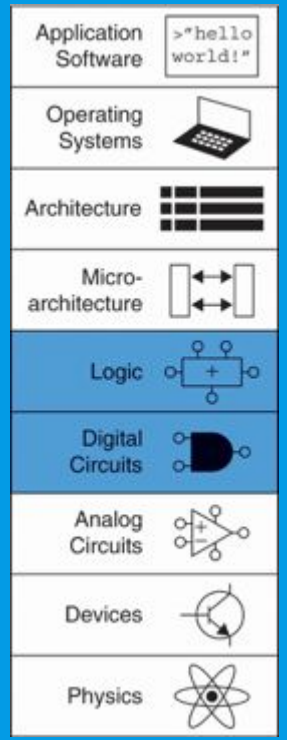
COSC175 (Systems I): Computer Organization & Design

Professor Lillian Pentecost
Fall 2024

Warm-Up September 12

- Where we were
  - Picking up with ***simplifying*** boolean equations and ***proving*** equality
  - Describing simple circuits in SystemVerilog
- Where we are going
  - Even more tools for simplification, more examples, a fan favorite: **K-Maps**
- Logistics, Reminders
  - Evening help sessions 7-9PM on Sundays, Tuesdays, Thursdays in C107
  - LP Office Hours at 2:30-4PM Today, CANCELED on Monday
  - Weekly Exercises Due Friday 5PM
  - Lab 1 due Monday 10PM
  - **Thank you to our volunteers, the lab sections are more balanced now!**
- Textbook Tags: 2.4, 2.5, 2.6, 2.7



Day2 Check-In (Recap)

A. Prove T9 (***covering***) theorem using each method (TT, other theorems)
B. Simplify the following expressions into ***two or fewer terms***, each with ***two or fewer literals***, then check your work with a truth table:

a. $Y = AC + BC + ABC$

b. $Y = \bar{A}\bar{B} + \bar{A}B\bar{C} + \overline{(A + \bar{C})}$

Number	Theorem	Dual	Name
T1	$B \bullet 1 = B$	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	$B + B = B$	Idempotency
T4	$\overline{\bar{B}} = B$		Involution
T5	$B \bullet \bar{B} = 0$	$B + \bar{B} = 1$	Complements

#	Theorem	Dual	Name
T6	$B \bullet C = C \bullet B$	$B+C = C+B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	$B + (C \bullet D) = (B+C)\,(B+D)$	Distributivity
T9	$B \bullet (B+C) = B$	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	$(B+C) \bullet (B+\bar{C}) = B$	Combining
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = (B \bullet C) + (\bar{B} \bullet D)$	$(B+C) \bullet (\bar{B}+D) \bullet (C+D) = (B+C) \bullet (\bar{B}+D)$	Consensus
T12	$\overline{B \bullet C \bullet D \ldots} = \bar{B}+\bar{C}+\bar{D}\ldots$	$\overline{B+C+D\ldots} = \bar{B} \bullet \bar{C} \bullet \bar{D}\ldots$	De Morgan's

Simplify B.b together on board

Define Minimization: **depends on your priorities!** For a boolean equation, we would aim for a sum of fewest, simplest possible ***implicants*** (or products), but for a circuit, we might want to minimize ***total number of gates***, or ***eliminate certain types of gates***, or ***reduce the "depth"*** of the circuit, to make it optimal.

b. $Y = \bar{A}\bar{B} + \bar{A}B\bar{C} + \overline{(A + \bar{C})}$

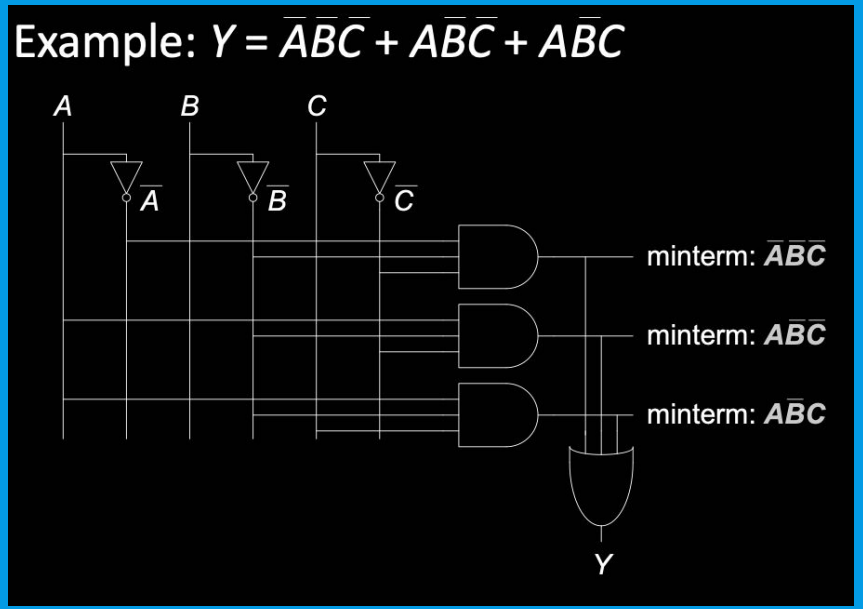
Number	Theorem	Dual	Name
T1	B • 1 = B	B + 0 = B	Identity
T2	B • 0 = 0	B + 1 = 1	Null Element
T3	B • B = B	B + B = B	Idempotency
T4	$\overline{\bar{B}} = B$		Involution
T5	B • $\overline{B}$ = 0	B + $\overline{B}$ = 1	Complements

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\bar{C}$) = B	(B+C) • (B+$\bar{C}$) = B	Combining
T11	(B•C) + ($\bar{B}$•D) + (C•D) = (B•C) + ($\bar{B}$•D)	(B+C) • ($\bar{B}$+D) • (C+D) = (B+C) • ($\bar{B}$+D)	Consensus
T12	$\overline{B•C•D\ldots} = \bar{B}+\bar{C}+\bar{D}\ldots$	$\overline{B+C+D\ldots} = \bar{B}•\bar{C}•\bar{D}\ldots$	De Morgan's

Organized Gate Schematics using PLAs

Programmable Logic Arrays are a general structure for constructing canon form schematics, making translating SOP, for example, very straightforward!

See DDCA 2.4, 2.5 for more tips and structures for schematics!



D, X, and Z as Values?

D for a **“Don’t Care”** value in the development of a design

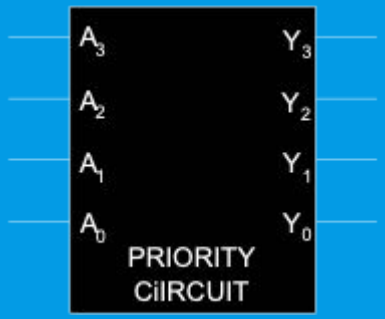
X for an **illegal value**, the result of a contentious or unclear operation

Z for a **floating value**, not knowably 0 or 1 or somewhere in between

“Don’t Care”s in Truth Tables: Example

Imagine a ***priority circuit:*** Output of “1” corresponding to most significant “1” input

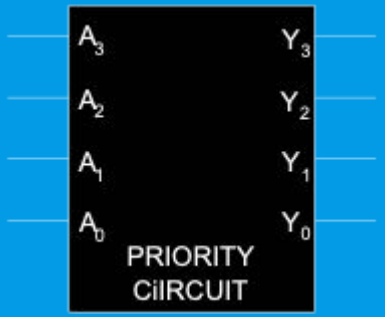
… starting with the truth table



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

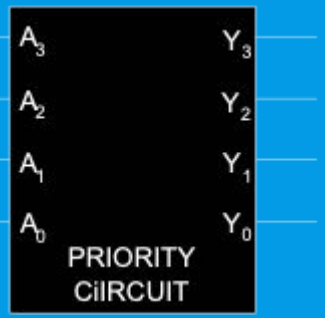
“Don’t Care”s in Truth Tables: Example

Imagine a ***priority circuit:*** Output of “1” corresponding to most significant “1” input



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

“Don’t Care”s in Truth Tables: Example



Imagine a ***priority circuit:*** Output of “1” corresponding to most significant “1” input

… we will use a “D” or an “X” in a truth table to denote “don’t care”

Remember your majority vote circuit – are there any potential “Don’t Care”s in that behavior?

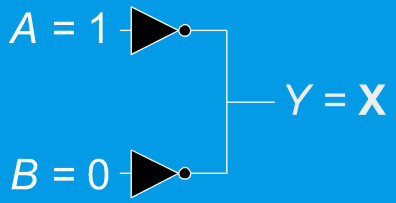
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	x	0	0	1	0
0	1	x	x	0	1	0	0
1	x	x	x	1	0	0	0

$$Y_3 = A_3$$
$$Y_2 = \overline{A_3}\, A_2$$
$$Y_1 = \overline{A_3}\, \overline{A_2}\, A_1$$
$$Y_0 = \overline{A_3}\, \overline{A_2}\, \overline{A_1}\, A_0$$

Contention: X

- **Contention:** circuit tries to drive output to 1 **and** 0
  - Actual value somewhere in between
  - Could be 0, 1, or in forbidden zone
  - Might change with voltage, temperature, time, noise
  - Often causes excessive power dissipation

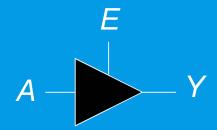
- **X is also used for:**
  - **Uninitialized values**
  - **Don't Care**
- **Warnings:**
  - Contention or uninitialized outputs usually indicate a **bug**.
  - **L**ook at the context to tell meaning



Floating: Z

- Floating, high impedance, open, high Z
- Floating output might be 0, 1, or somewhere in between
  - A voltmeter **won't** indicate whether a node is floating
  - But if you touch the node or wait a while, it may change values

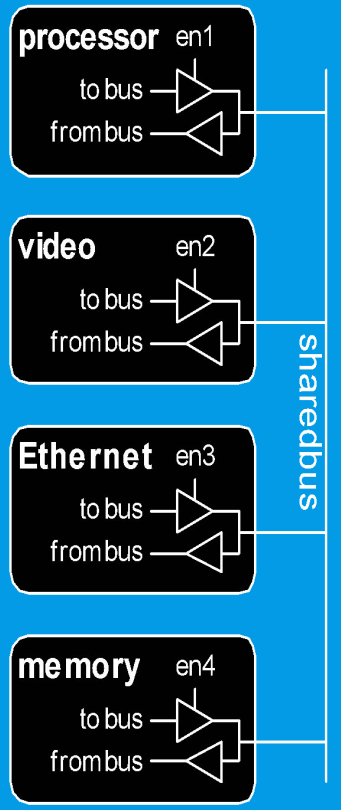
Tristate Buffer



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Tristate Busses for mediating components

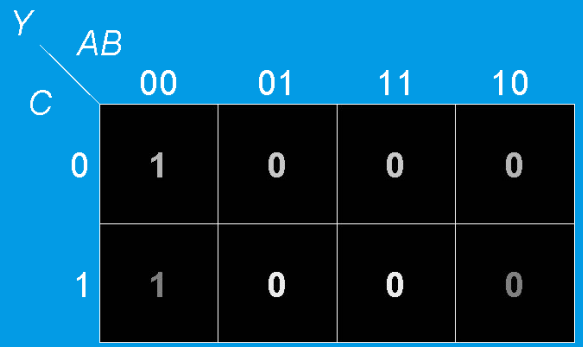
- Floating nodes are used in tristate busses
  - Many different drivers
  - Exactly one is active at once

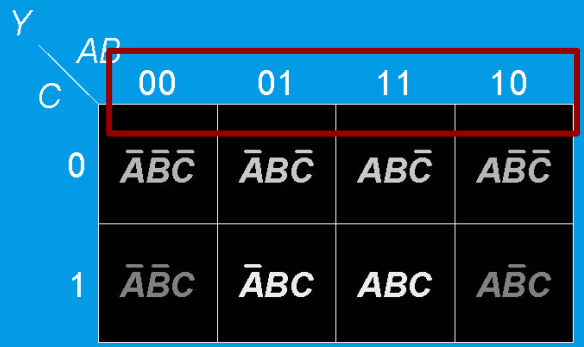


Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations ***graphically*** and ***efficiently***
- Multi-bit inputs are listed in ***gray code*** ordering

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

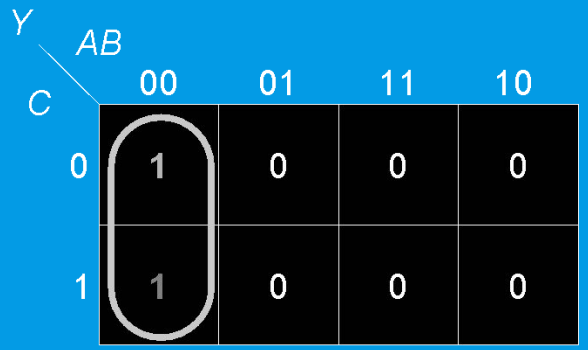


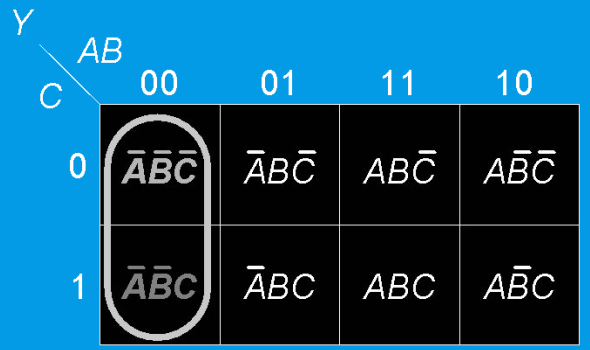


Making K-Maps Useful

- Circle 1’s in adjacent squares
- In groups of 1, 2, 4, 8… (powers of two)

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





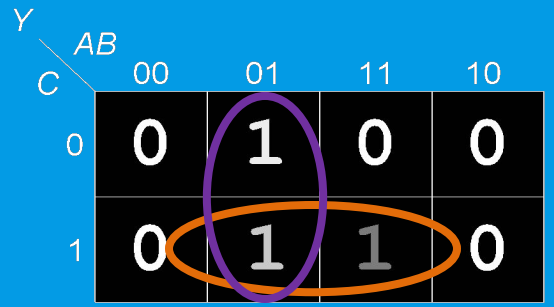
3-Input K-Map

- Circle 1’s in adjacent squares
- In Boolean expression: include only literals whose true **and** complement form are ***not*** in the circle

Truth Table

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

K-Map



$$Y = \bar{A}B + BC$$

Rules for K-Map Minimization

- **Every 1 must be circled** at least once
- Each circle must span a **power of 2** (i.e. 1, 2, 4) squares in each direction
- Each circle must be as **large** as possible
- A circle may **wrap around the edges**

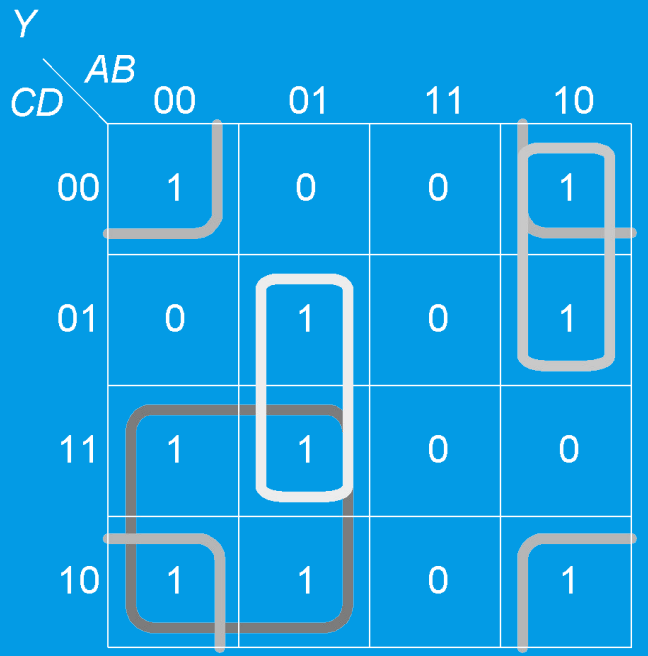
Crank it up!! 4-input example

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

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

Crank it up!! 4-input example

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



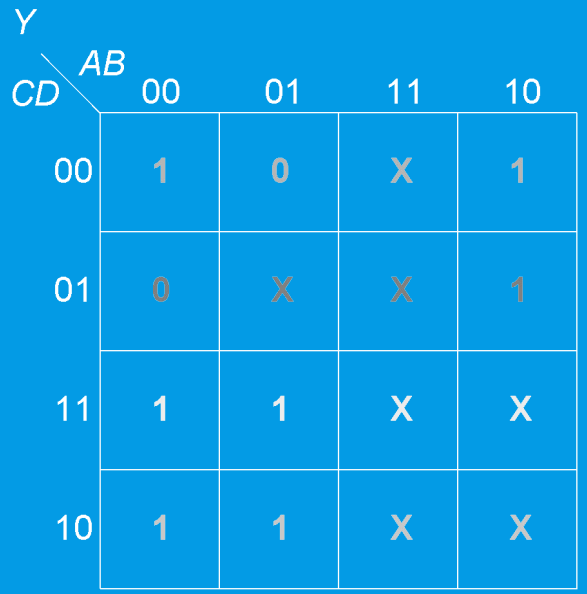
$$Y = \overline{A}C + \overline{A}BD + A\overline{B}\overline{C} + \overline{B}\overline{D}$$

… Now let’s add another rule

- **Every 1 must be circled** at least once
- Each circle must span a **power of 2** (i.e. 1, 2, 4) squares in each direction
- Each circle must be as **large** as possible
- A circle may **wrap around the edges**
- *Circle a **“don't care”** (D) **only if it helps** minimize the equation*

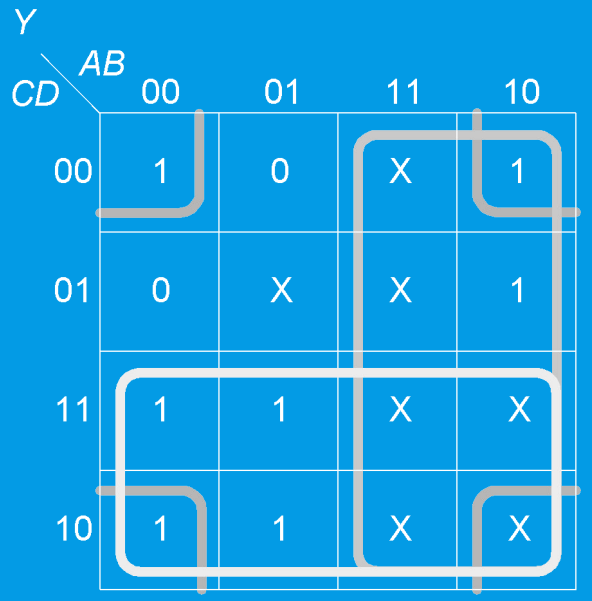
“Don’t Care”s in K-Maps

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



“Don’t Care”s in K-Maps

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



$Y = A + \overline{B}\overline{D} + C$

Wrap-Up September 12

- Coming up next!
  - Bigger, better, multi-component designs → **building computational units**
  - ***On your own: be sure to read DDCA 2.4 and 2.5 to inform your designs and schematic drawing!***
- Logistics, Reminders
  - Evening help sessions 7-9PM on Sundays, Tuesdays, Thursdays in C107
  - LP Office Hours at 2:30-4PM Today, CANCELED on Monday
  - Weekly Exercises Due Friday 5PM
  - Lab 1 due Monday 10PM (***you have keycard access to C107, the laptops and FPGAs must stay in that room at all times, no propping door open, clean up when you leave***)
  - **Thank you to our volunteers, the lab sections are more balanced now!**
- FEEDBACK
  - https://forms.gle/5Aafcm3iJthX78jx6