Homework

- Update on website issue
- Reading: Chapter 7
- Homework: All exercises at end of Chapter 7
- Due 9/26

Computer Science Foundations

COS 140: Foundations of Computer Science

Karnaugh Maps

Fall 2018

Computer Science Foundations

The problem

- Given a circuit specification, how can we make the best circuit possible?
- What constitutes "better" for circuits?
    - Reduce the number of gates
    - Reduce the number of inputs (pins)
- May also have to use only a particular set of gates
    - Some chips have only one type of gate, and may have that chip
    - NAND and NOR are cheaper to make
    - Must be in a functionally complete set to be able to realize all functions, e.g.: {AND, OR, NOT}, {NAND}, {NOR}

Equivalence

- Recall: two circuits are equivalent if they perform the same function, without regard for the gates used, the way the circuit is constructed, etc.

Equivalence

- Recall: two circuits are equivalent if they perform the same function, without regard for the gates used, the way the circuit is constructed, etc.
- Equivalence is also a more general concept
  - Basically, two entities are equivalent if, for all possible inputs, they have the same output
  - Equivalence allows computer scientists to use “the right tool for the job” by choosing the entity that best suits their needs

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$\ldots + ABC + \overline{A}BC + \ldots$$

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$... + ABC + \overline{A}BC + ...$$

- Can replace via laws of Boolean algebra:

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$\ldots + ABC + \overline{A}BC + \ldots$$

- Can replace via laws of Boolean algebra:
  $\ldots + (A + \overline{A})BC + \ldots$

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$... + ABC + \overline{A}BC + ...$$

- Can replace via laws of Boolean algebra:
  $... + (A + \overline{A})BC + ...$ (Distributive Law)

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$\ldots + ABC + \overline{A}BC + \ldots$$

- Can replace via laws of Boolean algebra:
  $\ldots + (A + \overline{A})BC + \ldots$ (Distributive Law)
  $\ldots + BC + \ldots$

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$... + ABC + \overline{A}BC + ...$$

- Can replace via laws of Boolean algebra:
$... + (A + \overline{A})BC + ...$ (Distributive Law)
$... + BC + ...$ (Inverse Law)

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$\ldots + ABC + \overline{A}BC + \ldots$$

- Can replace via laws of Boolean algebra:
  $\ldots + (A + \overline{A})BC + \ldots$ (Distributive Law)
  $\ldots + BC + \ldots$ (Inverse Law)
- In other words, the value of the variable doesn't matter, and it can be eliminated from that pair

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$... + ABC + \overline{A}BC + ...$$

- Can replace via laws of Boolean algebra:
  $... + (A + \overline{A})BC + ...$ (Distributive Law)
  $... + BC + ...$ (Inverse Law)
- In other words, the value of the variable doesn't matter, and it can be eliminated from that pair
- The pair is replaced by a new term having one fewer variable

Computer Science Foundations

An insight

- Given a Boolean circuit specification—say, an SOP—how would you proceed?
- Suppose two terms differ only by the "sign" of a variable – one has the variable, the other the complement (negation):

$$\ldots + ABC + \overline{A}BC + \ldots$$

- Can replace via laws of Boolean algebra:
  $\ldots + (A + \overline{A})BC + \ldots$ (Distributive Law)
  $\ldots + BC + \ldots$ (Inverse Law)
- In other words, the value of the variable doesn't matter, and it can be eliminated from that pair
- The pair is replaced by a new term having one fewer variable
- Process is repeated until minimal expression found

Computer Science Foundations

Difficulty with Boolean approach

- Problem: Can be difficult to see which terms to combine, in what order

Difficulty with Boolean approach

- Problem: Can be difficult to see which terms to combine, in what order

$$\overline{A}B\overline{C}D + ABCD + \overline{A}BCD + AB\overline{C}D$$
$$+ ABC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$$

Computer Science Foundations

Difficulty with Boolean approach

- Problem: Can be difficult to see which terms to combine, in what order

$$\overline{A}B\overline{C}D \;+\; ABCD \;+\; \overline{A}BCD \;+\; AB\overline{C}D$$

$$+\; ABC\overline{D} \;+\; A\overline{B}C\overline{D} \;+\; A\overline{B}\,\overline{C}\,\overline{D}$$

- It would be better if there was some way to *see* which terms can be combined

Computer Science Foundations

Karnaugh Maps

- A Karnaugh Map is a visual representation of a Boolean SOP expression
- Each term is represented by a cell in a table (map)
- Adjacent cells differ in the "sign" of only one variable
- E.g., ABC would be adjacent to $AB\overline{C}$, also $\overline{A}BC$, ...
- So how to draw the map?

Computer Science Foundations

Example: The magic of Karnaugh Maps

Suppose you want to create a circuit for the majority function

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

and you want to minimize the circuit, keeping it an SOP.

Computer Science Foundations

Example: The magic of Karnaugh Maps

Create a Karnaugh Map for the number of variables that you have in the expression.

Example: The magic of Karnaugh Maps

Create a Karnaugh Map for the number of variables that you have in the expression.

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

C \ AB	00	01	11	10
0				
1				

Computer Science Foundations

Example: The magic of Karnaugh Maps

Put a 1 in squares that correspond to the terms in the expression.

- Term $\Leftrightarrow$ square:
  - 1 if variable occurs in the term, 0 if complement occurs
  - E.g.: $\overline{A}B\overline{C}$ $\Leftrightarrow$ square 010
- For truth tables:
  - Match the input pattern for rows where output is 1 to the square's label
  - E.g.: 0 0 1 | 1 $\Leftrightarrow$ square 001

		AB			
		00	01	11	10
C	0				
	1				

Example: The magic of Karnaugh Maps

$$\underline{\overline{A}BC} + A\overline{B}C + AB\overline{C} + ABC$$

AB

C \ AB	00	01	11	10
0				
1		1		

Computer Science Foundations

Example: The magic of Karnaugh Maps

$$\overline{A}BC + \underline{A\overline{B}C} + AB\overline{C} + ABC$$

AB

C \ AB	00	01	11	10
0				
1		1		1

Computer Science Foundations

Example: The magic of Karnaugh Maps

$$\overline{A}BC + A\overline{B}C + \underline{AB\overline{C}} + ABC$$

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

 –

Example: The magic of Karnaugh Maps

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + \underline{ABC}$$

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

Computer Science Foundations

Example: The magic of Karnaugh Maps

Circle groups of powers of $2 \geq 2^1$ (2, 4, 8, etc.) until all ones have been circled. Circle the largest groups possible.

Example: The magic of Karnaugh Maps

Circle groups of powers of $2 \geq 2^1$ (2, 4, 8, etc.) until all ones have been circled. Circle the largest groups possible.

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Computer Science Foundations

Example: The magic of Karnaugh Maps

Circle groups of powers of $2 \geq 2^1$ (2, 4, 8, etc.) until all ones have been circled. Circle the largest groups possible.

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

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

Computer Science Foundations

Example: The magic of Karnaugh Maps

Circle groups of powers of $2 \geq 2^1$ (2, 4, 8, etc.) until all ones have been circled. Circle the largest groups possible.

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

AB

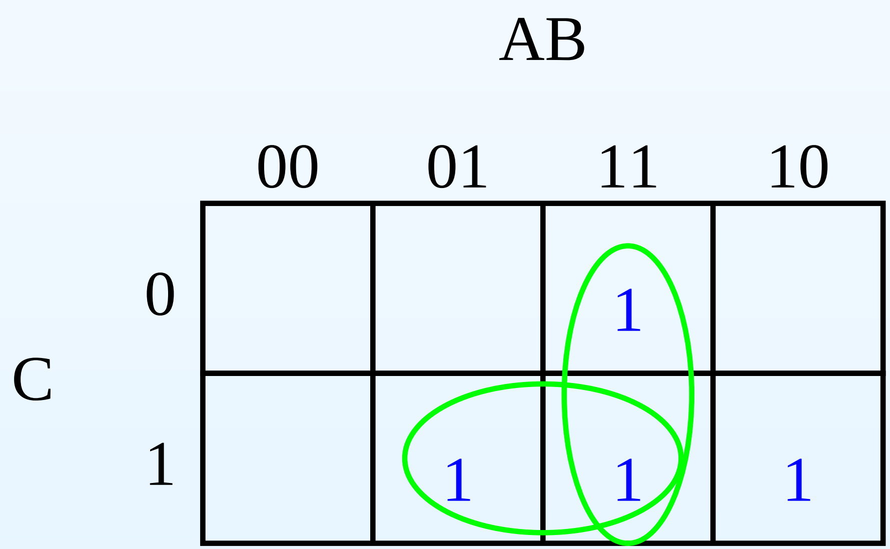
C	00	01	11	10
0			1	
1		1	1	1

Computer Science Foundations

Example: The magic of Karnaugh Maps

Circle groups of powers of $2 \geq 2^1$ (2, 4, 8, etc.) until all ones have been circled. Circle the largest groups possible.

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$



Example: The magic of Karnaugh Maps

Circle groups of powers of $2 \geq 2^1$ (2, 4, 8, etc.) until all ones have been circled. Circle the largest groups possible.

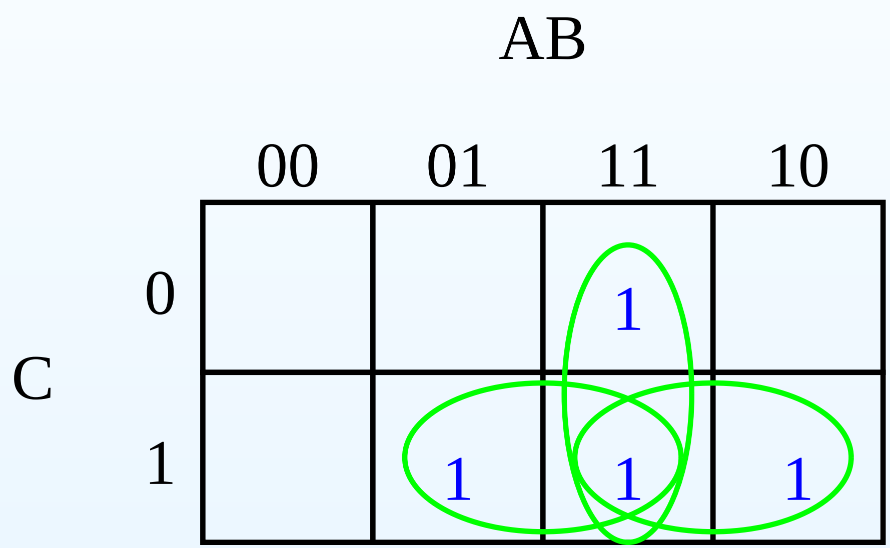
$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

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

Computer Science Foundations

Example: The magic of Karnaugh Maps

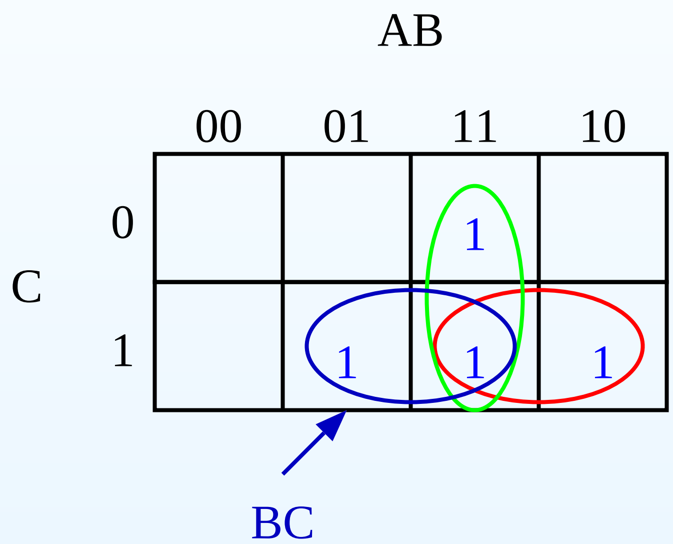
Read the terms from the circled items, leaving out variables that have different values within the group.



Computer Science Foundations

Example: The magic of Karnaugh Maps

Read the terms from the circled items, leaving out variables that have different values within the group.

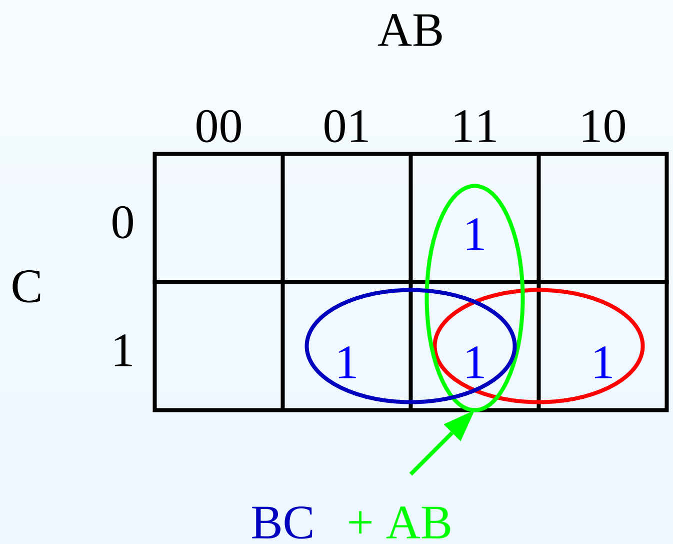


(Because B is same, C is same, but A = both 1 & 0)

Computer Science Foundations

Example: The magic of Karnaugh Maps

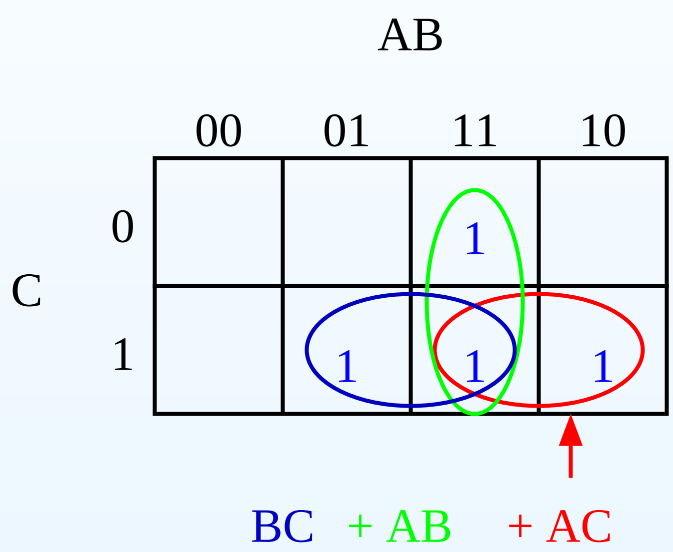
Read the terms from the circled items, leaving out variables that have different values within the group.



(Because A is same, B is same, but C = both 1 & 0)

Example: The magic of Karnaugh Maps

Read the terms from the circled items, leaving out variables that have different values within the group.



(Because A is same, C is same, but B = both 1 & 0)

Computer Science Foundations

Another Look at the Map

Values for variables are listed so that only one change of value occurs between neighbors. ("Gray code")

AB

C		00	01	11	10
	0				
	1				

With 4 variables ⇒ 4 rows, 4 columns.

Computer Science Foundations

What to Circle

- Circle: groups of size 2^n, $n > 0$
- Don't have to circle groups of 1
  - implicit circles
  - must remember to include them in minimized expression, though!
- Circle largest group possible to cover each 1
  - Larger groups $\Rightarrow$ fewer terms
  - Group of 2^n: n inputs are eliminated
- A 1 can be in > 1 group:
  - May be needed to increase size of multiple groups
  - Each group: must have at least one 1 not in any other group
- Circles can "wrap around" map:
  - side to side, top to bottom
  - all 4 corners

Example: Another Karnaugh Map

Design a minimal circuit for the following expression:

$$\overline{A}B\overline{C}D + ABCD + \overline{A}BCD + AB\overline{C}D$$
$$+ ABC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$$

Example: Another Karnaugh Map

Design a minimal circuit for the following expression:

$$\overline{A}B\overline{C}D + ABCD + \overline{A}BCD + AB\overline{C}D$$
$$+ ABC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$$

Draw the Karnaugh map and add the values:

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

Example: Another Karnaugh Map

Circle the groups and read the terms for the minimal circuit.

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

Example: Another Karnaugh Map

Circle the groups and read the terms for the minimal circuit.

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

Example: Another Karnaugh Map

Circle the groups and read the terms for the minimal circuit.

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

$$BD + A\overline{B}\,\overline{D} + AC\overline{D}$$

Computer Science Foundations

Including Don't Cares

- Put "Don't Cares" in Karnaugh Map as D
- Include them only in circles if it helps

Computer Science Foundations

Example

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	–
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	–
1	1	1	0	1
1	1	1	1	D

CD \ AB	00	01	11	10
00	1			D
01			D	
11		1	D	
10		1	1	

Example

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	–
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	–
1	1	1	0	1
1	1	1	1	D

AB

CD	00	01	11	10
00	1			D
01			D	
11		1	D	
10		1	1	

Example

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	–
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	–
1	1	1	0	1
1	1	1	1	D

AB

CD	00	01	11	10
00	1			D
01			D	
11		1	D	
10		1	1	

Example

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	–
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	–
1	1	1	0	1
1	1	1	1	D

AB

CD \ AB	00	01	11	10
00	1			D
01			D	
11		1	D	
10		1	1	

$$BC + \overline{B}\,\overline{C}\,\overline{D}$$

Computer Science Foundations

Advantages and Limitations of Karnaugh Maps

- Pros:
  - Easy to work with
  - Handles don't cares – no need to manipulate algebraic expression (as some other methods do)
- Cons:
  - Not meant for automation
  - Difficult to use with > 4 variables
    - 5 or 6 variables: map is cube
    - Handle by overlaying tables, but hard to visualize
    - > 6: hypercube

Computer Science Foundations

More about Karnaugh Maps and Minimizing Circuits

- Can be used for functions other than SOPs – map is read differently
- Other methods exist that can be automated:
  - Work with more variables
  - E.g., Quine-McKluskey Method
  - But QM is NP-hard (i.e., intractable for many-variable functions)

Computer Science Foundations