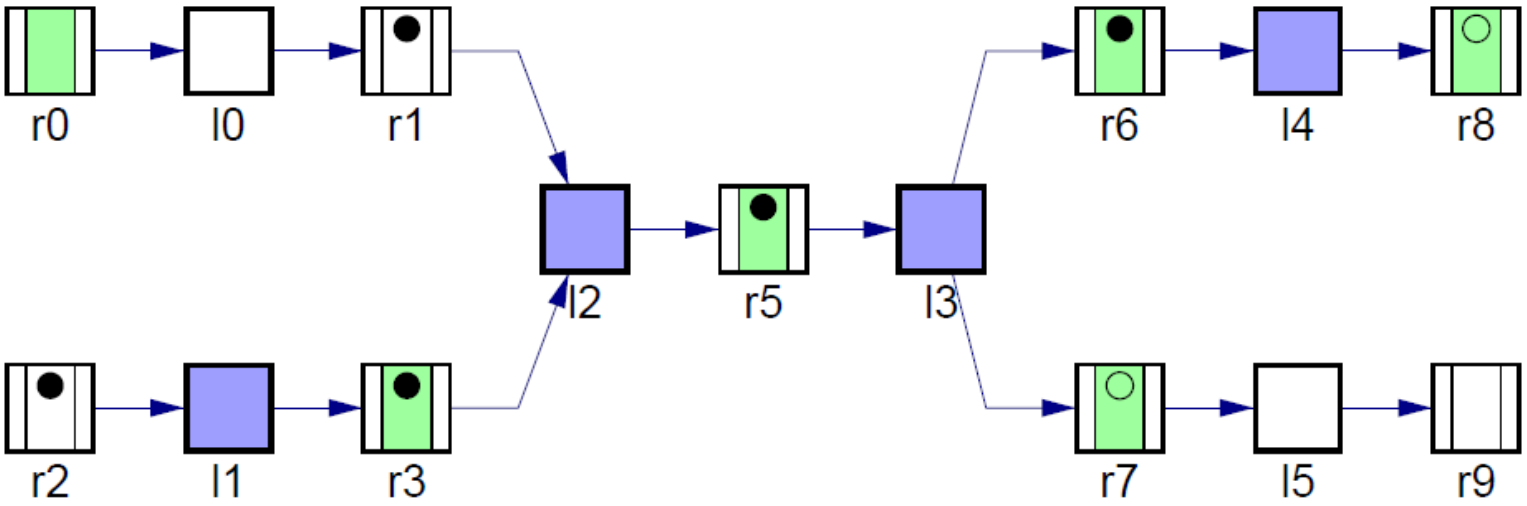
Static Data Flow Structures with Dynamic Elements

Ivan Poliakov, Charles Brej*,
Danil Sokolov, Alex Yakovlev

* University of Manchester

Static Data Flow Structures

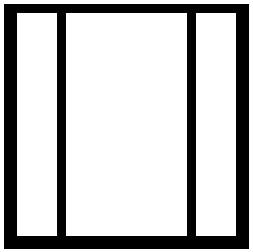


SDFS is a **directed graph**

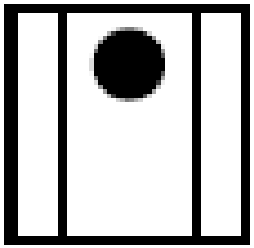
2 types of vertices (called nodes):

- combinational logic
- registers

Static Data Flow Structures - Registers



An unmarked register

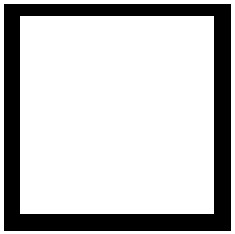


A marked register

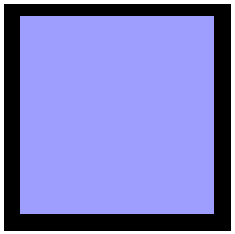
Register

- can contain a token
- token flow models data flow
- token transfer rules can be different

Static Data Flow Structures - Logic



A reset logic block

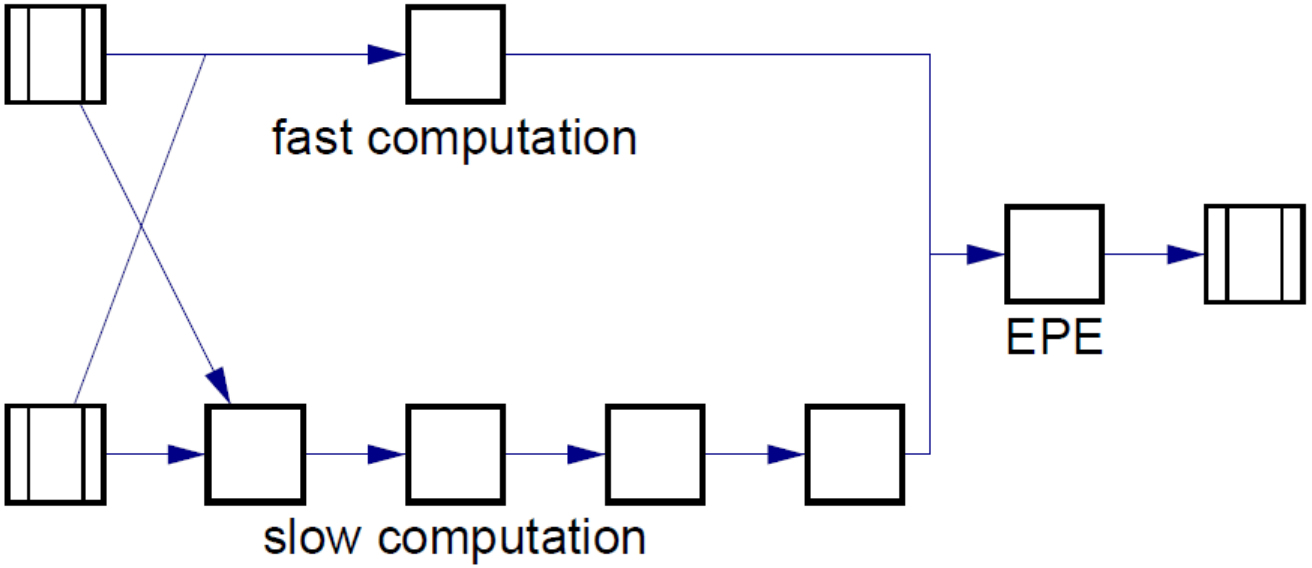


An evaluated logic block

Combinational logic

- does not contain tokens
- can be in two states: reset and evaluated
- token transfer rules can be different

SDFS drawbacks



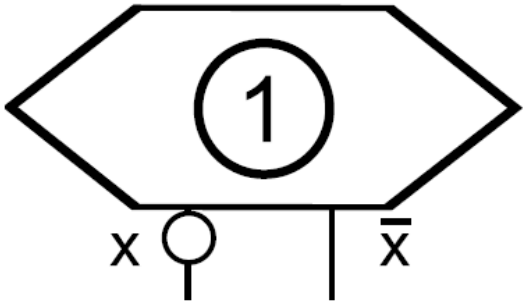
SDFS drawbacks

- determenistic behaviour
- choice made by the control cannot be modelled

Dynamic elements

- required to increase the flexibility of SDFS
- model the influence of the control path
- introduce data-dependent choice

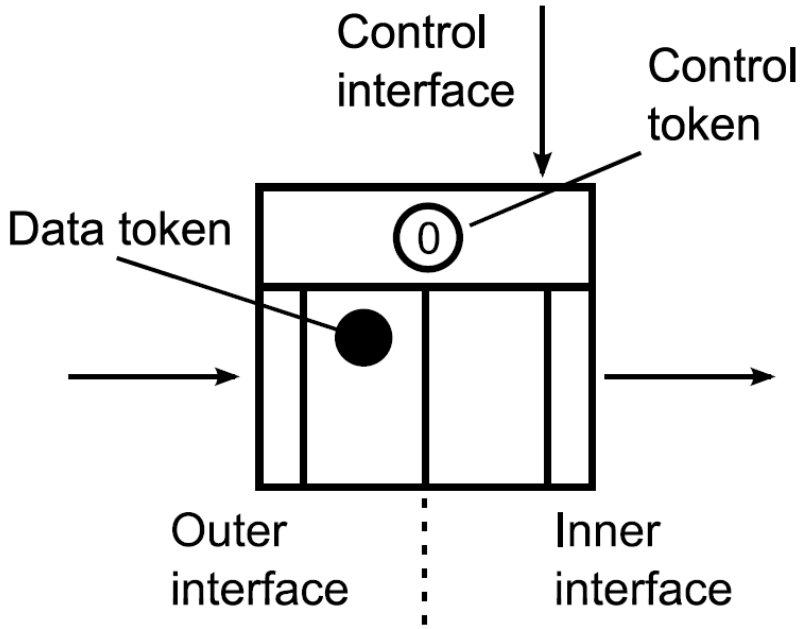
Dynamic elements - Controller



Controller

- tokens are associated with data values
- data values (0 and 1) represent control signals
- separate from data tokens
- a function can be assigned to modify token values
  (e.g. inversion)

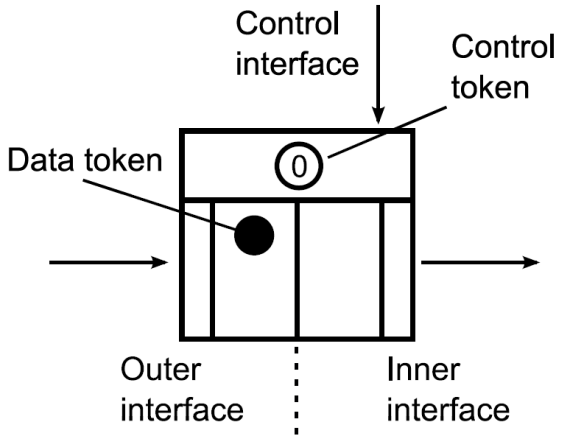
Dynamic elements — Push and Pop



Dynamic element

- contains two data interfaces: inner and outer, behaving as a regular SDFS registers
- contains a control interface (accepts data tokens)

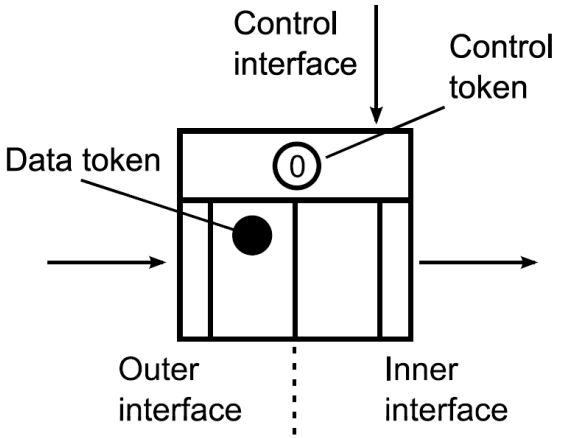
Dynamic elements - Push



Push

- synchronises a data token on the outer interface with a control token.
- If the control token is a 1-token, it forwards the data token
- If the data token is a 0-token, it allows the token to be removed from the outer interface without transferring it into the inner interface.

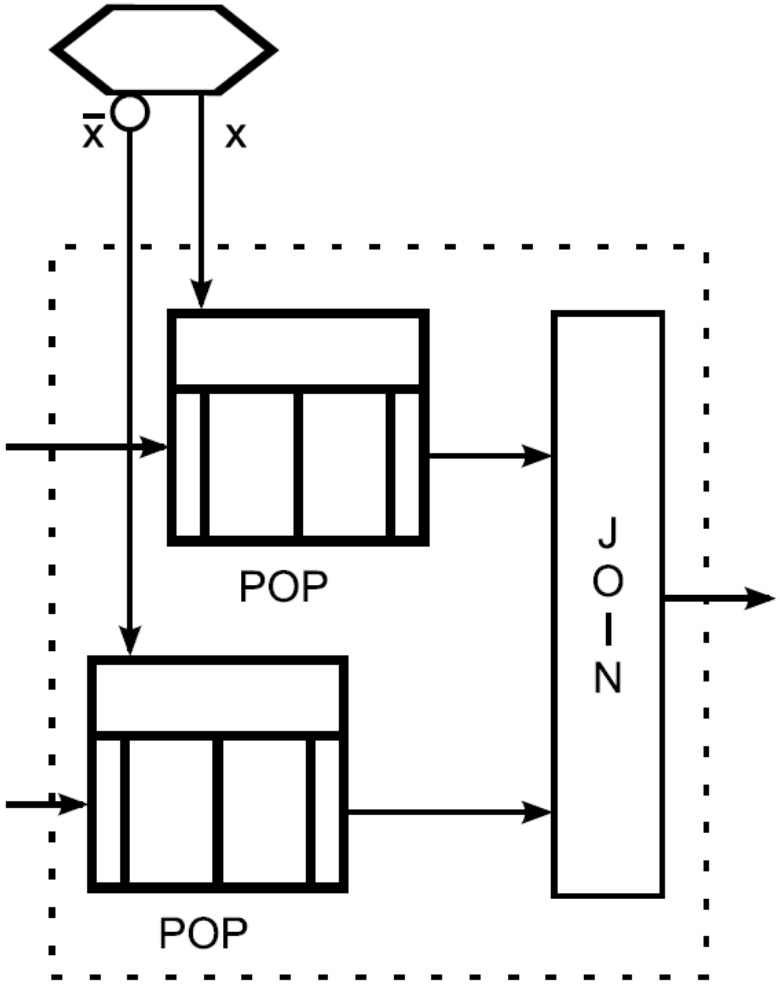
Dynamic elements - Pop



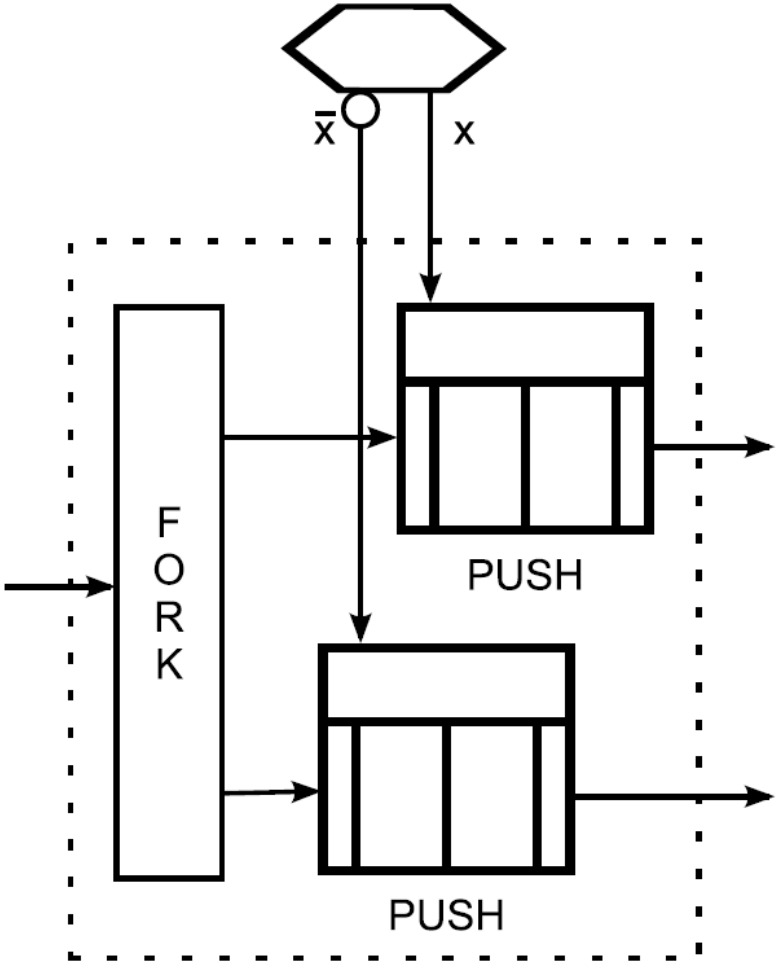
Pop

- first receives a control token
- if it is a 1-token, it then synchronises it with a data token on the inner interface and transfers it into the inner interface
- if it is a 0-token, it immediately produces a dummy data token on the inner interface

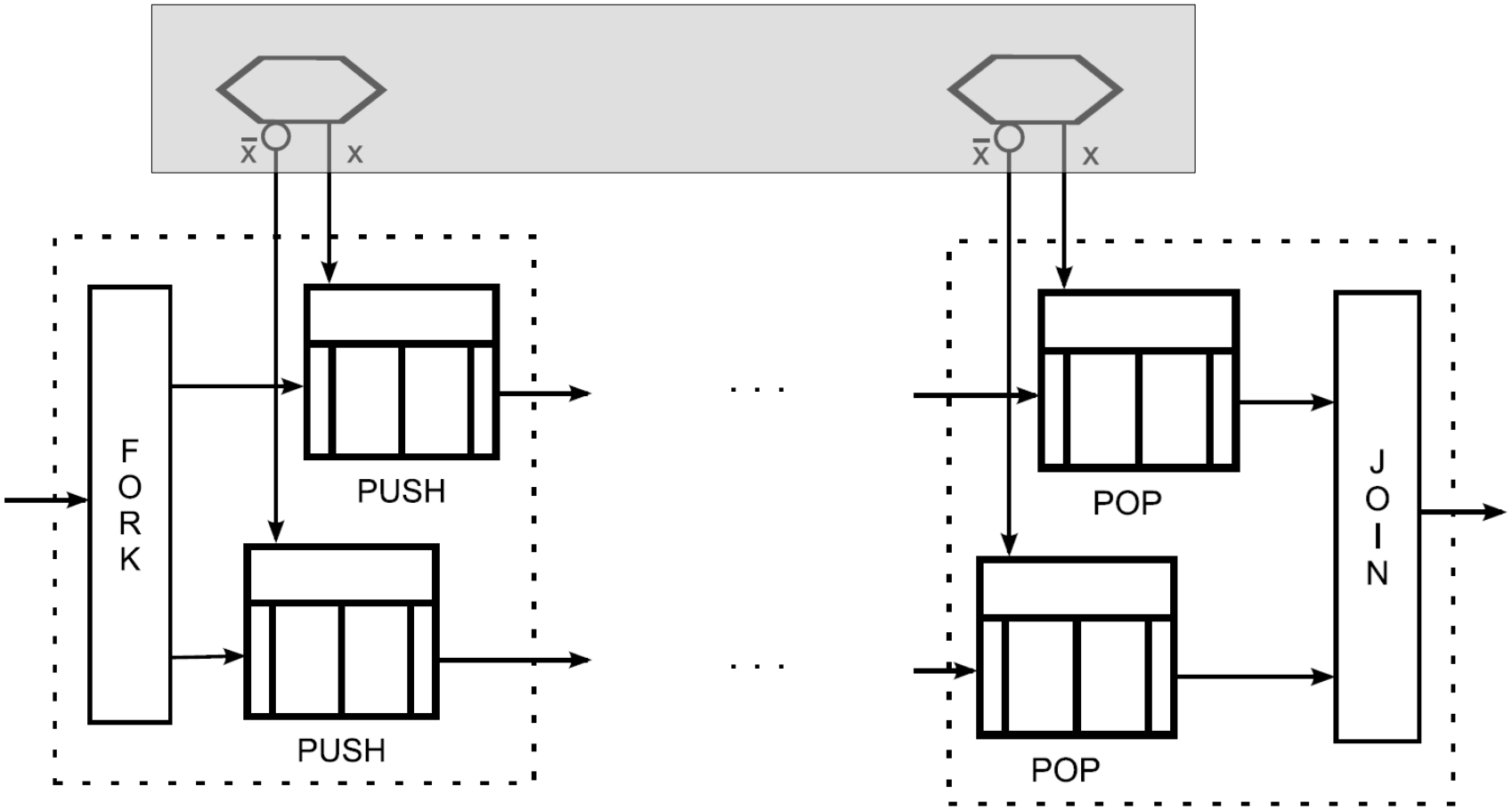
Dynamic elements (6) - Multiplexer



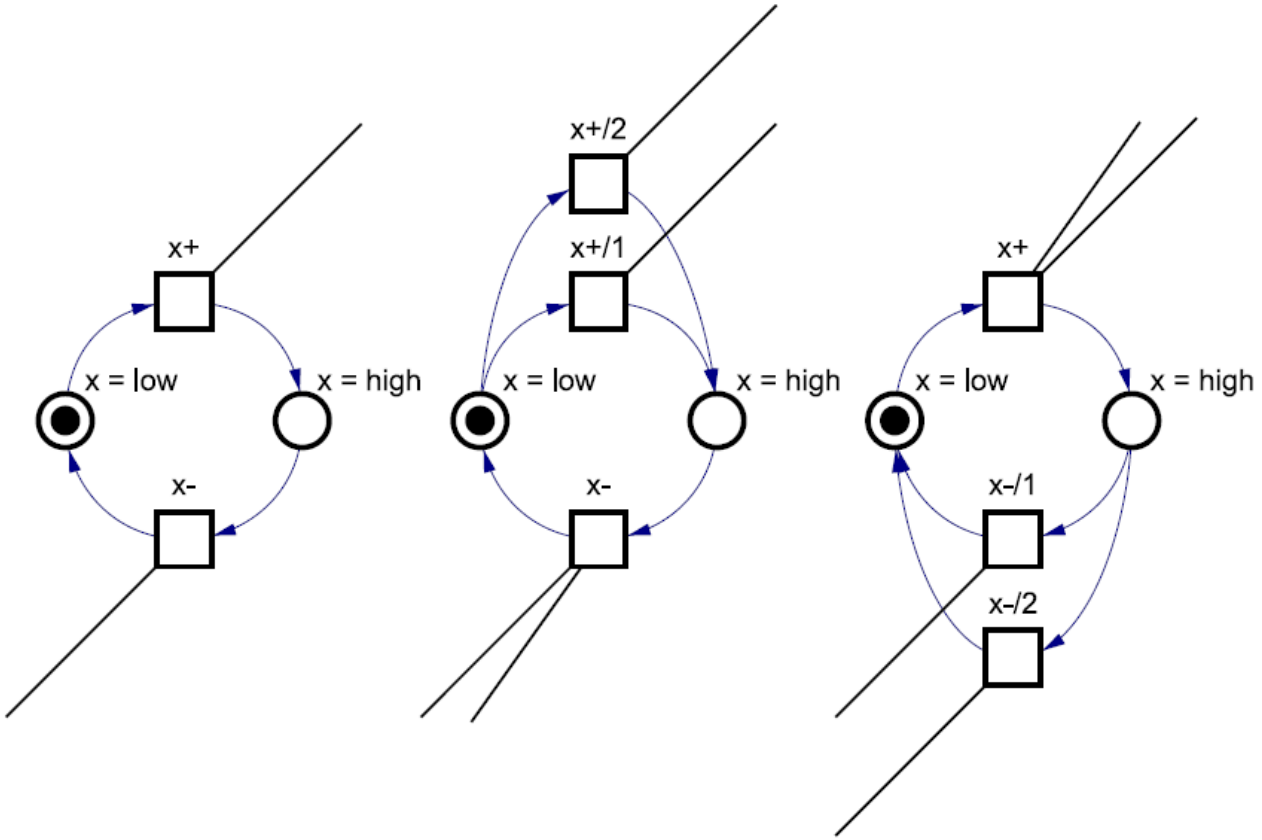
Dynamic elements (7) - Demultiplexer



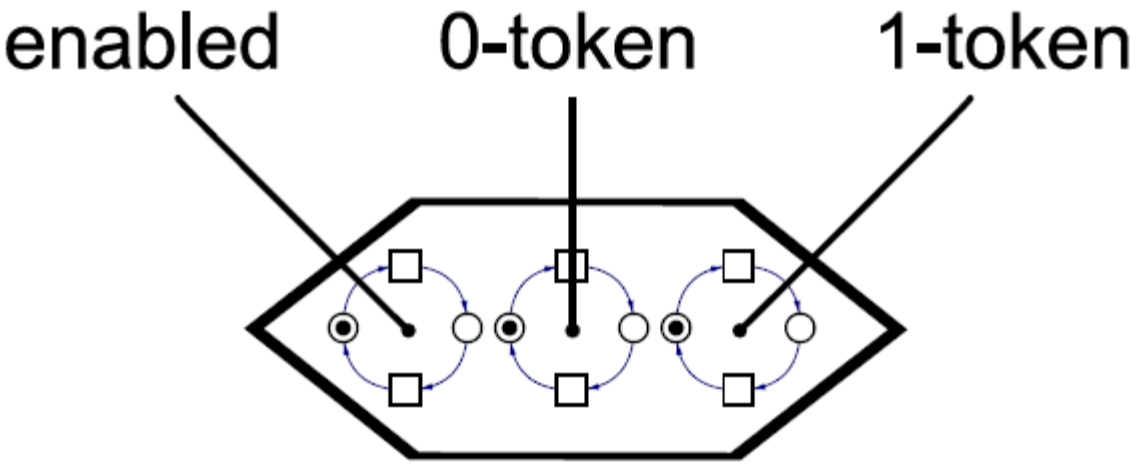
Dynamic elements (8) — data dependent choice



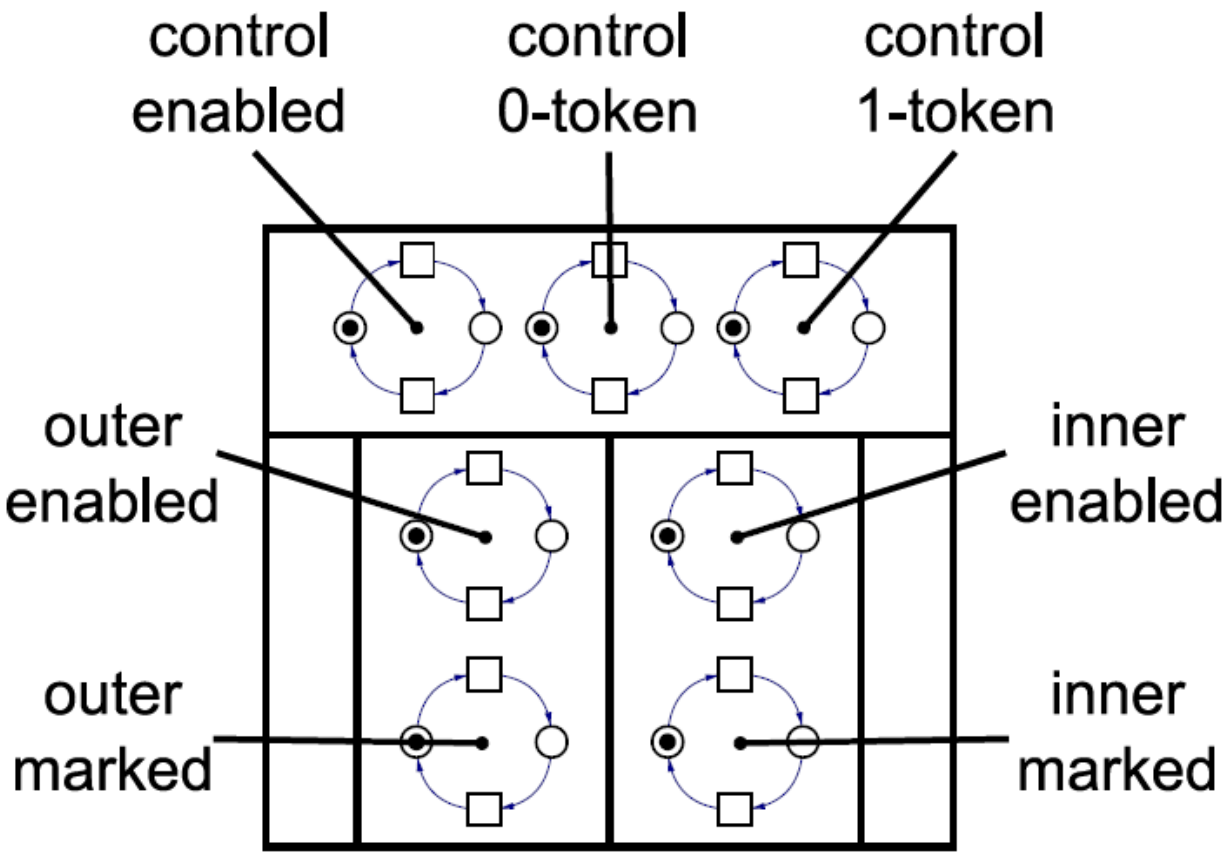
Verfication — Schematic Petri Nets



Verification — mapping of the controller element



Verification — mapping of the Push/Pop elements



Conclusions and future work

- Dynamic elements are introduced into the SDFS model
- A method for verification of SDFS with dynamic elements is proposed

The focus of the future work is a method for the automated translation of a higher level language (such as Balsa) description into an SDFS model

End

Thank you!

Questions?