Lecture 8: ROM & Programmable Logic Devices

Professor Peter Cheung
Department of EEE, Imperial College London

(Floyd 10.1, 10.3-5, 11.1-11.3)
(Tocci 12.1, 12.4-5, 13.1-13.4)

Points Addressed in this Lecture

- Read-only memory
- Implementing logic with ROM
- Programmable logic devices
- Implementing logic with PLDs
- Static hazards

Memory Terminology

- **Memory Cell**: circuit that stores 1-bit of information
- **Memory Word**: 8 – 64 bits
- **Byte**: a group of 8 bits
- **Capacity (=Density)**
  - 4096 20-bit words
  - 81,920 bits = 4096*20 = 4K*20
  - 1 M or 1 meg = $2^{20}$
  - 1 G or 1 giga = $2^{30}$
- **Address**
- **Read Operation** (=fetch operation)
- **Write Operation** (=store operation)

Addresses

<table>
<thead>
<tr>
<th>Addresses</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Word 0</td>
</tr>
<tr>
<td>001</td>
<td>Word 1</td>
</tr>
<tr>
<td>010</td>
<td>Word 2</td>
</tr>
<tr>
<td>011</td>
<td>Word 3</td>
</tr>
<tr>
<td>100</td>
<td>Word 4</td>
</tr>
<tr>
<td>101</td>
<td>Word 5</td>
</tr>
<tr>
<td>110</td>
<td>Word 6</td>
</tr>
<tr>
<td>111</td>
<td>Word 7</td>
</tr>
</tbody>
</table>
Read-only Memory (ROM)

- A ROM cell can store 1 bit of information
- Data can be read but not changed (written)
  - RAM is read-write capable
- ROM is non-volatile
  - the data is "remembered" even when the power supply to the chip is turned off
  - the data can be read after turning the power on again
  - RAM is volatile
- Applications
  - permanent storage of programmes for micro-processors
  - look-up tables of data
  - implementing combinational logic

Notes
- 6 address inputs - half for row select, half for column select
- row select energises all the switch transistors in one row
- column select uses a multiplexer to select just one column
- outputs are normally high but "pulled down" if a cell is programmed

A ROM Device
- E.g. 64x1 bit ROM

A ROM Cell
- A voltage is stored to represent a 0 or 1 as required
- If the “row-line” is addressed, the switch closes and the stored voltage appears on the “column-line”
- The switch is implemented with a (MOS) transistor
**Storage Mechanism**

- The storage mechanism for the 0 or 1 depends on the design of the ROM
- Mask Programmed ROM
  - the mask programmed ROM is programmed at the time of manufacture
  - the switch transistor is made to have a low threshold voltage to program a 0 and a high threshold to program a 1

![Mask Programmed Cell](image)

**Field Programmable ROM (PROM)**
- programmable using a PC-based system
- a semiconductor fuse is blown to program a cell to 1

**Electrically Erasable Programmable ROM (EEPROM)**
- programmable using a PC-based system
- the gate capacitance of a MOS transistor is charged (electrically) to store a 0 or discharged (electrically) to store a 1

![PROM Cell](image)

**Different ROM technology**

- In-circuit, electrically erasable byte-by-byte
- In-circuit, electrically erasable by sector or in bulk (all cells)
- UV erasable in bulk; erased and reprogrammed out of circuit
- Cannot be erased and reprogrammed

![Device complexity and cost](image)

**Implementing Logic with ROM**

- \(2^n \times 1\)-bit ROM devices have \(n\) inputs and 1 output
- they can be used to implement logic directly
- E.g. \(\text{OUT} = X \cdot Y + Z\)

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Programme the first 8 addresses according to the OUT column
- Connect X to A2, Y to A1 and Z to A0 of the ROM
- Usually "cheaper" than using gates for complex logic involving many variables
Programmable Logic Devices (PLDs)

- Several different "architectures" available
  - We will focus only on PAL and CPLD
- PALs and CPLDs are examples of PLDs
  - PAL: Programmable Array Logic
  - CPLD: Complex Programmable Logic Device
  - implement SOP expressions in canonical form
  - made from a Programmable AND section and a Fixed OR section
  - typically can implement 8 product terms

PALs are programmed like PROMs using fuses
  - one-time programmable
CPLDs are programmed like EEPROMs
  - Electrically erasable
PLAs
  - Devices with programmable AND and programmable OR

PAL Architecture

\[ f = A_0 \cdot A_1 \cdot A_2 \cdot A_4 \cdot \overline{A_6} + \overline{A_1} \cdot A_3 \cdot A_4 \]

Detail of AND Gates in PALs

4 bit example

other inputs not shown
Summary of Combinational Logic Building Blocks

- **Gates**
  - seven basic gates from which all other circuits are made
  - AND/NAND, OR/NOR, XOR/XNOR, NOT

- **Multiplexers**
  - act as switches to connect one output to one of a number of input signals
  - can also be used to implement logic

- **Decoders**
  - inverted multiplexers (sometimes called demultiplexers)
  - act as switches to connect one input to one of a number of output signals
  - also includes circuits such as Binary to 7 Segment decoders
  - four to seven bit decoders

- **Arithmetic Circuits**
  - binary adders, comparators, multipliers
  - issues of signed or unsigned number representation are important

- **Programmable Logic Devices**
  - **ROMs**
    - implement arbitrary logic functions
    - efficient for large combination logic circuits
  - **CPLDs**
    - implement canonical SOP Boolean expressions
    - **Advantages:**
      - reduction in chip count
      - easy upgrade by just reprogramming
    - **Disadvantages:**
      - programming equipment required
      - non-standard parts to stock and document

Static Hazards

- Gates have finite propagation delay
  - This can cause glitches in logic waveforms
- Consider an inverter
  - propagation delay of ~2nS
- E.g. Implementation of
  \[ f(A, B, C) = AB + \overline{AC} \]
  - If we use an inverter to generate \( \overline{A} \) from \( A \) then changes in \( \overline{A} \)
  - will be later than changes in \( A \).

Avoid Static Hazards

- Using a Karnaugh map, look for groups of minterms which do NOT overlap
  - These are potential hazards
- Avoid the hazard by introducing additional groups so that no non-overlapping groups remain
  \[
  
  \begin{array}{c|ccc}
  A \setminus BC & 00 & 01 & 11 & 10 \\
  \hline 
  0 & & & & \\
  1 & & & & \\
  \end{array}
  \]
- Groups are \( \overline{AC} + AB \) which do not overlap
  - Potential hazard
- Introduce the additional term \( BC \) to avoid the hazard
  \[ \overline{AC} + AB + BC \]