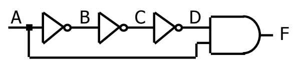
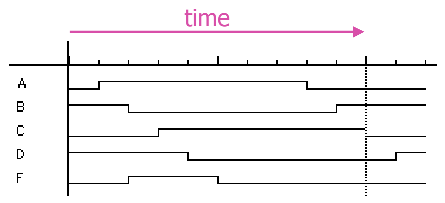
Overview

- Last lecture
  - Verilog
- Today
  - Timing diagrams
  - Multilevel logic
    - Multilevel NAND/NOR conversion
    - AOI and OAI gates
  - Hazards

Timing diagrams

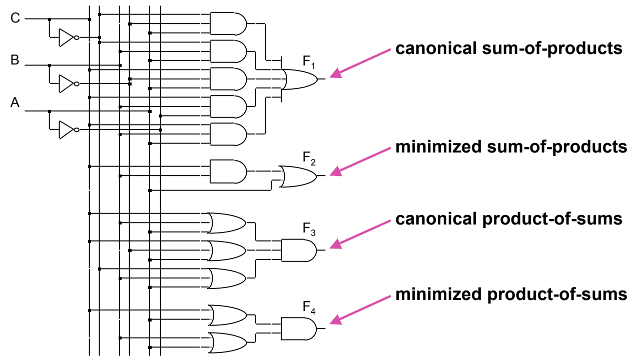
- "Sideways" truth tables
  - Show time-response of circuits
    - Real gates have real delays
  - Example: A' • A = 0
    - Delays cause transient F=1





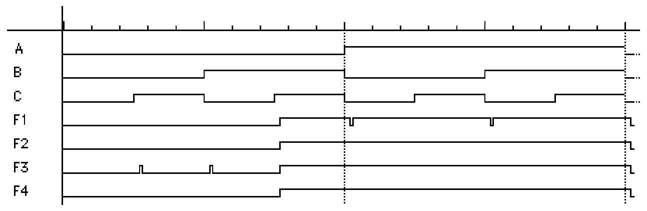
Some texts call timing diagrams "waveforms"

Example: F=A+BC in 2-level logic


C
B
A
F_1
F_2
F_3
F_4
canonical sum-of-products
minimized sum-of-products
canonical product-of-sums
minimized product-of-sums

Timing diagram for F = A + BC

- Time waveforms for F_1 – F_4 are identical
  - Except for timing hazards (glitches)
  - More on this shortly...



Multilevel logic

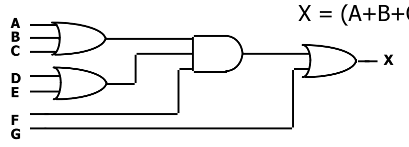
- Basic idea: Simplify logic using >2 gate levels
  - Time–space (speed versus gate count) tradeoff
- Two-level logic *usually*
  - Has smaller delays (faster circuits)
    - But more gates and more wires (more circuit area)
    - Sometimes has large fan-ins (slow)
  - Easier to eliminate hazards
- Multilevel logic *usually*
  - Has less gates (smaller circuits)
    - But can be slower (more gate delays)
  - Harder to eliminate hazards

Multilevel logic example

- Function X
  - SOP: X = ADF + AEF + BDF + BEF + CDF + CEF + G
    - X is minimized!
    - Six 3-input ANDs; one 7-input OR; 25 wires
  - Multilevel: X = (A+B+C)(D+E)F + G
    - Factored form
    - One 3-input OR, two 2-input OR's, one 3-input AND; 10 wires

3-level circuit

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

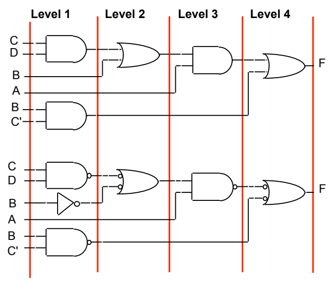


Multilevel NAND/NAND conversion

F = A(B+CD) + BC'

original AND-OR network

introduce bubbles (conserve inversions)

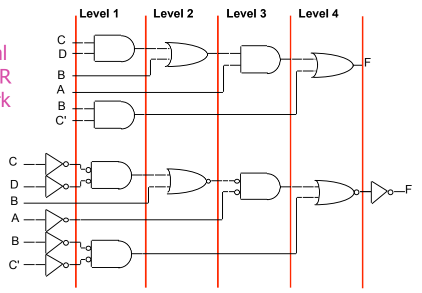


Multilevel NOR/NOR conversion

F = A(B+CD) + BC'

original AND-OR network

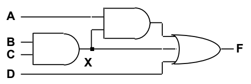
introduce bubbles (conserve inversions)



Generic multilevel conversion

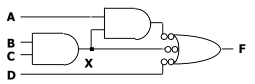
F = ABC + BC + D = AX + X + D

(a)



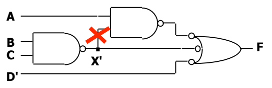
original circuit

(b)



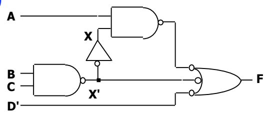
add double bubbles at inputs

(c)



distribute bubbles
some mismatches

(d)

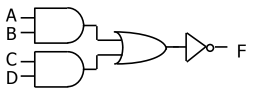


insert inverters to fix mismatches

AND-OR-Invert and OR-AND-Invert blocks

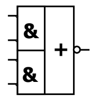
- AOI and OAI are dense 3-level building blocks
  - 2x2 AOI uses only 8 transistors
  - Minimal delay

logical concept

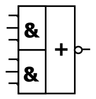


AND OR Invert

2x2 AOI symbol



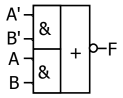
3x2 AOI symbol



Using AOI and OAI blocks

- Approach
  - Start by finding F'
    - ➩ If using AOI, find F' in minimized SOP form
    - ➩ If using OAI, find F' in minimized POS form
  - Form AOI as (F')'
- Example: F = A'B + AB'
  - SOP form: F' = A'B' + AB
  - AOI form: F = (A'B' + AB)'

		A
	0	1
B	1	0

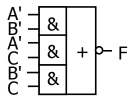


Example: AOI and OAI

			A	A
	0	1	1	1
C	0	0	1	0
		B	B	

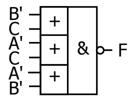
AOI form – Use SOP

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



OAI form – Use POS

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



Issues with multilevel design

- No global definition of "optimal" multilevel circuit
  - Optimality depends on user-defined goals
  - Synthesize an implementation that meets design goals
- Synthesis requires CAD-tool help
  - No simple hand methods like K-maps
  - CAD tools manipulate Boolean expressions
    - ➪ Factoring, decomposition, etc.
  - Covered in more detail in CSE467

Multilevel logic summary

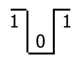
- Advantages over 2-level logic
  - Smaller circuits
  - Reduced fan-in
  - Less wires
- Disadvantages w.r.t 2-level logic
  - More difficult design
  - Less powerful optimizing tools
  - Dynamic hazards
- What you should know for CSE370
  - The basic multilevel idea
  - Multilevel NAND/NAND and NOR/NOR conversion
  - AOI gates

Hazards/glitches

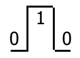
- Hazards/glitches: Undesired output switching
  - Occurs when different pathways have different delays
  - Wastes power; causes circuit noise
  - Dangerous if logic makes a decision while output is unstable
  - Dangerous if using asynchronous circuits
- Solutions
  - Design hazard-free circuits
    - ➪ Difficult when logic is multilevel
  - Wait until signals are stable
  - Use synchronous circuits

Types of hazards

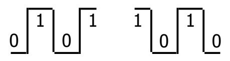
- Static 1-hazard
  - Output should stay logic 1
  - Gate delays cause brief glitch to logic 0



- Static 0-hazard
  - Output should stay logic 0
  - Gate delays cause brief glitch to logic 1

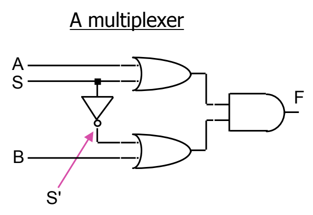


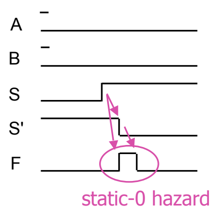
- Dynamic hazards
  - Output should toggle cleanly
  - Gate delays cause multiple transitions



Static hazards

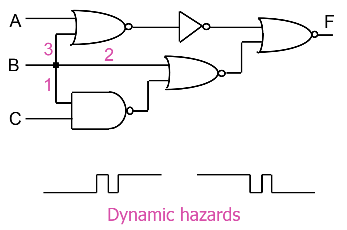
- Occur when a literal and its complement momentarily assume the same value
  - Through different paths with different delays
  - Causes an (ideally) static output to *glitch*

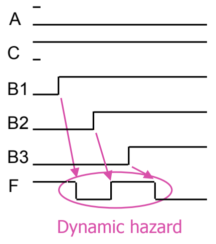




Dynamic hazards

- Occur when a literal assumes multiple values
  - Through different paths with different delays
  - Causes an output to toggle multiple times

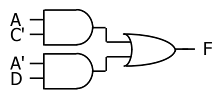


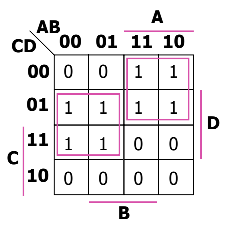


Eliminating static hazards

- In 2-level logic circuits
  - Assuming single-bit changes
- Key idea: Glitches happen when a changing input spans separate k-map encirclements
  - Example: 1101 to 0101 change can cause a static-1 glitch

F = AC' + A'D



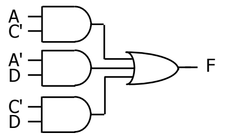


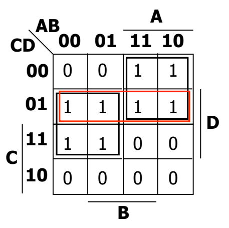
CD \ AB	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

Eliminating static hazards (con't)

- Solution: Add redundant k-map encirclements
  - Ensure that all single-bit changes are covered
  - First eliminate static-1 hazards: Use SOP form

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





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

- *Still need to eliminate static-0 hazards: Use POS form*

Summary of hazards

- We can eliminate static hazards in 2-level logic
  - For single-bit changes
  - Eliminating static hazards also eliminates dynamic hazards
- Hazard are a difficult problem
  - Multiple-bit changes in 2-level logic are hard
  - Static hazards in multilevel logic are harder
  - Dynamic hazards in multilevel logic are harder yet
- CAD tools and simulation/testing are indispensable
  - Test vectors probe a design for hazards