CIRCUITS: INTRODUCTION

- Some electronics
- From Boolean algebra to circuits

TAKING INTO ACCOUNT POWER (1)

Voltage has to be considered with a lot of attention because power P verifies:

$$P = \alpha C V^2 f$$

C: circuit capacitance

V: Supply voltage

f : Frequency;

Furthermore $f = \gamma V$.

Therefore Power varies as a cubic function of Voltage.

High voltage has three corollaries:

- High power therefore hot chips requiring specific cooling system
- High energy consumption therefore shorter battery life
- High frequencies (overclocking ☺)

n-type MOS TRANSISTOR

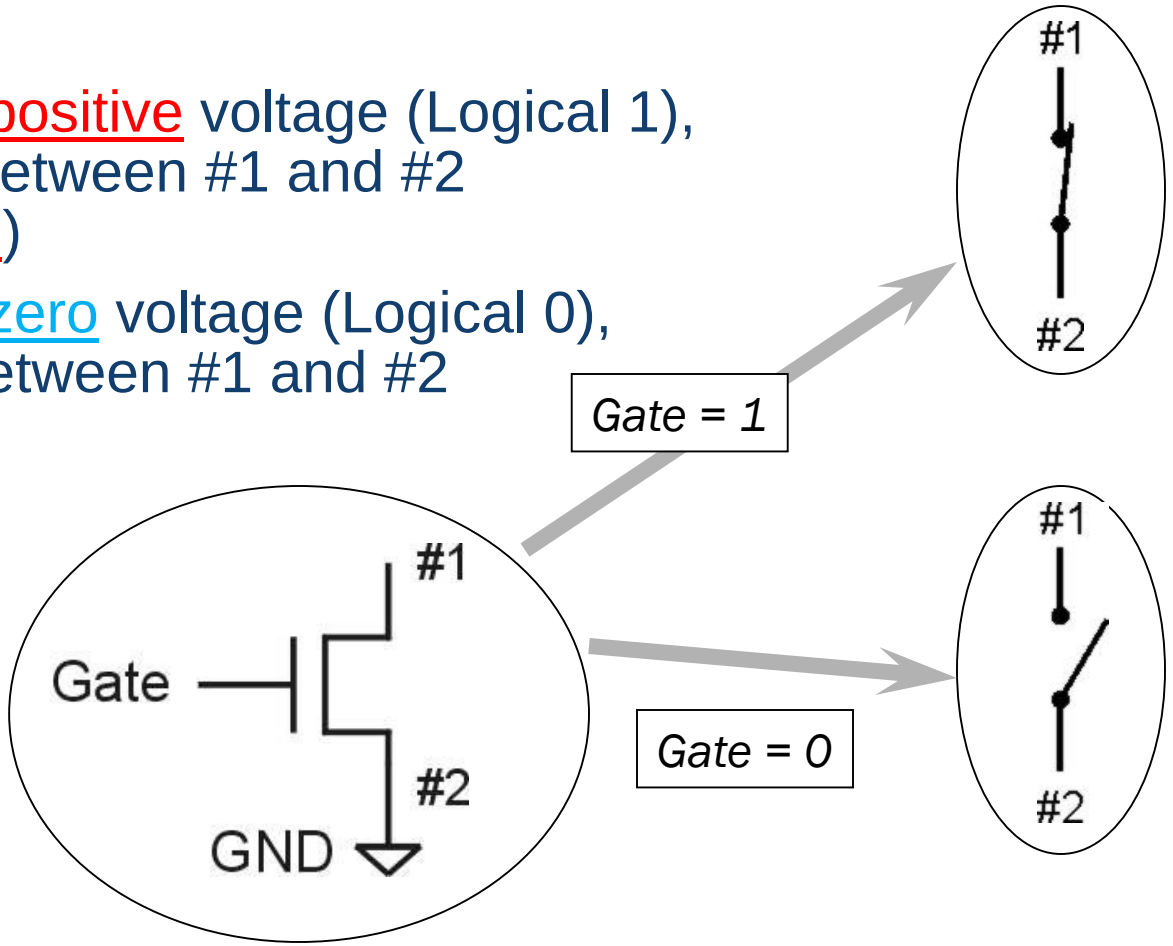
MOS = Metal Oxide Semiconductor

two types: n-type and p-type

n-type

when Gate has positive voltage (Logical 1),
short circuit between #1 and #2
(switch closed)

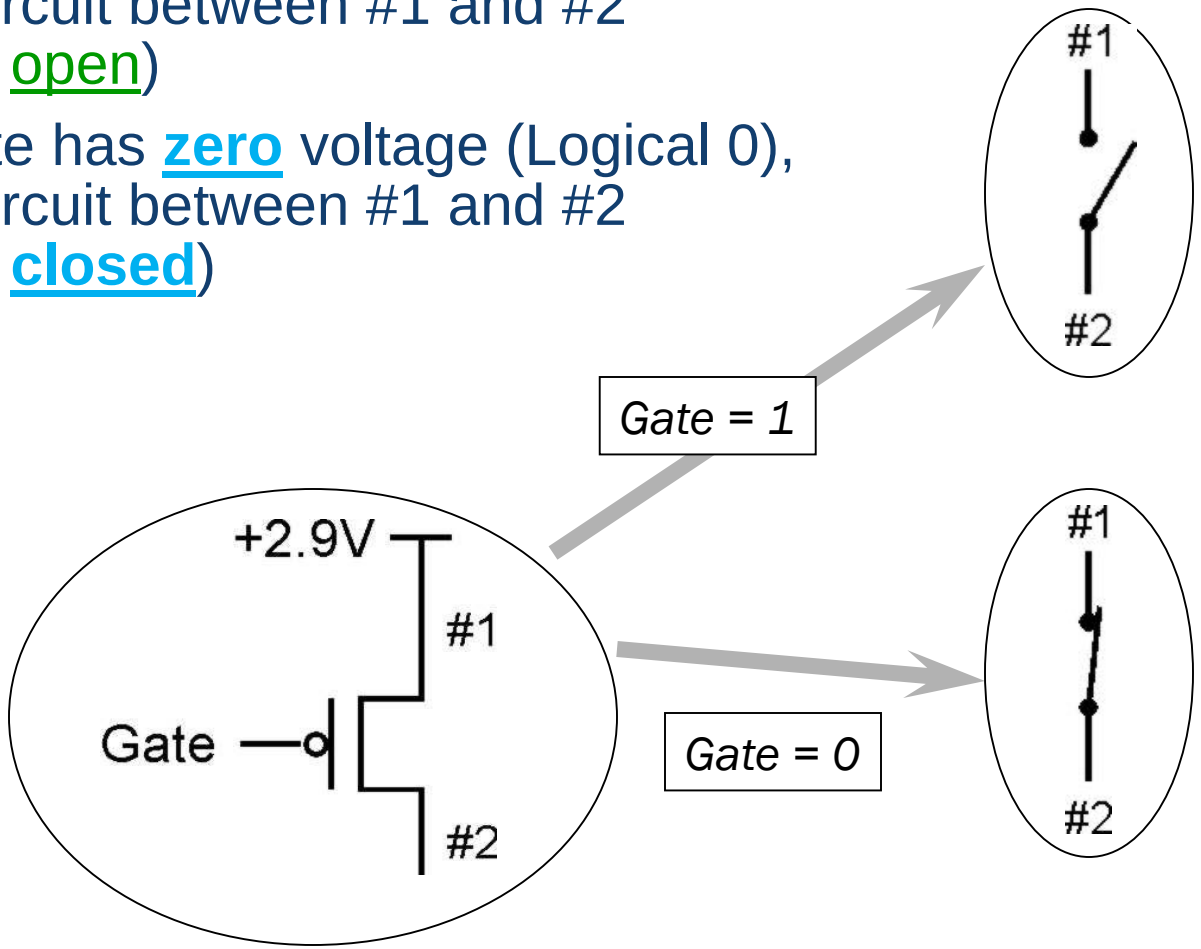
when Gate has zero voltage (Logical 0),
open circuit between #1 and #2
(switch open)



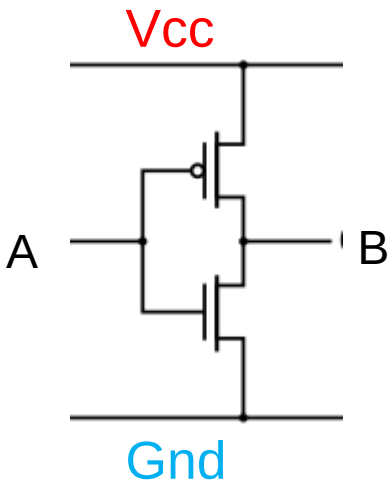
p-type MOS TRANSISTOR

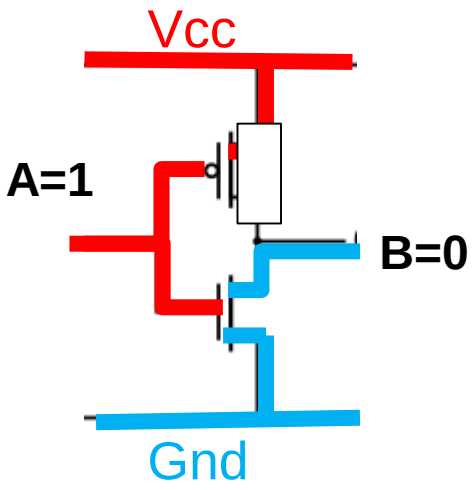
p-type is *complementary* to n-type

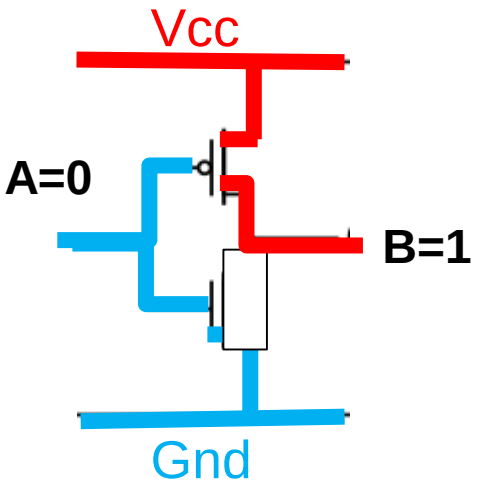
- when Gate has **positive** voltage (Logical 1)
  open circuit between #1 and #2
  (switch open)
- when Gate has **zero** voltage (Logical 0),
  short circuit between #1 and #2
  (switch **closed**)



CMOS INVERTER GATE

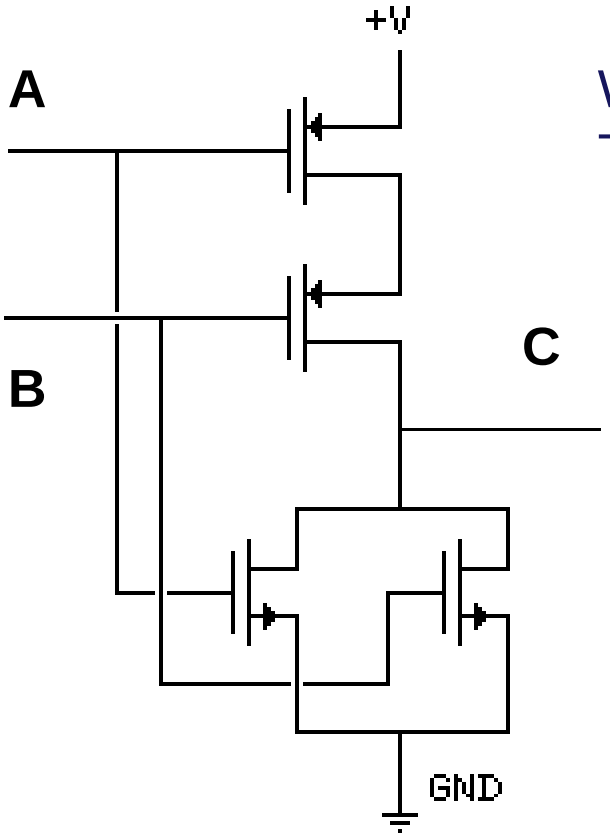






SOURCE OF THE DRAWINGS : WIKIPEDIA

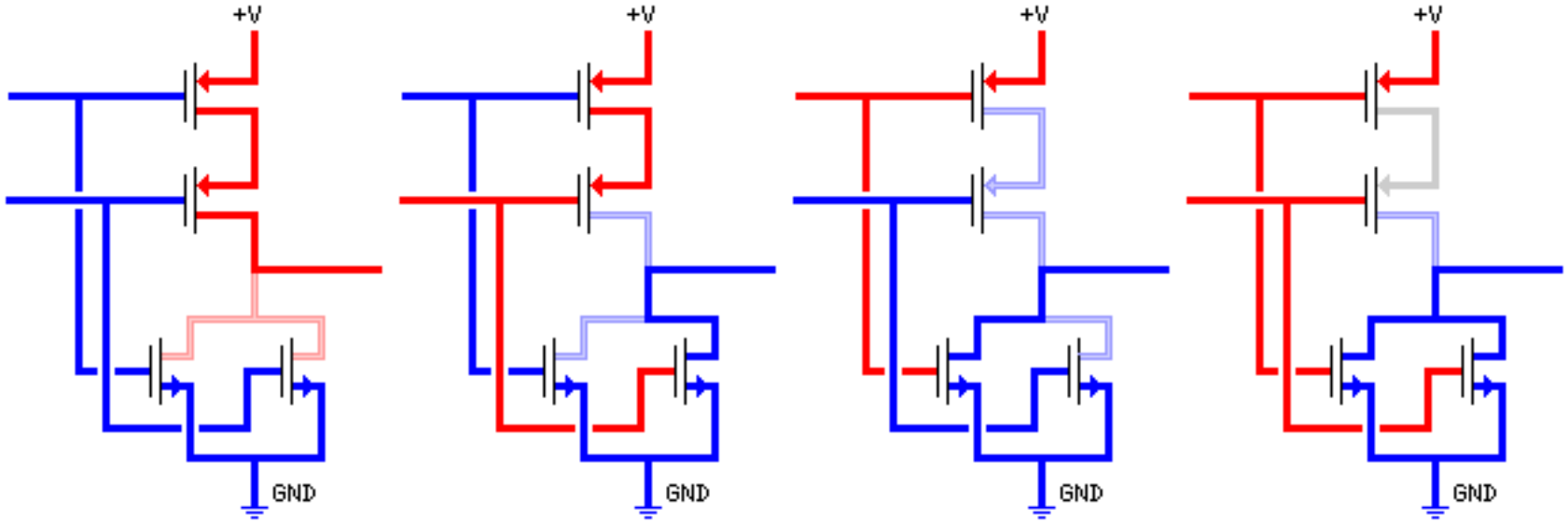
CMOS NOR GATE



WARNING: the symbols used for the N Type and P Type transistors is different!!

SOURCE OF THE DRAWINGS: http://www.quadibloc.com/comp/cp01.htm

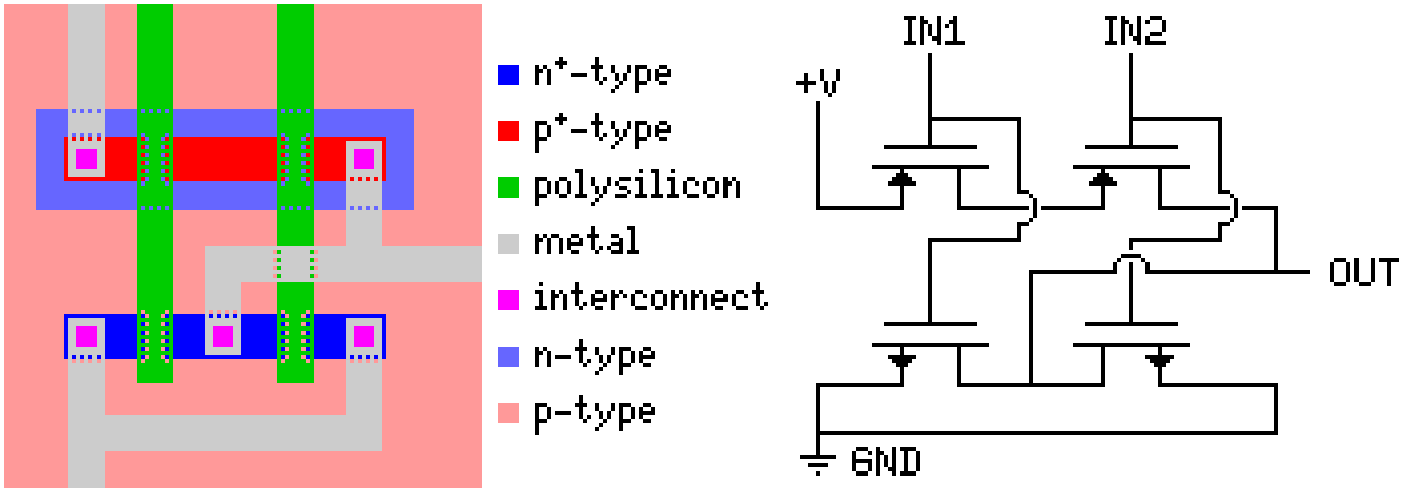
CMOS NOR GATE BEHAVIOR



A	B	C = NOR(A,B)
0	0	1
0	1	0
1	0	0
1	1	0

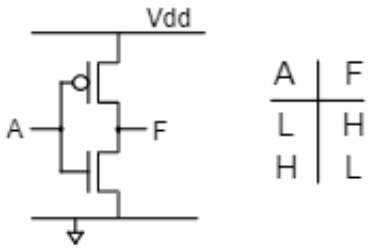
SOURCE OF THE DRAWINGS: http://www.quadibloc.com/comp/cp01.htm

INTEGRATED CIRCUIT LAYOUT OF A NOR GATE



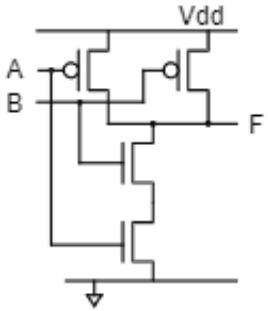
SOURCE OF THE DRAWINGS: http://www.quadibloc.com/comp/cp01.htm

BASIC CMOS GATES



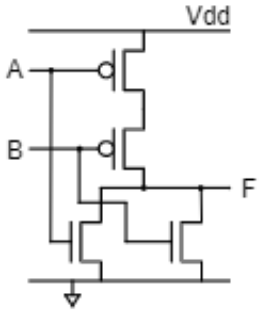
A	F
L	H
H	L

CMOS INVERTER



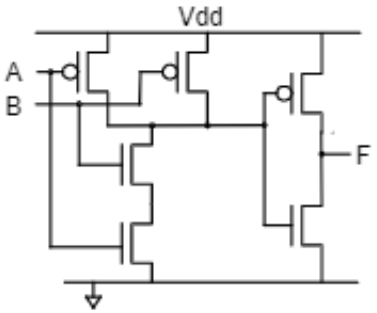
A	B	F
L	L	H
L	H	H
H	L	H
H	H	L

CMOS NAND



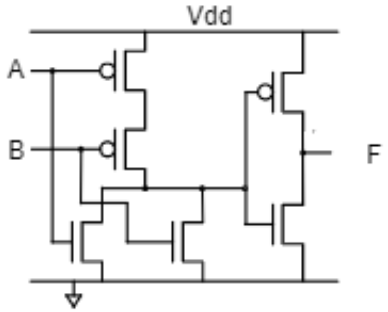
A	B	F
L	L	H
L	H	L
H	L	L
H	H	L

CMOS NOR



A	B	F
L	L	L
L	H	L
H	L	L
H	H	H

CMOS AND

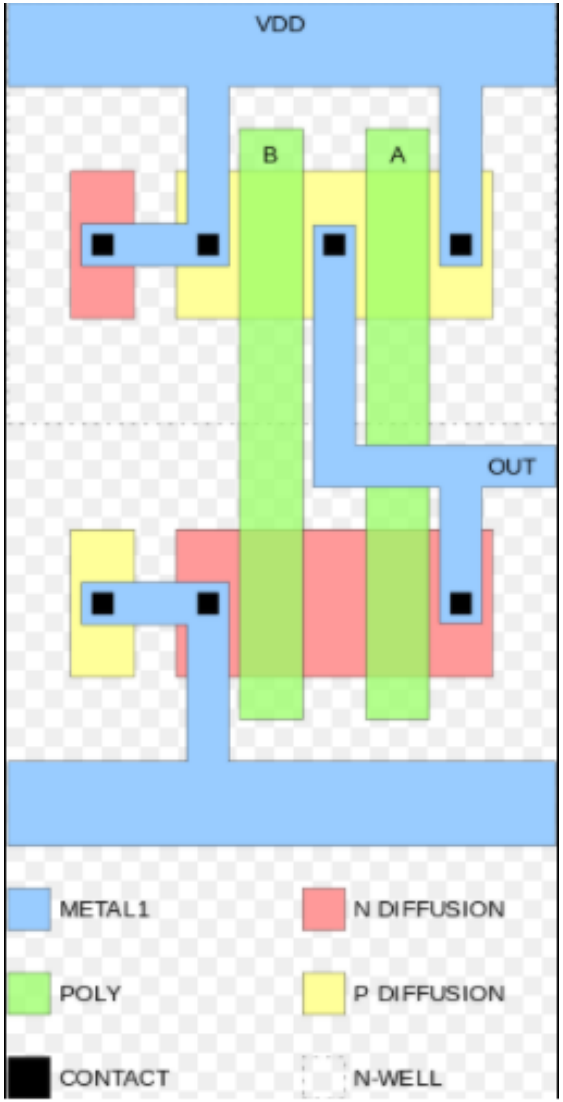


A	B	F
L	L	L
L	H	H
H	L	H
H	H	H

CMOS OR

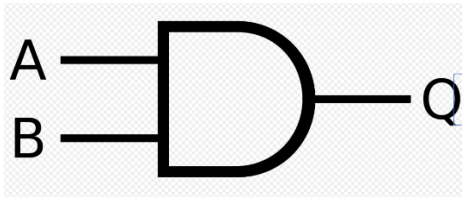
SOURCE OF THE DRAWINGS: https://learn.digilentinc.com/Documents/313

THE PHYSICAL LAYOUT OF A CMOS NAND

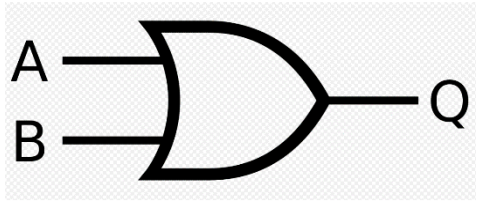


SOURCE OF THE DRAWINGS : WIKIPEDIA

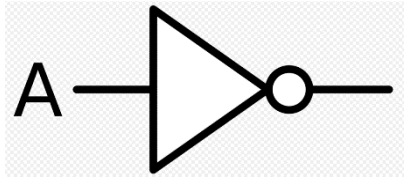
AND/OR/NOT STANDARD REPRESENTATION



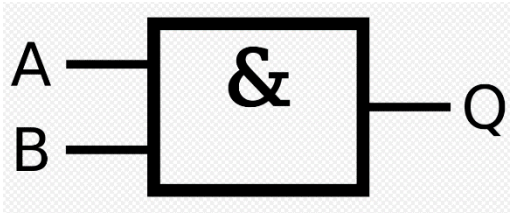
ANSI/MIL AND



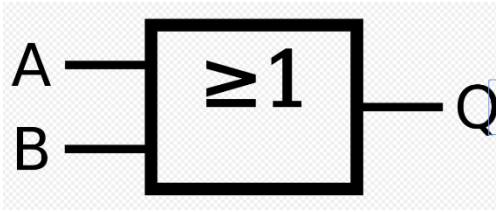
ANSI/MIL OR



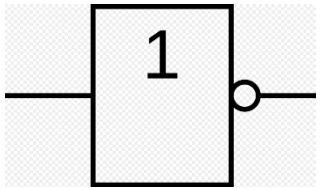
ANSI/MIL NOT



IEC AND



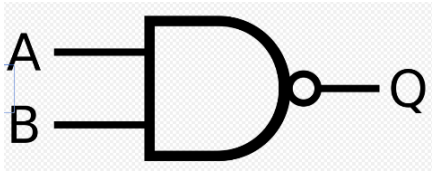
IEC OR



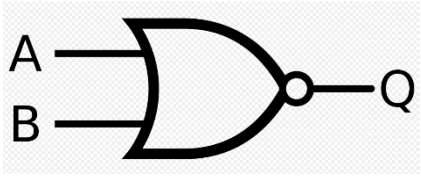
IEC NOT

SOURCE OF THE DRAWINGS : WIKIPEDIA

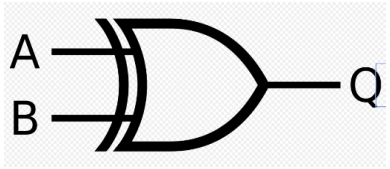
NAND/NOR/XOR/XNOR REPRESENTATIONS



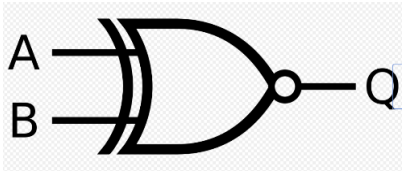
MIL/ANSI NAND



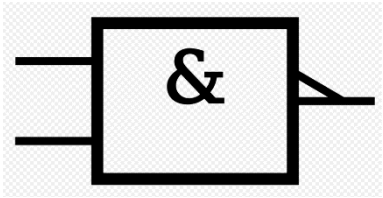
MIL/ANSI NOR



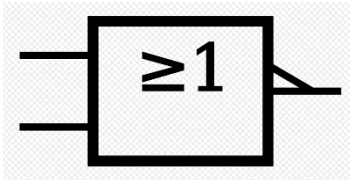
ANSI/MIL XOR



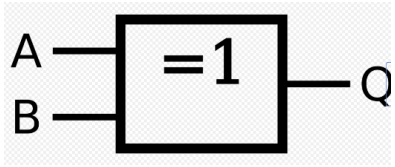
ANSI/MIL XNOR



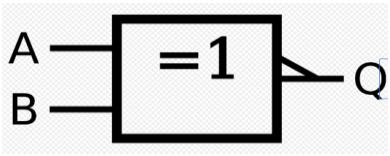
IEC NAND



IEC NOR



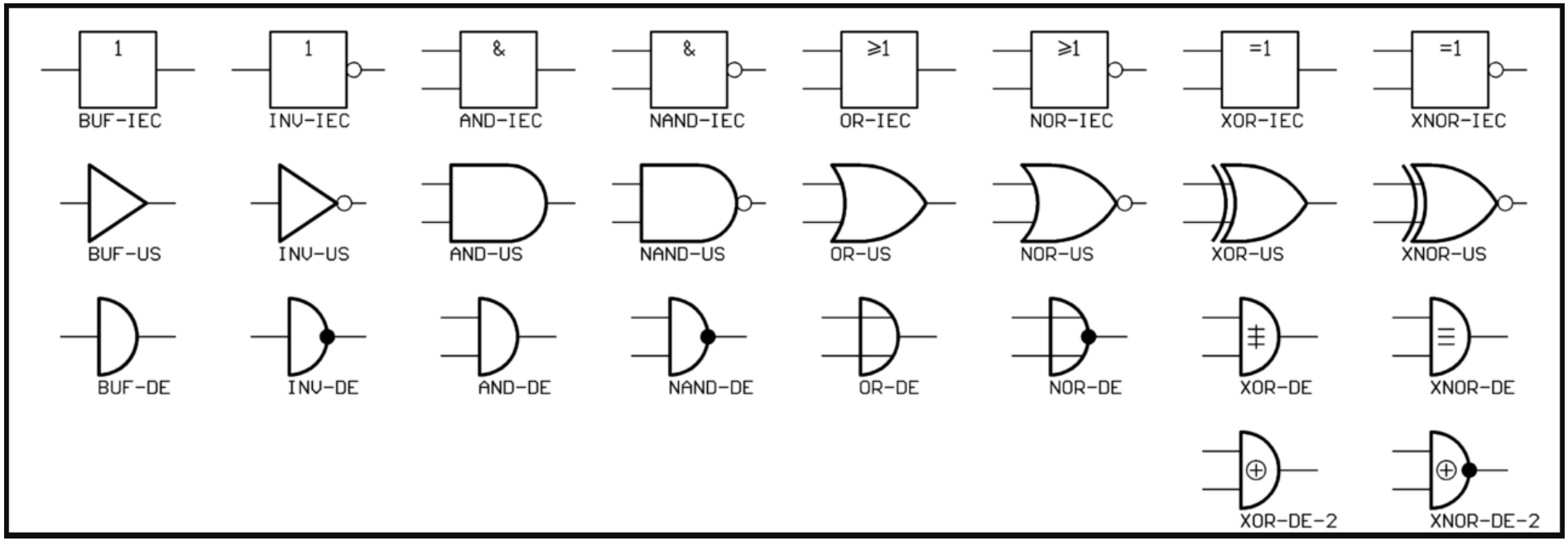
IEC XOR



IEC XNOR

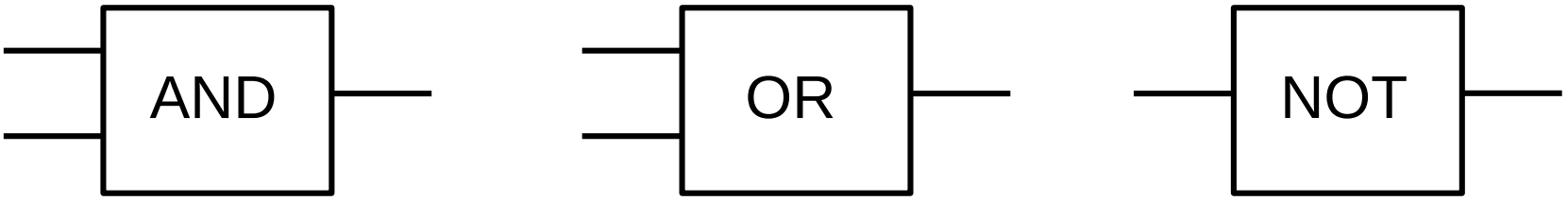
SOURCE OF THE DRAWINGS : WIKIPEDIA

EUROPEAN, US AND GERMAN NORMS

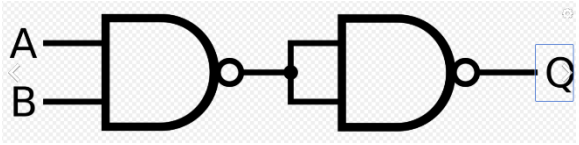


A PROPOSAL FOR EASY GATE DRAWING

Simply use rectangles and write inside the corresponding boolean function!! Non ambiguous and easy to draw.

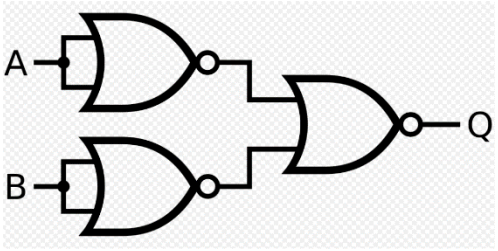


AND IMPLEMENTATIONS USING NAND / NOR



AND Using NAND gates

((ab)'(ab)')'=((ab)')'=ab

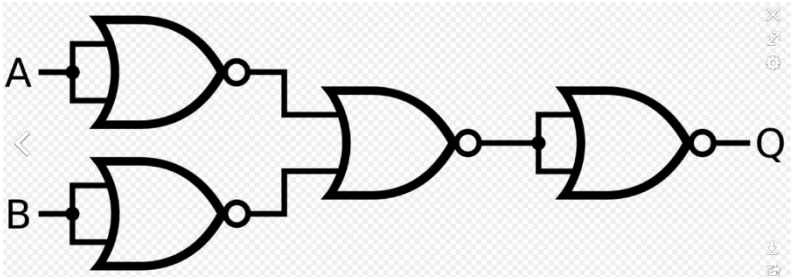


AND Using NOR gates

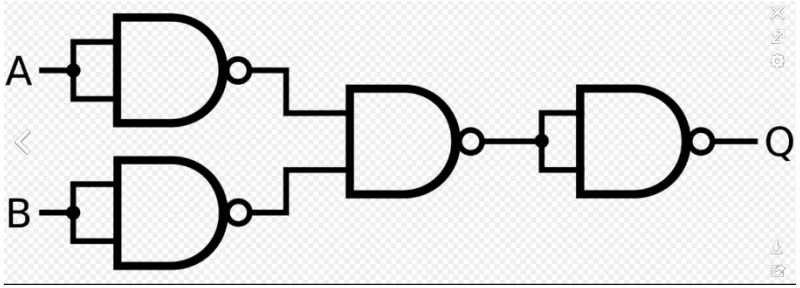
((a+a)' + (b+b)')'= (a'+b')'= ab

SOURCE OF THE DRAWINGS : WIKIPEDIA

NAND (resp. NOR) USING NOR (resp. NAND) GATES



NAND IMPLEMENTED USING NOR GATES
1 NAND NEEDS 4 NOR GATES



NOR IMPLEMENTED USING NAND GATES:
1 NOR NEEDS 4 NAND GATES

SOURCE OF THE DRAWINGS : WIKIPEDIA

FROM TRUTH TABLES TO CIRCUITS

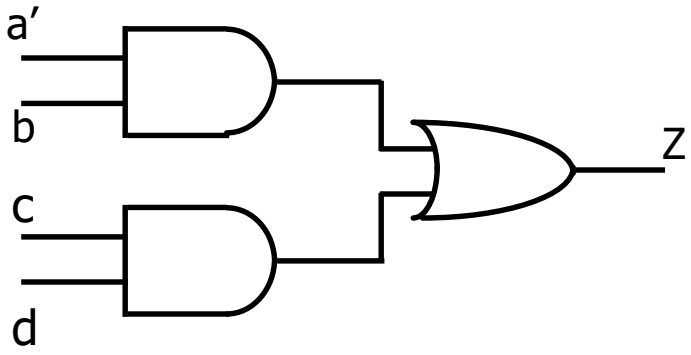
Given a truth table

1. Write the Boolean expression
2. Make the Boolean expression simpler (minimal SOP)
3. Draw as gates

Example:

a	b	c	F
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

$$F = a'bc'+a'bc+ab'c+abc$$
$$= a'b(c'+c)+ac(b'+b)$$
$$= a'b+ac$$



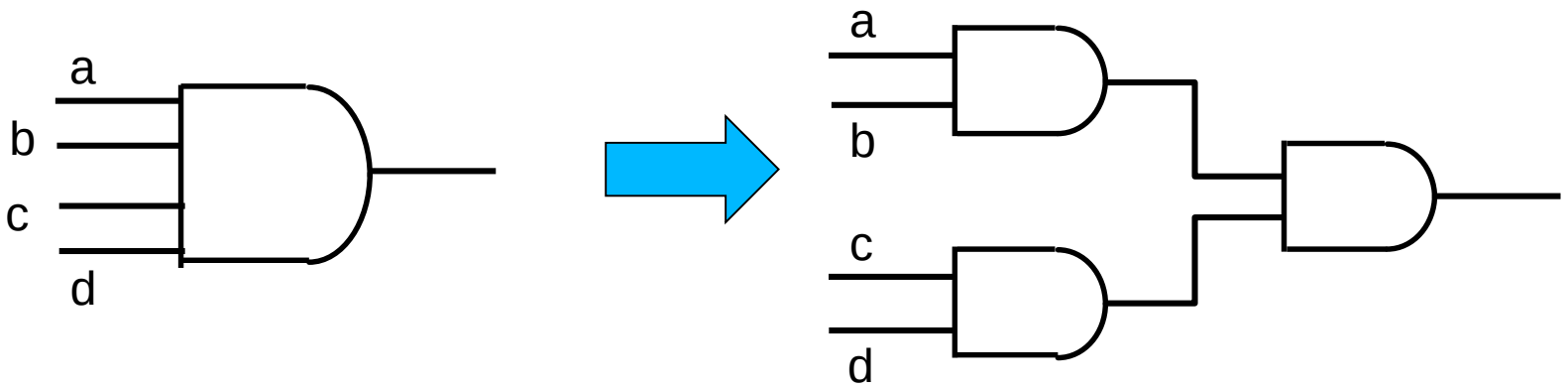
MULTI-INPUT GATES

AND/OR can take any number of inputs.

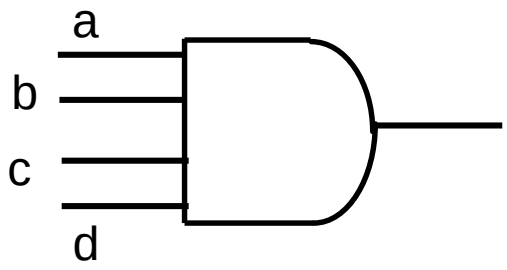
- AND = 1 if all inputs are 1, 0 otherwise: AND (a,b,c) = abc
- OR = 1 if any input is 1, 0 otherwise OR(a,b,c) = a+b+c

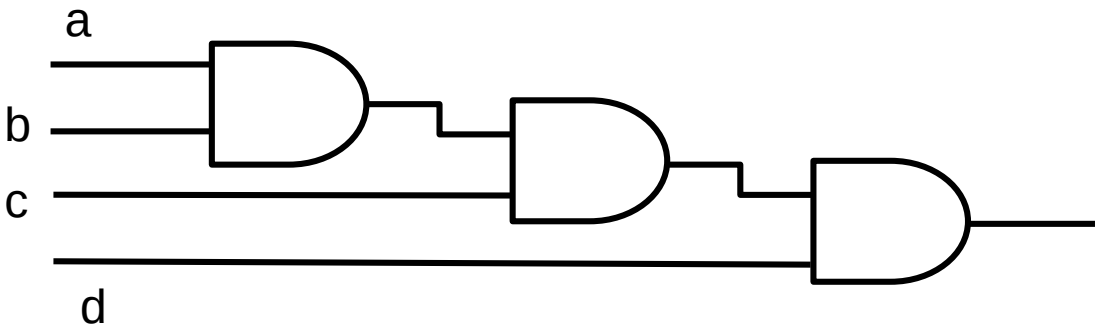
This can be extended to NAND/NOR. This is directly derived from operators associativity

Can implement with multiple two-input gates, or with single CMOS circuit.

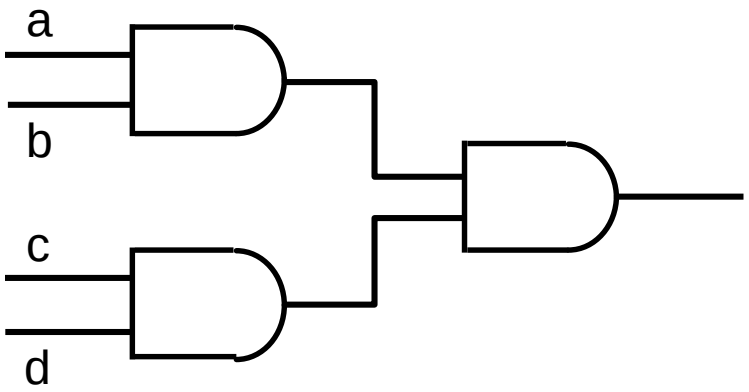


VARIOUS IMPLEMENTATIONS OF MULTI-INPUT





((a.b).c).d)



((a.b).(c.d))

REVISITING MORGAN'S LAWS

de Morgan's

Standard form:	a‘b' = (a + b)'	a' + b' = (ab)'
Inverted:	a + b = (a‘b')'	(ab) = (a' + b')'

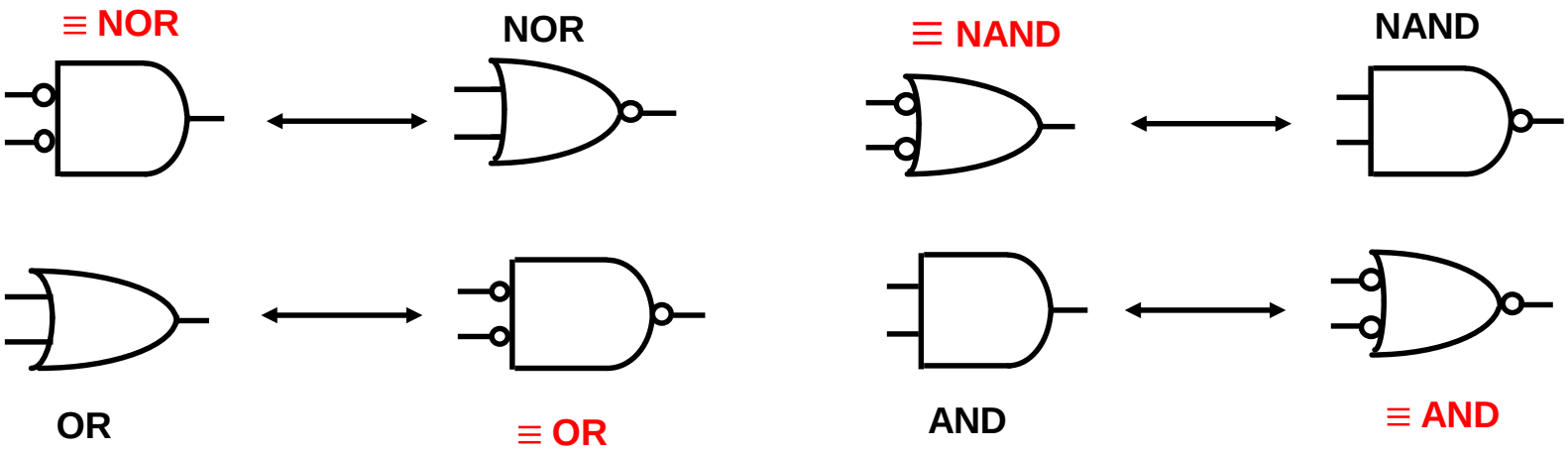
AND with complemented inputs ≡ NOR

OR with complemented inputs ≡ NAND

OR ≡ NAND with complemented inputs

AND ≡ NOR with complemented inputs

pushing the bubble



CONVERTING TO USE NAND/NOR (2)

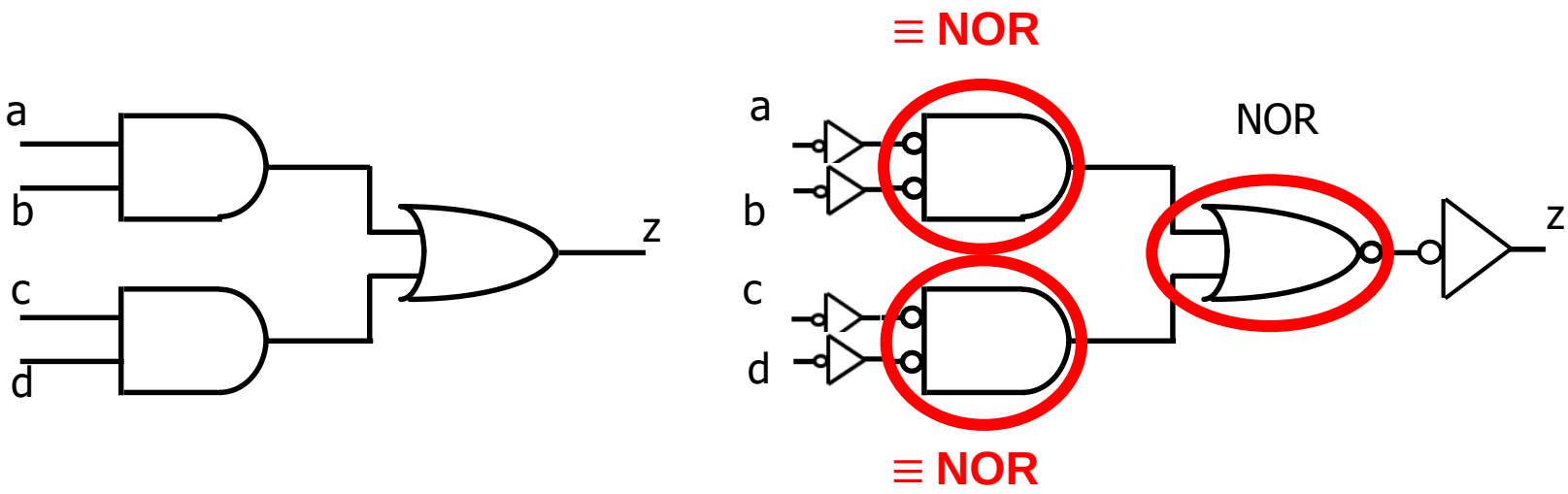
Example: AND/OR network to NOR/NOR

Z = ab+cd

= (a'+b')'+(c'+d')’

= {[(a'+b')'+(c'+d')’]’}’

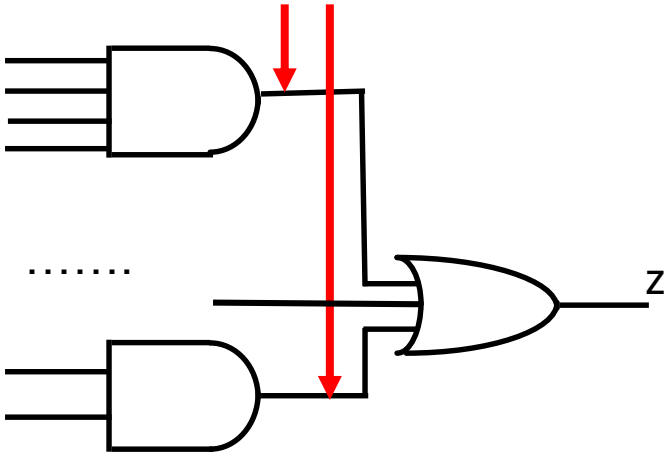
= {[((a’)’.(b’)’)+((c’)’)+(d’)’)]'}'

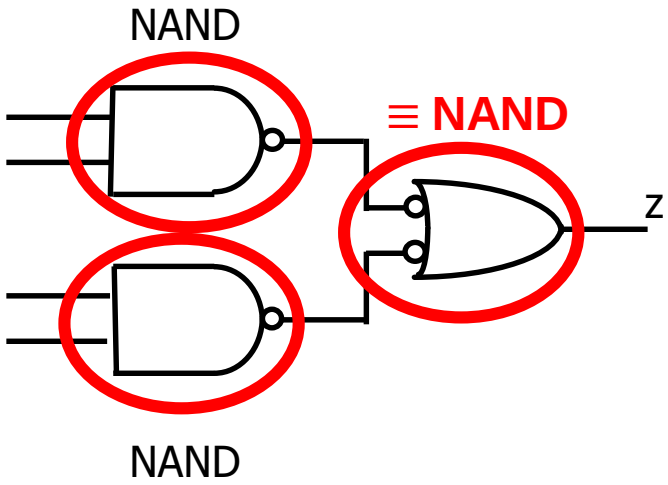


TRANSFORMING SOP'S INTO A BUNCH OF NANDS

AND/OR to NAND/NAND

On all wires, at both ends, add an inverter. At the input of the NAND gates, replace NOT by NAND





Acknowledgements

A lot of material was found in Wikipedia.

Some of these slides were inspired by slides developed by:

- University of Washington Computer Science & Engineering (CSE 370)
- Y.N. Patt (Univ of Texas Austin)
- S. J. Patel (Univ of Illinois Urbana Champaign)
- Walid A. Najjar (Univ California Riverside)
- Brian Linard (Univ California Riverside)
- G.T. Byrd (Univ North Carolina)