Lecture 10

- PLDs
  - ROMs
- Multilevel logic

Read-only memories (ROMs)

- Two dimensional array of stored 1s and 0s
  - Input is an address \( \Rightarrow \) ROM decodes all possible input addresses
  - Stored row entry is called a "word"
  - ROM output is the decoded word

ROM details

- Similar to a PLA but with a fully decoded and fixed AND array
- Completely flexible OR array (unlike a PAL)
Two-level logic using a ROM

- Use a ROM to directly store a truth table
  - No need to minimize logic

![ROM 8 words x 4 bits/word](image)

You specify whether to store 1 or 0 in each location in the ROM

ROMs versus PLAs/PALs

ROMs:
- Benefits
  - Quick to design, simple
- Limitations
  - Size doubles for each additional input
  - Can’t exploit don’t cares

PLAs/PALs:
- Benefits
  - Logic minimization reduces size
- Limitations
  - PAL OR-plane has hard-wired fan-in

Example: BCD to 7-segments

- The problem
  - Input is a 4-bit BCD digit (A, B, C, D)
  - Need signals to drive a display (7 outputs C0 – C6)

![BCD to 7-segment control-signal decoder](image)

Formalize the problem

- Truth table
  - Many don’t cares
- Choose implementation target
  - If ROM, we are done
  - Don’t cares imply PAL/PLA may be good choice
- Implement design
  - Minimize the logic
  - Map into PAL/PLA

![Not all rows of the truth table are listed separately](image)

SOP implementation

- Can do better than 15 product terms
  - Share terms among outputs ⇒ only 9 unique product terms
  - Each output not necessarily minimized

- For example:

![Better SOP for PLA](image)

- 15 unique product terms if we minimize individually

- 4 input, 7 output
- PLA: 15 AND gates
- PAL: 4 product terms per output (28 AND gates)
Better SOP for PLA

- 15 unique product terms if minimized individually
- 9 unique product terms if we try to share terms among outputs

\[
\begin{align*}
C_0 &= A + BD + C + B'D' \\
C_1 &= C'D' + CD + B' \\
C_2 &= B + C' + D \\
C_3 &= B'D' + CD' + BC'D + B'C \\
C_4 &= B'D' + CD' \\
C_5 &= A + C'D' + BD' + BC' \\
C_6 &= A + CD' + BC' + B'C \\
C_7 &= A + BC'D + CD + B'D' + BCD' + A \\
C_8 &= B'D + C'D' + CD + B'D' \\
C_9 &= B'D + BC'D + C'D' + CD + BCD' \\
C_{10} &= BC'D + B'D + B'D' + BCD' \\
C_{11} &= B'D' + BCD' \\
C_{12} &= BC'D + C'D' + A + BCD' \\
C_{13} &= B'C + BC' + BCD' + A \\
C_{14} &= B + C' + D \\
C_{15} &= B'D' + C'D' + A + BCD' \\
C_{16} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{17} &= BC'D + B'D + B'D' + BCD' \\
C_{18} &= B'D' + BCD' \\
C_{19} &= BC'D + C'D' + A + BCD' \\
C_{20} &= B'C + BC' + BCD' + A \\
C_{21} &= B + C' + D \\
C_{22} &= B'D' + C'D' + A + BCD' \\
C_{23} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{24} &= BC'D + B'D + B'D' + BCD' \\
C_{25} &= B'D' + BCD' \\
C_{26} &= BC'D + C'D' + A + BCD' \\
C_{27} &= B'C + BC' + BCD' + A \\
C_{28} &= B + C' + D \\
C_{29} &= B'D' + C'D' + A + BCD' \\
C_{30} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{31} &= BC'D + B'D + B'D' + BCD' \\
C_{32} &= B'D' + BCD' \\
C_{33} &= BC'D + C'D' + A + BCD' \\
C_{34} &= B'C + BC' + BCD' + A \\
C_{35} &= B + C' + D \\
C_{36} &= B'D' + C'D' + A + BCD' \\
C_{37} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{38} &= BC'D + B'D + B'D' + BCD' \\
C_{39} &= B'D' + BCD' \\
C_{40} &= BC'D + C'D' + A + BCD' \\
C_{41} &= B'C + BC' + BCD' + A \\
C_{42} &= B + C' + D \\
C_{43} &= B'D' + C'D' + A + BCD' \\
C_{44} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{45} &= BC'D + B'D + B'D' + BCD' \\
C_{46} &= B'D' + BCD' \\
C_{47} &= BC'D + C'D' + A + BCD' \\
C_{48} &= B'C + BC' + BCD' + A \\
C_{49} &= B + C' + D \\
C_{50} &= B'D' + C'D' + A + BCD' \\
C_{51} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{52} &= BC'D + B'D + B'D' + BCD' \\
C_{53} &= B'D' + BCD' \\
C_{54} &= BC'D + C'D' + A + BCD' \\
C_{55} &= B'C + BC' + BCD' + A \\
C_{56} &= B + C' + D \\
C_{57} &= B'D' + C'D' + A + BCD' \\
C_{58} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{59} &= BC'D + B'D + B'D' + BCD' \\
C_{60} &= B'D' + BCD' \\
C_{61} &= BC'D + C'D' + A + BCD' \\
C_{62} &= B'C + BC' + BCD' + A \\
C_{63} &= B + C' + D \\
C_{64} &= B'D' + C'D' + A + BCD' \\
C_{65} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{66} &= BC'D + B'D + B'D' + BCD' \\
C_{67} &= B'D' + BCD' \\
C_{68} &= BC'D + C'D' + A + BCD' \\
C_{69} &= B'C + BC' + BCD' + A \\
C_{70} &= B + C' + D \\
C_{71} &= B'D' + C'D' + A + BCD' \\
C_{72} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{73} &= BC'D + B'D + B'D' + BCD' \\
C_{74} &= B'D' + BCD' \\
C_{75} &= BC'D + C'D' + A + BCD' \\
C_{76} &= B'C + BC' + BCD' + A \\
C_{77} &= B + C' + D \\
C_{78} &= B'D' + C'D' + A + BCD' \\
C_{79} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{80} &= BC'D + B'D + B'D' + BCD' \\
C_{81} &= B'D' + BCD' \\
C_{82} &= BC'D + C'D' + A + BCD' \\
C_{83} &= B'C + BC' + BCD' + A \\
C_{84} &= B + C' + D \\
C_{85} &= B'D' + C'D' + A + BCD' \\
C_{86} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{87} &= BC'D + B'D + B'D' + BCD' \\
C_{88} &= B'D' + BCD' \\
C_{89} &= BC'D + C'D' + A + BCD' \\
C_{90} &= B'C + BC' + BCD' + A \\
C_{91} &= B + C' + D \\
C_{92} &= B'D' + C'D' + A + BCD' \\
C_{93} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{94} &= BC'D + B'D + B'D' + BCD' \\
C_{95} &= B'D' + BCD' \\
C_{96} &= BC'D + C'D' + A + BCD' \\
C_{97} &= B'C + BC' + BCD' + A \\
C_{98} &= B + C' + D \\
C_{99} &= B'D' + C'D' + A + BCD' \\
C_{100} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{101} &= BC'D + B'D + B'D' + BCD' \\
C_{102} &= B'D' + BCD' \\
C_{103} &= BC'D + C'D' + A + BCD' \\
C_{104} &= B'C + BC' + BCD' + A \\
C_{105} &= B + C' + D \\
C_{106} &= B'D' + C'D' + A + BCD' \\
C_{107} &= B'D + BC'D + C'D' + CD + BCD' \\
C_{108} &= BC'D + B'D + B'D' + BCD' \\
C_{109} &= B'D' + BCD' \\
C_{110} &= BC'D + C'D' + A + BCD' \\
C_{111} &= B'C + BC' + BCD' + A \\
C_{112} &= B + C' + D \\
C_{113} &= B'D' + C'D' + A + BCD' \\
C_{114} &= B'D + BC'D + C'D' + CD + BCD' \\
\end{align*}
\]

Multilevel logic

- Basic idea: Simplify logic using more than 2 gate levels
  - Time–space (speed versus gate count) tradeoff
  - Will talk about the speed issue with timing diagram
- Two-level logic usually
  - Has smaller delays (faster circuits)
  - More gates and more wires (more circuit area)
- Multilevel logic usually
  - Has fewer gates (smaller circuits)
  - More gate delays (slower circuits)

Example

- SOP: \( X = ADF + AEF + BDF + BEF + CDF + CEF + G \)
  - \( X \) is minimized!
  - Six 3-input ANDs; one 7-input OR; 26 wires
- Multilevel: \( X = (A+B+C)(D+E)F + G \)
  - Factored form
  - One 3-input OR, two 2-input OR’s, one 3-input AND; 11 wires

Multilevel NAND/NAND

\[
F = A(B+CD) + BC' 
\]

- Original AND-OR network
- Introduce bubbles (conserves inversions)

Multilevel NOR/NOR

\[
F = A(B+CD) + BC' 
\]

- Original AND-OR network
- Introduce bubbles (conserves inversions)
### Generic multilevel conversion

\[ F = ABC + BC + D = AX + X + D \]

(a) [Original circuit]

(b) [Add double bubbles at inputs]

(c) [Distribute bubbles, some mismatches]

(d) [Insert inverters to fix mismatches]

### Issues with multilevel design

- No global definition of “optimal” multilevel circuit
  - Optimality depends on user-defined goals
- Synthesis requires CAD-tool help
  - No simple hand methods like K-maps
  - CAD tools manipulate Boolean expressions
  - Covered in more detail in CSE467

### Multilevel logic summary

- Advantages over 2-level logic
  - Smaller circuits
  - Reduced fan-in
  - Less wires

- Disadvantages wrt 2-level logic
  - More difficult design
  - Less powerful optimizing tools