Lecture 13

- Logistics
  - HW4 up, due on Wednesday
- Last lecture
  - PLDs
- Today
  - Multilevel logic
  - Timing diagrams
  - Hazards

The "WHY" slide

- Multilevel logic
  - So far we talked about 2 level logic gate structure. But we can often express logic in many different ways and more levels than 2. Turns out expressing it in many levels could simplify circuit for the cost of slower processing time. If you want smaller circuits over speed, this is great to know.
- Timing diagram
  - Real gates have real delays and it is good to learn how to plot the information with respect to time.
- Hazards
  - Different delays can cause some output to get glitches. If these glitches are used in the next level of calculation, it could cause mis-calculation. It is good to know when that happens and how to fix it.

Multilevel logic

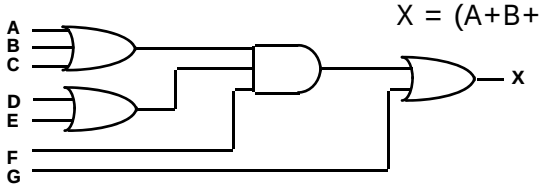
- Basic idea: Simplify logic using >2 gate levels
  - Time–space (speed versus gate count) tradeoff
    - Will talk about the speed issue with timing diagram
- Two-level logic *usually*
  - Has smaller delays (faster circuits)
  - more gates and more wires (more circuit area)
- Multilevel logic *usually*
  - Has fewer gates (smaller circuits)
  - more gate delays (slower circuits)

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; 26 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; 11 wires

3-level circuit

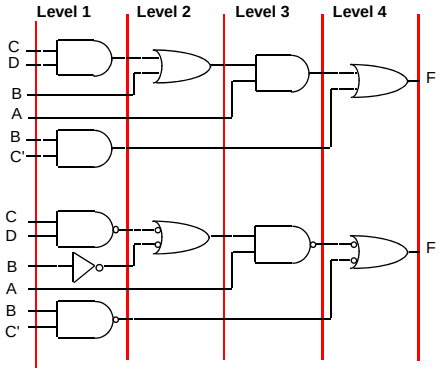
X = (A+B+C)(D+E)F + G



Multilevel NAND/NAND conversion

F = A(B+CD) + BC'

original AND-OR network

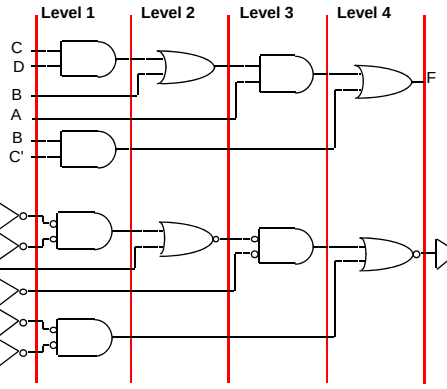


introduce bubbles (conserve inversions)

Multilevel NOR/NOR conversion

F = A(B+CD) + BC'

original AND-OR network

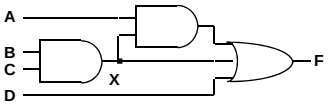


introduce bubbles (conserve inversions)

Generic multilevel conversion

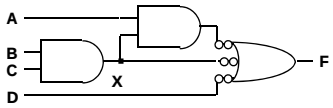
F = ABC + BC + D = AX + X + D

(a)



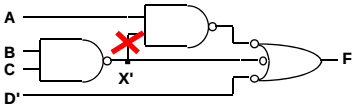
original circuit

(b)



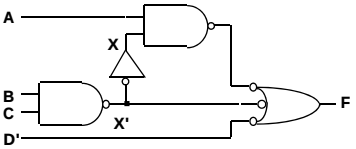
add double bubbles at inputs

(c)



distribute bubbles
some mismatches

(d)



insert inverters to fix mismatches

Issues with multilevel design

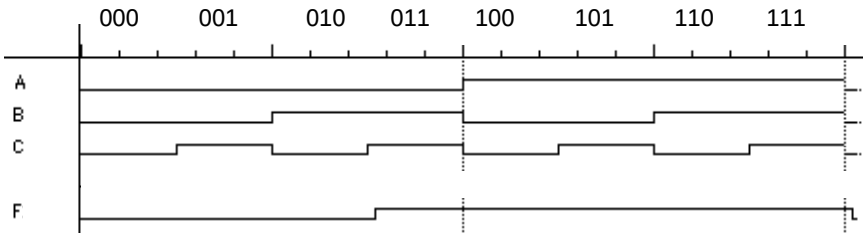
- No global definition of "optimal" multilevel circuit
  - Optimality depends on user-defined goals
- Synthesis requires CAD-tool help
  - No simple hand methods like K-maps
  - CAD tools manipulate Boolean expressions
  - Covered in more detail in CSE467

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
- What you should know for CSE370
  - The basic multilevel idea
  - Multilevel NAND/NAND and NOR/NOR conversion

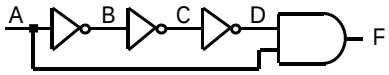
Timing diagram (aka waveforms)

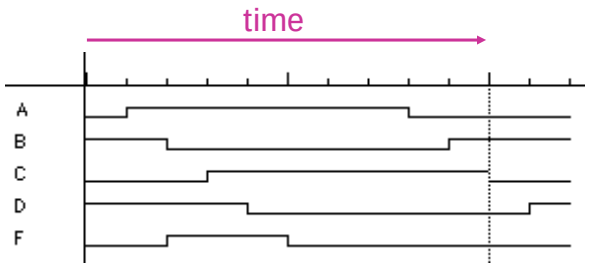
- Sideways truth tables
- Show time-response of circuits
- Example: F = A +BC



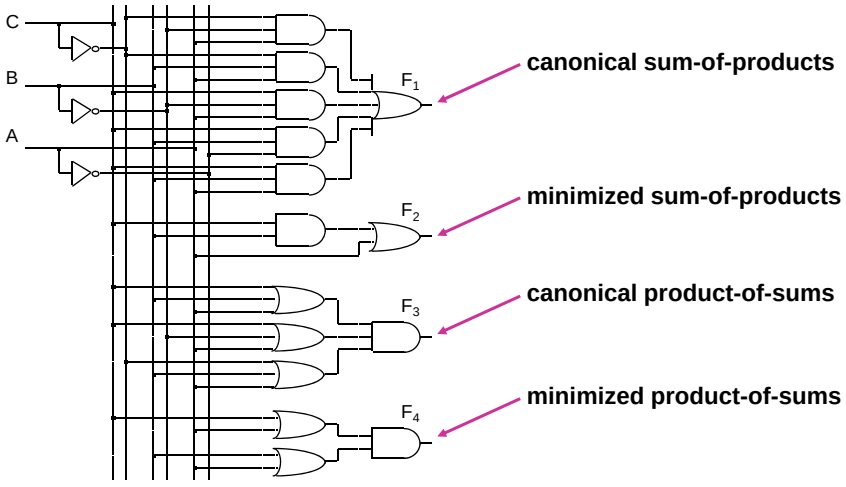
Timing diagrams

- Real gates have real delays
- Example: A' • A = 0
  - Delays cause transient F=1



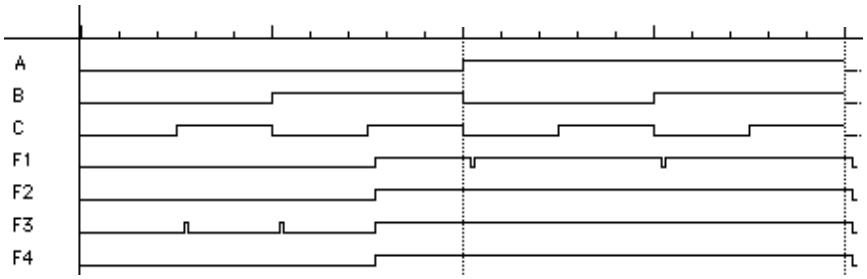


Example: F=A+BC in 2-level logic



Timing diagram for F = A + BC

- Time waveforms for F_1 – F_4 are identical
  - Except for timing hazards (glitches)



Hazards/glitches

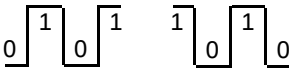
- Hazards/glitches: Undesired output switching
  - Occurs when different pathways have different delays
  - Wastes power; causes circuit noise
  - Dangerous if logic makes a decision while output is unstable
- Solutions
  - Design hazard-free circuits
    - Difficult when logic is multilevel
  - Wait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

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

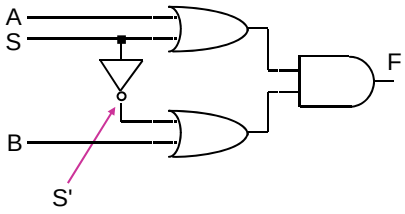
- Dynamic hazards
  - Output should toggle cleanly
  - Gate delays cause multiple transitions

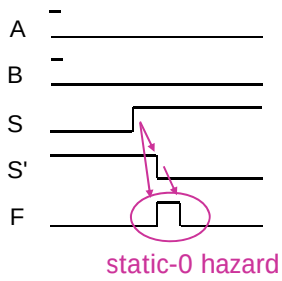


Static hazards

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

A multiplexer

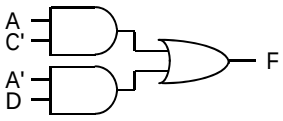


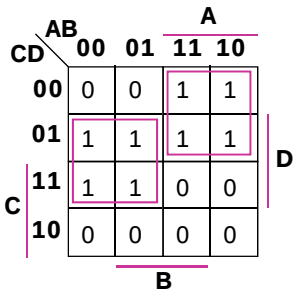


Eliminating static hazards (only in 2 level logic)

- 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

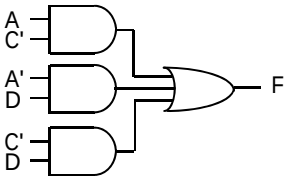


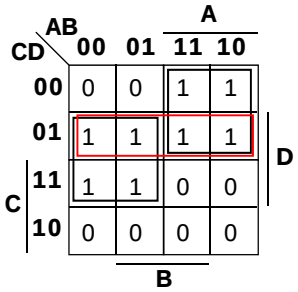


Eliminating static hazards (con't)

- 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





- *If 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
- Hazards 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