INPUT/OUTPUT ORGANIZATION

Module-2

Second internal portions

Direct Memory Access

Direct Memory Access (contd..)

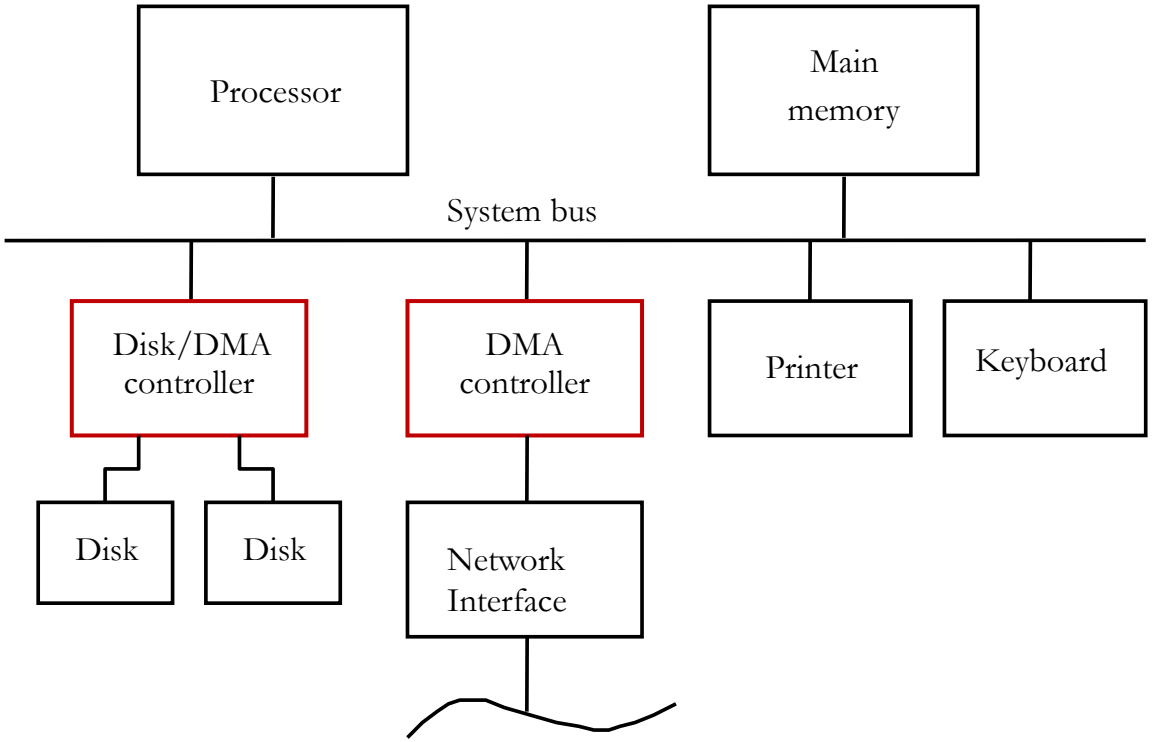
- Direct Memory Access (DMA):
  - A special control unit may be provided to transfer a block of data directly between an I/O device and the main memory, without continuous intervention by the processor.

- Control unit which performs these transfers is a part of the I/O device's interface circuit. This control unit is called as a **DMA controller**.

Direct Memory Access (contd..)

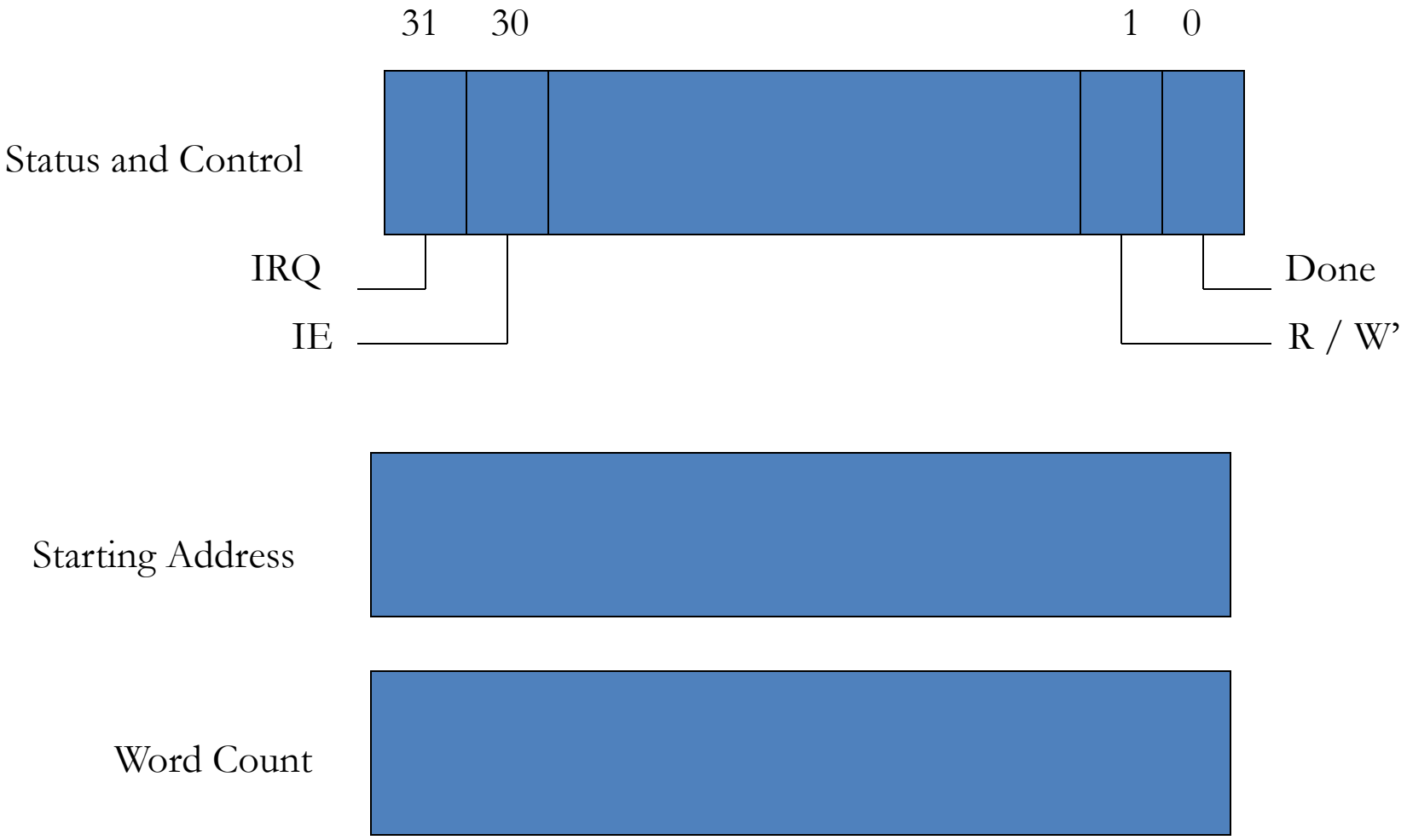
- DMA controller can transfer a block of data from an external device to the processor, without any intervention from the processor.
  - However, the operation of the DMA controller must be under the control of a program executed by the processor. That is, the processor must initiate the DMA transfer.
- To initiate the DMA transfer, the processor informs the DMA controller of:
  - Starting address,
  - Number of words in the block.
  - Direction of transfer (I/O device to the memory, or memory to the I/O device).
- Once the DMA controller completes the DMA transfer, it informs the processor by raising an interrupt signal.

Direct Memory Access



- *DMA controller connects a high-speed network to the computer bus.*
- *Disk controller, which controls two disks also has DMA capability. It provides two DMA channels.*
- *It can perform two independent DMA operations.*

Registers in a DMA Interface



Registers in a DMA Interface

- The DMA controller includes several registers :-
  - The **DMA Address Register** contains the starting address of the memory location to be used in the data transfer.
  - The **DMA Count Register**, also called Word Count Register, contains the no. of bytes of data to be transferred.
  - The **DMA Status Register and Control Register** accepts commands from the CPU.

Registers in a DMA Interface

- **R/W' Determines the direction of transfer .**
- When **R/W' =1, DMA controller read data from memory to I/O device.**
- **R/W' =0, DMA controller perform write operation.**
- **Done Flag=1, the controller has completed transferring a block of data and is** ready to receive another command.
- **IE=1, it causes the controller to raise an interrupt (interrupt Enabled) after it has** completed transferring the block of data.
- **IRQ=1, it indicates that the controller has requested an interrupt.**

Process of DMA Transfer

- To initiate a DMA transfer, the CPU loads the address of the first memory location of the memory block (to be read from or written into) into the DMA address register.

- It then writes the no. of bytes to be transferred into the DMA count register.

- Finally, it writes one or more commands to the DMA control register.

Process of DMA Transfer

- These commands specify transfer options such as the DMA transfer mode, the direction of the transfer- either from I/O to memory or from memory to I/O.
- The last command causes the DMA controller to initiate the transfer.
- When the DMA transfer is completed, the Done flag of Status and Control register is set.
- At the same time the IE bit is set and the DMA controller sends the interrupt request to the processor by setting the IRQ bit.

DMA Transfer Modes

- BURST mode
  - Sometimes called Block Transfer Mode.
  - An entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system buses by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU.
  - This mode is useful for loading programs or data files into memory, but it does render the CPU inactive for relatively long periods of time.

DMA Transfer Modes

- CYCLE STEALING Mode
    - Viable alternative for systems in which the CPU should not be disabled for the length of time needed for Burst transfer modes.
    - DMA controller obtains access to the system bus as in burst mode. However, it transfers one byte of data and, returns control of the system bus to the CPU. It continually issues request for system bus, transferring one byte of data per request, until it has transferred it's entire block of data.

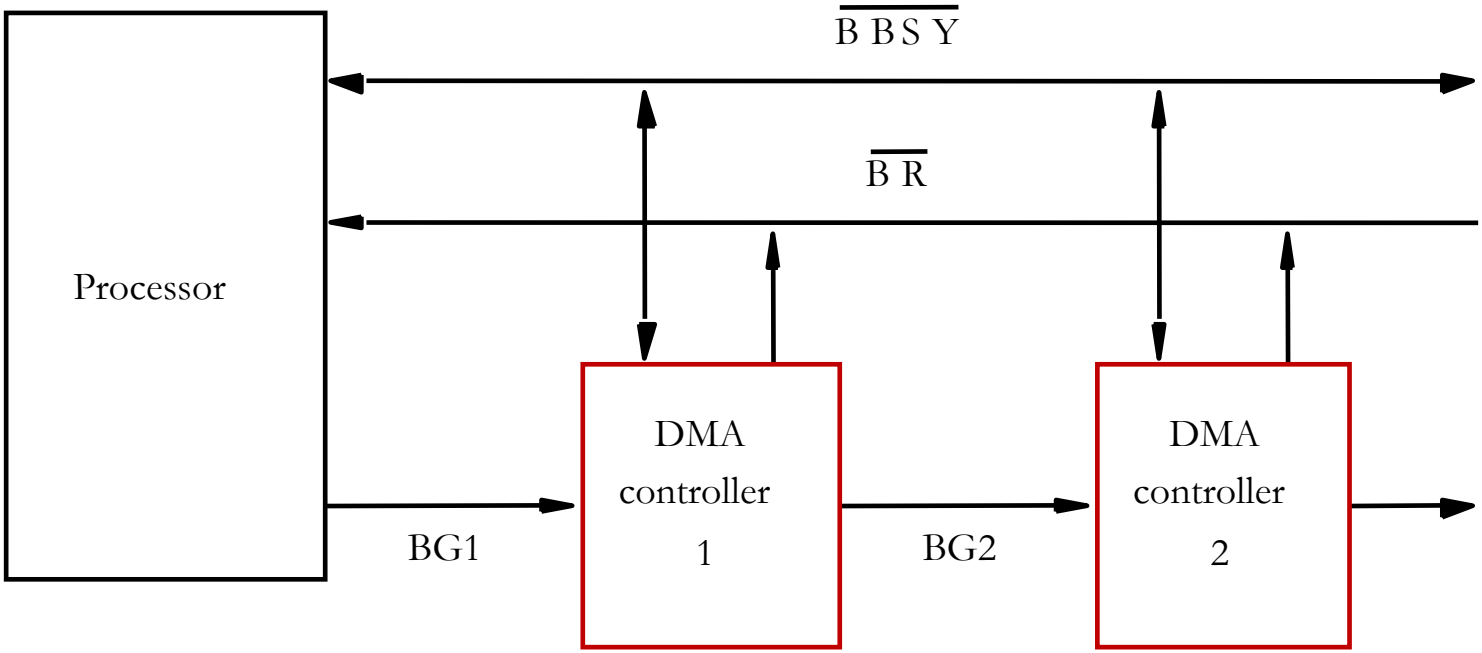
DMA Transfer Modes

- CYCLE STEALING Mode
  - By continually obtaining and releasing control of the system buses, the DMA controller essentially interleaves instruction execution & data transfers. The CPU processes an instruction, then the DMA controller transfers a data value, and so on.
  - The data block is not transferred as quickly as in burst mode, but the CPU is not idled for as long as in that mode.
  - Useful for controllers monitoring data in real time.

Bus arbitration

- Processor and DMA controllers both need to initiate data transfers on the bus and access main memory.
- The device that is allowed to initiate transfers on the bus at any given time is called the **bus master**.
- When the current bus master relinquishes its status as the bus master, another device can acquire this status.
  - The process by which the next device to become the bus master is selected and bus mastership is transferred to it is called bus arbitration.
- Centralized arbitration:
  - A **single** bus arbiter performs the arbitration.
- Distributed arbitration:
  - **All devices** participate in the selection of the next bus master.

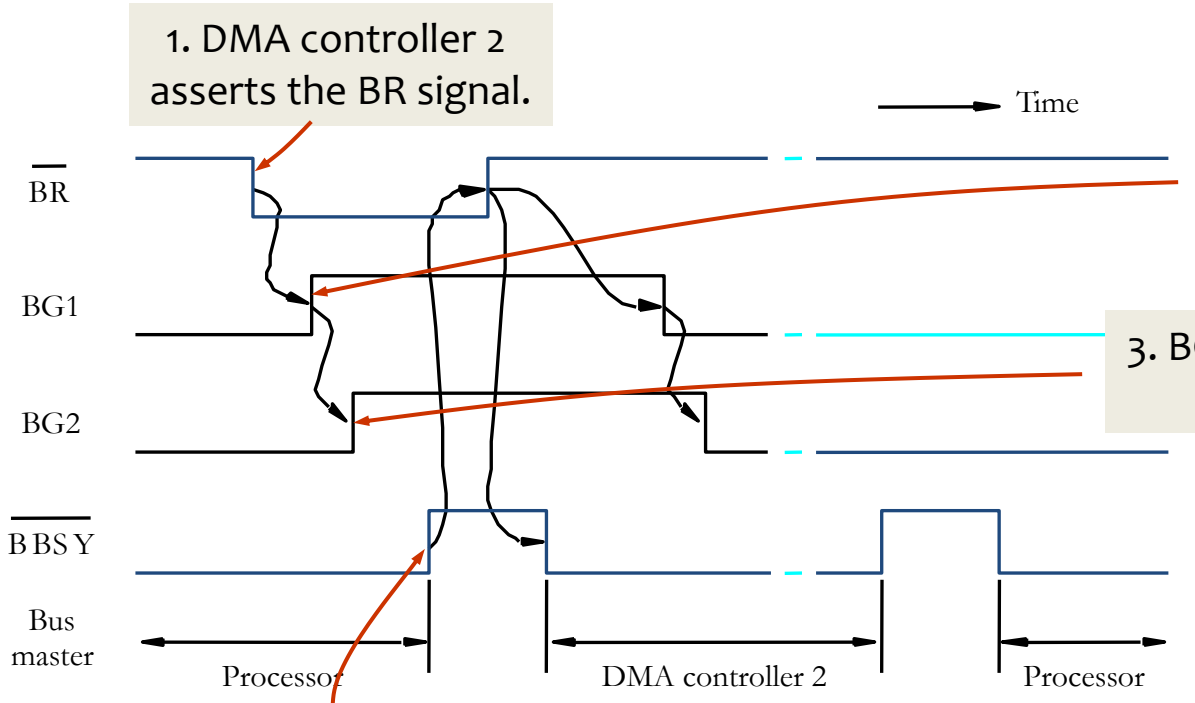
Centralized Bus Arbitration



Centralized Bus Arbitration(cont.,)

- **Bus arbiter may be the processor or a separate unit** connected to the bus.
- Normally, the **processor is the bus master**, unless it grants bus membership to one of the DMA controllers.
- DMA controller requests the control of the bus by asserting the **Bus Request** (BR) line.
- In response, the processor activates the **Bus-Grant1** (**BG1**) line, indicating that the **controller may use the bus when it is free**.
- BG1 signal is connected to all DMA controllers in a **daisy chain** fashion.
- **BBSY** signal is **0**, it indicates that the **bus is busy**. When BBSY becomes **1**, the DMA controller which asserted BR can **acquire** control of the bus.

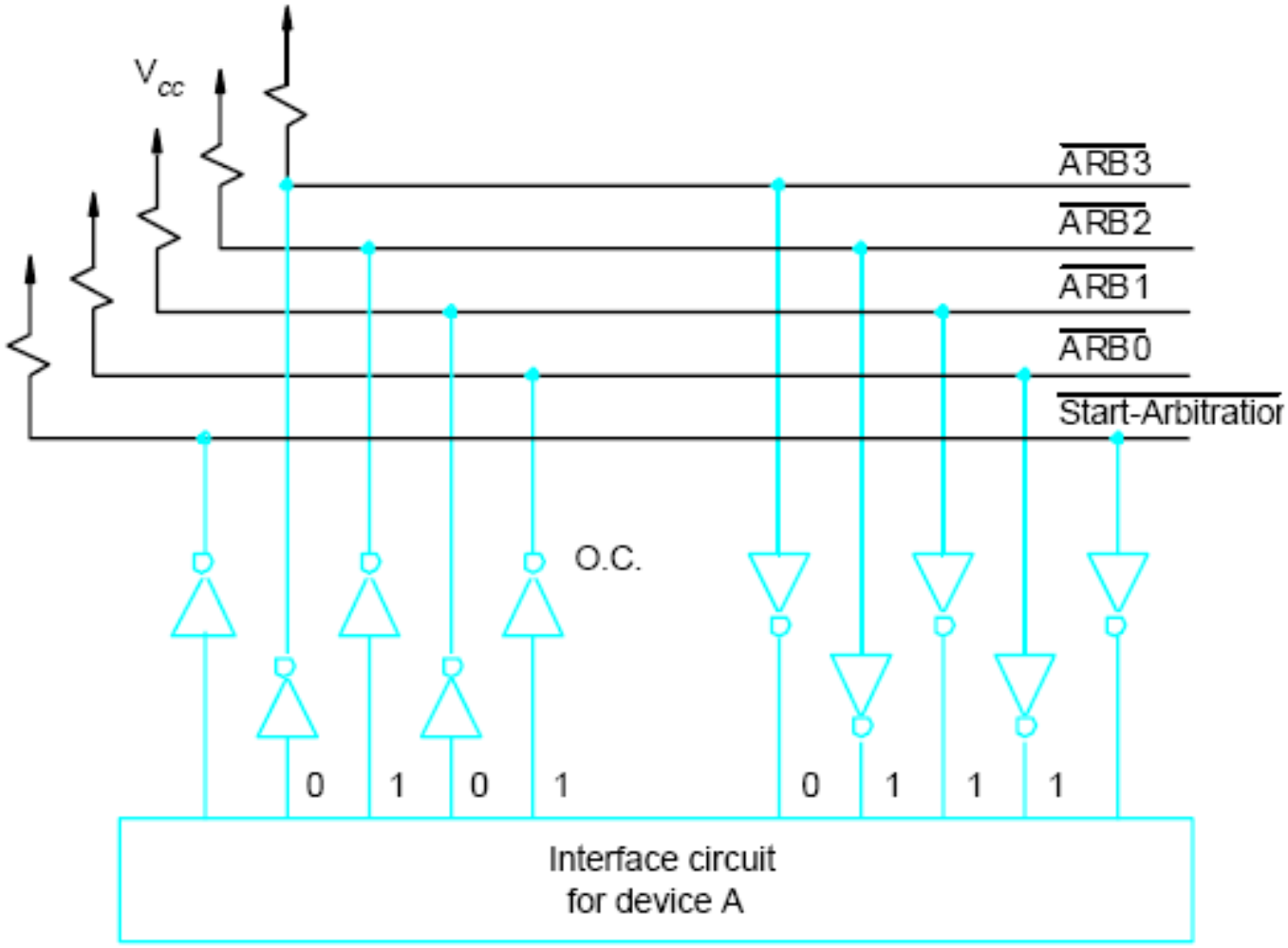
Centralized arbitration (contd..)



Distributed arbitration

- All devices waiting to use the bus share the responsibility of carrying out the arbitration process.
  - Arbitration process does not depend on a central arbiter and hence distributed arbitration has higher reliability.
- Each device is assigned a 4-bit ID number.
- All the devices are connected using 5 lines, 4 arbitration lines to transmit the ID, and one line for the Start-Arbitration signal.
- To request the bus, a device:
  - Asserts the Start-Arbitration signal.
  - Places its 4-bit ID number on the arbitration lines.
- The pattern that appears on the arbitration lines is the logical-OR of all the 4-bit device IDs placed on the arbitration lines.

Distributed arbitration



Distributed arbitration(Contd.,)

- *Arbitration process:*
  - *Each device compares the pattern that appears on the arbitration lines to its own ID, starting with MSB.*
  - *If it detects a difference at a bit position, it transmits zero for that and all lower bit positions.*
  - *The pattern that appears on the arbitration lines is the logical-OR of all the 4-bit device IDs placed on the arbitration lines.*

Distributed arbitration (contd..)

- *Device A has the ID 5 and wants to request the bus:*
  - *Transmits the pattern **0101** on the arbitration lines.*
- *Device B has the ID 6 and wants to request the bus:*
  - *Transmits the pattern **0110** on the arbitration lines.*
- *Pattern that appears on the arbitration lines is the logical OR of the patterns:*
  - *Pattern 0111 appears on the arbitration lines.*

Arbitration process:

- *Each device compares the pattern that appears on the arbitration lines to its own ID, starting with MSB.*
- *If it detects a difference, it transmits zeros on the arbitration lines for that and all lower bit positions.*
- *Device A compares its ID 5 with a pattern 0101 to pattern 0111.*
- *It detects a difference at bit position 1, as a result, it transmits a pattern 0100 on the arbitration lines.*
- *The pattern that appears on the arbitration lines is the logical-OR of 0100 and 0110, which is 0110.*
- *This pattern is the same as the device ID of B, and hence B has won the arbitration.*

Buses

Buses

- Processor, main memory, and I/O devices are interconnected by means of a bus.
- Bus provides a communication path for the transfer of data.
  - Bus also includes lines to support interrupts and arbitration.
- A bus protocol is the set of rules that govern the behavior of various devices connected to the bus, as to when to place information on the bus, when to assert control signals, etc.

Buses (contd..)

- Bus lines may be grouped into three types:
  - Data
  - Address
  - Control
- Control signals specify:
  - Whether it is a read or a write operation.
  - Required size of the data, when several operand sizes (byte, word, long word) are possible.
  - Timing information to indicate when the processor and I/O devices may place data or receive data from the bus.

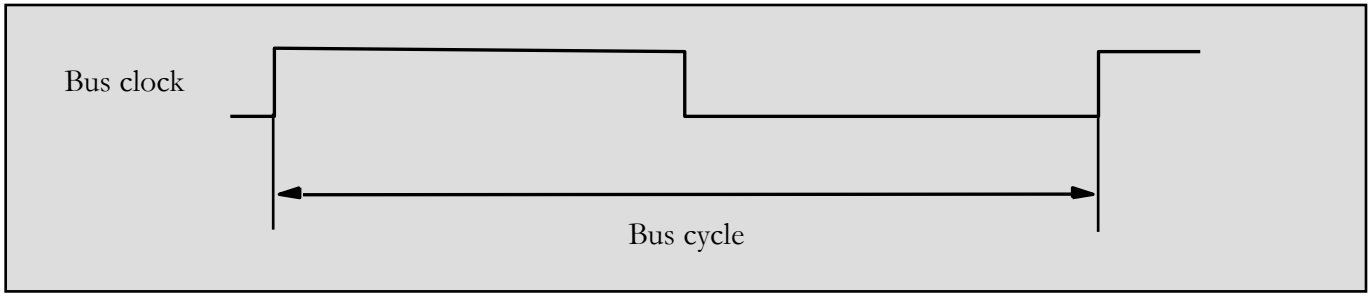
Buses (contd..)

- **Master** (Initiator) – initiates data transfer by issuing read or write command on the bus.
- processor or a DMA capable device can act as master.
- **Slave / Target -** The device addressed by the master.

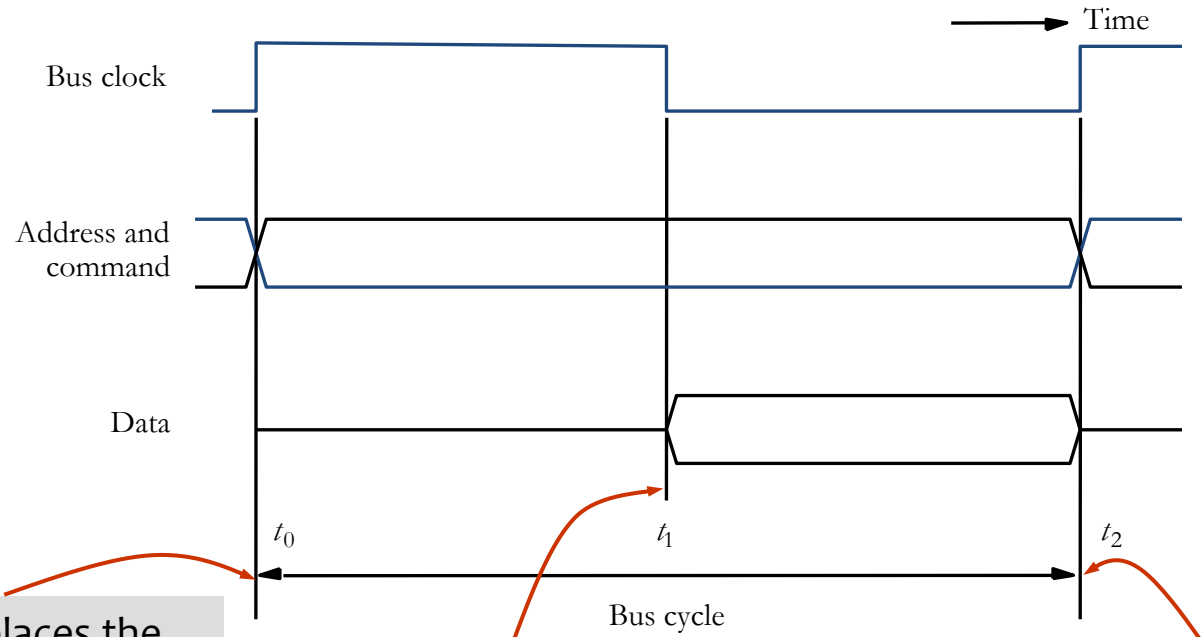
- Schemes for timing of data transfers over a bus can be classified into:
  - Synchronous,
  - Asynchronous.

Synchronous bus

- In synchronous bus, all devices derive timing information from a common clock line.
- Equally spaced pulses on this line define equal time intervals or bus cycles
- During a "**bus cycle" one data transfer can take place.**



Synchronous bus (contd..)



Master places the device address and command on the bus, and indicates that it is a Read operation.

Addressed slave places data on the data lines

Master "strobes" the data on the data lines into its input buffer, for a Read operation.

- *In case of a Write operation, the master places the data on the bus along with the address and commands at time t_0.*
- *The slave strobes the data into its input buffer at time t_2.*

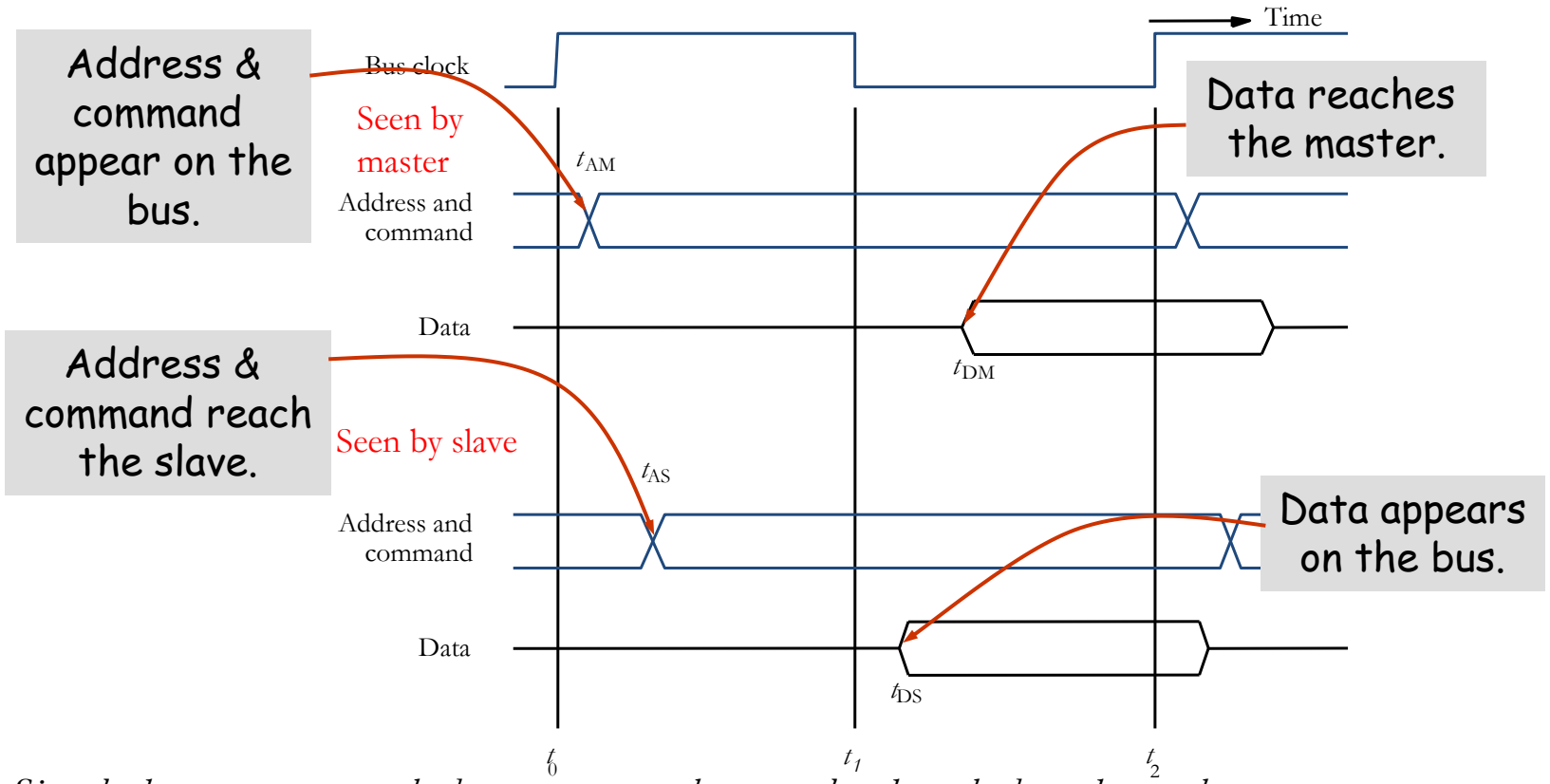
Synchronous bus (contd..)

- Once the master places the device address and command on the bus, it takes time for this information to propagate to the devices:
  - This time depends on the physical and electrical characteristics of the bus.
- Also, all the devices have to be given enough time to decode the address and control signals, so that the addressed slave can place data on the bus.
- Width of the pulse t_1 - t_0 depends on:
  - Maximum **propagation delay** between two devices connected to the bus,
  - time taken by all the devices to **decode the address** and **control signals**, so that the addressed slave can respond at time t_1.

Synchronous bus (contd..)

- At the end of the clock cycle, at time t_2, the master strobes the data on the data lines into its input buffer if it's a Read operation.
  - "Strobe" means to capture the values of the data and store them into a buffer.
- When data are to be loaded into a storage buffer register, the data should be available for a period longer than the setup time of the device.
- Width of the pulse $t_2 - t_1$ should be longer than:
  - Maximum **propagation time** of the bus plus
  - **set up time of the input buffer** register of the master.

Synchronous bus (contd..)



- *Signals do not appear on the bus as soon as they are placed on the bus, due to the propagation delay in the interface circuits.*
- *Signals reach the devices after a propagation delay which depends on the characteristics of the bus.*
- *Data must remain on the bus for some time after t_2 equal to the hold time of the buffer.*

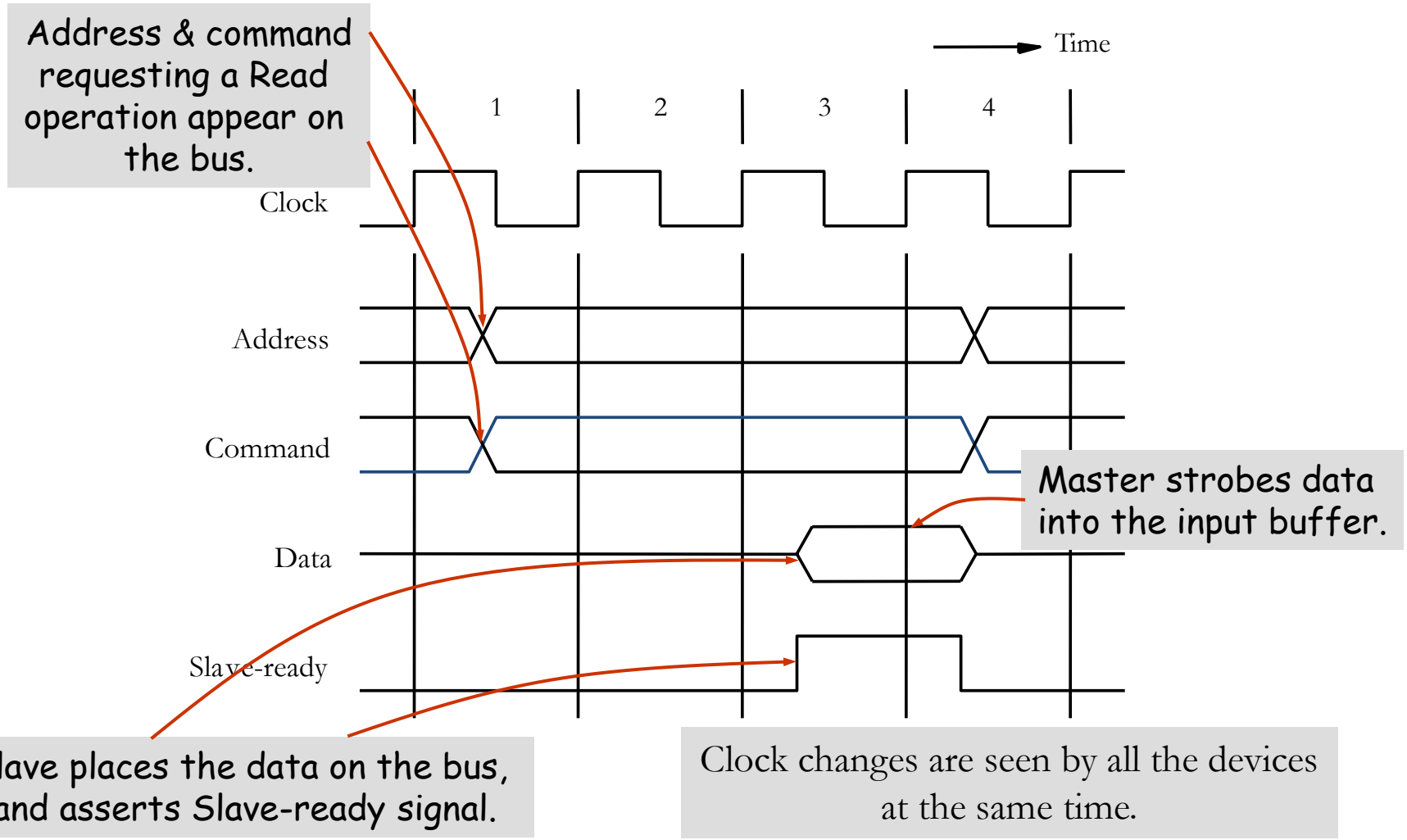
Synchronous bus (contd..)

- Data transfer has to be completed within one clock cycle.
  - Clock period t2 - must be such that the longest propagation delay on the bus and the slowest device interface must be accommodated.
  - Forces all the devices to operate at the speed of the slowest device.
- Processor just assumes that the data are available at t2 in case of a Read operation, or are read by the device in case of a Write operation.
  - What if the device is actually failed, and never really responded?

Synchronous bus (contd..)

- Most buses have control signals to represent a response from the slave.
- Control signals serve two purposes:
  - Inform the master that the slave has recognized the address, and is ready to participate in a data transfer operation.
  - Enable to adjust the duration of the data transfer operation based on the speed of the participating slaves.
- High-frequency bus clock is used:
  - Data transfer spans several clock cycles instead of just one clock cycle as in the earlier case.

Synchronous bus (contd..)



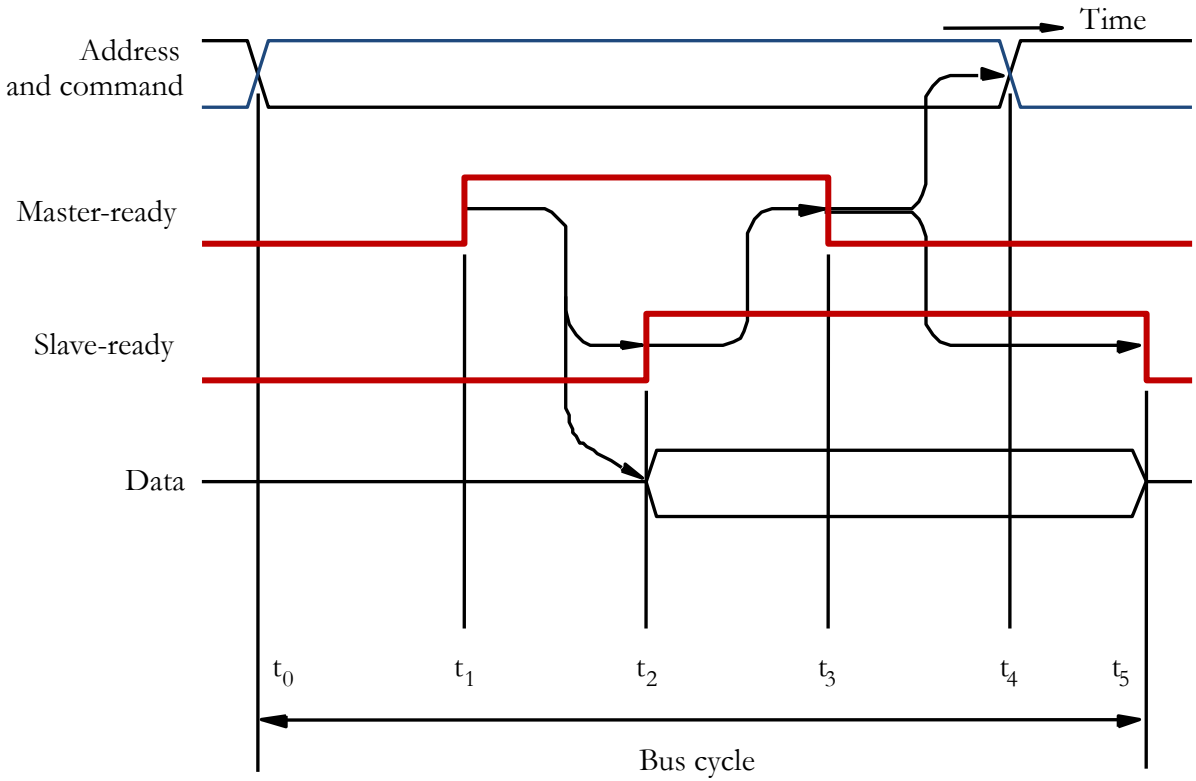
Asynchronous bus

- Data transfers on the bus is controlled by a handshake between the master and the slave.
- Common clock in the synchronous bus case is replaced by two timing control lines:
  - Master-ready,
  - Slave-ready.
- Master-ready signal is asserted by the master to indicate to the slave that it is **ready to participate** in a data transfer.
- Slave-ready signal is asserted by the slave **in response to the master-ready** from the master, and it indicates to the master that the **slave is ready to participate** in a data transfer.

Asynchronous bus (contd..)

- Data transfer using the handshake protocol:
  - Master places the address and command information on the bus.
  - Asserts the Master-ready signal to indicate to the slave that the address and command information has been placed on the bus.
  - All devices on the bus decode the address.
  - Addressed slave performs the required operation, and informs the processor it has done so by asserting the Slave-ready signal.
  - Master removes all the signals from the bus, once Slave-ready is asserted.
  - If the operation is a Read operation, Master also strobes the data into its input buffer.

Asynchronous bus- Timing diagram for i/p operation



t_0 - Master places the address and command information on the bus.

t_1 - Master asserts the Master-ready signal. Master-ready signal is asserted at t_1 instead of t_0

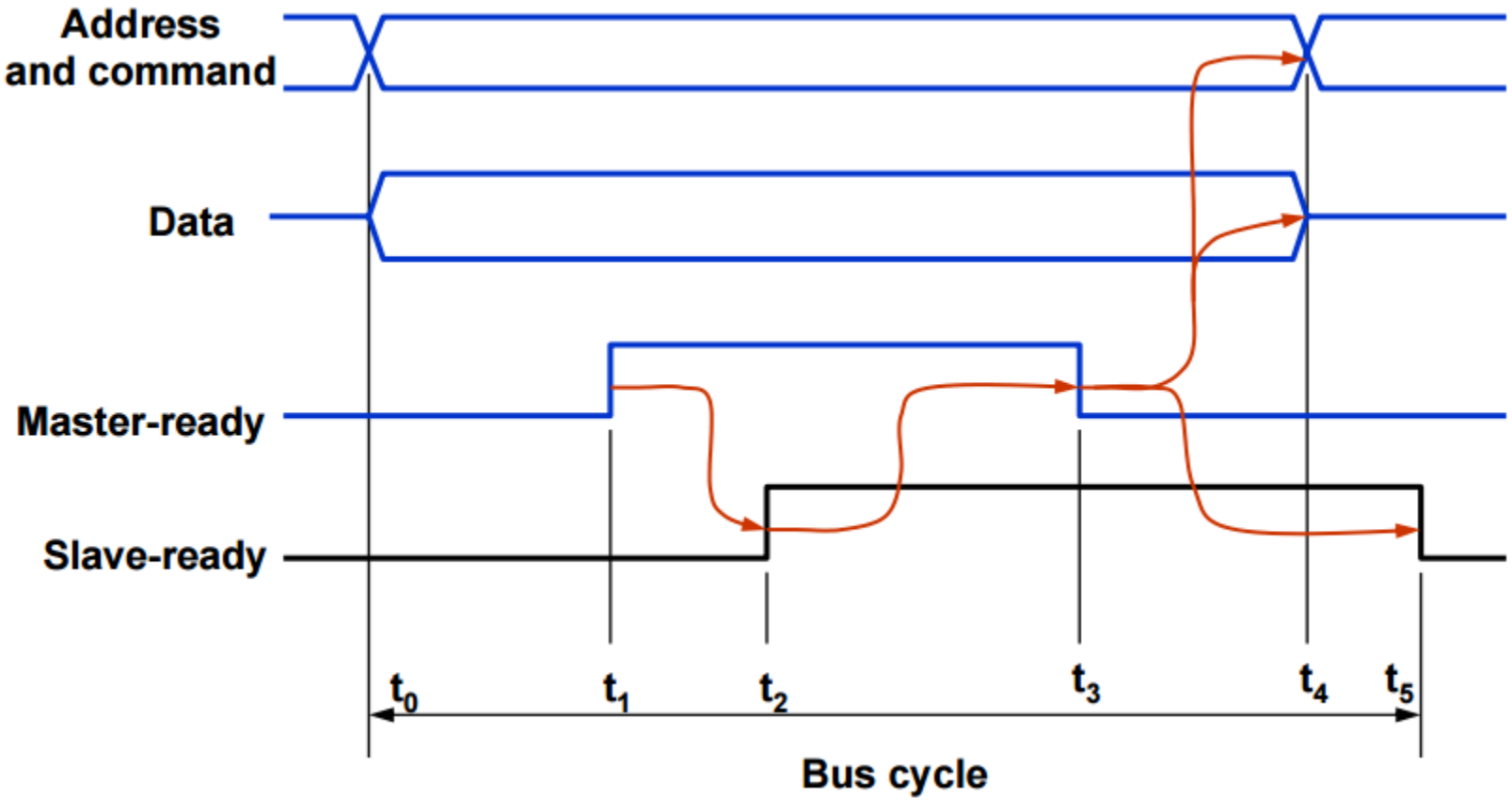
t_2 - Addressed slave places the data on the bus and asserts the Slave-ready signal.

t_3 - Slave-ready signal arrives at the master.

t_4 - Master removes the address and command information.

t_5 - Slave receives the transition of the Master-ready signal from 1 to 0. It removes the data and the Slave-ready signal from the bus.

Asynchronous bus- Timing diagram for o/p operation



Asynchronous vs. Synchronous bus

- Advantages of asynchronous bus:
  - Eliminates the need for synchronization between the sender and the receiver.
  - Can accommodate varying delays automatically, using the Slave-ready signal.
- Disadvantages of asynchronous bus:
  - Data transfer rate with full handshake is limited by two-round trip delays.
  - Data transfers using a synchronous bus involves only one round trip delay, and hence a synchronous bus can achieve faster rates.
  - However, for synchronous bus, Clock circuitry must be designed carefully to ensure proper synchronization, and delays must be kept within strict bounds