

# ForSyDe: A Formal Framework for Heterogeneous Models of Computation

**Axel Jantsch**

**Royal Institute of Technology**

**Sweden**

**ForSyDe Team:**

**Ingo Sander, Hossein Niaki,**

**Axel Jantsch, Zhonghai Lu,**

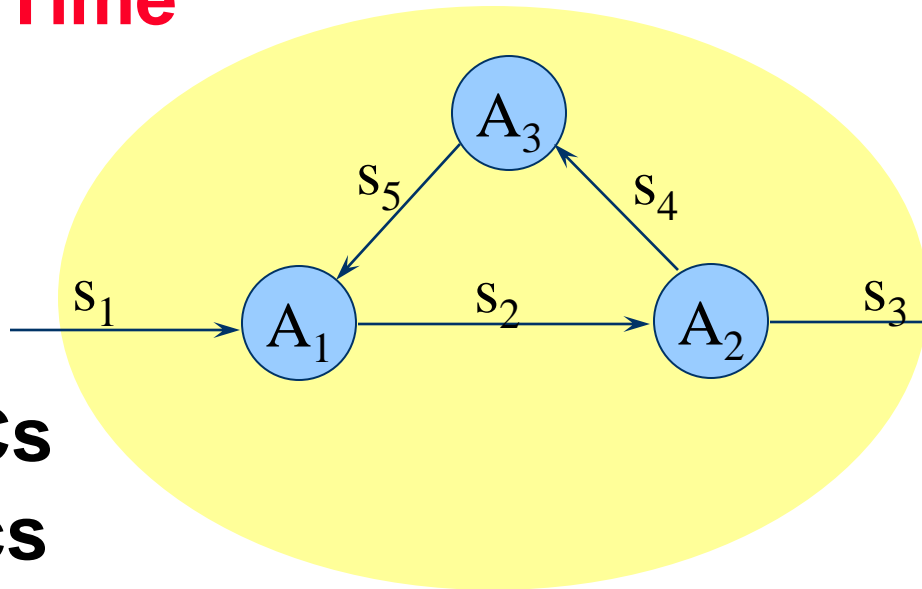
**Tarvo Raudvere, Jun Zhu, Alfonso Acosta**

# Overview

- **A denotational framework for MoCs**
  - **Semantic basis**
  - **Untimed MoC**
  - **Synchronous MoC**
  - **Discrete Time MoC**
  - **Continuous Time MoC**
- **Integration and Interaction of MoCs**
- **Parallel simulation**
- **Mapping onto parallel architectures**

# Models of Computation

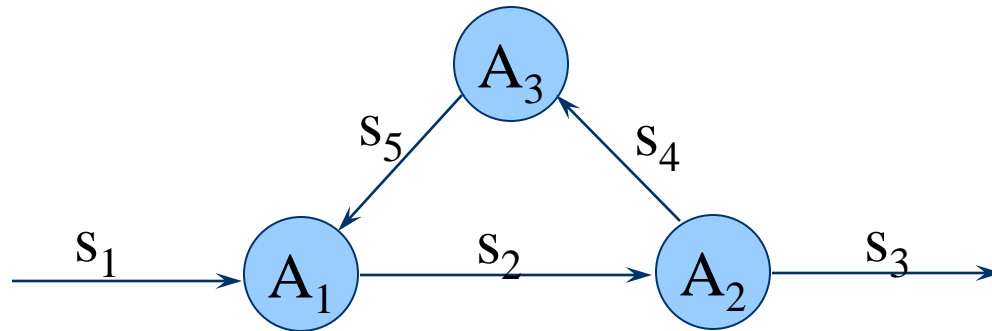
- **Concurrent process networks**
- **Formal definition of Communication**
- **Formal definition of Synchronization**
- **Formal definition of Time**
  - **Untimed**
  - **Synchronous Time**
  - **Discrete Time**
  - **Continuous Time**
- **Heterogeneous MoCs**
- **Execution mechanics**
- **Purpose of a model: simulation, synthesis, verification, performance analysis**



# ForSyDe Objectives

- **Model time explicitly**
- **Allow for time at all abstraction levels**
- **Allow to mix different time models**
- **Model process invocation explicitly**
- **Abstract execution mechanics**

# ForSyDe Features



## Processes

- Communicate through signals only;
- Functional
- State-full
- Blocking read
- Partition input and output signals
- Evaluate when required input is available

## Signals

- Sequences of events
- Preserve event order
- Have one writer and multiple readers
- Untimed MoC: Events are partially ordered
- Discrete Time MoCs: Signals carry timing information
- Continuous Time MoC: Signals are functions

# The ForSvDe Design Flow

## Ideal System Model

No resource limitation on

- Processors
- Communication bandwidth
- memory

## Implementation model

With finite resources

- Processors, HW blocks
- Reconfigurable resources
- Buffers
- Communication architecture
- Schedulers, arbiters

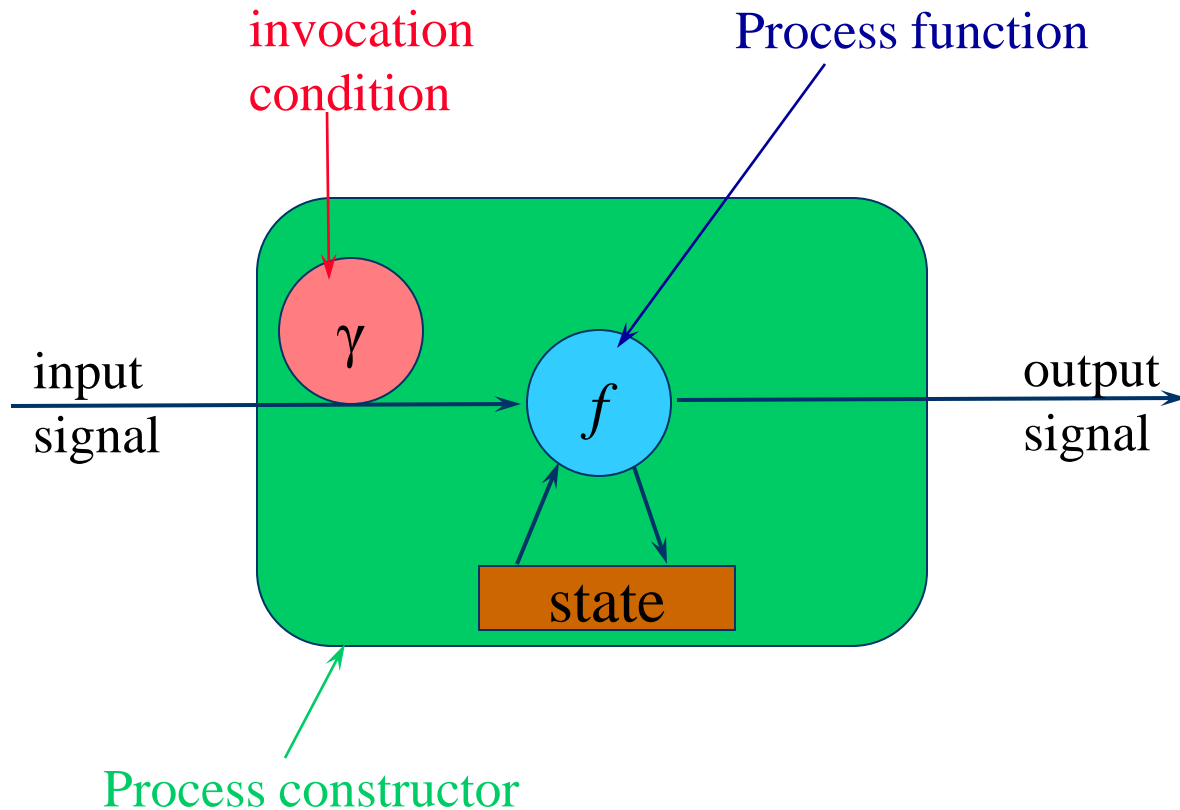
C program

VHDL design

SystemC model

# ForSyDe Process Constructors

Process = constructor + function + initialState + invocationCondition



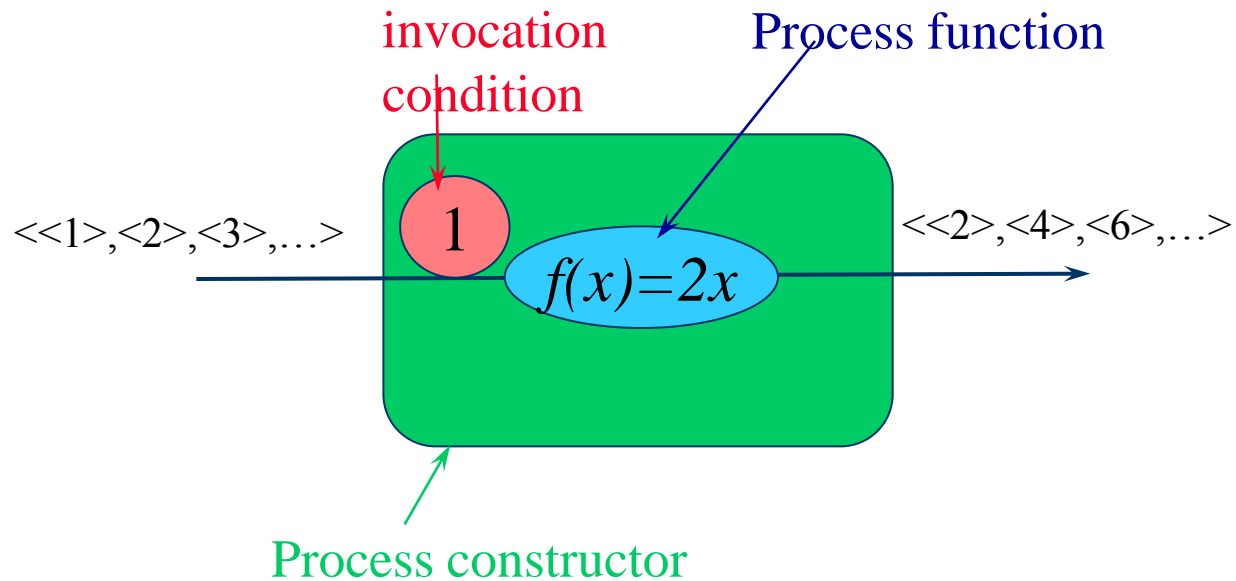
# The *combU* Process Constructor

$$\text{combU}(c, f) = p$$

where  $p(s) = s'$

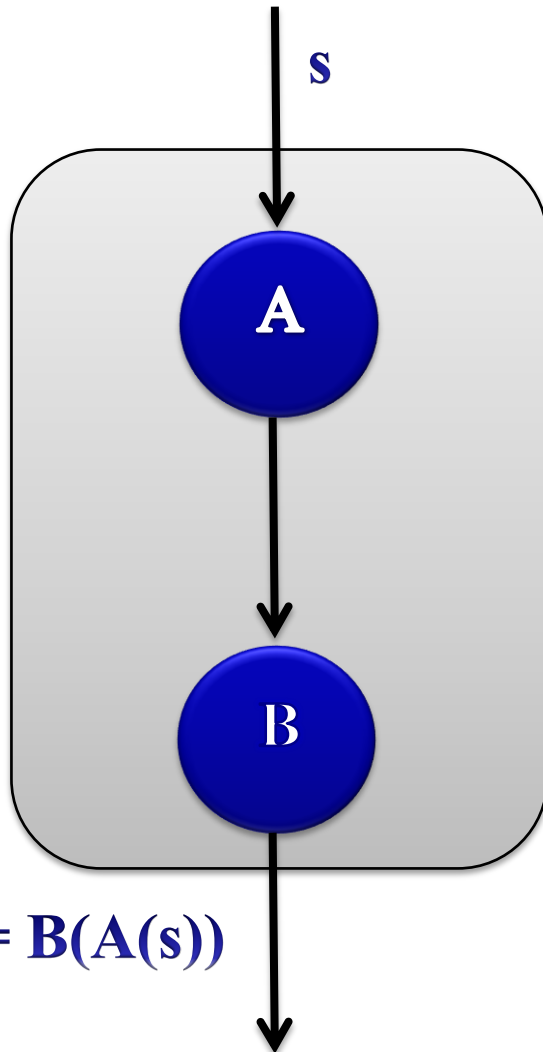
$$\text{combU}(c, f) = A$$

where  $c=1$   
 $f(x) = 2x$

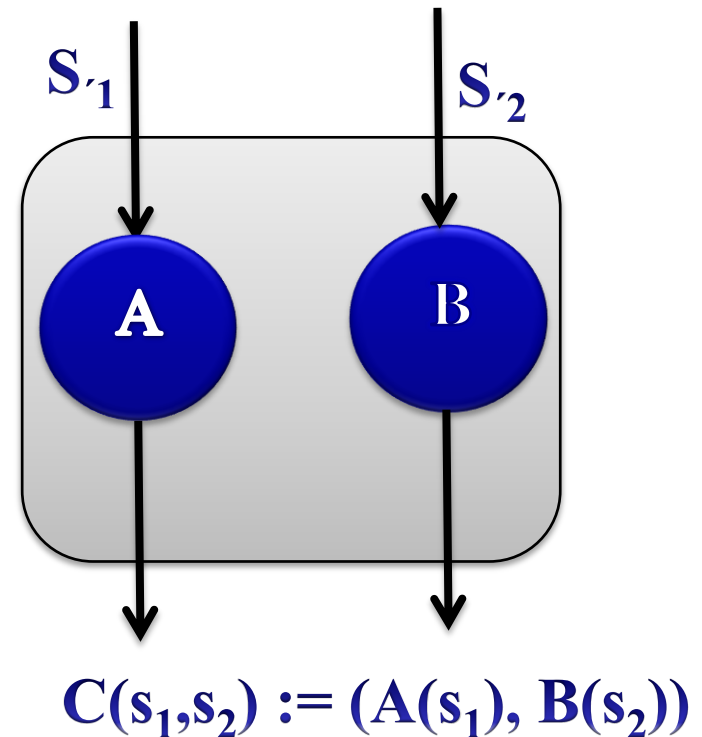


# Process Combinators

Sequential

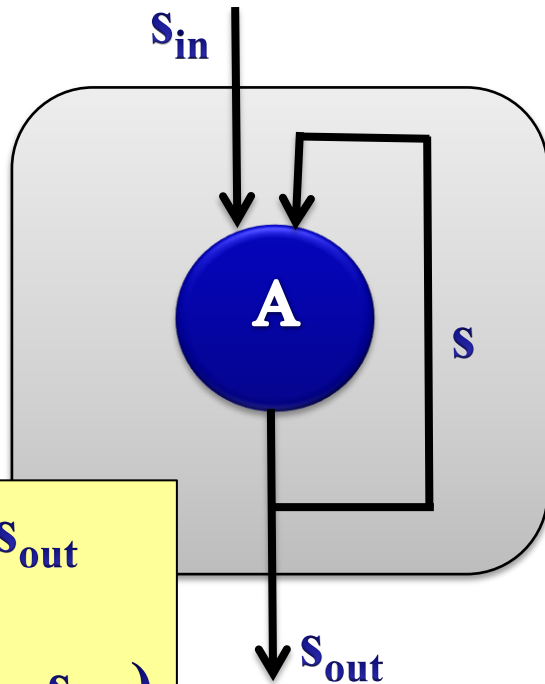


Sequential



# Process Combinators

## Feedback Semantics



$C(s_{in}) := s_{out}$   
where  
 $s_{out} = A(s_{in}, s_{out})$

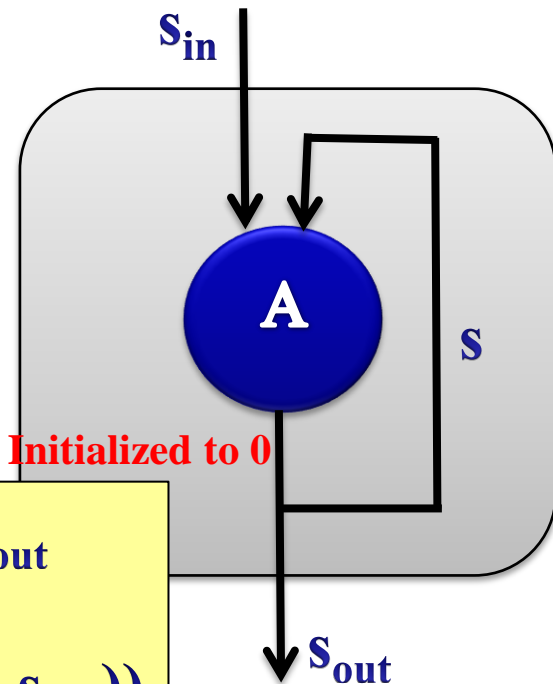
$f_A(x,y) = x+y$   
 $\rightarrow C \langle 1,2,3,4 \rangle = \langle \rangle$

$f_A(x,y) = \text{if } (y==3) \text{ then } 3 \text{ else } x$   
 $\rightarrow C \langle 1,2,3,4 \rangle = \langle \rangle$

Least fixed point over  
signal histories

# Process Combinators

## Feedback Semantics



$C(s_{in}) := s_{out}$   
where  
 $s_{out} = A(s_{in}, s_{out})$

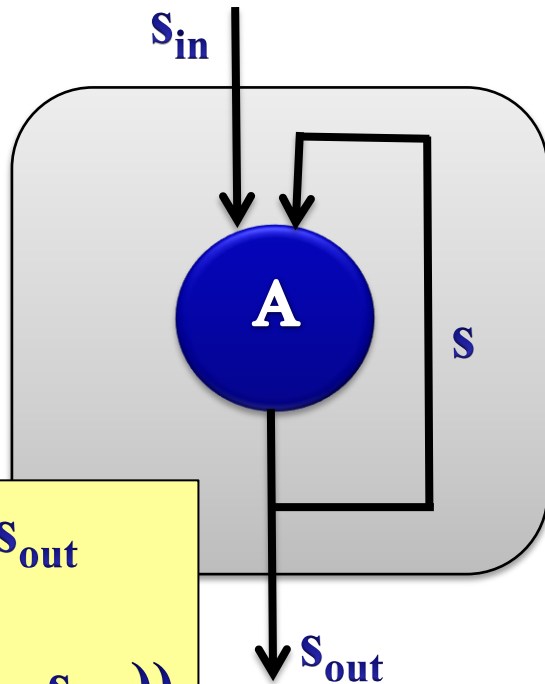
$$f_A(x, y) = x + y$$
$$\rightarrow C \langle 1, 2, 3, 4 \rangle = \langle 0, 1, 3, 6, 10 \rangle$$

$$f_A(x, y) = \text{if } (y == 3) \text{ then } 3 \text{ else } x$$
$$\rightarrow C \langle 1, 2, 3, 4 \rangle = \langle 0, 1, 2, 3, 3 \rangle$$

Least fixed point over  
signal histories

# Process Combinators

## Feedback Semantics



$C(s_{in}) := s_{out}$   
where  
 $s_{out} = A(s_{in}, s_{out})$

$$f_A(x, y) = x + y$$

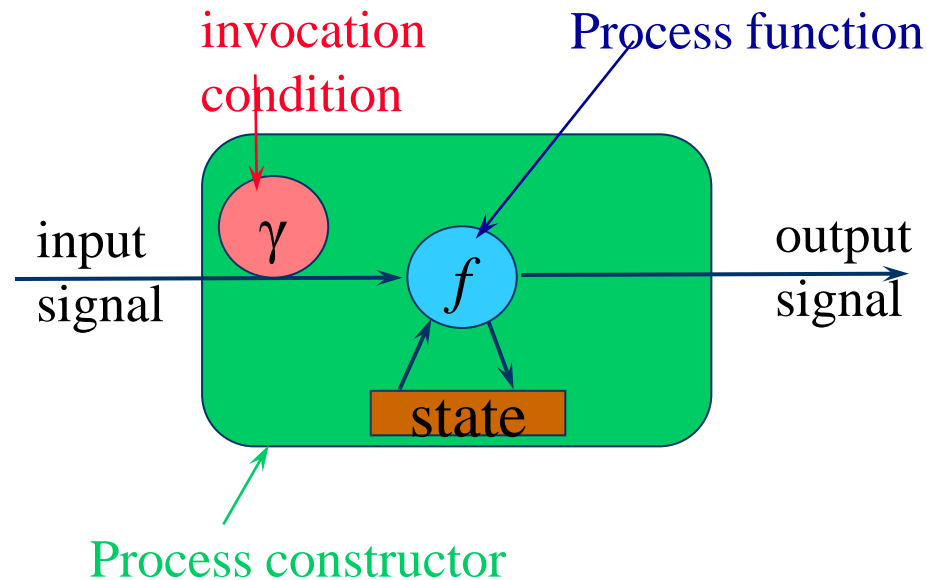
$$\rightarrow C \langle 1, 2, 3, 4 \rangle = \langle \perp, \perp, \perp, \perp \rangle$$

$$f_A(x, y) = \text{if } (y == 3) \text{ then } 3 \text{ else } x$$

$$\rightarrow C \langle 1, 2, 3, 4 \rangle = \langle \perp, \perp, \perp, \perp \rangle$$

Least fixed point over  
values in each  
computation cycle

# Models of Computation



## Process Constructors

- State-less
- Finite state machines
- Signal merge/split
- Source/sink

## Process Combinators

- Sequential composition
- Parallel composition
- Feed-back composition

## Models of Computation

- Untimed MoC (Dataflow, SDF)
- Synchronous MoC (Perfectly, Clocked)
- Discrete Time MoC
- Continuous Time MoC

# Definition of a Model of Computation

The **Untimed Model of Computation** is defined as

$U\text{-MoC} = (C, O)$ , with

$C = \{ \text{comb}U, \text{scan}U, \text{scand}U, \text{mealy}U, \text{moore}U, \text{zip}U, \text{unzip}U, \text{zipWith}U, \text{unzip}U, \text{source}U, \text{sink}U, \text{init}U \}$

$O = \{ ||, o, FP_p \}$

- **Synchronous MoC**
- **Clocked Synchronous MoC**
- **Discrete Time MoC**
- **Continuous MoC**

# Time and Process Invocation In MoCs

- **Untimed MoC:**
  - No explicit time, ordering of events
  - Invocation based on data availability
- **Synchronous MoC:**
  - Slot based time abstraction
  - Invocation in every slot
- **Discrete Time MoC:**
  - Physical, discrete time in seconds
  - Invocation based on data availability and progress of time
- **Continuous Time MoC:**
  - Physical, continuous time in seconds
  - Continuous invocation based on transfer functions

# The Integrated Model of Computation

The **Integrated Model of Computation** is defined as

$\text{HMoC} = (M, C, O)$ , with

$M = \{ \text{U-MoC}, \text{S-MoC}, \text{CS-MoC}, \text{T-MoC}, \text{CT-MoC} \}$

$C = \{ \textit{intSup}, \textit{intSdown}, \textit{intTup}, \textit{intTdown},$   
 $\textit{stripT2S}, \textit{stripT2U}, \textit{stripS2U},$   
 $\textit{insertS2T}, \textit{insertU2T}, \textit{insertU2S},$   
 $\textit{a2dConverter}, \textit{d2aConverter} \}$

$O = \{ \parallel, o, \text{FP}_p \}$

# MoC Interface Processes

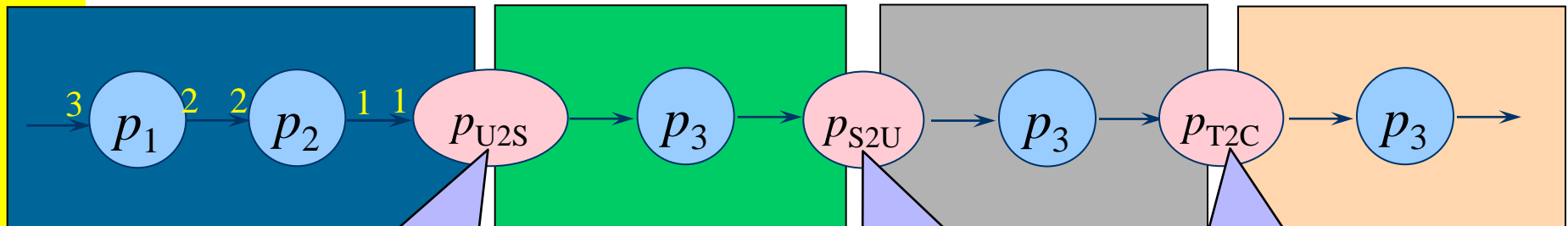
## Downstream the Time Abstraction

U-MoC

S-MoC

T-MoC

C-MoC



**n** number of tokens  
per synchronous  
slot

Each synchronous  
slot lasts **x** nano  
seconds;

Time relation is  
fixed already;

time structure propagates

time relation is fixed

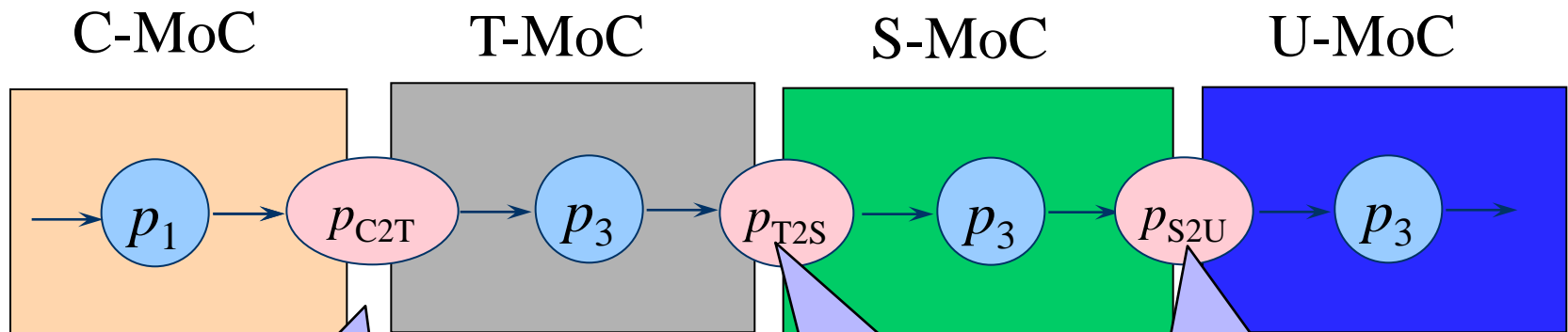
No Change in Accuracy

Interpolation of Data

Interpolation of Data

# MoC Interface Processes 2

## Upstream the Time Abstraction



Time relation is fixed;

Each synchronous slot lasts  $x$  nano seconds;

Only valid data are communicated;

time relation is fixed

time structure propagates

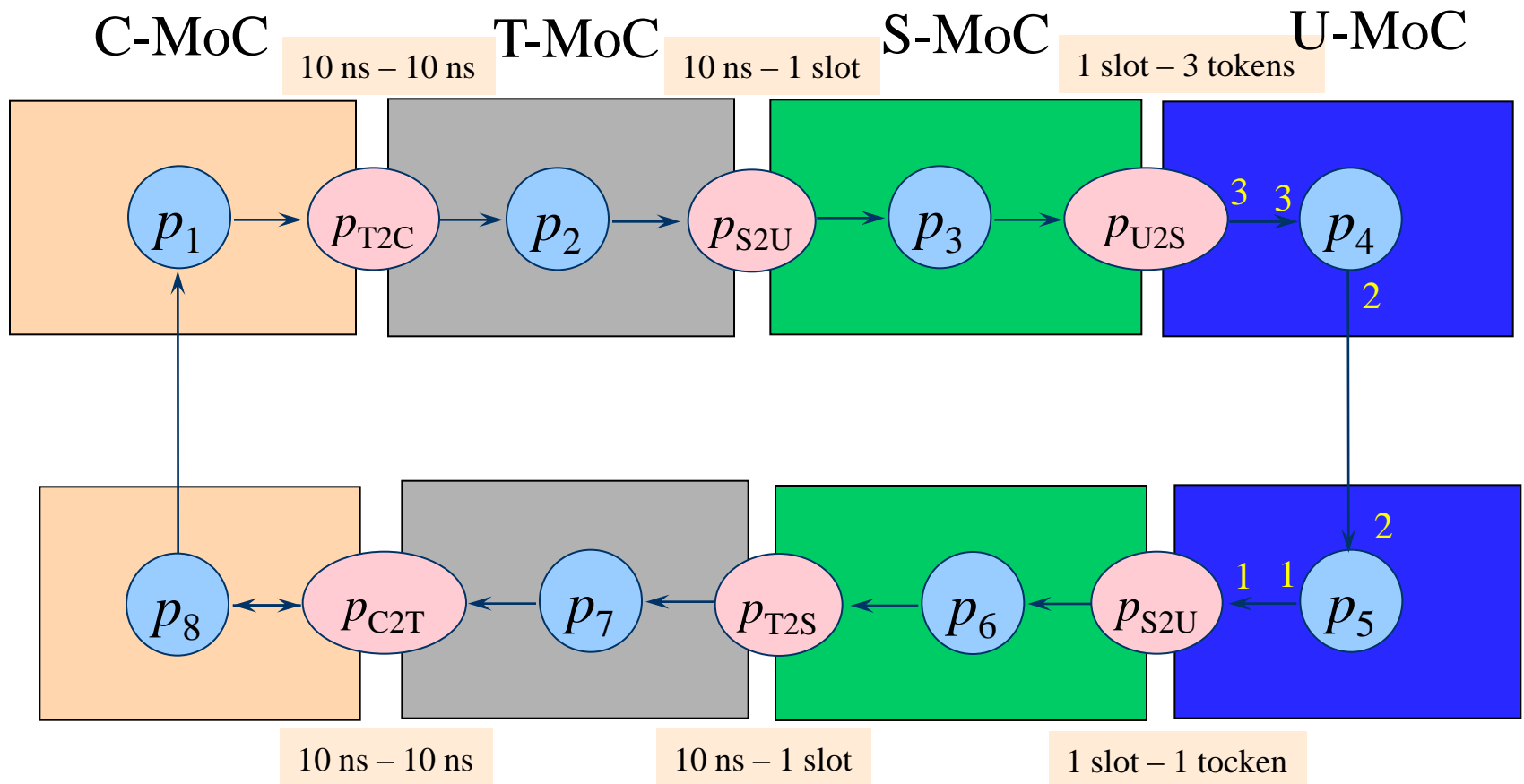
time structure does not propagate

Sampling of Data

Sampling of Data

No loss of Data

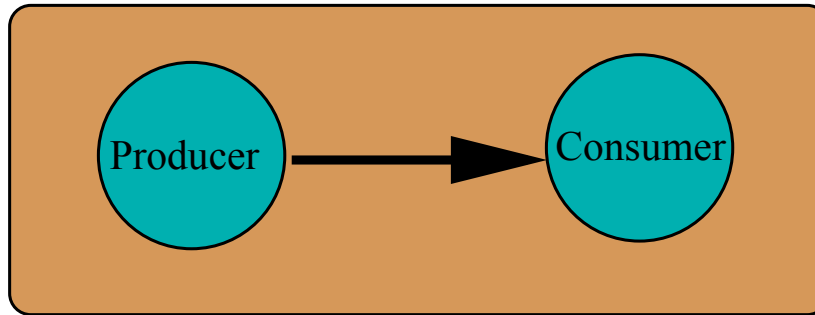
# Heterogeneous MoCs



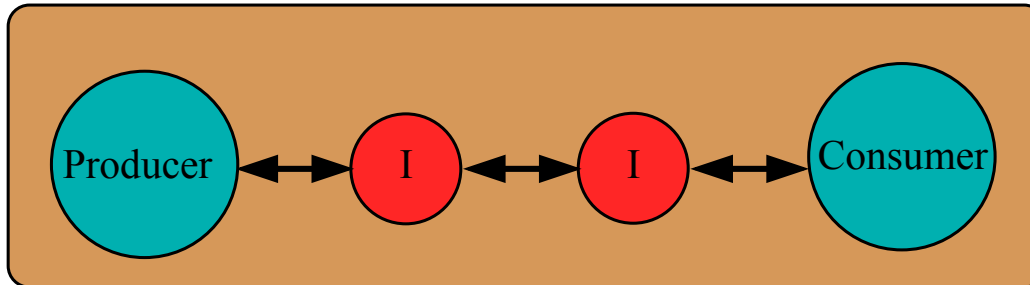
# Implementation

- **Process compilation and synthesis**
- **Communication Refinement**
- **Targets:**
  - **Hardware implementation**
  - **Multi-core software generation**
  - **Parallel simulation**

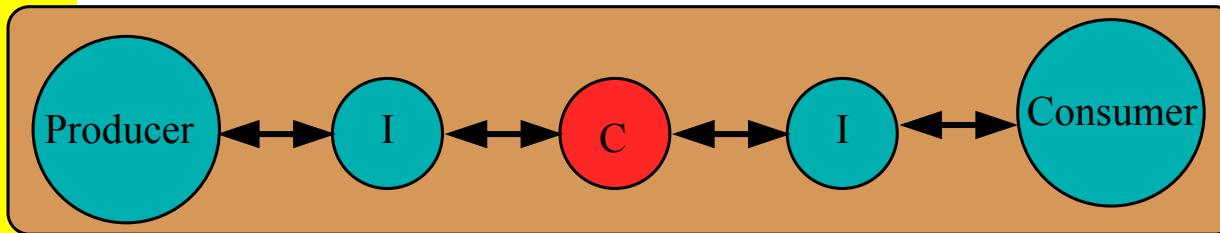
# Communication Channel Refinement



Ideal Communication

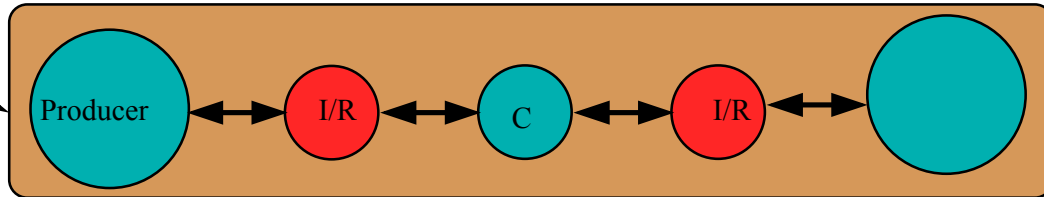


Asynchronous Interface

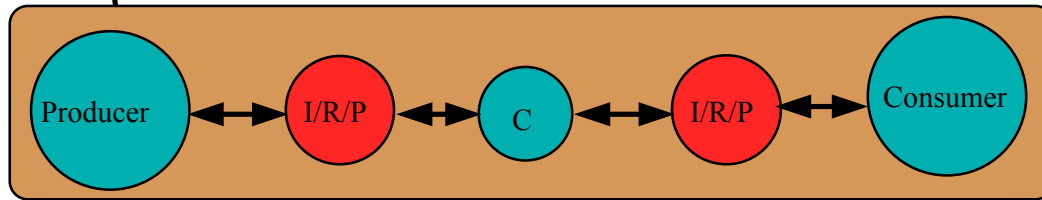


Realistic Channel Model

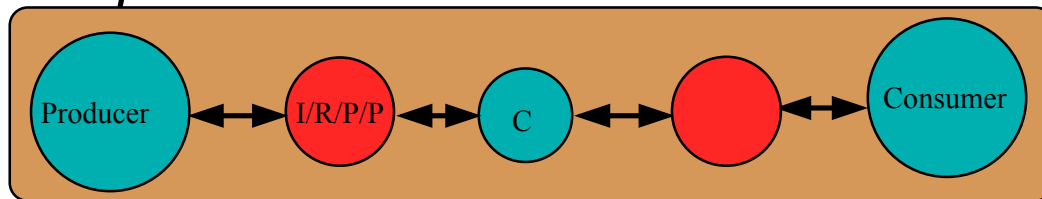
# Refinement and Validation



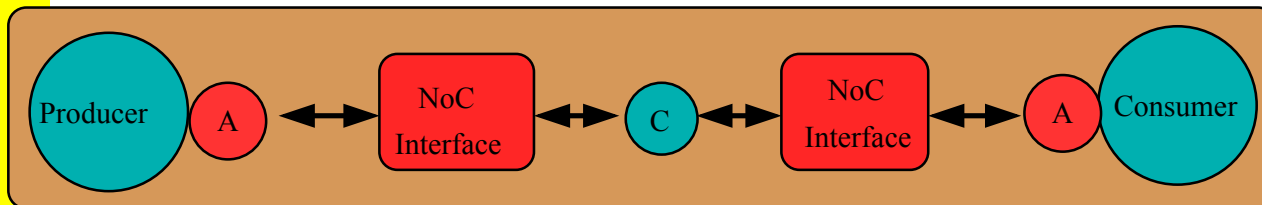
Refinement for Reliability



Refinement for Performance

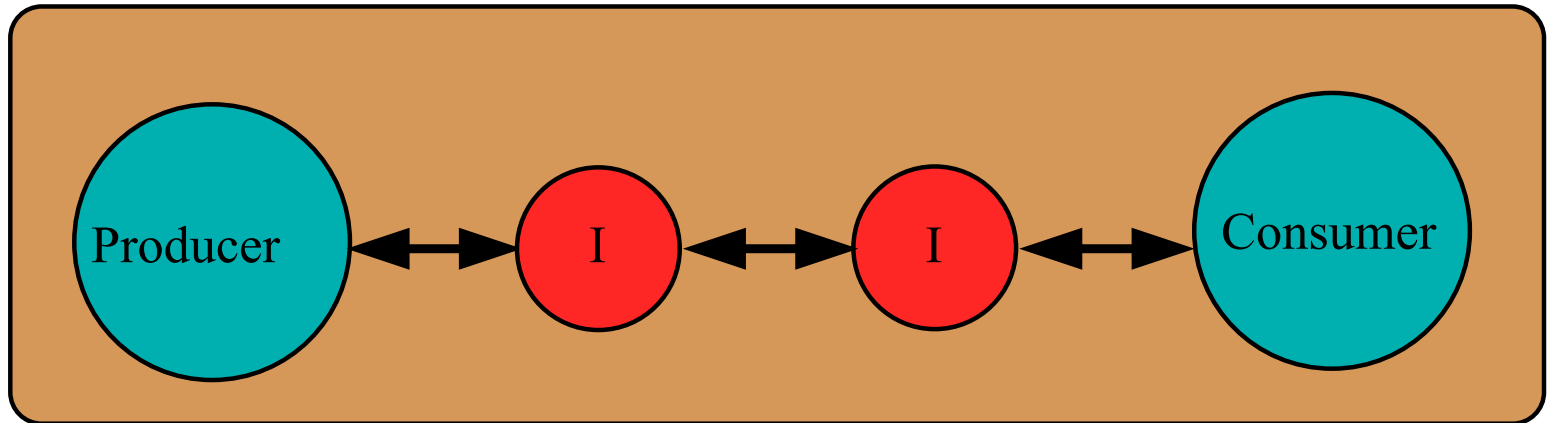


Refinement for Power



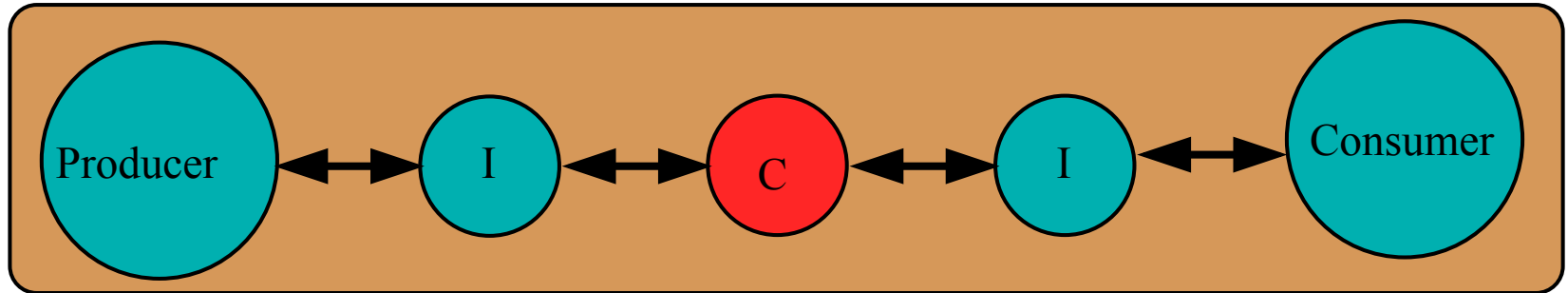
Mapping to NoC Services

# Asynchronous Interfaces



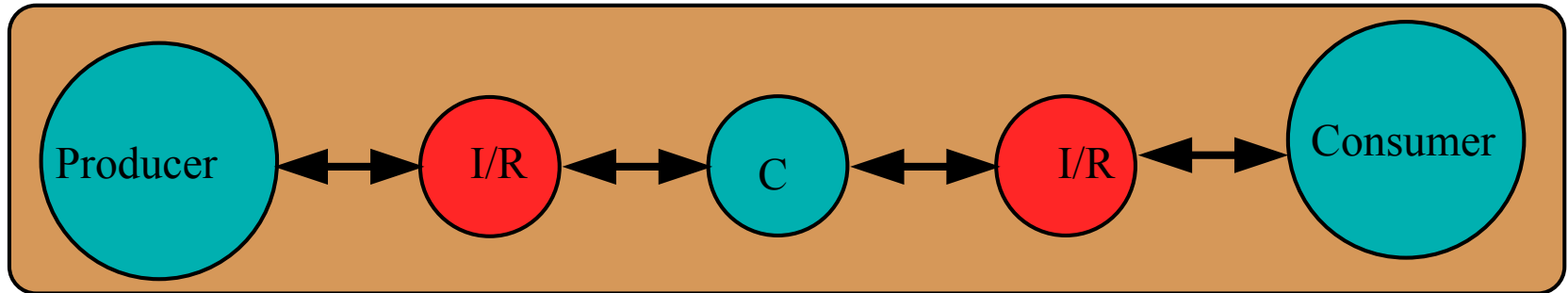
- **Selection of communication mechanism**
  - **Send/receive**
  - **Shared memory**
  - **Buffering**
- **Introduction of flow control**
- **Specification of requirements on**
  - **Performance**
  - **Power**
  - **reliability**

# Modelling the Channel



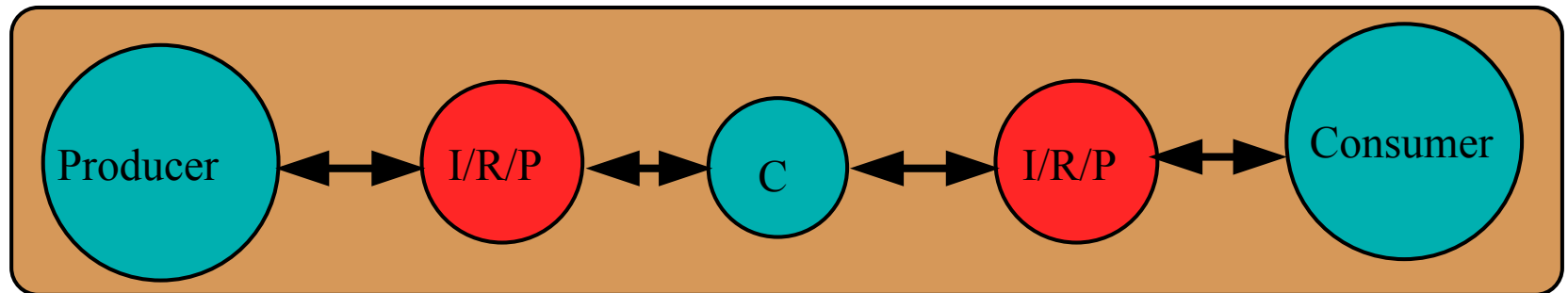
- **Delay**
- **Jitter**
- **Reliability**
- **Stochastic model**

# Refinement for Reliability



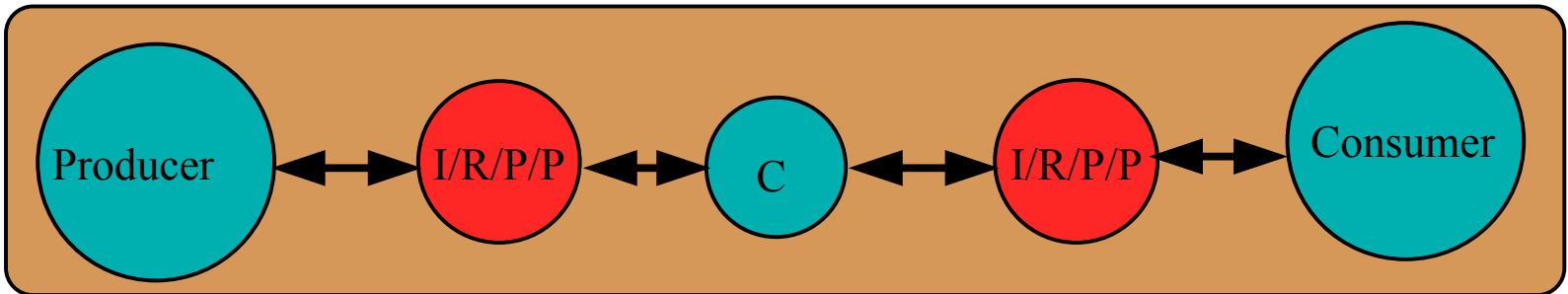
- **Design for a fault hypothesis**
- **Corrupted data**
- **Delay faults**
- **Transient faults**
- **Permanent faults**

# Refinement for Performance



- **Mapping to lower level services**
- **Buffer dimensioning for hiding jitter and delays**

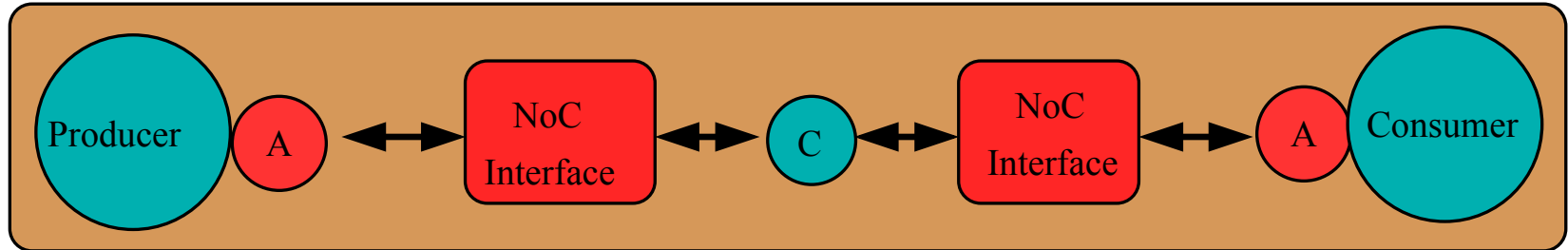
# Refinement for Power



## Options for power optimizations

- Mapping to lower level services
- Minimizing traffic

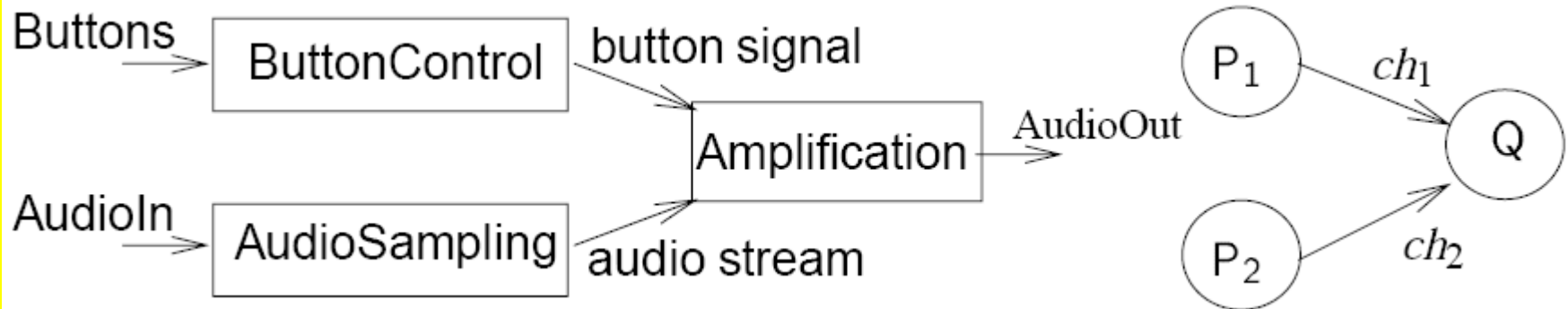
# Mapping onto NoC Services



- **Selecting type of service**
- **Static/Dynamic allocation of connections**
- **Merging of channels into connections**
- **Instantiation of adapters**

# Audio Amplifier Example

## Mapping a Synchronous MoC model onto an MPSoC



### Mapping Alternatives

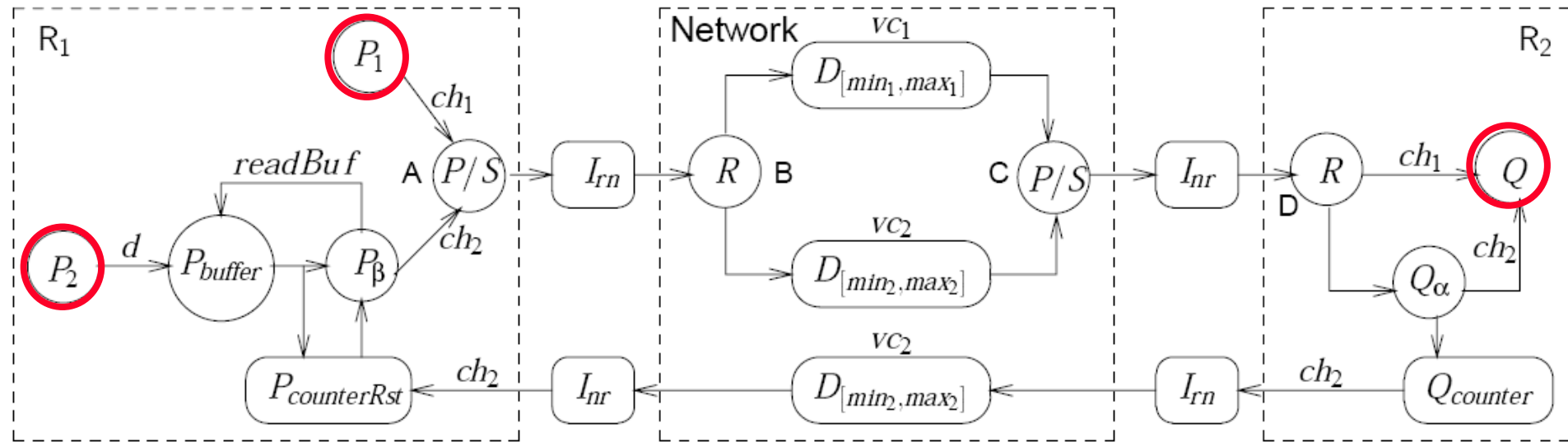
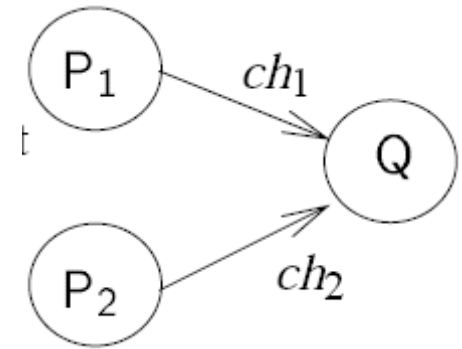
- $P1 \rightarrow R1$
- $P2 \rightarrow R2$
- $Q \rightarrow R3$
- $Ch1 \rightarrow vc1$
- $Ch2 \rightarrow vc2$

- $P1, P2 \rightarrow R1$
- $Q \rightarrow R3$
- $Ch1 \rightarrow vc1$
- $Ch2 \rightarrow vc2$

- $P1, P2 \rightarrow R1$
- $Q \rightarrow R3$
- $Ch1, ch2 \rightarrow vc1$

# Audio Amplifier Example

- $P_1, P_2 \rightarrow R_1$
- $Q \rightarrow R_3$
- $Ch1 \rightarrow vc1$
- $Ch2 \rightarrow vc2$



# ForSyDe Status

## Ideal System Model

No resource limitation on

- Processors
- Communication bandwidth
- memory

Stable Modeling technique  
U-MoC, S-MoC,  
D-MoC, C-MoC  
ForSyDe Libraries

## Implementation model

With finite resources

- Processors, HW blocks
- Reconfigurable resources
- Buffers
- Communication architecture
- Schedulers, arbiters

Set of transformations defined  
Verification of local  
transformations

C program

VHDL design

SystemC model

CoSimulation by Wrapping

Refinement by Replacement  
Methodology

# Selected References

- ForSyDe project home page: <http://www.ict.kth.se/forsyde/>
- Seyed Hosein Attarzadeh Niaki, Ingo Sander, "Semi-Formal Refinement of Heterogeneous Embedded Systems by Foreign Model Integration", Proceedings of the Forum on Design Languages (FDL), Oldenbour, Germany, September 2011.
- Seyed Hosein Attarzadeh Niaki and Ingo Sander, "Co-simulation of embedded systems in a heterogeneous MoC-based modeling framework", 6th IEEE International Symposium on Industrial Embedded Systems (SIES), IEEE, pp. 238--247, June 2011.
- Jun Zhu, Ingo Sander, and Axel Jantsch, "Performance analysis of reconfigurations in adaptive real-time streaming applications", ACM Transactions in Embedded Computing Systems, 2010.
- Zhonghai Lu, Ingo Sander, and Axel Jantsch. Towards performance-oriented pattern-based refinement of synchronous models onto NoC communication. In *9th Euromicro Conference on Digital System Design (DSD 2006)*, August 2006.
- Tarvo Raudvere, Ingo Sander, and Axel Jantsch. Application and verification of local non-semantic-preserving transformations in system design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(6):1091-1103, June 2008.
- Deepak Mathaikutty, Hiren Patel, Sandeep Shukla, and Axel Jantsch. SML-Sys: A functional framework for multiple models of computation for heterogeneous system design. *Design Automation for Embedded Systems*, 2008.
- Ingo Sander and Axel Jantsch. Modelling adaptive systems in ForSyDe. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 200(2):39-54, 2008.
- Axel Jantsch and Ingo Sander, "Models of Computation in the Design Process", SoC: Next Generation Electronics, IEE, edited by Bashir M Al-Hashimi, Invited contribution, 2005.
- Axel Jantsch and Ingo Sander, "Models of Computation and languages for embedded system design", IEE Proceedings on Computers and Digital Techniques, vol. 152, pp. 114-129, no. 2, Special issue on Embedded Microelectronic Systems; Invited paper, March 2005.
- Ingo Sander, Axel Jantsch, and Zhonghai Lu, "Development and Application of Design Transformations in ForSyDe", IEE Proceedings on Computers and Digital Technique, vol. 150, pp. 313-320, no. 5, September 2003.
- Ingo Sander and Axel Jantsch, "System Modeling and Transformational Design Refinement in ForSyDe", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, pp. 17-32, no. 1, January 2004.

**Questions ?**