Unit III

Syllabus

Bus Standards: Introduction, Asynchronous and Synchronous Buses, PCI, USB, FireWire (IEEE1394) etc.

Bus Arbitration

- The device that is allowed to initiate data transfers on the bus at any given time is called the bus master.
- When the current master releases the control of the bus, another device can acquire this status.
- Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.

- There are two approaches for bus arbitration:

1. Centralized arbitration
2. Distributed arbitration

Centralized Arbitration

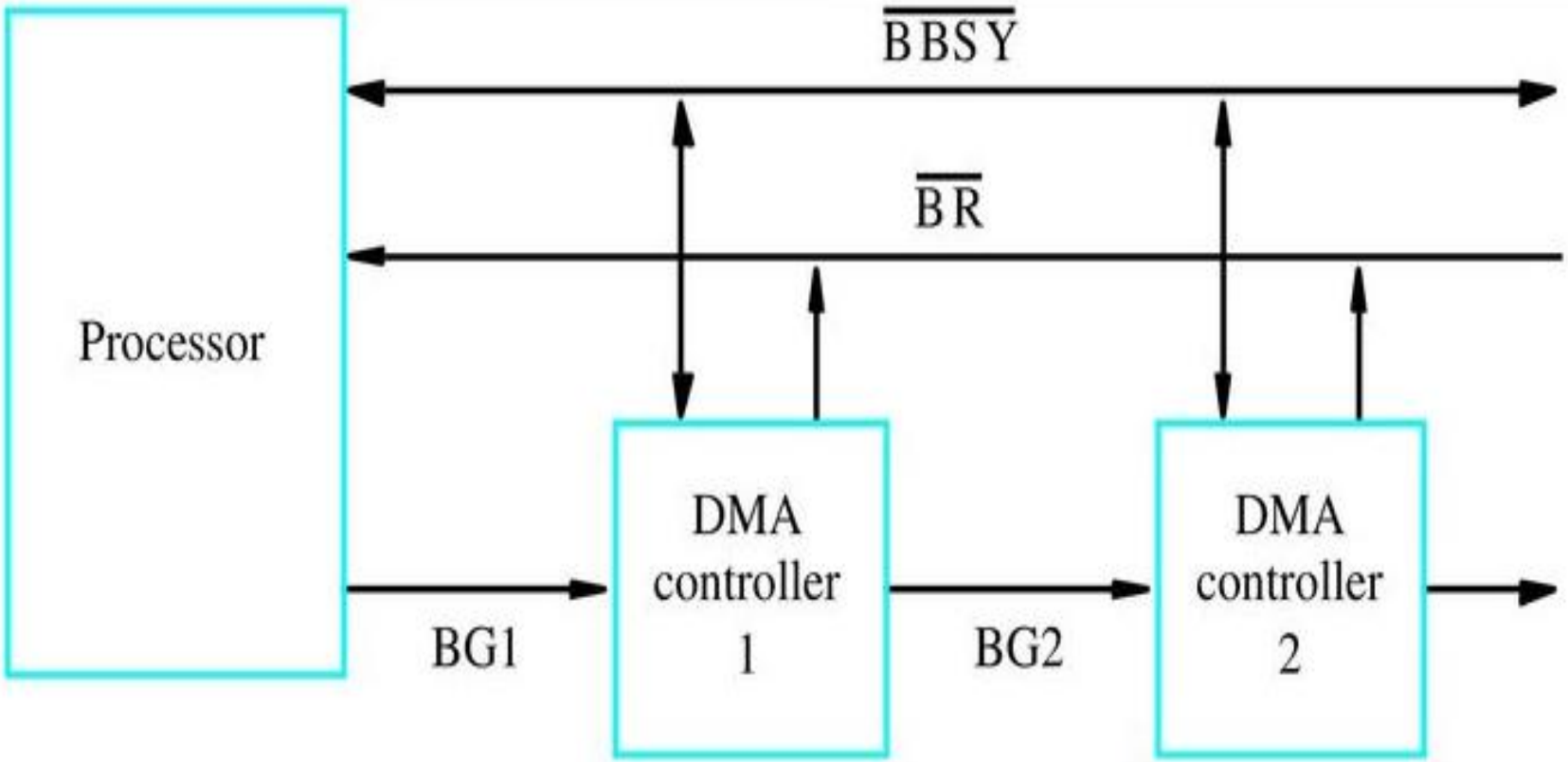
- In centralized arbitration a single bus arbiter performs the required arbitration.
- The bus arbiter may be the processor or a separate unit connected to the bus.
- Normally the processor is the bus master unless it grants bus mastership to one of the DMA controllers.

- A DMA controller indicates that it needs to become the bus master by activating the bus request line, BR.
- When the bus request is activated, the processor activates the bus grant signal, BG1, indicating to the DMA controller that they may use the bus when it become free.
- This signal is connected to all DMA controller using a daisy chain management.

- If DMA controller 1 is requesting the bus, it blocks the propagation of the grant signal to other devices. Otherwise it passes the grant downstream by asserting BG2.
- The current bus master indicates to all devices that it is using the bus by activating the Bus Busy signal, BBSY.

- After receiving the bus grant signal, a DMA controller waits for the Bus Busy signal to become inactive, then assumes the mastership of the bus.
- At this time, is activates the Bus Busy signal to prevent other devices from using the bus at the same time.

- The arbiter circuit ensures that only one request is granted at any given time, according to a predefined priority scheme.
- For example, if there are four bus request lines, BR1 to BR4, a fixed priority scheme may be used in which BR1 is given top priority and BR4 is given lowest priority.

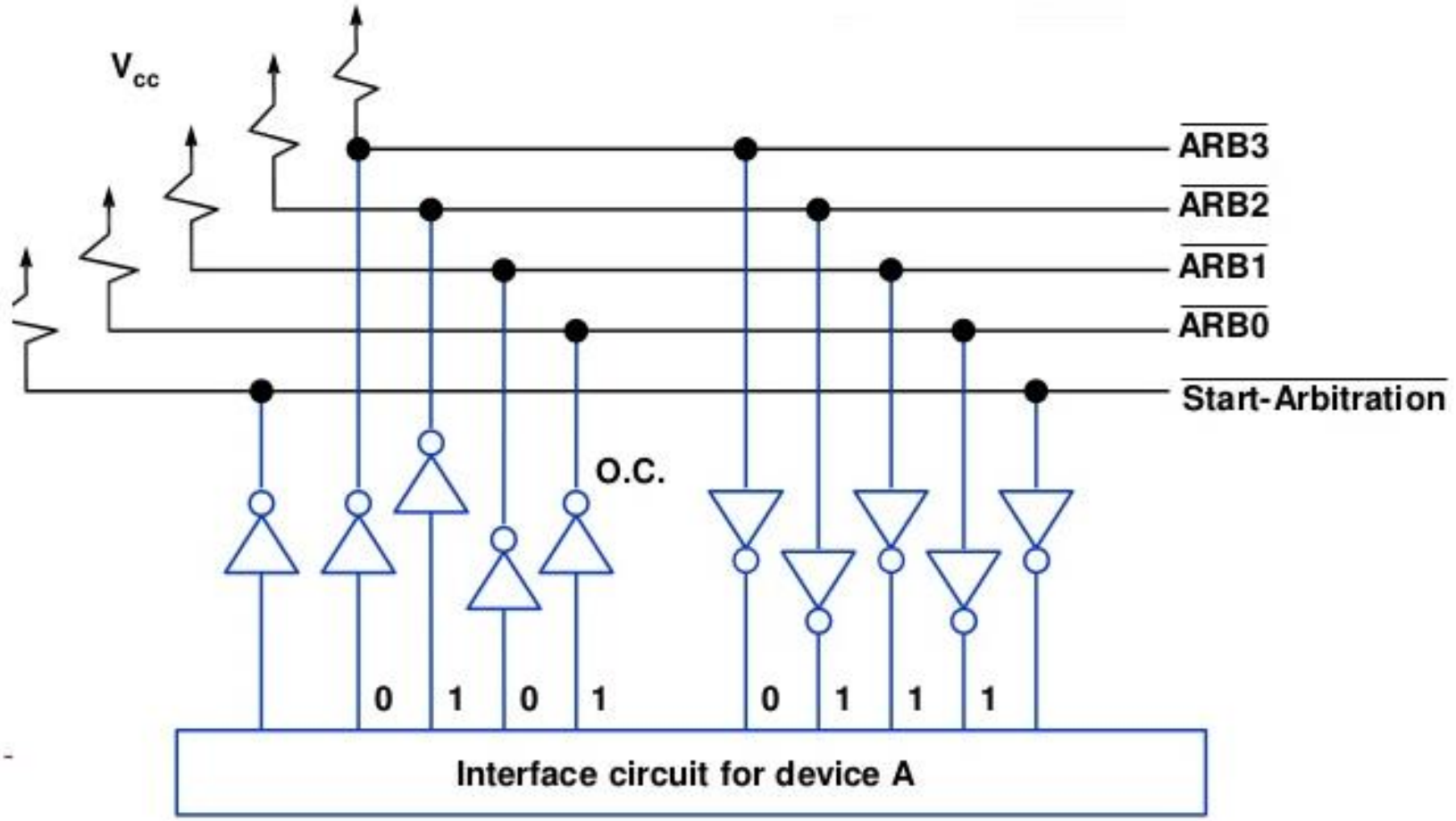


A simple arrangement for bus arbitration using a daisy chain

Distributed Arbitration

- Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.
- Each device on the bus is assigned a 4-bit identification number.
- When one or more devices request the bus, they assert the Start Arbitration signal and place their 4-bit ID numbers on four open collector lines, $\overline{\text{ARB0}}$ to $\overline{\text{ARB3}}$.

- A winner is selected as a result of the interaction among the signal transmitted over these lines by all contenders.
- The net outcome is that the code on the four lines represents the request that has the highest ID number.
- Decentralised arbitration has the advantage of offering the higher reliability, because the operation of the bus is not dependent on any single device.



A distributed arbitration scheme

Bus

- The processor, main memory, and I/O devices can interconnected by means of a common bus whose primary function is to provide a communication path for the transfer of data.
- The bus includes the lines needed for bus arbitration and interrupts.

- The bus lined used for transferring data may be grouped into three types: data, address, control lines.
- The control signals specify whether a read or write operation is to be performed.
- Usually, a single a R/W line is used, it specifies the READ when set to 1 and write when set to 0.
- When several operands sizes are possible, such as byte, word, or long word, the required size of data is indicated.

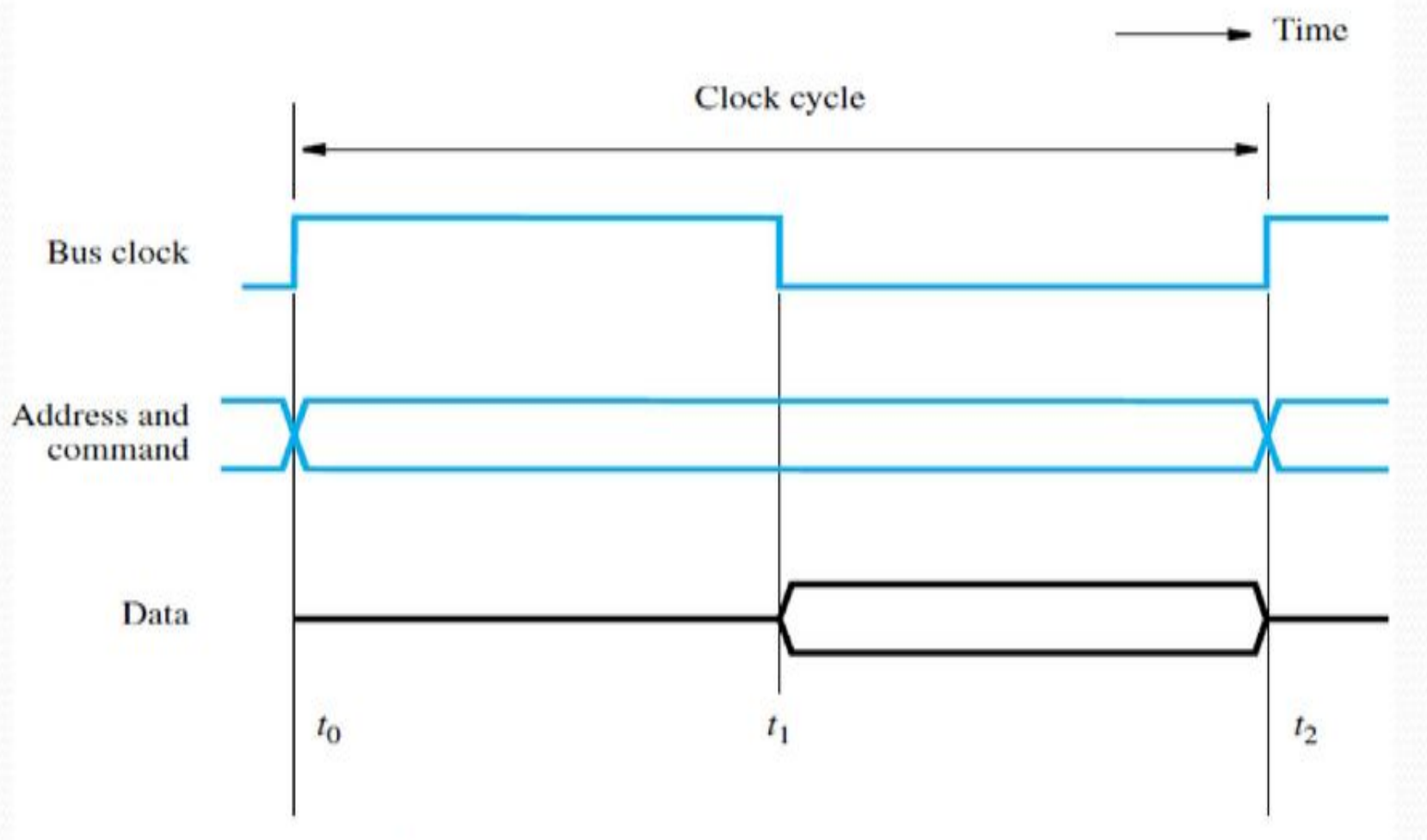
- The bus control signals also carry the timing information. They specify the times at which the processor and I/O devices may place the data on the bus or receive data from the bus.
- A variety of scheme have been devised for the timing of data transfers over a bus.
- These can be broadly classified as synchronous and asynchronous schemes.

- In any data transfer operation, one device plays the role of a master.
- This is the device that initiates data transfers by issuing read or write commands on the bus; hence it may be called an initiator.
- Normally, the processor acts as the master, but other devices with DMA capability may also become bus masters.
- The device addressed by the master is referred to as a slave or target.

Synchronous Bus

- In a synchronous bus, all devices derive timing information from a common clock line.
- Equally spaced pulses on this line define equal time intervals.
- In the simplest form of a synchronous bus, each of these intervals constitutes a bus cycle during which one data transfer can take place.
- The address and data lines in this and subsequent figures are shown as high and low at the same time.
- This is a common convention indicating that some lines are high and some low, depending on the particular address or data address being transmitted.

Timing of an input transfer on a Synchronous Bus



- At time t0, the master place the device address on the address lines and sends an appropriate commands on the control lines.
- In this case, the command will indicate an input operation and specify the length of the operand to be read, if necessary.
- Information travels over the bus at a speed determine by the its physical and electrical characteristics.
- The clock pulse width, t1-t0 must be longer than the maximum propagation delay between two devices connected to the bus.
- It also has to be long enough to allow all devices to decode the address and control signals so that the addressed device(the slave) can respond at time t1.
- It is important that slaves take no action or place any data on the bus before t1.
- The information on the bus is unreliable during the period t0 to t1 because signals are changing state.
- The addressed slave places the requested input data on the data lines at time t1.

- At the end of the clock cycle, at time t2, the master strobes the data on the data lines into its buffer.
- In this context, “strobe” means to capture the values of the data at a given time instant and store them into a buffer.
- For data to be loaded correctly into any storage device, such as register built with flip-flops, the data must be available at the input of that device for a period greater than the setup time of the device.
- Hence the period t2-t1 must be greater than maximum propagation time on the bus plus the setup time of the input buffer register of the master.

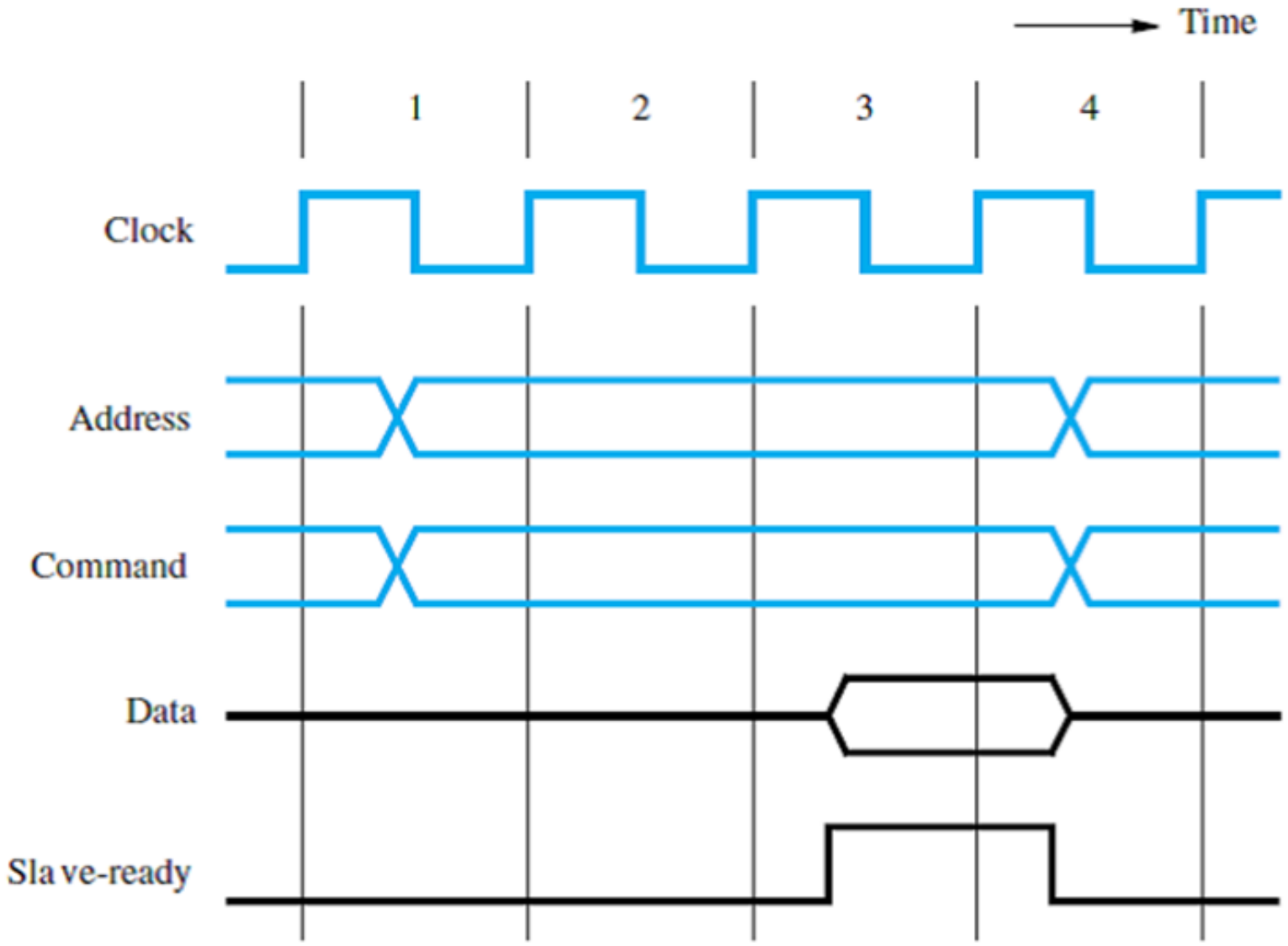
- A similar procedure is followed for an output operation. The master places the output data on the data lines when it transmits the address and commands information.
- At time t2, the addressed device strobes the data lines and loads the data into its data buffer.

Multiple Cycle Transfers

- The synchronous scheme has some limitations.
- Because in synchronous scheme a transfer has to be completed within one clock cycle, the clock period, t2-t0, must be chosen to accommodate the longest delays on the bus and the slowest device interface.
- This forces all devices to operate at the speed of the slowest device.
- Also, the processor has no way of determining whether the addressed device has actually responded.
- It simply assumes that at t2, the output data have been received by the I/O device or the input data are available on the on the data lines.
- If because of malfunction, the device does not responded, the error will not be detected.

Solution

- To overcome these limitation, most buses incorporate control signals that represent a response from the device.
- These signals inform the master that the slave has recognized its address and that it is ready to participate in a data transfer operation.
- They also make it possible to adjust the duration of the data transfer period to suit the needs of the participating device.
- To simplify this process, a high frequency clock signal is used such that a complete data transfer cycle would span several clock cycles.
- Then, the number of clock cycles involved can vary from one device to another.



An input transfer using multiple clock cycles

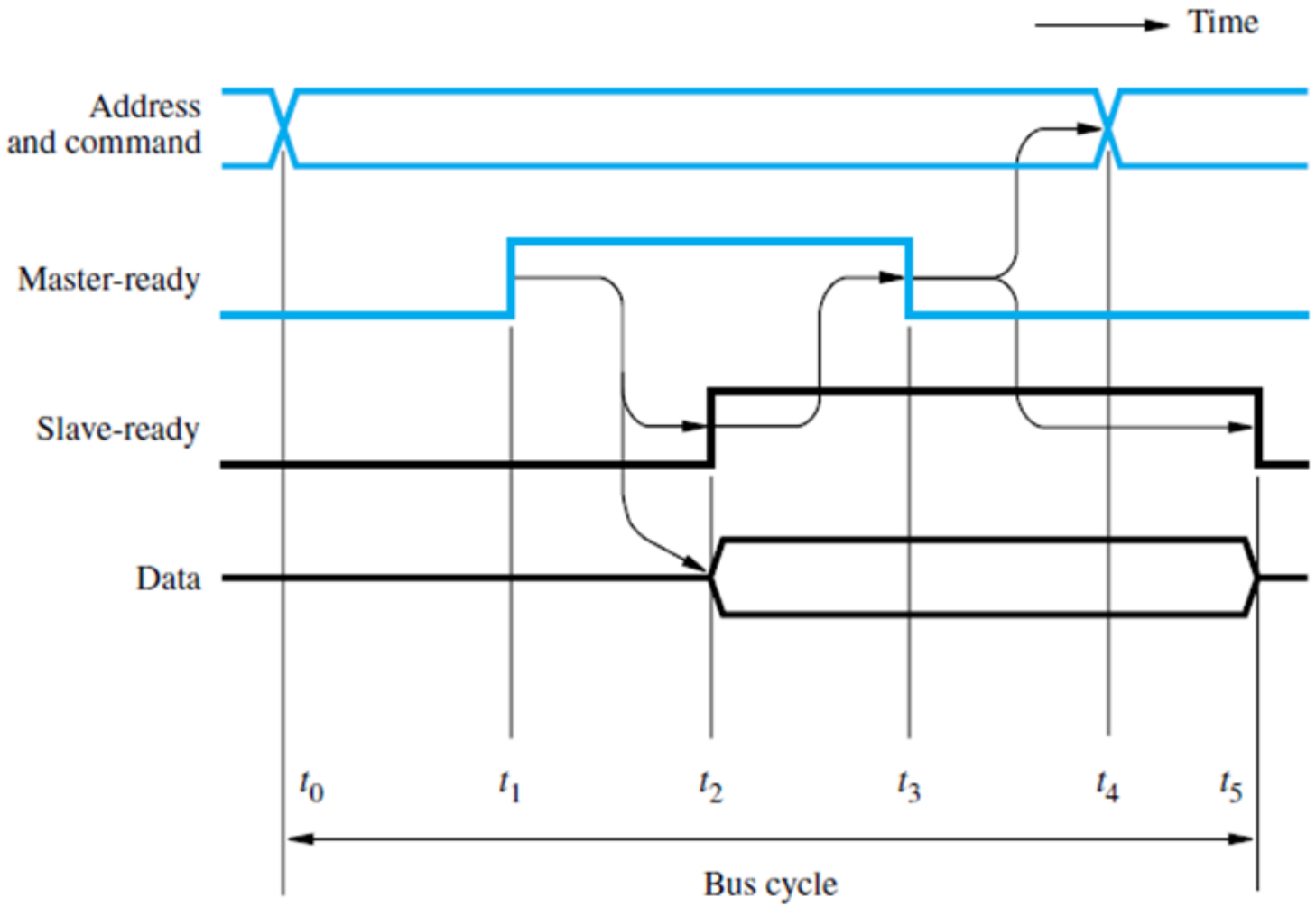
- Many computer buses, such as the processor buses of Pentium and ARM, use a multiple clock cycles transfer scheme for data transfer.

Asynchronous Bus

- An alternative scheme for controlling data transfers on the bus is based on the use of a handshake between the master and the slave.
- The common clock is replaced by two timing control lines, master ready and slave ready.
- The first is asserted by the master indicating that it is ready for a transaction and the second is a response from the slave.

- The master places the address and command information on the bus.
- Then it indicates to all devices that it has done so by activating the master ready line.
- This causes all devices on the bus to decode the address.
- The selected slave performs the required operation and inform the processor it has done so by activating the slave ready line.
- The master waits for slave ready to become asserted before it removes its signal from the bus.
- In the case of a read operation, it also strobes the data into its input buffer.

Figure


Time
Address and command
Master-ready
Slave-ready
Data
t_0
t_1
t_2
t_3
t_4
t_5
Bus cycle

- At t0- The master places the address and command information on the bus, and all devices on the bus begin to decode this information.
- At t1 – The master sets the Master-ready line to 1 to inform the I/O devices that the address and command information is ready. The delay t1-t0 is intended to allow for any skew that may occur on the bus.
- At t2 – The selected slave, having decoded the address and command information, performs the required input operation by placing the data from its register on the data lines. At the same time, it sets the slave ready signal to 1.

- At t3 – The slave ready signal arrives at the master, indicating that the input data are available on the bus. However since it was assumed that the device interface transmits the slave ready signal at the same time that it places the data on the bus, the master should allow for the bus skew. It must allow for the setup time needed by its input buffer. After a delay equivalent to the maximum bus skew and the minimum setup time, the master strobes the data into its input buffer. At the same time it drops the Master-ready signal, indicating that is has received the data.
- At t4 – The master removes the address and command information form the bus.
- At t5 – When the device interface receives the 1 to 0 transition of the master ready signal, it removes the data and the slave ready signal from the bus. This completes the input transfer.

- The timing information for an output operation is essentially the same as for an input operation.
- In this case, the master places the output data on the data lines at the same time that it transmits the address and command information.
- The selected slaves strobes the data into its output buffer when it receives the master-ready signal and indicates that it has done so by setting the slave ready signal to 1.

- The main advantage of the asynchronous bus is that the handshake process eliminates the need for synchronization of the sender and receiver clocks, thus simplifying the timing design.
- Delays, whether introduced by the interface circuits or by propagation over the bus wires, are readily accommodated.
- When these delays change, for example, due to change in load when an interface circuit is added or removed, the timing of data transfer adjusts automatically based on the new conditions.

- For a synchronous bus, clock circuitry must be designed carefully to ensure proper synchronization, and delays must be kept within strict bounds.
- The rate of data transfer on an asynchronous bus controlled by a full handshake is limited by the fact that each transfer involves two round trip delays. This can be readily seen as each transition on slave ready must wait for the arrival of a transition on master ready and vice versa.
- On synchronous buses, the clock period need only accommodate one end to end propagation delay. Hence, faster transfer rates can achieved.

PCI(Peripheral Component Interconnect)

- The PCI bus was introduced in 1992.
- The PCI was developed as a low cost bus that is truly processor independent.
- Its design anticipated a rapidly growing demand for bus bandwidth to support high speed disks and graphic and video devices.
- An important feature that PCI pioneered is a plug and play capability for connecting I/O devices.
- To connect a new device, the user simply connects the device interface board to the bus. The software takes care of the rest.

Data Transfers

- In today's computers, most memory transfers involve a burst of data rather than just one word.
- The reason is that modern processors include a cache memory.
- Data are transferred between the cache and the main memory in bursts of several words each.
- The words involved in such a transfer are stored at successive memory locations.

- When the processor (cache controller) specifies an address and requests a read operation from the main memory, the memory responds by sending a sequence of data words starting at that address.
- Similarly during a write operation, the processor sends a memory address followed by a sequence of data words, to be written in successive memory locations starting at that address.
- The PCI is designed primarily to support this mode of operation.
- A read or write operation involving a single word is simply treated as a burst of length one.

- The PCI bus supports three independent address space: memory, I/O, and Configuration.
- The configuration space is intended to give the PCI its plug and play capability.
- A 4-bit command that accompanies the address identifies which of three address spaces is being used in a given data transfer operation.
- In PCI bus the slave can store the address in its internal buffer. Thus the address is needed on the bus for one clock cycle only, freeing the address lines to be used for sending data in subsequent clock cycles.
- The result is a significant cost reduction because the number of wires on a bus is an important cost factor.

- At any given time, one device is the bus master. It has the right to initiate data transfers by issuing the read and write commands.
- A master is called an initiator in PCI terminology. This is either a Processor or DMA controller.
- The addressed device that responds to read and write commands is called a target.

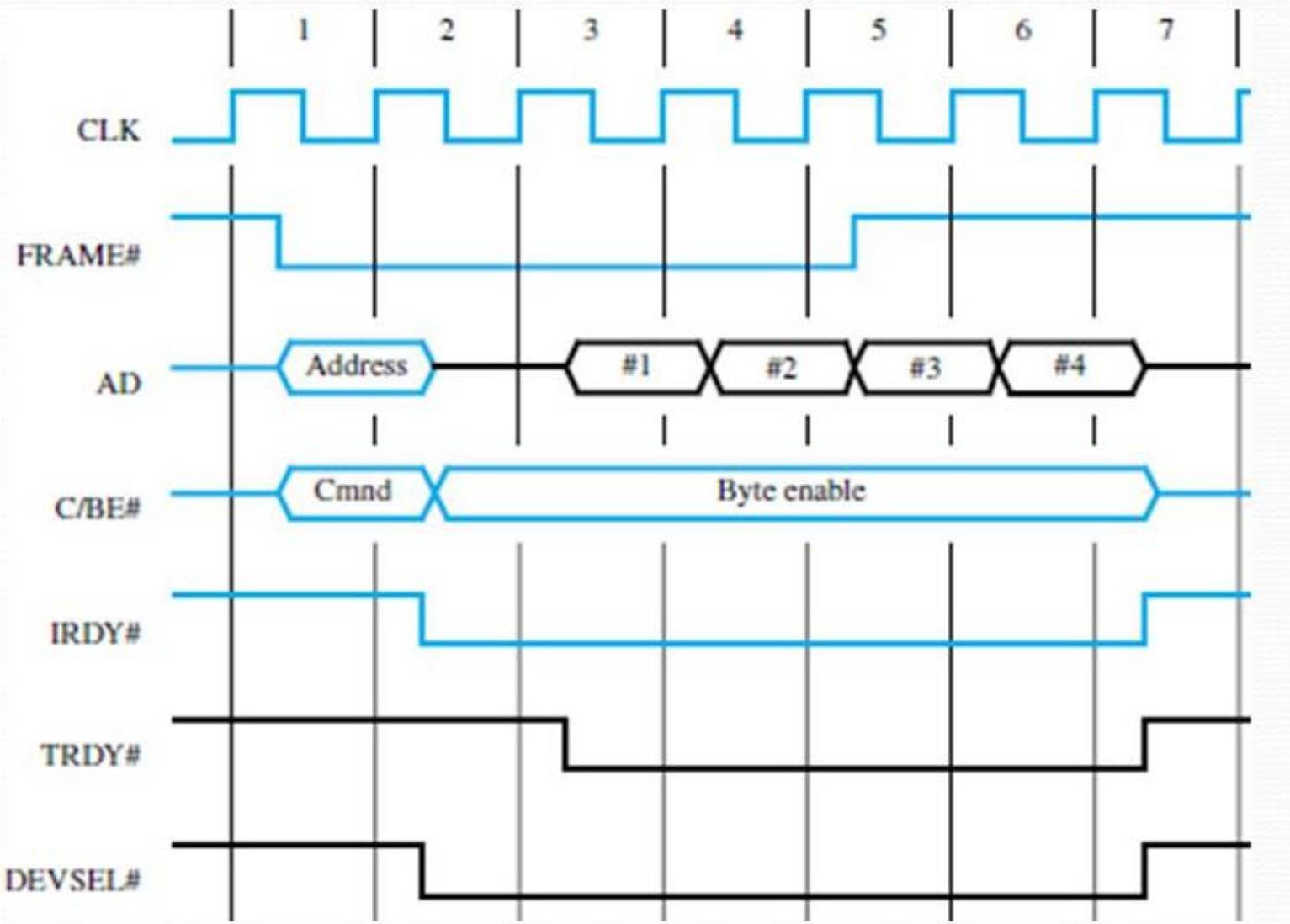
Data Transfers Signals on the PCI Bus

Name	Function
CLK	A 33Mhz or 66Mhz clock
FRAME#	Sent by the initiator to indicate the duration of a transaction
AD	32bit address/data lines, which may be increased to 64bit
C/BE#	4 command/byte enable line
IRDY#, TRDY#	Initiator ready and target ready signals
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction
IDSEL#	Initialization Device Select

- The signals whose name ends with # are asserted when in the low voltage state.
- A complete transfer operation on the bus, involving an address and a burst of data, is called a transaction.
- Individual word transfers within a transaction are called phases.

- Consider the bus transaction in which the processor reads four 32bit words from the memory.
- In this case the processor is the initiator and target is the memory.

Timing Diagram



- In clock cycle 1, the processors asserts FRAME# to indicate the beginning of a transaction.
- At the same time, it sends the address on the AD lines and a command on the C/BE# lines.
- In this case, the command will indicate that a read operation is requested and that the memory address space is being used.

- In clock cycle 2, the processor removes the address and disconnects its drivers from the AD lines.
- The selected target enables its drivers on the AD lines, and fetches the requested data to be placed on the bus during clock cycle 3.
- It asserts DEVSEL# and maintains it in the asserted state until the end of the transaction.
- The C/BE# lines, which are used to send a bus command in clock cycle 1, are used for a different purpose during the rest of the transaction.
- Each of these four lines is associated with one byte on the AD lines.

- The initiator sets one or more of the C/BE# lines to indicate which byte lines are to be used for transferring data.
- Assuming that the target is capable of transferring 32bits at a time, all four C/BE# lines are asserted.
- During clock cycle 3, the initiator asserts the initiator ready signal, IRDY#, to indicate that it is ready to receive data.
- If the target has data ready to send at this time, it asserts target ready, TRDY#, and send a word of data.
- The initiator loads the data into its input buffer at the end of the clock cycle.
- The target send three more words of data in clock cycle 4 to 6.

- The initiator uses the FRAME# signal to indicate the duration of the burst.
- It negates this signal during the second last word of the transfer.
- Since it wishes to read four words, the initiator negates FRAME# during clock cycle 5, the cycle in which it receives the third word.
- After sending the fourth word in the clock cycle 6, the target disconnects its drivers and negates the DEVSEL# at the beginning of the clock cycle 7.

How PCI Works

- Plug and Play (PnP) means that you can connect a device or insert a card into your computer and it is automatically recognized and configured to work in your system.
- PnP BIOS - The core utility that enables PnP and detects PnP devices. The BIOS also reads the ESCD for configuration information on existing PnP devices.
- Extended System Configuration Data (ESCD) - A file that contains information about installed PnP devices.

- PnP operating system - Any operating system, such as Windows XP, that supports PnP.
- PnP automates several key tasks that were typically done either manually or with an installation utility provided by the hardware manufacturer. These tasks include the setting of:
- Interrupt requests (IRQ) - An IRQ, also known as a hardware interrupt, is used by the various parts of a computer to get the attention of the CPU. For example, the mouse sends an IRQ every time it is moved to let the CPU know that it's doing something. Before PCI, every hardware component needed a separate IRQ setting. But PCI manages hardware interrupts at the bus bridge, allowing it to use a single system IRQ for multiple PCI devices.
- Direct memory access (DMA) - This simply means that the device is configured to access system memory without consulting the CPU first.

- Memory addresses - Many devices are assigned a section of system memory for exclusive use by that device. This ensures that the hardware will have the needed resources to operate properly.
- Input/Output (I/O) configuration - This setting defines the ports used by the device for receiving and sending information.

Example

- You open up your computer's case and plug the sound card into an empty PCI slot on the motherboard.
- You close the computer's case and power up the computer.
- The system BIOS initiates the PnP BIOS.
- The PnP BIOS scans the PCI bus for hardware. It does this by sending out a signal to any device connected to the bus, asking the device who it is.
- The sound card responds by identifying itself. The device ID is sent back across the bus to the BIOS.

- The PnP BIOS checks the ESCD to see if the configuration data for the sound card is already present. Since the sound card was just installed, there is no existing ESCD record for it.
- The PnP BIOS assigns IRQ, DMA, memory address and I/O settings to the sound card and saves the data in the ESCD.
- Windows XP boots up. It checks the ESCD and the PCI bus. The operating system detects that the sound card is a new device and displays a small window telling you that Windows has found new hardware and is determining what it is.
- In many cases, Windows XP will identify the device, find and load the necessary drivers, and you'll be ready to go. If not, the "Found New Hardware Wizard" will open up. This will direct you to install drivers off of the disc that came with the sound card.
- Once the driver is installed, the device should be ready for use. Some devices may require that you restart the computer before you can use them. In our example, the sound card is immediately ready for use.