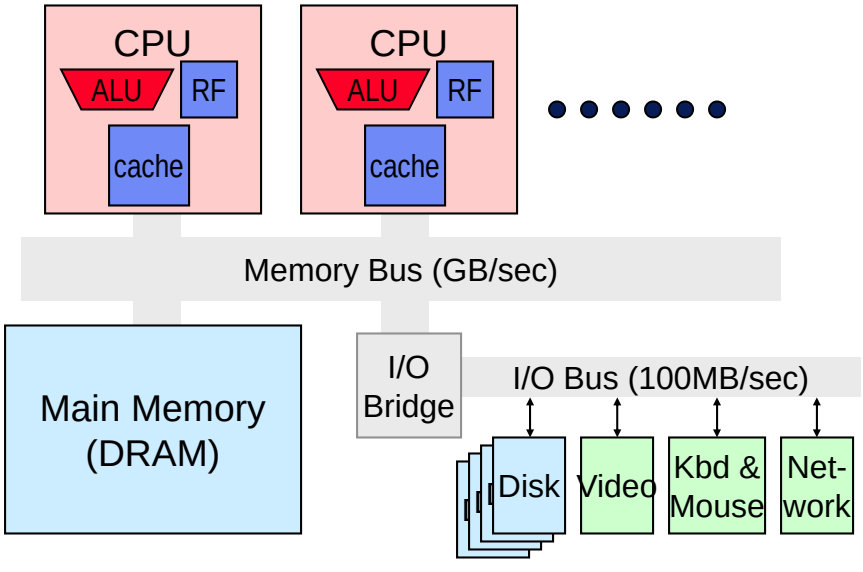
18-447 Lecture 25: Busses

James C. Hoe
Dept of ECE, CMU
April 27, 2009

Announcements: Project 4 due this week (no late check off)
HW 4 due today

Handouts: Practice Final Solutions

A Typical Computer Organization



Basic Bus Concepts

- A broadcast medium
  - a common datapath connecting multiple devices

    reducing interconnection cost
  - single driver, multiple receivers at a time
  - time-multiplexed by "transactions"

    bandwidth is shared
- Protocol
  - a set of handshake signals and rules of conduct to manage sharing by the bus devices
- Device types
  - masters: devices who initiate transactions
  - slaves: devices who respond to transactions

  a single device could have both master and slave functions
  - arbiter: a special bus device that manages shared bus usage

Bus Transactions

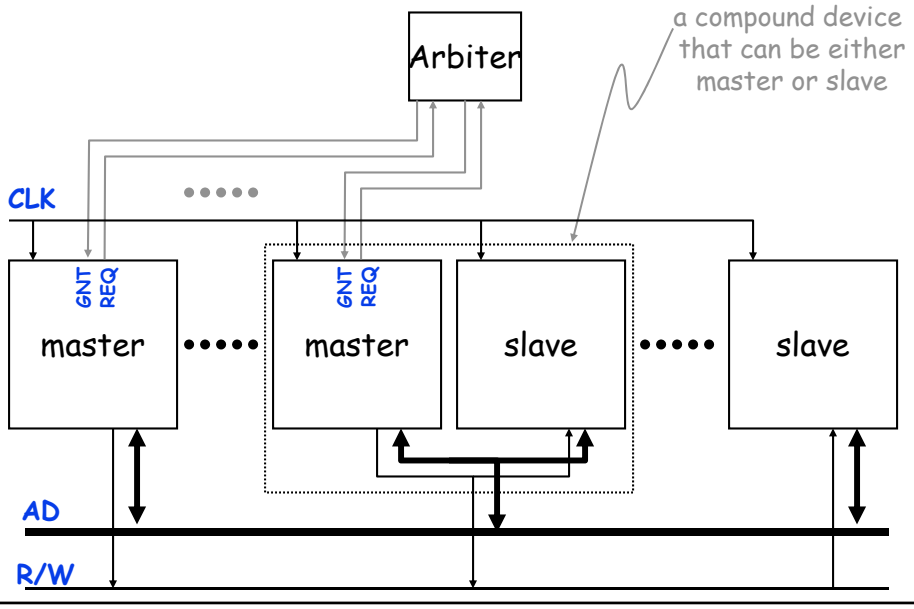
- Memory-like paradigm based on address, data, R/W
  - master issues read/write transactions to an address
  - each slave is assigned an address range to respond in a memory-like way
- Transaction phases
  - master requests ownership from arbiter
  - arbiter grants ownership to master
  - master drives address for all to see
  - a slave claims transaction
  - master (or slave) drives data (depending on read or write) for all to see
  - master terminates the transaction and bus ownership

Basic Bus Signals

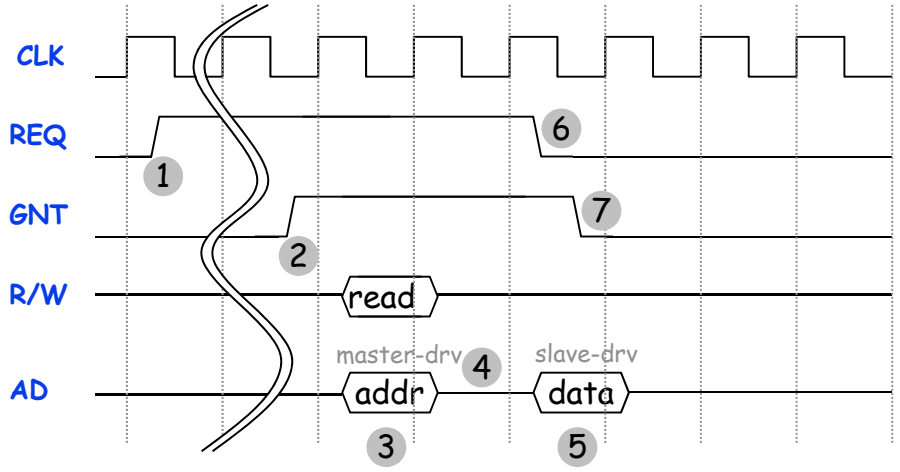
- **CLK**: all devices synchronized by a common clock
- Private signals to/from arbiter per master
  - **REQ** (output): assert to request ownership; de-assert to signal end of transaction
  - **GNT** (input): ownership is granted
- "Broadcast" signals shared by all devices
  - **AD** (address/data bus, bi-directional): master drives address during the address phase, master/slave drives data during the data phase

    why not have separate address bus and data bus?
  - **R/W** (bi-directional): bus commands, e.g. read vs. write

Bus Configuration

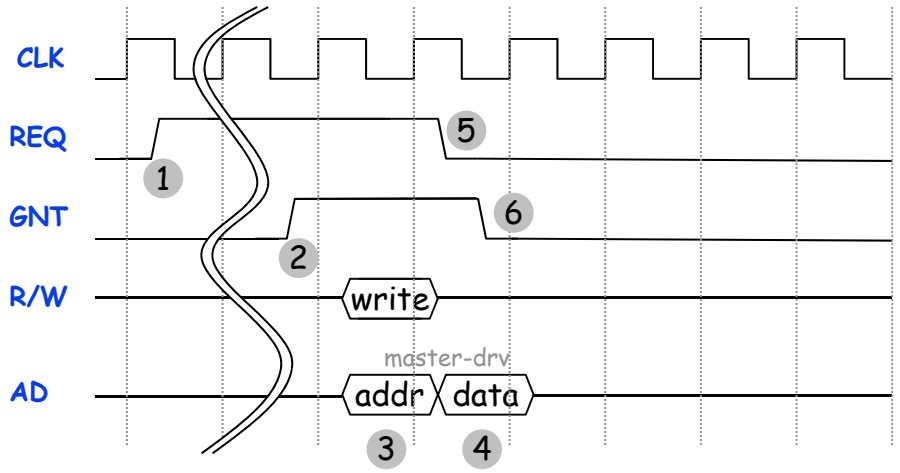


Simple Read Transaction



1. master requests bus
2. arbiter grants bus
3. master drives address/command, to be sampled on clock-edge
4. bus-turnaround cycle
5. slave drives data
6. master signals final cycle
7. arbiter acknowledges

Simple Write Transaction

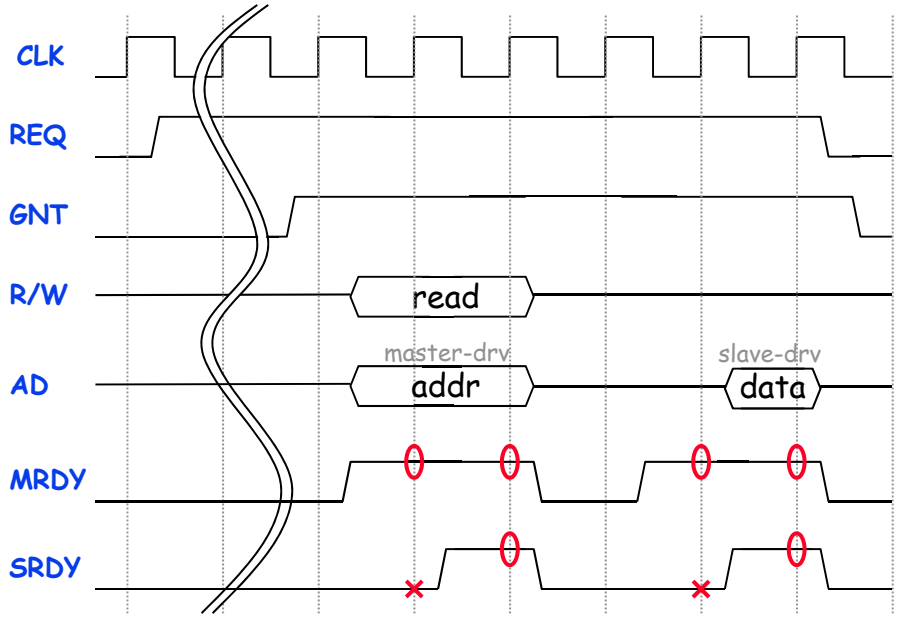


1. master requests bus
2. arbiter grants bus
3. master drives address/command, to be sampled on clock-edge
4. master drives data
5. master signals final cycle
6. arbiter acknowledges

Asynchronous Bus Protocols

- Previously, we assumed
  - slave can always decode address in 1 cycle
  - slave can always respond in 1 cycle
  - reasonable assumption for the "primary" bus where the allowed devices (processor, memory, chip-set) are restricted
  - unreasonable assumption for an "expansion" bus that has to work with both Gigabit ethernet and $2 sound cards
- Asynchronous handshaking
  - **REQ**/**GNT** is an example of asynchronous handshake
  - introduce **MRDY** and **SRDY** signals to coordinate **AD** usage
    - driver only asserts **RDY** when **AD** value is valid
    - receiver only asserts **RDY** when ready to accept **AD** value
  - a bus cycle is valid iff **MRDY**&&**SRDY**
    - receiver only pays attention if the driver is ready
    - driver repeats value until the receiver is ready
    - both driver and receiver can stall a transaction arbitrarily

Asyn Read Transaction



Bus Performance: Latency

- Request/Grant latency
  - a function of bus contention
  - a function of arbitration strategy
    - statically prioritized by expansion slots
    - FIFO, round-robin (and other so-called fair arbitrations)
- Transaction latency
  - a function of slave reaction time
  - synchronous protocols have less wastage but cannot allow variable latency slaves

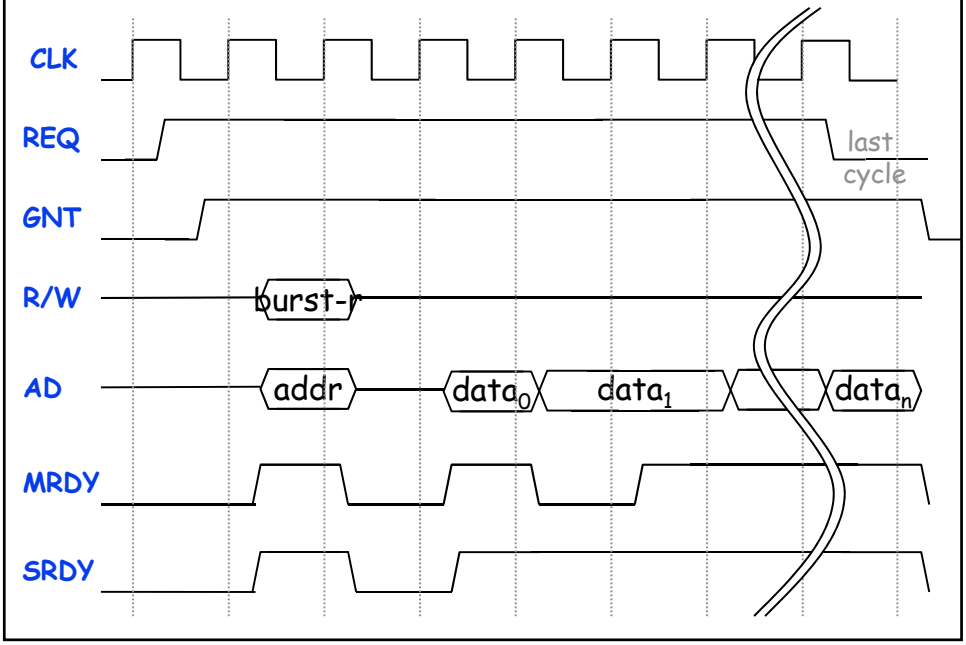
    advanced protocols can further shorten latency

Actual latency seen by the processor core is much longer than the raw bus latency because the processor core is biased to access the cache as quickly as possible at the expense of the infrequent need to go out to the bus

Bus Performance: Bandwidth

- Bandwidth (assume N-byte AD bus at frequency f)
  - peak bandwidth = $N \cdot f$ (what is often quoted)
  - must subtract overhead
- Each transaction has overhead cycles that don't contribute to data bandwidth
  - request and grant phases
  - address and claim phases
  - termination phase
- Best if these overhead cycles can be amortized over multiple data cycles
  - implied consecutive, successive addresses
  - fixed-sized burst on synchronous protocols (e.g., for transferring an entire cache line)
  - variable-sized burst on asynchronous protocols (e.g., for transferring to/from stream devices)

Burst Read Transaction (async)



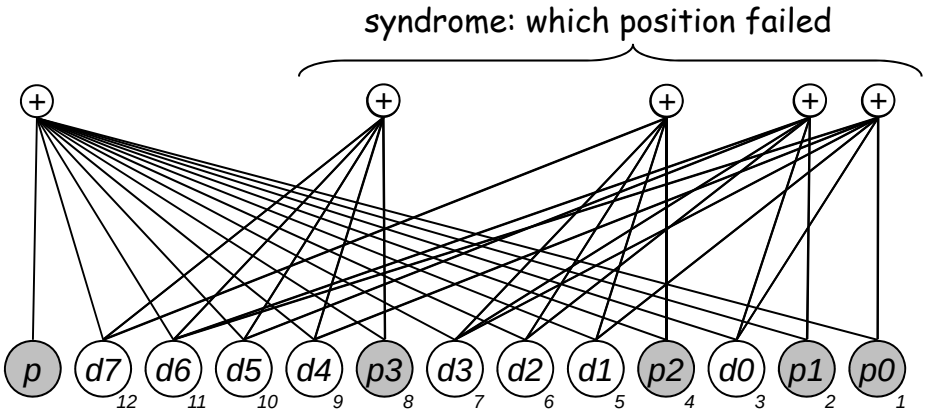
Error Detection

- Bus is not an electrically-friendly environment
  - long transmission lines with multiple taps that could either drive or receive
  - each device needs to switch a lot of external loads (e.g. **AD** bus) simultaneously
  - noises in and on the board
- **AD** bus transfers need to be protected by "information-redundant" coding to detect bit errors
- Even-parity scheme (most commonly used)
  - add an extra bit (the **parity** bit) to the **AD** bus
  - the driver set the **parity** bit such that XOR of all bus bits is 0
  - the receiver checks the parity of the received bus value

guaranteed to detect an odd number of bit flips but cannot detect an even number of bit flips or identify which bits flipped

SEC-DED

- Single error correction, double-error detection
  - insert parity bits in positions 2^i (count from 1)
  - each parity bit covers one-half of the bit positions, ones with i'th bit set in the index, e.g., p0 for all odd positions, p1 for all odd groups of 2, p2 for all odd groups of 4, etc.

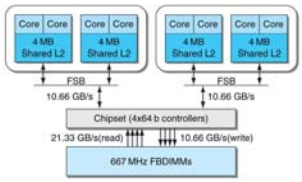


Advanced (Memory) Busses

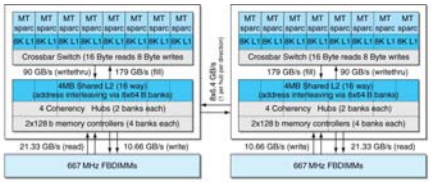
- Pipelined Bus
  - separate address and data busses
  - pipeline/overlap the request, address and data phases of 3 different transactions

    works best if all transactions have the same latency
- Out-of-order (split-phase) Bus
  - completely separate arbitration for the data bus
  - each address-bus transaction is assigned an unique tag and terminates without waiting for the data phase
  - when slave is read to respond, slave arbitrates for a data transaction on the data bus, using the tag to identify the master a slow slave doesn't stall the entire bus
- Switched Data Bus
  - with a split-phase protocol, the data bus doesn't have to be broadcast-based anymore
  - use a switched network to send data at dedicated/high BW

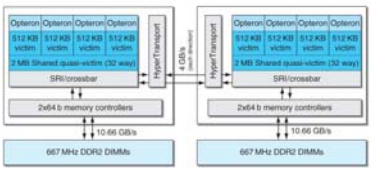
Some Current Examples



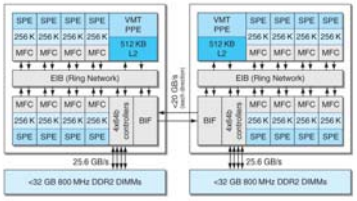
(a) Intel Xeon e5345 (Clovertown)



(c) Sun UltraSPARC T2 5140 (Niagara 2)



(b) AMD Opteron X4 2356 (Barcelona)



(d) IBM Cell QS20