

Karnaugh Maps and Other Topic on Combinatorial Logic

**CS 64: Computer Organization and Design Logic
Lecture #12
Fall 2018**

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- Lab #7 on Tuesday
- Due by **Sunday** at midnight
 - Not Friday... Happy Thanksgiving!
- **No class this Wednesday! 😊😊😊**

Lecture Outline

- Logic Functions and their Simplifications:
Truth Table Use vs. **Karnaugh Maps**

Scaling Up Simplification

- When we get to *more* than 3 variables, it becomes challenging to use truth tables
- We can instead use ***Karnaugh Maps*** to make it immediately apparent as to what can be simplified

Example of a K-Map

0
1
2
3

| A | B | f(A,B) |
|---|---|--------|
| 0 | 0 | a |
| 0 | 1 | b |
| 1 | 0 | c |
| 1 | 1 | d |

| B \ A | 0 | 1 |
|-------|---|---|
| | | |
| 0 | a | c |
| 1 | b | d |

| B \ A | 0 | 1 |
|-------|---|---|
| | | |
| 0 | 0 | 2 |
| 1 | 1 | 3 |

| A | B | f(A,B) |
|---|---|--------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| B \ A | 0 | 1 |
|-------|---|---|
| | | |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

K-Maps with 3 or 4 Variables

| $AB \backslash C$ | | A | | | |
|-------------------|---|-----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| C | 0 | 0 | 2 | 6 | 4 |
| | 1 | 1 | 3 | 7 | 5 |

Blue brackets indicate groupings: a horizontal bracket labeled B under the bottom row (C=1), and a vertical bracket labeled C to the left of the rightmost column (A=10).

Note the adjacent placement of:

00 01 11 10

It's NOT:

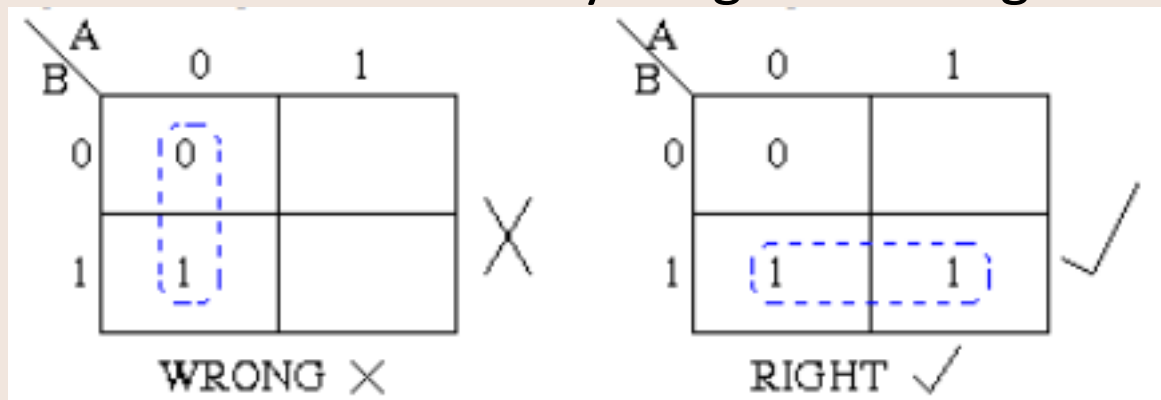
00 01 10 11

| $AB \backslash CD$ | | A | | | |
|--------------------|----|-----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| C | 00 | 0 | 4 | 12 | 8 |
| | 01 | 1 | 5 | 13 | 9 |
| | 11 | 3 | 7 | 15 | 11 |
| | 10 | 2 | 6 | 14 | 10 |

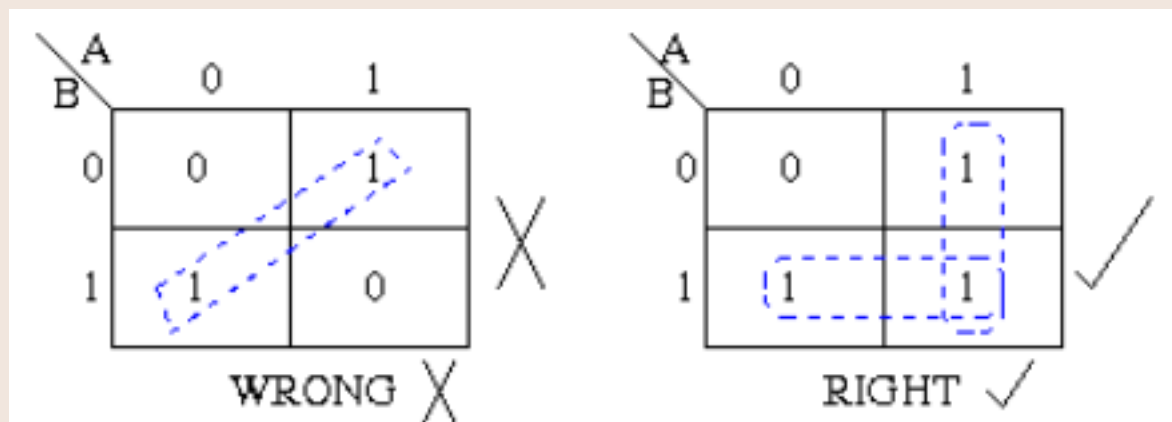
Blue brackets indicate groupings: a horizontal bracket labeled B under the bottom row (C=10), and a vertical bracket labeled D to the right of the rightmost column (A=10).

Rules for Using K-Maps for Simplification

1. Group together **adjacent cells** containing “1”
2. Groups should **not include** anything containing “0”



3. Groups may be horizontal or vertical, but **not diagonal**



Rules for Using K-Maps for Simplification

4. Groups must contain 1, 2, 4, 8, or in general 2^n cells.

| A \ B | 0 | 1 |
|-------|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |

Group of 2

RIGHT ✓

| AB \ C | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |

Group of 3

WRONG ✗

| A \ B | 0 | 1 |
|-------|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 1 |

Group of 4

RIGHT ✓

| AB \ C | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |

Group of 5

WRONG ✗

Rules for Using K-Maps for Simplification

5. Each group must be as large as possible

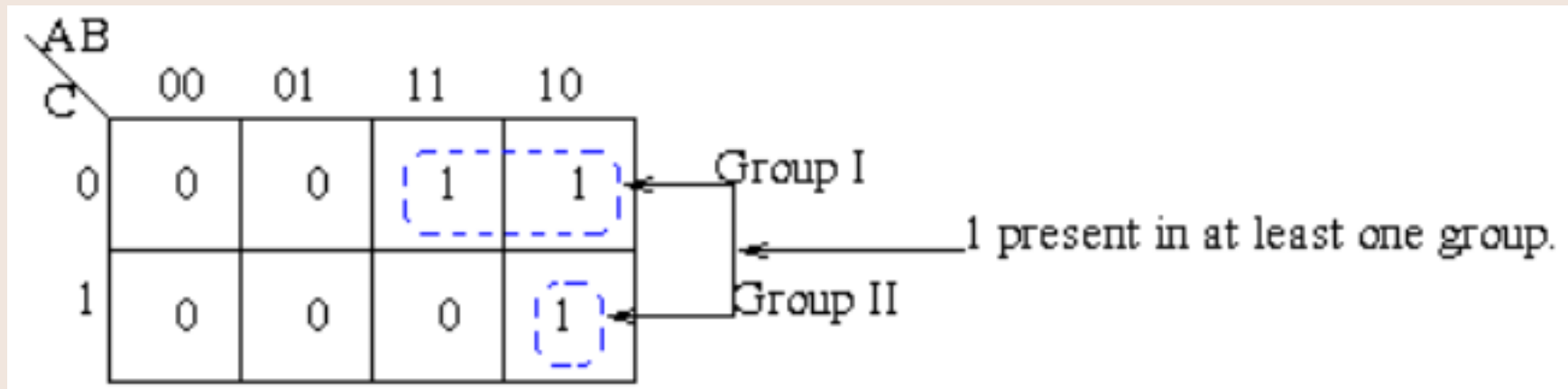
(Otherwise we're not being as minimal as we can be, even though we're not breaking any Boolean rules)

| AB \ C | | 00 | | | | 01 | | | | 11 | | | | 10 | | | |
|--------|--|---------|---|---|---|----|---|---|---|----|---|---|---|----|---|---|---|
| | | 0 | | | | 1 | | | | 0 | | | | 1 | | | |
| 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | RIGHT ✓ | | | | | | | | | | | | | | | |

| AB \ C | | 00 | | | | 01 | | | | 11 | | | | 10 | | | |
|--------|--|---------|---|---|---|----|---|---|---|----|---|---|---|----|---|---|---|
| | | 0 | | | | 1 | | | | 0 | | | | 1 | | | |
| 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | WRONG ✗ | | | | | | | | | | | | | | | |

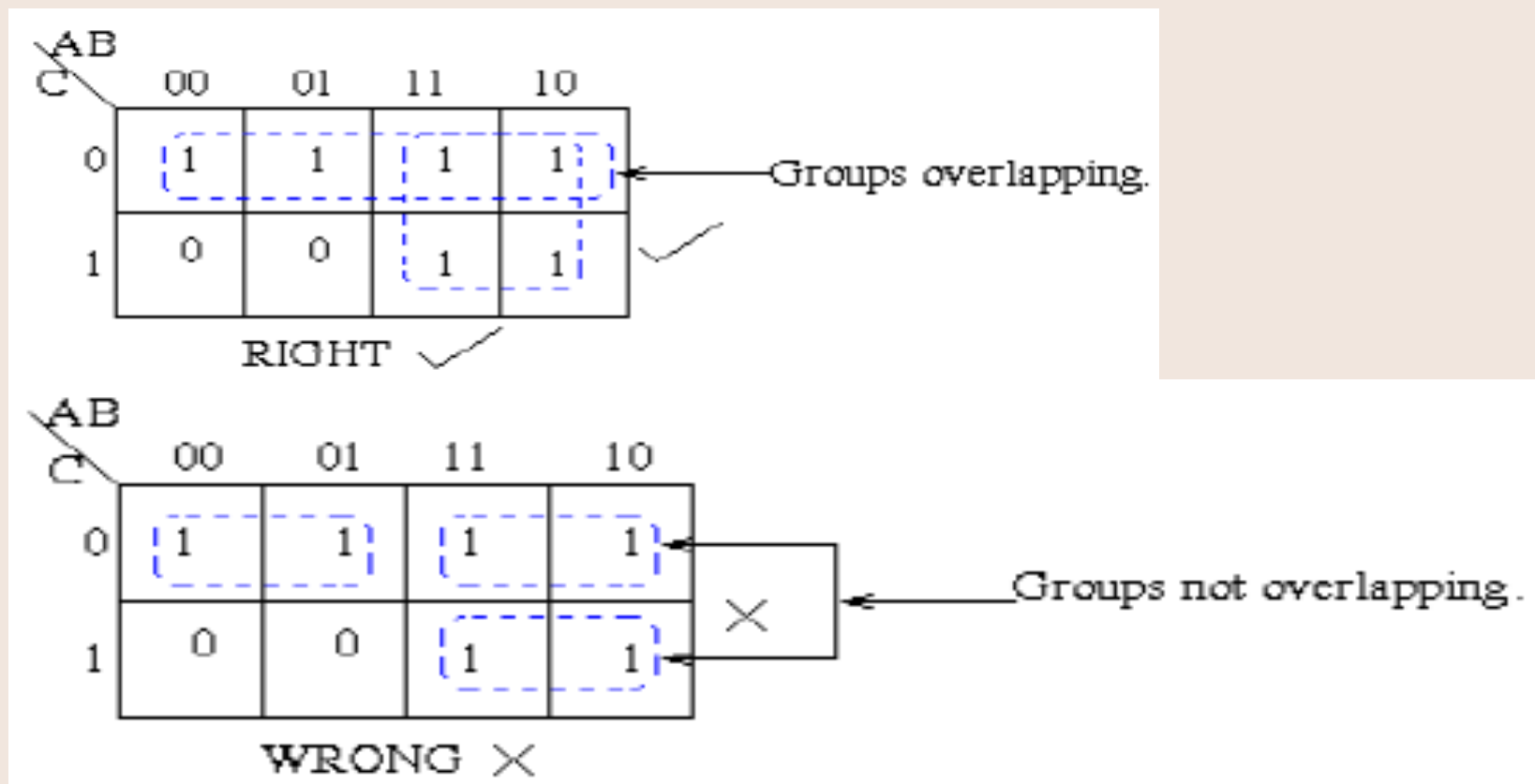
Rules for Using K-Maps for Simplification

6. Each cell containing a “1” must be at least in one group



Rules for Using K-Maps for Simplification

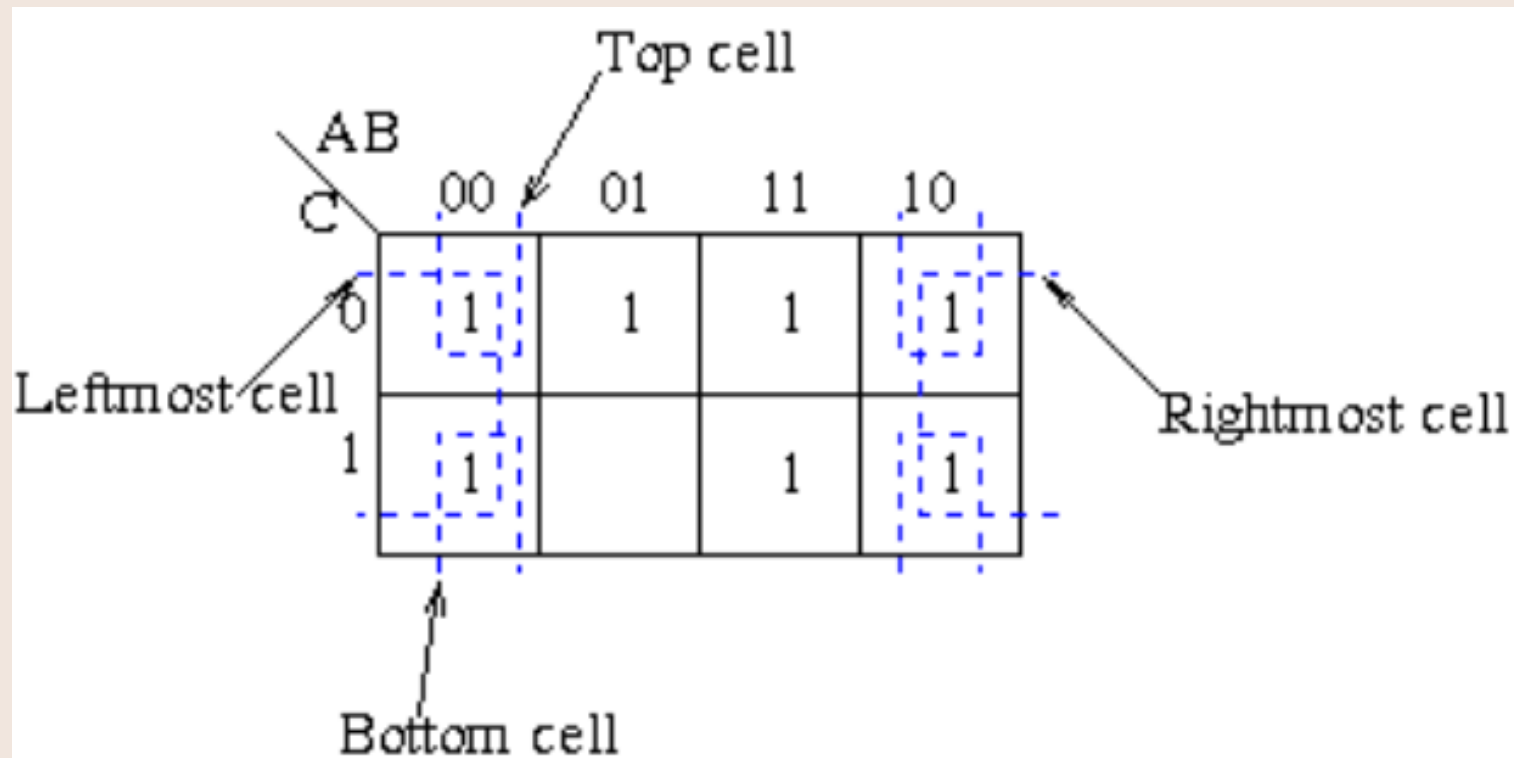
7. Groups may overlap esp. to maximize group size



Rules for Using K-Maps for Simplification

8. Groups may wrap around the table.

The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.



Example 1

2 vars

$F(X,Y)$

$$= XY + Y$$

$$= Y (X + 1)$$

$$= Y$$

Verifying results!

$$F(X,Y) = Y$$

Y = 1 column

| $\begin{array}{c} Y \\ \diagdown \\ X \end{array}$ | | 0 | 1 |
|----------------------------------------------------|--|---|---|
| 0 | | | 1 |
| 1 | | | 1 |

Example 2

3 vars

$F(X,Y,Z)$

$$= XZ + Z(X' + XY)$$

$$= XZ + ZX' + ZXY$$

$$= Z(X + X' + XY)$$

$$= Z(1 + XY)$$

$$= Z$$

A Karnaugh map for the function F(X,Y,Z) with variables X, Y, and Z. The map is a 2x4 grid. The columns are labeled with XY values: 00, 01, 11, 10. The rows are labeled with Z values: 0 and 1. The cell at (Z=1, XY=00) contains a red 1. The cell at (Z=1, XY=01) contains a red 1. The cell at (Z=1, XY=11) contains a red 1. The cell at (Z=1, XY=10) contains a red 1. A dashed red box encloses these four cells. A red arrow points from this box to the text F(X,Y,Z) = Z. Above the map, a bracket labeled Y=1 spans the columns 01 and 11. Another bracket labeled X=1 spans the columns 11 and 10.

| Z \ XY | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| 0 | | | | |
| 1 | 1 | 1 | 1 | 1 |

Verifying results!

$$F(X,Y,Z) = Z$$

Example 3

3 vars

Class Ex.

$$!A!B!C + !A!BC + !ABC + !AB!C + A!B!C + AB!C$$

| | | AB | | | |
|---|---|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| C | 0 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 1 | | |

$$F(X,Y,Z) = !C + !A$$

Example 4

4 vars

$$F(A,X,Y,Z)$$

$$= AX + Z(X+A'+Y)$$

$$= AX + ZX + ZA' + ZY$$

$$F(A,X,Y,Z) = ZA' + AX + ZY$$

A 4x4 Karnaugh map for the function F(A,X,Y,Z). The columns are labeled XY (00, 01, 11, 10) and the rows are labeled AZ (00, 01, 11, 10). The map contains 1s in the following cells: (01,01), (01,11), (01,10), (11,01), (11,11), (11,10), (10,11), and (10,10). Groupings are shown with brackets and dashed red boxes: a horizontal bracket for Y=1 covers columns 01 and 11; a horizontal bracket for X=1 covers columns 11 and 10; a vertical bracket for Z=1 covers rows 01 and 11; and a vertical bracket for A=1 covers rows 11 and 10. A red arrow points from the equation F(A,X,Y,Z) = ZA' + AX + ZY to the Z=1 grouping.

| AZ \ XY | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | 1 | 1 | 1 |
| 11 | | 1 | 1 | 1 |
| 10 | | | 1 | 1 |

Example 4

4 vars

Class Ex.

$F(A,B,C,D)$

$$= ABCD' + ABC'D + CD + A'B' + C'D$$

$$F(A,B,C,D) = A'B' + D + ABC$$

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 map contains 1s in the following cells: (00,00), (00,01), (00,11), (00,10), (01,00), (01,01), (01,11), (01,10), (11,00), (11,01), (11,11), (11,10), (10,00), (10,01), (10,11), (10,10). The map is annotated with groupings and prime implicants: a brown bracket at the top labeled 'B = 1' covers columns 01 and 11; a brown bracket at the top labeled 'A = 1' covers columns 11 and 10; a brown bracket on the left labeled 'D = 1' covers rows 01 and 11; a brown bracket on the left labeled 'C = 1' covers rows 11 and 10; a red dashed box covers the entire first column (AB=00); a red dashed box covers the entire second column (AB=01); a red dashed box covers the entire third column (AB=11); a red dashed box covers the entire fourth column (AB=10); a red arrow points from the text 'F(A,B,C,D) = A'B' + D + ABC' to the 'D = 1' grouping.

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | 1 | | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | | 1 | |

K-Map Rules Summary

1. Groups can contain only 1s
2. Only 1s in adjacent groups are allowed
3. Groups may ONLY be horizontal or vertical (no diagonals)
4. The number of 1s in a group must be a power of two (1, 2, 4, 8...)
5. Groups must be as large AND as few in no.s as “legally” possible
6. All 1s must belong to a group, even if it’s a group of one element
7. Overlapping groups are permitted
8. Wrapping around the map is permitted

Exploiting “Don’t Cares”

- An *output variable* that’s designated “don’t care” (symbol = X) means that it could be a **0** or a **1** (i.e. we “don’t care” which)
 - That is, it is **unspecified**,
usually because of invalid inputs

Example of a Don't Care Situation

- Consider coding all decimal digits (say, for a digital clock app):
 - 0 thru 9 --- requires how many bits?
 - 4 bits
 - But! 4 bits convey more numbers than that!
 - Don't forget A thru F!
- Not all binary values map to decimal



Example Continued...

| Binary | Decimal |
|--------|---------|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |

| Binary | Decimal |
|--------|---------|
| 1000 | 8 |
| 1001 | 9 |
| 1010 | X |
| 1011 | X |
| 1100 | X |
| 1101 | X |
| 1110 | X |
| 1111 | X |

Don't Care: So What?

- Recall that in a K-map, we can only group 1s
- Because the value of a don't care is irrelevant, we can treat it as a 1 ***if it is convenient to do so*** (or a 0 if that would be more convenient)

Example

- A circuit that calculates if the 4-bit binary coded single digit decimal **input % 2 == 0**
- So, although 4-bits will give me numbers from 0 to 15, I don't care about the ones that yield 10 to 15.

| I3 | I2 | I1 | I0 | R |
|----|----|----|----|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 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 |

Example as a K-Map

| $I_1 I_0$ | | | | | |
|-----------|----|----|----|----|----|
| $I_3 I_2$ | | 00 | 01 | 11 | 10 |
| | 00 | 1 | 0 | 0 | 1 |
| | 01 | 1 | 0 | 0 | 1 |
| | 11 | X | X | X | X |
| | 10 | 1 | 0 | X | X |

If We Don't Exploit “Don't Cares”

$$R = \bar{I}_1 \bar{I}_0 \bar{I}_3 + I_1 \bar{I}_0 \bar{I}_3 + \bar{I}_0 \bar{I}_1 \bar{I}_2 I_3$$

| $I_1 I_0$ | | | | | |
|-----------|--|----|----|----|----|
| $I_3 I_2$ | | 00 | 01 | 11 | 10 |
| | | | | | |
| 00 | | 1 | 0 | 0 | 1 |
| 01 | | 1 | 0 | 0 | 1 |
| 11 | | X | X | X | X |
| 10 | | 1 | 0 | X | X |

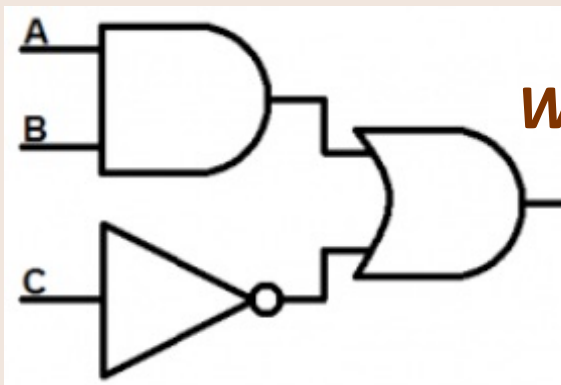
If We **DO** Exploit “Don’t Cares”

$$R = \bar{I}_0$$

| $I_1 I_0$ | | | | | |
|-----------|--|----|----|----|----|
| $I_3 I_2$ | | 00 | 01 | 11 | 10 |
| | | | | | |
| 00 | | 1 | 0 | 0 | 1 |
| 01 | | 1 | 0 | 0 | 1 |
| 11 | | X | X | X | X |
| 10 | | 1 | 0 | X | X |

Combinatorial Logic Designs

- When you *combine* multiple logic blocks together to form a more complex logic function/circuit



What is the output?

$$A.B + \overline{C}$$

What is its truth table?

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

What is its K-Map?

| C \ AB | | | | |
|--------|----|----|----|----|
| | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | | | 1 | |

Exercise 1

- Given the following truth table, draw the resulting logic circuit
 - **STEP 1:** Draw the K-Map and simplify the function
 - **STEP 2:** Construct the circuit from the now simplified function

| A | B | C | D | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

| A | B | C | D | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Exercise 1 – Step 1

Get the simplified function

| | | | | | |
|----|----|-------|----|-------|----|
| | | AB | | | |
| | | B = 1 | | A = 1 | |
| CD | | 00 | 01 | 11 | 10 |
| | 00 | | 1 | 1 | |
| | 01 | | | | |
| | 11 | | | 1 | 1 |
| | 10 | | | 1 | 1 |

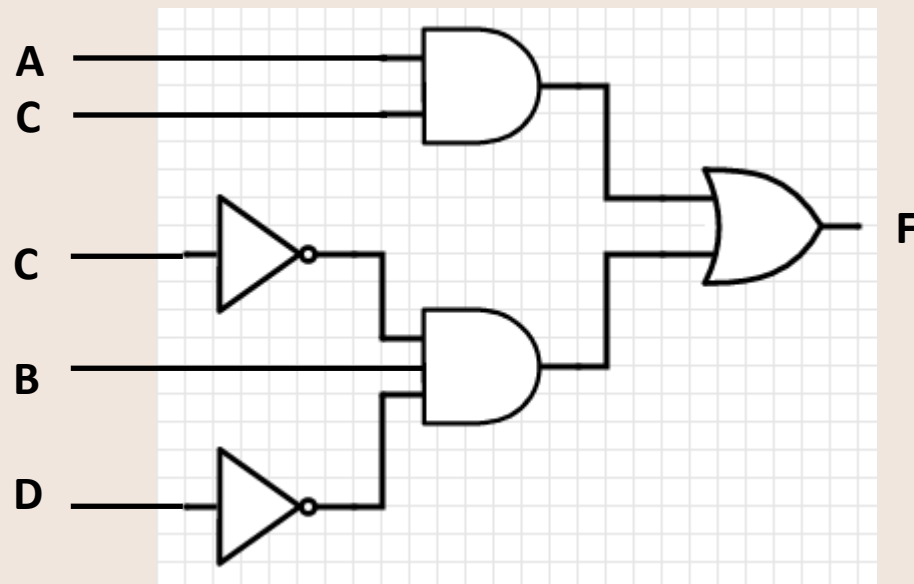
$D = 1$ (rows 11, 10)
 $C = 1$ (columns 01, 11)

$$F(A,B,C) = B.C'.D' + A.C$$

Exercise 1 – Step 2

Draw the logic circuit diagram

$$F(A,B,C) = B.C'.D' + A.C$$



Exercise 2

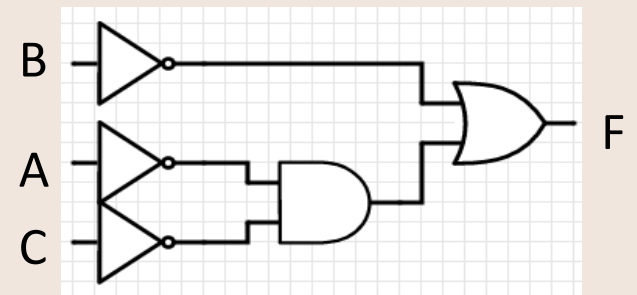
Class Ex.

- Given the following truth table, draw the resulting logic circuit

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

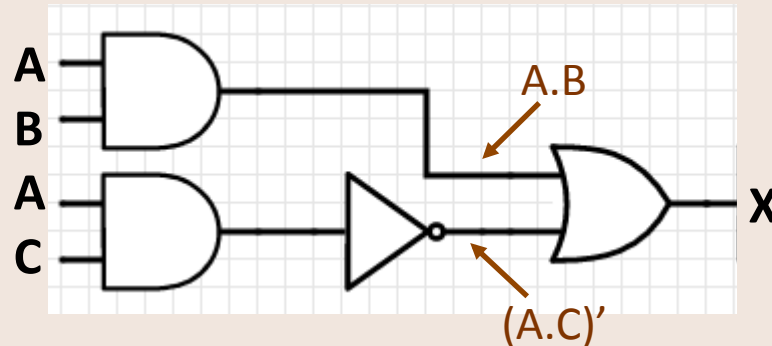
| | | AB | | | |
|---|---|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| C | 0 | 1 | 1 | | 1 |
| | 1 | 1 | | | 1 |

$$F(A,B,C) = B' + A'.C'$$



Exercise 3

- Given the following schematic of a circuit, (a) write the function and (b) fill out the truth table:



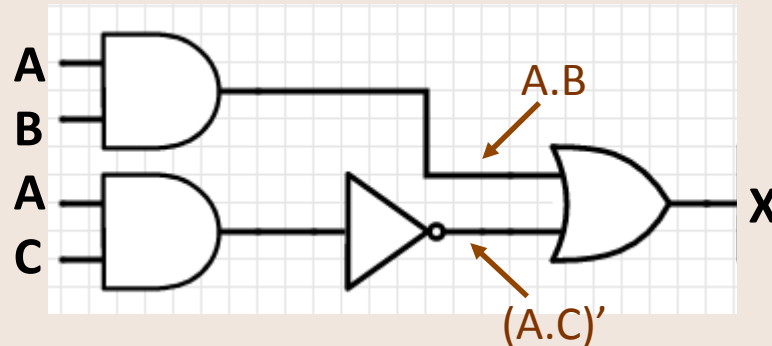
$$X = A.B + (A.C)'$$

(note that also means: $X = A.B + A' + C'$)

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

Exercise 3

- Given the following schematic of a circuit, (a) write the function and (b) fill out the truth table:



$$X = A.B + (A.C)'$$

(note that also means: $X = A.B + A' + C'$)

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Multiplexer

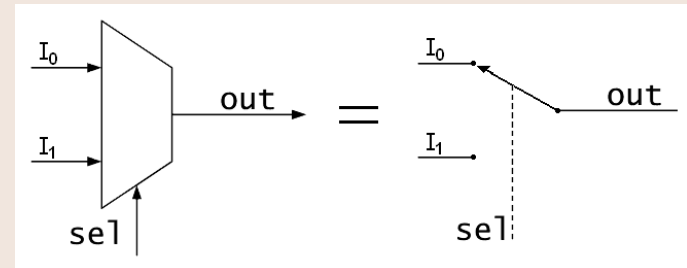
- A logical selector:
 - Select either input A or input B to be the output

```
// if s = 0, output is a
// if s = 1, output is b
int mux(int a, int b, int s)
{
    if (!s) return a;
    else return b;
}
```

Multiplexer

(Mux for short)

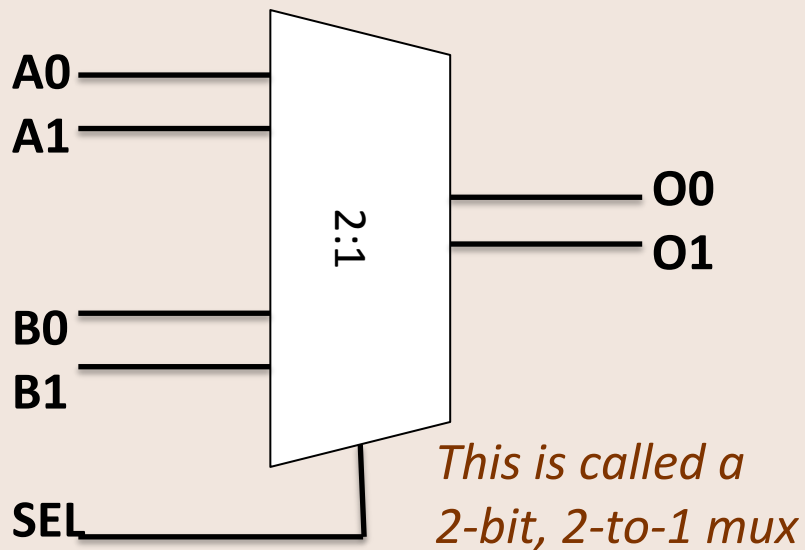
- Typically has 3 *groups of* inputs and 1 output
 - IN: 2 data , 1 select
 - OUT: 1 data



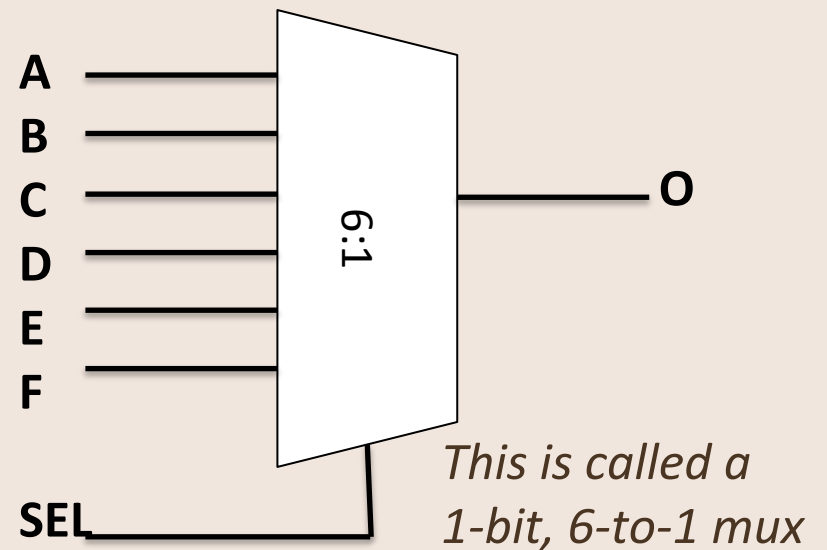
- 1 of the input data lines gets selected to become the output, based on the 3rd (select) input
 - If “Sel” = 0, then I_0 gets to be the output
 - If “Sel” = 1, then I_1 gets to be the output
- The opposite of a Mux is called a **Demultiplexer** (or **Demux**)

Mux Configurations

Muxes can have I/O that are multiple bits

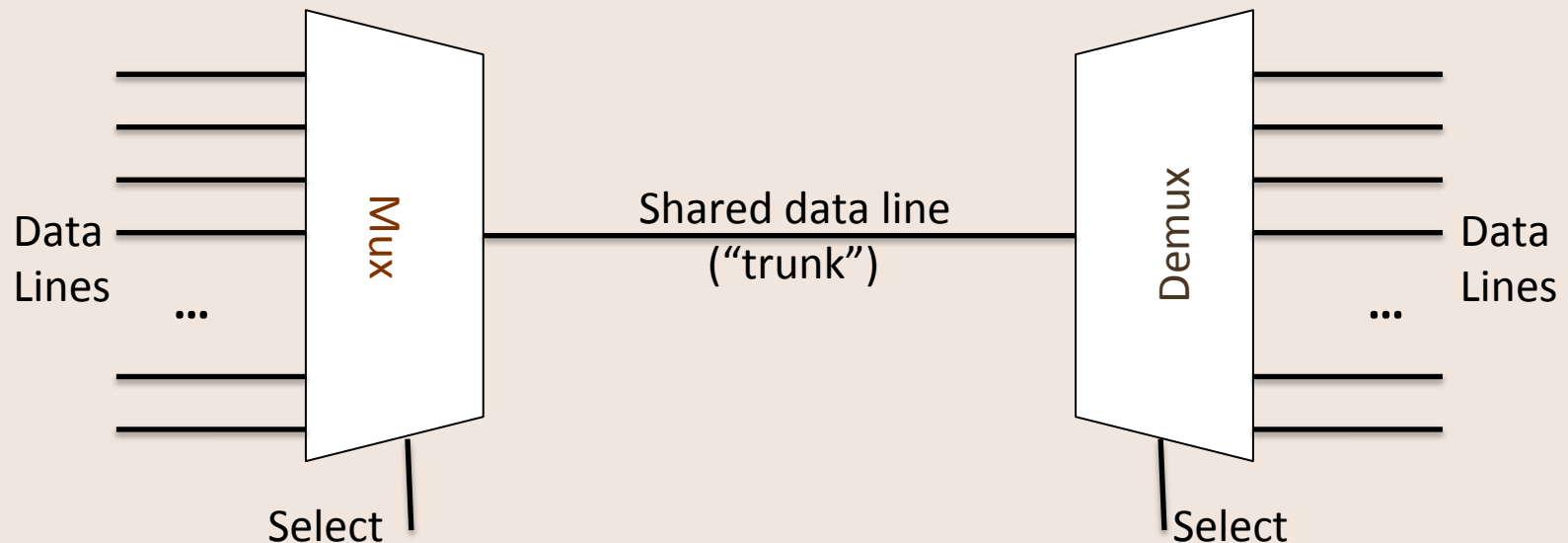


Or they can have more than two data inputs



The Use of Multiplexers

- Makes it possible for several signals (variables) to share one resource
 - Very commonly used in data communication lines



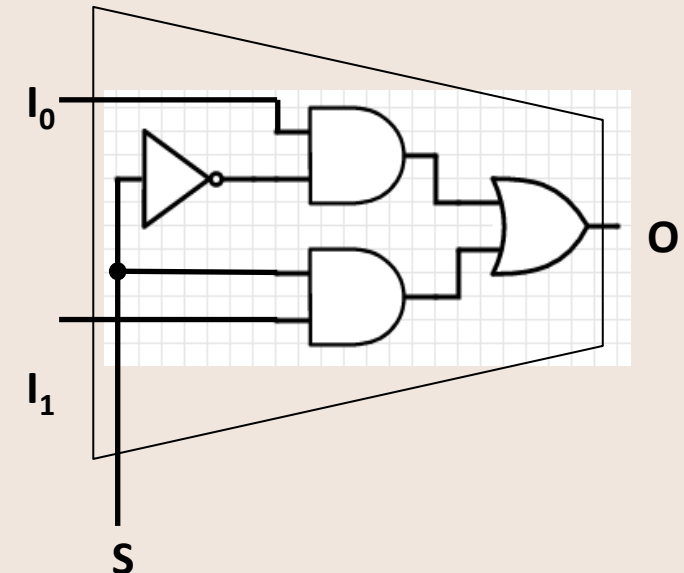
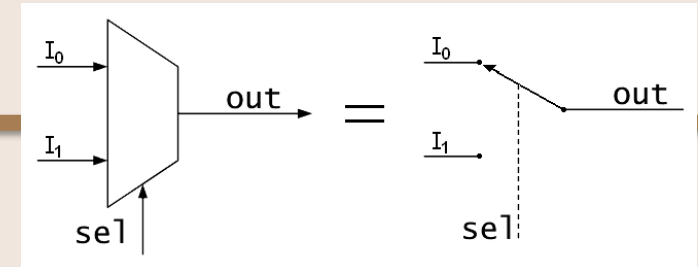
Mux Truth Table and Logic Circuit

1-bit Mux

| I_0 | I_1 | S | O |
|-------|-------|-----|-----|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

| S | $I_0 I_1$ | | | |
|-----|-----------|----|----|----|
| | 00 | 01 | 11 | 10 |
| 0 | | | 1 | 1 |
| 1 | | 1 | 1 | |

$$O = S \cdot I_1 + S' \cdot I_0$$



• = lines are physically connected

YOUR TO-DOs

- Lab #7 on Tuesday – Due by **Sunday** at midnight
- No class on Wednesday! 😊😊😊

</LECTURE>