and routing

Mapping and scheduling data-flow applications using the KRG model

Jean-Vivien Millo, Robert de Simone

EPI AOSTE, INRIA Sophia-Antipolis

Good afternoon everybody!

From Prof. Ramesh

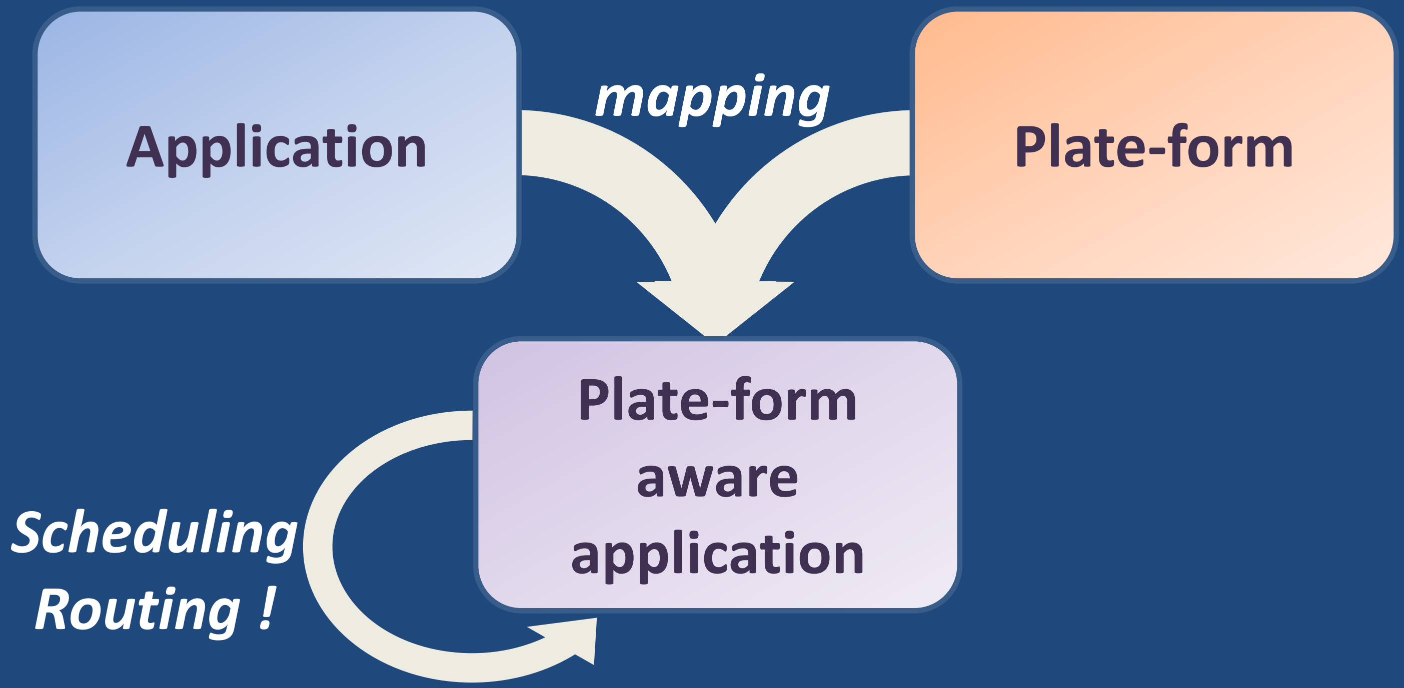
Outline

- Introduction
- The KRG model
- Front end required!
- Case study: All to all propagation algorithm

General view

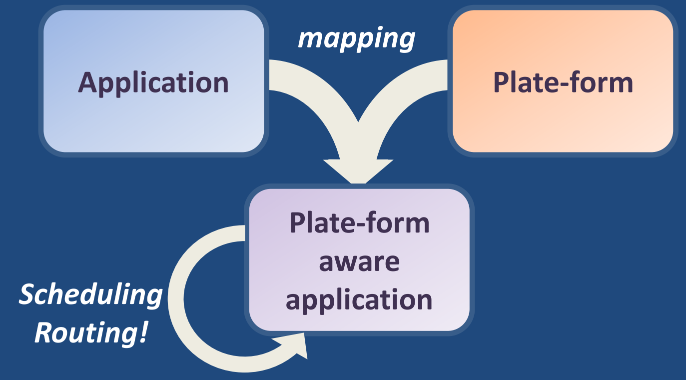
- Mapping and Scheduling data-flow application
  - AAA: Allocation Application / Architecture
  - Plate-form based design
  - Y-Chart approach [1]
- using the K-Periodically Routed Graph (KRG)
  - An incremental approach
  - Fine grain parallelism

General view



General view

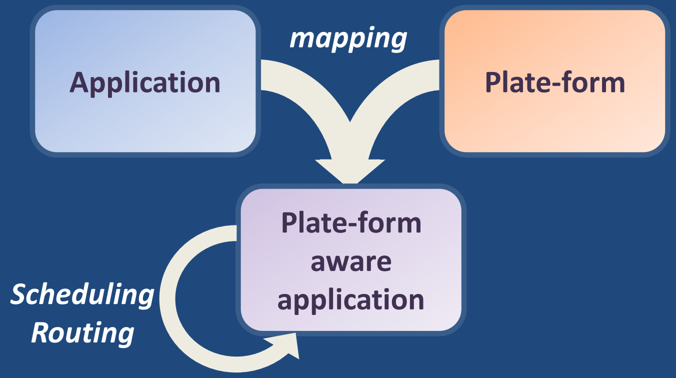
- Application
  - Data-flow
  - High parallelism



- Plate-form
  - Heterogeneous multi-core
  - On-chip network

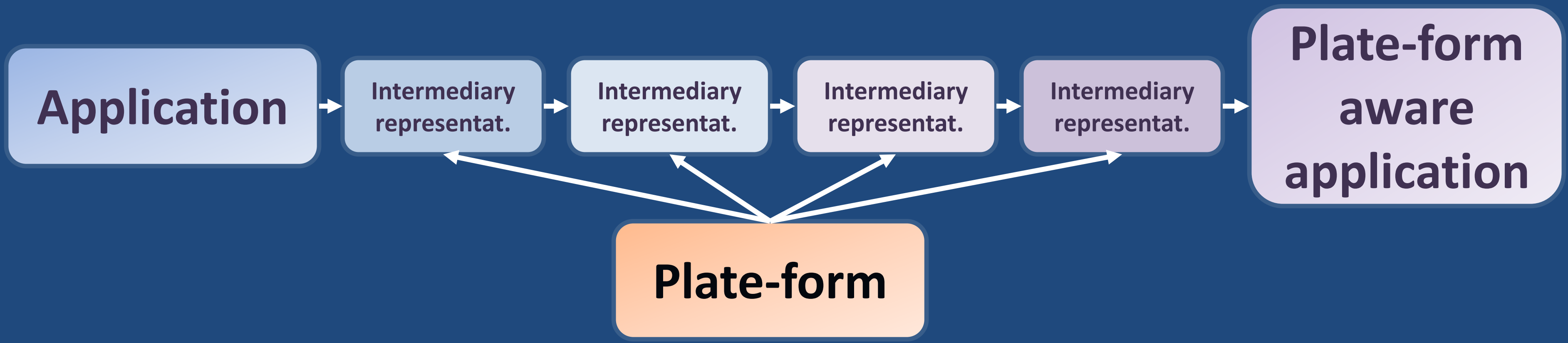
General view

- Mapping
  - Allocating tasks on cores
  - Allocating com. on network!



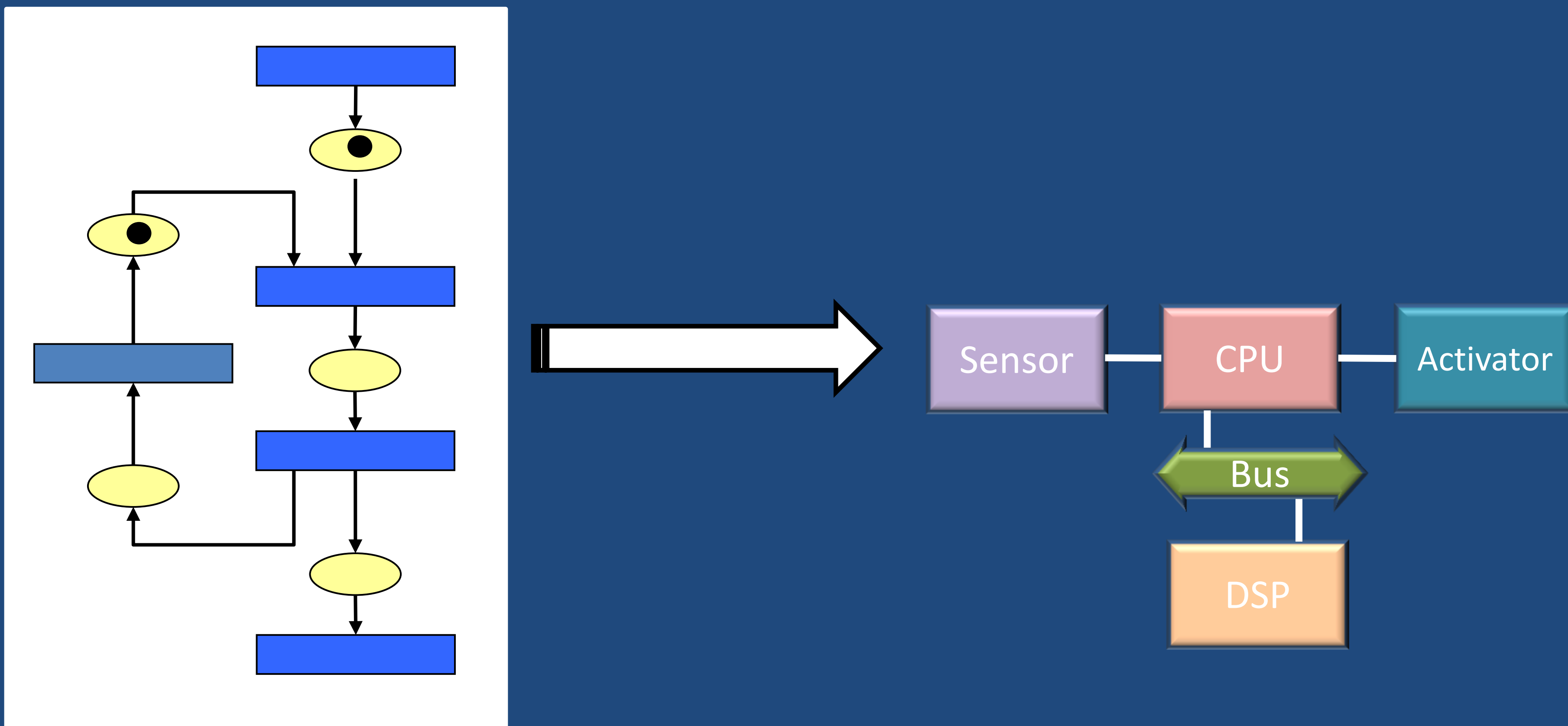
- Scheduling
  - Temporal allocation of task's occurrences
- Routing
  - Temporal allocation of com. Occurrences
- Not necessary in sequence

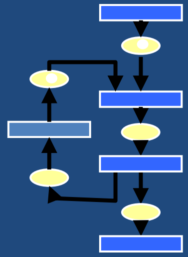
An incremental approach



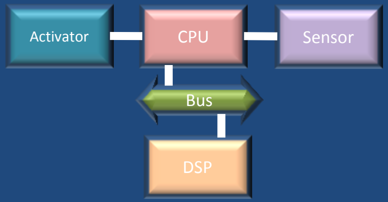
- At each step:
  - Analyses are possible
  - Back tracking based on analyses results

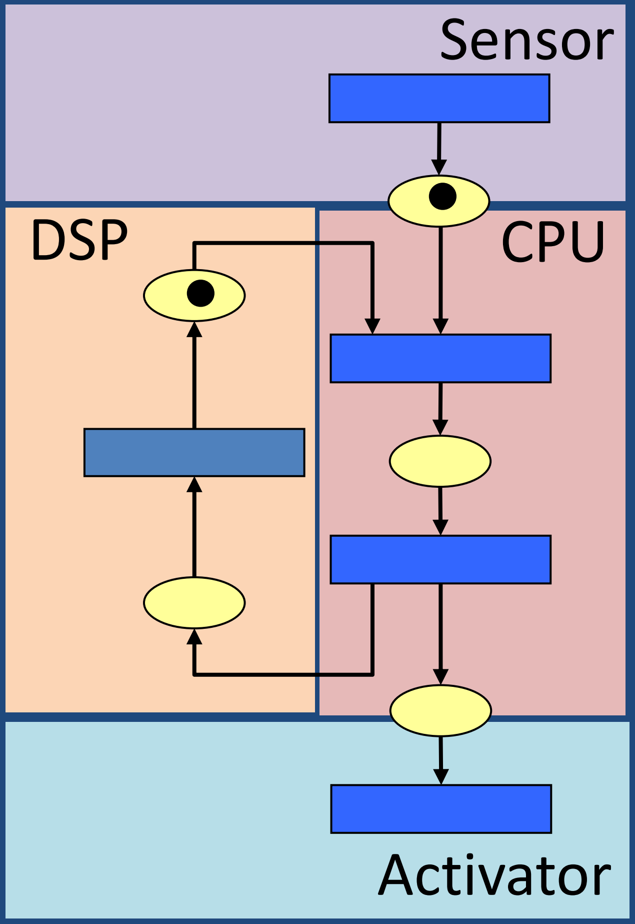
On an example

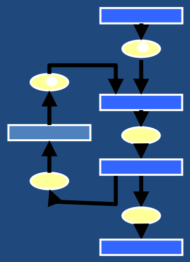




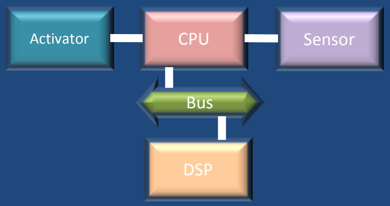
Mapping

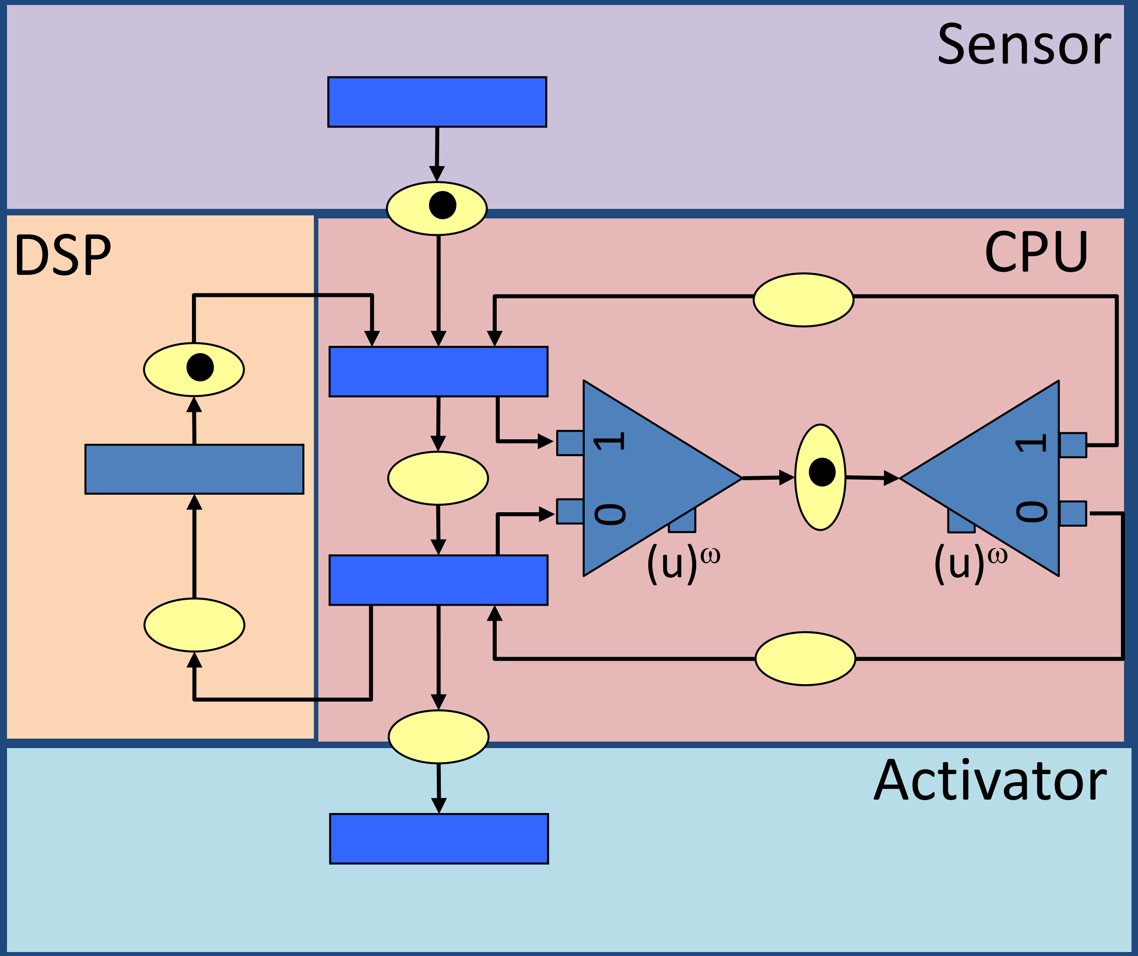


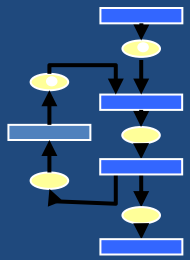


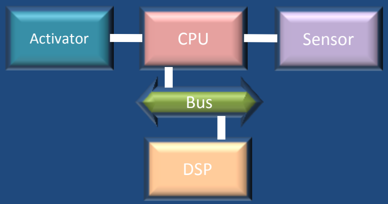


Mapping tasks



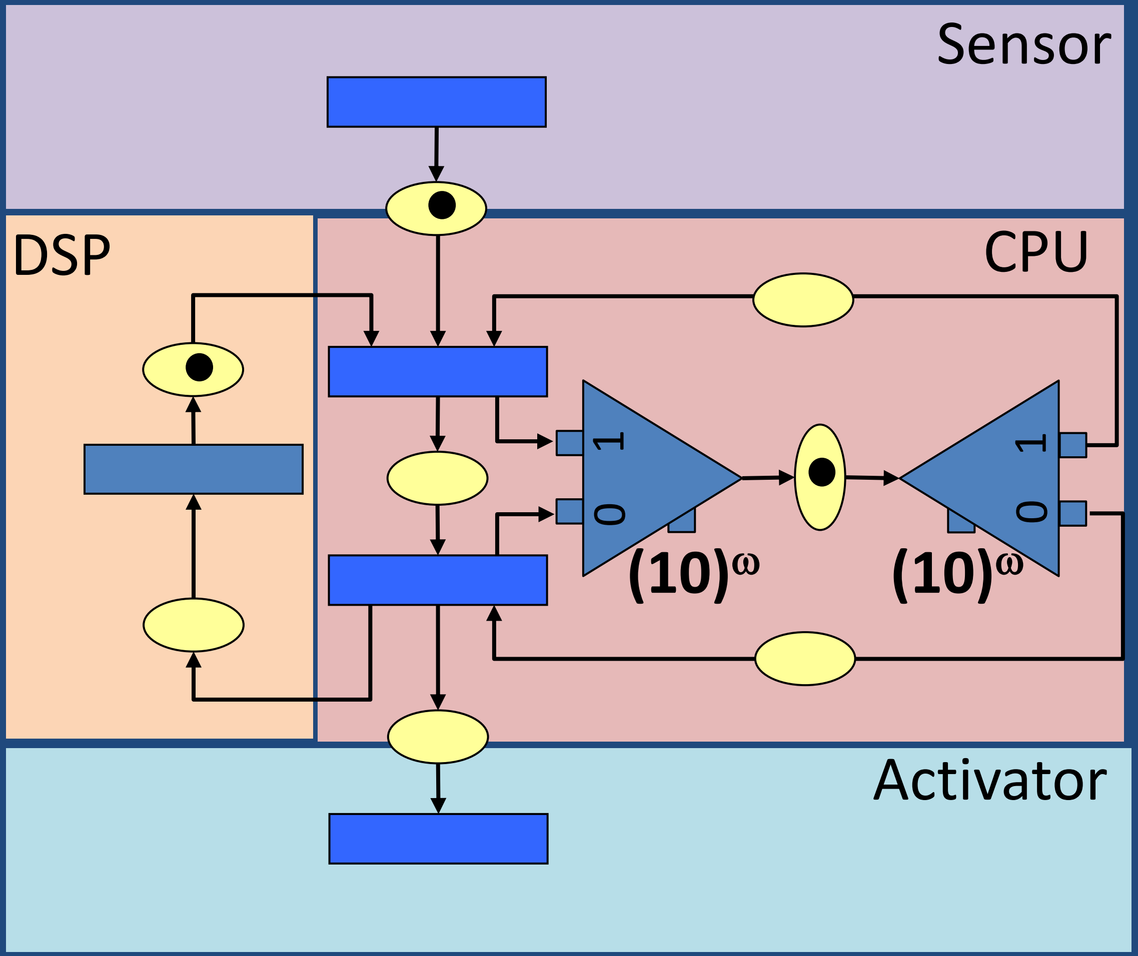




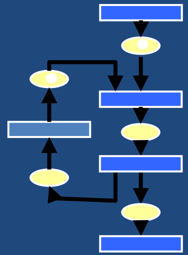


Activator
CPU
Sensor
Bus
DSP

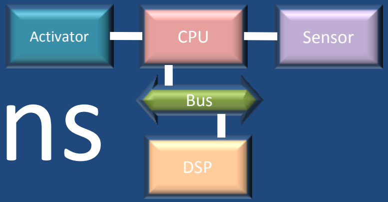
Scheduling

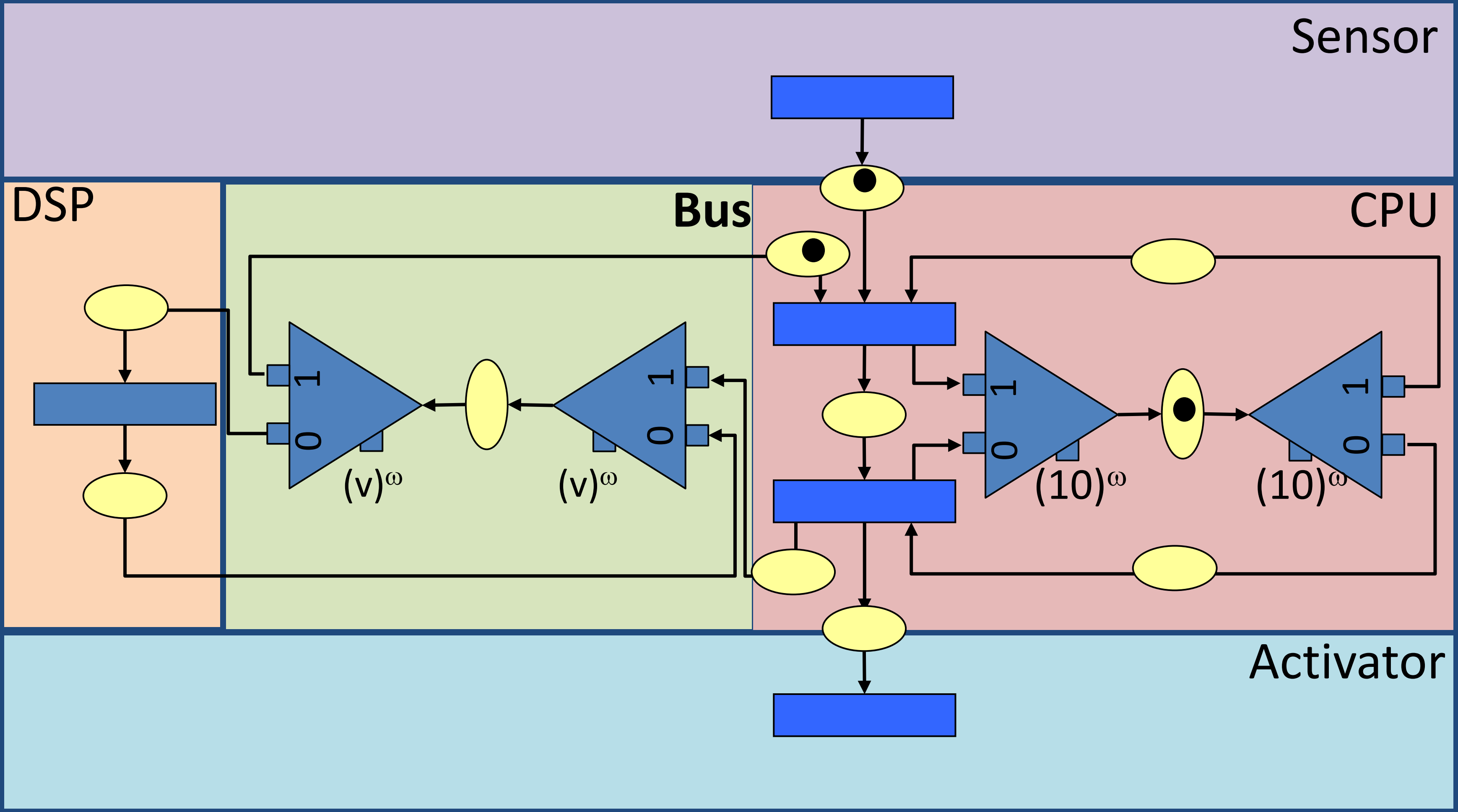


Sensor
DSP
CPU
1
0
(10)ω
1
0
(10)ω
Activator

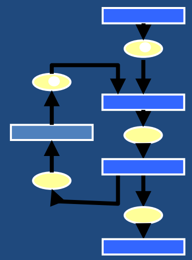


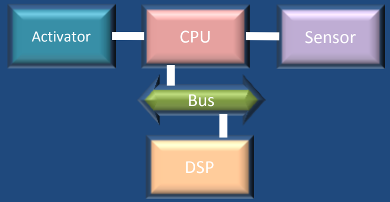
Mapping communications



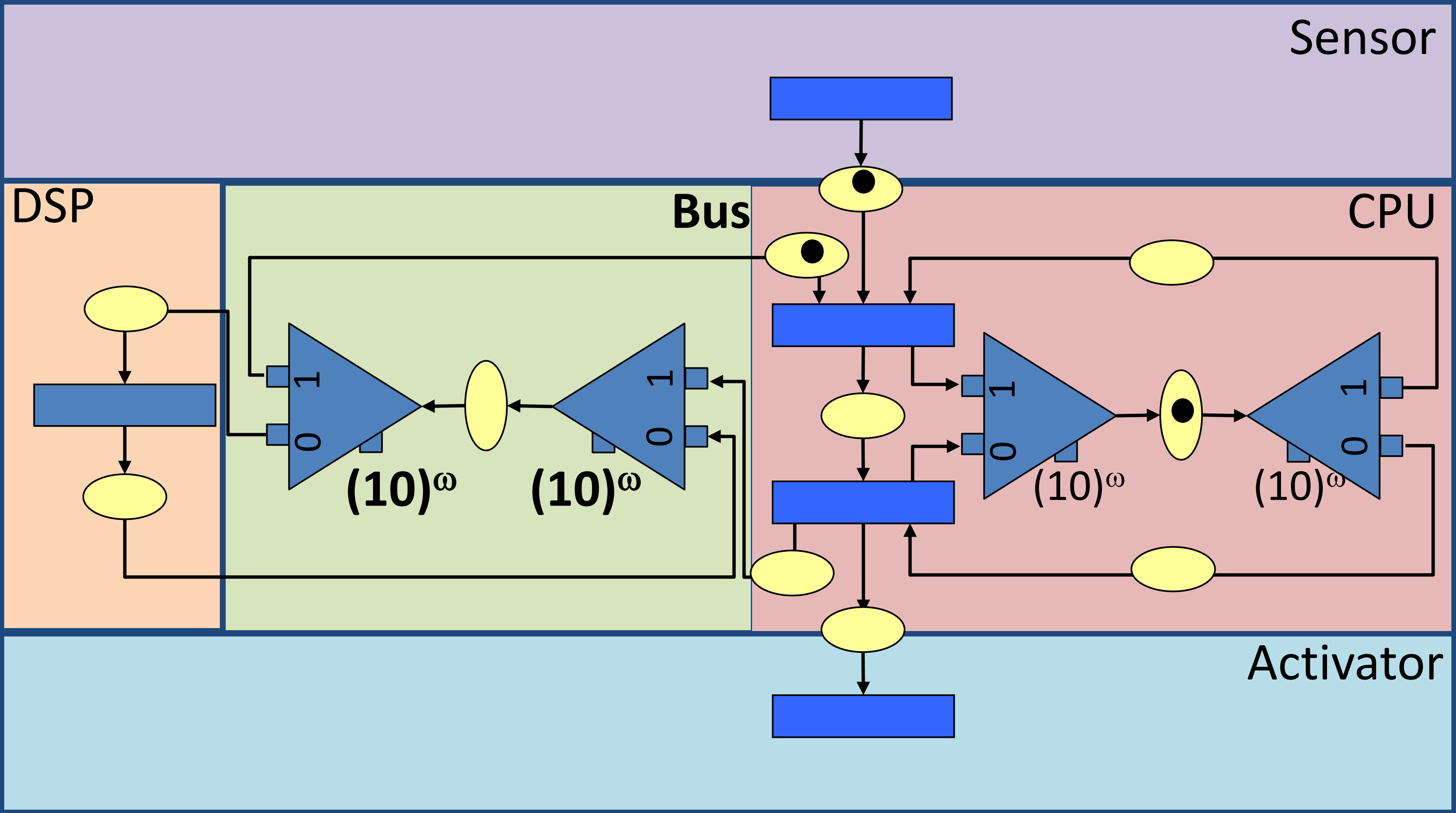


Routing





Activator
CPU
Sensor
Bus
DSP

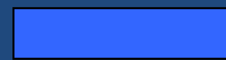
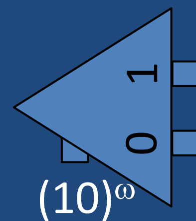


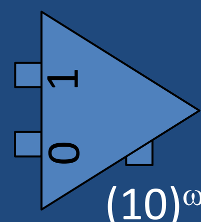
Sensor
DSP
Bus
CPU
1
0
(10)$^\omega$
(10)$^\omega$
1
0
(10)$^\omega$
(10)$^\omega$
Activator

Outline

- Introduction
- The KRG model
- Front end required!
- Case study: All to all propagation algorithm

The KRG model

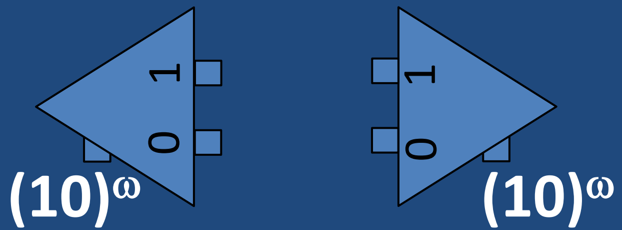
- The transition  consumes and produces on every port
- The select 

1 input/2 outputs
- The merge 

1 input/2 outputs, **blocking**
- The place  1 input/1 output
- The token  A piece of data
- The arc  Relating places and nodes

The KRG model

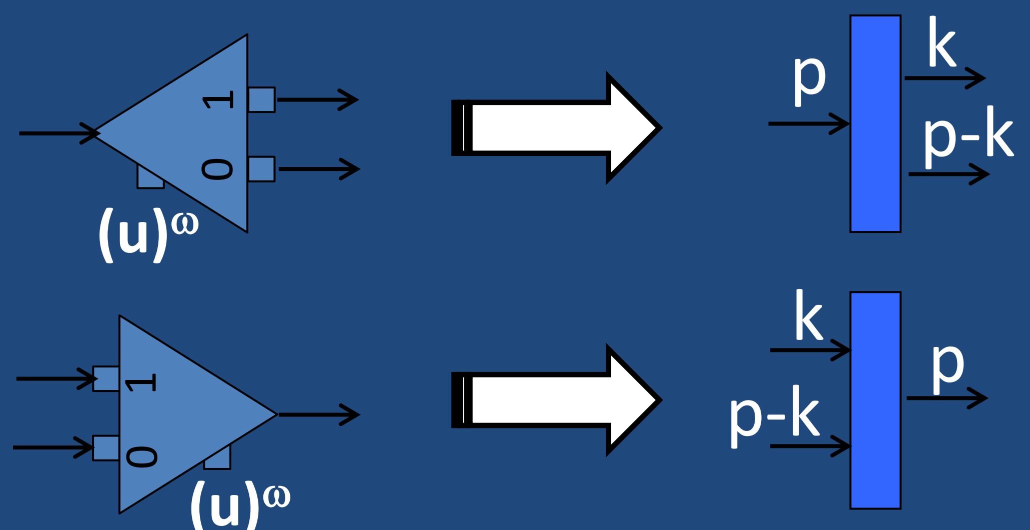
- The routing patterns:
  - Data independent
  - Decided at compile time
  - K-Periodic pattern

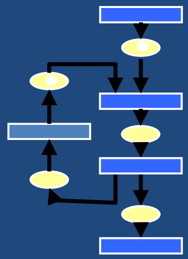


- About the KRG model:
  - Many properties are decidable on the model
    - Safety, (pseudo)-liveness.
    - Timing analysis

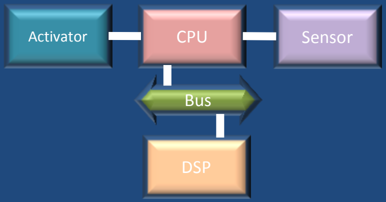
Abstraction in SDF

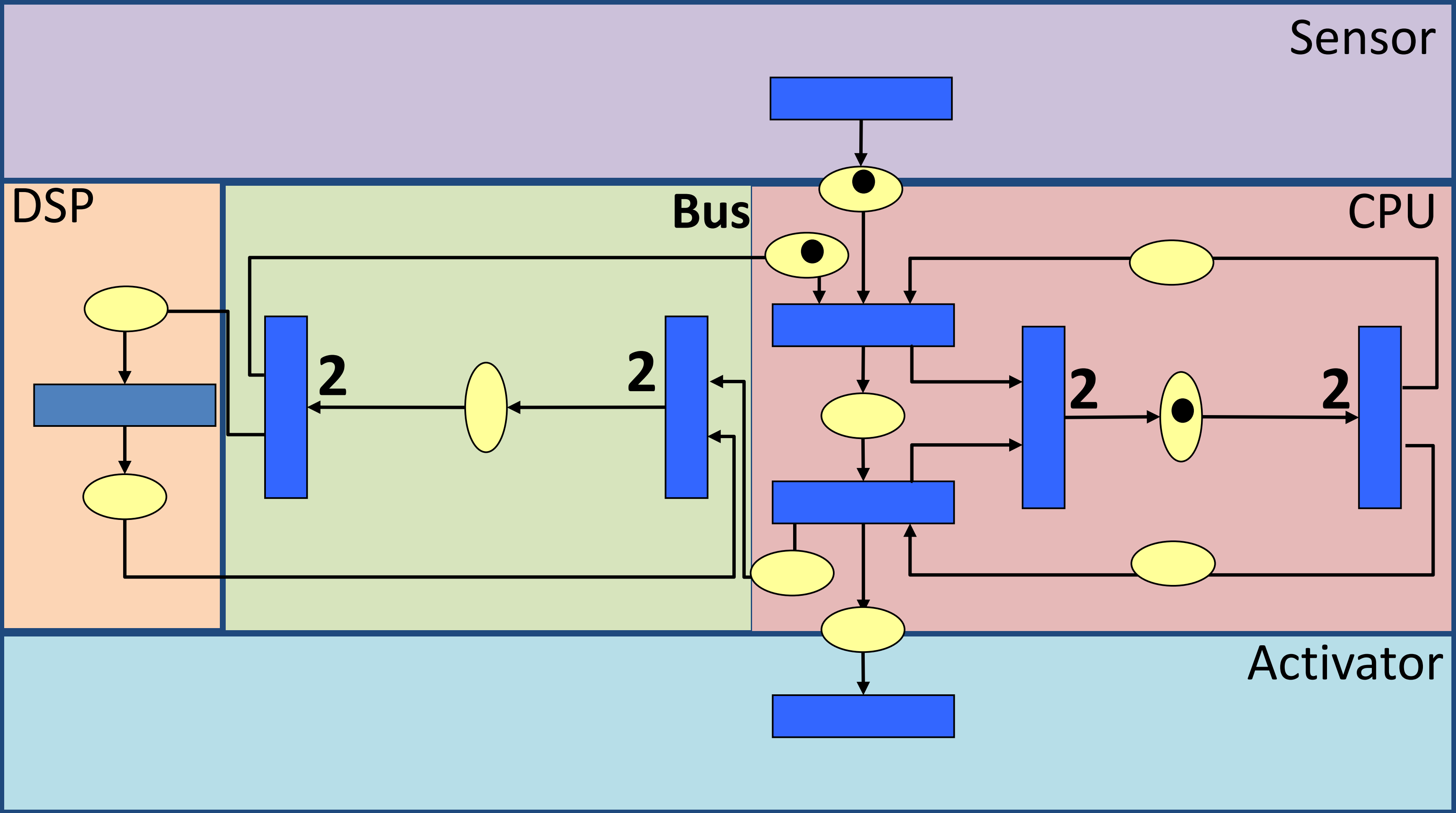
- Any KRG can be abstracted into an Synchronous Data Flow(SDF) graph.
  - Balanceness can be checked
  - Let $u \in \{0,1\}^*$ such that $|u|=p$ and $|u|_1=k$





Abstraction in SDF





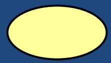
Normal form

- Every KRG has a normal form that preserves its flow and behavior
- Many transformation rules are defined
  - Equivalence between original application and plate-form aware application.

Synchronous KRG

- Let us assume a global clock

After expansion



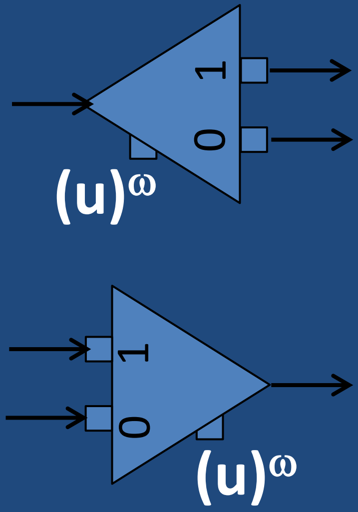
Latency ≥ 0

$\boxed{0 \geq \text{Latency}} \geq 1$

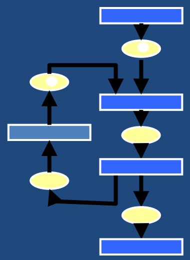
Latency ≥ 0

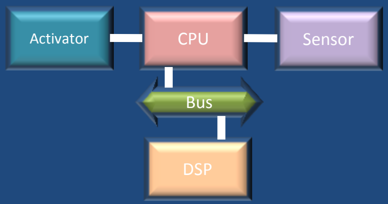
Latency $= 0$

No cycle with 0 latency

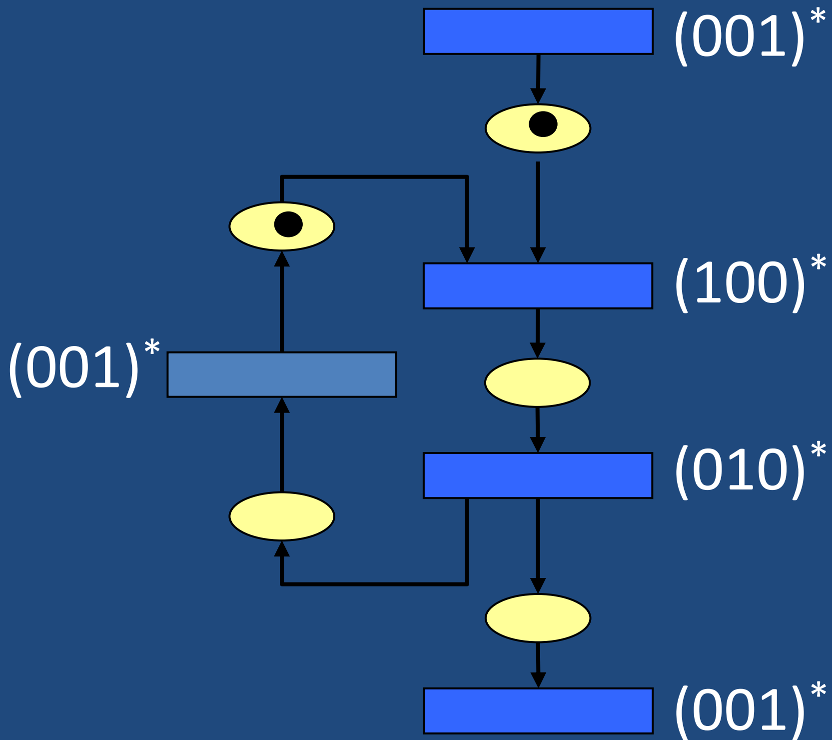


Latency $= 0$

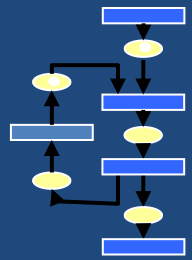


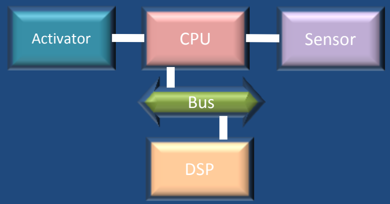


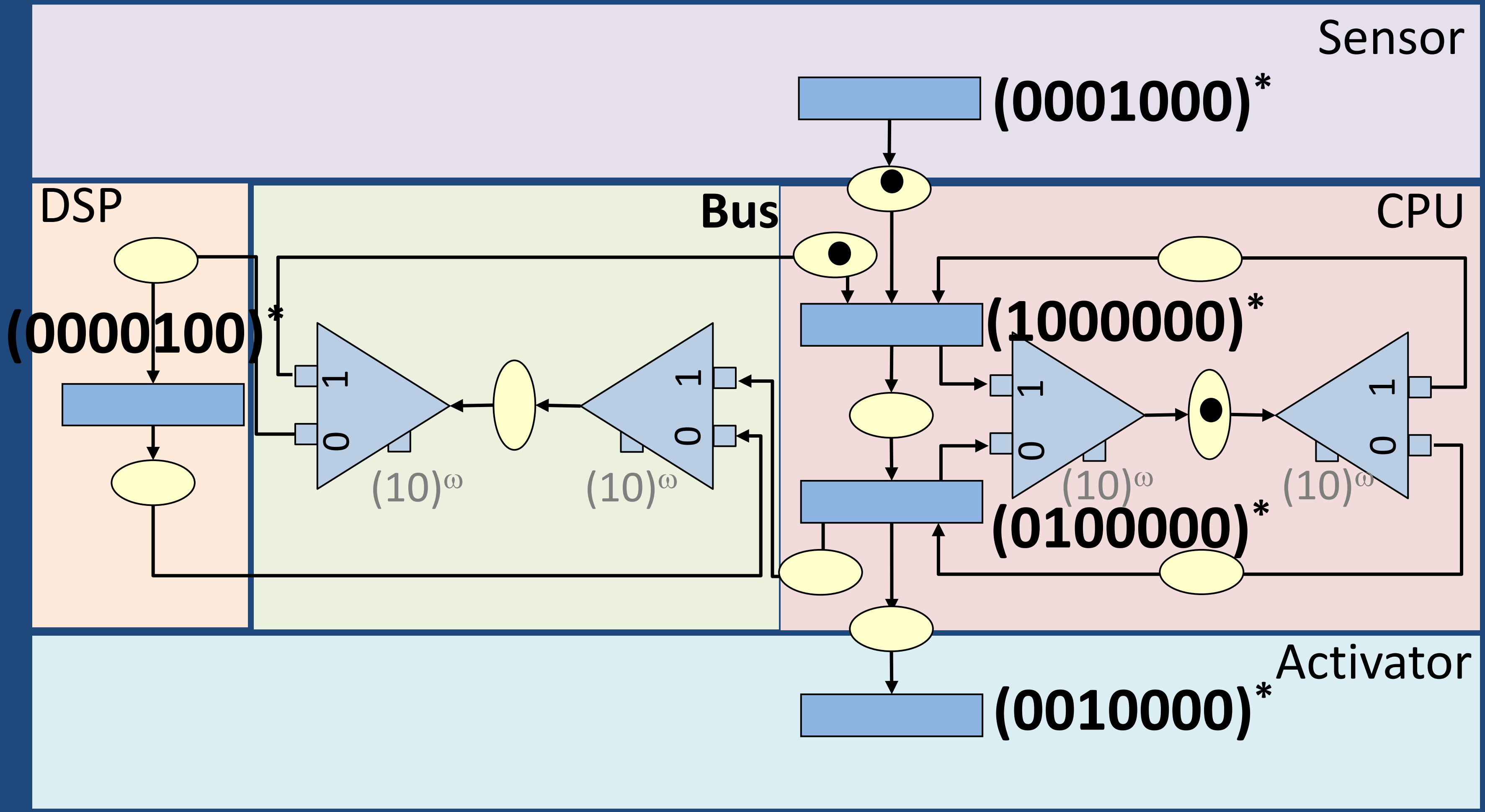
Scheduling

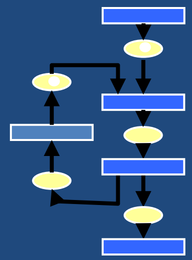


Scheduling

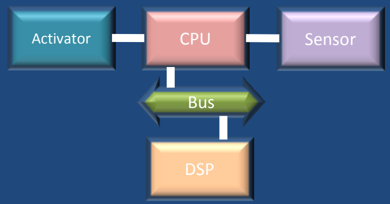



Activator
CPU
Sensor
Bus
DSP

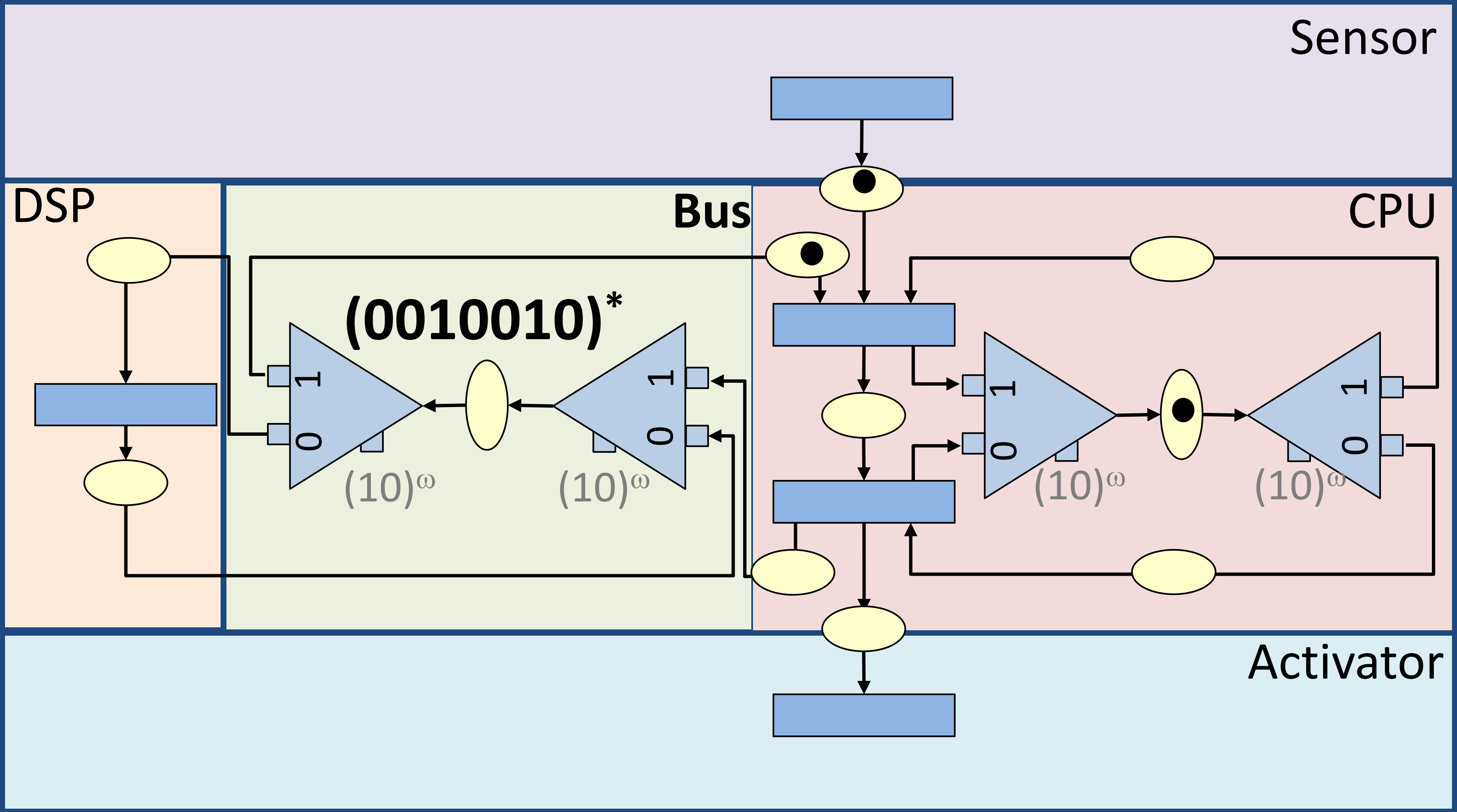

Sensor
(0001000)*
DSP
Bus
CPU
(0000100)*
(1000000)*
(0100000)*
(10)ω
(10)ω
(10)ω
(10)ω
Activator
(0010000)*



Routing



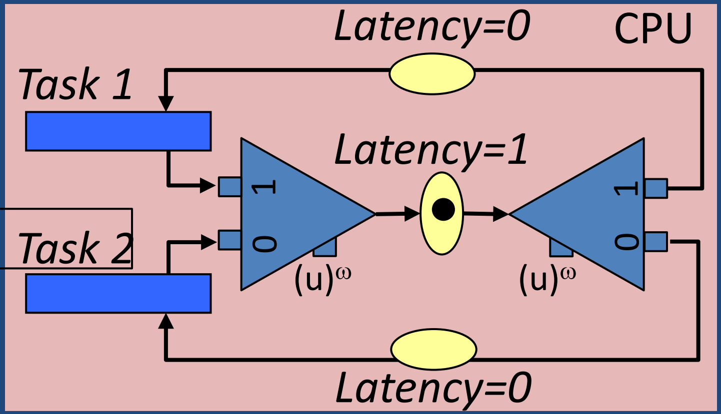
Activator
CPU
Sensor
Bus
DSP



Sensor
DSP
Bus
CPU
(0010010)*
1
0
(10)ω
(10)ω
(10)ω
(10)ω
Activator

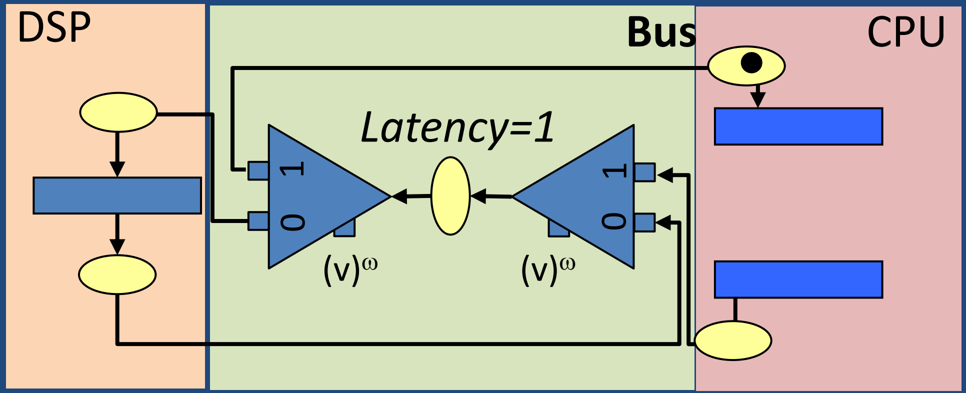
Incremental mapping

- Mapping constraints can be modeled as additional pieces of KRG



Task allocation

Memory allocation?

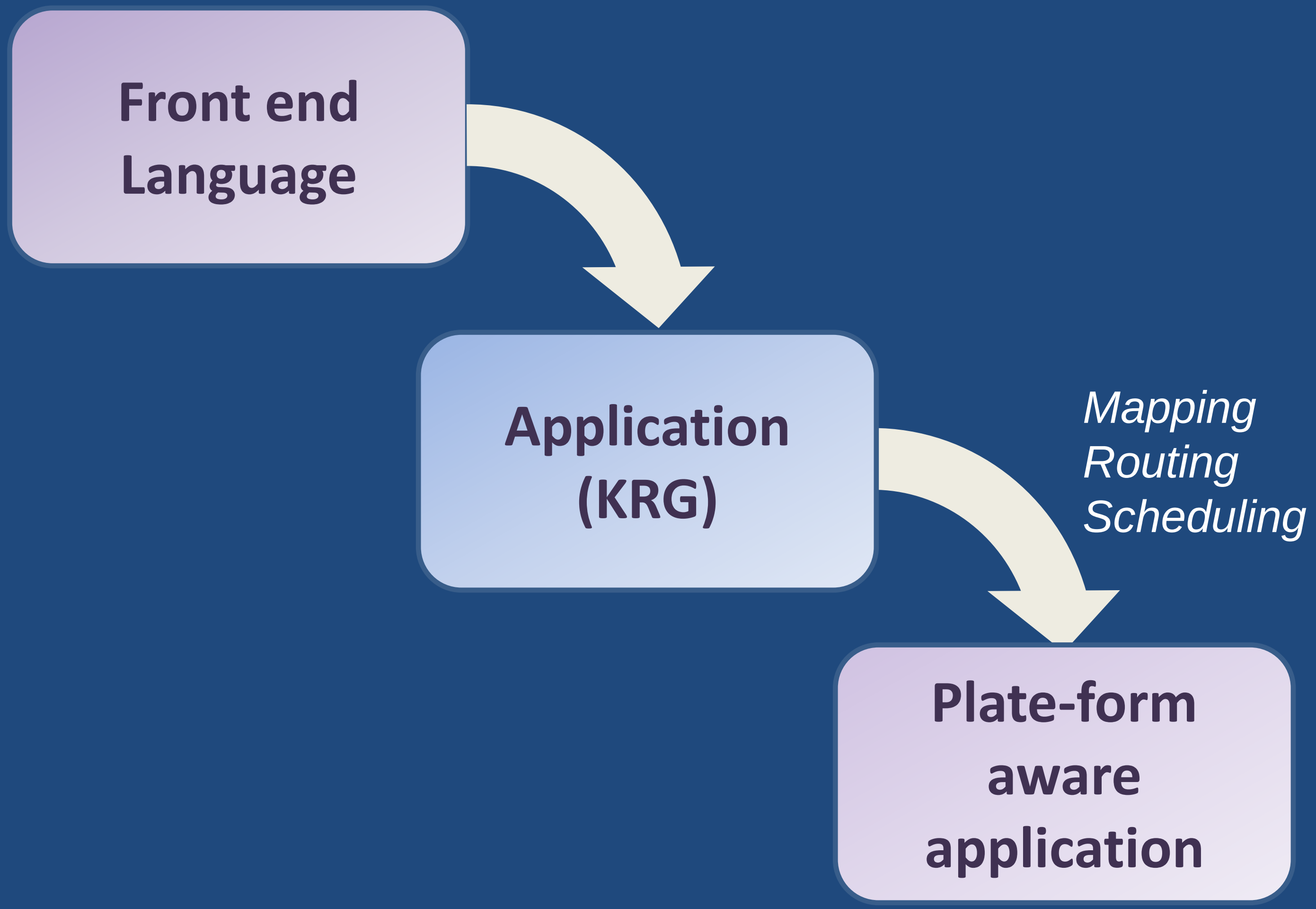


Communication allocation

Outline

- Introduction
- The KRG model
- Front end required!
- Case study: All to all propagation algorithm

Front end required



Front end

- Affine bounded nested loop
  - Stream it
  - Polyhedral process network
  - Compaan
- KRG
  - goes further with non-linear optimization
  - routes communications

Front end

- CCSL
  - Clock Constraint Specification Language
- Specification of the system
  - Introduction of the plate-form constraints
  - Capture the resulting switching conditions and schedules
- KRG and associated methodology would be the solving engine.

Outline

- Introduction
- The KRG model
- Front end required!
- Case study: All to all propagation algorithm
  - Routing communications in a NoC

Refining cellular automata with routing constraints

Jean-Vivien Millo, Robert de Simone
INRIA Sophia-Antipolis

Why

- In a CA, communications are free

- In the implementation of a CA on a multicore architecture,
  - communication are not free

- Game of life with neighborhood 2 and a grid of size 10: 2400 messages are exchanged.

Today

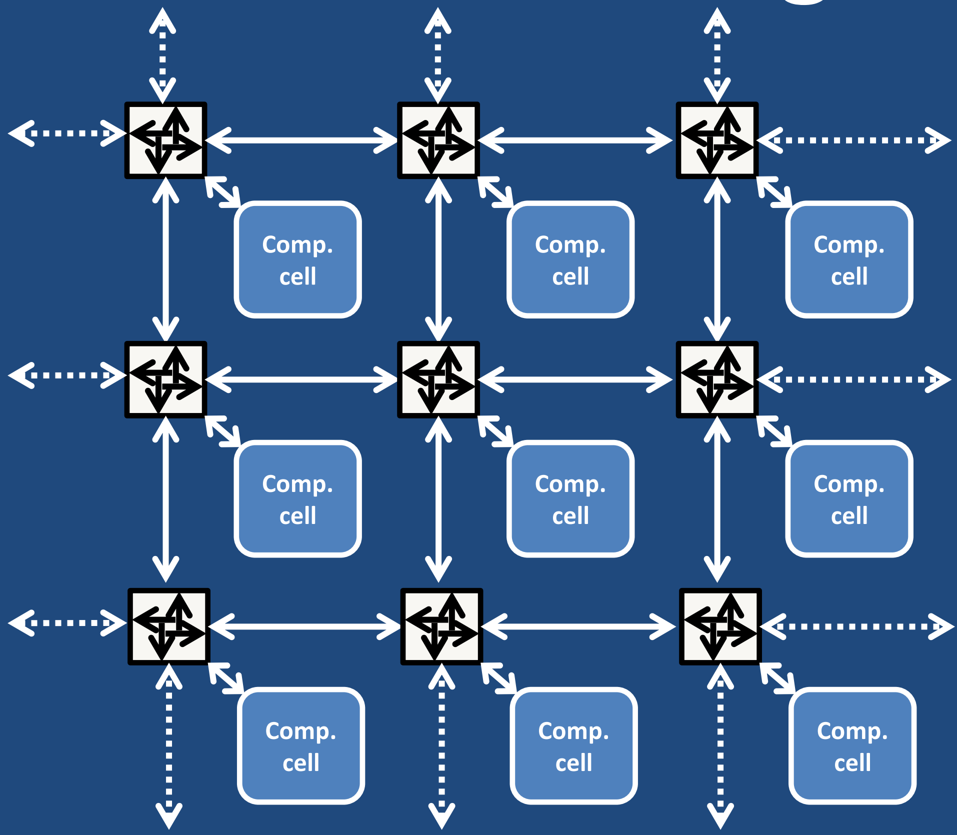
- Motivating example
  - Neighbor Broadcasting Algorithm
- How to perform routing:
  - Extract the routing directives from behavioral analysis of the algorithm

Cellular Automata

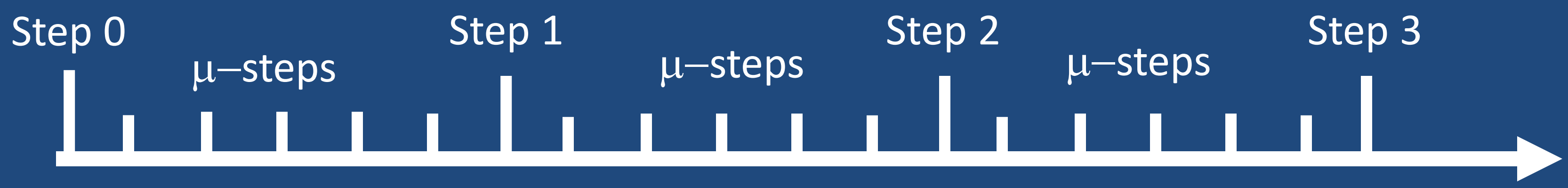
- Synchronous, infinite but periodic
- rectangular grid of dimension 2

- Manhattan distance: $|x_a-x_b|+|y_a-y_b|$
- Moore distance: MAX $(|x_a-x_b|, |y_a-y_b|)$
- Neighborhood $N^n(c)=(c_1/\ ||c-c_1||_\infty \leq n)$
- The radius is *n*

CA + routing



- Refinement of the timeline:



The router

Algorithm 1 describes the generic behavior of a router

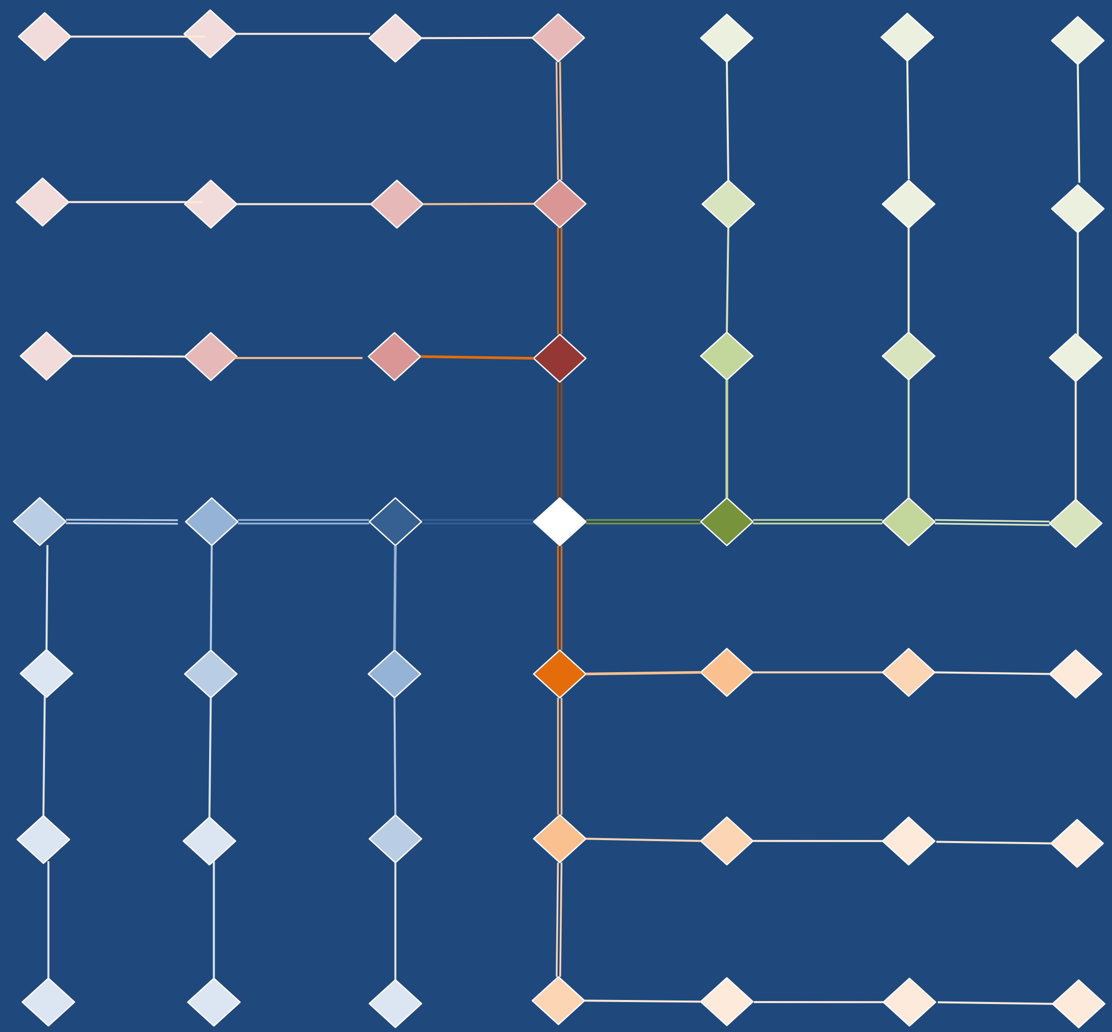
```
while true do
    in_ports = feeding_rule()
    for all in_port ∈ in_ports do
        data = in_port.read()
        destinations = propagation_rule(data)
        for all out_port ∈ destinations do
            out_port.write(data)
        end for
    end for
    pause
end while
```

- How to compute:
  - The feeding rule?
  - The propagation rule?

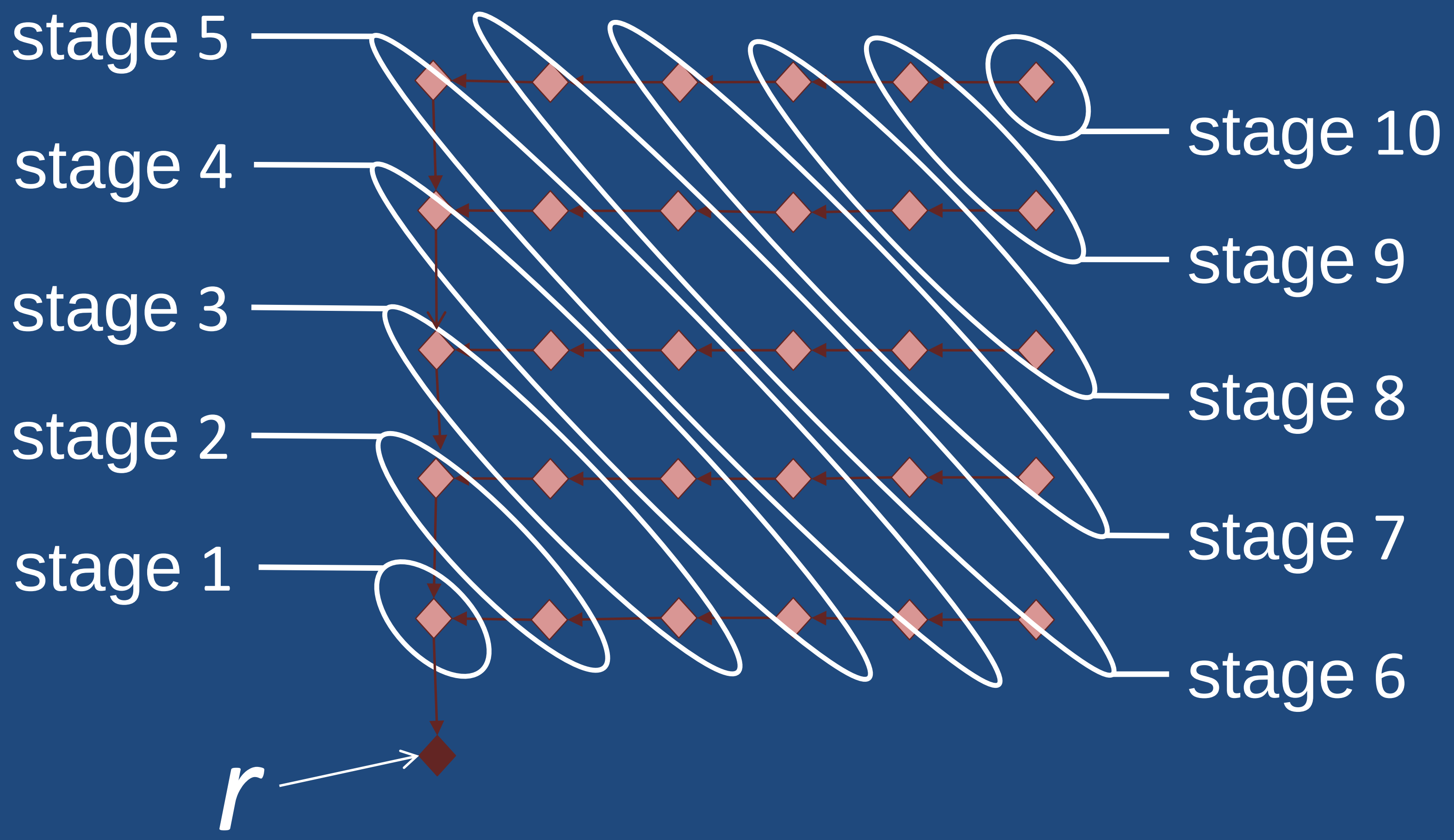
Neighbor Broadcasting Algorithm

- Propagate the current state to all our neighbors up to a radius *n*.
  - Using a predefined propagation pattern
  - Using multicasting
- All the cells do the same simultaneously

Propagation pattern

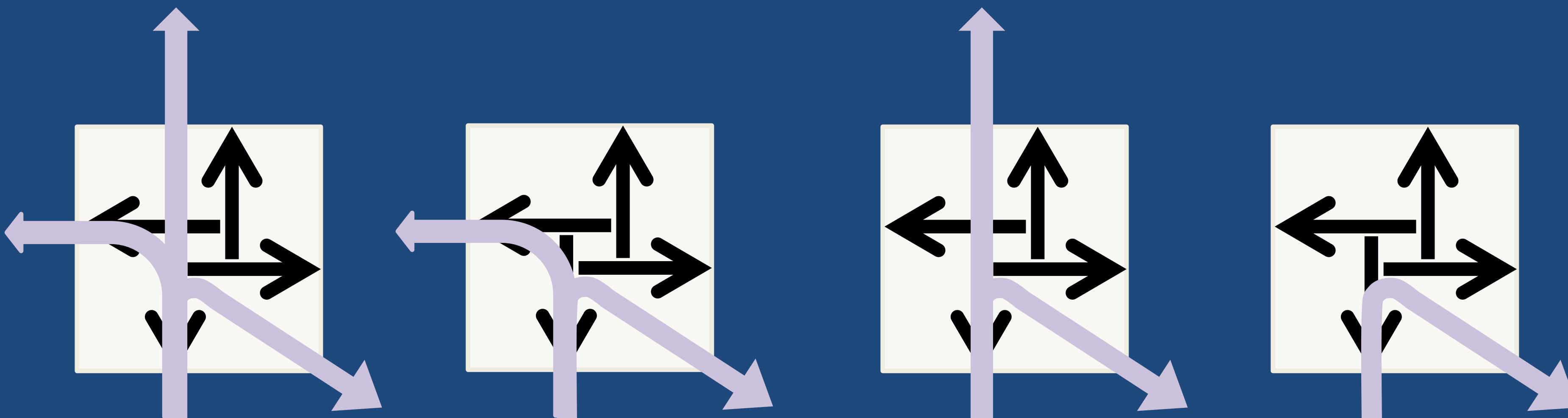


NBA μ–step by μ–step



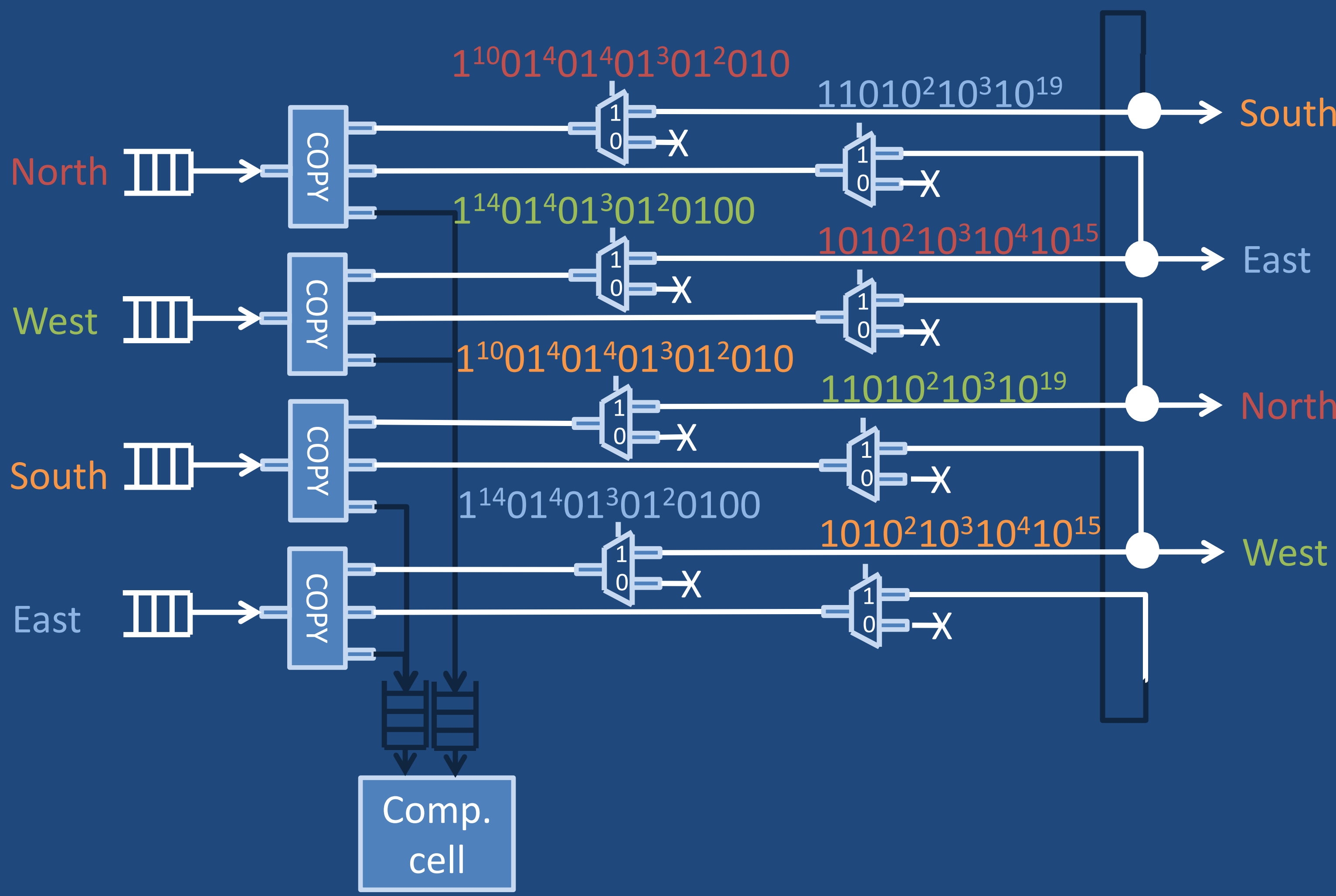
The router during the NBA

- The 4 propagations patterns



- Opposite sources do not interfere
  - Opposite sources can be processed in parallel
  - Here is the feeding rules

The propagation rule



Experimental results

- Simulation in System C
  - of the NBA

- For a radius *n*
  - Execution time is $2 \times n \times (n+1)$
  - Buffer size is *n*

- *Stencil application*

Conclusion

- We have presented an extension of CA with routing constraints
- We have illustrated our approach with the NBA
- Possible future directions:
  - Dimension > 2
  - Asynchronous CA
  - What if the CA is bigger than the NoC

Thank you

References

- [1] Bart Kienhuis, Ed F. Deprettere, Pieter van der Wolf, Kees A. Vissers: A Methodology to Design Programmable Embedded Systems - The Y-Chart Approach. Embedded Processor Design Challenges 2002: 18-37