Chapter 8 – Memory Basics
Overview

- Memory definitions
- Random Access Memory (RAM)
- Static RAM (SRAM) integrated circuits
- Arrays of SRAM integrated circuits
- Dynamic RAM (DRAM)
- Read Only Memory (ROM)
Memory Definitions

- **Memory** — A collection of storage cells together with the necessary circuits to transfer information to and from them.

- **Random Access Memory (RAM)** — a memory organized such that data can be transferred to or from any cell (or collection of cells) in a time that is not dependent upon the particular cell selected.

- **Memory Address** — A vector of bits that identifies a particular memory element (or collection of elements).
Memory Definitions (Continued)

- Typical data elements are:
  - **bit** — a single binary digit
  - **byte** — a collection of eight bits accessed together
  - **word** — a collection of binary bits whose size is a typical unit of access for the memory. It is typically a power of two multiple of bytes (e.g., 1 byte, 2 bytes, 4 bytes, 8 bytes, etc.)

- **Memory Data** — a bit or a collection of bits to be stored into or accessed from memory cells.

- **Memory Operations** — operations on memory data supported by the memory unit. Typically, *read* and *write* operations over some data element (bit, byte, word, etc.).
A basic memory system is shown here:

- \( k \) address lines are decoded to address \( 2^k \) words of memory.
- Each word is \( n \) bits.
- Read and Write are single control lines defining the simplest of memory operations.
Basic Memory Operations

- Memory operations require the following:
  - *Data* — data written to, or read from, memory as required by the operation.
  - *Address* — specifies the memory location to operate on. The address lines carry this information into the memory. Typically: $n$ bits specify locations of $2^n$ words.
  - *An operation* — Information sent to the memory and interpreted as control information which specifies the type of operation to be performed. Typical operations are READ and WRITE.
Basic Memory Operations (continued)

- **Read Memory** — an operation that reads a data value stored in memory:
  - Place a valid address on the address lines.
  - Activate the Read input.
  - Wait for the read data to become stable.

- **Write Memory** — an operation that writes a data value to memory:
  - Place a valid address on the address lines.
  - Apply the data to the data lines.
  - Toggle the memory write control line.
Basic Memory Operations (continued)

- Instead of separate Read and Write control lines, most ICs provide a Chip Select that selects the chip to be read from or written to and a Read/Write that determines the particular operation.
Basic Memory Operations (continued)

<table>
<thead>
<tr>
<th>Chip select CS</th>
<th>Read/Write R/W</th>
<th>Memory operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Write to selected word</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Read from selected word</td>
</tr>
</tbody>
</table>
RAM Integrated Circuits

- Types of random access memory
  - *Static* – information stored in latches
  - *Dynamic* – information stored as electrical charges on capacitors
    - Charge “leaks” off
    - Periodic *refresh* of charge required

- Dependence on Power Supply
  - *Volatile* – loses stored information when power turned off
  - *Non-volatile* – retains information when power turned off
Static RAM Cell

- Array of storage cells used to implement static RAM

- Storage Cell
  - SR Latch
  - Select input for control
  - Dual Rail Data Inputs B and \( \bar{B} \)
  - Dual Rail Data Outputs C and \( \bar{C} \)
Static RAM □ Bit Slice

- Represents all circuitry that is required for $2^n$ 1-bit words
  - Multiple RAM cells
  - Control Lines:
    - Word select $i$  
    - one for each word
    - Read / Write
    - Bit Select
  - Data Lines:
    - Data in
    - Data out
To build a RAM IC from a RAM slice, we need:

- **Decoder** decodes the n address lines to \(2^n\) word select lines
- A 3-state buffer
- on the data output permits RAM ICs to be combined into a RAM with \(c \times 2^n\) words
2^n-Word × 2-Bit RAM IC
Making Larger Memories

- Using the CS lines, we can make larger memories from smaller ones by tying all address, data, and R/W lines in parallel, and using the decoded higher order address bits to control CS.

- Using the 4-Word by 1-Bit memory from before, we construct a 16-Word by 1-Bit memory.
Making Larger Memories
Larger RAMs from Smaller RAMs

- We can use small RAMs as building blocks for making larger memories, by following the same principles as in the previous examples.
- As an example, suppose we have some 64K x 8 RAMs to start with:
  - $64K = 2^6 \times 2^{10} = 2^{16}$, so there are 16 address lines.
  - There are 8 data lines.
Making a Larger Memory

- We can put four 64K x 8 chips together to make a 256K x 8 memory.
- For 256K words, we need 18 address lines.
  - The two most significant address lines go to the decoder, which selects one of the four 64K x 8 RAM chips.
  - The other 16 address lines are shared by the 64K x 8 chips.
- The 64K x 8 chips also share WR and DATA inputs.
- This assumes the 64K x 8 chips have three-state outputs.
Analyzing the 256K x 8 RAM

- There are 256K words of memory, spread out among the four smaller 64K x 8 RAM chips.
- When the two most significant bits of the address are 00, the bottom RAM chip is selected. It holds data for the first 64K addresses.
- The next chip up is enabled when the address starts with 01. It holds data for the second 64K addresses.
- The third chip up holds data for the next 64K addresses.
- The final chip contains the data of the final 64K addresses.
Address Ranges

```
<table>
<thead>
<tr>
<th>Address Range</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 1111 1111 1111 1111 1111 (3ffff)</td>
<td>to</td>
</tr>
<tr>
<td>11 0000 0000 0000 0000 (30000)</td>
<td></td>
</tr>
<tr>
<td>10 1111 1111 1111 1111 (2ffff)</td>
<td>to</td>
</tr>
<tr>
<td>10 0000 0000 0000 0000 (20000)</td>
<td></td>
</tr>
<tr>
<td>01 1111 1111 1111 1111 (1ffff)</td>
<td>to</td>
</tr>
<tr>
<td>01 0000 0000 0000 0000 (10000)</td>
<td></td>
</tr>
<tr>
<td>00 1111 1111 1111 1111 (0ffff)</td>
<td>to</td>
</tr>
<tr>
<td>00 0000 0000 0000 0000 (00000)</td>
<td></td>
</tr>
</tbody>
</table>
```
Making Wider Memories

- To construct wider memories from narrow ones, we tie the address and control lines in parallel and keep the data lines separate.

- For example, to make a 4-word by 4-bit memory from 4, 4-word by 1-bit memories

- Note: Both 16x1 and 4x4 memories take 4-chips and hold 16 bits of data.
Making a Wider Memory

- Here is a 64K x 16 RAM, created from two 64K x 8 chips.
  - The left chip contains the most significant 8 bits of the data.
  - The right chip contains the lower 8 bits of the data.
Dynamic RAM (DRAM)

- Dynamic memory is built with capacitors.
  - A stored charge on the capacitor represents a logical 1.
  - No charge represents a logic 0.
- However, capacitors lose their charge after a few milliseconds. The memory requires constant refreshing to recharge the capacitors. (That’s what’s “dynamic” about it.)
- Dynamic RAMs tend to be physically smaller than static RAMs.
  - A single bit of data can be stored with just one capacitor and one transistor, while static RAM cells typically require 4-6 transistors.
  - This means dynamic RAM is cheaper and denser—more bits can be stored in the same physical area.
Dynamic RAM (DRAM)

- In practice, dynamic RAM is used for a computer’s main memory, since it is cheap and you can pack a lot of storage into a small space.
- The disadvantage of dynamic RAM is its speed.
- Real systems augment dynamic memory with small but fast sections of static memory called caches.
READ ONLY MEMORY (ROM)

- **Characteristics**
  - Perform read operation only, write operation is not possible
  - Information stored in a ROM is made permanent during production and cannot be changed
  - Organization

Information on the data output line depends only on the information on the address input lines.

--> Combinational Logic Circuit
An example

2 x 4 decoder

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