Digital Fundamentals: A Systems Approach

Thomas L. Floyd

Functions of Combinational Logic

Chapter 5

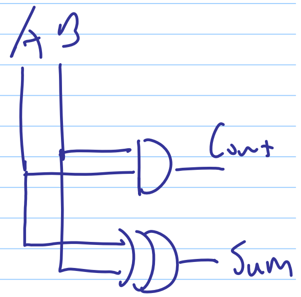
ALWAYS LEARNING PEARSON

Half Adder

①

inputs A	B	outputs C_{out}	Sum(Σ)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

③



②

$C_{out} = AB$

$Sum = \bar{A}B + A\bar{B} = A \oplus B$

Ch.5 Summary

Half-Adders

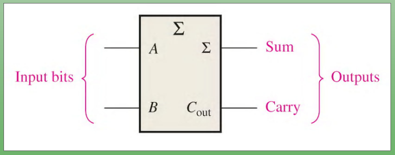
Basic rules of binary addition are performed by a **half adder**, which accepts two binary inputs (*A* and *B*) and provides two binary outputs (Carry-out and Sum).

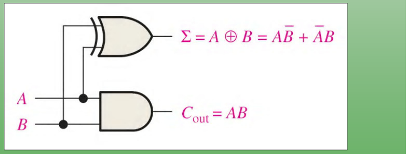
TABLE 5–1 • Half-adder truth table.

A	*B*	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum
C_{out} = output carry
A and *B* = input variables (operands)

The logic symbol and equivalent circuit are:





Full Adder

①

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

②

$C_{out} = \bar{A}BC_{in}$
$+ A\bar{B}C_{in}$
$+ AB\bar{C}_{in}$
$+ ABC_{in}$

$Sum = \bar{A}\bar{B}C_{in}$
$+ \bar{A}B\bar{C}_{in}$
$+ A\bar{B}\bar{C}_{in}$
$+ ABC_{in}$

② -1

$$C_{out} = \bar{A} B C_{in} + A \bar{B} C_{in} + A B \bar{C}_{in} + A B C_{in}$$

$\Leftarrow C_{in}(\bar{A}B + A\bar{B})$

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

$\Leftarrow AB(C_{in} + \bar{C}_{in}) = AB \cdot 1 = AB$

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

② -2

$$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

$= (\bar{A}\bar{B} + AB)\, C_{in}$

$+ (\bar{A}B + A\bar{B})\, \bar{C}_{in}$

$= (A \odot B)\, C_{in}$

$+ (A \oplus B)\, \bar{C}_{in}$

$\bar{X}Y + X\bar{Y} \Leftarrow$ $= \overline{A \oplus B} \cdot C_{in}$

$= X \oplus Y$ $\quad X = A \oplus B$ $\quad + (A \oplus B) \cdot \bar{C}_{in}$

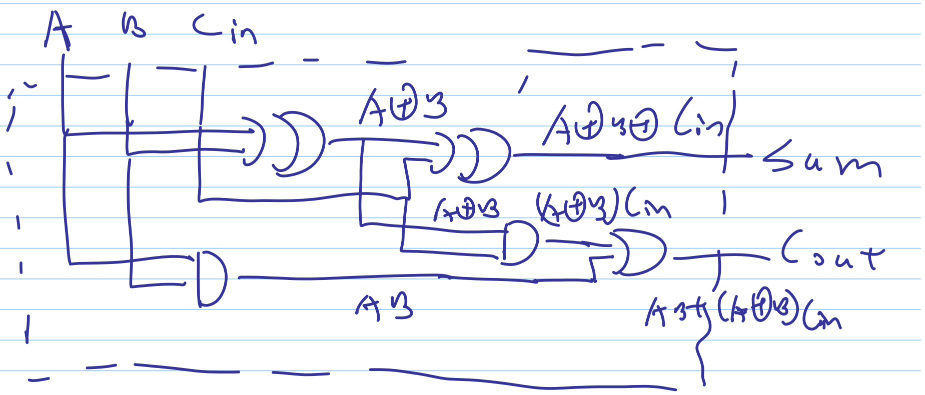
$Y = C_{in}$

$= A \oplus B \oplus C_{in}$

Sectin 5.2

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

③ $Sum = A \oplus B \oplus C_{in}$

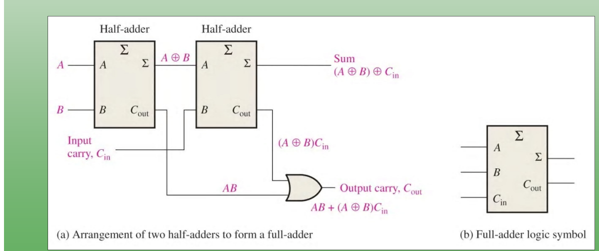


Ch.5 Summary

Full-Adders

A **full adder** accepts three binary inputs (*A*, *B*, and Carry-in) and provides two binary outputs (Carry-out and Sum). The truth table summarizes the operation.

A full-adder can be constructed from two half adders as shown:



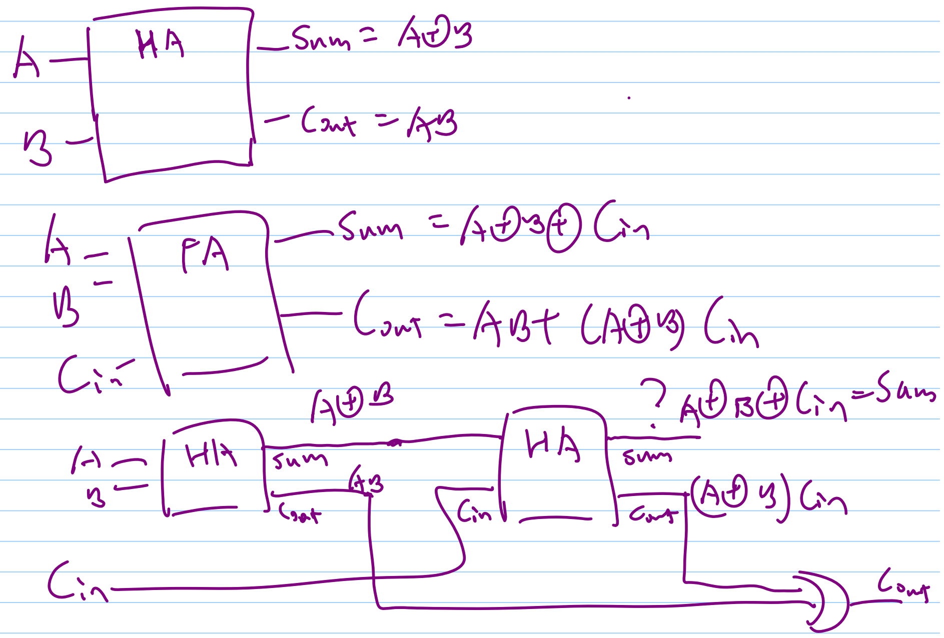
(a) Arrangement of two half-adders to form a full-adder

(b) Full-adder logic symbol

TABLE 5–2 • Full-adder truth table.

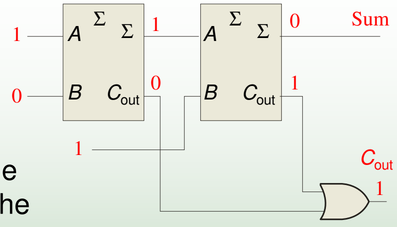
A	*B*	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = input carry, sometimes designated as *CI*
C_{out} = output carry, sometimes designated as *CO*
Σ = sum
A and *B* = input variables (operands)



Ch.5 Summary

Full-Adders

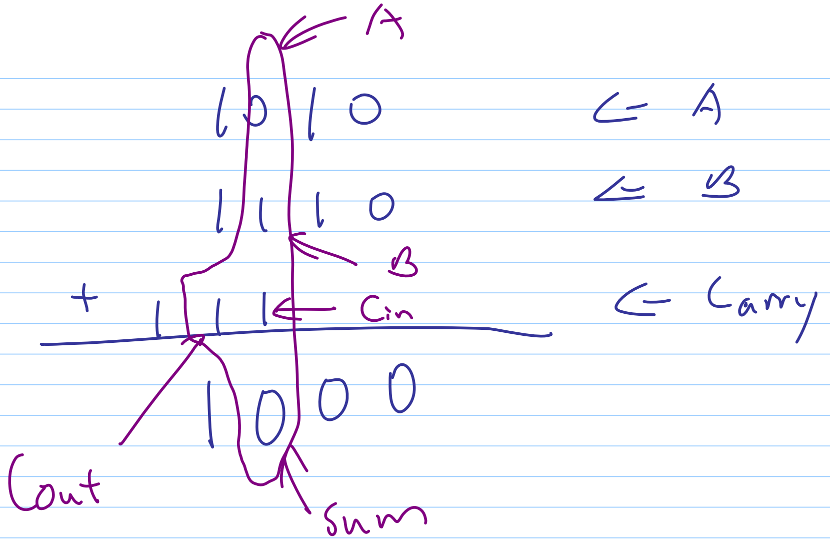


For the given inputs, determine the intermediate and final outputs of the full adder.

The first half-adder has inputs of 1 and 0; therefore the Sum =1 and the Carry out = 0.

The second half-adder has inputs of 1 and 1; therefore the Sum = 0 and the Carry out = 1.

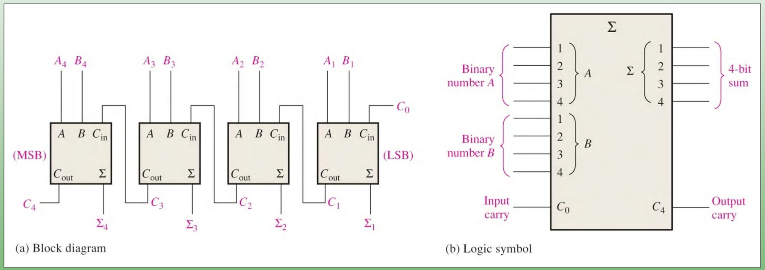
The OR gate has inputs of 1 and 0, therefore the final carry out = 1.



Ch.5 Summary

Parallel Adders

Full adders are combined into parallel adders that can add binary numbers with multiple bits. A 4-bit adder is shown.



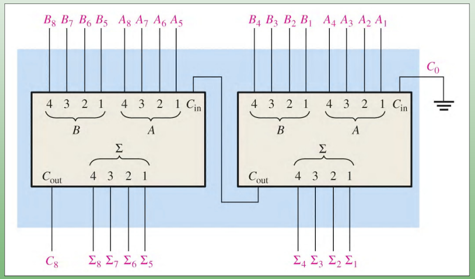
(a) Block diagram

(b) Logic symbol

The output carry (C_4) is not ready until it propagates through all of the full adders. This is called *ripple carry*, which delays the addition process.

Adder Expansion

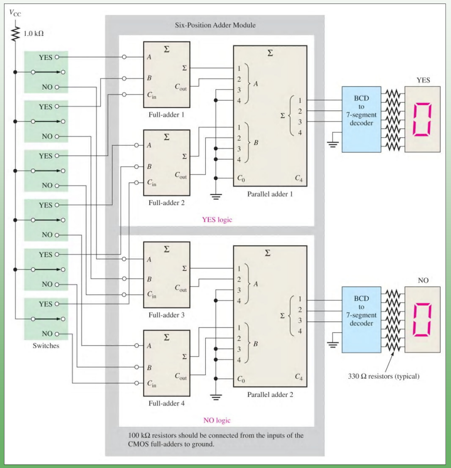
Two four-bit adders can be **cascaded** to form an 8-bit adder as shown.


B_8 B_7 B_6 B_5 A_8 A_7 A_6 A_5
B_4 B_3 B_2 B_1 A_4 A_3 A_2 A_1
4 3 2 1 4 3 2 1 C_{in}
B
A
Σ
C_{out} 4 3 2 1
C_0
C_8
Σ_8 Σ_7 Σ_6 Σ_5
Σ_4 Σ_3 Σ_2 Σ_1

The carry-in (C_0) pin on the lower-order adder is grounded and the carry-out pin is connected to the C_0 pin of the higher-order adder.

An Adder Application

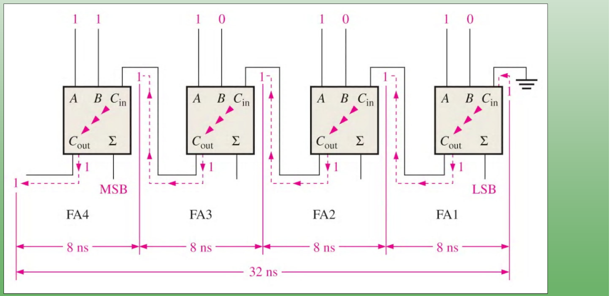
A voting system



Ripple Carry Adder

The parallel adder introduced earlier is a **ripple carry adder**. Because the carry from each adder is applied to the next, time must provided for the carry bits to "ripple" through the circuit.

If each adder has an 8 ns delay, it takes 32 ns for the carry to work its way through the adder (as shown).

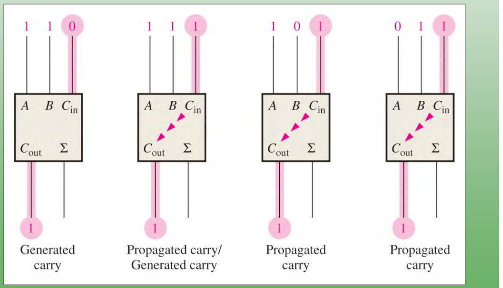


Look-Ahead Carry Adder

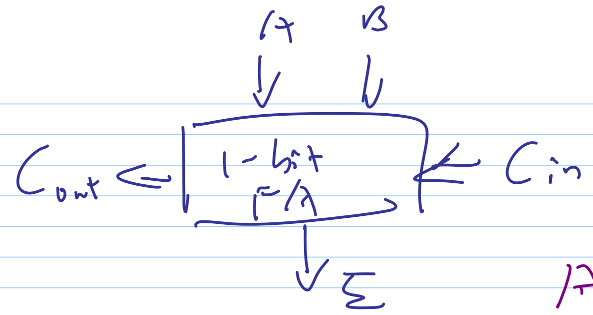
A **look-ahead carry adder** anticipates the carry that will be generated by each adder stage, and produces the required carry at one time, making the adder faster than a ripple carry adder.

Carry generation: When a carry signal is produced internally (i.e., a carry-in is not involved).

Carry propagation: When an input carry signal is involved in producing a carry out signal.



Section 5.3



$$A + A = A$$
$$A + A + A = A$$

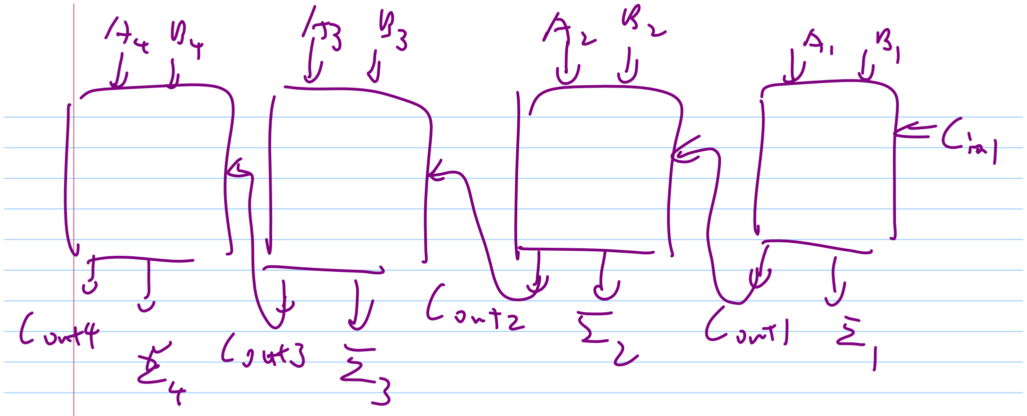
$$C_{out} = \bar{A} B C_{in} + A \bar{B} C_{in} + A B \bar{C}_{in} + A B C_{in}$$

$$= AB + AC_{in} + BC_{in}$$

$$= \underbrace{AB}_{C_g} + \underbrace{(A+B)}_{C_p} C_{in}$$

$$C_{g1} = A_1 B_1$$
$$C_{p1} = A_1 + B_1$$
$$C_{g2} = A_2 B_2$$
$$C_{p2} = A_2 + B_2$$
$$C_{g3} = A_3 B_3$$
$$C_{p3} = A_3 + B_3$$

$$C_{g4} = A_4 B_4$$
$$C_{p4} = A_4 + B_4$$



$$C_{out1} = C_{g1} + C_{p1} C_{in1} \quad (1)$$

$$C_{out2} = C_{g2} + C_{p2} C_{out1} \quad (2)$$

$$= C_{g2} + C_{p2} (C_{g1} + C_{p1} C_{in1})$$

$$= C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} C_{in1}$$

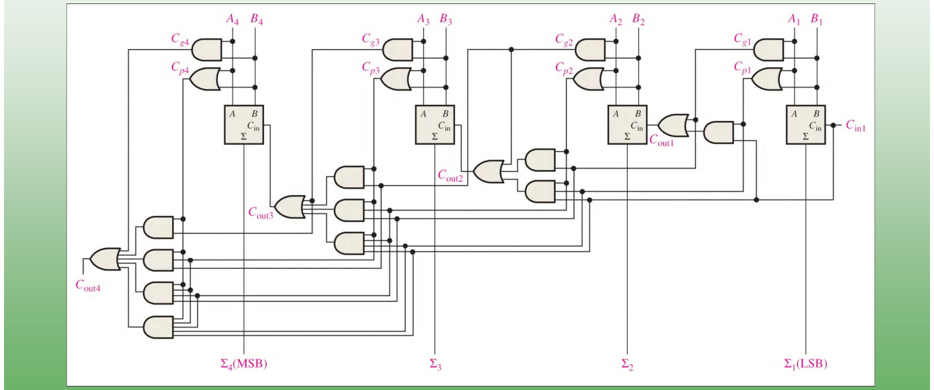
$$C_{out3} = C_{g3} + C_{p3} C_{out2}$$

$$C_{out4} = C_{g4} + C_{p4} C_{out3}$$

$$= f(C_{g1}, C_{p1}, C_{g2}, C_{p2}, C_{g3}, C_{p3}, C_{g4}, C_{p4}, C_{in1})$$

$$= f(A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4, C_{in})$$

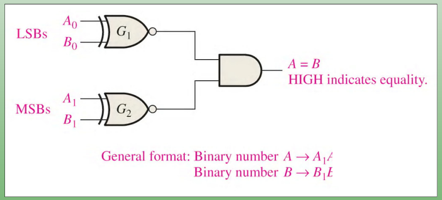
A Four-Stage Look-Ahead Carry Adder



Comparators

The function of a comparator is to compare the magnitudes of two binary numbers to determine the relationship between them. In the simplest form, a comparator can test for equality using XNOR gates.

How could you test two 2-bit numbers for equality?

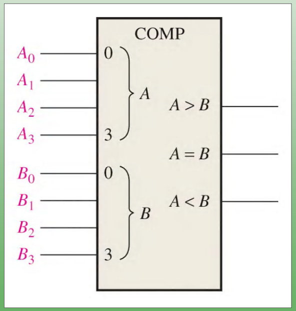

LSBs
A_0
B_0
G_1
MSBs
A_1
B_1
G_2
$A = B$
HIGH indicates equality.
General format: Binary number $A \rightarrow A_1A$
Binary number $B \rightarrow B_1B$

AND the outputs of two XNOR gates

Comparators

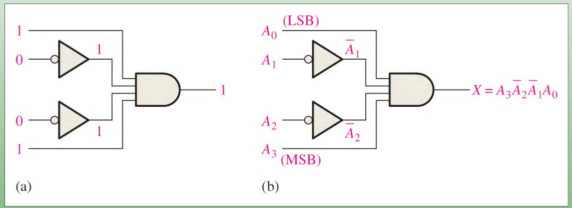
IC comparators provide outputs to indicate which of the numbers is larger or if they are equal. The bits are numbered starting at 0, rather than 1 as in the case of adders.

Cascading inputs are provided to expand the comparator to larger numbers.



Decoders

A **decoder** is a logic circuit that detects the presence of a specific combination of bits at its input. A simple decoder that detects the presence of the binary code 1001 is shown.

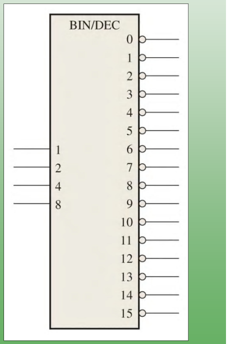


The circuit on the left has an active HIGH output for the inputs shown; the circuit on the right shows the logic expressions for the various gate outputs.

Decoders

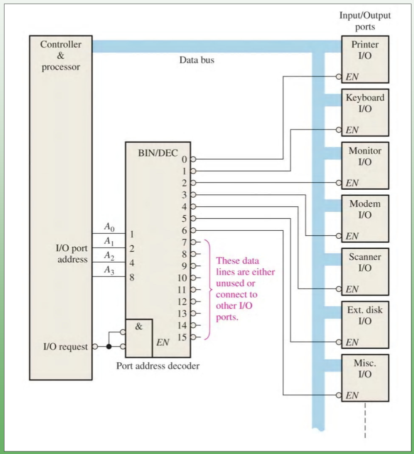
IC decoders have multiple outputs to decode any combination of inputs. For example the binary-to-decimal decoder shown here has 16 outputs – one for each combination of binary inputs. The first 8 lines of the circuit truth table are shown.

DECIMAL DIGIT	BINARY INPUTS A_3	A_2	A_1	A_0	DECODING FUNCTION	OUTPUTS 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1



A Decoder Application

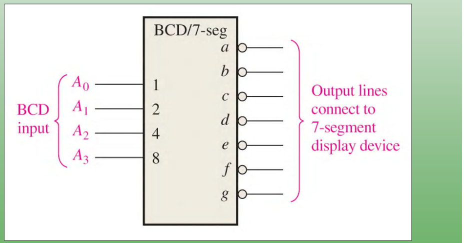
A simplified I/O Port System



BCD Decoder/Driver

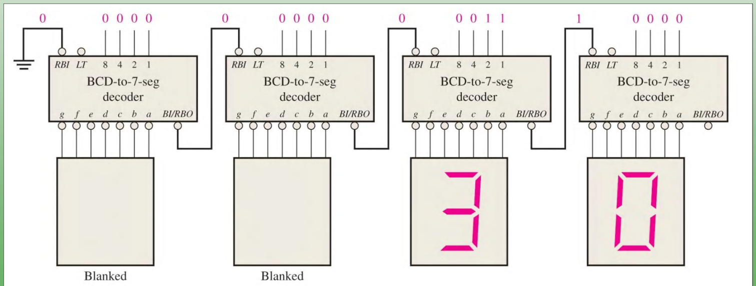
Another useful decoder is the 74LS47. This is a BCD-to-seven segment display with active LOW outputs.

The *a-g* outputs are designed for much higher current than most devices (hence the word driver in the component's name).



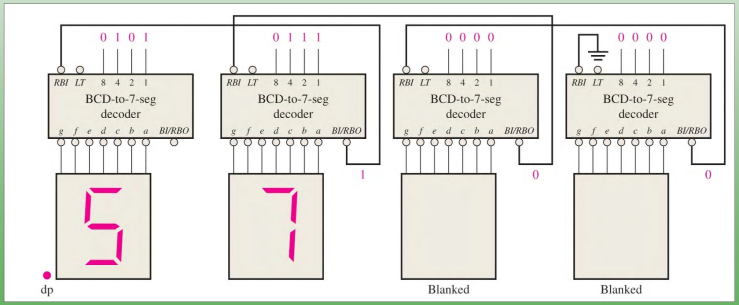
Leading Zero Suppression

The 74LS47 features leading zero suppression, which blanks unnecessary leading zeros but keeps significant zeros as illustrated here. The $\overline{BI}/\overline{RBO}$ output is connected to the $\overline{RBI}$ input of the next decoder.



Trailing Zero Suppression

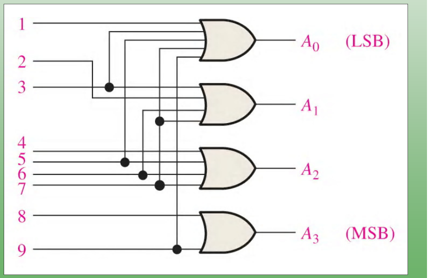
Trailing zero suppression blanks unnecessary trailing zeros to the right of the decimal point as illustrated here. The *RBI* input is connected to the *BI/RBO* output of the following decoder.



Encoders

An **encoder** accepts an active logic level on one of its inputs and converts it to a coded output, such as BCD or binary.

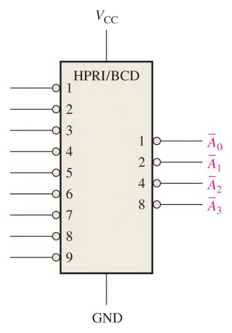
The decimal to BCD is an encoder with an input for each of the ten decimal digits and four outputs that represent the BCD code for the active digit. The basic logic diagram is shown. There is no zero input because the outputs are all LOW when the input is zero.



Encoders

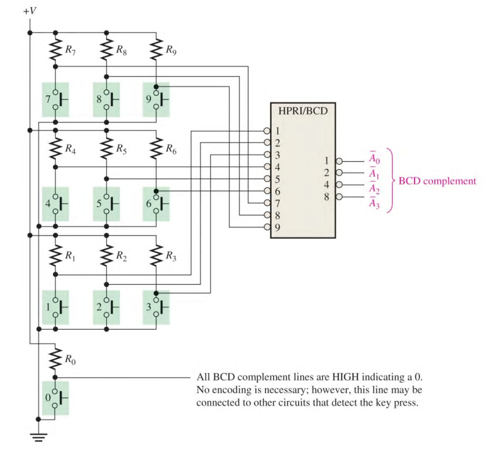
The 74HC147 is an example of an IC encoder. It is has ten active-LOW inputs and converts the active input to an active-LOW BCD output.

This device is a **priority encoder**. This means that if more than one input is active, the component responds to the highest numbered input.

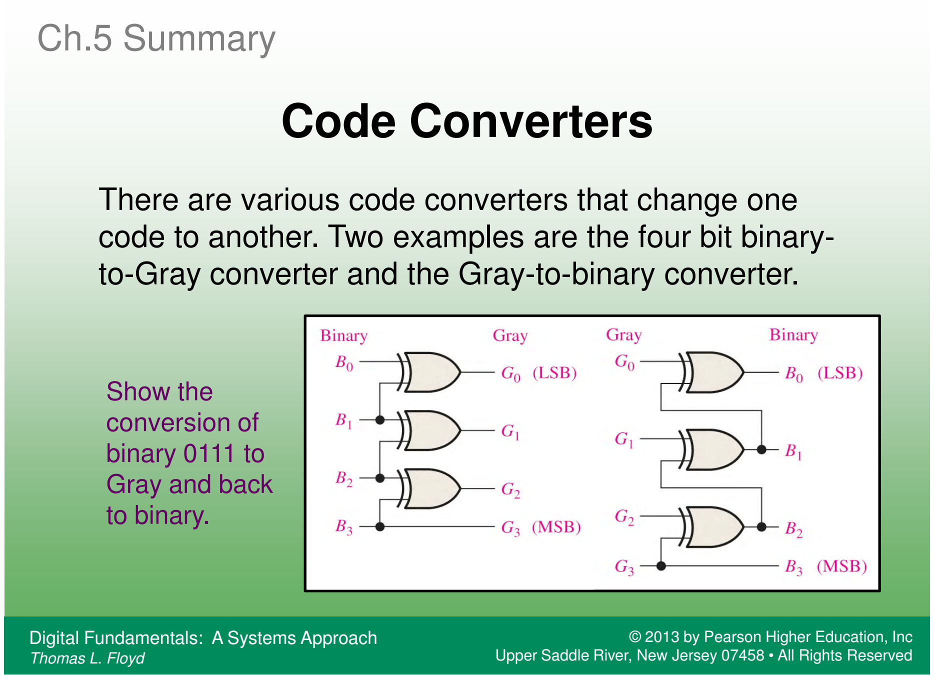


An Encoder Application

A keypad encoder



All BCD complement lines are HIGH indicating a 0. No encoding is necessary; however, this line may be connected to other circuits that detect the key press.



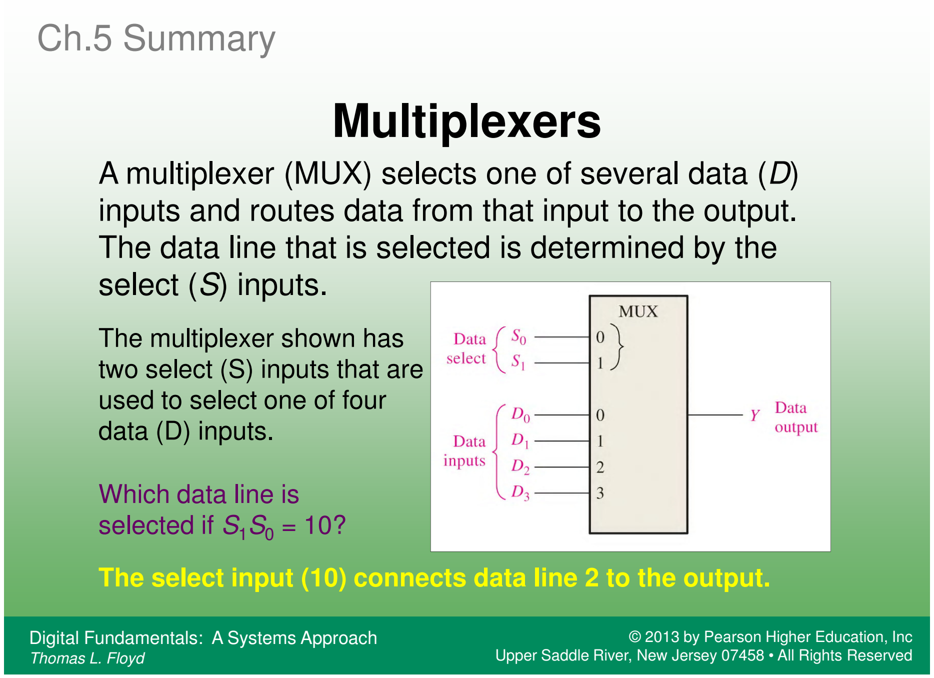
Ch.5 Summary

Code Converters

There are various code converters that change one code to another. Two examples are the four bit binary-to-Gray converter and the Gray-to-binary converter.

Show the conversion of binary 0111 to Gray and back to binary.

Digital Fundamentals: A Systems Approach
Thomas L. Floyd



Ch.5 Summary

Multiplexers

A multiplexer (MUX) selects one of several data (*D*) inputs and routes data from that input to the output. The data line that is selected is determined by the select (*S*) inputs.

The multiplexer shown has two select (S) inputs that are used to select one of four data (D) inputs.

Which data line is selected if $S_1S_0 = 10$?

The select input (10) connects data line 2 to the output.

Digital Fundamentals: A Systems Approach
Thomas L. Floyd

if $S_1S_0 == 00$ then $Y = D_0$

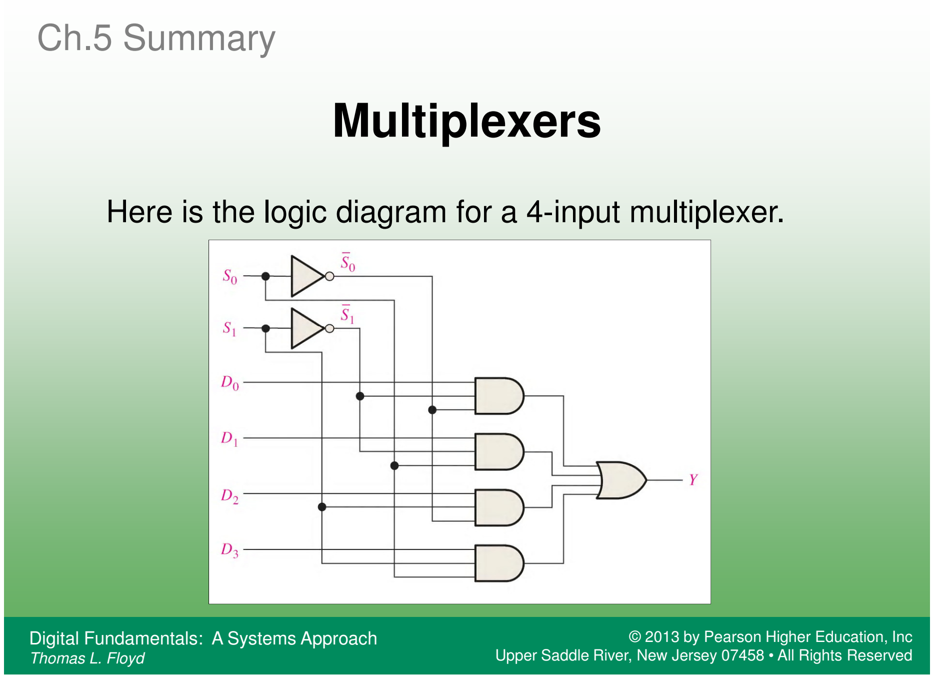
if $S_1S_0 == 01$ then $Y = D_1$

if $S_1S_0 = 10$ then $Y = D_2$

if $S_1S_0 = 11$ then $Y = D_3$

S_1S_0	Y
00	D_0
01	D_1
10	D_2
11	D_3

$$Y = \bar{S}_1\bar{S}_0D_0 + \bar{S}_1S_0D_1 + S_1\bar{S}_0D_2 + S_1S_0D_3$$



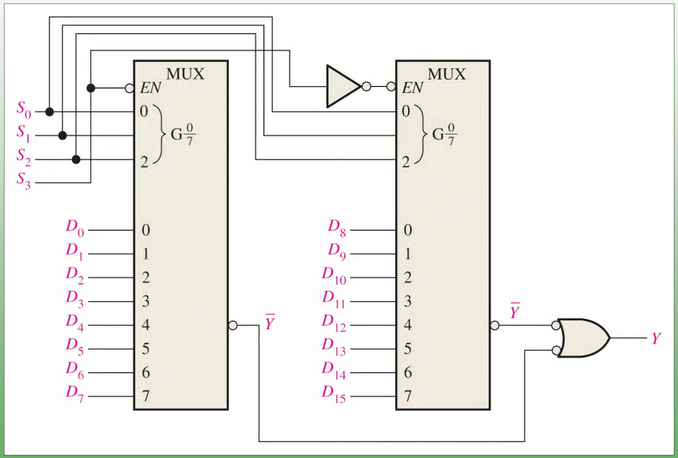
Ch.5 Summary

Multiplexers

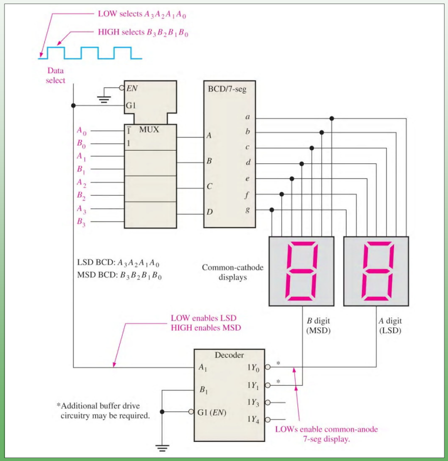
Here is the logic diagram for a 4-input multiplexer.

Digital Fundamentals: A Systems Approach
Thomas L. Floyd

A 16-Bit Multiplexer



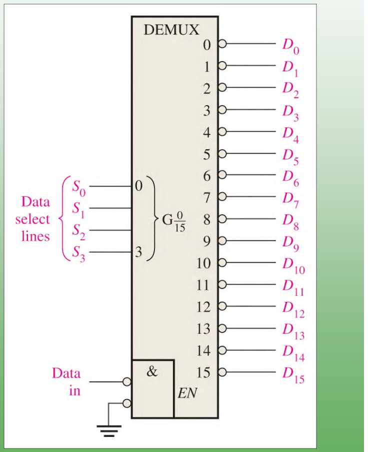
A 7-Segment Display Multiplexer



Demultiplexers

A demultiplexer (DEMUX) performs the opposite function from a MUX. It switches data from one input line to two or more data lines depending on the select inputs.

Data is applied to one of the data input pin, and routed to the selected output line depending on the select variables. Note that the outputs are active-LOW.



Parity Generators/Checkers

Parity is an error detection method that uses an extra bit appended to a group of bits to force them to be either odd or even. In even parity, the total number of ones is even; in odd parity the total number of ones is odd.

The ASCII letter S is 1010011. Show the parity bit for the letter S with odd and even parity.

S with odd parity = 11010011

S with even parity = 01010011

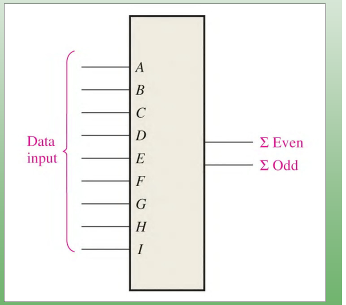
Ch.5 Summary

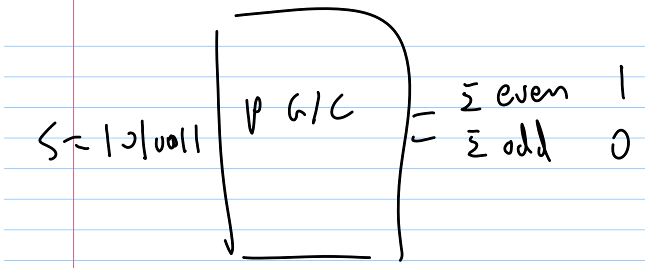
Parity Generators/Checkers

A 9-bit parity checker/generator can be used to generate a parity bit or to check an incoming data stream for even or odd parity.

Checker: The even output will normally be HIGH if the data lines have even parity; otherwise it will be LOW. Likewise, the odd output will normally be HIGH if the data lines have odd parity; otherwise it will be LOW.

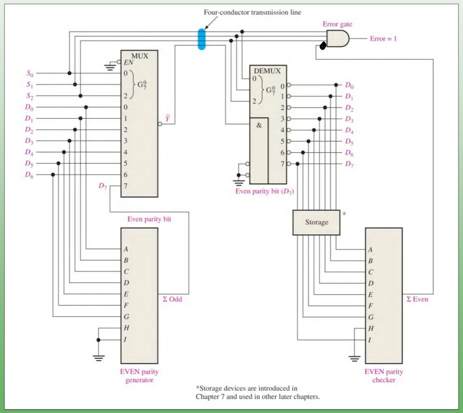
Generator: To generate even parity, the parity bit is taken from the odd parity output. To generate odd parity, the output is taken from the even parity output.



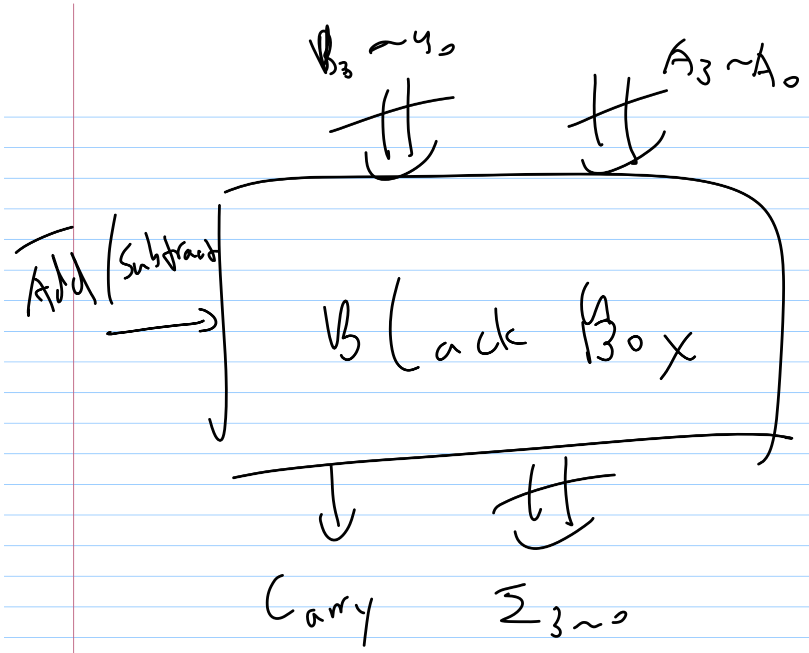


Ch.5 Summary

A Simplified Data Transmission System with Error Detection



*Storage devices are introduced in Chapter 7 and used in other later chapters.



XOR

$$B \oplus 0 = B\bar{0} + \cancel{\bar{B}\cdot 0} = B\cdot 1 = B$$

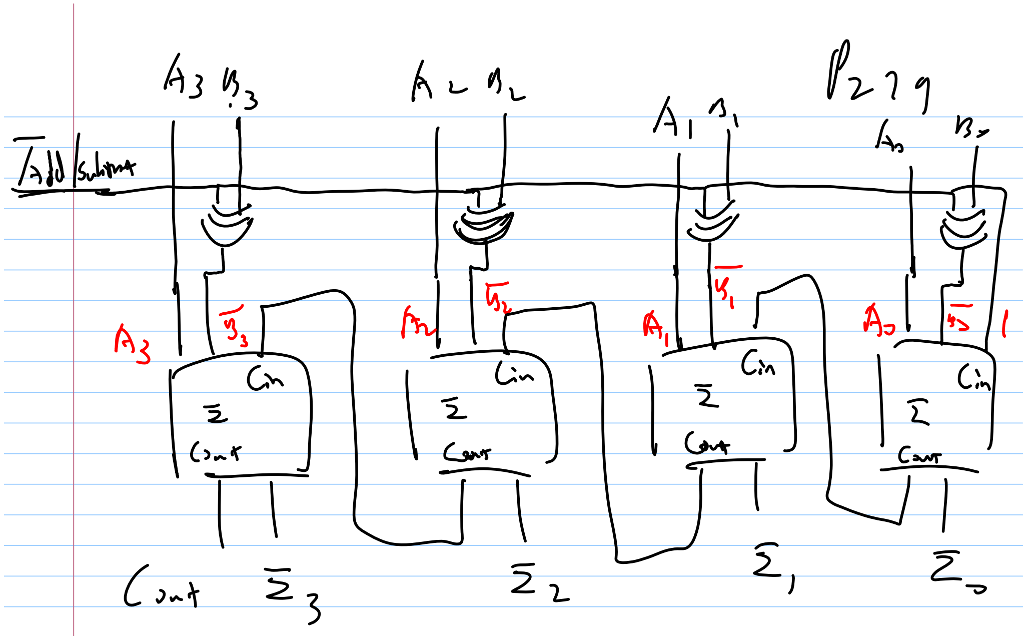
$$B \oplus 1 = \cancel{B\cdot\bar{1}} + \bar{B}\cdot 1 = \bar{B}\cdot 1 = \bar{B}$$

$$A - B = A_3 A_2 A_1 A_0$$

$$+ \left(-B_3 B_2 B_1 B_0\right)$$

$$\bar{B}_3 \bar{B}_2 \bar{B}_1 \bar{B}_0 \quad \text{flip}$$

$$+ \quad 1$$

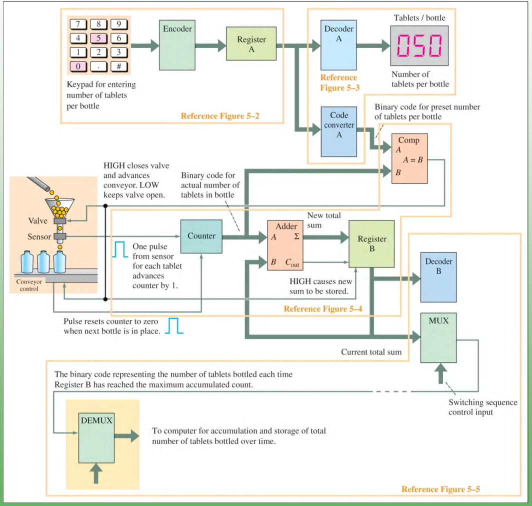


If $\overline{\text{add}}$ / subtract = 0

If $\overline{\text{add}}$ / subtract = 1

Ch.5 Summary

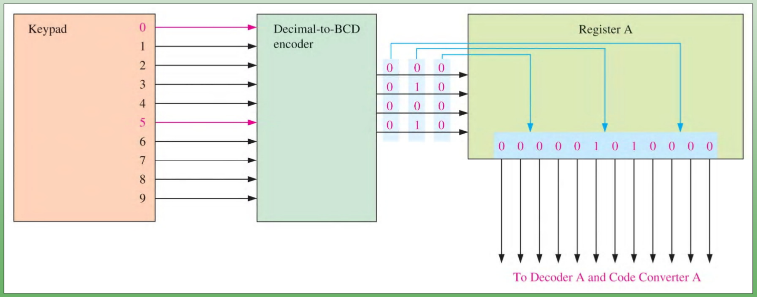
Bottling Control System


Keypad for entering number of tablets per bottle
Encoder
Register A
Reference Figure 5–2
Decoder A
Reference Figure 5–3
Tablets / bottle
050
Number of tablets per bottle
Binary code for preset number of tablets per bottle
Code converter A
Comp A A = B B
HIGH closes valve and advances conveyor. LOW keeps valve open.
Binary code for actual number of tablets in bottle
Valve
Sensor
Conveyor control
Counter
One pulse from sensor for each tablet advances counter by 1.
Adder A Σ B Cout
New total sum
Register B
HIGH causes new sum to be stored.
Reference Figure 5–4
Decoder B
MUX
Pulse resets counter to zero when next bottle is in place.
Current total sum
The binary code representing the number of tablets bottled each time Register B has reached the maximum accumulated count.
Switching sequence control input
DEMUX
To computer for accumulation and storage of total number of tablets bottled over time.
Reference Figure 5–5

Ch.5 Summary

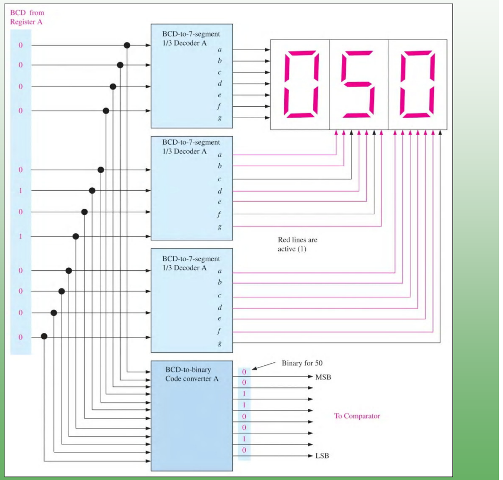
Bottling Control System

Information is entered into the system using a keypad, encoder, and register.



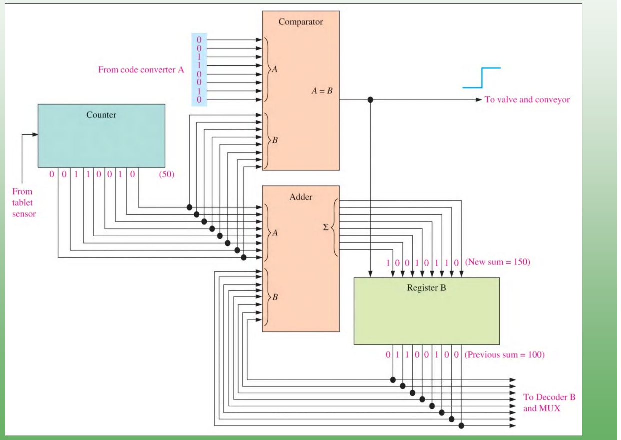
Bottling Control System

The display circuit contains decoder and code converter circuits.


BCD from Register A
0
0
0
0
0
1
0
1
0
0
0
0
BCD-to-7-segment 1/3 Decoder A
a
b
c
d
e
f
g
BCD-to-7-segment 1/3 Decoder A
BCD-to-7-segment 1/3 Decoder A
0 5 0
Red lines are active (1)
BCD-to-binary Code converter A
Binary for 50
0
0
1
1
0
0
1
0
MSB
LSB
To Comparator

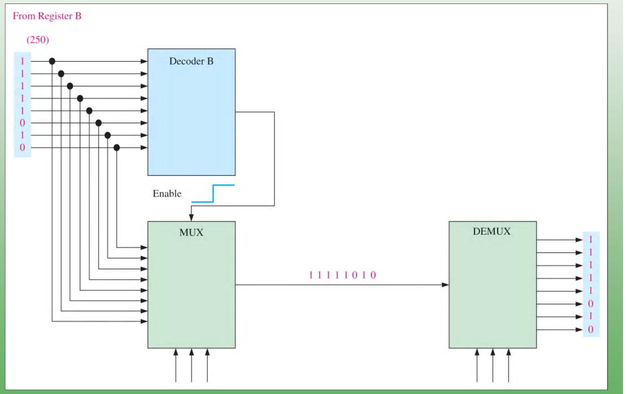
Bottling Control System

The comparator, counter, adder, and register circuits

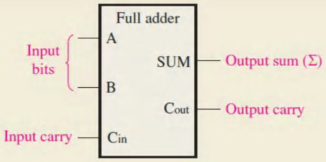


Bottling Control System

The decoder, multiplexer (MUX), and demultiplexer (DEMUX) circuits.



VHDL and Verilog Full Adder



$$\Sigma = (A \oplus B) \oplus C_{\text{in}}$$
$$C_{\text{out}} = AB + (A \oplus B)C_{\text{in}}$$

VHDL

```
entity FullAdder is
   port (A, B, Cin: in bit; SUM, Cout: out bit;
end entity FullAdder;

architecture FA_Function of FullAdder is
begin
    SUM <= (A xor B) xor Cin;
    Cout <= (A and B) or (A xor B) and Cin;
end architecture FA_Function;
```

Verilog

```
module FullAdder (A, B, Cin, SUM, Cout);
   input A, B, Cin;
   output SUM, Cout;
      assign SUM = (A ^ B) ^ Cin;
Cout = (A && B) || (A ^ B) && Cin;
endmodule
```

Ch.5 Summary

Key Terms

Full-adder A digital circuit that adds two bits and an input carry bit to produce a sum and an output carry.

Cascading Connecting two or more similar devices in a manner that expands the capability of one device.

Ripple carry A method of binary addition in which the output carry from each adder becomes the input carry of the next higher order adder.

Look-ahead carry A method of binary addition whereby carries from the preceding adder stages are anticipated, thus eliminating carry propagation delays.

Ch.5 Summary

Key Terms

Decoder A digital circuit that converts coded information into a familiar or noncoded form.

Encoder A digital circuit that converts information into a coded form.

Priority encoder An encoder in which only the highest value input digit is encoded and any other active input is ignored.

Multiplexer (MUX) A circuit that switches digital data from several input lines onto a single output line in a specified time sequence.

Demultiplexer (DEMUX) A circuit that switches digital data from one input line onto a several output lines in a specified time sequence.