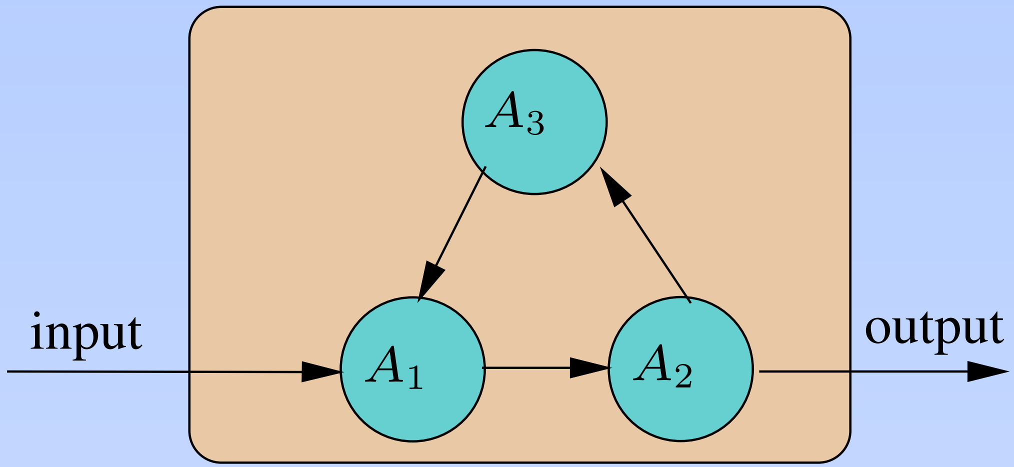
ForSyDe: A Denotational Framework for Heterogeneous Models of Computation

Axel Jantsch, Ingo Sander, Jun Zhu
Royal Institute of Technology, Stockholm

Models of Computation and Communication
ARTIST2 Workshop
November 2006

ForSyDe Features



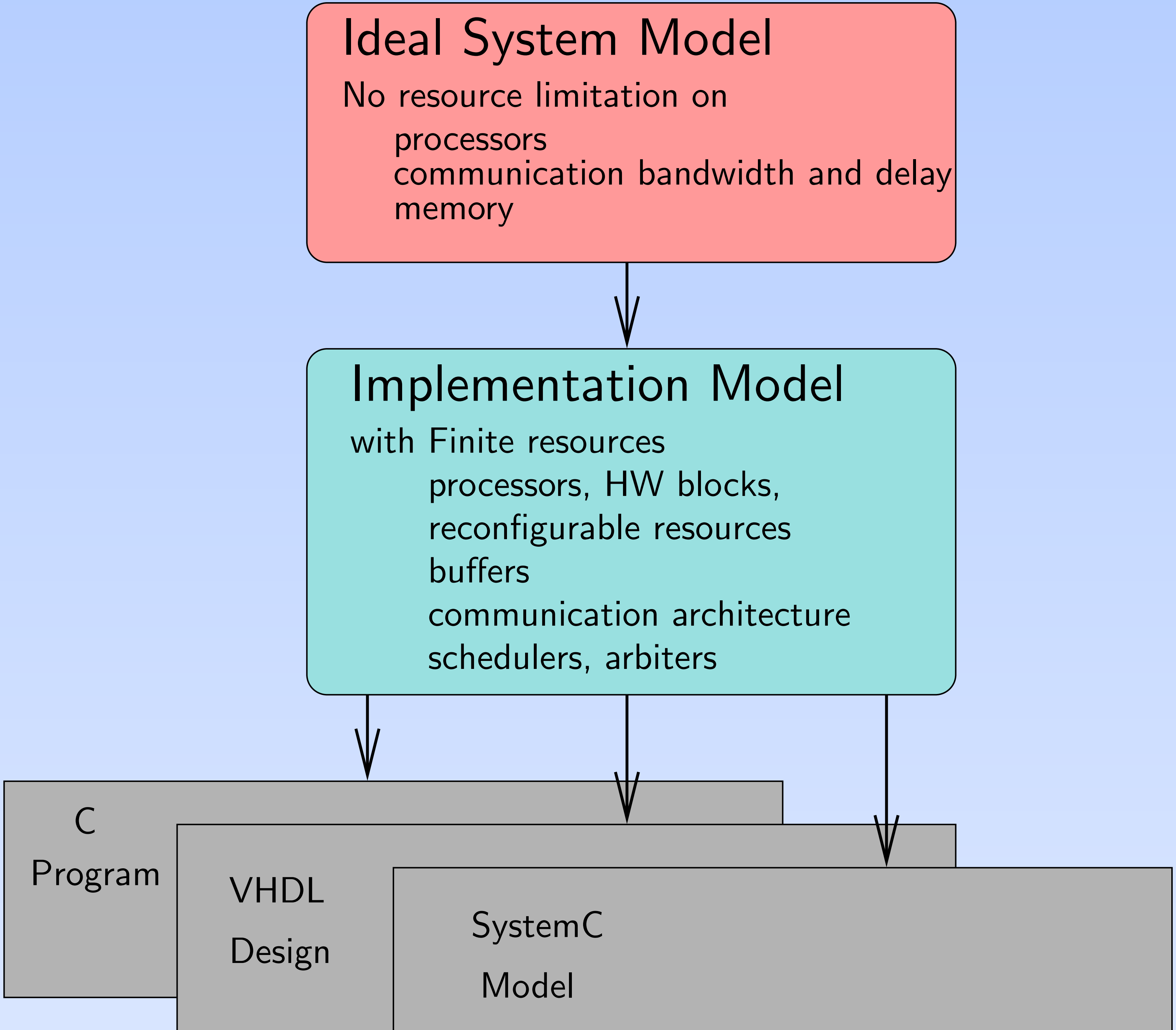
Processes

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

Signals:

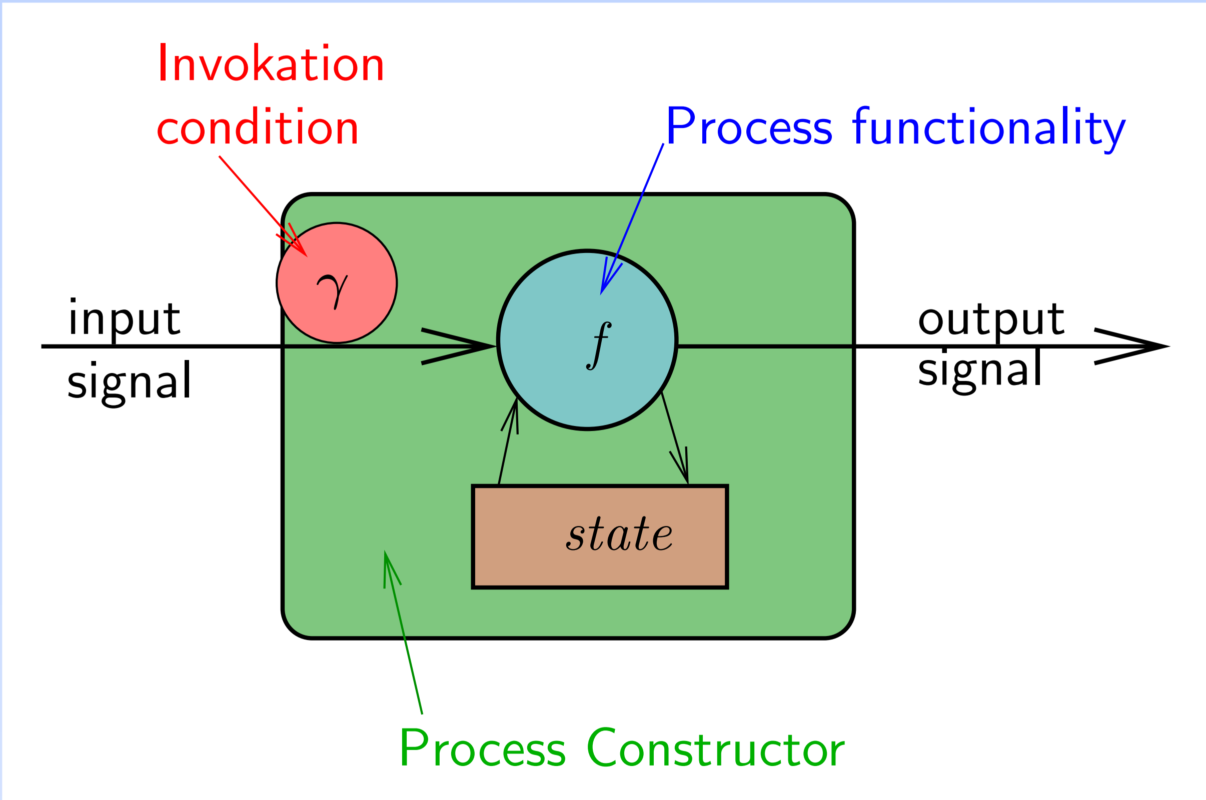
- Sequences of events
- Preserve event order
- Have one writer and multiple readers
- Untimed MoC: Events are partially ordered
- Discrete Time MoCs: Signals carry timing information

The ForSyDe Design Flow

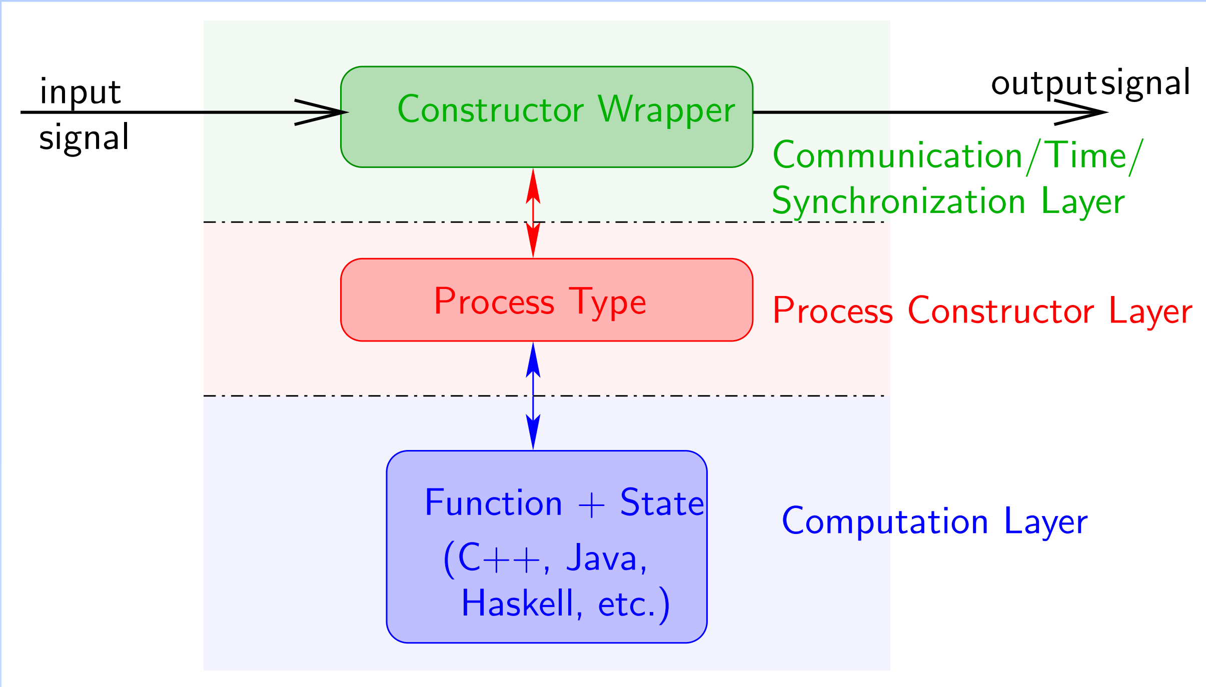


Process Constructors

process = constructor + function + initial_state + invokation_condition



Layered View of Process Constructors



Models of Computation

- Untimed MoC (Datflow, SDF, Rendezvous)
- Synchronous MoC (Perfectly, Clocked)
- Discrete Time MoC
- Soon: Continuous Time MoC

Process Combinators

- Sequential Composition
- Parallel Composition
- Feed-back Composition

Process Constructor Types

- State-less Processes
- FSM Machines
- Zip / Unzip Processes
- Sources and Sinks

The $\mathtt{mapU}$ Process Constructor

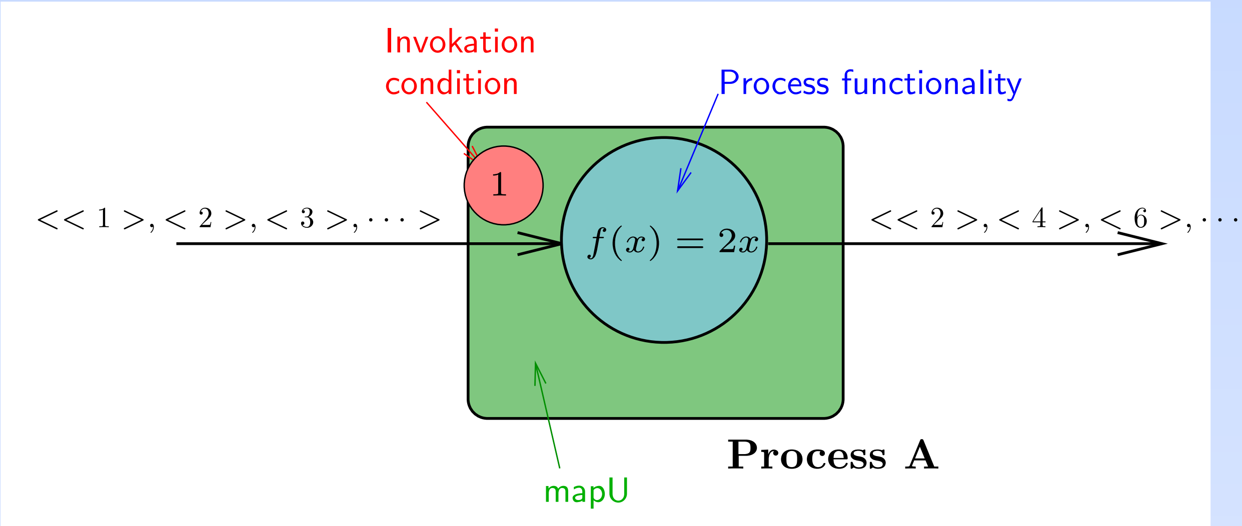
$$
\begin{aligned}
\mathtt{mapU}(c, f) \quad &= \quad p \\
\text{where} \quad & p(\dot{s}) = \dot{s}' \\
& f(\dot{a}_i) = \dot{a}_i' \\
& \pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle, \ \nu(i) = c \\
& \pi(\nu', \dot{s}') = \langle \dot{a}_i' \rangle, \ \nu'(i) = \# f(\dot{a}_i)
\end{aligned}
$$

The *mapU* Process Constructor

$$
\begin{aligned}
\mathtt{mapU}(c, f) &= p \\
\text{where} \quad & p(\dot{s}) = \dot{s}' \\
& f(\dot{a}_i) = \dot{a}'_i \\
& \pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,\ \nu(i) = c \\
& \pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,\ \nu'(i) = \# f(\dot{a}_i)
\end{aligned}
$$

Example:

$$
\begin{aligned}
A &= \mathtt{mapU}(c, f) \\
\text{where} \quad & c = 1 \\
& f(x) = 2x
\end{aligned}
$$



Definition of a Model of Computation

The **Untimed Model of Computation (Untimed MoC)** is defined as Untimed MoC$=(C, O)$, where

$$
\begin{aligned}
C = \{ & \mathtt{mapU}, \mathtt{scanU}, \mathtt{scandU}, \mathtt{mealyU}, \mathtt{mooreU}, \\
& \mathtt{zipU}, \mathtt{zipUs}, \mathtt{zipWithU}, \mathtt{unzipU}, \\
& \mathtt{sourceU}, \mathtt{sinkU}, \mathtt{initU}\} \\
O = \{ & \|, \circ, \mathbf{FB_P}\}
\end{aligned}
$$

- Synchronous Model of Computation
- Clocked Synchronous Model of Computation
- Discrete Time Model of Computation

The Integrated MoC

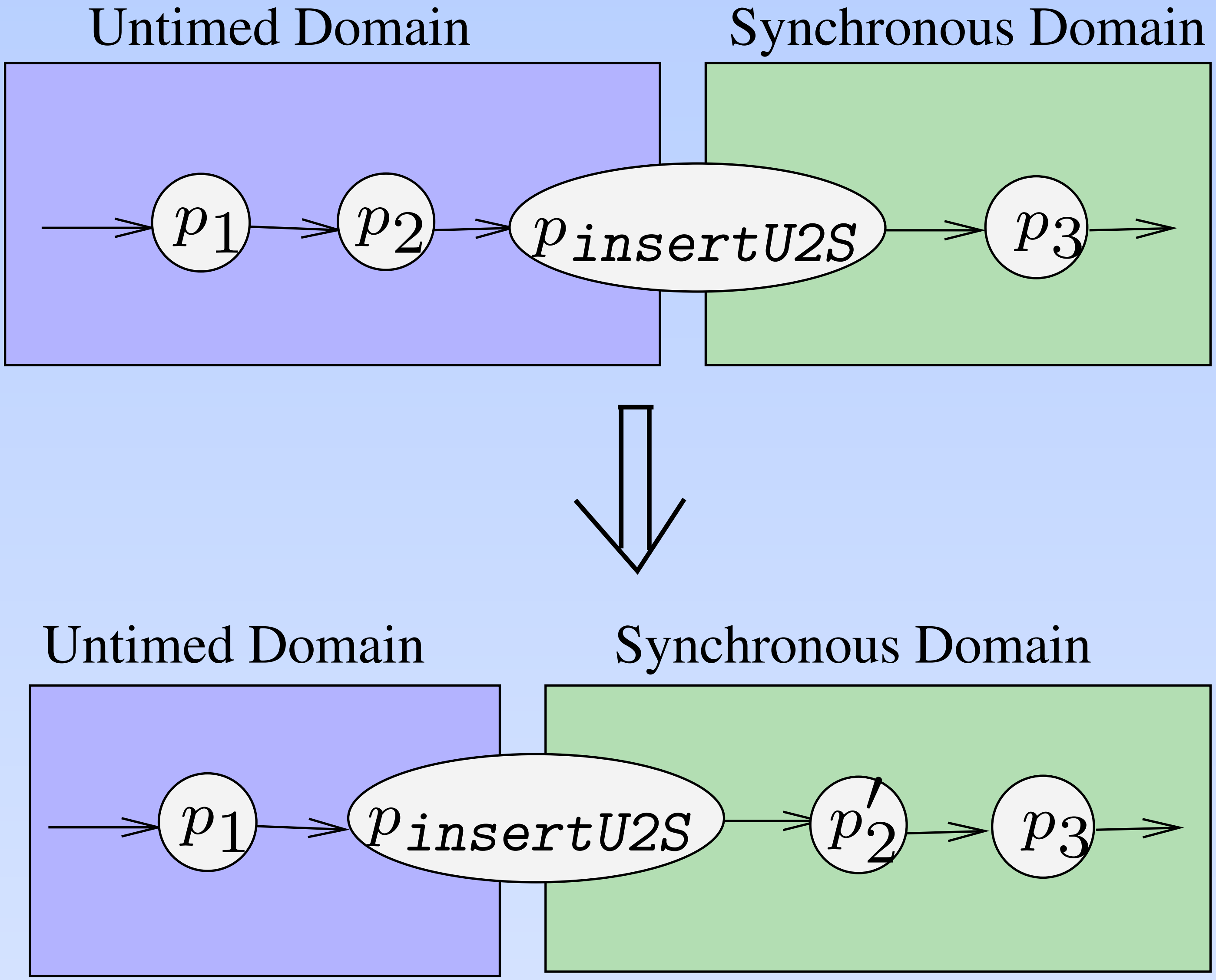
The **Integrated Model of Computation (Integrated MoC)** is defined as Integrated HMoC$=(M, C, O)$, where

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

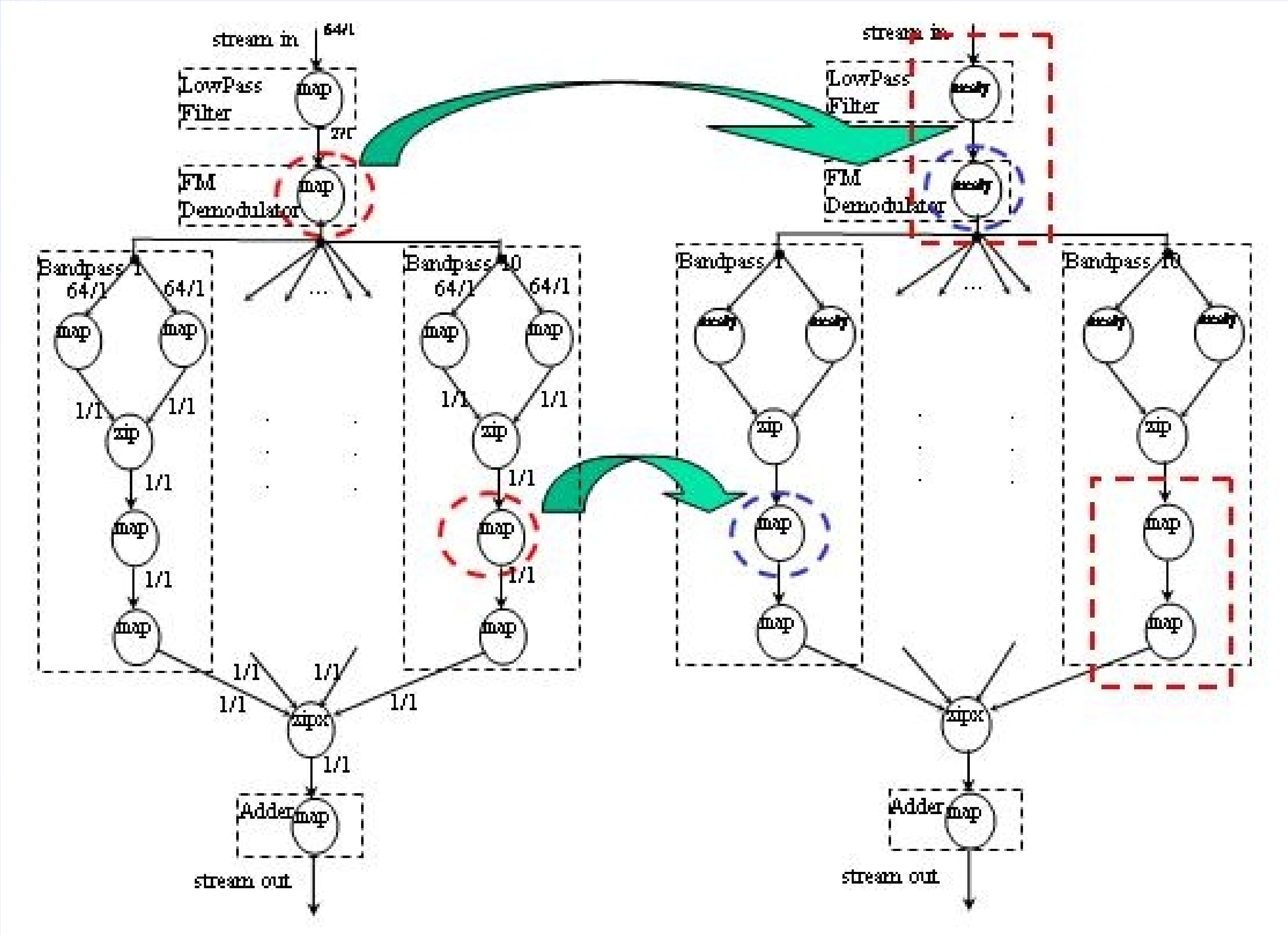
$$\begin{aligned} C = \{ & \mathtt{intSup}, \mathtt{intSdown}, \mathtt{intTup}, \mathtt{intTdown}, \\ & \mathtt{stripT2S}, \mathtt{stripT2U}, \mathtt{stripS2U}, \\ & \mathtt{insertS2T}, \mathtt{insertU2T}, \mathtt{insertU2S}\} \end{aligned}$$

$$O = \{\|, \circ, \mathbf{FB_P}\}$$

Process Migration



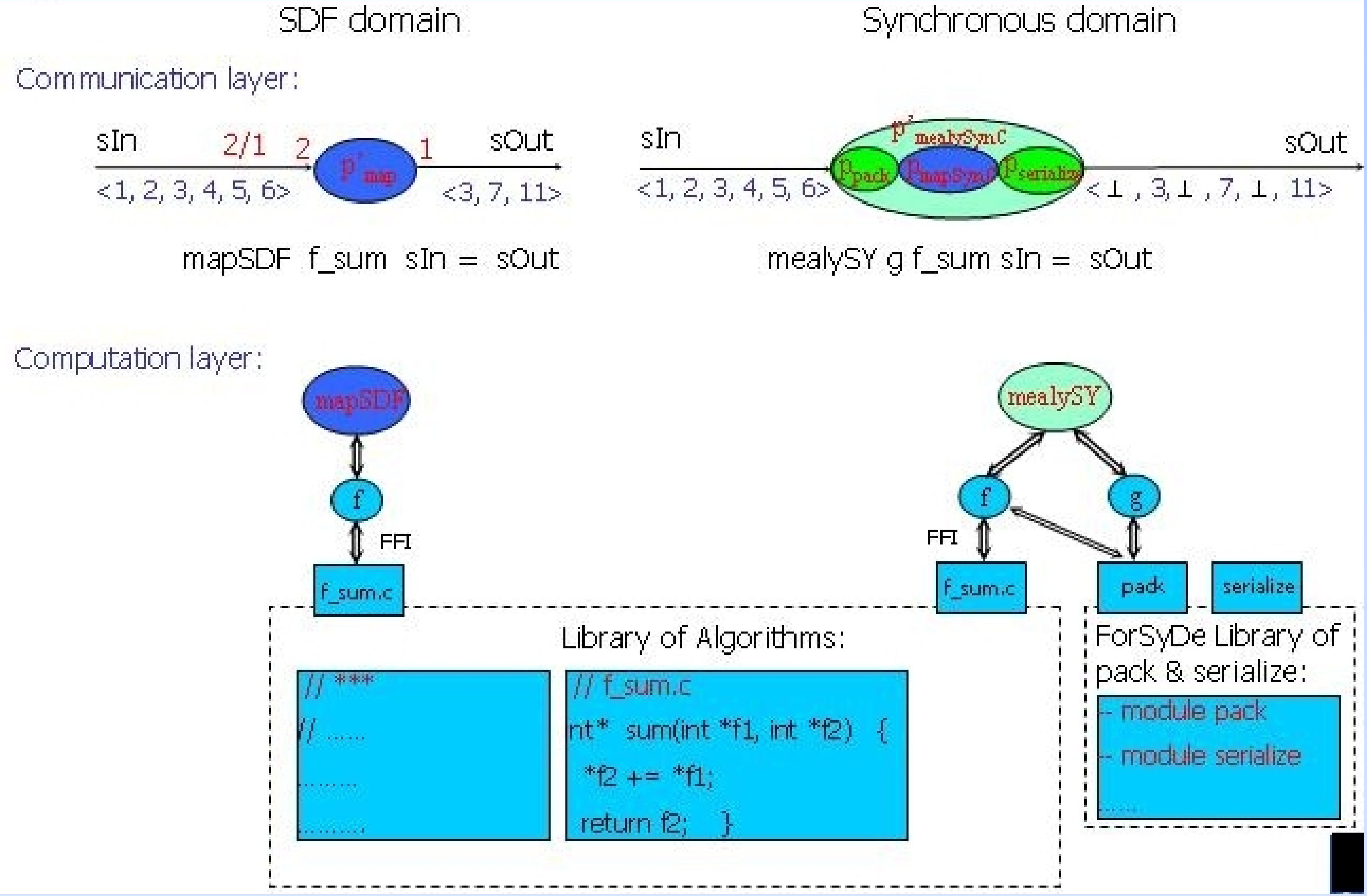
Process Refinement - FM Software Radio Example



SDF Model

Synchronous Model

Process Refinement - FM Software Radio Example



Transformation Rules - Scheduling

- A combinational process with m input signals is modeled with $zipWithSY_m(f)$
- In each event cycle the function f is applied to the current values of the input signals
- A large amount of computational resources may be required for these processes

Transformation Rules - Scheduling

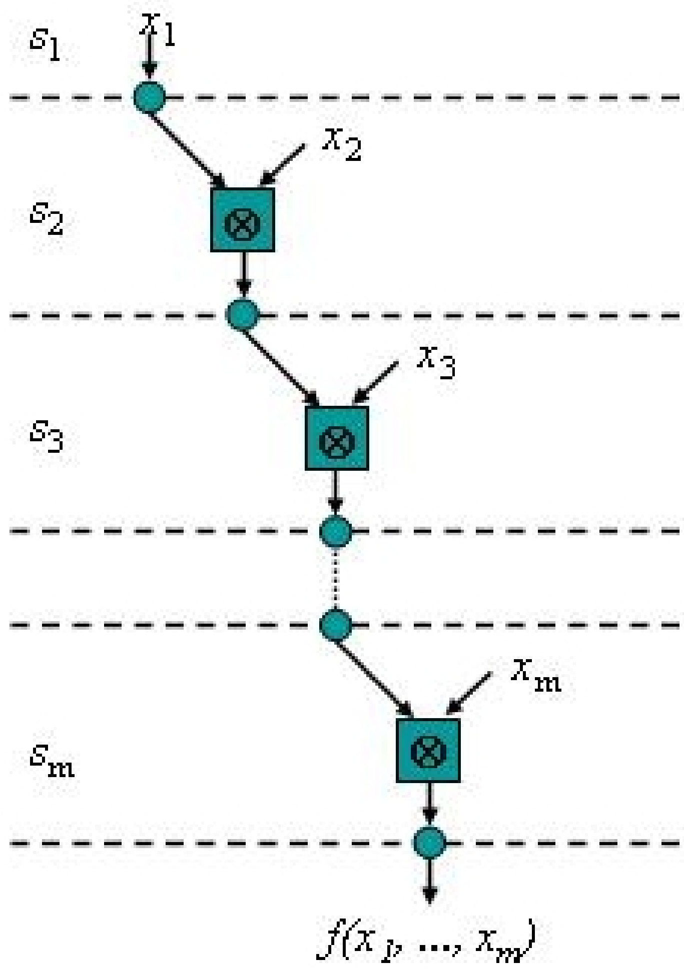
Combinational process

If

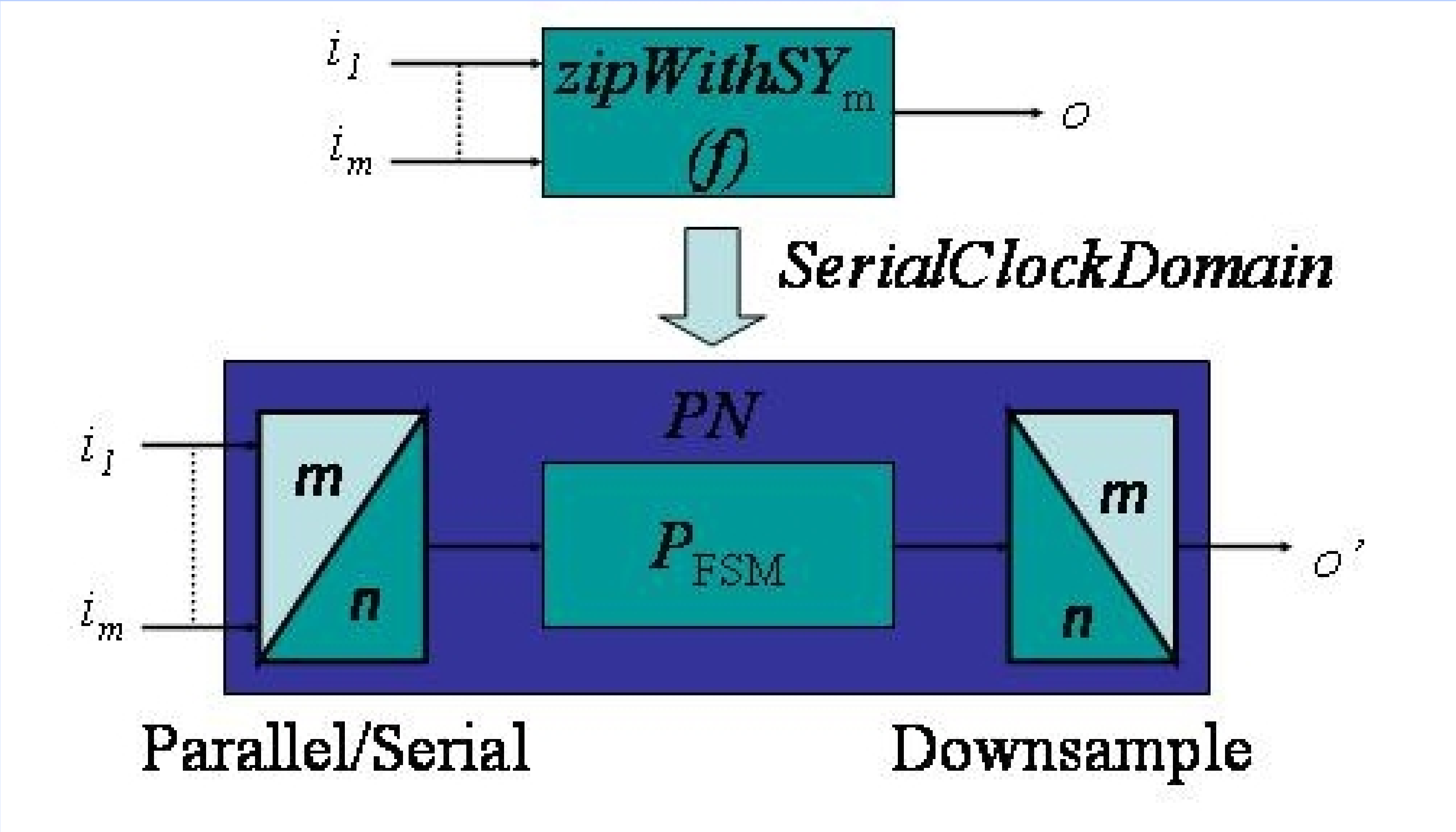
$$f(x_1, \ldots, x_m) = x_1 \otimes x_2 \otimes \ldots \otimes x_m$$

the following schedule using only one computational unit can be derived:

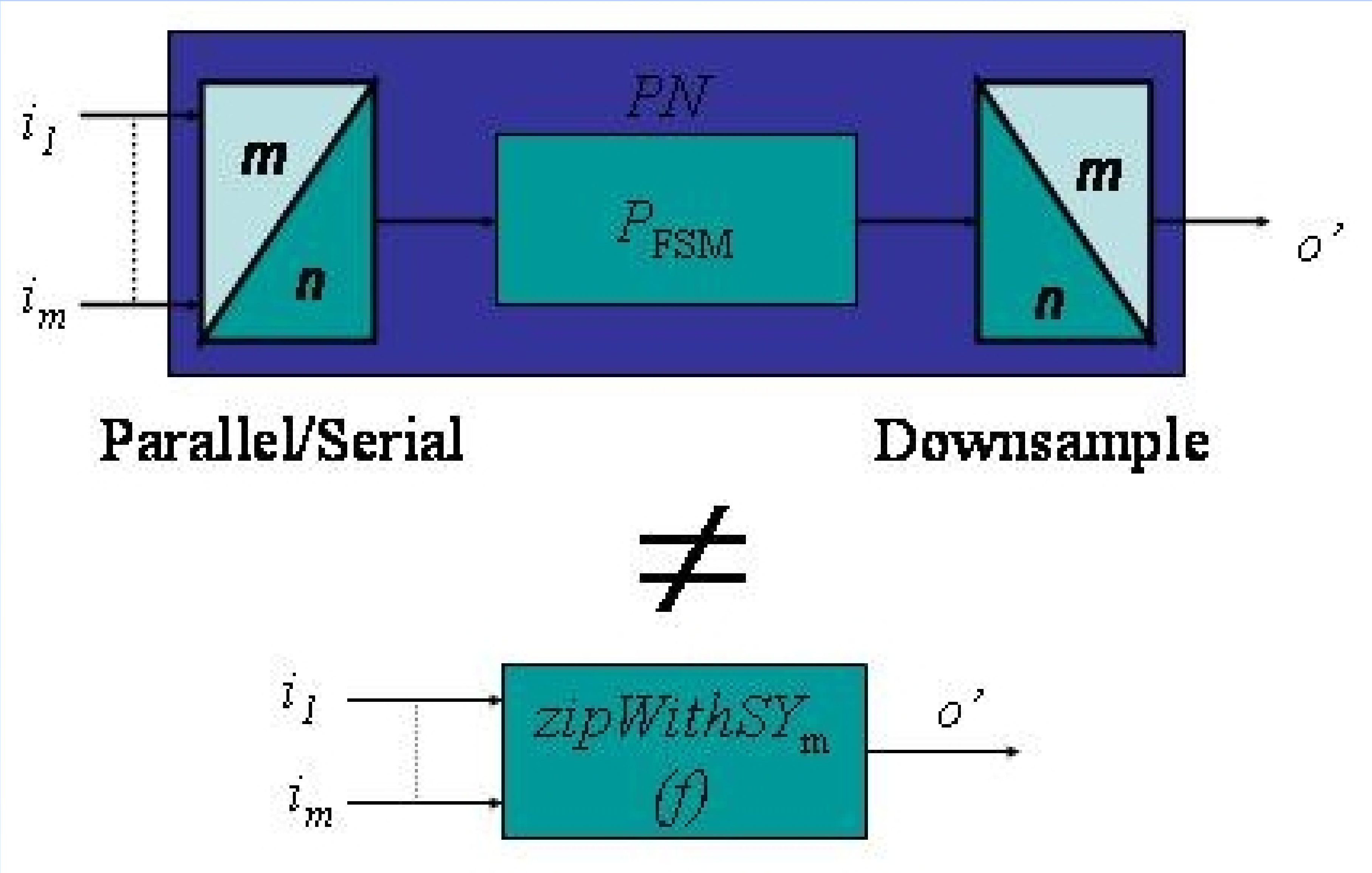
Schedule



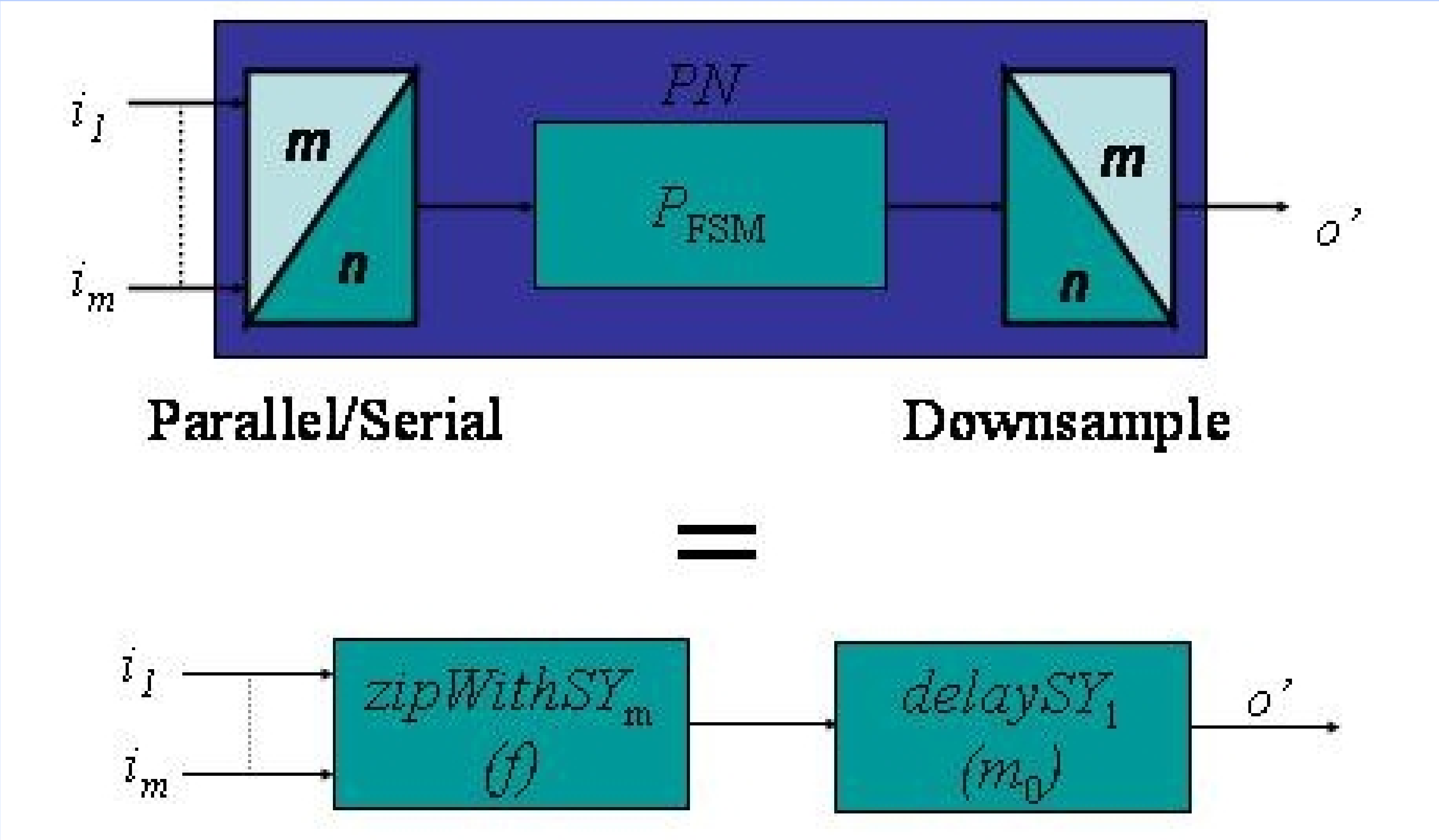
Transformation Rules - Scheduling



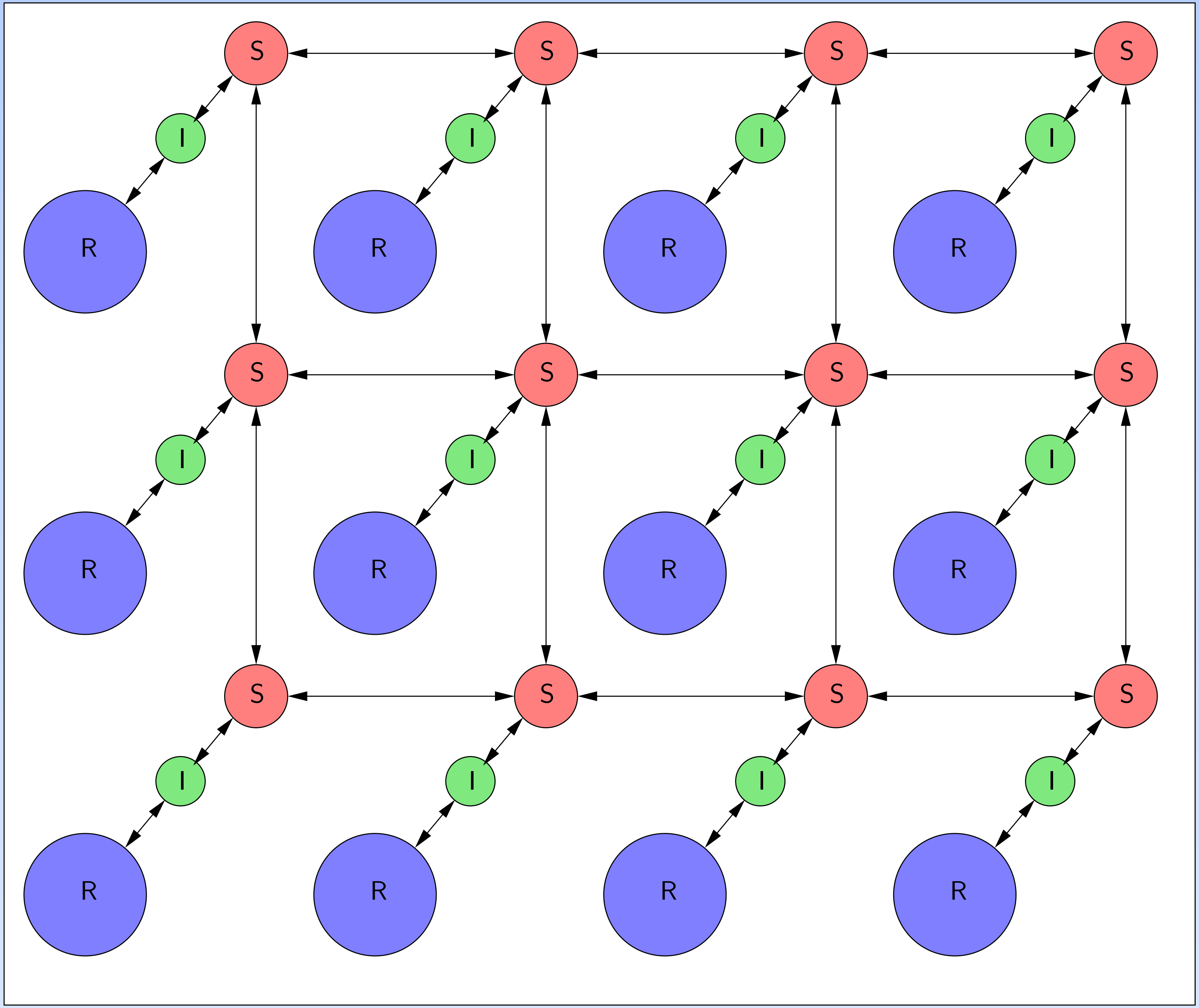
Transformation Rules - Scheduling



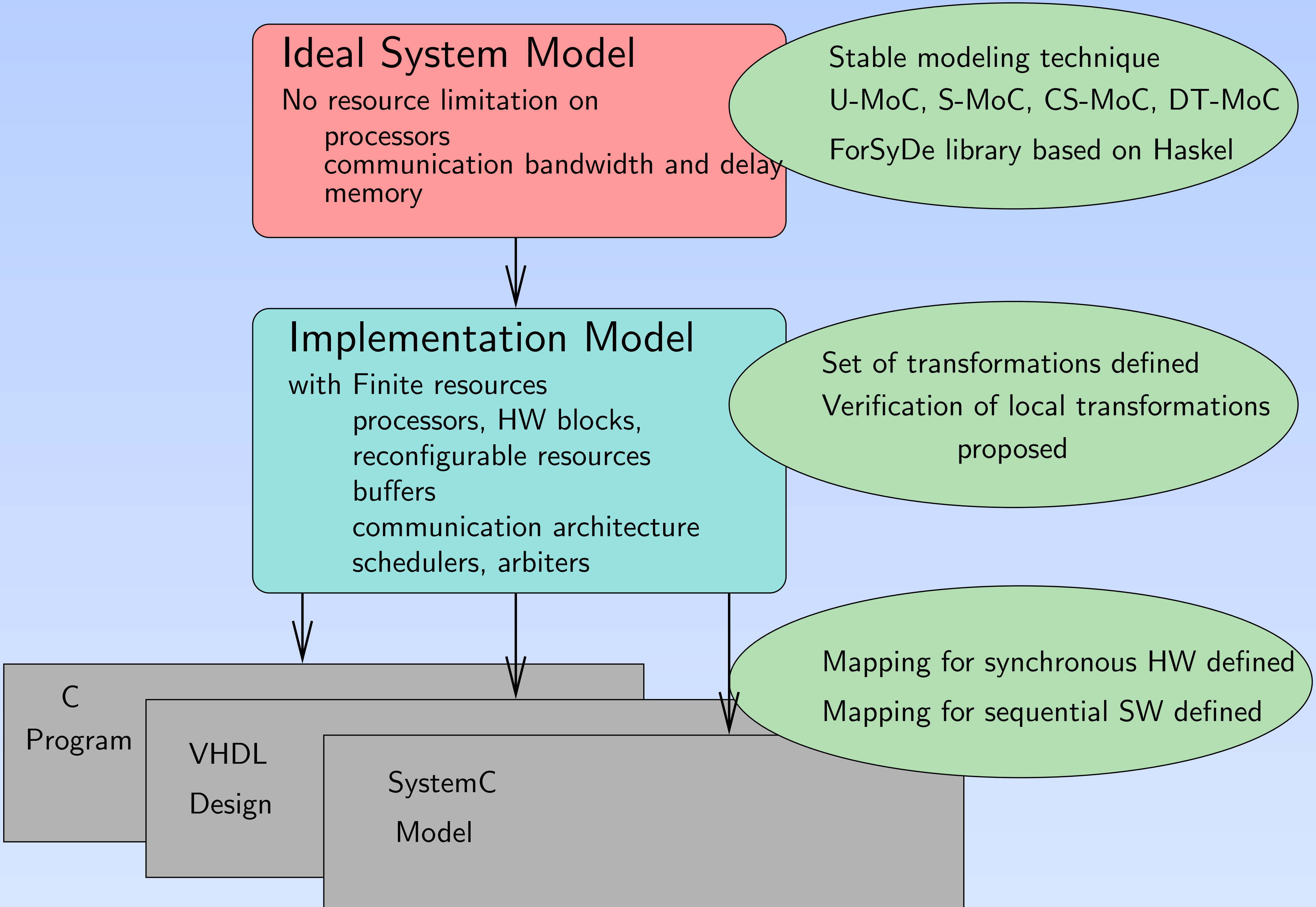
Transformation Rules - Scheduling



NoC Simulator Case Study



ForSyDe Status



ForSyDe Status

Ideal System Model

No resource limitation on
- processors
- communication bandwidth and delay
- memory

Stable modeling technique

U-MoC, S-MoC, CS-MoC, DT-MoC

ForSyDe library based on Haskel

Implementation Model

with Finite resources
- processors, HW blocks,
- reconfigurable resources
- buffers
- communication architecture
- schedulers, arbiters

Set of transformations defined

Verification of local transformations proposed

C

Program

Mapping for synchronous HW defined

Mapping for sequential SW defined

Ongoing work

Definition of a CT-MoC

Modeling of non-functional properties

Modeling of adaptive resources

GME based tool support for transformations

Communication refinement method