

Embedded Systems Design: A Unified Hardware/Software Introduction

Section D

Outline

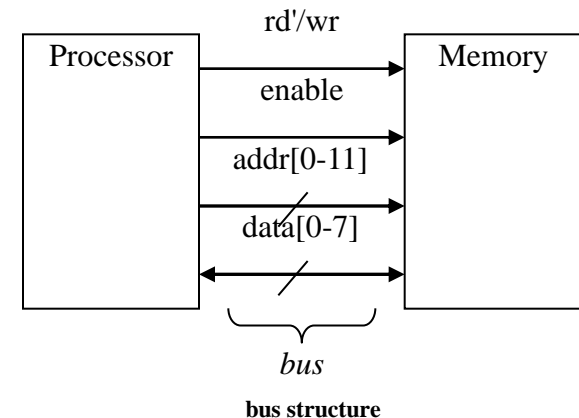
- Interfacing basics
 - Microprocessor interfacing
 - I/O Addressing
 - Interrupts
 - Direct memory access
 - Arbitration
 - Hierarchical buses
 - Protocols
 - Serial
 - Parallel
 - Wireless
-

Introduction

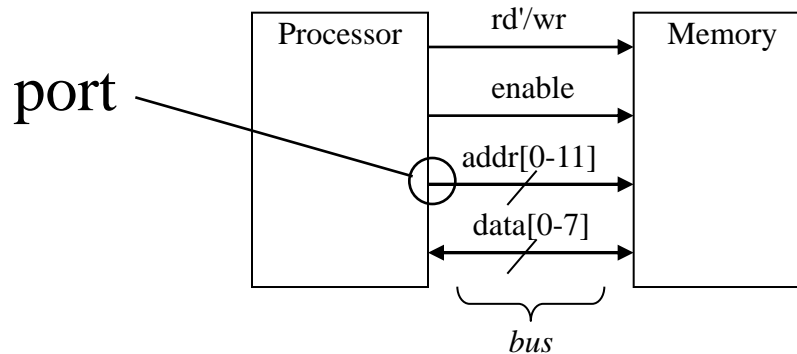
- Embedded system functionality aspects
 - Processing
 - Transformation of data
 - Implemented using processors
 - Storage
 - Retention of data
 - Implemented using memory
 - Communication
 - Transfer of data between processors and memories
 - Implemented using buses
 - Called *interfacing*
-

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
 - Or, entire collection of wires
 - Address, data and control
 - Associated protocol: rules for communication

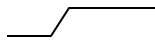
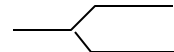


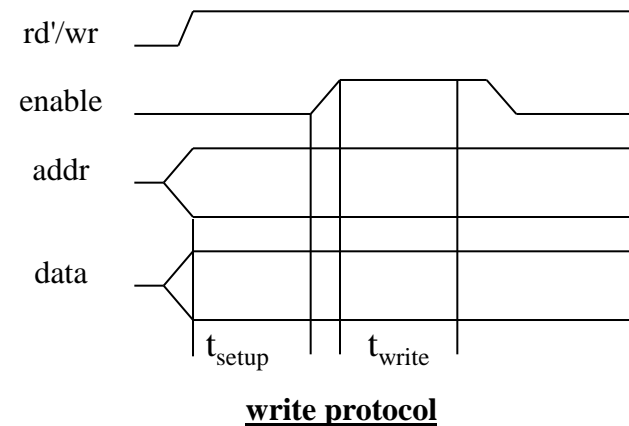
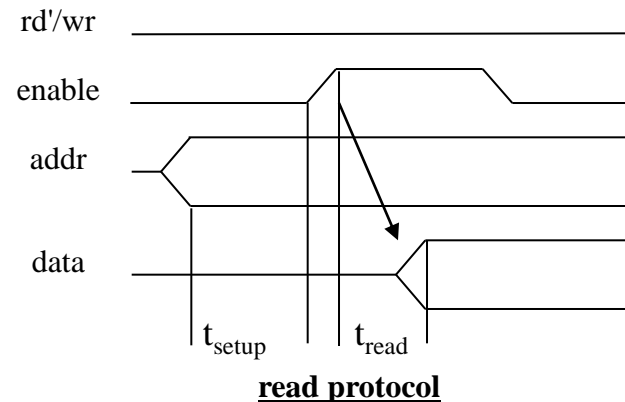
Ports



- Conducting device on 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
- Single wire or set of wires with single function
 - E.g., 12-wire address port

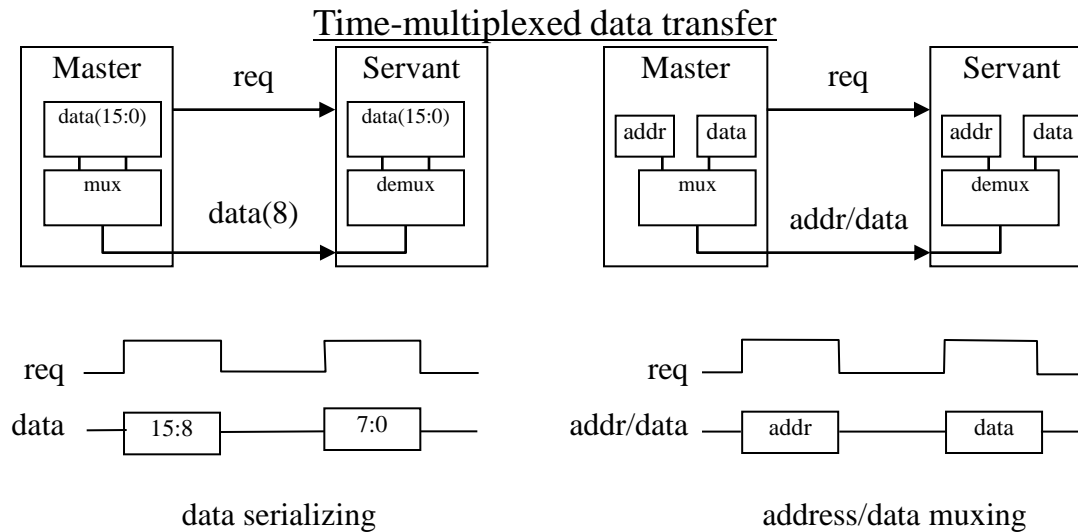
Timing Diagrams

- Most common method for describing a communication protocol
- Time proceeds to the right on x-axis
- Control signal: low or high 
 - May be active low (e.g., go' , $/go$, or go_L)
 - Use terms *assert* (active) and *deassert*
 - Asserting go' means $go=0$
- Data signal: not valid or valid 
- Protocol may have subprotocols
 - Called bus cycle, e.g., read and write
 - Each may be several clock cycles
- Read example
 - rd'/wr set low, address placed on $addr$ for at least t_{setup} time before $enable$ asserted, $enable$ triggers memory to place data on $data$ wires by time t_{read}

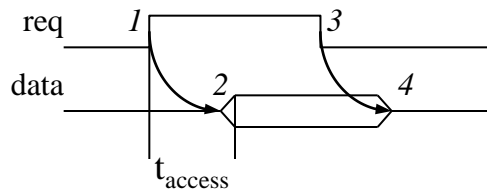
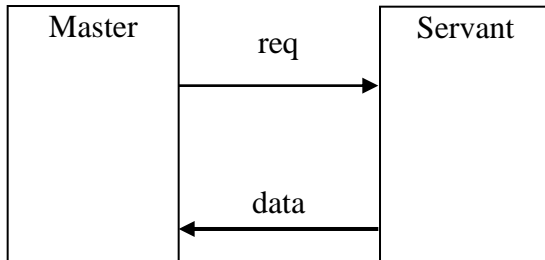


Basic protocol concepts

- Actor: master initiates, servant (slave) respond
- Direction: sender, receiver
- Addresses: special kind of data
 - Specifies a location in memory, a peripheral, or a register within a peripheral
- Time multiplexing
 - Share a single set of wires for multiple pieces of data
 - Saves wires at expense of time

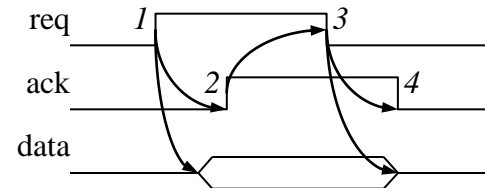
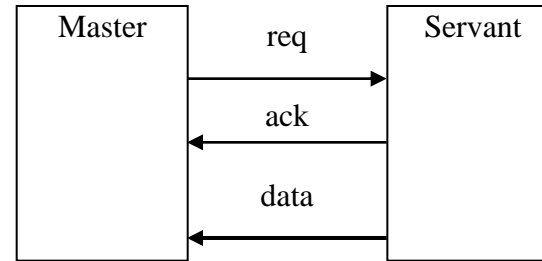


Basic protocol concepts: control methods



1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}**
3. Master receives data and deasserts *req*
4. Servant ready for next request

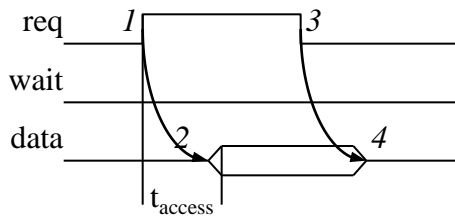
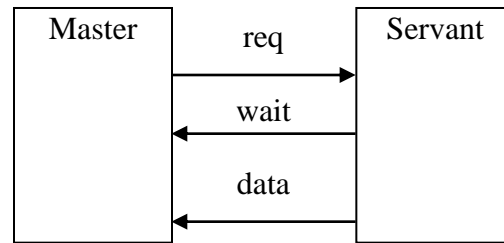
Strobe protocol



1. Master asserts *req* to receive data
2. Servant puts data on bus **and asserts *ack***
3. Master receives data and deasserts *req*
4. Servant ready for next request

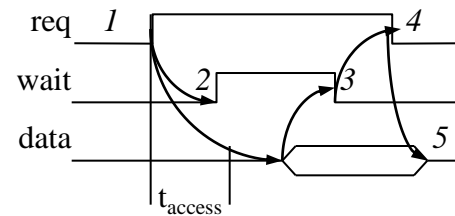
Handshake protocol

A strobe/handshake compromise



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

Fast-response case

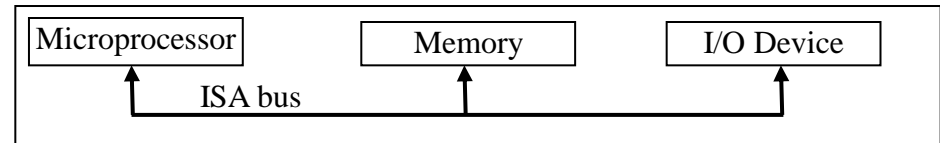


1. Master asserts *req* to receive data
2. Servant can't put data within t_{access} , **asserts *wait*** ack
3. Servant puts data on bus and **deasserts *wait***
4. Master receives data and deasserts *req*
5. Servant ready for next request

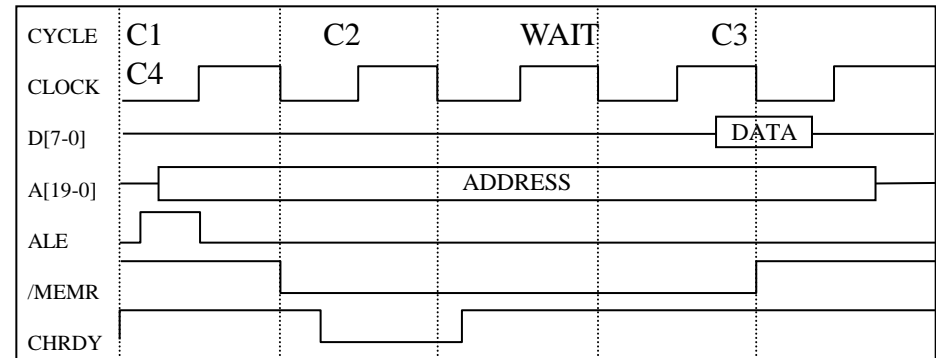
Slow-response case

ISA bus protocol – memory access

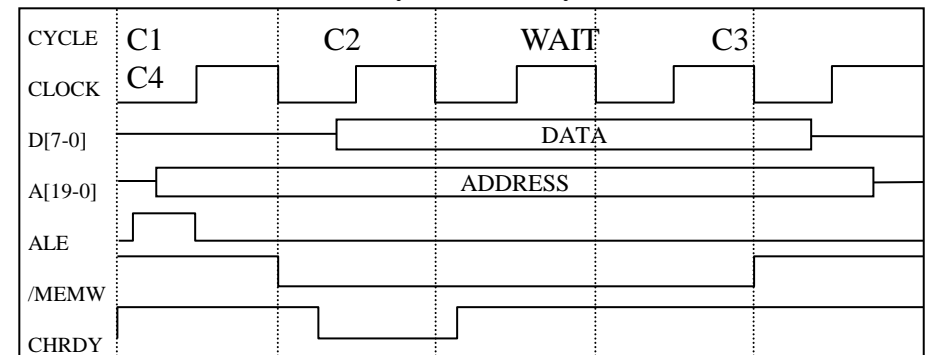
- ISA: Industry Standard Architecture
 - Common in 80x86's
- Features
 - 20-bit address
 - Compromise strobe/handshake control
 - 4 cycles default
 - Unless CHRDY deasserted – resulting in additional wait cycles (up to 6)



memory-read bus cycle



memory-write bus cycle

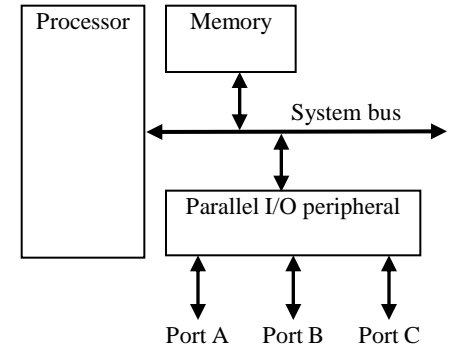


Microprocessor interfacing: I/O addressing

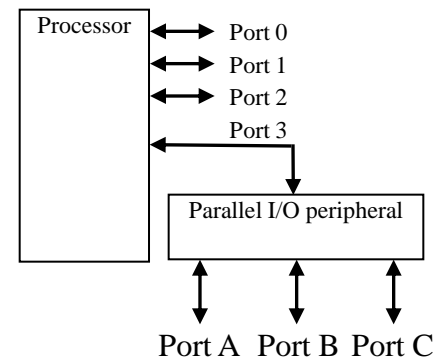
- A microprocessor communicates with other devices using some of its pins
 - Port-based I/O (parallel I/O)
 - Processor has one or more N-bit ports
 - Processor's software reads and writes a port just like a register
 - E.g., $P0 = 0xFF$; $v = P1.2$; -- P0 and P1 are 8-bit ports
 - Bus-based I/O
 - Processor 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

Compromises/extensions

- Parallel I/O peripheral
 - When processor only supports bus-based I/O but parallel I/O needed
 - Each port on peripheral connected to a register within peripheral that is read/written by the processor
- Extended parallel I/O
 - When processor supports port-based I/O but more ports needed
 - One or more processor ports interface with parallel I/O peripheral extending total number of ports available for I/O
 - e.g., extending 4 ports to 6 ports in figure



Adding parallel I/O to a bus-based I/O processor



Extended parallel I/O

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
-

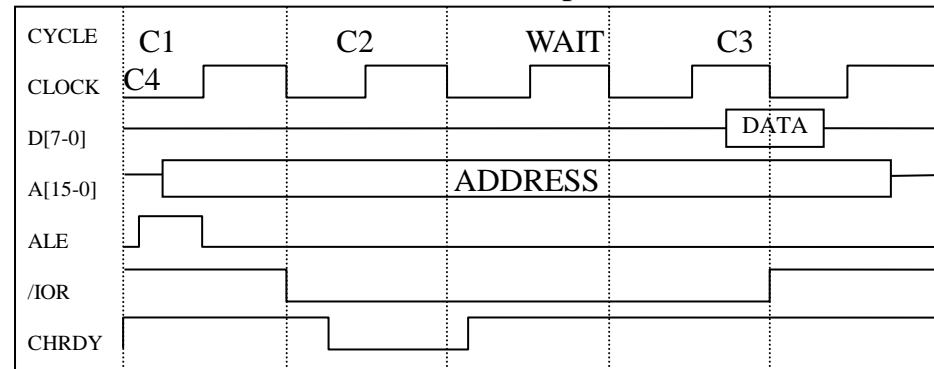
Memory-mapped I/O vs. Standard I/O

- 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

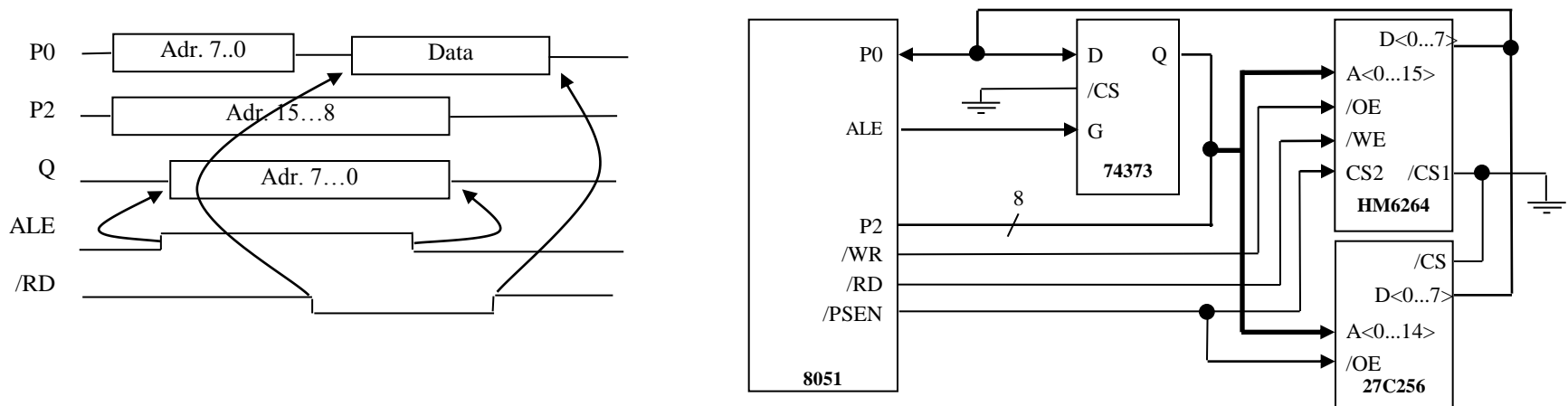
ISA bus

- 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



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