Technische Universiteit
Eindhoven
University of Technology

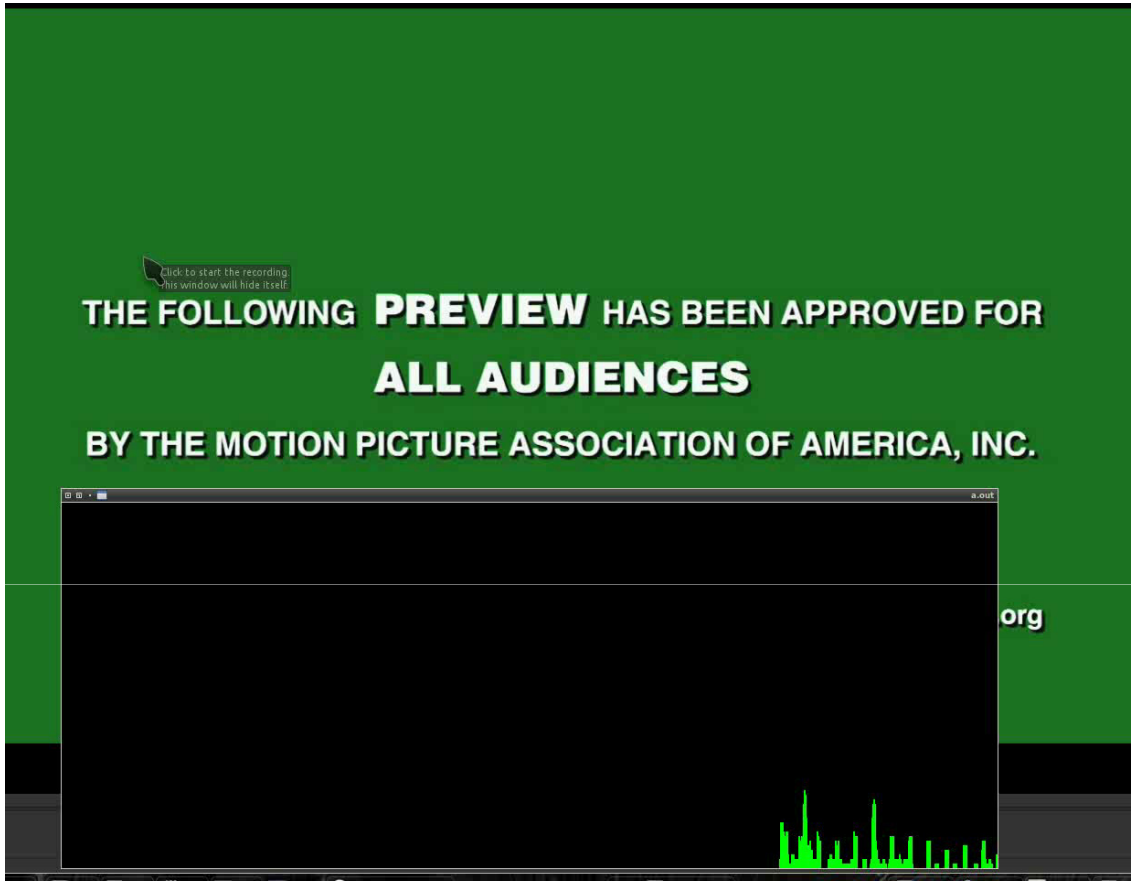
Mapping applications to predictable MPSoCs

Sander Stuijk, Marc Geilen, Twan Basten

CODES+ISSS 2011 tutorial
Taipei, Taiwan

Department of Electrical Engineering
Electronic Systems

Embedded streaming systems



Application trends

Dynamism

Concurrency

Predictability

The timing behavior of an application can be guaranteed independent of other applications running in the system

[Thanks to Martijn Koedam]

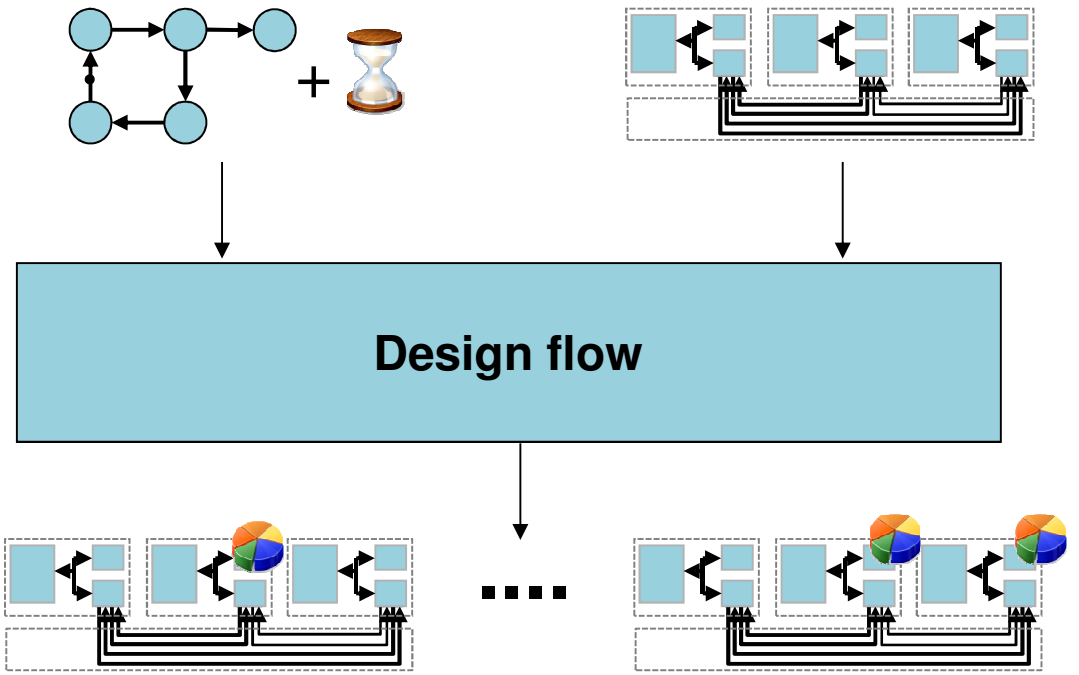
Model-based design

TU/e

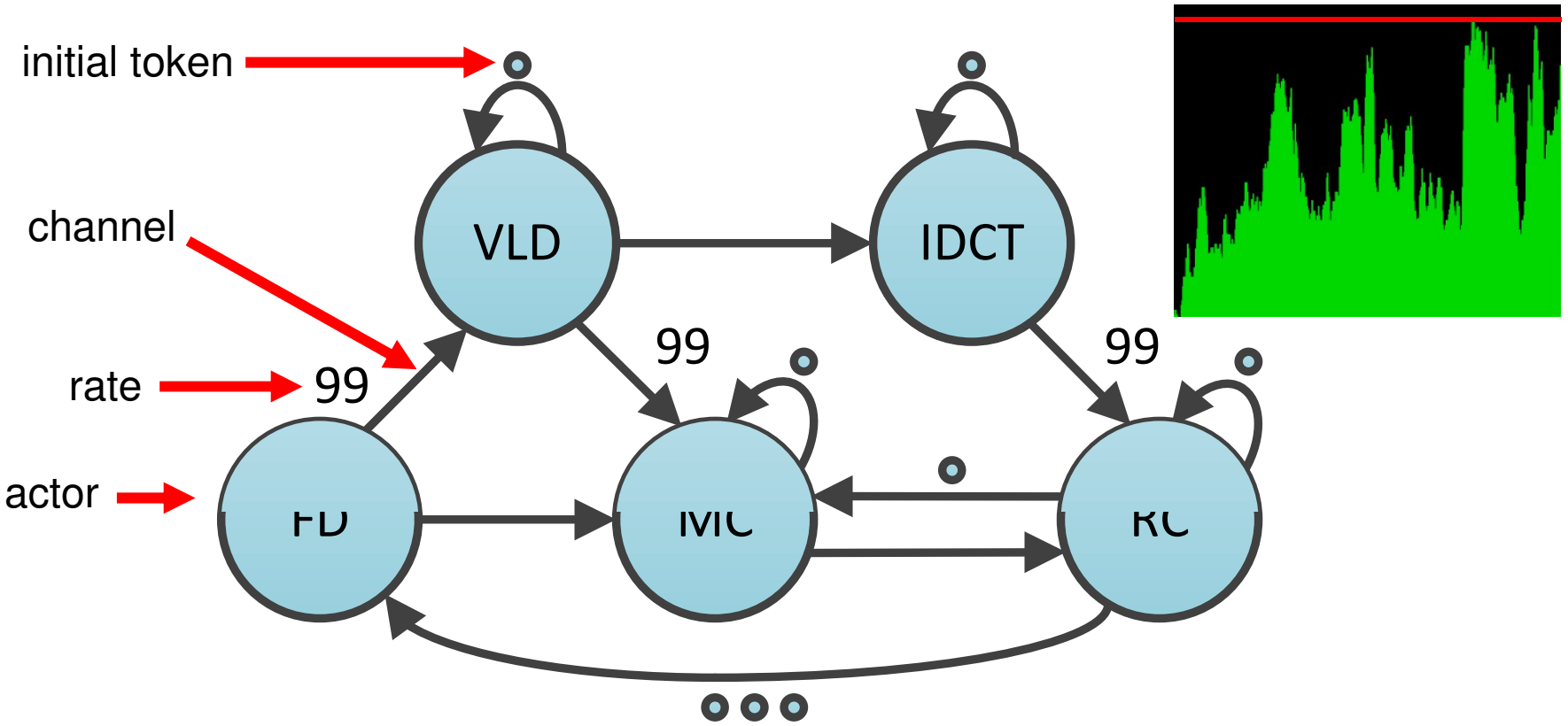
Modeling

Analysis

Implementation

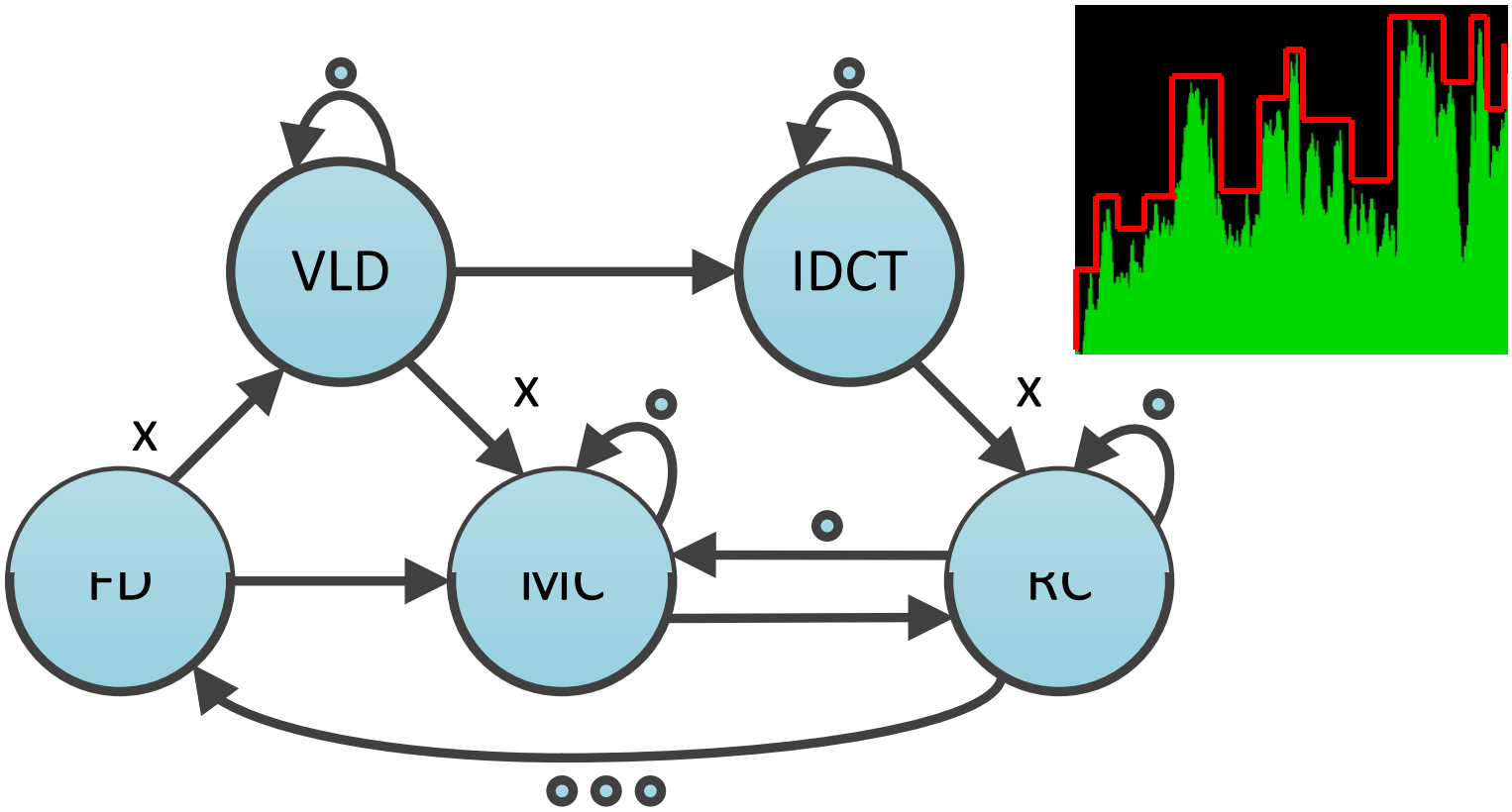


Model-of-Computation – SDF



- Model abstract from dynamic behavior
- Many efficient design-time analysis algorithms available
- Low implementation overhead
- SDF-based design approach may lead to resource over-allocation

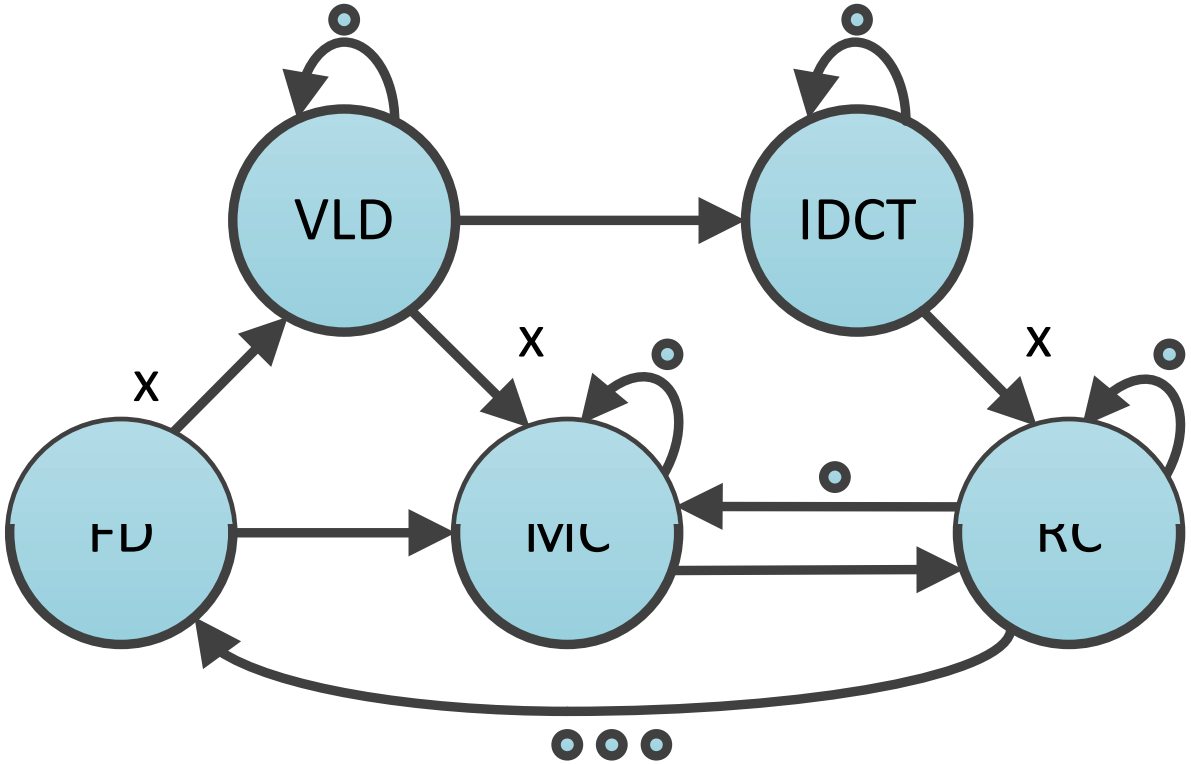
Scenario-aware dataflow



x = {0, 30, 40, 50, 60, 70, 80, 99}

- Dynamic behavior captured in scenarios
- Applications have relatively static behavior inside a scenario
- Trade-off between number of scenarios, run-time analysis techniques, and implementation efficiency

Scenario-aware dataflow – scenario switching



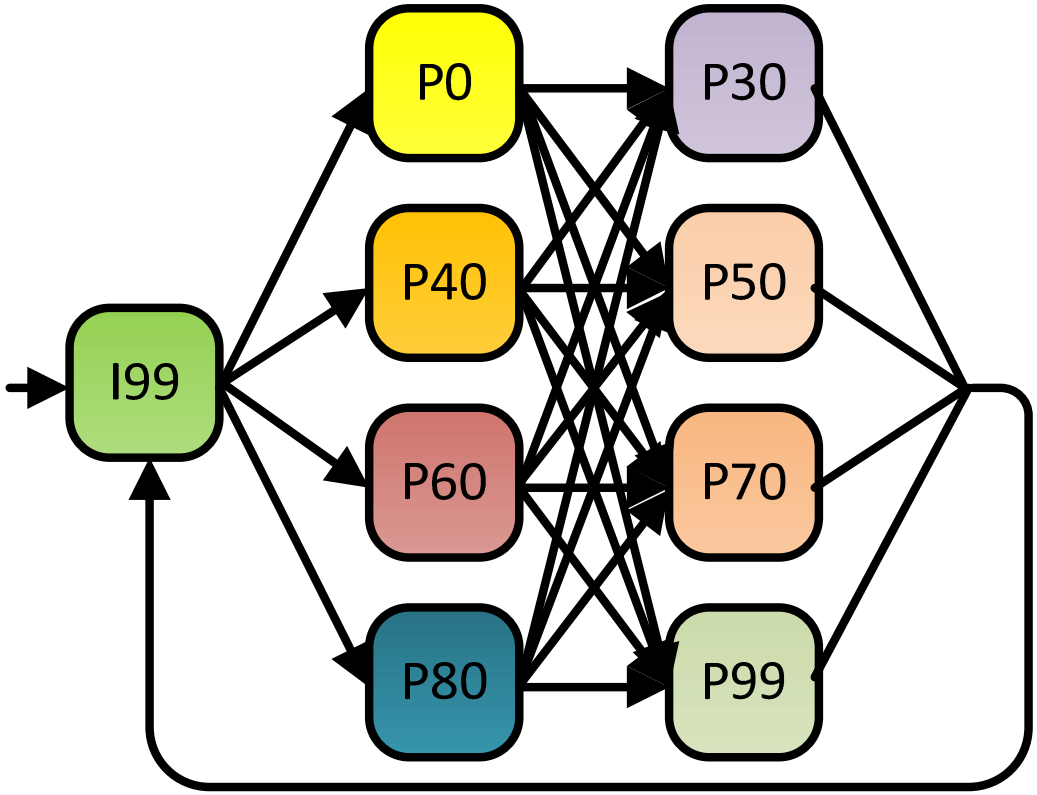
x = {0, 30, 40, 50, 60, 70, 80, 99}

- Similar to SDF, an iteration is a set of actor firings that have no net effect on the token distribution
- Different variants of SADF can model different scenario switching behavior

Scenario-aware dataflow – scenario switching

TU/e

- FSM-based SADF
  - Scenario executed for complete iteration
  - Each scenario corresponds to an SDF graph
  - FSM specifies possible scenario sequences



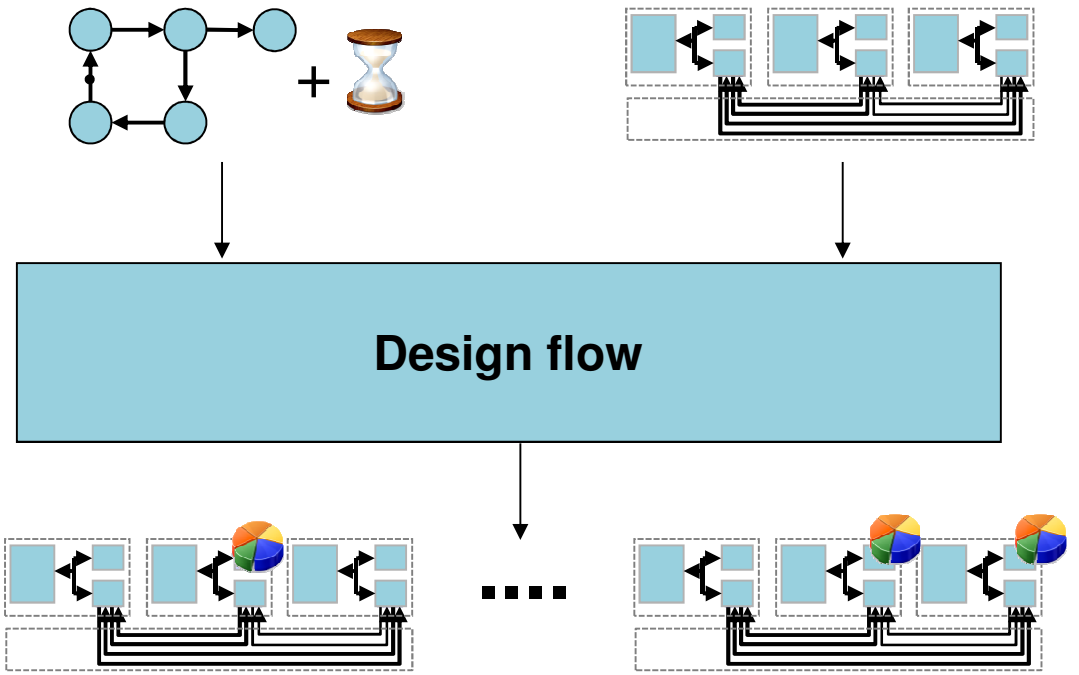
Model-based design

TU/e

Modeling

Analysis

Implementation



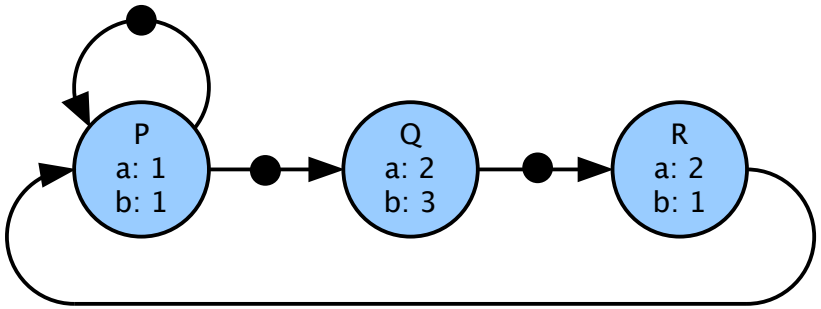
Analyzing SADF graphs

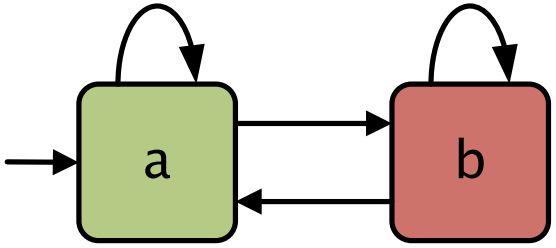
TU/e

- Analysis techniques
  - Throughput
  - Latency
  - Buffer requirements
- Techniques based on (max,+)-algebra

- Assumption
  - Relevant implementation aspects must be modeled in the graph

- Example: scenario aware dataflow graph with a static structure
  - Execution times vary with scenarios **a** and **b**

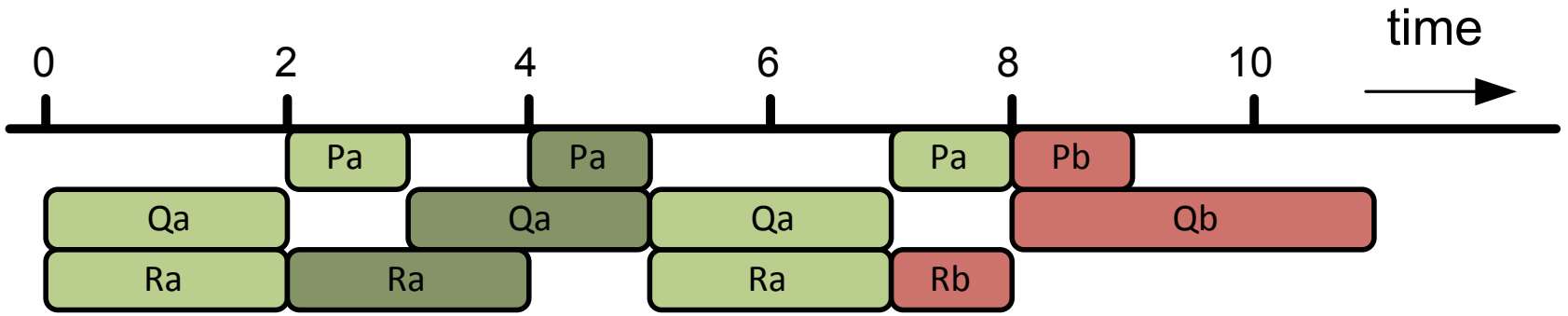


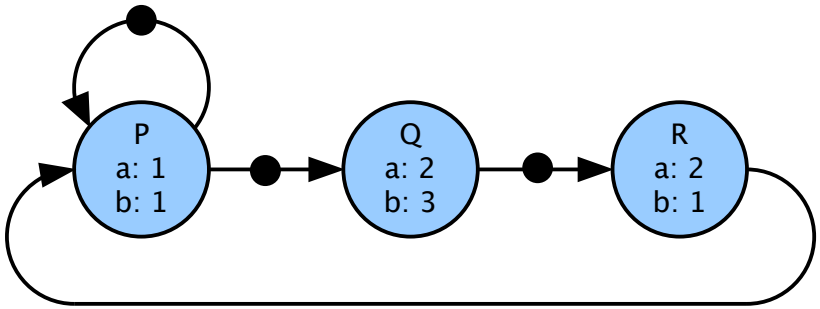


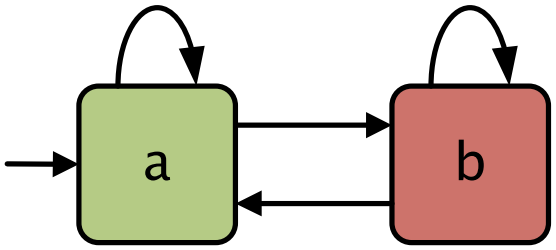
Analyzing SADF graphs

TU/e

- Gantt chart for the scenario sequence **aaab**



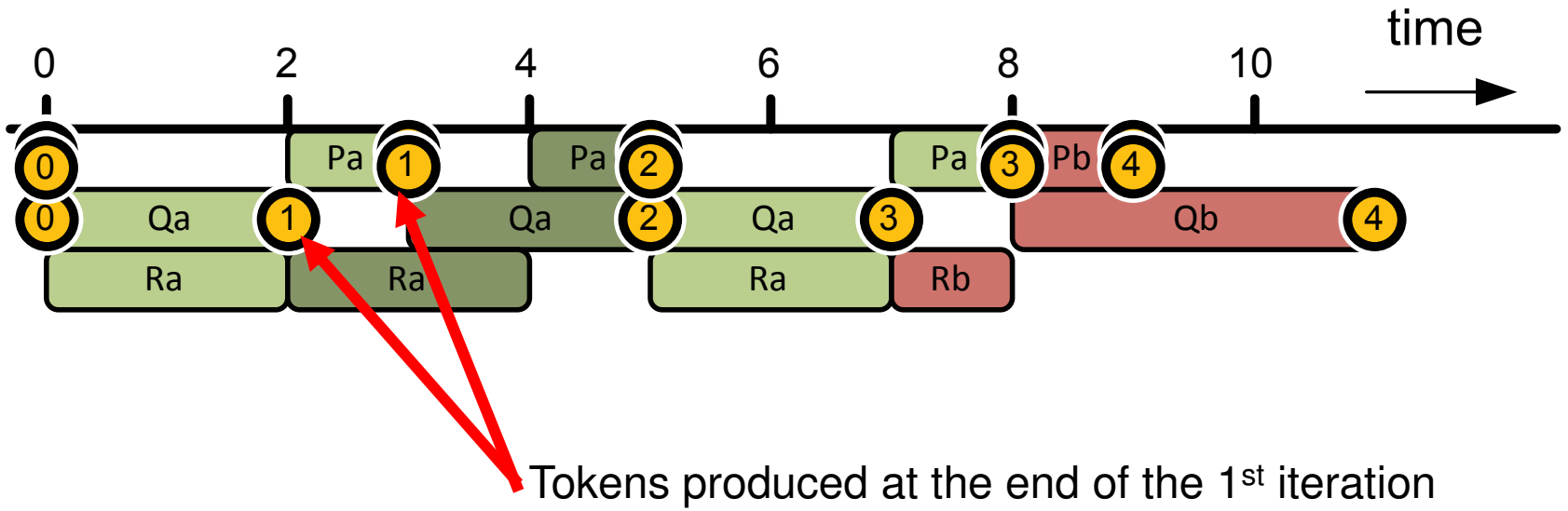


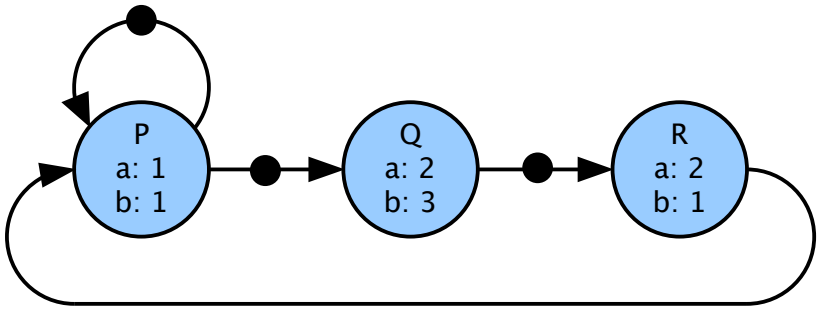


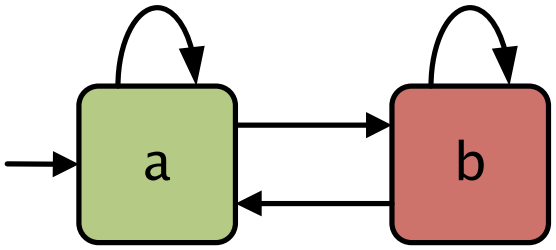
Analyzing SADF graphs

TU/e

- Gantt chart for the scenario sequence **aaab**



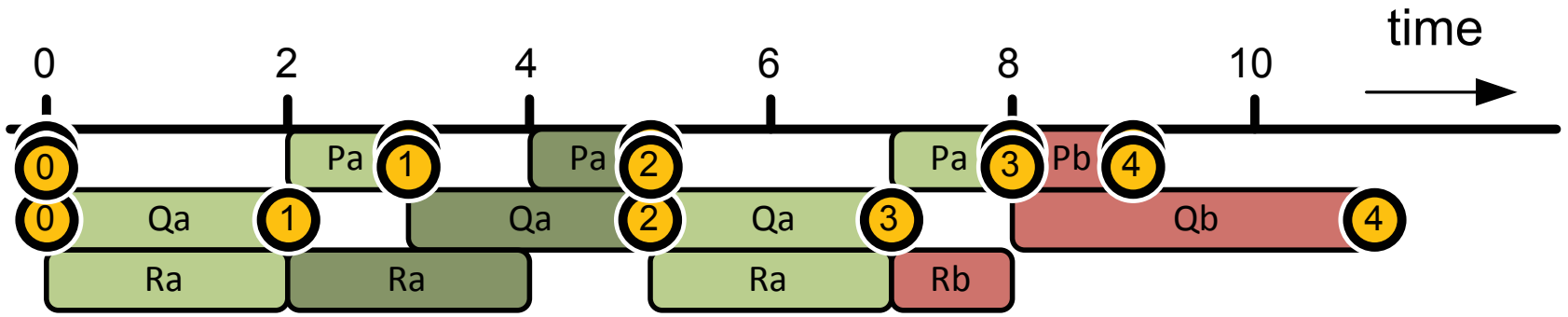




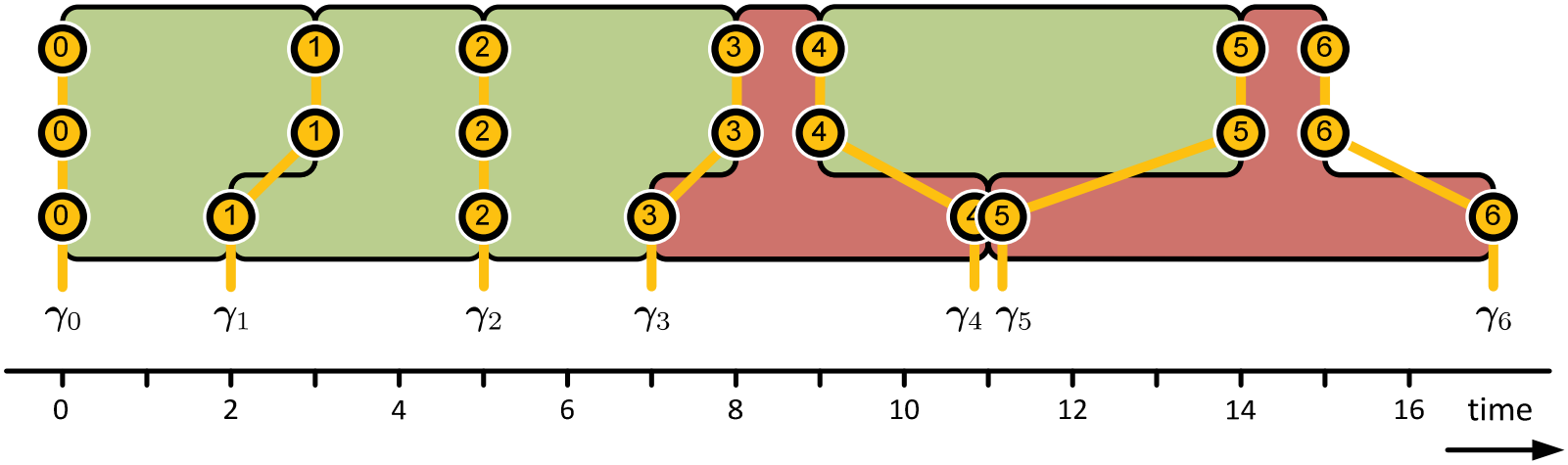
Analyzing SADF graphs

TU/e

- Gantt chart for the scenario sequence **aaab**



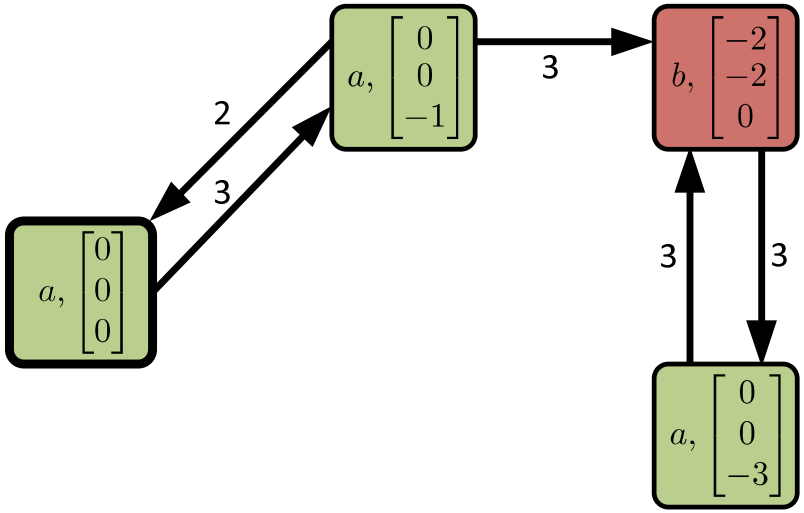
- Execution is a sequence of vector shapes
- Token time stamps (vector shapes) provide constraints for next iteration



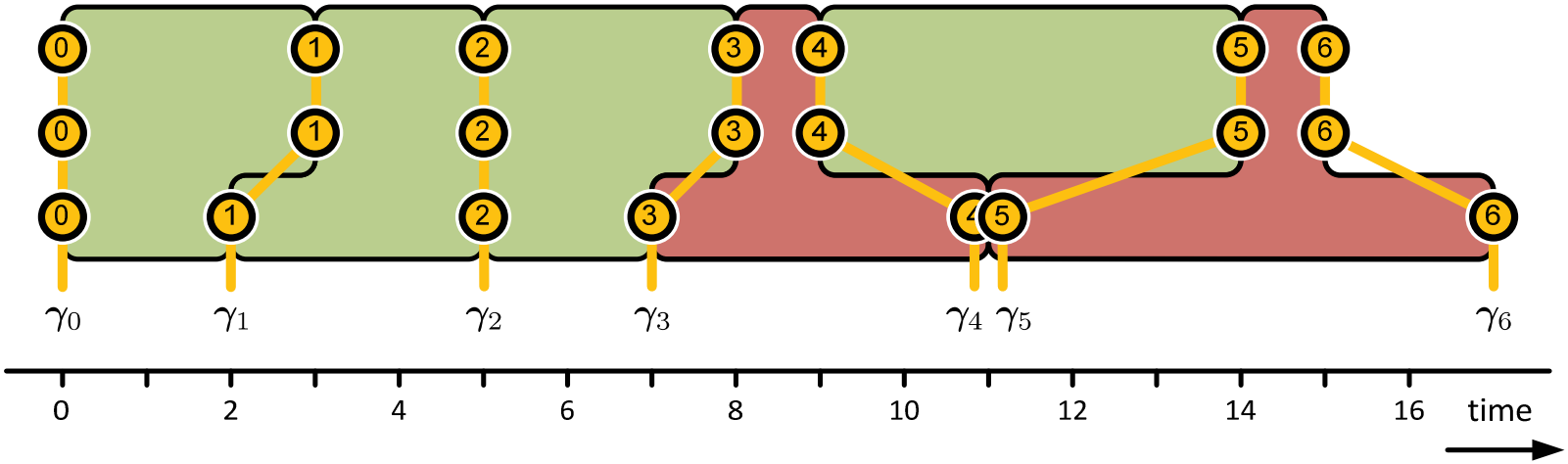
Analyzing SADF graphs

TU/e

- State space of the SADF



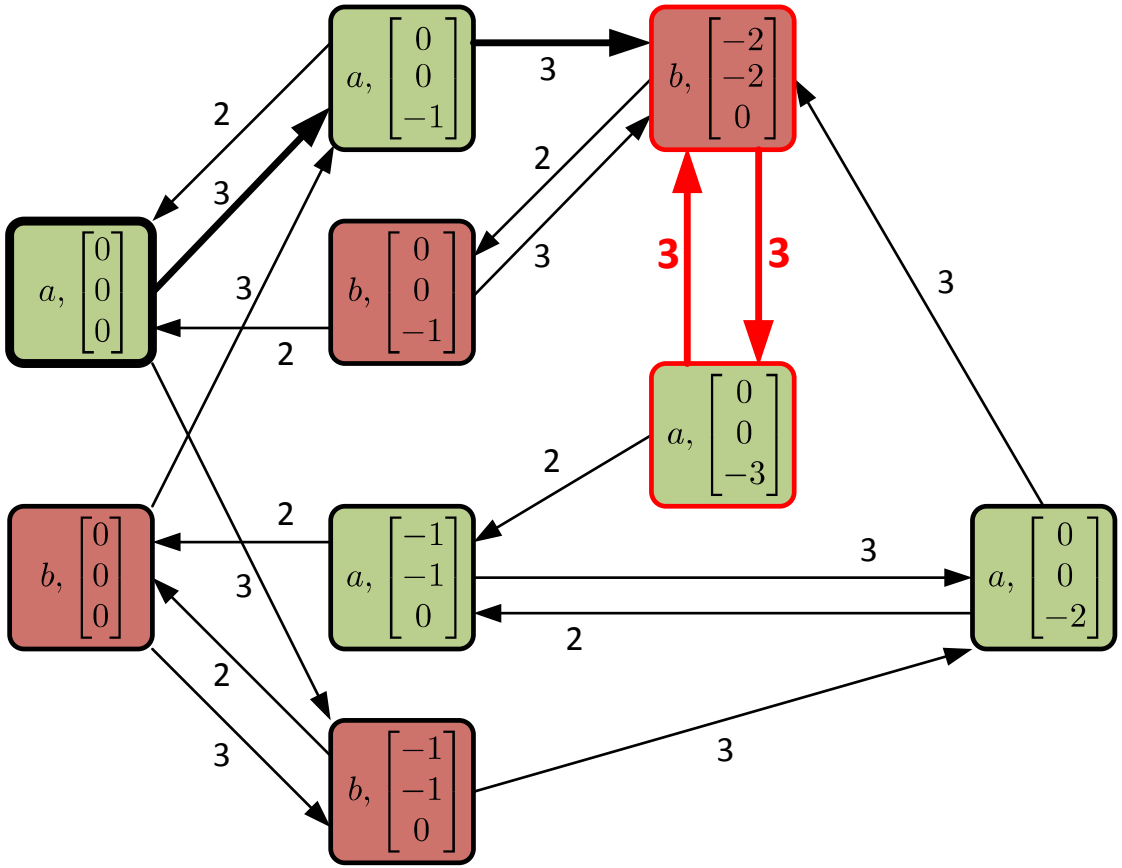
- Duration of sequence **aaabab** is: 3+2+3+3+3+3 = 17
- Execution is a sequence of vector shapes
- Token time stamps (vector shapes) provide constraints for next iteration



Analyzing SADF graphs

TU/e

- State space of the SADF



- Worst-case throughput determined by maximum cycle mean
- Latency can also be found through state space analysis

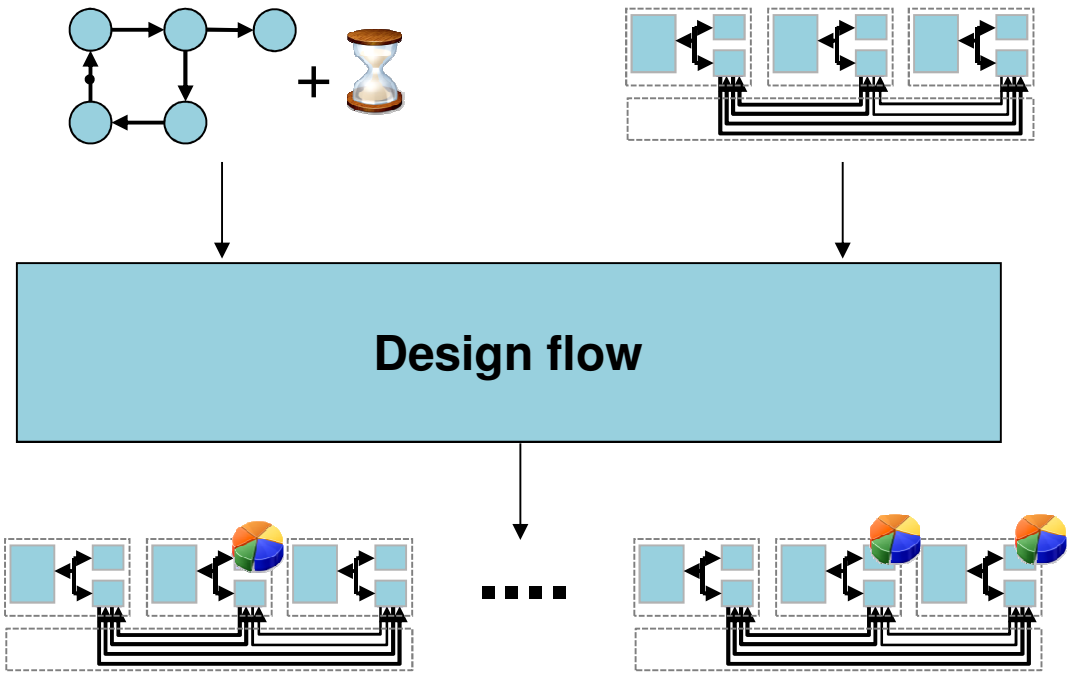
Model-based design

TU/e

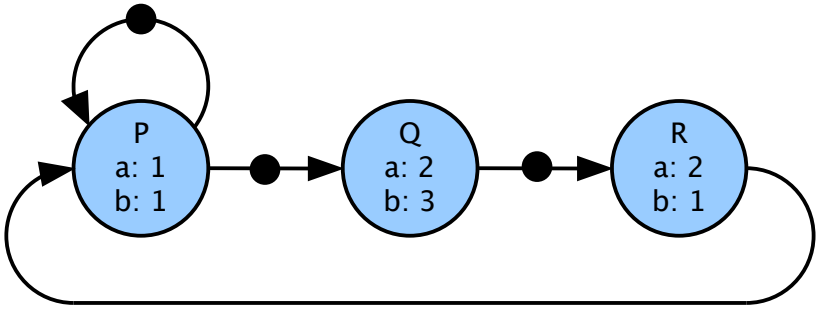
Modeling

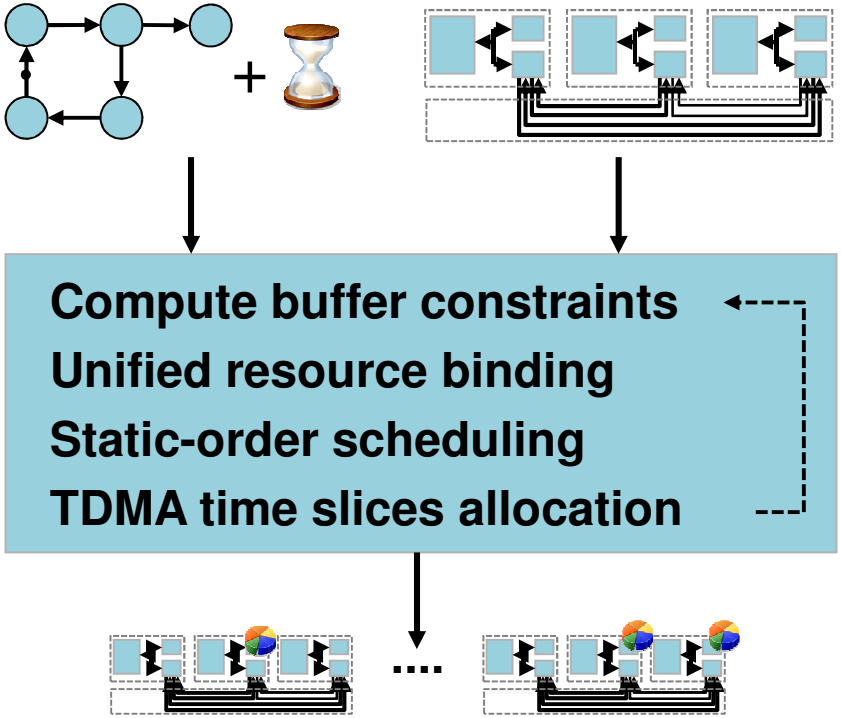
Analysis

Implementation

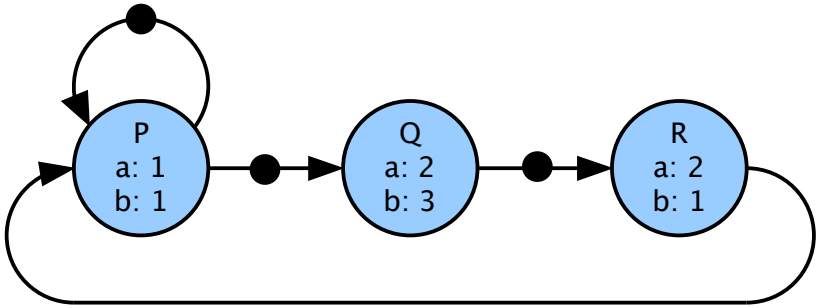


Predictable scenario-aware design-flow

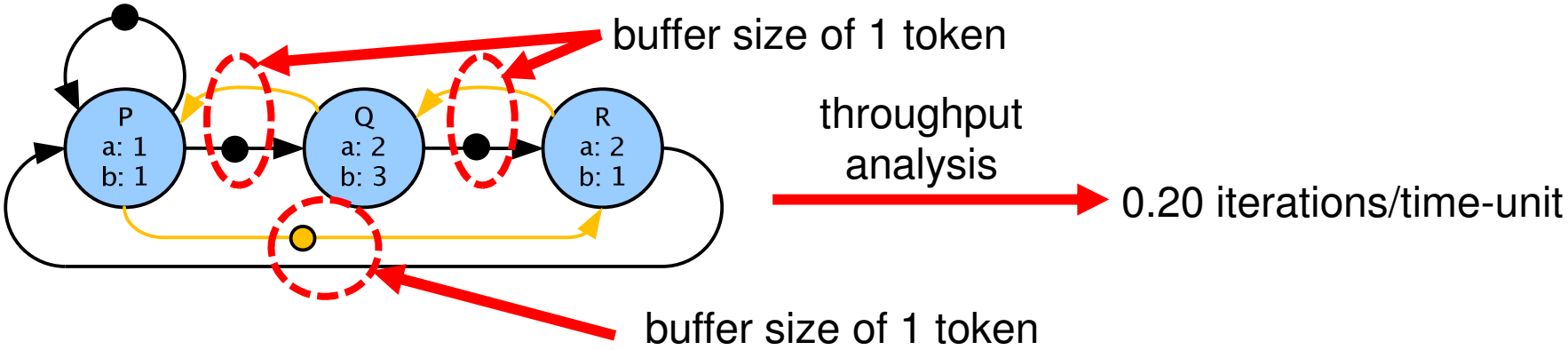


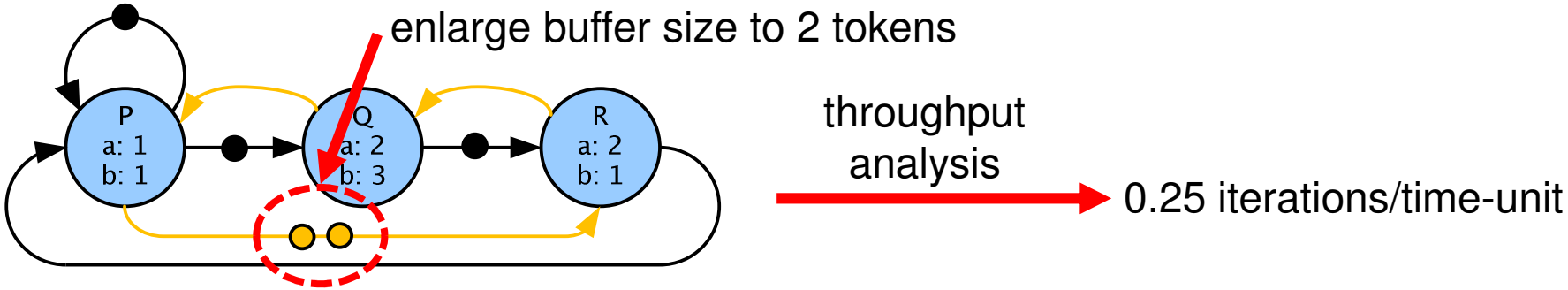


Compute buffer constraints



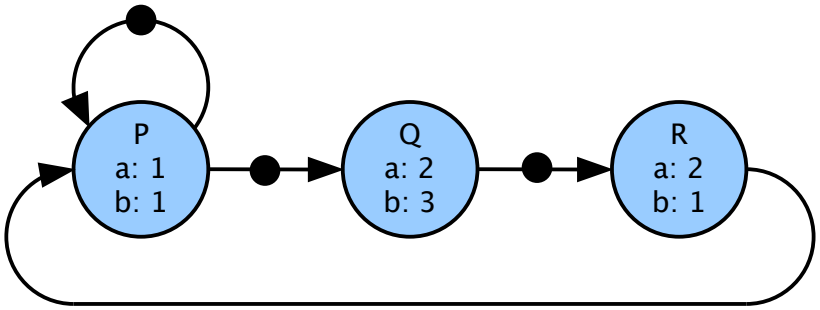
- Model buffer size constraints with back-edge with initial tokens

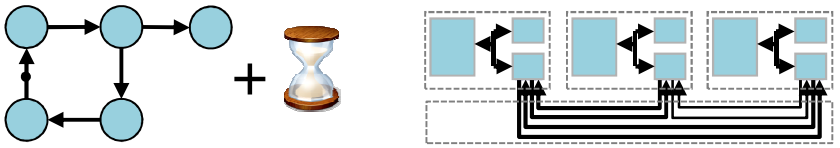




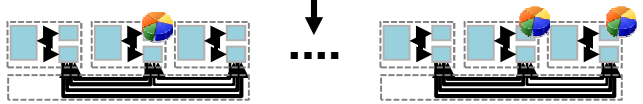
Predictable scenario-aware design-flow

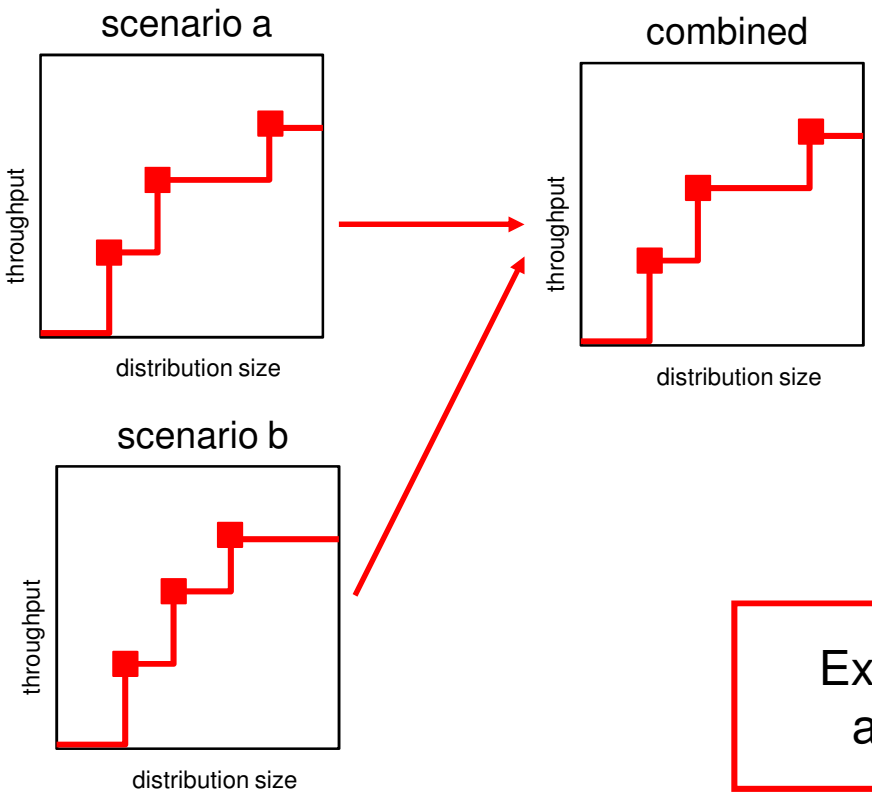
TU/e





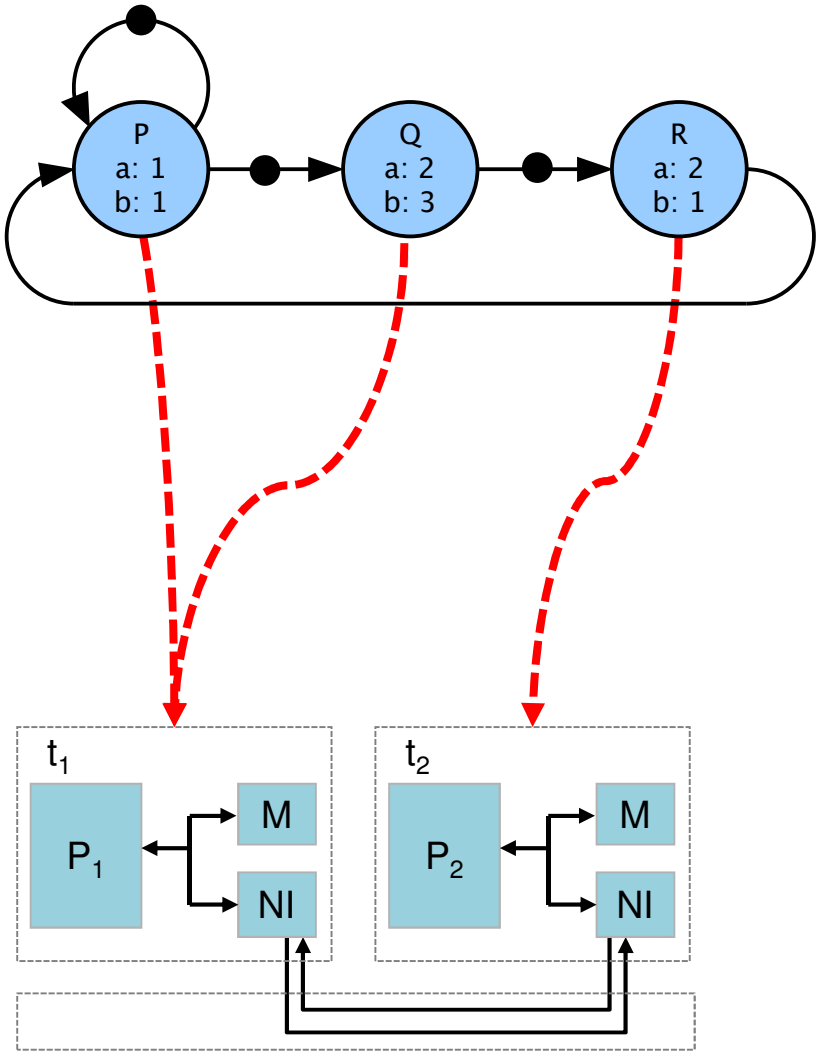
Compute buffer constraints
Unified resource binding
Static-order scheduling
TDMA time slices allocation

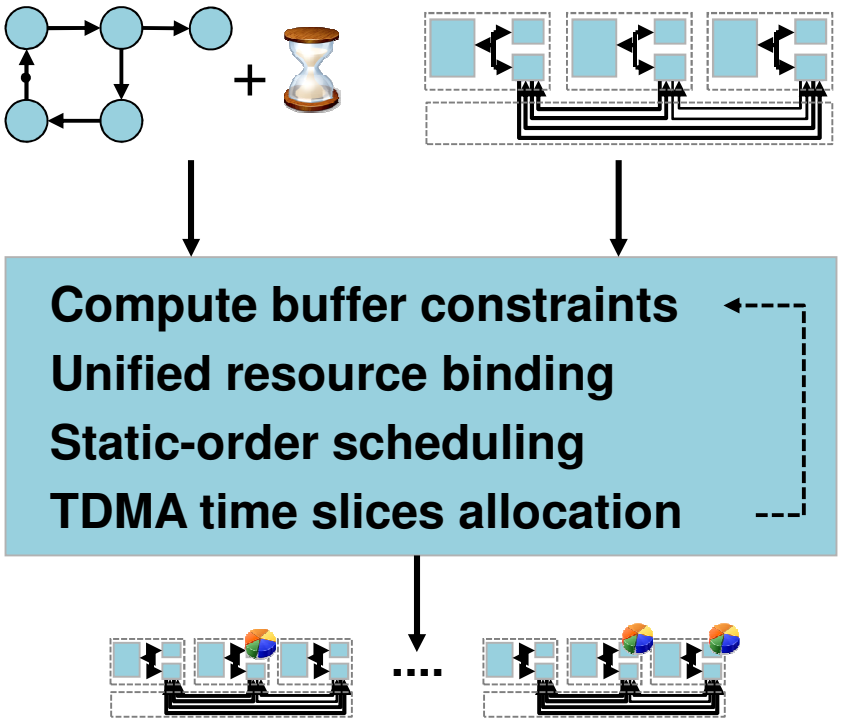




Existing buffer computation technique analysis each scenario individually

Predictable scenario-aware design-flow

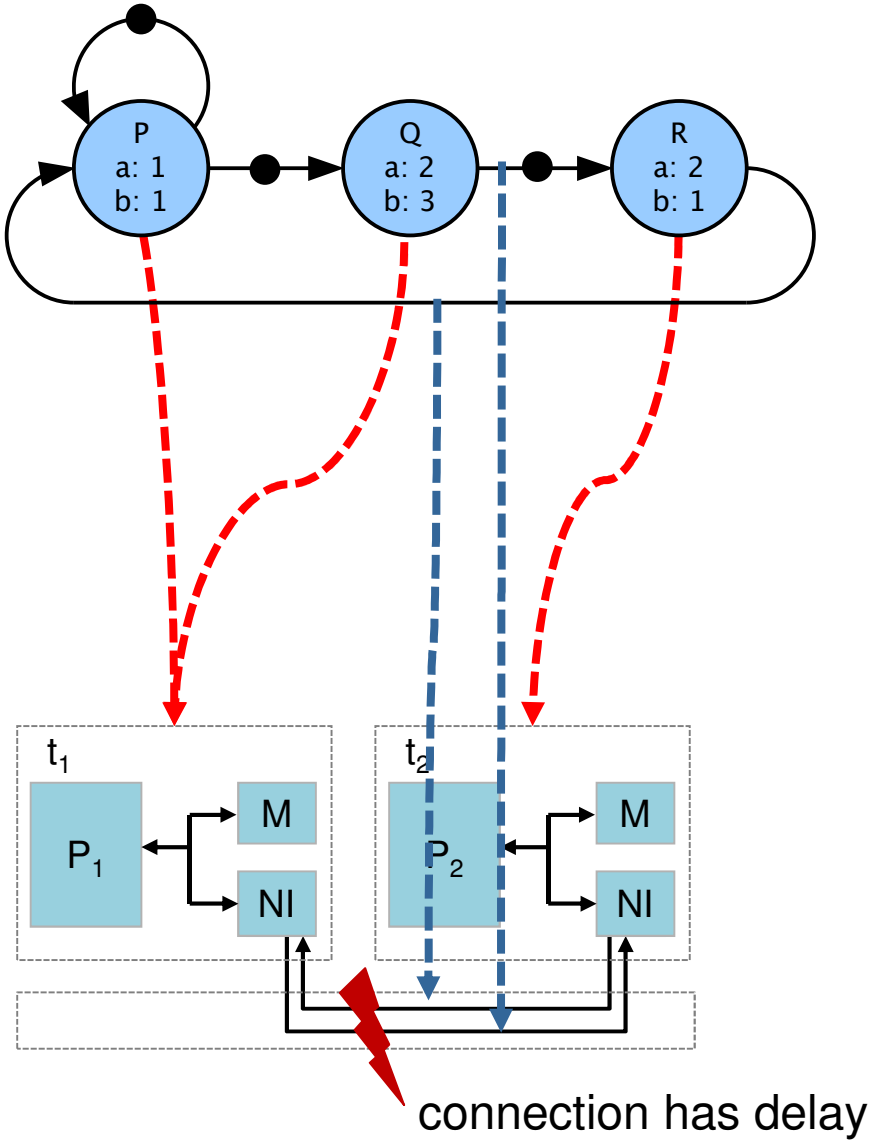




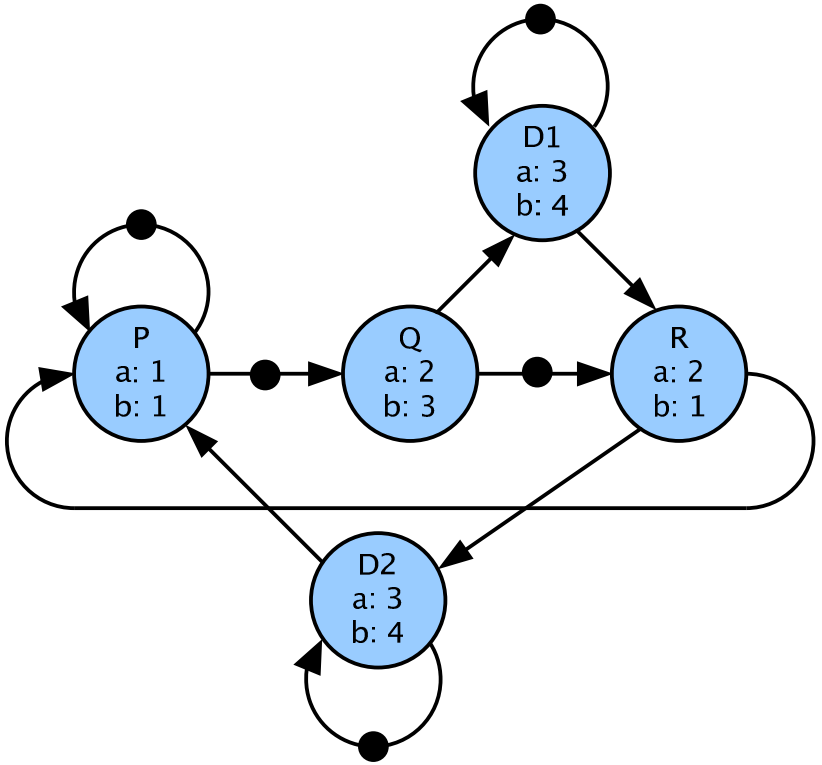
- Unified mapping avoids need for run-time reconfiguration mechanism
- Static-order schedules may change between scenarios

Modeling timing impact of platform

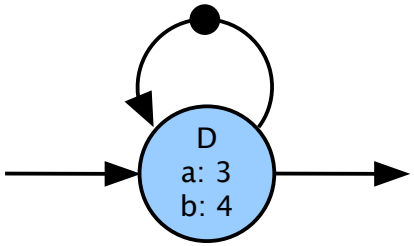
TU/e



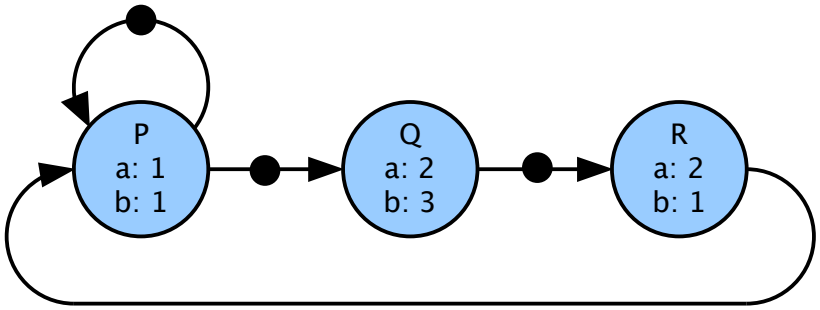
binding-aware dataflow graph

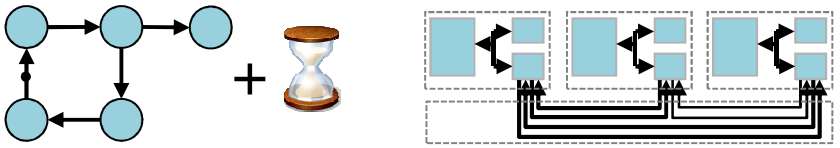


dataflow model for connection delay

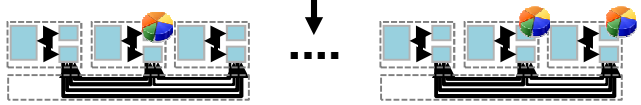


Predictable scenario-aware design-flow



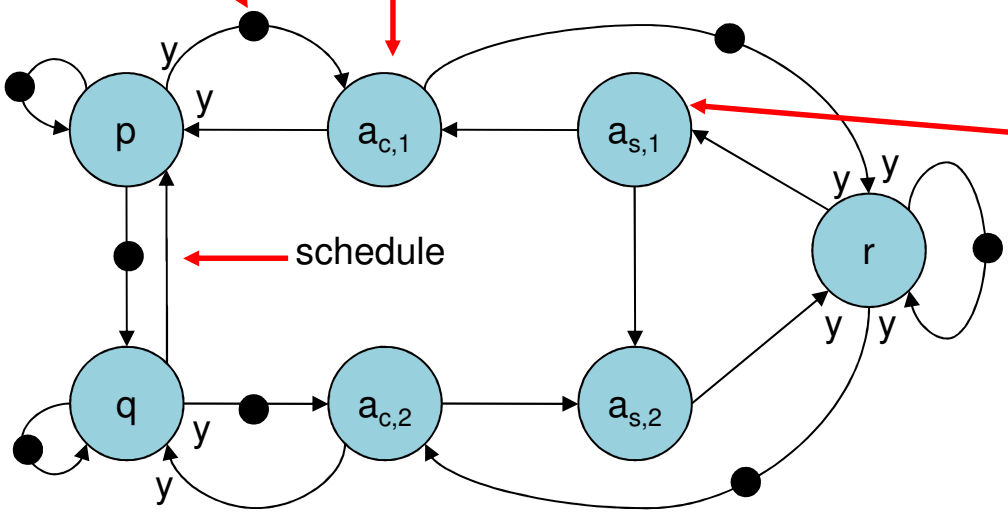


Compute buffer constraints
Unified resource binding
Static-order scheduling
TDMA time slices allocation



buffer constraint

interconnect delay

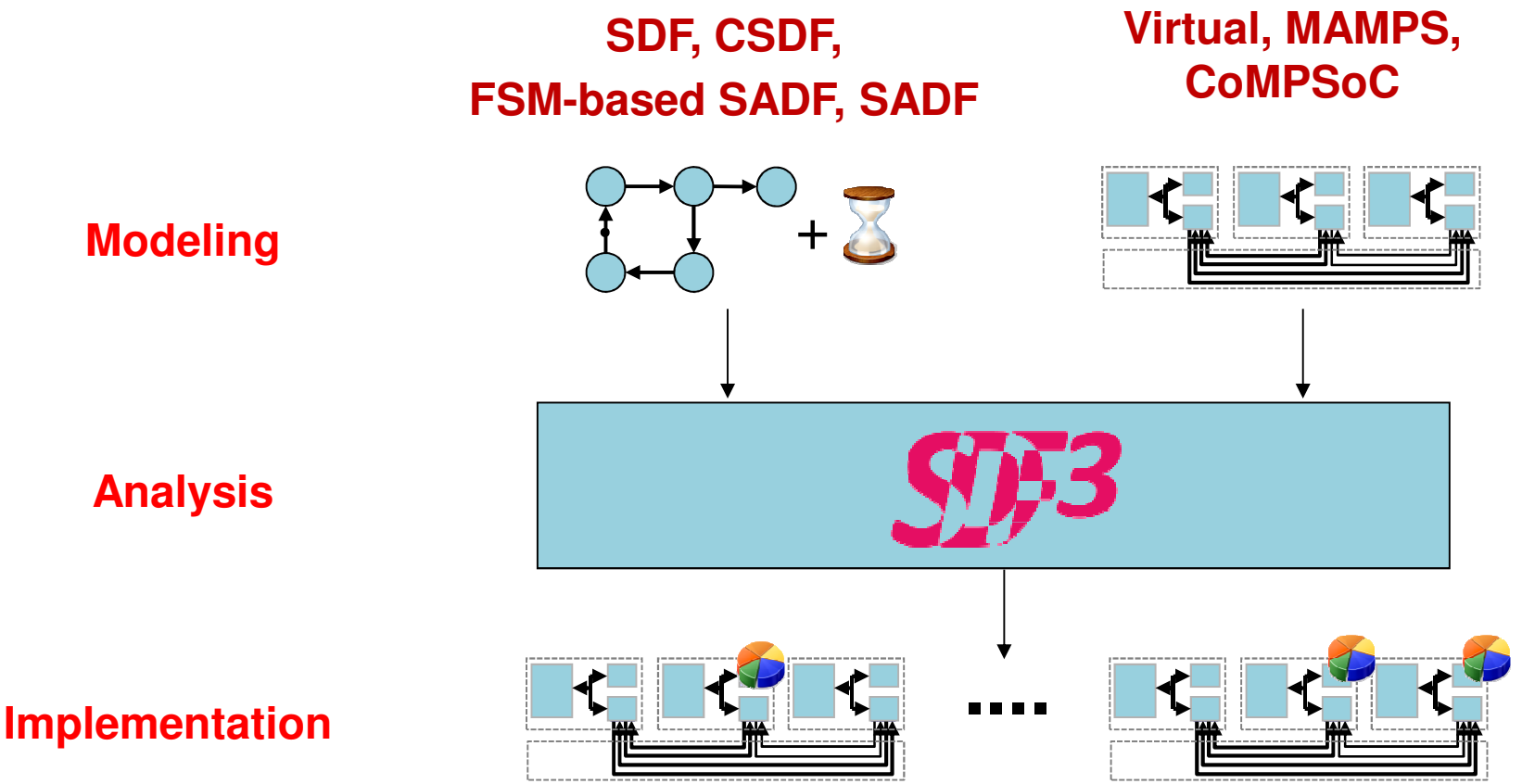


time wheel synchronization

schedule

no auto-concurrency

SDF³: SDF For Free



- Key features
  - Open-source GPL licensed software
  - Separation of analysis, transformation, and implementation techniques
  - Additional MPSoC platforms can be added with minimal effort

Conclusions

TU/e

- SADF Model-of-Computation
  - Scenarios capture dynamic (application) behavior
  - Provides many analysis techniques
  - Provides implementation trajectory
- Dataflow graph model captures
  - Application behavior
  - Timing impact of platform resources
- Use of single MoC enables model-based design of predictable systems
- Analysis and implementation techniques implemented in SDF^3 tool kit

  www.es.ele.tue.nl/sdf3