Implementation Technologies

- Standard gates (pretty much done)
  - gate packages
  - cell libraries
- Regular logic (we are here)
  - multiplexers
  - decoders
- Two-level programmable logic (a little later)
  - PALs, PLAs, PLDs
  - ROMs
  - FPGAs

Regular logic

- Need to make design faster
- Need to make engineering changes easier to make
- Simpler for designers to understand and map to functionality
  - harder to think in terms of specific gates
  - easier to think in terms of larger multi-purpose blocks
Making connections

- Direct point-to-point connections using wires
- Route one of many inputs to a single output --- multiplexer
- Route a single input to one of many outputs --- demultiplexer

Mux and demux (cont'd)

- Uses of multiplexers/demultiplexers in multi-point connections

![Diagram of multiplexers and demultiplexers](image-url)
Multiplexers/selectors

- **Multiplexers/selectors: general concept**
  - $2^n$ data inputs, $n$ control inputs (called "selects"), 1 output
  - used to connect $2^n$ points to a single point
  - control signal pattern forms binary index of input connected to output

\[
Z = A' I_0 + A I_1
\]

<table>
<thead>
<tr>
<th>$I_1$</th>
<th>$I_0$</th>
<th>$A$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **2:1 mux**: $Z = A' I_0 + A I_1$
- **4:1 mux**: $Z = A'B' I_0 + A'B I_1 + AB' I_2 + ABI_3$
- **8:1 mux**: $Z = A'B'C' I_0 + A'B'C I_1 + A'BC I_2 + A'BCI_3 + AB'C I_4 + AB'C'I_5 + ABC'I_6 + ABCI_7$

- **In general**: $Z = \Sigma_{k=0}^{2^n-1}(m_k | k)$
  - in minterm shorthand form for a $2^n:1$ Mux

---

Multiplexers/selectors (cont'd)

- 2:1 mux: $Z = A' I_0 + A I_1$
- 4:1 mux: $Z = A'B' I_0 + A'B I_1 + AB' I_2 + ABI_3$
- 8:1 mux: $Z = A'B'C' I_0 + A'B'C I_1 + A'BC I_2 + A'BCI_3 + AB'C I_4 + AB'C'I_5 + ABC'I_6 + ABCI_7$

- **In general**: $Z = \Sigma_{k=0}^{2^n-1}(m_k | k)$
  - in minterm shorthand form for a $2^n:1$ Mux

---
Gate level implementation of muxes

- 2:1 mux

- 4:1 mux

Multiplexers as general-purpose logic

- A $2^n$:1 multiplexer can implement any function of $n$ variables
  - with the variables used as control inputs and
  - the data inputs tied to 0 or 1
  - in essence, a lookup table (LUT)

- Example:
  - $F(A,B,C) = m_0 + m_2 + m_6 + m_7$
  - $= A'B'C' + A'BC' + ABC' + ABC$

$$Z = A'B'C'I_0 + A'B'CI_4 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$
Cascading multiplexers

- Large multiplexers can be made by cascading smaller ones

<table>
<thead>
<tr>
<th>I0</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>I6</th>
<th>I7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

control signals B and C simultaneously choose one of I0, I1, I2, I3 and one of I4, I5, I6, I7

control signal A chooses which of the upper or lower mux's output to gate to Z

Multiplexers as general-purpose logic (cont’d)

- A $2^{n-1}:1$ multiplexer can implement any function of n variables
  - with n-1 variables used as control inputs and
  - the data inputs tied to the last variable or its complement

- Example:
  - $F(A,B,C) = m_0 + m_2 + m_6 + m_7$
    - $= A'B'C' + A'BC' + ABC' + ABC$
    - $= A'B'(C') + A'B(C') + AB'(0) + AB(1)$
Multiplexers as general-purpose logic (cont’d)

- **Generalization**
  - n-1 mux control variables
  - single mux data variable

- **Example:**
  - G(A, B, C, D) can be realized by an 8:1 MUX
  - Choose A, B, C as control variables

Activity

- Realize F = B’CD’ + ABC’ with a 4:1 multiplexer and a minimum of other gates:
Activity

- Realize $F = B'CD' + ABC'$ with a 4:1 multiplexer and a minimum of other gates:

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

0 when $B'C'$

$D'$ when $B'C$

A when $BC'$

0 when $BC$

$x = B'C'(0) + B'C(D') + BC'(A) + BC(0)$

Demultiplexers/decoders

- **Decoders/demultiplexers: general concept**
  - Single data input, $n$ control inputs, $2^n$ outputs
  - Control inputs (called "selects" (S)) represent binary index of output to which the input is connected
  - Data input usually called “enable” (G)

<table>
<thead>
<tr>
<th>1:2 Decoder:</th>
<th>3:8 Decoder:</th>
</tr>
</thead>
<tbody>
<tr>
<td>O0 = G • S'</td>
<td>O0 = G • S2' • S1' • S0'</td>
</tr>
<tr>
<td>O1 = G • S</td>
<td>O1 = G • S2' • S1' • S0</td>
</tr>
<tr>
<td>2:4 Decoder:</td>
<td>3:8 Decoder:</td>
</tr>
<tr>
<td>O0 = G • S1' • S0'</td>
<td>O2 = G • S2' • S1' • S0</td>
</tr>
<tr>
<td>O1 = G • S1' • S0</td>
<td>O3 = G • S2' • S1 • S0</td>
</tr>
<tr>
<td>O2 = G • S1 • S0'</td>
<td>O4 = G • S2 • S1' • S0'</td>
</tr>
<tr>
<td>O3 = G • S1 • S0</td>
<td>O5 = G • S2 • S1' • S0</td>
</tr>
<tr>
<td></td>
<td>O6 = G • S2 • S1 • S0'</td>
</tr>
<tr>
<td></td>
<td>O7 = G • S2 • S1 • S0</td>
</tr>
</tbody>
</table>
Gate level implementation of demultiplexers

- **1:2 decoders**
  - Active-high enable
  - Active-low enable

- **2:4 decoders**
  - Active-high enable
  - Active-low enable

Demultiplexers as general-purpose logic

- A \( n:2^n \) decoder can implement any function of \( n \) variables
  - with the variables used as control inputs
  - the enable inputs tied to 1 and
  - the appropriate minterms summed to form the function
Demultiplexers as general-purpose logic (cont’d)

- \( F_1 = A'B'C'D + A'B'CD + ABCD \)
- \( F_2 = ABC'D' + ABC \)
- \( F_3 = (A' + B' + C' + D') \)

Cascading decoders

- 5:32 decoder
  - 1x2:4 decoder
  - 4x3:8 decoders