Interaction between delays and high-level transformations in Array-OL

Calin GLITIA

in collaboration with
Pr. Pierre BOULET

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

Outline

- Introduction
- Array-OL model of specification
- High-level transformations
- Inter-repetition dependence extension
- Interaction between the dependences and the transformations
- Conclusions

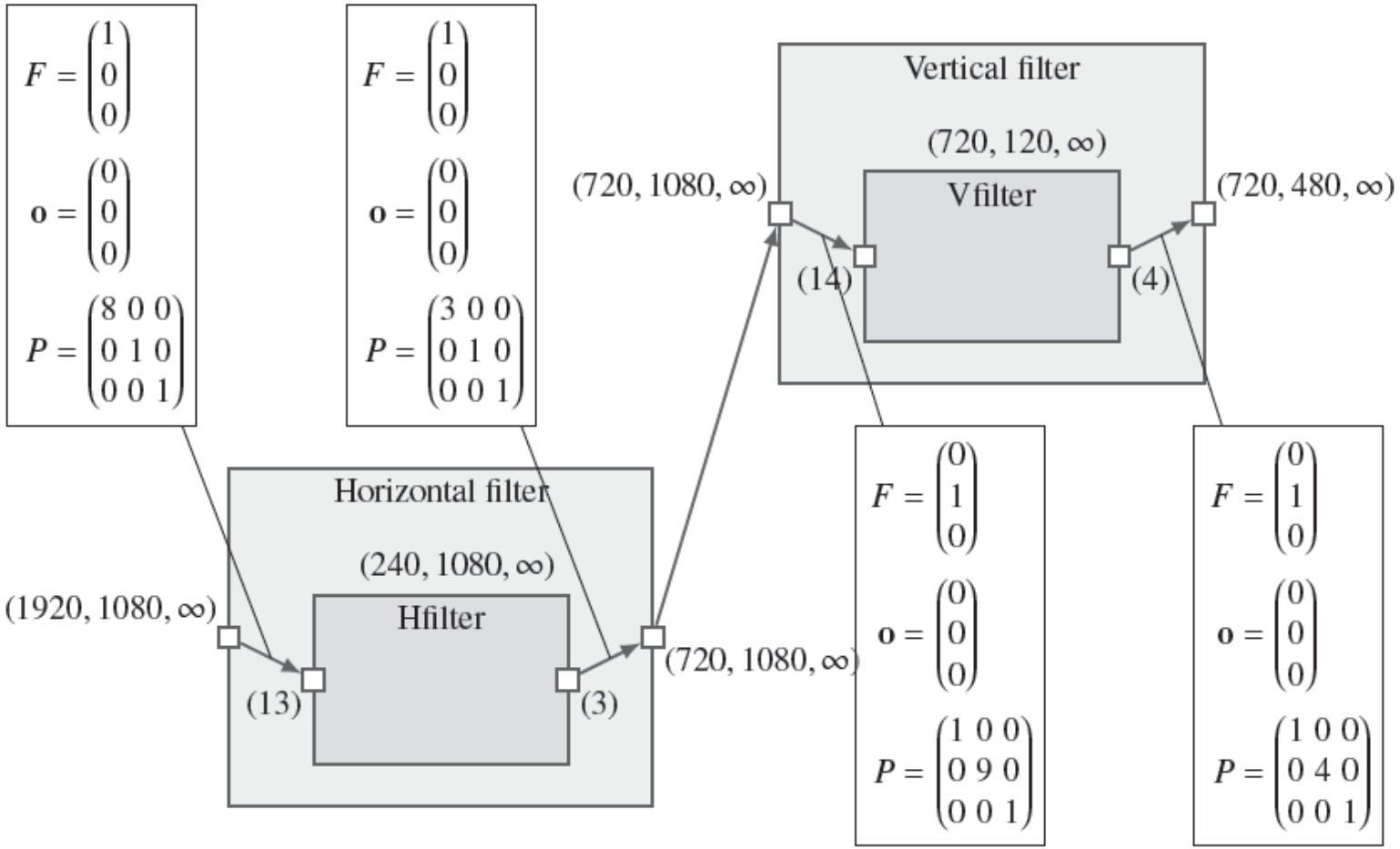
Introduction

- Computation intensive **multidimensional** applications
- A good specification language:
  - Expressive
  - Without compromising the usability
  - Allows a static schedule on parallel architectures
- Expressivity:
  - Access via sub-arrays
  - Support for sliding windows
  - Cyclic data accesses
  - Several sampling rates
  - Hierarchical constructions
  - *Express delays and state constructions*

Array-OL model of specification

- Graphical-textual language
- Focused on expressivity
- Separate the specification from execution
  - Uniform data processing
  - Time as dimension(s) of the data-structures
  - Single assignment formalism
- Express maximum of parallelism
  - **Task parallelism**
    - Hierarchical component-based model
  - **Data parallelism**
    - Repetitions (parallel by default)
    - Access to sub-arrays (patterns) expressed by *Tilers*

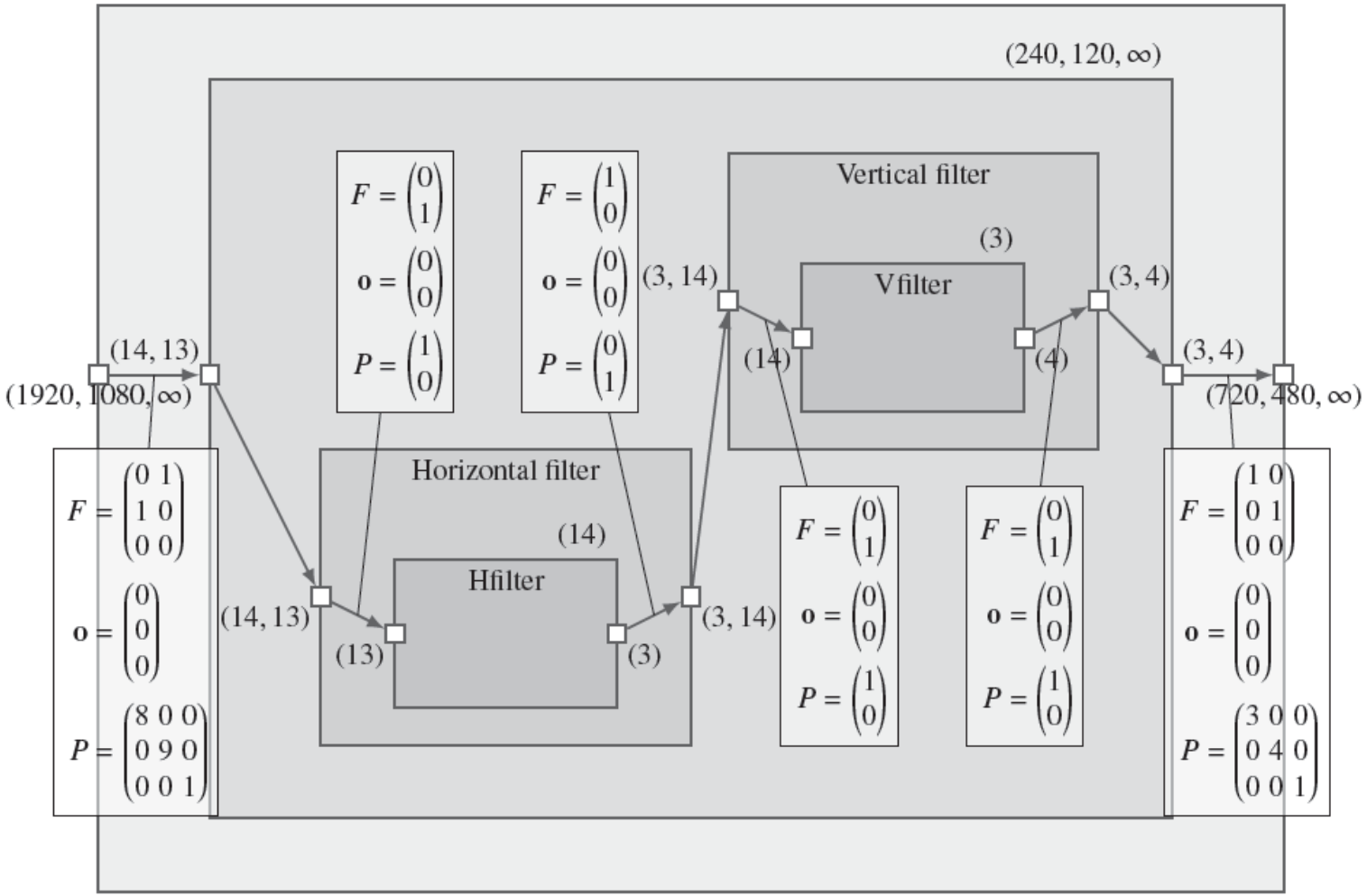
Downscaler Example



Array-OL transformations

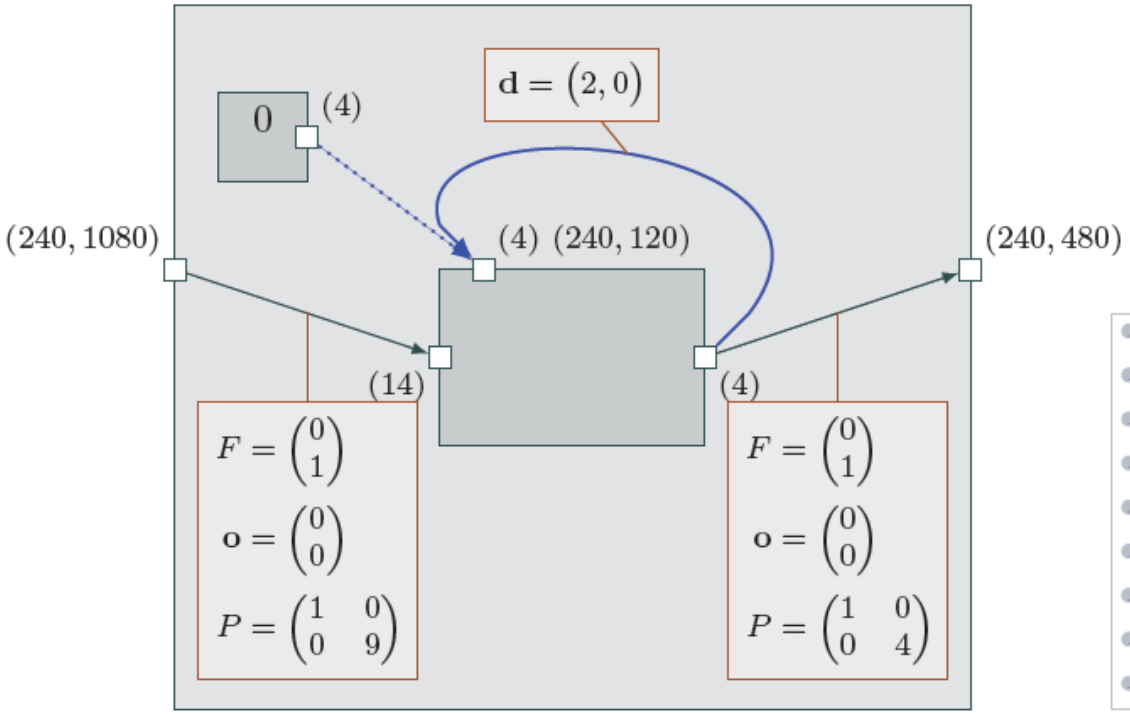
- Applicative optimizations
  - Reduce intermediary arrays
  - Increase task parallelism
  - Allow a pipeline execution
- Fusion transformation
  - Compute common repetition
  - Minimize intermediary array
  - Hierarchy creation
- Other transformations
  - Adjustment of the granularity of parallelism
    - Change paving, Tiling, Collapse
  - By a redistribution of repetitions through the hierarchy levels

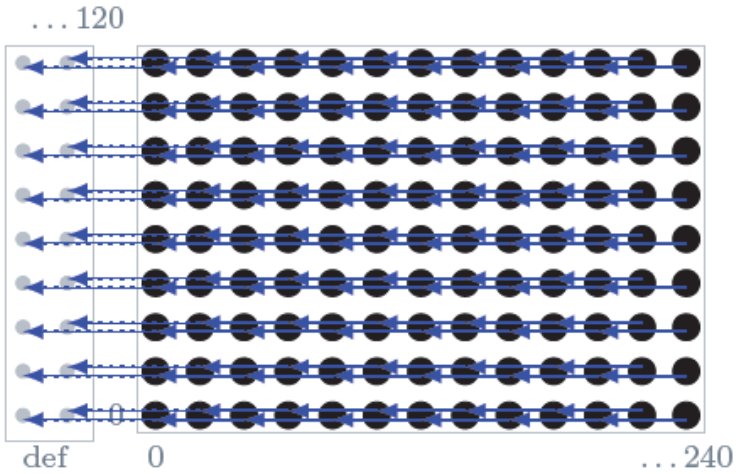
Downscaler after fusion



Inter-repetition dependences

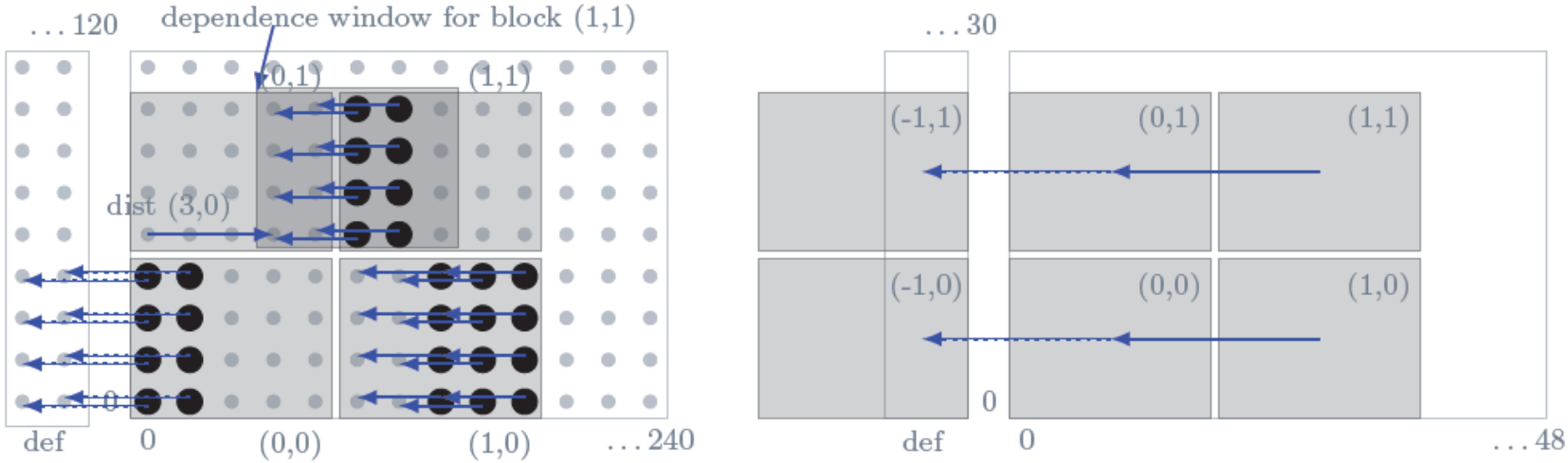
- Uniform dependences between repetitions
  - Dependence vector
  - Default values connector
    - Eventual *Tiler* - different default values for different repetitions



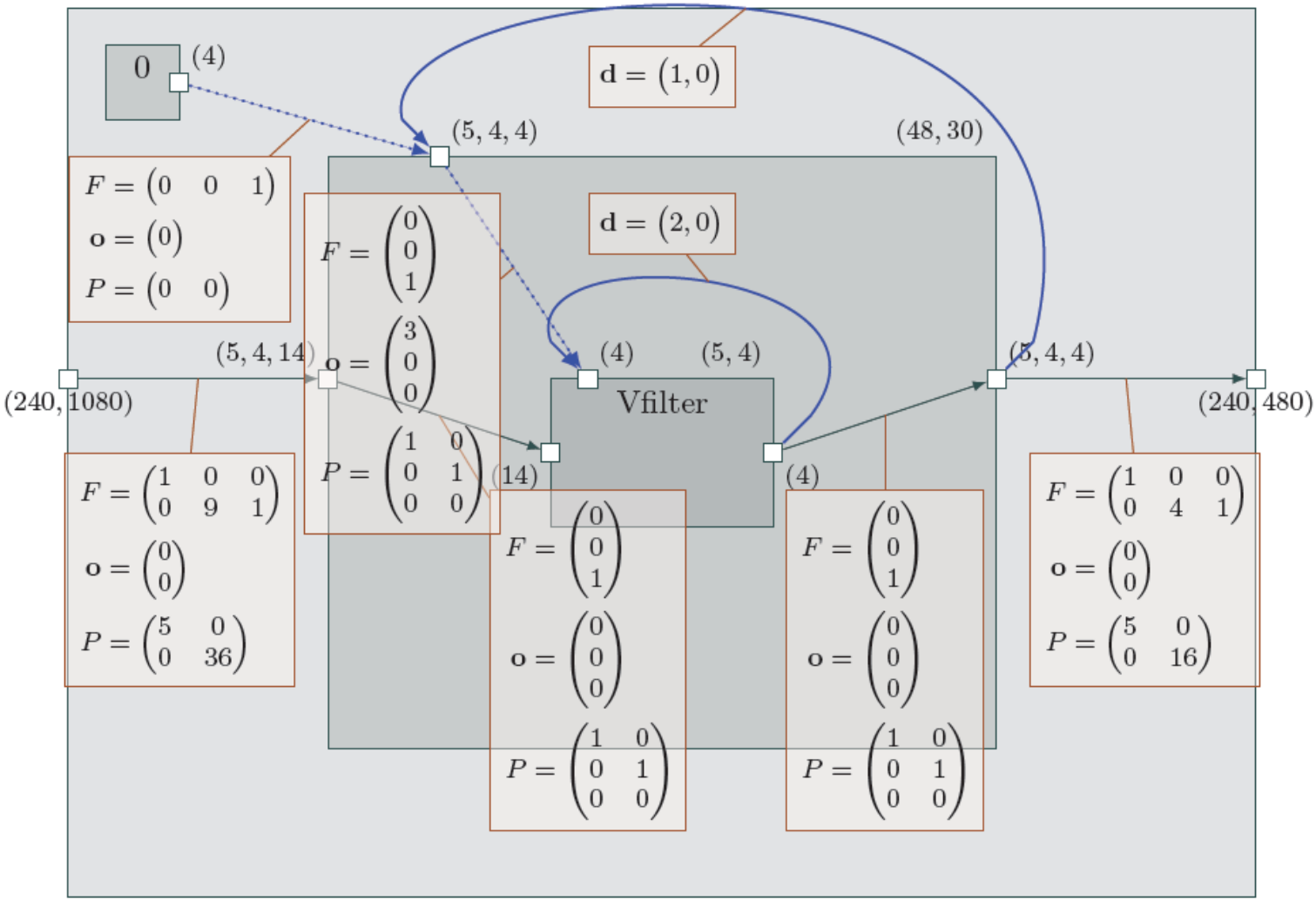


Example of tilling

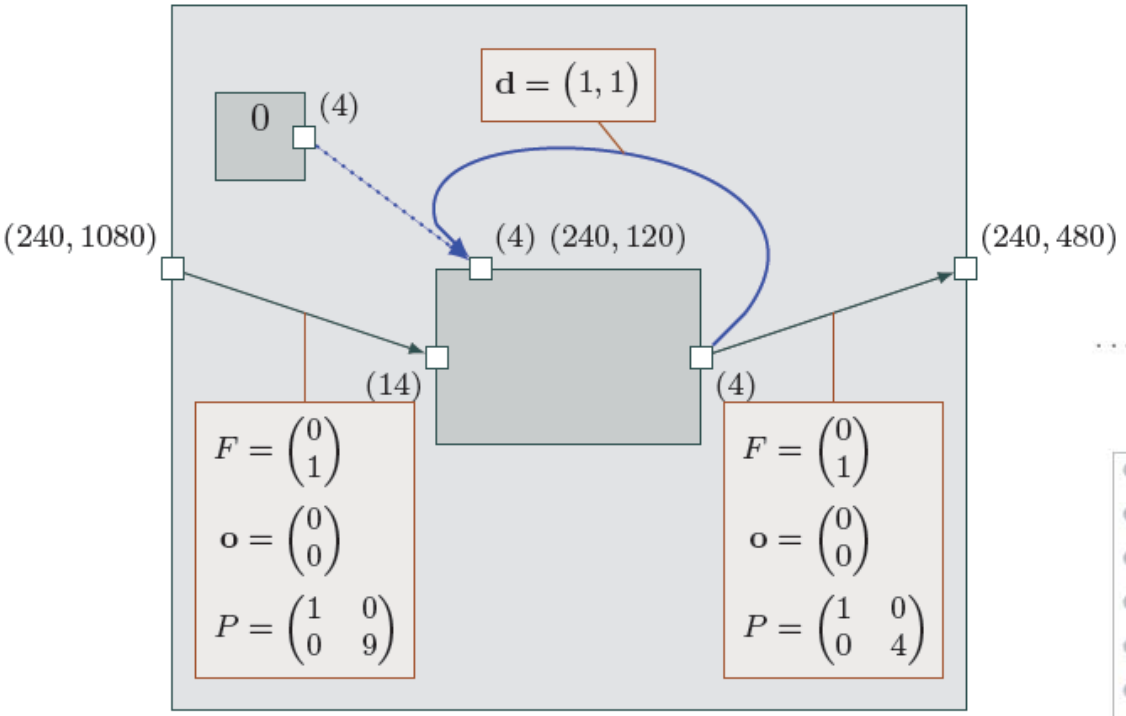
- Splitting the repetition space into blocks of (5,4)
  - Initial depending repetitions on the same hierarchy level
  - Result: depending repetitions in different blocks
- How to express such dependences?
  - Connections between dependences on different levels

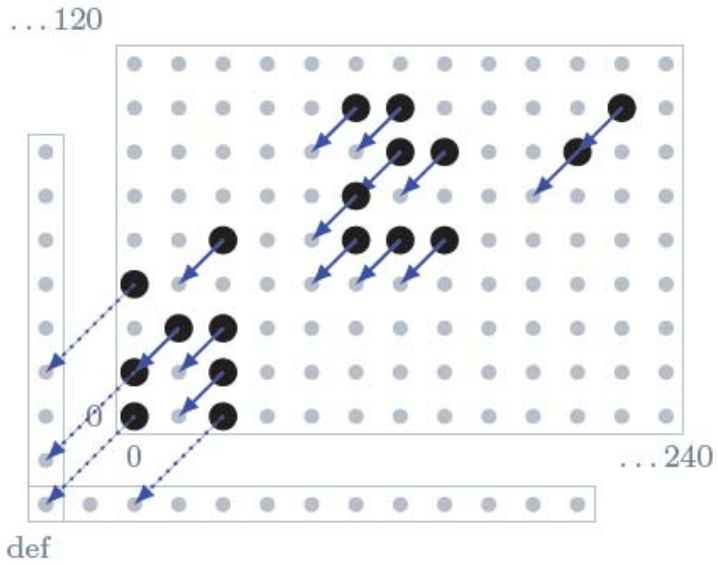


Inter-repetition dependences connected through the hierarchy



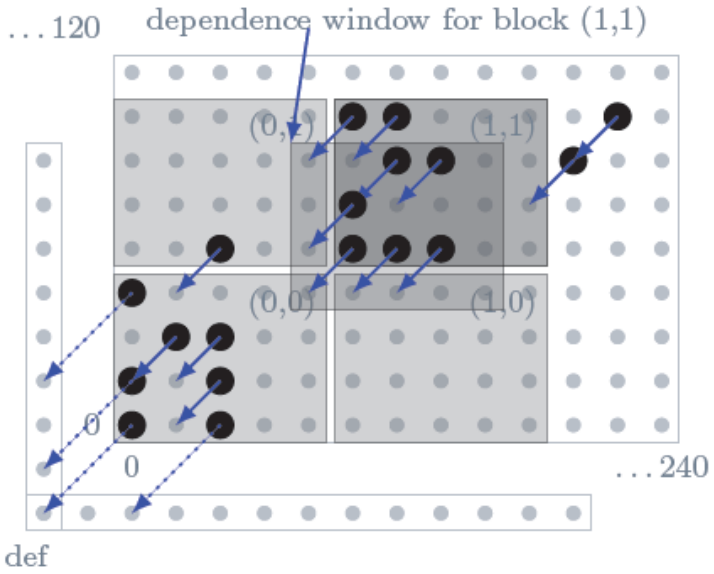
Multiple dependences/default connectors example: before transformation

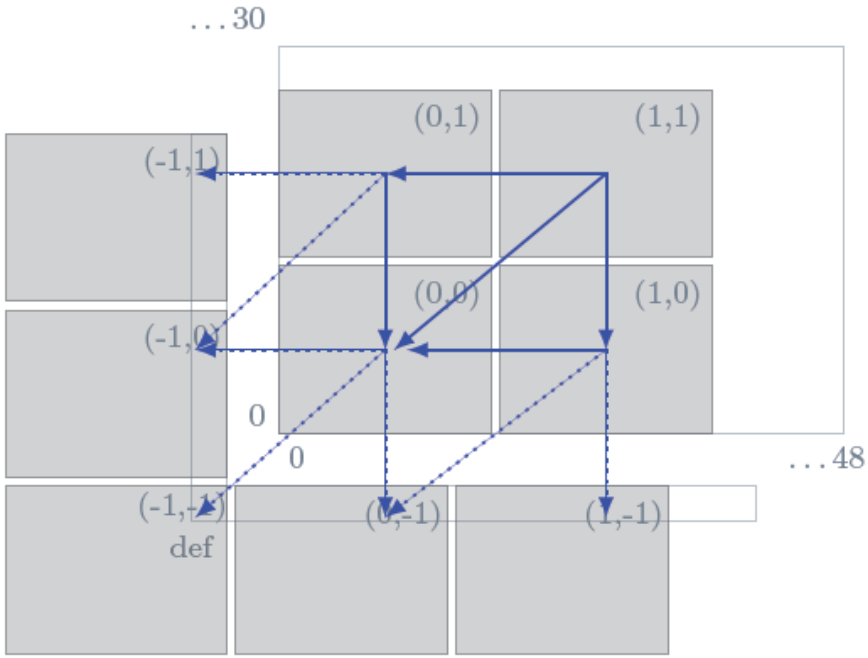




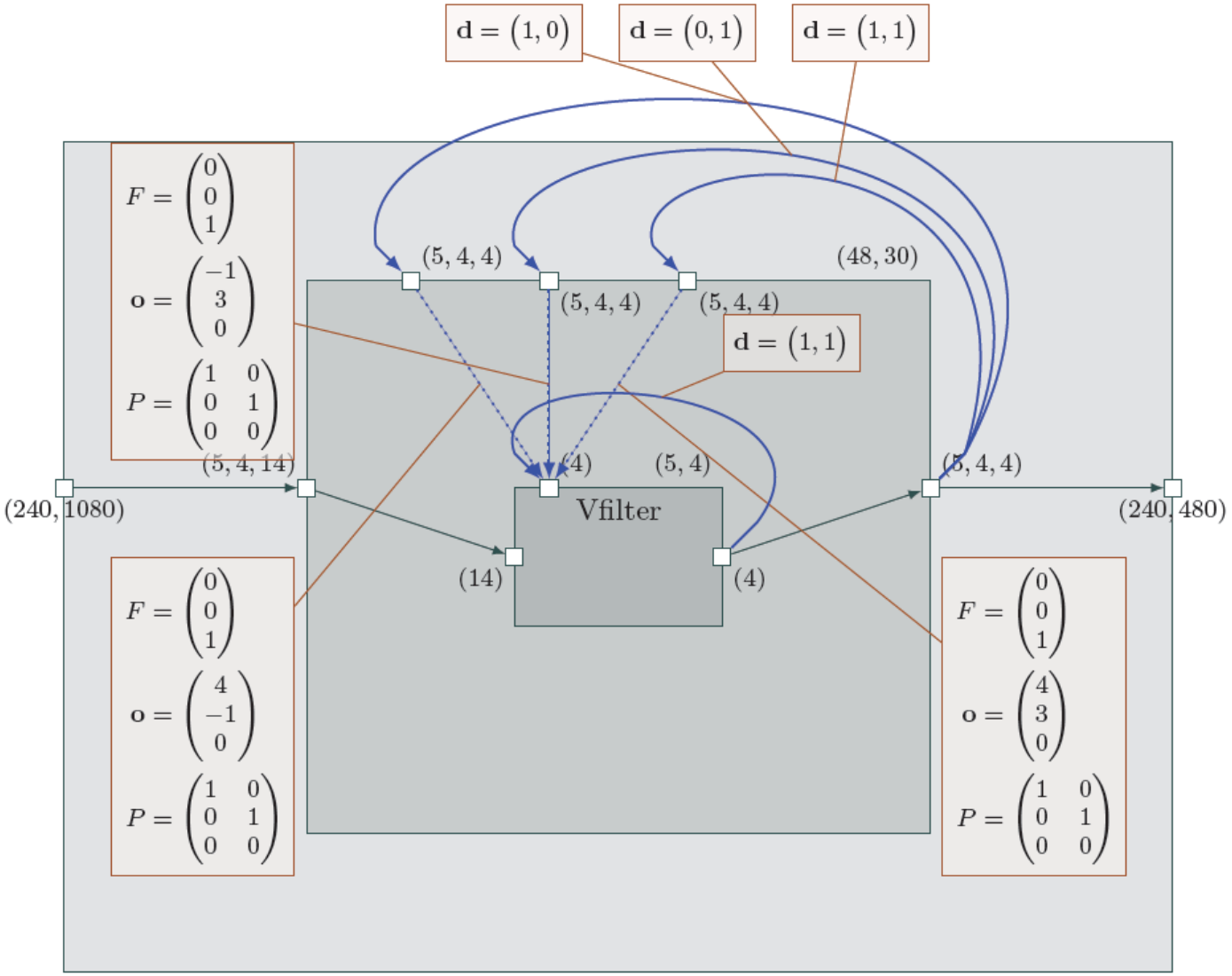
Multiple inter-block dependences

- Dependence window intersects three other blocks
  - Multiple dependences on block level
  - Multiple default connectors on lower level
- Choosing the right dependence done through *Tilers*





After transformation



Interaction with the transformations

- Array-OL transformations can be seen as a redistribution of repetitions between hierarchy levels
- A transformation effects only maximum two successive hierarchy levels:
  - Interchange of repetitions between the two
  - Hierarchy creation
  - Hierarchy suppression
- Automatic post-transformation stage to adapt the dependences
- Impact on inter-repetition dependences
  - Dependence creation/suppression
  - Dependence vector modification
  - Dependence relocate

Conclusions

- Inter-repetition dependence extension
  - Can express delays and state constructions
  - Complex dependences through the hierarchy
  - Model state machines for control
  - Repeated interconnected architectures (grid of processors)
- Reduced impact on Array-OL transformations
  - Facilitate the implementation
- Marte OMG (Modeling and Analysis of Real-Time Embedded Systems)
  - Standard UML Profile
  - Gaspard2
  - Model Transformations
  - Code Generation