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

(b) Truth table

<table>
<thead>
<tr>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Techniques to Connect to a Bus (cont’d)

Open collector register chip

(a) Connection diagram

(b) Logic symbol

<table>
<thead>
<tr>
<th>WE</th>
<th>WA1</th>
<th>WA0</th>
<th>D inputs to</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Word 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Word 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Word 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Word 3</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>None</td>
</tr>
</tbody>
</table>

(c) Write function table

<table>
<thead>
<tr>
<th>RE</th>
<th>RA1</th>
<th>RA0</th>
<th>Output from</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Word 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Word 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Word 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Word 3</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>None (Z)</td>
</tr>
</tbody>
</table>

(d) Read function table
Techniques to Connect to a Bus (cont’d)

**Tri-State Buffers**

(a) & (c) E = 0

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>DI</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

(b) & (d) E = 1


Techniques to Connect to a Bus (cont’d)

Two example tri-state buffer chips

74367
74368

© S. Dandamudi
Chapter 16: Page 11

Techniques to Connect to a Bus (cont’d)

(a) Connection diagram

(b) Logic symbol

8-bit tri-state register
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

Address lines

- A0
- A1

Write

- WR

Read

- RD

Chip select

- CS

4 X 3 memory

D0

D1

D2

Bidirectional data lines
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

Address bus A0 – A31

A31 A30 A29 A28 A27 A26

A2 – A25

Module A

A0 – A23
16M X 32
CS
D0 – D31

To data bus

A31 A30 A29 A28 A27 A26

A2 – A25

Module B

A0 – A23
16M X 32
CS
D0 – D31

To data bus
Memory Mapping (cont’d)

Partial mapping

Address bus \( A_0 \rightarrow A_{31} \)

Module A

\[
\begin{array}{c}
\text{CS} \\
16M \times 32 \\
D_0 \rightarrow D_{31}
\end{array}
\]

To data bus

Module B

\[
\begin{array}{c}
\text{CS} \\
16M \times 32 \\
D_0 \rightarrow D_{31}
\end{array}
\]

To data bus

\( A_{2} \rightarrow A_{25} \)
Alignment of Data

CPU

D0 – D31

D24 – D31

D16 – D23

D8 – D15

D0 – D7

Data bus

MEMORY

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

© S. Dandamudi

Chapter 16: Page 21

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)

Memory address

\[ n = m + r \text{ bits} \]

\begin{align*}
\text{r bits} & \quad \text{m bits} \\
\text{Memory unit number} & \quad \text{Address in a memory unit}
\end{align*}

(a) Normal memory address mapping

\begin{align*}
\text{m bits} & \quad \text{r bits} \\
\text{Address in a memory unit} & \quad \text{Memory unit number}
\end{align*}

(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

From address bus

Address in a memory unit (m-2 bits)

Memory unit #

2-to-4 decoder

Memory unit 0 → MDR → Memory unit 1 → MDR → Memory unit 2 → MDR → Memory unit 3

Data bus
Interleaved Memory (cont’d)

Interleaved memory allows pipelined access to memory.

(a) Noninterleaved memory access

(b) Interleaved memory access
Interleaved Memory (cont’d)

Independent access organization

From address bus

MAR

Memory unit 3

MAR

Memory unit 2

MAR

Memory unit 1

MAR

Memory unit 0

2-to-4 decoder

Address in a memory unit (m-2 bits)

2 bits

Data bus

Memory unit #
Interleaved Memory (cont’d)

• Number of banks
  * \( M \) = memory access time in cycles
  * To provide one word per cycle
    » Number of banks \( \geq M \)

• 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