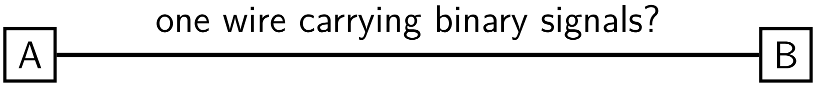
connecting things

how to (in hardware) connect A and B?

$\boxed{A}$ $\qquad\qquad\qquad\qquad\qquad\qquad$ $\boxed{B}$

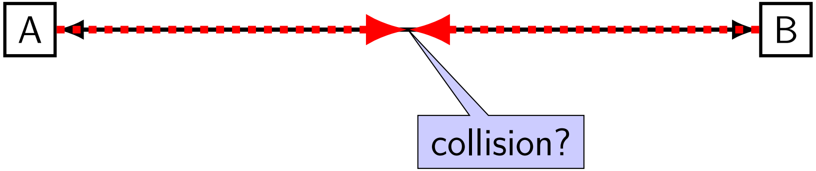
connecting things

how to (in hardware) connect A and B?



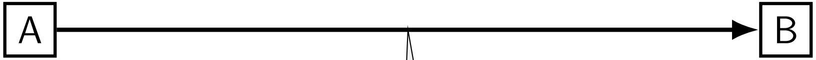
connecting things

how to (in hardware) connect A and B?



connecting things

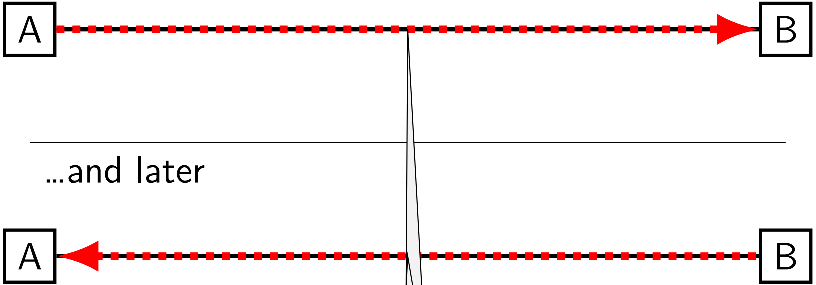
how to (in hardware) connect A and B?



one-way communication only?
called *simplex*

connecting things

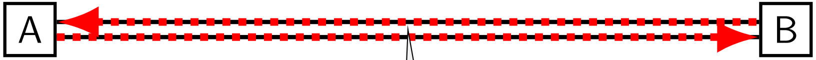
how to (in hardware) connect A and B?



taking turns, but one-way
called *half-duplex*
challenge: how to agree who's turn?

connecting things

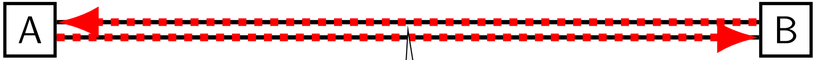
how to (in hardware) connect A and B?



both ways at the same time
called *full duplex* (or *duplex*)

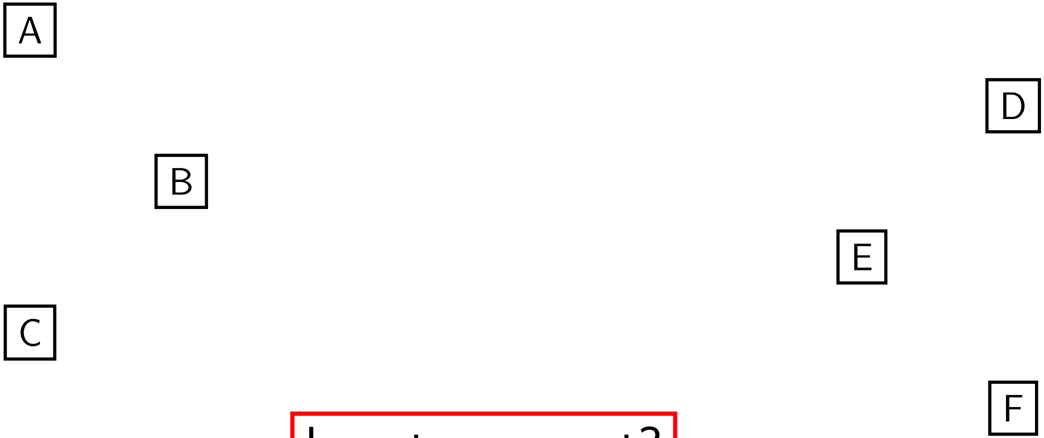
connecting things

how to (in hardware) connect A and B?



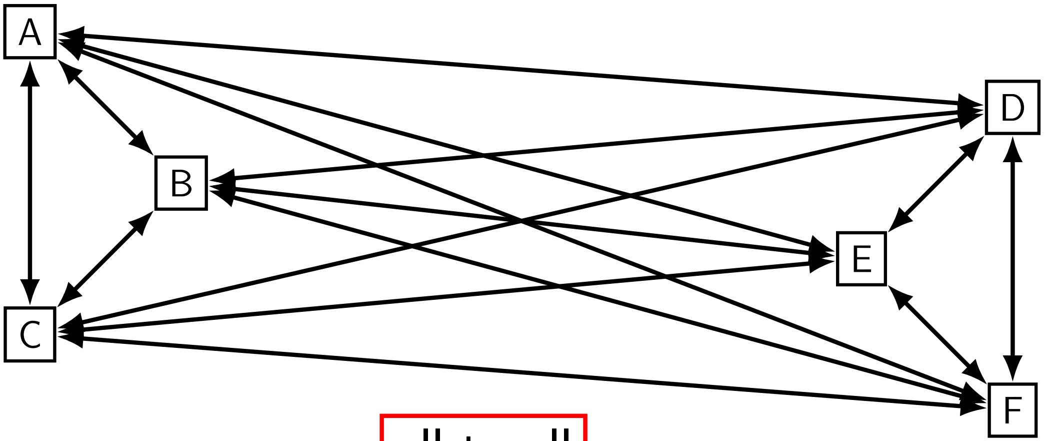
here: duplex via multiple wires (simplest scheme)
can achieve effect electrically/etc. via one wire
example: cable Internet
(how is topic for ECE class)

connecting things



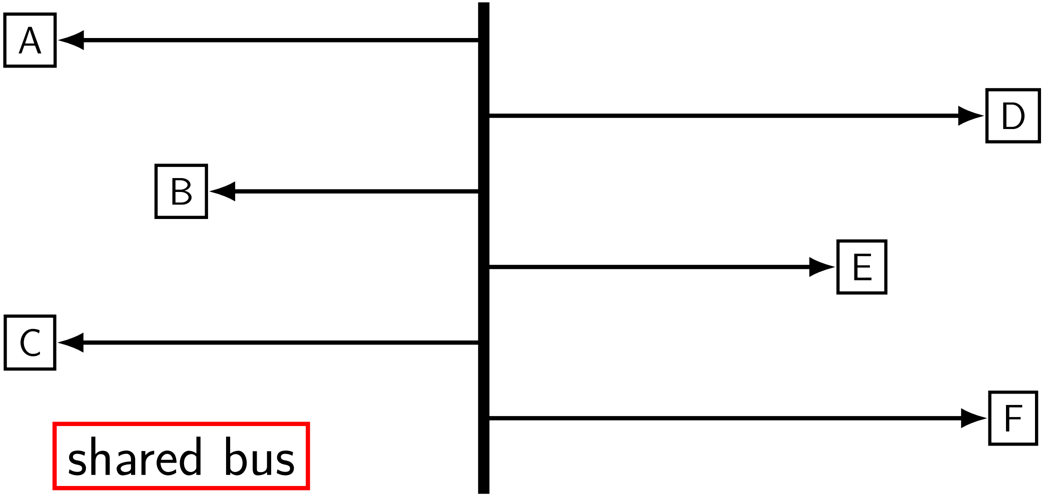
how to connect?

connecting things



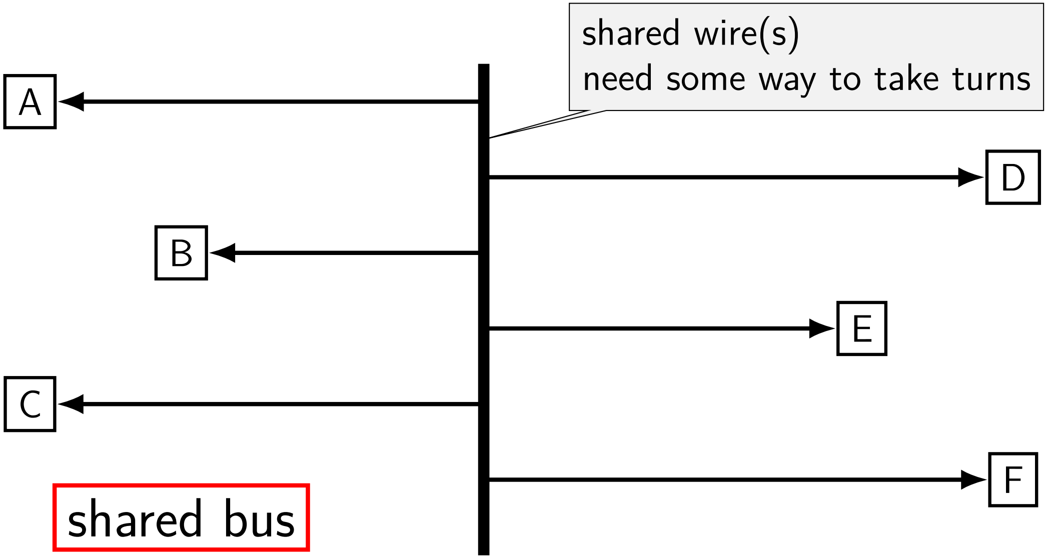
all-to-all

connecting things



shared bus

connecting things



shared bus, really?

common for parts of internals of computers (topic later)

model for wifi

- radio “channel” kinda similar to shared wire

how the early versions of Ethernet worked

- “vampire taps” physically attached to shared cable

shared bus, messages for who?

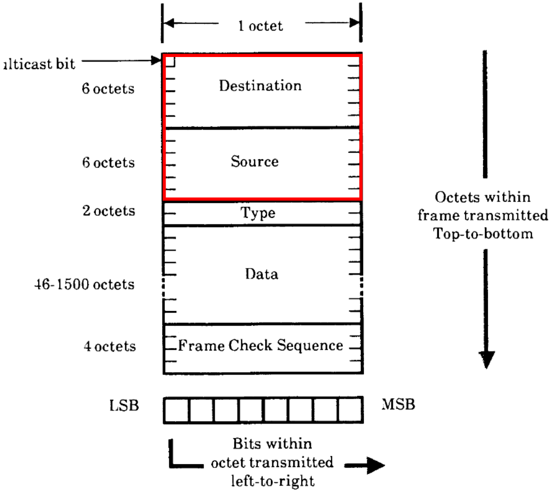


Figure 6-1: Data Link Layer Frame Format

messages needs a ‘header’ to tell who it’s to/from

everyone needs to filter out messages that aren’t theirs

Figure from Digital, Intel, and Xerox, “The Ethernet: A Local Area Network: Data Link Layer and Physical Layer Specification”, Version 2.0 (1982)

taking turns on shared bus?

token ring

- one machine has a 'token' = can send
- send special message to pass to another machine

free-for-all: collision detection + retry

- detect if you're transmitting when someone else is
- wait (usually randomized amount of time) and retry

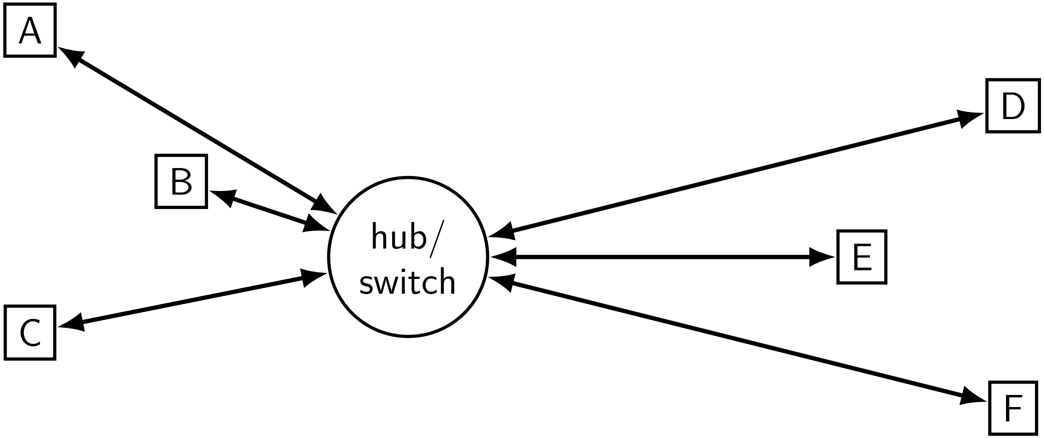
coordinating machine transmits timeslots

- part of common cellphone design (TDMA: time division multiple access)

make bus support multiple transmitters?

- requires understanding how interference works
- another part of common cell phone design

connecting things



what does the hub do?

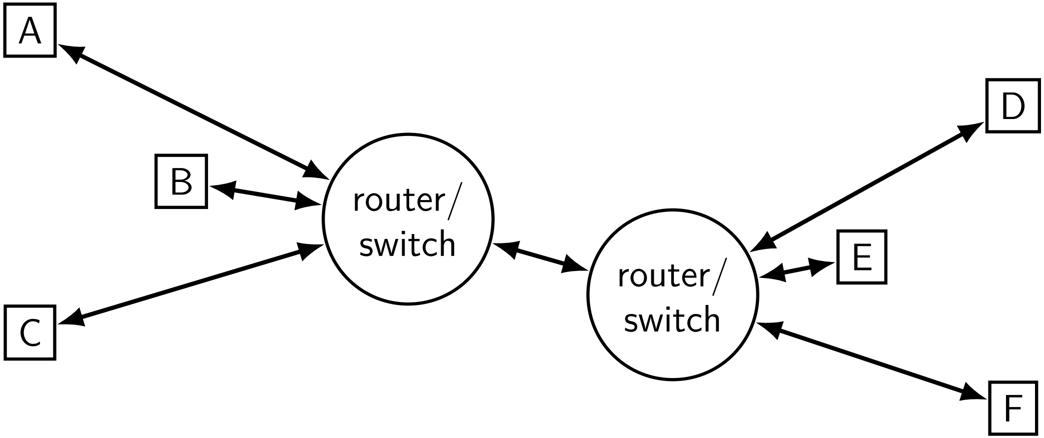
simple version:

- imitate shared bus: copy messages to everyone else
- something to handle two messages sent at once

less simple:

- read "header" on message + send to destination only
- requires some way to figure out destinations
- queue of messages waiting to be sent

connecting things



more complicated designs

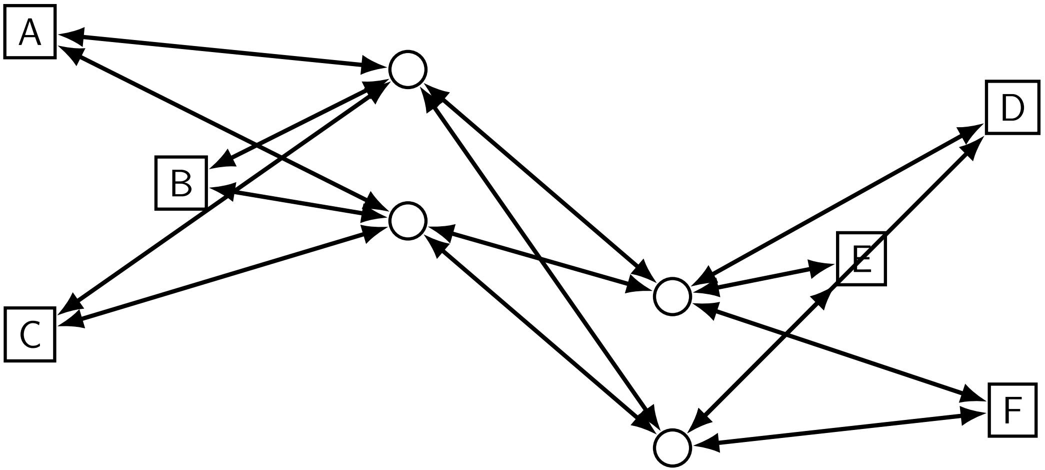
hierarchies

networks of networks

  "internetworks"

so far still have single points of failure

connecting things



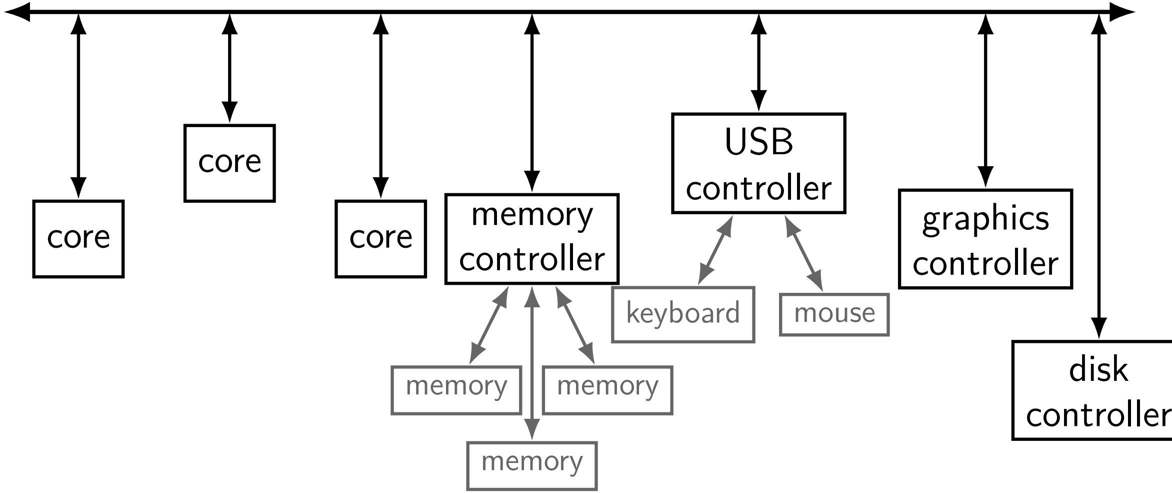
individual computers are networks

individual computers are (kinda) networks of...

- processors
- memories
- I/O devices

so what topology (layout) do those networks have?

the “bus”


core
core
core
memory controller
USB controller
graphics controller
disk controller
memory
memory
memory
keyboard
mouse

example: 80386 signal pins

name	**purpose**	
CLK2	clock for bus	timing
W/R#	write or read?	metadata
D/C#	data or control?	
M/IO#	memory or I/O?	
INTR	interrupt request	
...	other metadata signals	
BE0#-BE3#	(4) byte enable	address
A2-A31	(30) address bits	
DO-D31	(32) data signals	data

adapted from the Intel 80386 Hardware Reference Manual (1986)

example: AMD EPYC (1 socket)

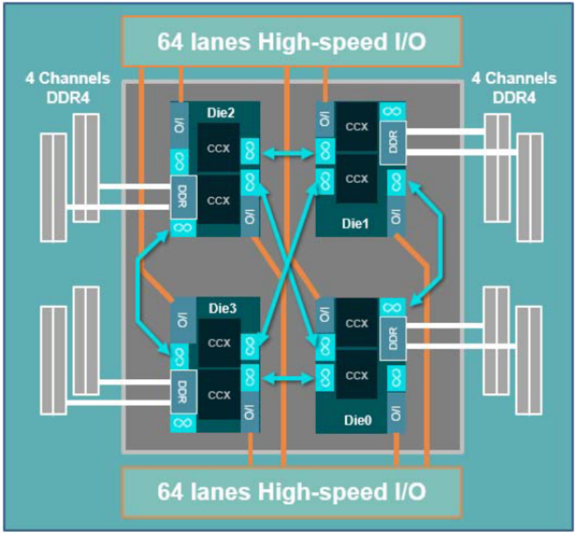


Fig. 21. Single-socket AMD EPYC$^{\text{TM}}$ system (SP3).

Figure from Burd et al,
" 'Zepllin': An SoC for Multichip Architectures" (IEEE JSSC Vol 54, No 1)

example: Intel Skylake-SP

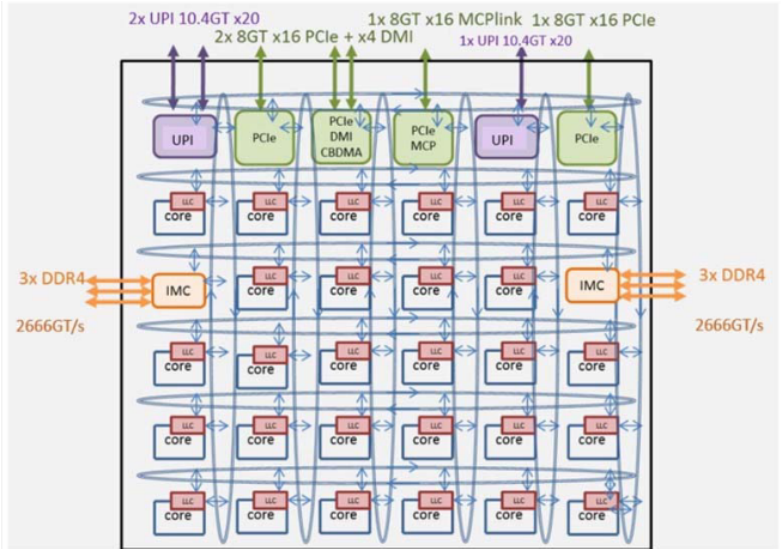
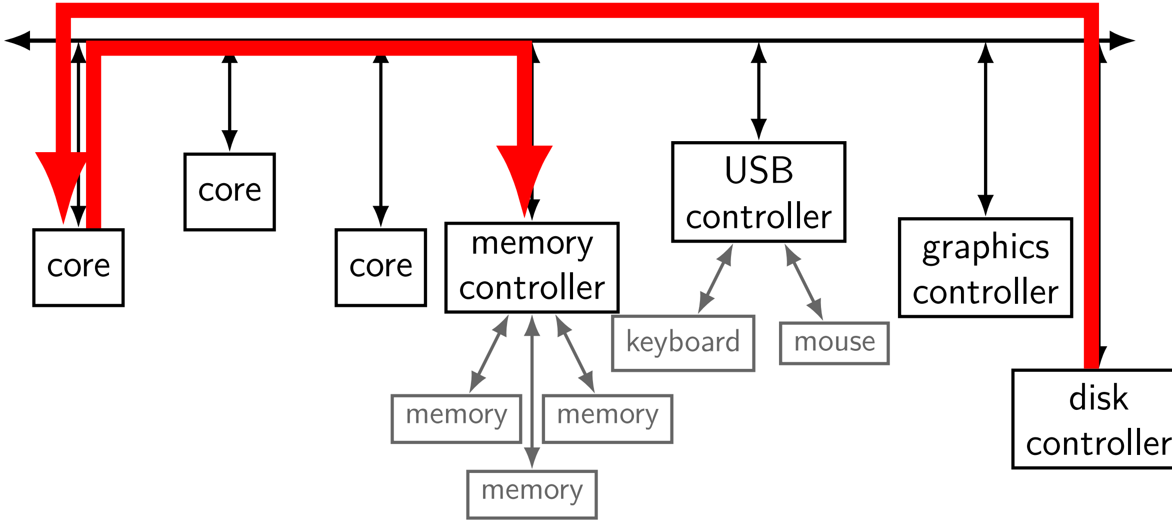
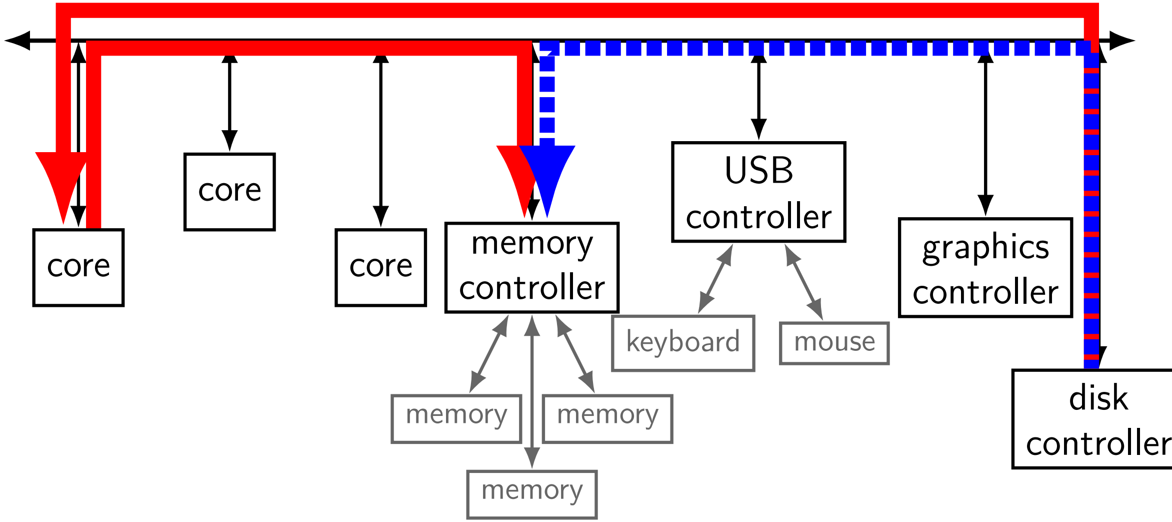


Figure from Tam et al, "SkyLake-SP: A 14nm 28-Core Xeon® Processor" (ISSCC 2018)

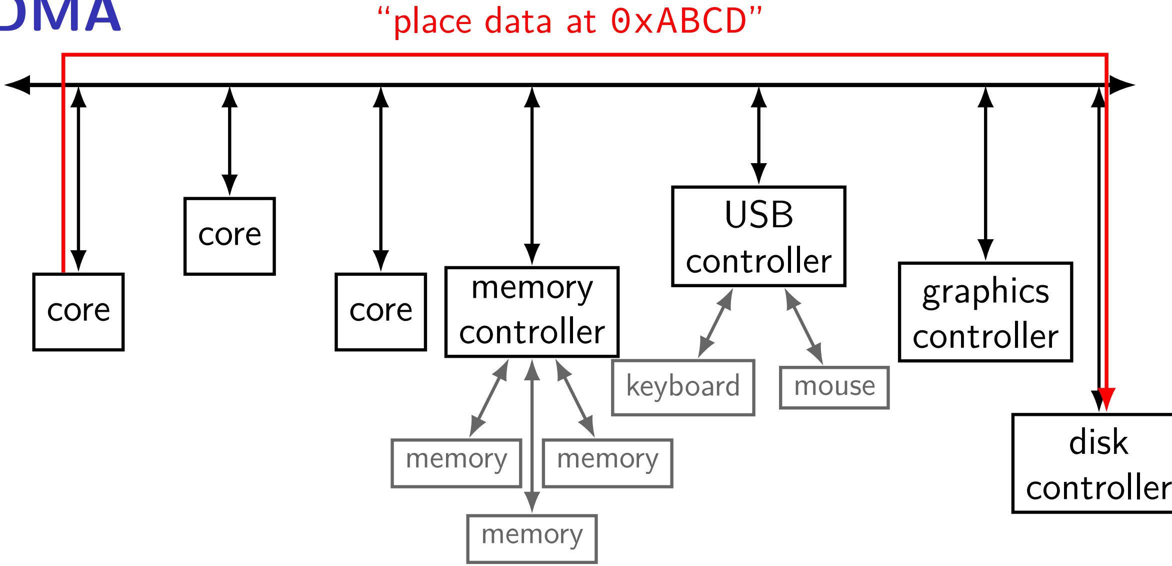
extra trips to CPU



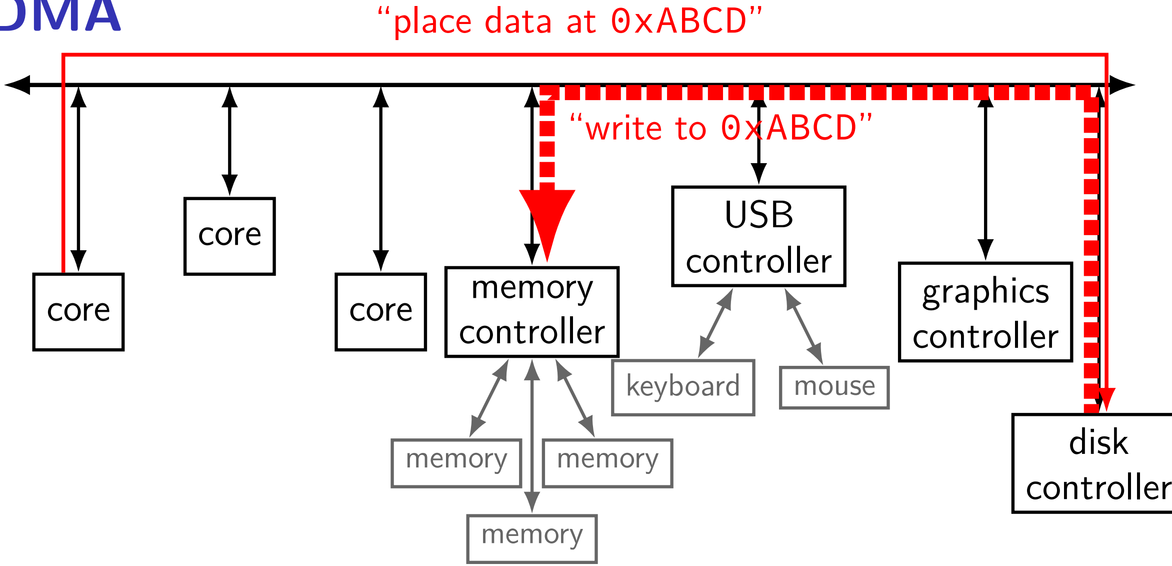
extra trips to CPU



DMA

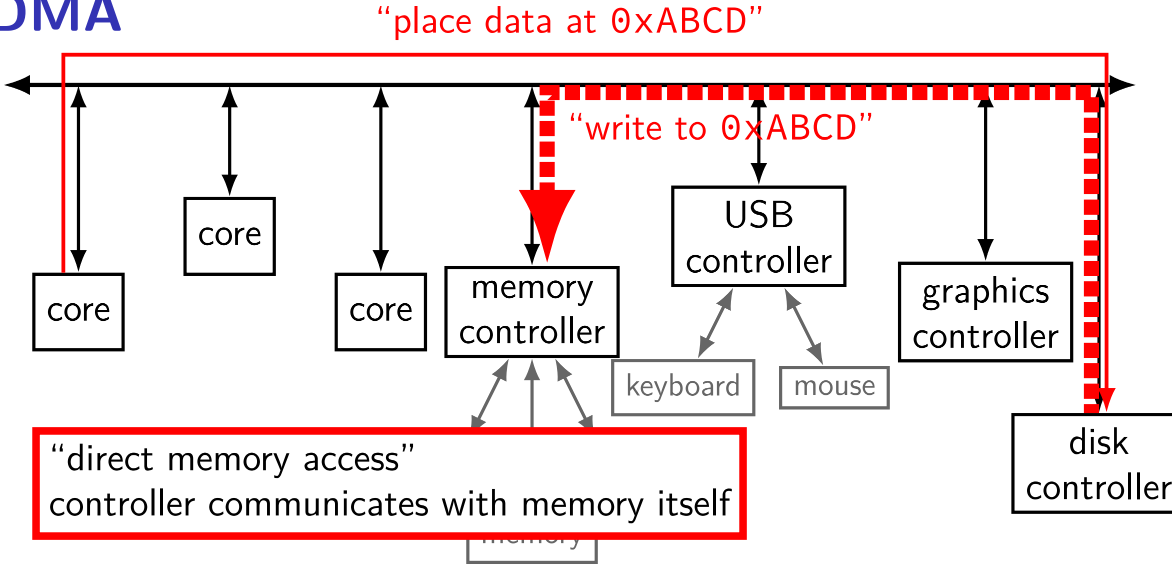

"place data at 0xABCD"
core
core
core
memory controller
USB controller
graphics controller
disk controller
keyboard
mouse
memory
memory
memory

DMA



"place data at 0xABCD"
"write to 0xABCD"
core
core
core
memory controller
USB controller
graphics controller
disk controller
keyboard
mouse
memory
memory
memory

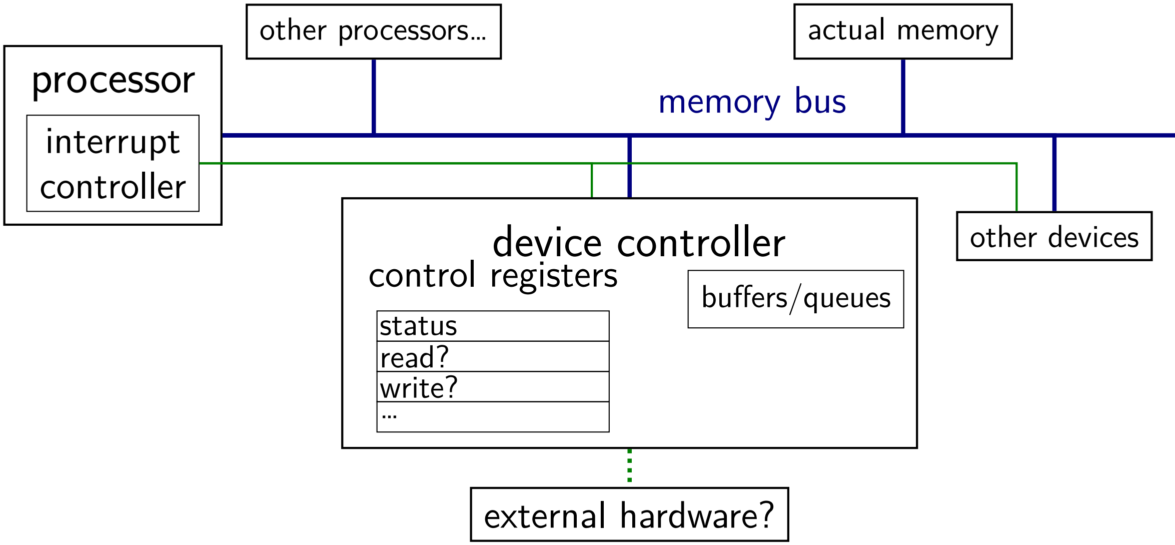
DMA



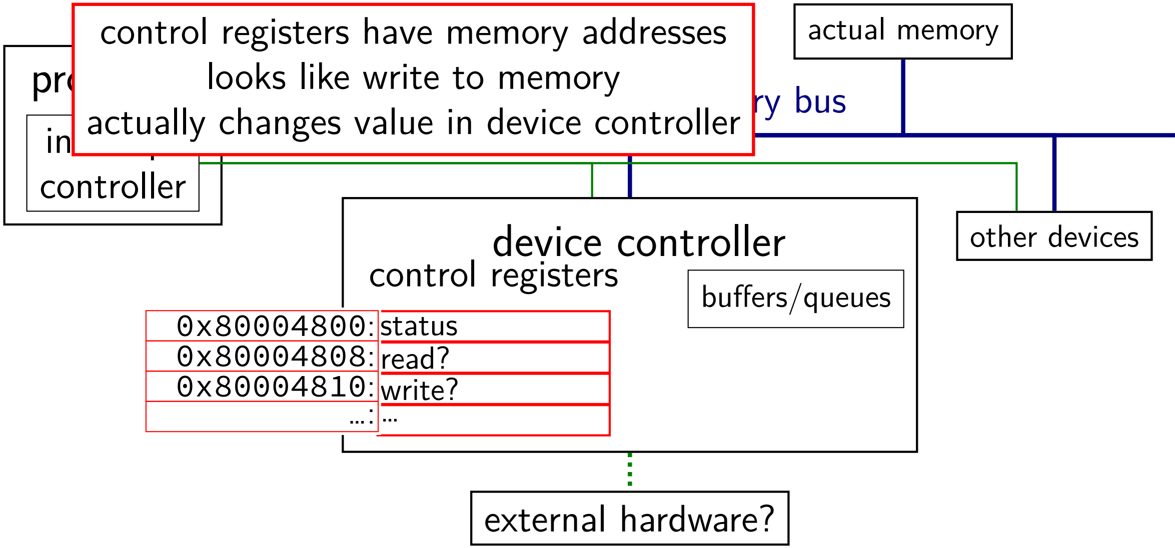
"direct memory access"
controller communicates with memory itself

backup slides

connecting devices

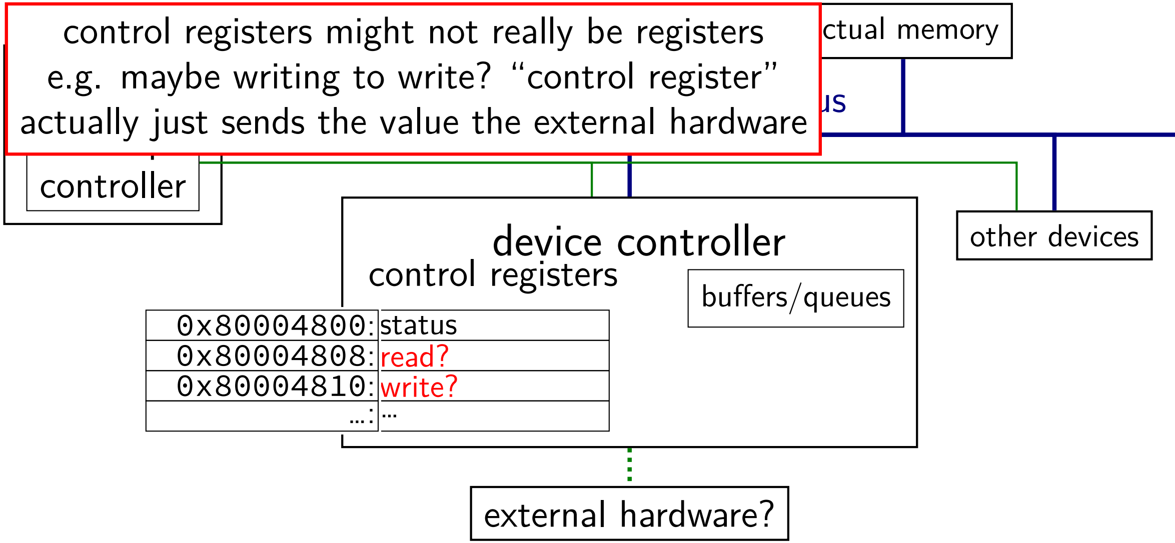


connecting devices

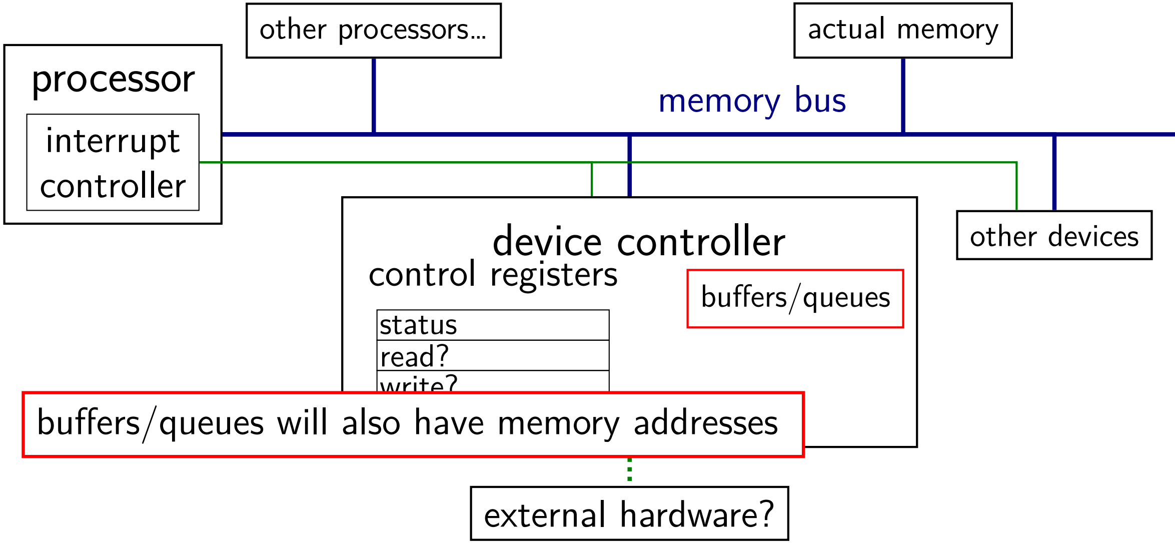


connecting devices

control registers might not really be registers
e.g. maybe writing to write? "control register" actually just sends the value the external hardware

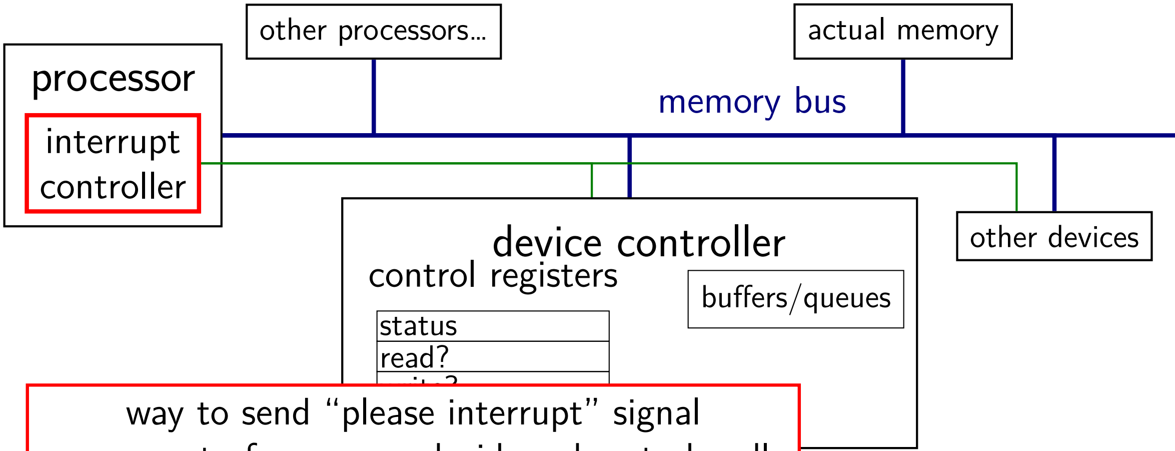


connecting devices



buffers/queues will also have memory addresses

connecting devices



way to send "please interrupt" signal
component of processor decides when to handle
(deals with ordering, interrupt disabling,
which of several processors handles it, ..., etc.)