Programmable logic arrays (PLA)

- Pre-fabricated building block of many AND/OR gates
  - actually NOR or NAND
  - "personalized" by making or breaking connections among the gates
  - programmable array block diagram for sum of products form

![Diagram of PLA](image-url)
Multiplexor Logic

- $F(a,b,c) = a'F(0,b,c) + aF(1,b,c)$
  - Let $G(b,c) = F(0,b,c)$
  - Let $H(b,c) = F(1,b,c)$

- $G(b,c) = bc$
- $H(b,c) = b + c$

- Example FullAdder Carry = $ab + ac + bc$
- $G = bc$  
  $H = b + c + bc = b + c$

- $F(a,b,c) = a'b'F(0,0,c) + a'bF(0,1,c) + ab'F(1,0,c) + abF(1,1,c)$
  - $F(a,b,c) = a'b'F0 + a'bF1 + ab'F2$ or $abF3$

CSE 370 - Fall 1999 - Introduction - 2
Programmable Logic Devices

- Shared product terms among outputs

**example:**

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

**input side:**

- 1 = uncomplemented in term
- 0 = complemented in term
- _ = does not participate

**output side:**

- 1 = term connected to output
- _ = no connection to output

<table>
<thead>
<tr>
<th>product term</th>
<th>inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A B C</td>
<td>F0 F1 F2 F3</td>
</tr>
<tr>
<td>AB</td>
<td>1 1 _</td>
<td>_ 1 1 _</td>
</tr>
<tr>
<td>B'C</td>
<td>_ 0 1</td>
<td>_ _ _ 1</td>
</tr>
<tr>
<td>AC'</td>
<td>1 _ 0</td>
<td>_ 1 _ _</td>
</tr>
<tr>
<td>B'C'</td>
<td>_ 0 0</td>
<td>1 _ 1 _</td>
</tr>
<tr>
<td>A</td>
<td>1 _ _</td>
<td>1 _ 0 1</td>
</tr>
</tbody>
</table>

This is a personality matrix with rows and columns

(Note common subexpression re-use)
Before programming

- All possible connections are available before "programming"
  - in reality, all AND and OR gates are NANDs

![Diagram with fuses]
After programming

- Unwanted connections are "blown"
  - fuse (normally connected, break unwanted ones)
  - anti-fuse (normally disconnected, make wanted connections)

F0 = A + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A
Alternate representation for high fan-in structures

- Short-hand notation so we don't have to draw all the wires
  - \( \times \) signifies a connection is present and perpendicular signal is an input to gate

\[
\begin{align*}
F_0 &= AB + A'B' \\
F_1 &= CD' + C'D
\end{align*}
\]
PLA as ROM

- Multiple functions of A, B, C
  - F1 = A B C
  - F2 = A + B + C
  - F3 = A' B' C'
  - F4 = A' + B' + C'
  - F5 = A xor B xor C

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</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>0</td>
<td>1</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>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<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>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Common Sub-Expression Extraction and Use

Implementement
F = Σm(5, 7, 10, 14, 15)
G = Σm(6, 7, 9, 13, 15)
PALs and PLAs: another design example

- Magnitude comparator

K-map for EQ

K-map for NE

K-map for LT

K-map for GT

CSE 370 - Fall 1999 - Introduction - 9
PALs and PLAs

- Programmable logic array (PLA)
  - what we've seen so far
  - unconstrained fully-general AND and OR arrays

- Programmable array logic (PAL)
  - Fixed OR array
  - faster and smaller OR plane
  - No term sharing

If not using a row, then make zero

A given column of the OR array has access to only a subset of the possible product terms

CSE 370 - Fall 1999 - Introduction - 10
Read-only memories

- Two dimensional array of 1s and 0s
  - entry (row) is called a "word"
  - width of row = word-size
  - index is called an "address"
  - address is input
  - selected word is output

Like complete, preprogrammed (N)AND-plane of PLA

internal organization

word lines (only one is active – decoder is just right for this)

word[$i$] = 0011

word[$j$] = 1010

bit lines (normally pulled to 1 through resistor – selectively connected to 0 by word line controlled switches)
ROMs and combinational logic

- Combinational logic implementation (two-level canonical form) using a ROM
- Put entire truth table into memory

\[
\begin{align*}
F_0 &= A' B' C' + A B' C' + A B' C \\
F_1 &= A' B' C + A' B C' + A B C \\
F_2 &= A' B' C' + A' B' C + A B' C' \\
F_3 &= A' B C + A B' C' + A B C'
\end{align*}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F0</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</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>0</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</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>1</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>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

truth table

ROM 8 words x 4 bits/word

block diagram
ROM structure

- Similar to a PLA structure but with a fully decoded AND array
  - completely flexible OR array (unlike PAL)
ROM vs. PLA

- ROM approach advantageous when
  - design time is short (no need to minimize output functions)
  - most input combinations are needed (e.g., code converters)
  - little sharing of product terms among output functions

- ROM problems
  - size doubles for each additional input (32x4 for Calendar example)
  - can't exploit don't cares

- PLA approach advantageous when
  - design tools are available for multi-output minimization
  - there are relatively few unique minterm combinations
  - many minterms are shared among the output functions
  - Supports multilevel implementation using feedback

- PAL problems
  - constrained fan-ins on OR plane
  - Difficulty of common term re-use??
Regular logic structures for two-level logic

- ROM – full AND plane, general OR plane
  - cheap (high-volume component)
  - can implement any function of n inputs
  - medium speed

- PAL – programmable AND plane, fixed OR plane
  - intermediate cost
  - can implement functions limited by number of terms
  - high speed (only one programmable plane that is much smaller than ROM's decoder)

- PLA – programmable AND and OR planes
  - most expensive (most complex in design, need more sophisticated tools)
  - can implement any function up to a product term limit
  - slow (two programmable planes)
Regular logic structures for multi-level logic

- Difficult to devise a regular structure for arbitrary connections between a large set of different types of gates
  - efficiency/speed concerns for such a structure
  - in 467 you'll learn about field programmable gate arrays (FPGAs) that are just such programmable multi-level structures
    - programmable multiplexers for wiring
    - **lookup tables for logic functions** *(programming fills in the table)*
      - multi-purpose cells (utilization is the big issue)

- Use multiple levels of PALs/PLAs/ROMs
  - output intermediate result
  - make it an input to be used in further logic
Combinational logic implementation summary

- Multi-level logic
  - conversion to NAND-NAND and NOR-NOR networks
  - transition from simple gates to more complex gate building blocks
  - reduced gate count, fan-ins, potentially faster
  - more levels, harder to design

- Time response in combinational networks
  - gate delays and timing waveforms
  - hazards/glitches (what they are and why they happen)

- Regular logic
  - multiplexers/decoders
  - ROMs
  - PLAs/PALs
  - advantages/disadvantages of each