

BROWN
School of Engineering

DIGITAL ELECTRONICS SYSTEM DESIGN

FALL 2019
PROF. IRIS BAHAR
 SEPTEMBER 18, 2019
 LECTURE 5: TIMING HAZARDS & COMBINATIONAL BLOCKS

MIDTERM EXAM


- Please mark your calendars:
- The midterm exam will be held on **Wednesday, October 30**
 - In class, 90 minutes
 - 15% of your total grade

REVIEW: PRIME IMPLICANTS

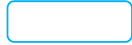
- An **implicant** is a product/sum term obtained by combining adjacent squares
- A **prime implicant** is a product/sum term obtained by combining the maximum number of adjacent squares
- An **essential prime implicant** is
 - A prime implicant
 - ... that must be included in order to cover a "one" in the function
 - This works with zeros to make Maxterms too
- To find a simplified expression that covers all "1" in the function:
 - First find the essential prime implicants
 - Then add prime implicants to cover the minterms that are not yet covered

PRIME IMPLICANTS EXAMPLE

1	0	0	0
1	1	1	0
0	1	1	1
0	1	0	0



is not essential (removing it does not uncover a "1")



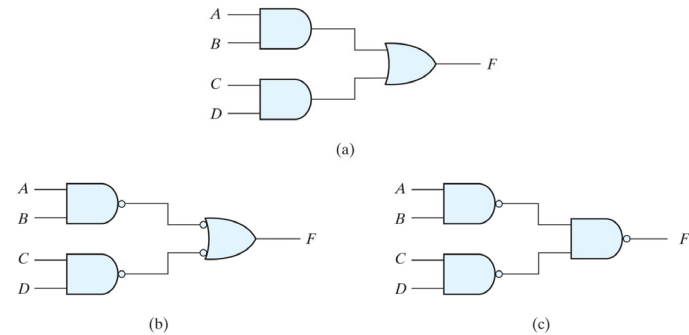
is essential

PROCEDURE FOR DESIGNING A COMBINATIONAL CIRCUIT

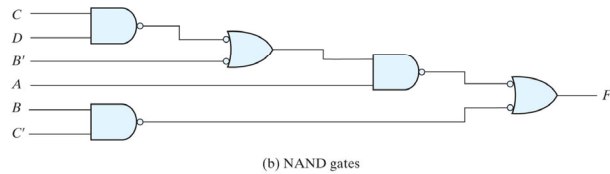
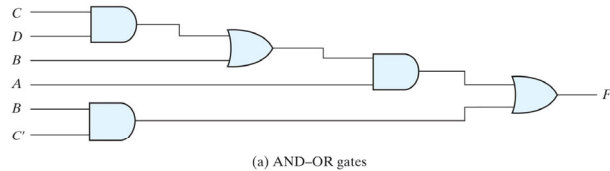
1. Write the truth table
2. Derive a simplified Boolean expression for each output variable via
 - Karnaugh-maps OR
 - Derive a standard SOP/POS and simplify via Boolean algebra
3. Draw the logic diagram
4. Wire gates together OR implement in Verilog

6

3 WAYS TO IMPLEMENT $F = AB + CD$

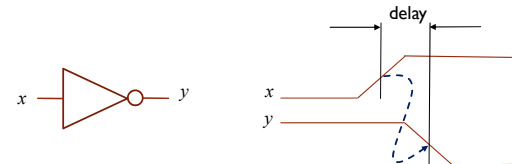


IMPLEMENTING $F = A(CD + B) + BC'$



STATIC VS. TRANSIENT BEHAVIOR

- So far we have only considered steady-state behavior of the logic circuits
- Signals at the output of gates do not change instantaneously



- *How may this impact our circuit designs?*

STATIC HAZARDS

- Glitch/hazard: A short pulse at the output of a circuit, when steady-state analysis predicts output does not change.
 - Result of differences in propagation delay between paths
- Example: $f(a,b,c) = m_3 + m_4 + m_6 + m_7$
 $= a'bc + ab'c' + abc' + abc$
- What does the K-map look like and what is the minimized Boolean expression for the function?

CORRESPONDING K-MAP

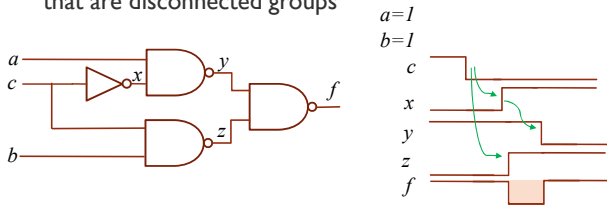
		b = 1			
		(0,0)	(0,1)	(1,1)	(1,0)
c = 0	0	0	0	1	1
c = 1	1	0	1	1	0
		a = 1			

$$f(a,b,c) = ac' + bc$$

- What does the circuit implementation look like?

TIMING HAZARDS IN CIRCUITS

- Hazard can occur when input change spans prime implicants that are disconnected groups



- Glitch corresponds to the transition $abc=111 \rightarrow 110$

REMOVING GLITCHES

		b = 1			
		(0,0)	(0,1)	(1,1)	(1,0)
c = 0	0	0	0	1	1
c = 1	1	0	1	1	0
		a = 1			

$$f(a,b,c) = ac' + bc + ab$$

- By adding the term ab we cover the transition $abc=111 \rightarrow 110$ with a single prime implicant
- No glitch!

PROBLEMS WITH GLITCHES

- Why are glitches bad?
 - Depending on how the circuit's output is used, a system's operation may or may not be adversely affected
 - May cause accidental update of data in memory units
 - Logic switching translates to voltage changes and circuit capacitances being charged and discharged
 - consequences in wasted energy consumption

$$i = C \frac{dV}{dt} \quad P = iV = CV \frac{dV}{dt} \approx CV^2$$

COMBINATIONAL BUILDING BLOCKS

MULTI-LEVEL LOGIC

- So far we have primarily focused on 2-level representations for combinational logic (SOP or POS)
- Multilevel logic is typically more compact (i.e., more cost-efficient) in practice

COMBINATIONAL BUILDING BLOCKS

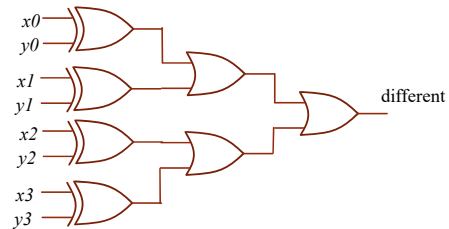
- More complex functions built from basic gates
 - Comparators
 - Multiplexors
 - Decoders
 - Encoders
- Typically tens to hundreds of transistors
- Common building blocks for digital systems

EQUALITY COMPARATORS WITH XORS

- 1-bit comparator

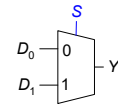


- 4-bit comparator



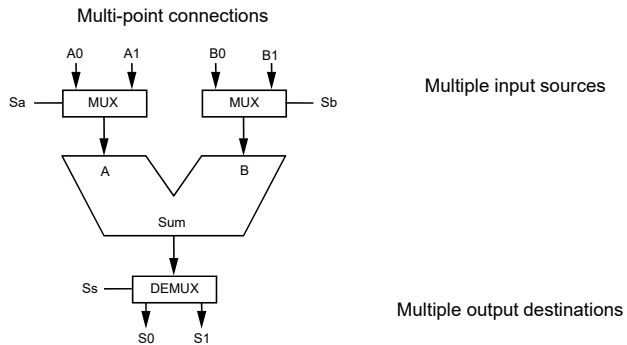
MULTIPLEXOR ("MUX")

- Connects one of n inputs to the output
 - "Select" control signals pick 1 of the n sources
 - $\log_2 n$ select bits
- Useful when multiple data sources need to be routed to a single destination
 - Often arises from resource sharing
 - Example: select 1-of- n data inputs to an adder

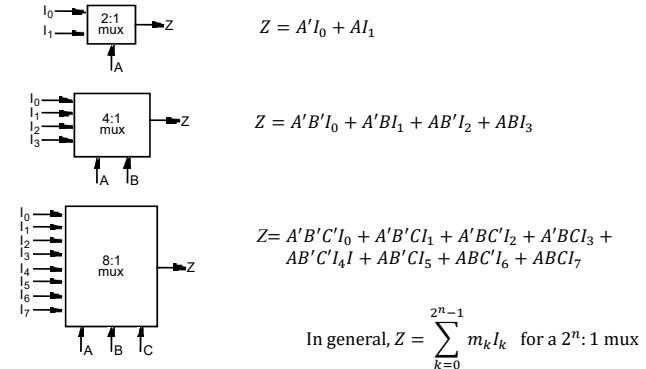


S	D ₁	D ₀	Y	S	Y
0	0	0	0	0	D ₀
0	0	1	1	1	D ₁
0	1	0	0		
0	1	1	1		
1	0	0	0		
1	0	1	0		
1	1	0	1		
1	1	1	1		

USE OF MULTIPLEXORS/SELECTORS

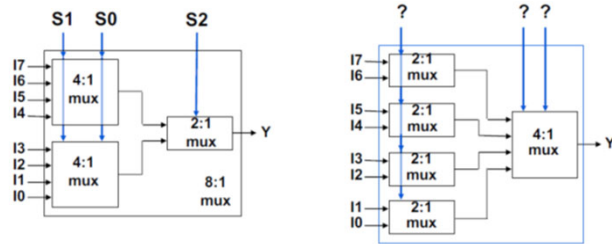


USE OF MULTIPLEXERS/SELECTORS



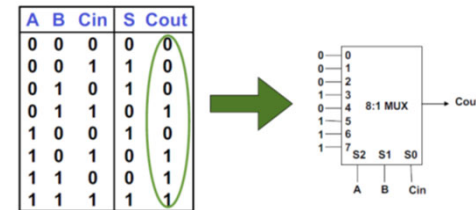
CASCADING MULTIPLEXORS

- Large multiplexors can be implemented by cascading smaller ones



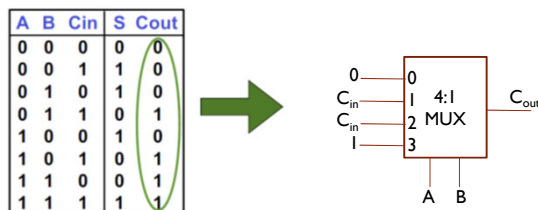
LOGIC FUNCTIONS USING MUXES

- Any function of n variables can be implemented with a $2^n:1$ multiplexor
 - Input variables connected to select inputs
 - Data inputs tied to 0 or 1 according to truth table



LOGIC FUNCTIONS USING MUXES

- Any function of n variables can be implemented with a $2^n:1$ multiplexor
 - How do we implement Cout with a single 4:1 MUX?



DECODER: DEFINITION

- N inputs, 2^N outputs
- One-hot outputs: only one output HIGH at once

