CSE 311

Foundations of Computing I

Pre-Lecture Problem

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

Use Logical Equivalences to show

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

Identity

$p \wedge \mathsf{T} \equiv p$

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

Domination

$p \vee \mathsf{T} \equiv \mathsf{T}$

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

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

$p \vee \neg p \equiv \mathsf{T}$

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

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

$p \to q \equiv \neg p \vee q$

Contrapositive

$p \to q \equiv \neg q \to \neg p$

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

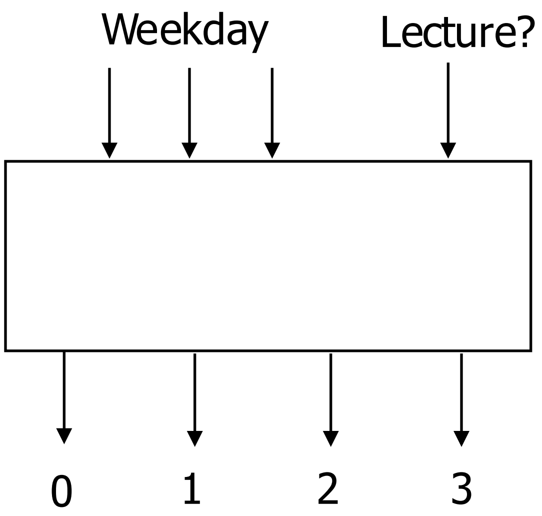
Implementation in Software

```
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	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)

	$d_2d_1d_0$	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

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)

	$d_2d_1d_0$	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

SUN, 000, 1 → $d_2' \cdot d_1' \cdot d_0' \cdot L$

MON, 001, 1 → $d_2' \cdot d_1' \cdot d_0 \cdot L$

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.

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
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:

- SUN 000, L=0 → $d_2' \cdot d_1' \cdot d_0' \cdot L'$
- MON 001, L=0 → $d_2' \cdot d_1' \cdot d_0 \cdot L'$
- TUE 010, L=0 → $d_2' \cdot d_1 \cdot d_0' \cdot L'$
- WED 011, L=0 → $d_2' \cdot d_1 \cdot d_0 \cdot L'$
- THU 100 → $d_2 \cdot d_1' \cdot d_0'$
- FRI 101, L=1 → $d_2 \cdot d_1' \cdot d_0 \cdot L$

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

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

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

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

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:

FRI 101 0 → $d_2 \cdot d_1' \cdot d_0 \cdot L'$

SAT 110 → $d_2 \cdot d_1 \cdot d_0'$

111 → $d_2 \cdot d_1 \cdot d_0$

Truth Table to Logic (Part 4)

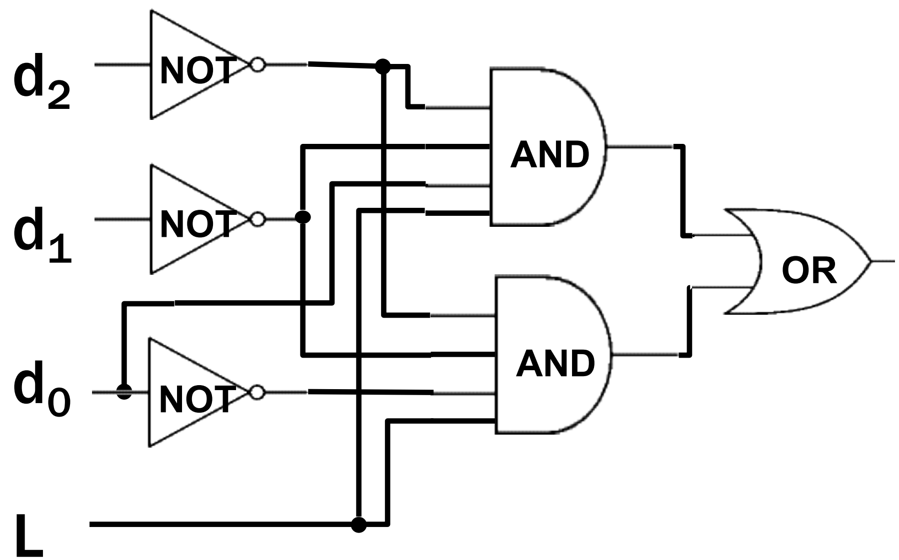
$c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1 \cdot d_0$

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

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

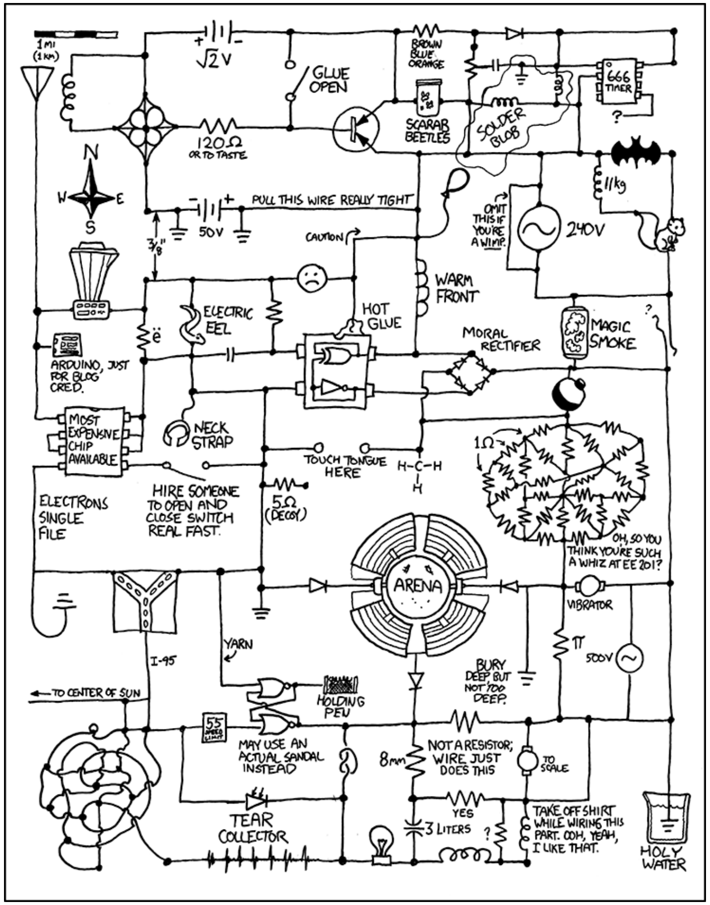
$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

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 { + , • }
  - 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. $X \bullet 1 = X$

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. $X \bullet X' = 0$

commutativity:

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

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

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

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	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

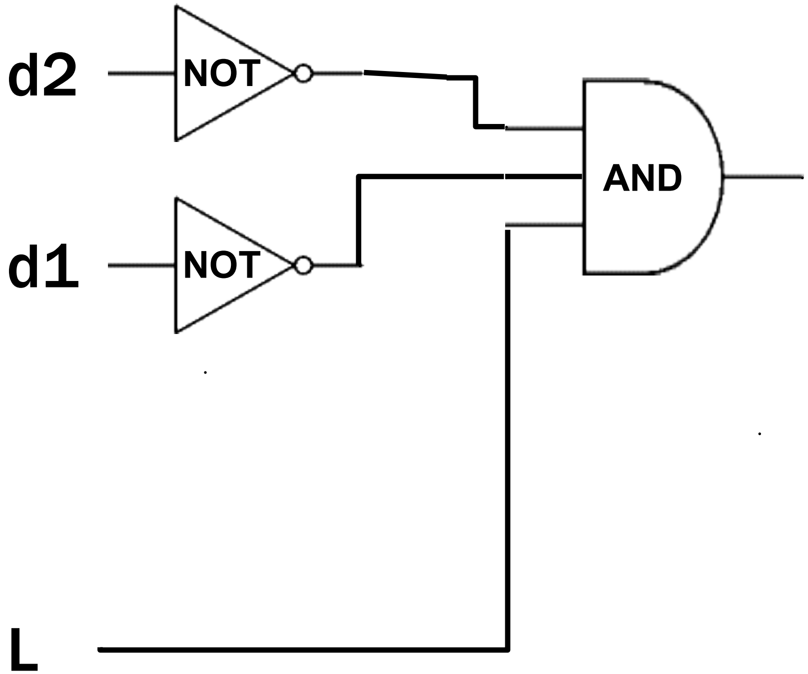
Simplifying using Boolean Algebra

c3 = d2’ • d1’ • d0’ • L + d2’ • d1’ • d0 • L

= d2’ • d1’ • (d0’ + d0) • L

= d2’ • d1’ • (1) • L

= d2’ • d1’ • L

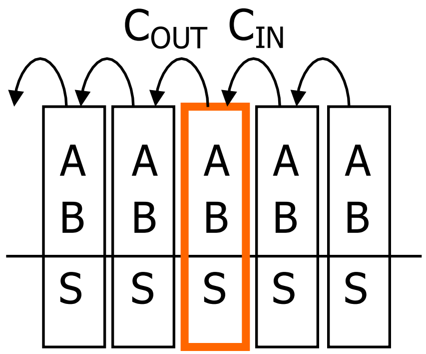


1-bit Binary Adder

A	$0 + 0 = 0$ (with $C_{OUT} = 0$)
+ B	$0 + 1 = 1$ (with $C_{OUT} = 0$)
S	$1 + 0 = 1$ (with $C_{OUT} = 0$)
(C_{OUT})	$1 + 1 = 0$ (with $C_{OUT} = 1$)

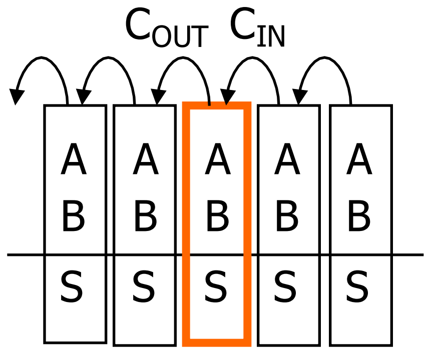
Idea: To chain these together, let's add a carry-in

(C_{IN})
A
+ B
S
(C_{OUT})



1-bit Binary Adder

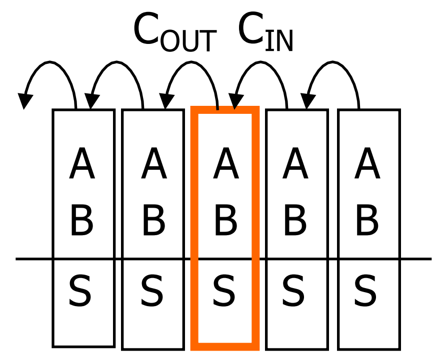
- **Inputs:** A, B, Carry-in
- **Outputs:** Sum, Carry-out



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

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

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

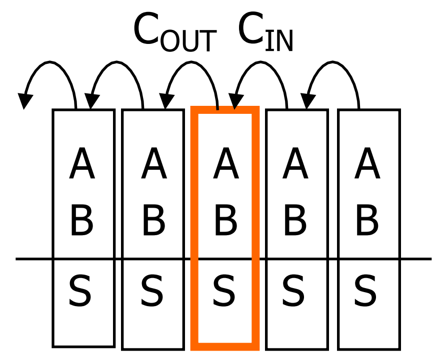
$A \cdot B' \cdot C_{IN}'$

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

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

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}

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

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

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

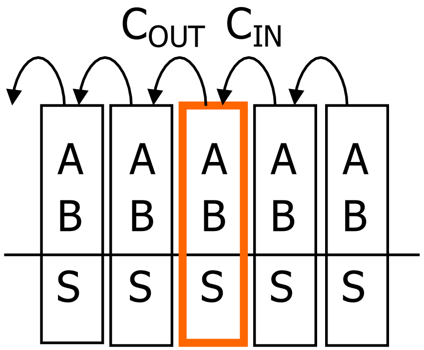
$A \cdot B \cdot C_{IN}$

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

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

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

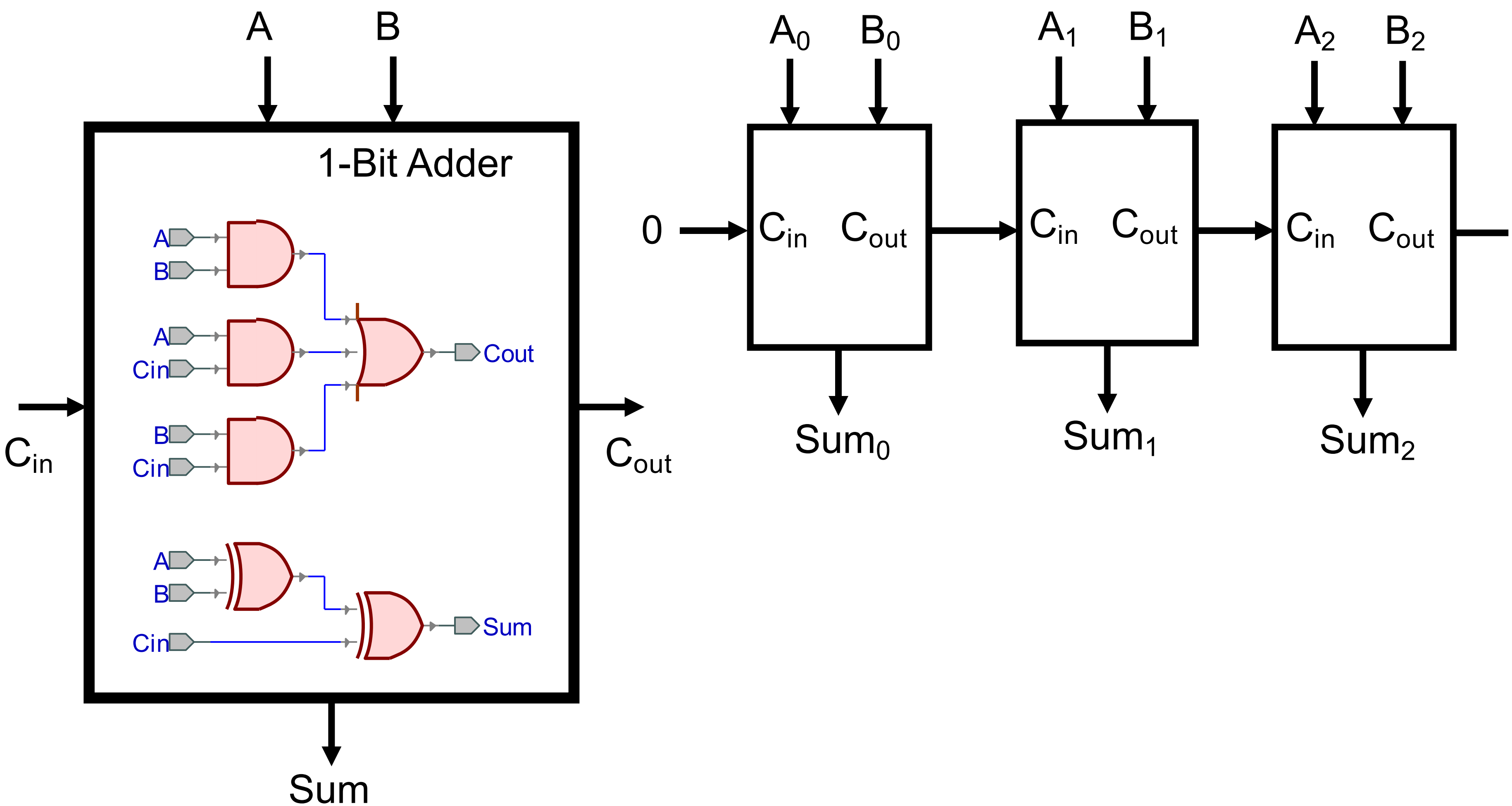
The theorems of Boolean algebra can simplify expressions

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

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

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

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