# ECE 587 – Hardware/Software Co-Design
# Lecture 07 Process-Based Models I

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

Feburary 2, 2026

# Reading Assignment

- This lecture: 3.1.2
- Next lecture: 3.1.2, 3.2

# Outline

Process-Based Models

Kahn Process Network (KPN)

Actor

# Process-Based Models

- A set of concurrent processes
  - There is no guarantee on the speed of the processes. Some may run very fast, while some may simply stop there for no apparent reason.
- Processes may communicate with each other to exchange data and establish dependencies among behaviors.
  - Semantics of communication differs for different process-based models.
  - If we need to distinguish to exchange data and to establish dependencies, we call the former communication and the later synchronization.

# Specifying and Implementing Processes

- Specify each process using state-based models.
  - As FSMs or sequential programs.
  - These processes run parallelly and independently.
- Implementing processes means to schedule them on processors (CPU) or processing elements (PE)
  - Scheduling a single process on a single CPU/PE is straight-forward.
  - OS usually provides support to schedule multiple processes on a single CPU/PE.
  - Otherwise, we need to provide our own scheduling, which will be introduced later.

# Specifying and Implementing Communications

- Two typical communication models
  - Message passing
  - Shared memory
- Communication via message passing
  - Synchronization is implied by the message.
  - Synchronous/rendezvous(request-reply) style
  - Asynchronous/queue-based style
- Communication via shared memory
  - Data (state) are shared – synchronization is the critical issue.
  - Highly depend on the availablity of synchronization primitives: e.g. mutex and semaphore as supported by OS through Inter-Process Communication (IPC) mechanisms, or memory barriers available from processors directly.
- Which one is more intuitive? Which one is more close to actual hardware?

# Deadlocks

- Process A holds resource a while waiting for resource b.
- Process B holds resource b while waiting for resource a.
- The system deadlocks.
  - No process can proceed and make progress.
- Deadlocks arise if there is a circular dependency on who is owning the resource and who is requesting the resource.
- Deadlocks can be avoided by either statically preventing such dependency to happen at design time or dynamically breaking them at runtime.
  - Still quite challenging for real systems.

# Deterministic Behavior

- A deterministic model will alway generate the same output given the same input.
    - Input/output behavior does not depend on the performance of processes and communication channels.
    - Determinism makes debugging and validation easier.
    - Otherwise, it is difficult to tell whether the model is correct.
- Additional determinism may be introduced to the model to specify the order the processes are executed.
    - For most systems, this may lead to overspecification, impacting the performance of the implementations, since otherwise the non-deterministic behavior may be exploited for better performance.

# Considerations for Process-Based Models

- How processes and communications are modeled?
- Can we prevent deadlocks?
- What kind of determinism is guaranteed?
- Let's focus on theoretical models with message passing communications first, and then discuss practical models with shared memory communications.

# Outline

Process-Based Models

Kahn Process Network (KPN)

Actor

# Kahn Process Network (KPN)

- Processes are represented by nodes.
  - As sequential programs.
- Communications are represented by (directed) arcs.
  - Usually known as *channel*s
  - Unidirectional
  - Point to point
- Communication via message passing
  - Unit of data/message: a *token*
  - Assume each arc has an unbounded first-in-first-out (FIFO) queue to hold unprocessed tokens.

# KPN Example

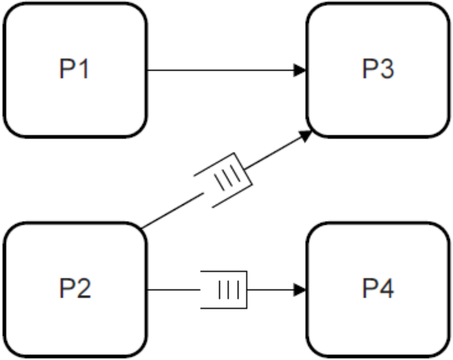

*FIGURE 3.1* Kahn Process Network (KPN) example

(Gajski et al., 2009)

# Interactions between Processes and Channels

- A process sending a token to a channel will never block.
  - Since the FIFO queue will take the token immediately.
- A process always blocks on reading a channel.
  - The process waits until a token is available and will retrieve the token immediately.
- The behavior of a process is deterministic.
  - Assume that the process cannot peek into a channel.
  - The process must decide which channel to read next and then block on that channel until a token arrives.

# Modeling System Inputs and Outputs

- Inputs can be modeled as processes without incoming arc.
  - So they simply generate tokens to be further processed.
- Outputs can be modeled as processes without outgoing arc.
  - So they will remove tokens from the system.

# Deadlocks and Determinism

- Certain KPNs may have deadlocks.
  - There is generally no way to decide if a KPN will have a deadlock or not.
- KPNs are deterministic.
  - The deterministic behavior of a process leads to the deterministic behavior of the system since it is independent of when a token arrives.

# Implementation Considerations

- Unbounded queue cannot be implemented.
- Sizes must be imposed on all queues.
    - And we want the sizes to be as small as possible to save memory for queues.
    - However, smaller sizes may lead to artificial deadlocks.
- Scheduling of processes may impact queue sizes.
- Determinism makes it possible for some algorithms to determine a set of queues with small sizes for certain schedules.
    - However, it is not always possible to find such sizes/schedule.

# Outline

Process-Based Models

Kahn Process Network (KPN)

Actor

# Motivation

- KPN cannot provide more guarantees than deterministic behavior because it is difficult to analyze those complex sequential programs.
  - The complexity generally comes from that they need to consume tokens that arrive without a fixed schedule.
- A typical sandwich structure used in those processes.
  - Receive some tokens.
  - Do some computations.
  - Send some tokens.
- We have better ways to model such structure!

# Actor

- Processes as actors.
  - Essentially FSMDs.
  - Each state transition is triggered by an incoming token.
  - Each state transition is usually modeled by a sequential program.
  - Output corresponds to outgoing tokens.
- Deadlocks and determinism depend on details of state transition.
  - E.g. how to consume and produce tokens.
- Widely adopted by programming languages and frameworks that need to support process-based models.

# Summary

- Processes and communications are critical for process-based models.
- KPN: sequential programs+queues
- Actor: FSMD+queues