# COMP2121: Microprocessors and Interfacing

## Computer Buses and Parallel Input/Output

http://www.cse.unsw.edu.au/~cs2121

Lecturer: Hui Wu

Term 2, 2019

1

1

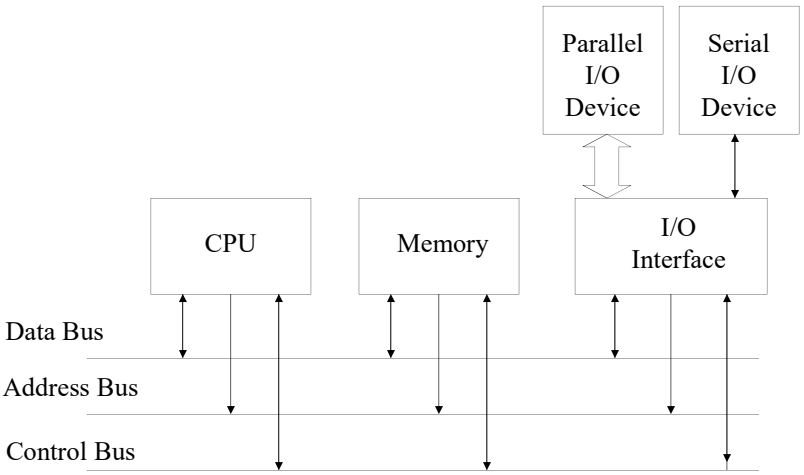
## Overview

- Buses
- Memory mapped I/O and Separate I/O
- I/O Synchronization
- Bus arbitration

2

2

# Bus Oriented Architecture




3

3

# Computer Buses (1/3)

- CPU is connected to memory and I/O devices via data, address and control buses.
- Data bus is bi-directional and transfers information (memory data and instruction, I/O data) to and from CPU.
- Address bus may be bi-directional (with more than one source of information) but is most often unidirectional because CPU is the only source of the addresses.
- Control bus carries all other signals required to control the operation of the system.

4

4

# Computer Buses (2/3)

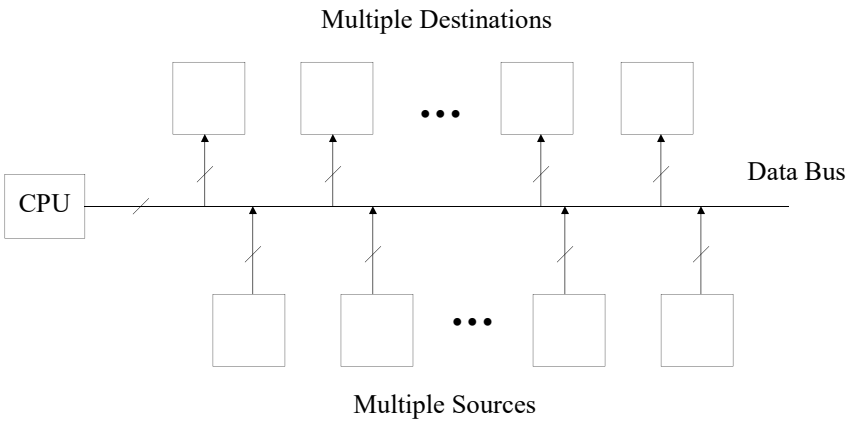
- Component level bus is defined by the signals on the microprocessor chip, such as READ/WRITE.
  - Component level signals are different for different manufacturers and used when designing single board computers or dedicated application systems.
- System level bus is defined by more generic signals such as MEMRD and IORD.
  - Often designed for use as a backplane into which printed circuit boards are plugged.
- Intersystem bus is used to connect different systems.

5

5

# Computer Buses (3/3)

- Each line of a bus may have multiple sources and destinations.




6

6

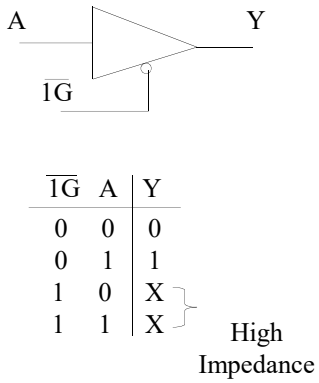
# Information Sources – The Input Interface

- The input interface provides three-state buffers between the source and the data bus.
  - For example, a parallel, eight-bit input interface can be constructed with eight three-state gates whose enable lines are tied together.
  - The open-collector gate is often used for control signal such as request for interrupts.

7

7

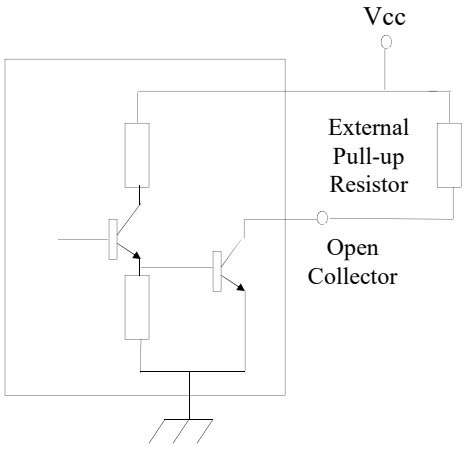
# Typical Bus Interface Gates




| $\overline{1G}$ | A | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | X |
| 1 | 1 | X |

High Impedance

(a) Three-state gate



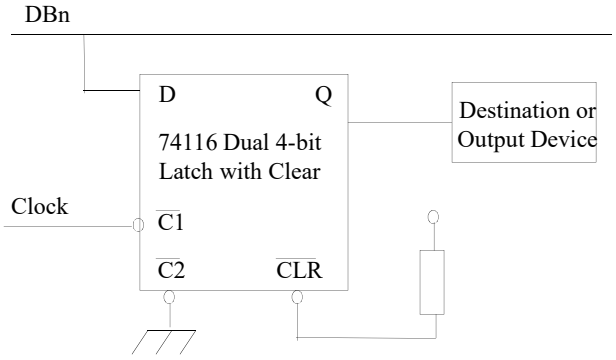

(b) Typical open-collector gate

8

8

# Information Destinations – The Output Interface

- The output interface between the data bus and a destination or output device is a latch.




9

9

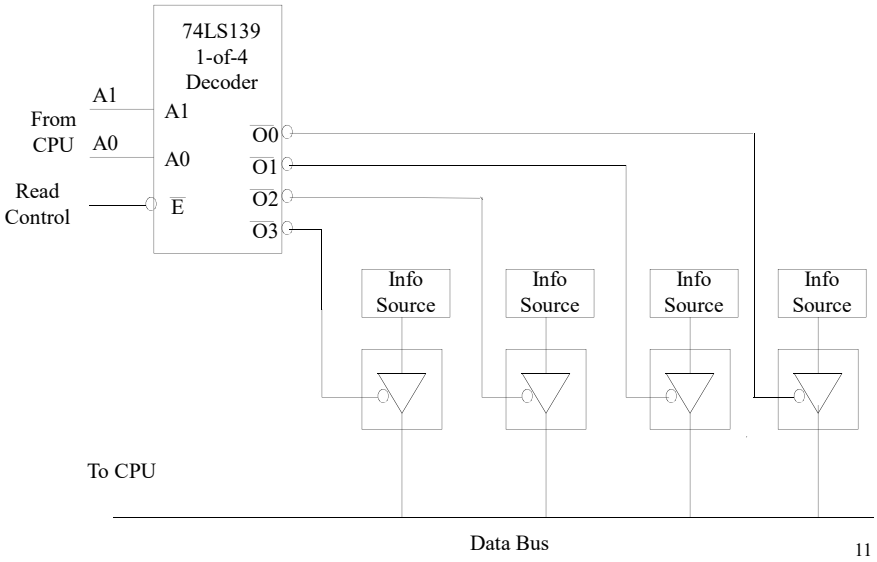
# Address Decoding

- The interface must provides the ability for CPU to select one of many sources and destinations.
  - Addressing and address decoding can select one out of many sources and destinations.

10

10

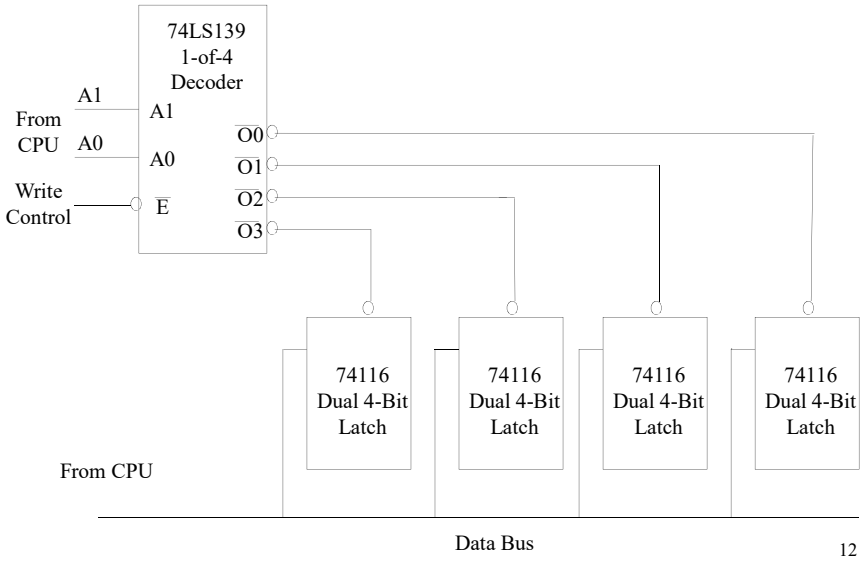
# Address Decoding for Input Devices




11

11

# Address Decoding for Output Devices




12

12

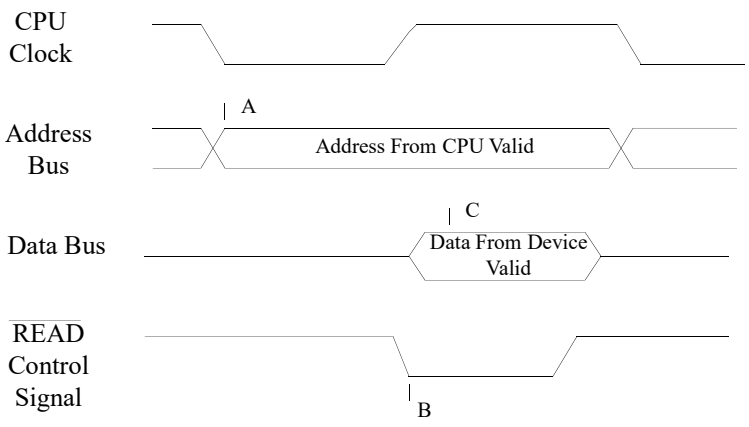
# CPU Timing Signals

- CPU must provide timing and synchronization so that the transfer of information occurs at the right time.
  - CPU has its own clock.
  - I/O devices may have a separate I/O clock.
  - Typical timing signals include **READ** and **WRITE**.

13

13

# Typical CPU Read Cycle (1/2)




14

14

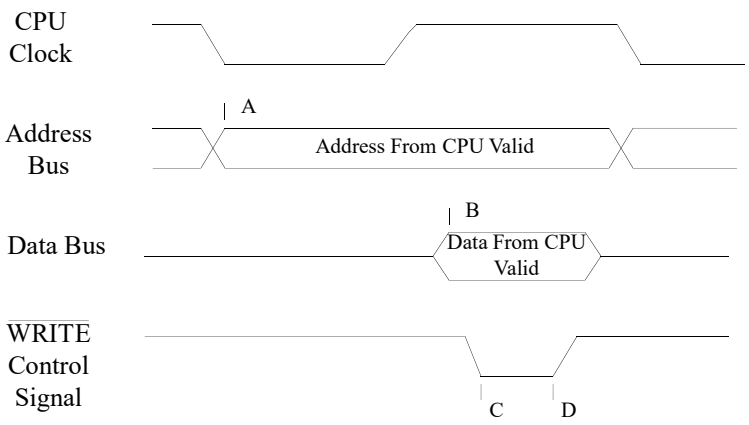
# Typical CPU Read Cycle (2/2)

- CPU places the address on the address bus at point A.
- The control signal **$\overline{\text{READ}}$** is asserted at point B to signal the external device that CPU is ready to take the data from the data bus.
- CPU reads the data bus at point C whether or not the input device has put it ready
  - ❑ If NOT, some form of synchronization is required.

15

15

# Typical CPU Write Cycle (1/2)




16

16

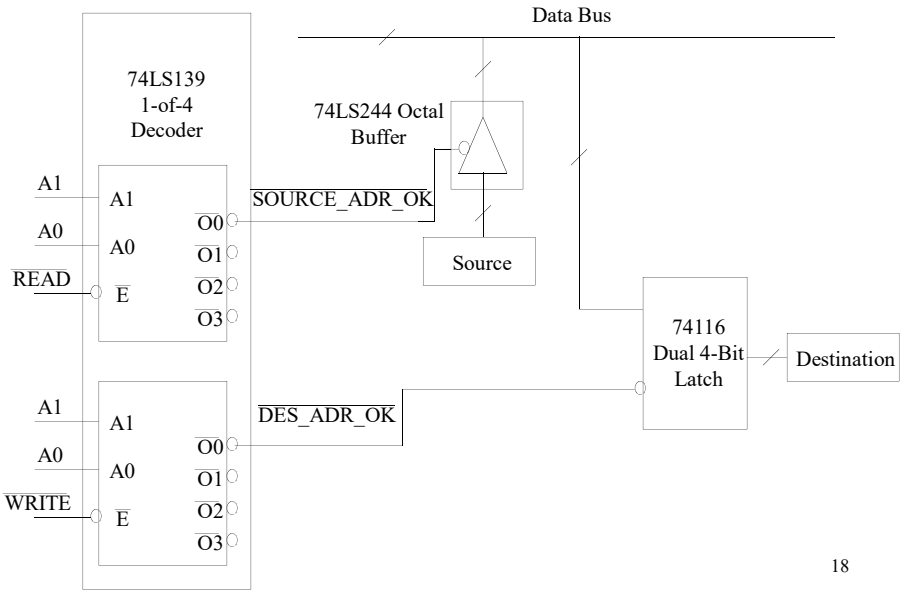
# Typical CPU Write Cycle (2/2)

- CPU places the address on the address bus at point A.
- The data bits are supplied by CPU at point B.
- The control signal **WRITE** is asserted by CPU at point C.
  - This signal is used to create the clock to latch the data at the correct time.
- Depending on the type of latch and when **WRITE** is asserted, the data may be captured on the falling edge or rising edge.

17

17

# Complete I/O Interface (1/2)




18

18

## Complete I/O Interface (2/2)

- **$\overline{\text{READ}}$** and **$\overline{\text{WRITE}}$** control the enable (**$\overline{\text{E}}$**).
- Three state enables and the latch clock signals are not asserted until the correct address is on the address bus AND the correct time in the read or write cycle has arrived.

19

19

## I/O Addressing

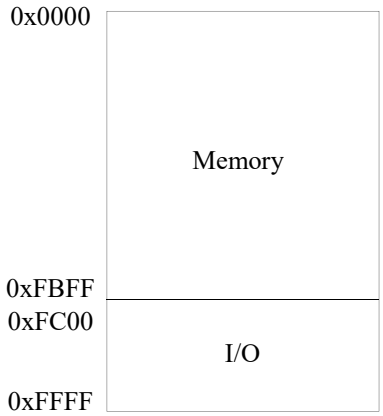
- If the same address bus is used for both memory and I/O, how does hardware distinguish between memory reads and writes and I/O reads and writes?
  - Two approaches:
    - Memory-mapped I/O.
    - Separate I/O.
  - AVR supports both.

20

20

# Memory Mapped I/O (1/2)

- The entire memory space is divided into memory space and I/O space.




21

21

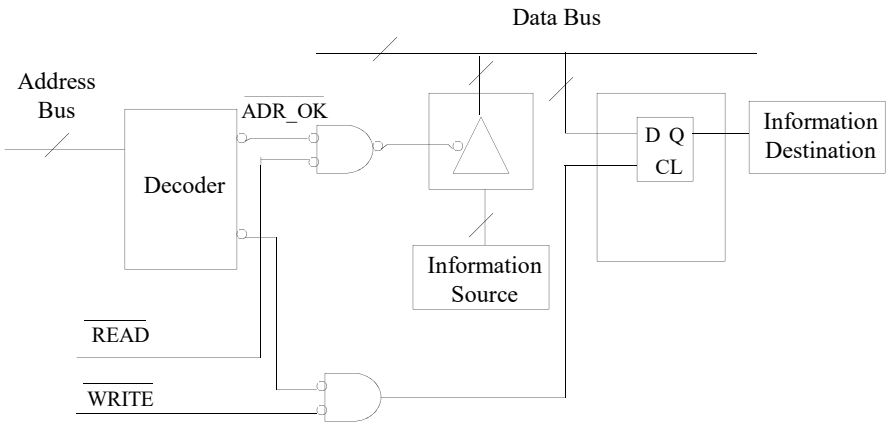
# Memory Mapped I/O (2/2)

- Advantages:
  - Simpler CPU design.
  - No special instructions for I/O accesses.
- Disadvantages:
  - I/O devices reduce the amount of memory available for application programs.
  - The address decoder needs to decode the full address bus to avoid conflict with memory addresses .

22

22

# I/O Interface for Memory-Mapped I/O




23

23

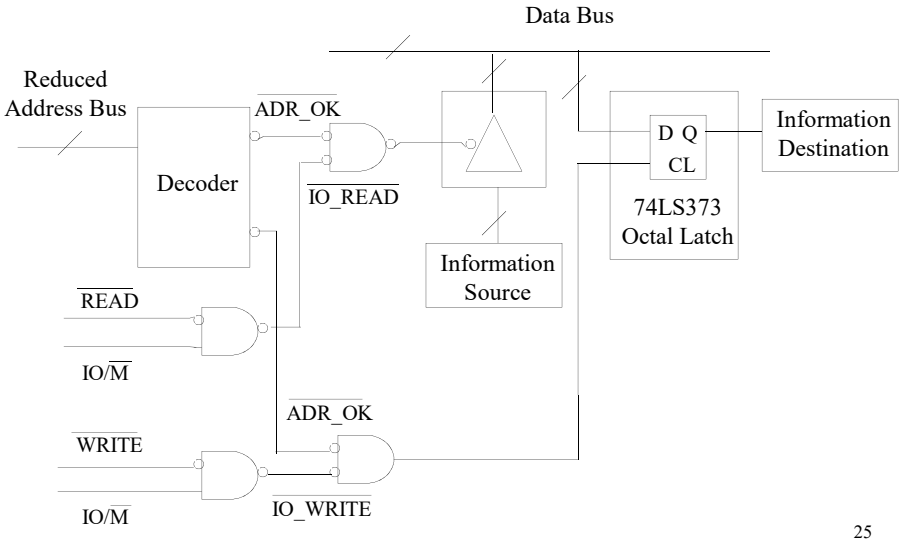
# Separate I/O

- Two separate spaces for memory and I/O.
  - Less expensive address decoders than those needed for memory-mapped I/O.
- Additional control signal, called IO/$\overline{M}$, is required to prevent both memory and I/O trying to place data on the bus simultaneously.
  - IO/$\overline{M}$ is high for I/O use and low for memory use.
- Special I/O instructions such as in and out are required.

24

24

# I/O Interface for Separate I/O




25

25

# I/O Synchronization

- CPU is typically much faster than I/O devices.
- I/O devices need to transfer data at unpredictable intervals.

Therefore, synchronization between CPU and I/O devices is required.

Two synchronization approaches:

- Software synchronization.
- Hardware synchronization.

26

26

## Software Synchronization

Two software synchronization approaches:

- Real-time synchronization.
  - ❑ Uses a software delay to match CPU to the timing requirements of the I/O device.
    - ❖ Sensitive to CPU clock frequency.
    - ❖ Wastes CPU time.
- Polled I/O.
  - ❑ A status register, with a **DATA_READY** bit, is added to the device. The software keeps reading the status register until the **DATA_READY** bit is set.
    - ❖ Not sensitive to CPU clock frequency.
    - ❖ Still wastes CPU time, but CPU can do other tasks.

27

27

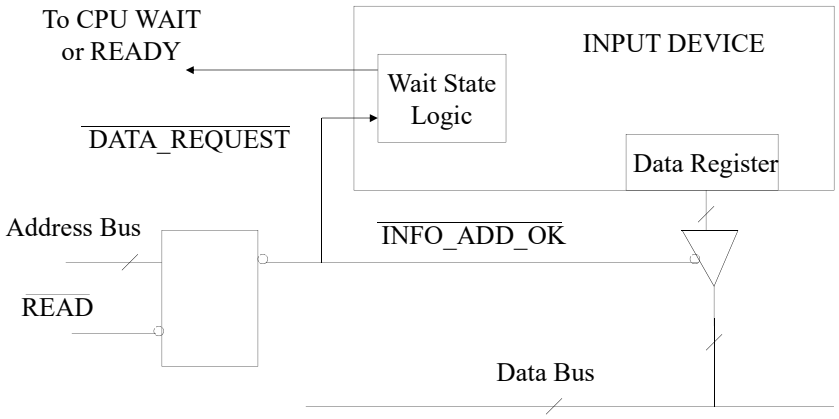
## Handshaking I/O

- This hardware synchronization approach needs a control signal **READY** or **WAIT**.
  - ❑ For an input device, when CPU is asking for input data, the input device will assert **WAIT** if the input data is NOT available. When the input data is available, it will deassert **WAIT**. While **WAIT** is asserted, CPU must wait until this control signal is deasserted.
  - ❑ For an output device, when CPU is sending output data via the data bus, the output device will assert **WAIT** if it is not ready to take the data. When it is ready, it will deassert **WAIT**. While **WAIT** is asserted, CPU must wait until this control signal is deasserted.

28

28

# Input Handshaking Hardware




29

29

# Bus Masters and Slaves

- A bus master is either a CPU or a hardware component (DMA Controller for instance) that controls the buses.
- A bus slave is a device that takes its orders from the bus master.
- What if two or more bus masters want to control the buses simultaneously?
  - Bus arbitration is required.

30

30

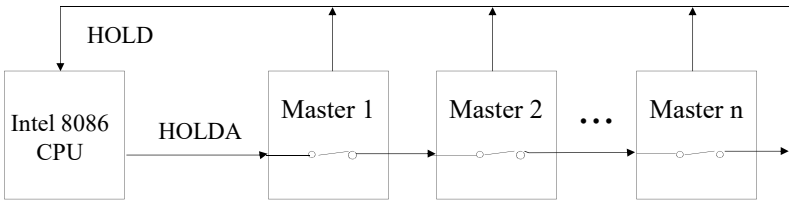
# Bus Arbitration

- Daisy chain bus arbitration.
- Hardware priority bus arbitration.

31

31

# Daisy Chain Bus Arbitration (1/2)




32

32

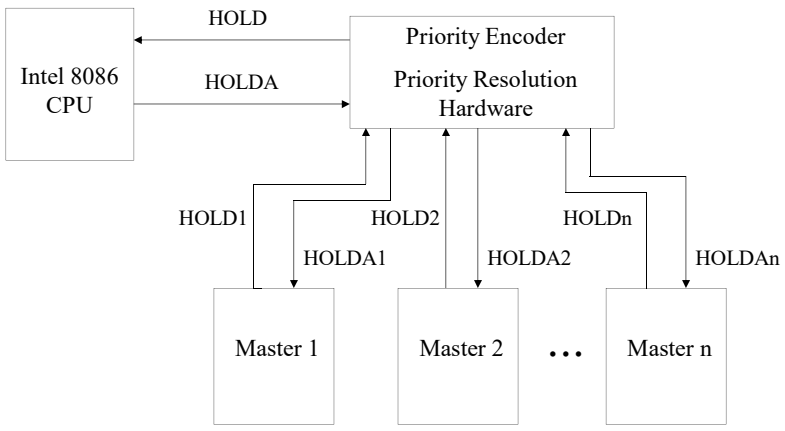
# Daisy Chain Bus Arbitration (2/2)

- Whenever a device wants to control the bus, it asserts HOLD and opens the switch in the HOLDA (HOLD_ACKNOLEDGE) line.
- When HOLDA is asserted by the CPU, it passes through each of the inactive one.
- If a bus master farther right on the chain asserts HOLD before another master is finished, HOLDA is not passed along until the higher priority device (closer to CPU) is finished with its tasks and closes its switch.

33

33

# Hardware Priority Bus Arbitration




34

34

## Hardware Priority Bus Arbitration

- Each device is pre-assigned a priority.
- Simultaneous HOLD signals are encoded so that only the highest priority device receives HOLDA.

35

35

## Reading

1. Chapter 7: Computer Buses and Parallel Input and Output. Microcontrollers and Microcomputers by Fredrick M. Cady.

36

36