Lecture 16: Address decoding

- Introduction to address decoding
- Full address decoding
- Partial address decoding
- Implementing address decoders
- Examples
Introduction to address decoding

- Although the memory space in the 68000 is said to be flat, it does not mean that the physical implementation of memory is homogeneous
  - Different portions of memory are used for different purposes: RAM, ROM, I/O devices
  - Even if all the memory was of one type, we still have to implement it using multiple ICs
  - This means that for a given valid address, one and only one memory-mapped component must be accessed

- Address decoding is the process of generating chip select (CS*) signals from the address bus for each device in the system

- The address bus lines are split into two sections
  - the N most significant bits are used to generate the CS* signals for the different devices
  - the M least significant signals are passed to the devices as addresses to the different memory cells or internal registers
A very simple example

- Let’s assume a very simple microprocessor with 10 address lines (1KB memory)
- Let’s assume we wish to implement all its memory space and we use 128x8 memory chips
- **SOLUTION**
  - We will need 8 memory chips (8x128=1024)
  - We will need 3 address lines to select each one of the 8 chips
  - Each chip will need 7 address lines to address its internal memory cells
Address decoding methods

- The previous example specified that all addressable memory space was to be implemented but
  - There are some situations where this requirement is not necessary or affordable

- If only a portion of the addressable space is going to be implemented there are two basic address decoding strategies
  - Full address decoding
    - All the address lines are used to specify a memory location
    - Each physical memory location is identified by a unique address
  - Partial address decoding
    - Since not all the address space is implemented, only a subset of the address lines are needed to point to the physical memory locations
    - Each physical memory location is identified by several possible addresses (using all combinations of the address lines that were not used)
Full address decoding

- Let’s assume the same microprocessor with 10 address lines (1KB memory)
  - However, this time we wish to implement only 512 bytes of memory
  - We still must use 128-byte memory chips
  - Physical memory must be placed on the upper half of the memory map

**SOLUTION**

<table>
<thead>
<tr>
<th>Device</th>
<th>A9</th>
<th>A8</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MEM 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MEM 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MEM 3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- Used for Address Decoding
- Used to reference memory cells on each memory IC

Memory map:

- MEM 0
- MEM 1
- MEM 2
- MEM 3
- Not used
- Not used
- Not used
Partial address decoding

- Let’s assume the same microprocessor with 10 address lines (1KB memory)
  - However, this time we wish to implement only 512 bytes of memory
  - We still must use 128-byte memory chips
  - Physical memory must be placed on the upper half of the memory map

- SOLUTION

![Memory map diagram]

<table>
<thead>
<tr>
<th>Device</th>
<th>A9</th>
<th>A8</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM 0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MEM 1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MEM 2</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MEM 3</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Implementing address decoders

- **Discrete logic**
  - High speed (propagation signals)
  - High chip-count
  - Lacks flexibility

- **Data decoders**
  - More appropriate than random logic
  - The selection of devices is determined by the physical wiring
  - All the memory blocks must have the same size

- **Programmable Read Only Memory (PROM)**
  - Versatile, since the selection of devices is determined by the programming
  - Memory blocks can be of different sizes
  - Lookup tables can become very large for more than 8 address lines

- **Other methods (beyond the scope of the lecture)** are
  - Field Programmable Gate Arrays (FPGA) and
  - Programmable Address Decoders
Example 1

- A circuit containing 64K words of RAM is to be interfaced to a 68000-based system, so that the first address of RAM (the base address) is at $480000
  - What is the entire range of RAM addresses?
  - Design a FULL address decoder using two 64K×8 RAM ICs

Solution

- The address range for the RAM is from $480000 to $480000+(128K=$20000)=$4A0000-1=$49FFFF
- The two ICs must be differentiated through UDS*/LDS* (since the 68000 DOES NOT have A0)

<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>
<th></th>
<th></th>
<th></th>
<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>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

These 7 address lines set the base address of the memory

These 16 address lines will select one of the $2^{16}$ (64K) locations inside each RAM IC

This address line is implemented with UDS*/LDS*
Example 2

- A 68000-based system is to be built with these memory requirements
  - a 16K word EPROM with a starting address of $60 0000
  - a 16K word RAM with a starting address of $70 0000

- Design a FULL address decoder for this application using 16K×8 chips for both EPROM and RAM

  - $60 0000 + (16KWord=32KB=$8000)-1=$60 7FFF
  - $70 0000 + (16KWord=32KB=$8000)-1=$70 7FFF

<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>
<th></th>
<th></th>
<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>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<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>
<th></th>
<th></th>
<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>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- ROMSEL
- RAMSEL
- UDS
- LDS

\[ \text{ROMSEL}_{D8-D15} \]
\[ \text{RAMSEL}_{D8-D15} \]
Example 3

- Design a PARTIAL address decoder for a 68000-based system with only 8K words of EPROM space, and a base address at $4000, using 8Kx8 memory chips.
Example 4

- **Design a partial address decoder for a 68000-based system that contains**
  - 2MB of EPROM at a starting address $00 0000$ using 512Kx8 chips
  - 2MB of RAM at a starting address $10 0000$ using 256Kx8 chips
  - 64KB I/O space starting at $FF0000$

- **SOLUTION**
  - For the EPROM we will need 4 512Kx8 chips, organized as 2 pairs of 512x8 chips (in order to use UDS*/LDS*). We will call these pairs ROM1 and ROM2
  - For the RAM we will need 8 256Kx8 chips, organized as 4 pairs of 256Kx8: RAM1 to RAM4

<table>
<thead>
<tr>
<th>A23 A22 A21 A20 A19 A18 A17 A16 A15 A14 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00</th>
<th>ROM1</th>
<th>ROM2</th>
<th>RAM1</th>
<th>RAM2</th>
<th>RAM3</th>
<th>RAM4</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 X X X X X X X X X X X X X X X X X X X</td>
<td>0 0 0 0 1 X X X X X X X X X X X X X X X X X X X</td>
<td>0 0 1 0 0 X X X X X X X X X X X X X X X X X X X</td>
<td>0 0 1 0 1 X X X X X X X X X X X X X X X X X X X</td>
<td>0 0 1 1 0 X X X X X X X X X X X X X X X X X X X</td>
<td>0 0 1 1 1 X X X X X X X X X X X X X X X X X X X</td>
<td>1 1 1 1 1 X X X X X X X X X X X X X X X X X X X</td>
<td></td>
</tr>
</tbody>
</table>