Hardware Accelerators

José Costa

Software for Embedded Systems

Departamento de Engenharia Informática (DEI)
Instituto Superior Técnico

2015-10-27

Outline

- Hardware Accelerators
- CPUs and Accelerators
- Accelerated System Design
- Partitioning
- Scheduling and Allocation
- System Integration and Debugging

Accelerated Systems (1/2)

Accelerators

Use additional computational unit dedicated to some functions

- hardwired logic
- extra CPU

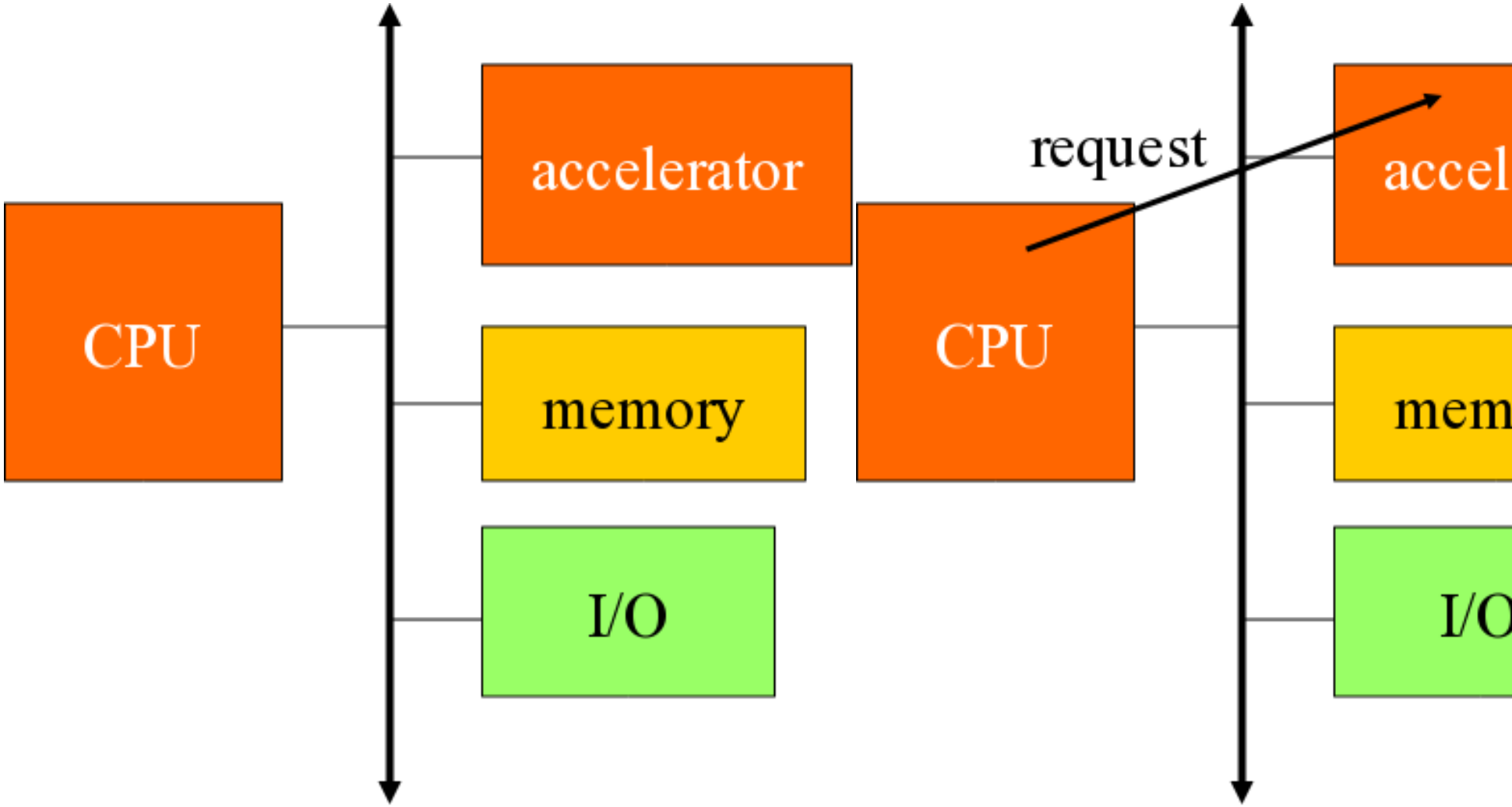
Hardware/software co-design

Joint design of hardware and software architectures

Accelerated Systems (2/2)

- May provide large performance increase
  - when execute code section where applications spends great deal of time
- May provide critical speedups for low-latency I/O functions
- Better meeting of our application's demands

Accelerated System Architecture



Accelerator vs. Co-Processor

Co-processor

A co-processor executes instructions

- connected to the internals of the CPU
- instructions are dispatched by the CPU
- example: Numerical co-processing in the Intel x86

Accelerator

An accelerator appears as a device on the bus

- the accelerator is controlled by registers or shared memory
- it does not execute instructions

Accelerator Implementations

Accelerators are designed to perform a specific function

Accelerator Implementations

- Application-specific integrated circuit (ASIC)
  - considerable design and manufacturing time
  - may be worth the effort for high volume designs
- Field-programmable gate array (FPGA)
  - can quickly be customized to a particular function
- Standard component
  - may have to design interface logic to mate it to our CPU
  - example: graphics processor

System Design Tasks

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

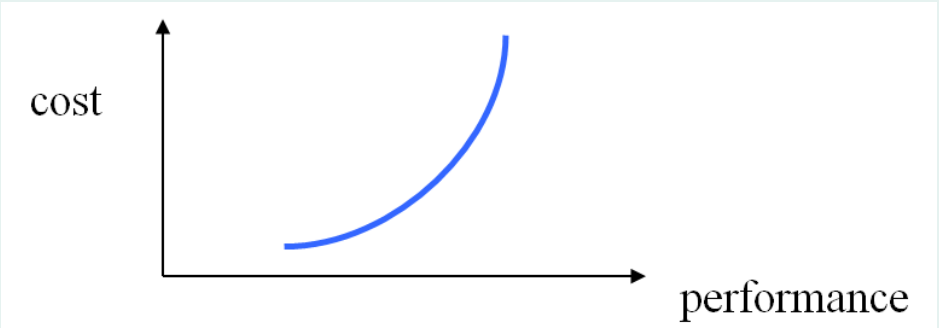
Why Accelerators? (1/4)

- They require considerable effort
- Thus, payoff should be considerable
- But, there are several good reasons to add accelerators to embedded systems

Why Accelerators? (2/4)

Better cost/performance

- Custom logic may be able to perform operation faster than a CPU of equivalent cost
- CPU cost is a non-linear function of performance

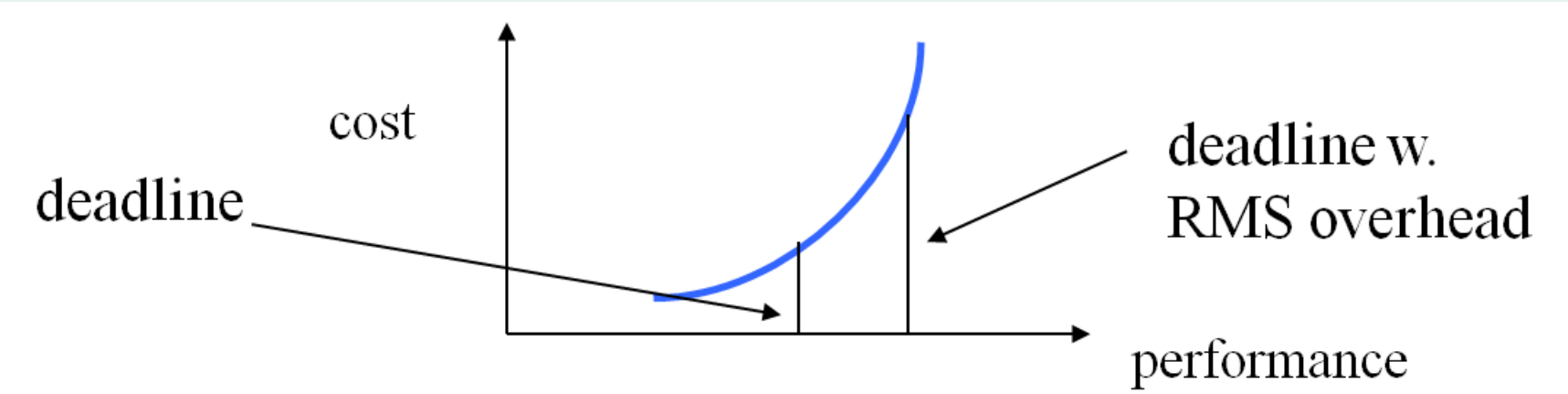


- Higher engineering costs and lead times must be factored into the project
- But even with the added cost of assembling multiple processing elements, the total system can be less expensive

Why Accelerators? (3/4)

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 Accelerators? (4/4)

- Some functions may not map well onto a CPU's data operations
  - bit level operations
  - use too many registers
- Good for processing I/O in real-time
  - more efficient when data must be read, processed and written to meet a tight deadline
- May consume less energy
- May be better at streaming data
- It may not be possible to do all the work on even the largest single CPU

Accelerated System Design

1. Determine that the system really needs to be accelerated
   - how much faster is the accelerator on the core function?
   - how much data transfer overhead?
2. Design the accelerator itself
   - translate the algorithm description into a hardware design
   - design the interface
3. Design CPU interface to accelerator
   - syncronize the operations of the accelerator with the rest of the application

Performance Analysis

Critical parameter is speedup

How much faster is the system with the accelerator?

- power consumption and manufacturing costs are less important

Computation of speedup not trivial

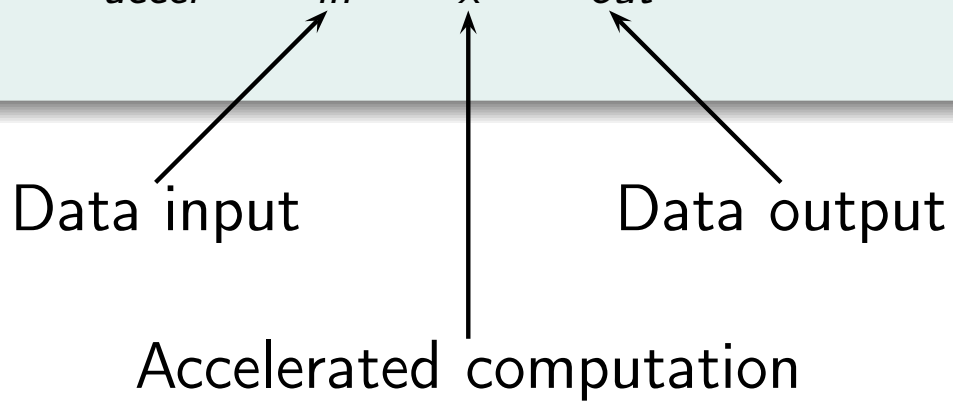
Must take into account

- accelerator execution time
- data transfer time
- synchronization with the master CPU

Accelerator Execution Time

Total accelerator execution time

$$t_{accel} = t_{in} + t_x + t_{out}$$



Data Input/Output Times

Total accelerator execution time

$$t_{accel} = t_{in} + t_x + t_{out}$$

t_{in} and t_{out} must reflect bus transaction times

Bus transactions include

- Flushing register/cache values to main memory (maintain cache-memory coherency)
- Time required for CPU to set up transaction
- Overhead of data transfers by bus packets, handshaking, etc.

Accelerator Gain

Example (How to compute accelerator gain)

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

Execution time on CPU

Single- vs. Multi-Threaded

One critical factor is available parallelism

Single-threaded/blocking

CPU waits for accelerator

Multithreaded/non-blocking

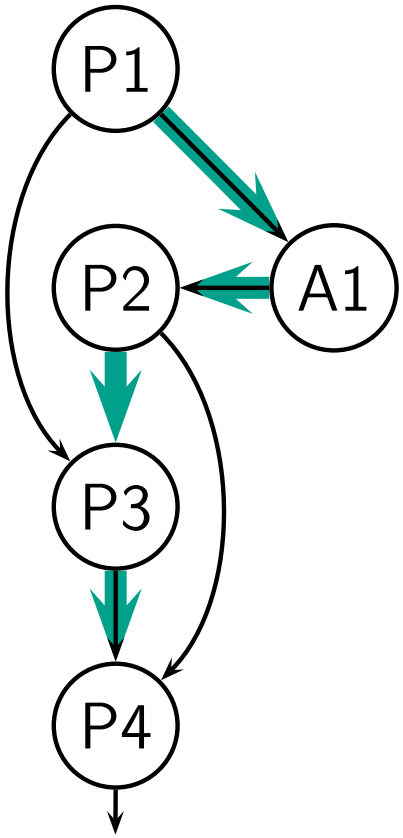
CPU continues to execute along with accelerator

- To multithread, CPU must have useful work to do
  - but software must also support multithreading

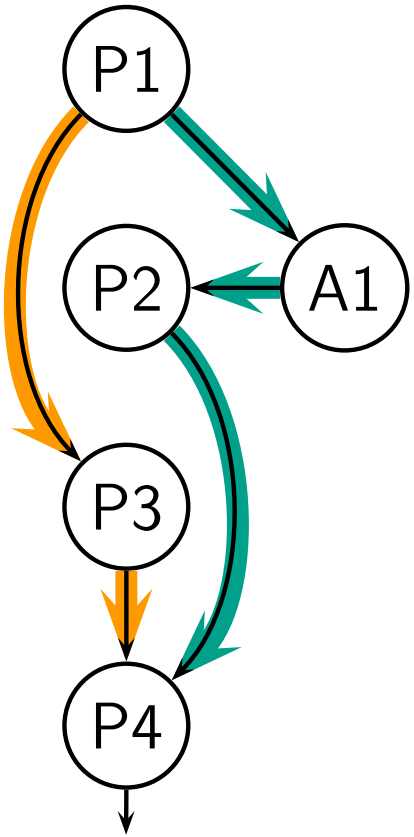
Total Execution Time

TÉCNICO LISBOA

Single Thread



Multiple Thread



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

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

Caching Problems

- Main memory provides the primary data transfer mechanism to the accelerator
- Programs must ensure that caching does not invalidate main memory data

Example (Caching problem)

- CPU reads location S
- accelerator writes location S
- CPU reads location S

Problem: CPU reads old value

Some invalidation mechanism must be present

Synchronization

- As with cache, main memory writes to shared memory may cause invalidation

Example (Synchronization problem)

- CPU reads location S
- accelerator writes location S
- CPU writes location S

Problem: Value written by accelerator was lost

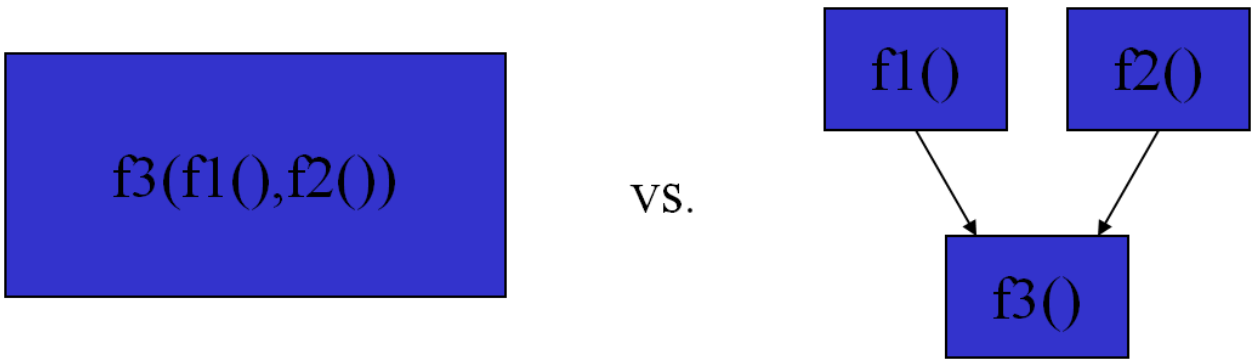
Some synchronization mechanism must be present

- CPU waits for accelerator to finnish
- CPU and accelerator use test-and-set atomic operations

Partitioning/Decomposition

- Divide functional specification into units
  - map units onto PEs
  - units may become processes

- Determine proper level of parallelism

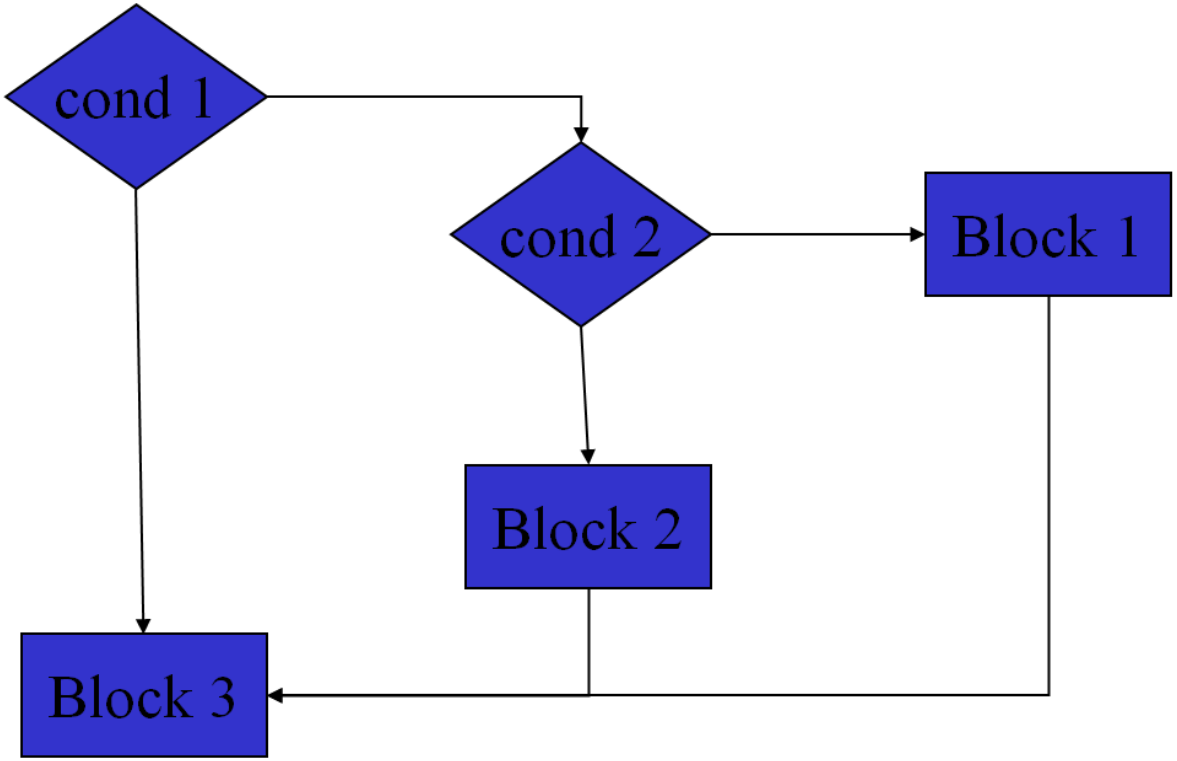


Decomposition Methodology

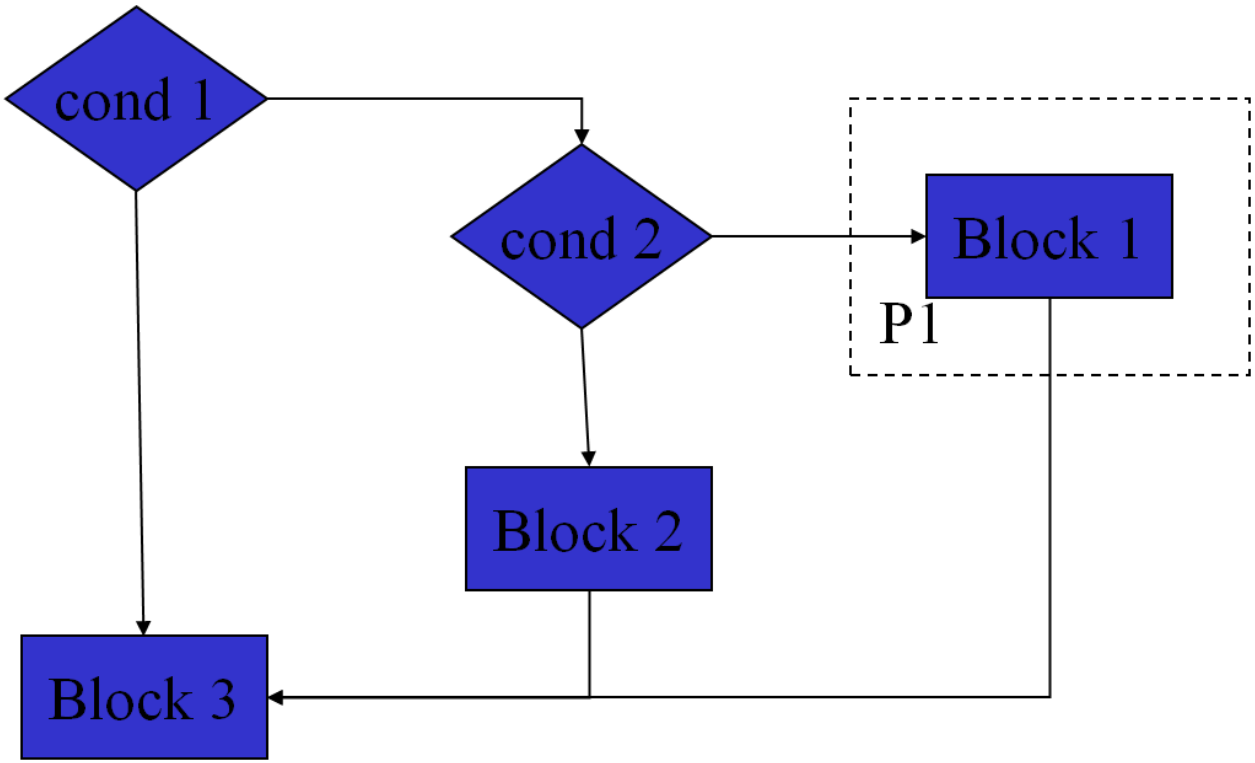
- Divide flow-graph (CDFG) into pieces, shuffle functions between pieces

- Hierarchically decompose CDFG to identify possible partitions

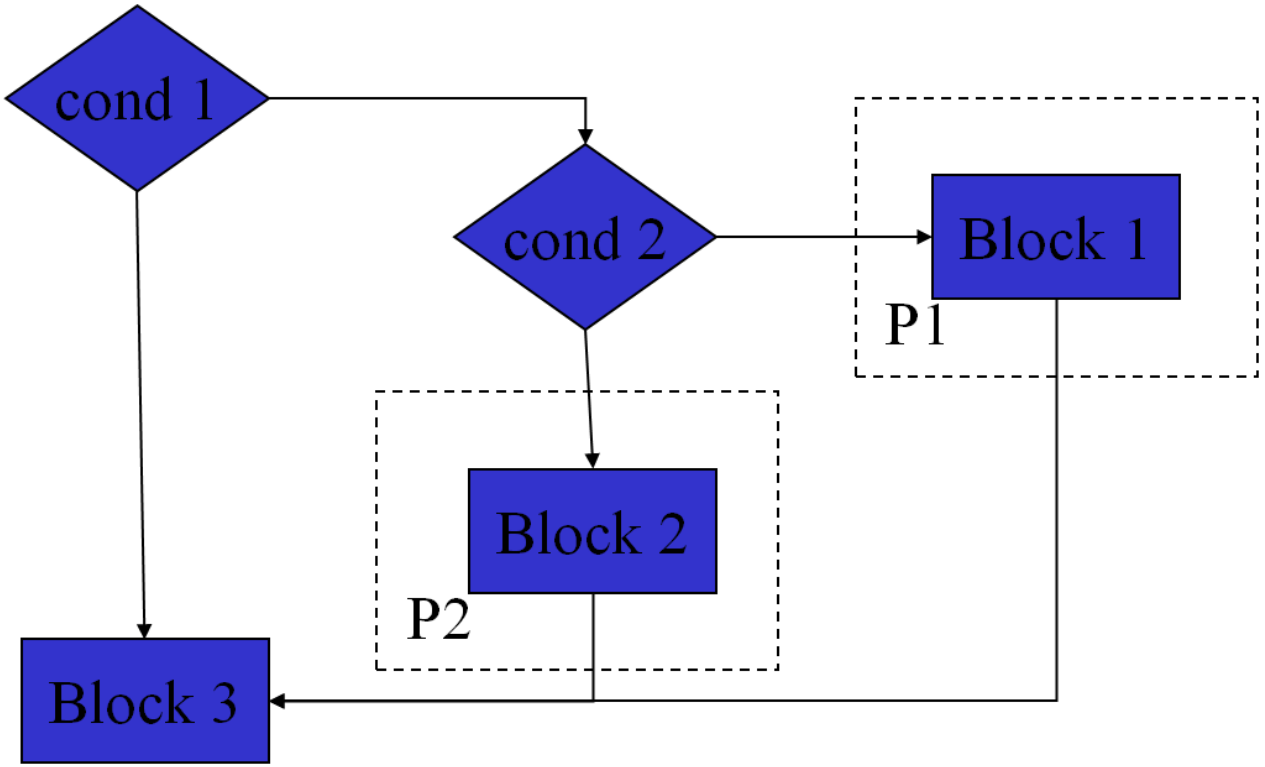
Decomposition Example



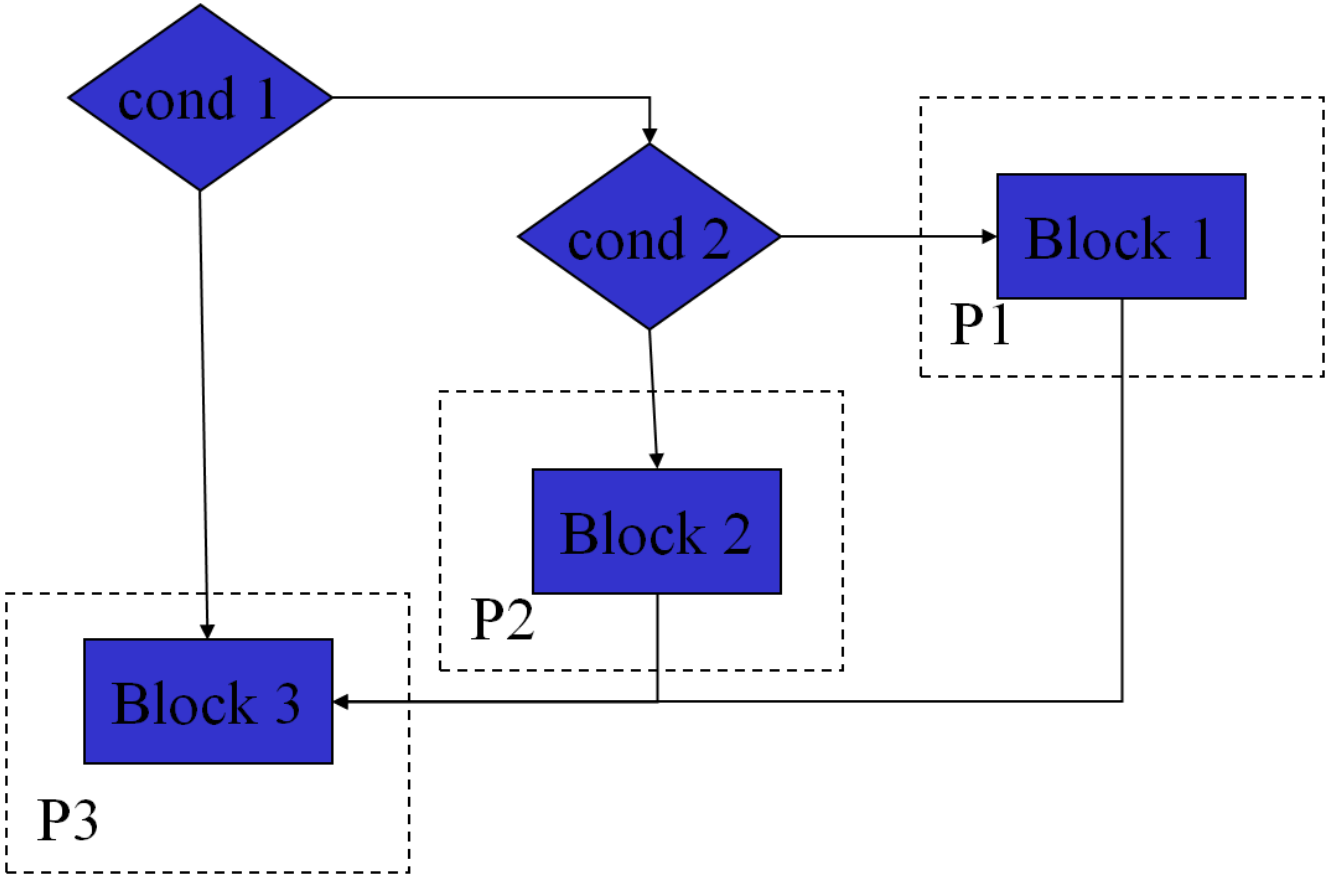
Decomposition Example



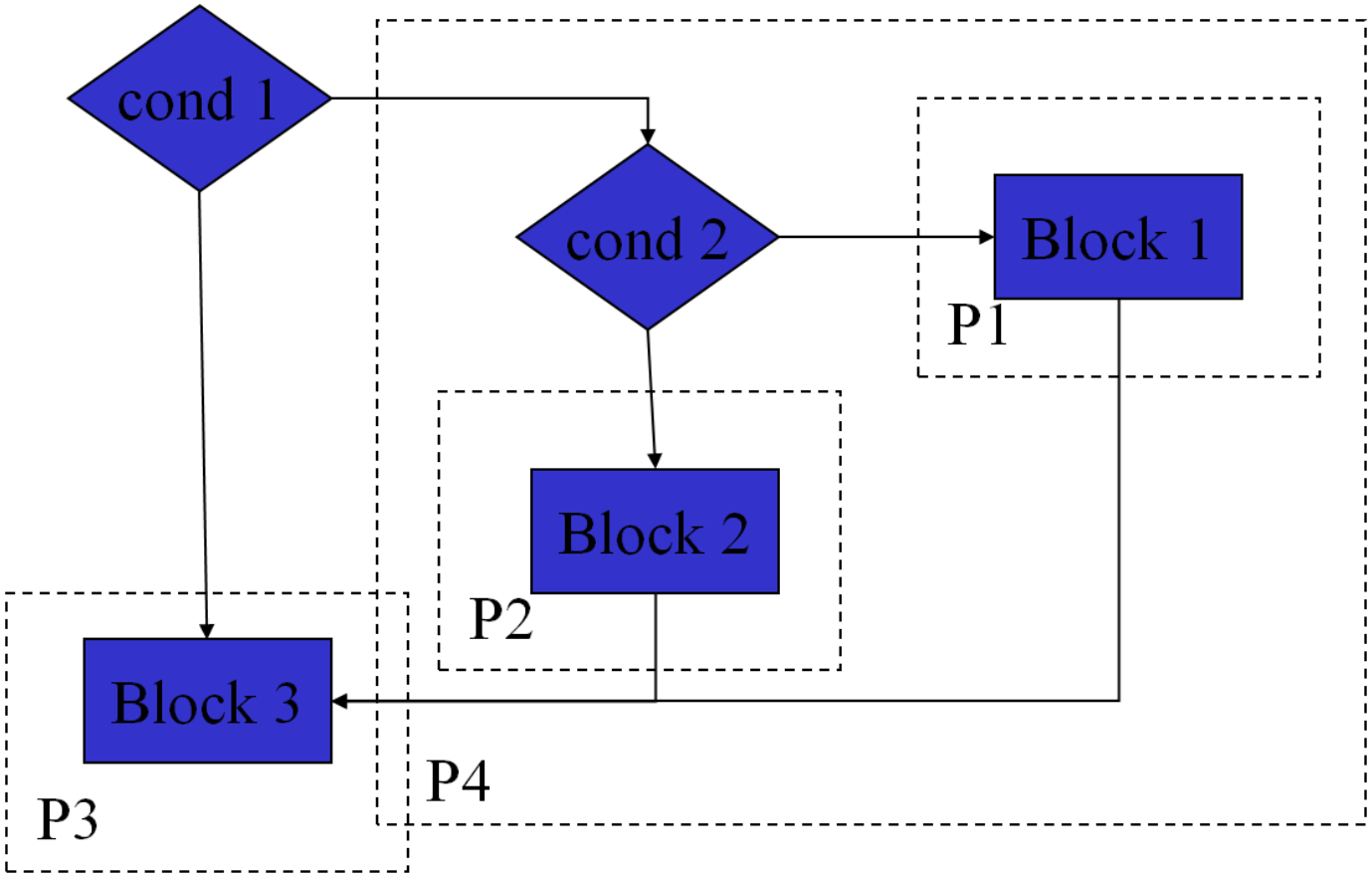
Decomposition Example



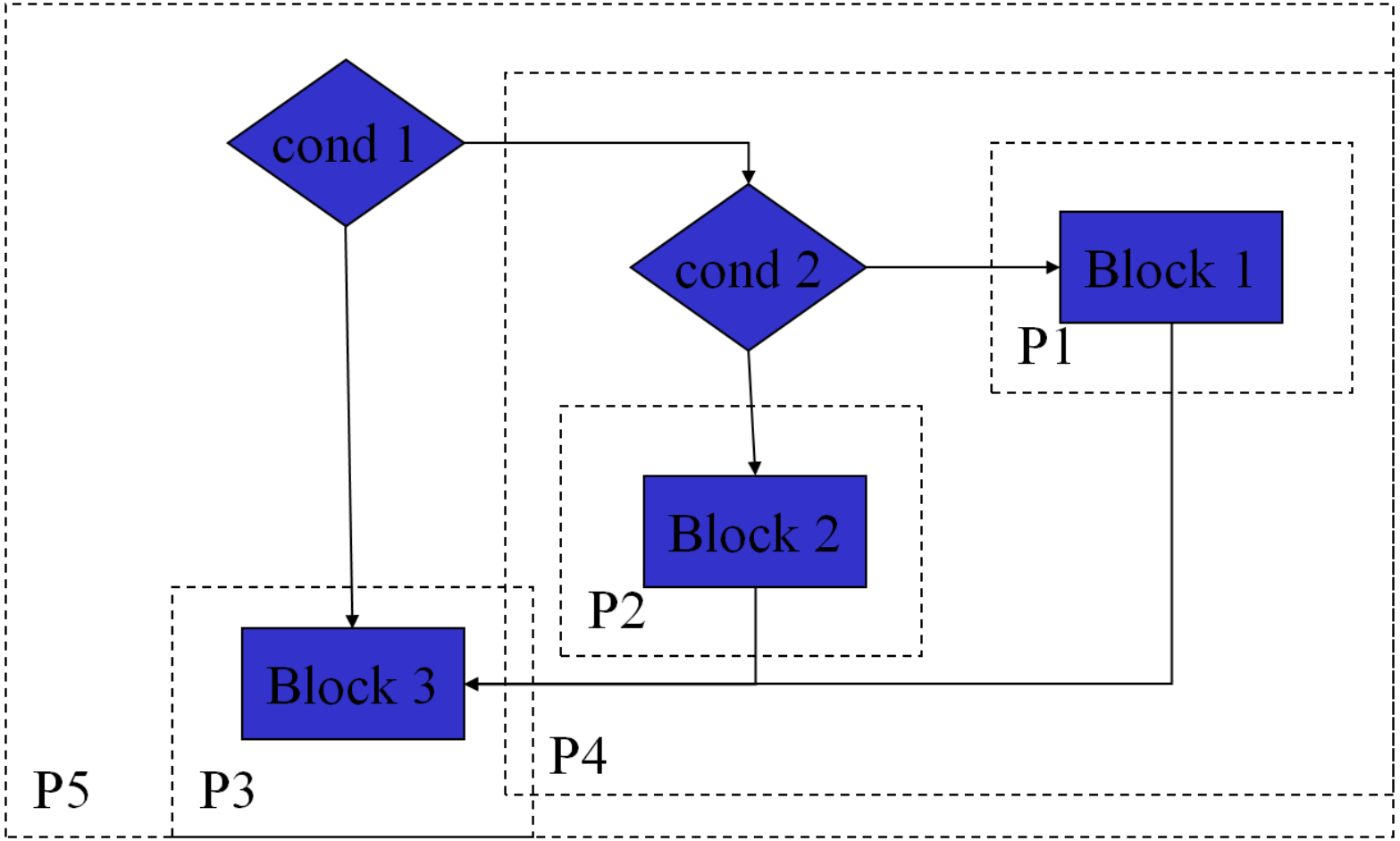
Decomposition Example



Decomposition Example



Decomposition Example

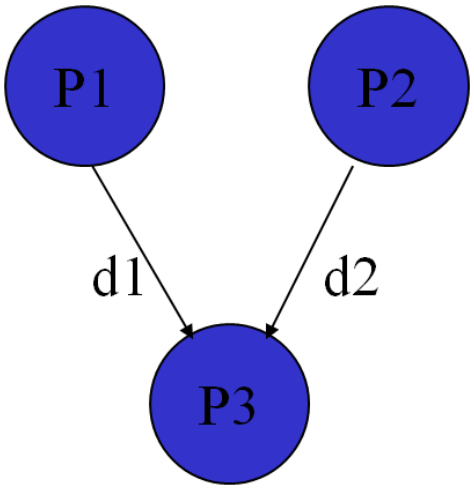


Scheduling and Allocation

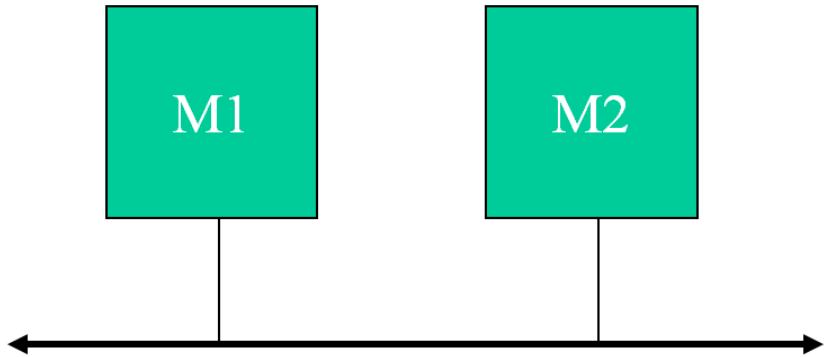
- Must:
  - schedule operations in time
  - allocate computations to processing elements

- Scheduling and allocation interact, but separating them helps
  - alternatively allocate, then schedule

Example: Scheduling and Allocation



Task graph



Hardware platform

Example: Process Execution Times

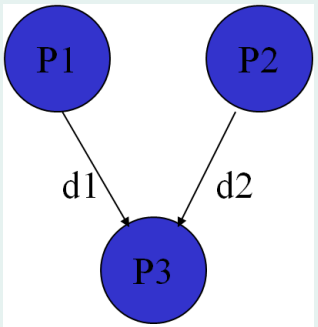
	M1	M2
P1	5	5
P2	5	6
P3	-	5

Example: Communication Model

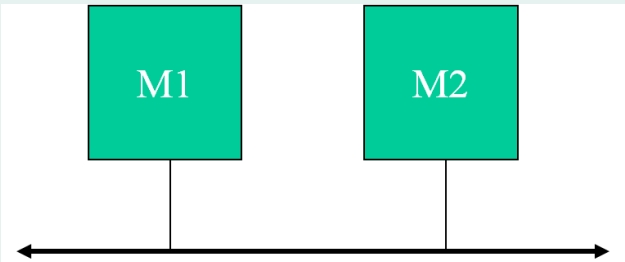
- Assume communication within PE is free
- Cost of P1->P3 communication is d1 = 2
- Cost of P2->P3 communication is d2 = 4

Example: Summary

Scheduling and allocation



Task graph



Hardware platform

Execution times

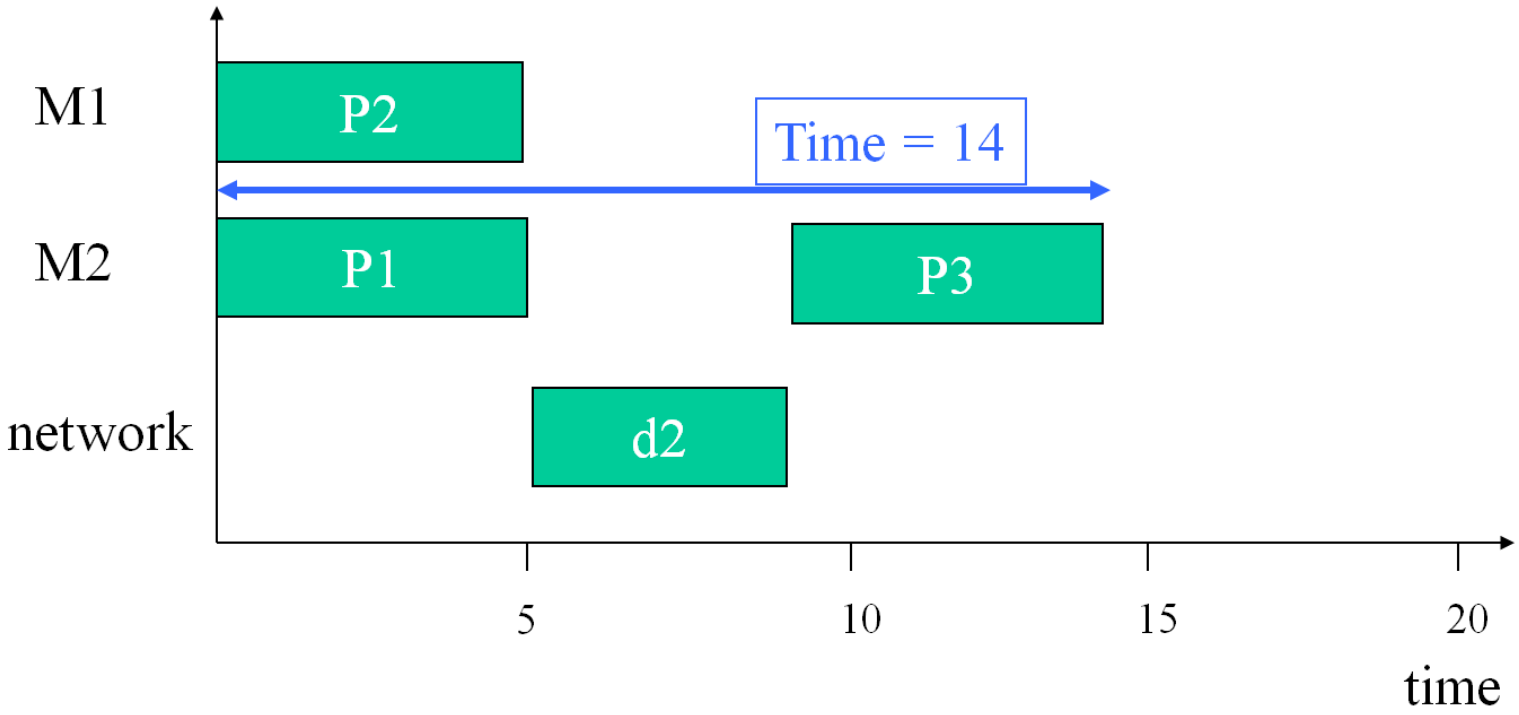
	M1	M2
P1	5	5
P2	5	6
P3	-	5

Communication model

- Assume communication within PE is free
- Cost of P1->P3 communication is d1 = 2
- Cost of P2->P3 communication is d2 = 4

Allocate

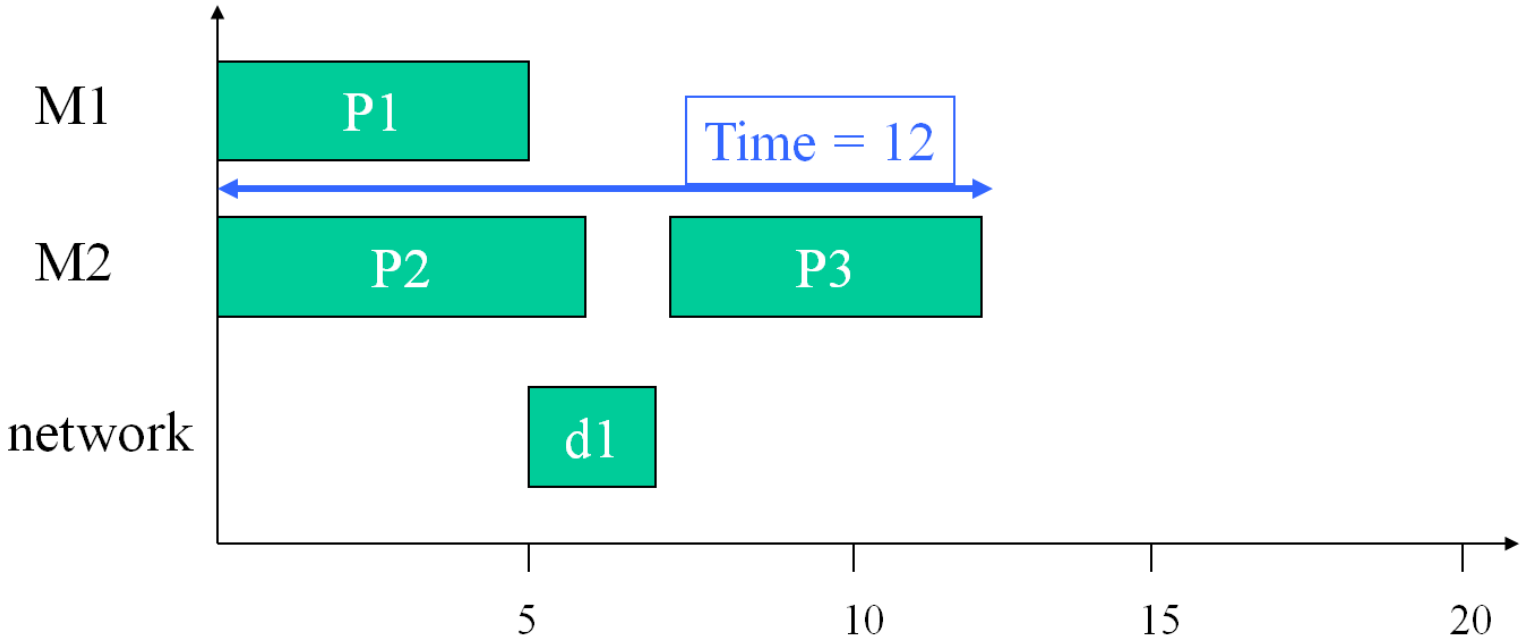
- P2 -> M1
- P1, P3 -> M2



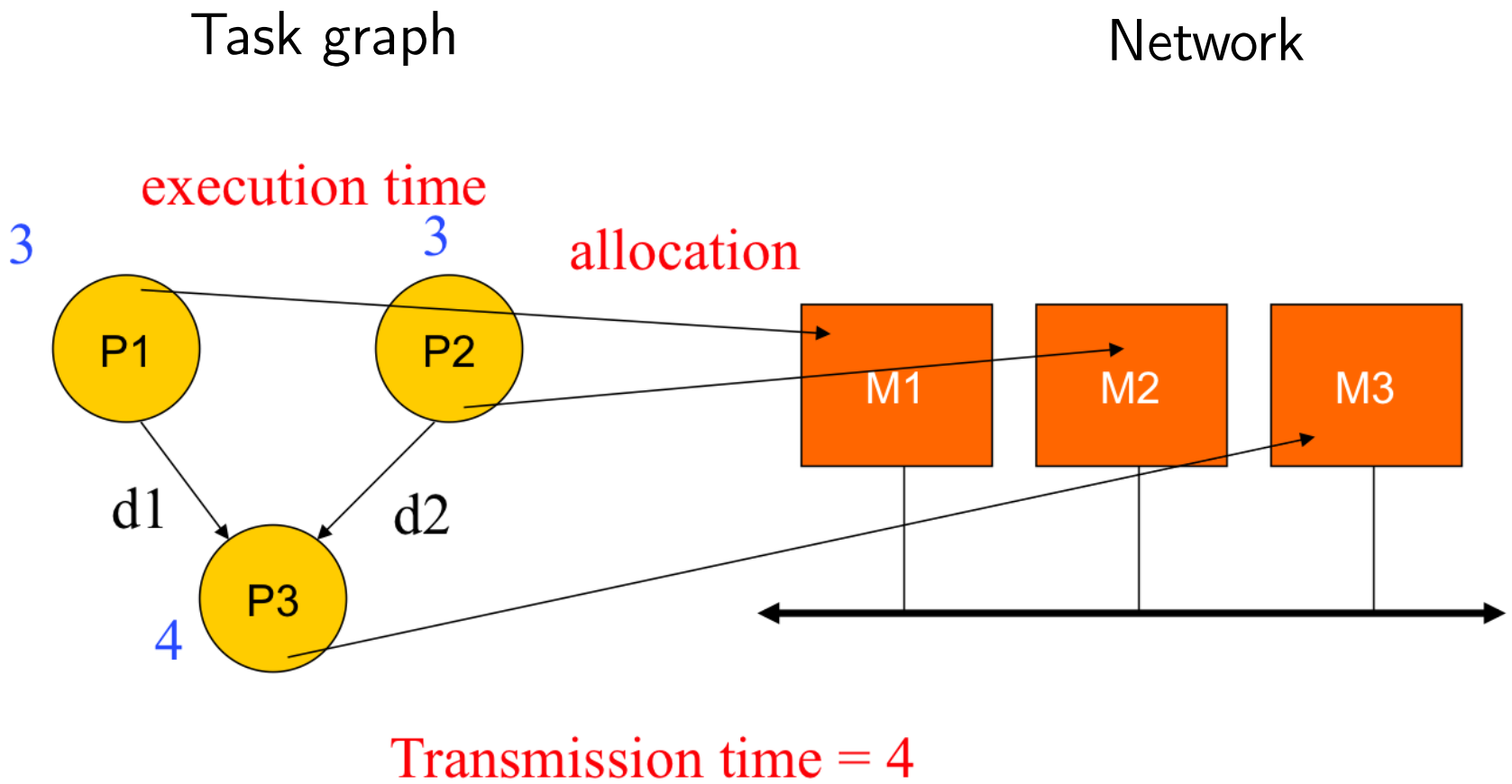
Second Design

Allocate

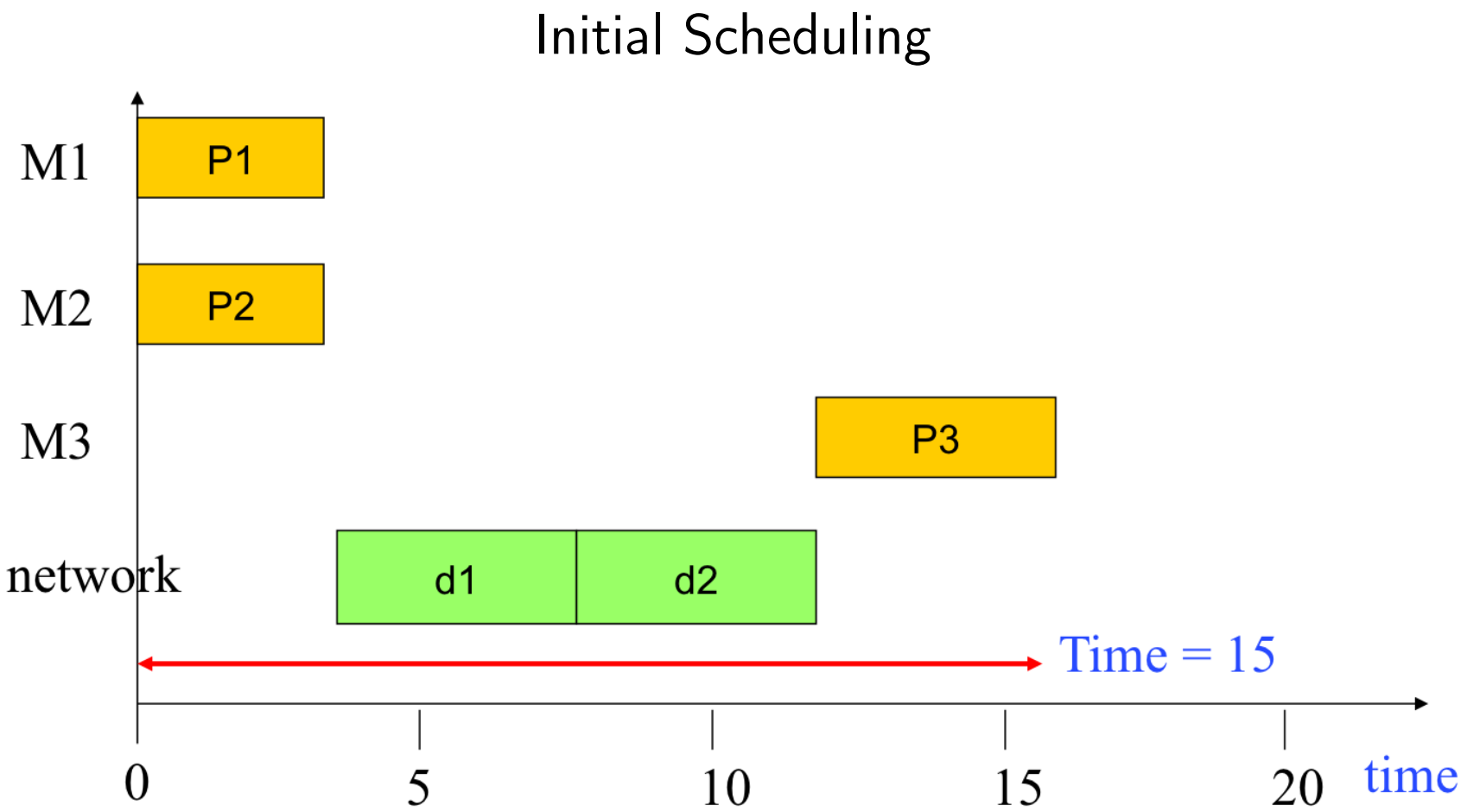
- P1 -> M1
- P2, P3 -> M2



Example: Reducing Delay (1/4)



Example: Reducing Delay (2/4)

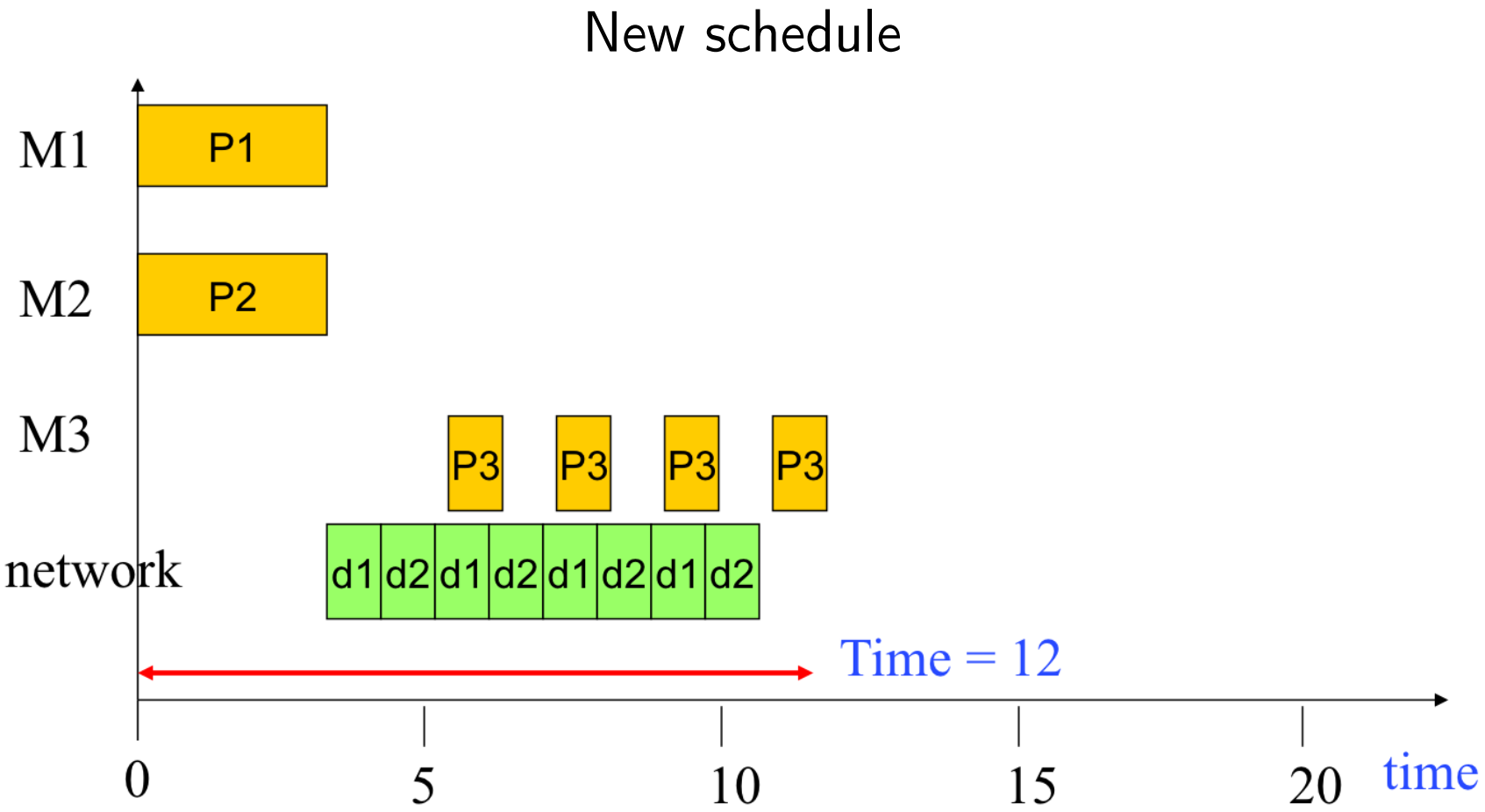


Example: Reducing Delay (3/4)

New Design: Modify P3

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

Example: Reducing Delay (4/4)



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

Outline

- Hardware Accelerators
- CPUs and Accelerators
- Accelerated System Design
- Partitioning
- Scheduling and Allocation
- System Integration and Debugging

References

Computers as Components: Principles of Embedded Computing System Design , Marylin Wolf. Morgan Kaufman. Ch 7.

Next Class

- System design