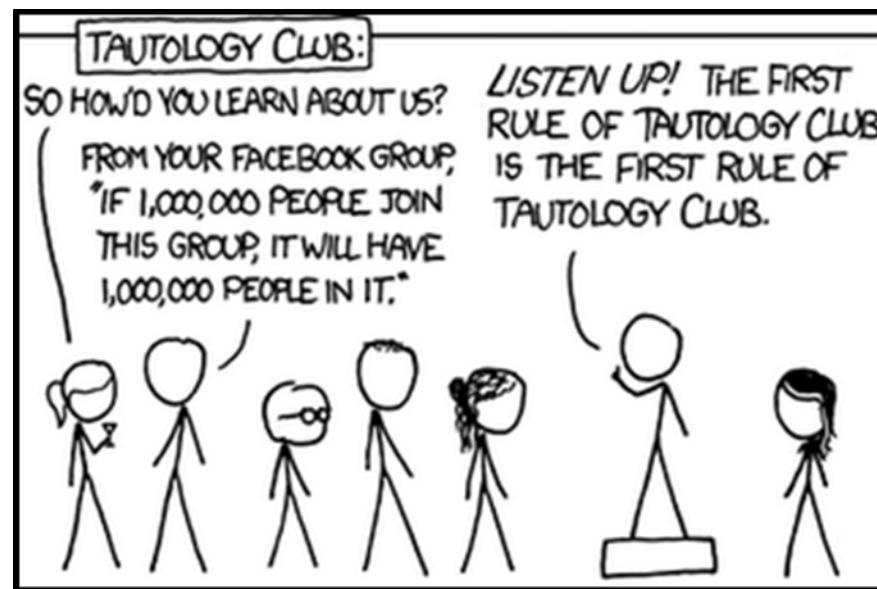


CSE 311: Foundations of Computing

Lecture 4: Boolean Algebra, Circuits, Canonical Forms



Boolean Logic

Combinational Logic

- $\text{output} = F(\text{input})$

Sequential Logic

- $\text{output}_t = F(\text{output}_{t-1}, \text{input}_t)$
 - output dependent on history
 - concept of a time step (clock, t)



Boolean Algebra: Another notation for logic consisting of...

- a set of elements $B = \{0, 1\}$
- binary operations $\{ +, \cdot \}$ (OR, AND)
- and a unary operation $\{ '\}$ (NOT)

Boolean Algebra

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

For any a, b, c in B :

1. closure:
2. commutativity:
3. associativity:
4. distributivity:
5. identity:
6. complementarity:
7. null:
8. idempotency:
9. involution:



$$\begin{aligned} a + b &\text{ is in } B \\ a + b &= b + a \\ a + (b + c) &= (a + b) + c \\ a + (b \cdot c) &= (a + b) \cdot (a + c) \\ a + 0 &= a \\ a + a' &= 1 \\ a + 1 &= 1 \\ a + a &= a \\ (a')' &= a \end{aligned}$$

$$\begin{aligned} a \cdot b &\text{ is in } B \\ a \cdot b &= b \cdot a \\ a \cdot (b \cdot c) &= (a \cdot b) \cdot c \\ a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\ a \cdot 1 &= a \\ a \cdot a' &= 0 \\ a \cdot 0 &= 0 \\ a \cdot a &= a \end{aligned}$$



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**

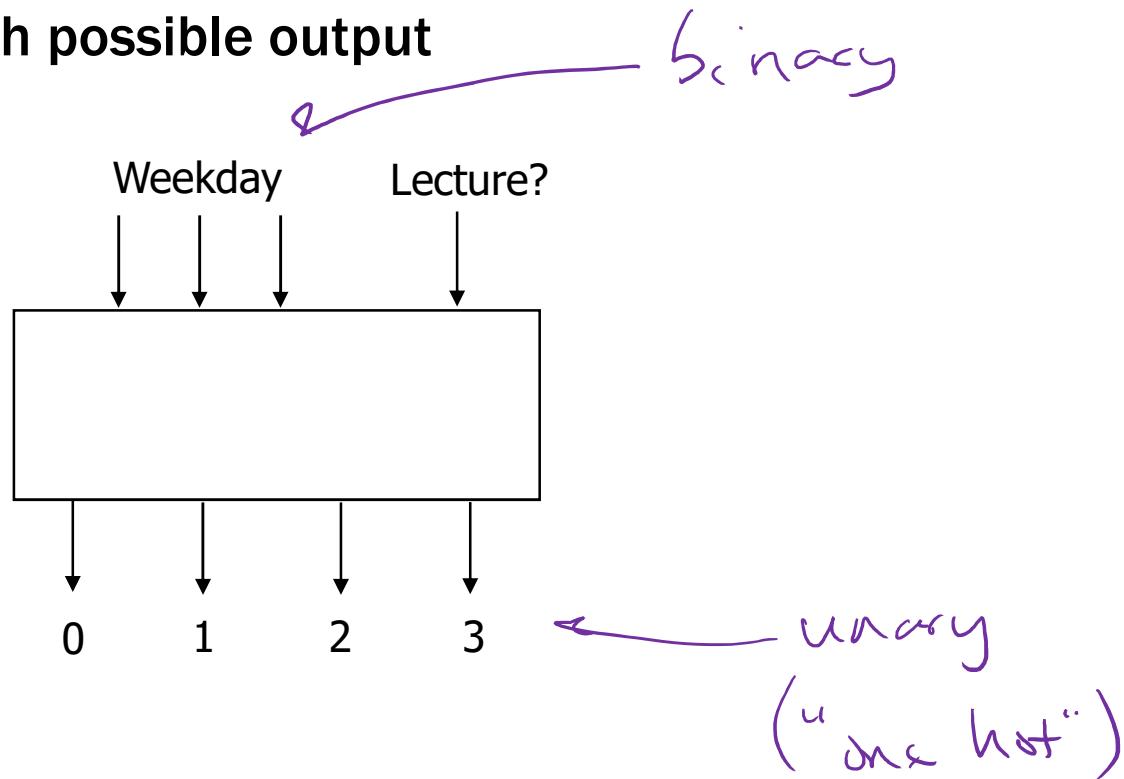
Implementation in Software

```
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

7 111

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	0	0	0	1
	MON	001	0	1	0	0
	MON	001	0	0	0	1
	TUE	010	0	1	0	0
	TUE	010	0	0	1	0
	WED	011	0	1	0	0
	WED	011	0	0	1	0
	THU	100	-	1	0	0
	FRI	101	0	0	1	0
	FRI	101	0	0	0	1
	SAT	110	-	0	1	0
	-	111	-	0	0	1

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	-	0	1	0
	FRI	101	0	1	0	0
	FRI	101	1	0	1	0
	SAT	110	-	1	0	0
	-	111	-	1	0	0

Truth Table to Logic (Part 1)

$d_2 d_1 d_0$	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)

$d_2 d_1 d_0$	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

DAY == SUN && L == 1

DAY == MON && L == 1

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

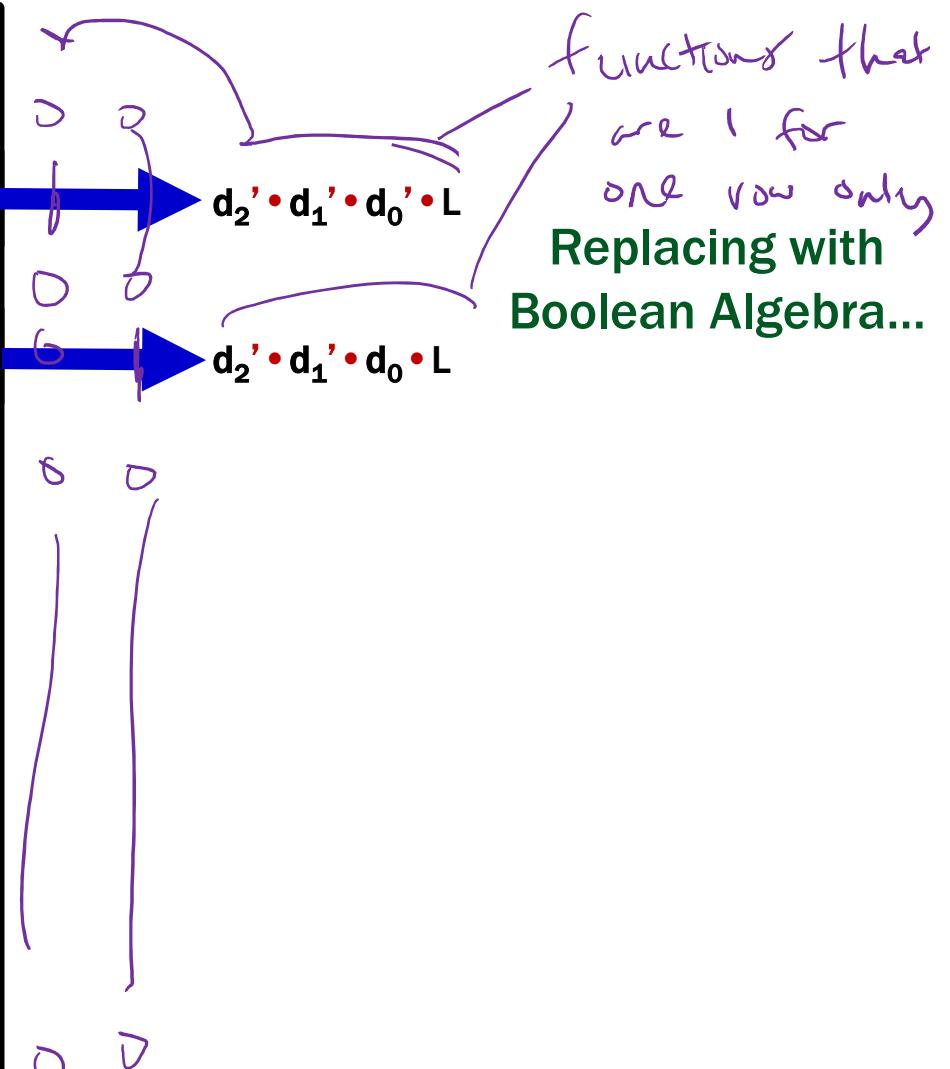
→ $d_2 == 0 \&\& d_1 == 0 \&\& d_0 == 0 \&\& L == 1$

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

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

Truth Table to Logic (Part 1)

$d_2 d_1 d_0$	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



Truth Table to Logic (Part 1)

$d_2 d_1 d_0$	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.

$$c_3 = \underbrace{d_2' \cdot d_1' \cdot d_0' \cdot L}_{\text{Term 1}} + \underbrace{d_2' \cdot d_1' \cdot d_0 \cdot L}_{\text{Term 2}}$$

Truth Table to Logic (Part 2)

	$d_2 d_1 d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$
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	$d_2' \cdot d_1 \cdot d_0' \cdot L$
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	
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	

Now, we do c_2 .

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

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

$$= c_2.$$

Truth Table to Logic (Part 3)

$d_2 d_1 d_0$	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

Now, we do c_1 :

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

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

Truth Table to Logic (Part 3)

$d_2 d_1 d_0$	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 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 :

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

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

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

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

???

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

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

Truth Table to Logic (Part 3)

$d_2 d_1 d_0$	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 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 :

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

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

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

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

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

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

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)

$d_2 d_1 d_0$	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 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 :

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

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

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

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

Truth Table to Logic (Part 4)

	$d_2 d_1 d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' +$ $d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' +$ $d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1 \cdot d_0 \cdot L$
SUN	000	1	0	0	0	1	$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$
MON	001	0	0	1	0	0	$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$
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	$d_2 \cdot d_1' \cdot d_0 \cdot L'$
FRI	101	1	0	1	0	0	
SAT	110	-	1	0	0	0	$d_2 \cdot d_1 \cdot d_0'$
-	111	-	1	0	0	0	$d_2 \cdot d_1 \cdot d_0$

Finally, we do c_0 :

Truth Table to Logic (Part 4)

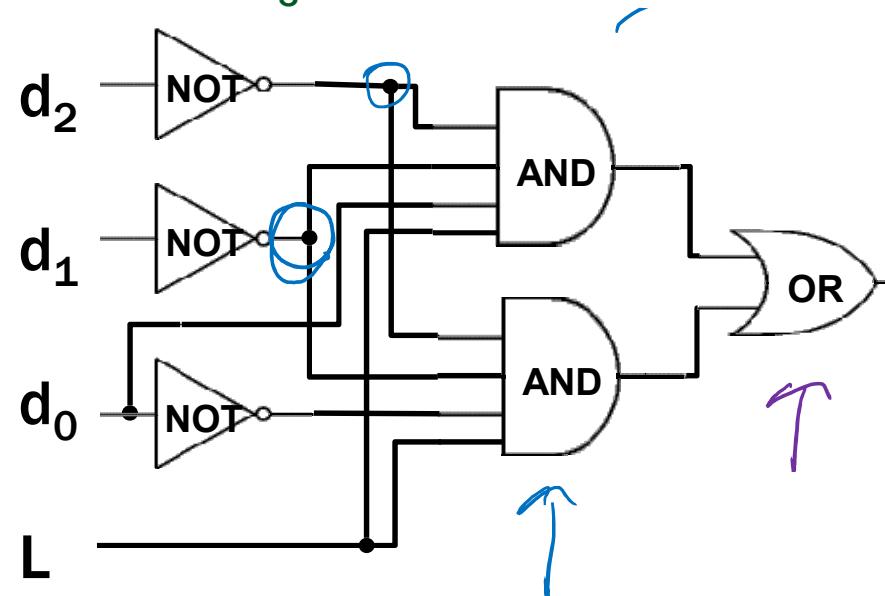
$$c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1 \cdot d_0$$

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

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

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

Here's c_3 as a circuit:



Boolean Algebra

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

For any a, b, c in B :

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



Simplification using Boolean Algebra

uniting:

$$10. \quad a \cdot b + a \cdot b' = a$$

$$10D. \quad (a + b) \cdot (a + b') = a$$

absorption:

$$11. \quad a + a \cdot b = a$$

$$11D. \quad a \cdot (a + b) = a$$

$$12. \quad (a + b') \cdot b = a \cdot b$$

$$12D. \quad (a \cdot b') + b = a + b$$

factoring:

$$13. \quad (a + b) \cdot (a' + c) = \\ a \cdot c + a' \cdot b$$

$$13D. \quad a \cdot b + a' \cdot c = \\ (a + c) \cdot (a' + b)$$

consensus:

$$14. \quad (a \cdot b) + (b \cdot c) + (a' \cdot c) = \\ a \cdot b + a' \cdot c$$

$$14D. \quad (a + b) \cdot (b + c) \cdot (a' + c) = \\ (a + b) \cdot (a' + c)$$

de Morgan's:

$$15. \quad (a + b + \dots)' = a' \cdot b' \cdot \dots$$

$$15D. \quad (a \cdot b \cdot \dots)' = a' + b' + \dots$$

Proving Theorems

- 2. commutativity:
- 3. associativity:
- 4. distributivity:
- 5. identity:
- 6. complementarity:
- 7. null:
- 8. idempotency:
- 9. involution:

$$\begin{aligned} a + b &= b + a \\ a + (b + c) &= (a + b) + c \\ a + (b \cdot c) &= (a + b) \cdot (a + c) \\ a + 0 &= a \\ a + a' &= 1 \\ a + 1 &= 1 \quad \leftarrow \\ a + a &= a \\ (a')' &= a \end{aligned}$$

$$\begin{aligned} a \cdot b &= b \cdot a \\ a \cdot (b \cdot c) &= (a \cdot b) \cdot c \\ a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\ a \cdot 1 &= a \\ a \cdot a' &= 0 \\ a \cdot 0 &= 0 \\ a \cdot a &= a \end{aligned}$$

Using the laws of Boolean Algebra:

prove the Uniting theorem:

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

dist
comp
identity

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

prove the Absorption theorem:

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

(con)

$$\begin{aligned} \text{identity} \quad X + X \cdot Y &= X \cdot 1 + X \cdot Y \\ \text{distr} \quad &= X \cdot (1 + Y) \\ \text{null} \quad &= X \cdot 1 \\ \text{identity} \quad &= X \end{aligned}$$

Proving Theorems

- 2. commutativity:
- 3. associativity:
- 4. distributivity:
- 5. identity:
- 6. complementarity:
- 7. null:
- 8. idempotency:
- 9. involution:

$$\begin{aligned} a + b &= b + a \\ a + (b + c) &= (a + b) + c \\ a + (b \cdot c) &= (a + b) \cdot (a + c) \\ a + 0 &= a \\ a + a' &= 1 \\ a + 1 &= 1 \\ a + a &= a \\ (a')' &= a \end{aligned}$$

$$\begin{aligned} a \cdot b &= b \cdot a \\ a \cdot (b \cdot c) &= (a \cdot b) \cdot c \\ a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\ a \cdot 1 &= a \\ a \cdot a' &= 0 \\ a \cdot 0 &= 0 \\ a \cdot a &= a \end{aligned}$$

Using the laws of Boolean Algebra:

prove the Uniting theorem:

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

distributivity

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

complementarity

identity

prove the theorem:

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

identity

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

distributivity

commutativity

null

identity

Proving Theorems

Using 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	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

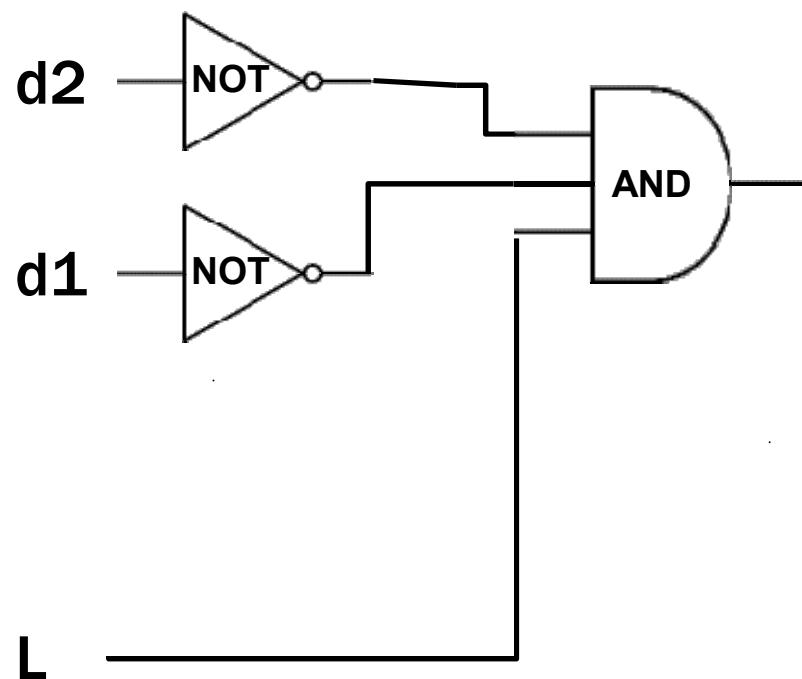
$$(X \bullet Y)' = X' + Y'$$

NAND is equivalent to OR
with inputs complemented

X	Y	X'	Y'	(X • Y)'	X' + Y'
0	0	1	1	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

$$\begin{array}{r} A \\ + B \\ \hline S \\ (\text{C}_{\text{OUT}}) \end{array}$$

$0 + 0 = 0$ (with $\text{C}_{\text{OUT}} = 0$)
 $0 + 1 = 1$ (with $\text{C}_{\text{OUT}} = 0$)
 $1 + 0 = 1$ (with $\text{C}_{\text{OUT}} = 0$)
 $1 + 1 = 0$ (with $\text{C}_{\text{OUT}} = 1$)

C_{out}

$\begin{array}{r} | \\ + 1 \\ + 1 \\ \hline 11 \end{array}$

$\begin{array}{r} 6 \\ 7 \\ \hline 13 \end{array}$

1-bit Binary Adder

A	$0 + 0 = 0$ (with $C_{OUT} = 0$)
<u>+ B</u>	$0 + 1 = 1$ (with $C_{OUT} = 0$)
S	$1 + 0 = 1$ (with $C_{OUT} = 0$)
(C_{OUT})	$1 + 1 = 0$ (with $C_{OUT} = 1$)

Idea: To chain these together, let's add a carry-in

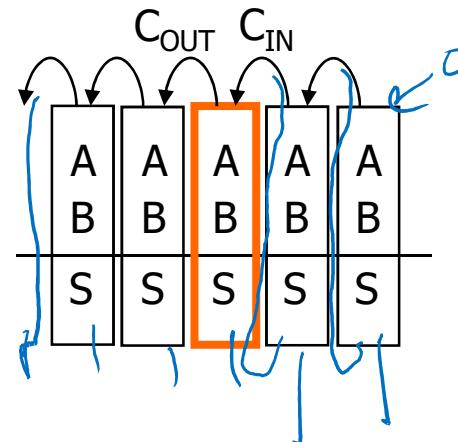
$$\begin{array}{r} & 1 \\ & \swarrow \curvearrowright \curvearrowleft \searrow \\ 1 & 1 & 0 & 1 \\ & \downarrow & & \downarrow \\ 0 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 \end{array}$$

1-bit Binary Adder

$$\begin{array}{r} A \\ + B \\ \hline S \\ (C_{OUT}) \end{array} \quad \begin{array}{l} 0 + 0 = 0 \text{ (with } C_{OUT} = 0) \\ 0 + 1 = 1 \text{ (with } C_{OUT} = 0) \\ 1 + 0 = 1 \text{ (with } C_{OUT} = 0) \\ 1 + 1 = 0 \text{ (with } C_{OUT} = 1) \end{array}$$

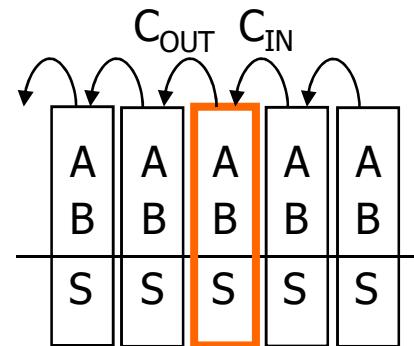
Idea: To chain these together, let's add a carry-in

$$\begin{array}{r} (C_{IN}) \\ A \\ + B \\ \hline S \\ (C_{OUT}) \end{array}$$



1-bit Binary Adder

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

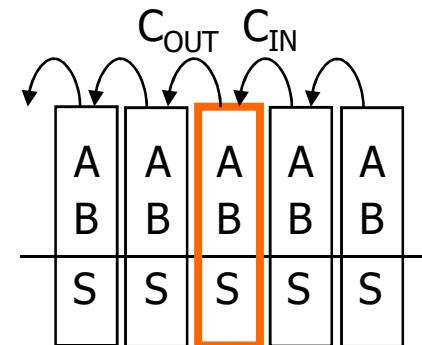


A	B	C _{IN}	C _{OUT}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



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

$$A' \cdot B' \cdot C_{IN}$$

$$A' \cdot B \cdot C_{IN}'$$

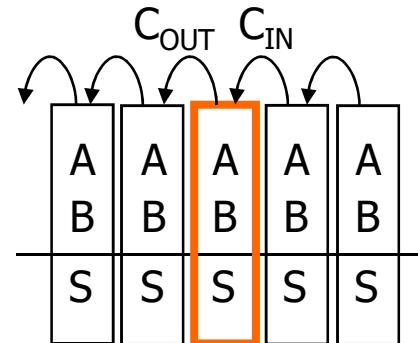
$$A \cdot B' \cdot C_{IN}'$$

$$A \cdot B \cdot C_{IN}$$

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

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}

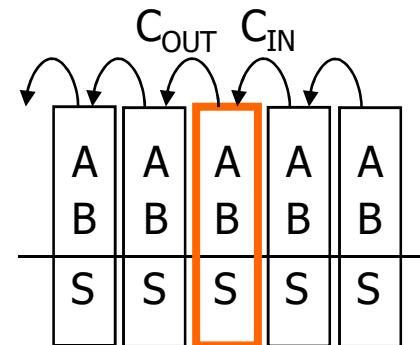
$$\begin{aligned}
 & A' \cdot B \cdot C_{IN} \\
 & A \cdot B' \cdot C_{IN} \\
 & A \cdot B \cdot C_{IN}' \\
 & A \cdot B \cdot C_{IN}
 \end{aligned}$$

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

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

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

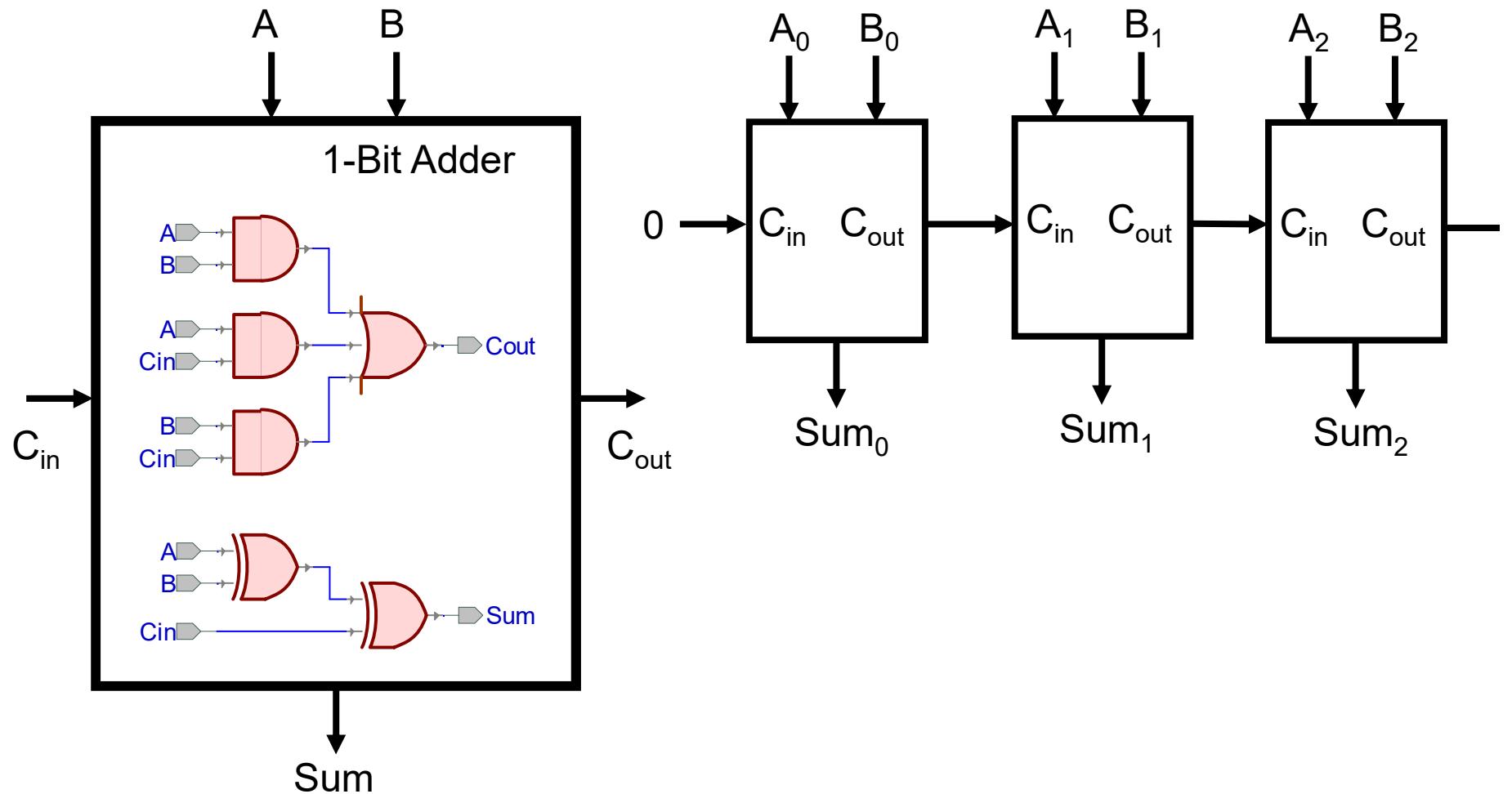
The theorems of Boolean algebra can simplify expressions

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

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

adding extra terms
creates new factoring
opportunities

A 2-bit Ripple-Carry Adder



Mapping Truth Tables to Logic Gates

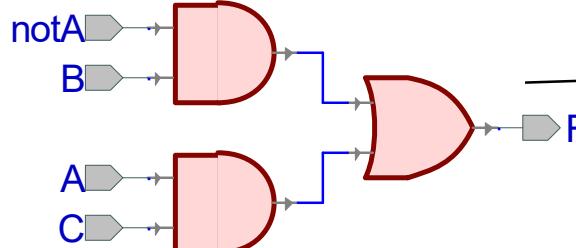
Given a truth table:

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

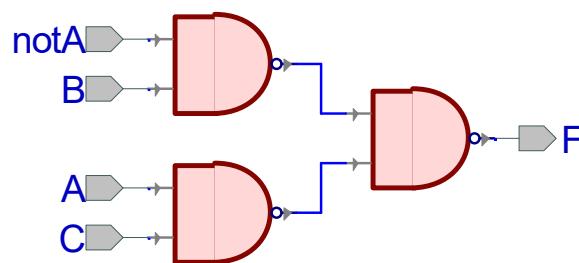
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(2)
$$\begin{aligned} F &= A'BC' + A'BC + AB'C + ABC \\ &= A'B(C' + C) + AC(B' + B) \\ &= A'B + AC \end{aligned}$$

(1)



(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

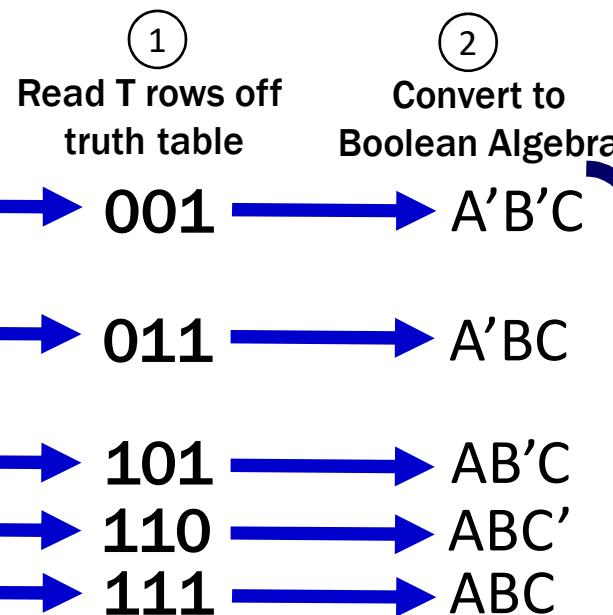
Sum-of-Products Canonical Form

- 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



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

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

Product-of-Sums Canonical Form

- AKA **Conjunctive Normal Form (CNF)**
- AKA **Maxterm Expansion**

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 F rows off
truth table

② Negate all
bits

④

Multiply the maxterms together

$F =$

③

Convert to
Boolean Algebra

F

Product-of-Sums Canonical Form

- AKA **Conjunctive Normal Form (CNF)**
- AKA **Maxterm Expansion**

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 F rows off
truth table

000

② Negate all
bits

111

③ Convert to
Boolean Algebra

$A + B + C$

010

101

$A + B' + C$

100

011

$A' + B + C$

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

Multiply the maxterms together

④

F

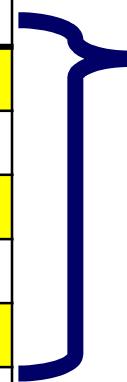
F

Product-of-Sums: Why does this procedure work?

Useful Facts:

- We know $(F')' = F$
- We know how to get a minterm expansion for F'

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



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

Product-of-Sums: Why does this procedure work?

Useful Facts:

- We know $(F')' = F$
- We know how to get a minterm expansion for F'

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

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

Taking the complement of both sides...

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

And using DeMorgan/Comp....

$$F = (A'B'C')' (A'BC')' (AB'C')'$$

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

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	$A+B+C$
0	0	1	$A+B+C'$
0	1	0	$A+B'+C$
0	1	1	$A+B'+C'$
1	0	0	$A'+B+C$
1	0	1	$A'+B+C'$
1	1	0	$A'+B'+C$
1	1	1	$A'+B'+C'$

F in canonical form:

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

canonical form \neq minimal form

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