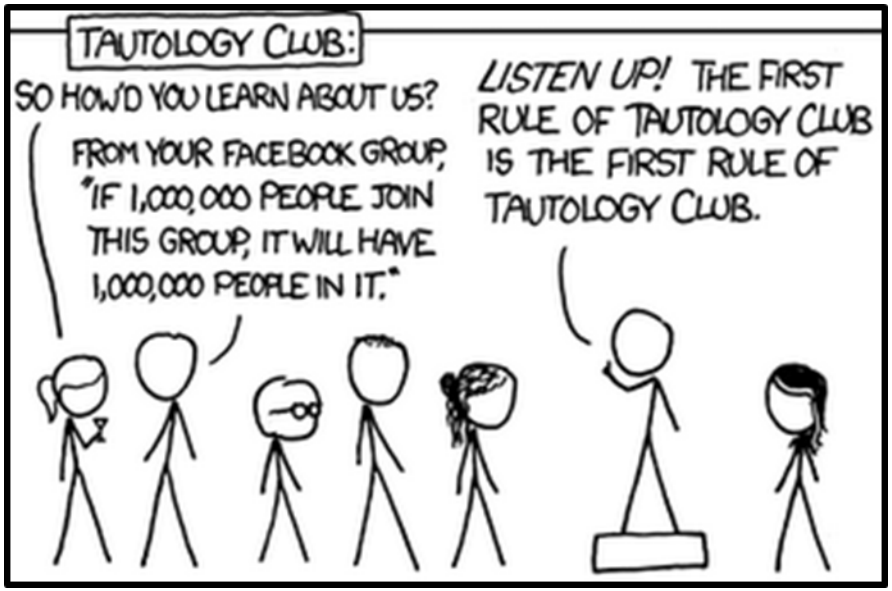
CSE 311: Foundations of Computing

Lecture 4: Circuits and Canonical Forms



Boolean algebra

Boolean Algebra: Another notation for propositional logic consisting of…

- a set of elements B = {0, 1}
- binary operations { + , • } (OR, AND)
- and a unary operation { ’ } (NOT)

George Boole
(1815-1864)

Example: $q \vee (r \wedge \neg s)$ written as $q + (r \cdot s')$

Notation used in circuit design

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**

Goal: Design a circuit implementing this function!

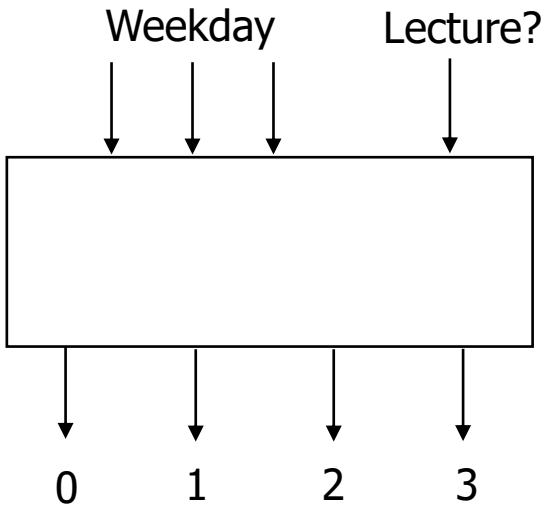
Implementation in Software

```
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				
SUN	000	1				
MON	001	0				
MON	001	1				
TUE	010	0				
TUE	010	1				
WED	011	0				
WED	011	1				
THU	100	-				
FRI	101	0				
FRI	101	1				
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	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

$\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L$

$\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L$

Replacing with Boolean logic…

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

$\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L$

$\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L$

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

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

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

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

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:

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

$$c_2 \equiv (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge L)$$

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
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:

$\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge \neg L$

$\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L$

$\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge \neg L$

$\neg d_2 \wedge d_1 \wedge d_0 \wedge \neg L$

???

$d_2 \wedge \neg d_1 \wedge d_0 \wedge L$

$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$

$c_2 \equiv (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge L)$

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
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:

$\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge \neg L$

$\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L$

$\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge \neg L$

$\neg d_2 \wedge d_1 \wedge d_0 \wedge \neg L$

$d_2 \wedge \neg d_1 \wedge \neg d_0$

L does not matter. So, we don't need L in the expression.

$d_2 \wedge \neg d_1 \wedge d_0 \wedge L$

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

$$c_2 \equiv (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge L)$$

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
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:

$\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge \neg L$

$\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L$

$\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge \neg L$

$\neg d_2 \wedge d_1 \wedge d_0 \wedge \neg L$

$d_2 \wedge \neg d_1 \wedge \neg d_0$

L does not matter. So, we don't need L in the expression.

$d_2 \wedge \neg d_1 \wedge d_0 \wedge L$

$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$

$c_2 \equiv (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge L)$

$c_1 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge \neg L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L) \vee (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge \neg L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge \neg L) \vee (d_2 \wedge \neg d_1 \wedge \neg d_0) \vee (d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$

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

$$c_1 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge \neg L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L) \vee (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge \neg L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge \neg L) \vee (d_2 \wedge \neg d_1 \wedge \neg d_0) \vee (d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

$$c_2 \equiv (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge L)$$

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

Finally, we do c_0:

$$d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L$$

$$d_2 \wedge d_1 \wedge \neg d_0$$

$$d_2 \wedge d_1 \wedge d_0$$

Truth Table to Logic (Part 4)

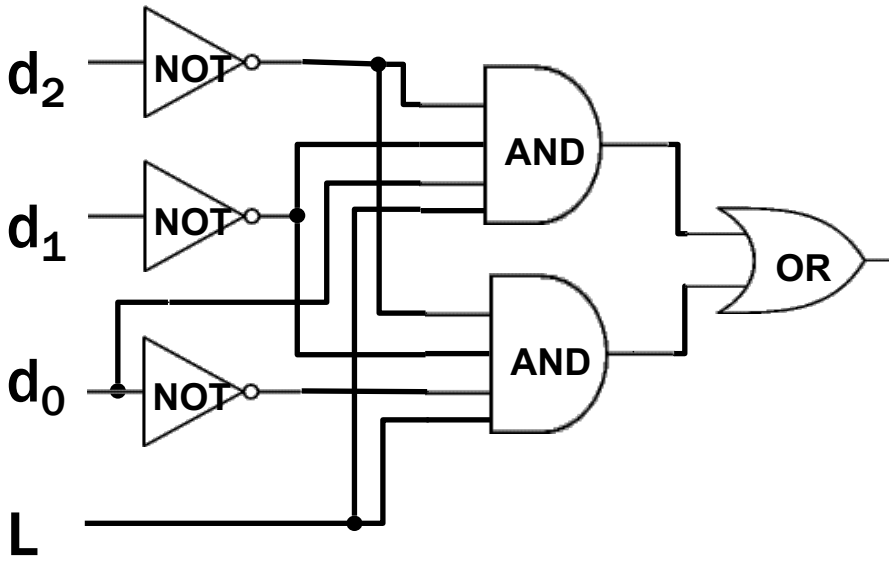
$c_0 \equiv (d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L) \vee (d_2 \wedge d_1 \wedge \neg d_0) \vee (d_2 \wedge d_1 \wedge d_0)$

$c_1 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge \neg L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge \neg L) \vee (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge \neg L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge \neg L) \vee (d_2 \wedge \neg d_1 \wedge \neg d_0) \vee (d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$

$c_2 \equiv (\neg d_2 \wedge d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge d_1 \wedge d_0 \wedge L)$

$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$

Here's c_3 as a circuit:



Simplifying using Boolean Algebra

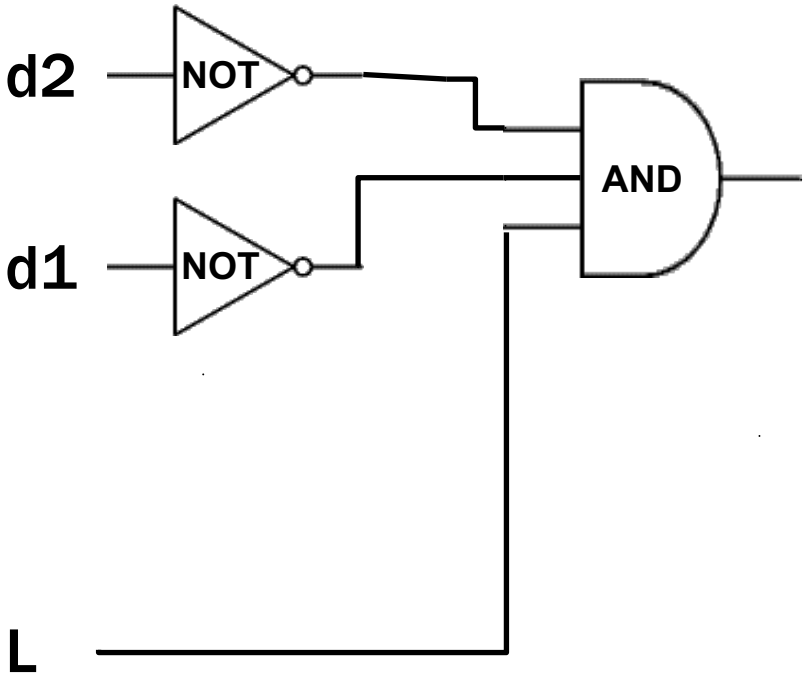
$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$

Simplifying using Boolean Algebra

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$
$$\equiv \neg d_2 \wedge \neg d_1 \wedge (\neg d_0 \vee d_0) \wedge L$$

Simplifying using Boolean Algebra

$$c_3 \equiv (\neg d_2 \wedge \neg d_1 \wedge \neg d_0 \wedge L) \vee (\neg d_2 \wedge \neg d_1 \wedge d_0 \wedge L)$$
$$\equiv \neg d_2 \wedge \neg d_1 \wedge (\neg d_0 \vee d_0) \wedge L$$
$$\equiv \neg d_2 \wedge \neg d_1 \wedge L$$



Lecture 4 Activity

- You will be assigned to **breakout rooms**. Please:
- Introduce yourself
- Choose someone to share screen, showing this PDF
- Today's task: Below you can find the truth table of a function c that depends on inputs d_0, d_1. Write $c \equiv \cdots$ in terms of the input variables and the operations $\wedge, \vee, \neg$

d_0	d_1	c
0	0	0
0	1	1
1	0	1
1	1	0

Then fill out the poll everywhere for **Activity Credit**!
Go to pollev.com/thomas311 and login with your UW identity

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)

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)

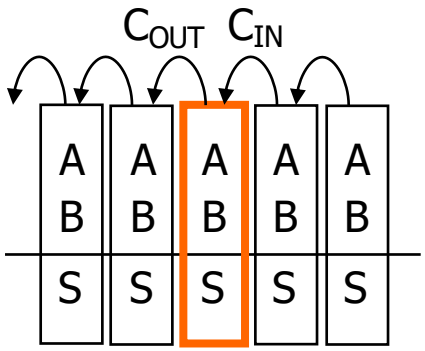
Idea: To chain these together, let's add a carry-in

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)

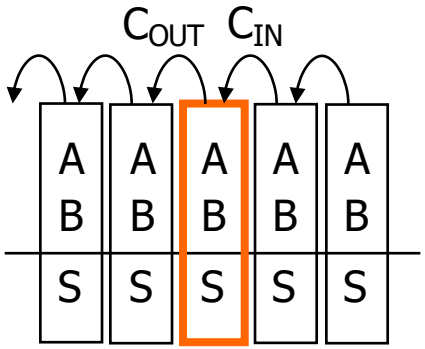
Idea: To chain these together, let's add a carry-in

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



1-bit Binary Adder

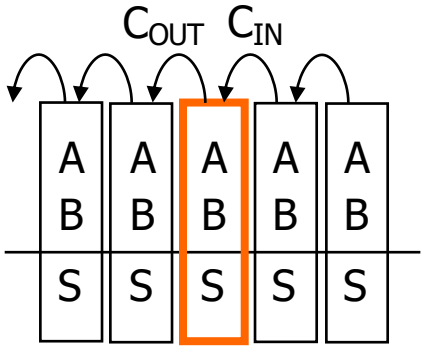
- **Inputs:** A, B, Carry-in
- **Outputs:** Sum, Carry-out



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

$\neg \boldsymbol{A} \wedge \neg \boldsymbol{B} \wedge \boldsymbol{C_{in}}$

$\neg \boldsymbol{A} \wedge \boldsymbol{B} \wedge \neg \boldsymbol{C_{in}}$

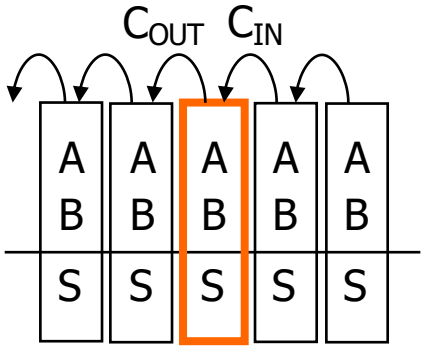
$\boldsymbol{A} \wedge \neg \boldsymbol{B} \wedge \neg \boldsymbol{C_{in}}$

$\boldsymbol{A} \wedge \boldsymbol{B} \wedge \boldsymbol{C_{in}}$

$$S \equiv (\neg A \wedge \neg B \wedge C_{in}) \vee (\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in}) \vee (A \vee B \vee C_{in})$$

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}

$\neg A \wedge B \wedge C_{in}$

$A \wedge \neg B \wedge C_{in}$

$A \wedge B \wedge \neg C_{in}$

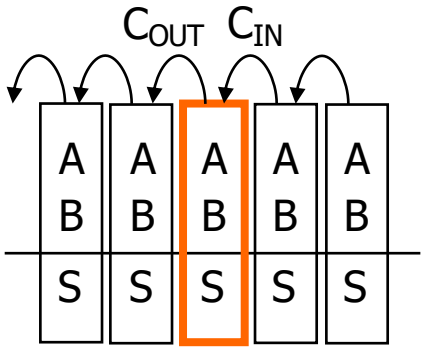
$A \wedge B \wedge C_{in}$

$$S \equiv (\neg A \wedge \neg B \wedge C_{in}) \vee (\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in}) \vee (A \vee B \vee C_{in})$$

$$C_{out} \equiv (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in})$$

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

$$S \equiv (\neg A \wedge \neg B \wedge C_{in}) \vee (\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in}) \vee (A \vee B \vee C_{in})$$

$$C_{out} \equiv (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in})$$

Apply Theorems to Simplify Expressions

The laws of propositional can simplify expressions

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

$$C_{out}$$
$$\equiv (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in})$$

Apply Theorems to Simplify Expressions

The laws of propositional can simplify expressions

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

$$
\begin{aligned}
& C_{out} \\
\equiv\ & (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in}) \\
\equiv\ & ((\neg A \wedge B \wedge C_{in}) \vee (A \wedge B \wedge C_{in})) \vee ((A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in})) \vee ((A \wedge B \wedge \neg C_{in}) \\
& \vee (A \wedge B \wedge C_{in}))
\end{aligned}
$$

Remark: We repeat terms in the 2nd $\equiv$ to obtain cancellations later

Apply Theorems to Simplify Expressions

The laws of propositional can simplify expressions

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

$$
\begin{aligned}
& C_{out} \\
\equiv\ & (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in}) \\
\equiv\ & ((\neg A \wedge B \wedge C_{in}) \vee (A \wedge B \wedge C_{in})) \vee ((A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in})) \vee ((A \wedge B \wedge \neg C_{in}) \\
& \vee (A \wedge B \wedge C_{in})) \\
\equiv\ & ((\neg A \vee A) \wedge B \wedge C_{in}) \vee (A \wedge (B \vee \neg B) \wedge C_{in}) \vee ((A \wedge B) \wedge (\neg C_{in} \vee C_{in}))
\end{aligned}
$$

Remark: We repeat terms in the 2nd ≡ to obtain cancellations later

Apply Theorems to Simplify Expressions

The laws of propositional can simplify expressions

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

$$
\begin{aligned}
& C_{out} \\
\equiv\ & (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in}) \\
\equiv\ & ((\neg A \wedge B \wedge C_{in}) \vee (A \wedge B \wedge C_{in})) \vee ((A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in})) \vee ((A \wedge B \wedge \neg C_{in}) \\
& \vee (A \wedge B \wedge C_{in})) \\
\equiv\ & ((\neg A \vee A) \wedge B \wedge C_{in}) \vee (A \wedge (B \vee \neg B) \wedge C_{in}) \vee ((A \wedge B) \wedge (\neg C_{in} \vee C_{in})) \\
\equiv\ & (T \wedge B \wedge C_{in}) \vee (A \wedge T \wedge C_{in}) \vee (A \wedge B \wedge T)
\end{aligned}
$$

Remark: We repeat terms in the 2nd ≡ to obtain cancellations later

Apply Theorems to Simplify Expressions

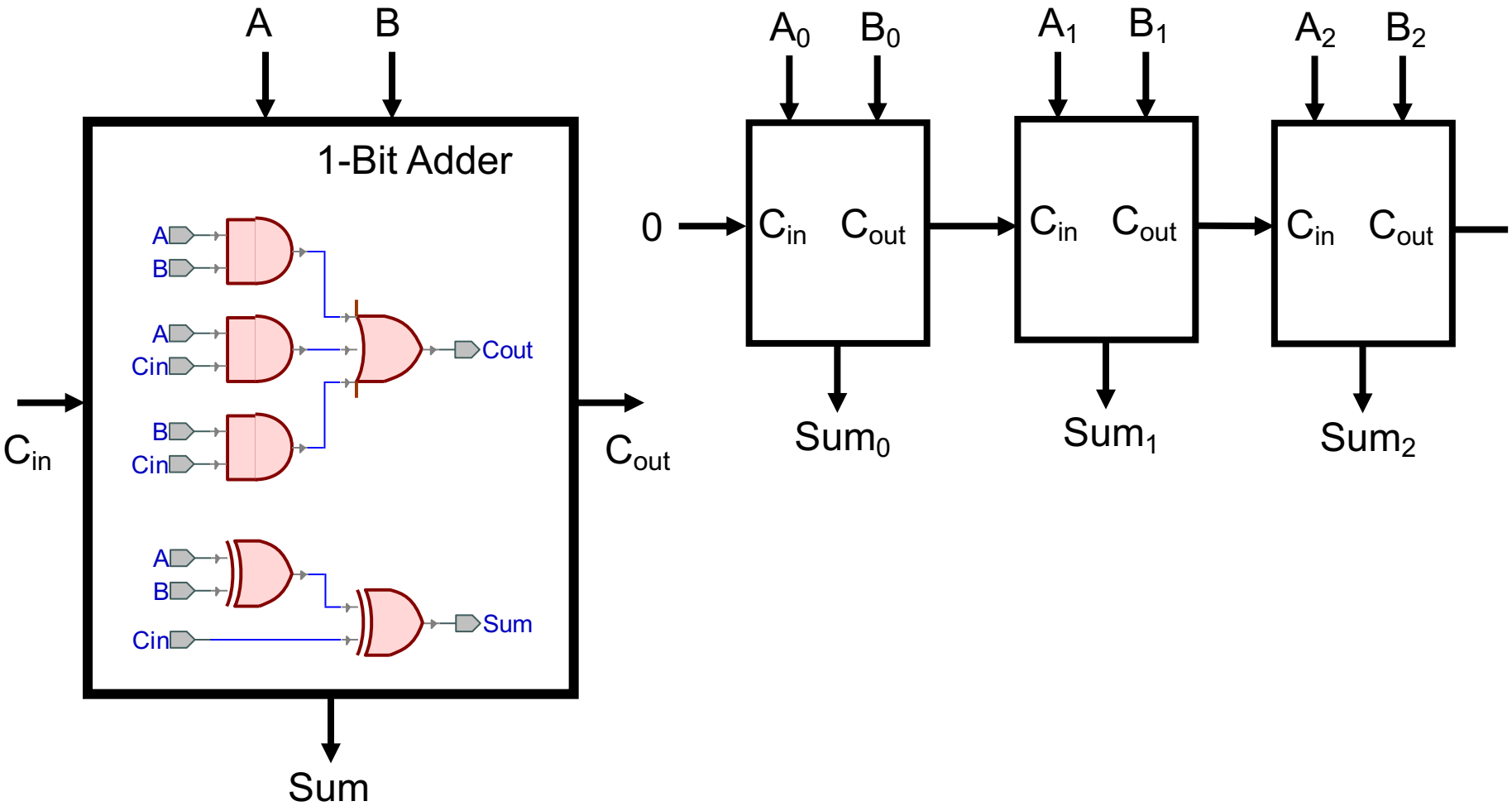
The laws of propositional can simplify expressions

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

$$
\begin{aligned}
& C_{out} \\
\equiv\ & (\neg A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge \neg C_{in}) \vee (A \wedge B \wedge C_{in}) \\
\equiv\ & ((\neg A \wedge B \wedge C_{in}) \vee (A \wedge B \wedge C_{in})) \vee ((A \wedge B \wedge C_{in}) \vee (A \wedge \neg B \wedge C_{in})) \vee ((A \wedge B \wedge \neg C_{in}) \\
& \vee (A \wedge B \wedge C_{in})) \\
\equiv\ & ((\neg A \vee A) \wedge B \wedge C_{in}) \vee (A \wedge (B \vee \neg B) \wedge C_{in}) \vee ((A \wedge B) \wedge (\neg C_{in} \vee C_{in})) \\
\equiv\ & (T \wedge B \wedge C_{in}) \vee (A \wedge T \wedge C_{in}) \vee (A \wedge B \wedge T) \\
\equiv\ & (B \wedge C_{in}) \vee (A \wedge C_{in}) \vee (A \wedge B)
\end{aligned}
$$

Remark: We repeat terms in the 2nd ≡ to obtain cancellations later

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)

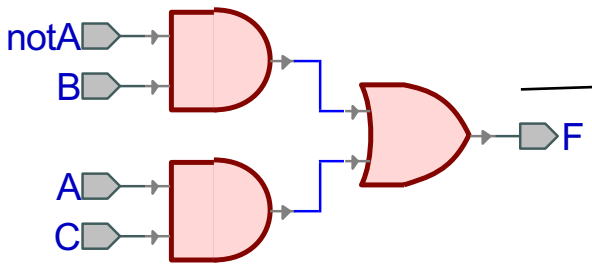
$$F \equiv (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C)$$

(2)

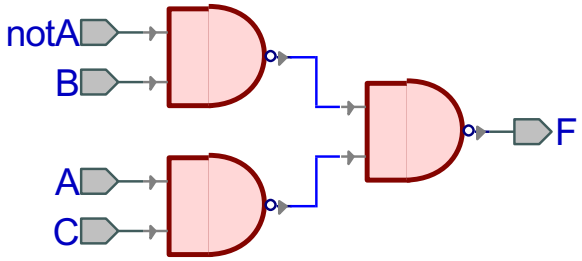
$$\equiv \big(\neg A \wedge B \wedge (\neg C \vee C)\big) \vee \big(A \wedge C \wedge (\neg B \vee B)\big)$$

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

(3)



(4)



Disjunctive Normal Form

- For a propositional logic variable q, the terms q and $\neg q$ are the literals
- A propositional logic formula is in disjunctive normal form (DNF), if it is an OR of AND terms of literals

Example for DNF: $(q \wedge \neg r \wedge s) \vee (\neg q \wedge \neg r) \vee (\neg r \wedge \neg s)$

Disjunctive Normal Form

- For a propositional logic variable q, the terms q and $\neg q$ are the literals
- A propositional logic formula is in disjunctive normal form (DNF), if it is an OR of AND terms of literals

  Example for DNF: $(q \wedge \neg r \wedge s) \vee (\neg q \wedge \neg r) \vee (\neg r \wedge \neg s)$

- Every propositional formula has an equivalent DNF.

However that DNF is not necessarily unique.

Disjunctive Normal Form

- For a propositional logic variable q, the terms q and $\neg q$ are the literals
- A propositional logic formula is in disjunctive normal form (DNF), if it is an OR of AND terms of literals

Example for DNF: $(q \wedge \neg r \wedge s) \vee (\neg q \wedge \neg r) \vee (\neg r \wedge \neg s)$

- Every propositional formula has an equivalent DNF. However that DNF is not necessarily unique.
- A logic formula is in full DNF if every variable appears exactly once in every conjunction. The full DNF is unique up to the order of the terms.

Example for full DNF: $(q \wedge \neg r \wedge s) \vee (\neg q \wedge \neg r \wedge \neg s) \vee (q \wedge \neg r \wedge \neg s)$

Construction of Disjunctive Normal Form (DNF)

- AKA Sum-of-Products Canonical Form
- AKA Minterm Expansion

3 Take disjunction of the conjunctions

$$F \equiv (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)$$

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 conjunction

001 → $\neg A \wedge \neg B \wedge C$

011 → $\neg A \wedge B \wedge C$

101 → $A \wedge \neg B \wedge C$

110 → $A \wedge B \wedge \neg C$

111 → $A \wedge B \wedge C$

Construction of Disjunctive Normal Form (DNF)

Note: Obtained (full) DNF is often not minimal

Full DNF:

$$F \equiv (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)$$

DNF (but minimized):

$$F \equiv (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)$$
$$\equiv \big(\big((\neg A \wedge \neg B) \vee (\neg A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B)\big) \wedge C\big) \vee (A \wedge B \wedge \neg C)$$
$$\equiv \Big(\big((\neg A \vee A) \wedge (\neg B \vee B)\big) \wedge C\Big) \vee (A \wedge B \wedge \neg C)$$
$$\equiv C \vee (A \wedge B \wedge \neg C)$$
$$\equiv C \vee (A \wedge B)$$

Conjunctive Normal Form

- A propositional logic formula is in conjunctive normal form (CNF), if it is an AND of OR terms of literals

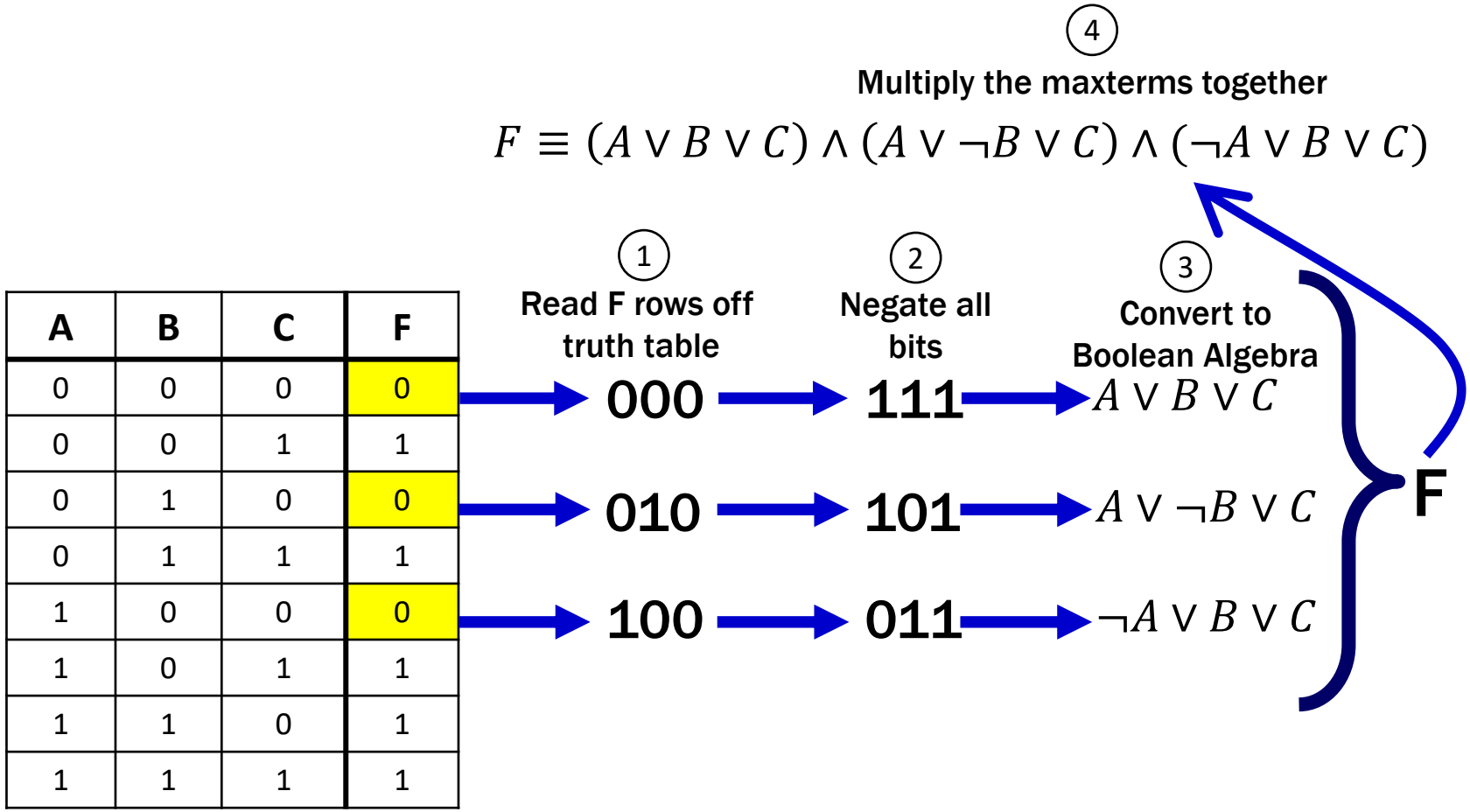
  Example for CNF: $(q \vee \neg r \vee s) \wedge (\neg q \vee \neg r) \wedge (\neg r \vee \neg s)$

- Every propositional logic formula has an equivalent CNF.

Again that CNF is not necessarily unique (but the full CNF is)

- Other names for CNF:
  - Product-of-Sums Canonical Form
  - Maxterm Expansion

Product-of-Sums Canonical Form



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

CNF: Why does this procedure work?

Useful Facts:

- **We know** $F \equiv \neg(\neg F)$
- **We know how to get a DNF for** $\neg F$

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

$$\neg F \equiv (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C)$$

Taking the complement of both sides…

$$F \equiv \neg((\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C))$$

And using DeMorgan/Comp….

$$F \equiv \neg(\neg A \wedge \neg B \wedge \neg C) \wedge \neg(\neg A \wedge B \wedge \neg C) \wedge \neg(A \wedge \neg B \wedge \neg C)$$

$$F \equiv (A \vee B \vee C) \wedge (A \vee \neg B \vee \neg C) \wedge (\neg A \vee B \vee C)$$