Overview of Chapter 6

- Part 1: Memories
  - Memory Definitions
  - Random Access Memory
    - Function, Operation, Timing
  - RAM Integrated Circuits
    - RAM Cell, RAM Bit Slice
    - 3-State Buffers
    - Cell Arrays and Coincident Selection
    - Dynamic RAM
  - Array of RAM Integrated Circuits
    - Making larger and wider memories
- Part 2: Programmable Logic Technologies
  - Introduction
  - Read-Only Memory
  - Programmable Logic Array (PLA)
  - Programmable Array Logic (PAL)
  - Examples and implementations
Memory Definitions

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

- **Memory Organization** - the basic architectural structure of a memory in terms of how data is accessed.

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

- **Memory Address** - A collection of binary digits that identify 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 sized data element (bit, byte, word, etc.).
Memory Organization

- Organized as an indexed **array of words**. Value of the index for each word is the memory address.
- Often organized to fit the needs of a particular computer architecture. Some historically significant computer architectures and their associated memory organization:
  - Digital Equipment Corporation PDP-8 – used a 12-bit address to address 4096 12-bit words.
  - IBM 360 – used a 24-bit address to address 16,777,216 8-bit bytes, or 4,194,304 32-bit words.
  - Intel 8080 – (8-bit predecessor to the 8086 and the current Intel processors) used a 16-bit address to address 65,536 8-bit bytes.

Memory Block Diagram

- 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.
Memory Organization (Example)

- Example memory contents:
  - A memory of 3 address bits, 8 data bits will have:
  - \( k = 3 \) and \( N = 8 \) so \( 2^3 = 8 \) Addresses labeled 0 to 7.
  - \( 2^3 = 8 \) Words of 8-bit data

<table>
<thead>
<tr>
<th>Memory Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
</tr>
<tr>
<td>1 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 0</td>
</tr>
<tr>
<td>1 1 1</td>
</tr>
</tbody>
</table>

Basic Memory Operations

- Memory operations require the following:
  - **Data** - data written to, or read from, memory as required by the operation.
  - **Address** - used to specify a range of indices the memory is to operate on. The address lines carry this information to 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 DATA, and WRITE DATA. Others are READ followed by WRITE and a variety of new operations associated with delivering block of data.
Basic Memory Operations (Continued)

- **Read Memory** -- These operations are used to read a data value stored in memory:
  - Place a valid address on the address lines.
  - Wait for the read data to become stable.
- **Write Memory** -- These operations are used to write a data value to memory:
  - Place a valid address on the address lines and valid data on the data lines.
  - Assert memory write enable line.
- Sometimes the read or write enable line is defined as a clock with precise timing information (e.g. Read Clock, Write Strobe).
  - Otherwise, it is just an interface signal.
  - Sometimes memory must acknowledge that it has completed the operation.

Memory Operation Timing

- The most basic memories are asynchronous
  - Storage is performed by latches or storage of electrical charge
  - Do not use a clock
- Controlled by application of control inputs and address
- Timing of signal application is critical to the operation
- See Figure 6-4 in text
  - Control Signals: Memory Enable, Read/Write
  - Relative timing of signals for Write and Read
RAM Integrated Circuits

- **Types of Random Access Memory (RAM)**
  - **Static** – Information stored in latches
  - **Dynamic** – Information stored as electrical charges on capacitors
    - Charge “leaks” off
    - Refresh required

- **Dependence on Power Supply**
  - **Volatile** – Lose 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**
- **Each storage cell consists of:**
  - A latch
  - Cell write logic
  - Cell read logic
- **See Figure 6-5 in text for example – A logical representation of electronic circuitry**
  - SR Latch for storage
  - Select input for control
  - Dual Rail Data Inputs B and ̅B
  - Dual Rail Data Outputs C and ̅C
Static RAM Bit Slice

- Represents all of the circuitry that is required to store multiple 1-bit words
- See Figure 6-6 in text as an example
  - Multiple RAM cells
  - Control Lines:
    - Word select i – one for each word
    - Read/Write
    - Bit Select
  - Data Lines:
    - Data in
    - Data out

n-Word × 1-Bit RAM IC

- To build a RAM IC from a RAM slice, we need:
  - A decoder decodes the \( \log_2 n \) address lines to \( n \) word select lines
  - A 3-state buffer on the data output permits RAM ICs to be combined into a RAM with \( c \times n \) words
- See Figure 6-7 in text as an example
  - Add 4-to 16 decoder with address inputs and word select outputs
  - Add 3-state buffer controlled by chip select
### 3-state Buffers and Logic

- **Three-State Logic** – Sometimes called tri-state logic but tri-state is a registered trademark of National Semiconductor – has three states for logic levels:
  - Active high state (output is driven high)
  - Active low state (output is driven low)
  - High impedance (Hi-Z) state (output is not driven)
- Especially useful for replacing "open collector" wire-OR or wire-AND oriented busses.
- Often registers made of latches and flip-flops have three-state outputs.
- Commonly used for memory components

### 3-State Buffer Basics

- The basic 3-State Buffer is shown:
- **Hi-Z:** Means the output is not "driven"
- The output "floats" to some level which is usually neither the "1" nor "0" in this state.
- When EN = "1", the output follows the input.
- The buffer can also invert data
- **EN** can be an inverted signal

<table>
<thead>
<tr>
<th>EN</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>Hi-Z</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>
3-State Logic Basics (Continued)

- Making Multiplexers with Tri-state Devices:

<table>
<thead>
<tr>
<th>EN1</th>
<th>EN0</th>
<th>IN1</th>
<th>IN0</th>
<th>OL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Hi-Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>damage</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>damage</td>
</tr>
</tbody>
</table>

Cell Arrays and Coincident Selection

- Memory arrays can be very large =>
  - Large decoders
  - Large fanouts for the bit lines
- The decoder size and fanouts can be reduced to approximately the $\sqrt{n}$ by using a coincident selection in a 2-dimensional array
  - Uses two decoders, one for words and one for bits
  - Word select becomes Row select
  - Bit select becomes Column select
- See Figure 6-10 for example
  - $A_3$ and $A_2$ used for Row select
  - $A_1$ and $A_0$ for Column select
RAM ICs with > 1 Bit/Word

- Word length can be quite high.
- To better balance the number of words and word length, use ICs with > 1 bit/word
- See Figure 6-11 for example
  - 2 Data input bits
  - 2 Data output bits
  - Row select selects 4 rows
  - Column select selects 2 pairs of columns

Dynamic RAM (DRAM)

- Basic Principle: Storage of information on capacitors.
- Charge and discharge of capacitor to change stored value
- Use of transistor as “switch” to:
  - Store charge
  - Charge or discharge
- See Figure 6-12 in text for circuit, hydraulic analogy, and logical model.
Dynamic RAM (DRAM)

- Block Diagram – See Figure 6-13 in text
- Refresh Controller and Refresh Counter
- Read and Write Operations
  - Application of row address
  - Application of column address
  - Why is the address split?
  - Why is the row address applied first?
- Timing – See Figure 6-15 in text

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 higher order address bits to decode CS.
- Using the 4-Word by 1-Bit memory from before, we construct a 16-Word by 1-Bit memory. ⇒
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.