# Modelos de computación
# Lección 8

Prof.Ing. Jeferson González G, M.Sc

CE-5303 Sistemas Embebidos

*Área de Ingeniería en Computadores*

*Instituto Tecnológico de Costa Rica*

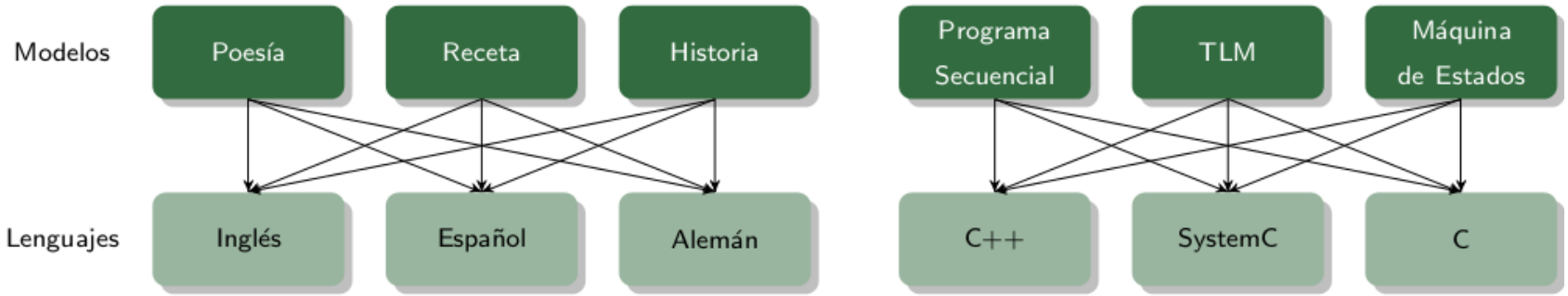
# Modelo de Computación - MoC

Un **Modelo de Computación** (MoC) es una manera abstracta y conceptual de describir el comportamiento de un sistema.

**Modelo es especificación**. Definen dos aspectos:

- Componentes: elementos y operaciones
- Comunicación entre componentes

# Modelos vs Lenguajes



# Modelos de Computación

Existen principalmente 3 tipos de modelos de computación:

- Programación secuencial (Von Neumann)
- Modelos basados en proceso
- Modelos basados en estados

# Programación secuencial

Modelo clásico de descripción de sistemas de computación (programas).

- Flujo secuencial de instrucciones no es adecuado para modelar hardware concurrente
- En general no considera requerimientos temporales (timeout/deadlocks)

# Programación secuencial II - Threads

## Threads

"The lack of timing in the core abstraction is a flaw, from the perspective of embedded software, and threads as a **concurrency model are a poor match for embedded systems**... Nontrivial software written with threads, semaphores, and mutexes is incomprehensible to humans." Edward Lee, 2005.

Desventajas:

- mecanismos de sincronización - afecta rendimiento
- condiciones de carrera
- deadlocks

**Conclusión**: Para describir en alto nivel, deben evitarse modelos basados en programación secuencial.

# Requisitos de un MoC para SM

- **Jerarquía** (modularidad)
- **Concurrencia**: Paralelismo
- **Ejecutabilidad**
- **Comunicación**
- **Tiempo**

# Modelos basados en procesos

Corresponden a un conjunto de procesos concurrentes, que internamente, poseen un modelo gramático secuencial.

- Set de bloques independientes de código que ejecutan en paralelo.
- Comunicación entre bloques se da por medio de tokens o mensajes.
- Determinísticos: Una misma entrada siempre va a producir una misma salida

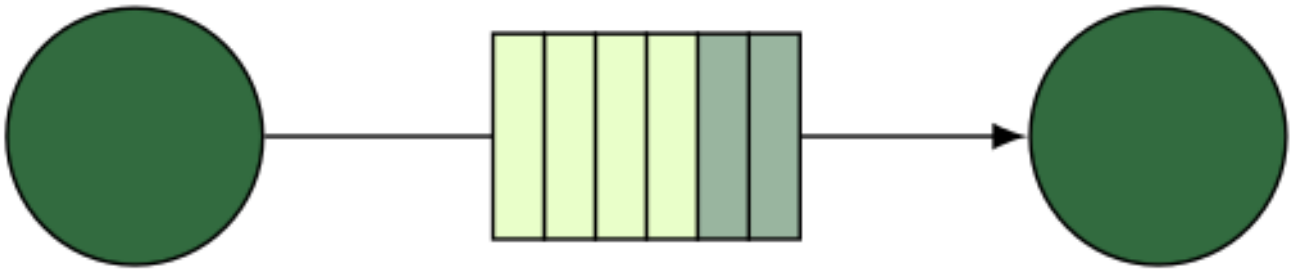
# Procesos: Flujo de datos

## Procesos

- Proceso activo: No requiere calendarización
- Proceso pasivo: Requiere calendarización

## Comunicación

- Comunicación mediante FIFO
- FIFOs infinitos (teóricamente)
- Lecturas destructivas

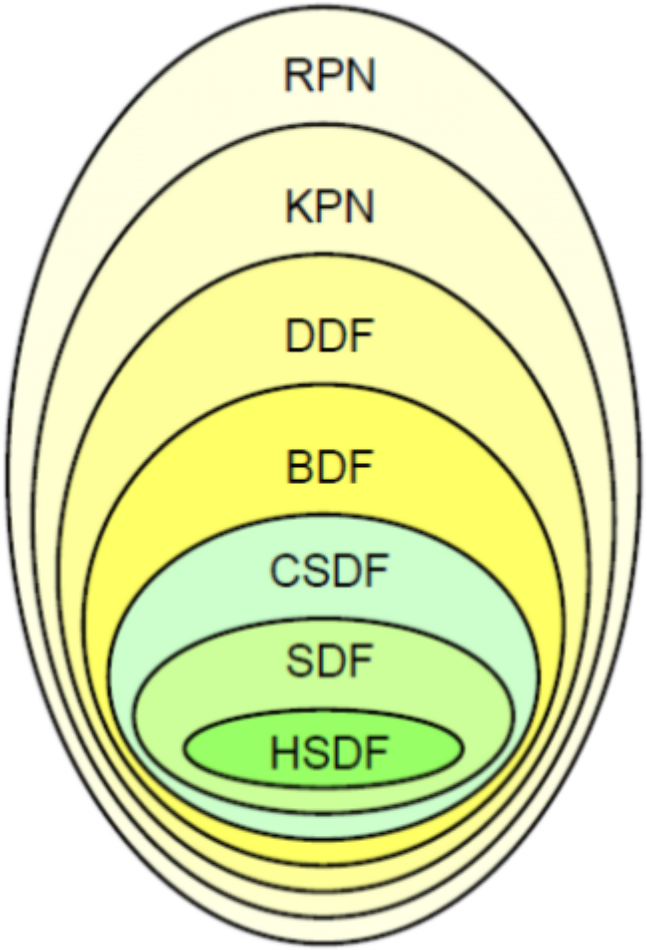


# Aplicaciones

Útil para describir sistemas que involucran flujo de datos:

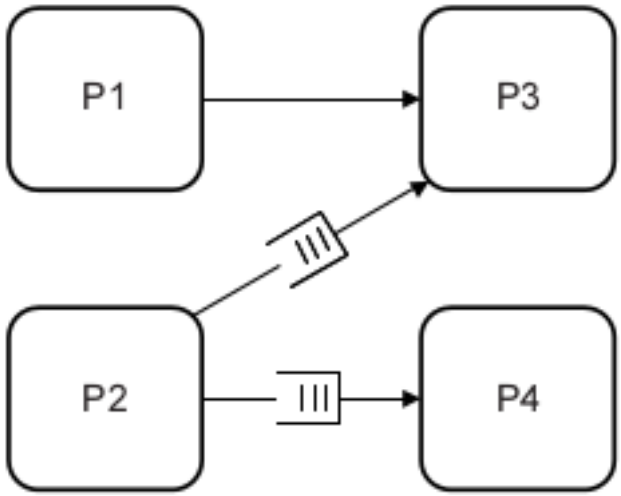
- Streaming: audio/video
- Filtado digital de señales
- Procesamiento de imágenes, etc

# Modelos basados en procesos



# Redes de procesos de Kahn

- Red de procesos **activos** concurrentes, comunicados a través de **FIFOs** unidireccionales (infinitos).
  - Lectura del FIFO es bloqueante.
  - Escritura del FIFO es no bloqueante.
- Cada FIFO tiene solo un productor y un consumidor.

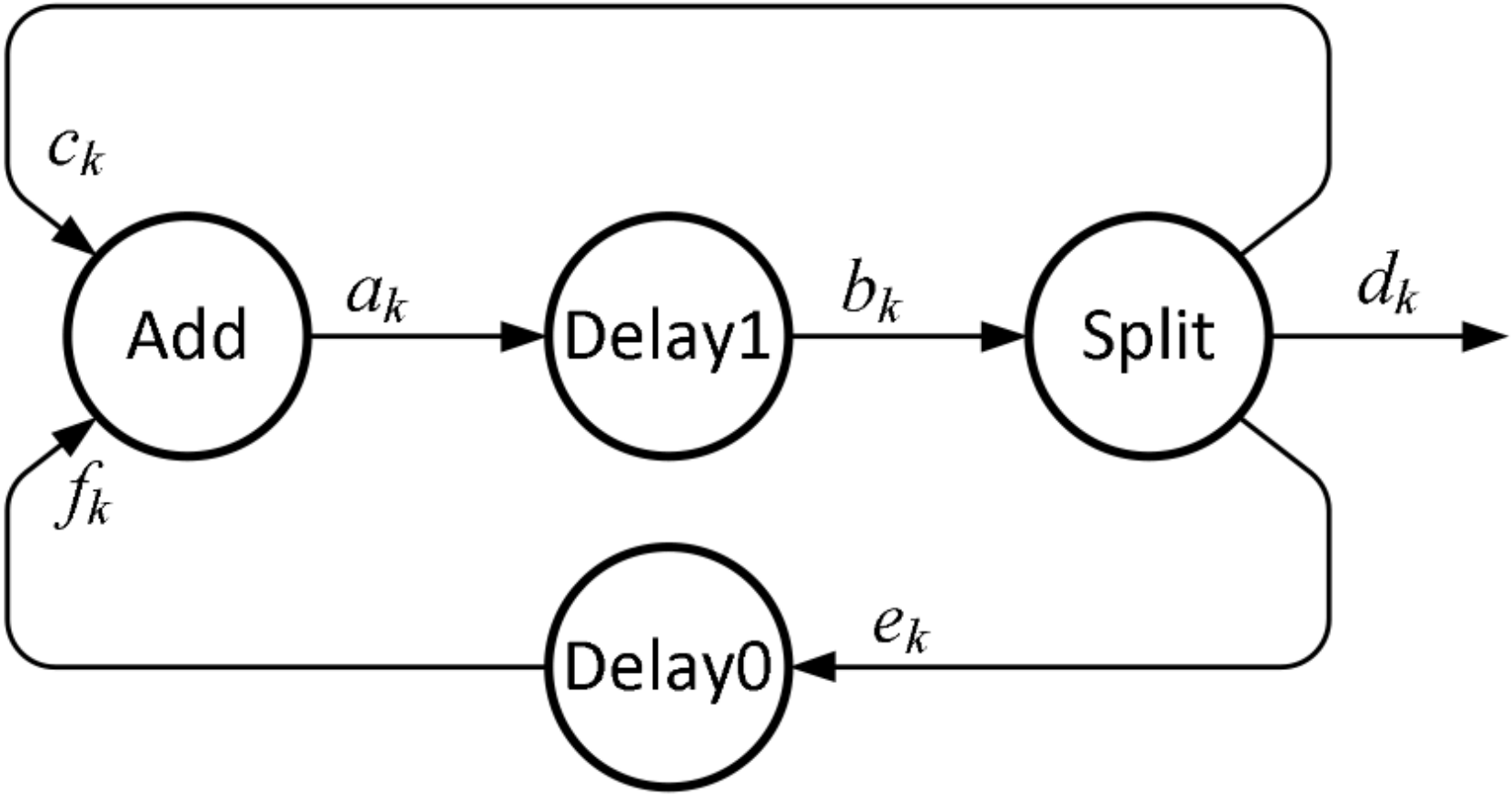


# Procesos en KPN

La implementación de procesos (internamente) en KPN se realiza en programación secuencial:

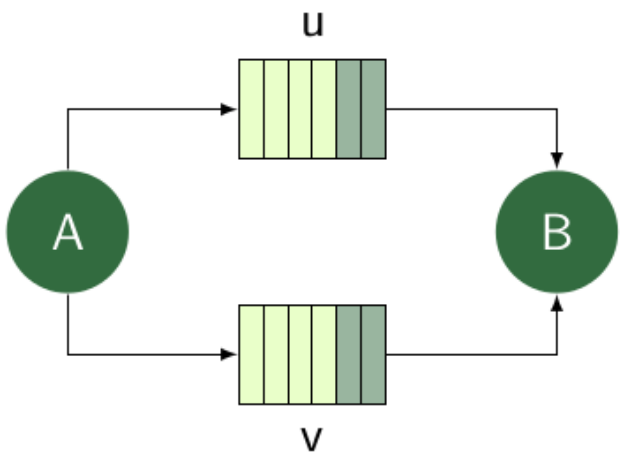
```
process Add(in x, in  y, out z) {
      int a;
      int b;
      int c;
      while (1) {
          a = x.read(); //bloqueante
          b = y.read(); //bloqueante
          c = a + b;
          z. write(b);
      }
}
```

# KPN ejemplo



# Deadlock

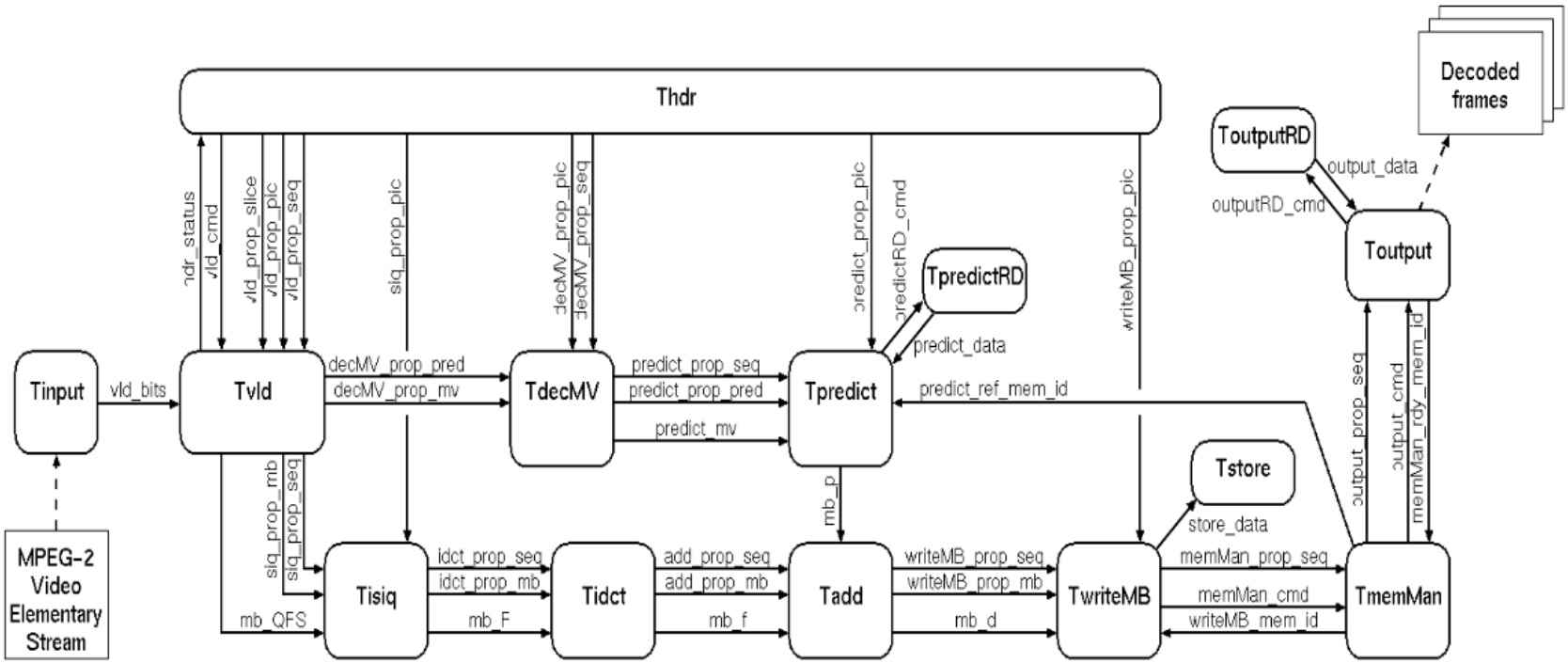
Estancamiento producido por utilizar buffers reales en KPNs



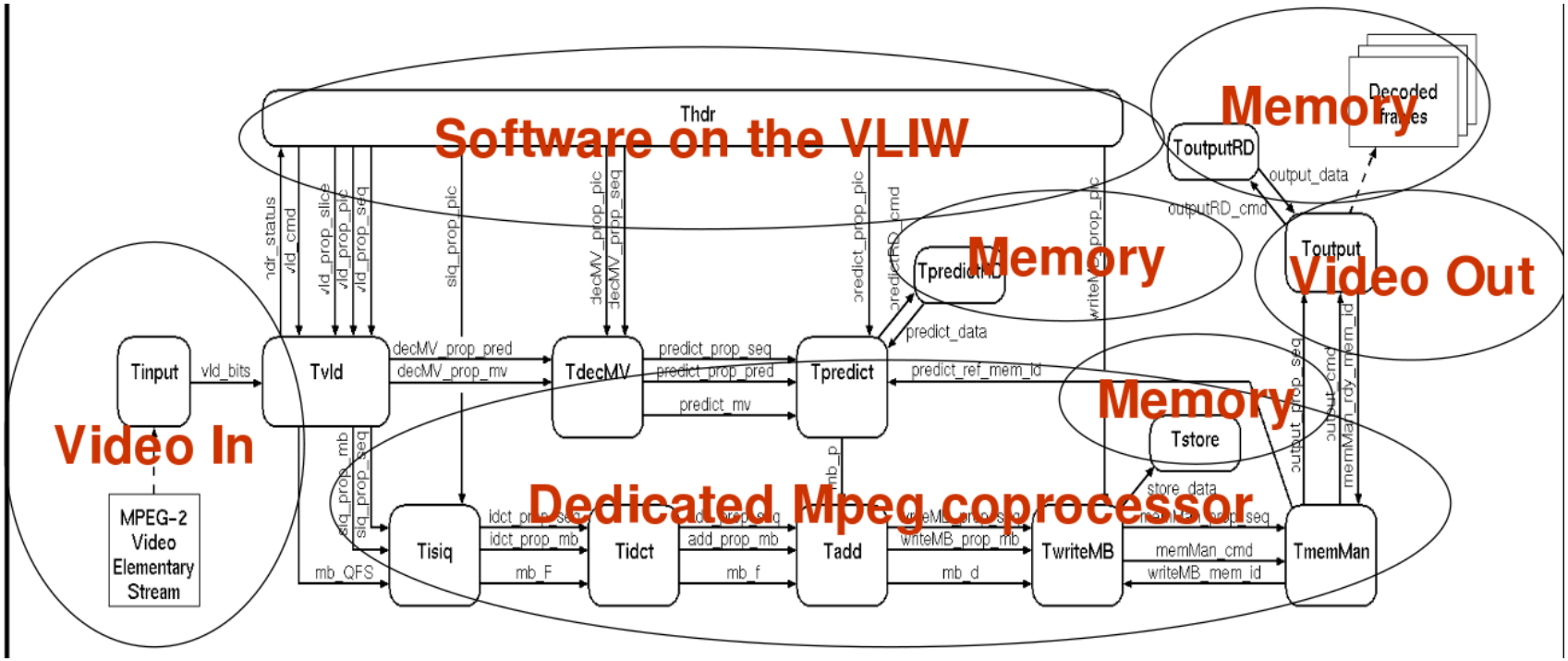

```
proceso A (out int u, out int v) {
  while(1) {
    send(1, u);
    send(1, u);
    send(1, v);
  }
}
```

```
proceso B (in int u, in int v) {
  while(1) {
    wait(v);
    wait(u);
    wait(u);
  }
}
```

# Ejemplo: MPEG video decoder



# Ejemplo: Mapeo (Binding)



# KPN

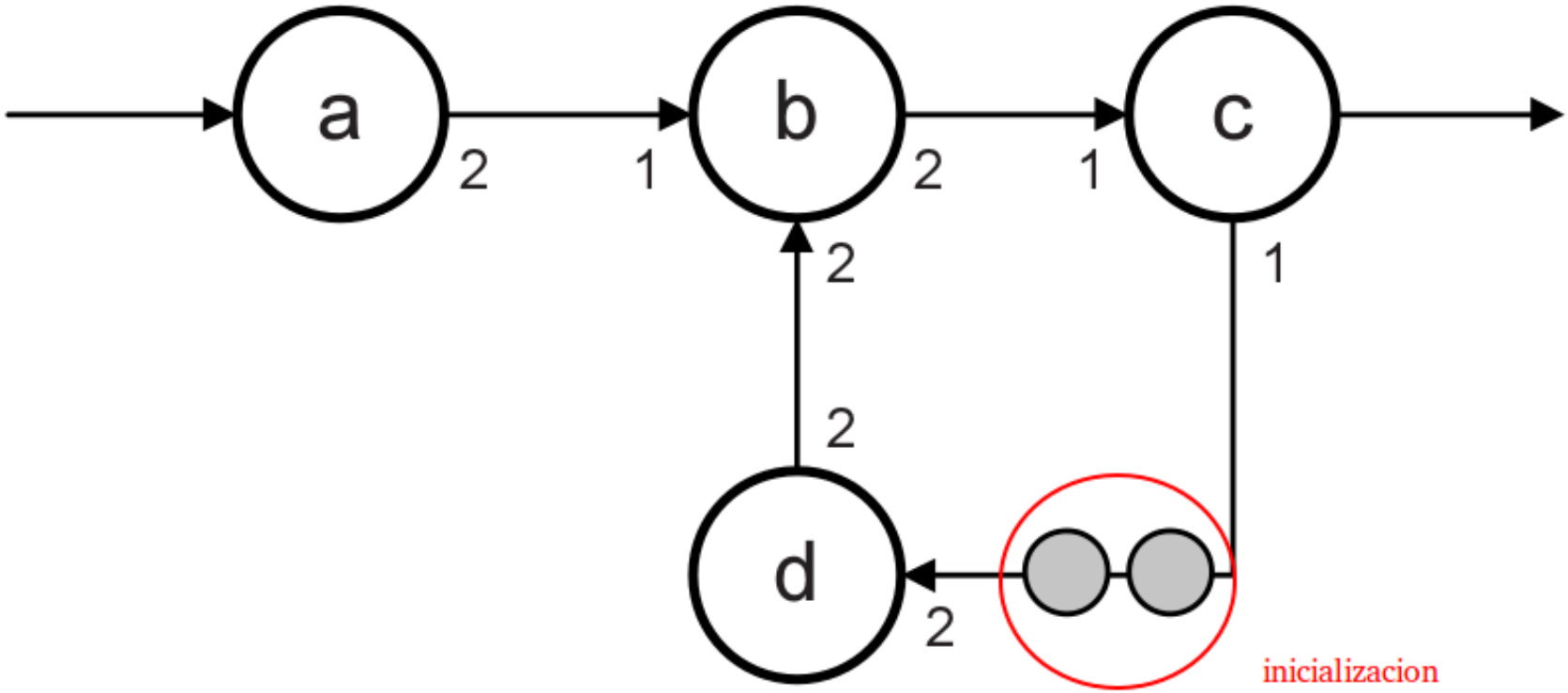
Desventaja **KNP:** Difícil de calendarizar en escenarios reales (FIFOs finitos) **Solución:** Modelo con calendarización pasiva.

# Synchronous Data Flow

Modelo basado en red de procesos pasivos cuncurrentes, comunicados a través de FIFOs.

- **Proceso pasivo**: Requiere un esquema de calendarización.
- Calendarización estática.
- Modelo de ejecución basado en **disparos**:
  - Reglas de disparo para número y condiciones de disparo.

# SDF



# Reglas de disparo

- Procesos se habilitan al tener tokens en FIFO de entrada.
- Procesos son disparados removiendo N cantidad de tokens de sus FIFOs de entrada y produciendo M tokens a los de salida.
- **Iteración**: Secuencia de disparos que lleva el SDF al estado inicial.

# Calendarización en SDF

Pasos para la calendarización:

- Determinar ecuaciones de balanceo por FIFOs (tokens producidos - tokens consumidos = 0)
- Establecer tasas de ejecución relativa (resolver ecuaciones de balanceo)
- Determinar una calendarización periódica (orden disparos en cada proceso)

# Ejemplo: Paso 1

**Ecuaciones de balanceo**

$b - 3c = 0$

$2c - a = 0$

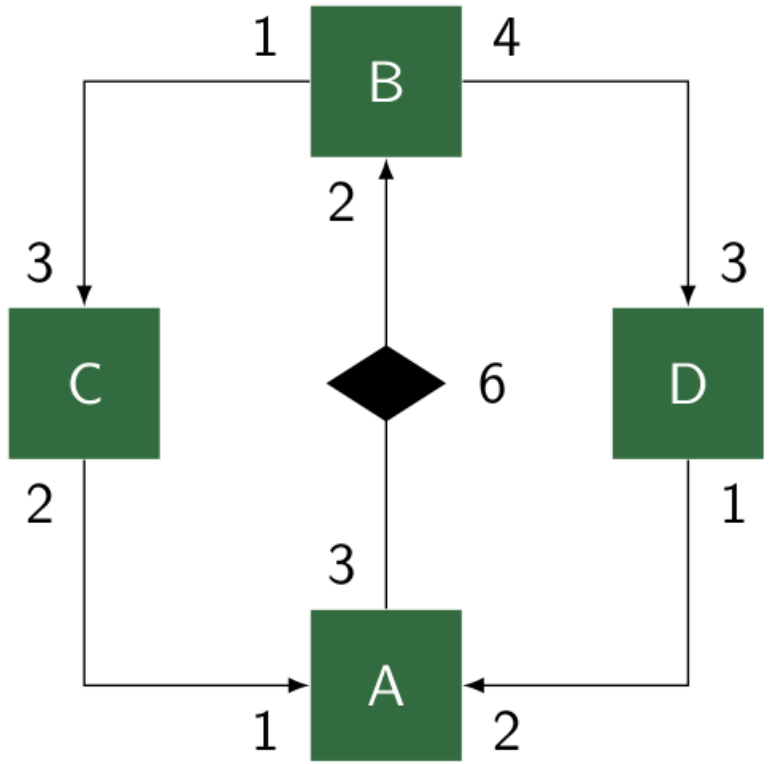
$d - 2a = 0$

$4b - 3d = 0$

$3a - 2b = 0$

**Matriz (Gauss-Jordan)**

$$\begin{bmatrix} 3 & -2 & 0 & 0 \\ 0 & 4 & 0 & -3 \\ 0 & 1 & -3 & 0 \\ -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = 0$$



# Ejemplo: Paso 2

Un grafo SDF con N procesos tiene una calendariación periódica si la matriz topológica tiene rango N-1.
**Clasificación**:

- Sistema consistente: Matriz tiene rango N-1. Existen múltiples soluciones. **Deseable: menor entero positivo**
- Sistemas inconsistentes: Única solución trivial (todo es cero)
- Sistemas desconectados: Tasa de algunos procesos está indefinida.

# Ejemplo: Paso 2

$$\begin{bmatrix} 3 & -2 & 0 & 0 \\ 0 & 4 & 0 & -3 \\ 0 & 1 & -3 & 0 \\ -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = 0 \quad \Rightarrow \quad \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 4 \end{bmatrix}$$

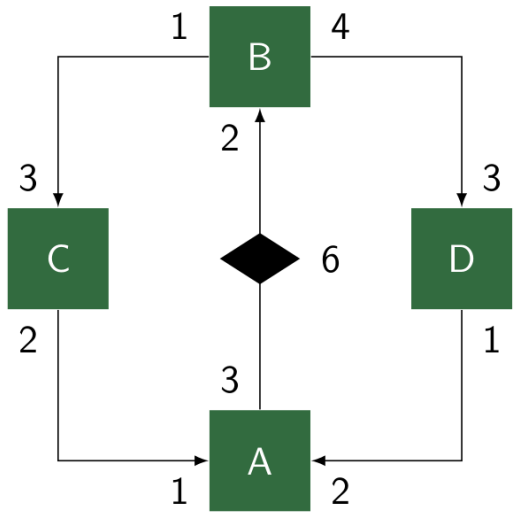
# Ejemplo: Paso 3

**Calendarizaciones posibles:**

- BBBCDDDDAA*
- BDBDBCADDA*
- BBDDBDDCAA*
- ...

**Reducción de tamaños de buffers

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 4 \end{bmatrix}$$



# Máquinas de estados finitos

Una máquina de estados se define por:

$$< S, I, O, f, h >$$

- **S**: Conjunto de estados
- **I**: Conjunto de entradas
- **O**: Conjunto de salidas
- **f**: Función de próximo estado $f : S \times I \rightarrow S$
- **h**: Función de salida
  - Mealy $h : S \times I \rightarrow O$
  - Moore $h : S \rightarrow O$

# FSM

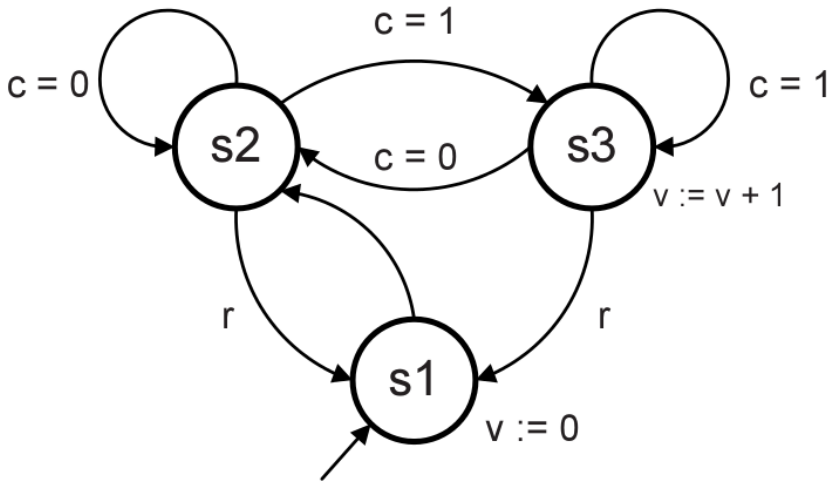
- Máquinas de estados finitos pueden volverse muy grandes (cantidad de estados) para sistemas complejos.
- Se debe incluir in estado para cada condición posible.
- Para reducir la complejidad se utilizan **Máquinas de estados finitas con datos** (FSMD)

# Máquina de estados finitos con datos

Una máquina de estados finita con datos se define por:

$$< S, I, O, V, f, h >$$

- Extiende la definición de FSM con un conjunto de variables V. De forma que:
  - $f : S \times I \times V \rightarrow S \times V$
  - $h : S \times I \times V \rightarrow O$



# FSMD

- Utilizados para representar implementaciones de Hardware de Procesadores en RTL.
- Variables y expresiones (v=v+1..etc) representan las operaciones en datapath.
- Estados y transiciones representan el controlador del procesador.

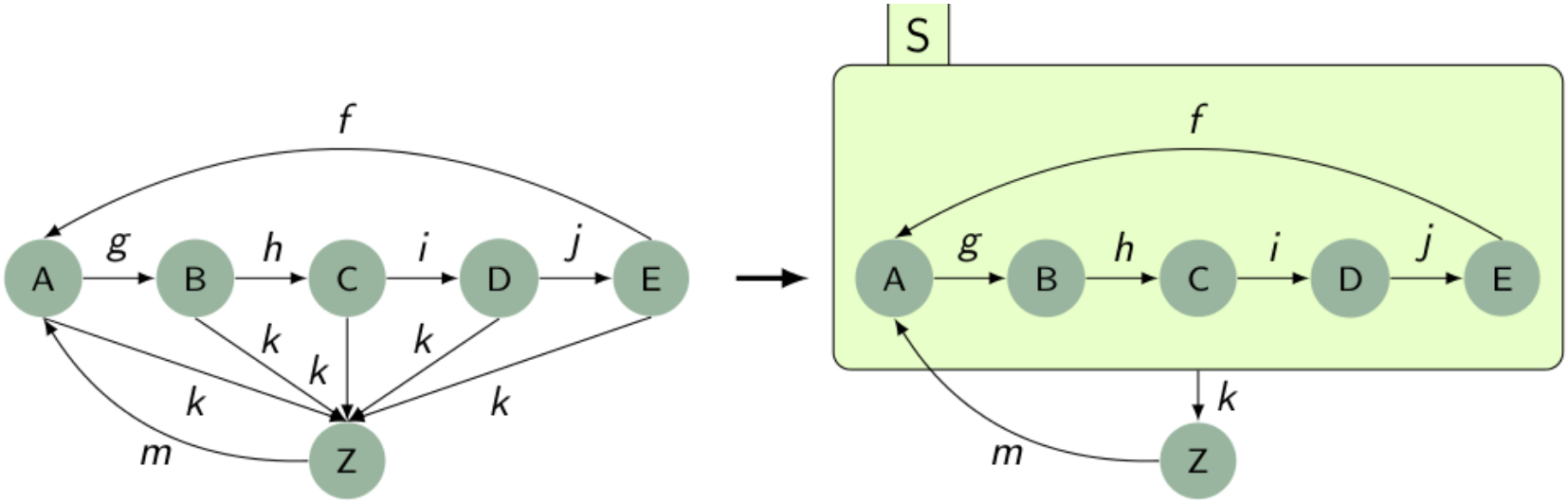
# Máquinas de estados con jerarquía y concurrencia (HCFSM)

En este modelo, se adiciona mecanismo de jerarquía y concurrencia:

- **Jerarquía**: Super estados $\rightarrow$ incluyen maquinas de estados (con datos) en su interior.
- **Concurrencia**: Estados ejecutándose simultáneamente.

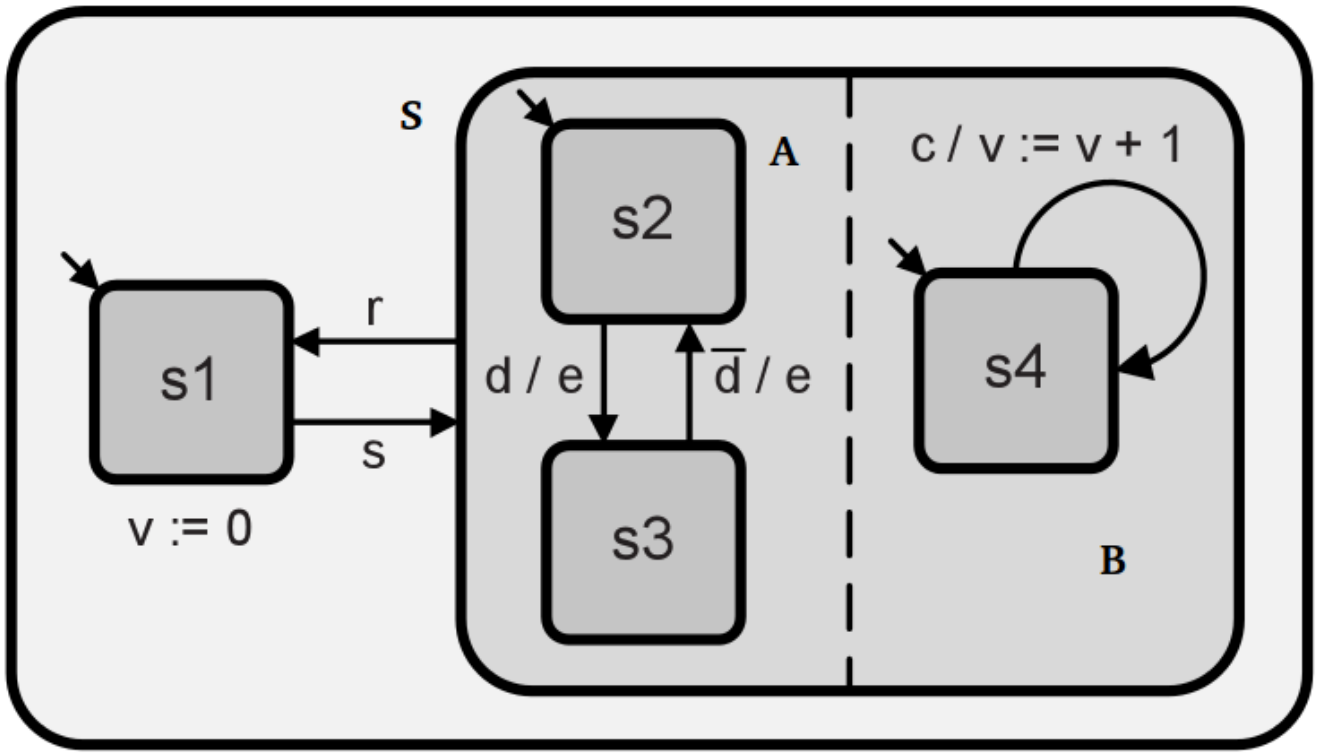
Conocidos también como **StateCharts**.

# Ejemplo Super estados




- Si **m** se da: La máquina entrará al super estado S, que representa por sí una máquina de estados.
- Si **k** se da: Sin importar en qué estado se encuentre la máquina dentro de S, se irá al estado z.

# Ejemplo concurrencia



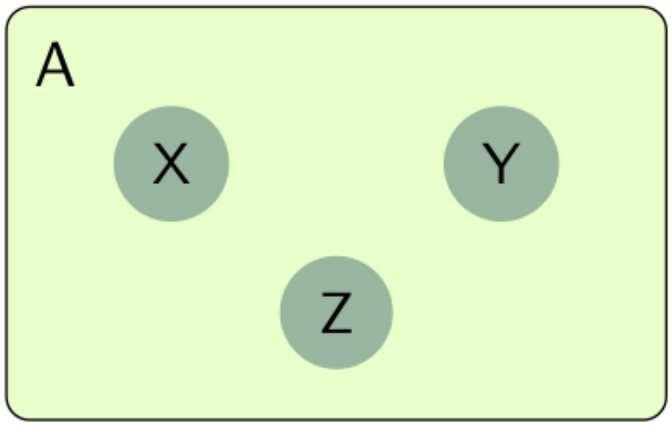

- Super estados S, A, B
- Estados concurrentes A, B

# Definiciones

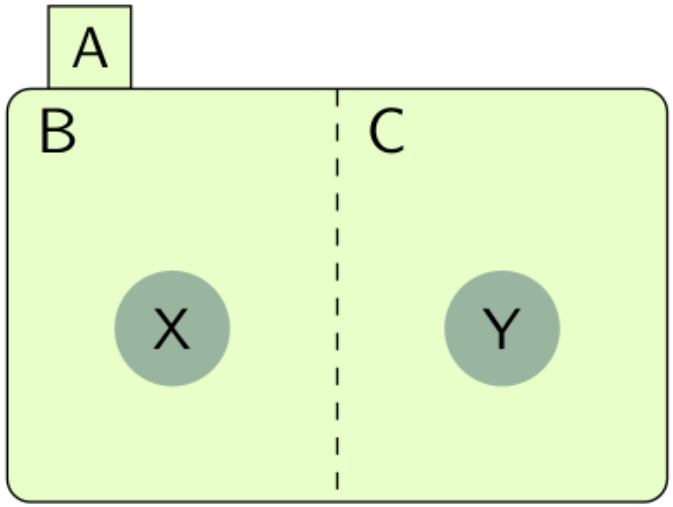
- **Estado activo**: Estados actuales de una máquina.
- **Estado básico**: Estados que no están compuestos de otros.
- **Super estados**: Estados que contienen otros estados.
- **Super estado Or**: Dentro del super estado, solamente uno de los estados estará activo.
- **Super estado And**: Dentro del super estado, puede haber dos o más estados activos (concurrencia)

# Super estados

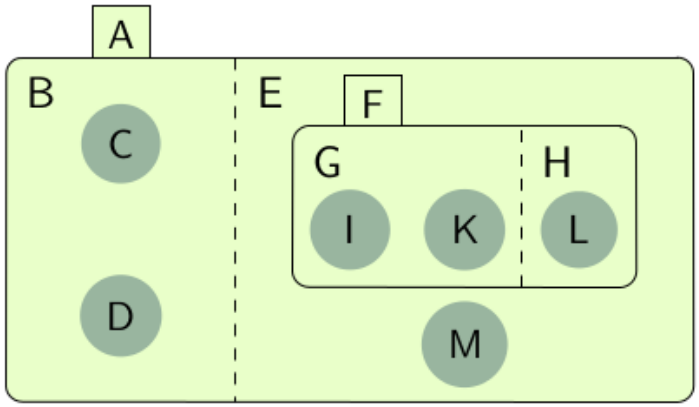
## Super Estado OR

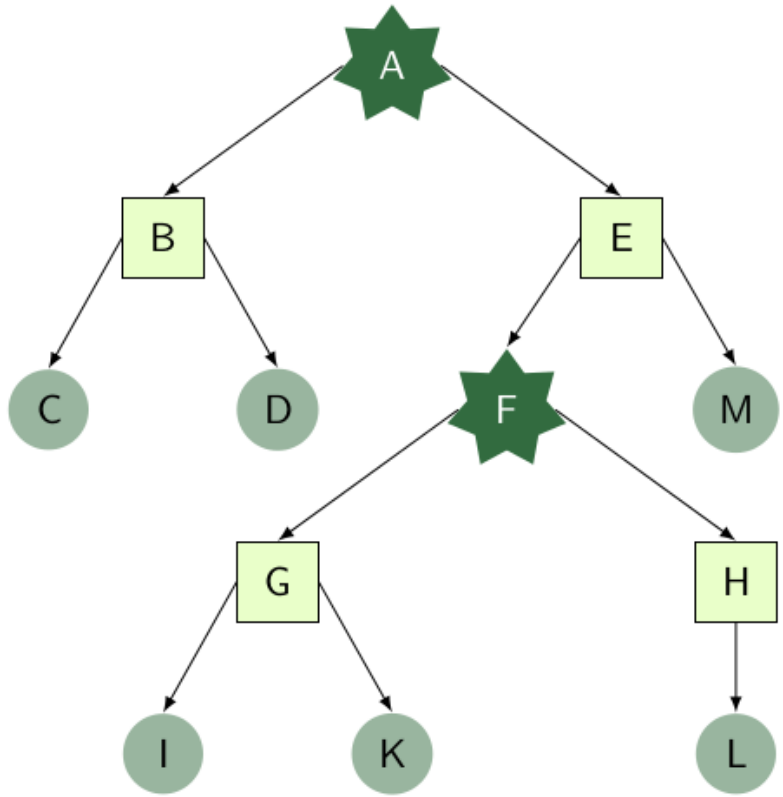



## Super Estado AND



# Representación en grafo de árbol Charts



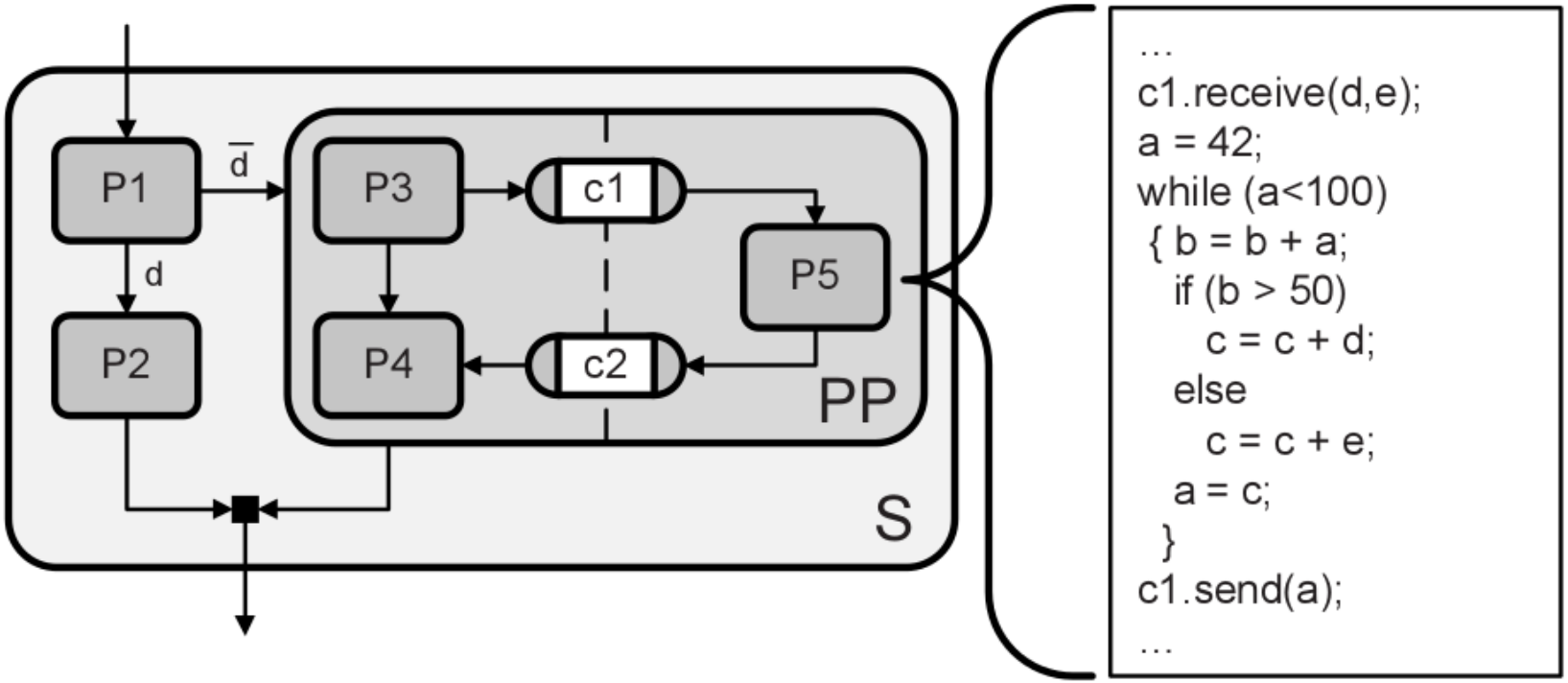



# Máquinas de estado de procesos

Modelo que combina redes de procesos con máquinas de estados (Statecharts).

- Cada estado puede componerse de uno o más estados en los que realiza un proceso secuencial, pero los estados pueden ejecutarse concurrentemente.
- Cada proceso tiene definido un inicio y un fin.
- Dos tipos de transiciones: Inmediata (señal/evento) y en terminación.
- Procesos paralelos corren asincrónicamente. Comunicación se da por variables compartidas, eventos y señales.
- Algunos lenguajes permiten encapsulamiento de comunicación, a través de canales.

# PSM


P1
d
P2
P3
P4
c1
c2
P5
PP
S
...
c1.receive(d,e);
a = 42;
while (a<100)
{ b = b + a;
if (b > 50)
c = c + d;
else
c = c + e;
a = c;
}
c1.send(a);
...

 Gajski, D.D., Abdi, S., Gerstlauer, A., Schirner, G (2009)
Embedded System Design - Modeling, Synthesis and Verification