Lecture 11: Speed & Communications

David Black-Schaffer
davidbbs@stanford.edu
EE183 Spring 2003

Overview

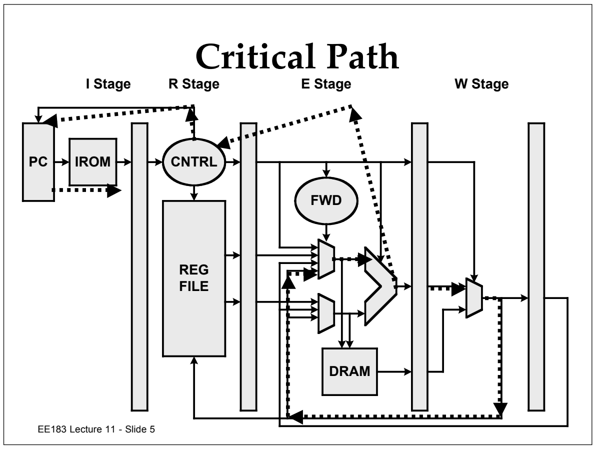
- Pipelining for Speed
  - Do a little bit of work in each stage (fast)
  - Use the pipe registers to break up the critical path
  - Remember that if you do less in each stage you need more stages, so it may not be faster overall
- Memory-mapped I/O
  - MUX the data lines based on the address
  - Use LOAD/STORE to special addresses to transfer data between external devices

Logistics

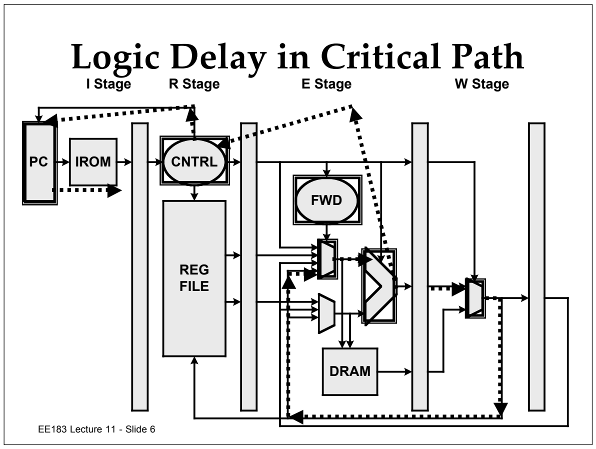
- Midterm this Wednesday 7-9pm in room 102
- This is the last EE183 lecture ever…

Today

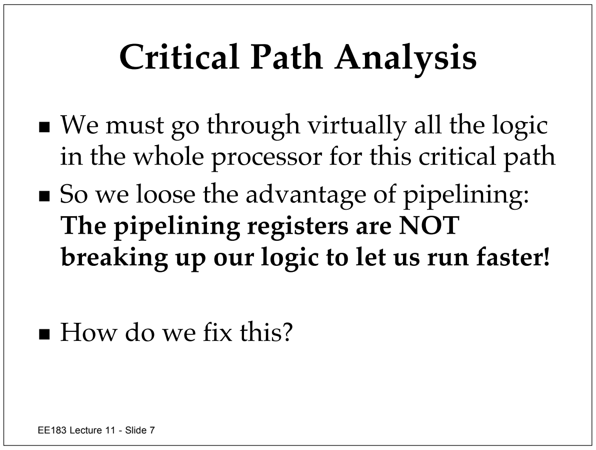
- Fixing your critical path
- Brief review of Memory-mapped I/O
- Issues with communications



Critical Path

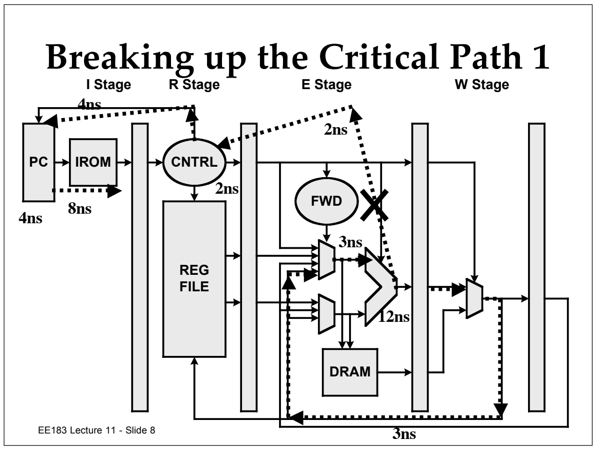


Logic Delay in Critical Path



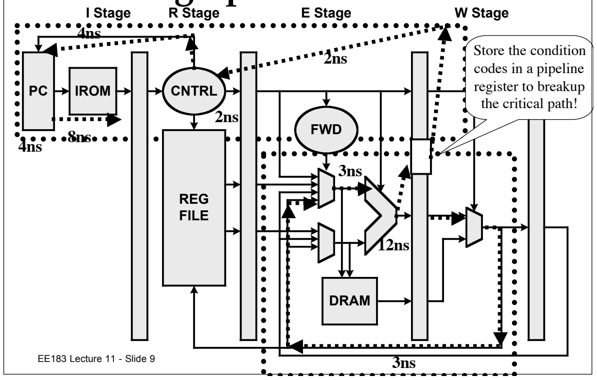
Critical Path Analysis

- We must go through virtually all the logic in the whole processor for this critical path
- So we loose the advantage of pipelining: **The pipelining registers are NOT breaking up our logic to let us run faster!**
- How do we fix this?



Breaking up the Critical Path 1

Breaking up the Critical Path 2



Store the condition codes in a pipeline register to breakup the critical path!

So what did we do?

- We stored the result of an intermediate calculation so we could breakup the critical path
- We now can run at a 2x faster clock speed, but we take an additional cycle
- Do we win?

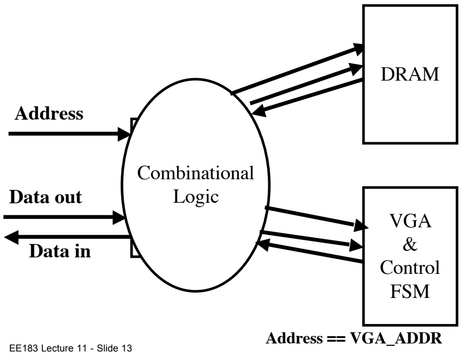
What you have to do

- If your design does not meet 50MHz you must go in and re-design your critical path to improve it
- You may run your design at 25MHz by dividing down the clock for the processor, but you may **only** cross the 25/50 boundary at a BRAM
  - Remember that the BRAM has two ports and they both have their own clocks
  - Run the processor side at 25MHz and the VGA side at 50MHz and you're okay
- Don't try to mix clocks if you can possibly avoid it

Memory-mapped I/O

- Add logic to look for LOAD/STORE to a particular address/range of addresses
- Re-route the signals to the external device
- Example:
  - If I do a STORE to 0xFFF then send that data not to the DRAM but to the VGA
  - If I do a LOAD from 0xFFD then take the data not from the DRAM but from the Timer

Memory-mapped I/O



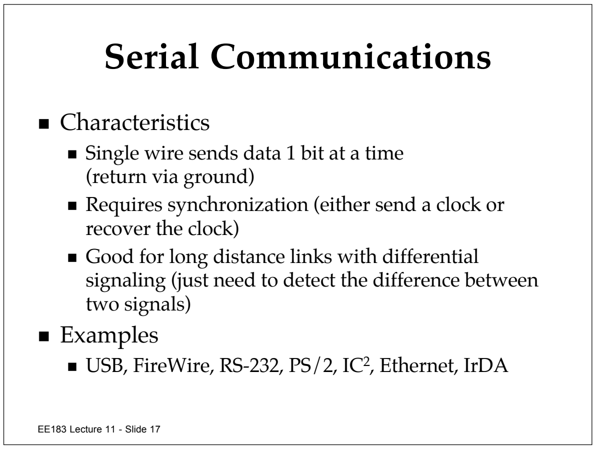
Implementation

- Simple: look at the RAM address bus and re-route the data and RE/WE signals based on the address (some MUXes)
- Have the external devices respond to these signals to send/receive data

Any Questions?

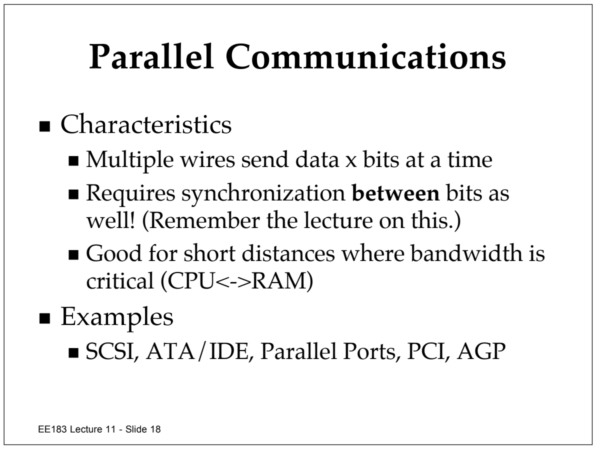
Communications

- Serial vs. Parallel
  - Serial: one bit at a time, slower, simpler
  - Parallel: many bits at a time, faster, more expensive
- Point-to-point vs. Distributed
  - Point-to-point: simple, but need multiple links to communicate to everything
  - Distributed: more expensive, but can add devices easily



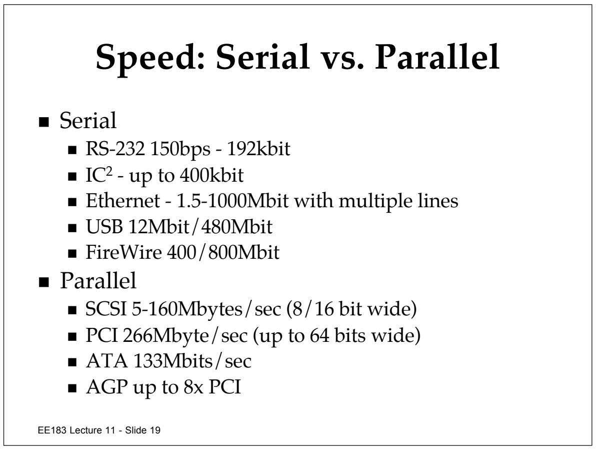
Serial Communications

- Characteristics
  - Single wire sends data 1 bit at a time
    (return via ground)
  - Requires synchronization (either send a clock or recover the clock)
  - Good for long distance links with differential signaling (just need to detect the difference between two signals)
- Examples
  - USB, FireWire, RS-232, PS/2, IC^2, Ethernet, IrDA



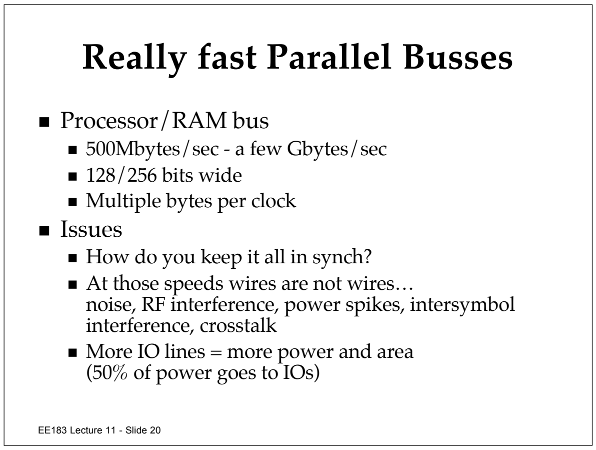
Parallel Communications

- Characteristics
  - Multiple wires send data x bits at a time
  - Requires synchronization **between** bits as well! (Remember the lecture on this.)
  - Good for short distances where bandwidth is critical (CPU<->RAM)
- Examples
  - SCSI, ATA/IDE, Parallel Ports, PCI, AGP



Speed: Serial vs. Parallel

- Serial
  - RS-232 150bps - 192kbit
  - IC^2 - up to 400kbit
  - Ethernet - 1.5-1000Mbit with multiple lines
  - USB 12Mbit/480Mbit
  - FireWire 400/800Mbit
- Parallel
  - SCSI 5-160Mbytes/sec (8/16 bit wide)
  - PCI 266Mbyte/sec (up to 64 bits wide)
  - ATA 133Mbits/sec
  - AGP up to 8x PCI



Really fast Parallel Busses

- Processor/RAM bus
  - 500Mbytes/sec - a few Gbytes/sec
  - 128/256 bits wide
  - Multiple bytes per clock
- Issues
  - How do you keep it all in synch?
  - At those speeds wires are not wires…
    noise, RF interference, power spikes, intersymbol interference, crosstalk
  - More IO lines = more power and area
    (50% of power goes to IOs)

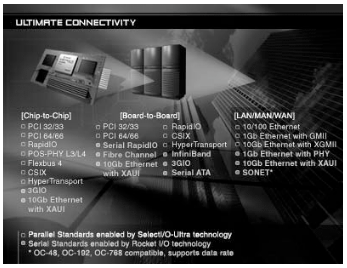
Where to go from here?

- **Everything is moving to differential serial communications (Serdes)**
  - 3-10Gbps over two wires
  - Handles both on-chip and off-chip signals
    - Why would you want it on chip? SoC!
  - Goes over both copper and fiber
- Examples:
  - USB, FireWire, Serial ATA, Fibre Channel, RapidIO, SONET, Rambus, Intel FSB, AMD Hypertransport

Why Serdes?

- Differential
  - Makes signal detection much easier despite interference (both lines see the same noise, so subtracting them gives you a much cleaner signal)
- Clock and Data Recovery (CDR)
  - Recovering the clock from the signals is possible which means you don’t have to send a separate synchronized clock
- Single-bit
  - Don’t have to synchronize with multiple other bits

What is available?



ULTIMATE CONNECTIVITY

[Chip-to-Chip]

○ PCI 32/33
○ PCI 64/66
○ RapidIO
○ POS-PHY L3/L4
○ Flexbus 4
○ CSIX
○ HyperTransport
● 3GIO
● 10Gb Ethernet with XAUI

[Board-to-Board]

○ PCI 32/33
○ PCI 64/66
● Serial RapidIO
● Fibre Channel
● 10Gb Ethernet with XAUI
○ RapidIO
○ CSIX
○ HyperTransport
● InfiniBand
● 3GIO
● Serial ATA

[LAN/MAN/WAN]

○ 10/100 Ethernet
○ 1Gb Ethernet with GMII
○ 10Gb Ethernet with XGMII
● 1Gb Ethernet with PHY
● 10Gb Ethernet with XAUI
● SONET*

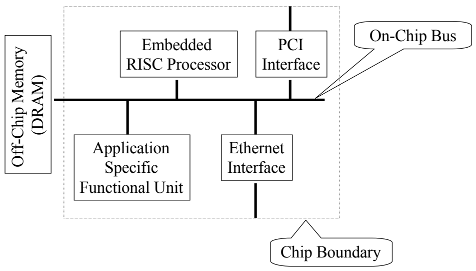
○ Parallel Standards enabled by SelectI/O-Ultra technology
● Serial Standards enabled by Rocket I/O technology
* OC-48, OC-192, OC-768 compatible, supports data rate

P2P vs. Distributed

- Point-to-point
  - Good for maximum performance
  - Pipelines, data paths, specialized units
  - Very complicated to connect everything this way
- Distributed
  - Good for multiple discrete functional units
  - Standardized interface
  - Standard network packet routing
- When to use which?
  - System on a Chip design: on-chip networks to route data
  - Custom processors: point-to-point for best throughput
  - What if it takes more than a cycle to get across your chip?

SoC Design

Prevalent Design Methodology (look familiar?)



Things to take home

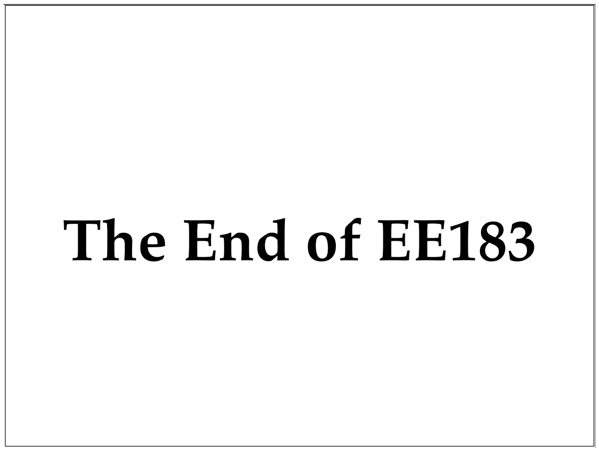
- Everything is moving to serial differential communications with clock recovery
- Transistors are free, so on-chip networks make more sense than trying to route large busses between every two points

Midterm

- Don’t forget the midterm this Wednesday.
- 7-9pm in SEQ 102 (next door)
- 45 minute “quiz”
- On the material in the lectures

Lecture 11 Key Points

- Use the pipeline to break up long (slow) calculations
- MUX the RAM data lines based on the address to interface to external devices
- Serial is the way of the future for communications
- Logistics
  - **Midterm Wednesday from 7-9pm in 102**



The End of EE183