Engr433: Digital Design

Review of Combinational Circuits

Dr. Curt Nelson

Combinational Review Topics

- Number Systems
  - Bases
    - Decimal
    - Binary
    - Hexadecimal
  - Conversions
    - Decimal <==> Binary
    - Decimal <==> Hexadecimal
    - Binary <==> Hexadecimal
- Basic gates
  - Inverters, And, Nand, Or, Nor, Xor, Xnor

Combinational Review Topics - Continued

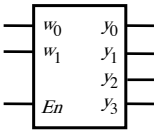
- Boolean Algebra
  - Basic Laws
  - De'Morgans Theorems
- Truth Tables
  - Minterms, Sum of Products
  - Maxterms, Product of Sums
- Karnaugh Maps
  - Sum of Products - Minterms
  - Product of Sums - Maxterms
  - Minimization
    - Don't Cares
  - Entered Variable Mapping (EVM)

Combinational Review Topics - Continued

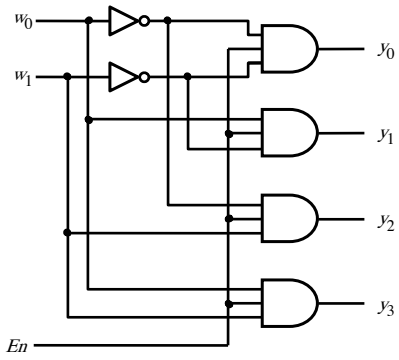
- Medium and Large Scale Integration (MSI and LSI)
  - Arithmetic: Adders / Subtractors
  - Data Format Converters: Encoders / Decoders
  - Data Selectors: Multiplexers/Demultiplexers
  - Other functions

Digital Abstraction

- Why do we use abstraction in the digital arena?
  - To manage complexity.

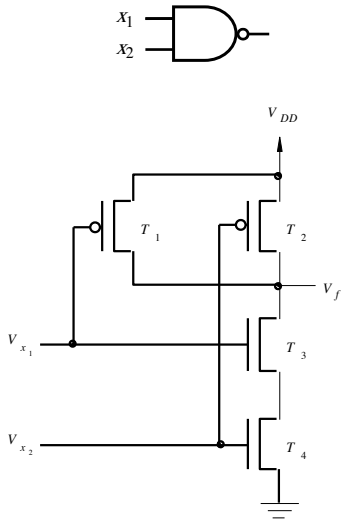


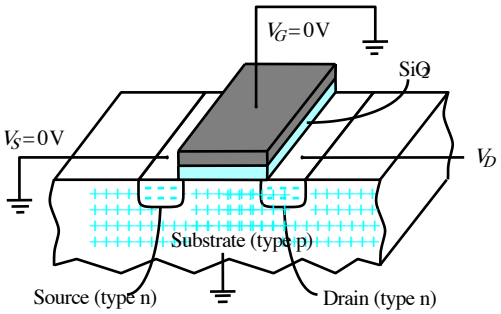
(b) Graphic symbol



(c) Logic circuit

Digital Abstraction





Truth Tables

- All combinations of inputs on the left;
- Outputs on the right;
- 2-input **AND** and **OR** functions shown below.

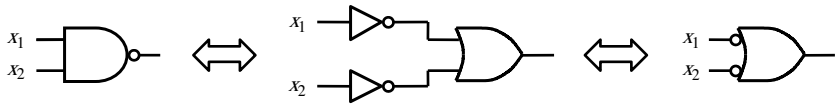
x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1
		AND	OR

Truth Table Proof of DeMorgan's Theorem

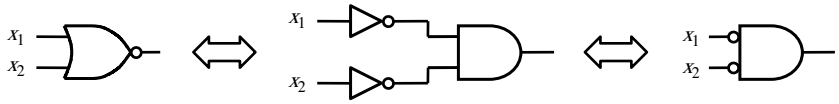
$\overline{x \cdot y} = \overline{x} + \overline{y}$ DeMorgan's Theorem

x	y	$x \cdot y$	$\overline{x \cdot y}$	$\overline{x}$	$\overline{y}$	$\overline{x} + \overline{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0
		LHS		RHS		

DeMorgan's Theorems in Terms of Logic Gates



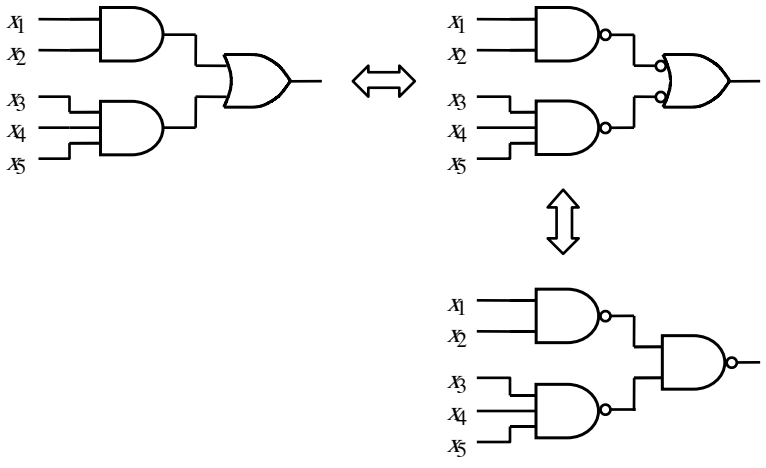
(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$



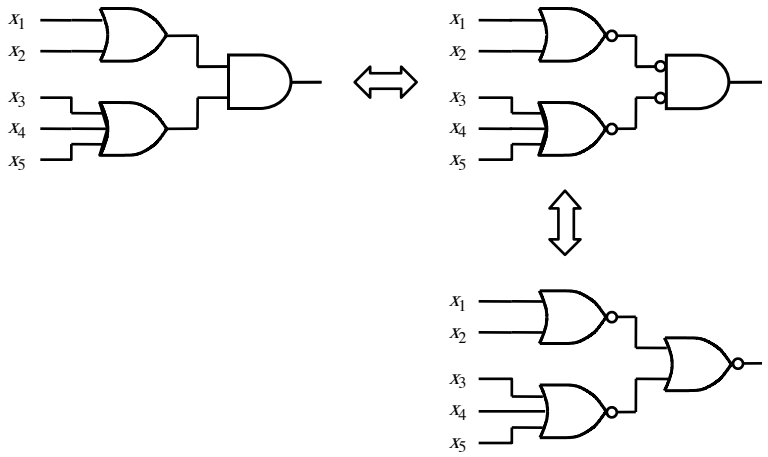
(b) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

- Function vs. Gate

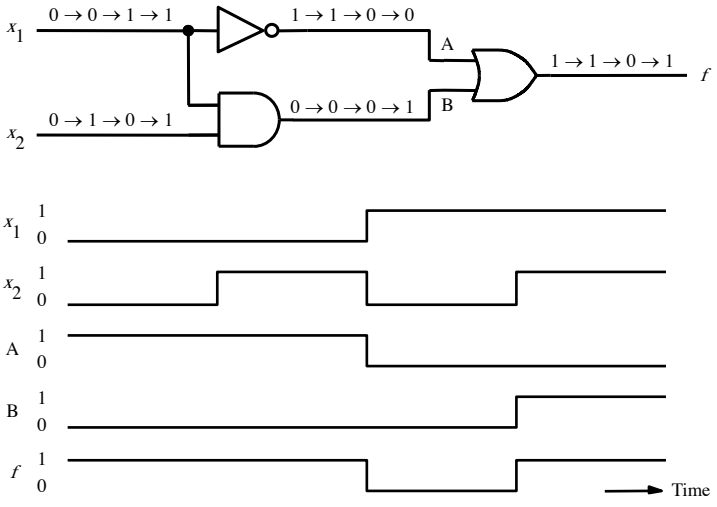
SOP Implementation Using NAND Gates



POS Implementation Using NOR Gates



Timing Diagram

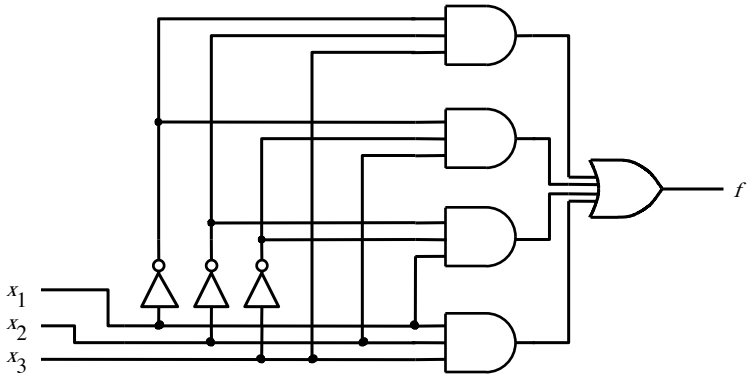


Logic Synthesis

- *Logic synthesis,* or *logic optimization,* is the process of translating a truth table, schematic, or VHDL code into a network of logic gates.
- Example:
  - Minterm form;
  - Maxterm form;
  - Minimum form.

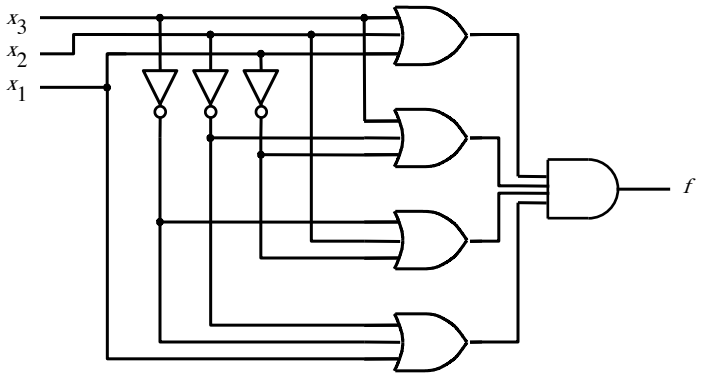
x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

SOP Implementation – NAND Gates



(a) Sum-of-products realization

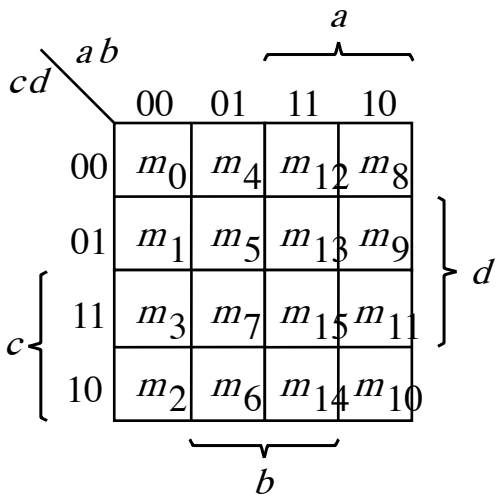
POS Implementation – NOR Gates



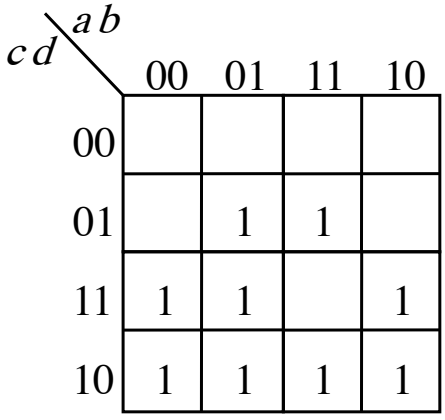
(b) Product-of-sums realization

Minimal Implementation

4-Variable Karnaugh Map (Kmap)



4-Variable Kmap Example



Four-variable function $f = \Sigma\, m(2, 3, 5, 6, 7, 10, 11, 13, 14)$

4-Variable Function with Don't Cares

cd \ ab	00	01	11	10
00	0	1	d	0
01	0	1	d	0
11	0	0	d	0
10	1	1	d	1

$f = \Sigma\, m(2, 4, 5, 6, 10) + d(12, 13, 14, 15)$

Entered Variable (EV) Mapping

- Sometimes called Map Entered Variables (MEV's);
- Allows many variables to be presented using a reduced size K-map;
- Occurs quite frequently in digital systems, especially state machines;
- May require K-map **compression** and **expansion**;
- References (entered-variable mapping or map-entered variables):
  - Tinder, Engineering Digital Design, Second Edition (Library);
  - Other digital logic textbooks;
  - Web – YouTube, etc.

Entered Variable (EV) Mapping Example

a	b	c	*f*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

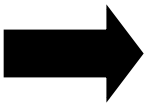
$f = \Sigma\, m(2, 5, 6, 7)$

c \ *ab*	00	01	11	10
0	0	1	1	0
1	0	0	1	1

f = a'bc' + ab'c + abc' + abc
= bc' + ac

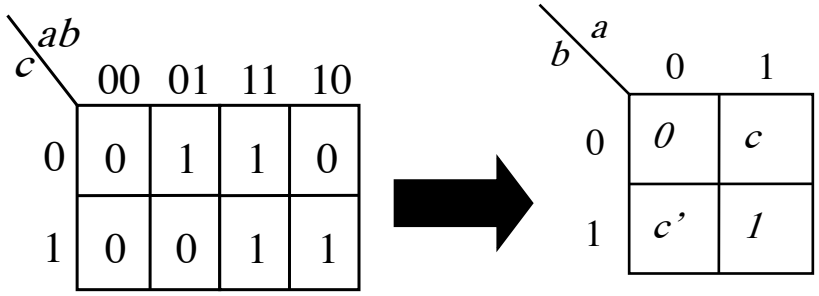
Entered Variable Truth Table Compression

a	b	c	*f*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



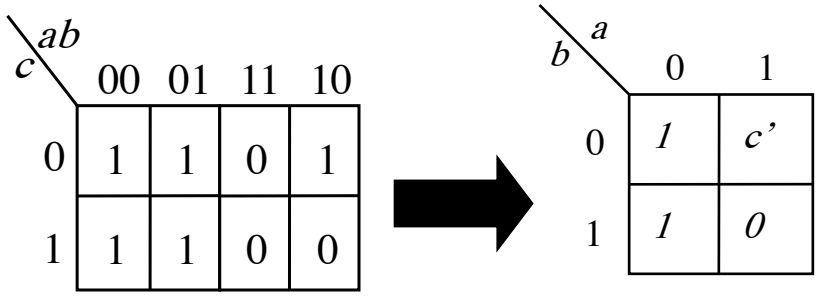
a	*b*	*f*
0	0	*0*
0	1	*c'*
1	0	*c*
1	1	*1*

Entered Variable Map Compression



f = a'bc' + ab'c + abc' + abc
= bc' + ac

Three Variable Compression Example



1 = c + c'

0 = cc'

Four Variable Compression Example

cd \ ab	00	01	11	10
00	1	1		
01		1	1	
11			1	1
10	1			1

The function $f = \Sigma\, m(0, 2, 4, 5, 10, 11, 13, 15)$

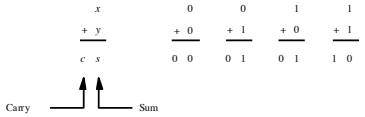
c \ a b	00	01	11	10
0	d’	1	d	0
1	d’	0	d	1

b \ a	0	1
0	c’d’+cd’ =d’	c’0+c1 = c
1	c’1 + c0 = c’	c’d+cd=d

Other Examples

- Expansion;
- Compression and expansion using don’t cares.

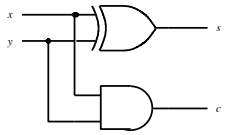
MSI Circuits – Half Adder



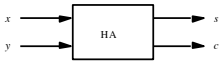
(a) The four possible cases

x	y	Carry c	Sum s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) Truth table



(c) Circuit

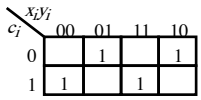


(d) Graphical symbol

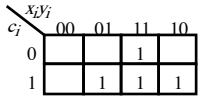
MSI Circuits – Full Adder

c_i	x_i	y_i	c_{i+1}	s_i
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

(a) Truth table

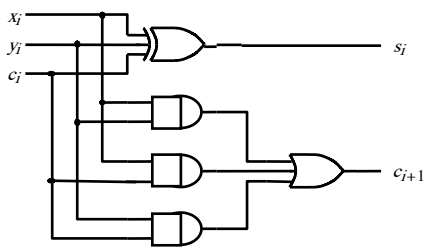


$s_i = x_i \oplus y_i \oplus c_i$



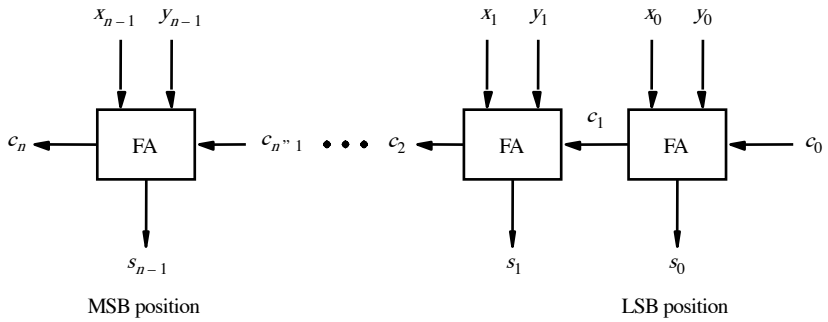
$c_{i+1} = x_iy_i + x_ic_i + y_ic_i$

(b) Karnaugh maps



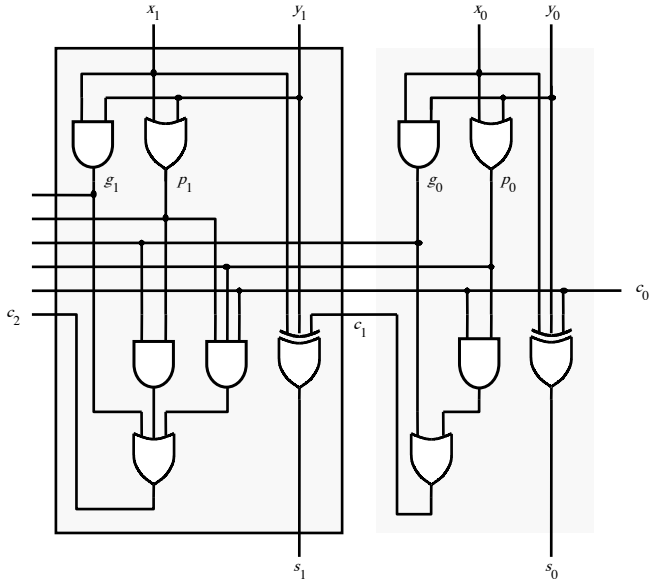
(c) Circuit

N-bit Ripple Carry Adder

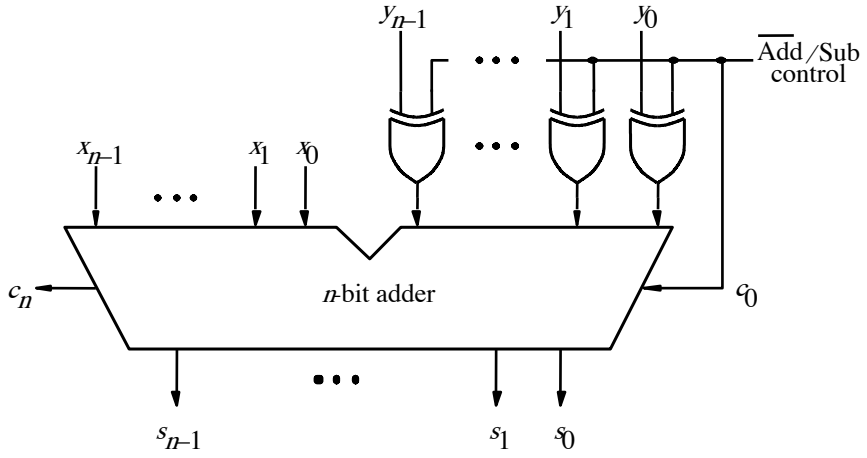


- Major issue – propagation delay.

Faster Adders – Carry Lookahead

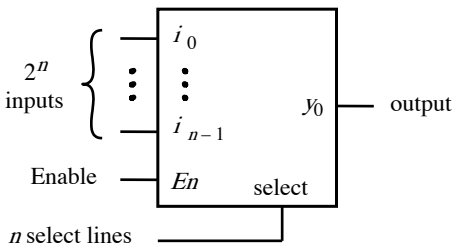


Adder / Subtractor Circuit



Multiplexers

- Devices that select one of many inputs to be routed to one output based on the binary value of select lines
  - Enable – used to enable or disable the complete function;
  - Select – used to select which one of the inputs gets routed to the output.



Synthesis of Logic Functions Using Mux's

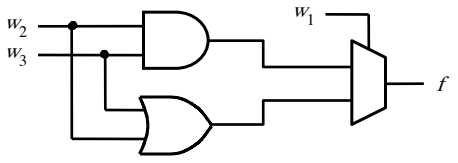
- Mux's can be used to synthesize logic functions as follows:
  - Create truth table;
  - Compress as necessary;
  - Implement.
- In general, an *N* variable function can be implemented with one *N-1* multiplexer and at most, one inverter.

Example: 3-Input Majority Function

w_1	w_2	w_3	f
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

w_1	f
0	$w_2 w_3$
1	$w_2 + w_3$

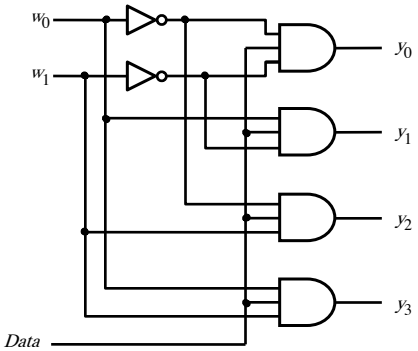
(a) Truth table



(b) Circuit

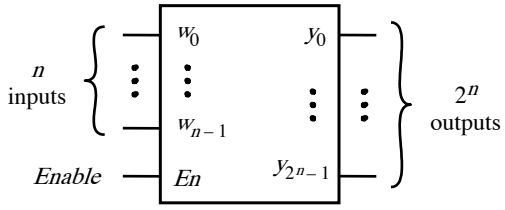
De-Multiplexers

- A de-multiplexer is a circuit which places the value of a single data input onto one of a number of outputs.



An *n*-to-2^n Decoder

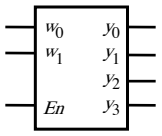
- A decoder is a device that activates one output based on the binary value of the inputs;
- A decoder is a minterm generator;
- Enable – used to enable or disable the complete decoder function.



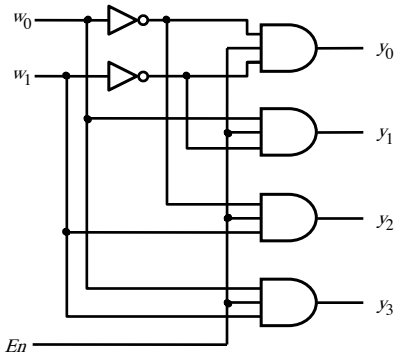
A 2-to-4 Decoder

En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table



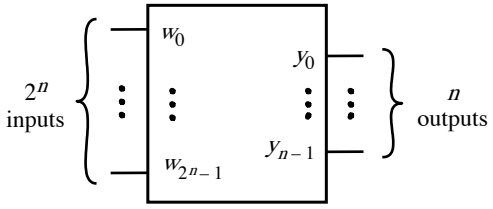
(b) Graphic symbol



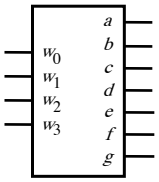
(c) Logic circuit

A 2^n-to-n Binary Encoder

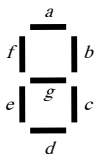
- An encoder is a device that outputs a binary code representing which one of many inputs is active.
- Priority encoder – assigns priority to certain inputs
  - Used in embedded computer systems to service interrupts.



BCD to 7-segment Code Converter



(a) Code converter

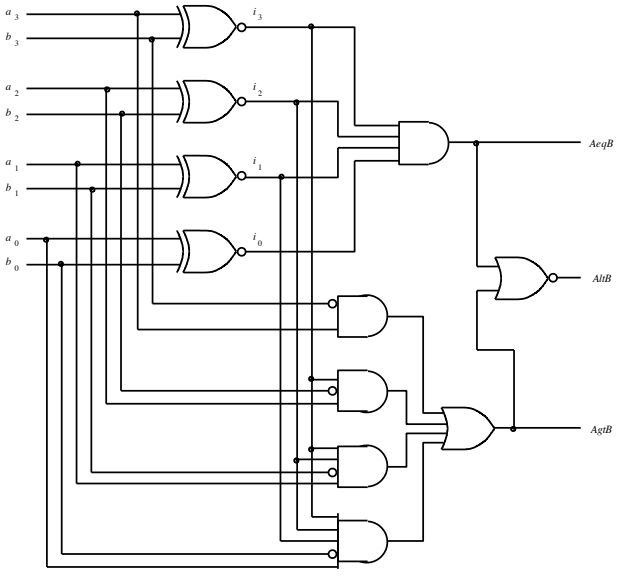


(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

(c) Truth table

A Four-bit Magnitude Comparator



Arithmetic Logic Units (74HC381)

Operation	Inputs s_2 s_1 s_0	Outputs F
Clear	0 0 0	0 0 0 0
B=A	0 0 1	$B = A$
A=B	0 1 0	$A = B$
ADD	0 1 1	$A + B$
XOR	1 0 0	A XOR B
OR	1 0 1	A OR B
AND	1 1 0	A AND B
Preset	1 1 1	1 1 1 1