

Lecture 4

- ◆ Logistics
 - HW1 due Wednesday at start of class
 - Office Hours:
 - ◊ Me: 12:20-1:00 CSE 668 plus one later this week
 - ◊ TAs: Today at 3:30, tomorrow at 12:30 & 2:30 in CSE 220
 - Lab2 going on this week
- ◆ Last lecture --- Boolean algebra
 - Axioms
 - Useful laws and theorems
 - Simplifying Boolean expressions
- ◆ Today's lecture
 - Logic gates and truth tables in detail
 - Implementing logic functions
 - Canonical forms

CSE370, Lecture 4

1

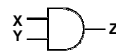
The "WHY" slide


- ◆ Logic Gates and Truth Tables
 - Now you know 0's and 1's and the basic Boolean algebra, now you are ready to go back and forth between truth table, Boolean expression, and logic gates. This ability to go back and forth is an extremely useful skill designing and optimizing computer hardware.
- ◆ Implementing Logic Functions
 - Now with these basic tools you learned, you can "implement" logic functions. We use Boolean algebra to implement logic functions that are used in the computers. And these logic functions are used by computer programs you write.
- ◆ Canonical forms
 - There are many forms to expression one Boolean function. It is good to have one standard way. A canonical form is the standard form for Boolean expressions. It has a nice property that allows you to go back and forth between truth table/expressions/gates easily.

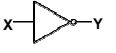
CSE370, Lecture 4

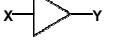
2

Logic gates and truth tables

- ◆ AND $X \cdot Y$ XY


X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1
- ◆ OR $X + Y$


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1
- ◆ NOT \bar{X} X'


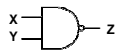
X	Y
0	1
1	0
- ◆ Buffer X



X	Y
0	0
1	1


CSE370, Lecture 4


3

Logic gates and truth tables (con't)

- ◆ NAND $\overline{X \cdot Y}$ \overline{XY}


X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0
- ◆ NOR $\overline{X + Y}$


X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0
- ◆ XOR $X \oplus Y$


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0
- ◆ XNOR $\overline{X \oplus Y}$


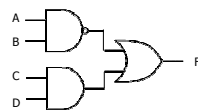
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

CSE370, Lecture 4

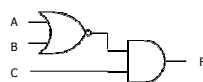
4

Boolean expressions \Leftrightarrow logic gates

- ◆ Example: $F = (A \cdot B)' + C \cdot D$



- ◆ Example: $F = C \cdot (A + B)'$



CSE370, Lecture 4

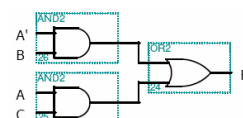
5

Truth tables \Leftrightarrow logic gates

- ◆ Given a truth table
 - Write the Boolean expression
 - Minimize the Boolean expression
 - Draw as gates
 - Example:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$\begin{aligned}
 F &= A'BC' + A'BC + AB'C + ABC \\
 &= A'B(C' + C) + AC(B' + B) \\
 &= A'B + AC
 \end{aligned}$$



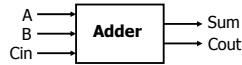
CSE370, Lecture 4

6

Example: A binary full adder

◆ 1-bit binary adder

- Inputs: A, B, Carry-in
- Outputs: Sum, Carry-out



A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\text{Sum} = A'B'Cin + A'BCin' + AB'Cin' + ABCin$$

$$\text{Cout} = A'BCin + AB'Cin + ABCin' + ABCin$$

Both Sum and Cout can be minimized.

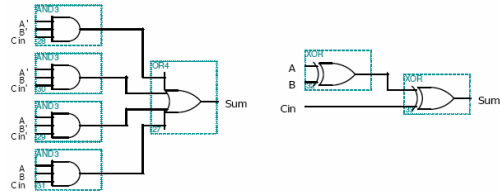
Full adder: Sum

Before Boolean minimization

$$\text{Sum} = A'B'Cin + A'BCin' + AB'Cin' + ABCin$$

After Boolean minimization

$$\text{Sum} = (A \oplus B) \oplus Cin$$



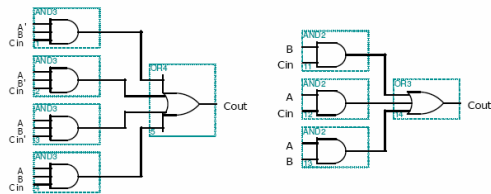
Full adder: Carry-out

Before Boolean minimization

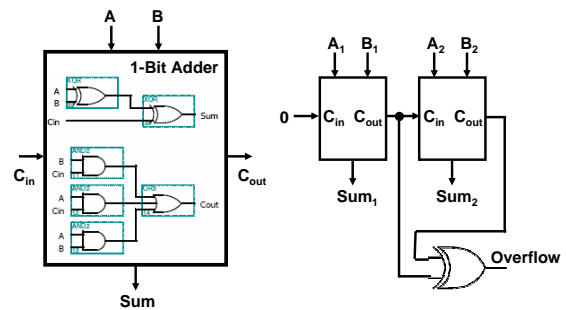
$$\text{Cout} = A'BCin + AB'Cin + ABCin' + ABCin$$

After Boolean minimization

$$\text{Cout} = BCin + ACin + AB$$



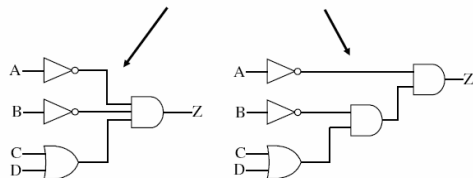
Preview: A 2-bit ripple-carry adder



Many possible mappings

◆ Many ways to map expressions to gates

- Example: $Z = \bar{A} \cdot \bar{B} \cdot (C + D) = \bar{A} \cdot \bar{B} \cdot (C + D)$



What is the optimal gate realization?

- ◆ We use the axioms and theorems of Boolean algebra to "optimize" our designs

◆ Design goals vary

- Reduce the number of gates?
- Reduce the number of gate inputs?
- Reduce number of chips and/or wire?

◆ How do we explore the tradeoffs?

- Logic minimization: Reduce number of gates and complexity
- Logic optimization: Maximize speed and/or minimize power
- CAD tools

Minimal set

- ◆ We can implement any logic function from NOT, NOR, and NAND

- Example: $(X \text{ and } Y) = \text{not } (X \text{ nand } Y)$

- ◆ In fact, we can do it with only NOR or only NAND

- NOT is just NAND or NOR with two identical inputs

X	Y	X nor Y	X	Y	X nand Y
0	0	1	0	0	1
1	1	0	1	1	0

- NAND and NOR are duals: Can implement one from the other

- ⇨ $X \text{ nand } Y = \text{not } ((\text{not } X) \text{ nor } (\text{not } Y))$

- ⇨ $X \text{ nor } Y = \text{not } ((\text{not } X) \text{ nand } (\text{not } Y))$

Canonical forms

- ◆ Canonical forms

- Standard forms for Boolean expressions

- Generally not the simplest forms

- ⇨ Can be minimized

- Derived from truth table

- ◆ Two canonical forms

- Sum-of-products (minterms)

- Product-of-sum (maxterms)

Sum-of-products canonical form (SOP)

- ◆ Also called disjunctive normal form (DNF)

- Commonly called a **minterm expansion**

A	B	C	F	F'	
0	0	0	0	1	
0	0	1	1	0	
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	0	

$F = A'B'C' + A'BC' + AB'C' + ABC'$

$F' = A'B'C' + A'BC' + AB'C'$

Minterms

- ◆ Variables appear exactly once in each minterm

- In true or inverted form (but not both)

A	B	C	minterms
0	0	0	A'B'C' m0
0	0	1	A'B'C m1
0	1	0	A'BC' m2
0	1	1	A'BC m3
1	0	0	AB'C' m4
1	0	1	AB'C m5
1	1	0	ABC' m6
1	1	1	ABC m7

F in canonical form:

$$F(A,B,C) = \Sigma m(1,3,5,6,7)$$

$$= m1 + m3 + m5 + m6 + m7$$

$$= A'B'C + A'BC + AB'C + ABC + ABC$$

canonical form → minimal form

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

$$= AB + C$$

short-hand notation

Product-of-sums canonical form (POS)

- ◆ Also called conjunctive normal form (CNF)

- Commonly called a **maxterm expansion**

A	B	C	F	F'	
0	0	0	0	1	
0	0	1	1	0	
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	0	

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

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

Maxterms

- ◆ Variables appears exactly once in each maxterm

- In true or inverted form (but not both)

A	B	C	maxterms
0	0	0	A+B+C M0
0	0	1	A+B+C' M1
0	1	0	A+B'+C M2
0	1	1	A+B'+C' M3
1	0	0	A'+B+C M4
1	0	1	A'+B+C' M5
1	1	0	A'+B'+C M6
1	1	1	A'+B'+C' M7

F in canonical form:

$$F(A,B,C) = \Pi M(0,2,4)$$

$$= M0 \cdot M2 \cdot M4$$

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

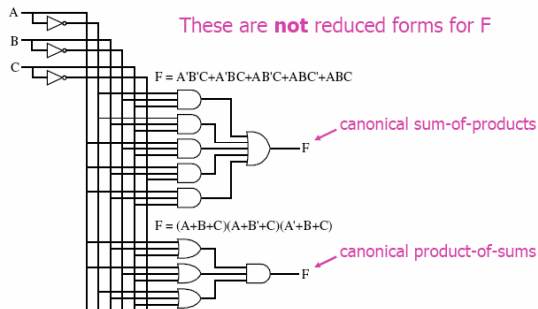
canonical form → minimal form

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

$$= AB + C$$

short-hand notation

Canonical implementations of $F = AB + C$



CSE370, Lecture 4

19

Conversion between canonical forms

- ◆ Minterm to maxterm
 - Use maxterms that aren't in minterm expansion
 - $F(A,B,C) = \sum m(1,3,5,6,7) = \prod M(0,2,4)$
- ◆ Maxterm to minterm
 - Use minterms that aren't in maxterm expansion
 - $F(A,B,C) = \prod M(0,2,4) = \sum m(1,3,5,6,7)$
- ◆ Minterm of F to minterm of F'
 - Use minterms that don't appear
 - $F(A,B,C) = \sum m(1,3,5,6,7)$ $F'(A,B,C) = \sum m(0,2,4)$
- ◆ Maxterm of F to maxterm of F'
 - Use maxterms that don't appear
 - $F(A,B,C) = \prod M(0,2,4)$ $F'(A,B,C) = \prod M(1,3,5,6,7)$

CSE370, Lecture 4

20

SOP, POS, and de Morgan's theorem

- ◆ Sum-of-products
 - $F' = A'B'C' + A'BC' + AB'C'$
- ◆ Apply de Morgan's to get POS
 - $(F')' = (A'B'C' + A'BC' + AB'C')'$
 - $F = (A+B+C)(A+B'+C)(A'+B+C)$
- ◆ Product-of-sums
 - $F' = (A+B+C')(A+B'+C')(A'+B+C')(A'+B'+C')$
- ◆ Apply de Morgan's to get SOP
 - $(F')' = ((A+B+C')(A+B'+C')(A'+B+C')(A'+B'+C'))'$
 - $F = A'B'C' + A'BC' + AB'C' + ABC' + ABC$

CSE370, Lecture 4

21