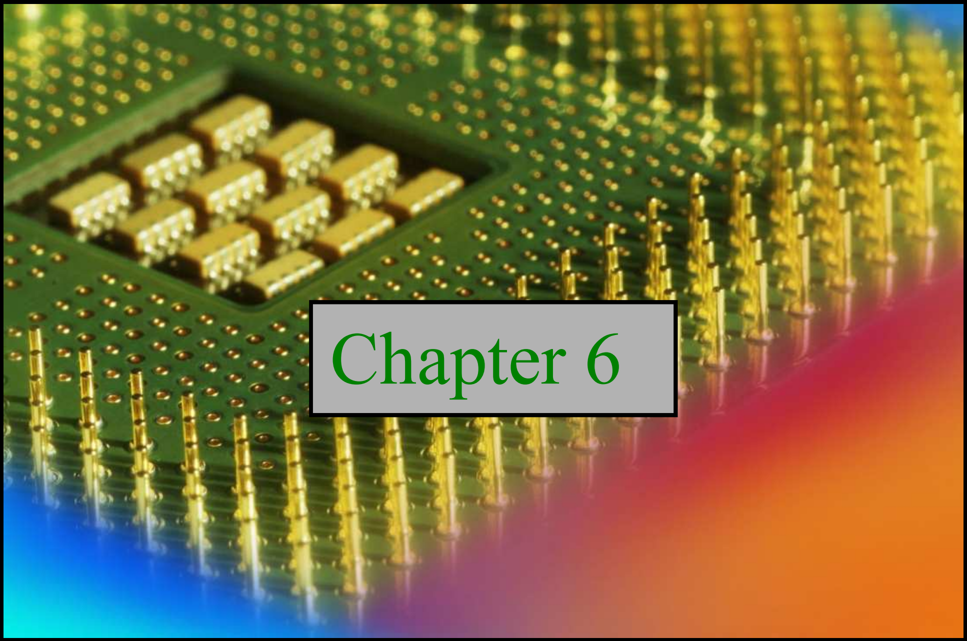
Digital Fundamentals

Tenth Edition

Floyd



Chapter 6

Summary

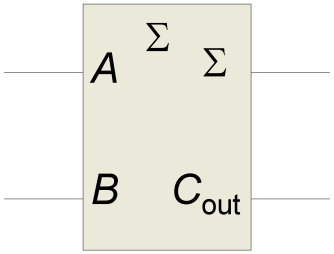
Half-Adder

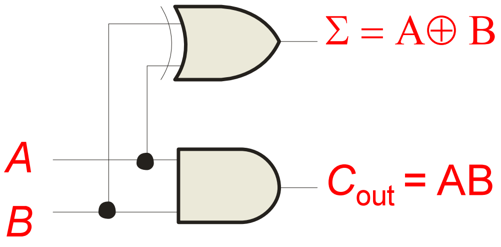
Basic rules of binary addition are performed by a **half adder**, which has two binary inputs (*A* and *B*) and two binary outputs (Carry out and Sum).

The inputs and outputs can be summarized on a truth table.

Inputs		Outputs	
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The logic symbol and equivalent circuit are:





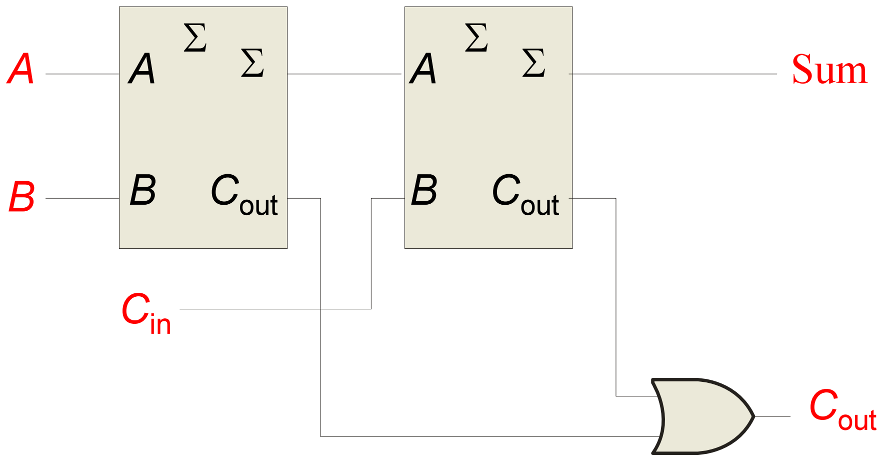
Summary

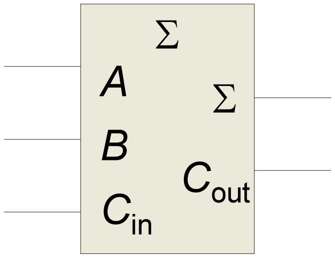
Full-Adder

By contrast, a **full adder** has three binary inputs (*A*, *B,* and Carry in) and 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:

Inputs			Outputs	
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

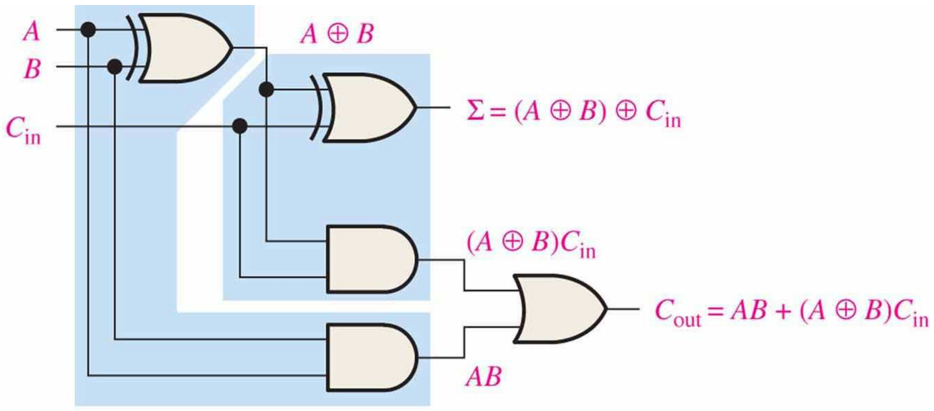




Symbol

Summary

Full-Adder

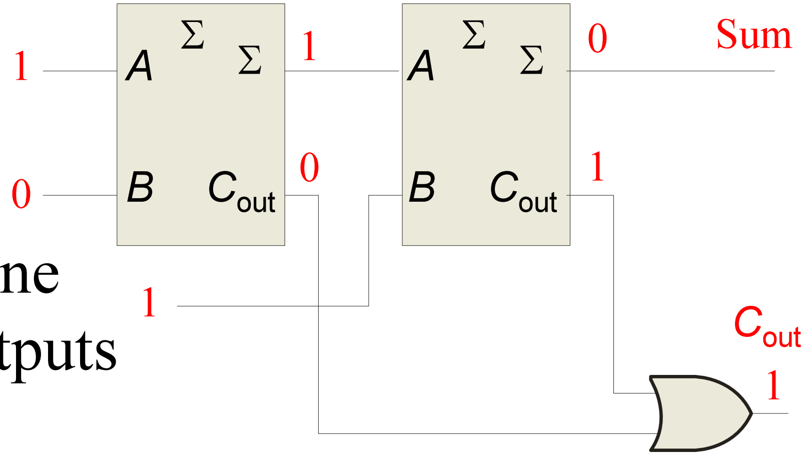

A
B
C_{in}
$A \oplus B$
$\Sigma = (A \oplus B) \oplus C_{in}$
$(A \oplus B)C_{in}$
$C_{out} = AB + (A \oplus B)C_{in}$
AB

(b) Complete logic circuit for a full-adder (each half-adder is enclosed by a shaded area)

Inputs			Outputs	
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

Summary

Full-Adder



Example

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

Solution 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.

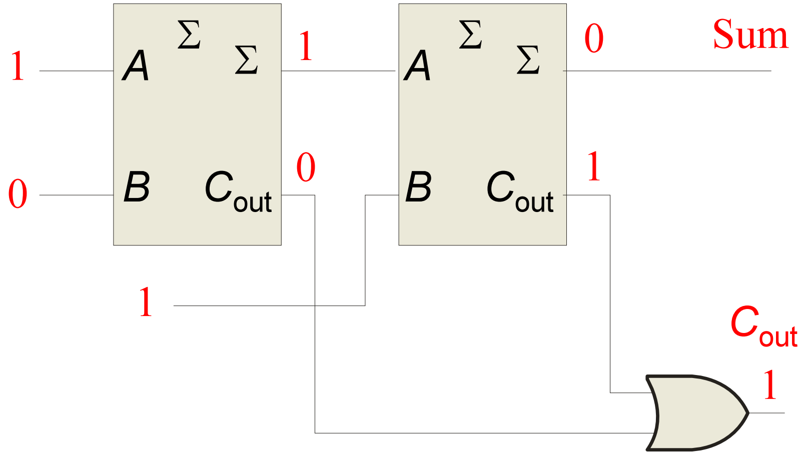
The OR gate has inputs of 1 and 0, therefore the final carry out = 1.

Summary

Full-Adder

Notice that the result from the previous example can be read directly on the truth table for a full adder.

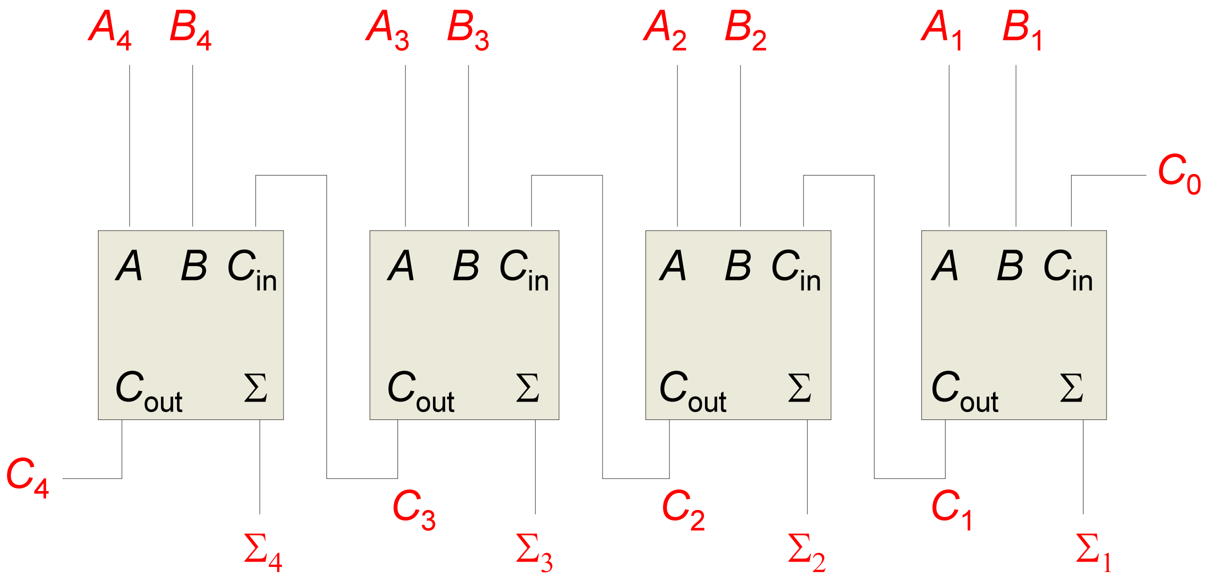
Inputs			Outputs	
A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Summary

Parallel Adders

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

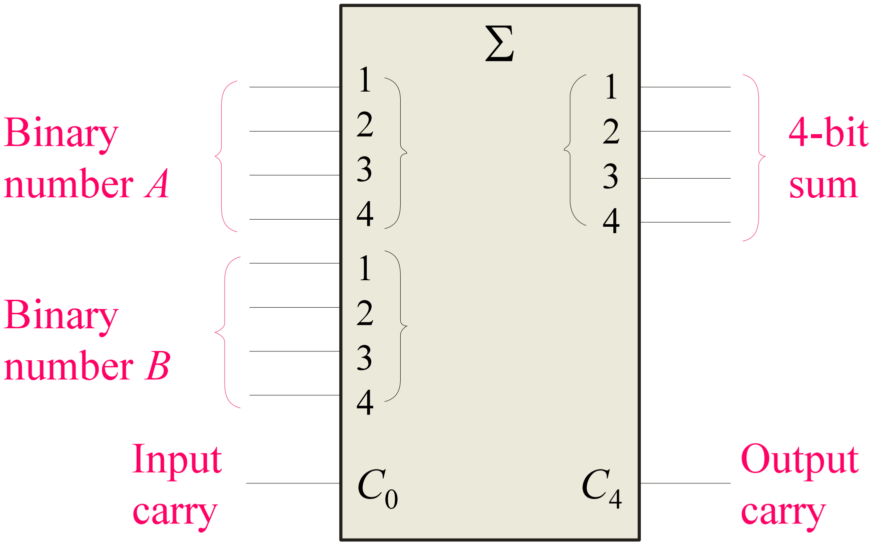


The output carry (C_4) is not ready until it propagates through all of the full adders. This is called *ripple carry*, delaying the addition process.

Summary

Parallel Adders

The logic symbol for a 4-bit parallel adder is shown. This 4-bit adder includes a carry in (labeled (C_0) and a Carry out (labeled C_4).



The 74LS283 is an example. It features *look-ahead carry*, which adds logic to minimize the output carry delay. For the 74LS283, the maximum delay to the output carry is 17 ns.

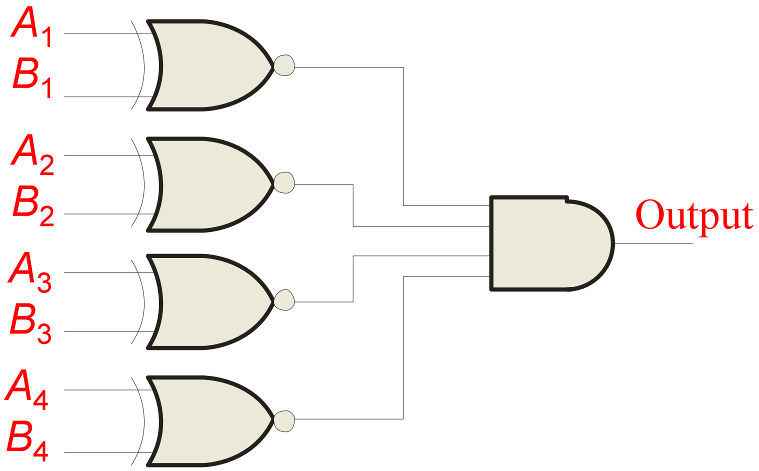
Summary

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.

Example How could you test two 4-bit numbers for equality?

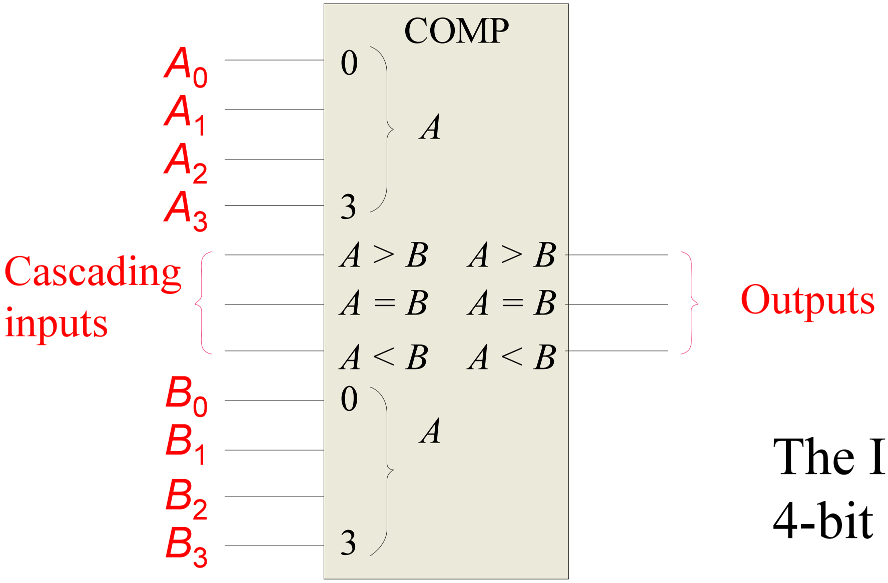
Solution AND the outputs of four XNOR gates.



Summary

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.

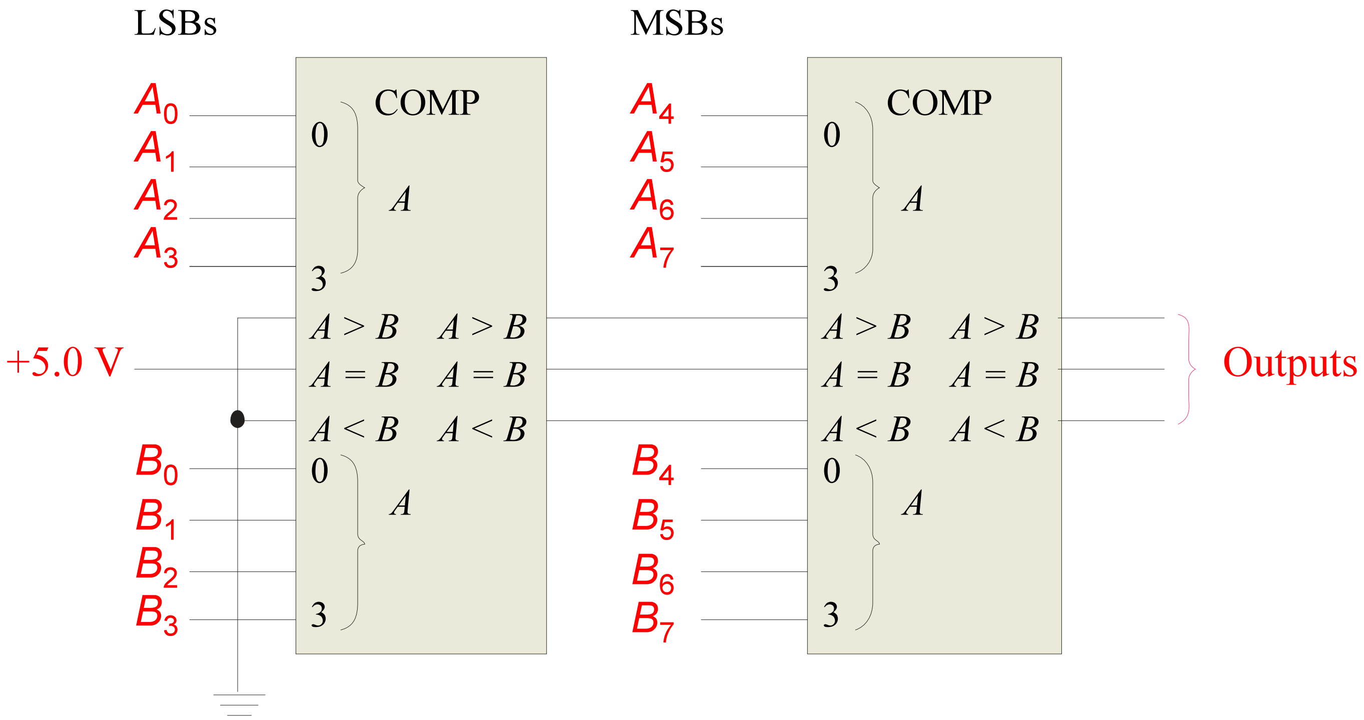


The IC shown is the 4-bit 74LS85.

Summary

Comparators

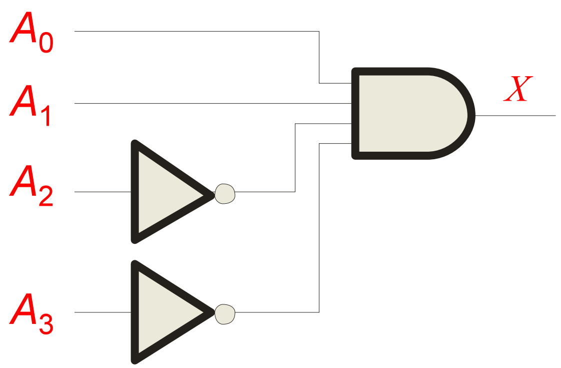
IC comparators can be expanded using the cascading inputs as shown. The lowest order comparator has a HIGH on the *A* = *B* input.



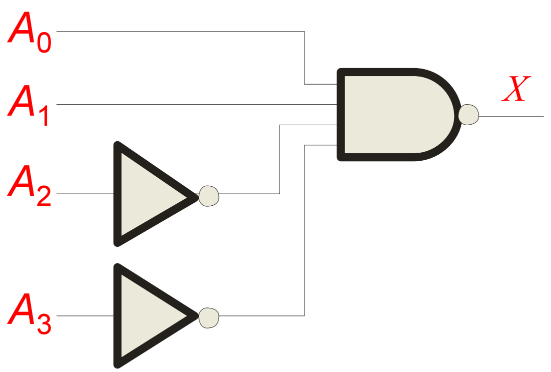
Summary

Decoders

A **decoder** is a logic circuit that detects the presence of a specific combination of bits at its input. Two simple decoders that detect the presence of the binary code 0011 are shown. The first has an active HIGH output; the second has an active LOW output.



Active HIGH decoder for 0011

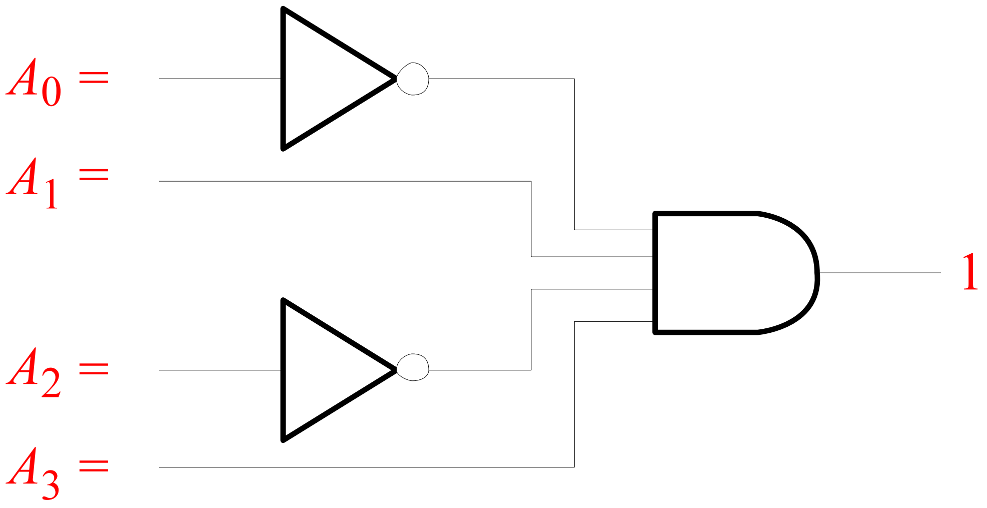


Active LOW decoder for 0011

Summary

Decoders

Question Assume the output of the decoder shown is a logic 1. What are the inputs to the decoder?



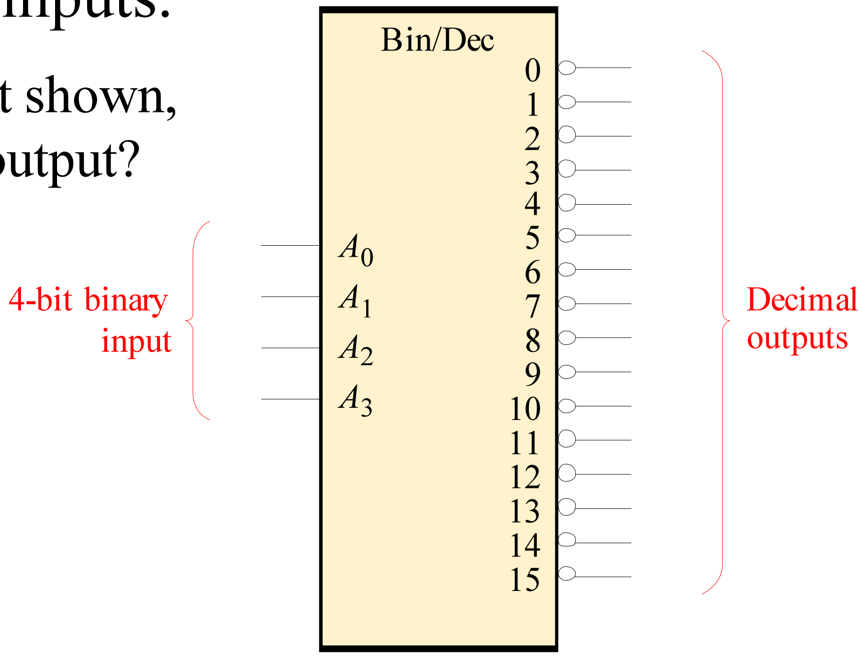
Summary

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.

Question

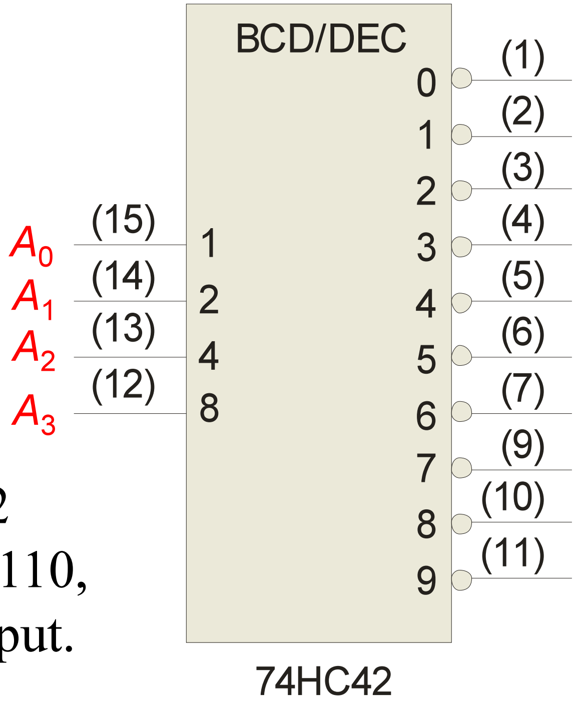
For the input shown, what is the output?



Summary

Decoders

BCD-to-decimal decoders accept a binary coded decimal input and activate one of ten possible decimal digit indications.



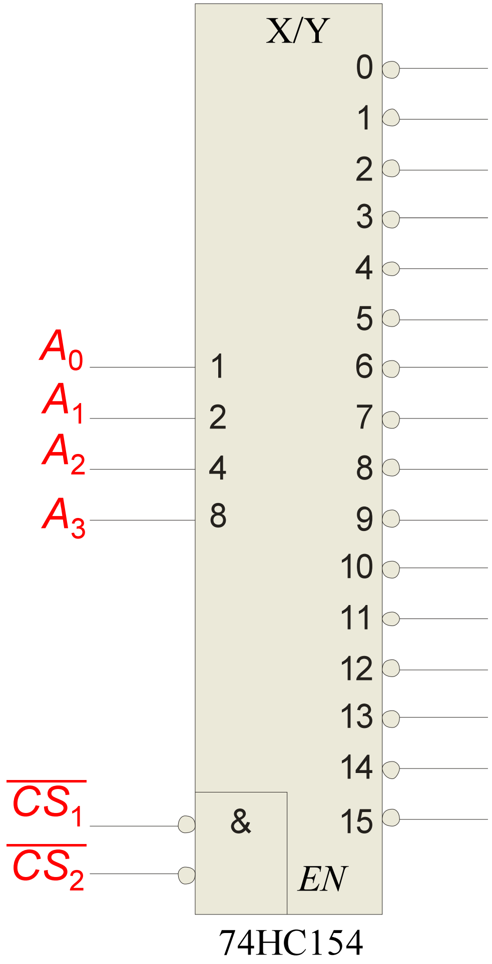
Example Assume the inputs to the 74HC42 decoder are the sequence 0101, 0110, 0011, and 0010. Describe the output.

Solution All lines are HIGH except for one active output, which is LOW. The active outputs are 5, 6, 3, and 2 in that order.

Summary

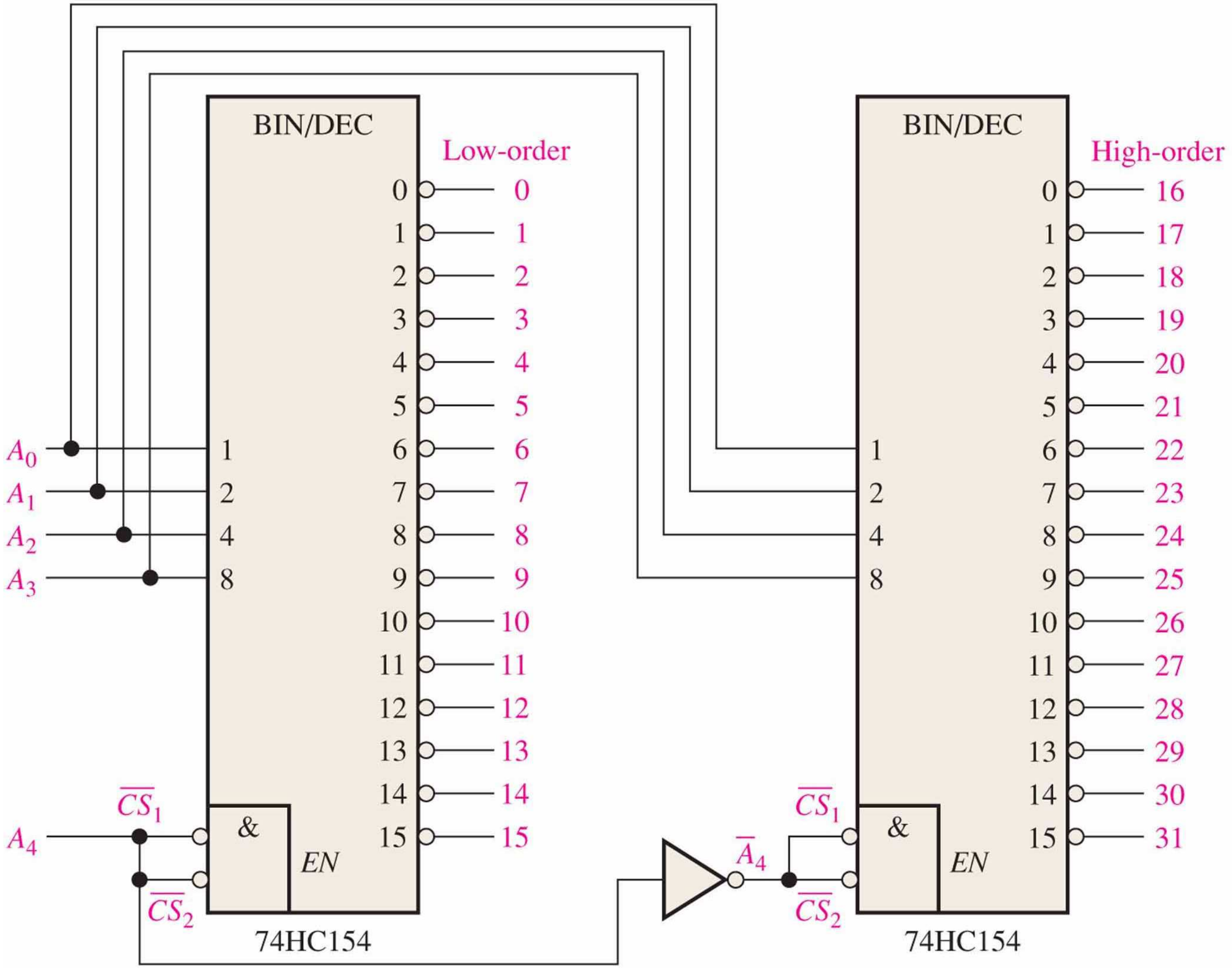
Decoders

A specific integrated circuit decoder is the 74HC154 (shown as a 4-to-16 decoder). It includes two active LOW chip select lines which must be at the active level to enable the outputs. These lines can be used to expand the decoder to larger inputs.



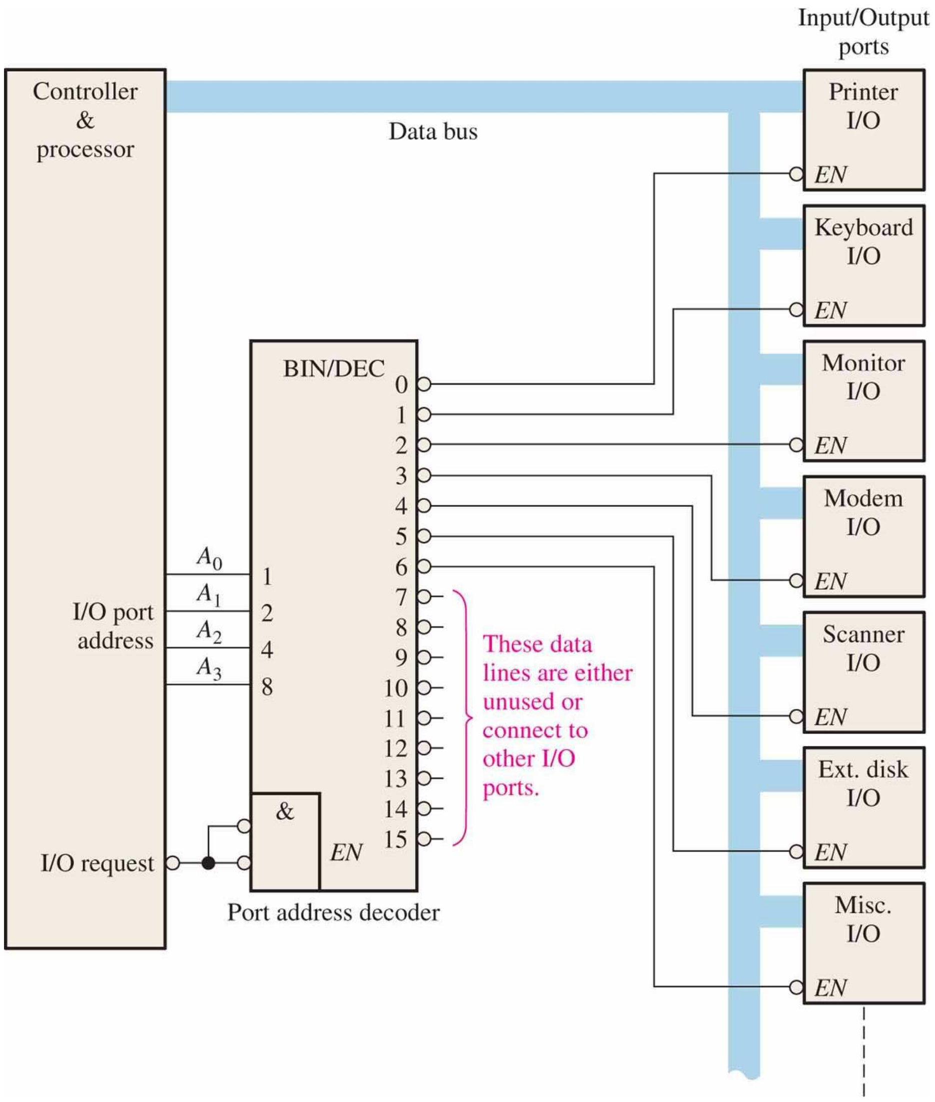
Summary

Decoders



Summary

Decoders

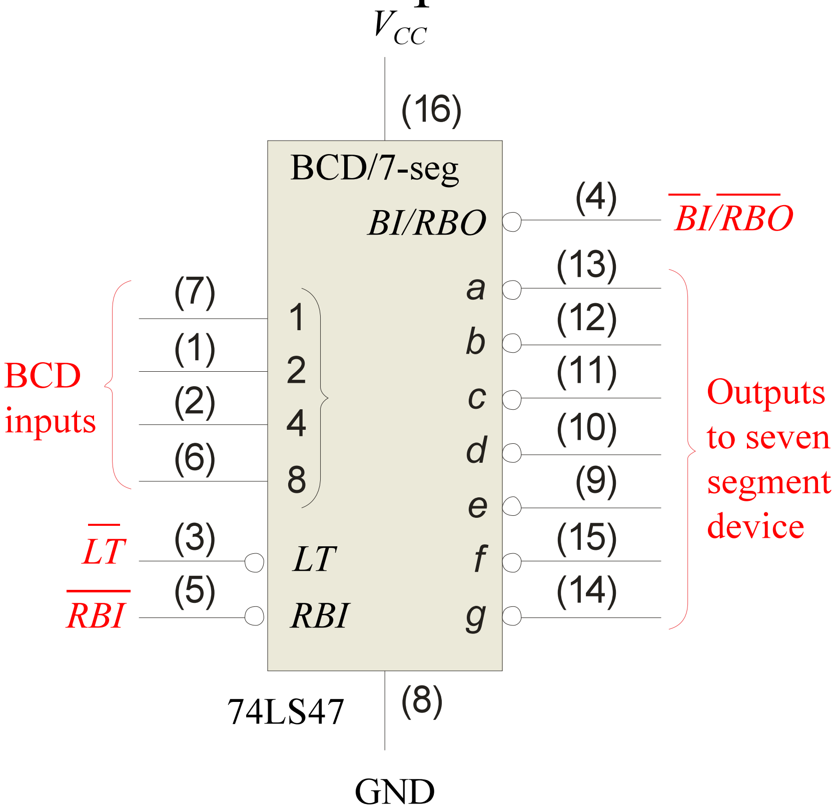


Summary

BCD Decoder/Driver

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

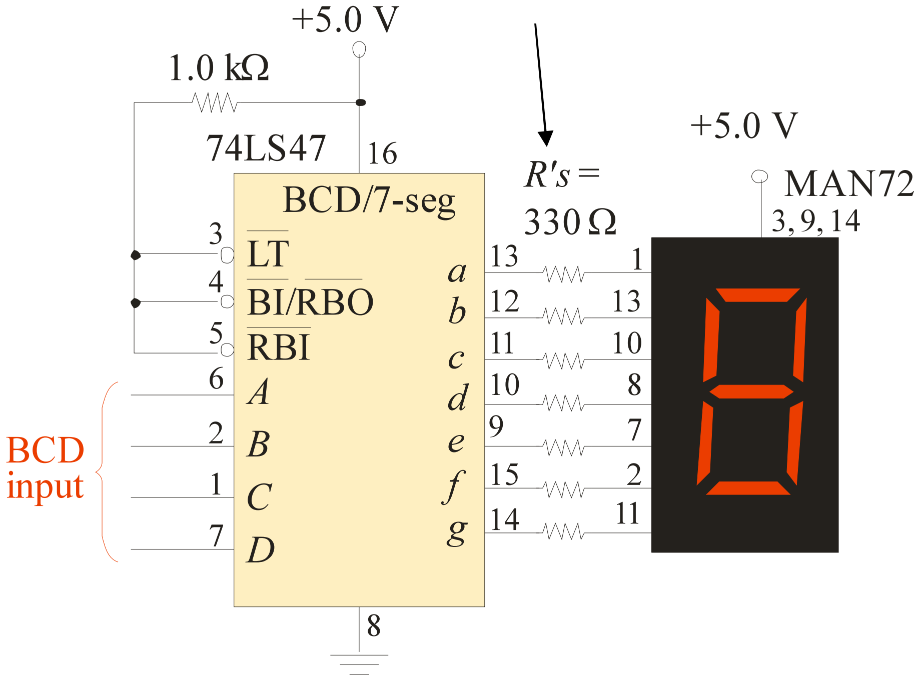
The *a*-*g* outputs are designed for much higher current than most devices (hence the word driver in the name).



Summary

BCD Decoder/Driver

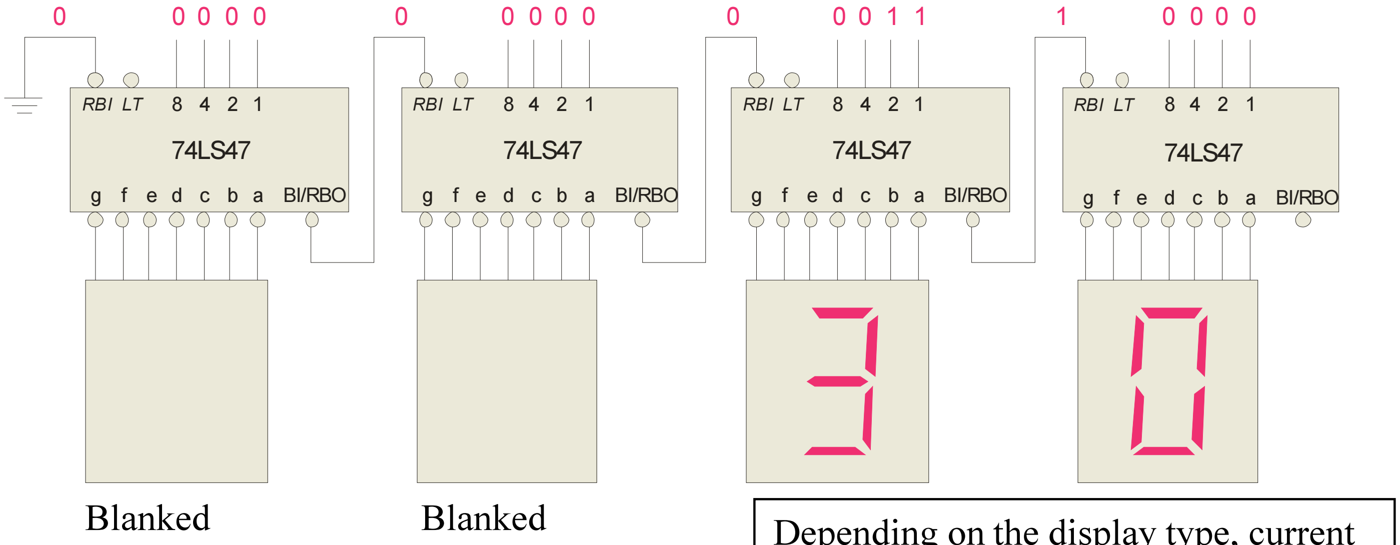
Here the 7447A is an connected to an LED seven segment display. Notice the current limiting resistors, required to prevent overdriving the LED display.



Summary

BCD Decoder/Driver

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.

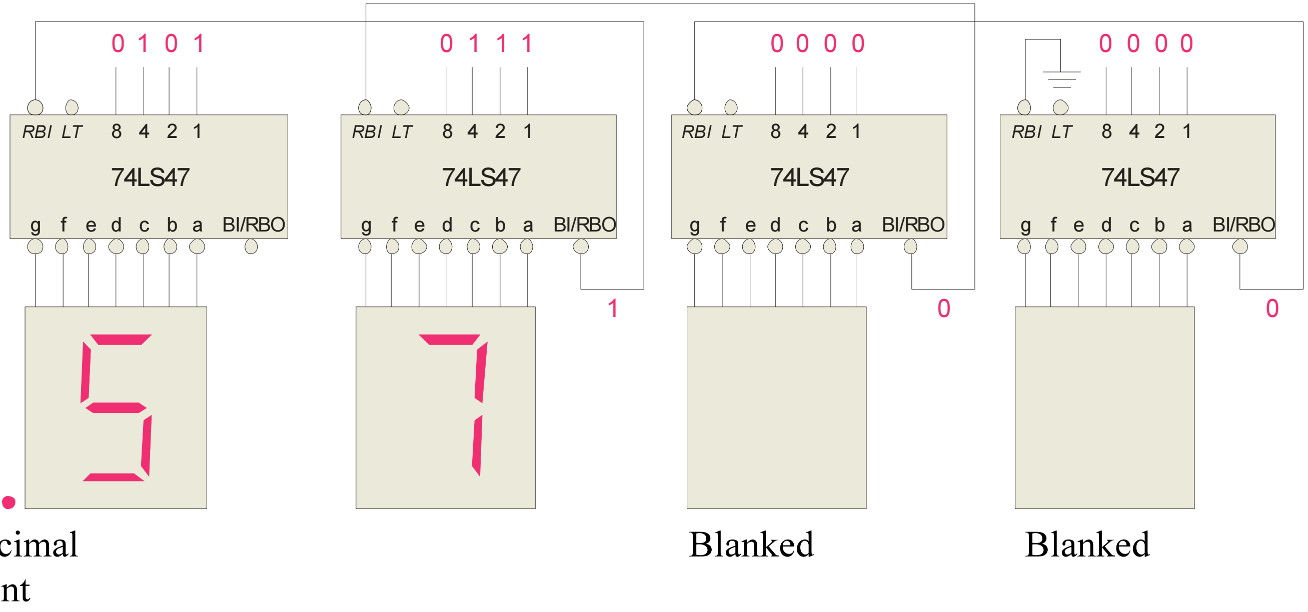


Depending on the display type, current limiting resistors may be required.

Summary

BCD Decoder/Driver

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

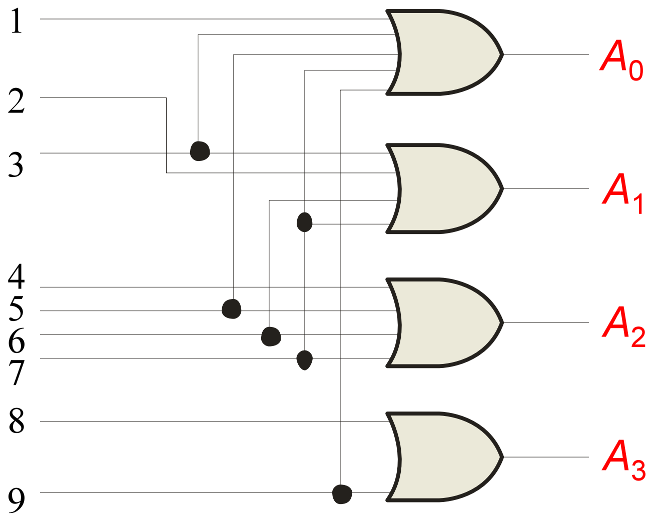


Summary

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.

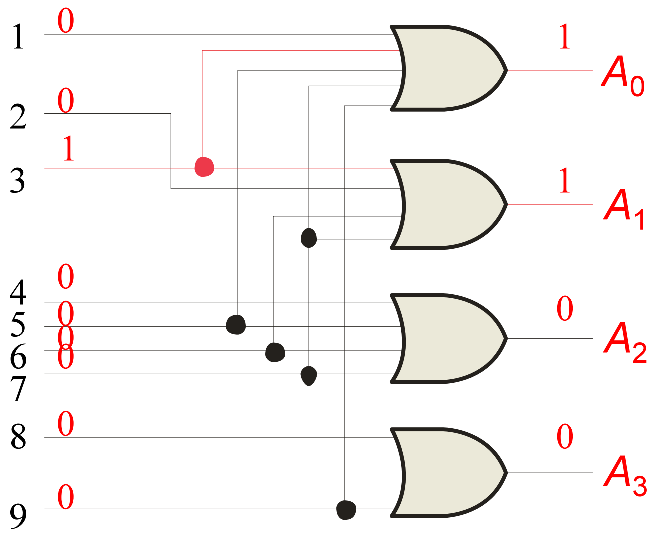


Summary

Encoders

Example Show how the decimal-to-BCD encoder converts the decimal number 3 into a BCD 0011.

Solution The top two OR gates have ones as indicated with the red lines. Thus the output is 0111.

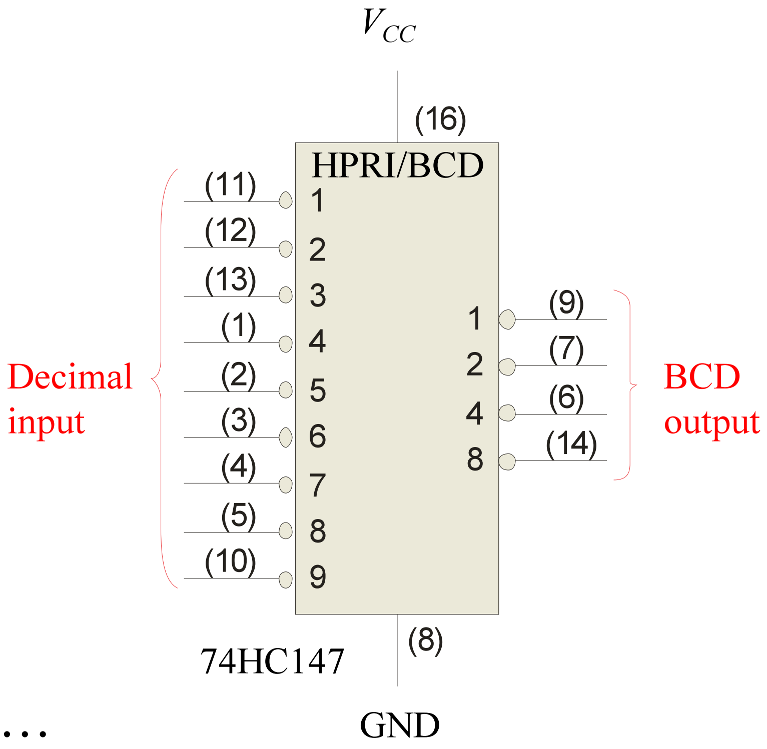


Summary

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 offers additional flexibility in that it is a **priority encoder**. This means that if more than one input is active, the one with the highest order decimal digit will be active.

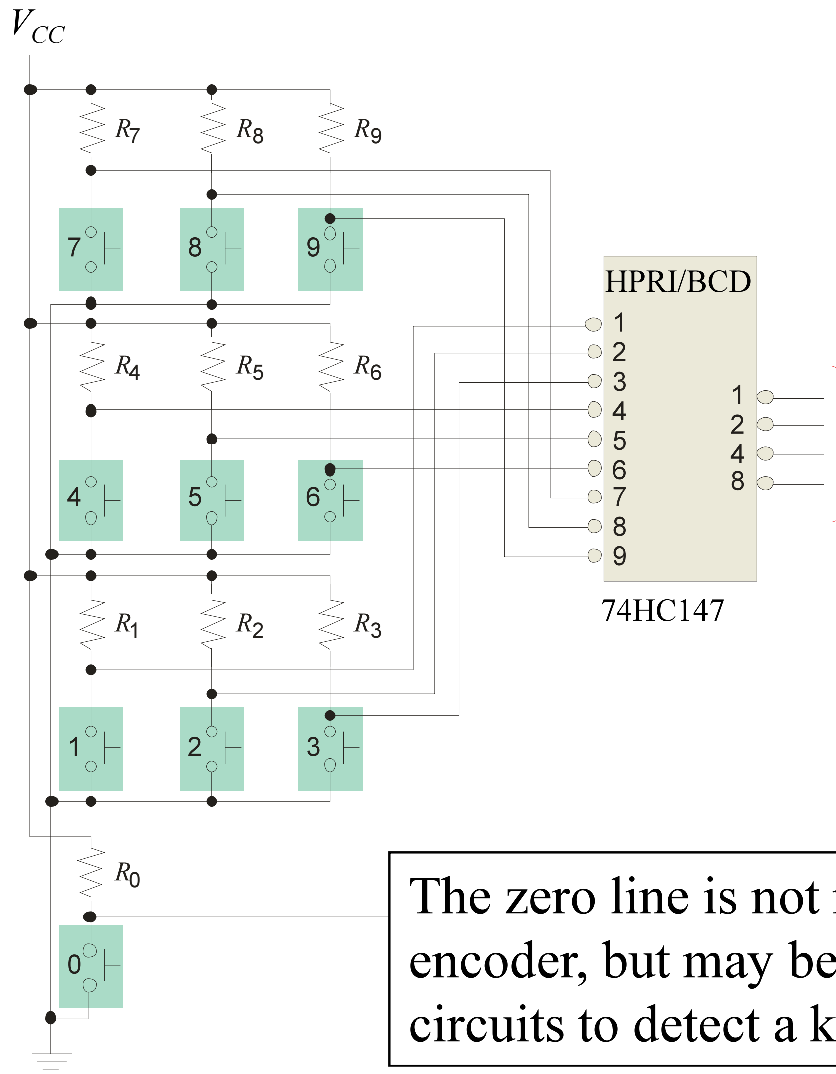


The next slide shows an application …

Summary

Encoders

Keyboard encoder



BCD complement of key press

The zero line is not needed by the encoder, but may be used by other circuits to detect a key press.

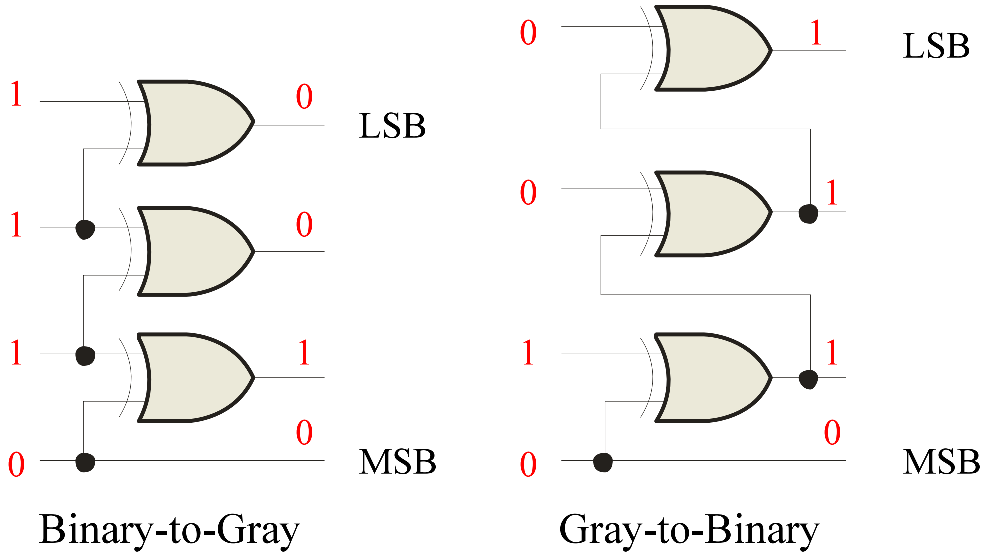
Summary

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.

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

Solution

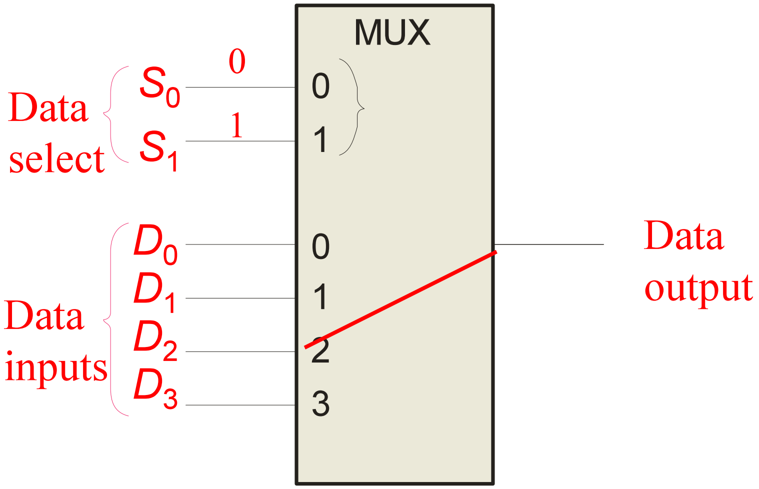


Summary

Multiplexers

A multiplexer (MUX) selects one data line from two or more input lines and routes data from the selected line to the output. The particular data line that is selected is determined by the select inputs.

Two select lines are shown here to choose any of the four data inputs.



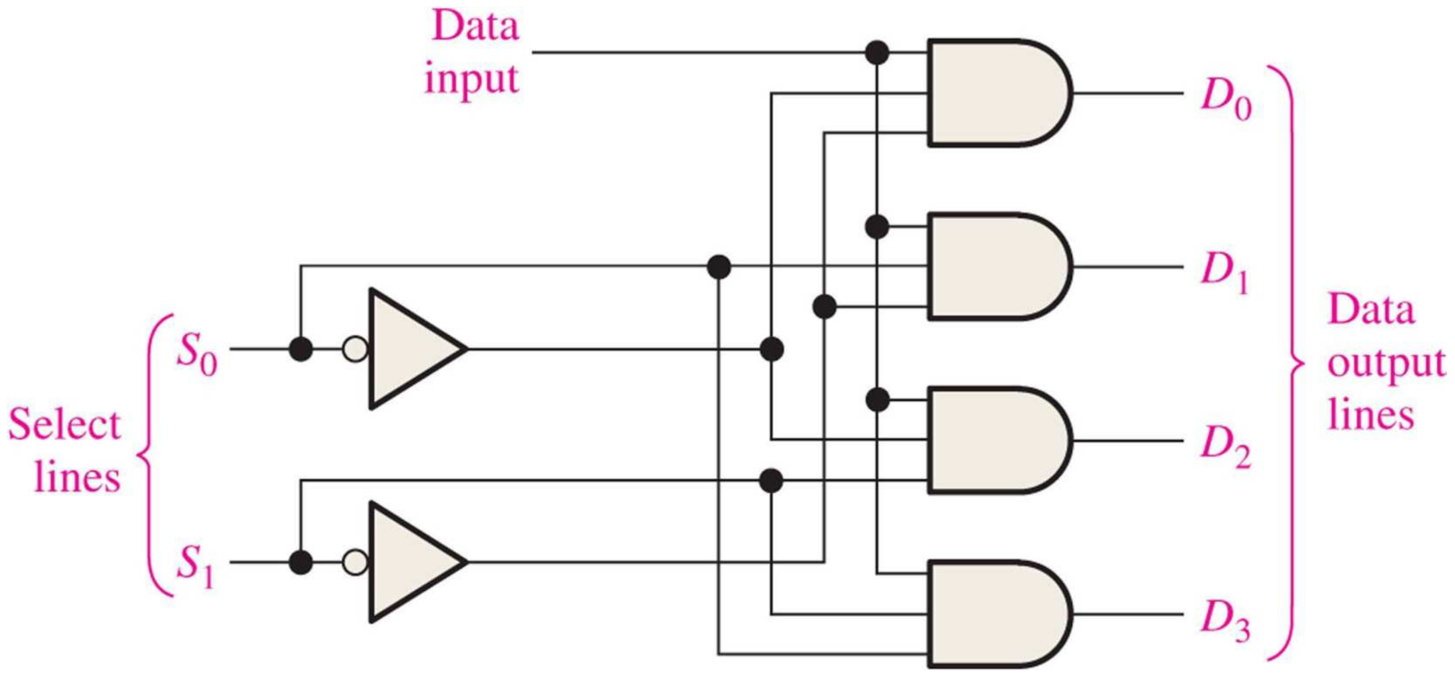
Question

Which data line is selected if $S_1S_0 = 10$? D_2

Summary

Demultiplexers

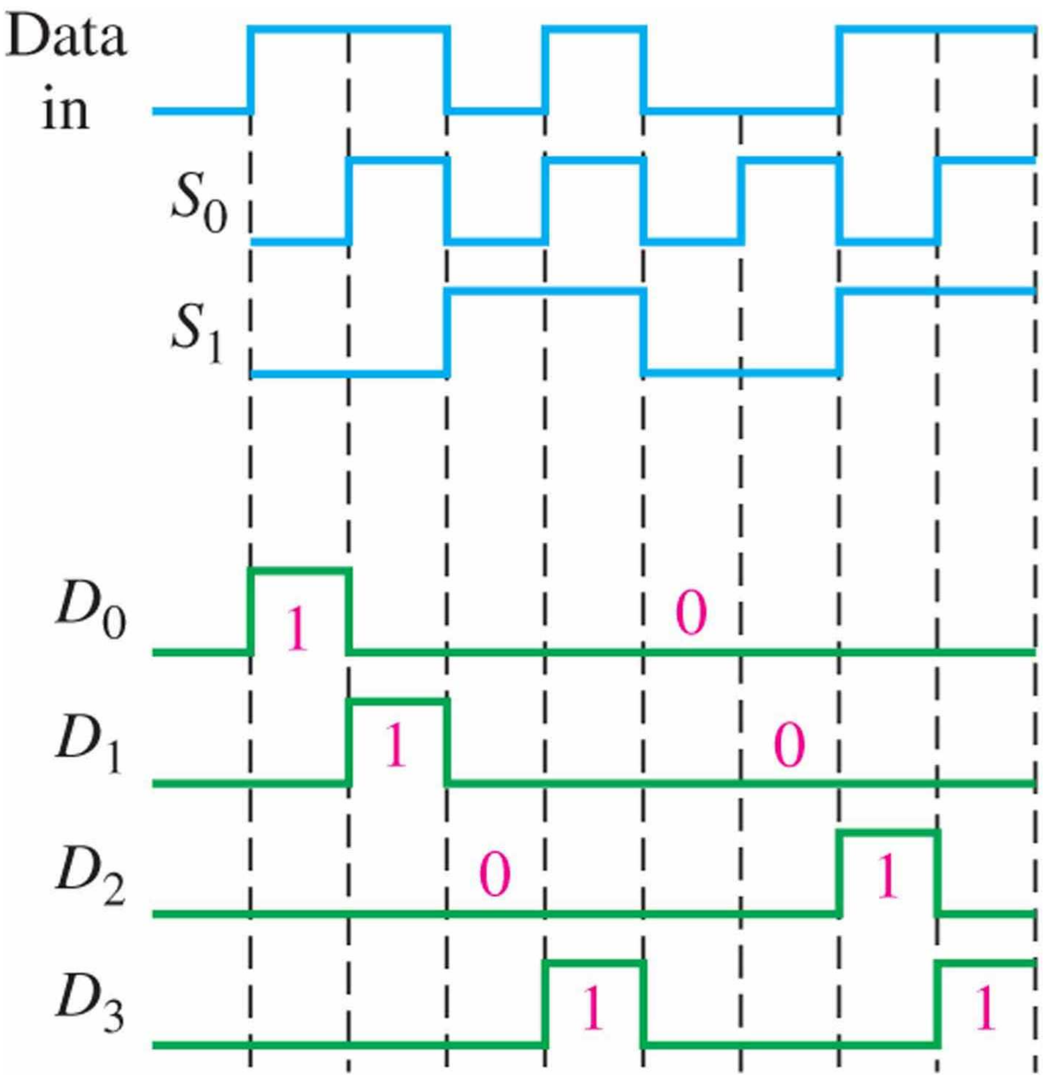
A demultiplexer (DEMUX) performs the opposite function from a MUX. It switches data from one input line to two or more data lines depending on the select inputs.



Summary

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.

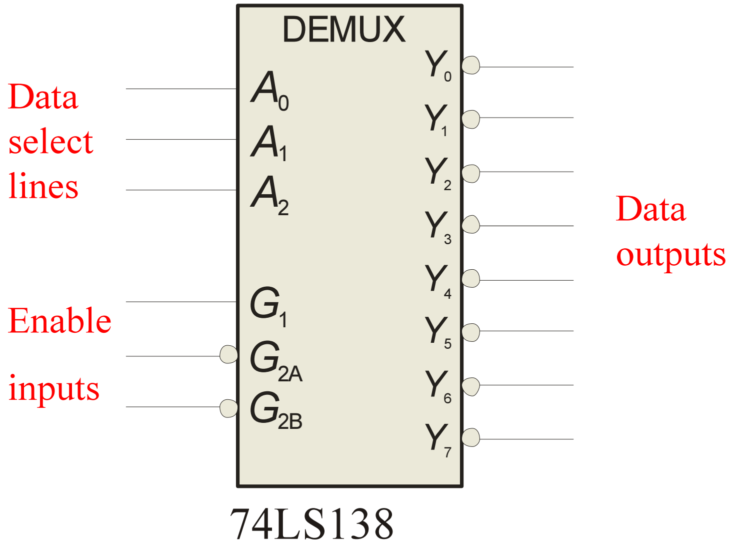


Summary

Demultiplexers

A decoder performs the similar function as a demultiplexer (DEMUX).

The 74LS138 was introduced previously as a decoder but can also serve as a DEMUX. When connected as a DEMUX, data is applied to one of the enable inputs, and routed to the selected output line depending on the select variables. Note that the outputs are active-LOW as illustrated in the following example…



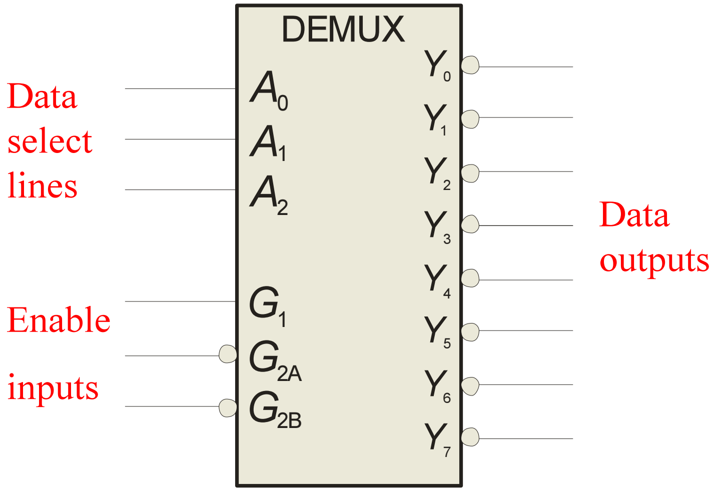
Demultiplexers

Example

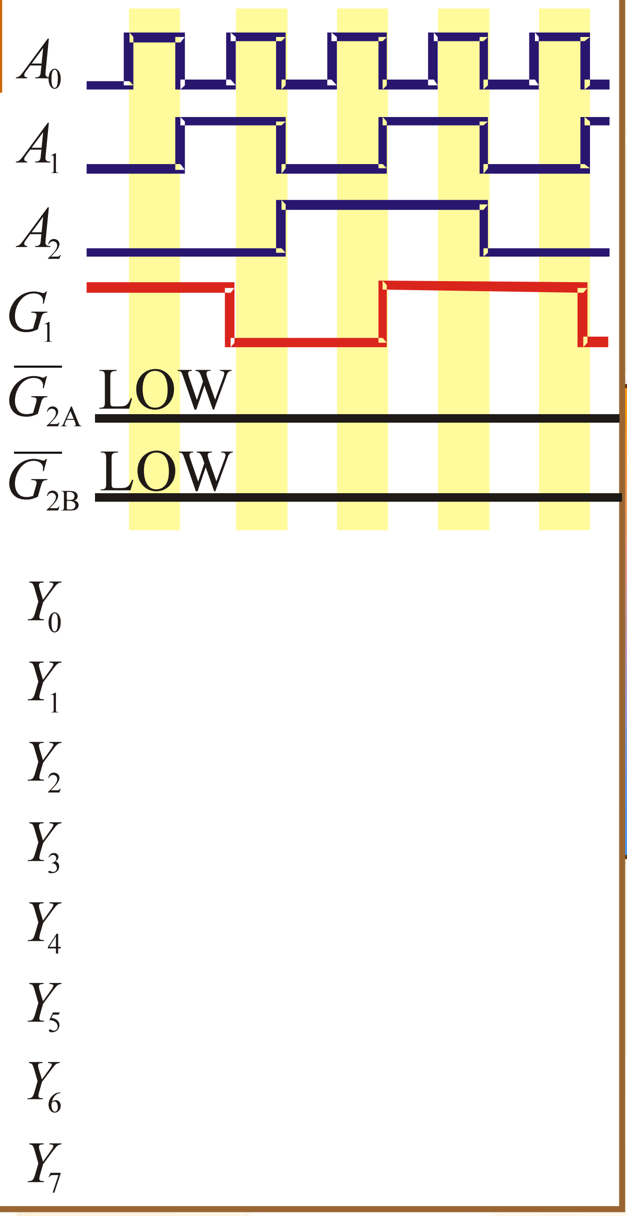
Determine the outputs, given the inputs shown.

Solution

The output logic is opposite to the input because of the active-LOW convention. (Red shows the selected line).

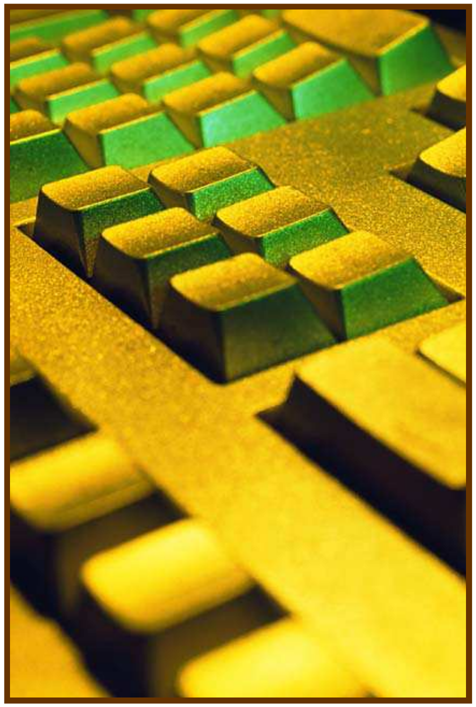


74LS138



Summary

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.

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

Solution S with odd parity = 11010011
S with even parity = 01010011

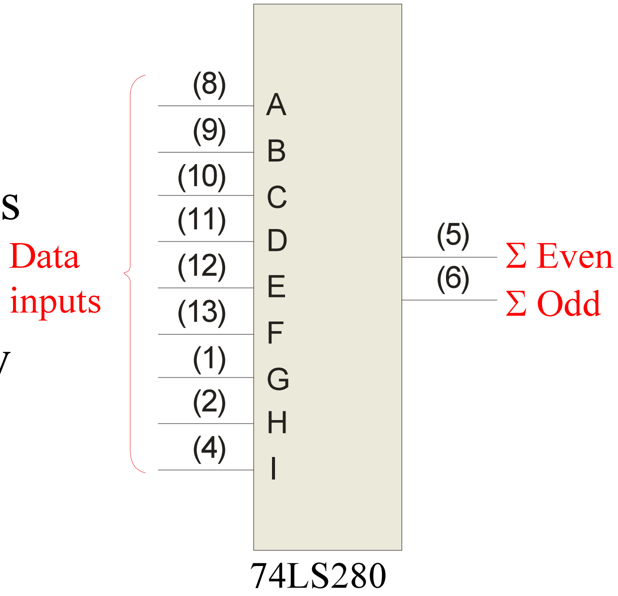
Summary

Parity Generators/Checkers

The 74LS280 can be used to generate a parity bit or to check an incoming data stream for even or odd parity.

Checker: The 74LS280 can test codes with up to 9 bits. 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.



Homework 9

- Chapter 6 (2, 7, 14, 22, 29, 32)