Embedded Systems Design: A Unified Hardware/Software Introduction

Chapter 6

Interfacing

Outline

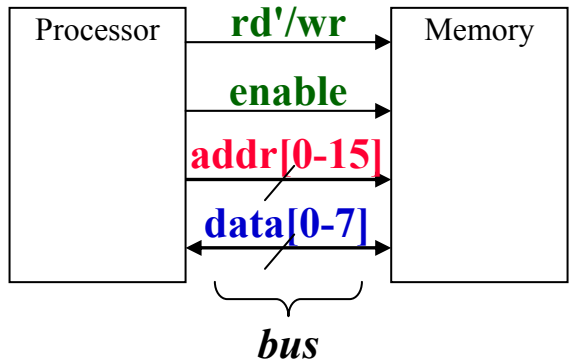
- Interfacing basics
- Microprocessor/Microcontroller interfacing
  - I/O Addressing
  - Interrupts
  - Direct memory access (DMA)
- Arbitration (decision/choice) techniques
- Hierarchical buses
- Protocols
  - Serial
  - Parallel
  - Wireless

Introduction

- Embedded system functionality aspects
  - **Processing**
    - Transformation of data
    - Implemented using **processors**
  - **Storage**
    - Retention of data
    - Implemented using **memory**
  - **Communication**
    - Transfer of data between processors, memories and peripherals
    - Implemented using **buses**
    - Called ***interfacing***

Interfacing Basics - A Simple Bus

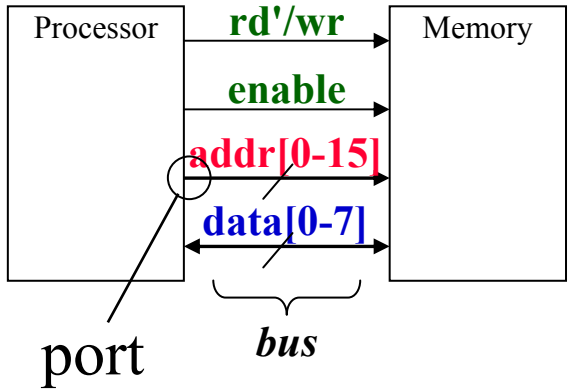
- Wires:
  - Uni-directional or bi-directional
  - One line may represent multiple wires
- Bus
  - Set of wires with a single function
    - **Address bus, data bus, control bus**
  - Or, entire collection of wires
    - Address, data and control
    - Associated protocol describes the rules for communicating over these wires



Bus Structure

Ports

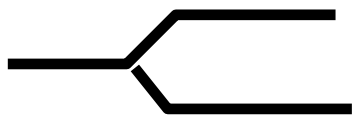
- Conducting device (like metal) on the periphery
- Connects bus to processor or memory
- Often referred to as a *pin*
  - Actual pins on periphery of IC package that plug into socket on printed-circuit board
  - Sometimes metallic balls instead of pins
  - Today, metal "pads" connecting processors and memories within single IC
- Distinction between a ***port*** and a ***bus*** is similar to the distinction between a ***house's driveway*** and a ***street***



Timing Diagrams

- Most common method for describing a communication protocol
- Control signal: low or high

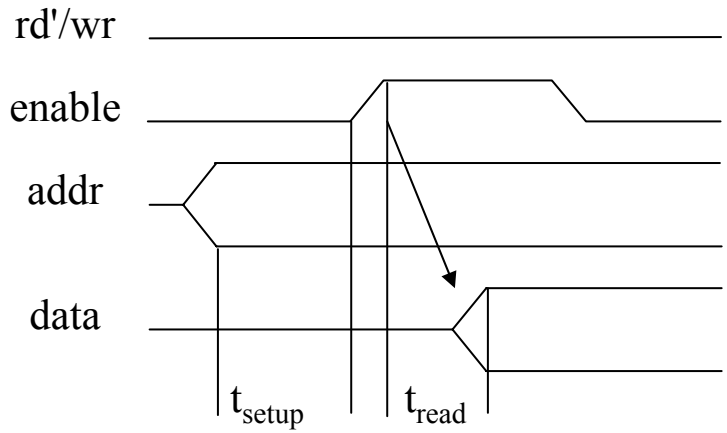
  - May be active low (e.g., go', /go, or go_L)
  - Or active high (e.g., go, or go_H)
  - Use terms *assert* (active) and *deassert* (deactive)
  - Asserting go' means go=0
- Data signal: not valid or valid



- Protocol may have sub-protocols
  - Called bus cycle, e.g., read protocol, write protocol
  - Each may be several clock cycles

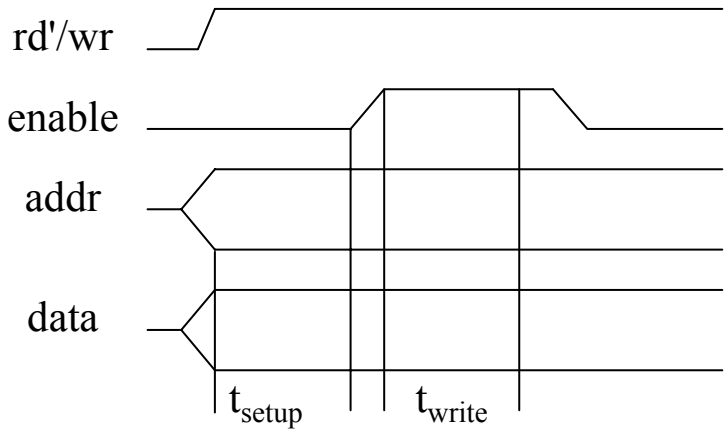
Timing Diagrams

- Read example
  - Processor sets *rd'/wr* low, places address on *addr* for at least t_{setup} time before *enable* asserted, enable triggers memory to place data on *data* wires by time t_{read}



read protocol

- Write example
  - Processor sets *rd'/wr* high, places address on *addr* and data on *data* before *enable* asserted, enable triggers memory to store data from *data* wires by time t_{write}



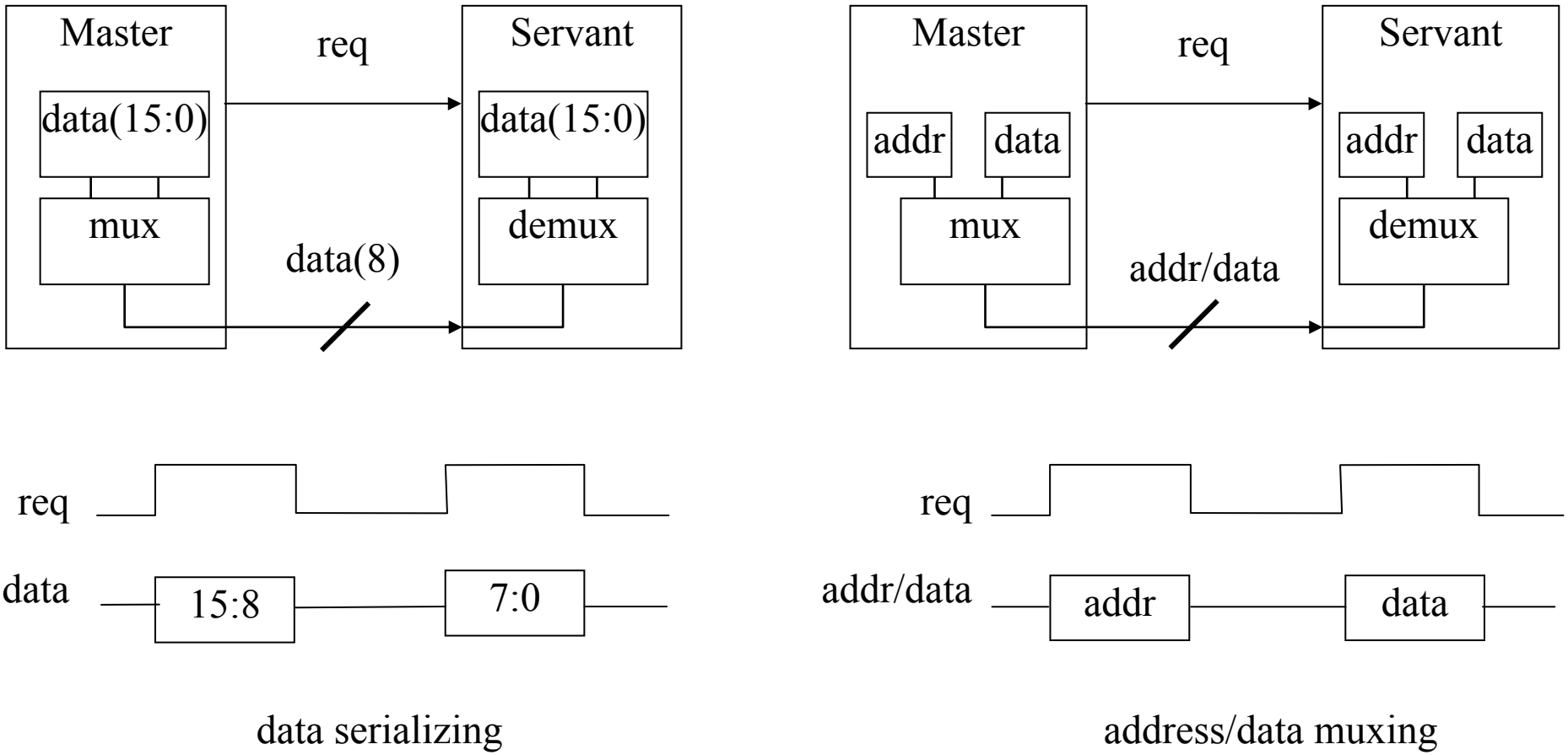
write protocol

Basic protocol concepts

- ***Actor*** is a processor or memory involved in the data transfer.
- A protocol typically involves two actors: a ***master*** initiates, and a ***servant*** (slave) responds
- Master usually processor and servant usually peripheral or memory or **could be another processor**
- Time multiplexing
  - Share a single set of wires for multiple pieces of data
  - Saves wires at the expense of time

Basic protocol concepts

Time-multiplexed data transfer

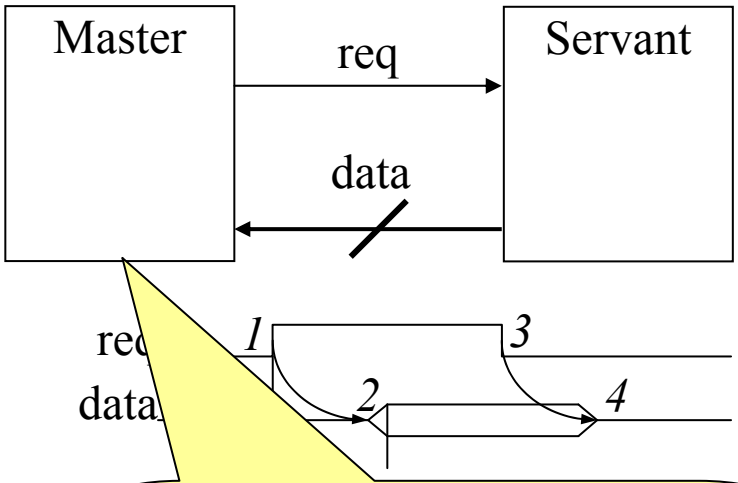


data serializing

address/data muxing

Basic protocol concepts: control methods

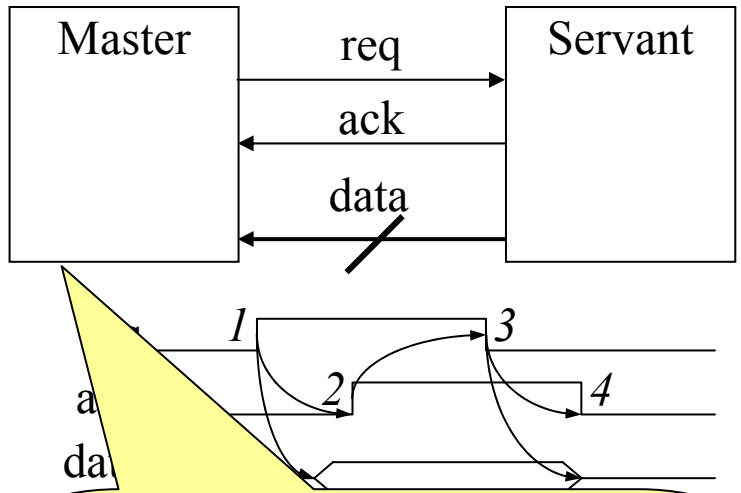
Strobe protocol



I want that report (the data) on my desk (the data bus) in an hour (t_{access}).

2. Servant puts data on bus within time $\mathbf{t_{access}}$

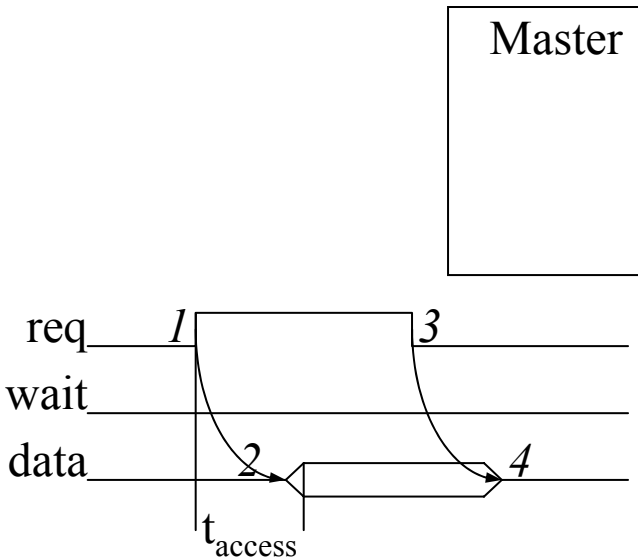
Handshake protocol



I want that report on my desk soon; let me know when it's ready.

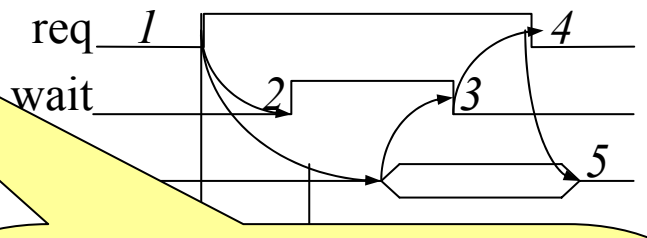
A strobe/handshake compromise

Fast-response case



1. Master asserts ***req*** to receive data
2. Servant puts data on bus within time $\mathbf{t_{access}}$ (wait line is unused)
3. Master receives data and deasserts ***req***
4. Servant ready for next request

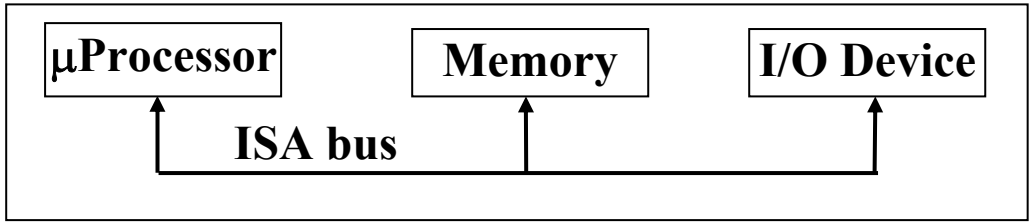
Slow-response case



1. I want that report on my desk
2. Servant can't put data within t_{access}; asserts ***wait***
3. by then, let me know that and
4. let me know when it's ready.
5. Servant ready for next request

ISA Bus Protocol – Memory Access

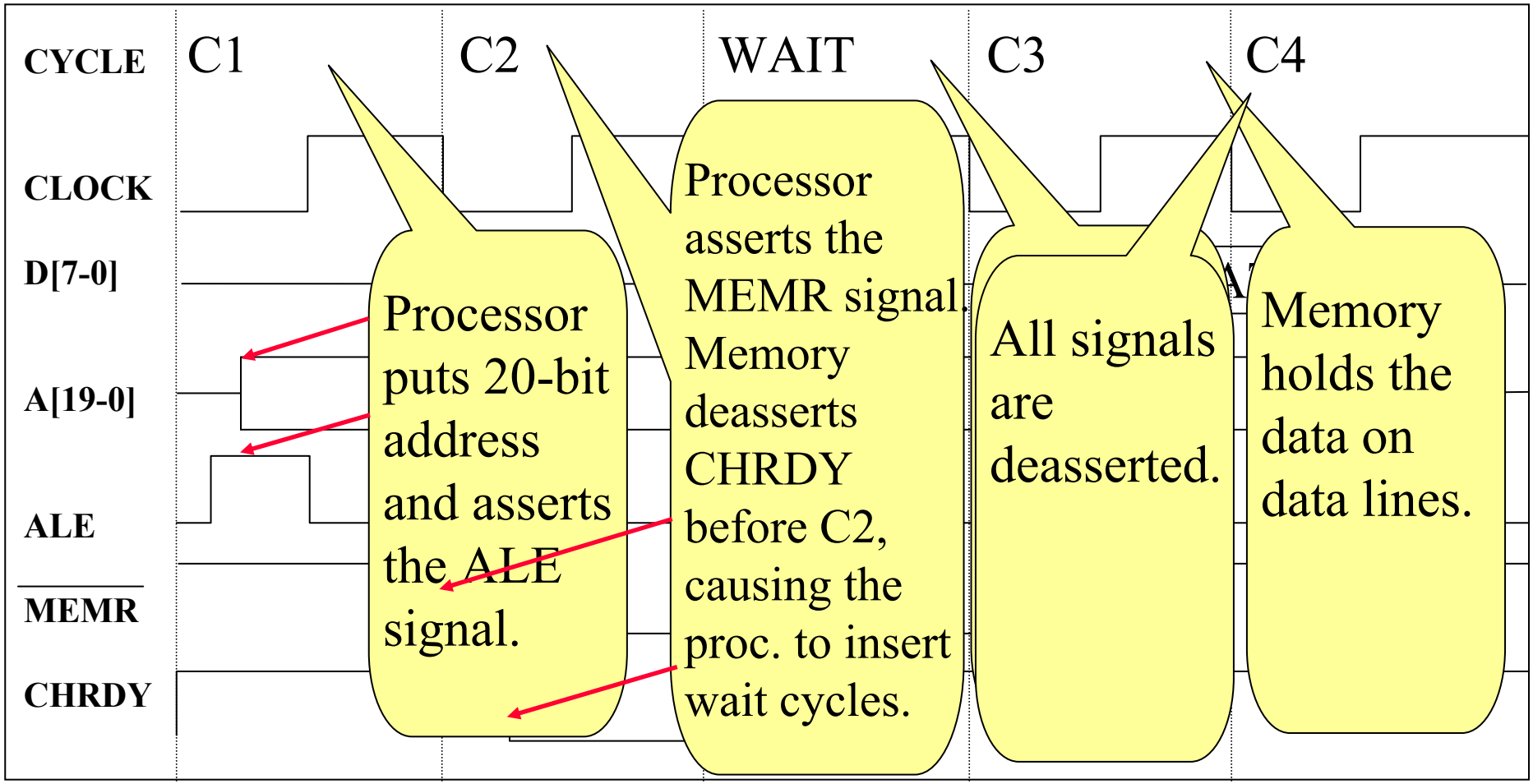
- ISA: Industry Standard Architecture



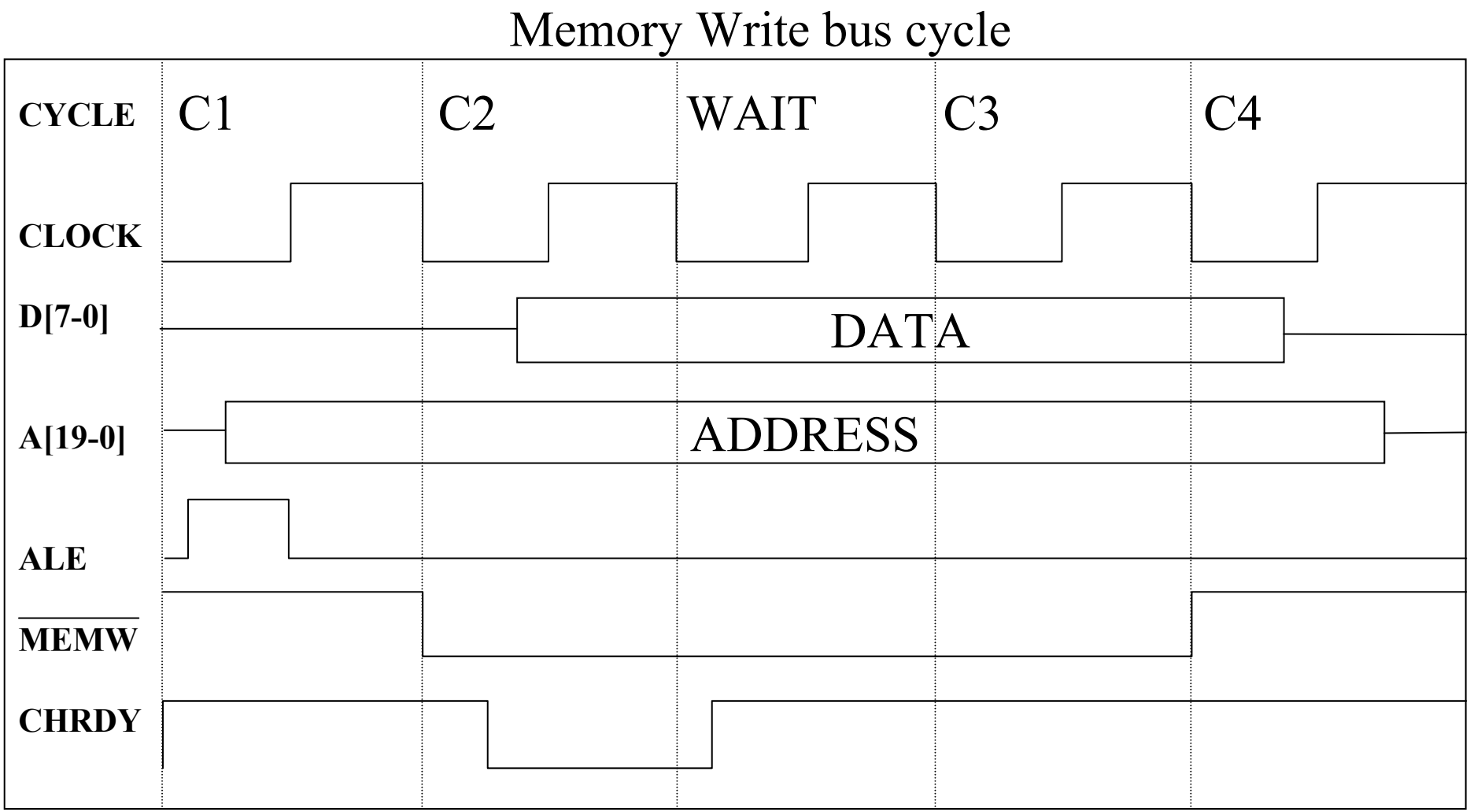
  - Common in 80x86's
  - 20-bit address
  - Compromise strobe/handshake control
  - 4 clock cycles default
  - Memory deasserted the CHRDY (channel ready signal) before the rising clock edge in 2nd clock cycle, causing the processor to insert wait cycles until CHRDY was reasserted
  - Up to six wait cycles can be inserted by a slow device

ISA Bus Protocol – Memory Access

Memory Read bus cycle



ISA Bus Protocol – Memory Access



Microprocessor interfacing: I/O addressing

- A microprocessor communicates with other devices using some of its pins
  - Port-based I/O (parallel I/O)
    - Processor has one or more N-bit ports
    - Processor's software reads and writes a port just like a register
  - Bus-based I/O
    - Processor has address, data and control ports that form a single bus
    - Communication protocol is built into the processor
    - A single instruction carries out the read or write protocol on the bus

Types of bus-based I/O: memory-mapped I/O and standard I/O

- Processor talks to both memory and peripherals using same bus – two ways to talk to peripherals
  - Memory-mapped I/O
    - Peripheral registers occupy addresses in same address space as memory
    - e.g., Bus has 16-bit address
      - lower 32K addresses may correspond to memory
      - upper 32k addresses may correspond to peripherals
  - Standard I/O (I/O-mapped I/O)
    - Additional pin (*M/IO*) on bus indicates whether a memory or peripheral access
    - e.g., Bus has 16-bit address
      - all 64K addresses correspond to memory when *M/IO* set to 0
      - all 64K addresses correspond to peripherals when *M/IO* set to 1

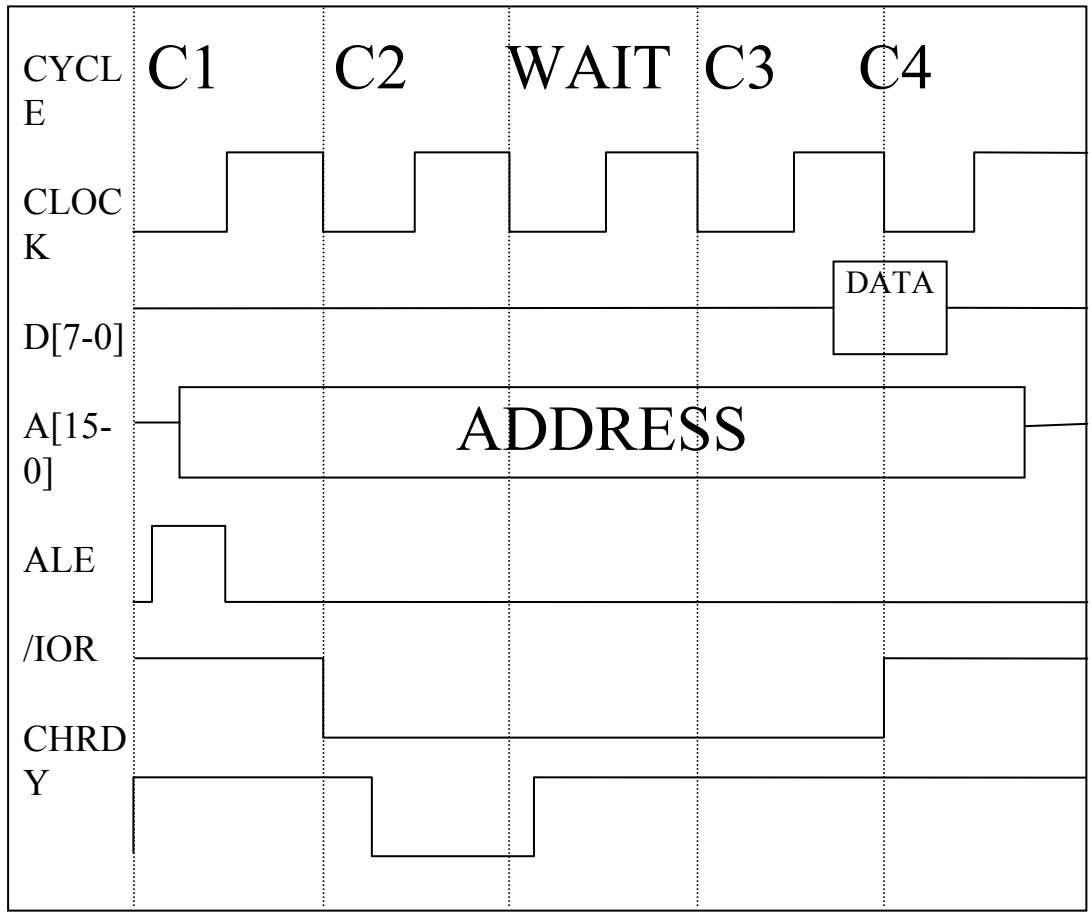
Memory-mapped I/O vs. Standard I/O

- Memory-mapped I/O
  - Requires no special instructions
    - Assembly instructions involving memory like MOV and ADD work with peripherals as well
    - Standard I/O requires special instructions (e.g., IN, OUT) to move data between peripheral registers and memory
- Standard I/O
  - No loss of memory addresses to peripherals
  - Simpler address decoding logic in peripherals possible
    - When number of peripherals much smaller than address space then high-order address bits can be ignored
      - smaller and/or faster comparators

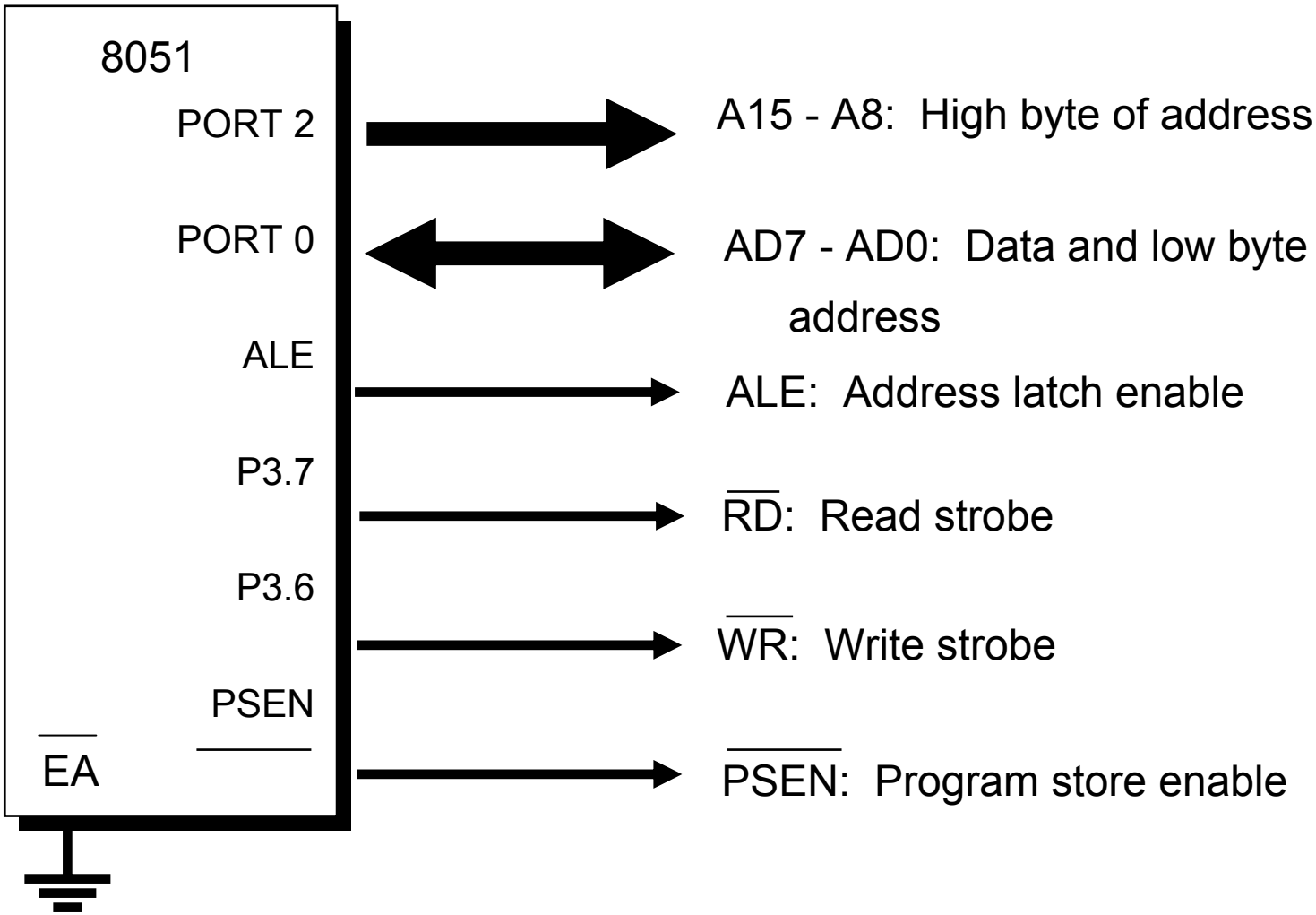
ISA bus

- ISA supports standard I/O
  - /IOR distinct from /MEMR for peripheral read
    - /IOW used for writes
  - 16-bit address space for I/O vs. 20-bit address space for memory
  - Otherwise very similar to memory protocol

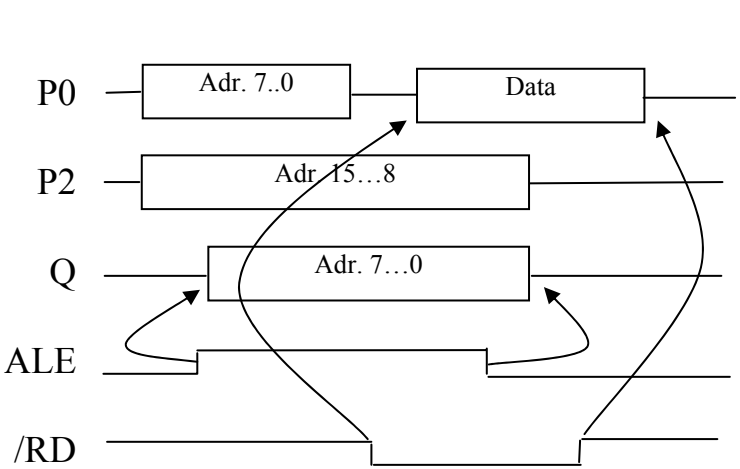
ISA I/O bus read protocol

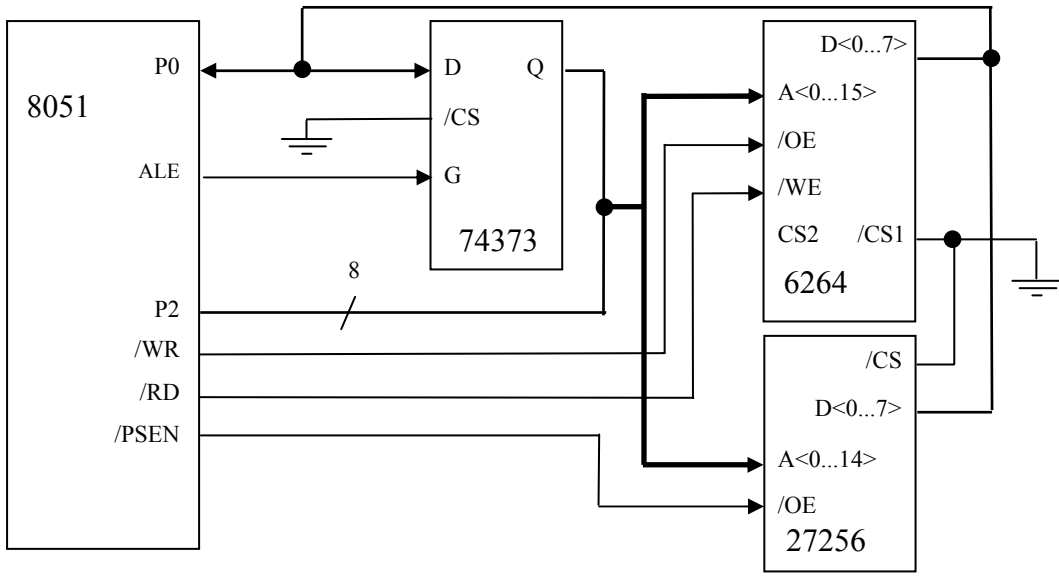


External Bus Expansion of 8051



A basic memory protocol

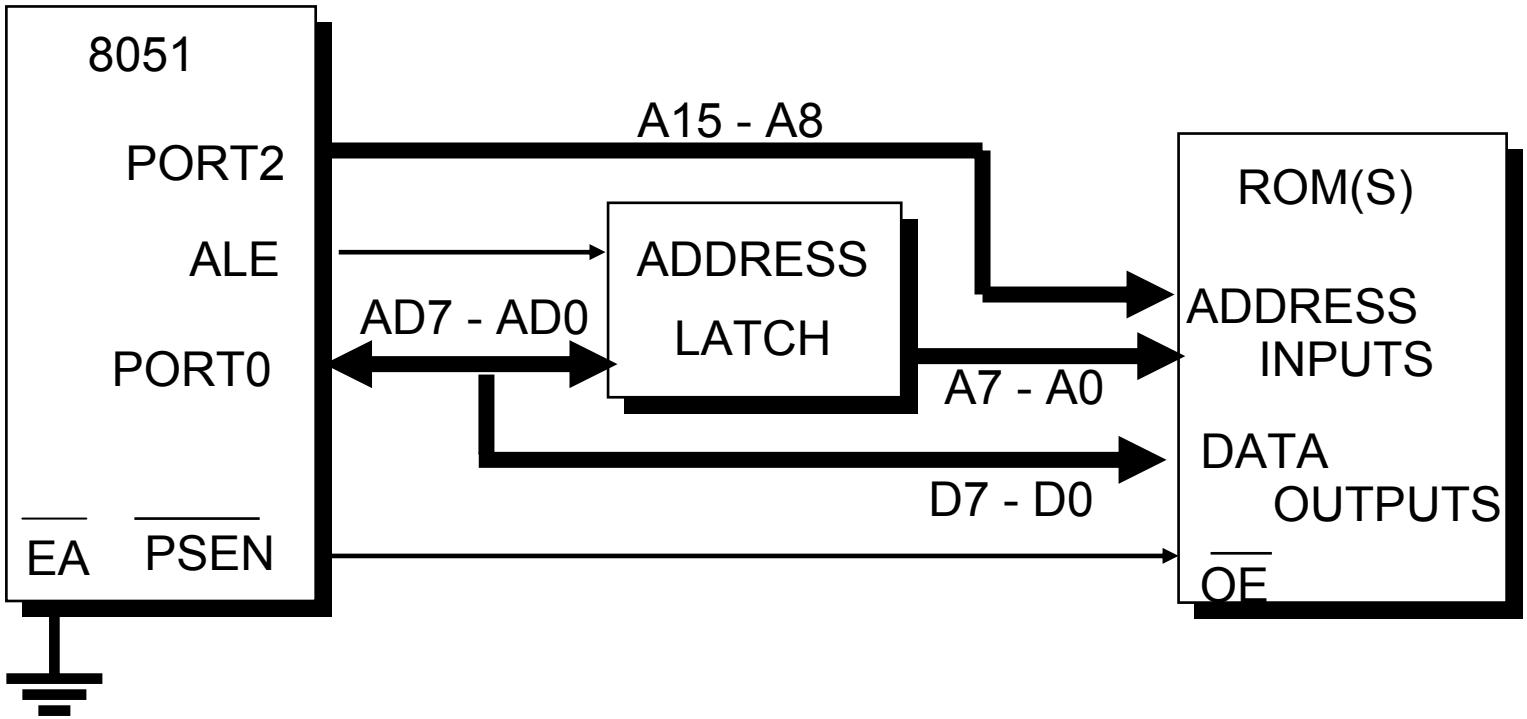




- Interfacing an 8051 to external memory
  - Ports P0 and P2 support port-based I/O when 8051 internal memory being used
  - Those ports serve as data/address buses when external memory is being used
  - 16-bit address and 8-bit data are time multiplexed; low 8-bits of address must therefore be latched with aid of ALE signal

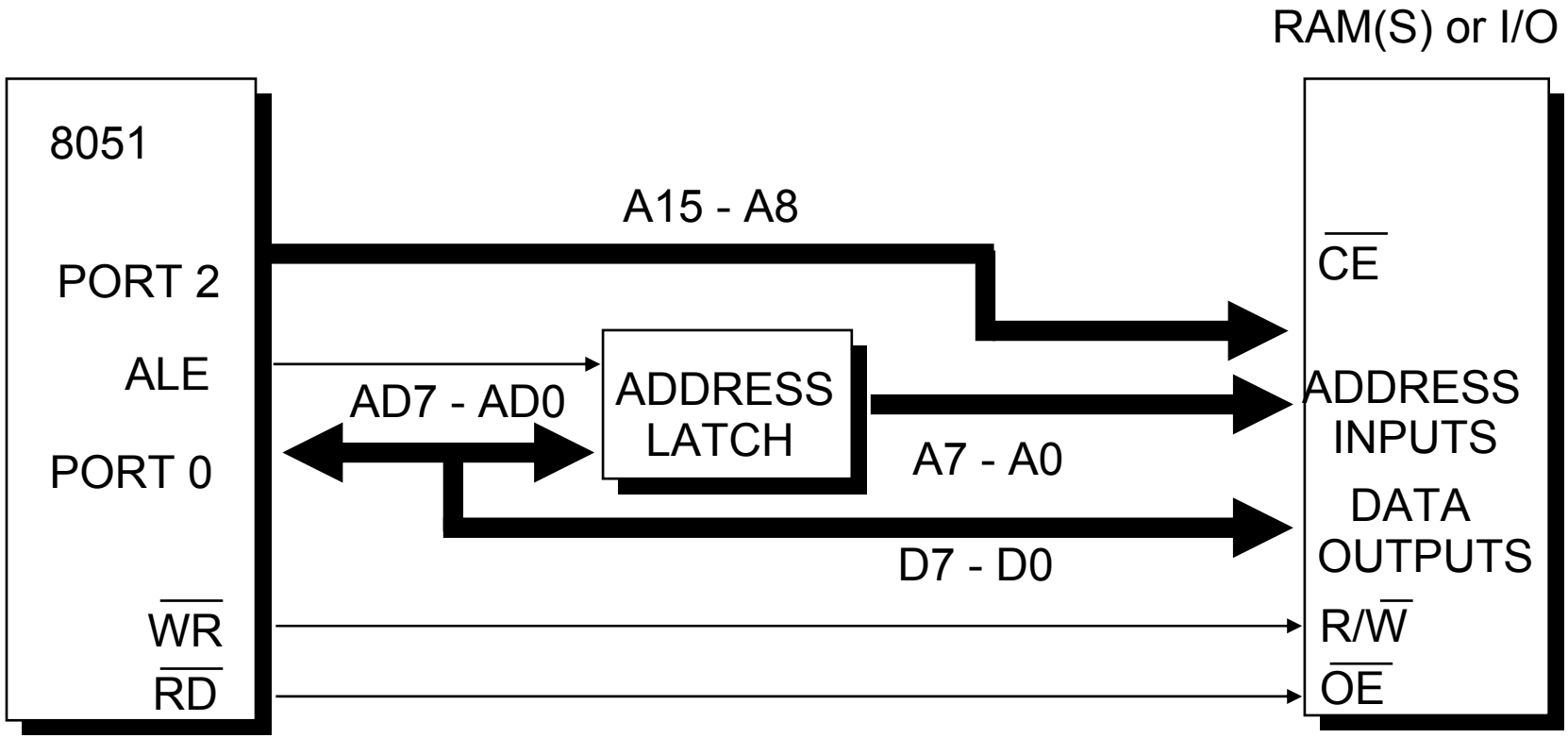
External Program Memory

- 64K byte address space.
- Enabled by PSEN signal.



External Data Memory

- 64K byte address space.
- Enabled by RD and WR signals.



Microprocessor interfacing: interrupts

- Suppose a peripheral intermittently receives data, which must be serviced by the processor
  - The processor can *poll* the peripheral regularly to see if data has arrived – wasteful
  - The peripheral can *interrupt* the processor when it has data
- Requires an extra pin or pins: Int
  - If Int is 1, processor suspends current program, jumps to an Interrupt Service Routine, or ISR
  - Known as interrupt-driven I/O
  - Essentially, "polling" of the interrupt pin is built-into the hardware, so no extra time!

Processing Interrupts (in 8051)

When an interrupt occurs and is accepted by CPU, the following actions occur:

- –Current instruction's complete execution
- –PC is saved on the stack
- –PC is loaded with the vector address of the ISR
- –ISR executes and takes action in response to interrupt
- –ISR finishes with a RETI instruction
- –PC is loaded with its old value from the stack
- –Execution of main program continues where it left off

Interrupt Vector Addresses

Interrupt	Flag	Vector Address	SFR & Bit Position
System Reset	RST	0000H	
External 0	IE0	0003H	TCON.1
Timer 0	TF0	000BH	TCON.5
External 1	IE1	0013H	TCON.3
Timer 1	TF1	001BH	TCON.7
Serial Port	RI or TI	0023H	SCON.0 or SCON.1

Direct memory access (DMA)

- Data being accumulated in a peripheral should be first stored in memory before being processed by a program running on the processor. E.g., packet data from an Ethernet card.
- This temporary storage called ***buffering***
- Microprocessor could handle this with an ISR
  - ISR would simply transfer data from peripheral to memory – then resuming its application
  - Storing and restoring its state results in inefficiency
- DMA controller eliminates these inefficiencies

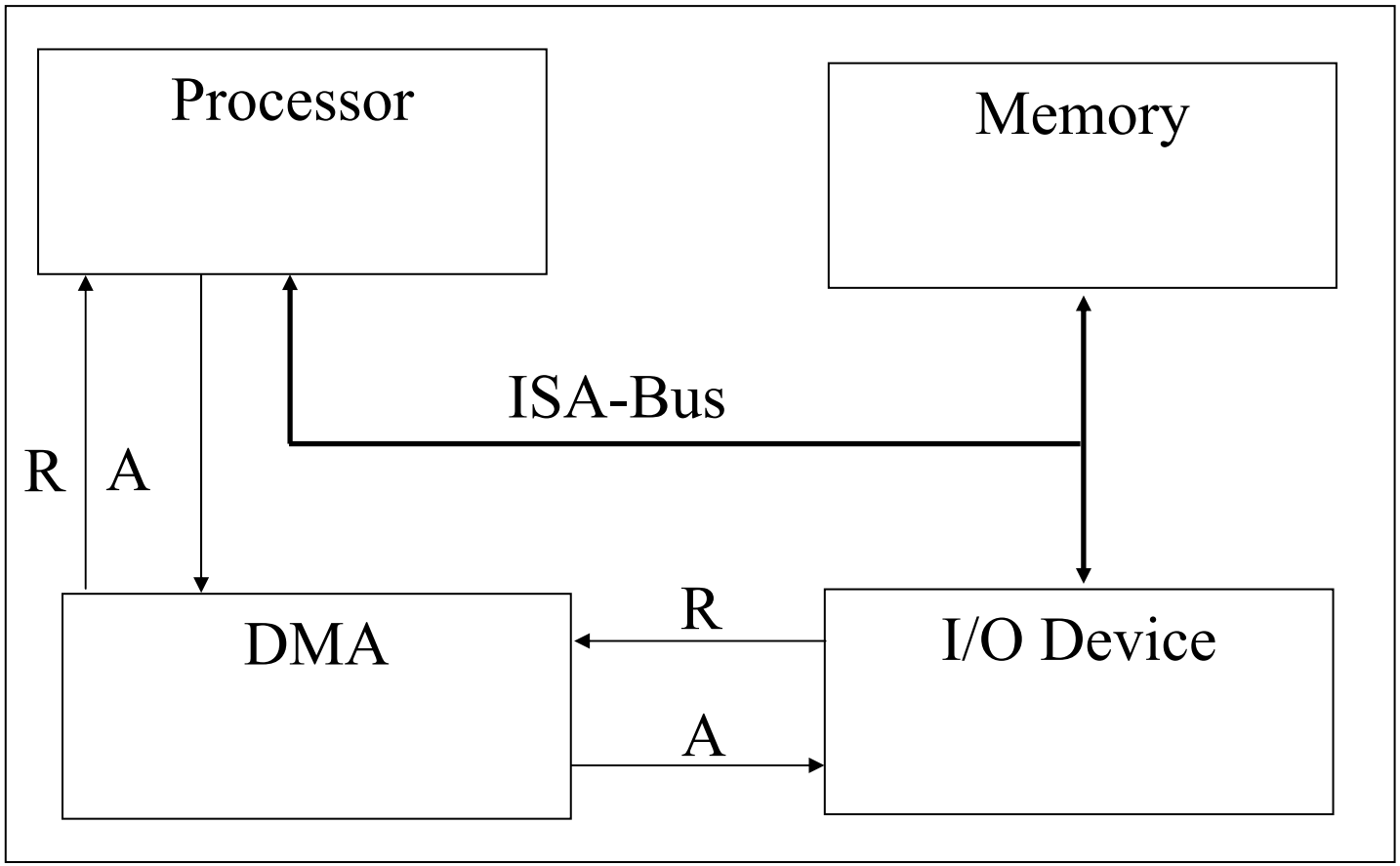
Direct memory access (DMA)

- DMA controller is more efficient because it is a separate single-purpose processor whose sole purpose is to transfer data between memories and peripherals
  - Peripheral requests servicing from DMA controller, which then requests control of the system bus from the processor by hardware (a pin - ***Dreq***)
  - Microprocessor gives control of the system bus to DMA controller (another pin – ***Dack***)
  - Microprocessor can meanwhile execute its regular program
    - No inefficient storing and restoring state due to ISR call
    - Regular program need not wait unless it requires the system bus

Direct memory access (DMA)

- Microprocessors which supports DMA are specially designed with these two pins.
- DMA controllers also connects to all the system bus signals i.e., address, data, and control lines.
- DMA controllers are configured to know addresses of all the peripherals and memory by a routine running on the processor during system initialization.
- In an embedded system addresses are guaranteed not to change, so can be hardcode directly into DMA controller.
- DMA controllers might transfer just one piece of data or (commonly) transfer numerous pieces of data (***block***), one right after the other.
- Usually used for CD-ROM players or disk controllers.
- Can control lots of peripherals like 128 by one DMA.

Block Diagram of DMA System



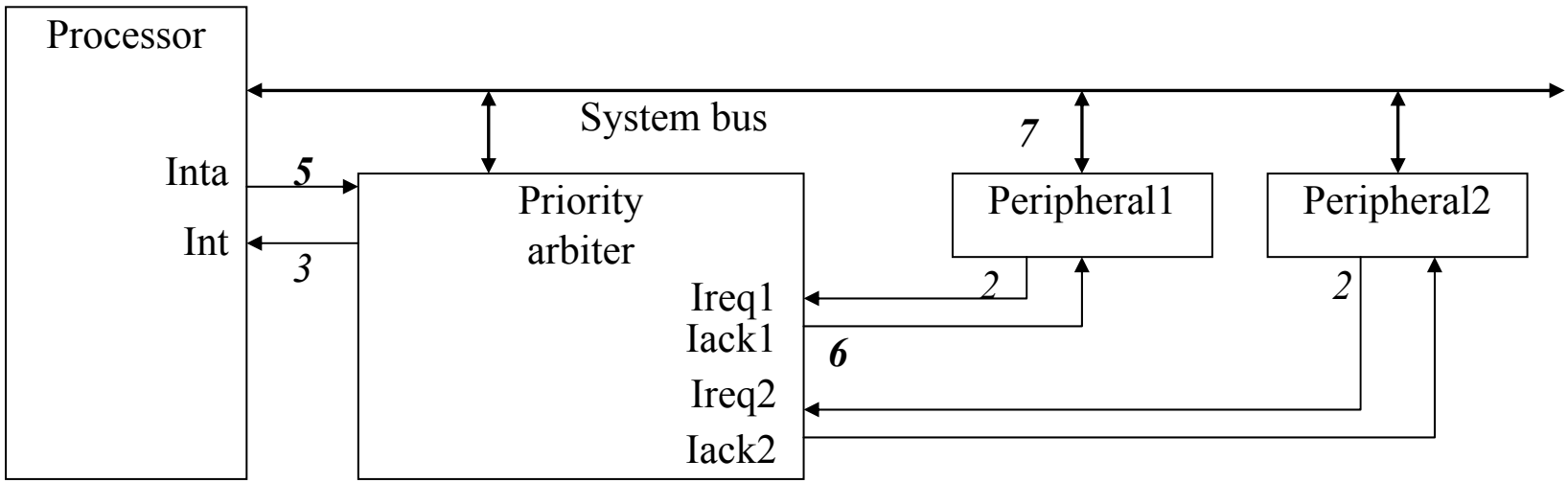
Arbitration

- Consider the situation where multiple peripherals request service from single resource (e.g., microprocessor, DMA controller) simultaneously - which gets serviced first and which should wait? Several methods

1. Priority arbiter

  - Single-purpose processor
  - Peripherals make requests to arbiter, arbiter makes requests to resource
  - Arbiter connected to system bus for configuration only

Arbitration using a priority arbiter



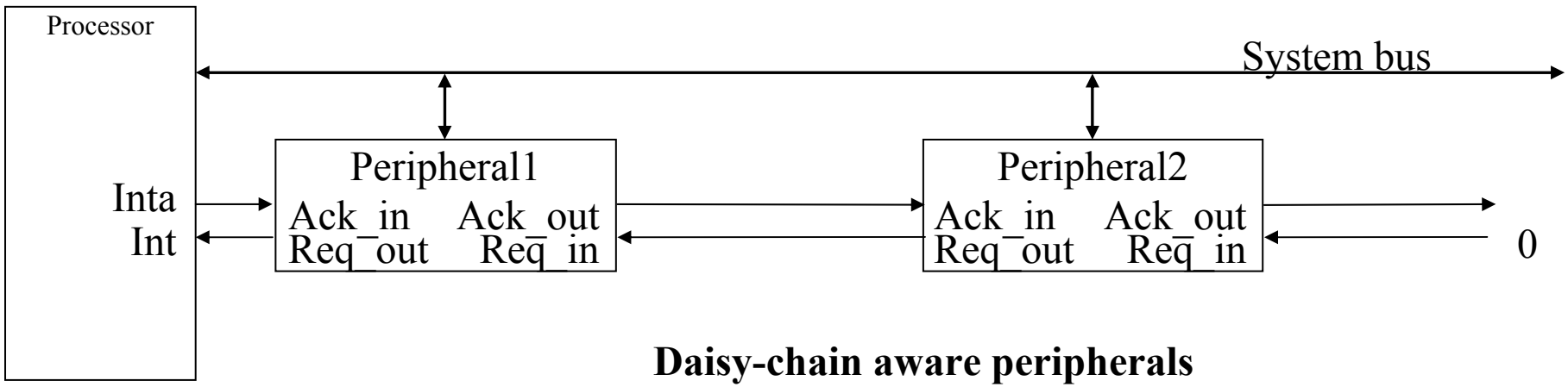
1. Processor is executing its program.
2. Peripheral1 needs servicing so asserts ***Ireq1***. Peripheral2 also needs servicing so asserts ***Ireq2***.
3. Priority arbiter sees at least one ***Ireq*** input asserted, so asserts ***Int***.
4. Processor stops executing its program and stores its state.
5. Processor asserts ***Inta***.
6. Priority arbiter asserts ***Iack1*** to acknowledge Peripheral1.
7. Peripheral1 puts its interrupt address vector on the system bus
8. Processor jumps to the address of ISR read from data bus, ISR executes and returns (and completes handshake with arbiter).
9. Processor resumes executing its program.

Arbitration: 1. Priority arbiter

- Types of priority
  - Fixed priority
    - each peripheral has unique rank e.g., 1, 2, 3, or 4
    - highest rank chosen first with simultaneous requests
    - It is preferred when clear difference in rank between peripherals
  - Rotating priority (round-robin) - more complex
    - priority changed based on history of servicing
    - better distribution of servicing especially among peripherals with similar priority demands
- Also called *interrupt controller*

Arbitration: 2. Daisy-chain arbitration

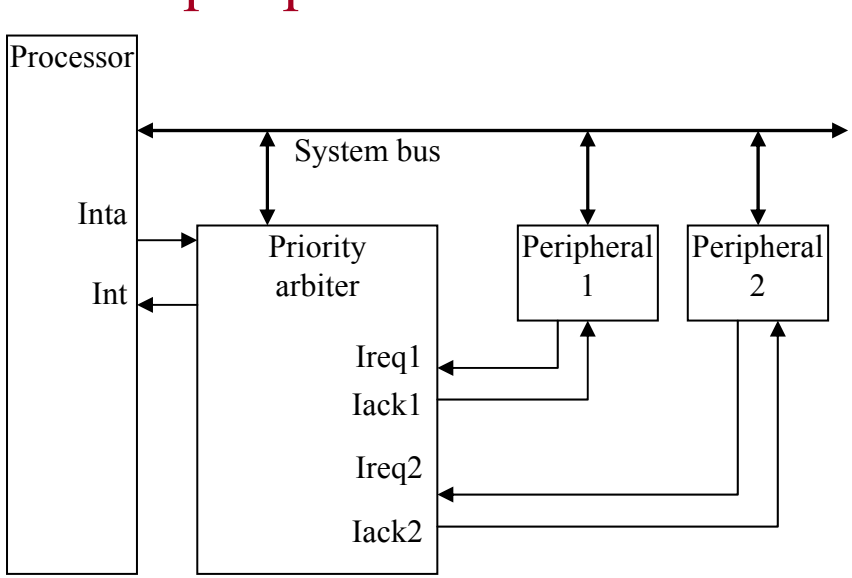
- Arbitration done by peripherals
  - Built into peripheral or external logic added
    - ***req*** input and ***ack*** output added to each peripheral
- Peripherals connected to each other in daisy-chain manner
  - One peripheral connected to resource, all others connected "upstream"
  - Peripheral's ***req*** flows "downstream" to resource, resource's ***ack*** flows "upstream" to requesting peripheral
  - Closest peripheral has highest priority, peripheral at the end of chain has lowest

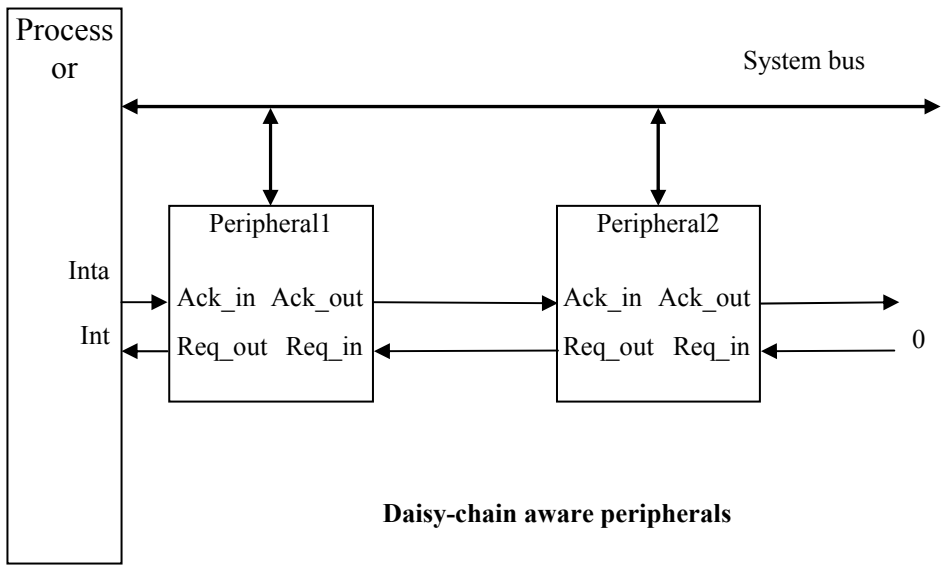


Daisy-chain aware peripherals

Arbitration: Daisy-chain arbitration

- Pros/cons
  - Easy to add/remove peripheral - no system redesign needed
  - Does not support rotating priority
  - One broken peripheral can cause loss of access to other peripherals

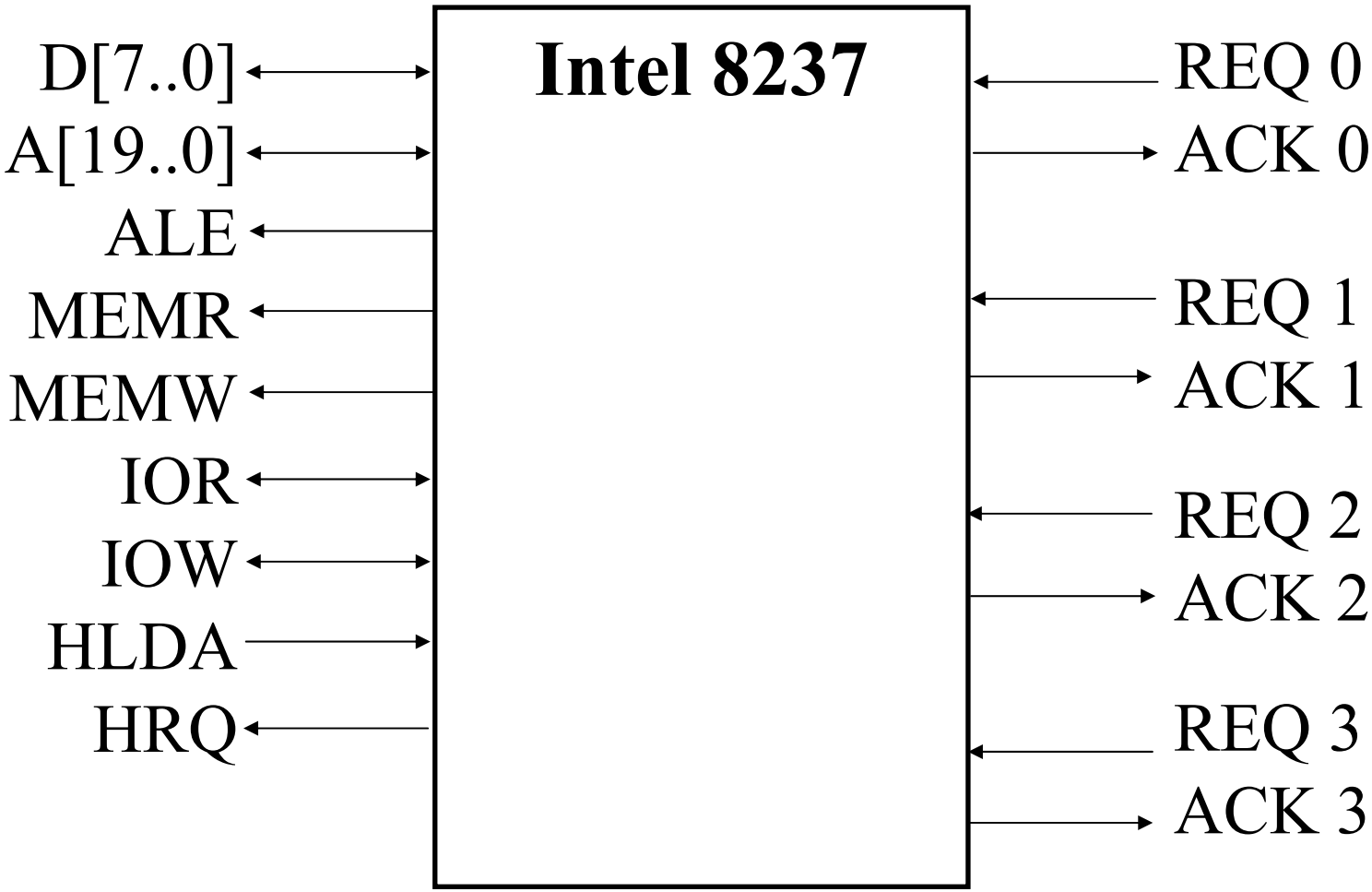




Arbitration: 3. Network-oriented arbitration

- Many embedded systems contain multiple processors communicate via a shared bus (sometimes called a network)
  - Arbitration in such cases typically built into bus protocol
  - Separate processors may try to write simultaneously causing collisions
    - Data must be resent
    - Protocols ensure not to start sending again at same time
      - statistical methods can be used to reduce chances
- Typically used for connecting multiple distant chips

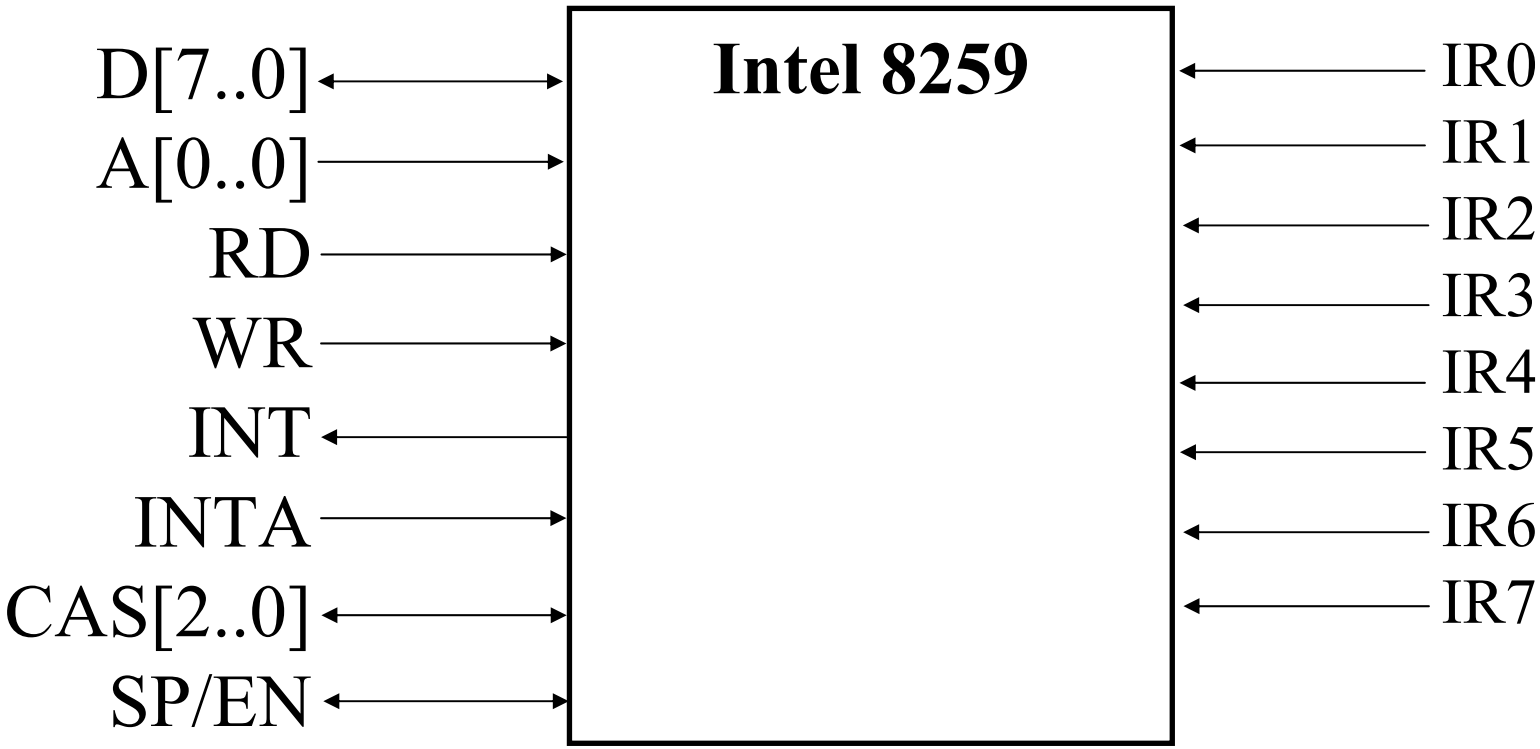
Intel 8237 DMA controller



Intel 8237 DMA controller

Signal	Description
D[7..0]	These wires are connected to the system bus (ISA) and are used by the microprocessor to write to the internal registers of the 8237.
A[19..0]	These wires are connected to the system bus (ISA) and are used by the DMA to issue the memory location where the transferred data is to be written to. The 8237 is
ALE*	This is the address latch enable signal. The 8237 use this signal when driving the system bus (ISA).
MEMR*	This is the memory write signal issued by the 8237 when driving the system bus (ISA).
MEMW*	This is the memory read signal issued by the 8237 when driving the system bus (ISA).
IOR*	This is the I/O device read signal issued by the 8237 when driving the system bus (ISA) in order to read a byte from an I/O device
IOW*	This is the I/O device write signal issued by the 8237 when driving the system bus (ISA) in order to write a byte to an I/O device.
HLDA	This signal (hold acknowledge) is asserted by the microprocessor to signal that it has relinquished the system bus (ISA).
HRQ	This signal (hold request) is asserted by the 8237 to signal to the microprocessor a request to relinquish the system bus (ISA).
REQ 0,1,2,3	An attached device to one of these channels asserts this signal to request a DMA transfer.
ACK 0,1,2,3	The 8237 asserts this signal to grant a DMA transfer to an attached device to one of these channels.
*See the ISA bus description in this chapter for complete details.	

Intel 8259 programmable priority controller



Intel 8259 programmable priority controller

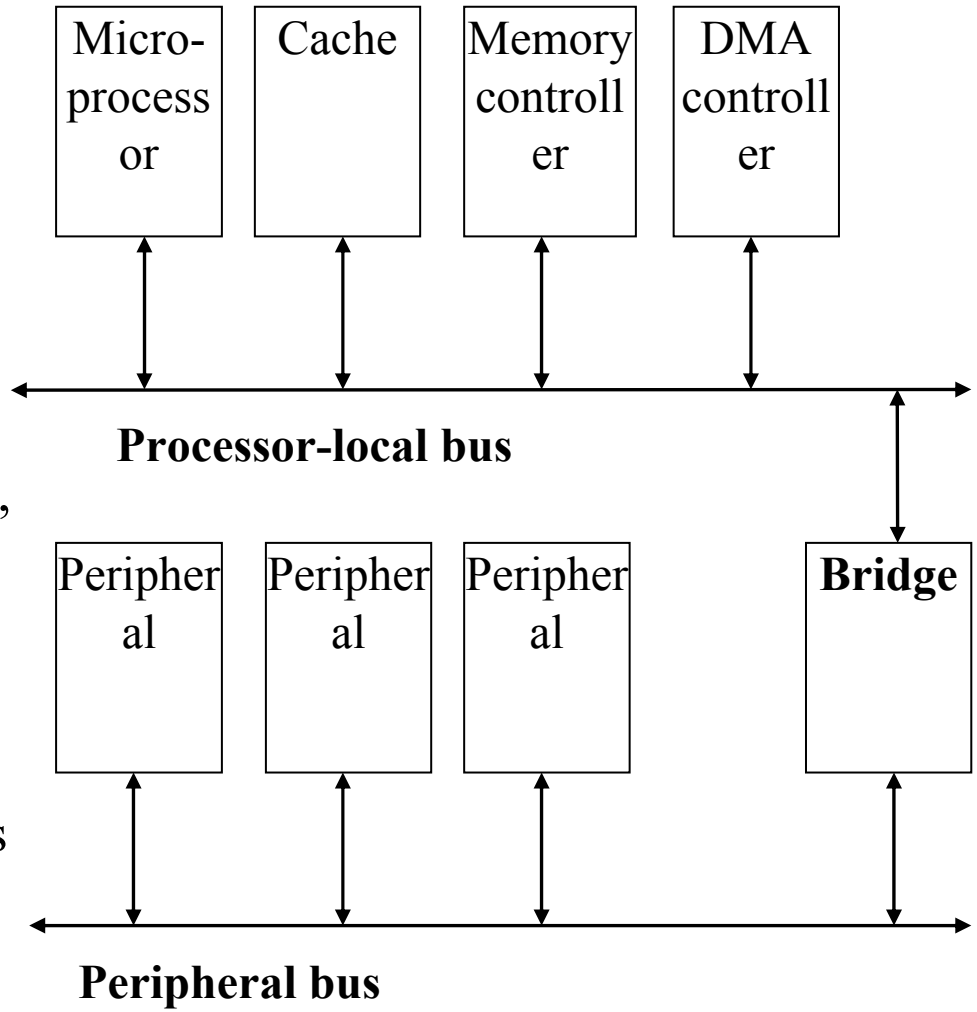
Signal	Description
D[7..0]	These wires are connected to the system bus and are used by the microprocessor to write or read the internal registers of the 8259.
A[0..0]	This pin actis in cunjunction with WR/RD signals. It is used by the 8259 to decipher various command words the microprocessor writes and status the microprocessor wishes to read.
WR	When this write signal is asserted, the 8259 accepts the command on the data line, i.e., the microprocessor writes to the 8259 by placing a command on the data lines and asserting this signal.
RD	When this read signal is asserted, the 8259 provides on the data lines its status, i.e., the microprocessor reads the status of the 8259 by asserting this signal and reading the data lines.
INT	This signal is asserted whenever a valid interrupt request is received by the 8259, i.e., it is used to interrupt the microprocessor.
INTA	This signal, is used to enable 8259 interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the microprocessor.
IR 0,1,2,3,4,5,6,7	An interrupt request is executed by a peripheral device when one of these signals is asserted.
CAS[2..0]	These are cascade signals to enable multiple 8259 chips to be chained together.
SP/EN	This function is used in conjunction with the CAS signals for cascading purposes.

Multilevel bus architectures

- Embedded systems have numerous types of communications varying in their frequencies and speed requirements.
- The frequent and high speed communications will likely be between the processor and its memories.
- Less frequent communications, requiring less speed, will be between the processor and its peripherals.
- A single high speed bus requires each peripheral to have a high speed bus interface. Also having too many peripherals on the same bus will slow down the bus.
- Two-level bus is one of the solutions.

Multilevel bus architectures

- Processor-local bus
  - High speed, wider (as wide as a memory word), most frequent communication
  - Connects microprocessor, cache, memory controllers, etc.
- Peripheral bus
  - Lower speed, narrower (few pins), less frequent communication
  - Typically industry standard bus (ISA, PCI) for portability
- Bridge
  - Single-purpose processor converts communication between buses
- Three-level is also in practice in more complex systems- for coprocessors



Advanced communication principles

- Until now - Interfacing methods to interconnect components within an IC via on-chip buses, or to interconnect ICs via on-board buses.
- Now communication from a more abstract point of view.
- Communication can take place over a number of different types of media, e.g., single wire, set of wires, radio waves, infrared waves, etc.
- Layering
  - Break complexity of communication protocol into pieces easier to design and understand
  - Lower levels provide services to higher level
    - Lower level might work with bits while higher level might work with packets of data
  - Physical layer
    - Lowest level in hierarchy
    - Medium to carry data from one actor (device or node) to another

Advanced communication principles

- Three basic types of communication
- **Parallel communication**
  - Physical layer capable of transporting multiple bits of data
- **Serial communication**
  - Physical layer transports one bit of data at a time
- **Wireless communication**
  - No physical connection needed for transport at physical layer

Parallel communication

- Multiple data, control, and possibly power wires
  - One bit per wire
- High data throughput with short distances
- Typically used when connecting devices on same IC or same circuit board
  - Bus must be kept short
    - long parallel wires result in high capacitance values which requires more time to charge/discharge
    - Data misalignment between wires increases as length increases
- Higher cost, bulky
  - Insulation must be used to prevent noise from each wire from interfacing with the other wires.
  - A 32-wire cable connecting two devices together will cost much more than a two-wire cable.

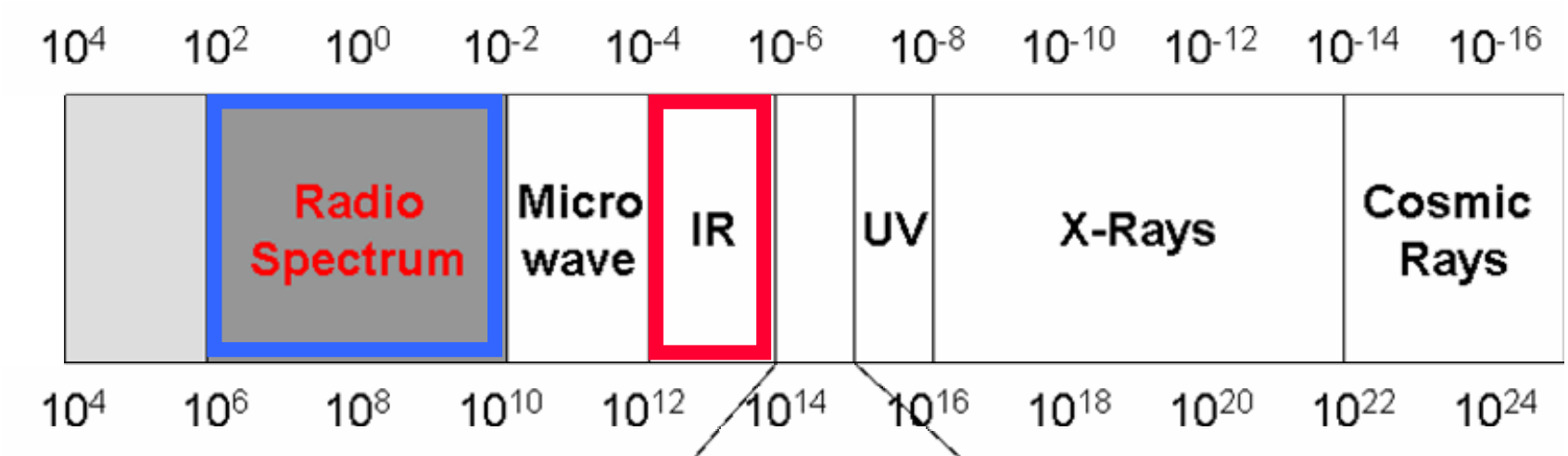
Serial communication

- Single data wire, possibly also control and power wires
- Words transmitted one bit at a time
- Higher data throughput with long distances
  - Less average capacitance, so more bits per unit of time
- Cheaper, less bulky
- More complex interfacing logic and communication protocol
  - Sender needs to decompose word into bits
  - Receiver needs to recompose bits into word
  - Control signals often sent on same wire as data (start, stop, parity bits etc.) - increasing protocol complexity

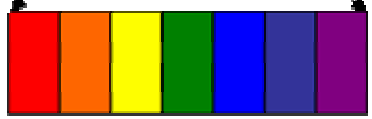
Wireless communication

- **Infrared (IR)**
  - Uses electronic wave frequencies just below visible light spectrum
  - Diode emits infrared light to generate signal
  - Infrared transistor detects signal, conducts when exposed to infrared light
  - Cheap to build transmitter and receiver circuits
  - Need line of sight, limited range
- **Radio frequency (RF)**
  - Uses electromagnetic wave frequencies in radio spectrum
  - Analog circuitry and antenna needed on both sides of transmission
  - Line of sight not needed, transmitter power determines range

Radio Spectrum



1MHz ==100m
100MHz ==1m
10GHz ==1cm



Visible light

< 30 KHz	VLF
30-300KHz	LF
300KHz – 3MHz	MF
3 MHz – 30MHz	HF
30MHz – 300MHz	VHF
300 MHz – 3GHz	UHF
3-30GHz	SHF
> 30 GHz	EHF

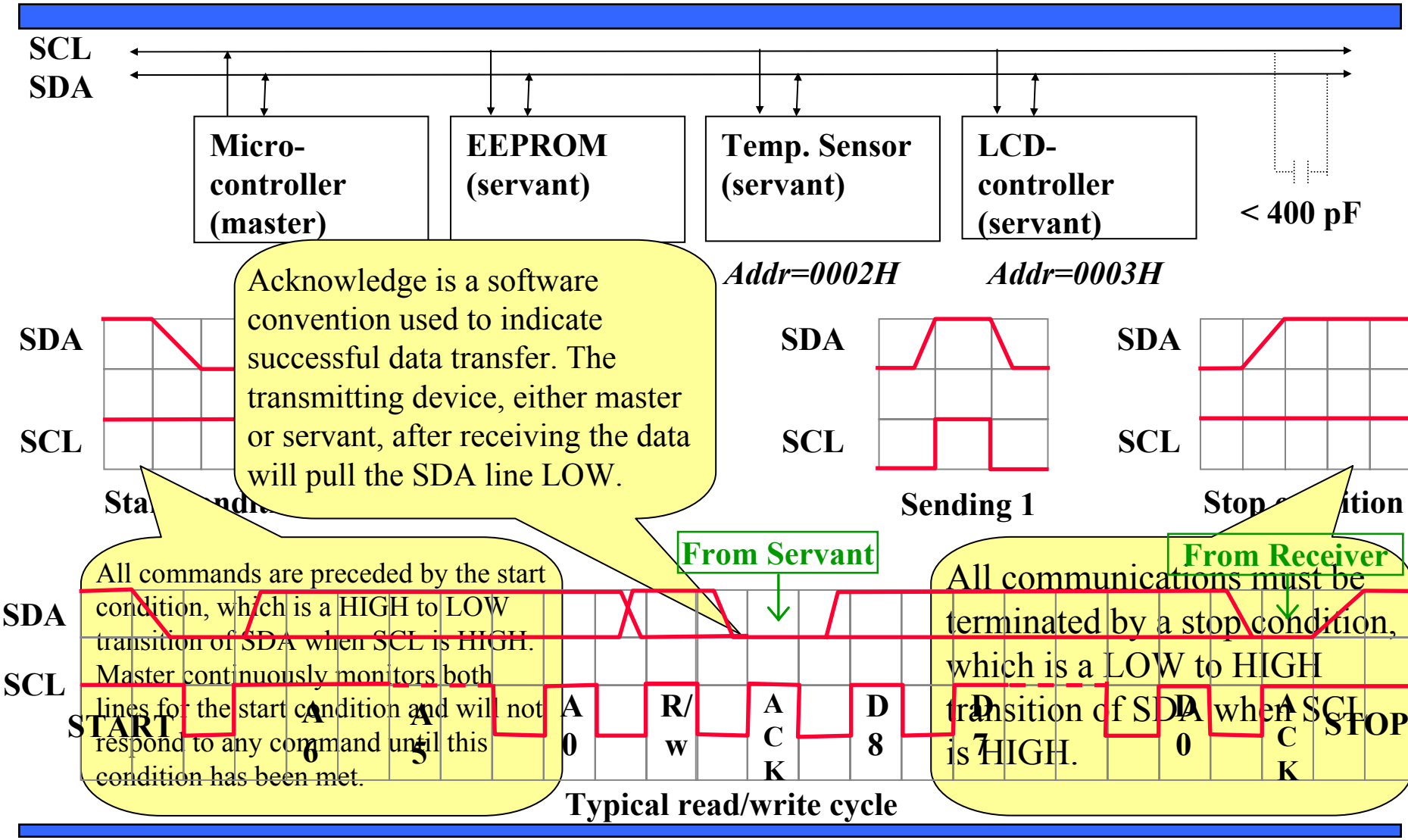
Error detection and correction

- Often part of bus protocol
- Error detection: ability of receiver to detect errors during transmission
- Error correction: ability of receiver and transmitter to cooperate to correct problem
  - Typically done by acknowledgement/retransmission protocol
- Bit error: single bit is inverted
- Burst of bit error: consecutive bits received incorrectly
- **Parity bit:** extra bit sent with word used for error detection
  - Odd parity: data word plus parity bit contains odd number of 1's
  - Even parity: data word plus parity bit contains even number of 1's
  - Always detects single bit errors, but not all burst bit errors
- **Checksum word:** extra word sent with data packet of multiple words
  - e.g., extra word contains XOR sum of all data words in packet. E.g., A packet consists of 4 words: 0000000, 0101010, 1010101, and 0000000. XOR checksum of these words 1111111

Serial protocols: I^2C

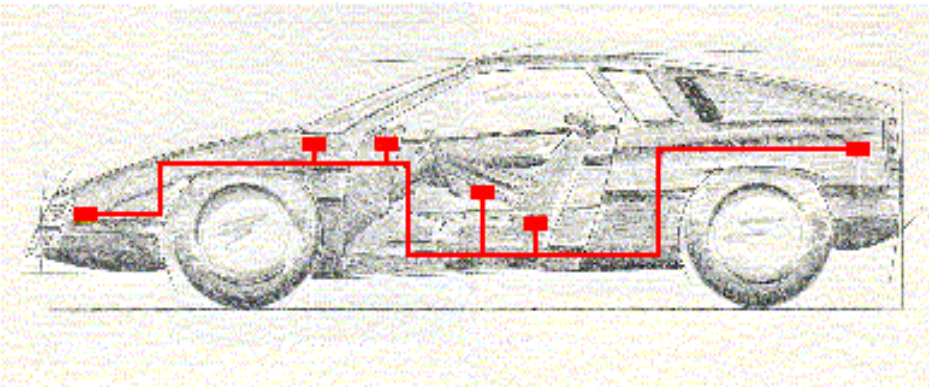
- **I^2C (Inter-IC)**
  - Bi-directional, two-wire serial bus protocol developed by Philips semiconductors nearly 20 years ago - extensively used in a variety of microcontroller-based professional, consumer applications such as televisions, VCRs, and audio equipment.
  - Good for embedded systems having mixed voltage environment (devices that operate under different voltages in a single system)-has automatic shifting circuit.
  - Enables peripheral ICs to communicate using simple communication hardware
  - Three data transfer speed modes for the I2C bus:
    - **Standard mode** is 100 Kbps.
    - **Fast mode** is 400 Kbps
    - **High-speed mode** supports speeds up to 3.4 Mbps.
    - All are backward compatible.
    - The I^2C bus supports 7-bit and 10-bit address space.

I²C bus structure



Serial protocols: CAN

- CAN (Controller Area Network)
  - Protocol for real-time applications
  - Developed by Robert Bosch GmbH
  - Originally for communication among components of cars
  - Applications now using CAN include:
    - elevator controllers, copiers, telescopes, production-line control systems, and medical instruments
  - Data transfer rates up to 1 Mbit/s and 11-bit addressing



Serial protocols: CAN

- It uses CSMA/CD+AMP (Carrier Sense Multiple Access/Collision Detection with Arbitration on Message Priority). Unlike Ethernet it will not stop transmission after collision.
- Actual physical design of CAN bus not specified in protocol
  - Requires devices to transmit/detect dominant and recessive signals to/from bus
  - e.g., '1' = dominant, '0' = recessive if single data wire used
  - Bus guarantees dominant signal prevails over recessive signal if asserted simultaneously
- Common devices interfacing with CAN:
  - 8051-compatible 8592 processor
  - Standalone CAN controllers - Dallas 80C400, etc.

Serial protocols: FireWire

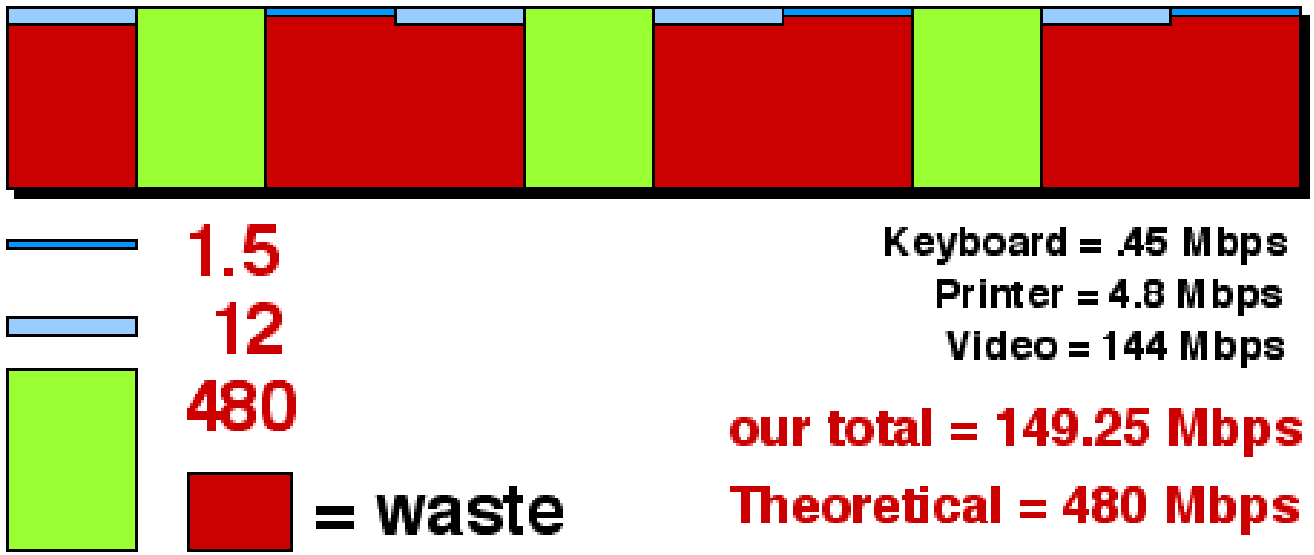
- FireWire (also I-Link, Lynx, IEEE 1394)
  - High-performance serial bus developed by Apple Computer Inc.
  - Designed for interfacing independent electronic components
    - e.g., disk drives, printers, scanners, cameras
  - Data transfer rates from 12.5 to 400 Mbits/s, 4.5m cable (FireWire 400)
  - Data transfer rate 800 Mbits/s, 100m cable (FireWire 800-IEEE1394b)
  - Plug-and-play capabilities
  - Packet-based layered design structure
  - Capable of supporting a LAN similar to Ethernet
    - 64-bit address:
      - 10 bits for network ids, 1023 subnetworks
      - 6 bits for node ids, each subnetwork can have 63 nodes
      - 48 bits for memory address, each node can have 281 terabytes of distinct locations

Serial protocols: USB

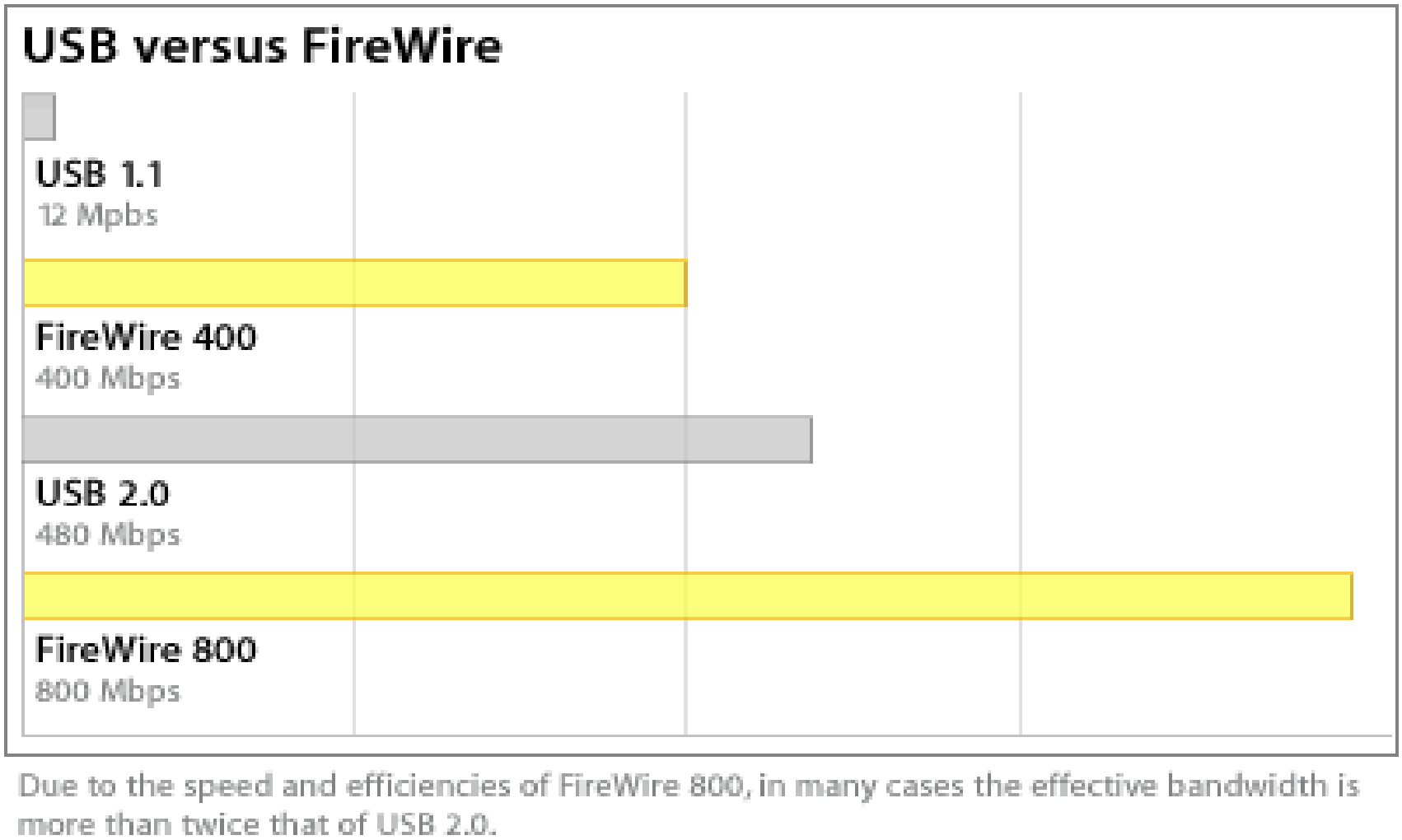
- USB (Universal Serial Bus)
  - Easier connection between PC and monitors, printers, digital speakers, modems, scanners, digital cameras, joysticks, multimedia game equipment
  - 2 data rates (USB 1.1):
    - 12 Mbps for increased bandwidth devices
    - 1.5 Mbps for lower-speed devices (joysticks, game pads)
  - USB 2.0:
    - 480 Mbps
  - Tiered star topology can be used
    - One USB device (hub) connected to PC
      - hub can be embedded in devices like monitor, printer, or keyboard or can be standalone
    - Multiple USB devices can be connected to hub
    - Up to 127 devices can be connected like this
  - USB host controller
    - Manages and controls bandwidth and driver software required by each peripheral
    - Dynamically allocates power downstream according to devices connected/disconnected

Serial protocols: USB

- Consider 3 devices, a keyboard (or mouse), a printer, and a new USB 2.0 video camera. 1/3 the performance would be used up at the lowest speed, and 1/3 at the middle speed, and what is left goes to the highest speed mode -- means 480 Mbps bus (data pipe) might really only achieve about 149 Mbps .



USB versus FireWire



Parallel protocols: PCI Bus

- PCI Bus (Peripheral Component Interconnect)
  - High performance bus originated at Intel in the early 1990's
  - Standard adopted by industry and administered by PCISIG (PCI Special Interest Group)
  - Interconnects chips, expansion boards, processor memory subsystems
  - Data transfer rates of 127.2 to 508.6 Mbits/s and 32-bit addressing
    - Later extended to 64-bit while maintaining compatibility with 32-bit schemes
  - Synchronous bus architecture (all transfers take place with respect to clk signal)
  - Multiplexed data/address lines

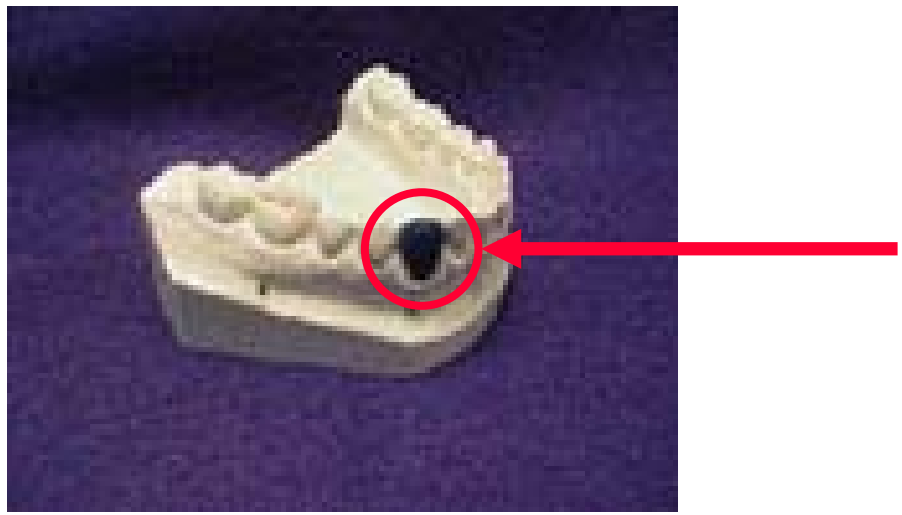
Parallel protocols: ARM Bus

- ARM Bus
  - Designed and used internally by ARM Corporation
  - Interfaces with ARM line of processors
  - Many IC design companies have own bus protocol
  - Data transfer rate is a function of clock speed
    - If clock speed of bus is X, transfer rate = 16 x X bits/s
  - 32-bit addressing

Wireless protocols: IrDA

- IrDA (Infrared Data Association)
  - Created and promoted by the Infrared Data Association (IrDA)
  - Supports short-range point-to-point infrared data transmission
  - Data transfer rate - between 9.6 kbps and 4 Mbps
  - IrDA hardware deployed in notebook computers, printers, PDAs, digital cameras, public phones, cell phones (small, semi-transparent, red window in laptops)
  - Lack of suitable drivers has slowed use by applications
  - Windows CE 1.0 was the first windows OS to provide built-in IrDA support
  - Windows 2000/98 include support
  - Becoming available on popular embedded OSs

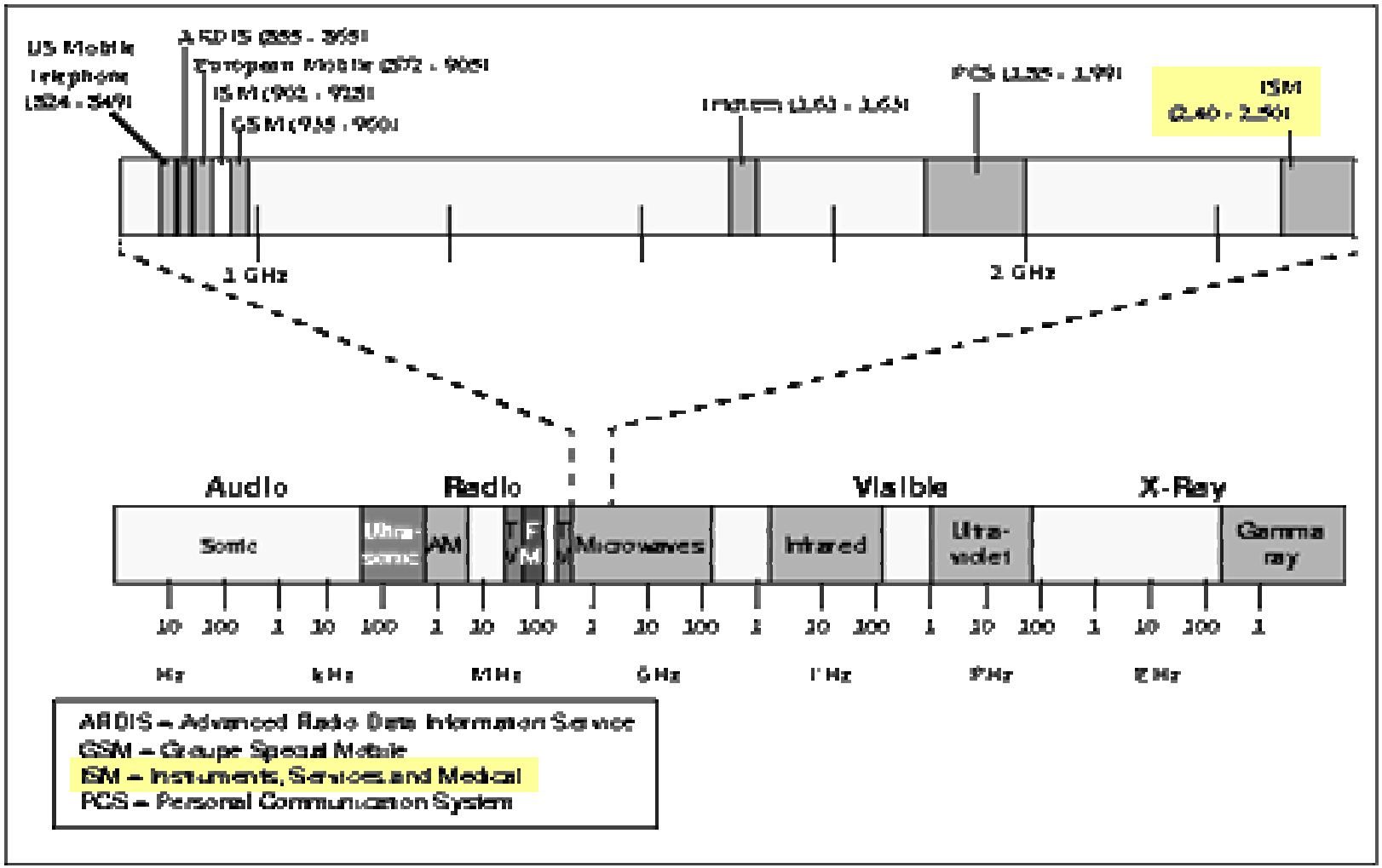
Wireless protocols: Bluetooth



Wireless protocols: Bluetooth

- Bluetooth
  - New, global standard for wireless connectivity
  - Based on low-cost, short-range radio link
  - Connection established when within 10 meters of each other
  - No line-of-sight required
    - e.g., Laptop connects to a printer in another room
  - Bluetooth communicates at a frequency of **2.45 gigahertz**, which has been set aside by international agreement for the use of Industrial, Scientific and Medical devices (ISM).
  - **Why is it called Bluetooth?**
    - Harald **Bluetooth** was king of Denmark in the late 900s (died in 986)
    - the Baltic region nations (including Denmark, Sweden, Norway and Finland) are leading in communications industry

Frequency Spectrum



Bluetooth Products

• The Sony Ericsson limited edition Car, with Bluetooth® wireless technology. It is a small race car, slightly larger than the size of a matchbox, that has two gears and is wirelessly controlled by a Bluetooth enabled Sony Ericsson mobile phone.

• Bluetooth USB adapter

• Laptop, PDA, mobile communicating through Bluetooth

Wireless Protocols: IEEE 802.11

- IEEE 802.11
  - Proposed standard for wireless LANs. There are two different ways to configure a network: ad-hoc and infrastructure.
  - Specifies parameters for PHY and MAC layers of network
    - PHY (Physical) layer
      - handles transmission of data between nodes
      - provisions for data transfer rates of 1 or 2 Mbps
      - operates in 2.4 to 2.4835 GHz frequency band for RF transmission
      - or 300 to 428,000 GHz for IR transmission
    - MAC (Medium Access Control) layer
      - specifies a set of protocols which is responsible for maintaining order in the use of a shared medium.
      - specifies a carrier sense multiple access with collision avoidance (CSMA/CA) protocol.

IEEE 802.11: Ad-hoc Network

- Computers are brought together to form a network "on the fly.“
- There is no structure to the network; no fixed points; and usually every node is able to communicate with every other node.
- Algorithms such as the spokesman election algorithm (SEA) have been designed to "elect" one machine as the master of the network with the others being slaves.
- Another algorithm in ad-hoc network architectures uses a broadcast and flooding method to all other nodes to establish who's who.

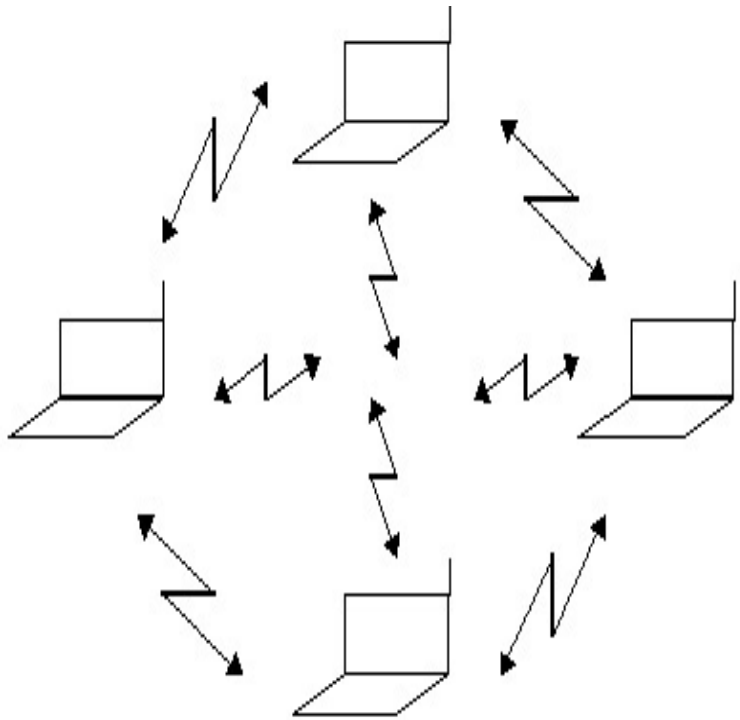


Figure 1: Ad-Hoc Network

IEEE 802.11: Infrastructure Network

- This architecture uses fixed network access points with which mobile nodes can communicate.
- These network access points are sometime connected to landlines to widen the LAN's capability by bridging wireless nodes to other wired nodes.
- If service areas overlap, handoffs can occur. This structure is very similar to the present day cellular networks around the world.

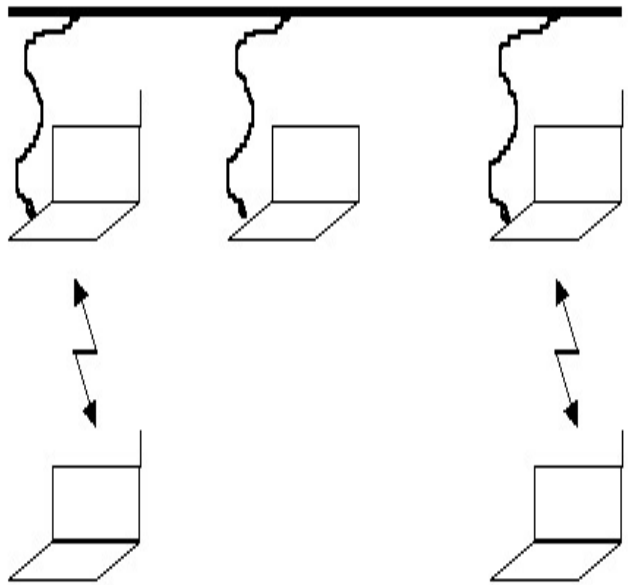


Figure 2: Infrastructure Network