

**CSE
31F**

**Foundations of
Computing I**

Pre-Lecture Problem

Do it! Do it now! What are you waiting for? 😊

Use Logical Equivalences to show

$$p \wedge ((p \rightarrow q) \vee (p \rightarrow r)) \equiv (r \vee q) \wedge p$$

Identity

$$p \wedge \top \equiv p$$

$$p \vee \text{F} \equiv p$$

Domination

$$p \vee \top \equiv \top$$

$$p \wedge \text{F} \equiv \text{F}$$

Idempotency

$$p \vee p \equiv p$$

$$p \wedge p \equiv p$$

Commutativity

$$p \vee q \equiv q \vee p$$

$$p \wedge q \equiv q \wedge p$$

Associativity

$$(p \vee q) \vee r \equiv p \vee (q \vee r)$$

$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$$

Distributivity

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$

Absorption

$$p \vee (p \wedge q) \equiv p$$

$$p \wedge (p \vee q) \equiv p$$

Negation

$$p \vee \neg p \equiv \top$$

$$p \wedge \neg p \equiv \text{F}$$

DeMorgan's Laws

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

Double Negation

$$\neg\neg p \equiv p$$

Law of Implication

$$p \rightarrow q \equiv \neg p \vee q$$

Contrapositive

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$

A Combinational Logic Example

Sessions of Class:

We would like to compute the number of lectures or quiz sections remaining *at the start* of a given day of the week.

- **Inputs:** Day of the Week, Lecture/Section flag
- **Output:** Number of sessions left

Examples: Input: (Wednesday, Lecture) Output: **2**

Input: (Monday, Section) Output: **1**

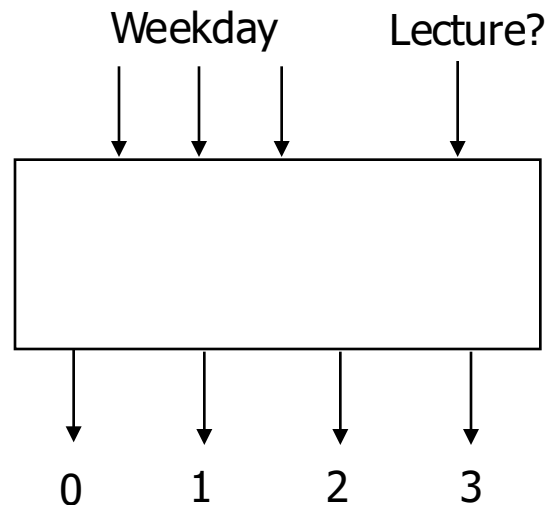
Implementation in Software

```
public int classesLeftInMorning(weekday, lecture_flag) {
    switch (weekday) {
        case SUNDAY:
        case MONDAY:
            return lecture_flag ? 3 : 1;
        case TUESDAY:
        case WEDNESDAY:
            return lecture_flag ? 2 : 1;
        case THURSDAY:
            return lecture_flag ? 1 : 1;
        case FRIDAY:
            return lecture_flag ? 1 : 0;
        case SATURDAY:
            return lecture_flag ? 0 : 0;
    }
}
```

Implementation with Combinational Logic

Encoding:

- How many bits for each input/output?
- Binary number for weekday
- One bit for each possible output



Defining Our Inputs!

Weekday Input:

- Binary number for weekday
- Sunday = 0, Monday = 1, ...
- We care about these in binary:

Weekday	Number	Binary
Sunday	0	(000) ₂
Monday	1	(001) ₂
Tuesday	2	(010) ₂
Wednesday	3	(011) ₂
Thursday	4	(100) ₂
Friday	5	(101) ₂
Saturday	6	(110) ₂

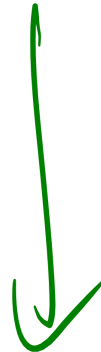
Converting to a Truth Table!

```

case SUNDAY or MONDAY:
    return lecture_flag ? 3 : 1;
case TUESDAY or WEDNESDAY:
    return lecture_flag ? 2 : 1;
case THURSDAY:
    return lecture_flag ? 1 : 1;
case FRIDAY:
    return lecture_flag ? 1 : 0;
case SATURDAY:
    return lecture_flag ? 0 : 0;

```

Weekday	Lecture?	c ₀	c ₁	c ₂	c ₃
SUN	000	0	1	0	0
SUN	000	0	0	0	1
MON	001				
MON	001				
TUE	010				
TUE	010				
WED	011				
WED	011				
THU	100	0	1	0	0
FRI	101				
FRI	101				
SAT	110				
-	111				



Converting to a Truth Table!

```
case SUNDAY or MONDAY:
    return lecture_flag ? 3 : 1;
case TUESDAY or WEDNESDAY:
    return lecture_flag ? 2 : 1;
case THURSDAY:
    return lecture_flag ? 1 : 1;
case FRIDAY:
    return lecture_flag ? 1 : 0;
case SATURDAY:
    return lecture_flag ? 0 : 0;
```

Weekday	Lecture?	c_0	c_1	c_2	c_3
SUN	000	0	1	0	0
SUN	000	1	0	0	1
MON	001	0	1	0	0
MON	001	1	0	0	1
TUE	010	0	1	0	0
TUE	010	1	0	1	0
WED	011	0	1	0	0
WED	011	1	0	1	0
THU	100	-	1	0	0
FRI	101	0	1	0	0
FRI	101	1	0	0	0
SAT	110	-	1	0	0
-	111	-	1	0	0

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Let's begin by finding an expression for c_3 . To do this, we look at the rows where $c_3 = 1$ (true).

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

e_1
DAY == SUN && L == 1

DAY == MON && L == 1

e_2

$e_1 \vee e_2$

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2d_1d_0 == 000 \ \&\& \ L == 1$

$d_2d_1d_0 == 001 \ \&\& \ L == 1$

Substituting DAY for the binary representation.

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2 == 0 \ \&\& \ d_1 == 0 \ \&\& \ d_0 == 0 \ \&\& \ L == 1$

$d_2 == 0 \ \&\& \ d_1 == 0 \ \&\& \ d_0 == 1 \ \&\& \ L == 1$

Splitting up the bits of the day;
so, we can write a formula.

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2' \cdot d_1' \cdot d_0' \cdot L$

$d_2' \cdot d_1' \cdot d_0 \cdot L$

Replacing with Boolean Algebra...

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2' \cdot d_1' \cdot d_0' \cdot L$

$d_2' \cdot d_1' \cdot d_0 \cdot L$

Either situation causes c_3 to be true. So, we "or" them.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 2)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Now, we do c_2 .

$$d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

Now, we do c_1 :

	$d_2 d_1 d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	$d_2' \cdot d_1' \cdot d_0' \cdot L'$
SUN	000	1	0	0	0	1	
MON	001	0	0	1	0	0	$d_2' \cdot d_1' \cdot d_0 \cdot L'$
MON	001	1	0	0	0	1	
TUE	010	0	0	1	0	0	$d_2' \cdot d_1 \cdot d_0' \cdot L'$
TUE	010	1	0	0	1	0	
WED	011	0	0	1	0	0	$d_2' \cdot d_1 \cdot d_0 \cdot L'$
WED	011	1	0	0	1	0	
THU	<u>100</u>	⊖	0	1	0	0	??? $(d_2 \cdot d_1' \cdot d_0) \cdot (L \cdot L')$
FRI	101	0	1	0	0	0	
FRI	101	1	0	1	0	0	$d_2 \cdot d_1' \cdot d_0 \cdot L$
SAT	110	-	1	0	0	0	
-	111	-	1	0	0	0	

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	$d_2' \cdot d_1' \cdot d_0' \cdot L'$
SUN	000	1	0	0	0	1	
MON	001	0	0	1	0	0	$d_2' \cdot d_1' \cdot d_0 \cdot L'$
MON	001	1	0	0	0	1	
TUE	010	0	0	1	0	0	$d_2' \cdot d_1 \cdot d_0' \cdot L'$
TUE	010	1	0	0	1	0	
WED	011	0	0	1	0	0	$d_2' \cdot d_1 \cdot d_0 \cdot L'$
WED	011	1	0	0	1	0	
THU	100	-	0	1	0	0	$d_2 \cdot d_1' \cdot d_0'$
FRI	101	0	1	0	0	0	
FRI	101	1	0	1	0	0	$d_2 \cdot d_1' \cdot d_0 \cdot L$
SAT	110	-	1	0	0	0	
-	111	-	1	0	0	0	

Now, we do c_1 :

No matter what L is, we always say it's 1. So, we don't need L in the expression.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	$d_2' \cdot d_1' \cdot d_0' \cdot L'$
SUN	000	1	0	0	0	1	
MON	001	0	0	1	0	0	$d_2' \cdot d_1' \cdot d_0 \cdot L'$
MON	001	1	0	0	0	1	
TUE	010	0	0	1	0	0	$d_2' \cdot d_1 \cdot d_0' \cdot L'$
TUE	010	1	0	0	1	0	
WED	011	0	0	1	0	0	$d_2' \cdot d_1 \cdot d_0 \cdot L'$
WED	011	1	0	0	1	0	
THU	100	-	0	1	0	0	$d_2 \cdot d_1' \cdot d_0'$
FRI	101	0	1	0	0	0	
FRI	101	1	0	1	0	0	$d_2 \cdot d_1' \cdot d_0 \cdot L$
SAT	110	-	1	0	0	0	
-	111	-	1	0	0	0	

Now, we do c_1 :

No matter what L is,
we always say it's 1.
So, we don't need L
in the expression.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

$$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 4)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Finally, we do c_0 :

$$d_2 \cdot d_1' \cdot d_0 \cdot L'$$

$$d_2 \cdot d_1 \cdot d_0'$$

$$d_2 \cdot d_1 \cdot d_0$$

Truth Table to Logic (Part 4)

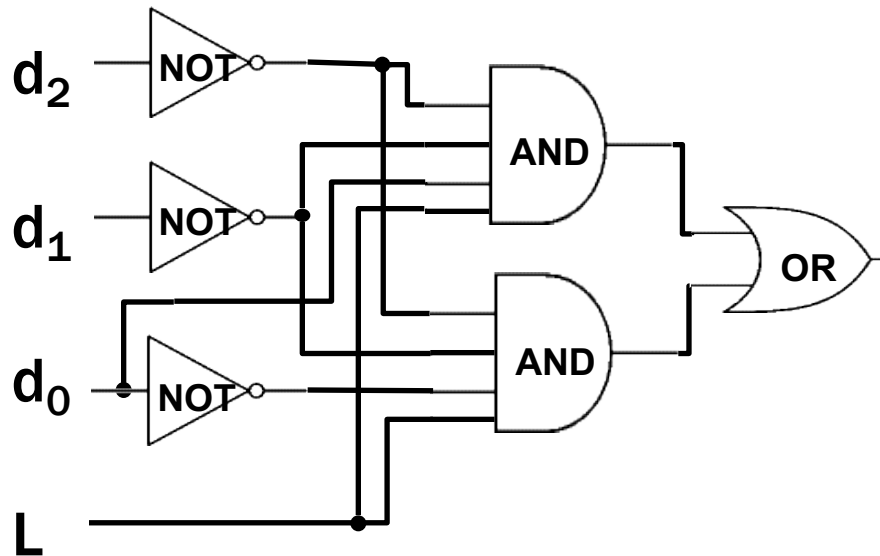
$$c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1 \cdot d_0$$

$$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

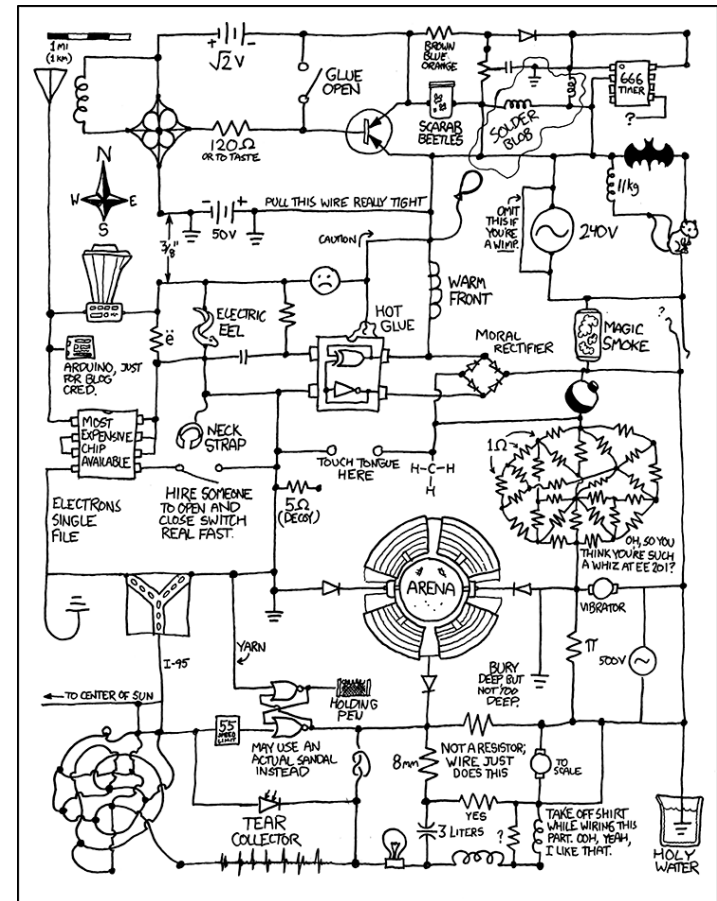
$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Here's c_3 as a circuit:



CSE 311: Foundations of Computing

Lecture 4: Boolean Algebra, Circuits, Canonical Forms



Boolean Algebra

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



1. the set B contains at least two elements: $0, 1$

For any a, b, c in B :

- | | | |
|---------------------|---|---|
| 2. closure: | $a + b$ is in B | $a \cdot b$ is in B |
| 3. commutativity: | $a + b = b + a$ | $a \cdot b = b \cdot a$ |
| 4. associativity: | $a + (b + c) = (a + b) + c$ | $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ |
| 5. identity: | $a + 0 = a$ | $a \cdot 1 = a$ |
| 6. distributivity: | $a + (b \cdot c) = (a + b) \cdot (a + c)$ | $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ |
| 7. complementarity: | $a + a' = 1$ | $a \cdot a' = 0$ |

Axioms and Theorems of Boolean Algebra

identity:

$$1. X + 0 = X$$

$$1D. X \cdot 1 = X$$

null:

$$2. X + 1 = 1$$

$$2D. X \cdot 0 = 0$$

idempotency:

$$3. X + X = X$$

$$3D. X \cdot X = X$$

involution:

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

complementarity:

$$5. X + X' = 1$$

$$5D. X \cdot X' = 0$$

commutativity:

$$6. X + Y = Y + X$$

$$6D. X \cdot Y = Y \cdot X$$

associativity:

$$7. (X + Y) + Z = X + (Y + Z)$$

$$7D. (X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$

distributivity:

$$8. X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$$

$$8D. X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$$

Axioms and Theorems of Boolean Algebra

uniting:

$$9. X \cdot Y + X \cdot Y' = X$$

$$9D. (X + Y) \cdot (X + Y') = X$$

absorption:

$$10. X + X \cdot Y = X$$

$$10D. X \cdot (X + Y) = X$$

$$11. (X + Y') \cdot Y = X \cdot Y$$

$$11D. (X \cdot Y') + Y = X + Y$$

factoring:

$$12. (X + Y) \cdot (X' + Z) = \\ X \cdot Z + X' \cdot Y$$

$$12D. X \cdot Y + X' \cdot Z = \\ (X + Z) \cdot (X' + Y)$$

consensus:

$$13. (X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = \\ X \cdot Y + X' \cdot Z$$

$$13D. (X + Y) \cdot (Y + Z) \cdot (X' + Z) = \\ (X + Y) \cdot (X' + Z)$$

de Morgan's:

$$14. (X + Y + \dots)' = X' \cdot Y' \cdot \dots$$

$$14D. (X \cdot Y \cdot \dots)' = X' + Y' + \dots$$

Proving Theorems (Rewriting)

Using the laws of Boolean Algebra:

prove the theorem:

$$X \cdot Y + X \cdot Y' = X$$

$$\begin{aligned} X \cdot Y + X \cdot Y' &= X \cdot (\cancel{Y} + \cancel{Y}') \\ &= X \cdot 1 \\ &= X \end{aligned}$$

prove the theorem:

$$X + X \cdot Y = X$$

$$X + X \cdot Y =$$

Proving Theorems (Rewriting)

Using the laws of Boolean Algebra:

prove the theorem:

$$X \cdot Y + X \cdot Y' = X$$

distributivity (8)

$$X \cdot Y + X \cdot Y' = X \cdot (Y + Y')$$

complementarity (5)

$$= X \cdot (1)$$

identity (1D)

$$= X$$

prove the theorem:

$$X + X \cdot Y = X$$

identity (1D)

$$X + X \cdot Y = X \cdot 1 + X \cdot Y$$

distributivity (8)

$$= X \cdot (1 + Y)$$

uniting (2)

$$= X \cdot (1)$$

identity (1D)

$$= X$$

Proving Theorems (Truth Table)

Using complete truth table:

For example, de Morgan's Law:

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

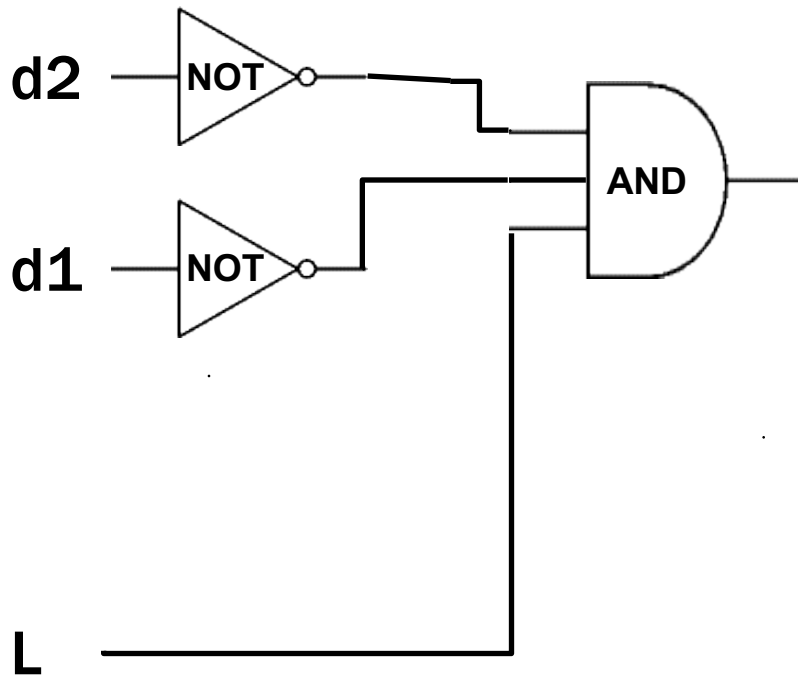
X	Y	X'	Y'	$(X + Y)'$	$X' \cdot Y'$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

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

X	Y	X'	Y'	$(X \cdot Y)'$	$X' + Y'$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Simplifying using Boolean Algebra

$$\begin{aligned}c3 &= (d2' \cdot d1' \cdot d0' \cdot L) + (d2' \cdot d1' \cdot d0 \cdot L) \\ &= d2' \cdot d1' \cdot (d0' + d0) \cdot L \\ &= d2' \cdot d1' \cdot (1) \cdot L \\ &= d2' \cdot d1' \cdot L\end{aligned}$$



1-bit Binary Adder

Let's make a circuit that adds three single bit inputs together into a single binary number

A	0	1	1	1
+ B	+ 0	+ 1	+ 0	+ 1
+ C	+ 0	+ 0	+ 1	+ 1
<u> </u>	<u> </u>	<u> </u>	<u> </u>	<u> </u>
S _c S	00	10	10	11

1-bit Binary Adder

Let's make a circuit that adds three single bit inputs together into a single binary number

A	0	1	1	1
+ B	+ 0	+ 1	+ 0	+ 1
<u>+ C</u>	<u>+ 0</u>	<u>+ 0</u>	<u>+ 1</u>	<u>+ 1</u>
S _C S	00	10	10	11

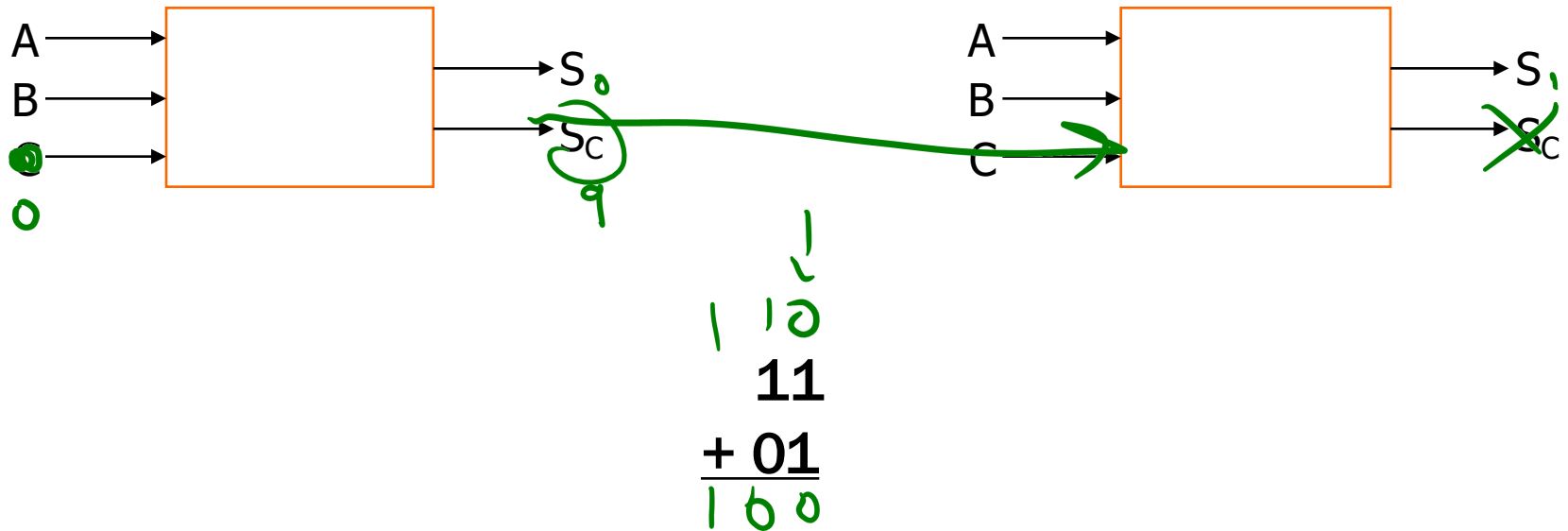
1-bit Binary Adder

- **Inputs:** A, B, C
- **Outputs:** Two-bit Sum

A	B	C	S_C	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

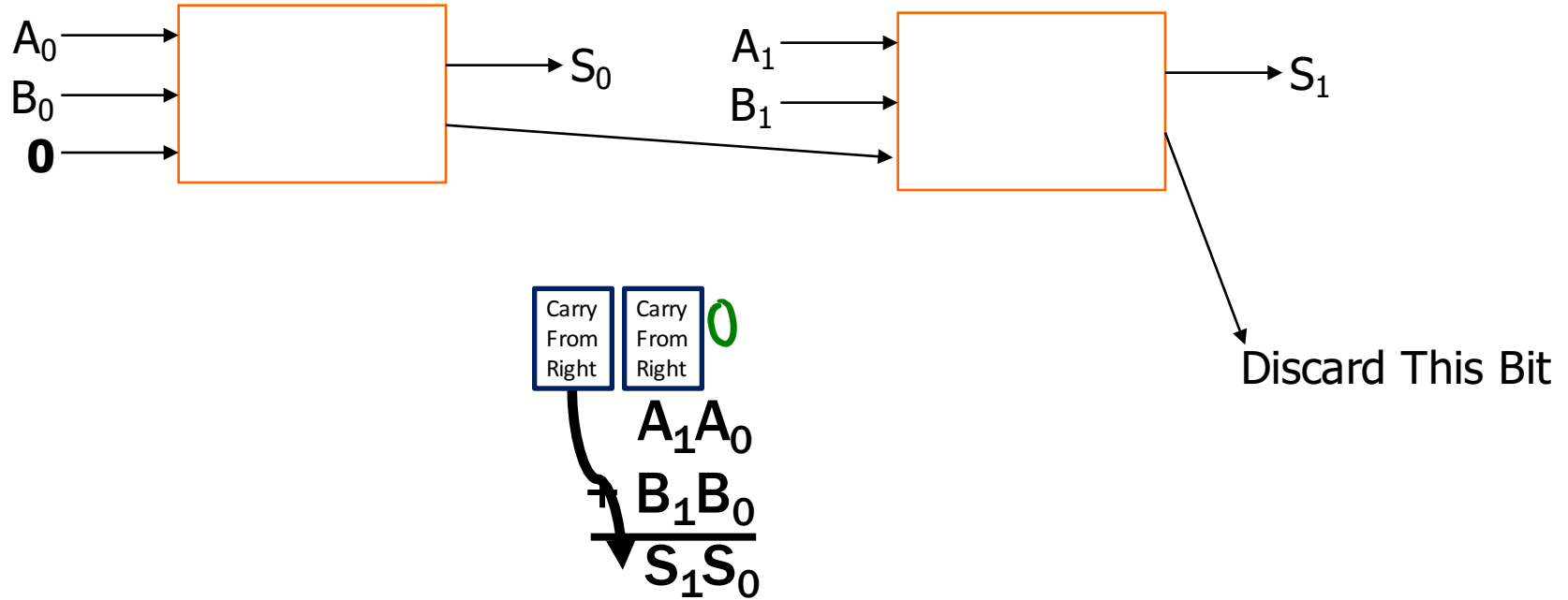


Larger Sum?

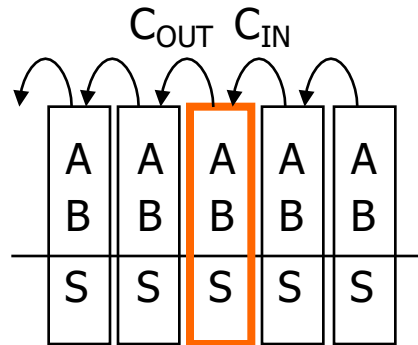
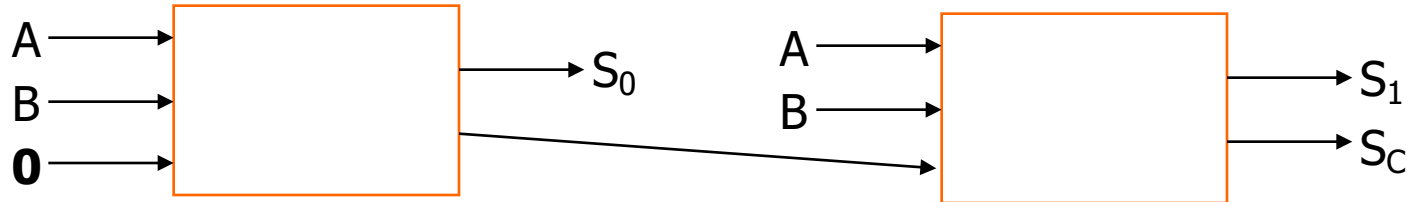


To get a sum of numbers with more bits, we can **combine** two of our original circuit together!

Larger Sum?



Larger Sum?

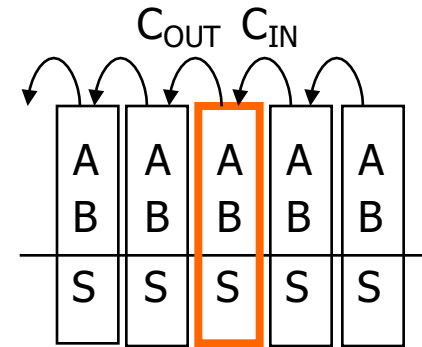


To support this idea, we rename our inputs/outputs.



1-bit Binary Adder

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

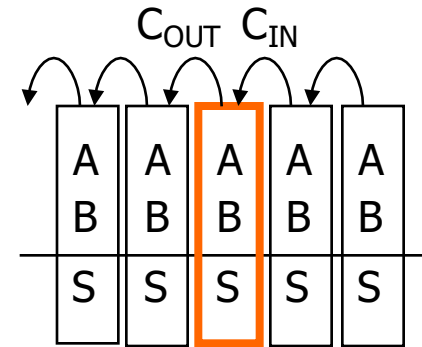


A	B	C _{IN}	C _{OUT}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



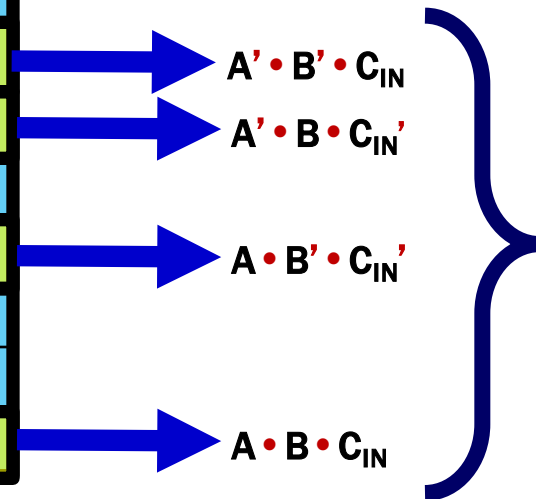
1-bit Binary Adder

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



A	B	C _{IN}	C _{OUT}	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

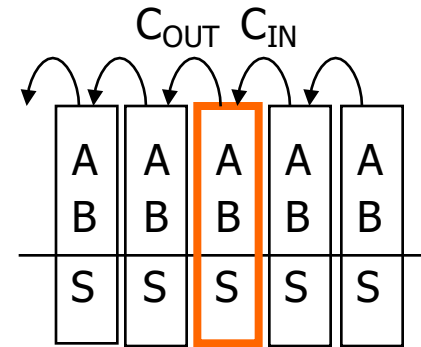
Derive an expression for S



$$S = A' \cdot B' \cdot C_{IN} + A' \cdot B \cdot C_{IN}' + A \cdot B' \cdot C_{IN}' + A \cdot B \cdot C_{IN}$$

1-bit Binary Adder

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



A	B	C _{IN}	C _{OUT}	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

Derive an expression for C_{OUT}

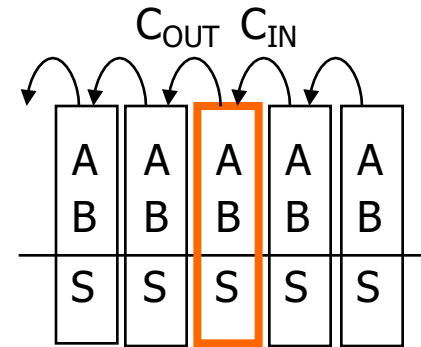
$$C_{OUT} = A' \cdot B \cdot C_{IN} + A \cdot B' \cdot C_{IN} + A \cdot B \cdot C_{IN}' + A \cdot B \cdot C_{IN}$$

The diagram shows the derivation of the expression for C_{OUT}. It starts with the truth table, where the rows where C_{OUT} is 1 are highlighted in yellow. Blue arrows point from these rows to the terms in the expression: A' · B · C_{IN}, A · B' · C_{IN}, A · B · C_{IN}', and A · B · C_{IN}. A large blue bracket groups these terms, leading to the final expression for C_{OUT}.

$$S = A' \cdot B' \cdot C_{IN} + A' \cdot B \cdot C_{IN}' + A \cdot B' \cdot C_{IN}' + A \cdot B \cdot C_{IN}$$

1-bit Binary Adder

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



A	B	C _{IN}	C _{OUT}	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

$$S = A' \cdot B' \cdot C_{IN} + A' \cdot B \cdot C_{IN}' + A \cdot B' \cdot C_{IN}' + A \cdot B \cdot C_{IN}$$

$$C_{OUT} = A' \cdot B \cdot C_{IN} + A \cdot B' \cdot C_{IN} + A \cdot B \cdot C_{IN}' + A \cdot B \cdot C_{IN}$$

Apply Theorems to Simplify Expressions

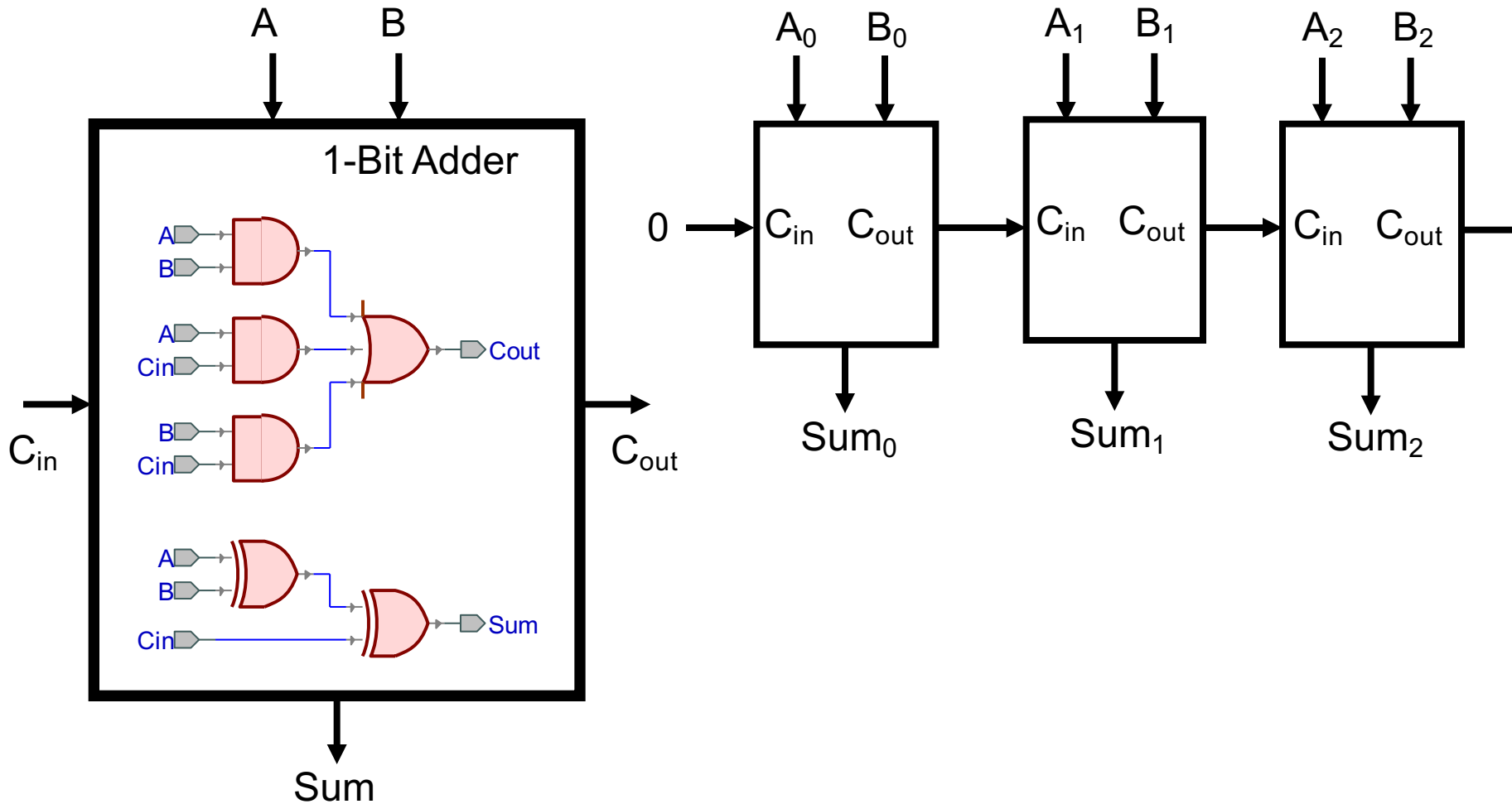
The theorems of Boolean algebra can simplify expressions

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

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

adding extra terms
creates new factoring
opportunities

A 2-bit Ripple-Carry Adder



Mapping Truth Tables to Logic Gates

Given a truth table:

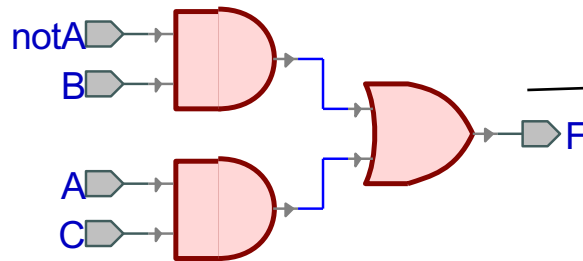
1. Write the Boolean expression
2. Minimize the Boolean expression
3. Draw as gates
4. Map to available gates

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

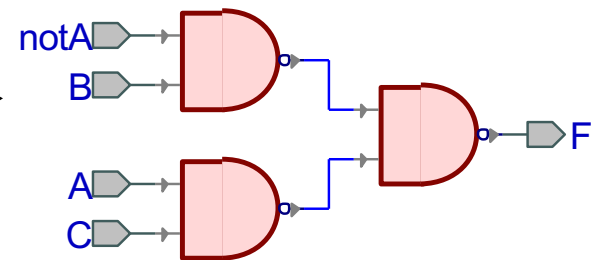
(2) ↓

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

(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

- AKA **Disjunctive Normal Form (DNF)**
- AKA **Minterm Expansion**

③

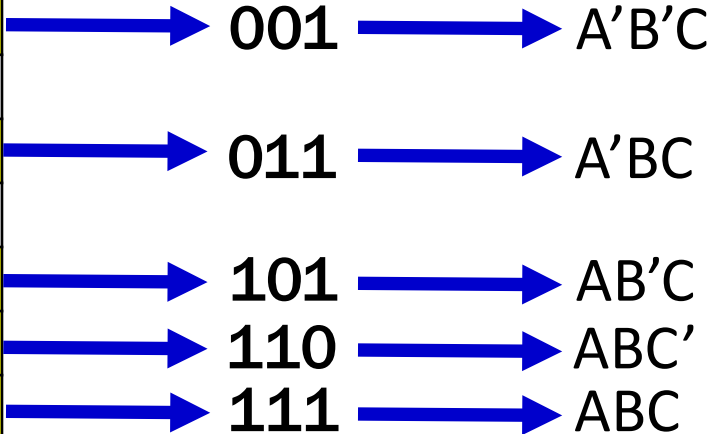
Add the minterms together

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

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

①
Read T rows off
truth table

②
Convert to
Boolean Algebra



F

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	$A'B'C'$
0	0	1	$A'B'C$
0	1	0	$A'BC'$
0	1	1	$A'BC$
1	0	0	$AB'C'$
1	0	1	$AB'C$
1	1	0	ABC'
1	1	1	ABC

F in canonical form:

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

canonical form \neq minimal form

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

$$A + C$$

$$(A \wedge B) \vee C$$