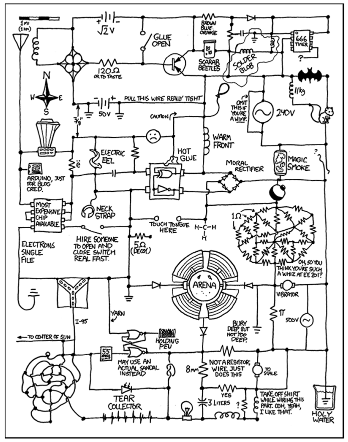
CSE 311: Foundations of Computing

Fall 2013

Lecture 4: Boolean algebra and circuits



announcements

Reading assignment

- **Boolean Algebra**
  - 12.1 – 12.3 7th Edition
  - 11.1 – 11.3 6th Edition

Homework 1 due today

- **Hand in at start of class**

Homework 2 available online later today

review: a quick combinational logic example

Calendar subsystem:

# of days in a month (to control watch display)

- used in controlling the display of a wrist-watch LCD screen

- **inputs:** month, leap year flag
- **outputs:** number of days

Example: (March, non-leap year) → **31**

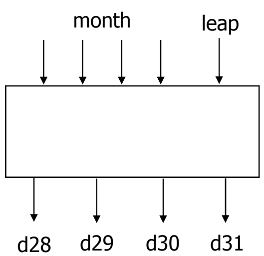
review: implementation in software

```
integer number_of_days (month, leap_year_flag){
   switch (month) {
      case 1: return (31);
      case 2: if (leap_year_flag == 1) then
               return (29) else return (28);
      case 3: return (31);
      ...
      case 12: return (31);
      default: return (0);
   }
}
```

review: implementation with combinatorial logic

Encoding:

- how many bits for each input/output?
- binary number for month
- four wires for 28, 29, 30, and 31



month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
1110	–	–	–	–	–
1111	–	–	–	–	–

review: implementation with combinatorial logic

Truth-table to logic to switches to gates

d28 = "1 when month=0010 and leap=0"

d28 = m8' • m4' • m2 • m1' • leap'

d31 = "1 when month=0001 or month=0011 or ... month=1100"

d31 = (m8' • m4' • m2' • m1) + (m8' • m4' • m2 • m1) +
... + (m8 • m4 • m2' • m1')

d31 = can we simplify more?

month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
...					
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

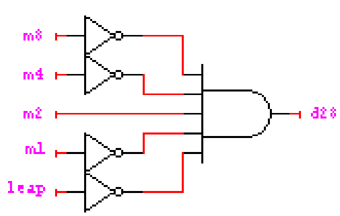
review: implementation with combinatorial logic

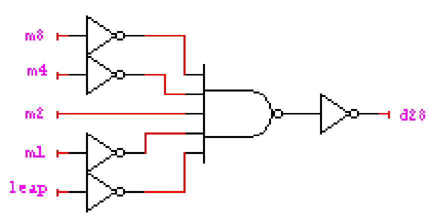
d28 = m8' • m4' • m2 • m1' • leap'

d29 = m8' • m4' • m2 • m1' • leap

d30 = (m8' • m4 • m2' • m1') + (m8' • m4 • m2 • m1') +
(m8 • m4' • m2' • m1) + (m8 • m4' • m2 • m1)
= (m8' • m4 • m1') + (m8 • m4' • m1)

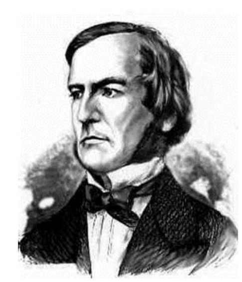
d31 = (m8' • m4' • m2' • m1) + (m8' • m4' • m2 • m1) +
(m8' • m4 • m2' • m1) + (m8' • m4 • m2 • m1) +
(m8 • m4' • m2' • m1') + (m8 • m4' • m2 • m1') +
(m8 • m4 • m2' • m1')





boolean algebra

- **Boolean algebra to circuit design**
- **Boolean algebra**
  - a set of elements B containing {0, 1}
  - binary operations { + , • }
  - and a unary operation { ' }
  - such that the following axioms hold:



1. the set B contains at least two elements: a, b		
2. closure:	a + b is in B	a • b is in B
3. commutativity:	a + b = b + a	a • b = b • a
4. associativity:	a + (b + c) = (a + b) + c	a • (b • c) = (a • b) • c
5. identity:	a + 0 = a	a • 1 = a
6. distributivity:	a + (b • c) = (a + b) • (a + c)	a • (b + c) = (a • b) + (a • c)
7. complementarity:	a + a' = 1	a • a' = 0

axioms and theorems of Boolean algebra

identity:
1. $X + 0 = X$ — 1D. $X \bullet 1 = X$

null:
2. $X + 1 = 1$ — 2D. $X \bullet 0 = 0$

idempotency:
3. $X + X = X$ — 3D. $X \bullet X = X$

involution:
4. $(X')' = X$

complementarity:
5. $X + X' = 1$ — 5D. $X \bullet X' = 0$

commutatively:
6. $X + Y = Y + X$ — 6D. $X \bullet Y = Y \bullet X$

associativity:
7. $(X + Y) + Z = X + (Y + Z)$ — 7D. $(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)$

distributivity:
8. $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$ — 8D. $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$

axioms and theorems of Boolean algebra

uniting:
9. $X \bullet Y + X \bullet Y' = X$ — 9D. $(X + Y) \bullet (X + Y') = X$

absorption:
10. $X + X \bullet Y = X$ — 10D. $X \bullet (X + Y) = X$
11. $(X + Y') \bullet Y = X \bullet Y$ — 11D. $(X \bullet Y') + Y = X + Y$

factoring:
12. $(X + Y) \bullet (X' + Z) = X \bullet Z + X' \bullet Y$ — 12D. $X \bullet Y + X' \bullet Z = (X + Z) \bullet (X' + Y)$

consensus:
13. $(X \bullet Y) + (Y \bullet Z) + (X' \bullet Z) = X \bullet Y + X' \bullet Z$ — 13D. $(X + Y) \bullet (Y + Z) \bullet (X' + Z) = (X + Y) \bullet (X' + Z)$

de Morgan's:
14. $(X + Y + \ldots)' = X' \bullet Y' \bullet \ldots$ — 14D. $(X \bullet Y \bullet \ldots)' = X' + Y' + \ldots$

proving theorems (rewriting)

Using the laws of Boolean algebra:

prove the theorem: $X \bullet Y + X \bullet Y' = X$

distributivity (8)	$X \bullet Y + X \bullet Y' = X \bullet (Y + Y')$
complementarity (5)	$= X \bullet (1)$
identity (1D)	$= X$

prove the theorem: $X + X \bullet Y = X$

identity (1D)	$X + X \bullet Y = X \bullet 1 + X \bullet Y$
distributivity (8)	$= X \bullet (1 + Y)$
uniting (2)	$= X \bullet (1)$
identity (1D)	$= X$

proving theorems (truth table)

Using complete truth table:

For example, de Morgan's Law:

$(X + Y)' = X' \bullet Y'$
NOR is equivalent to AND with inputs complemented

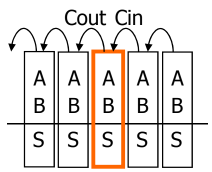
X	Y	X′	Y′	(X + Y)′	X′ • Y′
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

$(X \bullet Y)' = X' + Y'$
NAND is equivalent to OR with inputs complemented

X	Y	X′	Y′	(X • Y)′	X′ + Y′
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

a simple example: 1-bit binary adder

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



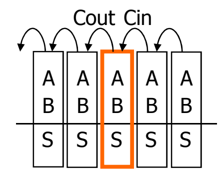
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A'\,B'\,\text{Cin} + A'\,B\,\text{Cin}' + A\,B'\,\text{Cin}' + A\,B\,\text{Cin}$$

$$\text{Cout} = A'\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin}$$

a simple example: 1-bit binary adder

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



$$\text{Cout} = A'\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin}$$

apply theorems to simplify expressions

The theorems of Boolean algebra can simplify expressions

– **e.g., full adder's carry-out function**

$$\begin{aligned}
\text{Cout} &= A'\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= A'\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + \boxed{A\,B\,\text{Cin} + A\,B\,\text{Cin}} \\
&= A'\,B\,\text{Cin} + A\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= (A' + A)\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= (1)\,B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin}' + \boxed{A\,B\,\text{Cin} + A\,B\,\text{Cin}} \\
&= B\,\text{Cin} + A\,B'\,\text{Cin} + A\,B\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= B\,\text{Cin} + A\,(B' + B)\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= B\,\text{Cin} + A\,(1)\,\text{Cin} + A\,B\,\text{Cin}' + A\,B\,\text{Cin} \\
&= B\,\text{Cin} + A\,\text{Cin} + A\,B\,(\text{Cin}' + \text{Cin}) \\
&= B\,\text{Cin} + A\,\text{Cin} + A\,B\,(1) \\
&= B\,\text{Cin} + A\,\text{Cin} + A\,B
\end{aligned}$$

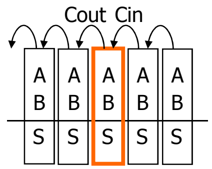
adding extra terms creates new factoring opportunities

rewrite using xor gates

$$S = A'\,B'\,\text{Cin} + A'\,B\,\text{Cin}' + A\,B'\,\text{Cin}' + A\,B\,\text{Cin}$$

a simple example: 1-bit binary adder

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



A	B	Cin	Cout	S
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

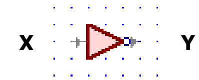
Cout = A′ B Cin + A B′ Cin + A B Cin′ + A B Cin
S = A′ B′ Cin + A′ B Cin′ + A B′ Cin′ + A B Cin

Cout = B Cin + A Cin + A B
S = A xor (B xor Cin)

recall gates

NOT

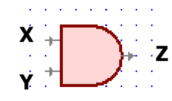
$X' \quad \bar{X} \quad \neg X$



X	Y
0	1
1	0

AND

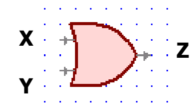
$X \cdot Y \quad XY \quad X \wedge Y$



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

OR

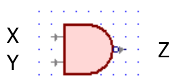
$X + Y \quad X \vee Y$



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

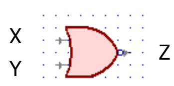
some other gates

NAND
$\neg(X \wedge Y)$



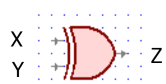
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR
$\neg(X \vee Y)$



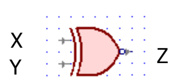
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

XOR
$X \oplus Y$



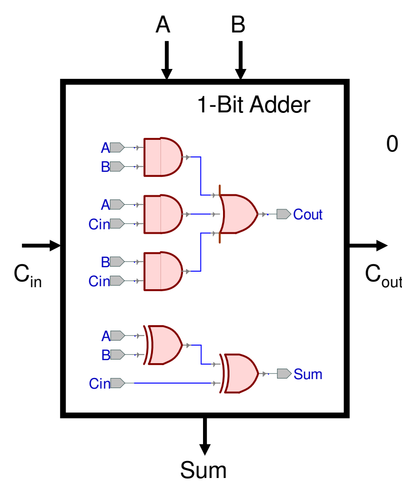
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

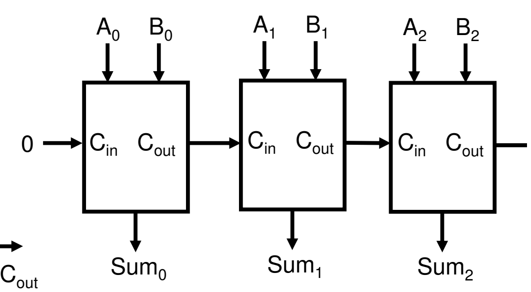
XNOR
$X \leftrightarrow Y, X = Y$



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

a 2-bit ripple-carry adder





mapping truth tables to logic gates

Given a truth table:

1. Write the Boolean expression
2. Minimize the Boolean expression
3. Draw as gates
4. Map to available gates

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

1

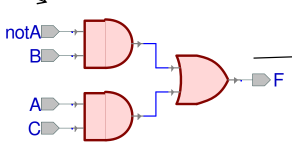
F = A'BC'+A'BC+AB'C+ABC

2

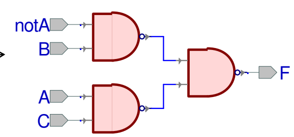
= A'B(C'+C)+AC(B'+B)

= A'B+AC

3



4



canonical forms

- Truth table is the unique signature of a Boolean function
- The same truth table can have many gate realizations
  - –we've seen this already
  - –depends on how good we are at Boolean simplification
- Canonical forms
  - – standard forms for a Boolean expression
  - – we all come up with the same expression

sum-of-products canonical form

- also known as Disjunctive Normal Form (DNF)
- also known as minterm expansion

F = *001* *011* *101* *110* *111*

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

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

sum-of-products canonical form

Product term (or minterm)

- – ANDed product of literals – input combination for which output is true
- – each variable appears exactly once, true or inverted (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

short-hand notation for minterms of 3 variables

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'

= (A'B' + A'B + AB' + AB)C + ABC'

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

= C + ABC'

= ABC' + C

= AB + C

product-of-sums canonical form

- Also known as Conjunctive Normal Form (CNF)
- Also known as maxterm expansion

F = *000* *010* *100*

F = (A + B + C) (A + B′ + C) (A′ + B + C)

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

product-of-sums canonical form

Sum term (or maxterm)

- ORed sum of literals – input combination for which output is false
- each variable appears exactly once, true or inverted (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

short-hand notation for maxterms of 3 variables

F in canonical form:

F(A, B, C) = ΠM(0,2,4)
= M0 • M2 • 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)
= (A + B + C) (A + B′ + C)
(A + B + C) (A′ + B + C)
= (A + C) (B + C)