ECE 2300
Digital Logic & Computer Organization
Spring 2024

Memories
Announcements

• Final exam is scheduled on Saturday May 11 @ 2pm

• Lab 3a due tomorrow

• In-class bonus questions
  – Bonus = 1 additional slip day
Review: Critical Path of 32-bit CLA

P’s and G’s for all CLA blocks are generated in parallel
### Review: Arithmetic Logic Unit (ALU)

<table>
<thead>
<tr>
<th>OP Name</th>
<th>OP</th>
<th>BSEL</th>
<th>CI</th>
<th>LOP</th>
<th>SOP</th>
<th>OSEL</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>000</td>
<td>00</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>Y = A + B + CI</td>
</tr>
<tr>
<td>SUB</td>
<td>001</td>
<td>01</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>Y = A + B' + 1</td>
</tr>
<tr>
<td>AND</td>
<td>011</td>
<td>00</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>01</td>
<td>Y = A AND B</td>
</tr>
<tr>
<td>OR</td>
<td>100</td>
<td>00</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>01</td>
<td>Y = A OR B</td>
</tr>
<tr>
<td>SHL</td>
<td>101</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>10</td>
<td>Y = A[6..0],SI</td>
</tr>
<tr>
<td>SHR</td>
<td>110</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>10</td>
<td>Y = SI,A[7..1]</td>
</tr>
<tr>
<td>PASS A</td>
<td>111</td>
<td>10</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>Y = A</td>
</tr>
</tbody>
</table>

**Diagram:**
- **Adder:** Add two inputs `A` and `B` and carry `CI`, producing a sum `Y` and carry `CO`.
- **Logical:** Perform AND, OR, and XOR operations on inputs `A` and `B` based on the LOP signal.
- **Shifter:** Shift left or right based on the SI and SOP signals.
- **Control Logic:** Determine the ALU operation based on the inputs `OP` and `OSEL`.
Our Microprocessor Needs (Large) Memory
Memory Organization: An Abstract View

- **Two dimensional array of bit cells**
  - Each bit cell stores 1 bit

- **Address selects a word with multiple bits**

![Diagram of memory organization]

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>010</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>01</td>
<td>110</td>
</tr>
<tr>
<td>00</td>
<td>011</td>
</tr>
</tbody>
</table>

A 3-bit word
General Memory Structure

Only one wordline is asserted at any given time
Types of Memories

• **Read-Only Memory (ROM)**
  – Truly read-only
    • Written in the factory, and never written after installation
  – Mostly read and rarely written
    • Much faster to read than write
  – Usually non-volatile

• **Random Access Memory (RAM)**
  – Dynamic RAM (DRAM), Static RAM (SRAM)
  – Read and write any location at similar speeds
  – Typically volatile: loses contents when powered off
    • New classes of non-volatile/persistent RAMs are emerging
Read-Only Memory (ROM) Structure

- n address lines
- decoder
- $2^n$ word lines
- memory array ($2^n \times m$)
- m bit lines
- fixed at factory or rarely written
Example ROM Implementation

address inputs

active-high decoder outputs

word line

resistors

decoder

$V_{DD}$

data output
Example ROM Implementation

Each NMOS "hardcodes" a 1

active-high decoder outputs

active-low bit lines, pulled LOW if transistor present

bit line

V_{DD}

resistors

data output

word line
Applications of ROM

- **Data & program storage**
  - e.g., Boot configuration for personal computers or application code for embedded systems

- **Combinational logic functions**
  - Lookup table
    - Address inputs = function inputs
    - Data outputs = function outputs
Using ROMs for Combinational Logic

\[
\begin{align*}
F_0 &= A' B' C' + A B' C + A B' C \\
F_1 &= A' B' C + A' B C' + A B C \\
F_2 &= A' B' C' + A' B' C + A B' C' \\
F_3 &= A' B C + A B' C' + A B C'
\end{align*}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F0</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Diagram:

- Decoder
- Memory array (8 words by 4 bits)
- 8 word lines
- F0 F1 F2 F3
Types of Memories

• Read-Only Memory (ROM)
  – Truly read-only
    • Written in the factory, and never written after installation
  – Mostly read and rarely written
    • Much faster to read than write
  – Usually non-volatile

• Random Access Memory (RAM)
  – Dynamic RAM (DRAM), Static RAM (SRAM)
  – Read and write any location at similar speeds
  – Typically volatile: loses contents when powered off
    • Note: new emerging classes of non-volatile/persistent RAMs are emerging
Dynamic RAMs (DRAMs)

• DRAM modules are usually off-chip and mounted on a separate printed circuit board

• In a typical DRAM, the smallest unit is the memory cell, which forms arrays
  – Multiple arrays make up banks; banks are organized into ranks
Common Organization of a DRAM Array*

- **Multiplexed address inputs to reduce pin count**
  - Row & column addresses are sent sequentially through a shared subset of pins
    - $\log_2 N$ bits each for an $N \times N$ bit cell array
    - Row address bits arrive first; followed by column address
  - **Row access**
    - Row-address strobe (RAS) loads row address bits into “row addr” latch to select 1 row
  - **Column access**
    - Column-address strobe (CAS) loads column address bits into “col addr” latch to select 1 column bit

---

*Optional reading: "How Does DRAM Work?" by Branch Education*
DRAM Bit Cell

- Capacitor accessed through a transistor

- Capacitor is charged or discharged to produce a 1 or a 0

- DRAM cells loses their state overtime and must be refreshed periodically, hence the name *dynamic*
Write a DRAM Bit Cell

- **Word line is asserted**

- **Bit line is driven with the desired value**
  - Capacitor is charged by the bit line to store a 1
  - Or discharged by the bit line to store a 0
Read a DRAM Bit Cell

- **Word line is asserted**

- **Bit line *precharged* halfway between 0 and 1**
  - Capacitor voltage pulls the bit line slightly higher or lower
  - *Sense amplifier* detects this small change (1 or 0)

- **Read destroys the stored value**
  - Need to rewrite the value afterwards
Read Access to a DRAM Array

DRAM bit cell array with 1024 rows x 1024 columns (1Mb)

1. 10 row address bits arrive first; RAS asserted
2. 1024 bits are read out to row data latch
3. 10 column address bits arrive later; CAS asserted
4. 1 bit returned through column MUX

Multiple DRAM arrays are read in parallel to retrieve a word
DRAM Refresh

• Capacitors discharge over time
  – Bit cells must be periodically refreshed (e.g., every 64ms, technology dependent)

• Refresh cycles recharge each memory bit
  – An entire row is refreshed at a time – each row periodically accessed using RAS (i.e., read access), which restores the charge
Static RAM (SRAM) Bit Cell

- **One cell requires six transistors**
  - The core is two cross-coupled inverters
- **Maintaining the state of the cell requires a constant power**
  - The cell is stable; no refresh cycles needed

6-Transistor (6T) SRAM Cell

BL = bit line
WL = word line
SRAM Write

- Drive one bit line high, the other low (depending on the desired value)
- Then turn on word line
- Bit lines over power cell with new value
SRAM Read

- Precharge both bit lines high
- Then turn on word line
- One of the two bit lines will be pulled down by the cell
  - Change detected by sensor amplifier
SRAM Architecture*

Address Decoder

Write Enable

Din_{n-1}  Din_{n-2}  ...  Din_1  Din_0

Column Driver  Column Driver  Column Driver  Column Driver

Sense Amp  Sense Amp  Sense Amp  Sense Amp

Dout_{n-1}  Dout_{n-2}  ...  Dout_1  Dout_0

BL  BL

BL  BL

BL  BL

BL  BL

*Optional reading: Understanding SRAM by SemiEngineering
Before Next Class

• H&H 7.1-7.3.4

Next Time

Single Cycle Microprocessor