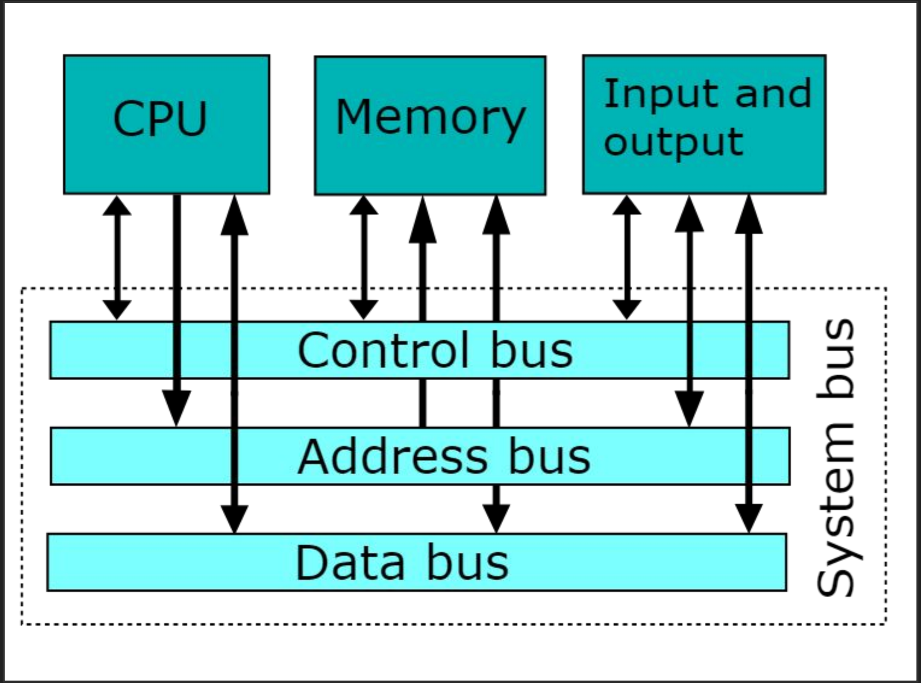
13 - Address Decoding

CEG 4330/6330 - Microprocessor-Based Embedded Systems
Max Gilson

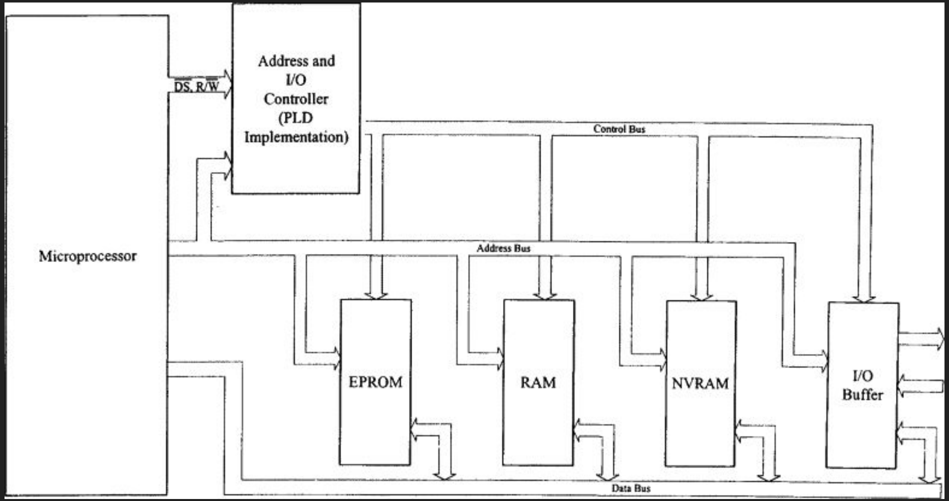
System Bus

- The communication between your processor and other components occurs over a system bus
- The system bus is made up of many smaller busses which are just bundles of wires or traces
  - Control bus
    - Selecting between different devices that share bus
  - Address bus
    - Providing an address for devices
  - Data bus
    - Providing data for devices
- Busses can be of different sizes, ex: 32-bit addresses and 8 bit data



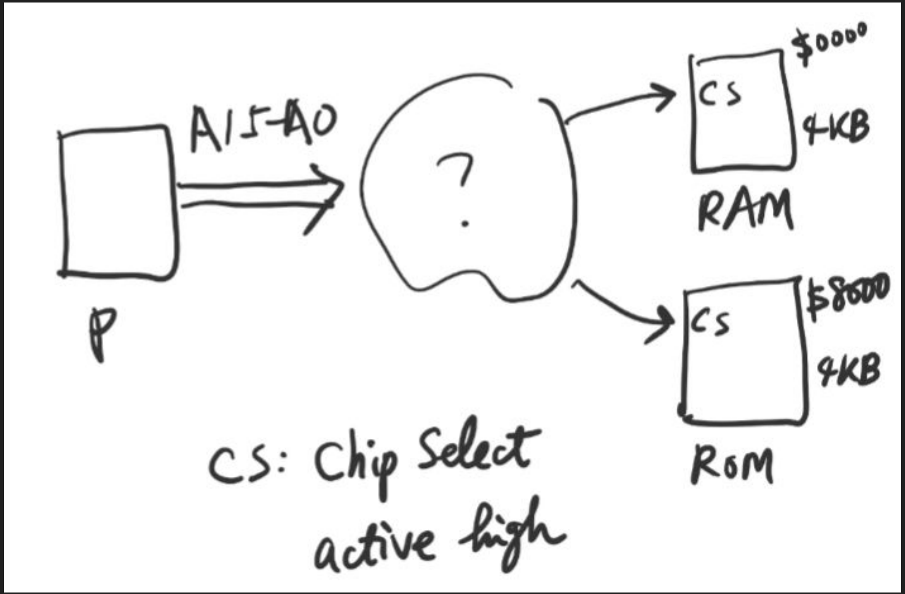
Address Decoding

- We will want to use the system bus to communicate with various devices
  - RAM, ROM, IO device 1, IO device 2, etc.
- We need a way of communicating with them independently while using the same bus
- We can take one of two approaches:
  - Centralized Address Decoder
    - One decoder for all devices
  - Distributed Address Decoder
    - Each device contains its own decoder



Centralized Address Decoding

- Assume we have a processor with 16-bit addresses
- RAM is 4096 bytes
    - Starting at address x0000
- ROM is 4096 bytes
    - Starting at address x8000
- How can we build a circuit to activate chip select in these ranges?



Partial vs Full Address Decoding

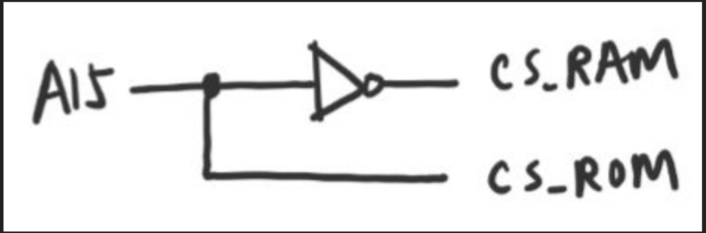
- We can take two approaches:
  - Partial address decoding
    - Ignore undefined areas of memory
    - Requires less hardware
  - Full address decoding
    - Only consider RAM and ROM sections of memory
    - Requires more hardware

Address	Data
x0000	RAM
…	
x0FFF	
x1000	Undefined
…	
x8000	ROM
…	
x8FFF	
x9000	Undefined

Partial Address Decoding

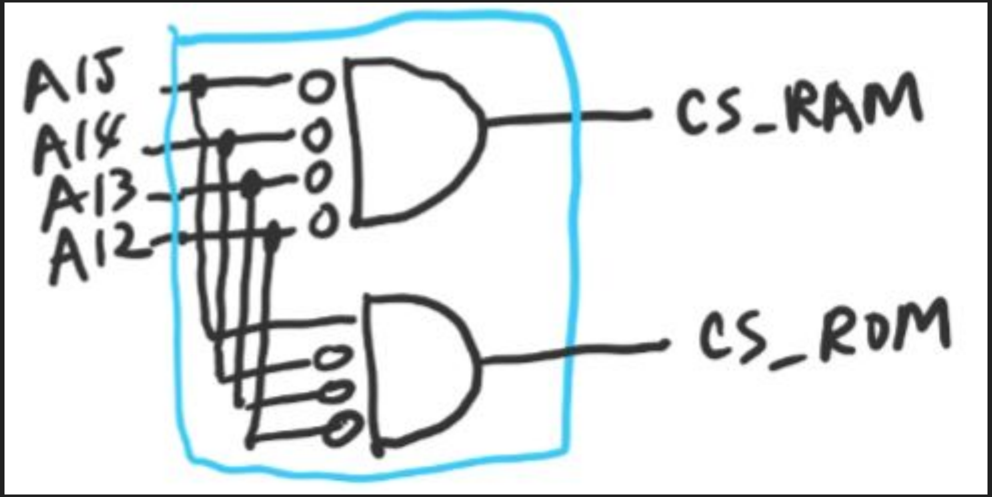
- For partial address decoding, we can use a not gate on the most significant bit
- This design sets the CS (chip select) pin of RAM and ROM when A15 is low or high, respectively

Address	Data
x0000	RAM
…	
x0FFF	
x1000	Undefined
…	
x8000	ROM
…	
x8FFF	
x9000	Undefined



Full Address Decoding

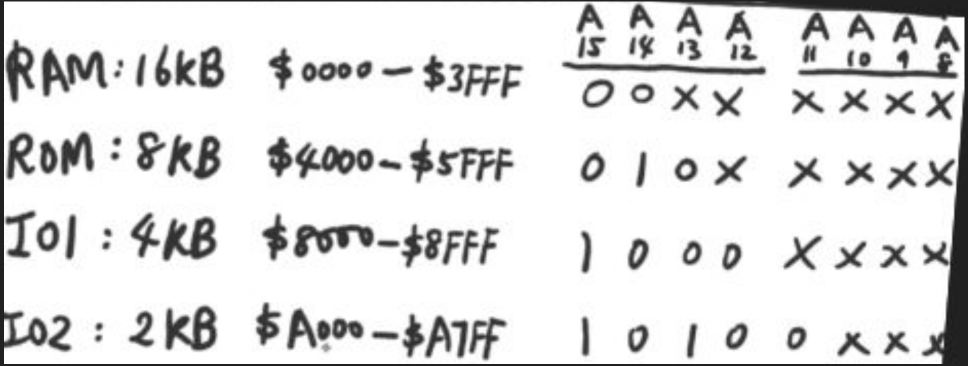
- For full address decoding, we must look at all 4 most significant bits



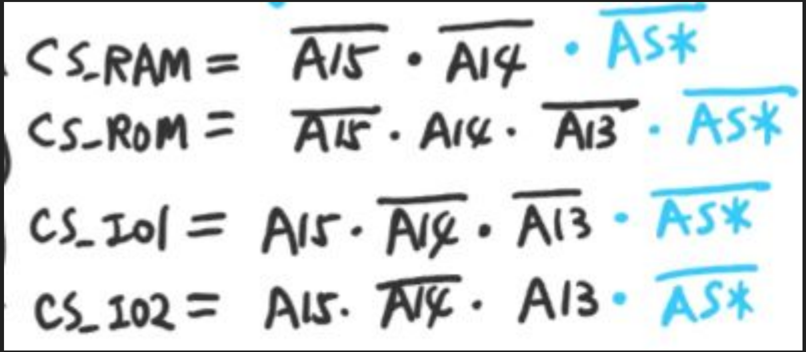
Address	Data
x0000	RAM
…	
x0FFF	
x1000	Undefined
…	
x8000	ROM
…	
x8FFF	
x9000	Undefined

Address Decoding Additional Example

- Assume we have a processor with 16-bit addresses
- RAM is 16,384 bytes
  - Starting at address x0000
- ROM is 8192 bytes
  - Starting at address x4000
- IO1 is 4096 bytes
  - Starting at address x8000
- IO2 is 2048 bytes
  - Starting at address xA000
- How can we build a circuit to activate chip select in these ranges?
  - AS* is address strobe



		A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8
RAM: 16kB	\$0000 – \$3FFF	0	0	X	X	X	X	X	X
ROM: 8KB	\$4000 – \$5FFF	0	1	0	X	X	X	X	X
IO1 : 4KB	\$8000 – \$8FFF	1	0	0	0	X	X	X	X
IO2 : 2KB	\$A000 – \$A7FF	1	0	1	0	0	X	X	X



$$CS\_RAM = \overline{A15} \cdot \overline{A14} \cdot \overline{AS*}$$

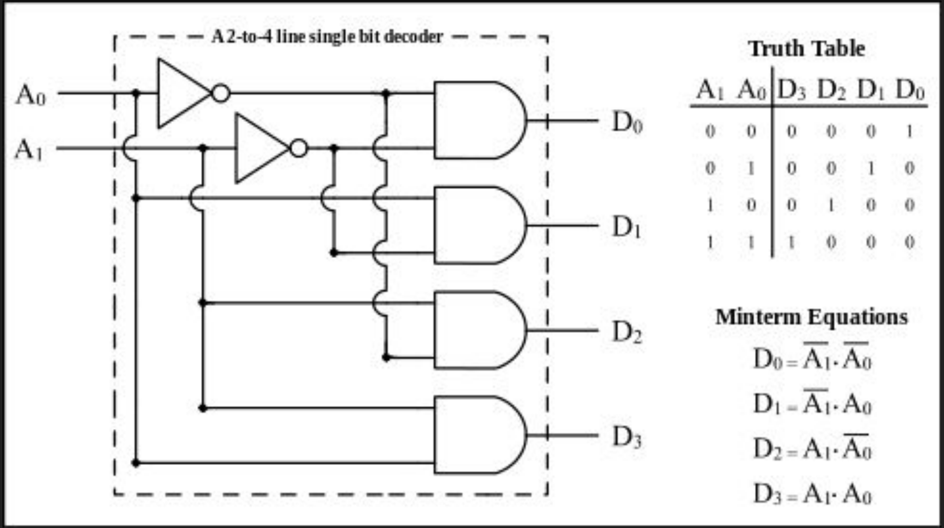
$$CS\_ROM = \overline{A15} \cdot A14 \cdot \overline{A13} \cdot \overline{AS*}$$

$$CS\_IO1 = A15 \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{AS*}$$

$$CS\_IO2 = A15 \cdot \overline{A14} \cdot A13 \cdot \overline{AS*}$$

Decoding Devices

- 2 to 4 Decoder
  - Converts a 2 bit address to 1 selection out of 4



Truth Table

A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Minterm Equations

$D_0 = \overline{A_1} \cdot \overline{A_0}$

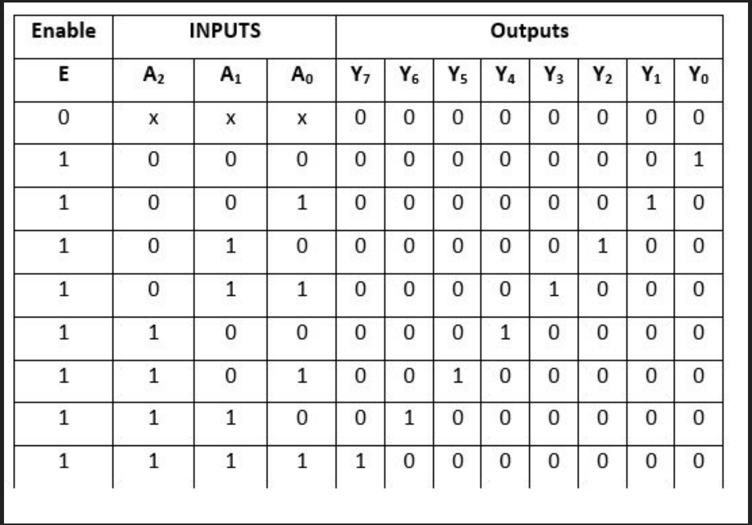
$D_1 = \overline{A_1} \cdot A_0$

$D_2 = A_1 \cdot \overline{A_0}$

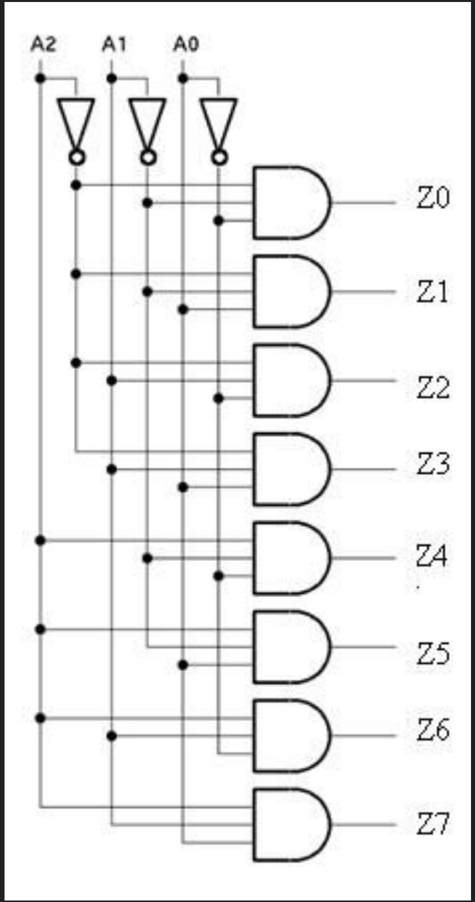
$D_3 = A_1 \cdot A_0$

Decoding Devices (cont.)

- 3 to 8 Decoder
  - Converts a 3 bit address to 1 selection out of 8

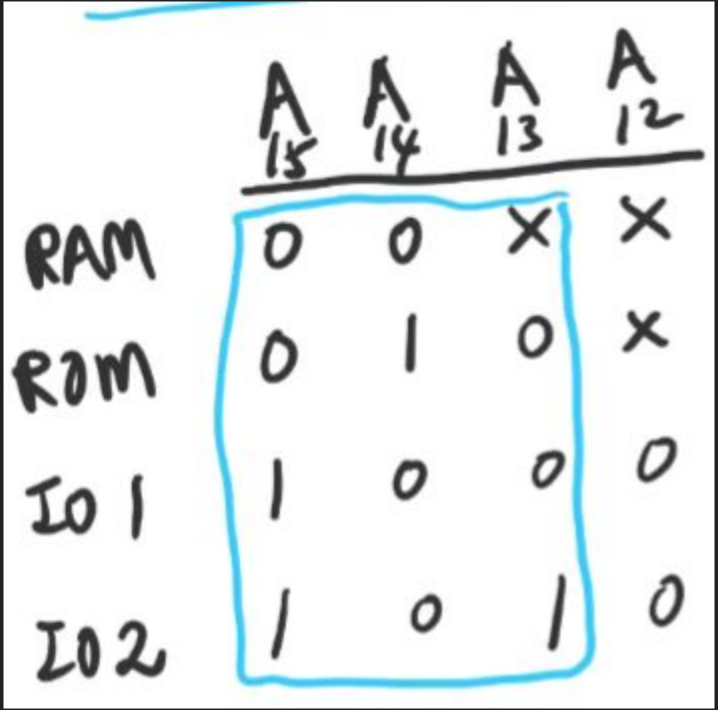


Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

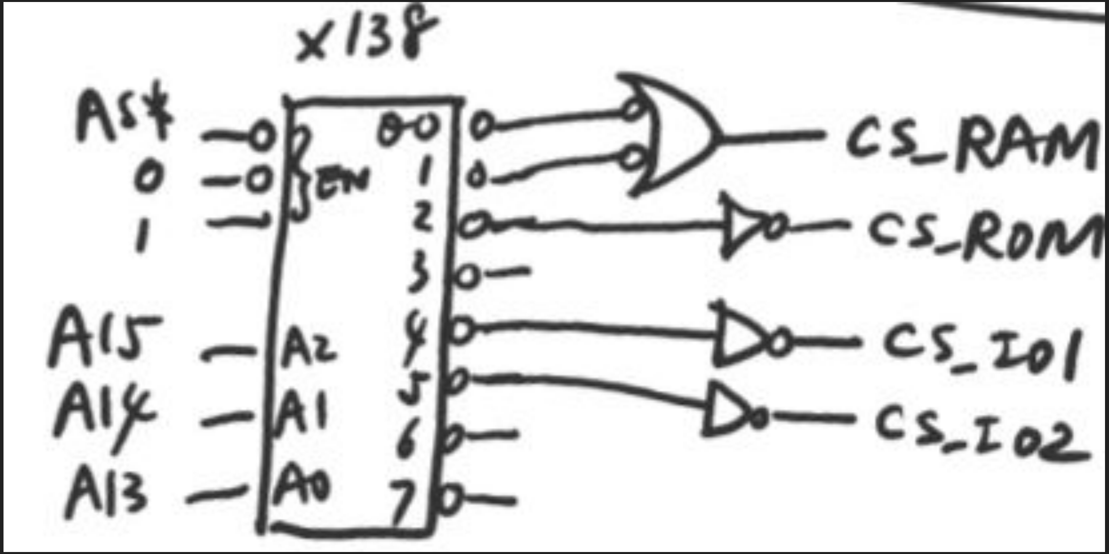


Designing Address Decoder

- 3 to 8 Decoder

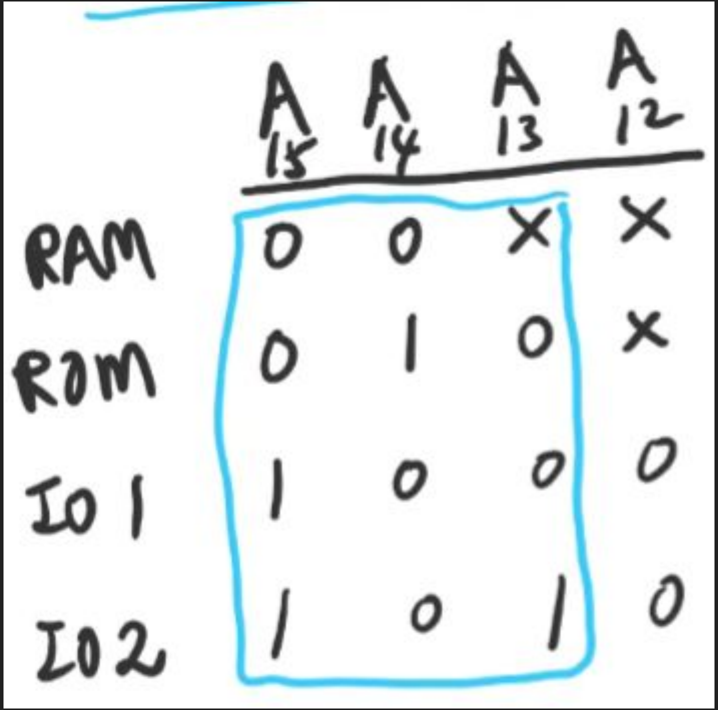


	A15	A14	A13	A12
RAM	0	0	X	X
ROM	0	1	0	X
IO1	1	0	0	0
IO2	1	0	1	0

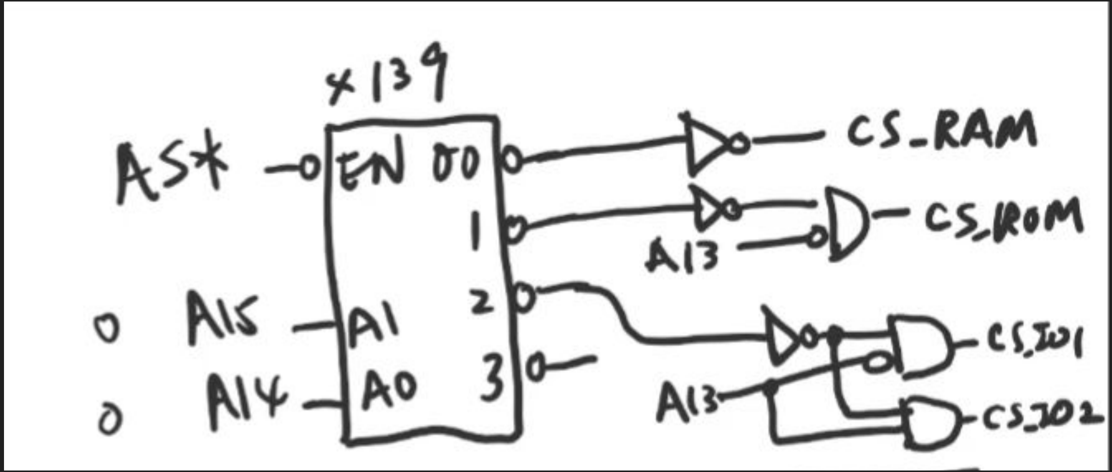


Designing Address Decoder

- 2 to 4 Decoder

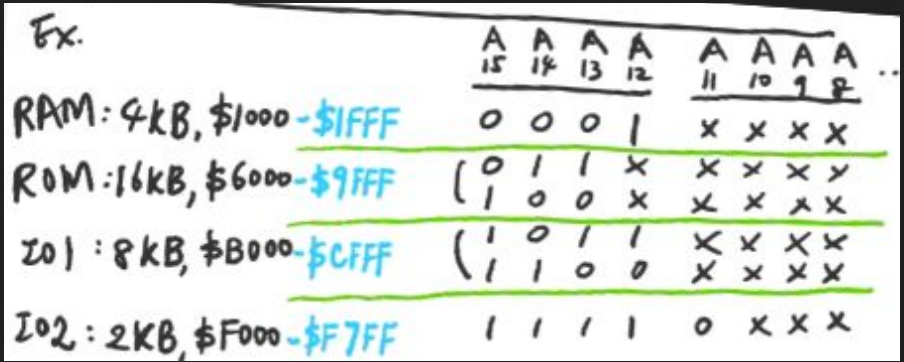


	A15	A14	A13	A12
RAM	0	0	X	X
ROM	0	1	0	X
IO1	1	0	0	0
IO2	1	0	1	0



Address Decoding Additional Example

- Assume we have a processor with 16-bit addresses
- RAM is 4096 bytes
  - Starting at address x1000
- ROM is 16,384 bytes
  - Starting at address x6000
- IO1 is 8192 bytes
  - Starting at address xB000
- IO2 is 2048 bytes
  - Starting at address xF000
- How can we build a circuit to activate chip select in these ranges?
  - Full address decoder?

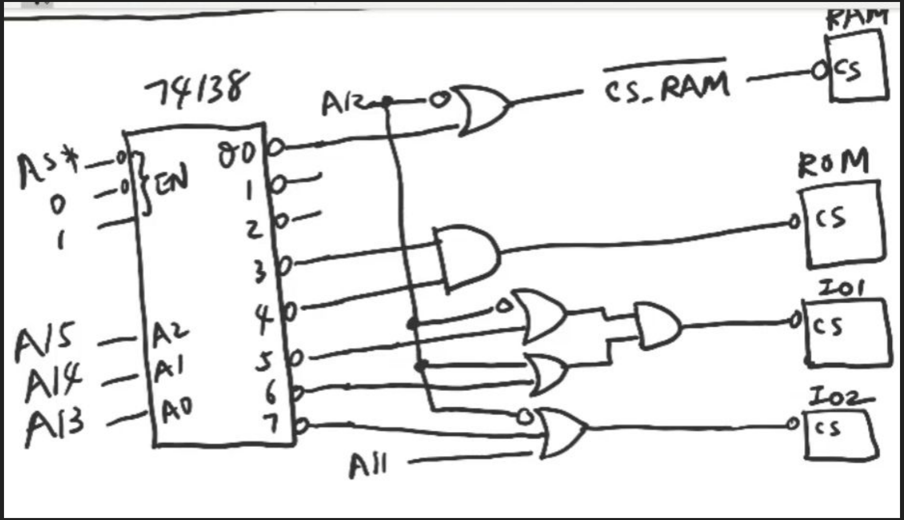


Ex.

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	..
RAM: 4kB, \$1000 - \$1FFF	0	0	0	1	x	x	x	x	
ROM: 16kB, \$6000 - \$9FFF	0	1	1	x	x	x	x	x	
	1	0	0	x	x	x	x	x	
IO1: 8kB, \$B000 - \$CFFF	1	0	1	1	x	x	x	x	
	1	1	0	0	x	x	x	x	
IO2: 2KB, \$F000 - \$F7FF	1	1	1	1	0	x	x	x	

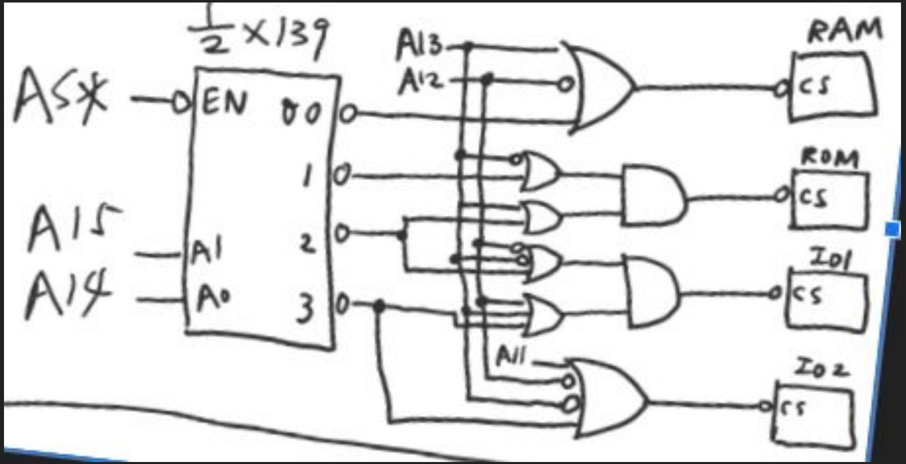
Designing Address Decoder

- 3 to 8 Decoder
- Hint: use DeMorgan's Law



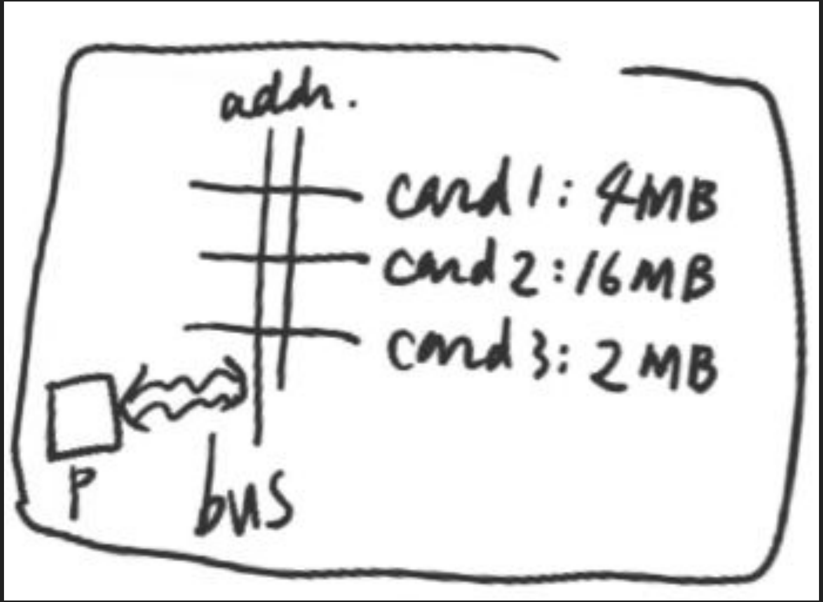
Designing Address Decoder

- 2 to 4 Decoder
- Hint: use DeMorgan's Law



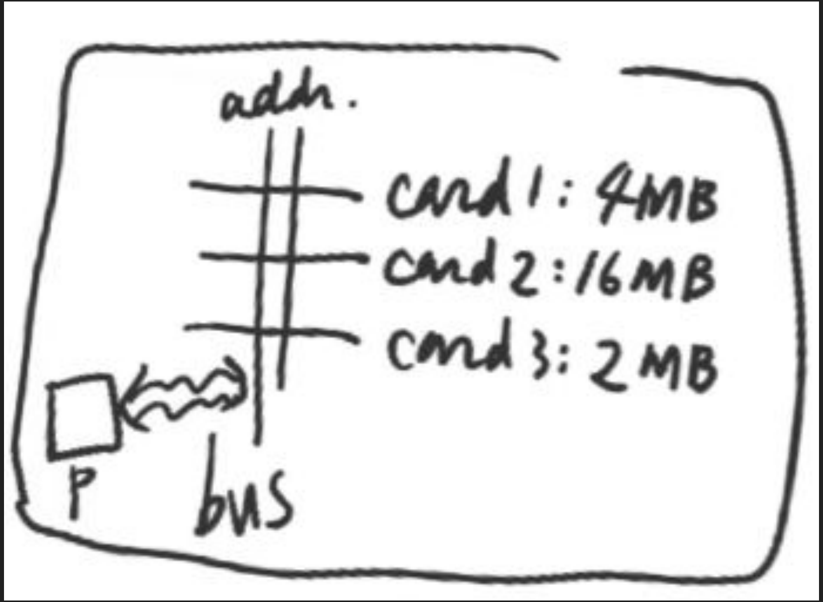
Distributed Address Decoding

- A bus may have a variety of different components that can utilize it
    - Think of a motherboard that supports adding new devices in card slots
    - The motherboard must support a wide variety of different devices that could be installed in any of the slots



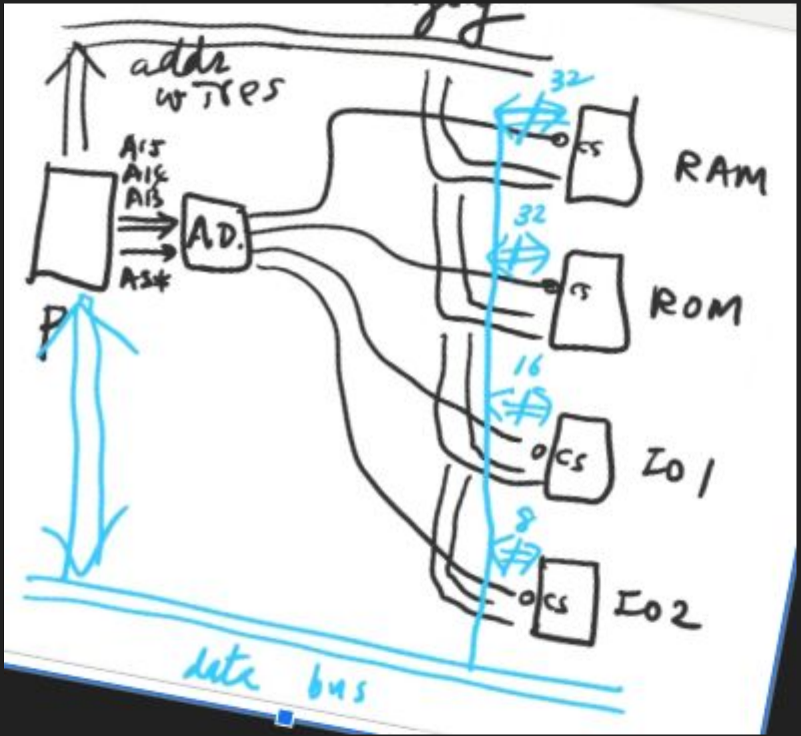
Distributed Address Decoding (cont.)

- Plug and Play (PnP) specifies that a device is capable of being installed on the bus without requiring physical modifications to the hardware or user interventions
- With PnP, at power up, each card on the bus reports its resource requirements
    - Memory size
    - Interrupts
    - Bus arbitration
- The BIOS firmware is responsible for assigning a starting address for each card based on memory size



Dynamic Bus Sizing

- Many devices will not use the same data amount as the bus for the processor
  - The processor may have 32 bit data bus
  - The IO card may have 8 bit data bus
  - The RAM may have 32 bit data bus
- If given the memory address x9F24 and processor is instructed to load 4 bytes from this address, how many bus accesses will this take?
  - Depends on the device being accessed (IO, RAM, etc.)
  - The address decoder must have this information and inform the processor



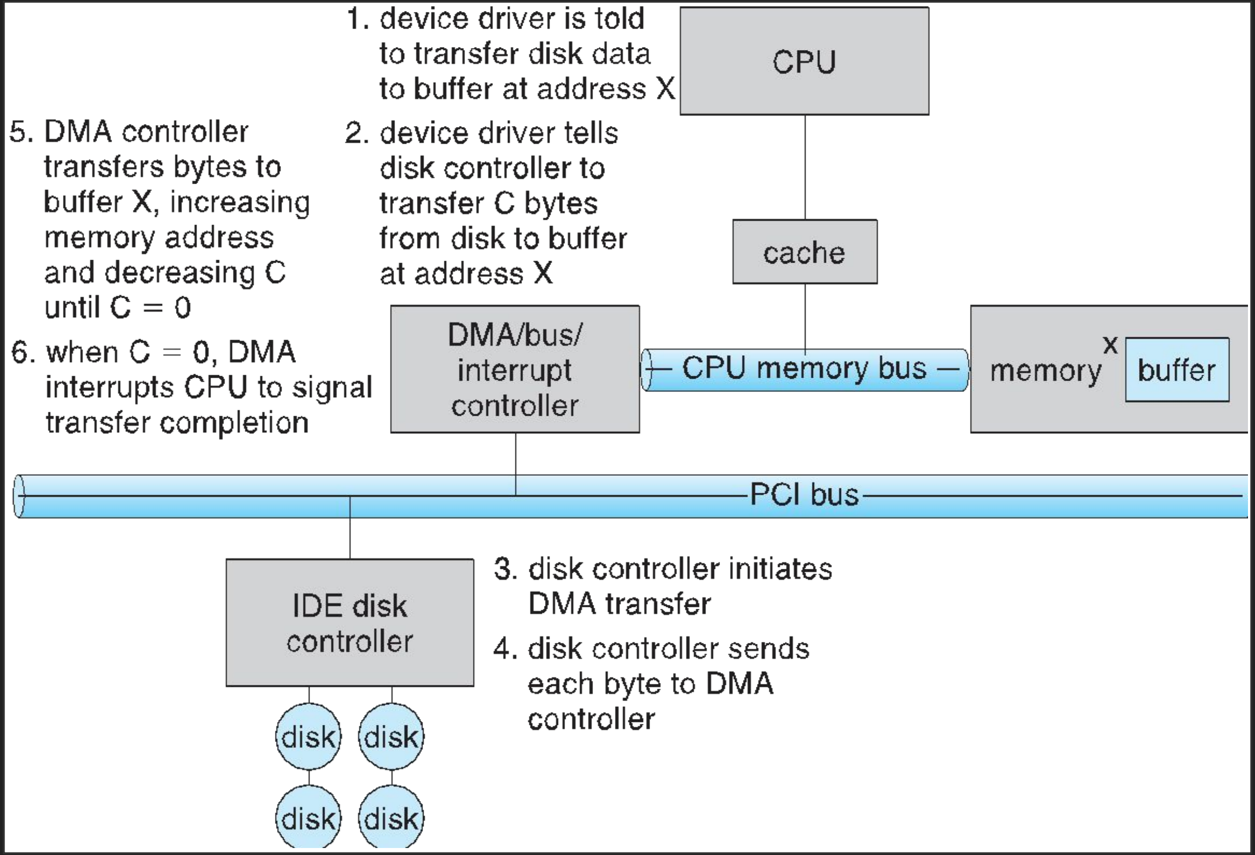
DMA and DMAC

- DMA: Direct Memory Access
- DMAC: Direct Memory Access Controller
- Many devices want to transfer large amounts of data at a time
  - Receiving data using single bytes from ports is too slow!
  - Imagine sending 4KB of data over a bus that is 4 bytes wide, this will either require 1000 interrupts or polling 1000 times (very slow!)
- Direct Memory Access (DMA) allows devices to read/write directly from/to memory without involving the CPU
- DMA requires a DMA controller that will map inputs/outputs from devices to specific areas of memory

DMA and DMAC (cont.)

- The CPU tells the DMA controller the source (specific device) and destination (memory address) for I/O accesses in memory
- Once the DMA controller has transferred all device data to memory, it interrupts the CPU so the CPU can go fetch that data
- The DMA controller and CPU are now competing for bus time but is still more efficient than requiring the CPU to read individual bytes from the device
- The DMA controller requires bus arbitration
    - It wants to use the bus to put data into memory, and the CPU wants to use the bus too, one of the devices will have to wait
- Lastly, the DMA controller sends interrupt to CPU once transfer is complete

DMA and DMAC (cont.)


1. device driver is told to transfer disk data to buffer at address X
CPU
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
cache
6. when C = 0, DMA interrupts CPU to signal transfer completion
DMA/bus/ interrupt controller
CPU memory bus
memory
x
buffer
PCI bus
IDE disk controller
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
disk
disk
disk
disk