271: Introduction to Digital Circuits and Systems

- Professor Scott Hauck, EEB-307Q
  (hauck@ee.washington.edu)
  - Office Hours: stop by or email w/schedule for a slot


- TAs (EEB-361):
  - Umaymah Khan (umaymah@uw.edu)
  - Lev Kurilenko (levkur@uw.edu)

- TA Office Hours: most times most weekdays (check website)

Grading

- 20% - Homeworks
- 30% - Labs
- 20% - Midterm Exam
- 30% - Final Exam
- Homework is due at the end of class on the specified date.
- Late penalties:
  - <24 hours: -10%
  - <48 hours: -30%
  - <72 hours: -60%
  - >72 hours: not accepted
Joint Work Policy

- Labs will be done alone, homeworks in groups of 1-2.
  - Students may not collaborate on labs/projects, nor between groups on the specifics of homeworks.
- OK:
  - Studying together for exams
  - Discussing lectures or readings
  - Talking about general approaches
  - Help in debugging, tools peculiarities, etc.
- Not OK:
  - Developing a lab together
  - Checking homework answers between groups
- Violation of these rules is at minimum:
  - Loss of twice the points of that assignment.
  - Report of Academic Misconduct to Dean’s Level.
  - Potentially fail class, be expelled from UW.

Class & Lab Meetings

- Labs:
  - Each student assigned a lab kit, can work where-ever.
  - **There are no specific assigned lab times.**
  - TAs have large blocks of office hours to help with labs, homeworks, class material, etc.
  - Signups for lab demos will be posted shortly.
- Midterm: Thu, Feb 8th, in class
- Final: Tues, March 13, 8:30-10:20
Motivation

- Readings: 1-1.4, 2-2.4
- Electronics an increasing part of our lives
  - Computers & the Internet
  - Car electronics
  - Robots
  - Electrical Appliances
  - Cellphones
  - Portable Electronics
- Class covers digital logic design & implementation

Example: Car Electronics

- Door Ajar (DriverDoorOpen, PassDoorOpen):
- High-beam indicator (lights, high beam selected)
Example: Car Electronics (cont.)

- Seat Belt Light (driver belt in):

- Seat Belt Light (driver belt in, passenger belt in, passenger present):

Basic Logic Gates

- AND: If A and B are True, then Out is True

- OR: If A or B is True, or both, then Out is True

- Inverter (NOT): If A is False, then Out is True
TTL Logic

Digital vs. Analog

Digital: only assumes discrete values
Binary/Boolean (2 values)
  yes, on, 5 volts, high, TRUE, "1"
  no, off, 0 volts, low, FALSE, "0"

Analog: values vary over a broad range continuously
Advantages of Digital Circuits

- Analog systems: slight error in input yields large error in output
- Digital systems more accurate and reliable
  - Readily available as self-contained, easy to cascade building blocks
- Computers use digital circuits internally
- Interface circuits (i.e., sensors & actuators) often analog

This course is about logic design, not system design (processor architecture), not circuit design (transistor level)

Combinational vs. Sequential Logic

**Sequential logic**

Network implemented from logic gates. The presence of feedback distinguishes between *sequential* and *combinational* networks.

**Combinational logic**

No feedback among inputs and outputs. Outputs are a function of the inputs only.
Black Box (Majority)

- Given a design problem, first determine the function
- Consider the unknown combination circuit a “black box”

Truth Table

![Black Box Diagram]

A B C → Out

“Black Box” Design & Truth Tables

- Given an idea of a desired circuit, implement it
  - Example: Odd parity - inputs: A, B, C, output: Out
**Boolean Elements**

*Algebra:* variables, values, operations

In Boolean algebra, the values are the symbols 0 and 1

- If a logic statement is false, it has value 0
- If a logic statement is true, it has value 1

**Operations:** AND, OR, NOT

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X AND Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>NOT X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X OR Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Boolean Equations**

*Boolean Algebra*

- values: 0, 1
- variables: A, B, C, …, X, Y, Z
- operations: NOT, AND, OR, …

- NOT X is written as \( \bar{X} \)
- X AND Y is written as \( X \cdot Y \), or sometimes \( X \& Y \) or \( X \& Y \)
- X OR Y is written as \( X + Y \)

**Deriving Boolean equations from truth tables:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Carry</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Carry =** OR'd together *product terms* for each truth table row where the function is 1

**Sum =** if input variable is 0, it appears in complemented form; if 1, it appears uncomplemented

---

15

16
### Boolean Algebra

**Another example:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Cout =

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Boolean Algebra (cont.)

**Reducing the complexity of Boolean equations**

Laws of Boolean algebra can be applied to carry out function to derive the following simplified expression:

Cout =

**Verify equivalence with the original Carry Out truth table:**

Place a 1 in each truth table row where the product term is true.

Each product term in the above equation covers exactly two rows in the truth table; a row can be "covered" by more than one term.