Read Only Memory (ROM)

- Device that allows **permanent** storage of information.
- Device has \( k \) input (address) lines and \( n \) output (data) lines.
- We can store \( 2^k \leq n \) bits of information inside the device.
- The address lines specify a memory location;
  - The data outputs at any time represents the value stored at the memory location specified on the address lines.
ROM Block Diagram

- High level block diagram for a ROM:

```
   k inputs  →  2^k x n ROM  →  n outputs
```

**k inputs**  →  **2^k x n ROM**  →  **n outputs**
ROM Block Diagram

- Uses an address decoder such that the k address lines selects one word of the $2^k$ words of data stored in the ROM.

- Each of the $2^k \leq n$ bits inside of the ROM are programmable via opening and/or closing switches.
Implementing Functions With ROM

- Can implement multi-input/multi-output logic functions inside of ROM.

- Data outputs are the logic functions and the address lines are the logic function inputs.

- We create a **ROM Table** to store the logic functions.
  - When an input (or address) is presented, the value stored in the specified memory location appears at the data outputs.
  - Each data output represents the correct value for its logic function.
Implementing Functions With ROM

- E.g., Implement the 3-input logics $f_0 = \sum (0, 1, 5, 7)$, $f_1 = \sum (0, 1, 2, 6)$ and $f_2 = \sum (2, 3, 4)$ using a ROM.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$f_2$</th>
<th>$f_1$</th>
<th>$f_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

3-to-8 decoder

8 x 3 ROM
Types of ROM

- Several technologies for implementing ROM:
  - PROM (Programmable Read-Only Memory):
    - PROM contains fuses giving logic value of 1 to all bits in device. Programming means “blowing” fuses to give some bits a logic value of 0.
    - Once programmed, that’s it - programming cannot be changed.
  - EPROM (Electrically Programmable Read-Only Memory):
    - Can be “erased” by exposure to UV light. Otherwise, same as PROM.
  - EEPROM (Electrically Erasable Programmable Read-Only Memory):
    - Can be “erased electrically”. Otherwise, same as a PROM.
  - In programming, these devices all have an extra pin (or extra pins) where the programming information (bit stream) is applied to do the programming.
Textbook

- ROM are described in Chapter 7, Section 7.5 of the textbook.
Random Access Memory (RAM)

- Storage device to which we can both read and write information.

![Diagram of RAM]

- data inputs
- address
- k
- read
- write
- $2^k \times n$
- RAM
- data outputs
- n
Random Access Memory (RAM)

- Internally, we need to be able to both read and write to bits of memory.

- Consider the following circuit that can function as a bit of memory:

- Note: circuit is not really made like this, but this will function correctly to explain the concept...
Random Access Memory (RAM)

- Take 1-bit memory and connect them into an array:
Random Access Memory (RAM)

- Can also share data lines with both input and output data using tri-state buffers (enabled by the read/write signal):

![Diagram of RAM with data inputs and outputs](image-url)

- 1 bit memory and select lines for input and output connections.
Textbook

- RAM is described in Chapter 7, Sections 7.2 and 7.3 of the course textbook.
Programmable Logic Array (PLA)

- Programmable device capable of implementing functions expressed in SOP.

- Consists of **input buffers and inverters** followed by:
  - Programmable **AND plane**, followed by
  - Programmable **OR plane**.
Programmable Logic Array (PLA)

- Can implement \( m \) logic functions of \( n \) variables. Limit is the number of product terms that can be generated inside of the device.

![Diagram of PLA with input buffers, AND plane, OR plane, and output functions.](image-url)
Programmable Logic Array (PLA)

- Example implementing 2 logic functions of 3 inputs using a 3-5-2 PLA.

\[ f_1 = x_1x_2 + x_1\overline{x}_3 + \overline{x}_1x_2x_3 \]

\[ f_2 = x_1x_2 + \overline{x}_1x_2x_3 + x_1x_3 \]
Programmable Array Logic (PAL)

- Similar to a PLA, but only has a programmable AND plane.
  - The OR plane is fixed.

- Not as flexible as a PLA since only certain AND gates feed each OR gate, but has fewer things that need programming.
Programmable Array Logic (PAL)

- Example of a PAL:

\[
\begin{align*}
    f_1 &= x_1x_2 + x_1\overline{x}_3 \\
    f_2 &= \overline{x}_2x_3 + x_1x_3
\end{align*}
\]
Programmable Array Logic (PAL)

- Sometimes the outputs are fed back internally and can be used to create product terms.
Programmable Logic Array (PLA)

- PLA and PAL are described in Chapter 7, Sections 7.6 and 7.7 of the course textbook.
Simple Programmable Logic Device (SPLD)

- To implement sequential circuits, take a PAL and add some flip-flops at the output of the OR plane.

- For example...

 Above circuit (plus SOP from the AND plane and OR gate) form a MacroCell.

- Several MacroCells together in the same IC is called an SPLD.
Complex Programmable Logic Device (CPLD)

- PLA, PAL and SPLD typically contain small number of outputs (e.g., 16 outputs) with many inputs (e.g., 36 inputs) and a fair number of product terms.

  Therefore only good for simple circuits where each equation has a wide fanin.

- Using a **Complex Programmable Logic Device (CPLD)** is the “next step” if we have a large complicated circuit...

- CPLD consists of many SPLD connected together by a **Programmable Routing Fabric** all in the **same IC**.
Complex Programmable Logic Device (CPLD)

- Typical architecture (each PAL-like block has many inputs - e.g., 36 - , many product terms - e.g., 80 - and several outputs - e.g., 16).
Complex Programmable Logic Device (CPLD)

- Can “zoom in” around one of the PAL blocks:

- In addition to programming the AND plane and MacroCells, also need to program the multiplexer select lines to “route” the correct signals into the PAL block.
Types of PLA, PAL, SPLD and CPLD

- Programming of these devices is similar to ROM; i.e., these devices are typically either PROM, EPROM or EEPROM.

- Programming info is generated (perhaps with a software tool), and the bit stream of program info is provided to one (or a few) additional pins on the device.

- Also possible (these days) to have SRAM-based PLDs...
  - In SRAM devices, the programming info is lost when power is turned off.
  - Necessary to re-program device every time the system is powered up.

- Often to see a configuration EPROM beside an SRAM based PLD on a circuit board.
  - Two chip solution... The EPROM holds the program that gets applied to the PLD upon power up.
Textbook

- SPLD and CPLD are described in Chapter 7, Section 7.8 of the textbook.
Introduction to FPGAs (1)

- Field Programmable Gate Arrays
- Back to basics: all programs are essentially a series of logic operations on bits

- The key idea is that FPGAs are custom-designed like ICs (ASICs), but are also software-reprogrammable
Introduce to FPGAs (2)

- You can in some sense think of an FPGA as a grid of wires connecting together logic gates. The joints between the wires are defined when you 'configure' the device.
- These wires have “fuses” between them - and the “fuses” can be “blown” or connected in software.
- At least, that was the original idea (Programmable Array Logic) - now they are far more sophisticated.
Field Programmable Gate Array (FPGA)

- Another type of programmable device capable of handling large circuits.

- Different from a CPLD:
  - Logic is not implemented in terms of Product Terms/MacroCells
  - Implemented using Lookup Table (LUT) which are like little memories
Instead of just AND/OR gates, FPGAs now use lookup table and flip-flop blocks, and include onboard memory (block RAM), hardware integer multipliers, fast I/O interconnects etc.
What is a reconfigurable computer?

- Idea whereby hardware can modify itself to suit executing program

- ‘Reconfigurable computing’ is sometimes used to refer to FPGAs alone.

- We use the term to refer to hybrid computers that include both conventional microprocessors and FPGA reconfigurable logic.
Field Programmable Gate Array (FPGA)

- Typical FPGA consists of many small logic blocks interconnected by programmable routing resources.
Field Programmable Gate Array (FPGA)

- Can “zoom in” around a logic block.

- Routing resources around the logic blocks need to be programmed so signals get “routed” to where they are needed.
Field Programmable Gate Array (FPGA)

- Can “zoom in” inside a logic block (e.g., 3-input logic block):

  ![Diagram of 3-input logic block](image)

- Can implement any 3-input function by properly programming the configuration bits.
Types of FPGA

- FPGA typically SRAM-based devices, but can be had in PROM, EPROM or EEPROM types.
Summary of CPLD and FPGA

☐ CPLD are:
  - Course-grained programmable architectures based on Product Term/MacroCell ideas (course-grained since each PAL-like block has many MacroCells, many inputs and many product terms).
  - Traditionally based on PROM, EPROM, EEPROM (but available SRAM-based).
  - Low to medium density - capable of implementing small to medium sized circuits.

☐ FPGA are:
  - Fine-grained programmable architectures based on Lookup Table ideas (fine-grained since each logic block implements a function of few inputs).
  - Traditionally based on SRAM (but available in PROM, EPROM and EEPROM).
  - Medium to high density - capable of implementing medium to large sized circuits.