CPSC 121: Models of Computation

Unit 2

Conditionals and Logical Equivalences

Based on slides by Patrice Belleville and Steve Wolfman

Pre-Class Learning Outcomes

- By the start of this class you should be able to
  - Translate back and forth between simple natural language statements and propositional logic, now with conditionals and biconditionals.
  - Evaluate the truth of propositional logical statements that include conditionals and biconditionals using truth tables
  - Given a propositional logic statement and an equivalence rule, apply the rule to create an equivalent statement.

Quiz 2 feedback

- Most frequent mistakes:
- Open-ended question?

In-Class Learning Goals

By the end of this unit, you should be able to:

- Explore alternate forms of propositional logic statements by application of equivalence rules, especially in order to simplify complex statements or massage statements into a desired form.
- Evaluate propositional logic as a "model of computation" for combinational circuits and identify at least one explicit shortfall (e.g., referencing gate delays, wire length, instabilities, shared sub-circuits, etc.)..

Where We Are inThe Big Stories

Theory
- How do we model computational systems?

Now:
- practicing a second technique for formally establishing the truth of a statement (logical equivalence proofs).
- (the first technique was truth tables.)

Hardware
- How do we build devices to compute?

Now:
- learning to modify circuit designs using our logical model
- gaining more practice designing circuits
- identifying a flaw in our model for circuits.

The Meaning of $\rightarrow$

- The meaning of **if p then q** in propositional logic is not quite the same as in normal language.
  - Consider:
    if it's at least 20ºC tomorrow, then I will come to UBC in shorts and T-shirt
  - Suppose it's -2ºC and snowing. Based on the above proposition, will I come to UBC in shorts and T-shirt?
    A. Yes
    B. No
    C. Maybe

The Meaning of $\rightarrow$

- Consider the proposition
  p: *If you fail the final exam, then you will fail the course*
- You need to distinguish between
  - The truth value of p (whether or not I lied).
  - The truth value of the conclusion (whether or not you failed the course).

The Meaning of $\rightarrow$

Consider again:
p: *If you fail the final exam, then you will fail the course*

- If you fail the final exam, will you pass the course?
  A. Yes
  B. No
  C. Maybe
- If you pass the final exam, will you pass the course?
  A. Yes
  B. No
  C. Maybe

More on the Meaning of $\rightarrow$

- Which of the following statements is equivalent to :

  if p then q ?

1. **if not p then not q**
   A. YES B. NO
2. **if not q then not p**
   A. YES B. NO
3. **p unless q**
   A. YES B. NO
4. **q unless not p**
   A. YES B. NO
5. **q only if p**
   A. YES B. NO
6. **p only if q**
   A. YES B. NO

p	q	$p \rightarrow q$	p unless q	p only if q
F	F	T		
F	T	T		
T	F	F		
T	T	T		

Note:
r unless s
means
if ~s then r

Equivalence Proofs

- How do we write a logical equivalence proof?
  - We state the theorem we want to prove.
  - We indicate the beginning of the proof by Proof:
  - We start with one side and work towards the other,
    - one step at a time, using the basic equivalences shown on next page
    - without forgetting to justify each step
    - usually we will simplify the more complicated proposition, instead of trying to complicate the simpler one.
  - We indicate the end of the proof by QED or $\square$

Basic Logical Equivalences

Name	Rule	Name	Rule
Identity law	$p \wedge T \equiv p$ $p \vee F \equiv p$	Domination law	$p \wedge F \equiv F$ $p \vee T \equiv T$
Idempotent law	$p \wedge p \equiv p$ $p \vee p \equiv p$	Commutative law	$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$
Associative law	$p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$ $p \vee (q \vee r) \equiv (p \vee q) \vee r$	Distributive law	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Absorption law	$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Negation law	$p \wedge \sim p \equiv F$ $p \vee \sim p \equiv T$
Double negative law	$\sim(\sim p) \equiv p$	DeMorgan's law	$\sim(p \wedge q) \equiv (\sim p) \vee (\sim q)$ $\sim(p \vee q) \equiv (\sim p) \wedge (\sim q)$
Definition of $\oplus$	$p \oplus q \equiv (p \vee q) \wedge \sim(p \wedge q)$	Definition of $\rightarrow$	$p \rightarrow q \equiv \sim p \vee q$
Contrapositive law	$p \rightarrow q \equiv (\sim q) \rightarrow (\sim p)$		

Writing an Equivalence Proof

Theorem: $(\sim a \wedge b) \vee a \equiv a \vee b$

Proof:

$(\sim a \wedge b) \vee a$
$\equiv a \vee (\sim a \wedge b)$ commutative
$\equiv (a \vee \sim a) \wedge (a \vee b)$ distribution
$\equiv$ ______________
$\equiv a \vee b$ identity

QED

What's missing?

a. $(a \vee b)$
b. $F \wedge (a \vee b)$
c. $a \wedge (a \vee b)$
d. None of these, but I know what it is.
e. None of these, and there's not enough information to tell.

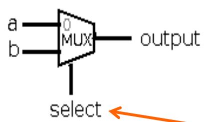
Examples of Equivalence Proofs

- Prove that
  - $\sim p \rightarrow \sim q \equiv q \rightarrow p$
  - $\sim p \wedge q \equiv (\sim p \vee q) \wedge \sim(\sim q \vee p)$
- We will do these on the board .

How Good Propositional Logic Is?

- Propositional Logic is not a perfect model of how gates work.
- To understand why, we will look at a multiplexer
  - A circuit that chooses between two or more values.
  - In its simplest form, it takes 3 inputs
    - An input **a**, an input **b**, and a control input **select**.
    - It outputs **a** if **select** is false, and **b** if **select** is true.

Multiplexer



call it **c**

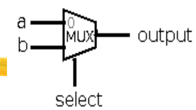
- Truth table:

a	b	c	output
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	T
T	F	T	F
T	T	F	T
T	T	T	T

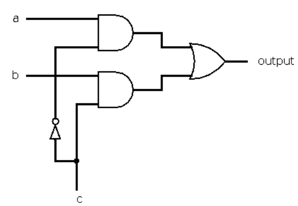
Outputs **a**'s value when **c** is F and **b**'s when **c** is T:

$(\sim a \wedge b \wedge c) \vee$
$(a \wedge \sim b \wedge \sim c) \vee$
$(a \wedge b \wedge \sim c) \vee$
$(a \wedge b \wedge c)$
$\equiv$
$(a \wedge \sim c) \vee (b \wedge c)$

MUX Design

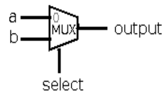


- Here is one possible implementation (call select "c"):



- Let us see why this may not work as we expect...

Truthy MUX

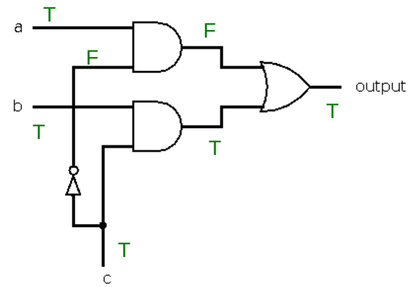


What is the intended output if both a and b are T?

A. T
B. F
C. Unknown... but could be answered given a value for c.
D. Unknown... and might still be unknown even given a value for c.

Glitch in MUX Design

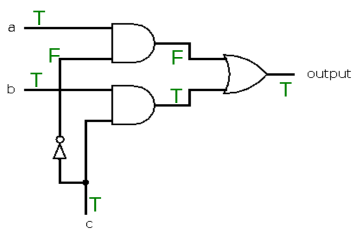
- Suppose the circuit is in steady-state with a, b, c all T



- Assume the gate delay is 10ns

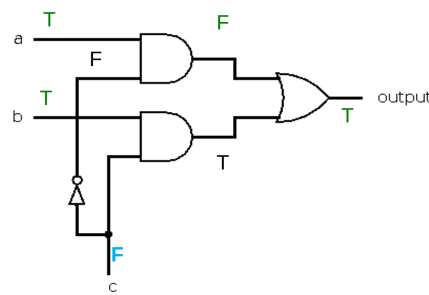
Trace

- How long will it take before output reflects any changes in a, b, c and is stable?
  - 5ns
  - 10ns
  - 20ns
  - 30ns
  - 40ns
  - It may never be stable
  - None of the above.



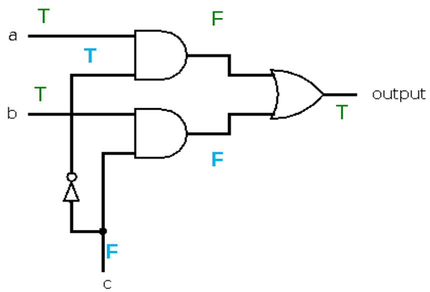
Trace – 5 ns

- Suppose that at time 0 we switch c to F.
- At time 5ns:



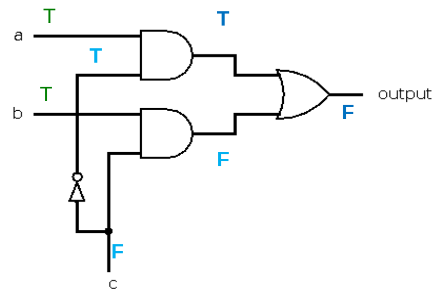
Trace – 10 ns

- At time 10ns:



Trace – 20 ns

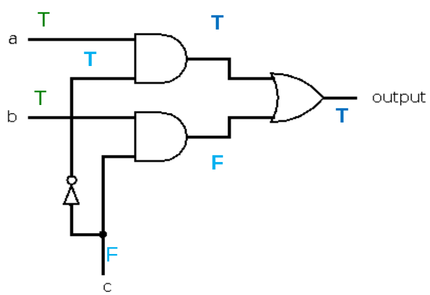
- At time 20ns:



- Note: the output is now F

Trace – 30 ns

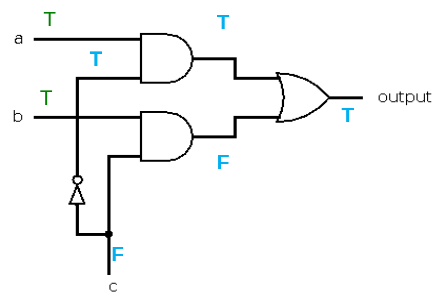
- At time 30ns:



- Note: the output is now T again.

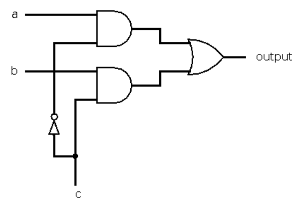
Trace – 40 ns

- At time 40ns:



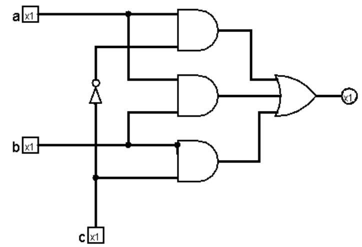
More MUX Glitches

- Cause of the problem: information from c travels two paths with different delays. Output can be incorrect until the longer path "catches up".
- Which one(s) of the following operation may cause an instability?
  - A. Changing a only
  - B. Changing b only
  - C. Changing c, when at least one of a, b is F
  - D. Both (a) and (b)
  - E. None of (a), (b) and (c)



A Correct Design for MUX

- Here is a multiplexer that avoid the instability:



- **Exercise:** Prove that it's logically equivalent to the original MUX
  - Hint: write (a ^ b) as (a ^ b ^ (c v ~c))

Exercises

- Consider the code:
  - if target = value then
    - if lean-left-mode = true then
      - call the go-left() routine
    - else
      - call the go-right() routine
  - else if target < value then
    - call the go-left() routine
  - else
    - call the go-right routine

Let gl mean "the go-left() routine is called". Complete the following:

- gl ↔

Exercises

- Consider the sentence: "Two strings s1 and s2 are equal if either both strings are null or neither s1 nor s2 is null and both strings have the same sequence of characters".
  - Let
    - n1: the string s1 is null
    - n2: the string s2 is null
    - eq: s1 and s2 are equal
    - s: the two strings have the same sequence of characters.
  - Is the given sentence logically equivalent to eq ↔ (n1 ^ n2) v s ?

Exercises

- Prove:

  (a ^ ~b) v (~a ^ b) ≡ (a v b) ^ ~ (a ^ b)

What is coming up?

- The third online quiz is due ____________________
  - Assigned reading for the quiz:
    - Epp, 4th edition: 2.5
    - Epp, 3rd edition: 1.5
    - Rosen, any edition: not much
      - http://en.wikipedia.org/wiki/Binary_numeral_system
    - Also read:
      - http://www.ugrad.cs.ubc.ca/~cs121/2009W1/Handouts/signed-binary-decimal-conversions.html
- Assignment #1 is due Friday, Jan 24 at 5:00pm.