Lecture 5
Multilevel Logic Synthesis

Hai Zhou

ECE 303

Advanced Digital Logic Design

Spring 2002

Outline

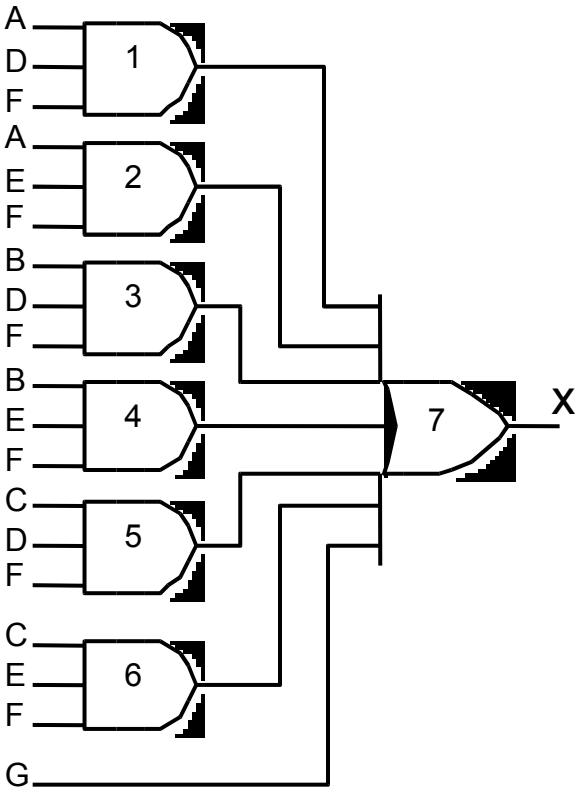
- Mapping 2-level logic to NAND logic, NOR logic
- CAD Tools for Multilevel Logic: MIS Algorithm
- Factored forms
- Flattening
- READING: Katz 3.1, 3.2

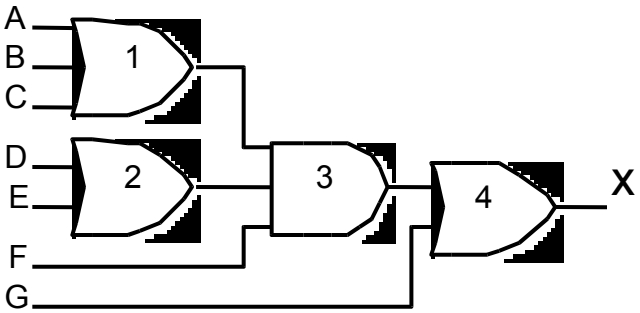
Multi-Level Logic: Advantages

Reduced sum of products form:

$$x = A\,D\,F + A\,E\,F + B\,D\,F + B\,E\,F + C\,D\,F + C\,E\,F + G$$

6 x 3-input AND gates + 1 x 7-input OR gate (may not exist!)
25 wires (19 literals plus 6 internal wires)





Factored form:

$$x = (A + B + C)\,(D + E)\,F + G$$

1 x 3-input OR gate, 2 x 2-input OR gates,
1 x 3-input AND gate
10 wires (7 literals plus 3 internal wires)

Multi-Level Logic: Conversion of Forms

NAND-NAND and NOR-NOR Networks

DeMorgan's Law: $(A + B)' = A' \cdot B'$; $(A \cdot B)' = A' + B'$

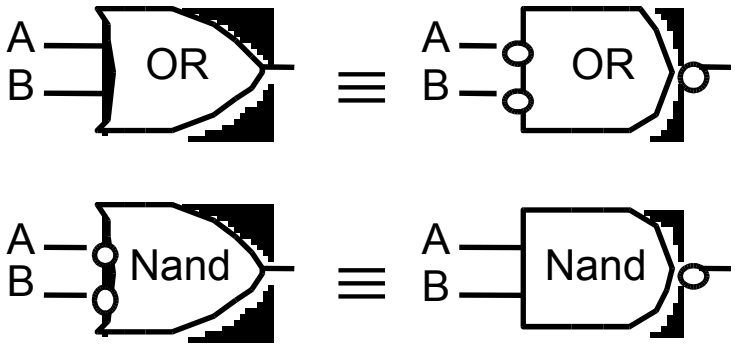
Written differently: $A + B = (A' \cdot B')'$; $(A \cdot B) = (A' + B')'$

In other words,

OR is the same as NAND with complemented inputs
AND is the same as NOR with complemented inputs
NAND is the same as OR with complemented inputs
NOR is the same as AND with complemented inputs

OR/NAND Equivalence

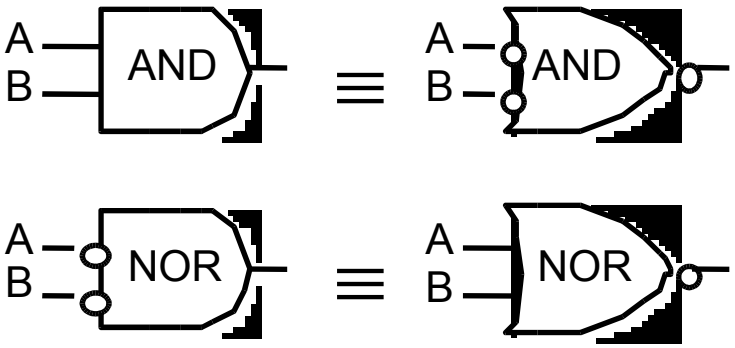
A	$\overline{A}$	B	$\overline{B}$	A + B	$\overline{A \cdot B}$	$\overline{A} + \overline{B}$	$\overline{A \cdot B}$
0	1	0	1	0	0	1	1
0	1	1	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	0	1	1	0	0



Multi-Level Logic: Conversion Between Forms

AND/NOR Equivalence

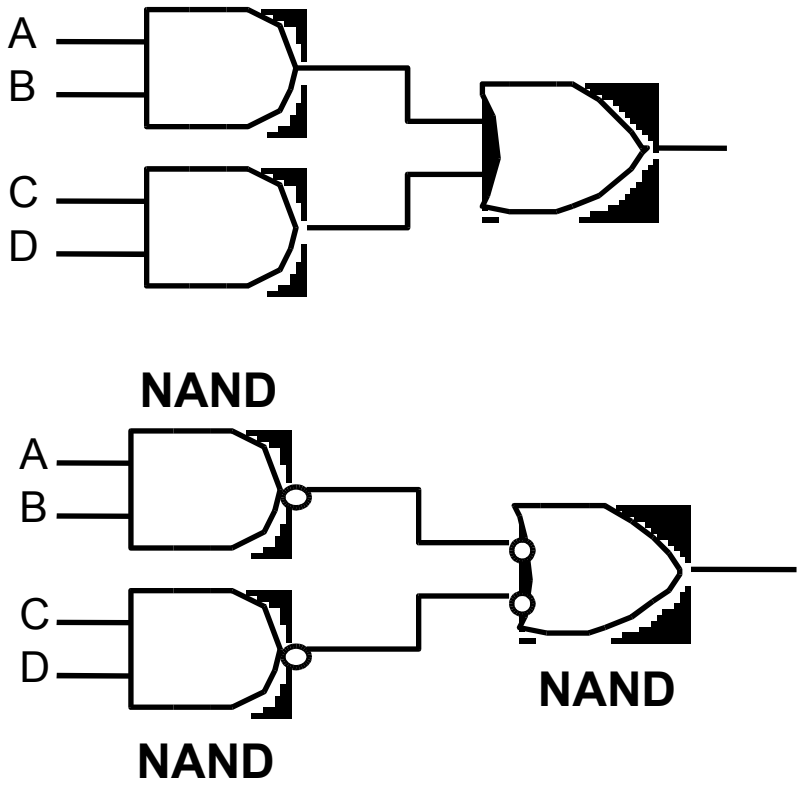
A	$\overline{A}$	B	$\overline{B}$	A • B	$\overline{A + B}$	$\overline{A} \cdot \overline{B}$	$\overline{A + B}$
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0



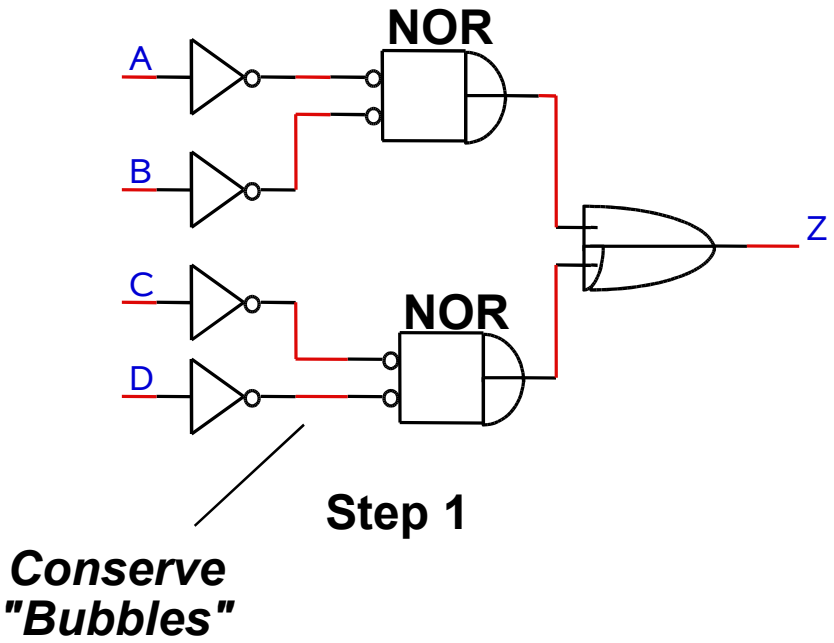
It is possible to convert from networks with ANDs and ORs to networks with NANDs and NORs by introducing the appropriate inversions ("bubbles")

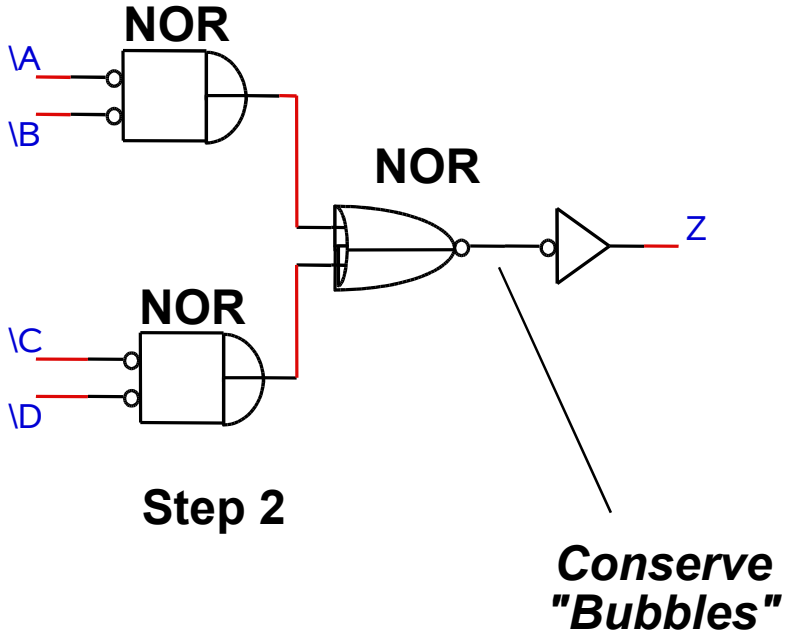
To preserve logic levels, each introduced "bubble" must be matched with a corresponding "bubble"

Map AND/OR network to NAND/NAND network

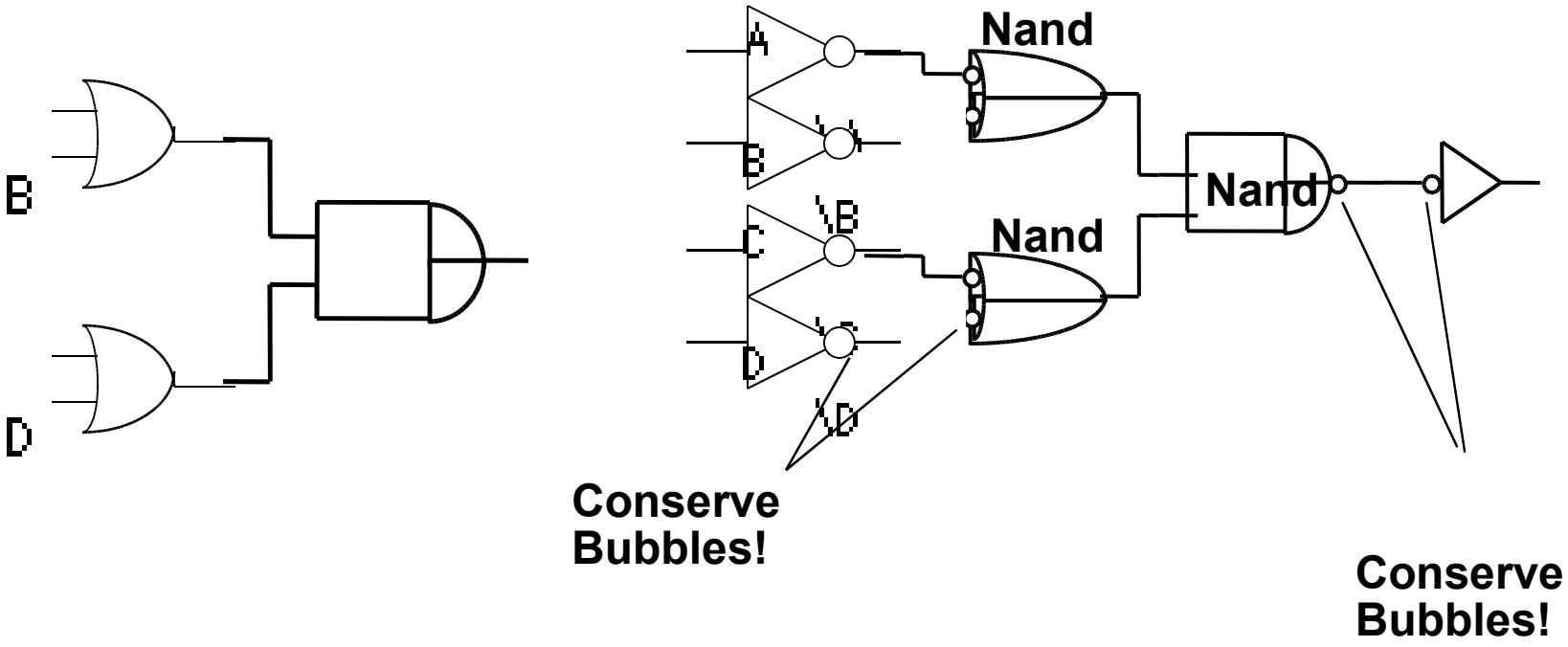


Map AND/OR Network to NOR/NOR Network





Map OR/AND Network to NAND/NAND



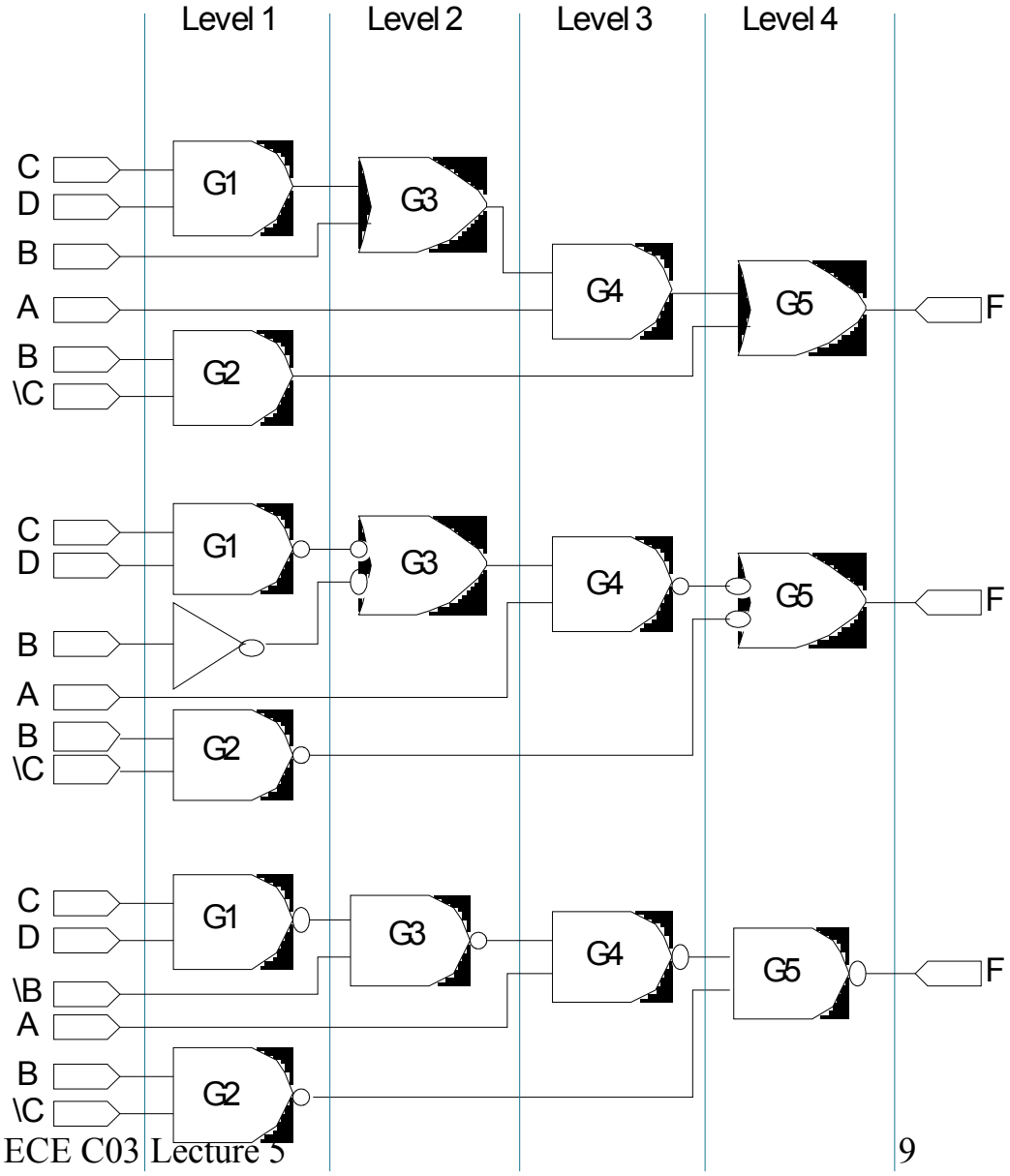
Multi-Level Logic: More than Two Levels

f = A (B + C D) + B C'

Original AND-OR Network

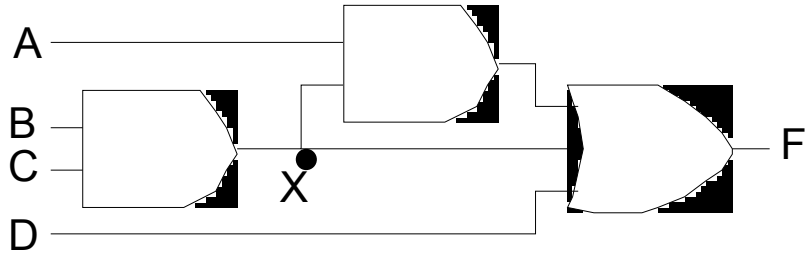
Introduction and Conservation of Bubbles

Redrawn in terms of conventional NAND Gates

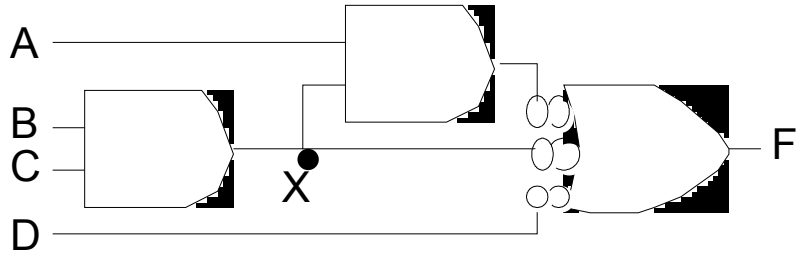


Multi-Level Logic: More than Two-Levels

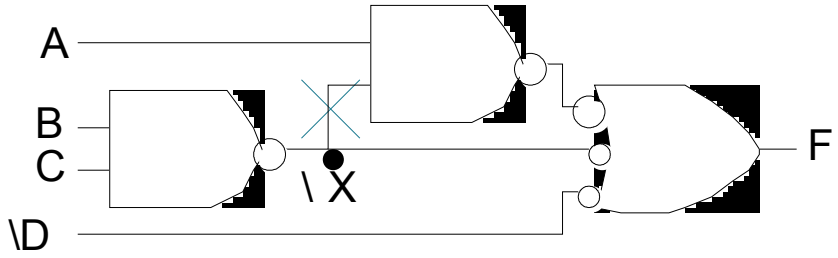
Conversion Example



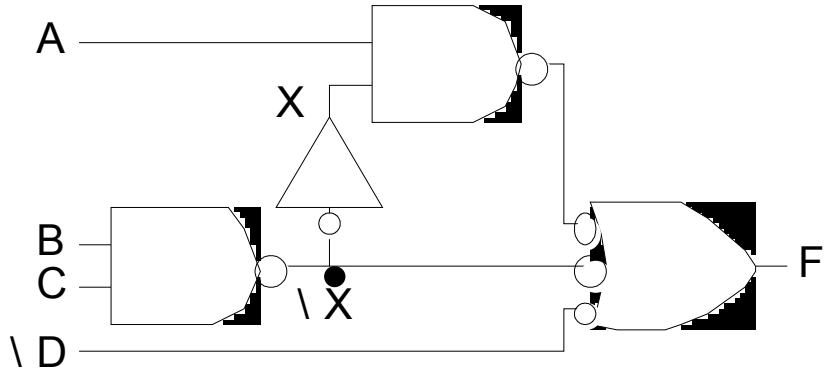
(a) **Original circuit**



(b) **Add double bubbles at inputs**



(c) **Distribute bubbles some mismatches**

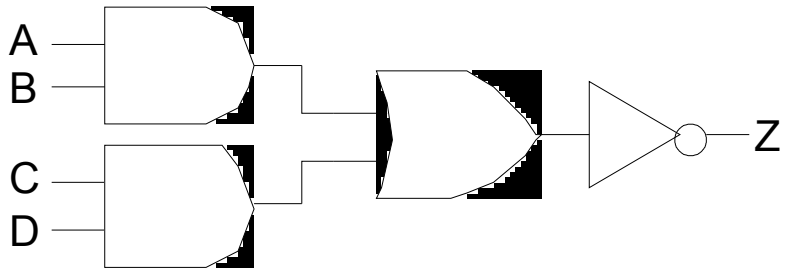


(d) **Insert inverters to fix mismatches**

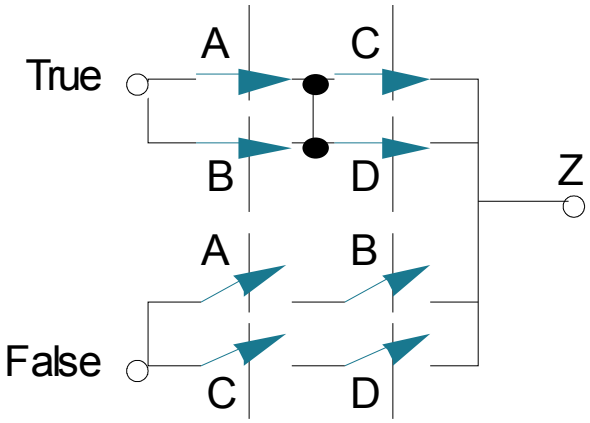
Multi-Level Logic: AND-OR-Invert Block

AOI Function: **Three stage logic— AND, OR, Invert**
Multiple gates "packaged" as a single circuit block

logical concept



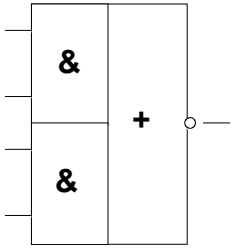
possible switch implementation



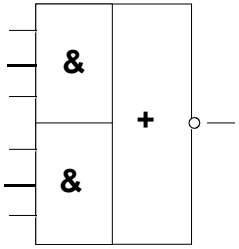
AND OR Invert

two-input two-stack

2x2 AOI Schematic Symbol



3x2 AOI Schematic Symbol



Multi-Level Logic: AND-OR-Invert

Example: XOR implementation

A xor B = A' B + A B'

= (?)'

AOI form

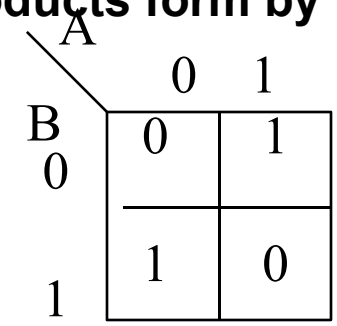
(A' B + A B')'

(A + B') (A' + B)

(A B + A' B')

General procedure to place in AOI form:

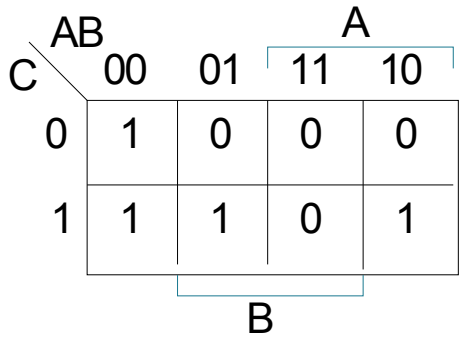
Compute the complement in Sum of Products form by circling the 0's in the K-map!



***f* = (A' B' + A B)'**

Multi-Level Logic: AND-OR-Invert

Example:



F' K-map

F = B C' + A C' + A B

F' = A' B' + A' C + B' C

Implemented by 2-input 3-stack AOI gate

F = (A + B) (A + C') (B + C')

F' = (B' + C) (A' + C) (A' + B')

Implemented by 2-input 3-stack OAI gate

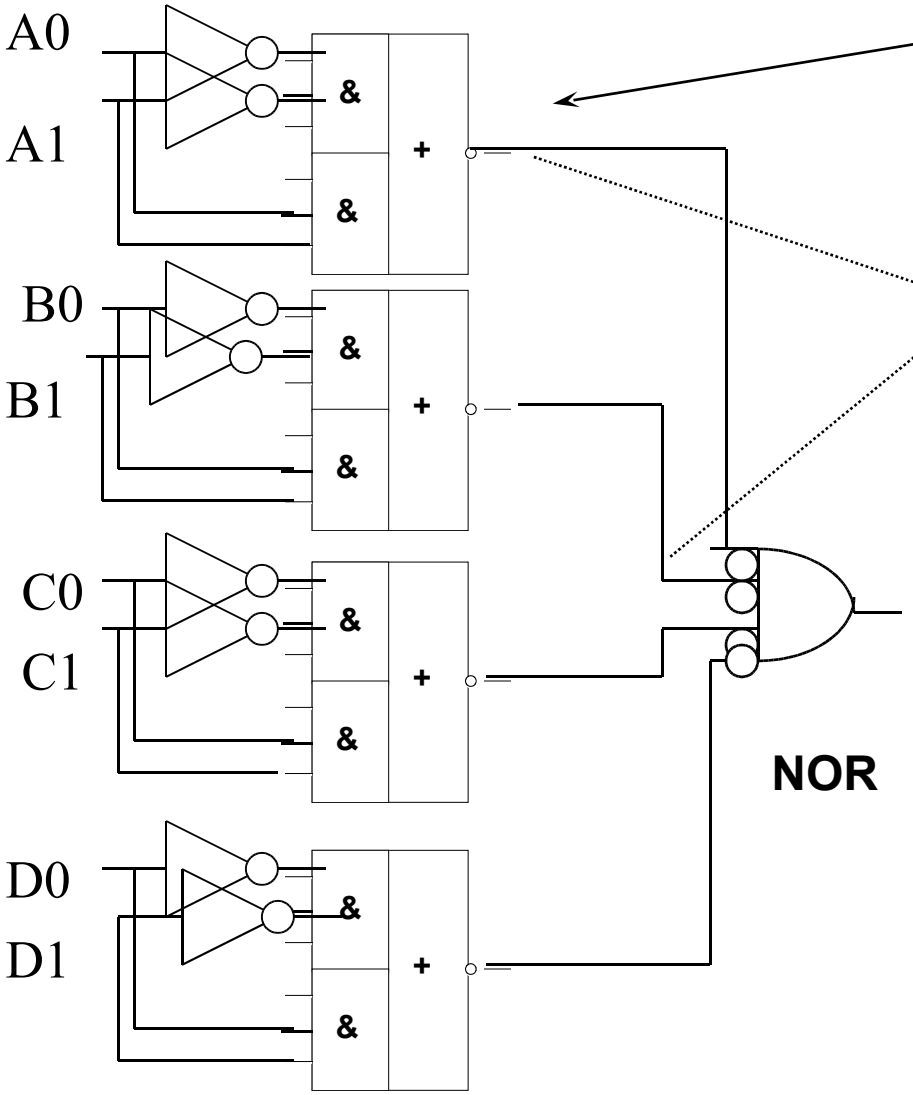
Example:

4-bit Equality Function

Z = (A0 B0 + A0' B0') (A1 B1 + A1' B1') (A2 B2 + A2' B2') (A3 B3 + A3' B3')

Each implemented in single 2x2 AOI gate

AOI Implementation of 4 bit Equality Checker



High if A0 ° B0, Low if A0 = B0

A = B active low

Conservation of bubbles

If all inputs are low
(asserted in negative logic)
then Ai = Bi, i=0,...,3
Output Z asserted

Multi-Level Logic: CAD Tools for Simplification

Multi-Level Optimization:

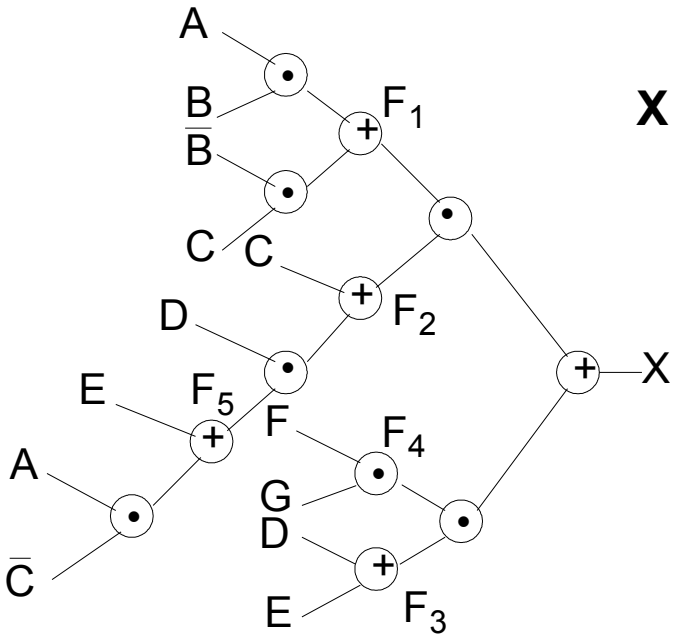
1. **Factor out common sublogic (reduce fan-in, increase gate levels), subject to timing constraints**

2. **Map factored form onto library of gates**

3. **Minimize number of literals (correlates with number of wires)**

Factored Form:

sum of products of sum of products . . .



X = (A B + B' C) (C + D (E + A C')) + (D + E)(F G)

Multi-Level Logic: CAD Tools for Simplification

Operations on Factored Forms:

- **Decompostion**
- **Extraction**
- **Factoring**
- **Substitution**
- **Collapsing**

Manipulate network by interactively issuing the appropriate instructions

There exists no algorithm that guarantees "optimal" multi-level network will be obtained

Decomposition

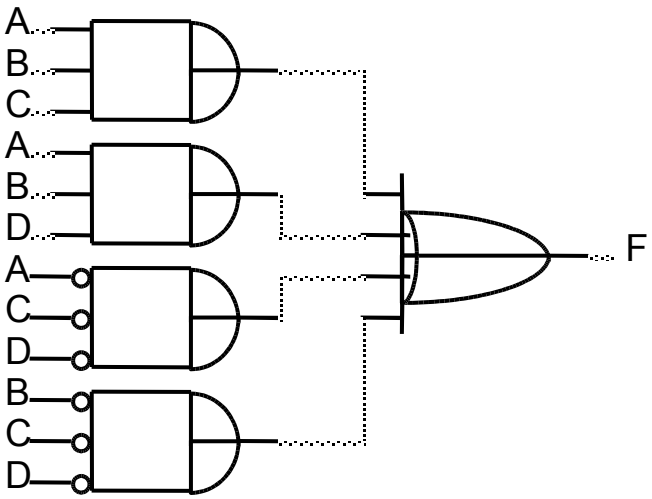
Take a single Boolean expression and replace with collection of new expressions:

F = A B C + A B D + A' C' D' + B' C' D' (12 literals)

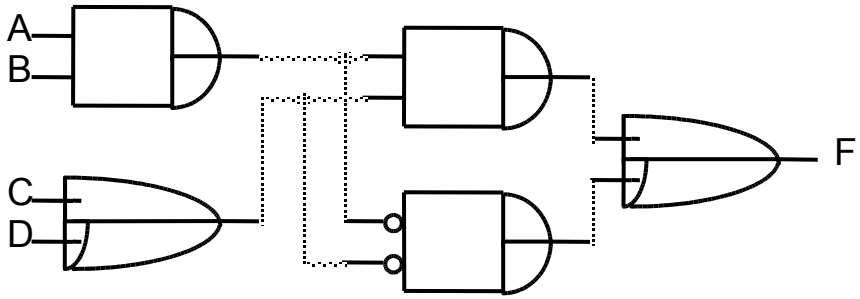
F rewritten as:

F = X Y + X' Y'
X = A B
Y = C + D

(4 literals)



Before Decomposition



After Decomposition

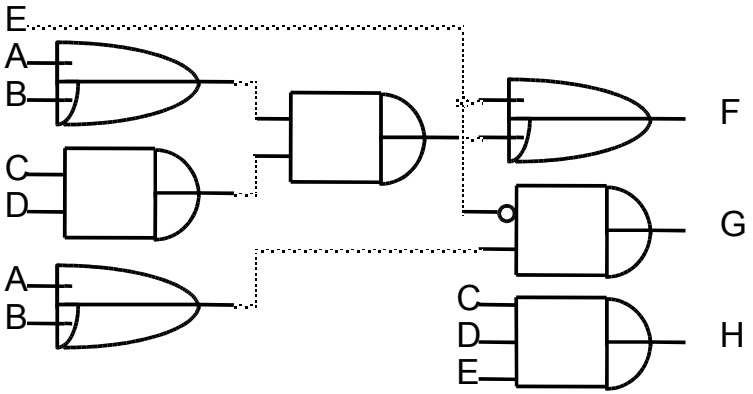
Extraction

Extraction: **common intermediate subfunctions are factored out**

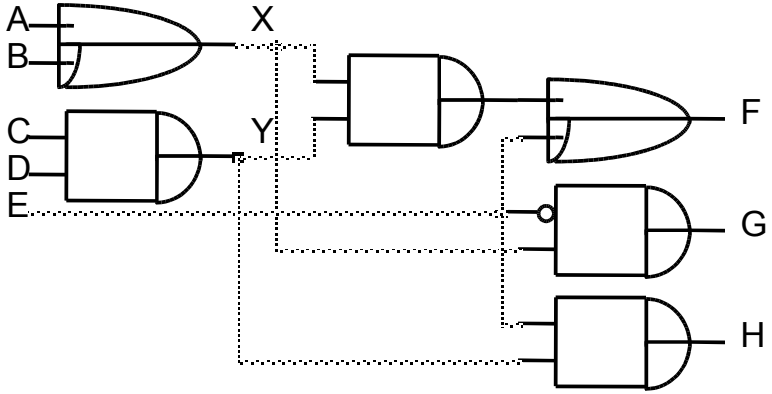
F = (A + B) C D + E **(11 literals)**
G = (A + B) E'
H = C D E

can be re-written as:

F = X Y + E **(7 literals)**
G = X E'
H = Y E
X = A + B **"Kernels": primary divisors**
Y = C D



Before Extraction



After Extraction

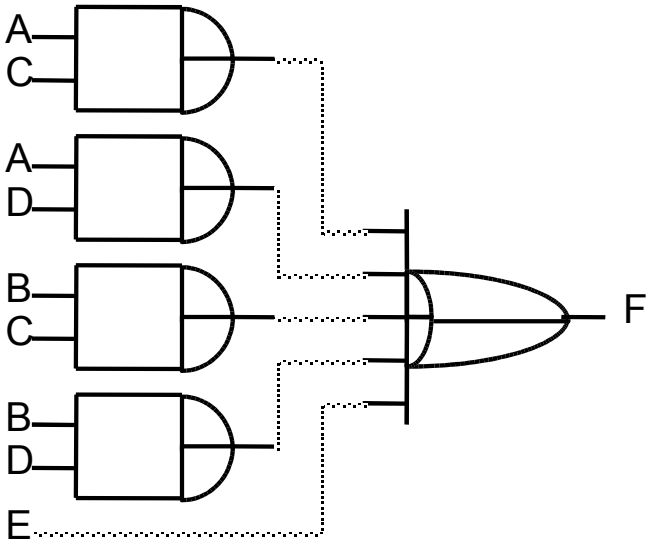
Factoring

Factoring: **expression in two level form re-expressed in multi-level form**

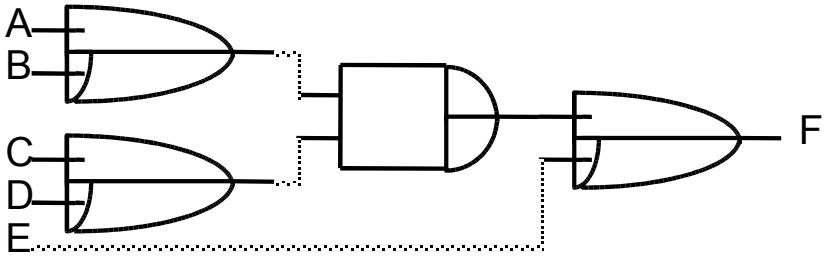
F = A C + A D + B C + B D + E (9 literals)

can be rewritten as:

F = (A + B) (C + D) + E (5 literals)



Before Factoring



After Factoring

Substitution

Substitution: **function G into function F, express F in terms of G**

F = A + B C **(5 literals)**
G = A + B

F rewritten in terms of G:

F = G (A + C) **(2 literals)**

Collapsing: **reverse of substitution; use to eliminate levels to meet timing constraints**

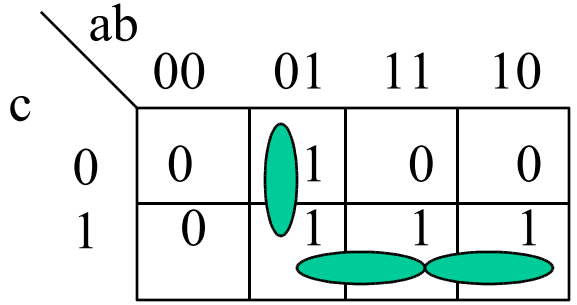
F = G (A + C)
= (A + B) (A + C)
= A A + A C + A B + B C
= A + B C ¦

Another Example of Resubstitution

- Given a set of Boolean functions
  - X = ac + ad + bc + bd + e
  - Y = a + b
  - This requires 11 literals
- The function Y is a divisor of X hence we can rewrite as
  - X = Y(c+d) + e
  - Y = a + b
  - This requires 6 literals

Node Simplification

- During multilevel logic synthesis, the sum of products form of each expression needs to be minimized
- Example:
  - F = af + bf + ag + cg + ade + bde + cde
- Conceptually you can think of this as using Karnaugh Maps, Quine McCluskey or ESPRESSO to simply these expressions using Boolean algebra
- Example:
  - F = a'b + ac + bc => a'b + bc



MIS CAD Tool Session

misII Session with the Full Adder

```
% misII
UC Berkeley, MIS Release #2.1 (compiled 3-Mar-89 at 5:32 PM)
misII> re full.adder
misII> p
    {co} = a b ci + a b ci' + a b' ci + a' b ci
    {sum} = a b ci + a b' ci' + a' b ci' + a' b' ci
misII> pf
   {co} = a b' ci + b (ci (a' + a) + a ci')
   {sum} = ci (a' b' + a b) + ci' (a b' + a' b)
misII> sim1 *
misII> p
    {co} = a b + a ci + b ci
    {sum} = a b ci + a b' ci' + a' b ci' + a' b' ci
misII> pf
   {co} = ci (b + a) + a b
   {sum} = ci (a' b' + a b) + ci' (a b' + a' b)
misII> gd *
misII> pf
   {co} = a [2] + b ci
   {sum} = a' [3]' + a [3]
   [2] = ci + b
   [3] = b' ci' + b ci
```

read eqntott equations

two level minimization

good decomposition

technology independent up to this point

MIS Script Session

```
misII> rlib msu.genlib
misII> map
misII> pf
   [361] = b' ci' + a'
   [328] = b'
   [329] = ci'
   {co} = [328]' [329]' + [361]'
   [3] = b ci' + b' ci
   {sum} = [3] a' + [3]' a
misII> pg
[361]      1890:physical 32.00
[328]      1310:physical 16.00
[329]      1310:physical 16.00
{co}       1890:physical 32.00
[3]        2310:physical 40.00
{sum}      2310:physical 40.00
misII> pat
 ... using library delay model
{sum}     : arrival=( 2.2  2.2)
{co}      : arrival=( 2.2  2.2)
[328]     : arrival=( 1.2  1.2)
[361]     : arrival=( 1.2  1.2)
[329]     : arrival=( 1.2  1.2)
[3]       : arrival=( 1.2  1.2)
ci        : arrival=( 0.0  0.0)
b         : arrival=( 0.0  0.0)
a         : arrival=( 0.0  0.0)
misII> quit
%
```

read library & perform technology mapping

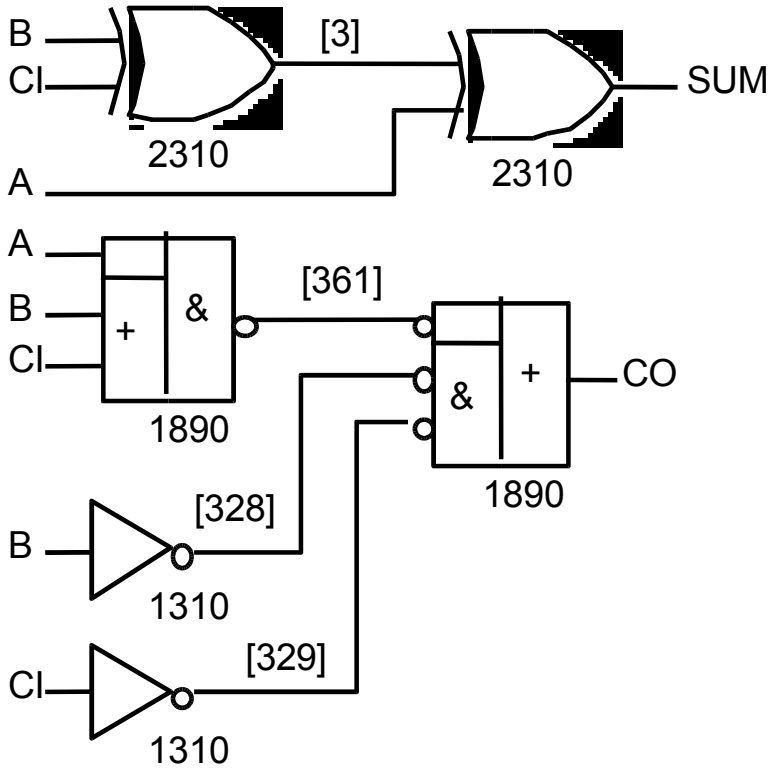
gates that implement the various nodes and their relative areas

timing simulation unit delay plus 0.2 time units per fan-out

Multi-Level Logic: CAD Tools for Simplification

misII and the MSU gate library

VLSI Standard Cells



NOTE: OR-AND-INVERT equivalent to INVERT-AND-OR

Number	Name	Function
1310	inv	A'
1120	nor2	(A+B)'
1130	nor3	(A+B+C)'
1140	nor4	(A+B+C+D)'
1220	nand2	(A•B)'
1230	nand3	(A•B•C)'
1240	nand4	(A•B•C•D)'
1660	and2/nand2	[A•B, (A•B)']
1670	and3/nand3	[A•B•C, (A•B•C)']
1680	and4/nand4	[A•B•C•D, (A•B•C•D)']
1760	or2/nor2	[A+B, (A+B)']
1770	or3/nor3	[A+B+C, (A+B+C)']
1780	or4	(A+B+C+D)
1870	aoi22	(A•B + C•D)'
1880	aoi21	(A + B•C)'
1860	oai22	[(A + B)(C + D)]'
1890	oai21	[A (B + C)]'
1970	ao22	A•B + D•E
1810	ao222	A•B + C•D + E•F
1910	ao2222	A•B + C•D + E•F + G•H
1930	ao33	A•B•C + D•E•F
2310	xor2	A•B' + A'•B
2350	xnor2	A•B + A'•B'

More Examples of MIS

mis with standard simplification script:

```
misII -f script -t pla <espresso truth table file>
```

Full Adder:

```
.model full.adder
.inputs a b ci
.outputs sum co
.names a b ci co sum
1--0 1
-1-0 1
--10 1
111- 1
.names a b ci co
11- 1
1-1 1
-11 1
.end
```

mis pla style outputs

input variables

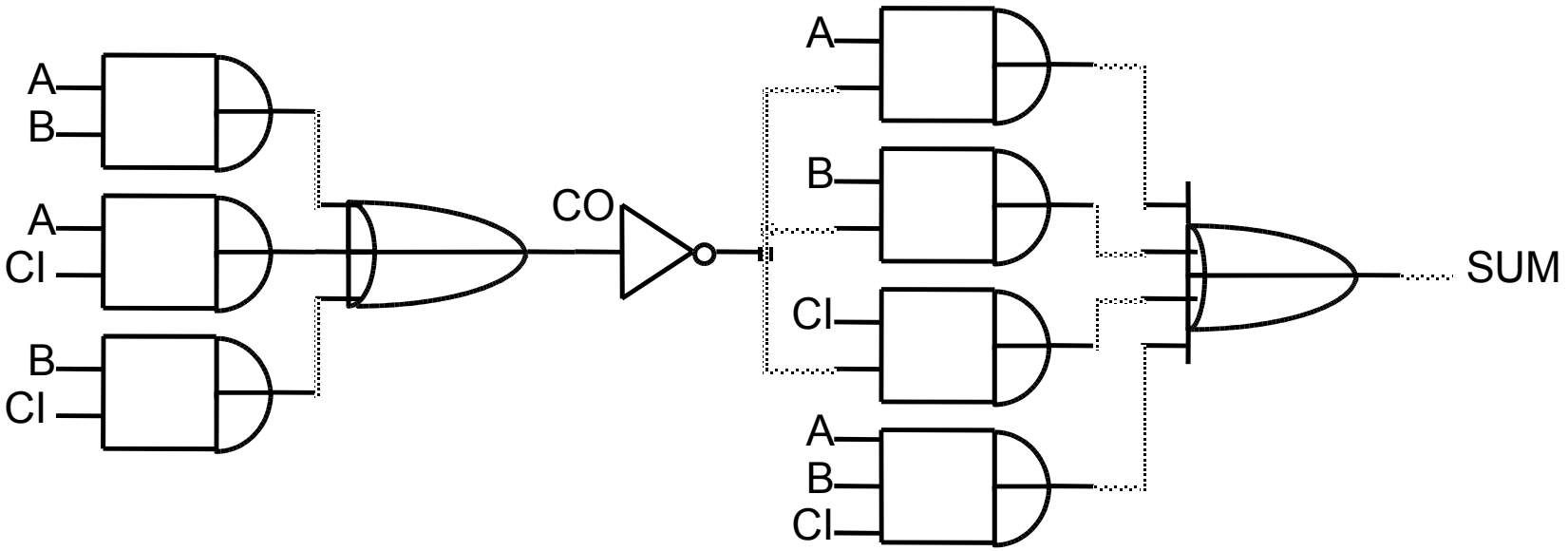
output variable

SUM = A CO' + B CO' + CI CO' + A B CI (9 literals)

CO = A B + A CI + B CI (6 literals)

Note that A xor B xor CI = A' B' CI + A B' CI' + A' B CI' + A B CI (12 literals!)

MIS Output of Full Adder



Multilevel Implementation of Full Adder: 5 Logic Levels!

MIS Implementation of Two-Bit Adder

```
.inputs a b c d
.outputs x y z
.names a c z [22] x
---1 1
11-- 1
-10- 1
.names a b c d x z [22] y
1---0-- 1
--1---1 1
-11-0-- 1
--110-- 1
---100- 1
.names a b c d z
-0-1 1
-1-0 1
0-10 1
.names a d z [22]
110 1
.end
```

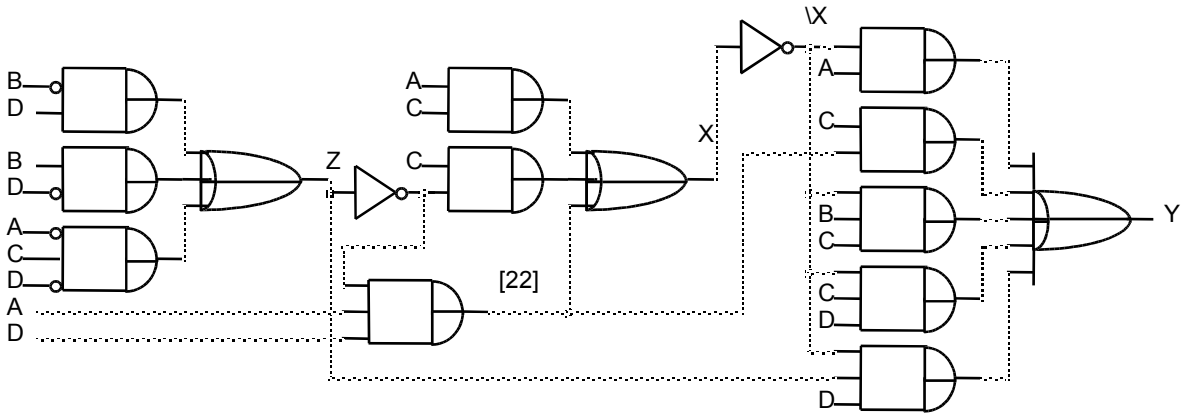
Mis Output

Z = B' D + B D' + A' C D'

[22] = A D Z'

X = [22] + A C + C Z'

Y = A X + C [22] + B C X' + C D X' + D X' Z'



8 logic levels!

MIS Implementation of BCD Incrementer

```
.model bcd.increment
.inputs a b c d
.outputs w x y z
.names a b c d z w
1---1 1
0111- 1
.names a b c w z x
01-0- 1
0-100 1
.names a c z y
-11 1
000 1
.names a b c d z
0--0 1
-000 1
.end
```

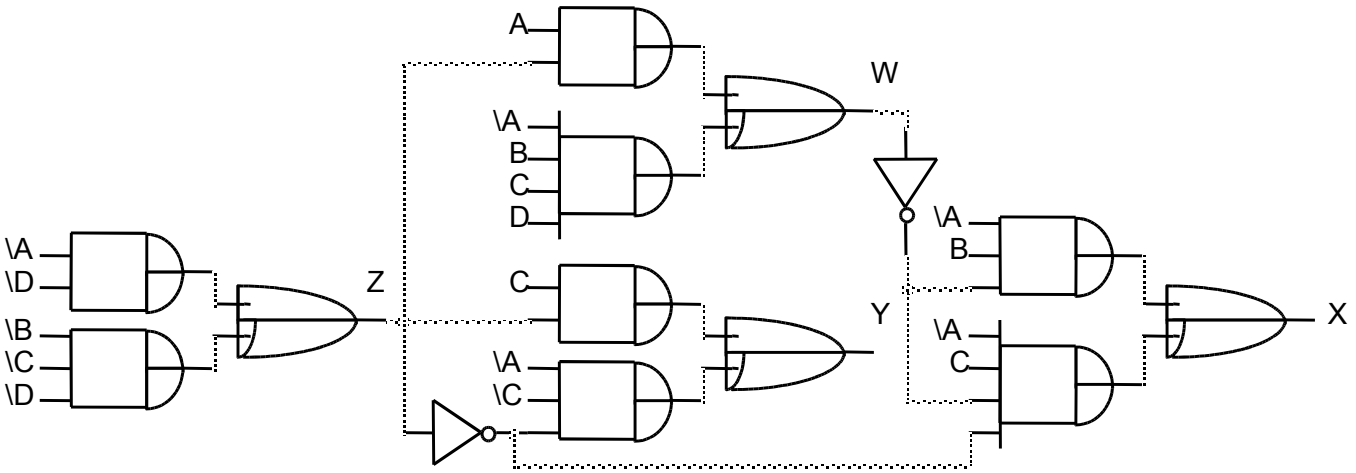
Mis Output

$$Z = A' D' + B' C' D'$$

$$Y = C Z + A' C' Z'$$

$$W = A Z + A' B C D$$

$$X = A' B W' + A' C W' Z'$$



Summary

- Mapping 2-level logic to NAND logic, NOR logic
- CAD Tools for Multilevel Logic: MIS Algorithm
- Factored forms
- Flattening
- NEXT LECTURE: Arithmetic Logic Circuits
- READING: Katz 5.2.1, 5.2.2, 5.2.4, 5.3, 4.6