



DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

COMPUTER ARCHITECTURES

PCI, PCI Express and USB interfaces

Prepared by: **Gábor Horváth**, ghorvath@hit.bme.hu

Presented by: **Gábor Lencse**, lencse@hit.bme.hu

Budapest,
2023. 03. 21.



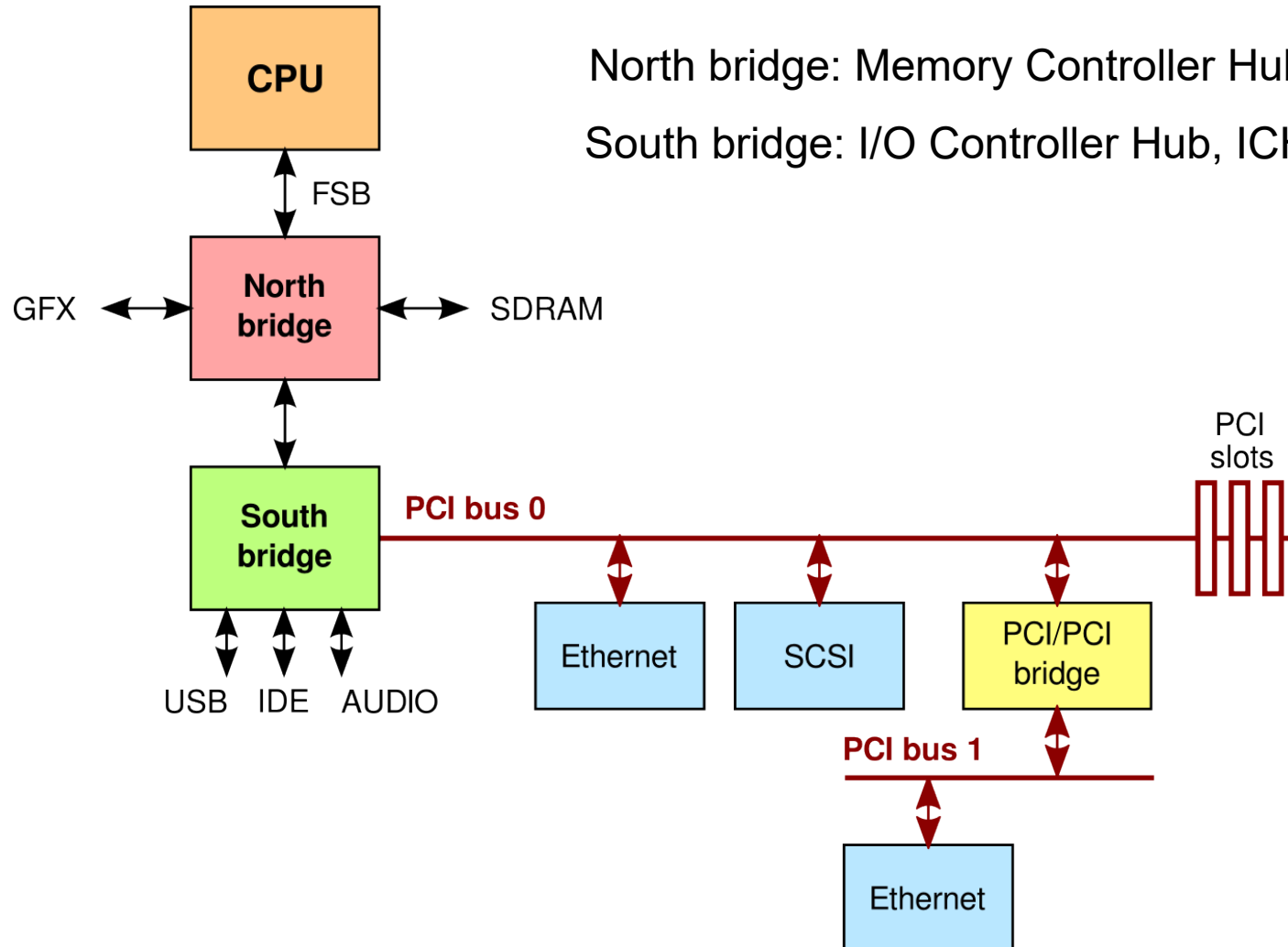
- 90's: chaos around PC peripheral interfaces
 - Thousands of vendor-specific interfaces/connectors
 - An expansion card was needed in most of the cases
 - The number of expansion slots is limited
 - Many interfaces → many different cables/connectors, etc.



- Solution:
 - PCI (= Peripheral Component Interface, 1992, Intel)
 - USB (= Universal Serial Bus, 1994, Compaq, DEC, Intel, Microsoft)
- Common features:
 - Architecture (CPU) independent
 - Support device auto-configuration
 - Hot-plug

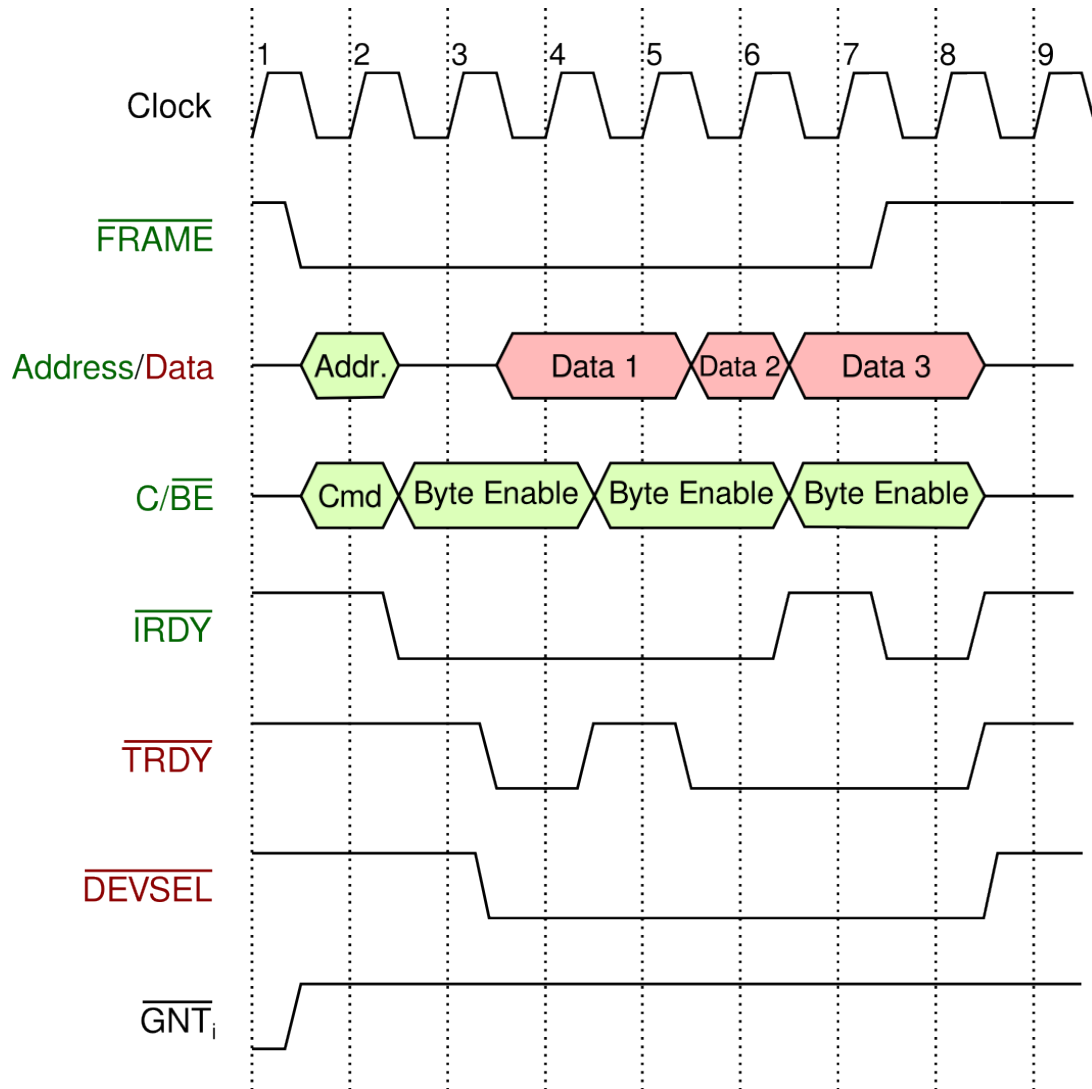


PCI



- Main components
 - **Host/PCI bridge**
 - I/O and memory requests of the CPU ↔ PCI transactions
 - **PCI devices**
 - Max. 32 devices/bus (max. 10 is more realistic)
 - 1-8 functions/device (logical I/O devices on the same physical device)
 - **PCI/PCI bridge**
 - Connects another PCI bus to the PCI bus
- Number of I/O devices in theory:
 - $8 \text{ functions/device} * 32 \text{ devices/bus} * 256 \text{ buses/system} = 65536 \text{ logical devices/system}$

- Each PCI device has max. 6 **windows**
 - A window can be
 - An address range in the memory address space
 - An address range in the I/O address space
 - The CPU performs a read/write to a window of a PCI device
→ it reads from/writes to the PCI device
- Allocation of windows:
 - Devices can tell how many and how large windows they need
 - Windows are assigned to devices by BIOS and the op. system
 - ... and tell the devices where their windows are
- Transaction Models:
 - Programmed I/O. Initiated by: CPU, target: PCI device
 - DMA. Initiated by: PCI device, target: system memory
 - Peer-to-peer transfer. Initiated by: PCI device, target: PCI device



Example: reading from a PCI device (initiator: host/PCI bridge)

Flow control:

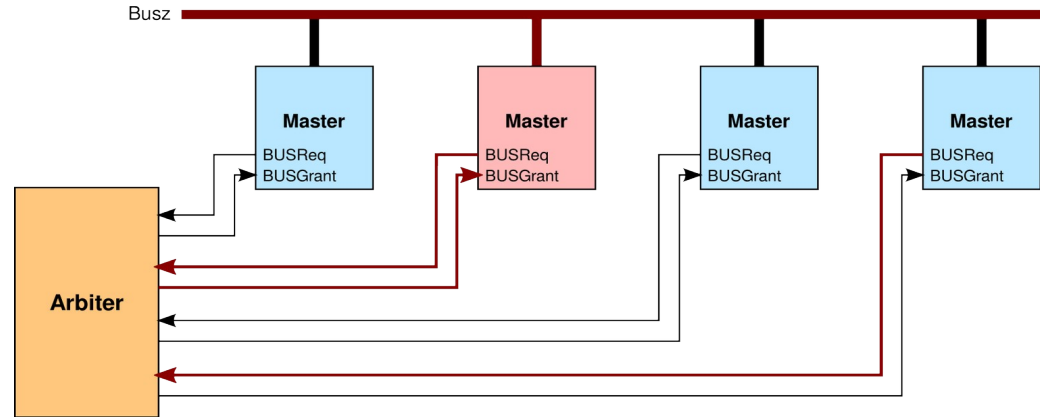
- $\overline{\text{IRDY}}$: initiator ready to transmit next data
- $\overline{\text{TRDY}}$: the target is ready to transmit next data

Phases:

- Addressing + command
- Target claims the transaction
- Frame transmission, with flow control for every data unit
- End of frame → end of transaction

- **Arbitration**

- Parallel arbitration:

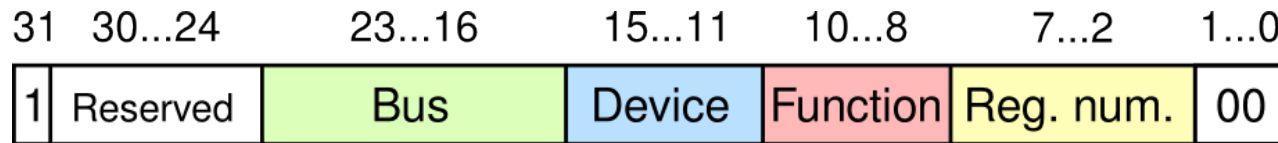


- Hidden arbitration:
 - The contention for the bus and the selection of the next winner occurs during the **current** transaction
- Fair: Can take the delay sensitivity of some devices into account

- **Interrupts**

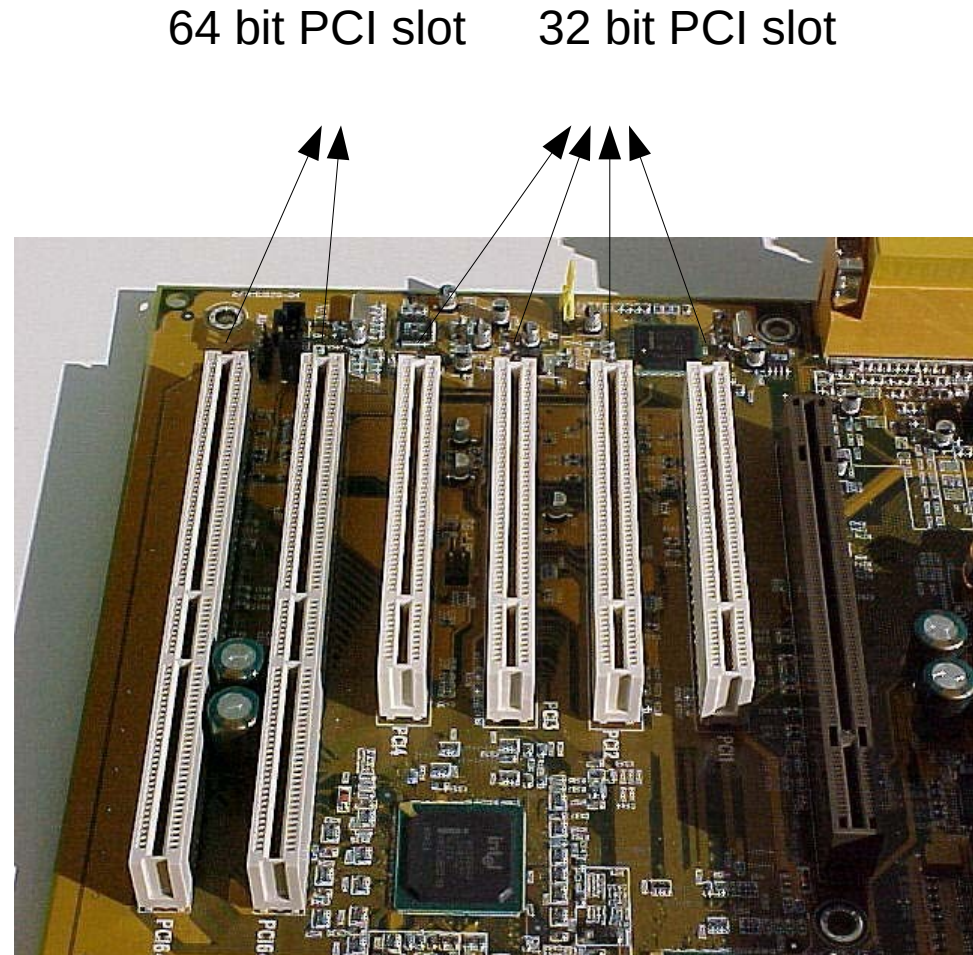
- Two ways:
 - By using one of the four interrupt lines of the PCI bus
 - Several devices can share the same interrupt
- By message signalled interrupts (MSI)
 - The PCI device writes a specific data to a specific address
 - The Host/PCI bridge listens to that address, and generates a CPU interrupt when the specific data is written

- Each device has 64 **configuration registers**, 32-bit each
 - 16 of them are fixed: VendorID, DeviceID, Revision, Base Address Register 0...5
- Accessing configuration registers:
 - Through the PCI host controller
 - Identifying a configuration register:



- Host/PCI bridge selects the device for configuration through a dedicated line
 - Reads/writes the conf. register by a read/write transaction
- During the boot process of the system:
 - The devices are enumerated systematically
 - The windows are allocated to devices
 - Device drivers are loaded
 - The device drivers configure the devices further using configuration transactions
 - ...

- Mandatory:
 - Clock, Reset
 - A/D[0...31]
 - C/ $\overline{\text{BE}}$ [0...3]
 - $\overline{\text{FRAME}}$
 - $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$
 - $\overline{\text{DEVSEL}}$, $\overline{\text{IDSEL}}$
 - Parity, $\overline{\text{PERR}}$, $\overline{\text{SERR}}$
 - $\overline{\text{REQ}}$, $\overline{\text{GNT}}$
- Optional:
 - A/D[32...63]
 - C/ $\overline{\text{BE}}$ [4...7]
 - $\overline{\text{INTA}}$, $\overline{\text{INTB}}$, $\overline{\text{INTC}}$, $\overline{\text{INTD}}$
 - $\overline{\text{REQ64}}$, $\overline{\text{ACK64}}$
 - $\overline{\text{CLKRUN}}$

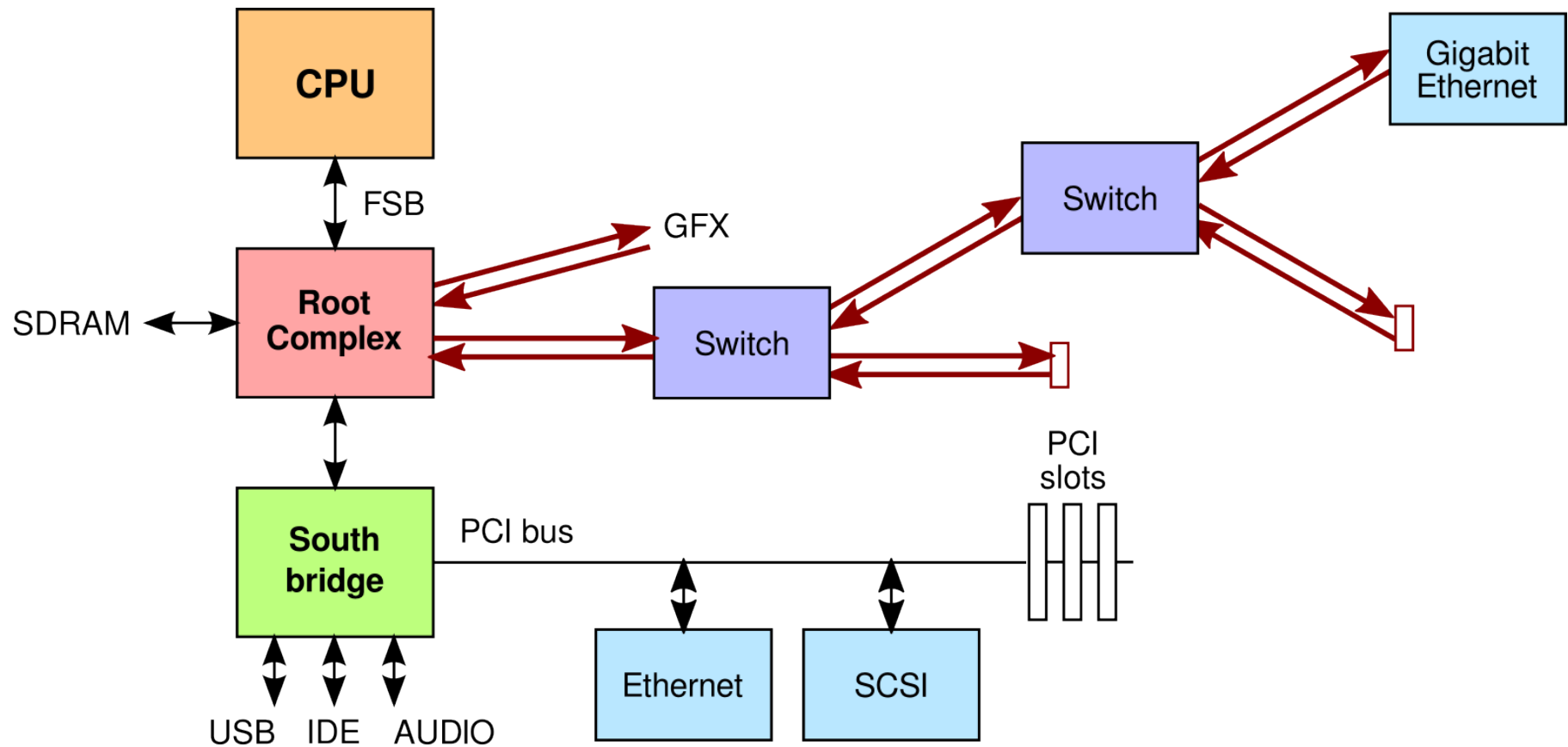




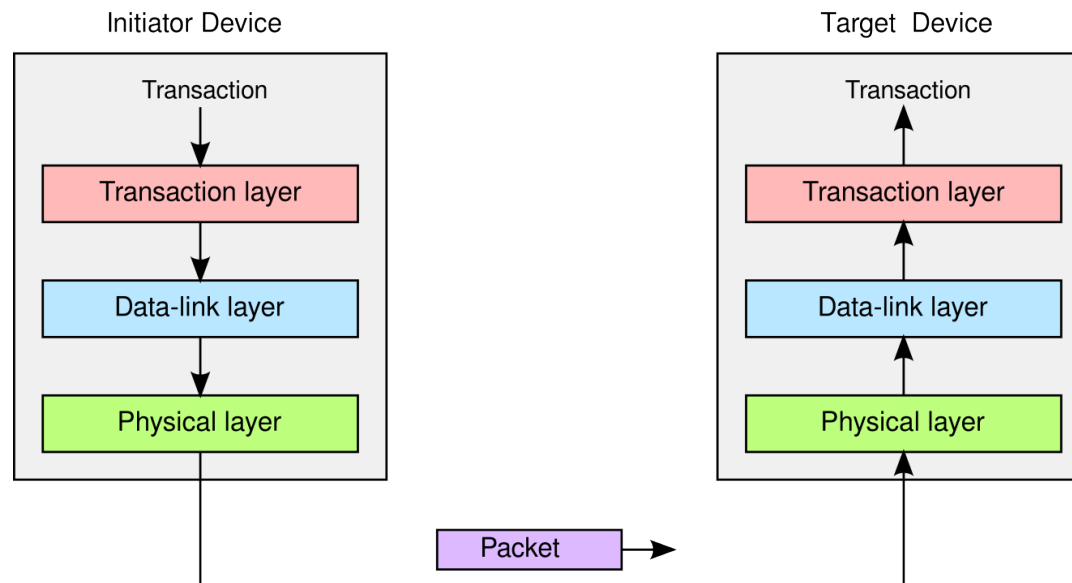
PCI Express

- Goals:
 - To achieve higher transmission speed than PCI
 - **To keep full software compatibility with PCI**
- Most significant changes:
 - **Point-to-point connections**
 - No shared medium → no contention, no waiting, no starvation, no arbitration
 - **Serial data transmission**
 - No clock signal (detected automatically from the 0/1 transitions)
 - A pair of wires: 32 Gbit/s (PCIe 5.0)
 - Full duplex communication:
 - 2 serial pair of wires for both directions
 - Two devices can be connected by more full duplex lines
 - The name of 1 full duplex serial line: **lane**
 - Part of the standard: 1x, 4x, 8x, 16x, 32x lanes
 - More lanes allow parallel transmission of data
 - But not in a synchronous way !!!
 - With 32x lanes we have $32 \times 32 \text{ Gbit/s} = 128 \text{ GB/s}$

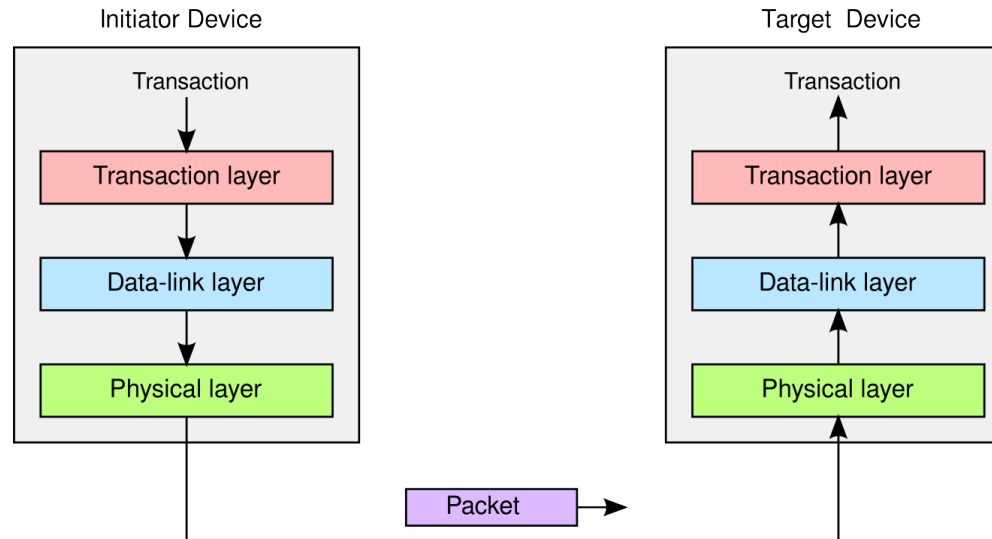
PCI EXPRESS BASED SYSTEMS



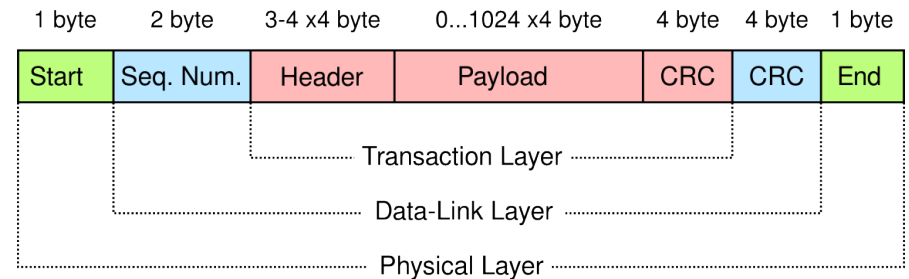
- Works like a packet switched network
 - Devices are connected to the root complex
 - Transactions are traveling from switch to switch as packets



- Transaction layer: creates packets from transactions, adds address info
- Data-link layer: sequence number generation / re-transmission on errors
- Physical layer: marks packet boundaries, 8b/10b (128b/130b) encoding

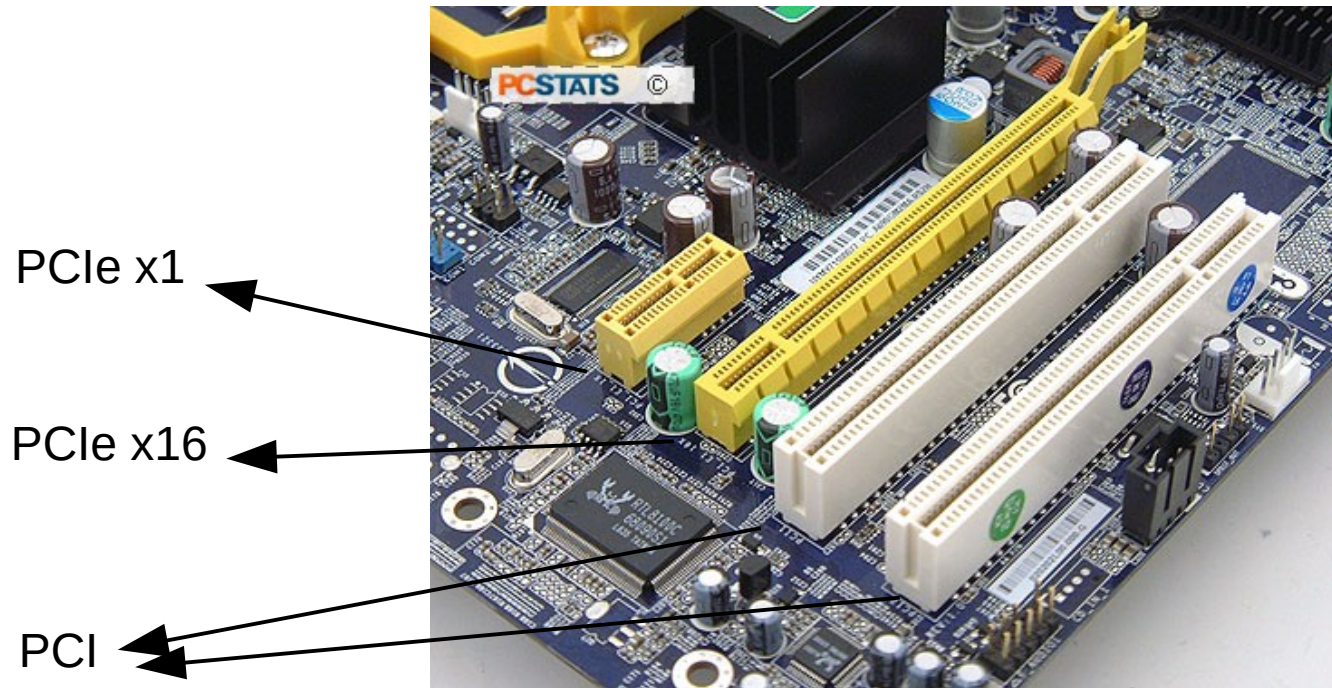


- Packet format:



- Overhead for transmitting 32 bits: $1+2+3*4+1*4+4+4+1 = 28$ byte
+ 8b/10b overhead, = 35 byte !!!

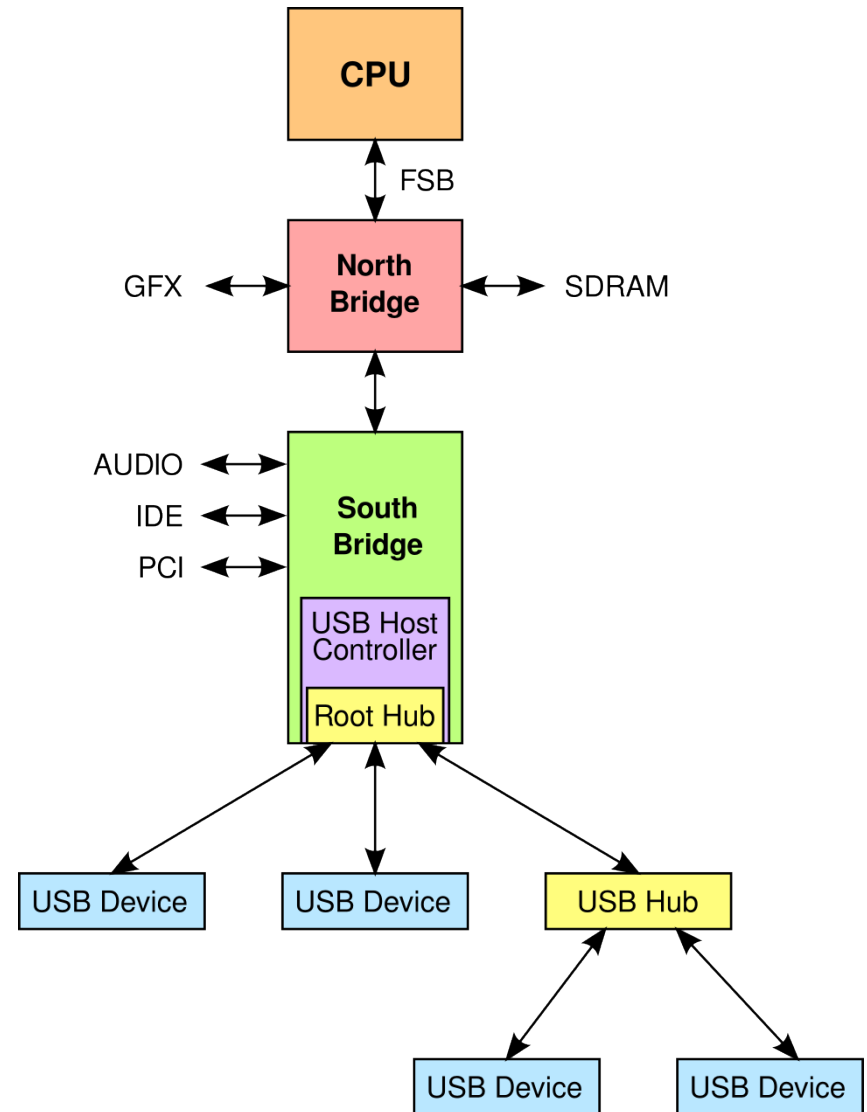
- Interrupts
 - Message based (MSI) → like in the PCI
 - Interrupt line based interrupts: deprecated
- Configuration
 - More configuration registers (1024)
 - First 64 of them: like in PCI (full compatibility)





USB

- Tree topology
 - Leafs: **USB devices**
 - Nodes: **USB hubs**
 - Root: **root hub**
(part of the host controller)
- An I/O device can be:
 - A simple device
 - A compound device
 - One device physically
 - More devices logically

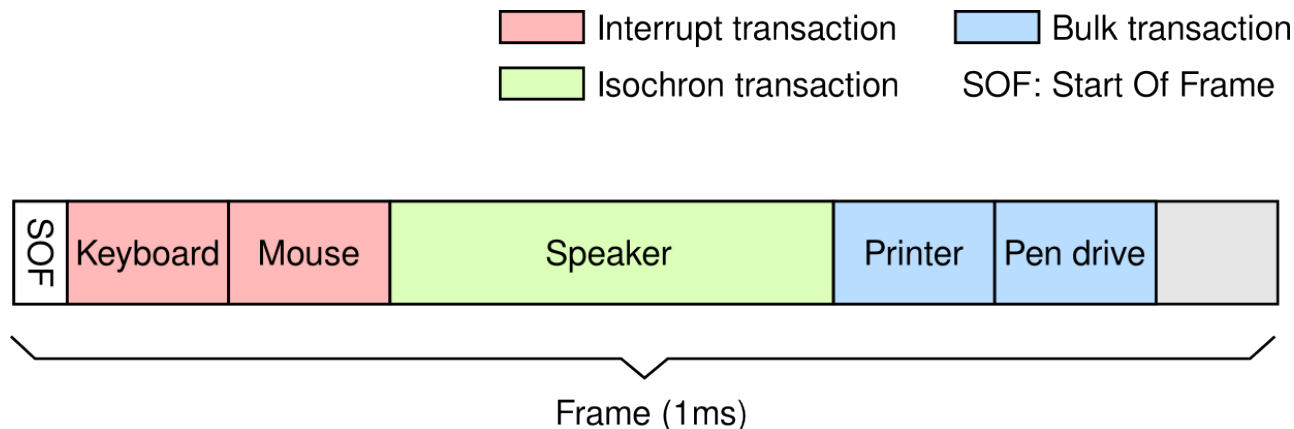


- We already know that a bus is a shared medium
 - Everybody hears what others talk
- But the **USB is a tree!**
- **It is still a bus**
 - It has a single master: the root hub
 - Everybody hears what the master says (it is broadcasted in the tree)
 - The target device listens, the other do not listen
 - Messages sent by the target device reach the master in a bottom-up manner
- Arbitration is not needed
 - Since we have only one master
 - Nobody can talk unless the master gives permission
- If it is a bus, it has its drawbacks as well:
 - The more devices are connected, to slower it is

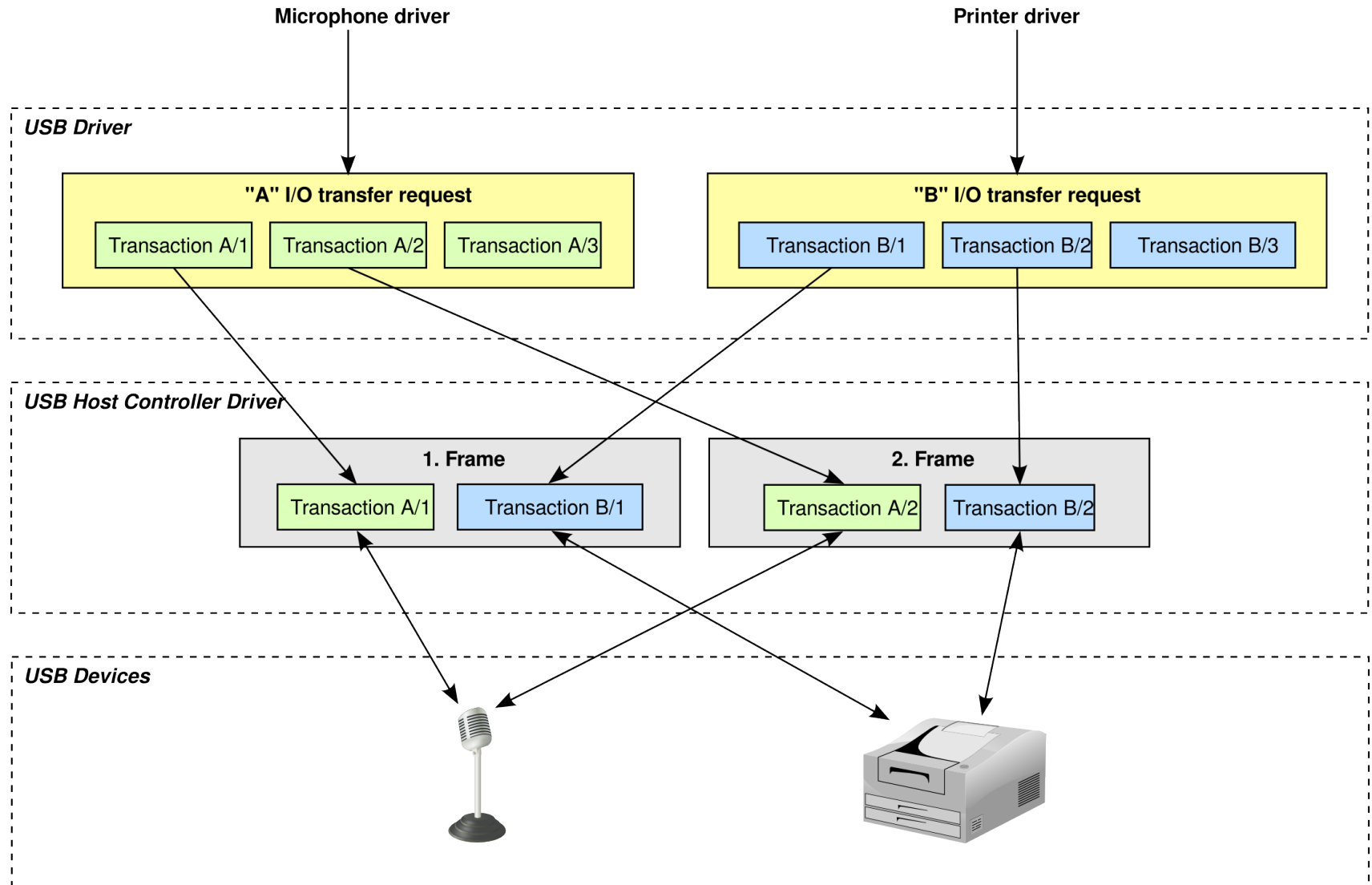
- The communication is organized in **transactions**
- All transactions are initiated by the master (root hub)
- Contents:
 - ID of the target device
 - Direction (input or output transaction)
 - The data to transmit
 - Output transactions: the data is driven by the root hub
 - Input transactions: the data is driven by the target device
- The transactions are carried by **frames**
 - Each frame takes a given amount of time
 - USB 1.1 full speed: 1 ms/frame, 1500 byte payload
 - USB 2.0 high speed: 125 μ s/frame, 7500 byte payload
 - Each frame can carry several transactions
 - Frames can carry only entire transactions (may not be fully filled)

- **Transfer mode:** how to transmit data
 - **Bulk**
 - To transmit large amount of data, in an error-free way
 - No delay guarantee
 - Examples: external hard disk, printer, scanner, pendrive, etc.
 - **Isochronous**
 - Real-time transfer for delay sensitive devices
 - Does not ensure error-free transmission
 - Examples: USB microphone, webcam, etc.
 - **Interrupt**
 - There is no interrupt in USB → it is simulated by polling!
 - Interrupts are transmitted with delay guarantee and in an error-free way
 - Examples: human interface devices (keyboard, mouse, etc.)
 - **Control**
 - Error-free transmission with delay guarantee
 - Message format is specified by the USB standard

- Bandwidth management
 - Isochronous and interrupt transactions have priority
 - But can not occupy more than 90% of the frame capacity
 - If a new device is connected, and the 90% limit would be exceeded with it
→ the device is not allowed to join the USB system
 - Control transactions have priority in the remaining 10% of the frame
 - Rest: for bulk transactions



THE DATA TRANSMISSION PROCESS



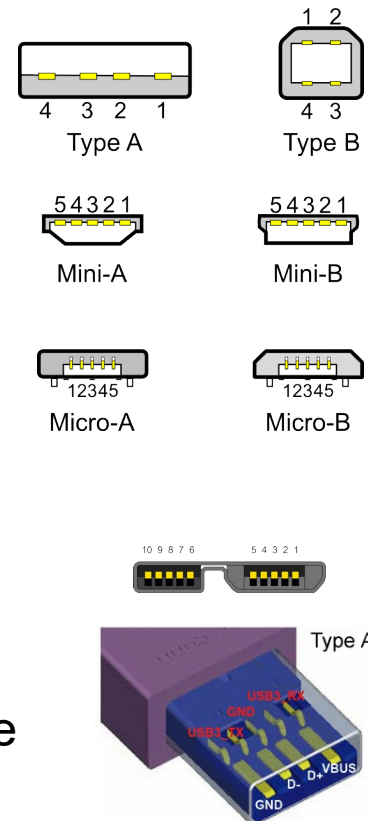
- When the system boots, or when a new device is connected
- The hub detects that a new device is connected
- The USB driver asks all hubs whether a new device is connected from time to time. If yes,
 - It tells the hub to reset that port
After the Reset the device answers transactions sent to address 0
 - Reads the device descriptors of the device
(By control transactions sent to address 0)
 - It contains the vendor ID, device ID, device class, parameters, etc.
 - Assigns a unique address to the device
(By a control transaction sent to address 0)
 - Checks if it can enter the USB system
 - Can its power demand be served
 - Can its isochronous and interrupt bandwidth demand be served
- If yes, it can enter and start answering transactions

- USB 1.1:
 - **LS**: low speed, 1.5 Mb/s
 - **FS**: full speed, 12 Mb/s
- USB 2.0:
 - **HS**: high speed, 480 Mb/s
 - LS, FS remains (by maintaining the compatibility is not trivial)
 - Smaller differences
 - In flow control
 - In frame sizes
- USB 3.0:
 - **SS**: super speed, 5 Gb/s
 - Completely different physical transmission of the transactions!
- USB 3.1:
 - **SS+**: super speed+, 10 Gb/s
- USB 3.2:
 - Even higher throughput with more lanes

- What did not change:
 - Tree topology, with hubs at the nodes
 - The 4 transfer types
 - Configuration is performed the same way
 - Root hub is still the only master
 - No change at software level
- What did change:
 - Almost everything at the physical level. A completely new system.
 - Compatibility: USB 3.0 is composed by two buses next to each other
 - An USB 2.0 bus for LS, FS, and HS transactions
 - A separate bus for the SS transactions
 - **It does not broadcast transactions any more**
 - Transactions are routed
 - Hubs forward the data to a single port only, where the target device is
 - **Store and forward packet transmission**
 - The hub waits till the entire transaction arrives, and forwards it only after that
 - **There are four wires instead of two to transmit data**
 - To allow parallel communication in up- and downward direction
 - Just like in the case of PCI Express (bit scrambling, 8/10 encoding, etc.)

- There are „ground” and „+5V” wires also present in the USB cable
→ USB devices can drain current from it!
- Drawing current:
 - Maximum:
 - USB 1.0 - 2.0: max. 500 mA
 - USB 3.0: max. 900 mA
 - Devices can take no more than 100mA in the configuration phase
 - Devices tell the power demand during the configuration phase
 - If it is more than it is available, the device can not enter the USB system

- Cable: 4 wires
 - Power: 0V, 5V
 - Signal: D+, D–, differential, serial transmission
- USB 1.1 connectors
 - Type A: connects device to hub
 - Type B: connects hub to device
- USB 2.0 connectors
 - New sizes: mini, micro
 - There are sockets supporting both “A” and “B” connectors
 - Connecting „A” they behave as a host
 - Connecting „B” they behave as a device
 - 5th lag: “role”: host if grounded, device if open
- USB 3.0:
 - Cable: includes the twisted pair of USB 2.0
And the two twisted pairs of USB 3.0 → thick and expensive
 - Connector: two parts: one for USB 2.0 and one for USB 3.0



- Type-C cable:

- Fully reversible plugs
- The same plug for the master and the slave
 - They negotiate on the roles through the CC (configuration channel)

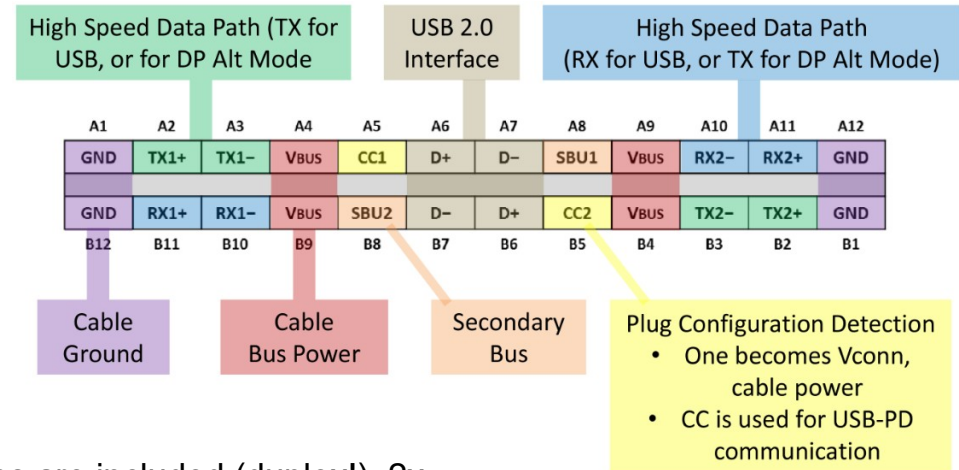
- Cables:

- USB 2.0 wires are included
- USB 3 SuperSpeed/SuperSpeed+ wires are included (duplex!), 2x
- „SideBand” wires for future use
 - Including analog audio
- Alternate (non-USB) mode: the cable can transmit other technologies as well
 - DisplayPort, Thunderbolt, HDMI
- It is for speed up to 40 Gb/s

- The cable is active

- Power delivery

- Current up to 5A, voltage up to 20V (100W!!!!)
- The cable actively participates in the negotiation of the power demand of the devices





DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

