Chapter # 4: Programmable and Steering Logic

Section 4.1
PALs and PLAs

Pre-fabricated building block of many AND/OR gates (or NOR, NAND) "Personalized" by making or breaking connections among the gates

Programmable Array Block Diagram for Sum of Products Form

Inputs

Dense array of AND gates

Product terms

Dense array of OR gates

Outputs
**PALs and PLAs**

- PLA -- Programmable Logic Array
- PAL -- Programmable Array Logic

- A typical TTL field-PLA might have 16 inputs, 48 product terms, 8 outputs -- 24 data pins
- Equivalent -- forty-eight 16-input AND gates and eight 48-input OR gates.
- 12 data-pin SSI package gives four 2-input gate
PALs and PLAs

Key to Success: Shared Product Terms

Equations

F0 = A' + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A

Example:

Personality Matrix

<table>
<thead>
<tr>
<th>Product term</th>
<th>Inputs A B C</th>
<th>Outputs F0 F1 F2 F3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B</td>
<td>1 1 -</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>B C</td>
<td>0 1 0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>A C</td>
<td>1 0 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>B C</td>
<td>0 0 0</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>A</td>
<td>1 - -</td>
<td>0 0 0 1</td>
</tr>
</tbody>
</table>

Input Side:
1 = asserted in term
0 = negated in term
- = does not participate

Output Side:
1 = term connected to output
0 = no connection to output

Need a 3-input, 5-product term, 4-output --- PLA device
PALs and PLAs

Example Continued

All possible connections are available before programming
• Programmer -- hardware device which personalizes the array
• Programming process depends on the particular IC
• Frequent technique -- places fuses between all possible inputs to a gate and the gate itself. Programmer hardware breaks the connections by placing a high current across selected fuses
PALs and PLAs

Example Continued

Unwanted connections are "blown"

F0 = A + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A

Note: some array structures work by making connections rather than breaking them
**Function Generator of 3 inputs: A, B, C**

The circuit should implement the logic functions AND, OR, NAND, NOR, XOR, XNOR

- **F1** = $A \land B \land C$
- **F2** = $A \lor B \lor C$
- **F3** = $A \land B \land C = A' + B' + C'$
- **F4** = $A \lor B \lor C = A'B'C'$
- **F5** = $A \oplus B \oplus C = A'B'C + A'BC' + AB'C' + ABC$
- **F6** = $A \oplus B \oplus C = ABC' + A'BC + AB'C + A'B'C'$
**PALs and PLAs**

*Alternative representation for high fan-in structures*

4-input, 4-output, 4 product terms

Short-hand notation so we don't have to draw all the wires!

Notation for implementing

\[ F_0 = A \cdot B + A' \cdot B' \]
\[ F_1 = C \cdot D' + C' \cdot D \]
What is difference between Programmable Array Logic (PAL) and Programmable Logic Array (PLA)?

PAL concept — implemented by Monolithic Memories. Programmable AND array but connections between product terms and specific OR gates are hardwire (constrained topology of the OR Array).

PLA concept — generalized topologies in AND and OR planes. Can be programmed in any way.

PLA can take advantage of shared product terms. PAL cannot. PLA is slower because of the relative resistance of programmable and hardwired connections.
**PALs and PLAs**

*Design Example: BCD to Gray Code Converter*

**Truth Table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<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>0</td>
<td>1</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>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</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>0</td>
<td>1</td>
<td>0</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>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</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>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**K-maps**

- **K-map for W**
- **K-map for X**
- **K-map for Y**
- **K-map for Z**

**Minimized Functions:**

- \( W = A + B \overline{D} + B \overline{C} \)
- \( X = B \overline{C}' \)
- \( Y = B + C \)
- \( Z = A'B'C'D + B \overline{C} \overline{D} + A \overline{D}' + B' \overline{C} \overline{D}' \)
No shared product terms, PAL implementation is best.

Programmed PAL:

4 inputs,
16 Programmable ANDS
4 4-input ORS
4 outputs

4 product terms per each OR gate
PALs and PLAs

Code Converter Discrete Gate Implementation

1: 7404 hex inverters
2,5: 7400 quad 2-input NAND
3: 7410 tri 3-input NAND
4: 7420 dual 4-input NAND

5 SSI Packages vs. 1 PLA/PAL Package!
PALs and PLAs

Another Example: Magnitude Comparator

What is more efficient, a PLA or PAL?
PALs and PLAs

Another Example: Magnitude Comparator (cont’d)

Since AC’ and A’C are used twice, (shared), a PLA-based implementation is better than a PAL.