ROM AND PLA
Compiled by: Suresh S. Balpande
contact:sbalpande@yahoo.com
BY:SURESH BALPANDE
Read-Only Memories

- Read-Only Memories are nonvolatile
  - Retain their contents when power is removed
- Mask-programmed ROMs use one transistor per bit
  - Presence or absence determines 1 or 0

Compiled by: Suresh S. Balpande
contact:sbalpande@yahoo.com
ROM Example

- 4-word x 6-bit ROM
  - Represented with dot diagram
  - Dots indicate 1’s in ROM

Word 0: 010101
Word 1: 011001
Word 2: 100101
Word 3: 101010

Looks like 6 4-input pseudo-nMOS NORs
ROM Array Layout

- Unit cell is 12 x 8 (about 1/10 size of SRAM)
Row Decoders

- ROM row decoders must pitch-match with ROM
  - Only a single track per word!
PROMs and EPROMs

- Programmable ROMs
  - Build array with transistors at every site
  - Burn out fuses to disable unwanted transistors

- Electrically Programmable ROMs
  - Use floating gate to turn off unwanted transistors
  - EPROM, EEPROM, Flash
Building Logic with ROMs

- Use ROM as lookup table containing truth table
  - n inputs, k outputs requires __ words x __ bits
  - Changing function is easy – reprogram ROM

- Finite State Machine
  - n inputs, k outputs, s bits of state
  - Build with __________ bit ROM and ____ bit reg
Building Logic with ROMs

- Use ROM as lookup table containing truth table
  - $n$ inputs, $k$ outputs requires $2^n$ words x $k$ bits
  - Changing function is easy – reprogram ROM
- Finite State Machine
  - $n$ inputs, $k$ outputs, $s$ bits of state
  - Build with $2^{n+s} \times (k+s)$ bit ROM and $(k+s)$ bit reg
ROM Implementation

- 16-word x 5 bit ROM

![Diagram of a 4:16 decoder and a ROM with inputs S1, S0, L, R, and outputs TL, TR, F]

4:16 DEC

<table>
<thead>
<tr>
<th>0000</th>
<th>0001</th>
<th>0010</th>
<th>0011</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
</tr>
<tr>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>

Compiled by: Suresh S. Balpande
contact: sbalpande@yahoo.com
ROM Implementation

- 16-word x 5 bit ROM

<table>
<thead>
<tr>
<th>S_1</th>
<th>S_0</th>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
</tr>
<tr>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
</tr>
<tr>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
</tr>
</tbody>
</table>

S_1 S_0 TR TL F'

Compiled by: Suresh S. Balpande
contact: sbalpande@yahoo.com
PLAs

- A Programmable Logic Array performs any function in sum-of-products form.
- Literals: inputs & complements
- Products / Minterms: AND of literals
- Outputs: OR of Minterms

Example: Full Adder

\[ s = \overline{a} \overline{b} \overline{c} + \overline{a}bc + \overline{ab}c + abc \]

\[ c_{out} = ab + bc + ac \]
NOR-NOR PLAs

- ANDs and ORs are not very efficient in CMOS
- Dynamic or Pseudo-nMOS NORs are very efficient
- Use DeMorgan’s Law to convert to all NORs
PLA Schematic & Layout

AND Plane

OR Plane

bc
ac
ab
abc
abc
abc
abc

out

c

Compiled by: Suresh S. Balpande
contact: sbalpande@yahoo.com
PLAs vs. ROMs

- The OR plane of the PLA is like the ROM array
- The AND plane of the PLA is like the ROM decoder
- PLAs are more flexible than ROMs
  - No need to have $2^n$ rows for $n$ inputs
  - Only generate the minterms that are needed
  - Take advantage of logic simplification
RoboAnt Dot Diagram

\[ S_1' = S_1S_0' + \overline{L}S_1 + \overline{L}RS_0 \]
\[ S_0' = R + L\overline{S}_1 + LS_0 \]
\[ TR = S_1\overline{S}_0 \]
\[ TL = S_0 \]
\[ F = S_1 + S_0 \]

Compiled by: Suresh S. Balpande
contact: sbalpande@yahoo.com