EECE 218
Microcontrollers

Address decoder design
Connecting devices to the external bus

- **Processor:**
  - Data bus (D0 – D15) I/O
  - Address bus (A0 – A15) O
  - Control bus (E, R/~W, etc.) O

- **Device**
  - Data bus (D0 – D8: Typical) (I/O)
  - Address bus (A0 – Ax) I
  - Control bus (typical)
    - ~CS: Chip Select: ‘enables’ the chip
    - ~OE: Output enable: ‘chip can send data’ (~RE: Read enable)
    - ~WE: Write enable: ‘chip should latch data internally’
Connecting devices to the external bus

- Connections:
  - Processor Data ↔ Device data (2 8-bit chips)
  - Processor R/~W → Device R/~W (or equivalent)
    - On this processor:
      - E clock high means DATA ACCESS
      - Thus: ~RE = ~(E AND R/~W)
      - ~WE = ~(E AND ~(R/~W))
  - Processor Address lower-order bits (A0 – Ax)
    → Device address bits (A0 – Ax)
  - Processor Address higher-order bits
    → **Address decoder** → Device ~CS/~CE signals
Example: 8Kbyte RAM

Should be located at a full multiple of segment size, i.e. at $0000, $2000, $4000, $6000, .... $C000, $E000.

Choose: $C000
Example: 8Kbyte RAM

Must go LOW whenever the processor access the 8kb segment starting at address $C000.

\[ \sim\text{OE} = \sim(E \times (\sim R / \sim W)) \]
\[ \sim\text{WE} = \sim(E \times \sim(R / \sim W)) \]
Example: 8Kbyte RAM

- Address map: a table showing the address bit combinations for each address range.

- Content:
  
  $0/1$ : if L/H range bits agree, $X$: otherwise

- Example: Range: $\$C000 -- \$DFFF

  
  \[
  \begin{array}{cccccc}
  \text{A15} & \text{A14} & \text{A13} & \text{A12} & \text{A11} & \text{A10} \\
  \text{$\$C000$} & 1 & 1 & 0 & 0 & 0 & \ldots \\
  \text{$\$DFFF$} & 1 & 1 & 0 & 1 & 1 & \ldots \\
  \text{i.e. x}: & 1 & 1 & 0 & X & X & \ldots \\
  \end{array}
  \]

  i.e.: the address decoder: $(A15 * A14 * \sim A13)$
Adress decoders

- Rule: Lower bits of the A-bus go directly to the memory device, higher bits are used to generate the Chip Select.

- Full address decoding:
  » All lines are used: the address decoder uses all higher-order address bits.
    - Example: 32 byte device
      » Needs 5 bits to select within the 32 byte block
      » Address decoder has 11 bits as input
    → Not economic!
Address decoders

- Partial address (‘Block’) decoding:
  » Some higher-order lines are not used in the decoding process.

- Example: Microcontroller evaluation board
  - Memory is divided into 8, 8 kbyte segments
  - One 8 k segment is divided into 8, 1kbyte segments (I/O)

- Chip to help: ‘138: 3-to-8-line decoder

---

**EECE 218 – Microcontrollers**

---

* G2 = G2A + G2B
H = High Level, L = Low Level, X = Don’t Care
Address decoders

- Solution for the example:

  » Use one 138 to implement the 8-way split
    
    A15 → C [4]
    A14 → B [2]
    A13 → A [1]
    
    000 – selects: ~Y0 → means: $0000 (note 3 MSB bits!)
    111 – selects: ~Y7 → means: $E000

  » Use a second 138 to implement the 2\textsuperscript{nd} 8-way split

    E.g. I/O is at $4000-5FFF
    
    [#1 138] → ~Y2 output → ~G2 input of [#2 138]
    
    On #2 138: A12 → C / A11 → B / A10 → A
Address decoders

- Useful constants:
  
  \[ 1\text{K} = 1024 = 10 \text{ bits}, \ 1\text{K} = \$400, \ \$1000 = 4\text{K} \]

In the example design: connect a 32-byte I/O device to \(~\text{Y}0\) of \#2 138

Device is mapped to \$4000 --- \$43FF

**BUT:** A9 --- A5 bits are NOT used in the decoding!

Effect: the setting of these address bits do not change whether the device is selected or not. The device is selected regardless of these bits → the device registers appear at multiple addresses.

Example: \$4000 \rightarrow \$4020 \rightarrow \$4040 \rightarrow \ldots \rightarrow \$43E0
Systematic design of address decoders

- **Process:**
  - Given: memory layout
  - Result: address decoding circuit

- **Key:** address map

- **Example:**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Starts at</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM1</td>
<td>$E000</td>
<td>8K</td>
</tr>
<tr>
<td>ROM2</td>
<td>$C000</td>
<td>8K</td>
</tr>
<tr>
<td>RAM1</td>
<td>$A000</td>
<td>4K</td>
</tr>
<tr>
<td>RAM2</td>
<td>$8000</td>
<td>4K</td>
</tr>
<tr>
<td>IO1</td>
<td>$2000</td>
<td>256</td>
</tr>
<tr>
<td>IO2</td>
<td>$2100</td>
<td>256</td>
</tr>
<tr>
<td>IO3</td>
<td>$2200</td>
<td>256</td>
</tr>
<tr>
<td>IO4</td>
<td>$2300</td>
<td>256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>LS138</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>YO</td>
</tr>
<tr>
<td>Select</td>
<td>Y1</td>
</tr>
<tr>
<td></td>
<td>Y2</td>
</tr>
<tr>
<td></td>
<td>Y3</td>
</tr>
<tr>
<td></td>
<td>Y4</td>
</tr>
<tr>
<td></td>
<td>Y5</td>
</tr>
<tr>
<td></td>
<td>Y6</td>
</tr>
<tr>
<td></td>
<td>Y7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G1</th>
<th>G2</th>
<th>C</th>
<th>B</th>
<th>A</th>
<th>YO</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>Y5</th>
<th>Y6</th>
<th>Y7</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>H</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

* G2 = G2A + G2B

H = High Level, L = Low Level, X = Don’t Care
Systematic design of address decoders

- Address map

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM1</td>
<td>$E000</td>
<td>$FFFF</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ROM2</td>
<td>$C000</td>
<td>$DFFF</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RAM1</td>
<td>$A000</td>
<td>$AFFF</td>
<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>RAM2</td>
<td>$8000</td>
<td>$8FFF</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IO1</td>
<td>$2000</td>
<td>$20FF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IO2</td>
<td>$2100</td>
<td>$21FF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>IO3</td>
<td>$2200</td>
<td>$22FF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>IO4</td>
<td>$2300</td>
<td>$23FF</td>
<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>
</tbody>
</table>
Systematic design of address decoders

- Circuit: