CSE140L: Digital Systems Laboratory

Introduction

Instructor: Mohsen Imani

Slides from:
Dr. Pietro Mercati & Prof. Tajana Simunic Rosing
Welcome to CSE 140L!

• **Instructor:** Mohsen Imani
  • Email: moimani@ucsd.edu;
    – please put “CSE140L” in the subject line
• **Office Hours:**
  – Mon 2-4pm CSE 2217
• **Website:** [https://cseweb.ucsd.edu/classes/su17_2/cse140L-a/index.html](https://cseweb.ucsd.edu/classes/su17_2/cse140L-a/index.html)
  – Homeworks and slides will be here
• **TAs and Tutors**
  – Office hours listed on the class website/Piazza
• **Discussion sessions:** TA will hold both lab and office hour
• **Grades:** [https://tritoned.ucsd.edu/](https://tritoned.ucsd.edu/)
• **Announcements and online discussion:** [https://piazza.com](https://piazza.com)
  → SIGN UP SOON !!!
• **Lab:** TBD
Course Description

• Prerequisites:
  – CSE 20 or Math 15A, and CSE 30.
  – CSE 140 must be taken concurrently

• Objective:
  – Introduce digital components and system design concepts through hands-on experience in a lab

• Grading:
  – Class participation using iClicker: 5%
  – Homeworks (5): #1: 0%, #2,#3: 10% , #4, #5: 15%
  – Final: 45%

• Homeworks:
  – Can be solved individually or in teams of two.
  – Each homework solution should be checked-off with a tutor or a TA before the end of the day on the due date
  – Detailed instructions will be provided on the homework
Textbook and Recommended Readings

- **Recommended textbook:**
  - Contemporary Logic Design by R. Katz & G. Borriello

- **Recommended textbook:**
  - Digital Design by F. Vahid

- Lecture slides are derived from the slides designed for both books
Demo Overview
Outline

• Transistors
  – How they work
  – How to build basic gates out of transistors
  – How to evaluate delay
Combinational circuit building blocks:
Transistors, gates and timing
The Scope of CSE140L

- We start with Boolean algebra $Y = A \text{ and } B$
- End up with the design of a simple CPU
Switches

• Electronic switches are the basis of binary digital circuits
  – Electrical terminology
    • **Voltage**: Difference in electric potential between two points
      – Analogous to water pressure
    • **Current**: Flow of charged particles
      – Analogous to water flow
    • **Resistance**: Tendency of wire to resist current flow
      – Analogous to water pipe diameter
    • \( V = I \times R \) (Ohm’s Law)

“Binary Digital” = all values are either 0 (low voltage) or 1 (high voltage)
The CMOS Switches

- CMOS circuit (Complementary – MOS)
  - Consists of N and PMOS transistors
  - Both N and PMOS are similar to basic switches
  - Rp ~ 2 Rn => PMOS in series is much slower than NMOS

Silicon -- not quite a conductor or insulator:

**Semiconductor**

1 = high voltage
0 = low voltage
Logic gates: CMOS NOT Gate

\[ Y = \overline{A} \]

<table>
<thead>
<tr>
<th>A</th>
<th>P1</th>
<th>N1</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>on</td>
<td>off</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>off</td>
<td>on</td>
<td>0</td>
</tr>
</tbody>
</table>
CMOS delay: resistance

- Resistance:
  - Function of:
    - resistivity $r$, thickness $t$: defined by technology
    - Width $W$, length $L$: defined by designer
  - Approximate ON transistor with a resistor
    - $R = r' \frac{L}{W}$
    - $L$ is usually minimum; change only $W$

\[
R = \frac{\rho L}{tW} = \frac{\rho}{t} \frac{L}{W}
\]

Source: Prof. Subhashish Mitra
CMOS delay: capacitance & timing estimates

- Capacitor
  - Stores charge \( = \) (capacitance \( C \); voltage \( V \))
  - Current: \( = \)

- Timing estimate
  - \( = \)

- Delay: time to go from 50% to 50% of waveform

Source: Prof. Subhashish Mitra
Charge/discharge in CMOS

• Calculate on resistance
• Calculate capacitance of the gates circuit is driving
• Get RC delay & use it as an estimate of circuit delay

\[ R_p \approx 2R_N \]

Source: Prof. Subhashish Mitra
CMOS Two Input NAND Gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P1</th>
<th>P2</th>
<th>N1</th>
<th>N2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>0</td>
</tr>
</tbody>
</table>
CMOS Two Input NOR Gate

\[(a+b)'\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P1</th>
<th>P2</th>
<th>N1</th>
<th>N2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>0</td>
</tr>
</tbody>
</table>
Rules for making gates

The Mathematical Method

- Given a logic function
  \[ F = f(a, b, c) \]

- Reduce (using DeMorgan) to eliminate inverted operations
  - inverted variables are OK, but not operations (NAND, NOR)

- Form pMOS network by complementing the inputs
  \[ F_p = f(\overline{a}, \overline{b}, \overline{c}) \]

- Form the nMOS network by complementing the output
  \[ F_n = f(a, b, c) = \overline{F} \]

- Construct \( F_n \) and \( F_p \) using AND/OR series/parallel MOSFET structures
  - series = AND, parallel = OR

- **EXAMPLE:**
  \[ F = \overline{ab} \Rightarrow \]
  \[ F_p = \overline{a} \overline{b} = a+b; \quad \text{OR/parallel} \]
  \[ F_n = \overline{ab} = ab; \quad \text{AND/series} \]
Another way of making CMOS gates

• Reducing Logic Functions
  - fewest operations ⇒ fewest txs
  - minimized function to eliminate txs
  - Example: \( x y + x z + x v = x (y + z + v) \)
    5 operations: 3 operations:
    3 AND, 2 OR 1 AND, 2 OR
    # txs = 30   # txs = 18

\[
\begin{align*}
\text{AND} &= \text{NAND} + \text{INV} \\
6 + xs &\quad 4 + xs &\quad 2 + xs
\end{align*}
\]

\[
\begin{align*}
\text{OR} &= \text{NOR} + \text{INV} \\
6 + xs &\quad 4 + xs &\quad 2 + xs
\end{align*}
\]

• Suggested approach to implement a CMOS logic function
  - create nMOS network
    • invert output
    • reduce function, use DeMorgan to eliminate NANDs and NORs
    • implement using series for AND and parallel for OR
  - create pMOS network
    • complement each operation in nMOS network
CMOS Example

- Construct the function below in CMOS
  \[ F = a + b \cdot (c + d); \] remember AND operations occur before OR

- **Step 1, invert output and find nMOS**
  - nMOS: implement \( a + b \cdot (c + d) \)
  - Group 1: c & d in parallel
  - Group 2: b in series with G1
  - Group 3: a parallel to G2

- **Step 2, complement operations**
  - pMOS
  - Group 1: c & d in series
  - Group 2: b parallel to G1
  - Group 3: a in series with G2
A CMOS design example

- Implement $F$ using CMOS: $F = A^*(B + C')$

\[
F_0 = F = A^* (B + C') = \overline{A} + (\overline{B} + \overline{C}) = \overline{A} + \overline{B} \cdot C
\]

\[
F_0 = F (\overline{A}, \overline{B}, \overline{C}) = A^* (\overline{B} + C) \Rightarrow \begin{cases} \overline{B} \text{ and } C \text{ parallel} \\ \overline{A} \text{ series with } \overline{A} \end{cases}
\]
What we’ve covered thus far

• Delay estimates
• Transistor design
• Building basic gates from CMOS