Announcements
- Homework #2 due today
- Homework #3 due next Thursday
- Labs start tomorrow

Class Material
- Last lecture
  - IC Manufacturing and Design Rules
- Today’s lecture
  - Overview of semiconductor memory
- Reading (Chapter 12.1, 12.2.3, 12.3.1)

Why Memory?

Semiconductor Memory Classification

<table>
<thead>
<tr>
<th>Read-Write Memory</th>
<th>Non-Volatile Read-Write Memory</th>
<th>Read-Only Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Access</td>
<td>Non-Random Access</td>
<td>EPROM</td>
</tr>
<tr>
<td>SRAM</td>
<td>FIFO</td>
<td>E²PROM</td>
</tr>
<tr>
<td>DRAM</td>
<td>LIFO</td>
<td>FLASH</td>
</tr>
<tr>
<td>Shift Register</td>
<td>CAM</td>
<td>Mask-Programmed Programmable (PROM)</td>
</tr>
</tbody>
</table>
Random Access Memories (RAM)

- **STATIC (SRAM)**
  - Data stored as long as supply is applied
  - Larger (6 transistors/cell)
  - Fast
  - Differential (usually)

- **DYNAMIC (DRAM)**
  - Periodic refresh required
  - Smaller (1-3 transistors/cell)
  - Slower
  - Single Ended

Basic Static Memory Element

- If D is high, D_b will be driven low
- Which makes D stay high
- Positive feedback

Random Access Chip Architecture

- Conceptual: linear array
  - Each box holds some data
  - But this does not lead to a nice layout shape
  - Too long and skinny

- Create a 2-D array
  - Decode Row and Column address to get data

Basic Memory Array

- **CORE:**
  - Keep square within a 2:1 ratio
  - Rows are word lines
  - Columns are bit lines
  - Data in and out on columns

- **DECODERS:**
  - Needed to reduce total number of pins: N+M address lines for \(2^{N+M}\) bits of storage.
  - Ex. if N+M=20 → \(2^{20}=1\text{Mb}\)

- **MULTIPLEXING:**
  - Used to select one or more columns for input or output of data

Writing into a Cross-Coupled Pair

- Access transistor must be able to overpower the feedback
Writing a “1”

Memory Cell

Complementary data values are written (read) from two sides.

SRAM Array Layout

65nm SRAM

ST/Philips/Motorola

Access Transistor

Pull down

Pull up

SRAM Column

Decoders

Intuitive architecture for N x M memory
Too many select signals:
N words == N select signals

Decoder reduces the number of select signals:
\[ K = \log_2 N \]
Row Decoders

Collection of $2^N$ complex logic gates
Organized in regular and dense fashion

\[
\begin{align*}
W_{L_0} &= \overline{A_0} \cdot A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7 \\
W_{L_{11}} &= \overline{A_0} \cdot A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \\
WL &= \overline{A_0}
\end{align*}
\]

NOR Decoder

\[
\begin{align*}
W_{L_0} &= \overline{A_0} + \overline{A_1} + \overline{A_2} + \overline{A_3} + \overline{A_4} + \overline{A_5} + \overline{A_6} + \overline{A_7} \\
W_{L_{11}} &= A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 + A_9
\end{align*}
\]

Decoder Design Example

- Look at decoder for 256x256 memory block (8KBytes)

Possible AND8

- Build 8-input NAND gate using 2-input gates and inverters
- Is this the best we can do?
- Is this better than using fewer NAND4 gates?

Possible Decoder

- 256 8-input AND gates
  - Each built out of tree of NAND gates and inverters
- Need to drive a lot of capacitance (SRAM cells)
  - What’s the best way to do this?

Problem Setup

- Goal: Build fastest, lowest possible power decoder with static CMOS logic
- What we know
  - Basically need 256 AND gates, each one of them drives one word line

Next Lecture

- Buffer delay optimization