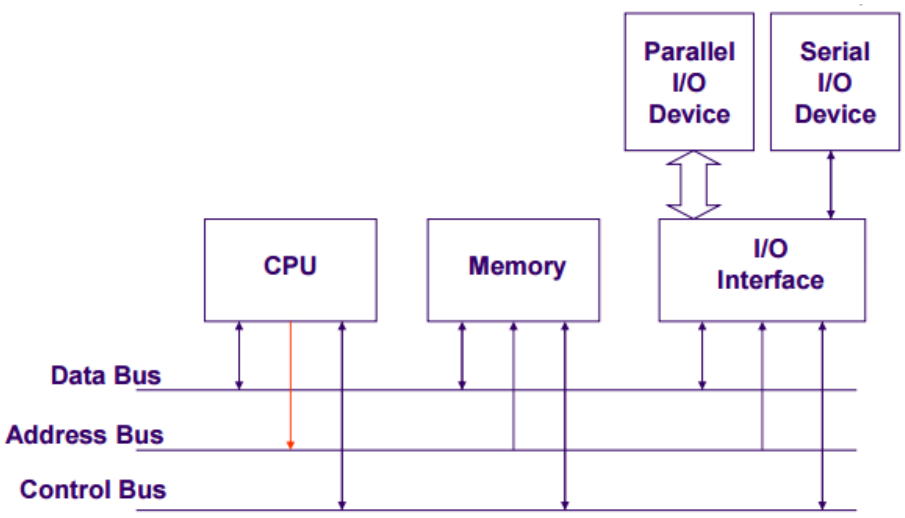
3. Buses and Parallel Input/Output

COMP2121 • KC Notes

3.1 Buses

- **Buses**: a parallel, bidirectional and binary information pathway between multiple sources and multiple destinations
    - **Data bus** – transfers the actual data
    - **Address bus** – transfers information about where the data should go
    - **Control bus** – transfers control signals
- **Characteristics of buses**
    - **Bus width** – amount of data that can be transmitted at a time, e.g. 16 bits, 32 bits
    - **Clock speed (MHz)** – how often data can be transferred on the buses

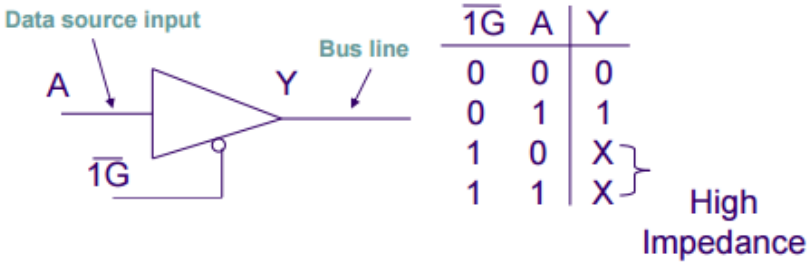
3.2 Computer buses



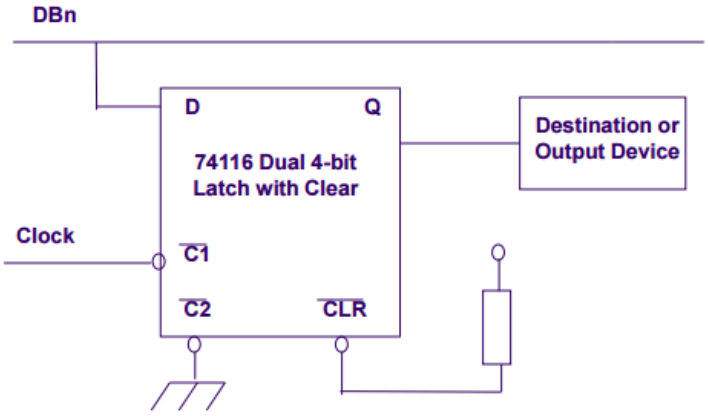
- **Computer buses**: connected to memory and I/O devices through all three buses
    - **Data bus** is bi-directional – transfers information (data, instructions) to and from the CPU
    - **Address bus** is usually unidirectional – CPU is usually the **source of addresses**
    - **Control bus** carries control signals for data transfer operations
- Each line has **multiple sources and destinations**

3.3 Input and output interface

- **Input interface** is needed that **connects multiple data sources** but only one source can drive a signal to the bus at a time
    - **Open-collector gates**: when outputs of several
    - Implemented as **three-state buffers for data buses**
        - Parallel eight bit input data is connected to eight three-state gates
        - The eight 1G lines are all connected to each other
        - When data is sent to bus, the eight three-state gates are enabled (1G reads 1)
        - The 1G signal is checked to be equal to 0 to activate output



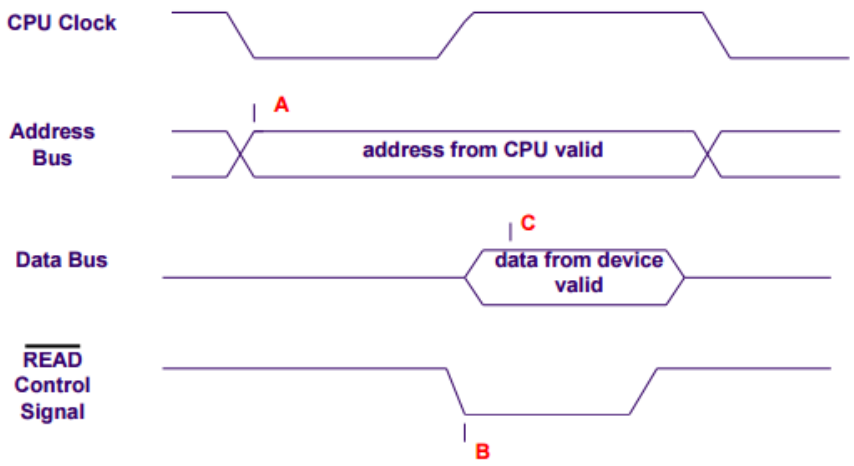
    - **Open-collector gate**: used for control signals, e.g. request for interrupts
        - Are LOW or float – so all gates are low unless changed by an external pull-up resistor
- **Output interface**: between the data bus and the output device contains a latch
    - Control signals are generated from the sequence controller, e.g. a clock



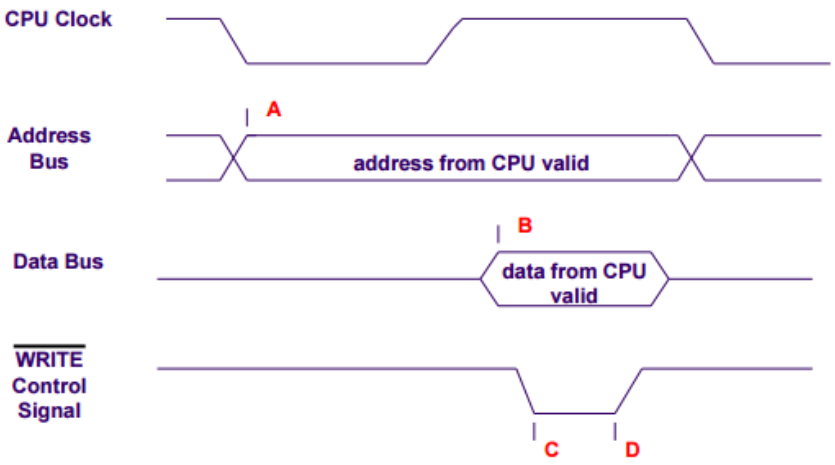
- The interface must let CPU **select one source and one destination**, and this is done using **the address decoder**
    - CPU provides timing and synchronisation (has its own clock, I/O devices may have a separate clock)
    - Data is only taken from or placed onto the bus at the correct time

3.4 Read and Write Cycles

- **Read cycle**: data transfer from an external source to the CPU
  - At A: **address placed on the address bus** by CPU
  - At B: **READ signal set to 0** to signal input device to know CPU is ready for data
  - At C: CPU **takes data from data bus** – if device not ready, synchronisation required



- **Write cycle**: data transfer from a register to an output data latch
  - At A: **address placed on the address bus** by CPU
  - At B: **data is placed by the CPU onto the data bus** (from C in READ)
  - At C: **WRITE signal set to 0** to signal the external device that data is ready
  - At D: **WRITE** signal stays asserted enough to let data be latched – captured either in falling or rising edge



- Three-state enable and latch clock signals are not asserted (set to 0) until **correct address is on the bus and correct time in read/write cycle** has arrived

3.5 I/O Addressing

- Same address bus is used by **memory and I/O devices** and this needs to be distinguished
    - AVR supports both memory-mapped and separate I/O
- **Memory-Mapped I/O**: any **instructions that read/write memory also read/writes I/O**
    - Address space is **split into memory and I/O address spaces**
        - AVR: 480 external I/O registers are mapped into memory space
        - Accesses to I/O registers use memory access (e.g. lds, sts)
    - Advantages: simpler CPU design, no special instructions
    - Disadvantages: I/O devices reduce application memory space, full address needs to be decoded to avoid confusion between addresses
- **Separate I/O**: **separate I/O instructions** for read/write
    - Separated maps, meaning **I/O map is much smaller** than the memory map
        - Results in **fewer I/O address bits** and **cheaper address decoders**
        - **IO/M**: additional control signal to prevent both memory and I/O from simultaneously placing data onto the bus
        - AVR: first 64 I/O registers with specific instructions (in, out)

3.6 I/O Synchronisation

- Need to synchronise I/O and CPU, because **CPU faster than I/O devices**
- **Real-time synchronisation**: **software delay**, e.g. a loop, to make the CPU wait for the I/O device
    - Timing must be known, sensitive to CPU clock frequency, wastes CPU time
- **Polled I/O**: **status register** with a DATA_READY bit
    - Software can read the status register and keep reading until it is set
    - Not sensitive to CPU clock frequency, wastes CPU time but it can do other tasks
- **Handshaking I/O**: **hardware method** with READY and WAIT
    - Input device will assert **WAIT** if **input data is not available**
    - Output device will assert **WAIT** if **not ready to take data**
    - Switches between WAIT and READY

3.7 Parallel I/O in AVR

- AVR ports are **configurable to receive or send data** – physical pins, use instructions in/out
- Each pin has three register bits:
    - DDxn: selects the **direction of the pin**: 1 if output, 0 if input
    - PORTxn: the data register
        - if pin is input, **pull-up resistor** can be activated/deactivated
    - PINxn: port input pins