

Lecture #9-10: Communication Methods

Kunle Olukotun
EE183
February 10, 2003

Lab Issues

- Lab 2 Writeup is due tonight at Midnight
- Lab 3 Questions?
 - The starter is available on the web
- ASM183
- Modelsim
 - Verilog simulation without synthesis
 - Tutorial and installation instructions EE183 webpage

Final Lab

- Start thinking about final lab now!
- Need 1 paragraph description by Feb 17
- Email to David and I
- Project meetings/feedback by Feb 24

Lab	Pre-Lab Report Due	Demo Due by 5 pm	Final Report Due
1	Jan 17	Jan 24	Jan 27
2	Jan 31	Feb 7	Feb 10
3	Feb 14	Feb 21	Feb 24
4	Feb 28	Mar 14	Mar 17

Communication methods for digital systems

- Communication methods
 - wires and signaling conventions used to transmit data between digital devices
 - we'll only deal with digital baseband communication
 - other methods include radio freq. (RF), infra-red (IR), freq. modulation (FSK), etc.
- Outline
 - Options
 - Trends
 - PCI example

Communication Options

- Orthogonal elements of communication methods
 - number of wires - serial or parallel
 - speed – bits/bytes/words per second
 - timing methodology – synchronous or asynchronous
 - number of destinations/sources – single or multiple
 - arbitration scheme – daisy-chain, centralized, distributed
 - *Copper or Fiber* ← New one!

Communication/Packaging Hierarchy

- On-Chip
- On a board
- Between Boards
- Between Racks/Boxes → Inside a data center
- How far you have to go makes a big difference on what technology you select
 - Crossing a level of the hierarchy is a big deal
 - All solutions could be used at any level
 - Engineering experience helps you decide

One Wire

- Serial
 - single wire to transmit information one bit at a time
requires synchronization between sender and receiver
sometimes includes extra wires for clock and/or
handshaking
 - good for inexpensive connections (e.g., terminals)
 - good for long-distance connections (e.g., LANs)
 - e.g.: RS-232, Ethernet, Apple desktop bus (ADB),
Philips inter-integrated circuit bus (I2C)
- Is this really one wire?
 - How does the current get back to you?

Many Wires

- Parallel
 - multiple wires to transmit information one byte or word
at a time
 - good for high-bandwidth requirements (CPU to
memory)
 - more expensive wiring/connectors/current requirements
 - e.g.: SCSI, EISA bus (PCs), NuBus (Mac), PCI (in
newer PCs and Macs), PCMCIA (in laptops)

Speed/Bandwidth

- **Serial**
 - low-speed, cheap connections – RS-232 1.2K - 38Kbits/sec, copper wire
 - medium-speed efficient connections – I2C 0 - 400Kbits/sec, board traces
 - high-speed, expensive connections – Ethernet 1.5 - 10Mbytes/sec, co-axial cable
 - Today: 100 Mbit ethernet over twisted pair(telephone line)
- **Parallel, low-medium-speed, not too wide**
 - EISA bus, 20Mbytes/sec, 16 bits wide
 - SCSI bus, 5 - 40Mbytes/sec, 8 bits wide
 - NuBus, 40Mbytes/sec, 32 bits wide
 - PCI, 266 Mbytes/sec, up to 64 bits wide

Speed/Bandwidth

- **Parallel, high-speed, very wide**
 - memory systems in large multi-processors
 - DRAM interfaces
 - 500Mbytes/sec–few Gbytes/sec, 128 bits wide
 - LDT (AMD) is bringing this to PC
- **Issues**
 - length of the wires (attenuation, noise, capacitance)
 - environment (RF interference, noise)
 - current switching (spikes on supply voltages)
 - number and types of wires (cost of connectors, cross-talk)

Timing methodology

- Asynchronous
 - fewer wires (no clock)
 - clock recovery
 - no skew concerns
 - synchronization overhead
 - appropriate for loosely-coupled systems (CPU and peripherals)
 - common in serial schemes
- Synchronous
 - Clock wires and skew concerns
 - no synchronization overhead
 - can be high-speed if delays are small and can be controlled
 - appropriate for tightly-couple systems (CPU and memory)
 - Common in parallel schemes

Timing Issues

- clock period and wire delay
- synchronization and skew
- power consumption (directly related to amount of switching on high-capacitance wires)
 - Half of ASIC power can be in IOs

Number of devices communicating

- Single source – single destination
 - point-to-point
 - cheap connections, no tri-stating necessary
- Single source – multiple destination
 - fanout limitations
 - addressing scheme to direct data to one destination
- Multiple source – multiple destination
 - arbitration between senders
 - tri-stating capability is necessary
 - addressing scheme
 - priority scheme
 - fairness considerations

Arbitration schemes

- Daisy-chain or token passing
 - devices either act or pass to next
 - fixed priority order (leads to fairness issues via starvation)
 - as many wires as devices
- Centralized
 - request to central arbiter
 - central arbiter implements priority scheme
 - wires from/to each device can be costly
 - can be dynamically changing priority/fairness
- Distributed
 - no central arbiter
 - common set of wires driven and observed by all devices
 - fixed priority/fairness scheme

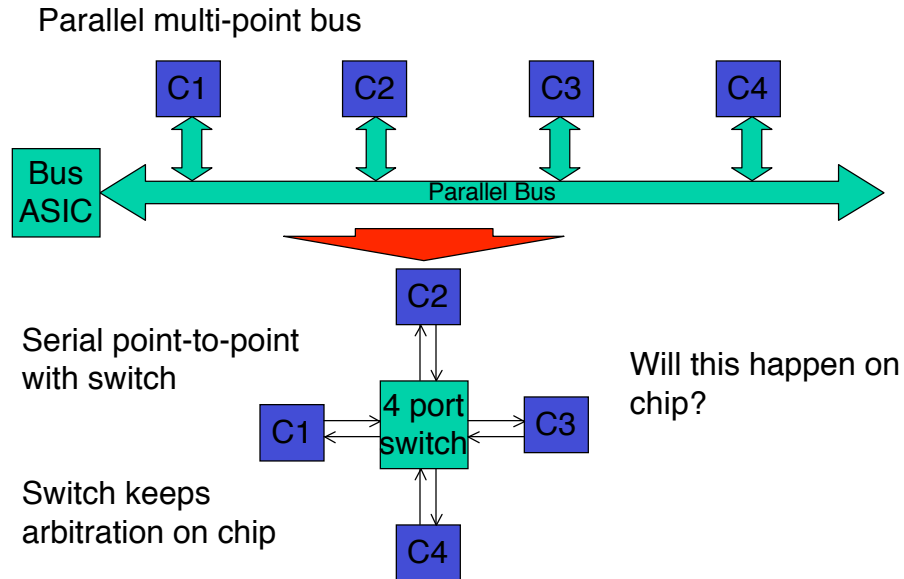
“Speeds and Feeds”

- Moore’s Law is only useful if you can get the bits into and out of the system
- Traditional engineering tradeoff is that parallel is preferred for short distances
- However, parallel wires have crosstalk and intersymbol interference at the speeds we want—in short they look analog

Trends

- number of wires → Serial via serdes (serializer, deserializer)
- timing methodology → Clock and Data Recovery (CDR)
- speed → 3Gbps per differential pair
- number of destinations/sources → point-to-point
- arbitration scheme → currently centralized but moving to distributed
- Copper or Fiber → Both, depends on distance—between racks/boxes is often fiber

Trends for Boards and Beyond



Plethora of Standards

- Xilinx Virtex Series IO Capabilities

ULTIMATE CONNECTIVITY

[Chip-to-Chip]	[Board-to-Board]	[LAN/MAN/WAN]
<ul style="list-style-type: none"> □ PCI 32/33 □ PCI 64/66 □ RapidIO □ POS-PHY L3/L4 □ Flexbus 4 □ CSIX □ HyperTransport ● 3GIO ● 10Gb Ethernet with XAU1 	<ul style="list-style-type: none"> □ PCI 32/33 □ PCI 64/66 ● Serial RapidIO ● Fibre Channel ● 10Gb Ethernet with XAU1 □ RapidIO □ CSIX □ HyperTransport ● InfiniBand ● 3GIO ● Serial ATA 	<ul style="list-style-type: none"> □ 10/100 Ethernet □ 1Gb Ethernet with GMII □ 10Gb Ethernet with XGMII ● 1Gb Ethernet with PHY ● 10Gb Ethernet with XAU1 ● SONET*

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

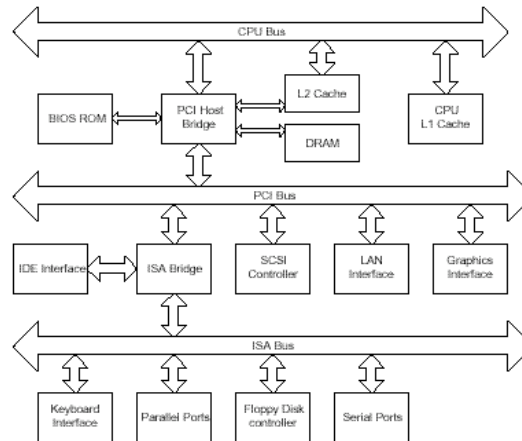
The Serialization of Communications

- Pin count is a key limiter for systems today
 - Need to route them to destination
 - Serdes power and knowhow is still an issue
- Many standards are moving to serdes
 - 3GIO(Pci)
 - Serial ATA (IDE hard disk)
 - AMD Hypertransport versus Intel Front Side Bus (FSB)
- EE273 has lots more info

Case Study: PCI Bus

- Origin of PCI Bus
 - first version of the specification was released in June 1992
 - The current specification of the PCI bus is revision 2.2, which was released in February 1999
- PCI is widely used
 - All PC systems today contain PCI slots,
 - Apple and workstations (SUN, SGI, HP)
 - Other bus standards based on PCI
- Benefits of PCI
 - Peak rate of 64 bit data path and 66MHz clock 524MBytes/sec.
 - 32 bit, 64-bit and 33MHz, 66 MHz versions, 133 MHz
 - Expandable to a large number of slots using PCI to PCI bridges.
 - Low Power 0-66MHz
 - Auto-configuration using software (no jumpers)

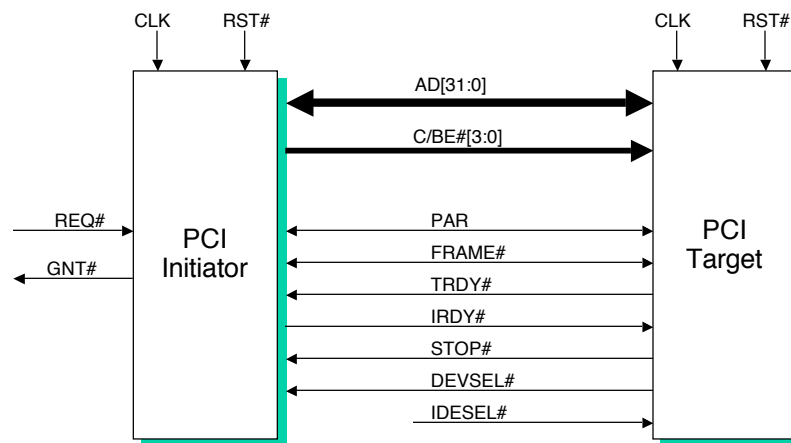
PCI in a System



PCI Terminology

- Transaction
 - Address phase
 - Claiming transaction
 - Data phase(s)
- Burst Transfer
 - Transaction with two or more data phases
- Initiator
 - Bus master
- Target
 - Device addressed (slave)

32 bit PCI Signals



PCI Signals

- System
 - CLK
 - RST#: global async. reset signal
- Address/data/command
 - AD : multiplexed address/ data
 - C/BE#: multiplexed command/ byte enable
- Interface control
 - PAR: even parity signal
 - FRAME#: start transaction, end transaction
 - IRDY#: master ready to transfer data
 - TRDY#: target ready to transfer data
 - DEVSEL#: target must acknowledge transaction in 4 cycles
 - STOP#: used by target to abort transaction
 - IDSEL#: used by master for system configuration

PCI Signals (cont)

- Arbitration
 - REQ#: used by bus master to request bus
 - GNT#: used by arbiter to grant bus to a particular master
 - bus granted: when GNT# asserted and bus is idle
 - bus idle : FRAME# and IRDY# de-asserted
- Arbitration Algorithm
 - No algorithm given in PCI spec.
 - Arbitration must be fair enough avoid deadlocks and starvation

PCI Commands

Commands set using C/BE# signals

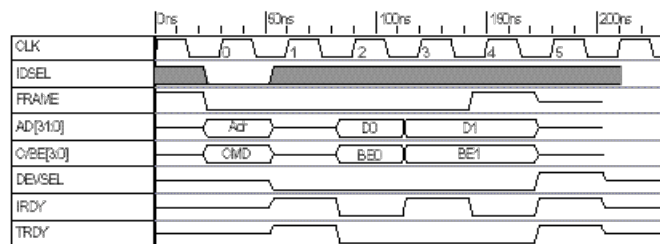
- Memory read
 - Read one or more locations from target memory
- Memory read line
 - Read a cache line
- Memory read multiple
 - Read one or more cache lines
- Memory write
 - Write one or more locations in target memory
- I/O read and write
 - Read or write target I/O space
- Configuration read and write
 - Read or write target configuration space

PCI Configuration Space

- 256 bytes long
- PCI standard defines first 64 bytes

Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address Register 0				10h
Base Address Register 1				14h
Base Address Register 2				18h
Base Address Register 3				1Ch
Base Address Register 4				20h
Base Address Register 5				24h
CardBus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved				34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

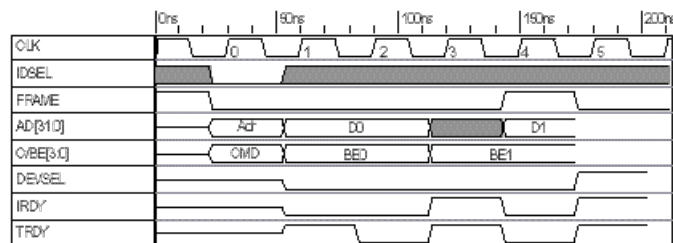
PCI Read



PCI Read Explanation

1. Master begins the transaction by driving FRAME# low, the requested address on AD[31:0], and the requested command on C/BE#[3:0] (I/O Read, Memory Read, Memory Read Line, or Memory Read Multiple).
2. At Clock 1 the Target responds by asserting DEVSEL# low.
3. At clock 2 both IRDY# and TRDY# are low, so the 1 st data word is read (D0).
4. At clock 3 the Master is not ready to accept the new word, since IRDY# is high. The Target is ready to transfer the next word since TRDY# is low. Since not both are low, no data is transferred.
5. At clock 4 the master is able to receive data again, since IRDY# is low, and the 2nd data word is transferred. Since FRAME# was high during this last cycle, both the Master and the Target end the cycle.
6. At clock 5 the Target tri-states AD[31:0] (turnaround), and drives TRDY# high. The Master also drives IRDY# high. This is done since IRDY# and TRDY# are sustained-tri-state lines

PCI Write



PCI Write Explanation

1. At Clock 0 the Master begins the transaction by driving FRAME# low, the requested address on AD[31:0], and the requested command on C/BE#[3:0] (I/O Write, Memory Write, or Memory Write Line).
2. At Clock 1 the Target responds by asserting DEVSEL# low. Since this is a write transaction, the master keeps driving AD[31:0] (this time with data. IRDY# is driven low to indicate data is ready for write. Since the Target is driving TRDY# high, it is not ready to accept the data. C/BE#[3:0] are loaded with the appropriate byte enables.
3. At clock 2 both IRDY# and TRDY# are low, so the 1st data word is written (D0).
4. At clock 3 the Master is not ready to send a new word (as indicated by the invalid content of AD[31:0]), so IRDY# is driven high. The Target is ready to accept the next word since TRDY# is low, but since IRDY# is high, no data transfer takes place.
5. At clock 4 the master is able to write data again, since IRDY# is low, and data is transferred. Since FRAME# was high during this last cycle, both the Master and the Target end the cycle.

Split-Transaction Bus

- A *split-transaction* bus splits each transaction into two largely independent parts, a request (address part) and a reply (data part for reads). Replies may appear in any order.
- Just as masters arbitrate to initiate a transaction on the address bus, slaves must now arbitrate to put their reply on the data bus.
- Other transactions may intervene
 - Improves bandwidth dramatically
 - Response is matched to request
 - Buffering between bus and cache controllers

Communications is Complicated

- Great opportunity for errors since your design is by definition talking to someone else's design
- “Be forgiving on the inputs and conservative on the outputs”
 - Some smart CS person
 - Applies to both logical and physical layer