Demultiplexers

Programmable Logic Devices
- Programmable logic array (PLA)
- Programmable array logic (PAL)

Switching networks logic blocks
- Multiplexer (MUX)
  - Routes one of many inputs to a single output
  - Also called a selector
- Demultiplexer (DEMUX)
  - Routes a single input to one of many outputs
  - Also called a decoder

Logic sharing example

Demultiplexers
- Basic concept
  - Single data input; n control inputs ("selects"); 2^n outputs
  - Single input connects to one of 2^n outputs
  - "Selects" decide which output is connected to the input

Demultiplexers
- The input is called an “enable” (G)
Demultiplexer as logic block

- A n:2^m demux can implement any function of n variables
  - Use variables as select inputs
  - Tie enable input to logic 1
  - Sum the appropriate minterms (extra OR gate)

Demultiplexer "decodes" appropriate minterms from the control signals

Cascading demultiplexers

- 5:32 demux

Programmable logic

- Concept: Large array of uncommitted AND/OR gates (actually NAND/NOR gates)
  - You program the array by making or breaking connections
  - Programmable block for sum-of-products logic

All two-level functions available

- You "program" the wire connections

Example

- F0 = A + B'C'
- F1 = AC + AB
- F2 = B'C' + AB
- F3 = B'C + A
Short-hand notation
- Draw multiple wires as a single wire or bus
- \( \times \) signifies a connection

Before Programming
\[
\begin{array}{ccc}
A & B & C \\
\hline
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

After Programming
\[
\begin{array}{ccc}
A & B' & C' \\
\hline
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

PLA example
- \( F_1 = ABC \)
- \( F_2 = A + B + C \)
- \( F_3 = A'B'C' \)
- \( F_4 = A' + B' + C' \)
- \( F_5 = A \text{xor} B \text{xor} C \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F_0</th>
<th>F_1</th>
<th>F_2</th>
<th>F_3</th>
<th>F_4</th>
<th>F_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Think of as a memory-address decoder

Memory bits

Example: BCD to Gray code

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>W</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Wiring a PLA

Minimized functions:
- \( W = A + BC + BD \)
- \( X = BC' \)
- \( Y = B + C \)
- \( Z = A'B'CD + BCD + AD' + B'CD' \)

Wiring a PAL

Minimized functions:
- \( W = A + BC + BD \)
- \( X = BC' \)
- \( Y = B + C \)
- \( Z = A'B'CD + BCD + AD' + B'CD' \)

Fine example for the use of PAL
(because no shared AND terms)

Many AND gates wasted, but
still faster and cheaper than PLA

PLAs versus PALs
- We've been looking at PLAs
  - Fully programmable AND/OR arrays
- Programmable array logic (PAL)
  - Programmable AND array
  - OR array is prewired
    - Cheaper and faster than PLAs

Example: BCD to Gray code

K-map for W

K-map for X

K-map for Y

K-map for Z
### Implementation comparison

**PLA:**
- No shared logic terms in this example
- 10 decoded functions (10 AND gates)

**PAL:**
- Z requires 4 product terms
- Need a PAL that handles 4 product terms for each output
- 16 decoded functions (16 AND gates)
- 6 unused AND gates

**Example is a good candidate for PALs:**
- 10 of 16 possible inputs are decoded
- No sharing among AND terms