

Working with Combinational Logic

- Simplification
 - two-level simplification
 - exploiting don't cares
 - algorithm for simplification
- Logic realization
 - two-level logic and canonical forms realized with NANDs and NORs
 - multi-level logic, converting to NAND and NOR networks
- Time behavior of circuits
 - It takes time to compute Boolean logic
 - Transistors and wires have delays
 - Circuits don't instantaneously behave as their logic functions

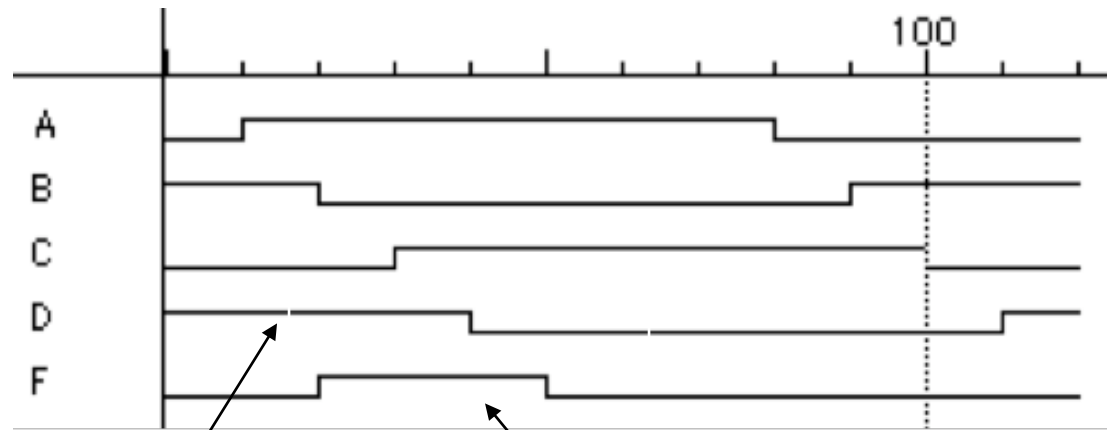
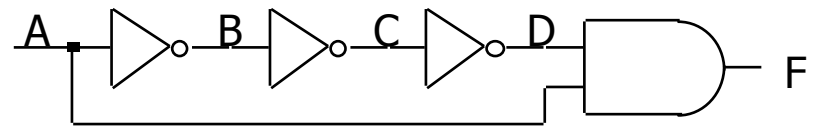
Time behavior of combinational networks

- Timing diagrams (or waveforms)
 - visualization of values carried on signal wires over time
 - useful in explaining sequences of events (changes in value)
- Simulation tools can be used to create these timing diagrams
 - input to the simulator includes gates and their connections
 - as well as input stimulus, that is, input signal waveforms that describe how the inputs change
- Some terms
 - gate or propagation delay — time needed for a change at an input to cause a change at an output
 - min delay – typical/nominal delay – max delay
 - careful designers design for the worst case (not always max delay)
 - 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 width — time that an output stays high or stays low between changes

Momentary changes in outputs

- Can be useful — pulse shaping circuits
- Can be a problem — incorrect circuit operation (glitches/hazards)
- Example: pulse shaping circuit

- $A' \cdot A = 0$
- delays matter
- in this case, pulse width = 3 when the propagation delay of inverters = 1

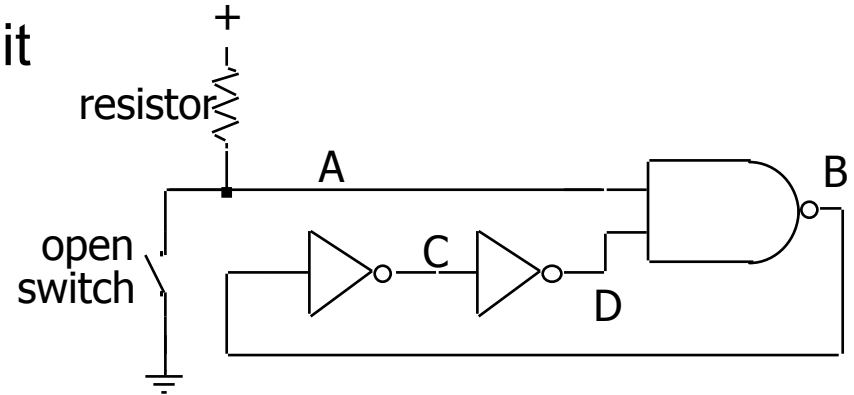


D remains high for three gate delays after A changes from low to high

F is not always 0 pulse 3 gate-delays wide

Oscillatory behavior

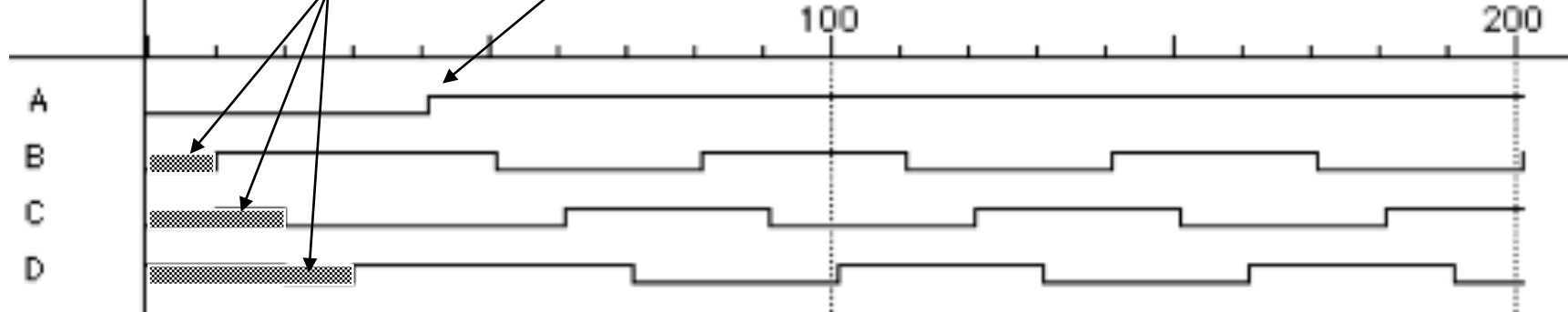
- Another pulse shaping circuit



close switch

initially
undefined

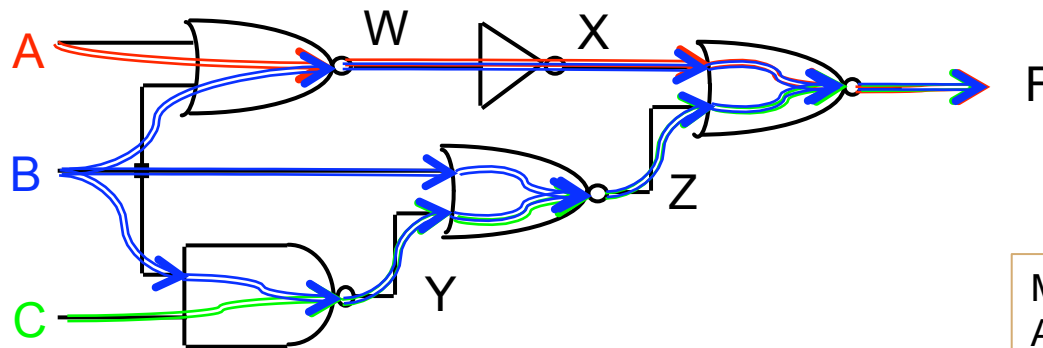
open switch



Computing circuit delays

- Consider every path from inputs to each output
 - add delay along path, max at gates
 - (max,+) algebra

A, B arrive at time 0, C at 4
 Max delay = $\max(3, 3, 2, 3, 4+3) = 7$



A, C arrive at time 0, B at 2
 Max delay = $\max(3, 2+3, 2+2, 2+3, 3) = 5$

A, B, C all arrive at time 0
 Assume delays all = 1
 A -> F: 3 gate delays
 B -> F: 3, 2, or 3 gate delays
 C -> F: 3 gate delays
 Max delay = $\max(3, 3, 2, 3, 3) = 3$

Make gate Y have delay of 2
 A, B, C all arrive at time 0
 A -> F: 3 gate delays
 B -> F: 3, 2, or 4 gate delays
 C -> F: 4 gate delays
 Max delay = $\max(3, 3, 2, 4, 4) = 4$

Make gate Y have delay of 2
 A, B arrive at time 0, C at 4
 Delay at W = $\max(A+1, B+1) = 1$
 Delay at X = $\max(W+1) = 2$
 Delay at Y = $\max(B+1, C+2) = 6$
 Delay at Z = $\max(B+1, Y+1) = 7$
 Delay at F = $\max(X+1, Z+1) = 8$

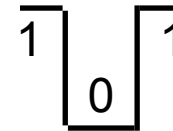
Hazards/glitches

- 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
 - Synchronous circuits (we'll come back to this concept later on)

Types of hazards

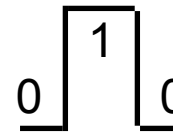
- Static 1-hazard

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



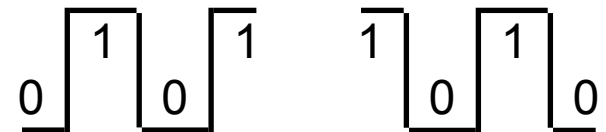
- Static 0-hazard

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



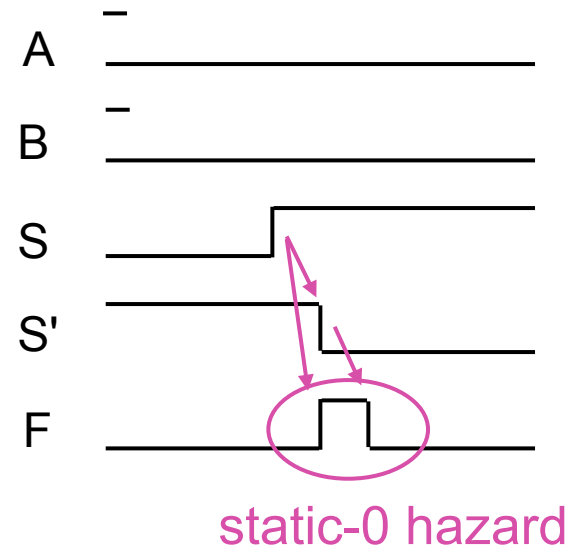
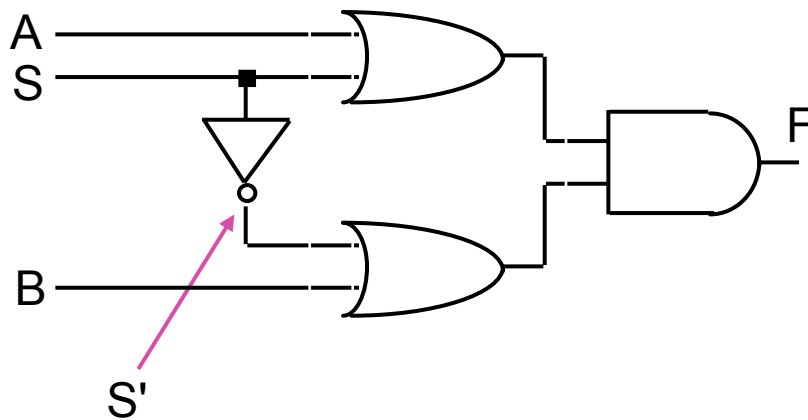
- Dynamic hazards

- Output should toggle cleanly
- Gate delays cause multiple transitions



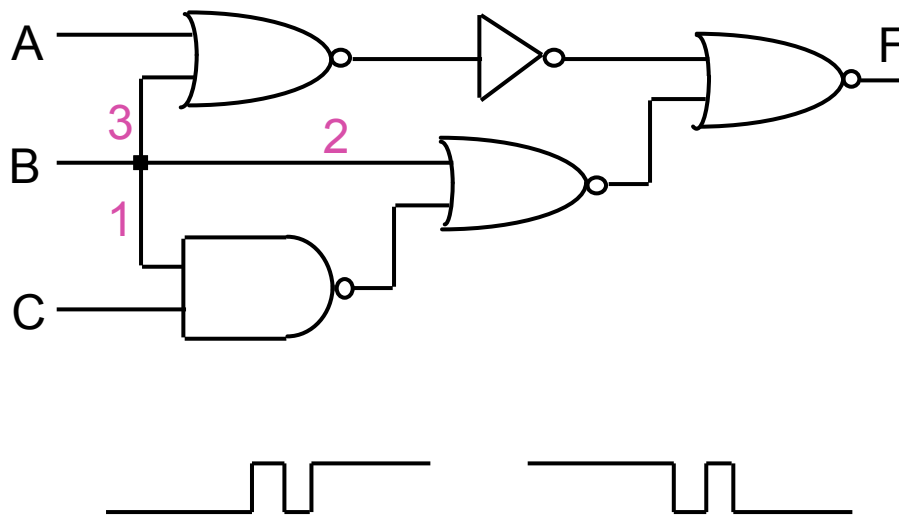
Static hazards

- 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*
 - Example: a multiplexer – a VERY common circuit
 - Width of hazard is equal to delay difference in paths

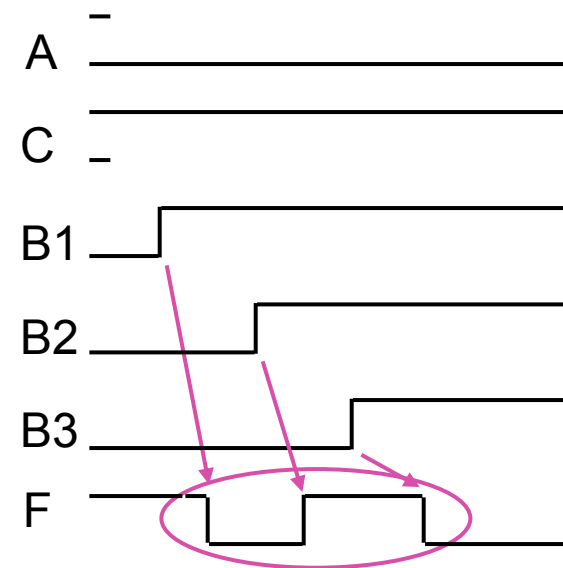


Dynamic hazards

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



Dynamic hazards

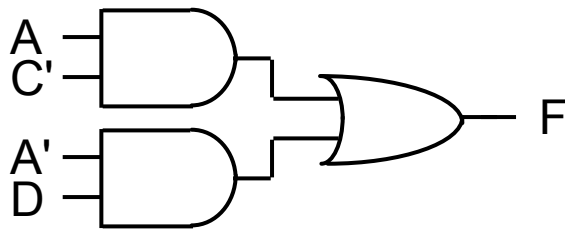


Dynamic hazard

Eliminating static hazards

- In 2-level logic circuits
 - Assuming single-bit changes
- Glitches happen when a changing input spans separate k-map implicants used in the gate realization
 - Example: 1101 to 0101 change can cause a static-1 glitch

$$F = AC' + A'D$$

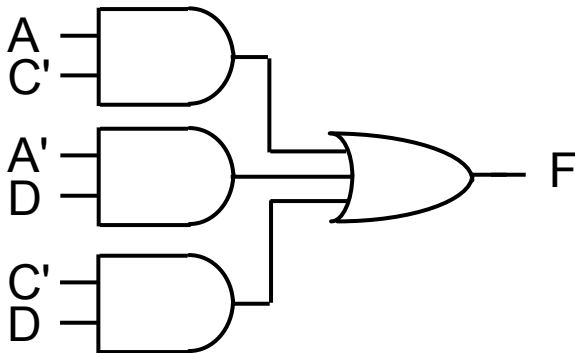


CD \ AB		A			
		00	01	11	10
C	00	0	0	1	1
	01	1	1	1	1
	11	1	1	0	0
	10	0	0	0	0
		B		D	

Eliminating static hazards (con't)

- Solution: Add redundant prime implicant to cover transition
 - Ensure that all single-bit changes are covered
 - Eliminates static-1 hazards: use SOP form

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



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

The table is a 4x4 Karnaugh map for variables A, B, C, and D. The columns are labeled AB (00, 01, 11, 10) and the rows are labeled CD (00, 01, 11, 10). The output values are 0 or 1. A red box highlights the 2x2 sub-region where C=01 and B=01, 11, which corresponds to the prime implicant C'D.

- 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
 - Test vectors probe a design for hazards

Combinational logic lessons

- Multi-level logic advantages over 2-level logic
 - Smaller circuits
 - Reduced fan-in
 - Less wires
 - May be faster due to smaller gates
- Multi-level disadvantages with respect to 2-level logic
 - More difficult design
 - Less powerful optimizing tools
 - Dynamic hazards
 - May be faster due to less gate levels