UNIVERSITY of WASHINGTON

CSS 422 Hardware and Computer Organization

Bus organization and memory design

Professor: Yang Peng

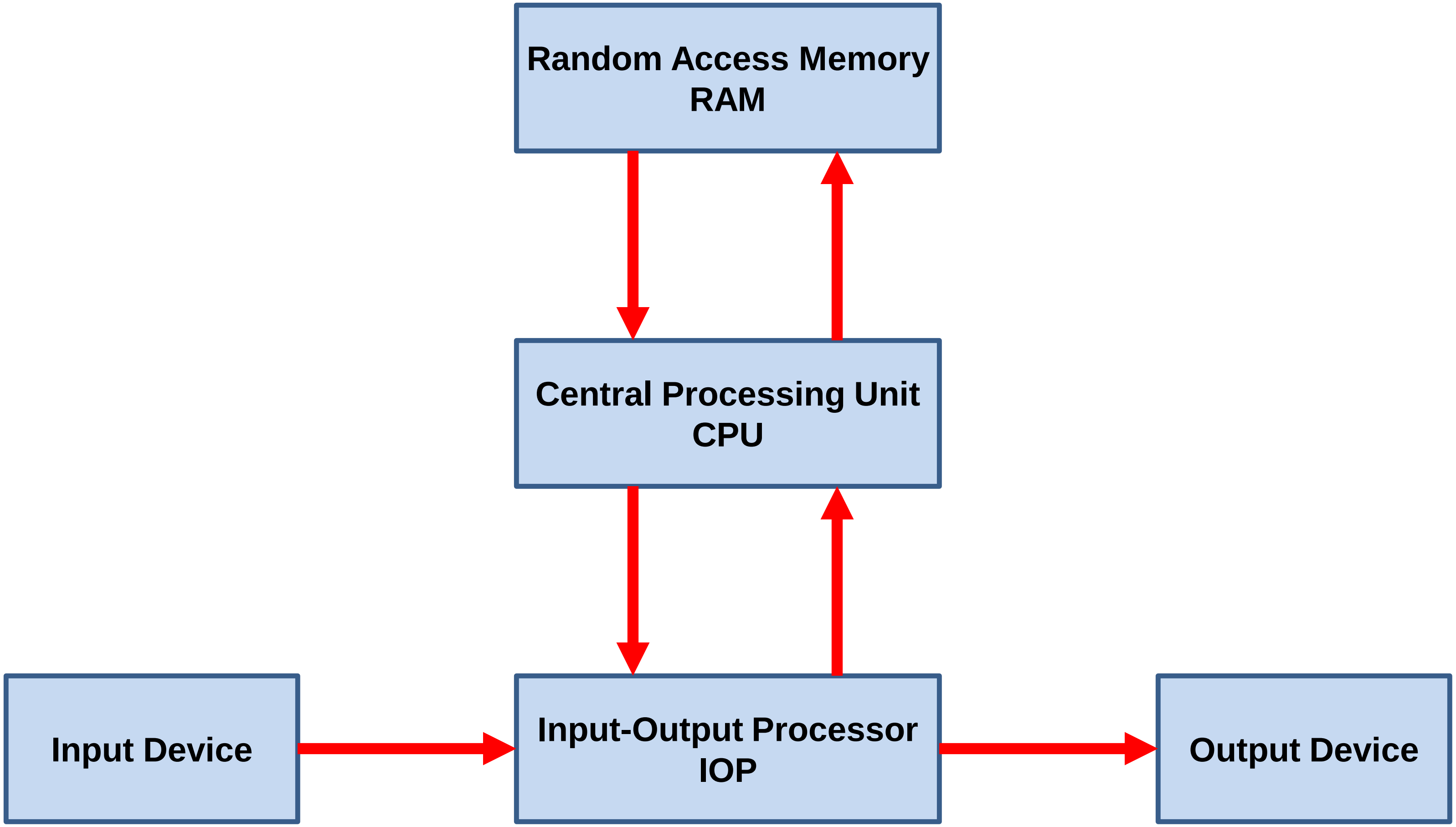
The slides are re-produced by the courtesy of
Dr. Arnie Berger and Dr. Wooyoung Kim

Topic

- Bus organization and memory design
  - Chapter 6 by Berger
  - Chapter 4 by Null

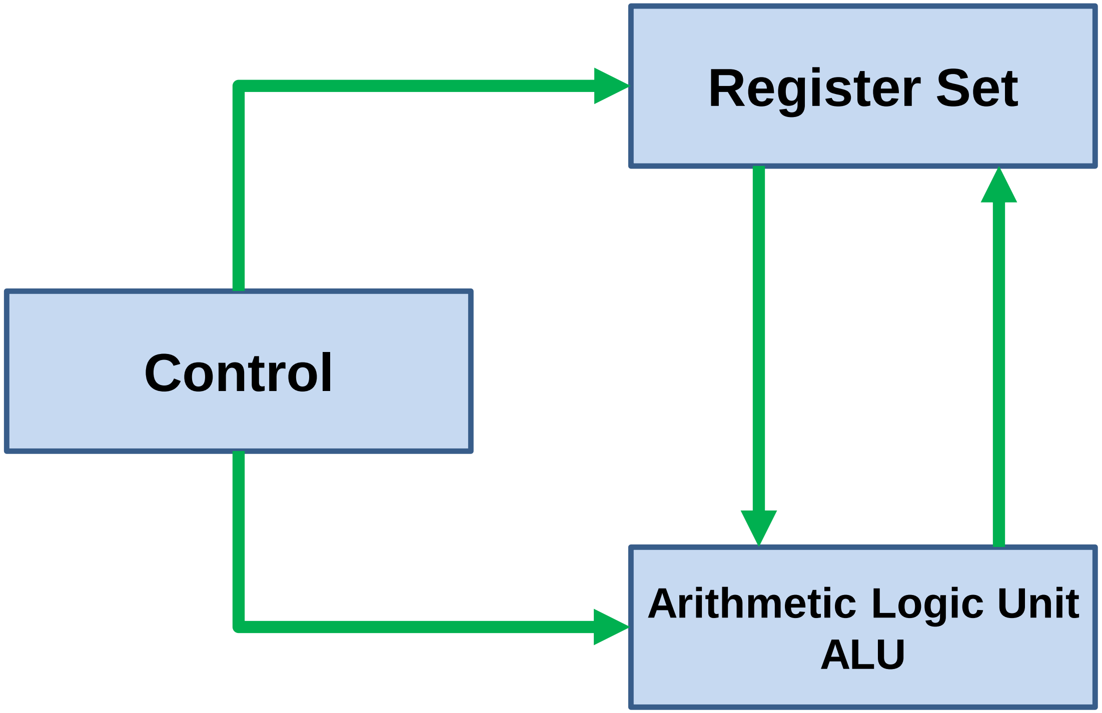
Diagram of a Digital Computer

- The components are connected with ***external*** buses

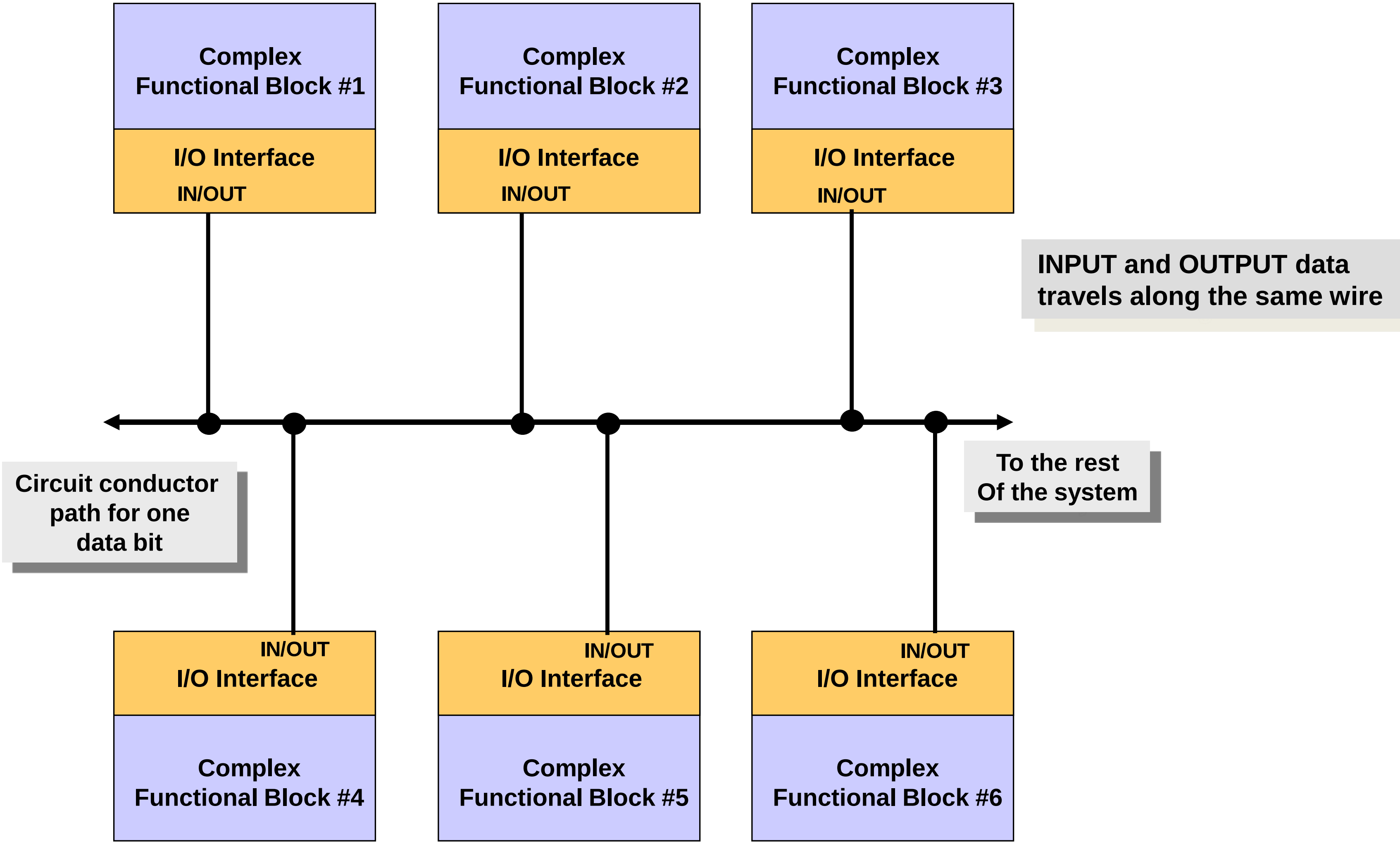


CPU

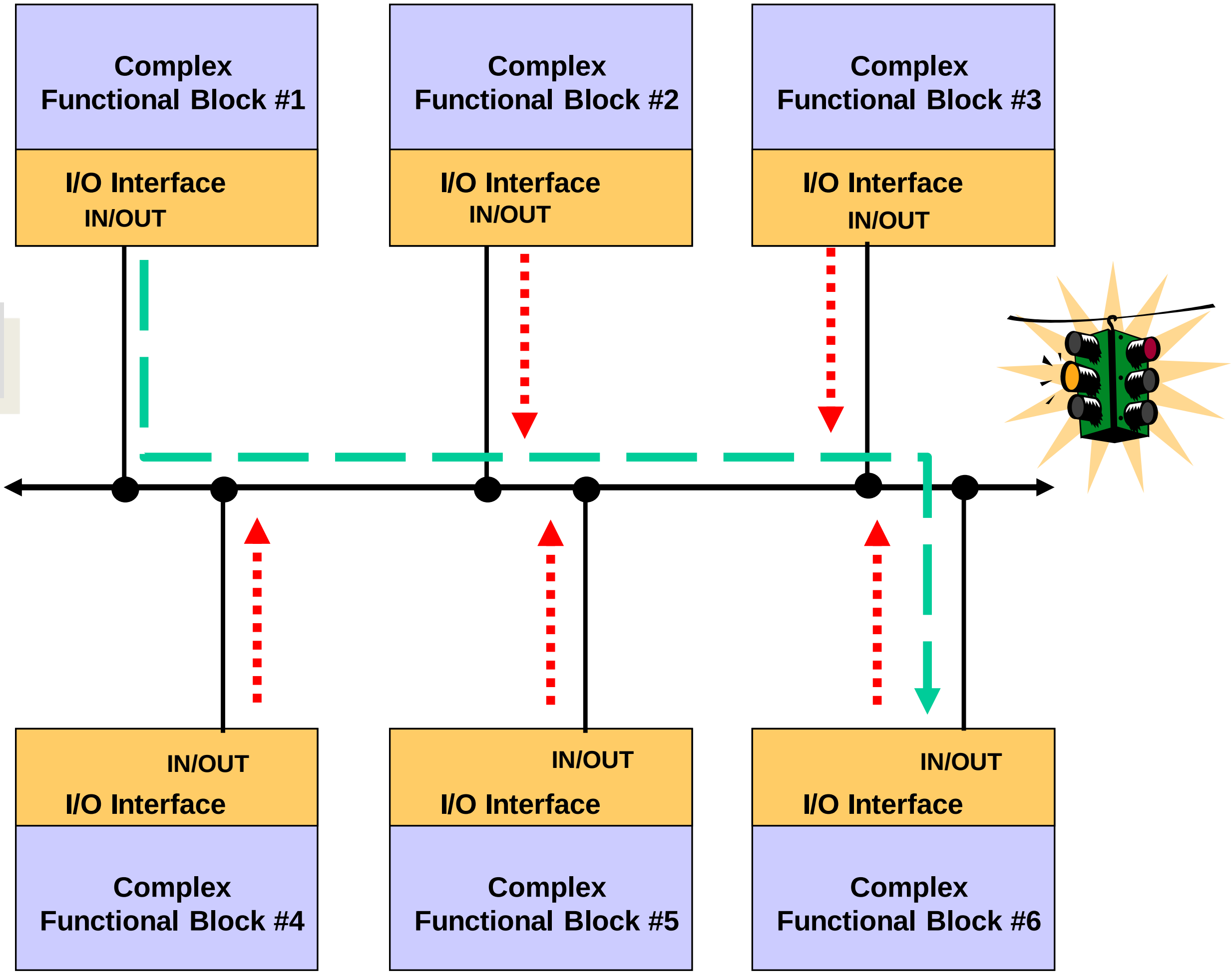
- The components in CPU is connected with ***internal*** buses



Bus Organization



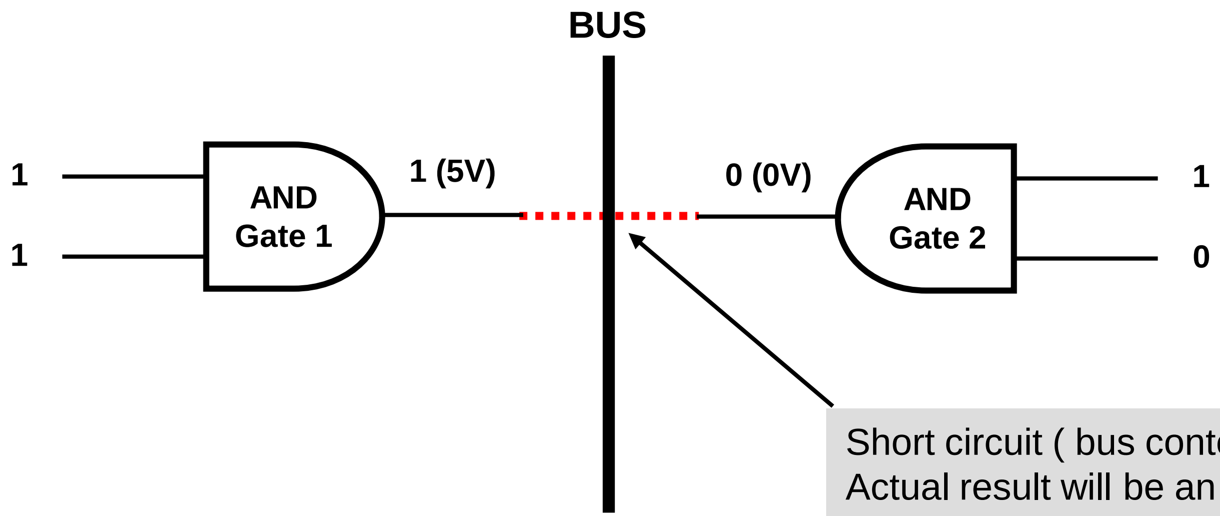
Bus Organization



How do we manage the flow of information?

Bus Organization

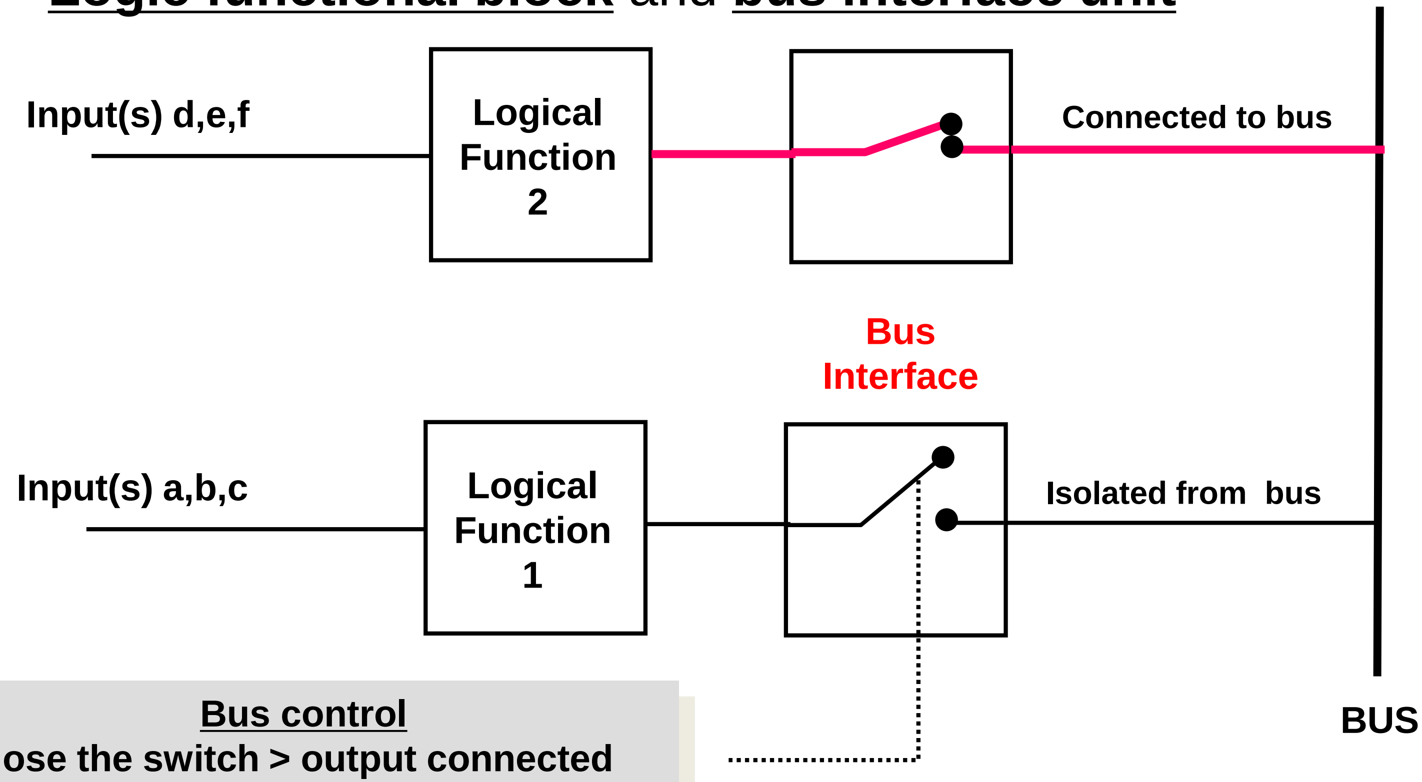
- Busses were invented in order to **simplify the organization and flow** of data within computer systems
  - Busses allow many devices to connect to the same data path
    - Allow for efficient exchange of data between devices
- Question: How do I ***connect outputs together and not get a short circuit***?



Short circuit (bus contention)
Actual result will be an indeterminate logic level

Bus Organization (2)

- Answer: All logic devices that connect to a bus are actually **divided into two parts**
  - **Logic functional block** and **bus interface unit**

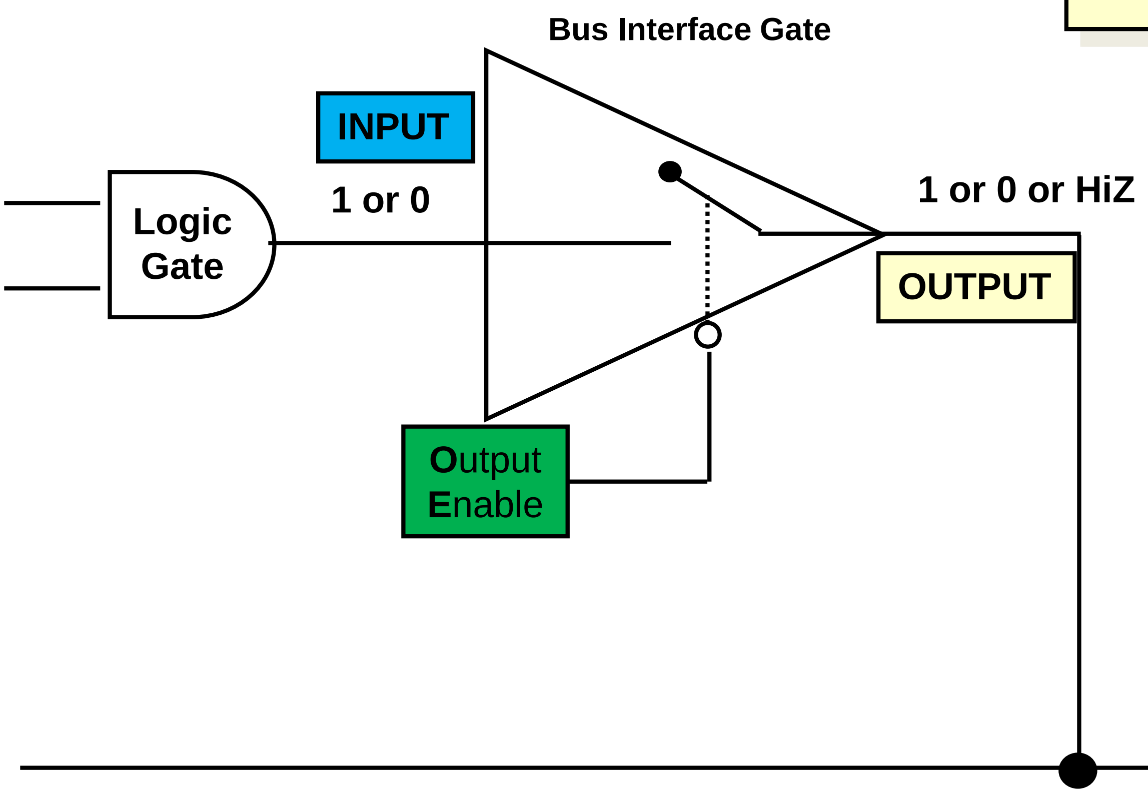


Bus control
- **Close the switch > output connected**
- **Open the switch > output disconnected**

Bus Interface Structure: Tristate

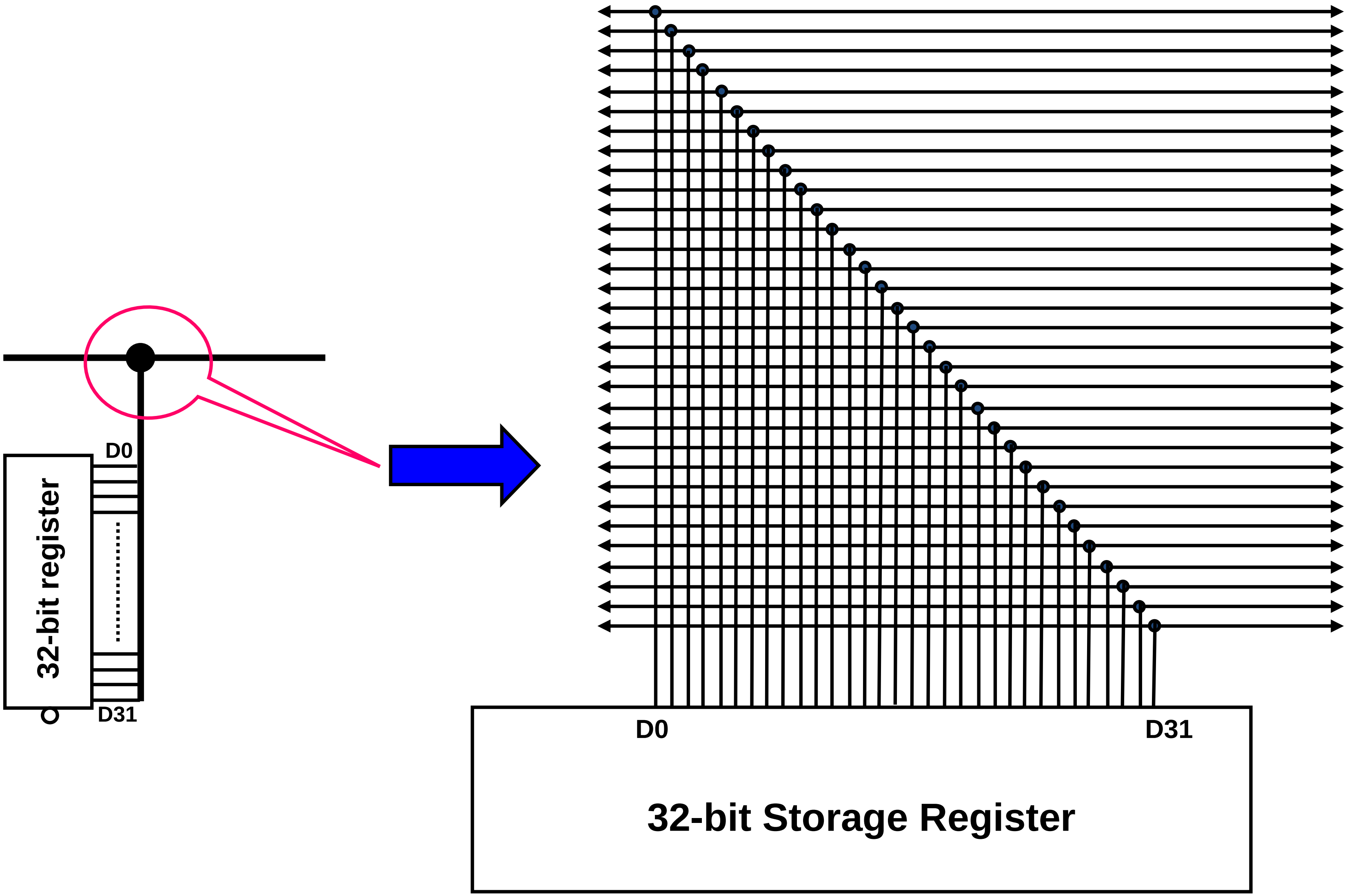
Truth table for Bus Interface Gate →

Output Enable	INPUT	OUTPUT
0	1	1
0	0	0
1	1	Hi Z
1	0	Hi Z



Hi Z: removing the gate from the circuit (No signal to the output)

Bus Organization

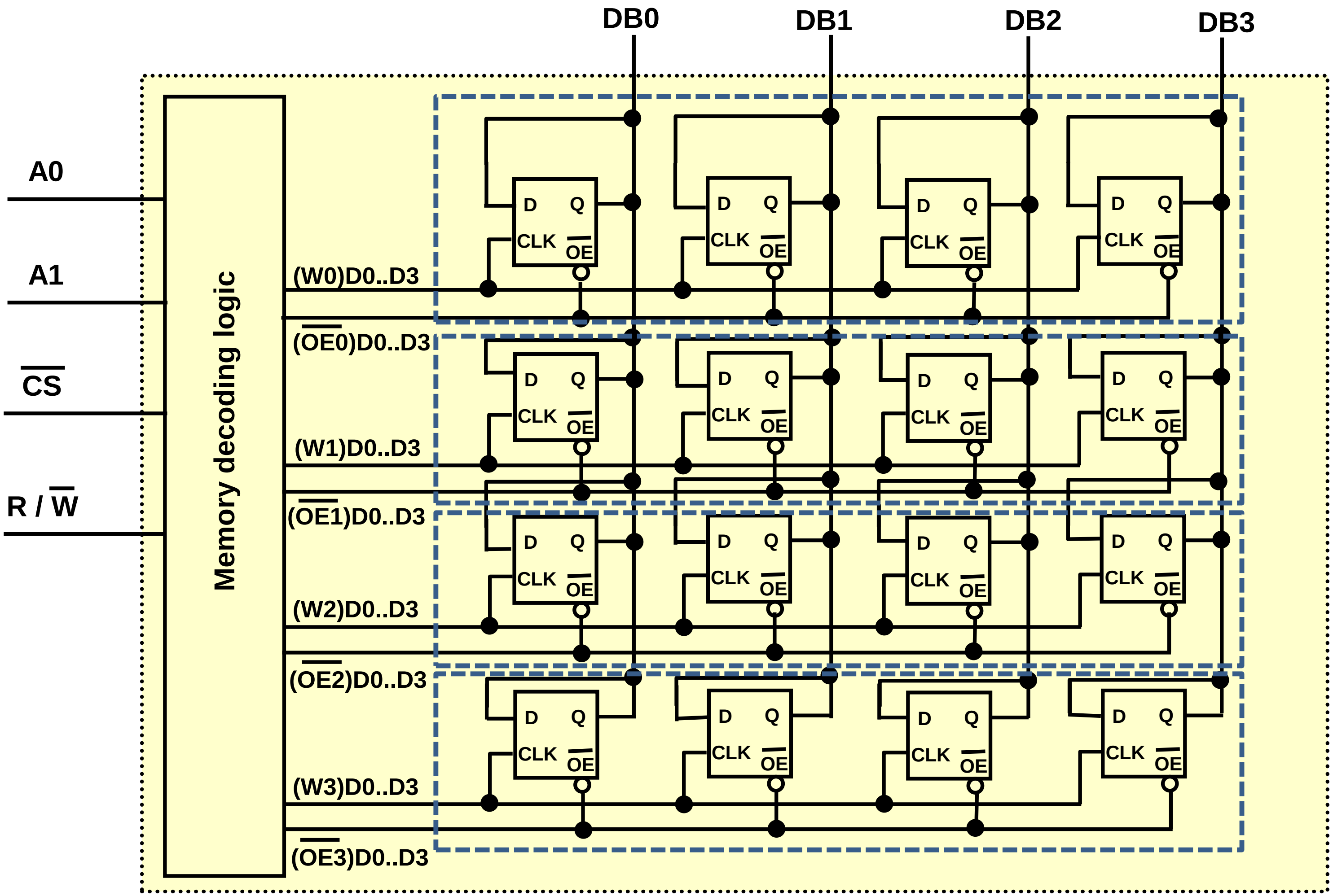


Data Path (Input/Output) Width

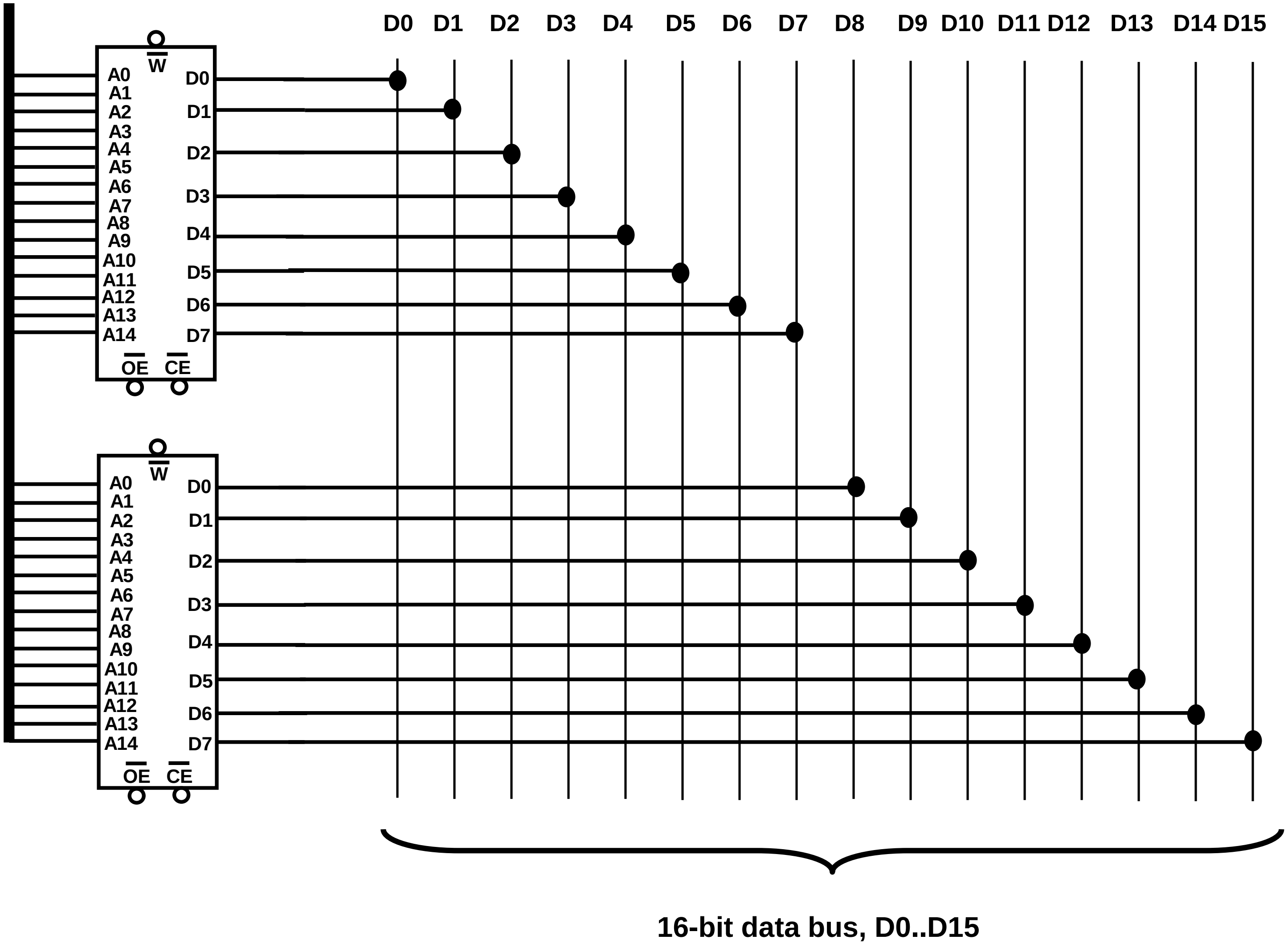
- Summary of where the various bus widths are most common
  - 4/8 bits - Appliances, modems, simple applications
  - 16 bits - Industrial controllers, automotive
  - 32 bits - Telecomm, laser printers, high-performance apps
  - 64 bits - PC’s, workstations
  - 128/256 bits - Next generation “computing” devices

- Internal and external data paths may differ in size
  - Narrower memory is more economical
  - MC68000: 32-bit internal/16-bit external
  - 80C188: 16-bit internal/8-bit external

Memory Organization



Expanding Memory by Width



Byte Packing – Big Endian

When the size of the data element (byte) is smaller than the width of the available memory, we use byte packing to save space.

- **Big Endian –** The **most significant byte** (the "big end") of the data is **placed at** the byte with **the lowest address**. The rest of the data is placed in order in the next three bytes in memory

000000	Byte 3	Byte 2	Byte 1	Byte 0
000004	Byte 3	Byte 2	Byte 1	Byte 0
000008	Byte 3	Byte 2	Byte 1	Byte 0
00000C	Byte 3	Byte 2	Byte 1	Byte 0
000010	Byte 3	Byte 2	Byte 1	Byte 0

FFFFF0	Byte 3	Byte 2	Byte 1	Byte 0
FFFFF4	Byte 3	Byte 2	Byte 1	Byte 0
FFFFF8	Byte 3	Byte 2	Byte 1	Byte 0
FFFFFC	Byte 3	Byte 2	Byte 1	Byte 0

Byte Packing – Little Endian

When the size of the data element (byte) is smaller than the width of the available memory, we use byte packing to save space.

- **Little Endian –** The **least significant** **byte** (the "little end") of the data is **placed at** the byte with **the lowest address**. The rest of the data is placed in order in the next three bytes in memory.

000000	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
000004	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
000008	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
00000C	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
000010	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**

⋮

FFFFF0	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
FFFFF4	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
FFFFF8	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**
FFFFFC	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**

Class Exercise

Given data 0xFF00AA11, copy the bytes to memory address $4000 using big endian and little endian orders.

Little Endian	
Address	Contents
4003	
4002	
4001	
4000	

Big Endian	
Address	Contents
4003	
4002	
4001	
4000	

Class Exercise

Given data 0xFF00AA11, copy the bytes to memory address $4000 using big endian and little endian orders.

Little Endian	
Address	Contents
4003	FF
4002	00
4001	AA
4000	11

Big Endian	
Address	Contents
4003	11
4002	AA
4001	00
4000	FF

Why we need to know the endian system?

- Intel 80x86: little-endian
- Macintosh: big-endian

- As long as the electronics is consistent, either byte order works within a computer system
  - Usually you don't need to think about which order is used

- However, problem will occur when data is transferred between systems using different endians
  - Which byte ordering (little or big endian) was used?
  - How to convert from big- to little-endian or vice versa?