Overview of Chapter 6 – Part 2

- RAM Integrated Circuits
  - Dynamic RAM
- Array of RAM Integrated Circuits
  - Arrays of Static and Dynamic RAMs
- Programmable Logic Technologies
  - Introduction
  - Read-Only Memory
  - Programmable Logic Array
  - Programmable Array Logic
  - VLSI Programmable Logic
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.
Programmable Logic

**IC Cost Dilemma:**
- IC logic circuit density increases exponentially with time.
- The more ICs you make, the cheaper they get.
- Complex logic ICs have very specific functions (so you make fewer).

**Question:** How do I make very high volume parts that are very dense?
- Answer #1: You make microprocessors or ICs (like automobile ignition controllers) that have very large volume. OR
- Answer #2: You make programmable parts.

Programmable Part Types

- Semiconductor manufacturers have developed several types of regular programmable logic elements. Three important ones are:
  - Read Only Memory (ROM) -- a fixed array of AND gates and a programmable array of OR gates.
  - Programmable Array Logic (PAL) -- a programmable array of AND gates feeding a fixed array of OR gates.
  - Programmable Logic Array (PLA) -- a programmable array of AND gates feeding a programmable array of OR gates.

- All of the above use regular structures of logic elements that can be thought of as a "memory array". There are also programmable devices that look more like programmable logic cells with programmable interconnect.
Programming Devices

- Devices may be:
  - Permanently programmed at the time of IC manufacture,
  - Programmed at the time of use (board level manufacturing), or
  - Dynamically re-programmed during use.
- Permanent programming techniques done at the time of manufacture include final level interconnect addition via metallization or device alteration through laser or e-beam programming.
- Use time programming techniques include shorting diodes, blowing fuses, shorting devices, and dumping charge into wells.
- Dynamically reprogrammed devices can be bulk erased and reprogrammed, or incrementally erased and reprogrammed.

Read Only Memory

- Read Only Memories (ROM) or Programmable Read Only Memories (PROM) have:
  - N input lines,
  - M output lines, and
  - $2^N$ decoded minterms.
- The N input lines are connected to a fixed decoder AND array of $2^N$ lines. Each line represents a minterm of N variables. Thus there are $2^N$ decoded minterms.
- Each of the M outputs lines are connected to an OR gate which has a programmable number of input connections. Any (or all) of the minterms may be ORed together for each of the M output lines.
- A program map for a PROM (or ROM) LOOKS LIKE A MULTIPLE OUTPUT FUNCTION TABLE.
Read Only Memories (Continued)

- **Example: A 8 X 4 ROM**
  
  \( N = 3 \) input lines, \( M = 4 \) output lines

- The fixed "AND" array is a decoder with 3 input bits to one-of-8 minterm output lines.

- The programmable "OR" array is shown as a "Wire-OR" function.
  A "Dot" in the array corresponds to including that minterm.

\[
\begin{array}{cccc}
\text{D7} & \text{D6} & \text{D5} & \text{D4} \\
A & B & C \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{D3} & \text{D2} & \text{D1} & \text{D0} \\
A_2 & A_1 & A_0 \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{F0} & \text{F1} & \text{F2} & \text{F3} \\
\end{array}
\]

Read Only Memories (Continued)

- **The 32 X 8 ROM example corresponds to the multiple output truth table:**

- The "internal organization" of the memory array often does not match the "logical organization".

- For example, one manufacture sells an \( N=13 \) (inputs decoded to 8192 minterms) by \( M=8 \) (outputs) PROM that is internally organized as a 128 rows by 512 columns bit array. An output MUX selects a group of 8 bits.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>F0 F1 F2 F3</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 1 0 0</td>
</tr>
</tbody>
</table>
Programmable Array Logic (PAL)

- PAL devices are closely related to the ROM in that the device is organized as a regular array of programmable elements.
- The PAL has a programmable set of AND terms, combined with a limited number of fixed OR terms.
- Where the ROM array is guaranteed to implement any function of N inputs, the PAL may run out of OR terms. Thus, it may be very important to minimize the number of OR terms in order to use a PAL.
- Another difference is that a ROM does not easily allow multi-level implementations. The designer must use separate ROMs for multiple levels. The PAL allows outputs from OR terms to be used as inputs to AND terms, making multi-level design easy.

Programmable Array Logic (Cont.)

- Example: 4 Input, 3 Output PAL with fixed, 3-input OR terms and programmable polarity outputs.
- This device is unprogrammed.
Programmable Array Logic (Cont.)

- An "X" at a cross line includes that variable in the AND term. An "X" in an AND gate removes that term. An "X" at the EXOR forms a "TRUE" term, else the output is complemented. Thus we have:
  - Out1 = (In1 In2 In3 + In1)

- Note that Out1 leaves the PAL in COMPLEMENT form, since the EXOR is tied to one. Out2 is shown in "TRUE" form.

What are the equations for the other terms?

Logic and Computer Design Fundamentals
© 2001 Prentice Hall, Inc
Programmable Logic Array (PLA)

- The last type of programmable logic element we will discuss is the PLA which has a programmable array of AND and OR terms.
- A PLA typically has a large number of inputs and outputs and can be used to implement equations that are impractical for a ROM (because of the number of inputs required).
- Generally the product terms limit the application of a PLA. Use minimization techniques to reduce the number of product terms in an implementation if it is to fit in a PLA.
- The program for a PLA is very similar to the connection array for a multiple output Boolean function, such as that generated by CAFE.

Summary of Programmable Logic

- **Device Characteristics**
- **Device Uses**

<table>
<thead>
<tr>
<th>Type</th>
<th>AND terms</th>
<th>OR terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>$2^N$ Fixed terms</td>
<td>Programmable</td>
</tr>
<tr>
<td>PAL</td>
<td>Programmable</td>
<td>Fixed</td>
</tr>
<tr>
<td>PLA</td>
<td>Programmable</td>
<td>Programmable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>Single level, SOP, multiple output functions. Directly implements function tables.</td>
</tr>
<tr>
<td>PAL</td>
<td>Two or more level SOP implementations with output polarity adjust. Can't implement all functions due to limited OR terms.</td>
</tr>
<tr>
<td>PLA</td>
<td>Two or more level SOP. Limited AND and OR terms can limit applicability.</td>
</tr>
</tbody>
</table>
ROM, PAL, PLA Examples

- **RECALL: Square Root of a Number (INT)**
  
  \[
  \begin{array}{c|c|c}
  \text{INT(0)} & y & x \\
  \hline
  1 & 1 & 1 \\
  1 & 1 & 1 \\
  1 & 1 & 1 \\
  1 & 1 & 1 \\
  \end{array}
  \quad
  \begin{array}{c|c|c}
  \text{INT(1)} & y & x \\
  \hline
  1 & 1 & 1 \\
  1 & 1 & 1 \\
  1 & 1 & 1 \\
  \end{array}
  \]

  \[
  \begin{array}{c|c}
  w & z \\
  \hline
  1 & 1 \\
  1 & 1 \\
  1 & 1 \\
  \end{array}
  \quad
  \begin{array}{c|c}
  w & z \\
  \hline
  1 & 1 \\
  1 & 1 \\
  \end{array}
  \]

- **Implementations**

<table>
<thead>
<tr>
<th>Device</th>
<th>RND</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MUX</td>
<td>Yes (3)</td>
<td>Yes (2)</td>
</tr>
<tr>
<td>Decoder</td>
<td>Yes (2)</td>
<td>Yes(2)</td>
</tr>
</tbody>
</table>

\[
\text{INT(1)} = w + x \\
\text{INT(0)} = x \cdot y + x \cdot z + w \cdot x
\]

ROM, PAL, PLA Implementations

- **Device**
  
<table>
<thead>
<tr>
<th>Device</th>
<th>RND</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>16x2</td>
<td>16x3</td>
</tr>
<tr>
<td>PAL</td>
<td>Maybe</td>
<td>Yes</td>
</tr>
<tr>
<td>PLA</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Note 1:** "OR" terms might limit application.

**Function Table**

<table>
<thead>
<tr>
<th>NUM</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00</td>
</tr>
<tr>
<td>0001</td>
<td>01</td>
</tr>
<tr>
<td>0010</td>
<td>01</td>
</tr>
<tr>
<td>0011</td>
<td>01</td>
</tr>
<tr>
<td>0100</td>
<td>10</td>
</tr>
<tr>
<td>0101</td>
<td>10</td>
</tr>
<tr>
<td>0110</td>
<td>10</td>
</tr>
<tr>
<td>0111</td>
<td>10</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>1001</td>
<td>11</td>
</tr>
<tr>
<td>1010</td>
<td>11</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>11</td>
</tr>
<tr>
<td>1101</td>
<td>11</td>
</tr>
<tr>
<td>1110</td>
<td>11</td>
</tr>
<tr>
<td>1111</td>
<td>11</td>
</tr>
</tbody>
</table>
ROM Implementation

- This implementation is trivial – the function table serves as the fuse map. A 16-word by 2-bit PROM is needed (we will use a 16x4 PROM as shown):
- Try programming it yourself first.

ROM Implementation (Continued)

- Here is the solution (check it again the function table).
- DOT means a connection exists.

<table>
<thead>
<tr>
<th>NUM</th>
<th>INT</th>
<th>NUM</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00</td>
<td>0000</td>
<td>10</td>
</tr>
<tr>
<td>0001</td>
<td>01</td>
<td>0001</td>
<td>11</td>
</tr>
<tr>
<td>0010</td>
<td>01</td>
<td>0010</td>
<td>11</td>
</tr>
<tr>
<td>0011</td>
<td>01</td>
<td>0011</td>
<td>11</td>
</tr>
<tr>
<td>0100</td>
<td>10</td>
<td>0100</td>
<td>11</td>
</tr>
<tr>
<td>0101</td>
<td>10</td>
<td>0101</td>
<td>11</td>
</tr>
<tr>
<td>0110</td>
<td>10</td>
<td>0110</td>
<td>11</td>
</tr>
<tr>
<td>0111</td>
<td>10</td>
<td>0111</td>
<td>11</td>
</tr>
</tbody>
</table>
PAL Implementation

- For the PAL implementation, we go back to the minimized equations:
  \[ \text{INT}(1) = w + x \]
  \[ \text{INT}(0) = x y + x z + w x \]
- Try your hand at programming it!

PAL Implementation (Continued)

- Here's an implementation:
  \[ \text{INT}(1) = w + x \]
  \[ \text{INT}(0) = x y + x z + w x \]
- Is it correct?
PLA Implementation

- For the PAL implementation, we also use the minimized equations:
  \[ \text{INT}(1) = w + x \]
  \[ \text{INT}(0) = x y + x z + w x \]

- Try your hand at programming it!

PLA Implementation (Continued)

- Here is an implementation:
  \[ \text{INT}(1) = w + x \]
  \[ \text{INT}(0) = x y + x z + w x \]

- Is it correct?