Chapter # 3: Multi-Level Combinational Logic

Contemporary Logic Design

Randy H. Katz
University of California, Berkeley

June 1993

Chapter Overview

- ***Multi-Level Logic***

  Conversion to NAND-NAND and NOR-NOR Networks

  DeMorgan's Law and Pushing Bubbles

  AND-OR-Invert Building Blocks

  CAD Tools for Multi-Level Optimization

- ***Time Response in Combinational Networks***

  Gate Delays and Timing Waveforms

  Hazards/Glitches and How To Avoid Them

Boolean Algebra

Commutative Laws:

$a + b = b + a$ $\qquad$ $a \bullet b = b \bullet a$

Associative Laws:

$(a+b)+c = a+(b+c)$ $\qquad$ $(ab)c = a(bc)$

Identities:

$a + 0 = a$ $\qquad$ $a \bullet 0 = 0$

$a \bullet 1 = a$ $\qquad$ $a + 1 = 1$

Distributive Laws:

$a + (b \bullet c) = (a+b) \bullet (b+c)$ $\qquad$ $a \bullet (b+c) = (a \bullet b) + (a \bullet c)$

Boolean Algebra

Complement:

$a + a = 1$ $\qquad$ $a \bullet a = 0$

$a + a = a$ $\qquad$ $a \bullet a = a$

Theorems:

$a + ab = a$ $\qquad$ $ab + ab = b$

DeMorgan's Theorem:

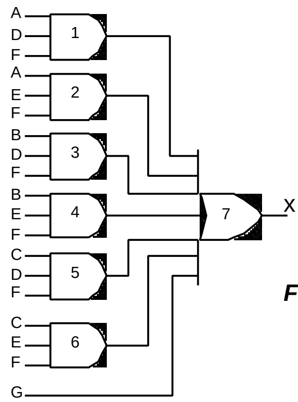
$a \bullet b = a + b$ $\qquad$ $a + b = a \bullet b$

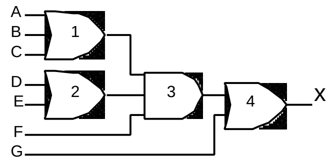
Multi-Level Logic: Advantages

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

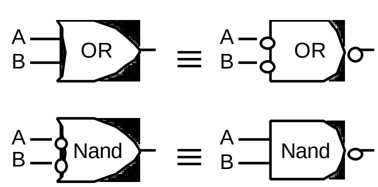
DeMorgan's Law: $(A + B)' = A' \bullet B'; \quad (A \bullet B)' = A' + B'$

$$A + B = (A' \bullet B')'; \quad (A \bullet B) = (A' + B')'$$

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

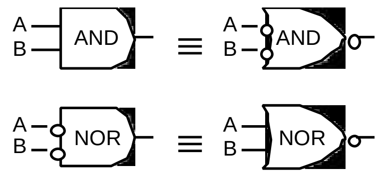
A	$\bar{A}$	B	$\bar{B}$	A + B	$\overline{\bar{A} \bullet \bar{B}}$	$\bar{A} + \bar{B}$	$\overline{A \bullet B}$
0	1	0	1	0	0	1	1
0	1	1	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	0	1	1	0	0



Mult-Level Logic: Conversion Between Forms

AND/NOR Equivalence

A	$\bar{A}$	B	$\bar{B}$	$A \bullet B$	$\overline{\bar{A} + \bar{B}}$	$\bar{A} \bullet \bar{B}$	$\overline{A + B}$
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

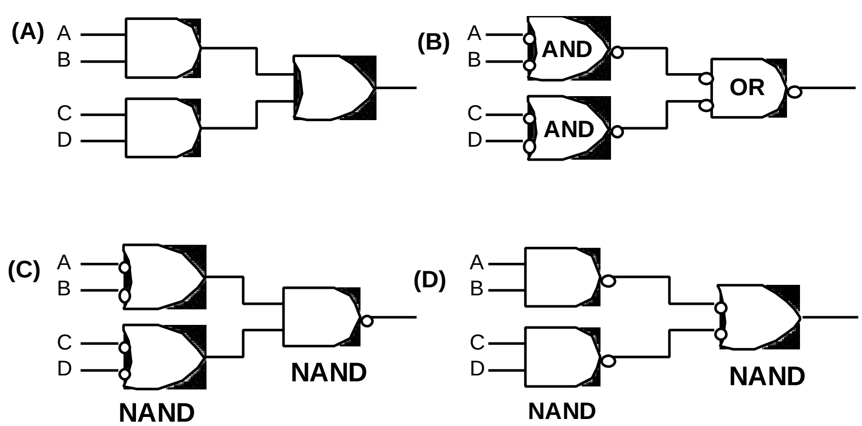


It is possible to convert from networks with ANDs and ORs to networks with NANDs and NORs by introducing the appropriate inversions ("bubbles")

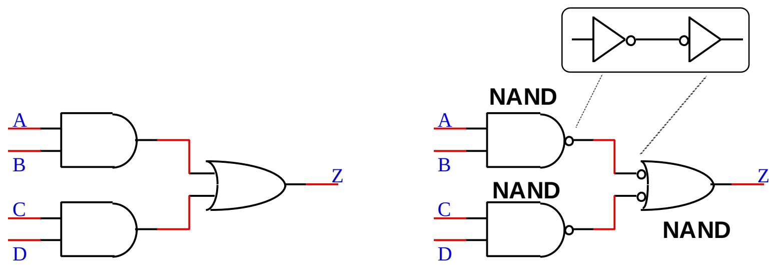
To preserve logic levels, each introduced "bubble" must be matched with a corresponding "bubble"

Multi-Level Logic: Conversion of Forms

Example: Map AND/OR network to NAND/NAND network



Example: Map AND/OR network to NAND/NAND network



$$Z = [(A \cdot B)' \ (C \cdot D)' \ ]'$$

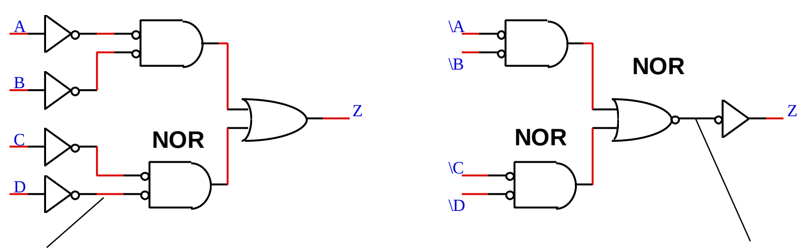
$$= [(A' + B')\ (C' + D')]'$$

$$= [(A' + B')' \bullet (C' + D')']$$

$$= (A \bullet B) + (C \bullet D) \ ¦$$

This is the easy conversion!

Example: Map AND/OR network to NOR/NOR network



Conserve "Bubbles"

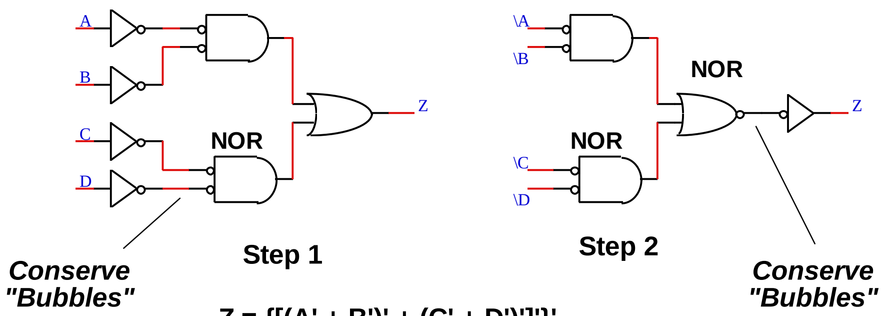
Step 1

Step 2

Conserve "Bubbles"

Z =

Example: Map AND/OR network to NOR/NOR network



$$Z = \{[(A' + B')' + (C' + D')']'\}'$$

$$= \{(A' + B') \cdot (C' + D')\}'$$

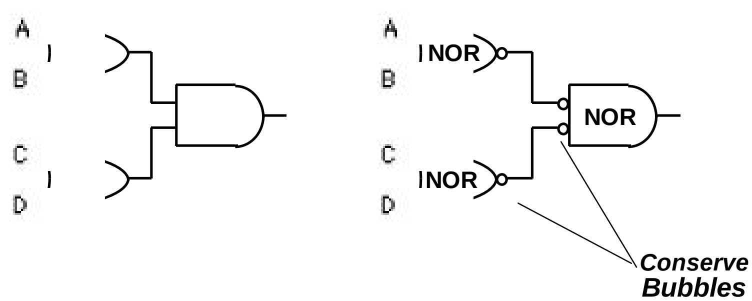
$$= (A' + B')' + (C' + D')'$$

$$= (A \cdot B) + (C \cdot D)$$

This is the hard conversion!

AND/OR to NAND/NAND more natural

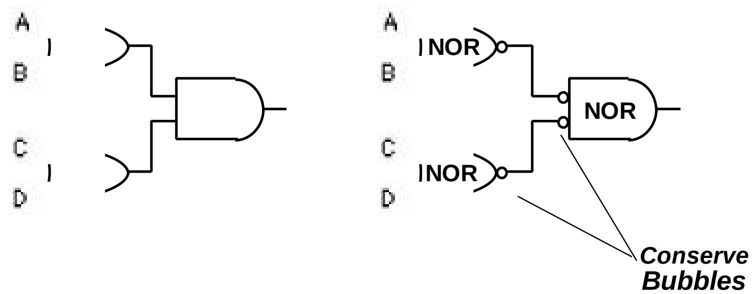
Example: Map OR/AND network to NOR/NOR network



Verify equivalence of the two forms

Z =

Example: Map OR/AND network to NOR/NOR network



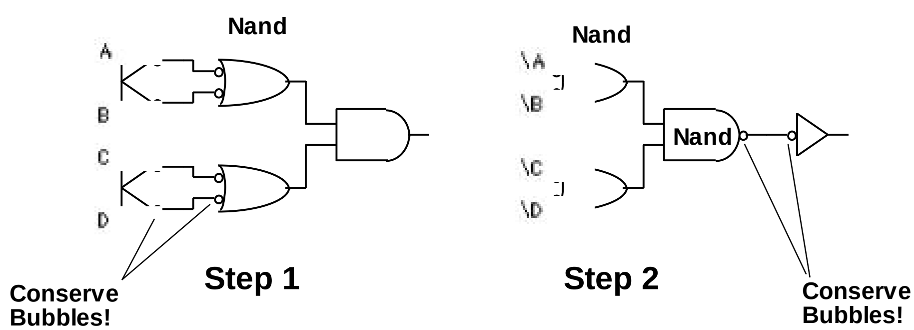
$$Z = [(A + B)' + (C + D)']'$$

$$= \{(A + B)'\}' \bullet \{(C + D)'\}'$$

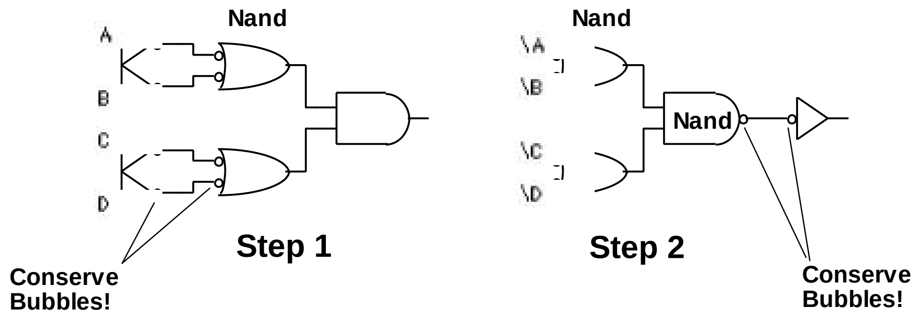
$$= (A + B) \bullet (C + D)$$

This is the easy conversion!

Example: Map OR/AND network to NAND/NAND network



Example: Map OR/AND network to NAND/NAND network



$$Z = \{[(A' \bullet B')' \bullet (C' \bullet D')']'\}'$$

$$= \{(A' \bullet B') + (C' \bullet D')\}'$$

$$= (A' \bullet B')' \bullet (C' \bullet D')'$$

$$= (A + B) \bullet (C + D)$$

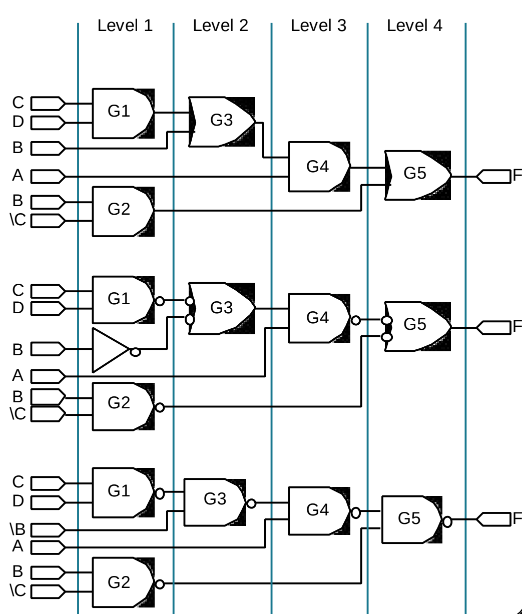
This is the hard conversion!

OR/AND to NOR/NOR more natural

Multi-Level Logic: More than Two Levels

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

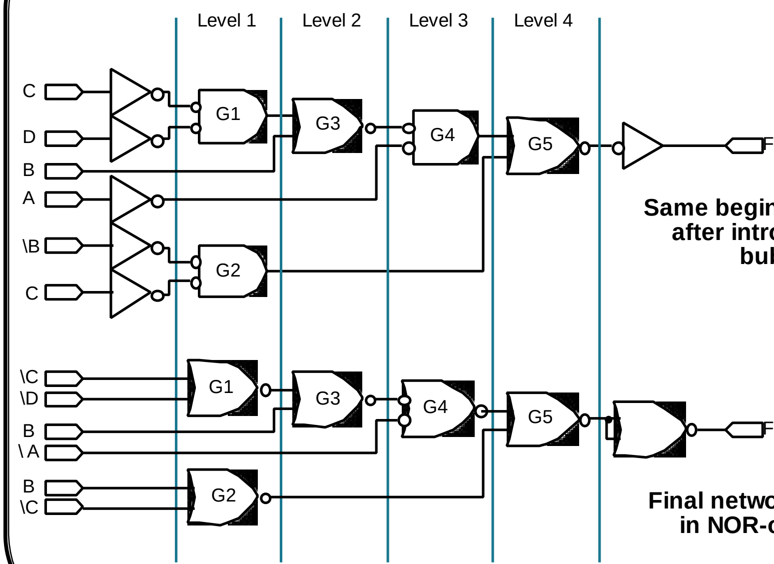
**Original
AND-OR Network**



**Introduction and
Conservation of Bubbles**

**Redrawn in terms
of conventional
NAND Gates**

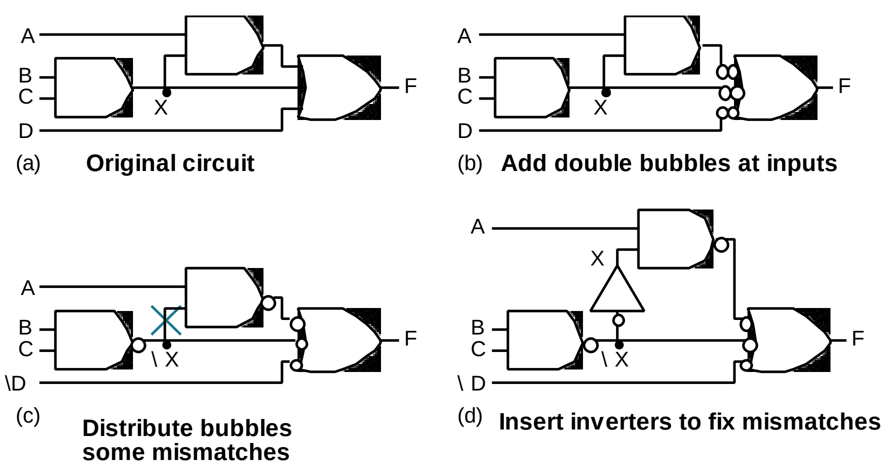
Multi-Level Logic: More than Two Levels



Same beginning network after introduction of bubbles

Final network, redrawn in NOR-only form

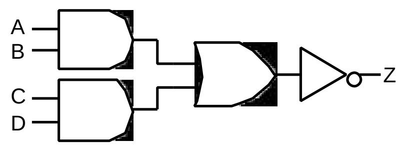
Conversion Example

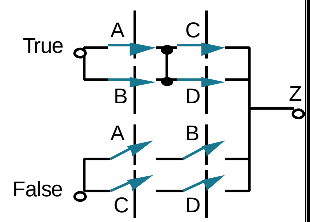


Multi-Level Logic: AND-OR-Invert Block (AOI)

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

logical concept



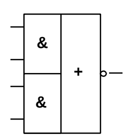


AND **OR** **Invert**

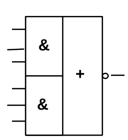
two-input two-stack

Multi-Level Logic: AND-OR-Invert Block (AOI)

2x2 AOI Schematic Symbol



3x2 AOI Schematic Symbol



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)'**

Example:

C \ AB	00	01	11	10
0	1	0	0	0
1	1	1	0	1

F K-map

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

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

2-input 3-stack AOI gate

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

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

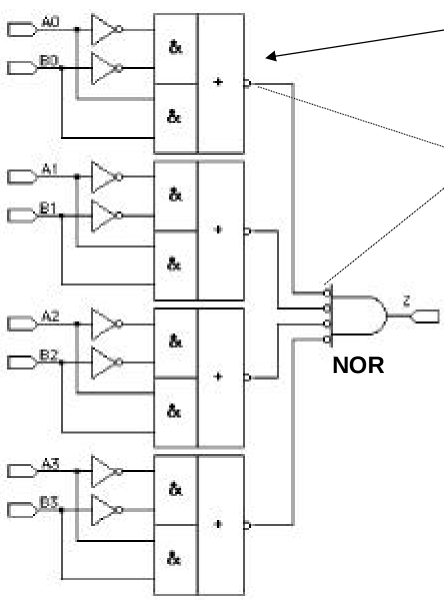
2-input 3-stack OAI gate

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

Example: AOI Implementation of a 4-Bit Equality Tester



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

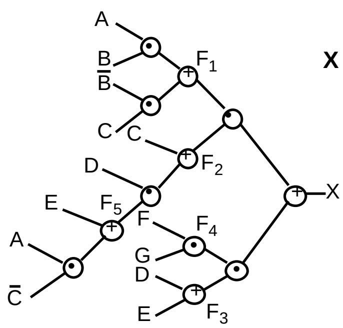
Multi-Level Logic: CAD Tools for Simplification

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

Time Response in Combinational Networks

- emphasis on timing behavior of circuits
- waveforms to visualize what is happening
- simulation to create these waveforms
- momentary change of signals at the outputs: ***hazards***
  - can be useful— pulse shaping circuits
  - can be a problem — ***glitches****:* incorrect circuit operation

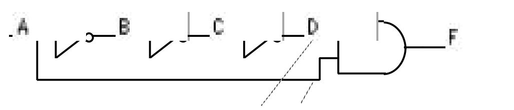
Terms:

gate delay - time for change at input to cause change at output
minimum delay vs. typical/nominal delay vs. maximum delay
careful designers design for the worst case!

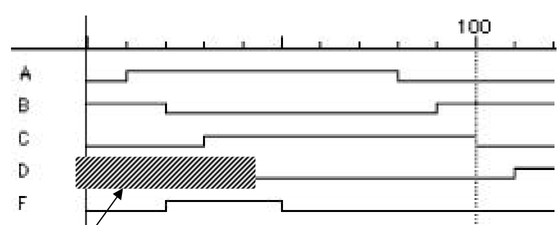
rise time - time for output to transition from low to high voltage

fall time - time for output to transition from high to low voltage

Pulse Shaping Circuit



A' • A = 0

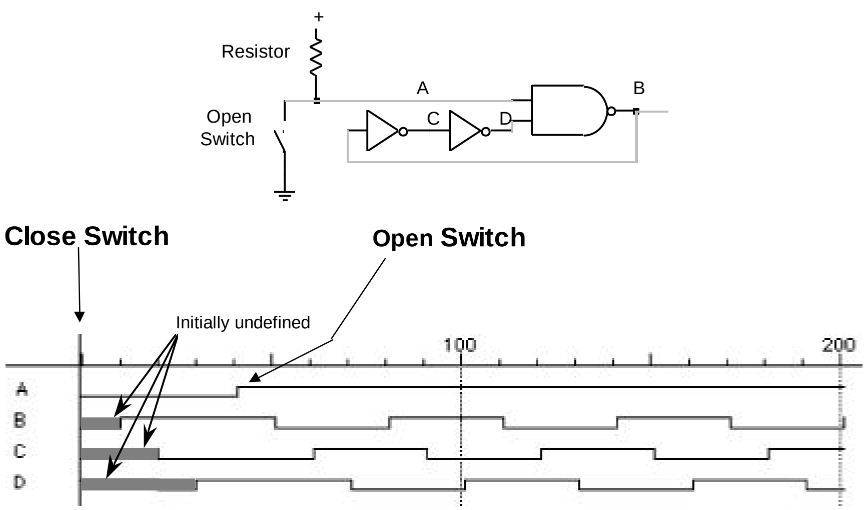


3 gate delays

D remains high for three gate delays after A changes from low to high

F is not always 0!

Another Pulse Shaping Circuit



Hazards/Glitches and How to Avoid Them

Unwanting switching at the outputs

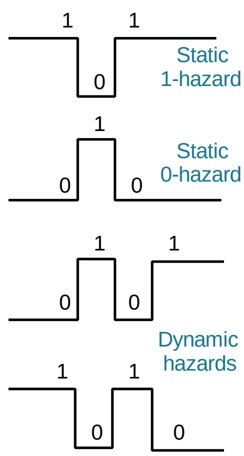
Occur because delay paths through the circuit experience different propagation delays

Danger if logic "makes a decision" while output is unstable OR hazard output controls an *asynchronous* input (these respond immediately to changes rather than waiting for a synchronizing signal called a *clock*)

Usual solutions:
- wait until signals are stable (by using a clock)
- never, never, never use circuits with asynchronous inputs
- design hazard-free circuits

Suggest that first two approaches be used, but we'll tell you about hazard-free design anyway!

Hazards/Glitches and How to Avoid Them



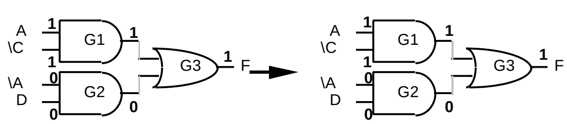
Input change causes output to go from **1** to **0** to **1**

Input change causes output to go from **0** to **1** to **0**

Input change causes a double change
from **0** to **1** to **0** to **1** OR
from **1** to **0** to **1** to **0**

Kinds of Hazards

Glitch Example

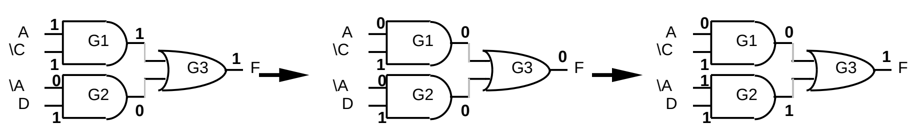


ABCD = 1100 ABCD = 1101

input change within product term

CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

F = A' D + A C'



ABCD = 1101 ABCD = 0101 ($\overline{A}$ is still 0) ABCD = 0101 ($\overline{A}$ is 1)

input change that spans product terms
output changes from 1 to 0 to 1

Glitch Example

<u>General Strategy</u>: add redundant terms

F = A' D + A C' becomes **A' D + A C' + C' D**

This eliminates 1-hazard? How about 0-hazard?

Re-express F in PoS form:

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

Glitch present!

Add term: (C' + D)

This expression is equivalent
to the hazard-free SoP form of F

CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

<u>*Glitch Example*</u>

Start with expression that is free of static 1-hazards

F = A C' + A' D + C' D

Work with complement:

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

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

= A C + A C D' + C D' + A' C D' + A' D'

= A C + C D' + A' D'

covers all the adjacent 0's in the K-map

free of static-1 and static-0 hazards!

Detecting Static Hazards in Multi-Level Circuits

Calculate transient output function

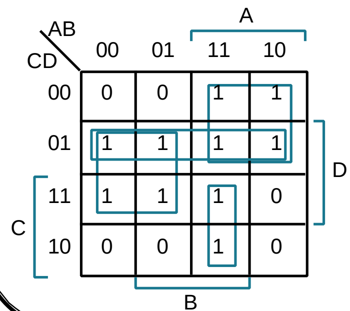
variables and complements are treated as independent variables

cannot use X + X' = 1 or X • X' = 0 for simplifications

Example:

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

F1 = A B C + A A' + A C' + A' D + C' D **2-level form**



ABCD: 1111 to 1110, covered by term ABC, so no 1-hazard present

ABCD: 1110 to 1100, term ABC goes low while term AC' goes high

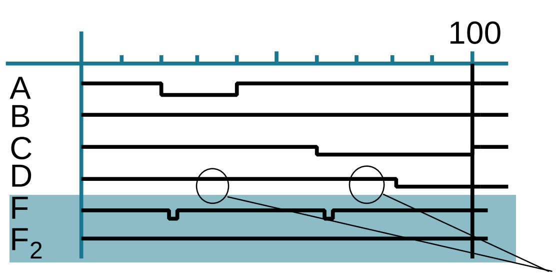
some static hazards are present!

Static 1-hazards

Solution:

Add redundant terms to insure all adjacent transitions are covered by terms

F2 = A C' + A' D + C' D + A B + B D



1's hazards in F corrected in F2

Static 0-Hazards

Similar to previous case, but work with the complement of F

If terms of the transient output function cover all 0 transitions, then no 0-hazards are present

$\overline{F}$ = [A B C + (A + D) (A' + C')]'

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

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

= A' D' + A B' C

+ B' C D'

CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	0
10	0	0	1	0

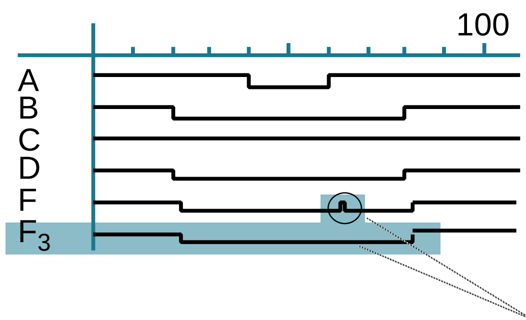
F = (A + D) (A' + B + C') (B + C' + D)

0-hazard on transition from 1010 to 0010

0-hazard free

equivalent to F2 on last slide

Static 0-Hazards



0-Hazard Corrected in F3

Designing Networks for Hazard-free operation

Simply place transient output function in a form that guarantees that all adjacent ones are covered by a term

no term of the transient output function contains both a variable and its complement

CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	0
10	0	0	1	0

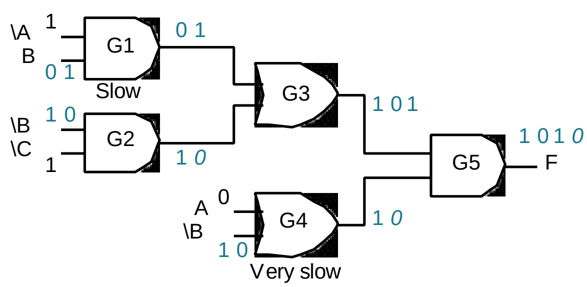
F(A,B,C,D) = • m(1,3,5,7,8,9,12,13,14,15)

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

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

(factored by distributive law, which does not introduce hazards since it does not depend on the complementarity laws for its validity)

Dynamic Hazards



Three different paths from B or B' to output

ABC = 000, F = 1 to ABC = 010, F = 0

different delays along the paths:
G1 slow, G4 very slow

Handling dynamic hazards very complex

Beyond our scope

Elements of a Data Sheet

A data sheet contains all the relevant documentation that you need to use the component:

1. Description of Function
2. A function/truth table
3. A logic schematic with labeled I/Os
4. Boolean expression of function in terms of I/Os
5. Alternative package pint-outs
6. Internal transistor shcematics
7. Operating specifications
8. Recommended operating conditions
9. Electrical characteristics.
10. Switching characteristics.

Operating Specifications: the absolute worst-case conditions under which the component can operate or be stored. Max input volt: 7v, Temp: 0 to 70 C.

Recommended Operating Conditions: the normal operating conditions for the supply voltage, input voltages, output currents, and temperature.

V_{HI}: min input volt recognized as a logic 1 (2v)

V_{IL}: max input volt recognized as a logic 0 (0.8v)

I_{OH}: max current gate can supply to maintain volt of logic 1 (-0.4 mA)

I_{OL}: min current gate can supply to maintain volt of logic 0 (8 mA)

Electrical Characteristics: voltages and currents that can be observed at the inputs and outputs.

V_{OH}: min output high volt (2.7v min, 3.4v typical)

V_{OL}: max output low volt (0.4v max, 0.25v typical)

I_{IH}: max current into input when high (20uA)

I_{IL}: max current into input when low (-0.4 mA)

The voltages determine the **noise margin**: 0.7v margin on logic 1, and 0.4v on logic 0.

Switching Characteristics: the typical and maximum gatdelays under specified test conditions.

t_{PLH}: delay from low to high (9ns typical, 15ns worst)

t_{PHL}: delay from high to low (10ns typical, 15ns worst)

Fan-Out: a given output can drive only a finite number of inputs before the output signal levels become degraded and are no longer recognized as good logic 1/0s.

To determine the **fan-out** examine the **I_{OH}** of the driving gate. This value must exceed the sum of the **I_{IH}** values of the inputs that the gate is driving.

Similarly, the **I_{OL}** of the gate must exceed the sum of the **I_{IL}** values of the inputs to which it is connected.

Example: $I_{IH:}$ 20uA, I_{OH}: -0.4mA

I_{IL}: -0.4mA, I_{OL}: 8mA

It can drive **20** gates to logic **1** and to logic **0**.

Technology Metrics

There are differences in the underlying technologies that may make one technology more attractive than another. The main technology metrics are:

1. Gate Delay: the time delay between the changes. Om general, bipolar techs are faster than MOS (ECL the fastest).

2. Degree of Integration: the area required to implement a given function in the underlying tech. MOS pack much more densely than bipolar.

SSI: up to 10 gates
MSI: up to 100 gates (not important)
LSI / VLSI: up to 1000 gates (MOS has advantage)

3. Power Dissipation: the power consumption and heat generated that must be dissipated.

- Bipolar generate more heat and consume more power
- ECL consume the most power
- MOS, especially CMOS, consume very little power

4. Noise Margin: the max volt that can be added to or subtracted from the login voltages and still have the ckt interpret the voltage as the correct logic value.

- Modern TTL / CMOS have good noise margins
- ECL has tighter noise margin

5. Component Cost: TTL, MOS, ECL