Memory Architecture

Memory Cell Technologies

Programmable Logic Devices
Memory Architecture
Memory Interface

Data stored in *word* units

A word is several bytes (powers of two are typical) *write* operations store data to memory

*read* operations retrieve data from memory

General ROM:

$2^k \times n$ words

$n$ bits per word
Conceptual View of Memory

Memory is an array of cells.
Each cell stores a single bit.

One cell
Implementation of cell depends on type of memory.
Address is decoded into set of wordlines.  
Wordlines select row to be read/written.  
Only one wordline=1 at a time.
Multiple cells read in parallel, setting values of multiple bitlines.
Coincident Selection Saves Decode Logic

- 15-bit address
- 32,768 row selects
- 32,800 gates
- Top 9 bits of address
- 512 row selects
- 64 col selects
- Bottom 6 bits of address
- 608 gates
Memory Cell Technologies
Static Random-Access Memory Cell (SRAM)
Dynamic RAM Cell

![Diagram of a Dynamic RAM Cell with connections labeled Bit line and Word line.](image-url)
# CMOS Mask-Programmed ROMs

<table>
<thead>
<tr>
<th>Add. Data</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>011</td>
</tr>
<tr>
<td>01</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>11</td>
<td>010</td>
</tr>
</tbody>
</table>

ROM “programmed” by selectively connecting drain wires.

Active-high wordlines.
EPROMs and FLASH use Floating-Gate MOSFETs
## Volatile Storage Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Flip-Flop</th>
<th>SRAM</th>
<th>DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transistors/Bit</strong></td>
<td>Approx. 20</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td><strong>Density</strong></td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td><strong>Access Time</strong></td>
<td>Fast</td>
<td>Medium</td>
<td>Slow</td>
</tr>
<tr>
<td><strong>Destructive Read?</strong></td>
<td>No</td>
<td>No</td>
<td>Yes(^1)</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

\(^1\)Therefore refresh required
Atari Space Race, 1973
Atari Space Race PCB

Front

Back (mirrored)
The Space Race ROM
The decoder or “AND plane”

In a RAM or ROM, computes every minterm

Pattern is not programmable
The Space Race ROM

The contents or “OR plane”

One term for every output

Pattern is programmable = the contents of the ROM
The Space Race ROM

Can we do better?
Simplifying the Space Race ROM ($D_0$-$D_7$)

Essential minterms mean don’t expand these
The Space Race ROM – using PAL

\[ D_0 = 32\bar{10} \]
The Space Race ROM – using PAL

$D_0 = 32\overline{1}0$

$D_1 = 32\overline{1}$
The Space Race ROM – using PAL

\[
D_0 = 32\bar{1}0
\]
\[
D_1 = 32\bar{1}
\]
\[
D_2 = 3\bar{2}10 + 32\bar{1}0
\]

Saved two ANDs
The Space Race ROM – using PAL

\[
\begin{align*}
D_0 &= 32\overline{1}0 \\
D_1 &= 32\overline{1} \\
D_2 &= 3210 + 32\overline{1}0 \\
D_3 &= \overline{3210} + 31\overline{0} + 32\overline{1}0 + 3210 \\
\end{align*}
\]
The Space Race ROM – using PAL

\[ D_0 = 32\bar{1}0 \]
\[ D_1 = 32\bar{1} \]
\[ D_2 = 3210 + 32\bar{1}0 \]
\[ D_3 = \overline{3210} + 31\bar{0} + 32\bar{1}0 + 3210 \]
\[ D_4 = \overline{3210} + \overline{3210} + 32\bar{1}0 + 3210 \]
The Space Race ROM – using PAL

\[ D_0 = 32\overline{10} \]
\[ D_1 = 32\overline{1} \]
\[ D_2 = \overline{3210} + 32\overline{10} \]
\[ D_3 = \overline{3210} + 31\overline{0} + 32\overline{10} + 3210 \]
\[ D_4 = \overline{3210} + \overline{3210} + 32\overline{10} + 3210 \]
\[ D_5 = 31\overline{0} + 20 + 21 + \overline{3210} + \overline{3210} \]
The Space Race ROM – using PAL

\[ D_0 = \overline{3210} \]
\[ D_1 = \overline{321} \]
\[ D_2 = \overline{3210} + \overline{3210} \]
\[ D_3 = \overline{3210} + \overline{310} + \overline{3210} + 3210 \]
\[ D_4 = \overline{3210} + \overline{3210} + \overline{3210} + 3210 \]
\[ D_5 = \overline{310} + 20 + 21 + \overline{3210} + \overline{3210} \]
\[ D_6 = \overline{3210} + \overline{3210} \]
The Space Race ROM – using PAL

\[ D_0 = 32\bar{1}0 \]
\[ D_1 = 32\bar{1} \]
\[ D_2 = 3\bar{2}10 + 32\bar{1}0 \]
\[ D_3 = 3\bar{2}10 + 31\bar{0} + 3\bar{2}10 + 3210 \]
\[ D_4 = 3\bar{2}10 + 3\bar{2}10 + 3\bar{2}10 + 3210 \]
\[ D_5 = 31\bar{0} + 20 + 21 + 3\bar{2}10 + 3210 \]
\[ D_6 = 3\bar{2}10 + 32\bar{1}0 \]
\[ D_7 = 3\bar{2}10 + 32\bar{1}0 \]

Saved two ANDs
Field-Programmable Gate Arrays (FPGAs)

16×1 RAM

programmable switch

Switch Block

Switch Box: 6 programmable switches