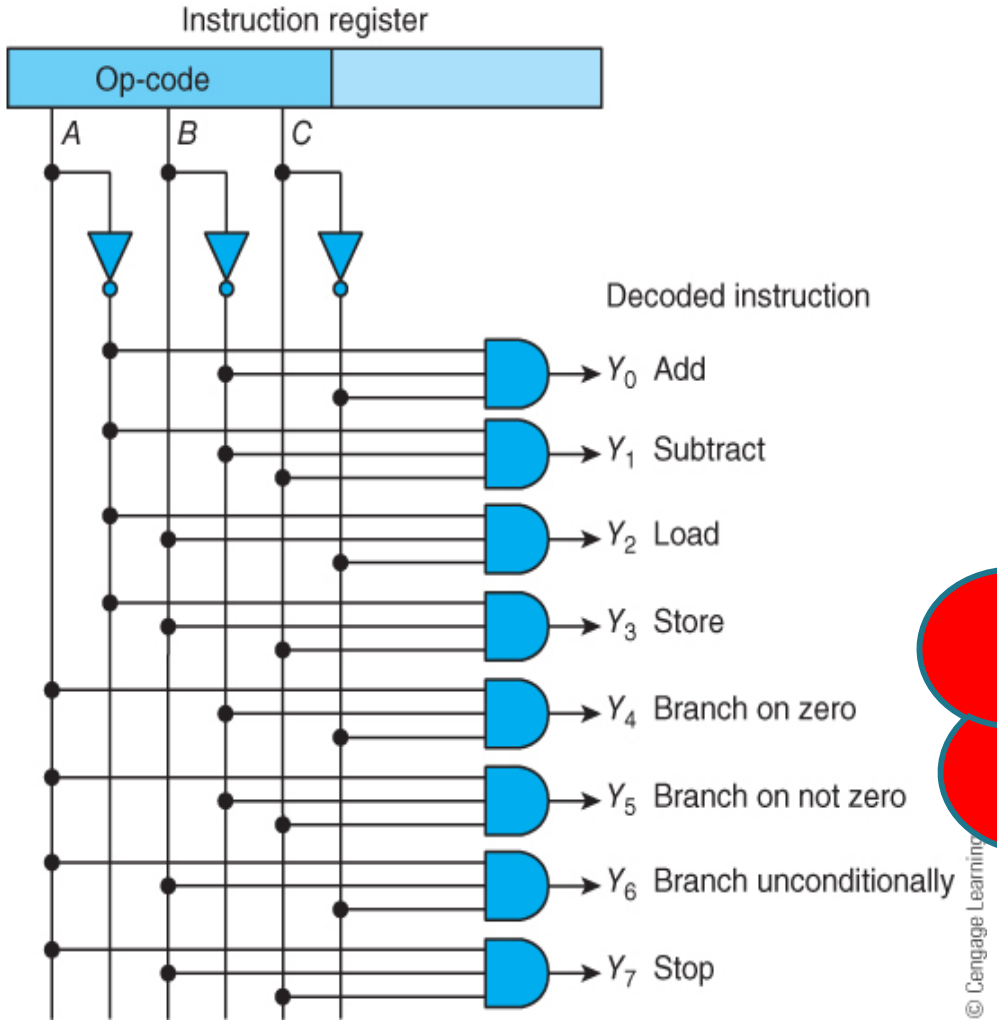
FIGURE 2.27 The adder/subtractor



The Decoder

- Figure 2.29 has ***three*** inputs A, B, and C, and ***eight*** outputs Y0 to Y7.
- The ***three*** inverters generate the complements of the inputs A, B, and C.
- Each of the ***eight*** ***AND*** gates is connected to ***three*** of the six lines .
  - each of the ***three*** variables appear in either its true or complemented form.

FIGURE 2.28 Application of a decoder



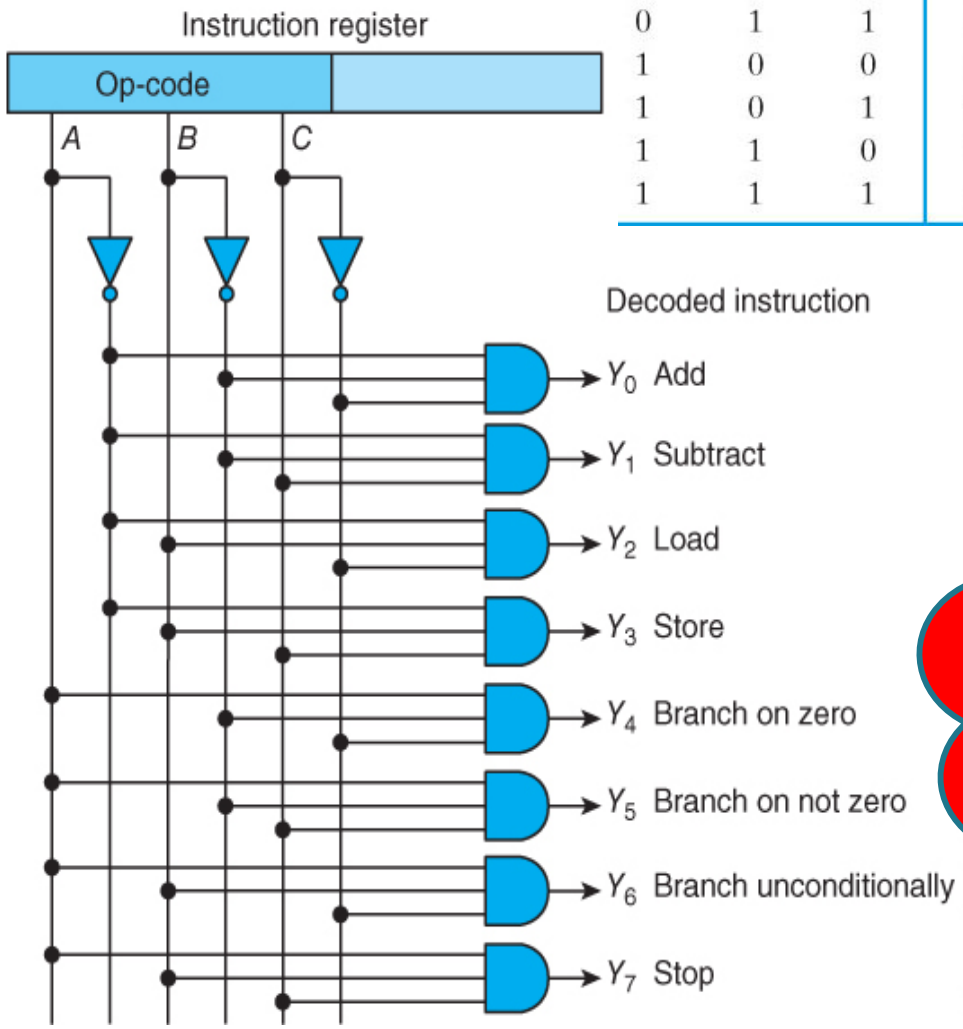
A ***decoder*** is combinational logic circuit that converts binary information from the n-bits coded input to a maximum of 2^n unique outputs.

The Decoder

TABLE 2.15 The Decoder

Inputs			Outputs							
A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

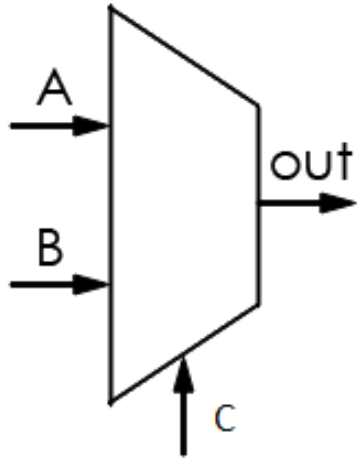
FIGURE 2.28 Application of a decoder



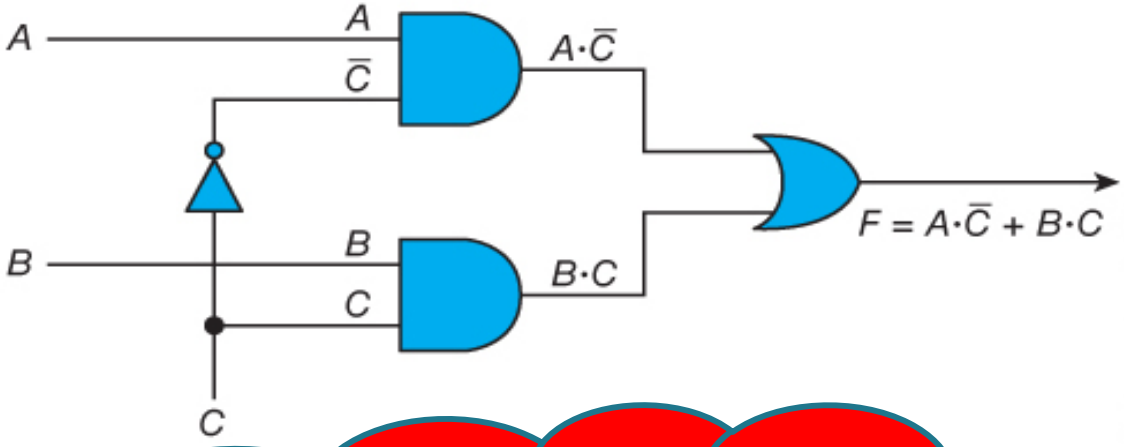
A ***decoder*** is combinational logic circuit that converts binary information from the n-bits coded input to a maximum of 2^n unique outputs.

The Multiplexer

- When C = 0, the output is A
- When C = 1, the output is B
- C works as a selector to select either A or B to go



Alternative representation of the two-input multiplexer



Truth table

C	A	B	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A ***multiplexer*** is combinational logic circuit that has up to 2^n binary input lines and n select lines, where the n select-lines are used to forward one of the input values to the output line.

The Multiplexer

- When C = 0, the output is A
- When C = 1, the output is B
- C works as a selector to select either A or B to go

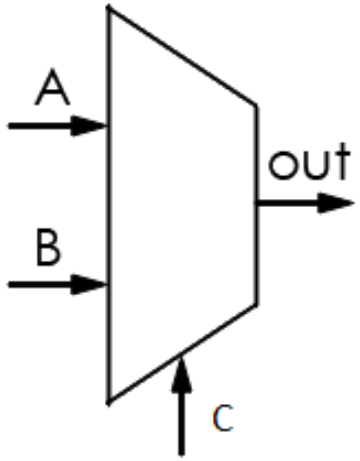
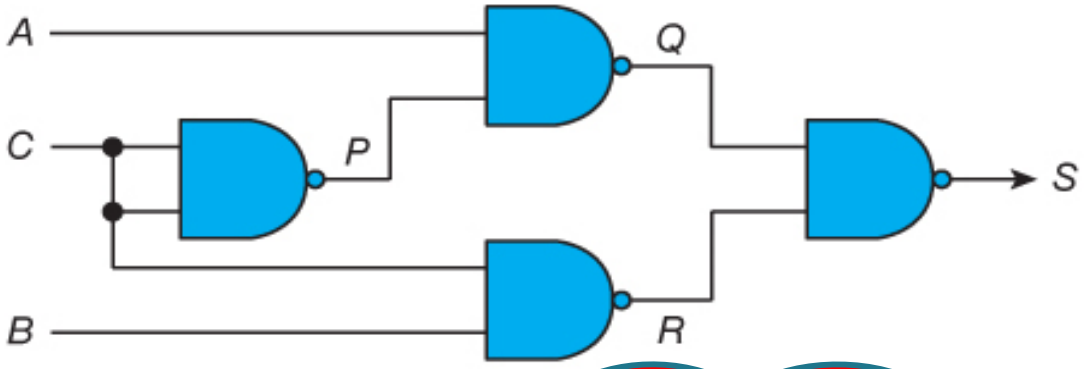


FIGURE 2.29 The two-input multiplexer and its truth table

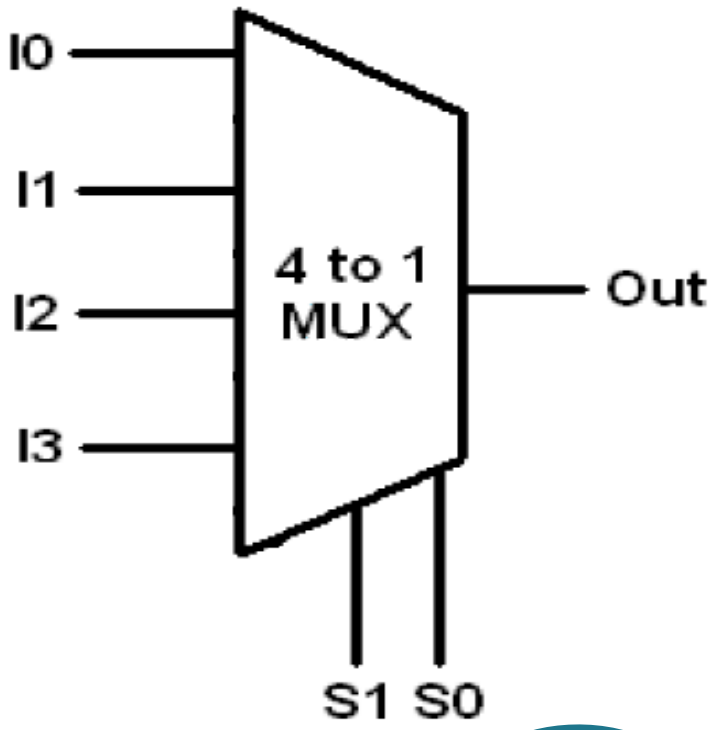


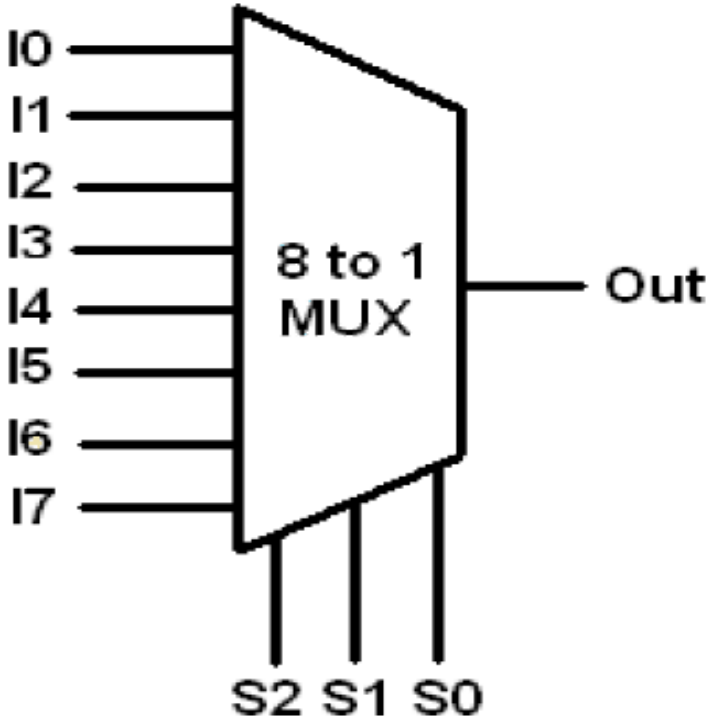
Truth table

C	A	B	P	Q	R	S
0	0	0	1	1	1	0
0	0	1	1	1	1	0
0	1	0	1	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	0
1	0	1	0	1	0	1
1	1	0	0	1	1	0
1	1	1	0	1	0	1

A ***multiplexer*** is combinational logic circuit that has up to 2^n binary input lines and n select lines, where the n select-lines are used to forward one of the input values to the output line.

The Multiplexer





A *multiplexer* is combinational logic circuit that has up to 2^n binary input lines and n select lines, where the n select-lines are used to forward one of the input values to the output line.

One Bit of an ALU

- This diagram describes one-bit of a primitive ALU that can perform five operations on bits A and B (**XOR**, **AND**, **OR**, **NOT A** and **NOT B**).
- The function to be performed is determined by the *three-bit control signal* F2,F1,F0.
- The five functions are generated by the five gates on the left.
- On the right, five **AND** gates are used to gate the selected function to an **OR** gate to produce the output.