Digital Electronic Circuits
Section 1 (EE, IE)

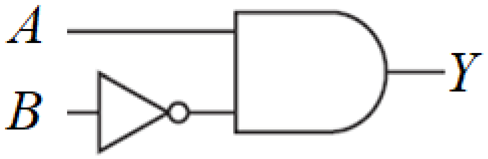
Lecture 14[#]

[#]Class 13 was on discussion related to Term Paper.
It has no distributable Lecture 13 slides but only email instruction.

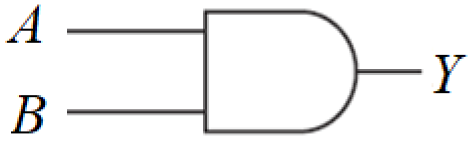
Decoder

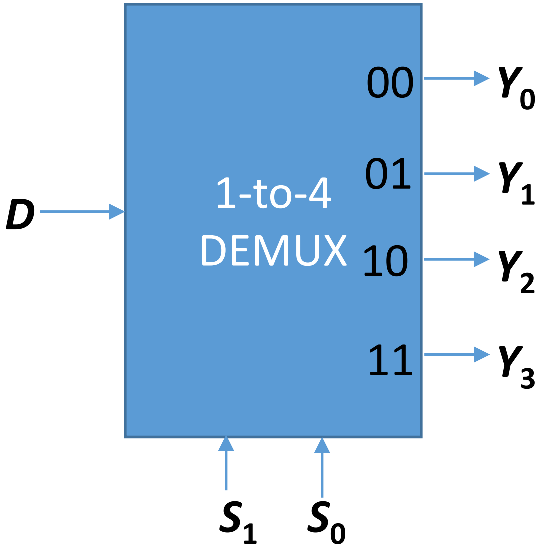
A decoder decodes input bit pattern by appropriate logic and activates the output when specific combination is present.

To decode AB = 01
(active HIGH)



To decode AB = 11



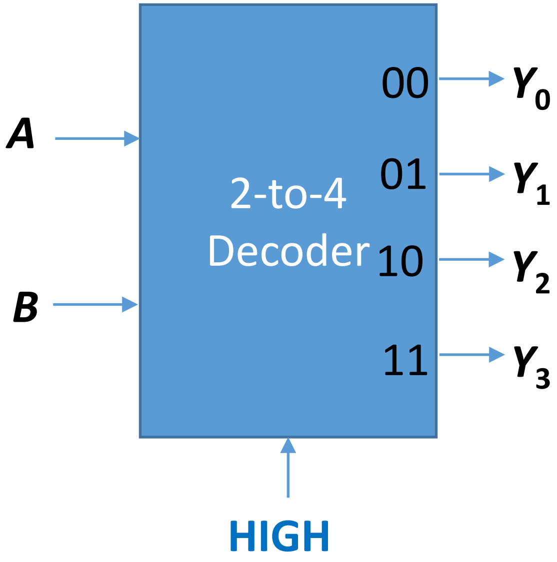


$Y_0 = S_1'S_0'.D$

$Y_1 = S_1'S_0.D$

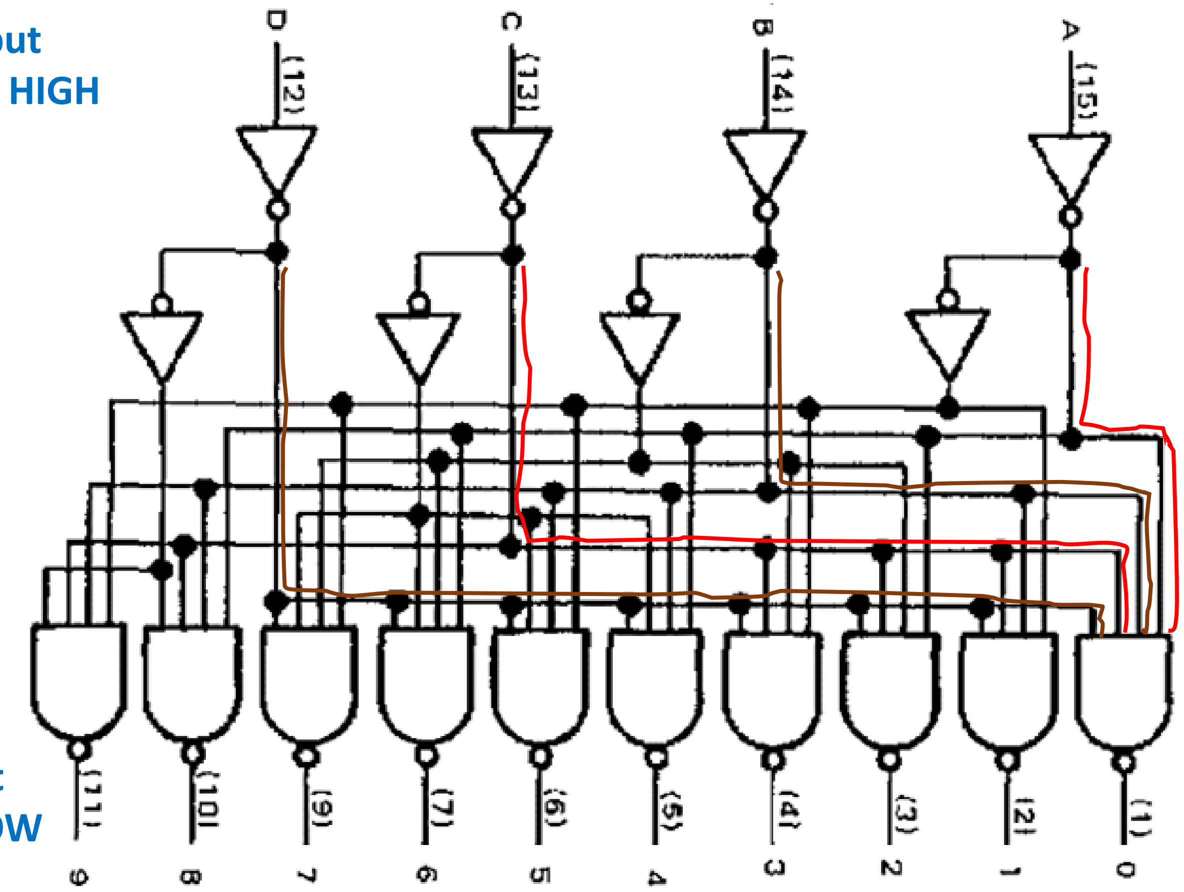
$Y_2 = S_1S_0'.D$

$Y_3 = S_1S_0.D$



BCD-to-Decimal Decoder

Input
Active HIGH



Output
Active LOW

IC 7445: BCD-to-Decimal Decoder / Driver

Open collector output

SN7445 can sink up to 80 mA

(SN7404, standard inverter sinks 16 mA max.)

***DCBA*: LLLL, 0: L, 1-9: H**

DCBA: LLLH, 0: H, 1: L, 2-9: H

DCBA: LLHL, 0-1: H, 2: L, 3-9: H

…

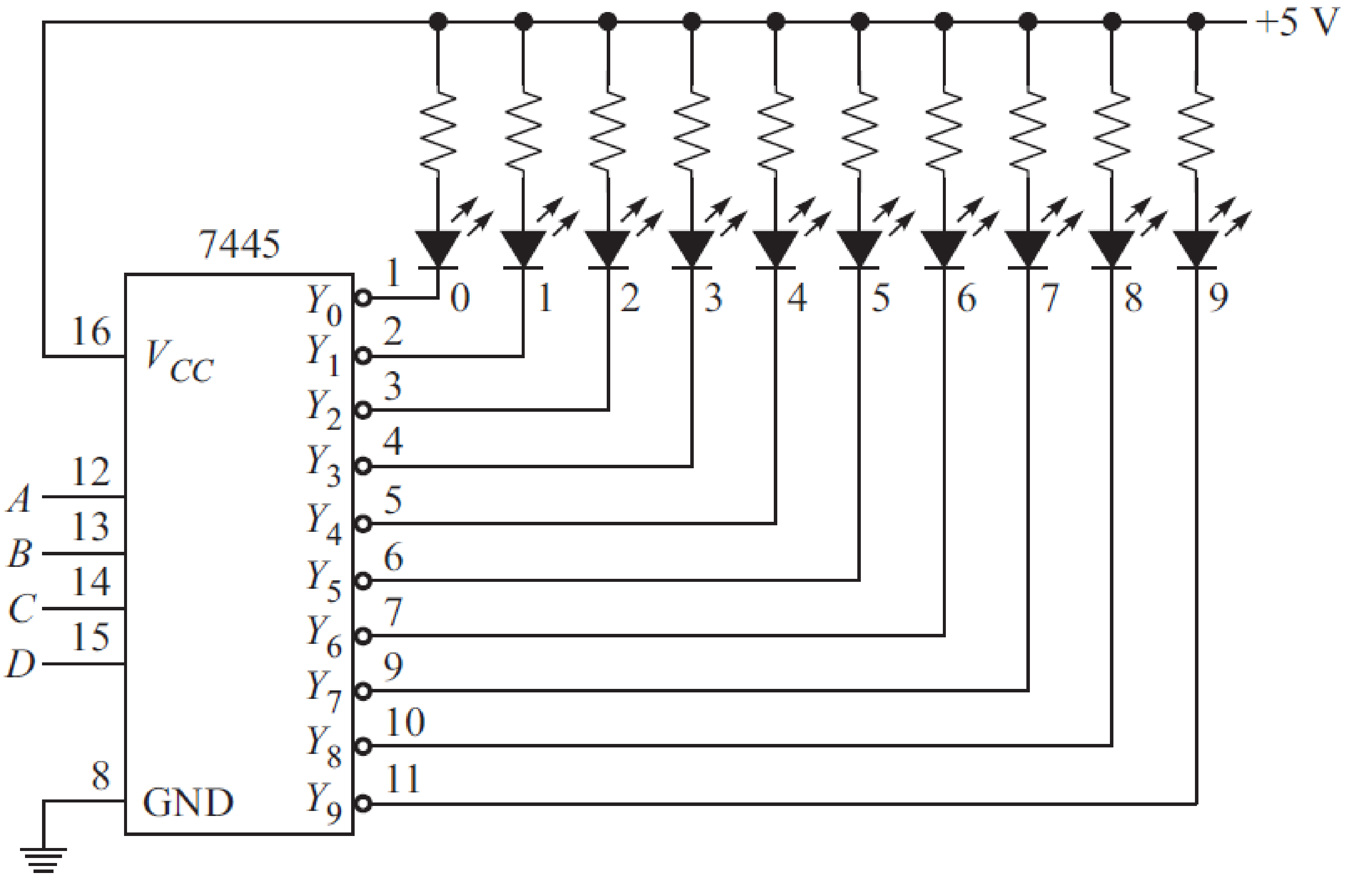
DCBA: HLLH, 0-8: H, 9: L

DCBA: HLHL, 0-9: H,

…

DCBA: HHHH, 0-9: H

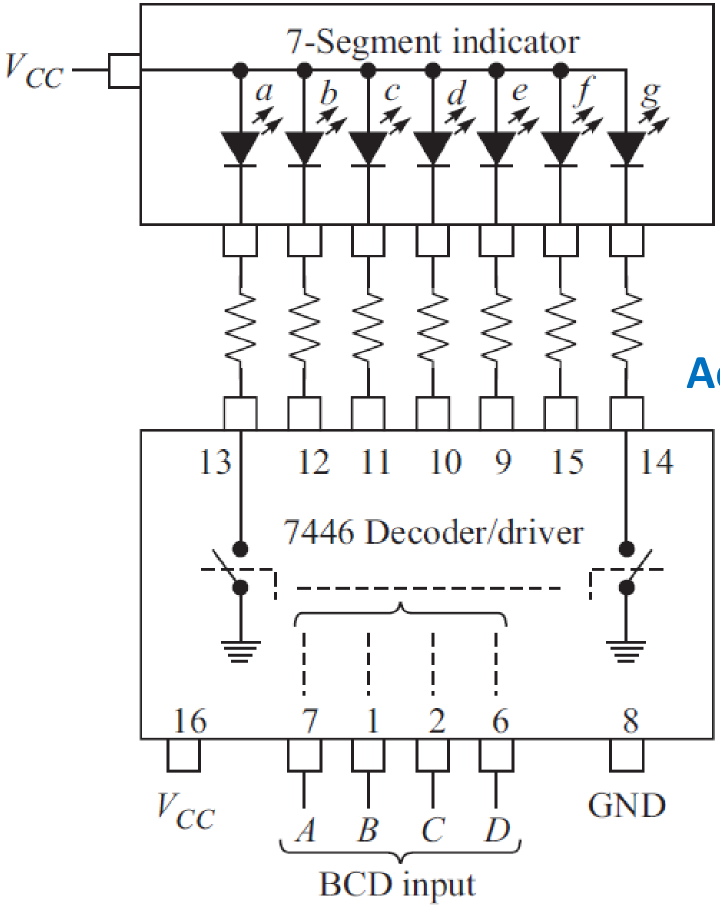
Decoder / Driver driving LED



If LED drop = 1.6 V
Resistance = 330 Ω
$V_{CE(Sat.)}$ = 0.1 V

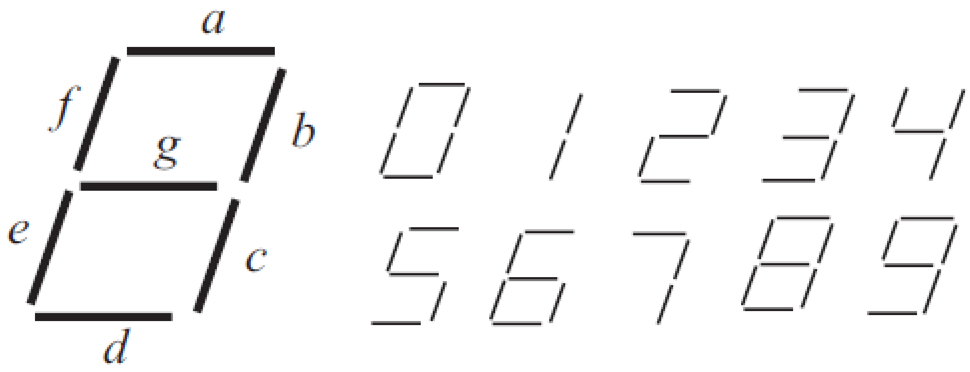
LED current
= (5 – 1.6 – 0.1) / 330
= 10 mA

BCD-to-7 Segment Decoder / Driver



Common Anode type

Output Active LOW



$a = F(A,B,C,D) = \sum m(0,2,3,5,6,7,8,9)$

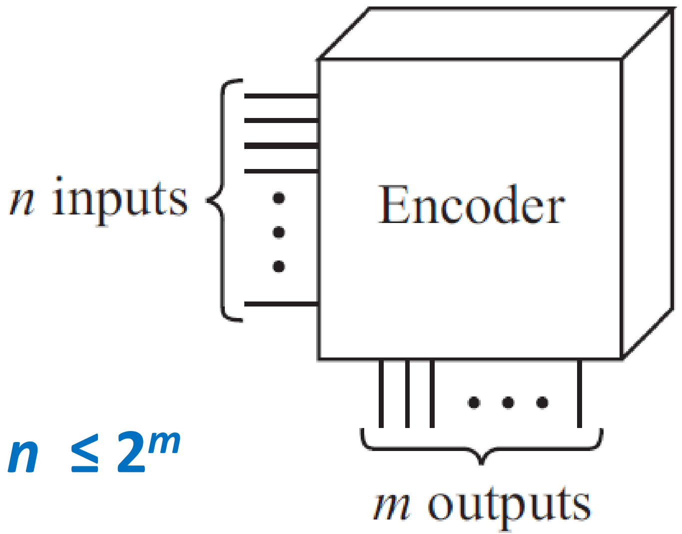
$b = F(A,B,C,D) = \sum m(0,1,2,3,4,7,8,9)$

...

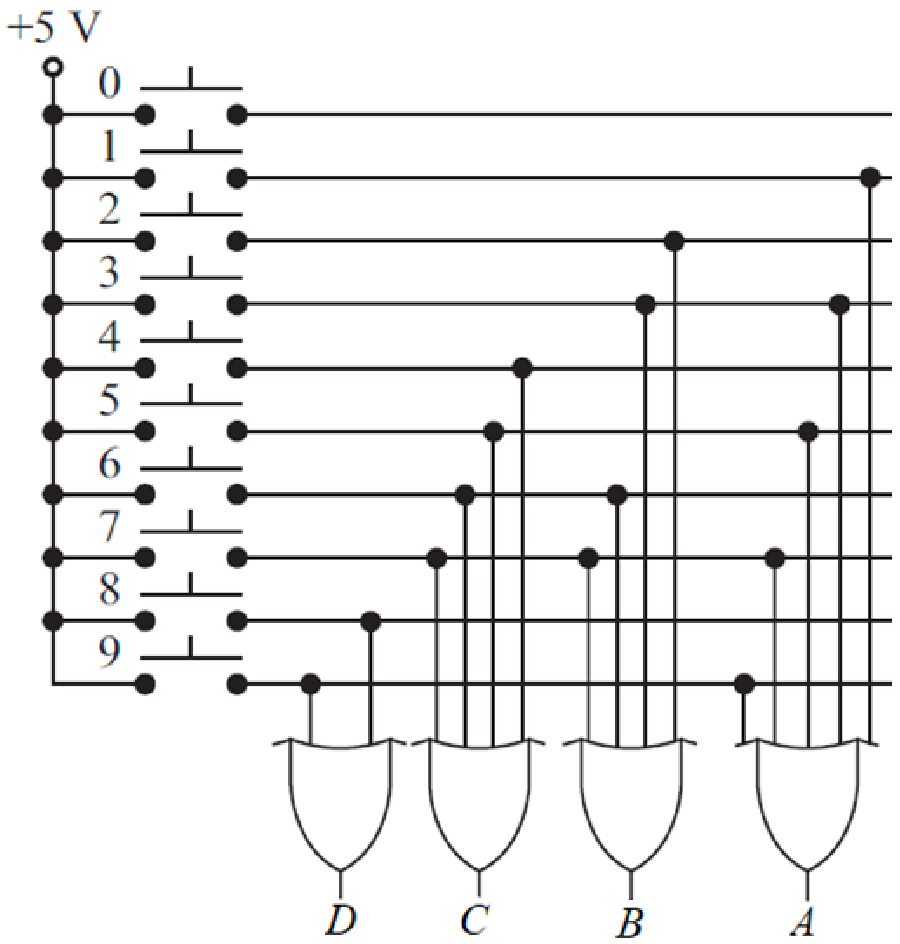
IC 7446 has other control inputs for additional functionality e.g. blanking of leading zero, lamp test etc.

Encoder

An encoder converts an active input signal to a coded output signal.



$n \leq 2^m$



A decimal-to-BCD encoder

Only one of the 0 to 9 input is to remain connected

0: *DCBA* = 0000
1: *DCBA* = 0001
2: *DCBA* = 0010
…
1 & 2: *DCBA* = 0011
1 & 3: *DCBA* = 0011

Concept of Priority Encoder

D_3	D_2	D_1	D_0	C_1	C_0
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

$C_1 = D_3 + D_2$

$C_0 = D_3 + D_2'.D_1$

(D_1 if not D_2, due to priority)

C_1

D_3D_2 \ D_1D_0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

D_2

D_3

C_0

D_3D_2 \ D_1D_0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

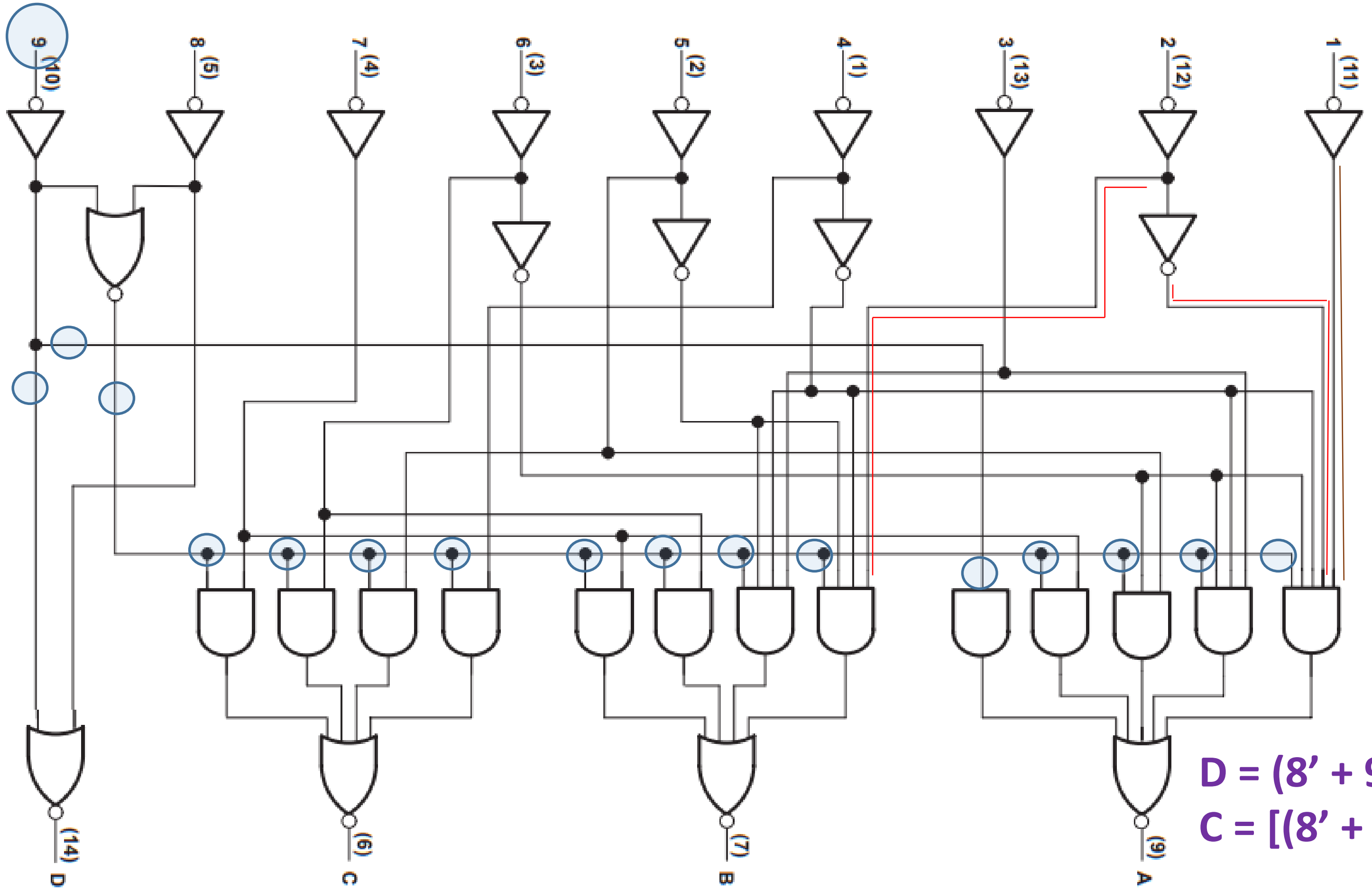
$D_2'.D_1$

D_3

Also, from Karnaugh Map

In a different context, in $Y = D_1 + D_1'.D_0$ which of D_1, D_0 is of higher priority?

Priority Encoder: Decimal-to-BCD



IC 74147: Decimal-to-BCD Priority Encoder

Higher number has higher priority.

Input and Output: Active Low

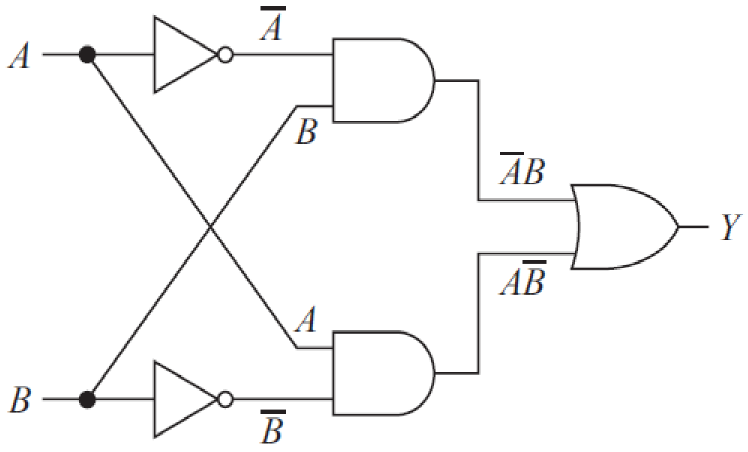
9: L, DCBA: LHHL
9: H, 8: L, DCBA: LHHH
9-8: H, 7: L, DCBA: HLLL
…
9-2: H, 1: L, DCBA: HHHL
9-1:H, DCBA: HHHH

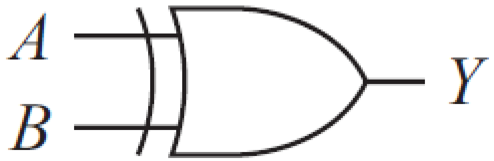
D = (8’ + 9’)’
C = [(8’ + 9’)’.(7’+6’+5’+4’)]’

Part 2

Exclusive-OR Gate

A	B	Y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

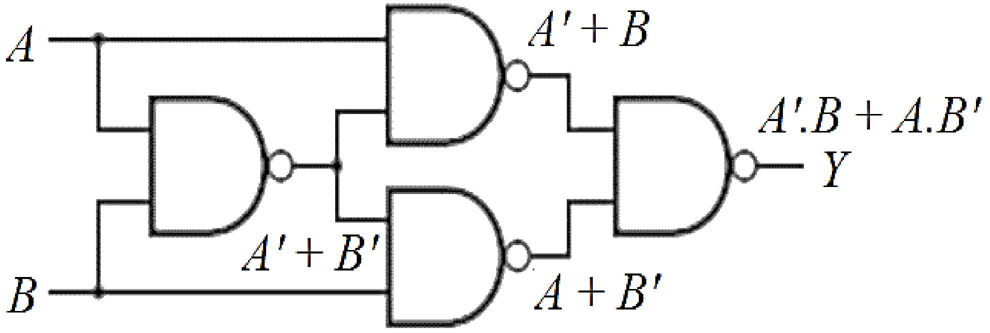




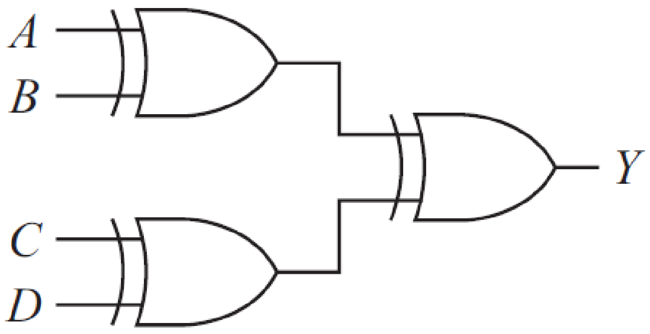
A⊕0 = A

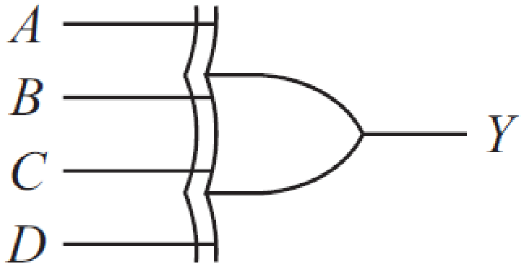
A⊕1 = A'

Y = A'.B + A.B' = A⊕B



Multi-input Exclusive-OR





$Y = A \oplus B \oplus C \oplus D$

A	B	C	D	Y
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**0**
0	1	1	1	**1**

If $A = 0$, $Y = B \oplus C \oplus D$

If $A = 1$, $Y = (B \oplus C \oplus D)'$

A	B	C	D	Y
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**0**

$\mathbf{A \oplus 0 = A}$

$\mathbf{A \oplus 1 = A'}$

- If no. of 1s in input combination is odd, then the output is 1, else, output is 0.
- This happens for any number of inputs e.g. **2,3,4,**5,6,7, …

Even Parity and Odd Parity

- **Even Parity:** In the input bits, there is even number of 1s.

Example: For, 4-bit input: 1001, 1100, 1111
8-bit input: 11000101, 01111011
16-bit input: 1011001110110011

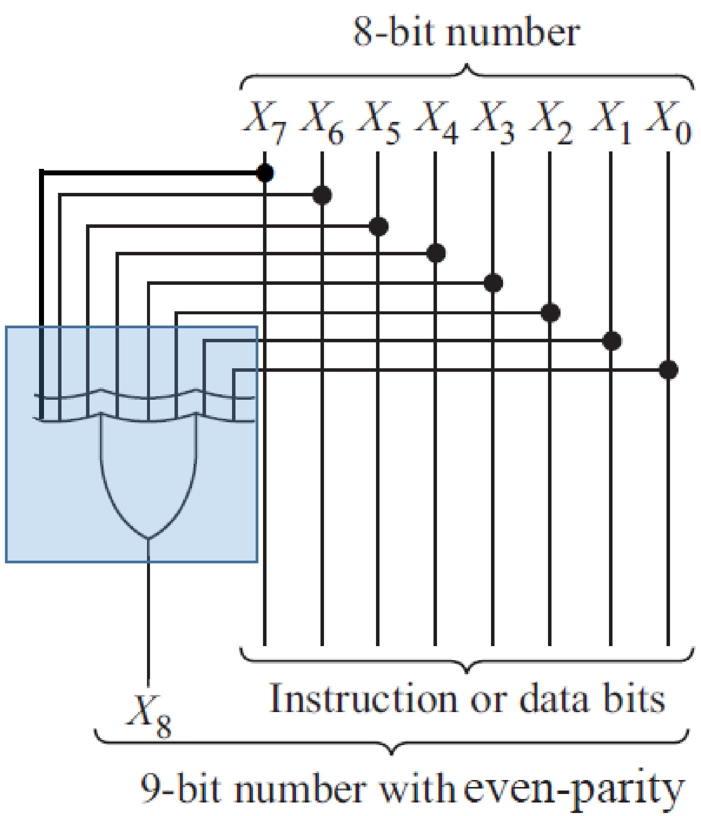
- **Odd Parity:** In the input bits, there is odd number of 1s.

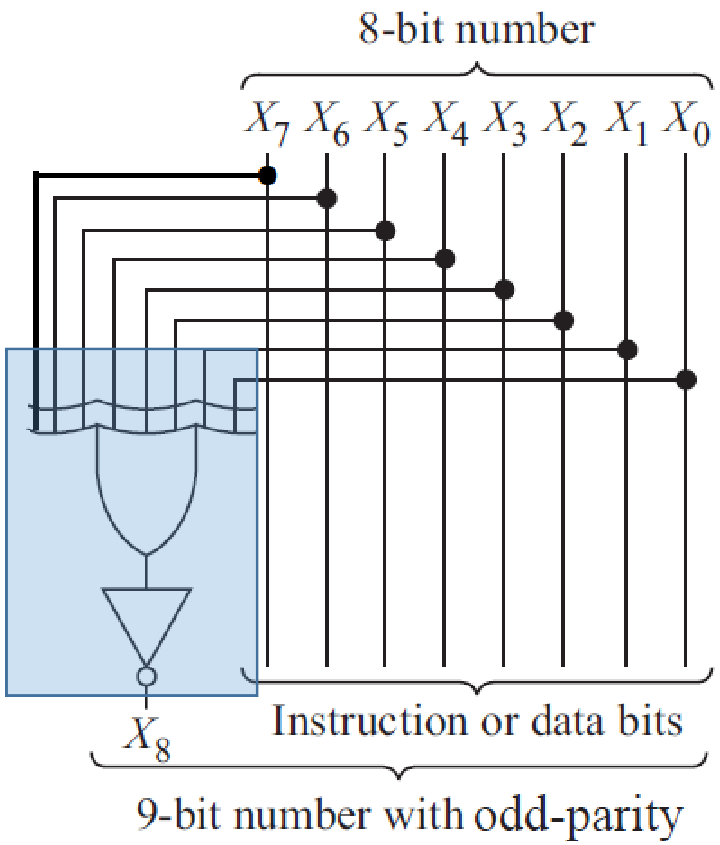
Example: For, 4-bit input: 0001, 1101, 1110
8-bit input: 11000001, 01011011
16-bit input: 1011001010110011

- Prior information about parity (even or odd) in the message can help in detecting **1-bit error.**

TI's DS90C031,DS90C032 line transmitter, receiver under test condition gives bit error rate (BER) less than 1 per 10^{12} bits.

Parity Generation



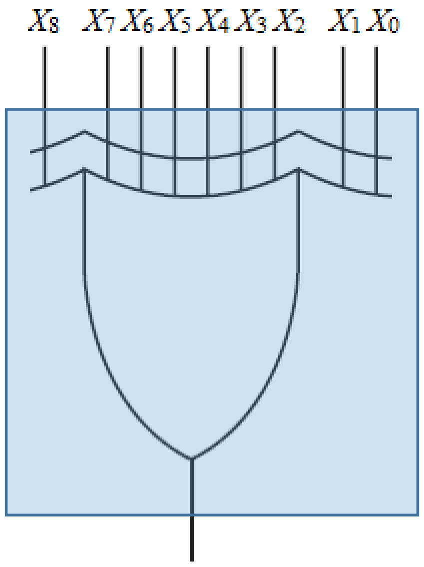


- **The aim here is to add a parity bit with the message so that message and parity bit together make it of a predefined parity (even or odd).**
- **The message is random i.e. can be of even or odd parity.**

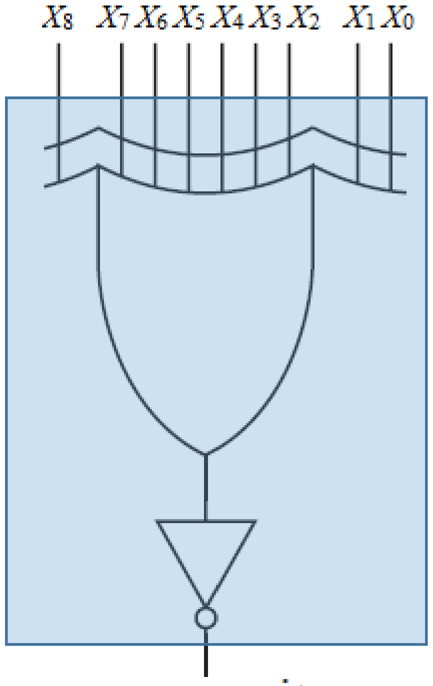
Example: Instead of 8 bits (message), 9 bits (message + parity bit) sent.

Parity Checking

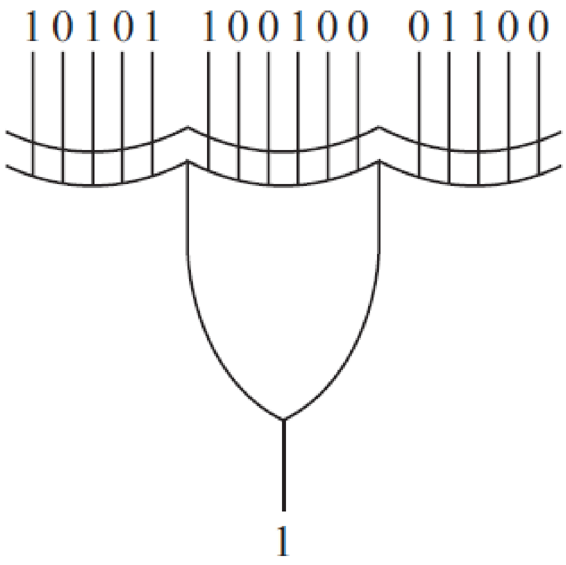
For 9-bit data received with pre-defined parity (even or odd).



odd-parity



even-parity



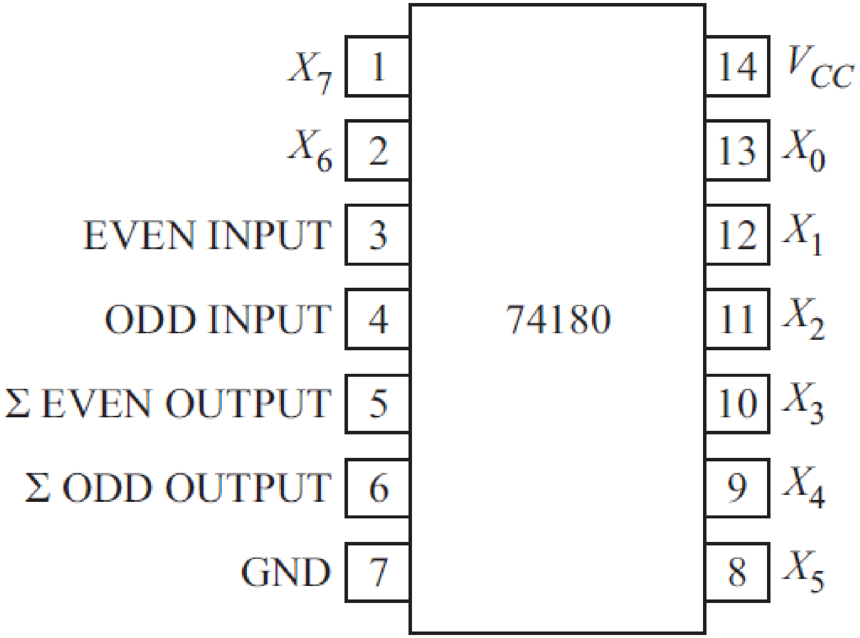
Example of 16-bit odd parity checking

Odd parity checking by Ex-OR, Odd parity generation by Ex-NOR (i.e. EX-OR then NOT)

Similarly, for Even Parity.

IC 74180

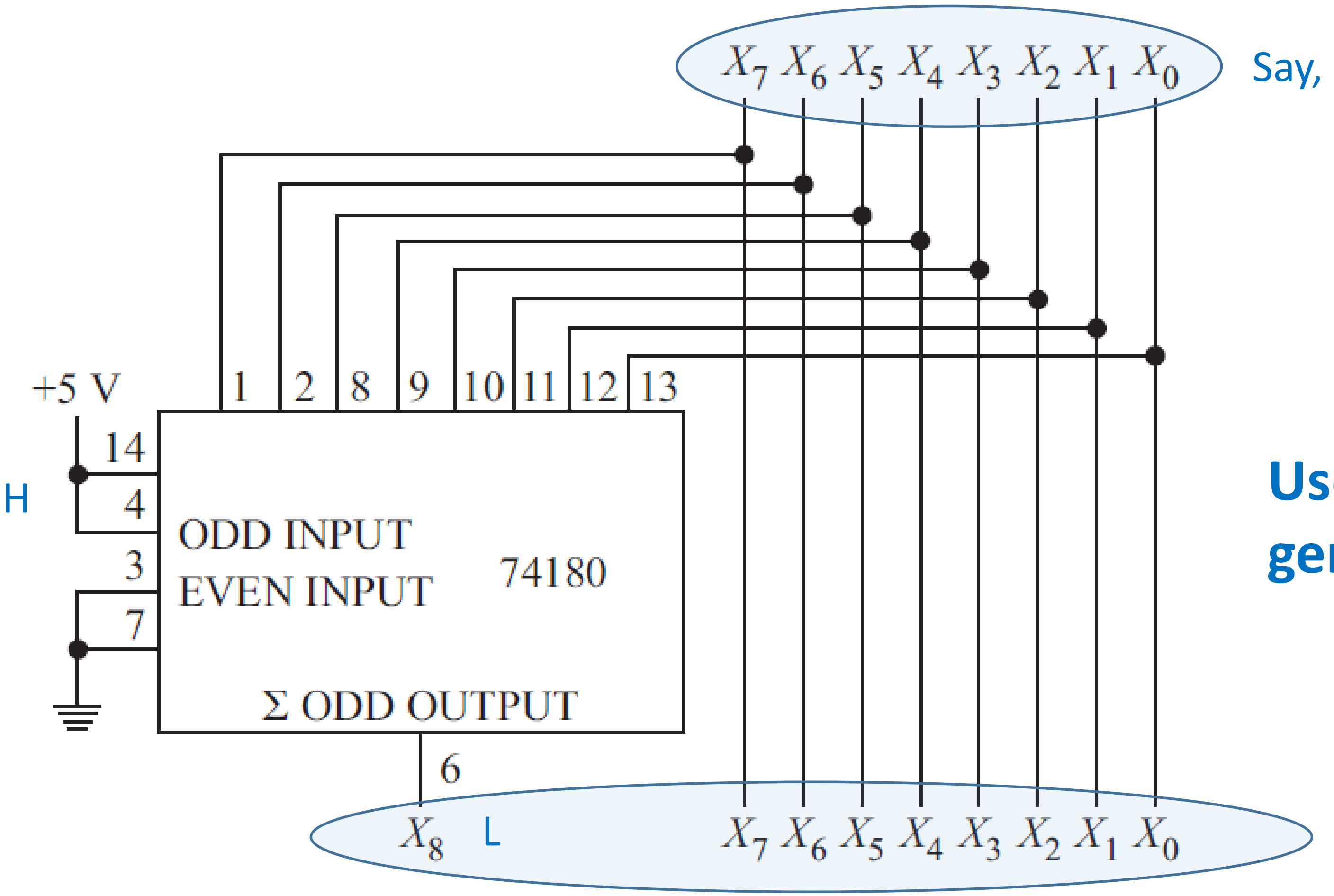
9-bit (8 data bits, 1 parity bit) parity generator / checker.



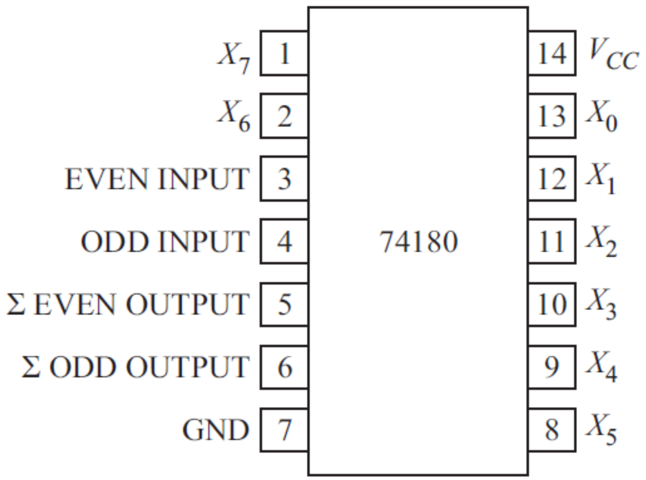
Inputs			Outputs	
Σ of H's at X_7 to X_0	*Even*	*Odd*	*Σ even*	*Σ odd*
Even	*H*	*L*	*H*	*L*
Odd	*H*	*L*	*L*	*H*
Even	*L*	*H*	*L*	*H*
Odd	*L*	*H*	*H*	*L*
X	*H*	*H*	*L*	*L*
X	*L*	*L*	*H*	*H*

- **For 9-bit parity checking even / odd input can be used as 9th bit.**
- **It can also be used for cascading to work on larger number of bits.**

Example with IC 74180



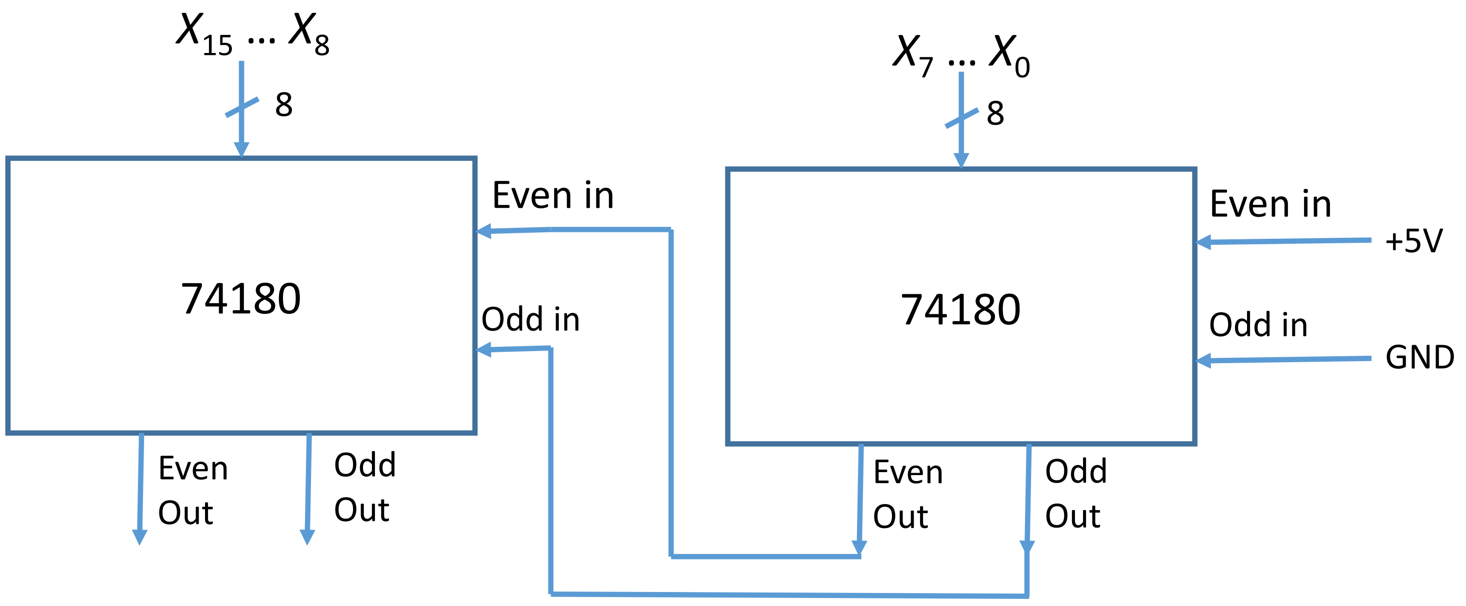
Say, odd no. of H



Use of IC 74180 to generate odd parity

Together, odd no. of H

Higher Order from Lower Order



16-bit parity checking by cascading two 74180

Lower order from Higher Order: Connect unused inputs to GND.

Part 3

Working of an odometer

Decimal system (we are used to)	Octal system	Binary system
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 0 0 0 1	**0 0 0 0 1**	**0 0 0 0 1**
0 0 0 0 2	0 0 0 0 2	**0 0 0 1 0**
0 0 0 0 3	0 0 0 0 3	0 0 0 1 1
0 0 0 0 4	0 0 0 0 4	0 0 1 0 0
0 0 0 0 5	0 0 0 0 5	0 0 1 0 1
0 0 0 0 6	0 0 0 0 6	0 0 1 1 0
0 0 0 0 7	0 0 0 0 7	0 0 1 1 1
0 0 0 0 8	**0 0 0 1 0**	0 1 0 0 0
0 0 0 0 9	0 0 0 1 1	0 1 0 0 1
0 0 0 1 0	0 0 0 1 2	0 1 0 1 0

Concept of place value

0	0	0	1	0

$\xrightarrow{\text{Decimal}}$ $1 \times 10^1 + 0 \times 10^0$

$\xrightarrow{\text{Octal}}$ $1 \times 8^1 + 0 \times 8^0$

$\xrightarrow{\text{Binary}}$ $1 \times 2^1 + 0 \times 2^0$

1010 in binary: $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

12 in octal: $1 \times 8^1 + 2 \times 8^0$

Base or radix – r

$d_i = \{0,1, \ldots, r - 1)$

$$d_n \ldots d_1 d_0 . d_{-1} d_{-2} \ldots d_{-m}$$

$$= d_n \times r^n + \ldots + d_1 \times r^1 + d_0 \times r^0$$
$$+ d_{-1} \times r^{-1} + d_{-2} \times r^{-2} + \ldots + d_{-m} \times r^{-m}$$

Binary to Decimal Conversion

2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}
b_3	b_2	b_1	b_0	.	b_{-1}	b_{-2}	b_{-3}	b_{-4}
8	4	2	1		$^1/_2$	$^1/_4$	$^1/_8$	$^1/_{16}$

binary point

$(1010.101)_2$
$= (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
$+ 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})_{10}$
$= (8 + 2 + 0.5 + 0.125)_{10}$
$= (10.625)_{10}$

Binary Number	Decimal Value
1	$2^0 = 1$
10	$2^1 = 2$
100	$2^2 = 4$
1000	$2^3 = 8$
10000	$2^4 = 16$
10000 0000	$2^8 = 256$
100 0000 0000	$2^{10} = 1024$ (1 K)
1000 0000 0000	$2^{11} = 2048$ (2 K)
10000 0000 0000 0000 0000	$2^{20} = 1024$ K (1 M)

Binary	Decimal
11	3
111	7
1111	15
1111 1111	255

Decimal to Binary Conversion

Conversion of $(10)_{10}$

Number	Divide by	Quotient	Remainder
10	2	5	0 (LSB)
5	2	2	1
2	2	1	0
1	2	0	1 (MSB)

1010

Integer

$(10)_{10} = (1010)_2$

Number	Multiplied by	Carry	Fraction
0.625	2	1 (MSB)	0.25
0.25	2	0	0.5
0.5	2	1 (LSB)	0

101

Conversion of $(0.625)_{10}$

Fraction

$(0.625)_{10} = (0.101)_2$

More Example

$(23.6)_{10} = (10111.10011)_2$

(fractional part truncated at 5 bits)

- Convert $(23.6)_{10}$ to binary

No.	÷	Quot.	Remainder
23	2	11	1
11	2	5	1
5	2	2	1
2	2	1	0
1	2	0	1

Conversion of 23

Integer part

No.	X	Carry	Fraction
0.6	2	1	0.2
0.2	2	0	0.4
0.4	2	0	0.8
0.8	2	1	0.6
0.6	2	1	0.2
0.2	…	…	…

Conversion of 0.6

Fractional part

Octal and Hexadecimal Representation

Similar to binary

Octal / Hex to Decimal:
Using base 8 / 16

Decimal to Octal / Hex:
Int.: Division by 8 / 16
Frac.: Multi. by 8 / 16

Octal: Base – 8

Valid digits: 0,1,2,3,4,5,6,7

8^2	8^1	8^0		8^{-1}	8^{-2}
o_2	o_1	o_0	.	o_{-1}	o_{-2}

Octal point

Octal: 35.6 ⇨ Binary: 011101.110
= 11101.11

Binary: 1101110.1011 ⇨ Octal: 156.54
= 001101110.101100

Hexadecimal: Base – 16

Valid digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

16^2	16^1	16^0		16^{-1}	16^{-2}
h_2	h_1	h_0	.	h_{-1}	h_{-2}

Hexadecimal point

Hex: 3D.6 ⇨ Binary: 00111101.0110
= 111101.011

Binary: 1101110.101101 ⇨ Hex: 6E.B4
= 01101110.10110100

Fixed-point Representation

- In practice, fixed number of bits are available to represent a binary number.
- In fixed-point representation, (i) width and (ii) position of binary point are defined.
- Similar to integer representation which is a special case where no bit assigned after binary point.

Consider 8 bits stored in memory is: 10011110.

When represented as *fixed*(8,3), it is

$10011.110 = (1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2})_{10} = (19.75)_{10}$

When represented as *fixed*(8,4), it is

$1001.1110 = (1 \times 2^3 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3})_{10} = (9.875)_{10}$

When represented as *fixed*(8,2), it is

$100111.10 = (1 \times 2^5 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1})_{10} = (39.5)_{10}$

Note that, one left shift of binary point is division by 2 and one right shift is multiplication by 2.

Due to similarity, integer arithmetic circuit can be used directly here.

Floating Point Representation

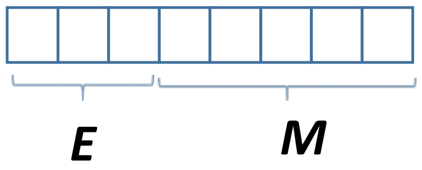
For *fixed*(8,4), precision is 2^{-4} = 0.0625; range is 0000.0000 to 1111.1111 i.e. 0 to 15.9375
For *fixed*(8,3), precision is 2^{-3} = 0.125; range is 00000.000 to 11111.111 i.e. 0 to 31.875

***fixed*(8,4) has better precision over *fixed*(8,3) but lower range.**

If a and b are the bits before and after binary point, respectively then the range is $2^a - 2^{-b}$.

In floating-point representation, the binary point is not fixed.
Gap between consecutive numbers is high for larger numbers, small for smaller numbers.

M: Mantissa
E: Exponent



Number = $M \times 2^E$

Normalized representation: 1101.01 = (1.10101) x 2^3

01110101

References:

- Donald P. Leach, Albert P. Malvino, and Goutam Saha, Digital Principles & Applications 8e, McGraw Hill
- Technical documents from http://www.ti.com accessed on Oct. 08, 2018