

The Embedded Computing Platform

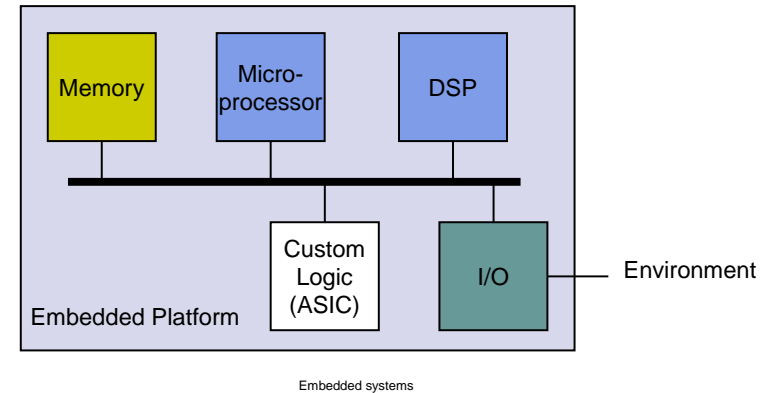
Ingo Sander
ingo@imit.kth.se

Based on Slides of Mats Brorsson and Wayne Wolf



What is a platform?

- A platform is the basic hardware and software architecture needed for an embedded application



Hardware platform architecture

Contains several elements:

- CPU;
- bus;
- memory;
- I/O devices: networking, sensors, actuators, etc.

How big/fast much each one be?

Software architecture

Functional description must be broken into pieces:

- division among people;
- conceptual organization;
- performance;
- testability;
- maintenance.

Hardware and software architectures



Hardware and software are intimately related:

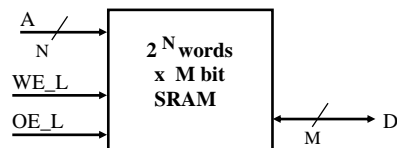
- software doesn't run without hardware;
- how much hardware you need is determined by the software requirements:
 - speed;
 - memory.

What do we need to define a hardware platform?



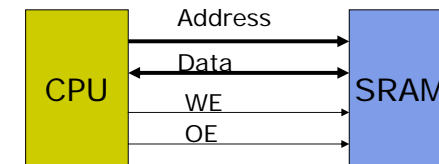
- Processor interface
 - Memory interface
 - I/O device interface
 - Means of communication
-
- It makes sense to use a standard!

Logic Diagram of a Typical SRAM



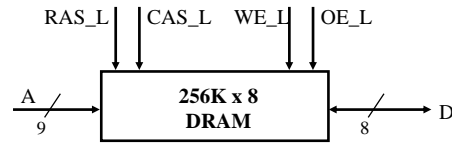
- Write Enable is usually active low (WE_L)
- Din and Dout are combined to save pins:
 - A new control signal, output enable (OE_L) is needed
 - WE_L is asserted (Low), OE_L is disasserted (High)
 - D serves as the data input pin
 - WE_L is disasserted (High), OE_L is asserted (Low)
 - D is the data output pin
 - Both WE_L and OE_L are asserted:
 - Result is unknown. Don't do that!!!

Simple Processor-Memory Communication



- The CPU and SRAM interfaces matches
- Communication is synchronous
- The set of wires between the CPU and the SRAM chip is called a *bus*

Logic Diagram of a Typical DRAM



- Control Signals (RAS_L, CAS_L, WE_L, OE_L) are all active low
- Din and Dout are combined (D):
 - WE_L is asserted (Low), OE_L is disasserted (High)
 - D serves as the data input pin
 - WE_L is disasserted (High), OE_L is asserted (Low)
 - D is the data output pin
- Row and column addresses share the same pins (A)
 - RAS_L goes low: Pins A are latched in as row address
 - CAS_L goes low: Pins A are latched in as column address
 - RAS/CAS edge-sensitive
- DRAMs need extra circuitry to match a bus interface

September 14, 2004

2B1447 Embedded Systems

9

Why Distributed Resources?

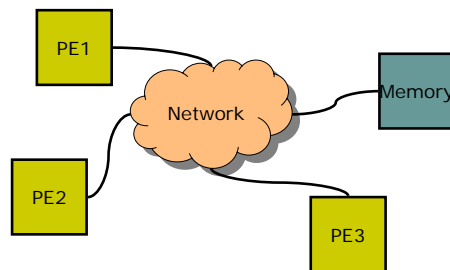
- Single Resource does not meet performance requirements
- Higher performance at lower cost
- Physically distributed activities
 - time constants may not allow transmission to central site
- Improved debugging
 - use one CPU in network to debug others
- May include IP-subsystems that have to be integrated into the rest of the system

September 14, 2004

2B1447 Embedded Systems

10

Interconnection Network



Ideally

- Full connectivity between components
- Unlimited bandwidth
- Very small delay
- Non-blocking
- Scalable

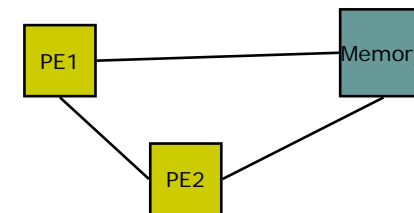
But in reality the capacity of the network is limited!

September 14, 2004

2B1447 Embedded Systems

11

Point-to-point networks



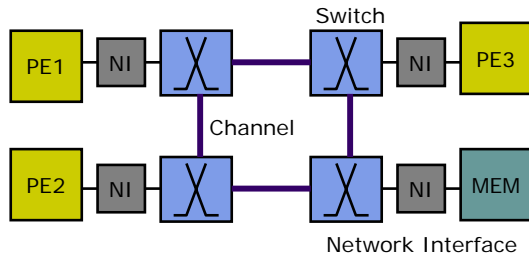
- All network elements are connected via point-to-point links with each other
- This is in general not possible
 - Many links are required
 - Many components do not have multiple ports and cannot serve two requests at same time (Memory)
 - At least components have to be extended by arbiters-interfaces

September 14, 2004

2B1447 Embedded Systems

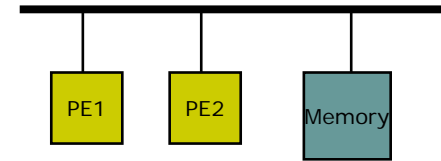
12

Networks



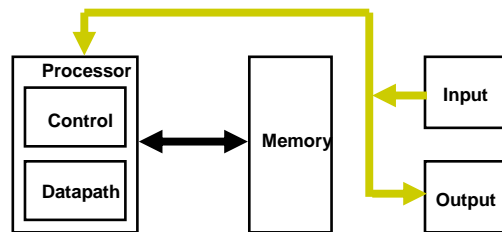
- The resources are connected to the network via network interfaces
- The topology of the network and the capability of the switches and communication channels determines the capacity of the network

Bus-Based Networks



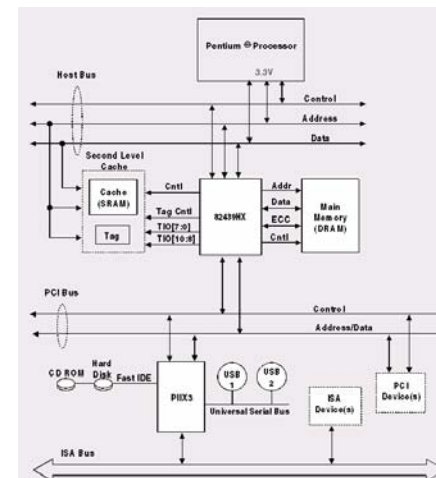
- The bus provides a communication link which all processing elements can access
- Only one bus master can use the bus at each instance of time

A Bus is:



- shared communication link
- single set of wires used to connect multiple subsystems
- A Bus is also a fundamental tool for composing large, complex systems
 - systematic means of abstraction

Example: Pentium System Organization

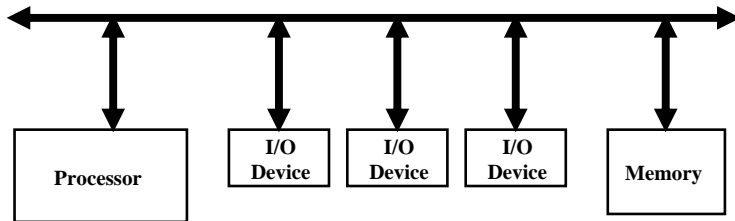


Processor/Memory Bus

PCI Bus

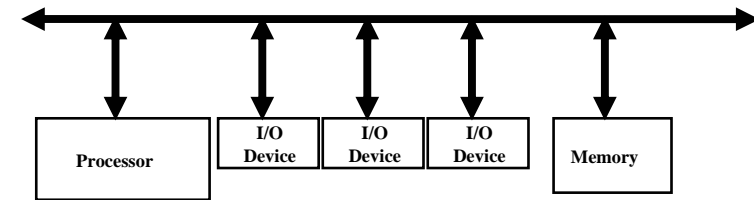
I/O Buses

Advantages of Buses



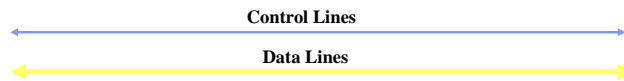
- Versatility:
 - New devices can be added easily
 - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
 - A single set of wires is shared in multiple ways
- Manage complexity by partitioning the design

Disadvantage of Buses



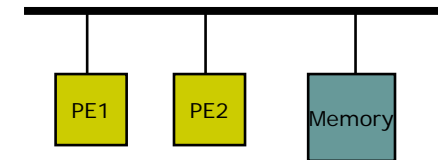
- It creates a communication bottleneck
 - The bandwidth of that bus can limit the maximum I/O throughput
- The maximum bus speed is largely limited by:
 - The length of the bus
 - The number of devices on the bus
 - The need to support a range of devices with:
 - Widely varying latencies
 - Widely varying data transfer rates

The General Organization of a Bus



- Control lines:
 - Signal requests and acknowledgments
 - Indicate what type of information is on the data lines
- Data lines carry information between the source and the destination:
 - Data and Addresses
 - Complex commands

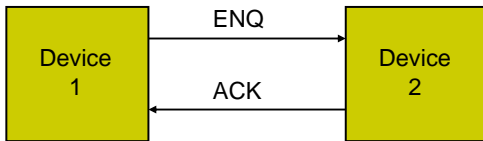
Summary Buses



- ↑ A bus is used to allow the communication between several devices
- ↑ The bus provides a *single* shared communication link between devices
- ↑ It is easy to add new devices to a bus
- ↓ A bus can cause a communication bottleneck! Thus scalability is limited!

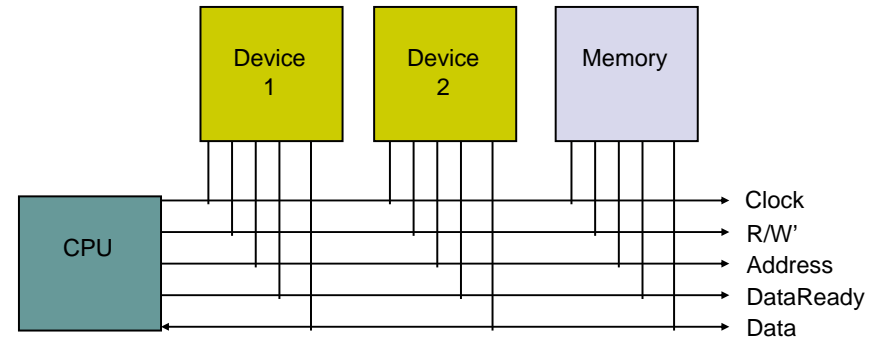
Bus Protocol

Four Cycle Handshake



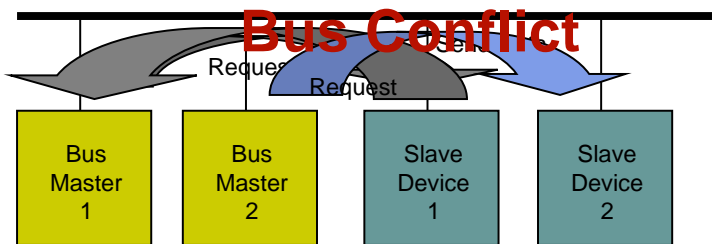
- Device 1 signals an enquiry (ENQ = active)
- Device 2 is ready to receive (ACK = active)
- When data transfer is complete, device 2 sets ACK = inactive
- After receiving ACK = inactive device 1 sets ENQ = inactive

A simple bus



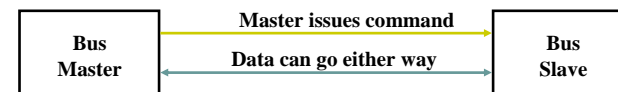
CPU controls the bus, devices and memory act on request (Memory-Mapped I/O)

Several Masters need an Arbitration Protocol



- *Bus Master 1* requests data from *Slave 1*
- *Slave 1* starts sending data
- *Bus Master 2* requests data from *Slave 2*
 - => Bus Conflict!

Master versus Slave



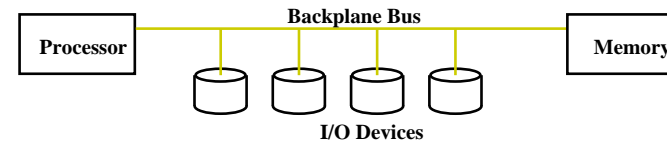
- A bus transaction includes two parts:
 - Issuing the command (and address) – request
 - Transferring the data – action
- Master is the one who starts the bus transaction by:
 - issuing the command (and address)
- Slave is the one who responds to the address by:
 - Sending data to the master if the master asks for data
 - Receiving data from the master if the master wants to send data



Types of Buses

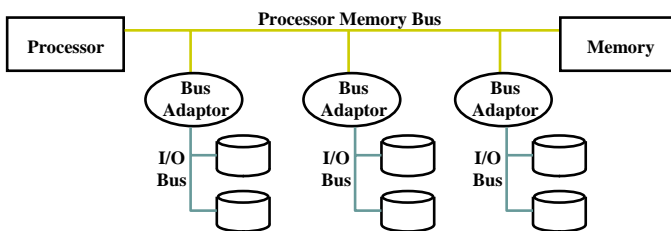
- Processor-Memory Bus (design specific)
 - Short and high speed
 - Only need to match the memory system
 - Maximize memory-to-processor bandwidth
 - Connects directly to the processor
 - Optimized for cache block transfers
- I/O Bus (industry standard)
 - Usually is lengthy and slower
 - Need to match a wide range of I/O devices
 - Connects to the processor-memory bus or backplane bus
- Backplane Bus (standard or proprietary)
 - Backplane: an interconnection structure within the chassis
 - Allow processors, memory, and I/O devices to coexist
 - Cost advantage: one bus for all components

A Computer System with One Bus: Backplane Bus



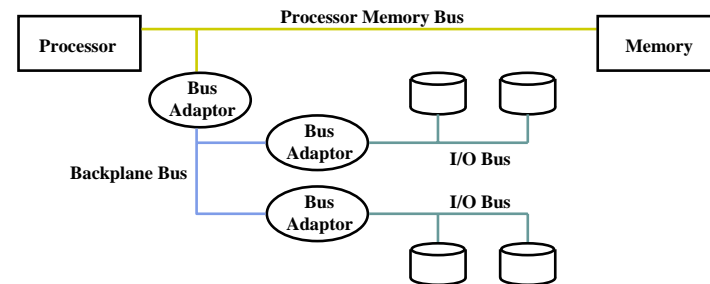
- A single bus (the backplane bus) is used for:
 - Processor to memory communication
 - Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC - AT

A Two-Bus System



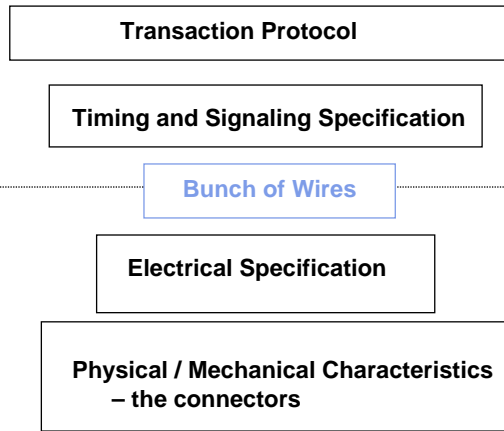
- I/O buses tap into the processor-memory bus via bus adaptors:
 - Processor-memory bus: mainly for processor-memory traffic
 - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
 - NuBus: Processor, memory, and a few selected I/O devices
 - SCCI Bus: the rest of the I/O devices

A Three-Bus System

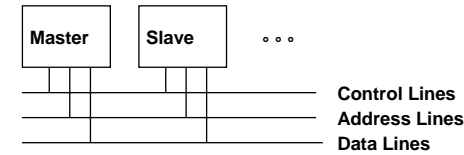


- A small number of backplane buses tap into the processor-memory bus
 - Processor-memory bus is used for processor memory traffic
 - I/O buses are connected to the backplane bus
- Advantage: loading on the processor bus is greatly reduced

What defines a bus?



Busses so far



Multibus: 20 address, 16 data, 5 control, 50ns Pause

- Bus Master: has ability to control the bus, initiates transaction
- Bus Slave: module activated by the transaction
- Bus Communication Protocol: specification of sequence of events and timing requirements in transferring information.

- Asynchronous Bus Transfers: control lines (req, ack) serve to orchestrate sequencing.
- Synchronous Bus Transfers: sequence relative to common clock.

Synchronous and Asynchronous Bus



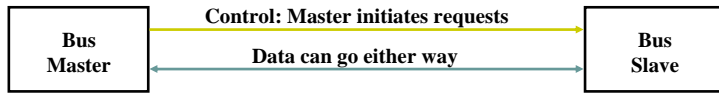
- Synchronous Bus:
 - Includes a clock in the control lines
 - A fixed protocol for communication that is relative to the clock
 - Advantage: involves very little logic and can run very fast
 - Disadvantages:
 - Every device on the bus must run at the same clock rate
 - To avoid clock skew, they cannot be long if they are fast
- Asynchronous Bus:
 - It is not clocked
 - It can accommodate a wide range of devices
 - It can be lengthened without worrying about clock skew
 - It requires a handshaking protocol

Bus Transaction



- Arbitration
- Request
- Action

Arbitration: Obtaining Access to the Bus



- One of the most important issues in bus design:
 - How is the bus reserved by a devices that wishes to use it?
- Chaos is avoided by a master-slave arrangement:
 - Only the bus master can control access to the bus:
 - It initiates and controls all bus requests
 - A slave responds to read and write requests
- The simplest system:
 - Processor is the only bus master
 - All bus requests must be controlled by the processor
 - Major drawback: the processor is involved in every transaction

Multiple Potential Bus Masters: the Need for Arbitration



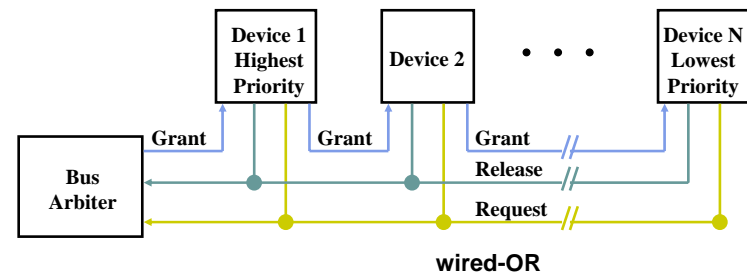
- Bus arbitration scheme:
 - A bus master wanting to use the bus asserts the bus request
 - A bus master cannot use the bus until its request is granted
 - A bus master must signal to the arbiter after finish using the bus
- Bus arbitration schemes usually try to balance two factors:
 - Bus priority: the highest priority device should be serviced first
 - Fairness: Even the lowest priority device should never be completely locked out from the bus

Multiple Potential Bus Masters: the Need for Arbitration



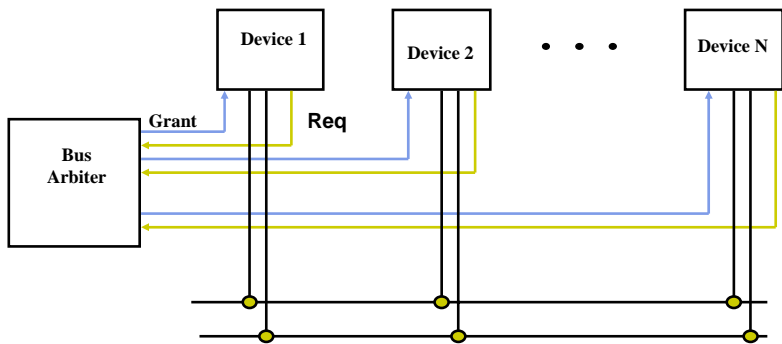
- Bus arbitration schemes can be divided into four broad classes:
 - Daisy chain arbitration: single device with all request lines.
 - Centralized, parallel arbitration: see next-next slide
 - Distributed arbitration by self-selection: each device wanting the bus places a code indicating its identity on the bus.
 - Distributed arbitration by collision detection: Ethernet uses this.

The Daisy Chain Bus Arbitration Scheme



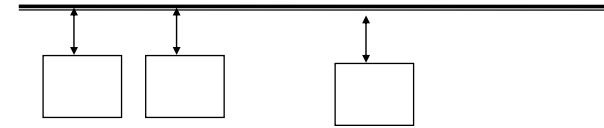
- Advantage: simple
- Disadvantages:
 - Cannot assure fairness:
 - A low-priority device may be locked out indefinitely
 - The use of the daisy chain grant signal also limits the bus speed

Centralized Parallel Arbitration



- Used in essentially all processor-memory busses and in high-speed I/O busses

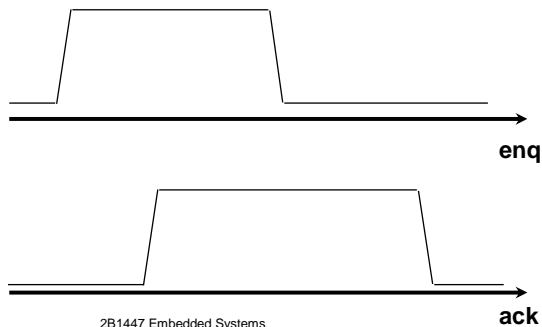
Simplest bus paradigm



- All agents operate synchronously
- All can source / sink data at same rate
- => simple protocol
 - just manage the source and target

Timing diagrams

- A timing diagram shows a trace through the operation of a system.
 - Generally used for asynchronous machines with timing constraints.



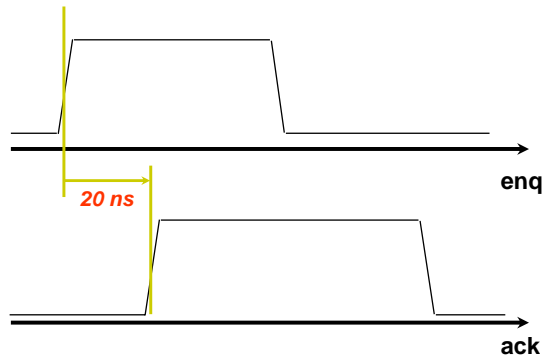
Timing diagram syntax

- Constant value:
 - 1 _____
 - 0 _____
- Stable:
 - =====
 - =====
- Changing:
 - ↗
- Unknown:
 - ⋈

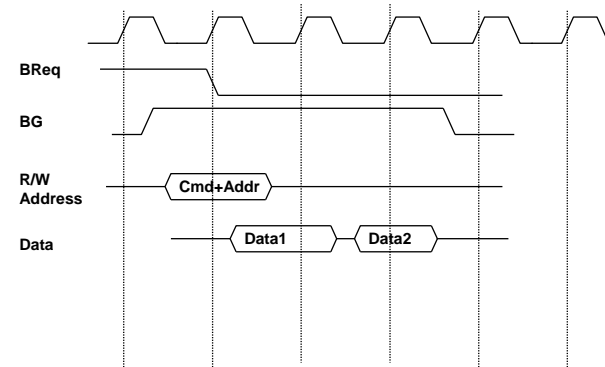


Timing constraints

- Minimum time between two events:



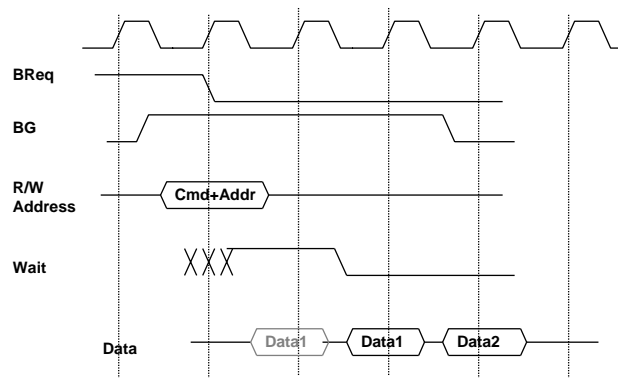
Simple Synchronous Protocol



- Even memory busses are more complex than this
 - memory (slave) may take time to respond
 - it need to control data rate



Typical Synchronous Protocol



- Slave indicates when it is prepared for data transfer
- Actual transfer goes at bus rate



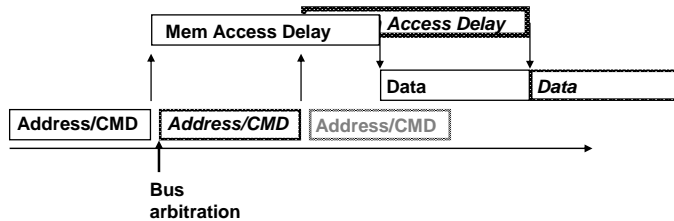
Increasing the Bus Bandwidth

- Separate versus multiplexed address and data lines:
 - Address and data can be transmitted in one bus cycle if separate address and data lines are available
 - Cost: (a) more bus lines, (b) increased complexity
- Data bus width:
 - By increasing the width of the data bus, transfers of multiple words require fewer bus cycles
 - Example: SPARCstation 20's memory bus is 128 bit wide
 - Cost: more bus lines
- Block transfers:
 - Allow the bus to transfer multiple words in back-to-back bus cycles
 - Only one address needs to be sent at the beginning
 - The bus is not released until the last word is transferred
 - Cost: (a) increased complexity (b) decreased response time for request



Split-Transaction (pipelined) Bus

- Consists of two separate busses: request bus (for add&cmd) and response bus (for data)
- Split bus transaction into request and response sub-transactions
 - Separate arbitration for each phase
- Other transactions may intervene
 - Improves bandwidth dramatically
 - Response is matched to request
 - Buffering between bus and cache controllers
- Reduce serialization down to the actual bus arbitration



September 14, 2004

2B1447 Embedded Systems

45



Increasing Transaction Rate on Multimaster Bus

- Overlapped arbitration
 - perform arbitration for next transaction during current transaction
- Bus parking
 - master can hold the bus and performs multiple transactions as long as no other master makes request
- Overlapped address / data phases (prev. slide)
 - requires one of the above techniques
- Split-phase (or packet switched) bus
 - completely separate address and data phases
 - arbitrate separately for each
 - address phase yield a tag which is matched with data phase
- "All of the above" in most modern memory busses

September 14, 2004

2B1447 Embedded Systems

46



The I/O Bus Problem

- Designed to support wide variety of devices
 - full set not know at design time
- Allow data rate match between arbitrary speed device
 - fast processor – slow I/O
 - slow processor – fast I/O

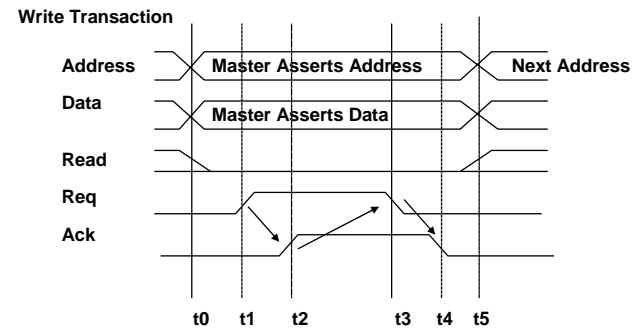
September 14, 2004

2B1447 Embedded Systems

47



Asynchronous Handshake



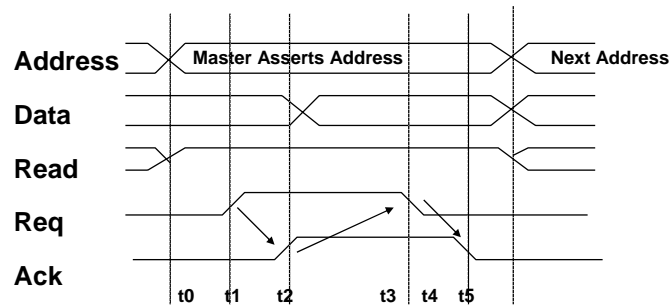
- t0 : Master has obtained control and asserts address, direction, data
 - Waits a specified amount of time for slaves to decode target
- t1: Master asserts request line
- t2: Slave asserts ack, indicating data received
- t3: Master releases req
- t4: Slave releases ack

September 14, 2004

2B1447 Embedded Systems

48

Read Transaction



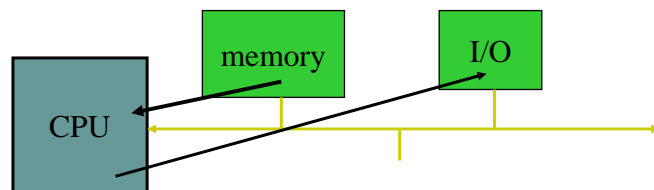
- t0 : Master has obtained control and asserts address, direction, data
 - Waits a specified amount of time for slaves to decode target
- t1: Master asserts request line
- t2: Slave asserts ack, indicating ready to transmit data
- t3: Master releases req, data received
- t4: Slave releases ack

High Speed I/O Bus

- Examples
 - graphics
 - fast networks
- Limited number of devices
- Data transfer bursts at full rate
- DMA transfers important
 - small controller spools stream of bytes to or from memory
- Either side may need to squelch transfer
 - buffers fill up

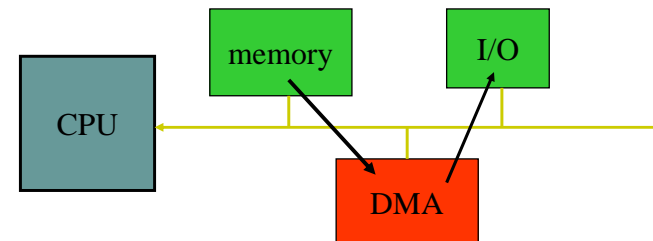
I/O may block the CPU

- If lots of data has to be send between I/O and memory, the CPU as only bus master is blocked, since all traffic has to be send via the CPU



Direct memory access (DMA)

- DMA provides parallelism on bus by controlling transfers without CPU
 - CPU can do another job
 - BUT: CPU may not occupy the bus when DMA has the bus.



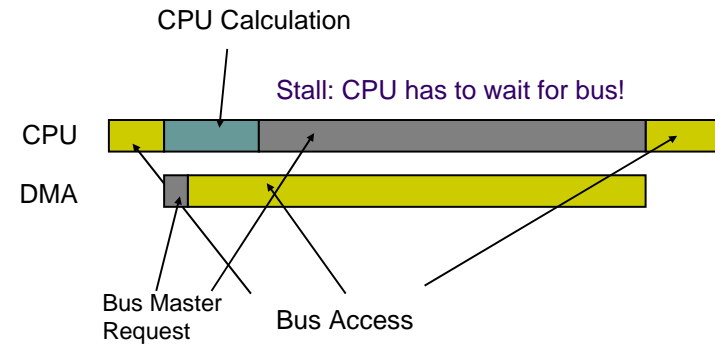


DMA operation

- CPU sets up DMA transfer:
 - Start address.
 - Length.
 - Transfer block length.
 - Style of transfer.
- DMA controller performs transfer, signals when done:
 - Cycle-stealing.
 - Priority.



DMA can block CPU



Cyclic Scheduling of DMA Request



- DMA uses bus only for a limited number of cycles and allows then the CPU to take control of the bus
- Long CPU Stalls are avoided!



Embedded busses

- Current system-on-chips are advanced enough to need a hierarchy of busses
- A new set of bus standards have been defined to be used in SoCs:
 - ARM Amba
 - Altera Avalon
 - ...



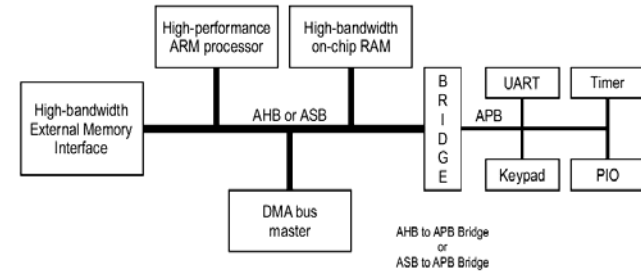
AMBA Specification

- The AMBA specification defines an on-chip communications standard for designing high-performance embedded micro-controllers
- Three buses are defined
 - Advanced High-Performance Bus (AHB)
 - Advanced System Bus (ASB)
 - Advanced Peripheral Bus (APB)
- A test methodology is included within AMBA which provides an infrastructure for modular macrocell test and diagnostic access



System based on an AMBA Bus

- An AMBA system typically contains a high speed bus (ASB or AHB) for CPU, fast memory and DMA and a bus for peripherals (APB), which is connected via a bridge to the high-speed bus



AMBA Buses

- AMBA AHB (new standard)
 - High Performance
 - Pipelined Operation
 - Multiple Bus Masters
 - Burst Transfers
 - Split Transactions
- AMBA ASB (older standard)
 - High Performance
 - Pipelined Operation
 - Multiple Bus Masters
- AMBA APB
 - Low Power
 - Latched Address and Control
 - Simple Interface
 - Suitable for many peripherals



AMBA AHB System

- AHB Master
 - A bus master is able to initiate read and write informations by providing address and control information. Only one bus master can use the bus at the same time
- AHB Slave
 - A bus slave responds to a read and write operation within a given address-space range. The bus slave signals back to the active bus master the success, failure or waiting of the data transfer

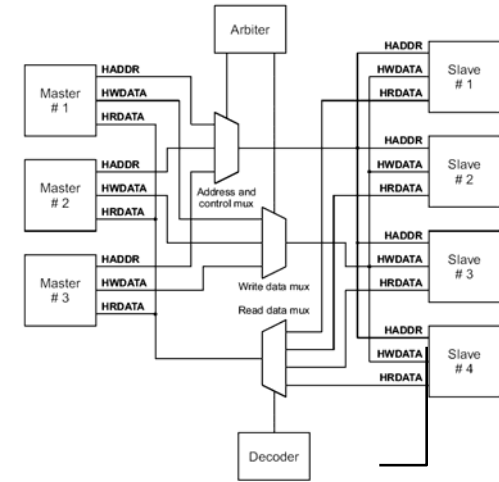


AMBA AHB System

- AHB Arbiter
 - The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements
 - An AHB includes only one arbiter
- AHB Decoder
 - The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer
 - A single centralized decoder is required in all AHB implementations



AMBA AHB Bus Interconnection

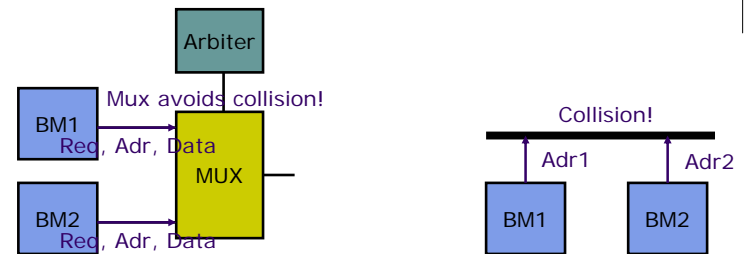


AMBA AHB Bus Interconnection

- AHB Protocol is based on a central multiplexor interconnection scheme
- All bus masters send their request in form of address and control signals
- The arbiter chooses one master. The address and control signals are routed to all slaves
- The decoder selects the signals from the slave that is involved in the transfer with the bus master



Comparison: Multiplexor Bus and Tri-State Bus



Multiplexor Bus

- Bus Master can send their request including address and data (for write) at the same time
- Arbiter selects a bus master

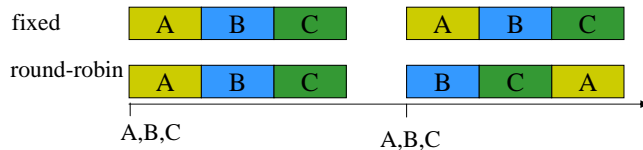
Tri-State Bus

- Only one bus master can output address or data (otherwise collision)
- A Bus Grant is needed to output address or data

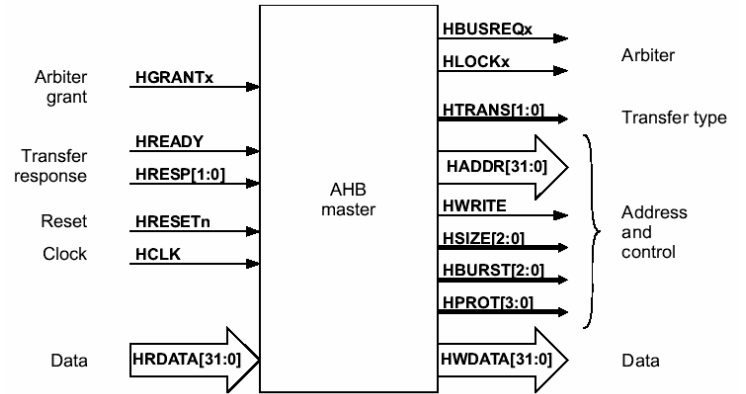


Bus arbitration

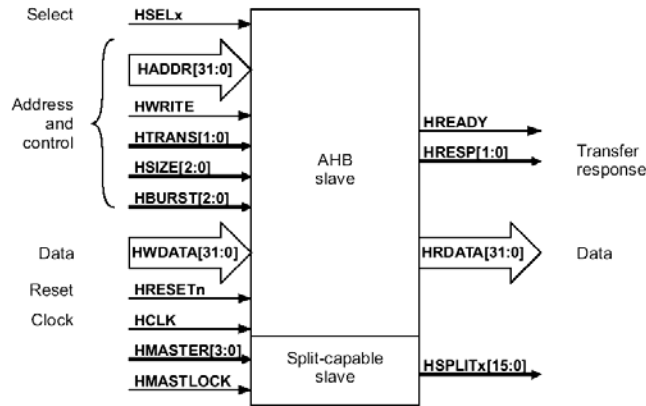
- The designer is free to define the priority arbitration in AMBA as long as it complies to the AMBA protocol
- Fixed:** Same order of priority every time.
 - Low Priority traffic may have to wait very long
- Fair:** every PE has same access over long periods.
 - round-robin:** rotate top priority among PEs



Interface AHB Bus Master



Interface AHB Bus Slave



AMBA APB System

- The APB is optimized for minimal power consumption and reduced interface complexity
- The APB appears as a local secondary bus that is encapsulated as a single AHB or ASB slave device
- The APB should be used to interface any peripherals which are low bandwidth and do not require the high performance of AHB or ASB
- An APB implementation usually contains a single APB bridge

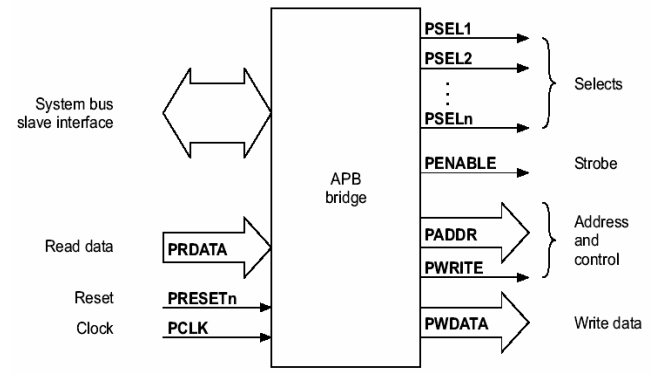


AMBA APB System

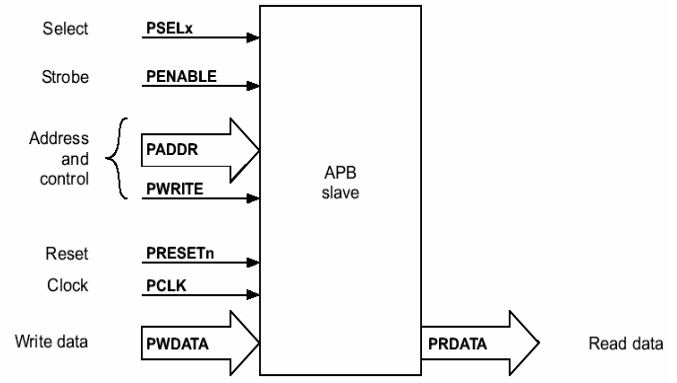
- All other other modules in the APB are slaves
- Interface Specification
 - address and control valid throughout the access (unpipelined)
 - zero-power interface during non-peripheral bus activity (peripheral bus is static when not in use)
 - timing can be provided by decode with strobe timing (unlocked interface)
 - write data valid for the whole access (allowing glitch-free transparent latch implementations).



Interface APB Bridge



Interface APB Slave



Intellectual Properties and Buses

- There exist many intellectual property devices, which can give a great benefit to your system
- IP's are usually available for the main bus protocols (like AMBA)
- If you do not use a standard bus protocol you may have to design adapters for each IP you use



System components

- Memory.
- Busses and interconnect.



Memory parameters

- Size.
 - Address width.
- Aspect ratio.
 - Data width.



Types of memory

- ROM:
 - Mask-programmable.
 - Flash programmable.
- RAM:
 - DRAM.
 - SRAM.

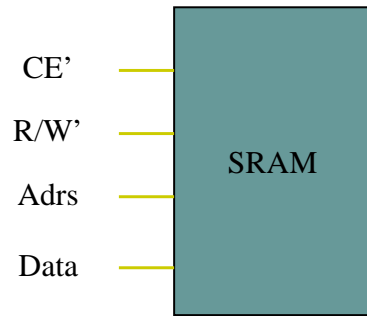


SRAM vs. DRAM

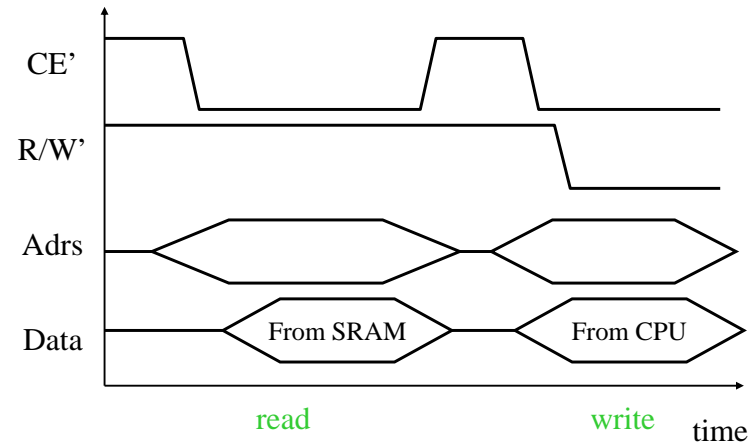
- SRAM:
 - Faster.
 - Easier to integrate with logic.
 - Higher power consumption.
- DRAM:
 - Denser.
 - Must be refreshed.



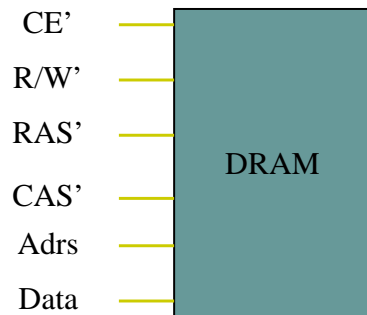
Typical generic SRAM



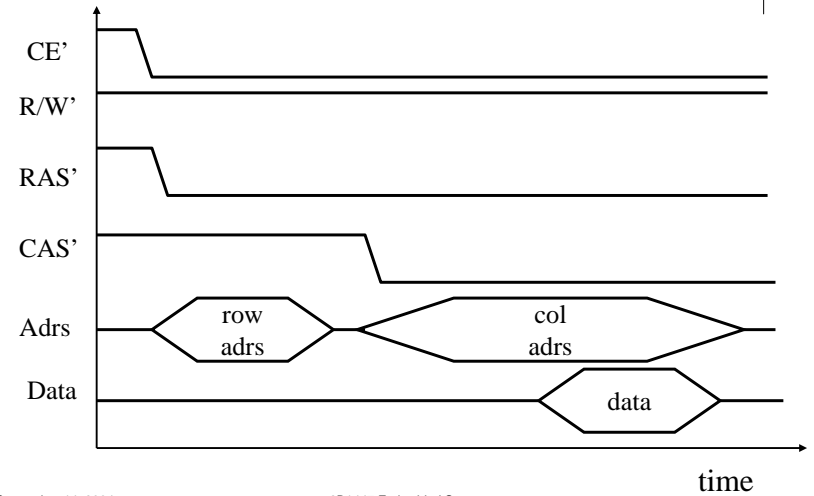
Generic SRAM timing



Generic DRAM device

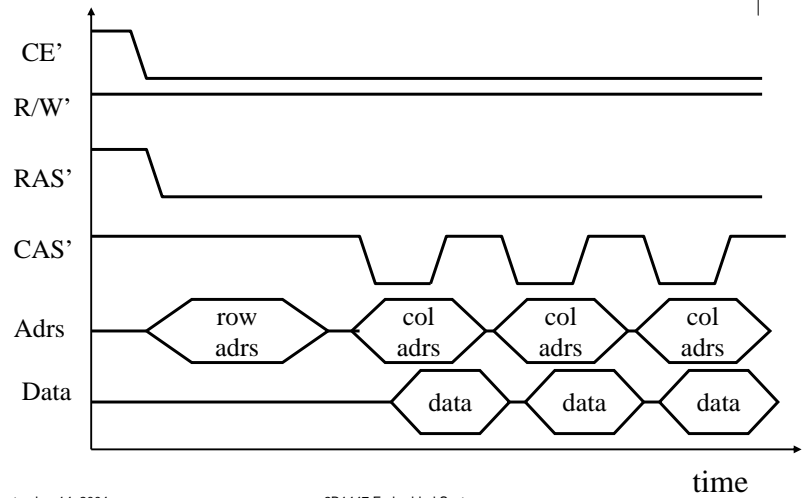


Generic DRAM timing





Page mode access



RAM refresh

- Value decays in approx. 1 ms.
- Refresh value by reading it.
 - Can't access memory during refresh.
- CAS-before-RAS refresh.
- Hidden refresh.



Other types of memory

- Extended data out (EDO): improved page mode access.
- Synchronous DRAM: clocked access for pipelining.
- Rambus: highly pipelined DRAM.



Flash issues

- Flash is programmed at system voltages.
- Erasure time is long.
- Must be erased in blocks.
- Flash memories keep their value, if the power is disconnected (SRAM and DRAM lose their values!)



I/O devices

- I/O devices:
 - serial links
 - timers and counters
 - keyboards
 - displays
 - analog I/O



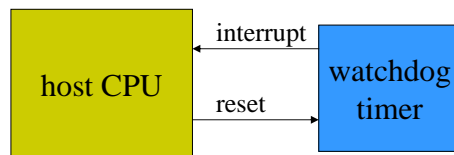
Timers and counters

- Very similar:
 - a **timer** is incremented by a periodic signal;
 - a **counter** is incremented by an asynchronous, occasional signal.
- Rollover causes interrupt.



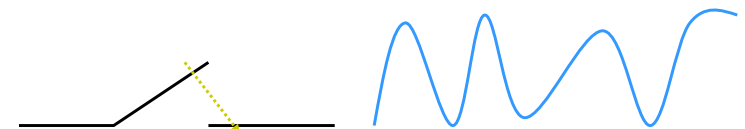
Watchdog timer

- Watchdog timer is periodically reset by system timer.
- If watchdog is not reset, it generates an interrupt to reset the host.



Switch debouncing

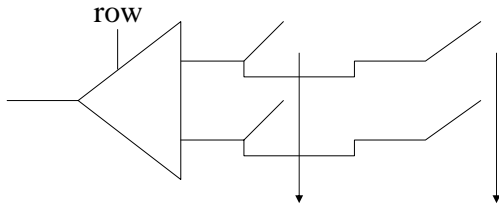
- A switch must be debounced to multiple contacts caused by eliminate mechanical bouncing:





Encoded keyboard

- An array of switches is read by an encoder.
- **N-key rollover** remembers multiple key depressions.



September 14, 2004

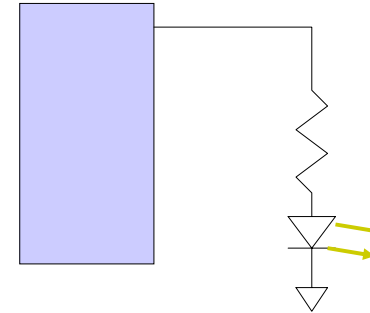
2B1447 Embedded Systems

89



LED

- Must use resistor to limit current:



September 14, 2004

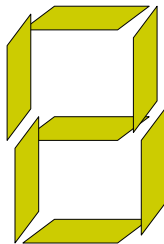
2B1447 Embedded Systems

90



7-segment LCD display

- May use parallel or multiplexed input.



September 14, 2004

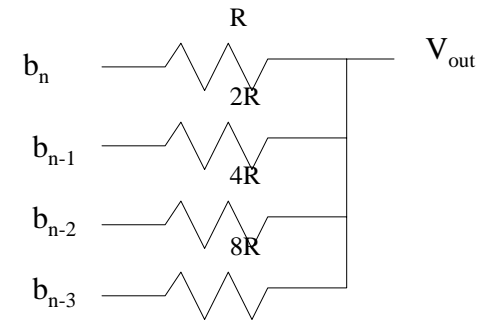
2B1447 Embedded Systems

91



Digital-to-analog conversion

- Use resistor tree:



September 14, 2004

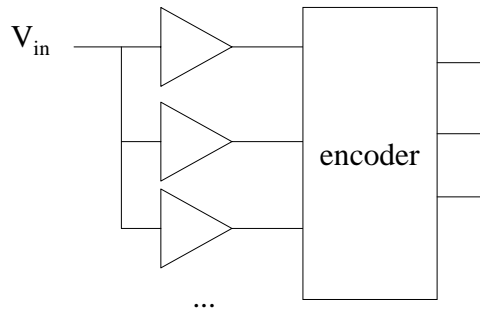
2B1447 Embedded Systems

92



Flash A/D conversion

- N-bit result requires 2^n comparators:



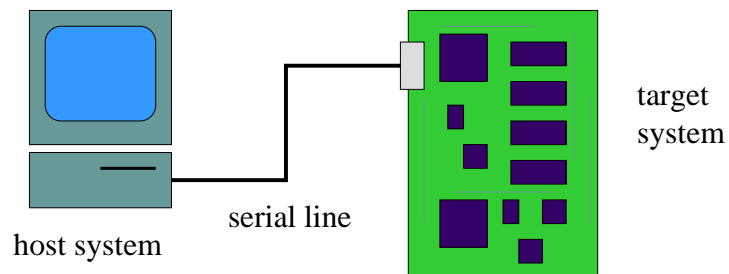
Software design techniques

- Want to develop as much code as possible on a standard platform:
 - friendlier programming environment;
 - easier debugging.
- May need to devise software stubs to allow testing of software elements without the full hardware/software platform.



Host/target design

- Use a host system to prepare software for target system:



Host-based tools

- Cross compiler:
 - compiles code on host for target system.
- Cross debugger:
 - displays target state, allows target system to be controlled.



Evaluation boards

- Designed by CPU manufacturer or others.
- Includes CPU, memory, some I/O devices.
- May include prototyping section.
- CPU manufacturer often gives out evaluation board netlist---can be used as starting point for your custom board design.



Adding logic to a board

- **Programmable logic devices (PLDs)** provide low/medium density logic.
- **Field-programmable gate arrays (FPGAs)** provide more logic and multi-level logic.
- **Application-specific integrated circuits (ASICs)** are manufactured for a single purpose.



Debugging embedded systems

- Challenges:
 - target system may be hard to observe;
 - target may be hard to control;
 - may be hard to generate realistic inputs;
 - setup sequence may be complex.



Software debuggers

- A monitor program residing on the target provides basic debugger functions.
- Debugger should have a minimal footprint in memory.
- User program must be careful not to destroy debugger program, but , should be able to recover from some damage caused by user code.



Breakpoints

- A breakpoint allows the user to stop execution, examine system state, and change state.
- Replace the breakpointed instruction with a subroutine call to the monitor program.



Breakpoint handler actions

- Save registers.
- Allow user to examine machine.
- Before returning, restore system state.
 - Safest way to execute the instruction is to replace it and execute in place.
 - Put another breakpoint after the replaced breakpoint to allow restoring the original breakpoint.



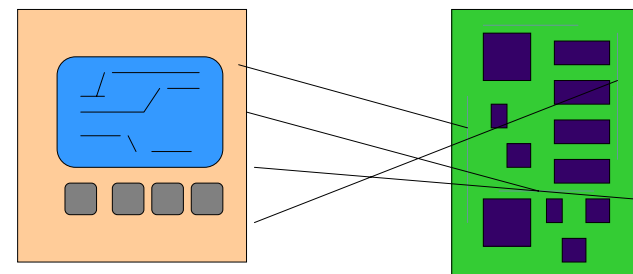
In-circuit emulators

- A microprocessor in-circuit emulator is a specially-instrumented microprocessor.
- Allows you to stop execution, examine CPU state, modify registers.



Logic analyzers

- A logic analyzer is an array of low-grade oscilloscopes:





LEDs for debugging

- LEDs or other visualizing devices are very good for debugging
- Work like a "printf" statement in C



How to exercise code

- Run on host system.
- Run on target system.
- Run in instruction-level simulator.
- Run on cycle-accurate simulator.
- Run in hardware/software co-simulation environment.



Simulation and Co-Simulation

- There are simulators for hardware and software
- In an embedded system there is a need for co-simulation
 - Hardware and software are simulated together
 - Problem: HW and SW simulate at different events
 - HW: clock tick
 - SW: instruction
- Never forget that the real system is much faster than the simulator
 - Do not trust the simulator!



Manufacturing testing

- Goal: ensure that manufacturing produces defect-free copies of the design.
- Can test by comparing unit being tested to the expected behavior.
 - But running tests is expensive.
- Maximize confidence while minimizing testing cost.



Testing concepts

- **Yield:** proportion of manufactured systems that work.
 - Proper manufacturing maximizes yield.
 - Proper testing accurately estimates yield.
- **Field return:** defective unit that leaves the factory.



Faults

- Manufacturing problems can be caused by many things.
- **Fault model:** model that predicts effects of a particular type of fault.
- **Fault coverage:** proportion of possible faults found by a set of tests.
 - Having a fault model allows us to determine fault coverage.



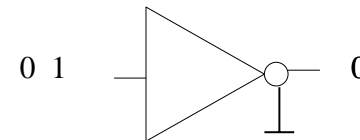
Software vs. hardware testing

- When testing code, we have no fault model.
 - We verify the implementation, not the manufacturing.
 - Simple tests work well to verify software manufacturing.
- Hardware requires manufacturing tests in addition to implementation verification.



Hardware fault models

- Stuck-at 0/1 fault model:
 - output of gate is always 0/1.





Combinational testing

- Every gate can be stuck-at-0, stuck-at-1.
- Usually test for single stuck-at-faults.
 - One fault at a time.
 - Multiple faults can mask each other.
- We can generate a test for a gate by:
 - controlling the gate's input;
 - observing the gate's output through other gates.



Sequential testing

- A state machine is combinational logic + registers.
- Sequential testing is considerably harder.
 - A single stuck-at fault affects the machine on every cycle.
 - Fault behavior on one cycle can be masked by same fault on other cycles.



Scan chains

- A scannable register operates in two modes:
 - normal;
 - scan---forms an element in a shift register.
- Using scan chains reduces sequential testing to combinational testing.
 - Unloading/unloading scan chain is slow.
 - May use partial scan.



Test generation

- **Automatic test pattern generation (ATPG)** programs: produce a set of tests given the logic structure.
- Some faults may not be testable---redundant.
 - Timeout on a fault may mean hard-to-test or untestable.



Boundary scan

- Simplifies testing of multiple chips on a board.
 - Registers on pins can be configured as a scan chain.

