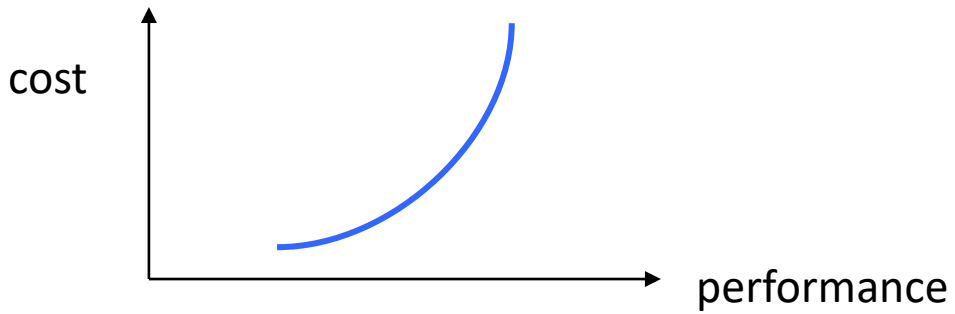
Networks and Multiprocessors

- Why networks and multiprocessors?
- Categories of multiprocessors.
- MPSoCs and shared memory multiprocessors.

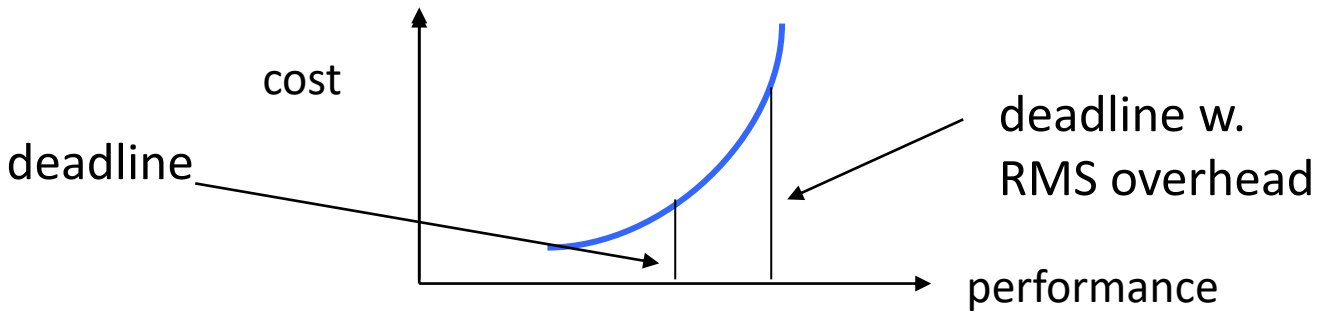
Why multiprocessors?

- Better cost/performance.
  - Match each CPU to its tasks or use custom logic (smaller, cheaper).
  - CPU cost is a non-linear function of performance.



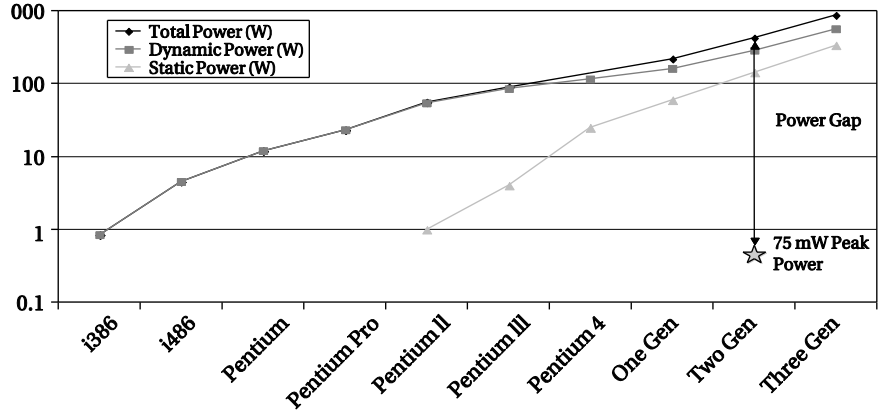
Why multiprocessors? cont'd.

- Better real-time performance.
  - Put time-critical functions on less-loaded processing elements.
  - Remember RMS utilization---extra CPU cycles must be reserved to meet deadlines.



Why multiprocessors? cont'd.

- Using specialized processors or custom logic saves power.
- Desktop uniprocessors are not power-efficient enough for battery-powered applications.

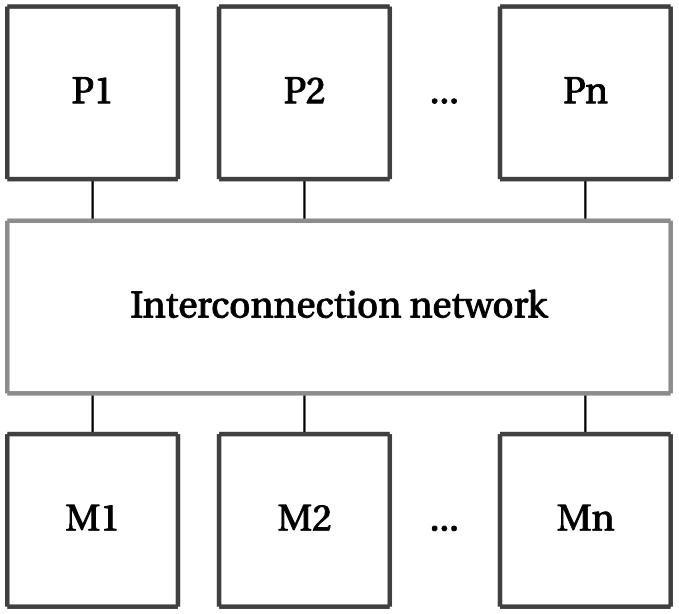


[Aus04] © 2004 IEEE Computer Society

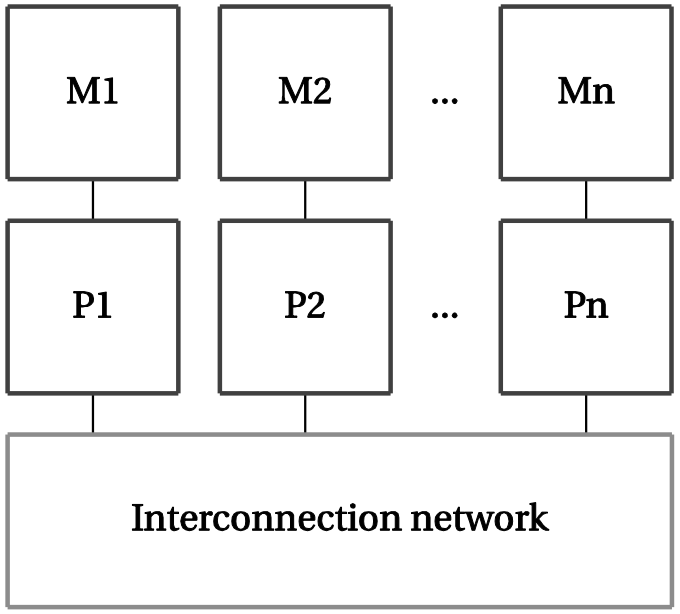
Why multiprocessors? cont'd.

- Good for processing I/O in real-time.
- May consume less energy.
- May be better at streaming data.
- May not be able to do all the work on even the largest single CPU.

Categories of multiprocessors



Shared memory



Message passing

Multiprocessor taxonomy

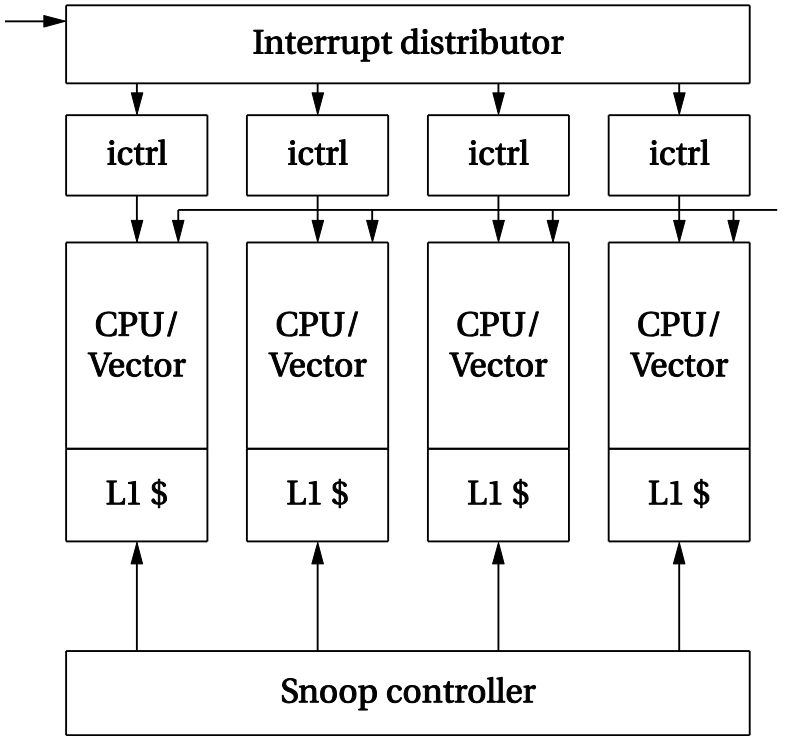
- PE: processing element.
- PEs and memory connected by interconnection network.
- Shared memory: all PEs talk to a pool of memory.
- Message passing: each PE has its own memory, communicates with messages.
- Shared memory and message passing are equivalent.

Varieties of multiprocessor

- System-on-chip is complete system, usually a multiprocessor, with I/O and memory.
- A distributed system has multiple chips and significant communication delay.

ARM MPCore

- Up to four CPUs.
- Distributed interrupt system.
- Snooping cache controller.



Why distributed?

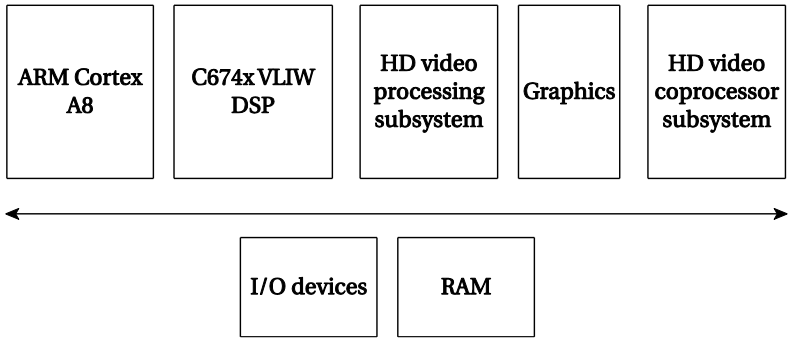
- Higher performance at lower cost.
- Physically distributed activities---time constants may not allow transmission to central site.
- Improved debugging---use one CPU in network to debug others.
- May buy subsystems that have embedded processors.

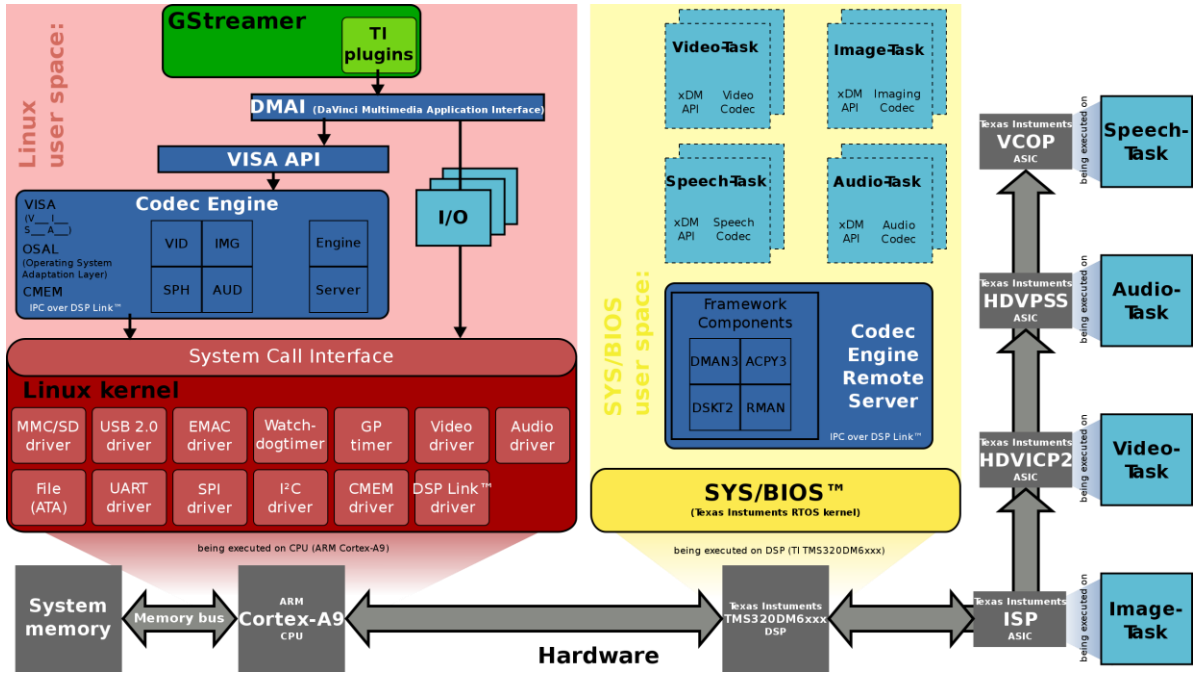
Shared memory multiprocessors

- Multi-core systems are symmetric multiprocessors with identical PEs.
- Many embedded multiprocessors are heterogeneous:
  - Multiple programmable CPU types.
  - Hardwired accelerators.
- Heterogeneous multiprocessors use less energy, are less expensive.

TI TMS320DM816x DaVinci

- ARM Cortex A8 plus C674x VLIW DSP.
- HD video coprocessor accelerates H.264, MPEG-4, etc.
- HD video processing susystem provides additional video processing.
- Graphics unit performs 3D graphics.

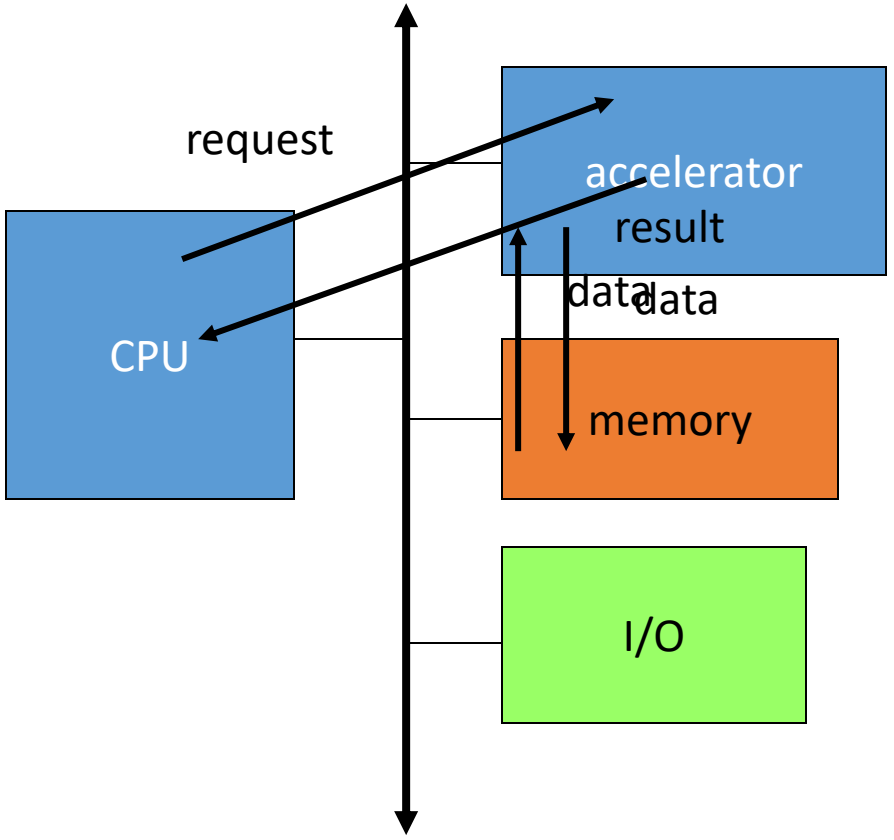




Accelerated systems

- Use additional computational unit dedicated to some functions?
  - Hardwired logic.
  - Extra CPU.
- Hardware/software co-design: joint design of hardware and software architectures.

Accelerated system architecture



Accelerator vs. co-processor

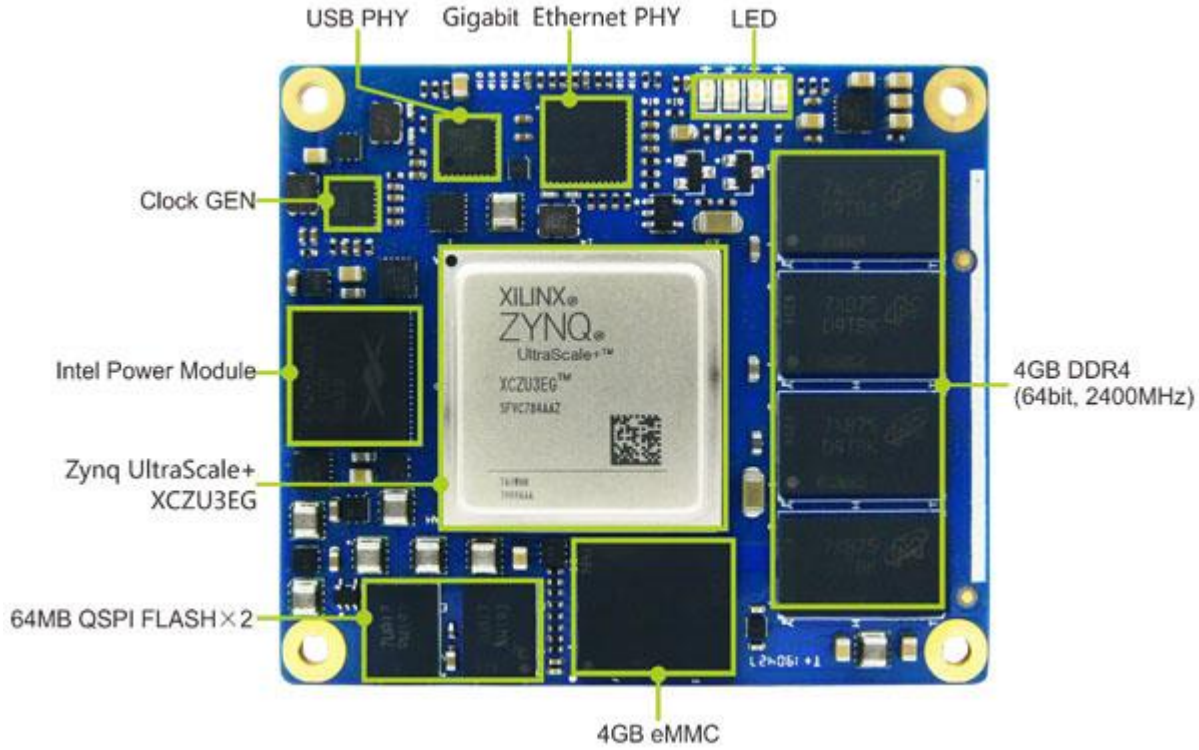
- A co-processor executes instructions.
  - Instructions are dispatched by the CPU.
- An accelerator appears as a device on the bus.
  - The accelerator is controlled by registers.

Accelerator implementations

- Application-specific integrated circuit.
- Field-programmable gate array (FPGA).
- Standard component.
  - Example: graphics processor.

Xilinx Zynq UltraScale+

- Quad-core ARM Cortex-A53 and dual-core ARM Cortex-R5.
- Mali graphics unit.
- FPGA fabric and block RAM.
- PCI3, SATA, USB, CAN, SPI, GPIO.



System design tasks

- Design a heterogeneous multiprocessor architecture.
  - Processing element (PE): CPU, accelerator, etc.
- Program the system.

Accelerated system design

- First, determine that the system really needs to be accelerated.
  - How much faster is the accelerator on the core function?
  - How much data transfer overhead?
- Design the accelerator itself.
- Design CPU interface to accelerator.

Accelerated system platforms

- Several off-the-shelf boards are available for acceleration in PCs:
  - FPGA-based core;
  - PC bus interface.

Accelerator/CPU interface

- Accelerator registers provide control registers for CPU.
- Data registers can be used for small data objects.
- Accelerator may include special-purpose read/write logic.
  - Especially valuable for large data transfers.

System integration and debugging

- Try to debug the CPU/accelerator interface separately from the accelerator core.
- Build scaffolding to test the accelerator.
- Hardware/software co-simulation can be useful.

Caching problems

- Main memory provides the primary data transfer mechanism to the accelerator.
- Programs must ensure that caching does not invalidate main memory data.
  - CPU reads location S.
  - Accelerator writes location S.
  - CPU writes location S.

BAD

Synchronization

- As with cache, main memory writes to shared memory may cause invalidation:
  - CPU reads S.
  - Accelerator writes S.
  - CPU reads S.

Multiprocessor performance analysis

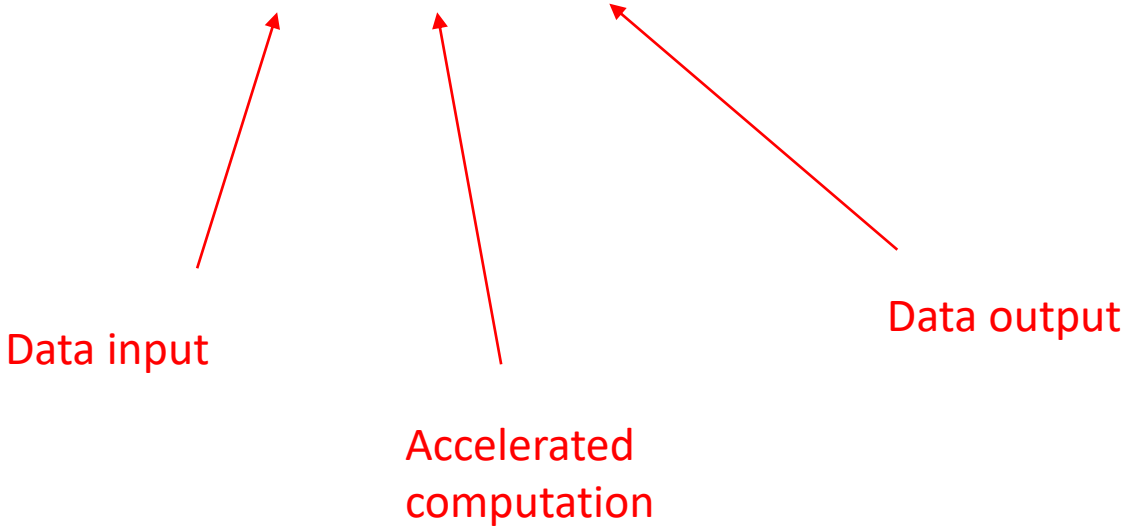
- Effects of parallelism (and lack of it):
  - Processes.
  - CPU and bus.
  - Multiple processors.

Accelerator speedup

- Critical parameter is speedup: how much faster is the system with the accelerator?
- Must take into account:
  - Accelerator execution time.
  - Data transfer time.
  - Synchronization with the master CPU.

Accelerator execution time

- Total accelerator execution time:
  - $t_{accel} = t_{in} + t_x + t_{out}$



Accelerator speedup

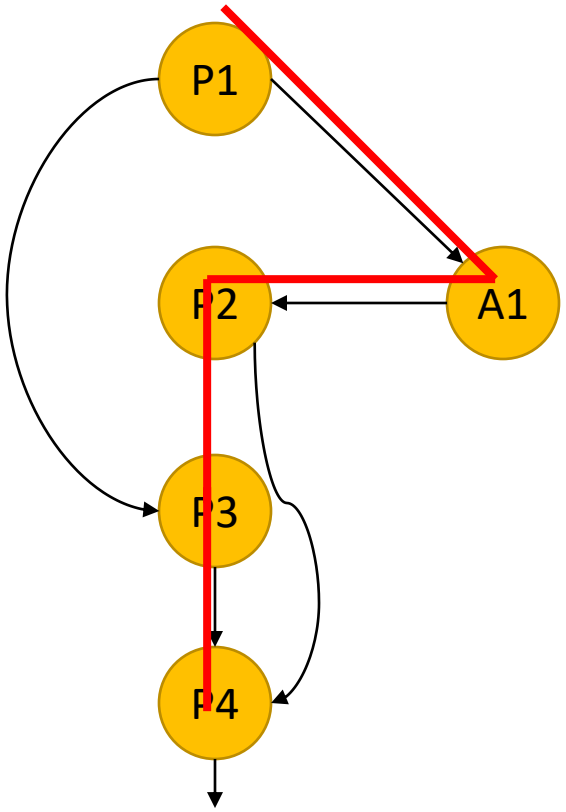
- Assume loop is executed n times.
- Compare accelerated system to non-accelerated system:
  - $S = n(t_{CPU} - t_{accel})$
  - $= n[t_{CPU} - (t_{in} + t_x + t_{out})]$

Single- vs. multi-threaded

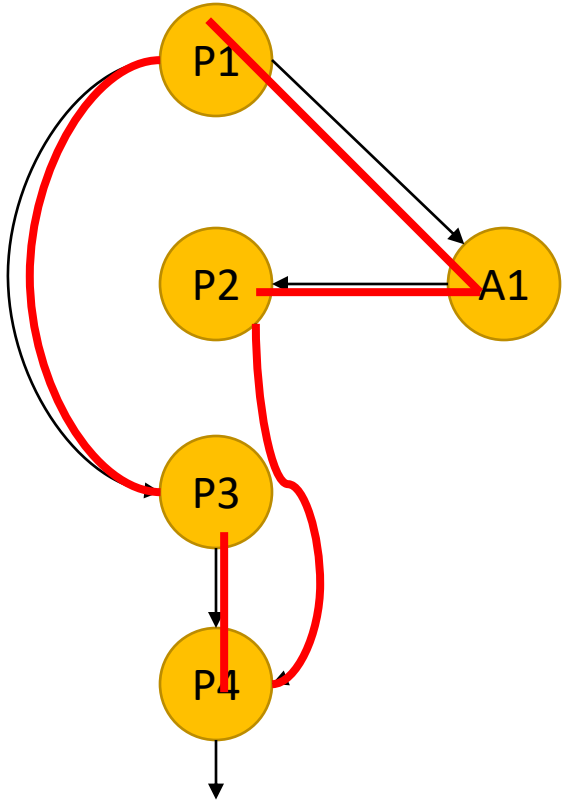
- One critical factor is available parallelism:
  - single-threaded/blocking: CPU waits for accelerator;
  - multithreaded/non-blocking: CPU continues to execute along with accelerator.
- To multithread, CPU must have useful work to do.
  - But software must also support multithreading.

Total execution time

- Single-threaded:



- Multi-threaded:



Execution time analysis

- Single-threaded:
  - Count execution time of all component processes.
- Multi-threaded:
  - Find longest path through execution.

Sources of parallelism

- Overlap I/O and accelerator computation.
  - Perform operations in batches, read in second batch of data while computing on first batch.
- Find other work to do on the CPU.
  - May reschedule operations to move work after accelerator initiation.

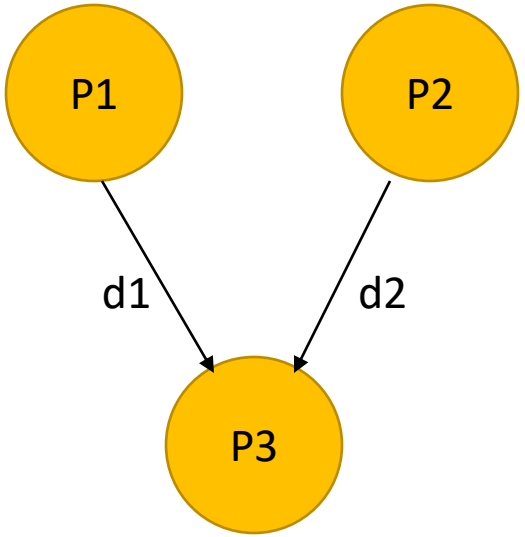
Data input/output times

- Bus transactions include:
  - flushing register/cache values to main memory;
  - time required for CPU to set up transaction;
  - overhead of data transfers by bus packets, handshaking, etc.

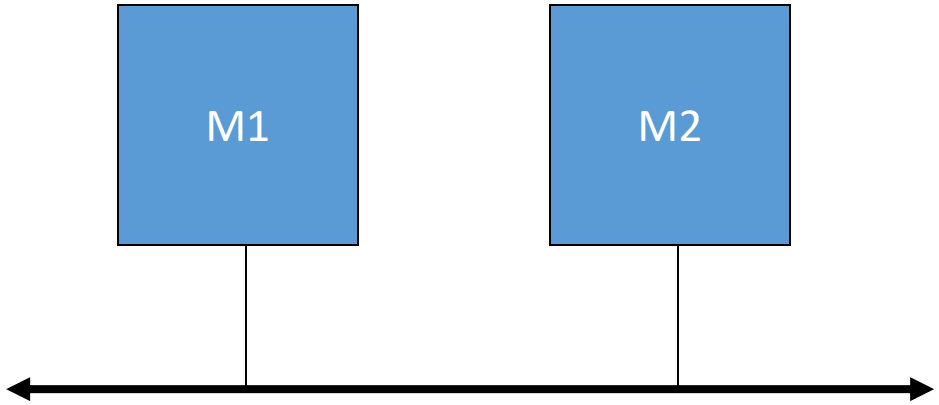
Scheduling and allocation

- Must:
  - schedule operations in time;
  - allocate computations to processing elements.
- Scheduling and allocation interact, but separating them helps.
  - Alternatively allocate, then schedule.

Example: scheduling and allocation



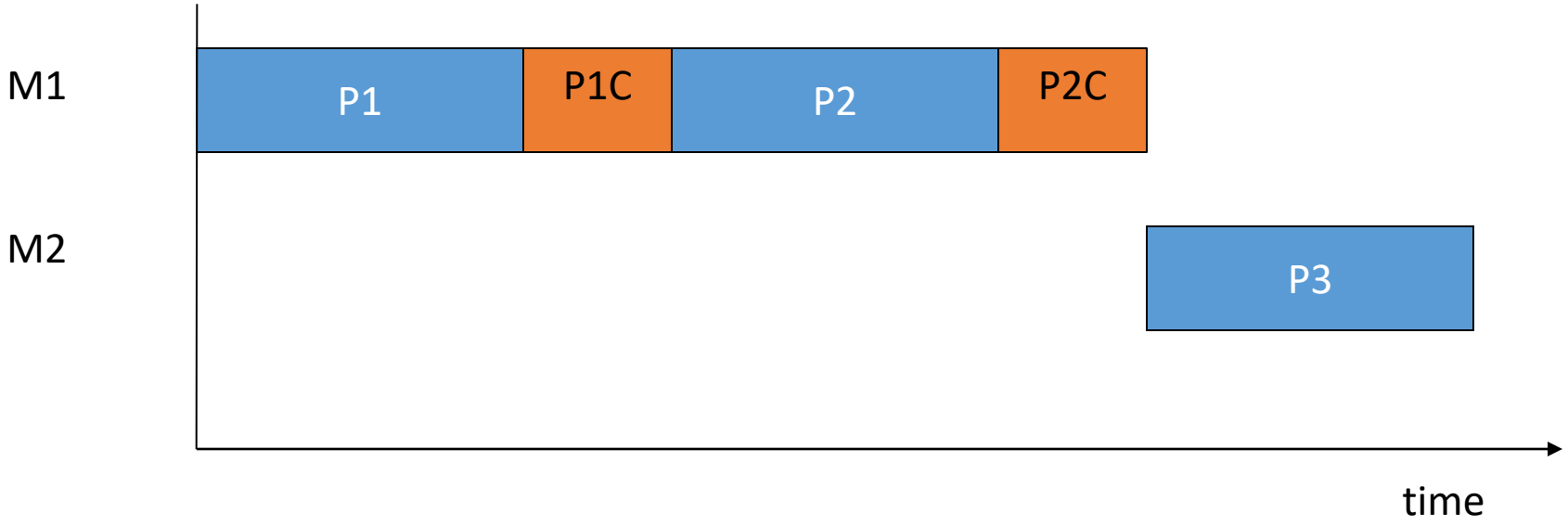
Task graph



Hardware platform

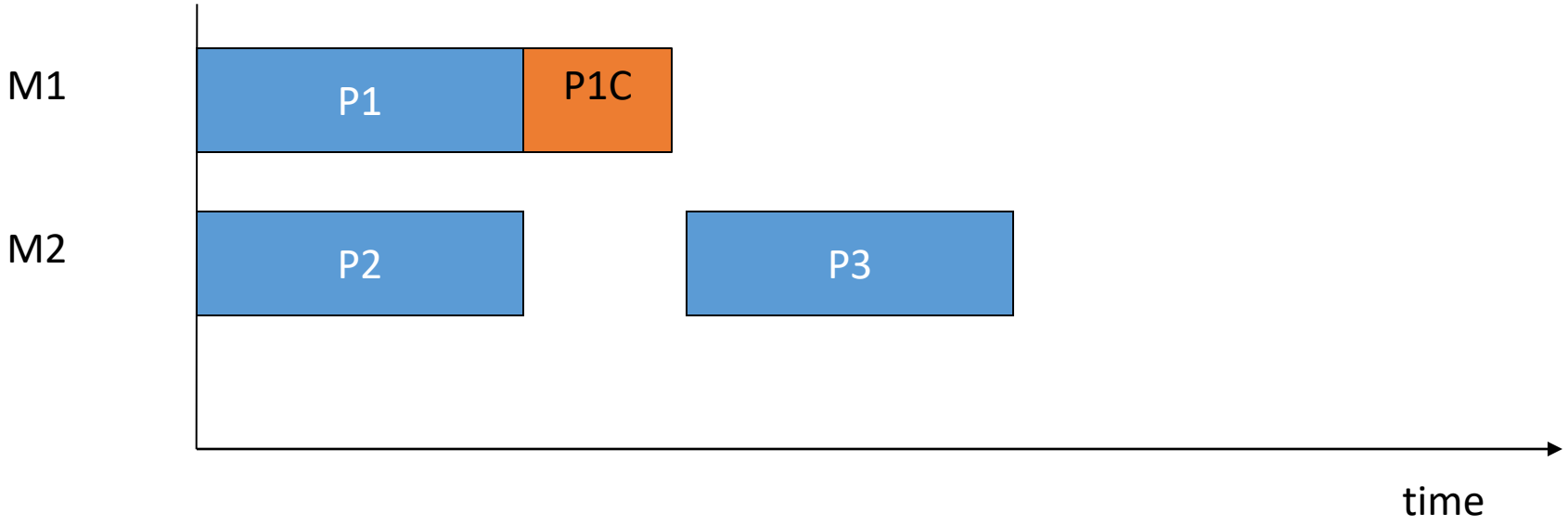
First design

- Allocate P1, P2 -> M1; P3 -> M2.



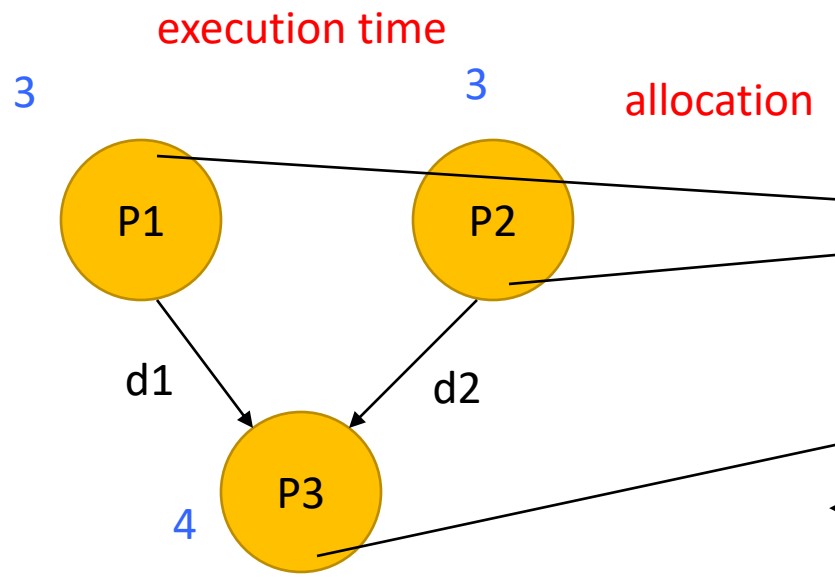
Second design

- Allocate P1 -> M1; P2, P3 -> M2:

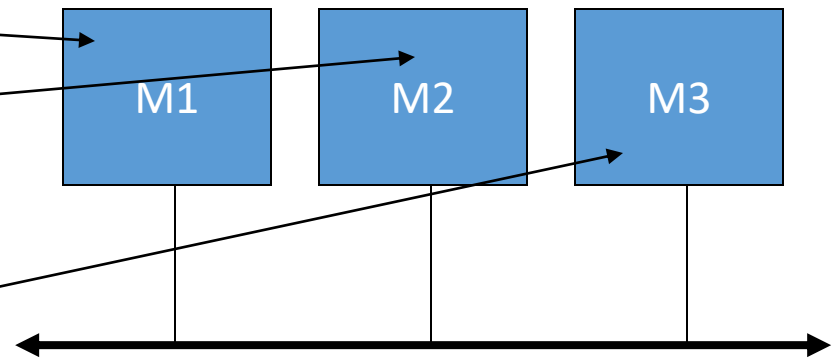


Example: adjusting messages to reduce delay

- Task graph:

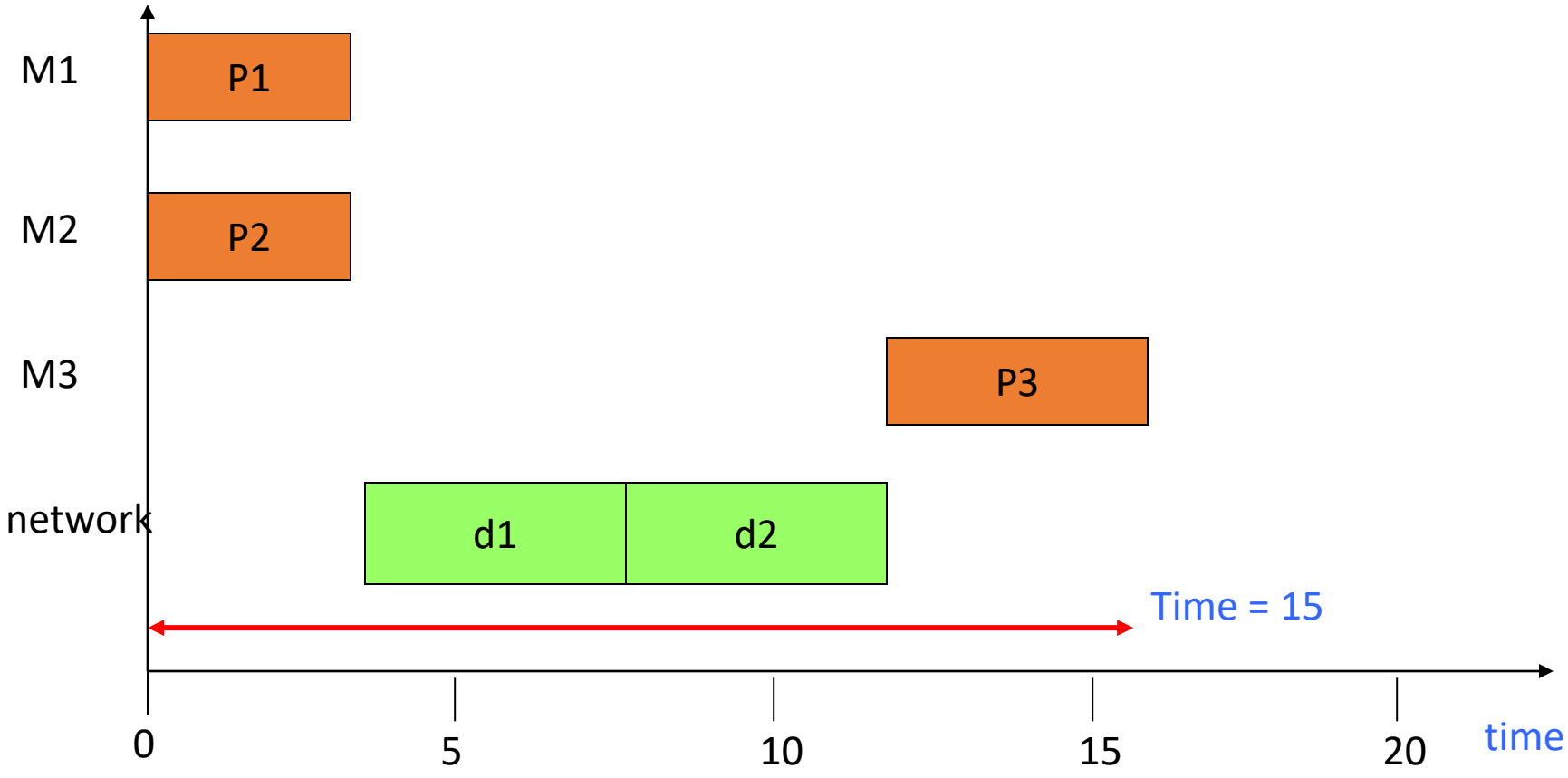


- Network:



Transmission time = 4

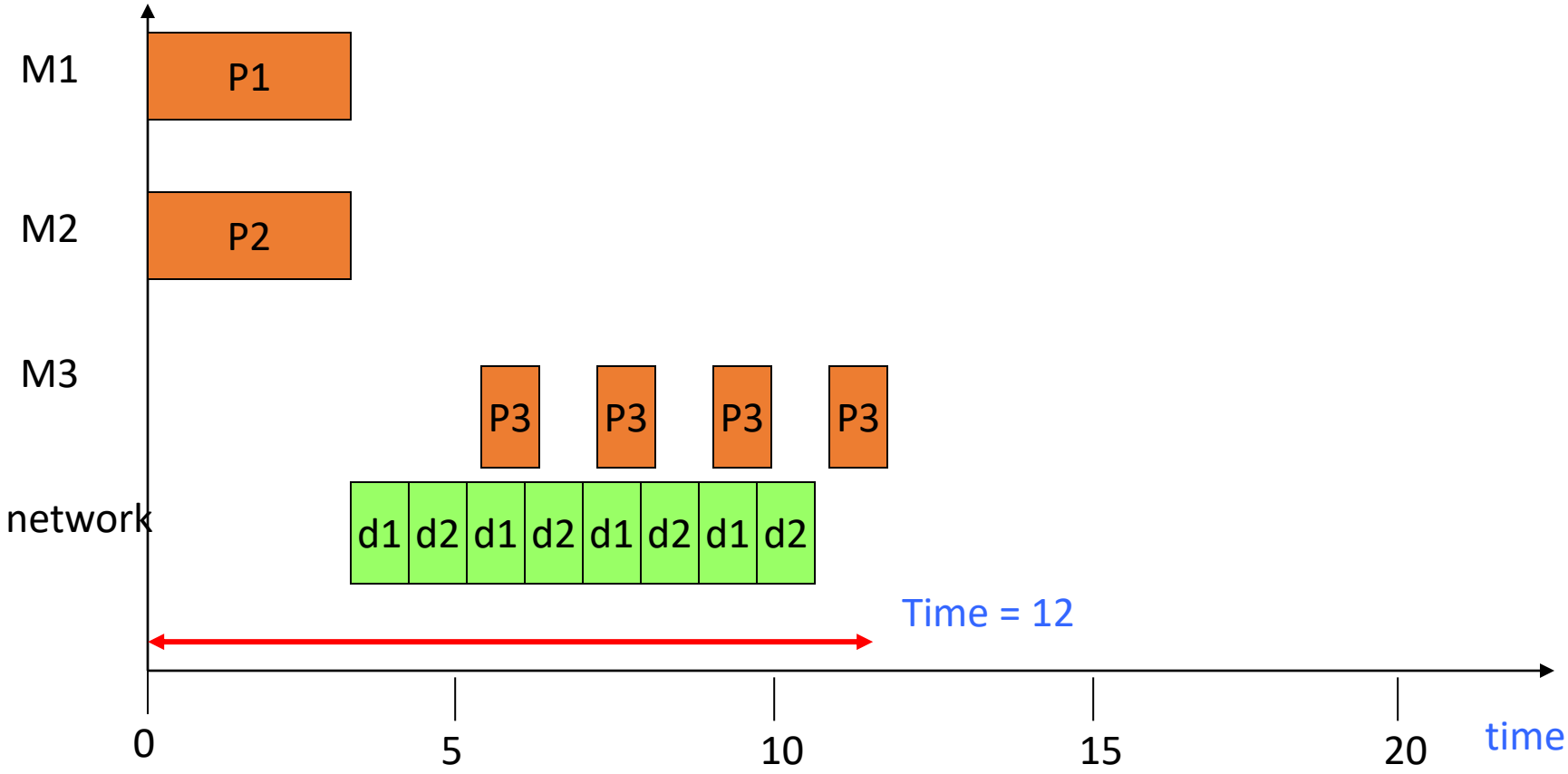
Initial schedule



New design

- Modify P3:
  - reads one packet of d1, one packet of d2
  - computes partial result
  - continues to next packet

New schedule

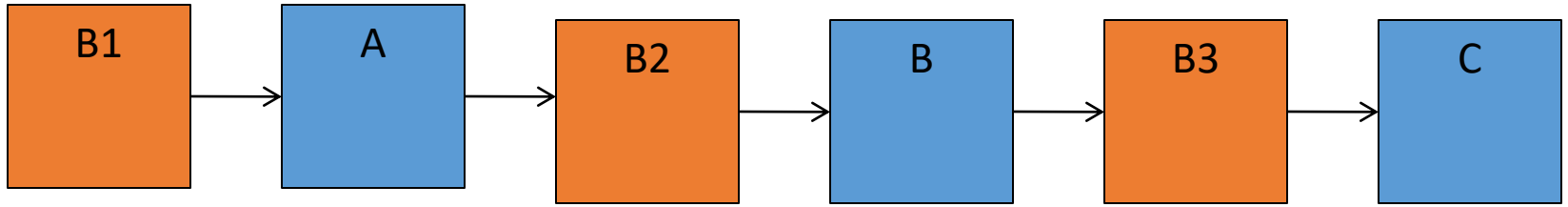


Buffering and performance

- Buffering may sequentialize operations.
  - Next process must wait for data to enter buffer before it can continue.
- Buffer policy (queue, RAM) affects available parallelism.

Buffers and latency

- Three processes separated by buffers:



Buffers and latency schedules

A[0]
A[1]
…
B[0]
B[1]
…
C[0]
C[1]
…

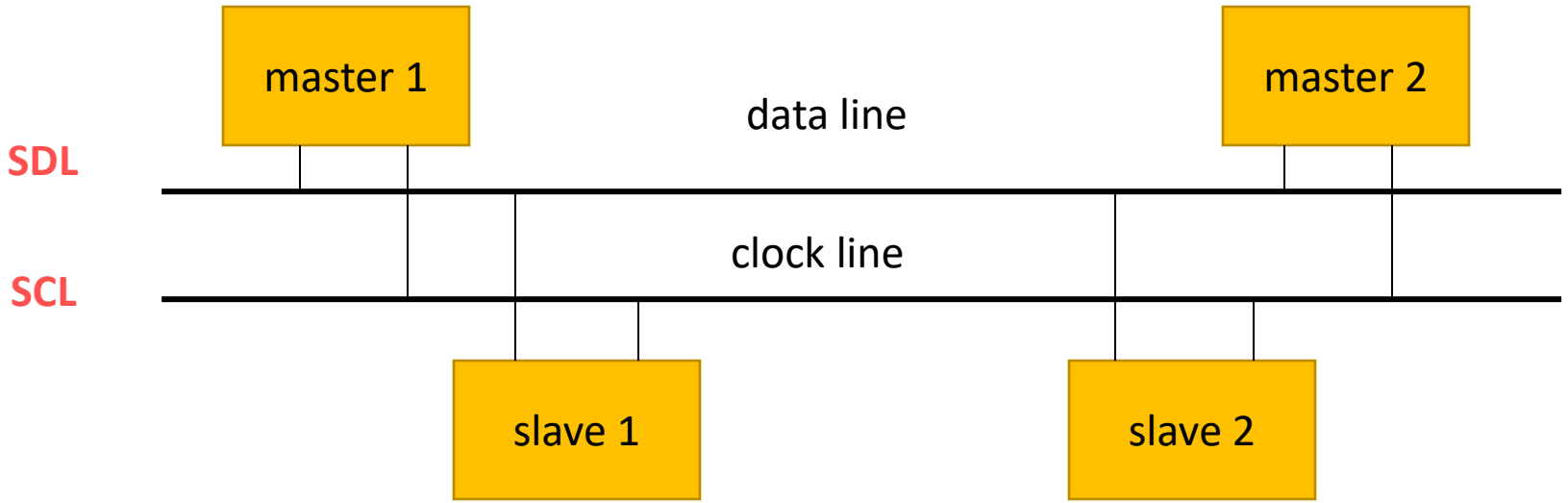
Must wait for all of A before getting any B

A[0]
B[0]
C[0]
A[1]
B[1]
C[1]
…

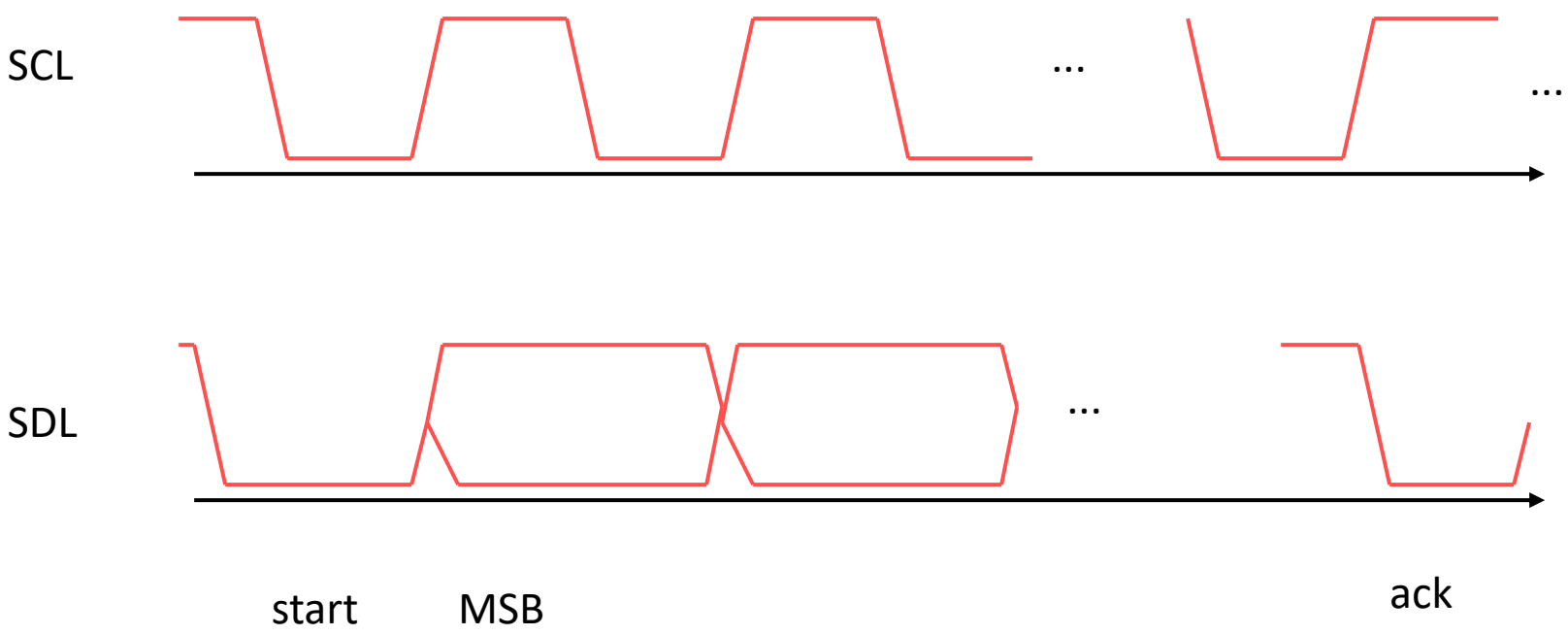
I^2C bus

- Designed for low-cost, medium data rate applications.
- Characteristics:
  - serial;
  - multiple-master;
  - fixed-priority arbitration.
- Several microcontrollers come with built-in I^2C controllers.

I^2C physical layer

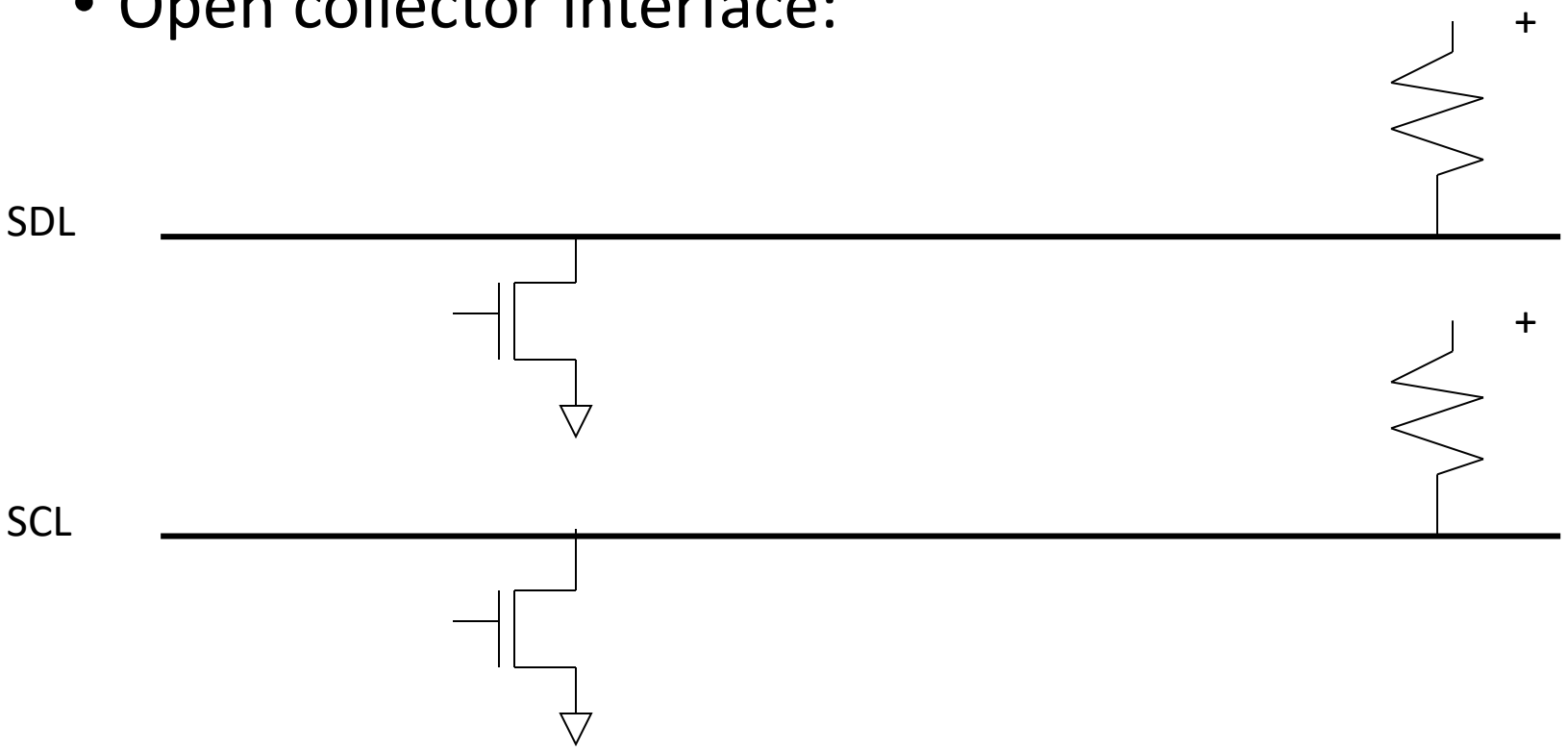


I^2C data format



I^2C electrical interface

- Open collector interface:



I^2C signaling

- Sender pulls down bus for 0.
- Sender listens to bus---if it tried to send a 1 and heard a 0, someone else is simultaneously transmitting.
- Transmissions occur in 8-bit bytes.

I^2C data link layer

- Every device has an address (7 bits in standard, 10 bits in extension).
  - Bit 8 of address signals read or write.
- General call address allows broadcast.

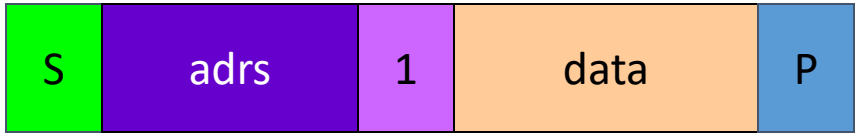
I^2C bus arbitration

- Sender listens while sending address.
- When sender hears a conflict, if its address is higher, it stops signaling.
- Low-priority senders relinquish control early enough in clock cycle to allow bit to be transmitted reliably.

I^2C transmissions

multi-byte write

read from slave



write, then read