Computer Architecture Midterm Notes:

1.0) At a top level, a computer consists of **CPU (central processing unit), memory, and I/O components**, with one or more modules of each type. These components are interconnected in some fashion to achieve the basic function of the computer, which is to execute programs.

1.1) Thus, at a top level, we can characterize a computer system by describing:

1.11) The external behavior of each component, that is, the data and control signals that it exchanges with other components

1.12) External Behavior is defined by **Data and Control Signals**

1.2) the interconnection structure and the controls required to manage the use of the interconnection structure.

1.3) **Von Neumann Architecture:**

Virtually all contemporary computer designs are based on concepts developed by J**ohn von Neumann** at the Institute for Advanced Studies, Princeton. and is based on three key concepts:

1.31) Data and instructions are stored in a **single read–write memory**.

1.32) The contents of this memory are **addressable by location, without regard to the type** of data contained there.

1.32) **Execution occurs in a sequential fashion** (unless explicitly modified) from one instruction to the next.

1.4) With General Purpose (Van Neuman) Hardware, input into the system is data and control signals, and output is a process.

1.41) Programs are a series of arithmetic or logical operations performed on some data. At each step control signal(s) must be executed.

1.412) Programs, however, do not invariably executed sequentially; it may jump around (e.g., the IAS jump instruction).

1.413) Operations on data may require access to more than just one element at a time in a predetermined sequence.

1.414) **Memory, or Main Memory,** to distinguish it from external storage or peripheral devices, is the place to store temporarily both instructions and data.

1.4141) Von Neumann pointed out that the same memory could be used to store both instructions and data.

1.5 The CPU Memory registers:

1.51 **Memory Address Register (MAR)**

1.511) Specifies the address in memory for the next read or write

1.52) **Memory Buffer Register (MBR)**, which

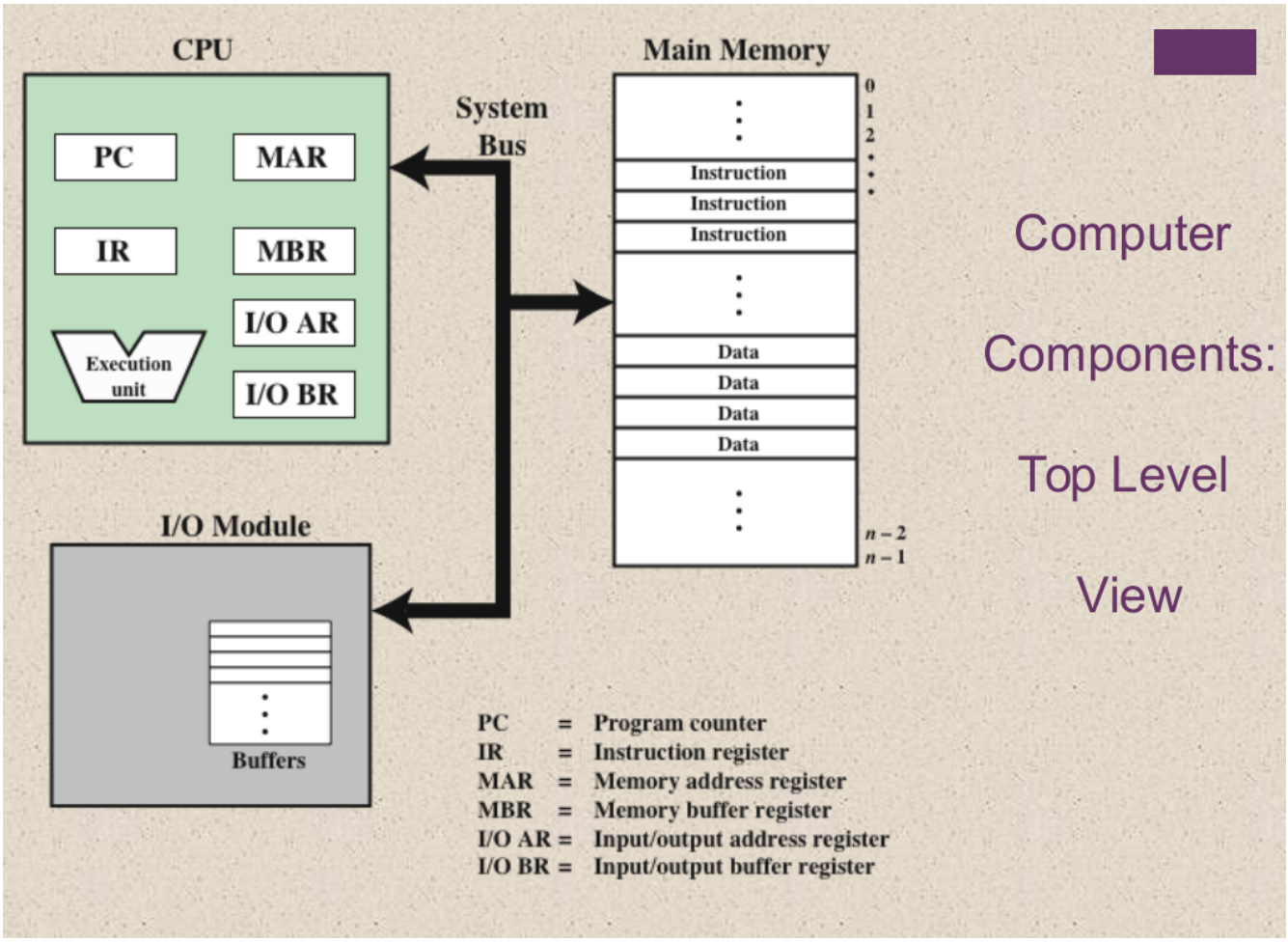
1.521) contains the data to be written into memory or receives the data read from memory.

1.53 **I/O address register (I/OAR)**

1.531) Specifies the address for a particular I/O device.

1.54) **I/O buffer (I/OBR) register**

1.541) contains data for the exchange of data between an I/O module and the CPU.



2.0 Basic Instruction Cycle

2.01 Fetch and Execute Cycle:

2.1 PC = Program Counter: A register in the CPU that holds the address for the next Memory to be fetched. It is autoicremented unless an instruction changes its stored address. It autoincrements in Bytes (a 8 bit machine autoincrements 8 bits at a time).

2.2 IR = Instruction Register: A register in the CPU that holds the data for the instruction that the CPU is to take.

2.4 AC = Temporary CPU memory storage

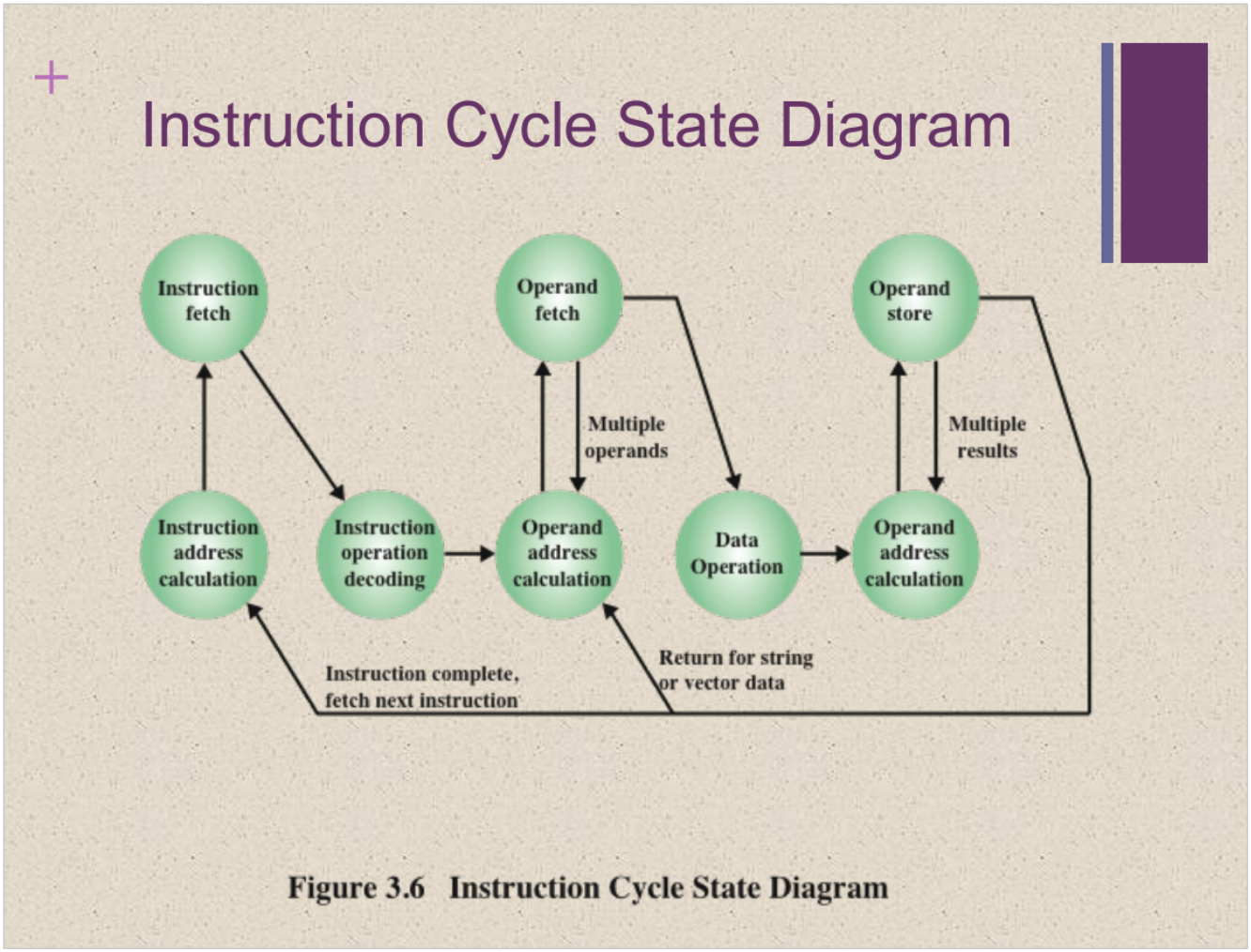
2.5 Fetch Cycle:

- 2.51 Read the PC
- 2.52 Increment the PC
- 2.53 Fetch the at the memory address that was read from the PC and store it in the IR
- 2.54 Interpret data in the IR and execute it
  - 2.541 Four Types of Actions that might be executed:
    - .5411: Processor-memory: Move data to and from processor and main memory
    - .5412: Processor-I/O: Move data from and to in and out of I/O peripheral through the I/O Module
    - .5413: Data Processing: Arithimatic or Logic Operation on Data
    - .5414: Control: Change the sequence of operation by altering the PC

2.6 Instruction Formats: Instruction formats can be varied but they usually have an optcode segment and a memory location segment. For example, you can have an instruction format that is a 16 bit word that can be stored in memory and fetched by the CPU and stored in an IR. A 16 bit word instruction might have 4 bits of **optcode** and 12 bits of to describe memory locations. Therefor, under this scenario, the CPU can perform 2^4 (16) optcodes and address 2^12 (4096) memory locations directly.

2.7 Generalized Instruction Cycle:

2.701: In General, execution cycles and optcodes might require more than one access to memory, or access to i/o modules. The Generalized Instruction Cycle can therefor be better diagrammed as follows:



This is a state diagram of the Instruction Cycle:

2.7011 *State Diagram: State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented in series of events, that could occur in one or more possible states. Hereby "each diagram usually represents objects of a single class and track the different states of its objects through the system".[1]*

State diagrams can be used to graphically represent finite state machines. This was introduced by C.E. Shannon and W. Weaver in their 1949 book "The Mathematical Theory of Communication". Another source is Taylor Booth in his 1967 book "Sequential Machines and Automata Theory". Another possible representation is the State transition table.

https://en.wikipedia.org/wiki/State_diagram

2.71 Instruction Cycle States: For any Instruction Cycle, not all states are used and some can be used more than one time.

2.72 **State Descriptions:**

2.721 **Instruction Address Calculation (IAC)**: Adding the appropriate integer to the PC Counter which determines the address of the next instruction to execute/

2.722 **Instruction Fetch (IF)**: Read an instruction from a memory location to the processor

2.723 I**nstruction Operation Decoding: (IOD)**: Determine the operation to be preformed the operands to be inputted into the operation.

2.724 **Operand Address Calculation (OAC):** Determining the address location for operands in memory or from I/O

3.0 **Interupts:** The execution of the instruction cycle is serial and predetermined by the instructions of the program. However, there are times when the serialization of the instructions need to be interceded in order to allow for parallelization of computing resources or other hardware issues. There are four classes of interupts:

3.01: Program: As a result of some execution, and interrupt occurs, most commonly due to a programmatic error. Examples include:

3.011 Arithmetic Overflow

3.012 Divide by Zero

3.013 illegal machine instruction

3.014 reference to memory location outside of userspace

3.02: Timer: CPU initiated interrupt generated by the CPU timer that allows some kind of system maintenance or prioritization of tasks

3.03: I/O : Generated by an I/O controller which stops execute in order allow for messaging to be received from the I/O modules. Without such interruptions, programming execution must wait in real time for disk writes and network socket routines and other slow routines and are operating system controlled.

3.04: Hardware Failure: including parity errors, power problems, are hard drive failures.

.

3.1 Interrupt and control transfer: For the purposes of I/O processing and speeding the program execute involves reaching an interrupt, handing control to the interrupt module, going back to normal execution, receiving an interrupt from the I/O module when it is module is finished writing or reading, and then continuing.

Transfer of Control via Interrupts

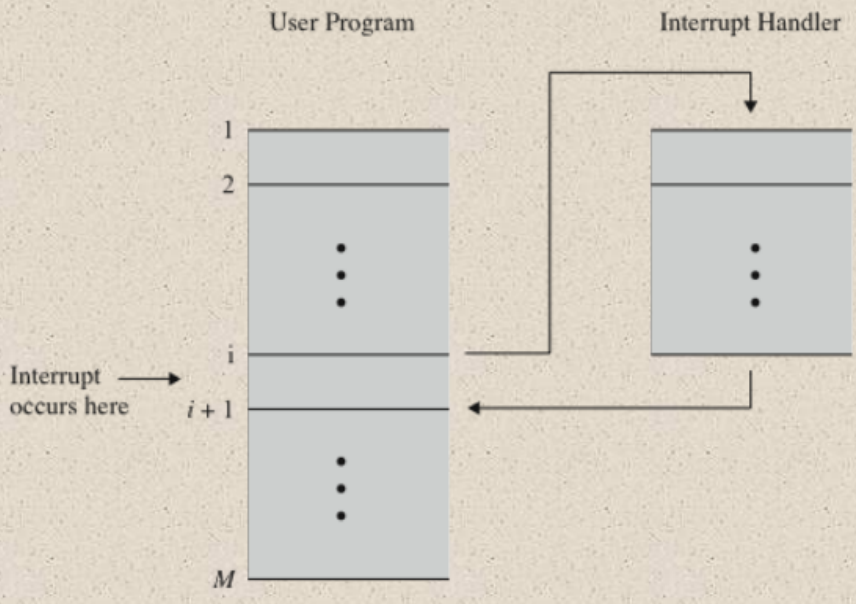


Figure 3.8 Transfer of Control via Interrupts

3.11 Interrupt States: Interrupts and the executing program must interface with each other. The operating system promotes this interaction to allow for multitasking and speeder program execution. Interrupts can be long and short.

3.111 Short Interrupts: These occur when an interrupt happens it returns back to a program prior to another trigger to an interrupt. For example, when a write command is executed, it triggers an interrupt from the I/O module of the operating system. The program is stored away, and the interrupt is begun as the PC index is changed to the location of the interrupt handler. The handler acquires what information it needs, and hands it off to the I/O module, and then returns the program to its preserved state, which the I/O module works on writing to the peripheral device (like a hard drive). The program continues until it is interrupted again, this time by the interrupt handler to give the write results to the program. Therefore several execution cycles of the program can progress which the hardware is busy storing the data it acquired.

3.112 Long Interrupts: Similar to short interrupts, however, multiple interrupt triggers are encountered prior to the interrupt handlers return to the program of successful I/O results. In that case, new I/O processes need to be blocked. Write requests, for example, get cued.

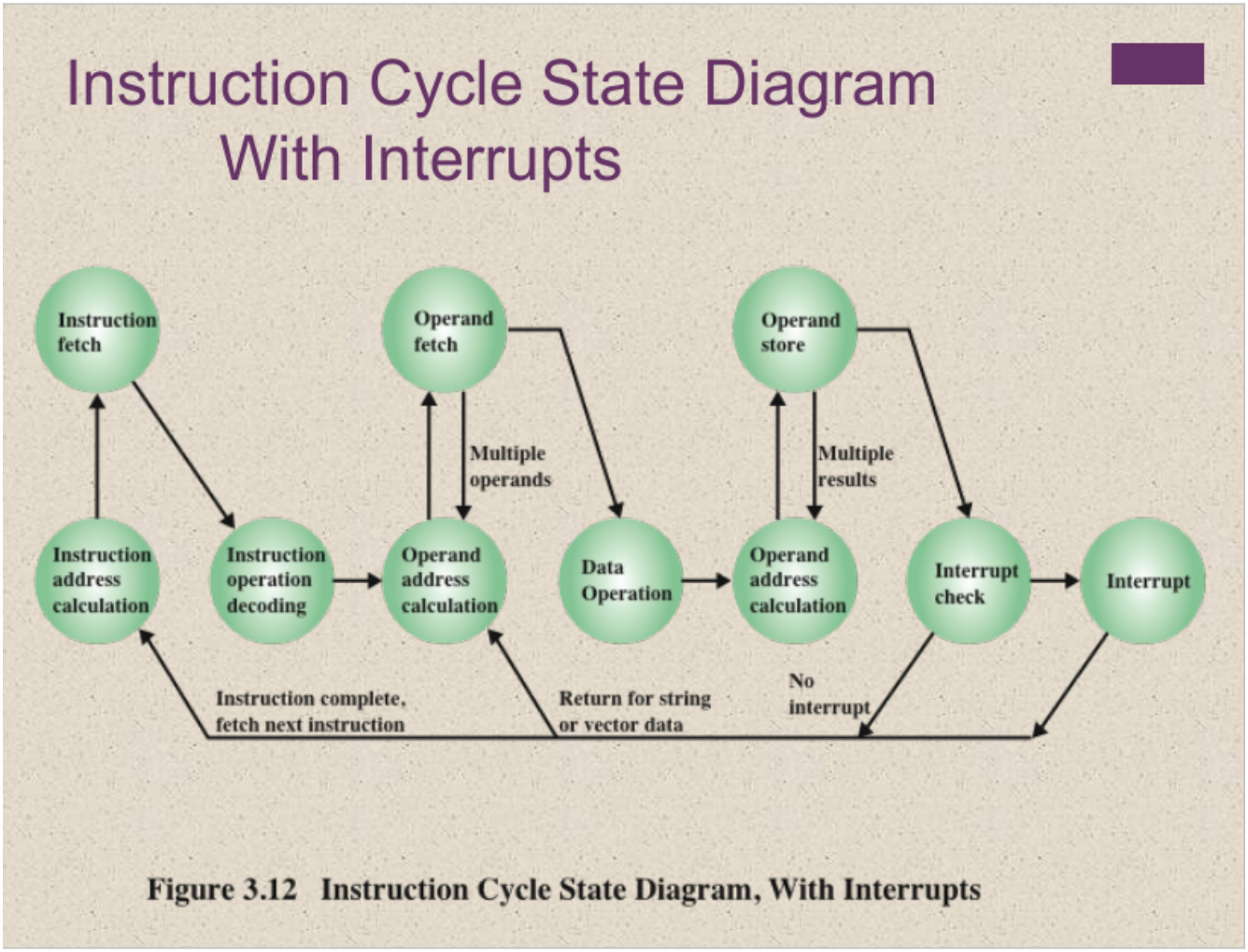


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

3.113 The Interrupt cycle is added to the instruction state diagram to account for interrupts and program caches.

3.114 Multiple Interrupt Strategies:

.1141 **Disabled Interrupt**: The Processor can and will ignore interrupts until interrupts are enabled. Once re-enabled, any later interrupts are processed in order.

.1142 **Priority Interrupt:** In this case, interrupt processes themselves can be interrupted by higher priority interrupts. When the higher interrupt, serviced by an **Interrupt Service Routine**, is finished, the CPU drops back to the stack of running ISR's. Each interrupt is handled according to its priority. Incoming higher priorities are handled first, and lower priority interrupts are cued.

4.0 **Input and Output Functions:**

4.01 Interrupt functions are handled by I/O Modules

4.02 I/O Modules can exchange data directly with the CPU.

4.03 Processor identifies a module associated with a device to read and write from

4.04 Processor uses module I/O instructions similar to memory Optcodes and Addresses, but for an I/O Device Module

4.05 Direct Memory Access - The Processor can give the I/O module direct access to the memory for reading or writing, those freeing the processor (See interrupts notes above)

5.0 Data Communication and Computer Buss:

5.01 Memory to Processor: Through Memory Words with serial addresses

.001 Read and Write Control signals

5.02 Processor to Memory

5.03 I/O to Processor via an I/O Module

.031 An I/O Module might control more than one device

.031 Each interface to a device is a port

.032 External Data Paths

.033 Capable of interrupt signals

5.03 Processor to I/O to the device

5.04 I/O to or from Memory : DMA

5.05 Processor Interface:

.051 Reads Instructions

.052 Reads Data

.053 Uses Control Signals to control operations of the system

.054 Receives Interrupts

5.1 Two General kinds of interconnection structures:

5.11 The **Bus and Multiple Bus Structures**

5.112 Bus: a communication pathway between two devices

5.113 **Shared transmission medium**

5.114 A signal to one device is available to all devices on the Bus

5.115 Only one device at a time can talk on the bus, otherwise the message becomes garbled

5.116 Data Lines provide a Path for Data moving among system modules.

5.117 System Bus: A bus that contains major components (processor, memory, I/O)

5.118 Each line can carry one bit at a time

(cont)

5.119 Data Bus Width: The system bus typically consists of approximately 50 to 100 separate lines which are grouped into subunits of 32,64, 128 or even more of these separate lines to provide parallel communication of word sizes across the system.

5.11.10 **Address Lines** : Used to designate the source or destination of data for memory or I/O devices

5.11.101 Address Line Width determines the maximum capacity for memory addressing

5.11.11 **Control Lines:** Command and Timing Signals that control access to the address bus and the data bus

111 Timing Signals indicate the validity of data and address information

112 Command Signals specify which operations to perform

113 Typical Control Lines:

1131 Memory Write

1132 Memory Read

1133 I/O Write

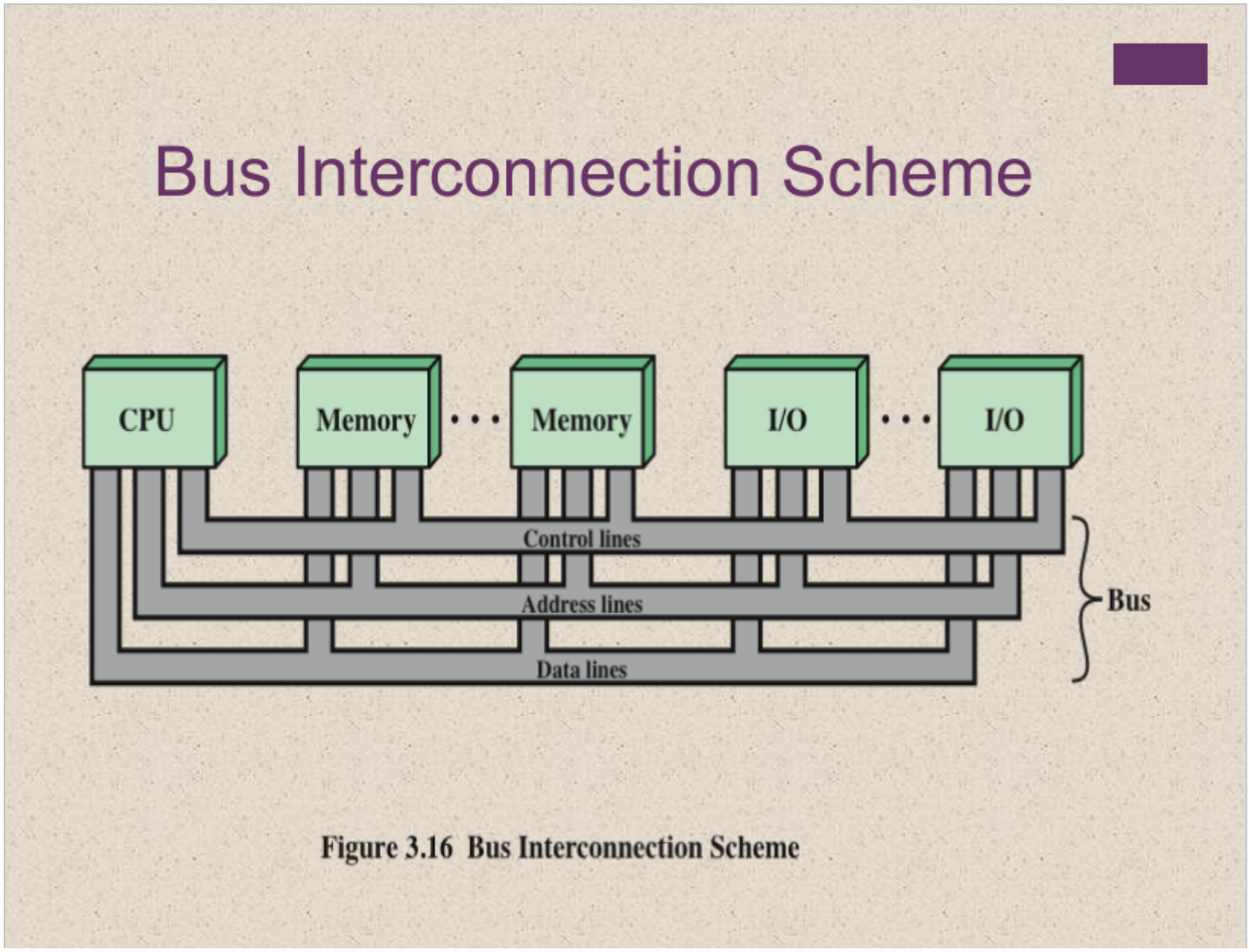


Figure 3.16 Bus Interconnection Scheme

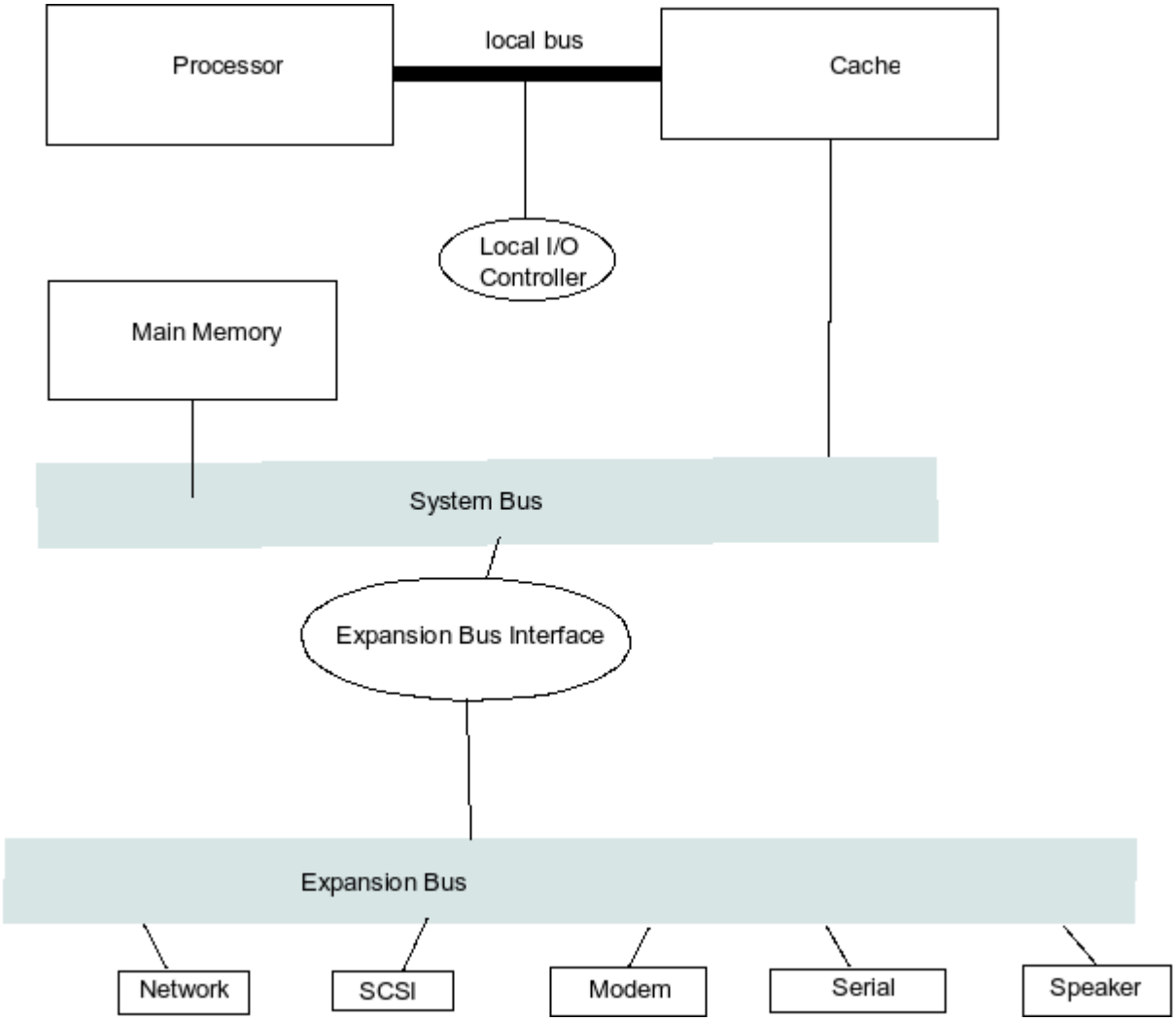
5.11.12 Bus Limitations:

121 The more devices on a bus, the longer the bus and the longer the propagation delay

122 The longer the propagation delay the hard it is for devices to switch and take control of the bus.

123 Aggregate Data Capacity of the bus are limited.

1231 This is affected by word widths and clock speeds.



1232 Memory Cache: Connected to the Processor system bus and the system bus.

12321 In modern systems, cache is attached to the processor.

5.12 Point to Point interconnection structures with Packet technology

5.13 Bus Design:

5.131 Dedicated

.1311 Assigned to one function or one subset of computer modules

.1312 Time Multiplexing - A bus system where the bus is controlled by an initial send of an address which is read by all the devices on the bus. The device that is assigned that address recognizes the communication and then activates on the bus, sending reads and writes over the shared bus until finished.

.1313 Address Valid Line - what the hell is that???

.1314 Physical Dedication: A bus which connects to a single device or subset of modules for the same purpose. An example is a SCSI bus used for SCIS i/o devices

.1315 Method of Arbitration: Devices and modules compete for bus access. They must arbitrate for bus time and resources:

.13151 Centralized Arbitration: Central Bus Controller or Processor controls bus access.

.13152 Distributed Arbitration: No controller and all the devices on the bus use logic to control the bus action.

.13153 Both Methods appoint a master device which then selects a slave for communication and data transfers.

5.132 Device Timing:

5.1321 Synchronous - All the devices on the bus obey a bus clock to send and receive signals. All devices are restricted to the same clock and must be of the same timing

5.1322 Asynchronous - Uses a handshake and determines clock speeds from transitions on the signal (autobaud) and all the devices need not share the same clock features, thus allowing new and old technology share the bus.

http://www.eecs.umich.edu/eecs/courses/eecs373/Lectures/stever_old_lectures/lec10.pdf

5.132 Multiplex

5.2 Point to Point Interconnect: With increasing capacity and multiple core processor, the latency problems built into shared bases have become increasingly problematic. The solution for this is increasingly a movement towards Point to Point Interconnection

5.21 Quick Path Interconnect (2008): Point to Point Interconnection

5.211 Multiple Directly Connected devices

5.212 Layered Controlled Protocols used (like TCP/IP)

5.213 Packet data technology used

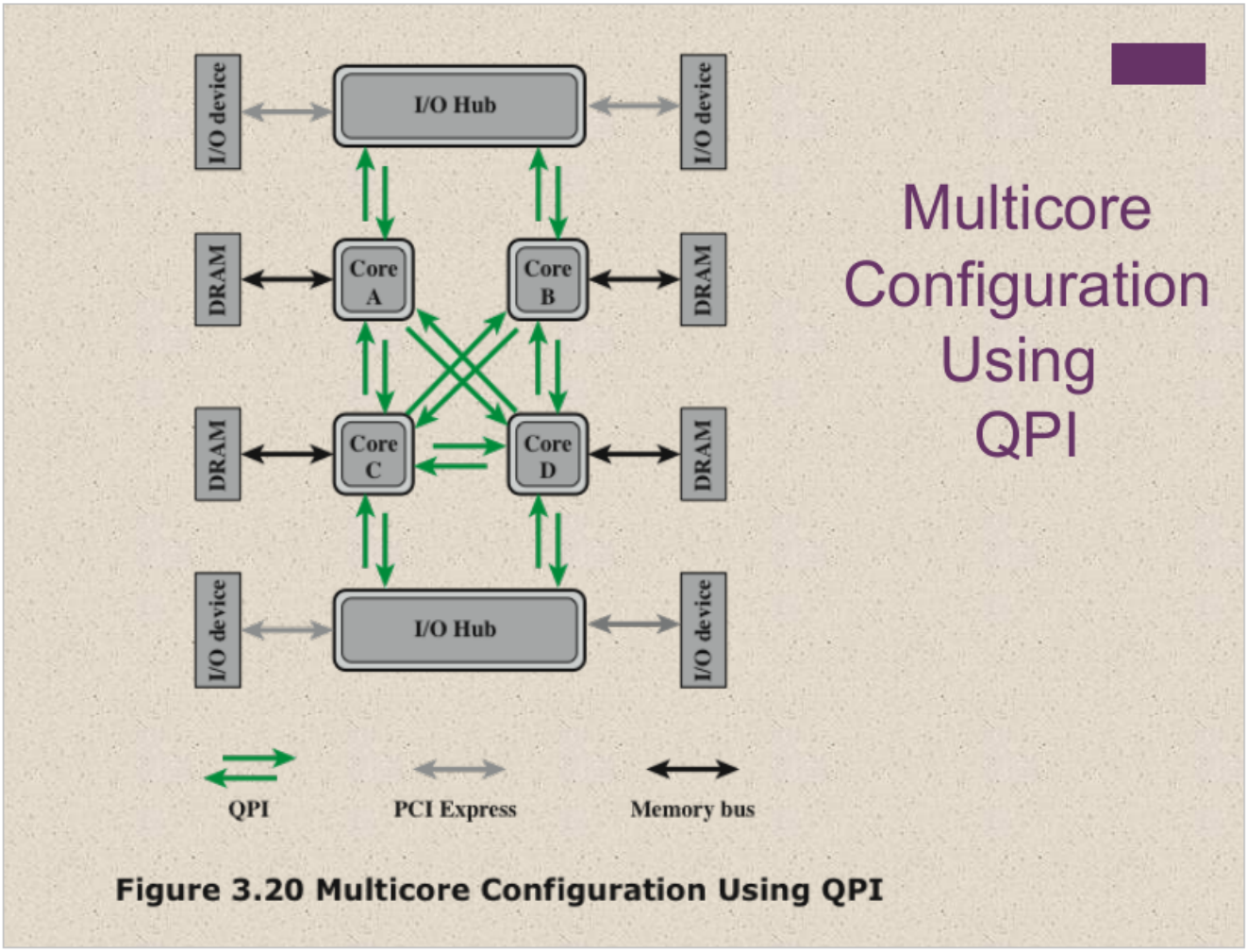


Figure 3.20 Multicore Configuration Using QPI

5.213 Creates a "network" of cpu's and the "I/O Hub"

5.214 The I/O hub used PCI Express technology to talk to i/o devices

5.215: Four Layers Architecture:

2151 Physical: Unit of transfer is 20 bits (phit - Physical Unit)

.2152 Link: Responsible for Flow Control and reliable transmission

21521 Unit of transfer 80 bit (flit)

.2153 Routing:

.2154 Protocol: High Level rules for packages of FLITS to move around the network

5.216 QPI Port: 84 links grouped as pairs of wire that transmit one bit as at a time.

2161 Each pair of wires is called a **lane**

2162 There are 20 data lanes in each direction

2163 One Clock Lane in each direction

2164 Therefore each QPI Port can transmit one phit (20 bits) in each direction at a time

2165 Lanes are grouped in 4 to allow for lower power consumption if desired

2166 **Differential Signaling or Balanced transmission** - Signal is sent in one wire and out the other side on the other wire. The difference between the voltage on each side determines the binary information.

2167 Multilane Distribution: Physical Layer translation of 80bit flit as 20bit phit by round robin injection of the bits into lanes allowing the stream to be processed in a parallel fashion.

2168 Link Layer Flow Control: Flips - 72 bit message payload and 8 bit CRC for error control and flow control between all processors and the I/O hubs.

2169 Routing Layer: Defined by firmware and limited by the small number of cpu and i/o hub devices on a system

216.10 Protocol Layer: Handles a procedure used to assure consistency between cache and main memory

5.3 PCI: Peripheral Component Interconnect: A high speed bus used for peripherals.

5.4 PCIe Peripheral Component Interconnect Express: A Point to Point, high speed peripheral bus designed to handle high speed data i/o devices such as Gigabit Ethernet and time dependent applications like video and audio.

PCIe Configuration

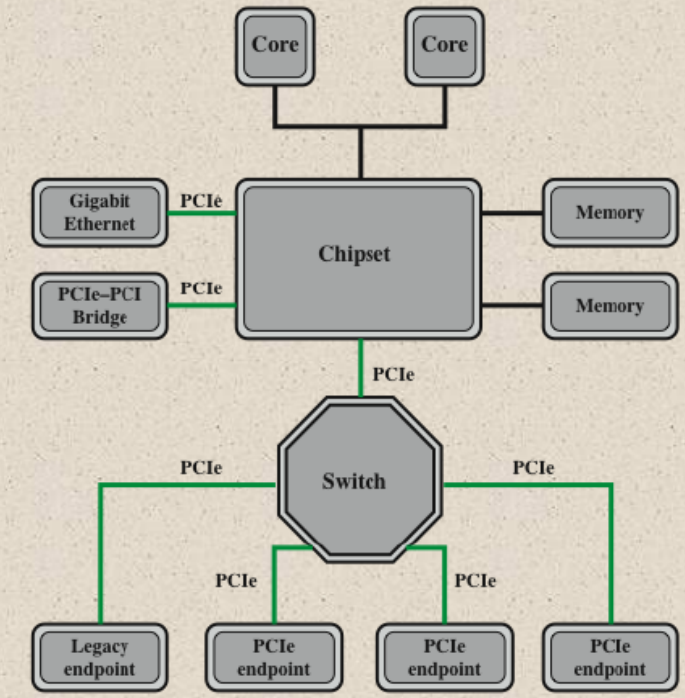


Figure 3.24 Typical Configuration Using PCIe

5.41 Chipset: or host bridge connects the PCIe to to the processors and memory.

5.411 Translates between PCIe protocols and Processor and Memory control and data protocols.

5.412 Connects to PCIe switches and devices. PCIe are considered ports on the Chipset

5.42 PCIe Layers:

5.421 **Physical**

5.422 **Datalink** - responsible for transmision and flow control. Creates packets called DataLink Layer Packets (DLLP)

5.423 **Transaction Layer:** Generates and Consumes data packets between components on the PCIe switch, or through the chipset. Responsible for processing read and write requests by PCIe I/O devices using the Transaction Layer Packets (TLP)

.4231 **Split Transaction Technique:**

.42311 Request Packet Sent

.42312 Competition Packet Awaited to be received

.42313 **Between the Request and Complete the Bus can be used for other activities, rather than both devices holding the lines for exclusive**

communication.

42314 The Device sending the Completion Packet has organized data and is set to complete the asked for task prior to sending the competition packet.

.4232 32 bit and extended 64 bit memory addressing in TLP

.4233 No-snop, Priority and relaxed ordering is available for TLP

.4234 PCIe has 4 address spaces:

41 Memory: System Main Memory and addresses that map into I/O devices

42 I/O: For Legacy PCI and I/O devices which have reserved memory ranges

43 Configuration: Configuration Registers associated with I/O Devices

44 Message: Control Signals for all the usual suspects: Error Handling, Flow Control, Power Management

.4235 The PCIe can lock transactions for Read and Write atomic processes

PCIe TLP Transaction Types

Address Space	TLP Type	Purpose
Memory	Memory Read Request	Transfer data to or from a location in the system memory map.
	Memory Read Lock Request	
	Memory Write Request	
I/O	I/O Read Request	Transfer data to or from a location in the system memory map for legacy devices.
	I/O Write Request	
Configuration	Config Type 0 Read Request	Transfer data to or from a location in the configuration space of a PCIe device.
	Config Type 0 Write Request	
	Config Type 1 Read Request	
	Config Type 1 Write Request	
Message	Message Request	Provides in-band messaging and event reporting.
	Message Request with Data	
Memory, I/O, Configuration	Completion	Returned for certain requests.
	Completion with Data	
	Completion Locked	
	Completion Locked with Data	

.4236 Type 1 and 0 configuration cycles are available for backward PCI devices:

5.424 PCIe Multilane Distribtuion?

.4241 PCIe ports can have 1,4,6,16 or 32 lanes.

.4242 Each lane can handle 16 Bytes at a time

.4242 Each 16 bytes (128 bits) is encoded into 130 bits (128b/130b encoding) which reduces efficient by 128/130

.4243 The extra bits are used to synchronize the sending and receiving devices.

.4244 Scramble: In order to help synchronize, signal is scrambled over lanes to produce a more uniform looking data stream that can reduce the likelihood of false lows and highs, which are used for synchronization signals.

.4245 On receiving the data needs to be reduced back to 128 bits and descrambled:

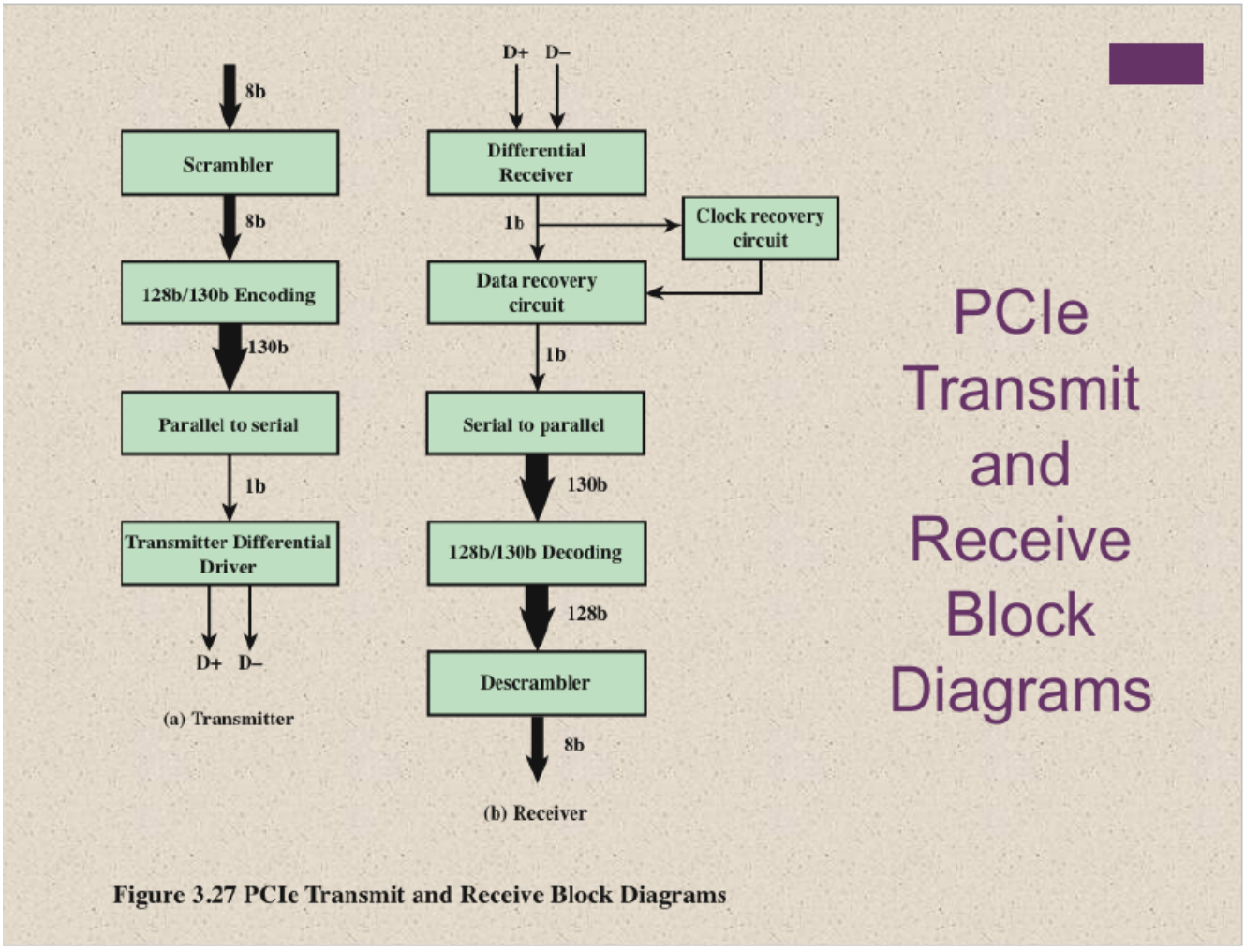


Figure 3.27 PCIe Transmit and Receive Block Diagrams

5.425 TL Packet Structure:

5.4251 Header

5.5452 Data - Upto 4096 bits

5.5453 ECRC - Optional End to End CRC control

5.5454 Length of data in DW (4 bits)

5.5455 Attributes two bits (relaxed ordering, nosnoop)

5.5456 EP - Poison Bit

5.5457 TE - Say CRC is included

5.5458 Traffic Class - Prioritization

5.5459 Type/Format -

5.545.10 First DW Bytes Enabled

5.545.11 Last DW Bytes Enabled