Memory System Design

Chapter 16

S. Dandamudi

Outline

- Introduction
- A simple memory block
    - Memory design with D flip flops
    - Problems with the design
- Techniques to connect to a bus
    - Using multiplexers
    - Using open collector outputs
    - Using tri-state buffers
- Building a memory block
- Building larger memories
- Mapping memory
    - Full mapping
    - Partial mapping
- Alignment of data
- Interleaved memories
    - Synchronized access organization
    - Independent access organization
    - Number of banks

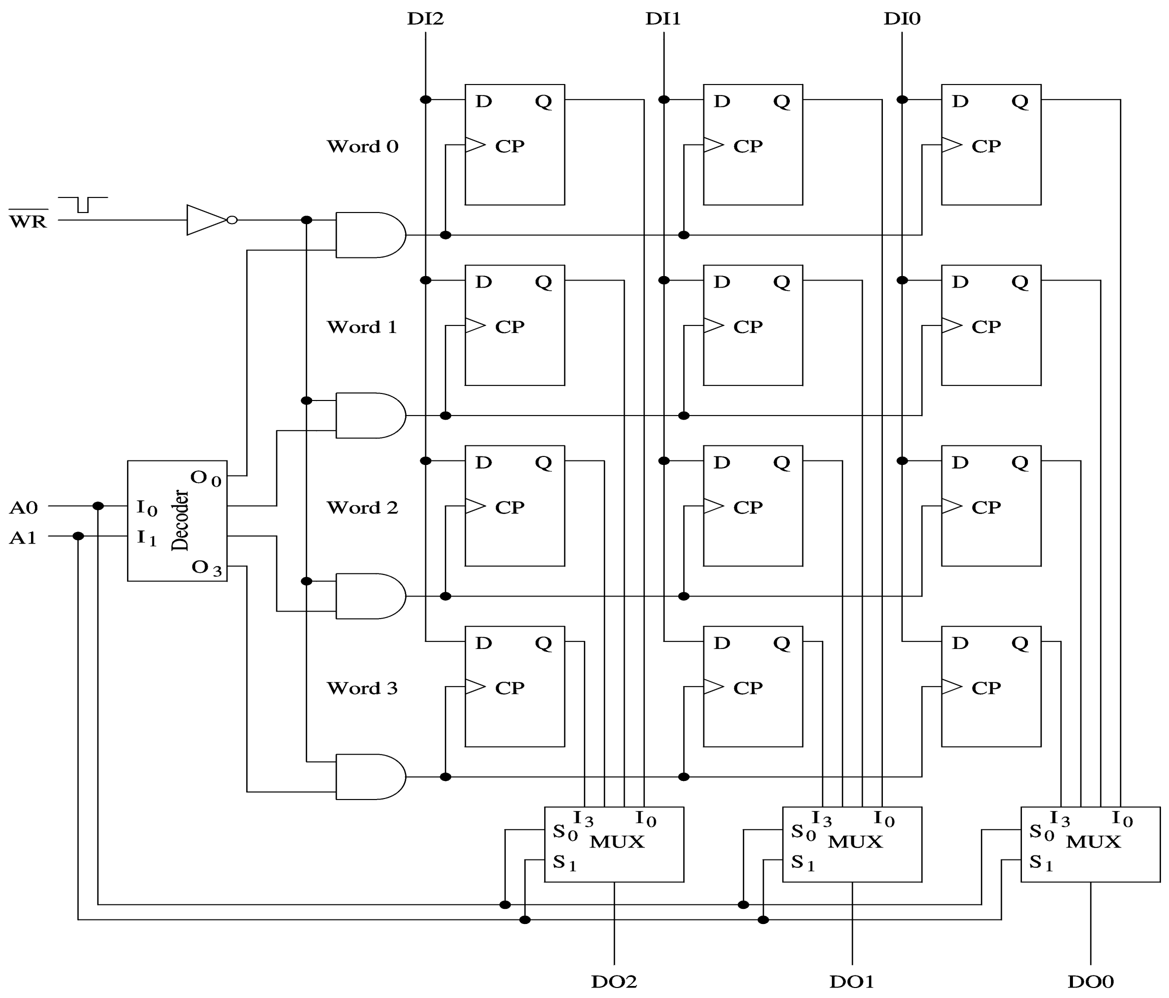
Introduction

- To store a single bit, we can use
  - ∗ Flip flops or latches
- Larger memories can be built by
  - ∗ Using a 2D array of these 1-bit devices
    - » "Horizontal" expansion to increase word size
    - » "Vertical" expansion to increase number of words
- Dynamic RAMs use a tiny capacitor to store a bit
- Design concepts are mostly independent of the actual technique used to store a bit of data

Memory Design with D Flip Flops

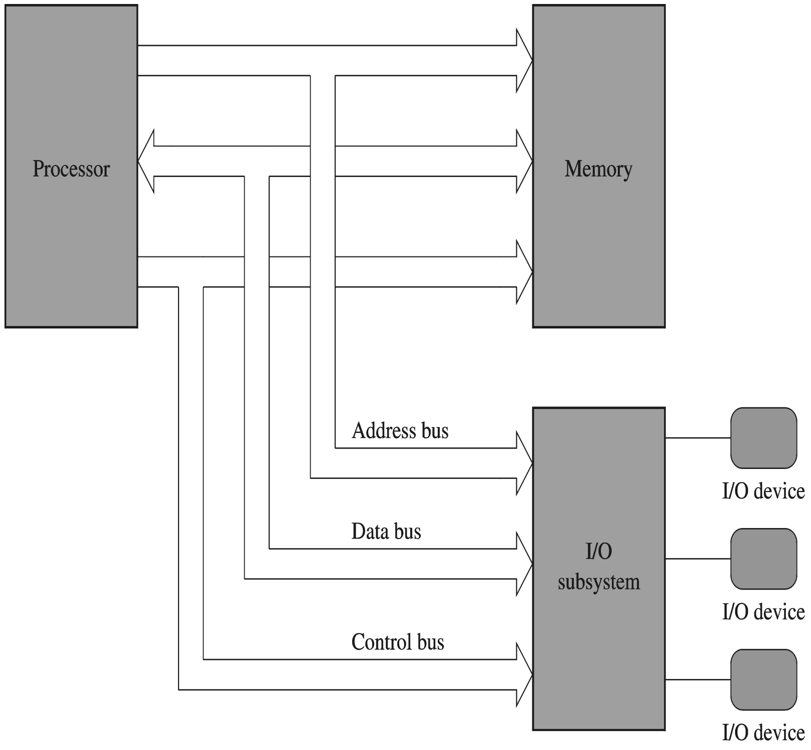
- An example
  - ∗ 4X3 memory design
  - ∗ Uses 12 D flip flops in a 2D array
  - ∗ Uses a 2-to-4 decoder to select a row (i.e. a word)
  - ∗ Multiplexers are used to gate the appropriate output
  - ∗ A single WRITE (WR) is used to serve as a write and read signal
    - – zero is used to indicate write operation
    - – one is used for read operation
  - ∗ Two address lines are needed to select one of four words of 3 bits each

Memory Design with D Flip Flops (cont'd)



Memory Design with D Flip Flops (cont'd)

- Problems with the design
  - ∗ Requires separate data in and out lines
    - » Cannot use the bidirectional data bus
  - ∗ Cannot use this design as a building block to design larger memories
    - » To do this, we need a chip select input
- We need techniques to connect multiple devices to a bus

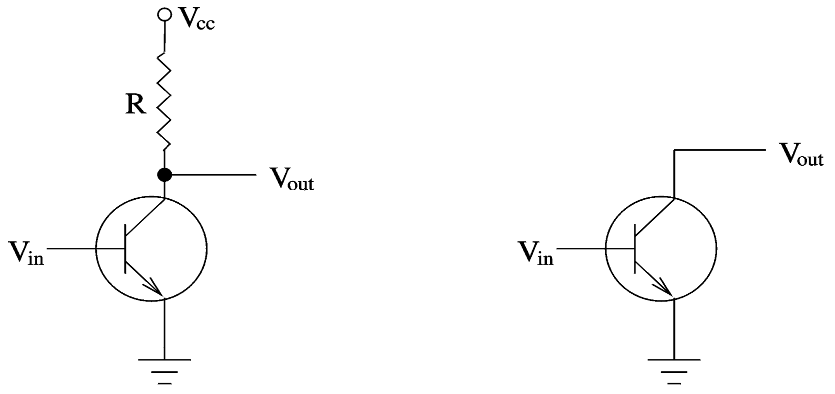


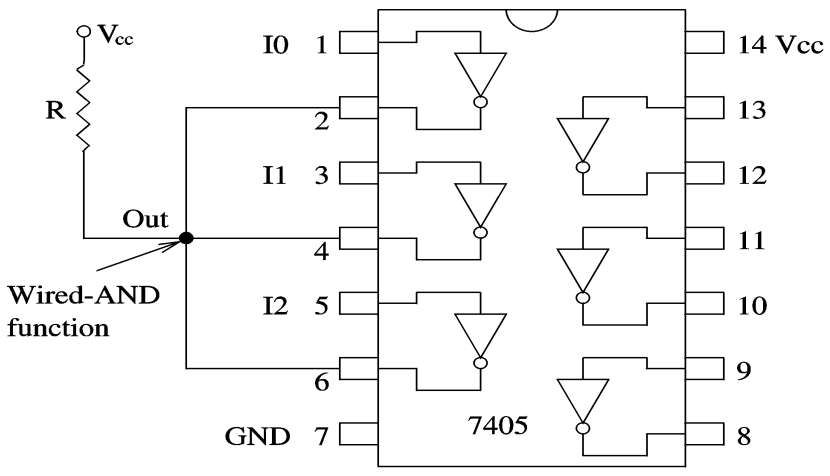
Techniques to Connect to a Bus

- Three techniques
  - ∗ Use multiplexers
    - » Example
      - – We used multiplexers in the last memory design
    - » We cannot use MUXs to support bidirectional buses
  - ∗ Use open collector outputs
    - » Special devices that facilitate connection of several outputs *together*
  - ∗ Use tri-state buffers
    - » Most commonly used devices

Techniques to Connect to a Bus (cont'd)

Open collector technique



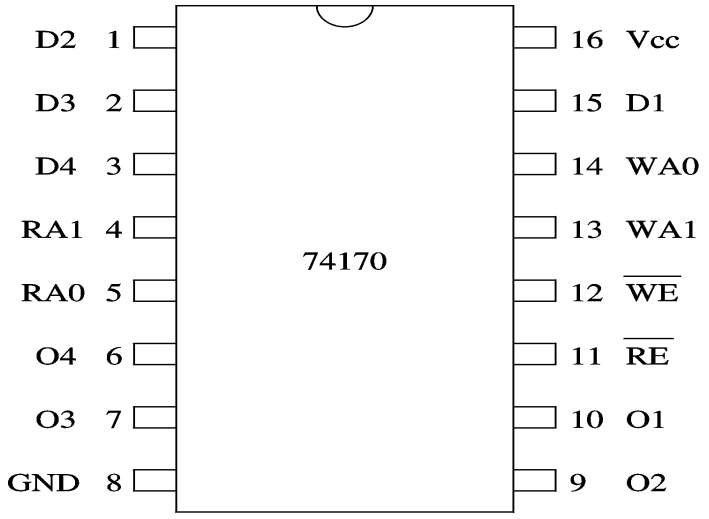


(a) Connection diagram

I2	I1	I0	Out
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

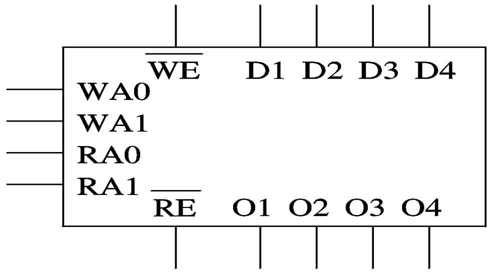
(b) Truth table

Techniques to Connect to a Bus (cont'd)



(a) Connection diagram

Open collector register chip



(b) Logic symbol

$\overline{\text{WE}}$	WA1	WA0	D inputs to
0	0	0	Word 0
0	0	1	Word 1
0	1	0	Word 2
0	1	1	Word 3
1	X	X	None

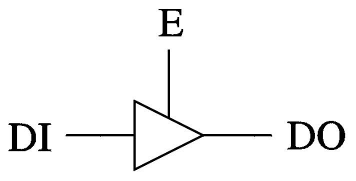
(c) Write function table

$\overline{\text{RE}}$	RA1	RA0	Output from
0	0	0	Word 0
0	0	1	Word 1
0	1	0	Word 2
0	1	1	Word 3
1	X	X	None (Z)

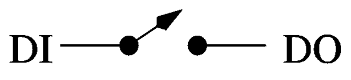
(d) Read function table

Techniques to Connect to a Bus (cont'd)

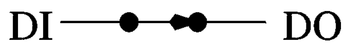
Tri-State Buffers



(a)



(b) E = 0



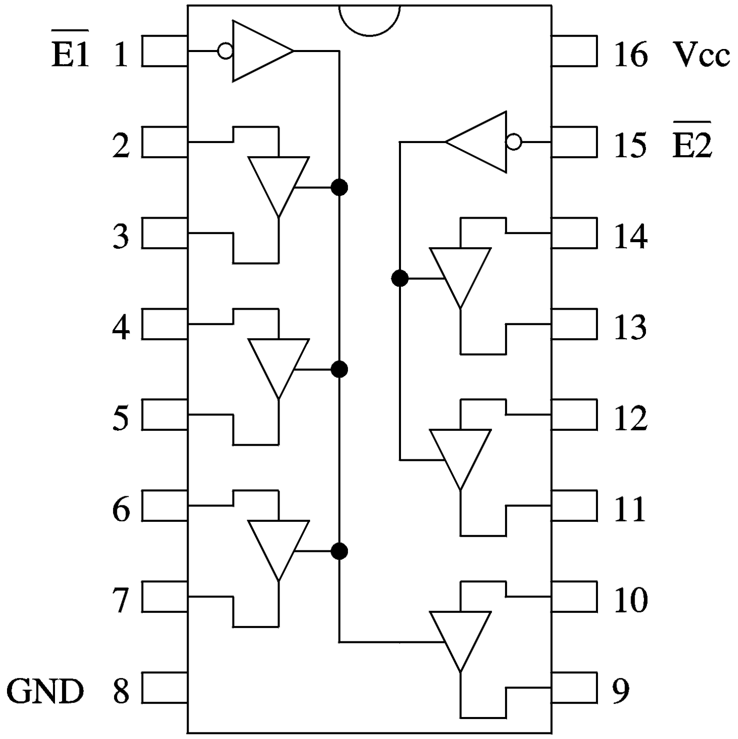
(c) E = 1

Inputs		Output
E	DI	DO
1	0	0
1	1	1
0	X	Z

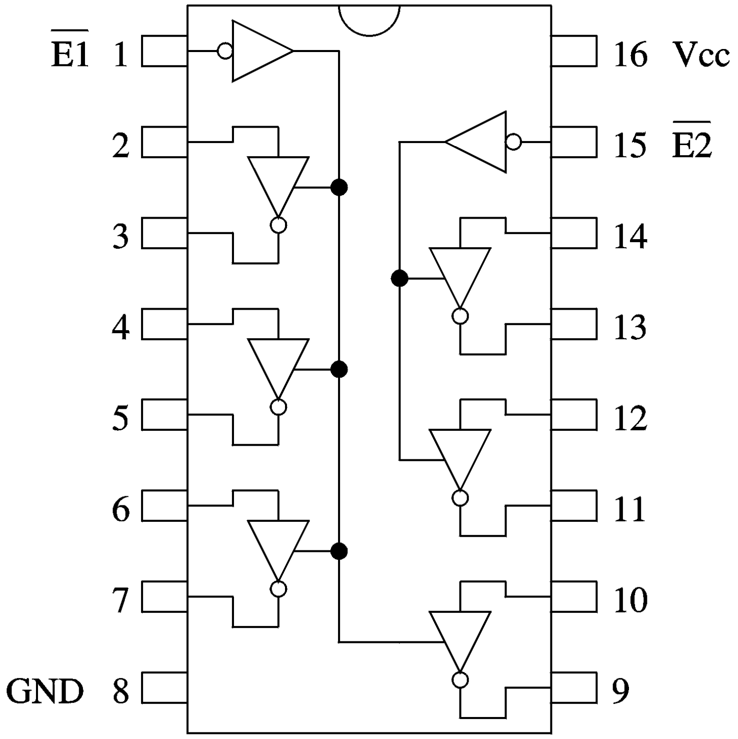
(d)

Techniques to Connect to a Bus (cont'd)

Two example tri-state buffer chips

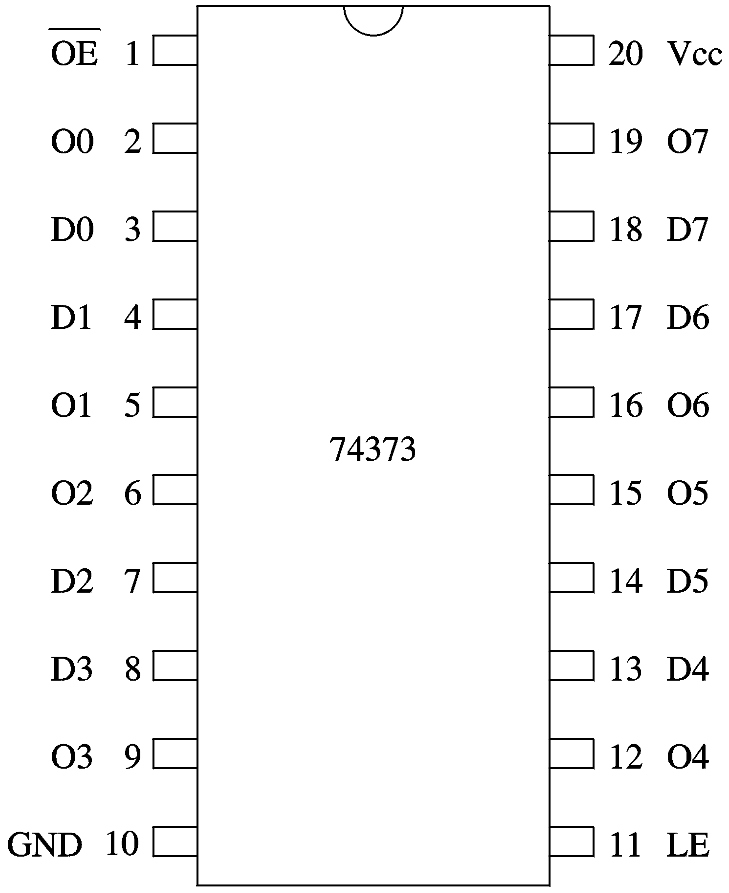


74367



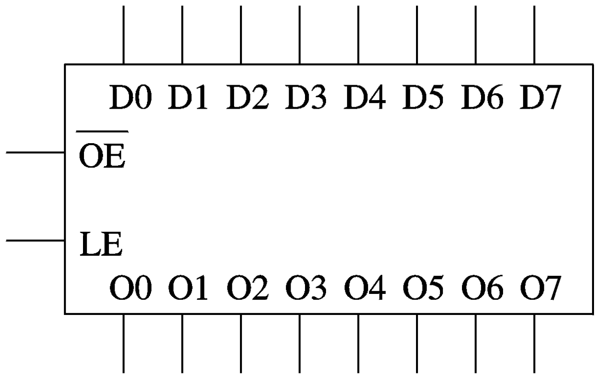
74368

Techniques to Connect to a Bus (cont'd)



(a) Connection diagram

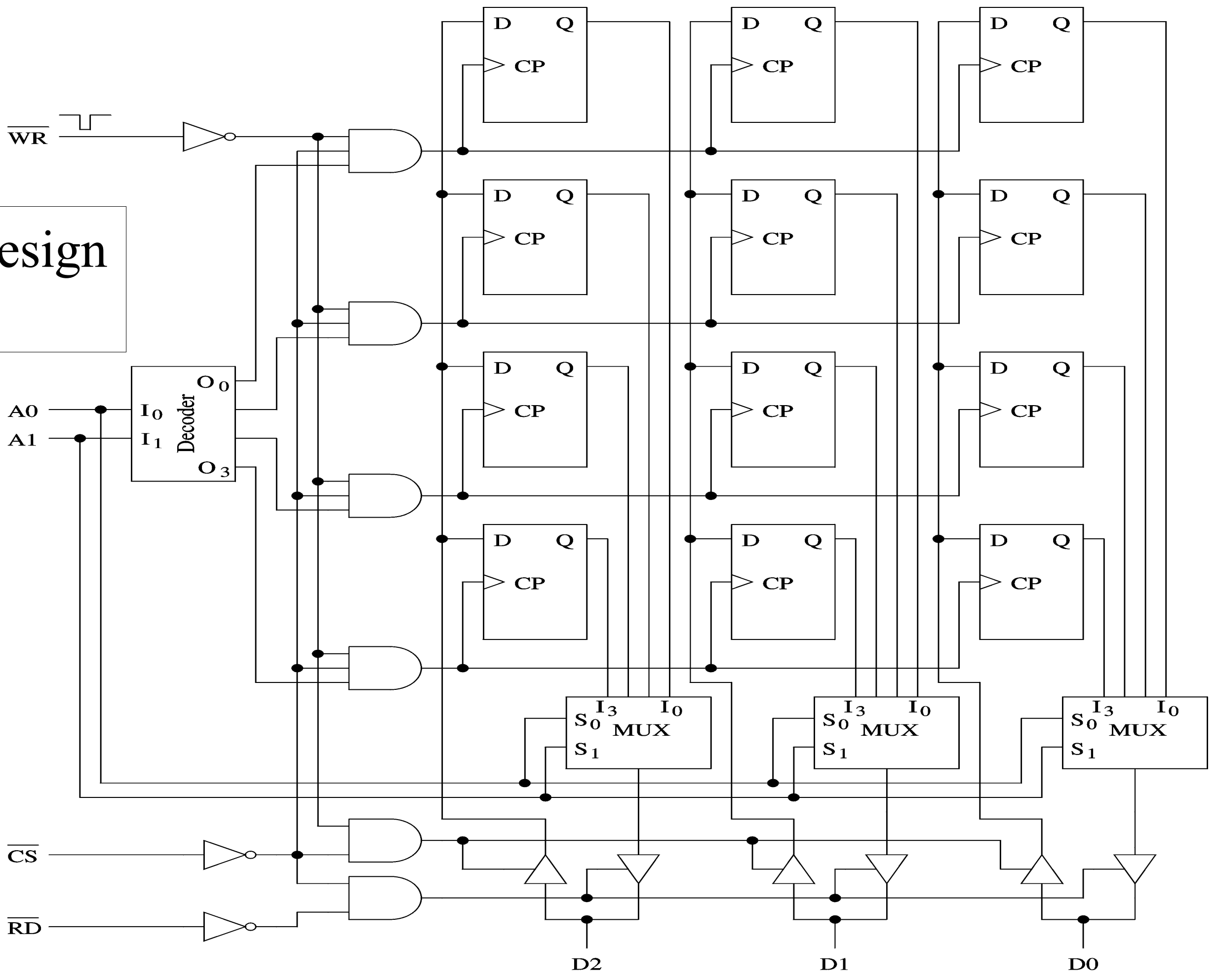
8-bit tri-state register



(b) Logic symbol

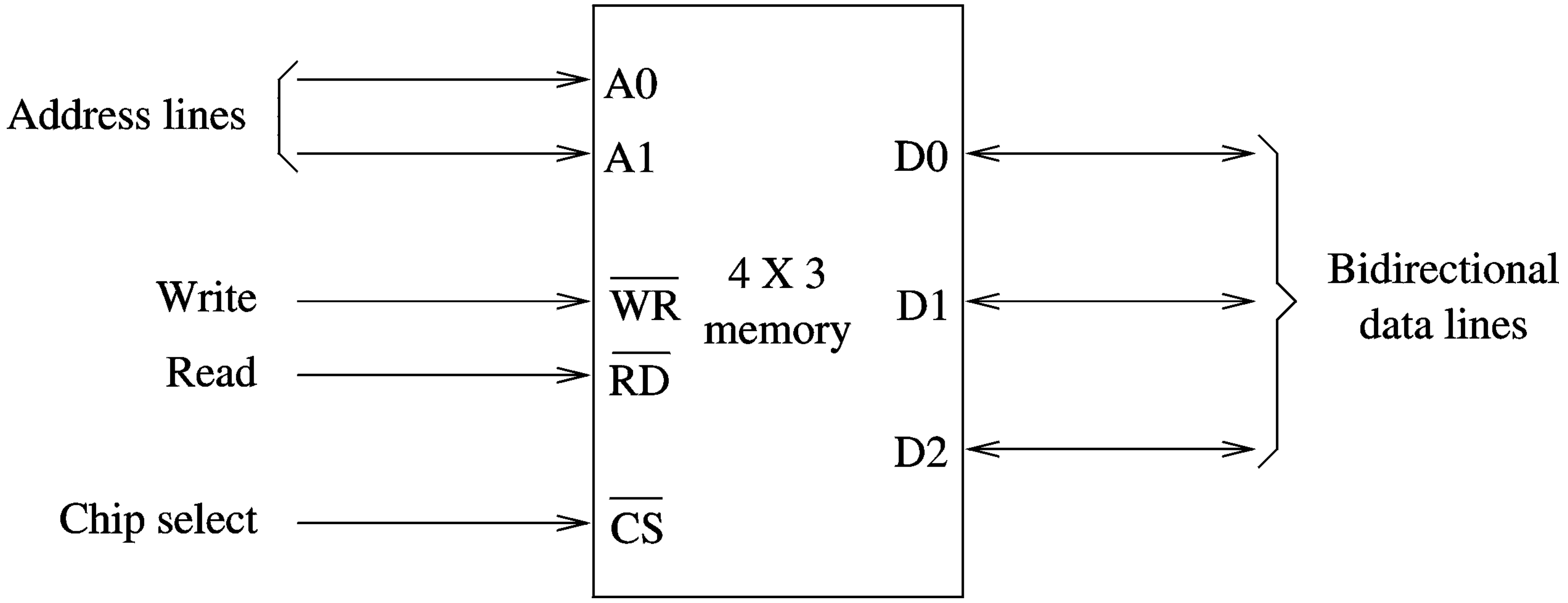
Building a Memory Block

A 4 X 3 memory design using D flip-flops



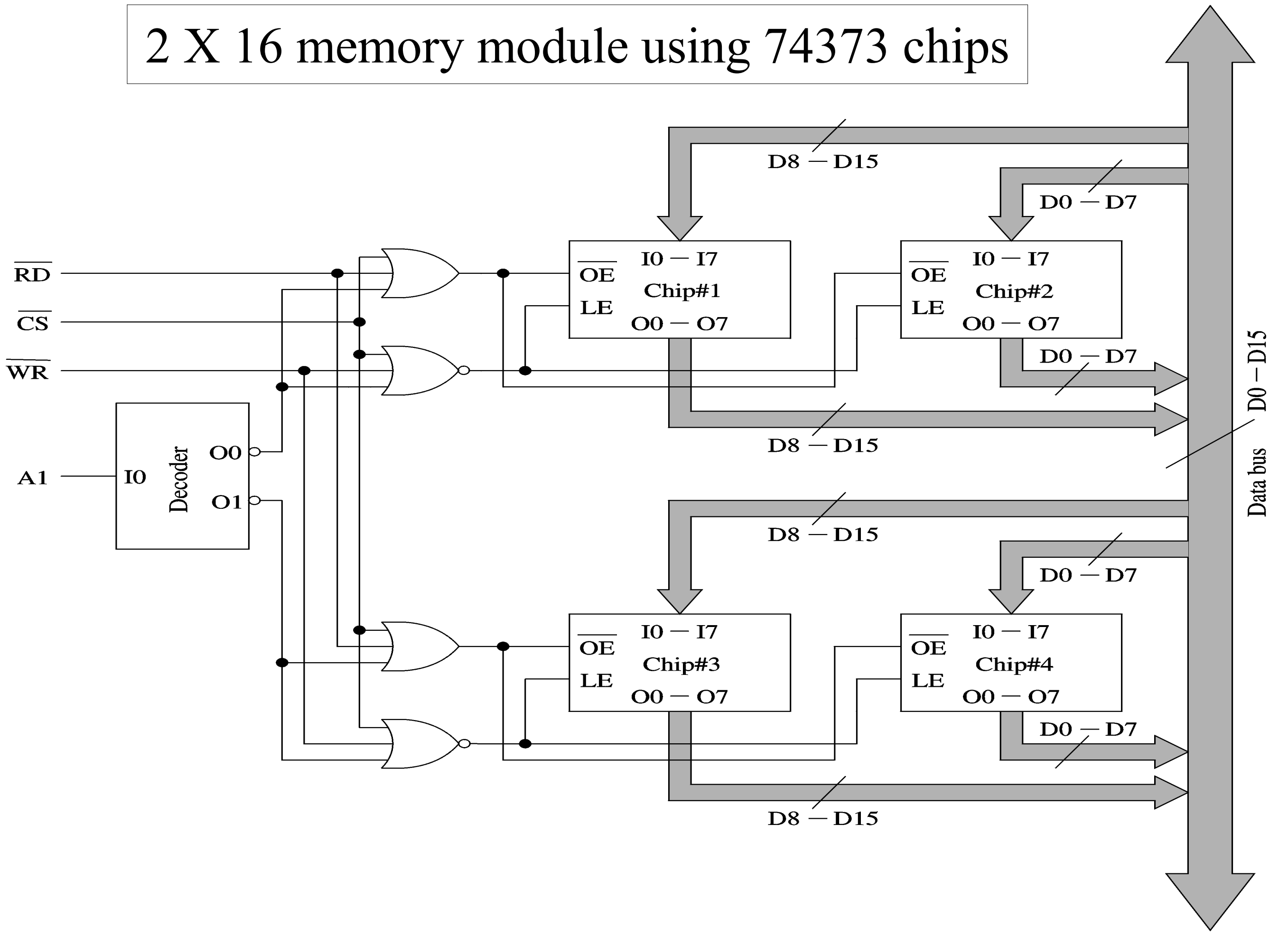
Building a Memory Block (cont'd)

Block diagram representation of a 4x3 memory



Building Larger Memories

2 X 16 memory module using 74373 chips

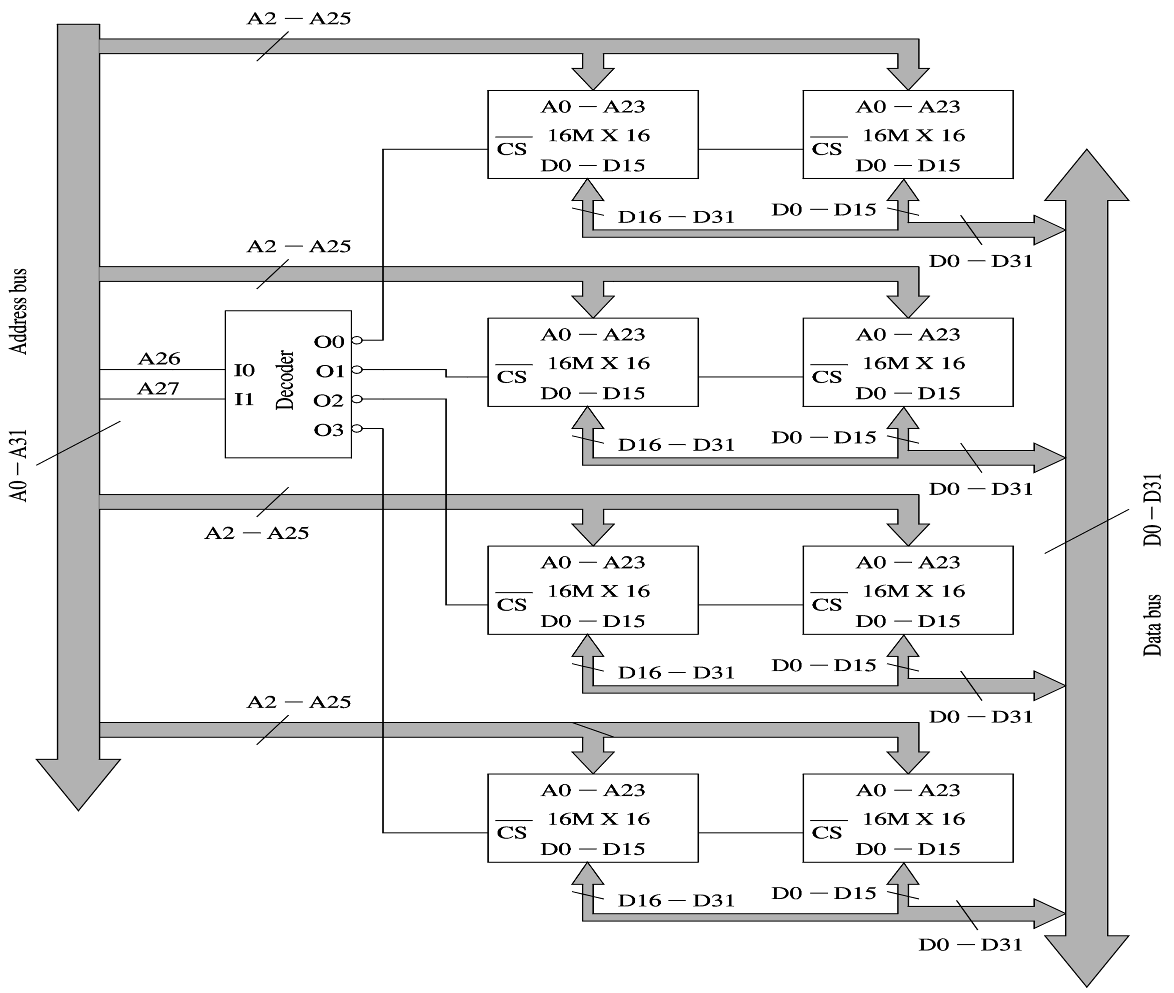


Designing Larger Memories

- Issues involved
  - ∗ Selection of a memory chip
    - » Example: To design a 64M X 32 memory, we could use
      - – Eight 64M X 4 in 1 X 8 array (i.e., single row)
      - – Eight 32M X 8 in 2 X 4 array
      - – Eight 16M X 16 in 4 X 2 array
- Designing M X N memory with D X W chips
  - ∗ Number of chips = M·N/D·W
  - ∗ Number of rows = M/D
  - ∗ Number of columns = N/W

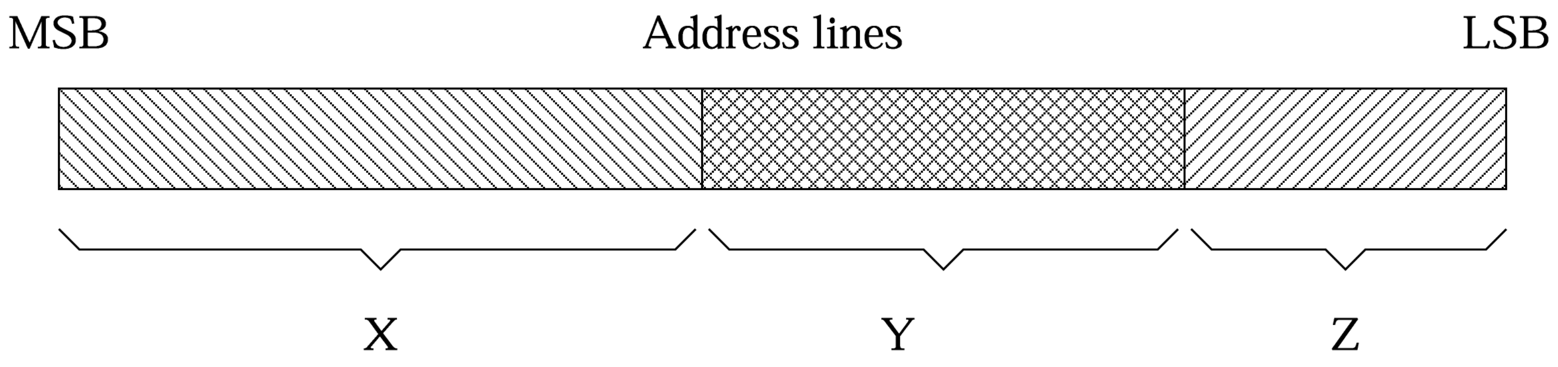
Designing Larger Memories (cont'd)

64M X 32 memory using 16M X 16 chips



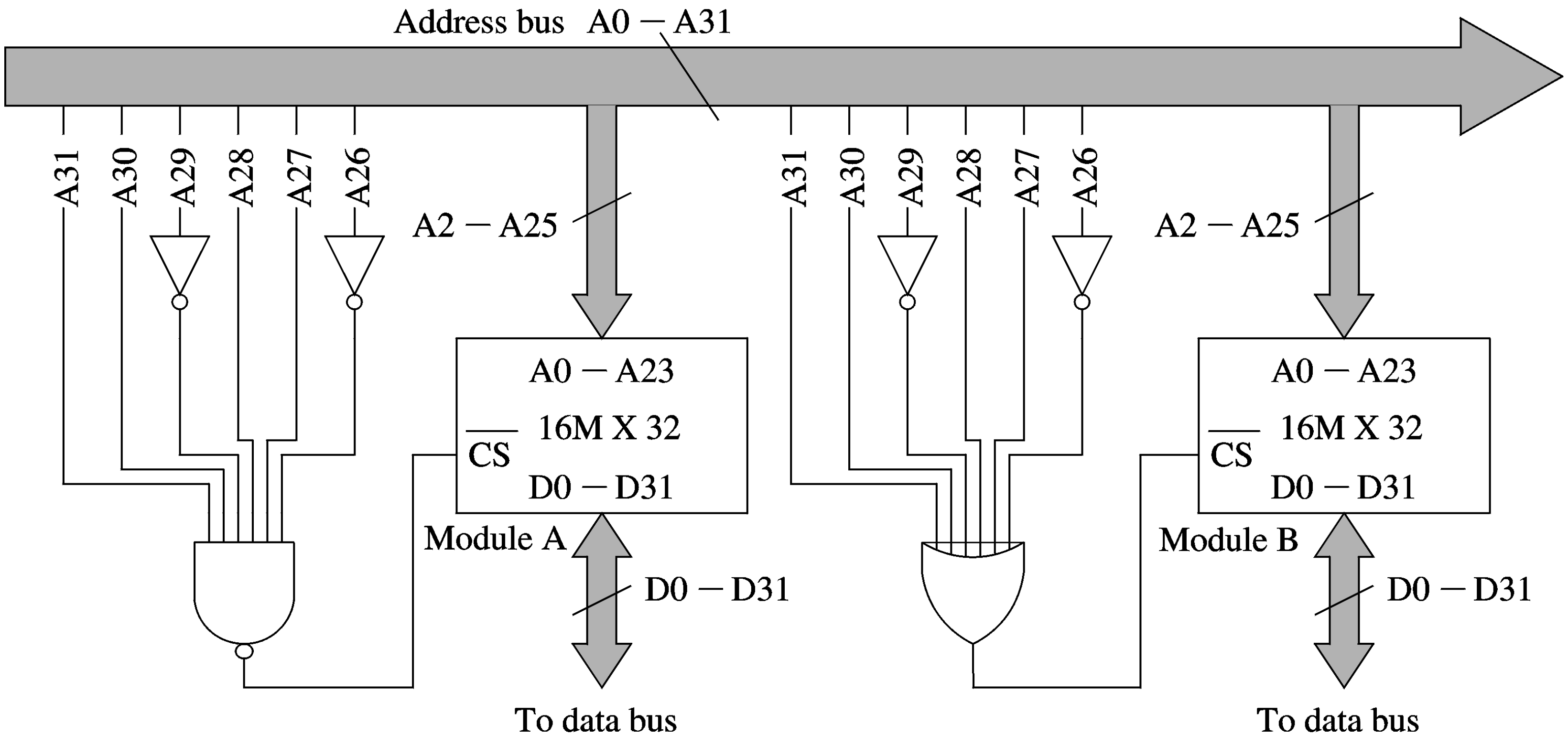
Designing Larger Memories (cont'd)

- Design is simplified by partitioning the address lines (M X N memory with D X W memory chips)
  - Z bits are not connected ($Z = \log_2(N/8)$)
  - Y bits are connected to all chips ($Y = \log_2 D$)
  - X remaining bits are used to map the memory block
    - » Used to generate chip selects



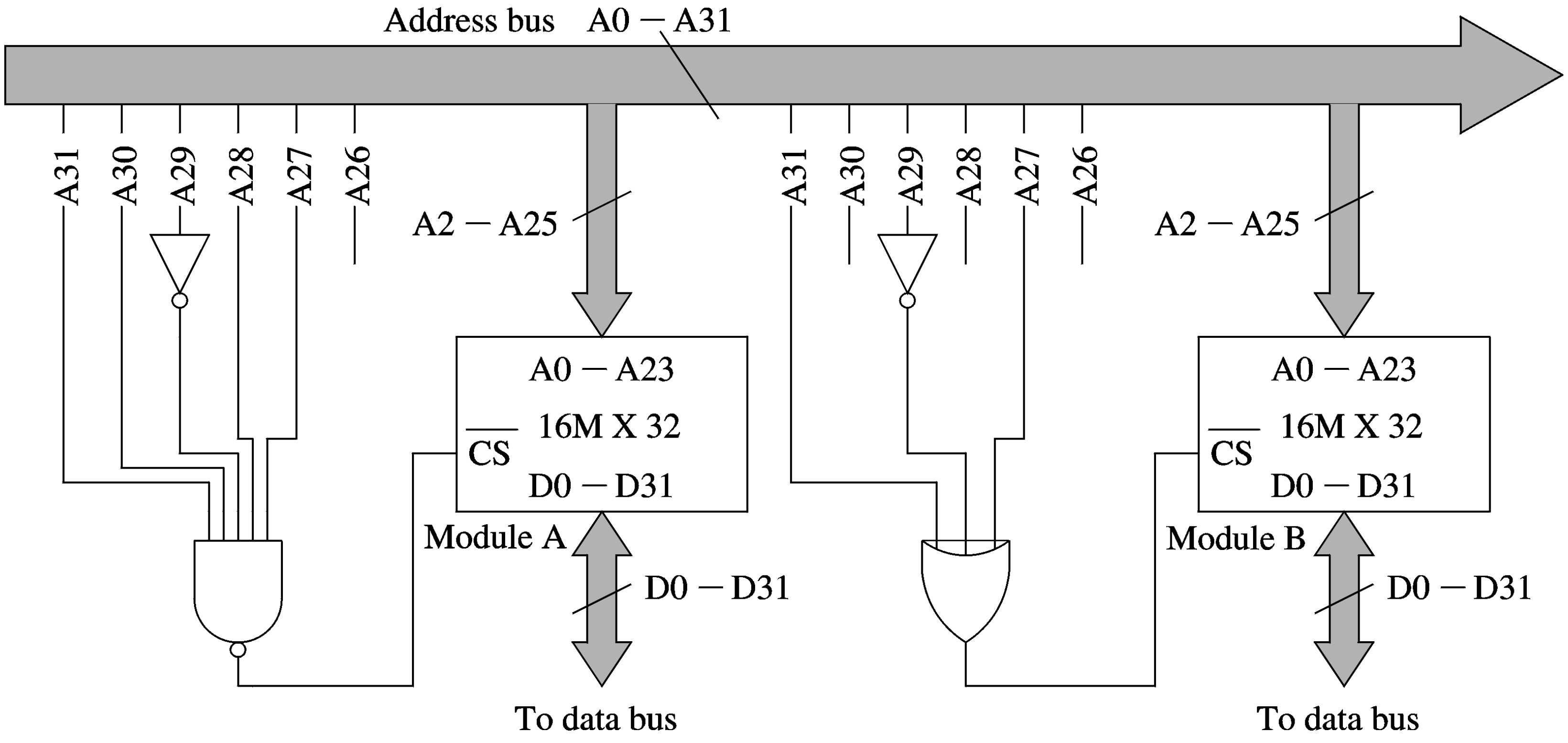
Memory Mapping

Full mapping

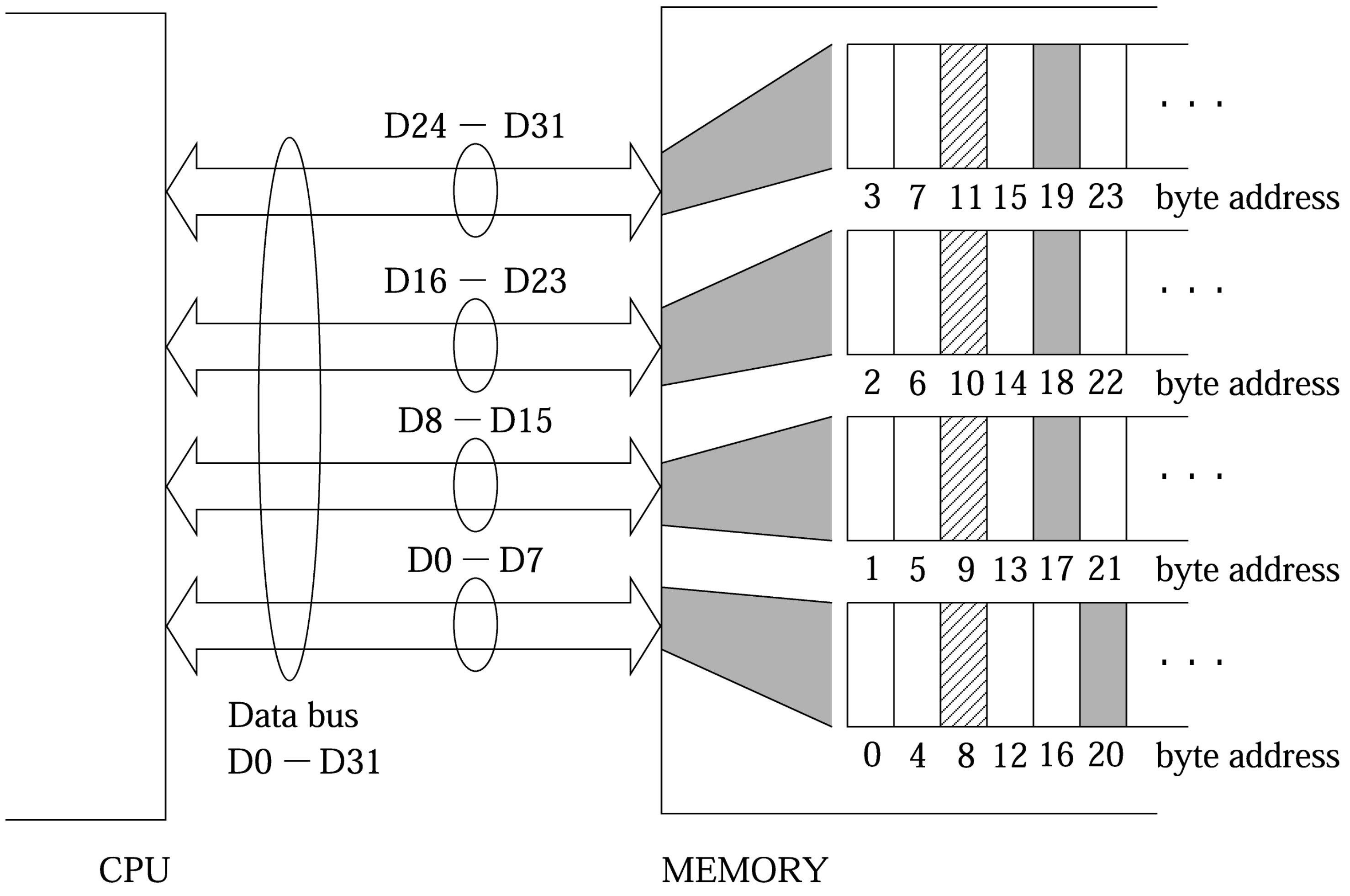


Memory Mapping (cont'd)

Partial mapping



Alignment of Data


D24 – D31
D16 – D23
D8 – D15
D0 – D7
Data bus
D0 – D31
3 7 11 15 19 23 byte address
2 6 10 14 18 22 byte address
1 5 9 13 17 21 byte address
0 4 8 12 16 20 byte address
. . .
CPU
MEMORY

Alignment of Data (cont’d)

- Alignment
  - ∗ 2-byte data: Even address
    - » Rightmost address bit should be zero
  - ∗ 4-byte data: Address that is multiple of 4
    - » Rightmost 2 bits should be zero
  - ∗ 8-byte data: Address that is multiple of 8
    - » Rightmost 3 bits should be zero
  - ∗ Soft alignment
    - » Can handle aligned as well as unaligned data
  - ∗ Hard alignment
    - » Handles only aligned data (enforces alignment)

Interleaved Memory

- In our memory designs
  - Block of contiguous memory addresses is mapped to a module
    - One advantage
      - Incremental expansion
    - Disadvantage
      - Successive accesses take more time
        - Not possible to hide memory latency
- Interleaved memories
  - Improve access performance
    - Allow overlapped memory access
    - Use multiple banks and access all banks simultaneously
      - Addresses are spread over banks
        - Not mapped to a single memory module

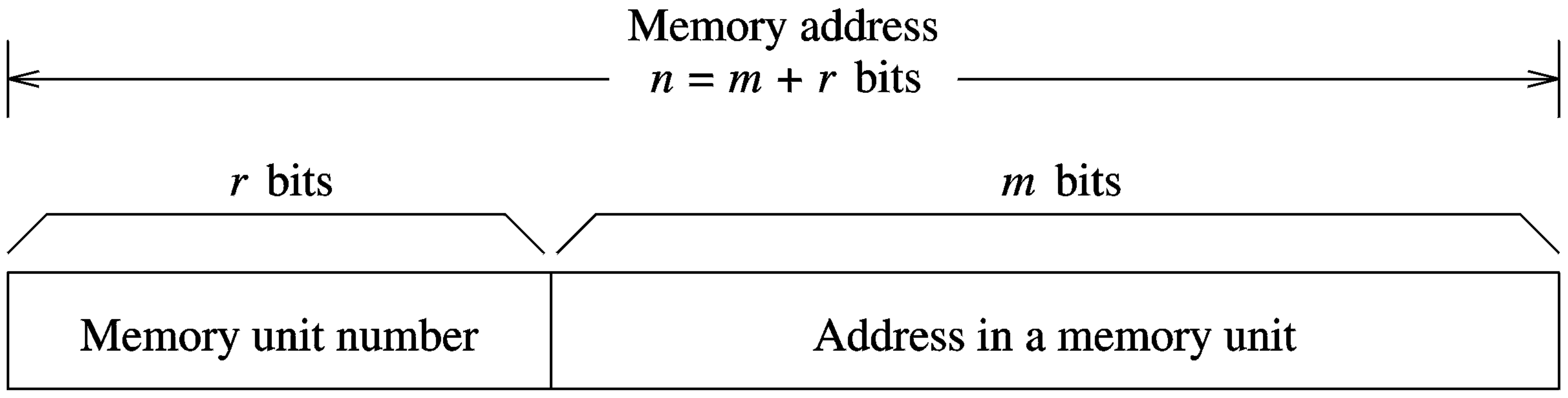
Interleaved Memory (cont'd)

- The ***n***-bit address is divided into two ***r*** and ***m*** bits:

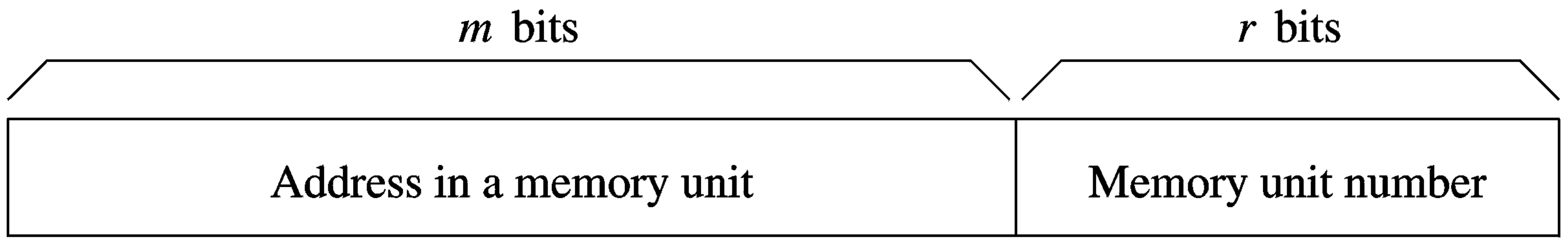
  $$n = r + m$$

- Normal memory
  - Higher order ***r*** bits identify a module
  - Lower order ***m*** bits identify a location in the module
    - » Called high-order interleaving
- Interleaved memory
  - Lower order ***r*** bits identify a module
  - Higher order ***m*** bits identify a location in the module
    - » Called low-order interleaving
  - Memory modules are referred to as memory banks

Interleaved Memory (cont'd)



(a) Normal memory address mapping



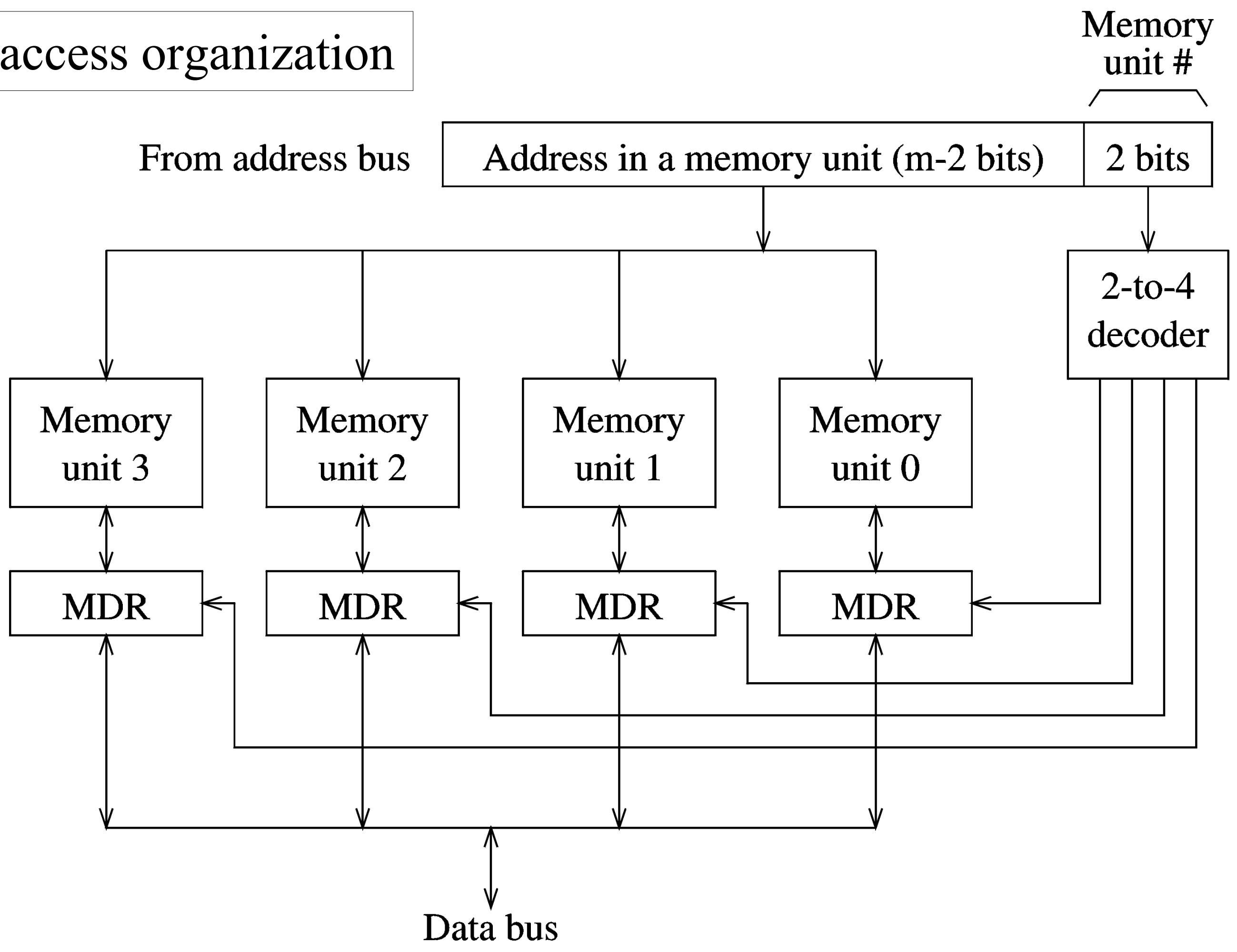
(b) Interleaved memory address mapping

Interleaved Memory (cont'd)

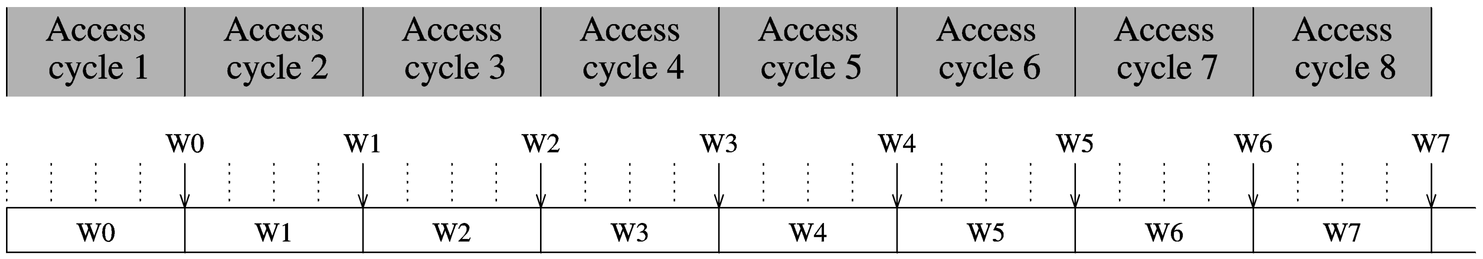
- Two possible implementations
  - Synchronized access organization
    - Upper ***m*** bits are presented to all banks simultaneously
    - Data are latched into output registers (MDR)
    - During the data transfer, next ***m*** bits are presented to initiate the next cycle
  - Independent access organization
    - Synchronized design does not efficiently support access to non-sequential access patterns
    - Allows pipelined access even for arbitrary addresses
    - Each memory bank has a memory address register (MAR)
      - No need for MDR

Interleaved Memory (cont'd)

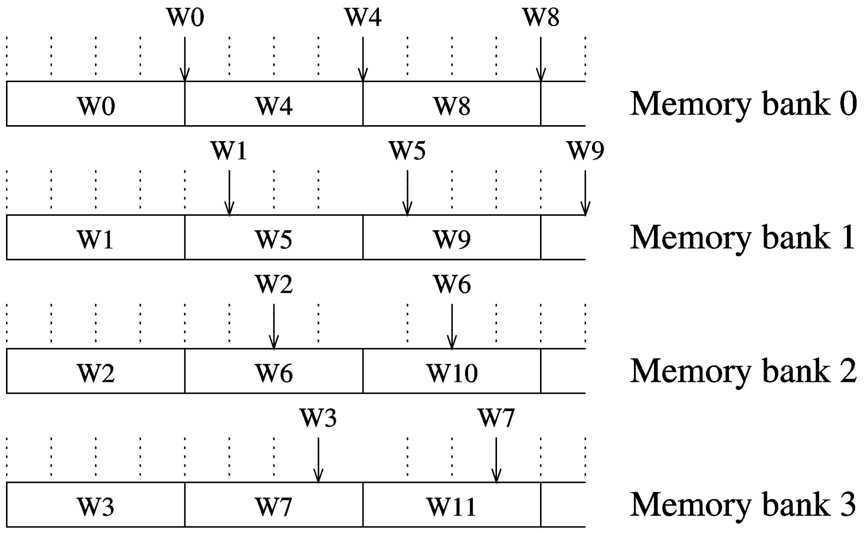
Synchronized access organization



Interleaved Memory (cont'd)



(a) Noninterleaved memory access

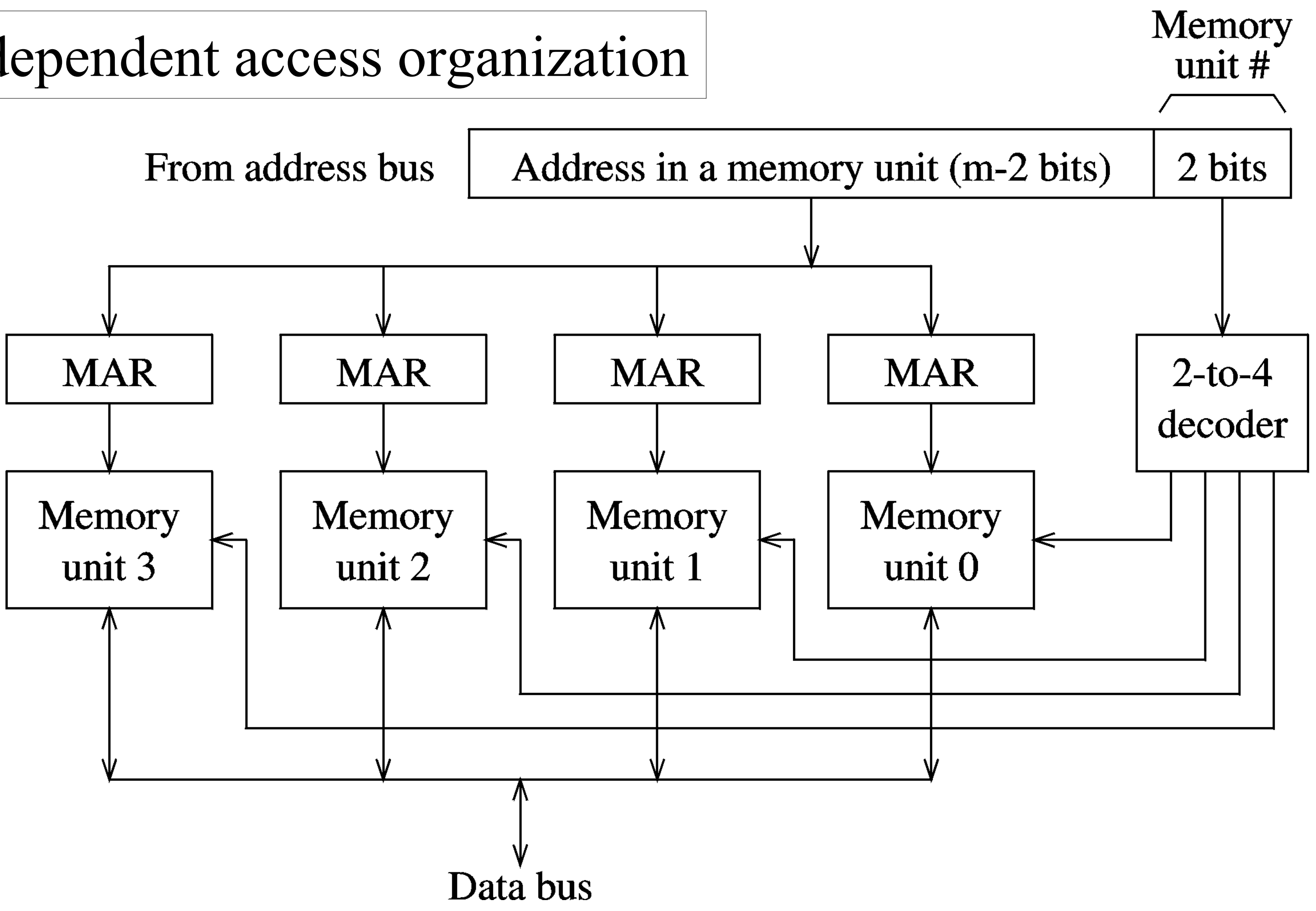


Interleaved memory allows pipelined access to memory

(b) Interleaved memory access

Interleaved Memory (cont'd)

Independent access organization



Interleaved Memory (cont'd)

- Number of banks
  - ∗ ***M*** = memory access time in cycles
  - ∗ To provide one word per cycle
    - » Number of banks ≥ ***M***
- Drawbacks of interleaved memory
  - ∗ Involves complex design
    - » Example: Need MDR or MAR
  - ∗ Reduced fault-tolerance
    - » One bank failure leads to failure of the whole memory
  - ∗ Cannot be expanded incrementally

Last slide