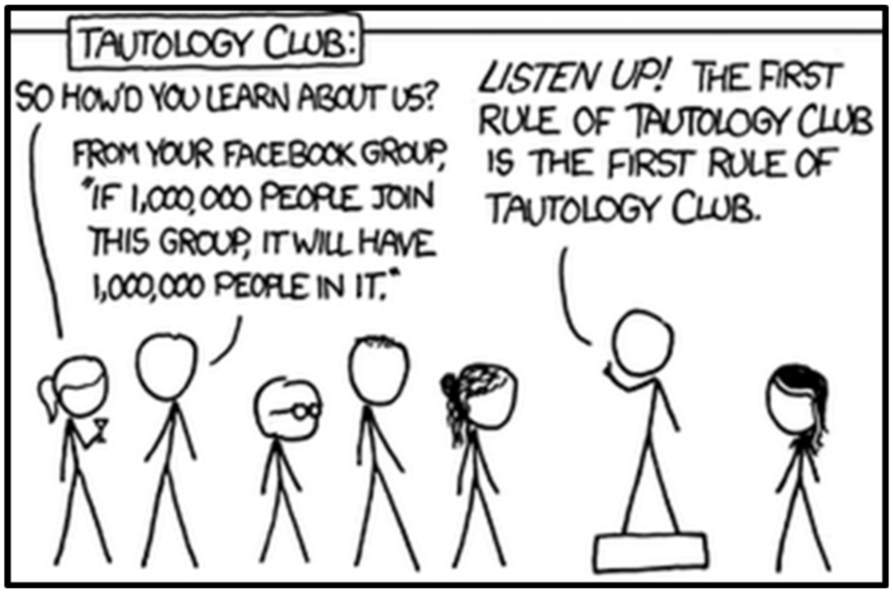
CSE 311: Foundations of Computing

Lecture 3: Digital Circuits



1^{st} HW out

Section materials & solutions posted.

OH posted.

Review: Propositional Logic

Propositions

- atomic propositions are T/F-valued variables
- combined using logical connectives (not, and, or, etc.)
- can be described by a truth table

  shows the truth value of the proposition in
  each combination of truth values of the atomic propositions

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Logical equivalence

- used to simplify logical expressions

First application

- Simplifying English sentences

Truth Table to show tautology

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

$$(p \wedge r) \rightarrow (r \vee p) \equiv \mathbf{T}$$

$\boldsymbol{p}$	$\boldsymbol{r}$	$\boldsymbol{p \wedge r}$	$\boldsymbol{r \vee p}$	$\boldsymbol{(p \wedge r) \rightarrow (r \vee p)}$
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

Logical Proofs of Equivalence

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

Use a series of equivalences like so:

$(p \wedge r) \rightarrow (r \vee p)$	$\equiv \neg(p \wedge r) \vee (r \vee p)$	Law of Implication
	$\equiv (\neg p \vee \neg r) \vee (r \vee p)$	De Morgan
	$\equiv \neg p \vee (\neg r \vee (r \vee p))$	Associative
	$\equiv \neg p \vee ((\neg r \vee r) \vee p)$	Associative
	$\equiv \neg p \vee (p \vee (\neg r \vee r))$	Commutative
	$\equiv (\neg p \vee p) \vee (\neg r \vee r)$	Associative
	$\equiv (p \vee \neg p) \vee (r \vee \neg r)$	Commutative (twice)
	$\equiv \mathbf{T} \vee \mathbf{T}$	Negation (twice)
	$\equiv \mathbf{T}$	Domination/Identity

Identity
- $p \wedge \mathrm{T} \equiv p$
- $p \vee \mathrm{F} \equiv p$

Domination
- $p \vee \mathrm{T} \equiv \mathrm{T}$
- $p \wedge \mathrm{F} \equiv \mathrm{F}$

Idempotent
- $p \vee p \equiv p$
- $p \wedge p \equiv p$

Commutative
- $p \vee q \equiv q \vee p$
- $p \wedge q \equiv q \wedge p$

Logical Proofs of Equivalence/Tautology

- Not smaller than truth tables when there are only a few propositional variables...
- ...but usually *much shorter* than truth table proofs when there are many propositional variables
- A big advantage will be that we can extend them to a more in-depth understanding of logic for which truth tables don't apply.

Another key application: Digital Circuits

Computing With Logic

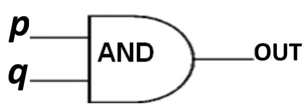
- T corresponds to 1 or "high" voltage
- F corresponds to 0 or "low" voltage

Gates

- Take inputs and produce outputs (functions)
- Several kinds of gates
- Correspond to propositional connectives (most of them)

Circuits: AND, OR, NOT Gates

AND Gate

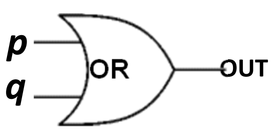


p	q	OUT
1	1	1
1	0	0
0	1	0
0	0	0

1 ↔ T
0 ↔ F

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

OR Gate

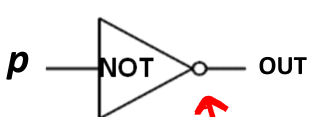


$\vee$

p	q	OUT
1	1	1
1	0	1
0	1	1
0	0	0

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

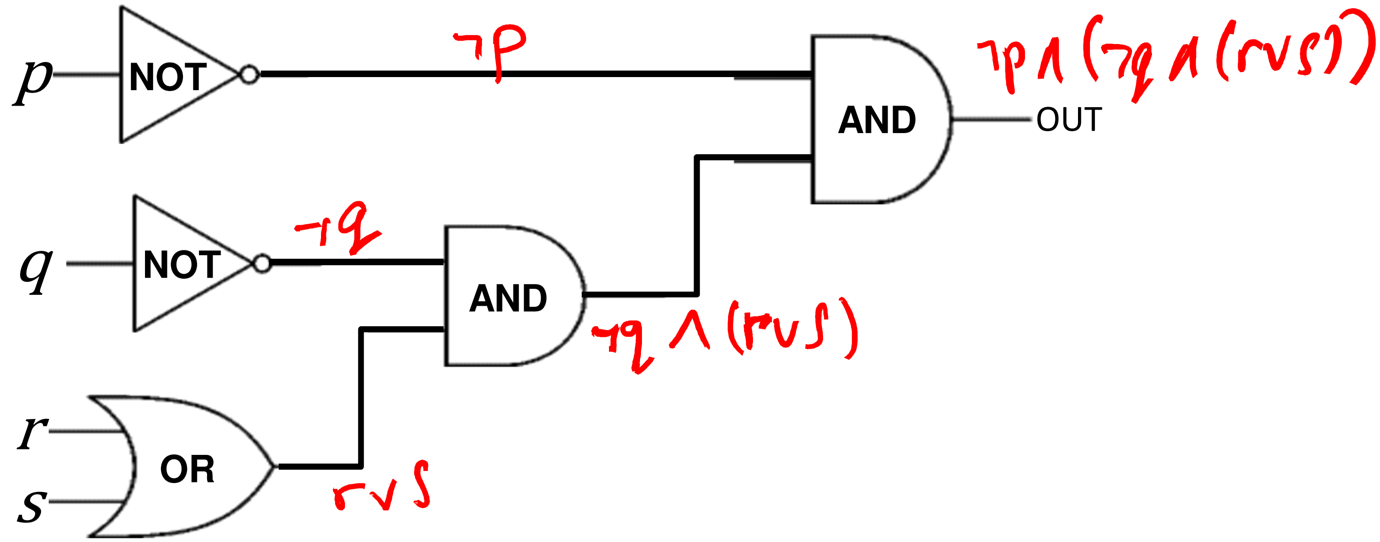
NOT Gate



p	OUT
1	0
0	1

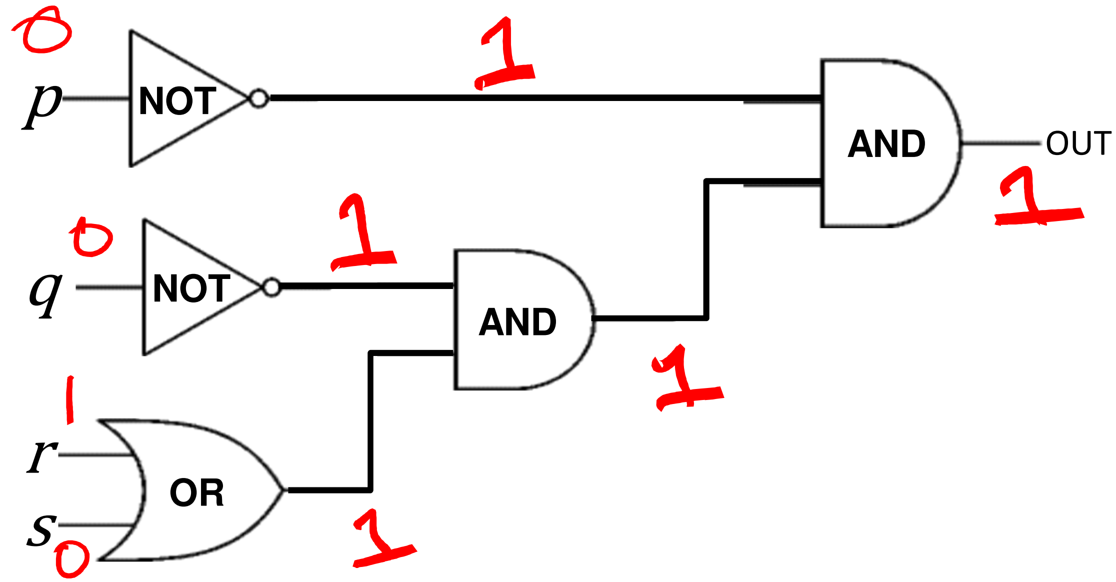
p	$\neg p$
T	F
F	T

Combinational Logic Circuits



Values get sent along wires connecting gates

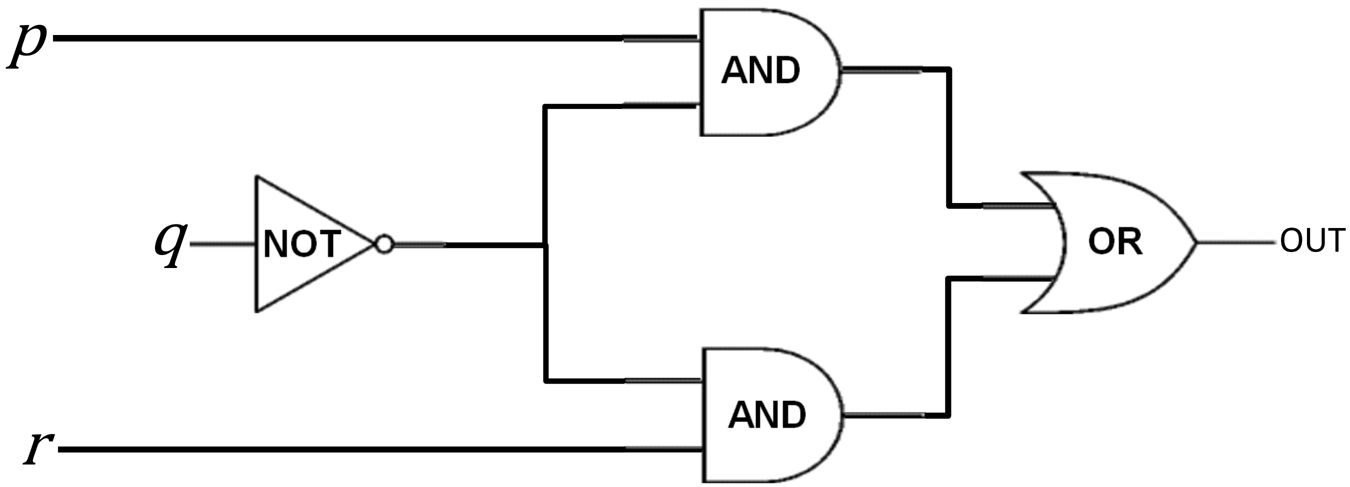
Combinational Logic Circuits



Values get sent along wires connecting gates

$$\neg p \wedge (\neg q \wedge (r \vee s))$$

Combinational Logic Circuits



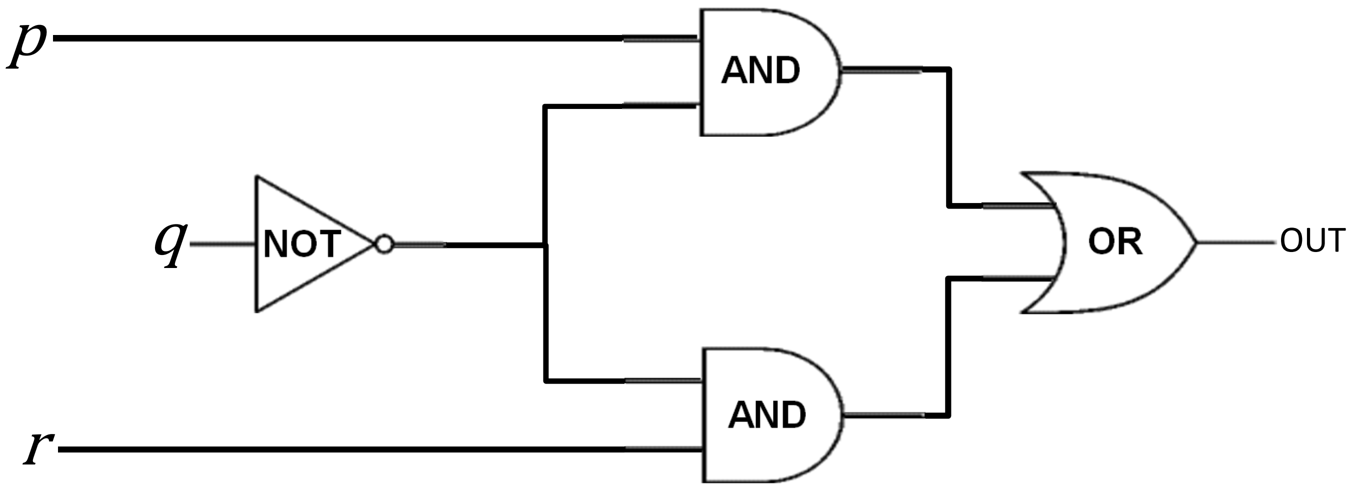
Wires can send one value to multiple gates!

$(p \wedge \neg q) \vee (\neg q \wedge r)$ ✓

$(p \wedge \neg q) \vee (r \wedge \neg q)$ ✓

$\equiv (p \vee r) \wedge \neg q$ ✓

Combinational Logic Circuits



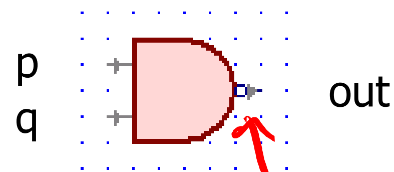
Wires can send one value to multiple gates!

$$(p \land \neg q) \lor (\neg q \land r)$$

Other Useful Gates

NAND

$$\neg(p \land q)$$

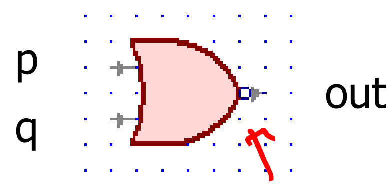


0

p	q	out
0	0	1
0	1	1
1	0	1
1	1	0

NOR

$$\neg(p \lor q)$$

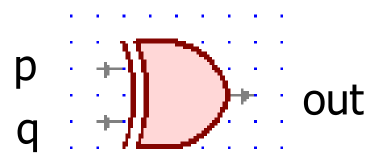


0

p	q	out
0	0	1
0	1	0
1	0	0
1	1	0

XOR

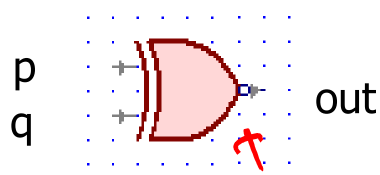
$$p \oplus q$$



p	q	out
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

$$p \leftrightarrow q$$



p	q	out
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Logic

Combinational Logic

– output = F(input)

Sequential Logic

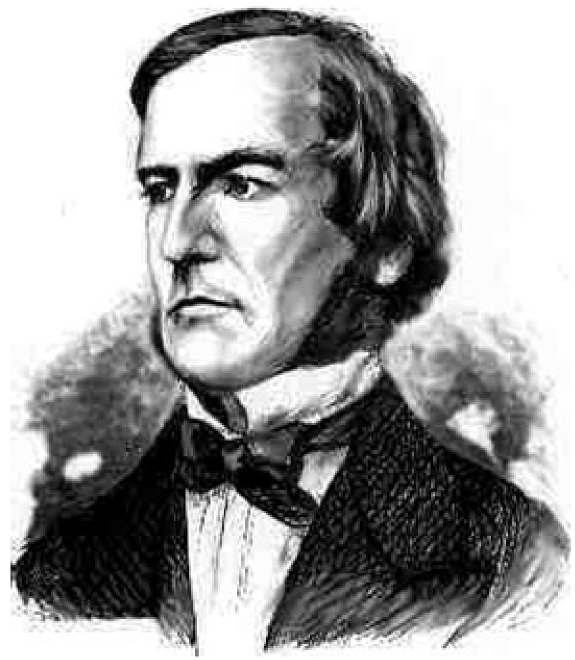
– $output_t = F(output_{t-1}, input_t)$
  - output dependent on history
  - concept of a time step (clock, t)

- Covered in CSE 369

Boolean Logic

Combinational Logic

– output = F(input)



Boolean Algebra: another notation for logic consisting of…

– a set of elements B = {0, 1}

– binary operations { + , • } (OR, AND)

– and a unary operation { ’ } (NOT)

x' $\neg x$

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

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

More generally $(a + b + c + \cdots)' = a' \bullet b' \bullet c' \bullet \cdots$

$$(a \bullet b \bullet c \bullet \cdots)' = a' + b' + c' + \cdots$$

Proving Theorems

2. commutativity:	a + b = b + a	a • b = b • a
3. associativity:	a + (b + c) = (a + b) + c	a • (b • c) = (a • b) • c
4. distributivity:	a + (b • c) = (a + b) • (a + c)	a • (b + c) = (a • b) + (a • c)
5. identity:	a + 0 = a	a • 1 = a
6. complementarity:	a + a’ = 1	a • a’ = 0
7. null:	~~a + 1 = 1~~	a • 0 = 0
8. idempotency:	a + a = a	a • a = a
9. involution:	(a’)’ = a	

Using the laws of Boolean Algebra:

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

Dist

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

$= X \cdot$

prove the “Absorption theorem”: $X + X \bullet Y = X$

$X + X \bullet Y =$

Proving Theorems

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

Using the laws of Boolean Algebra:

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

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

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

identity	$X + X \bullet Y = X \bullet 1 + X \bullet Y$
distributivity	$= X \bullet (1 + Y)$
commutativity	$= X \bullet (Y + 1)$
null	$= X \bullet 1$
identity	$= X$

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**

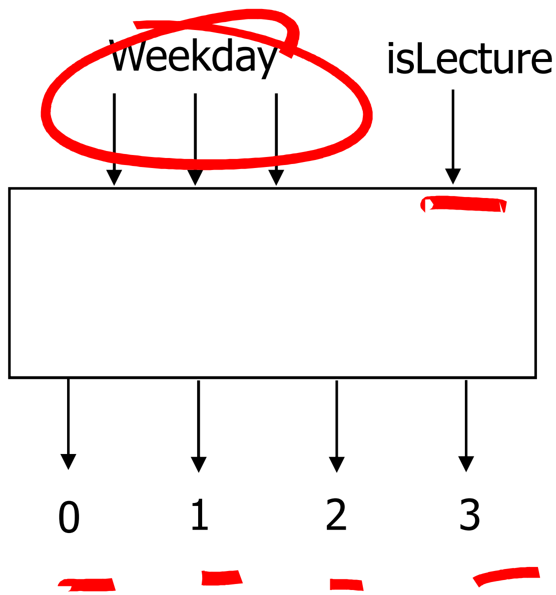
Implementation in Software

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

Weekday		isLecture	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 isLecture ? 3 : 1;
case TUESDAY or WEDNESDAY:
    return isLecture ? 2 : 1;
case THURSDAY:
    return isLecture ? 1 : 1;
case FRIDAY:
    return isLecture ? 1 : 0;
case SATURDAY:
    return isLecture ? 0 : 0;
```

Weekday		isLecture	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

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

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

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

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

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

How do we combine them?

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)

	$d_2d_1d_0$	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:

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

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

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

$c_2 = \underline{d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L}$

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:

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

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

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
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:

$d_2' \bullet d_1' \bullet d_0' \bullet L'$

$d_2' \bullet d_1' \bullet d_0 \bullet L'$

$d_2' \bullet d_1 \bullet d_0' \bullet L'$

$d_2' \bullet d_1 \bullet d_0 \bullet L'$

$d_2 \bullet d_1' \bullet d_0'$

$d_2 \bullet d_1' \bullet d_0 \bullet L$

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

$$c_3 = d_2' \bullet d_1' \bullet d_0' \bullet L + d_2' \bullet d_1' \bullet d_0 \bullet L$$
$$c_2 = d_2' \bullet d_1 \bullet d_0' \bullet L + d_2' \bullet d_1 \bullet d_0 \bullet L$$

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

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

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

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

Finally, we do c_0:

$d_2 \bullet d_1' \bullet d_0 \bullet L'$

$d_2 \bullet d_1 \bullet d_0'$

$d_2 \bullet d_1 \bullet d_0$

Truth Table to Logic (Part 4)

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

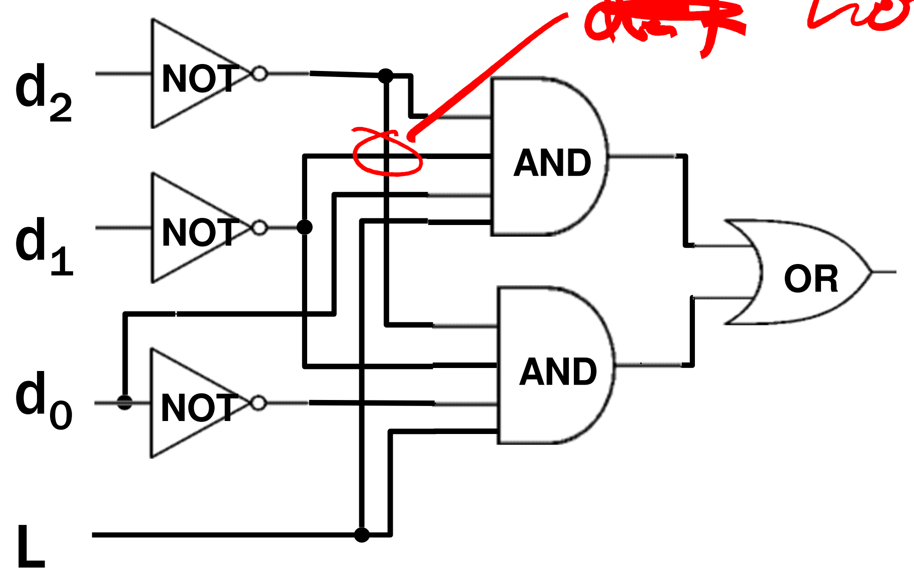
$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$

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

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

Here's c_3 as a circuit:

~~dot~~ not touching



Simplifying using Boolean Algebra

c3 = d2'•d1'•d0'•L + d2'•d1'•d0•L

distributive

= d2'•d1'•(d0' + d0)•L

= d2'•d1'•1•L

= d2'•d1'•L