http://workcraft.org/

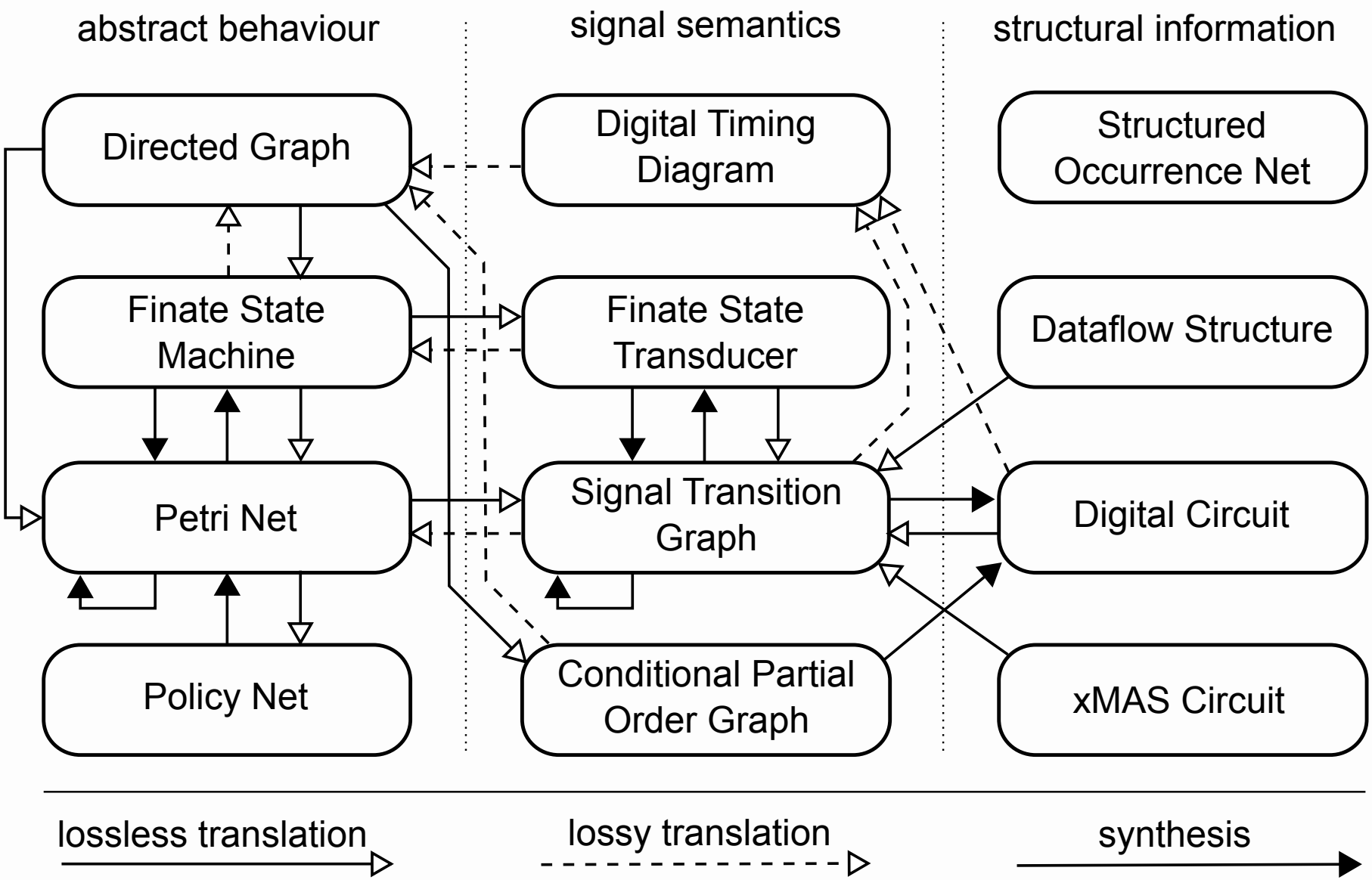
What is WORKCRAFT?

- Framework for interpreted graph models
  - Interoperability between different abstraction levels
  - Consistency for users; convenience for developers
- Elaborate graphical user interface
  - Visual editing, analysis, and simulation
  - Easy access to common operations
  - Possibility to script specialised actions
- Interface to back-end tools for synthesis and verification
  - Reuse of established theory and tools (PETRIFY, MPSAT, PUNF)
  - Command log for debugging and scripting

Why to use WORKCRAFT?

- Availability
  - Open-source front-end and plugins
  - Permissive freeware licenses for back-end tools
  - Frequent releases (4-6 per year)
  - Specialised tutorials and online training materials
- Extendibility
  - Plugins for new formalisms
  - Import, export and converter plugins
  - Interface to back-end tools
- Usability
  - Elaborated GUI developed with much user feedback
- Portability
  - Distributions for Windows, Linux, and OS X

Supported graph models

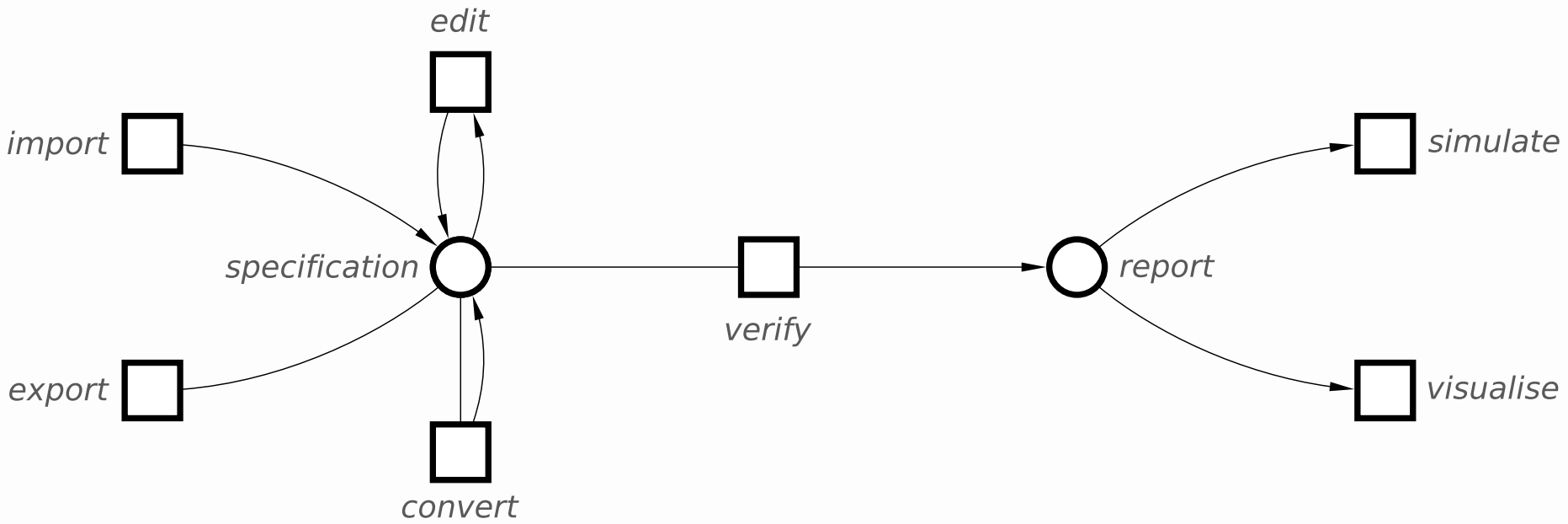


Supported features

Model	Supported features			
	Editing	Simulation	Verification	Synthesis
abstract behaviour				
Directed Graph	Yes	Yes	Yes	n/a
Finite State Machine	Yes	Yes	Yes	Yes[1)]
Petri Net	Yes	Yes	Yes	Yes[2)]
Policy Net	Yes	Yes	Yes	n/a
signal semantics				
Digital Timing Diagram	Yes	No	n/a	n/a
Finite State Transducer	Yes	Yes	Yes	Yes[3)]
Signal Transition Graph	Yes	Yes	Yes	Yes[4)]
Conditional Partial Order Graph	Yes	Some	No	Yes
structural information				
Structured Occurrence Net	Yes	Yes	Yes	n/a
Dataflow Structure	Yes	Yes	Yes	No
Digital Circuit	Yes	Yes	Yes	n/a
xMAS Circuit	Yes	Yes	Some	No

[1)] synthesis into Petri Net
[2)] re-synthesis into simpler Petri Net
[3)] synthesis into Signal Transition Graph
[4)] synthesis into Digital Circuit and re-synthesis into simpler Petri Net

Design flow



- Import: ASTG, Verilog
- Export: ASTG, Verilog, SVG/Dot/PDF/EPS
- Convert: synthesis or translation
- Verify: reachability analysis (REACH predicates, SVA-like invariants)
- Visualise: CSC conflict cores, circuit initialisation, bottleneck

Design flow: Asynchronous circuits

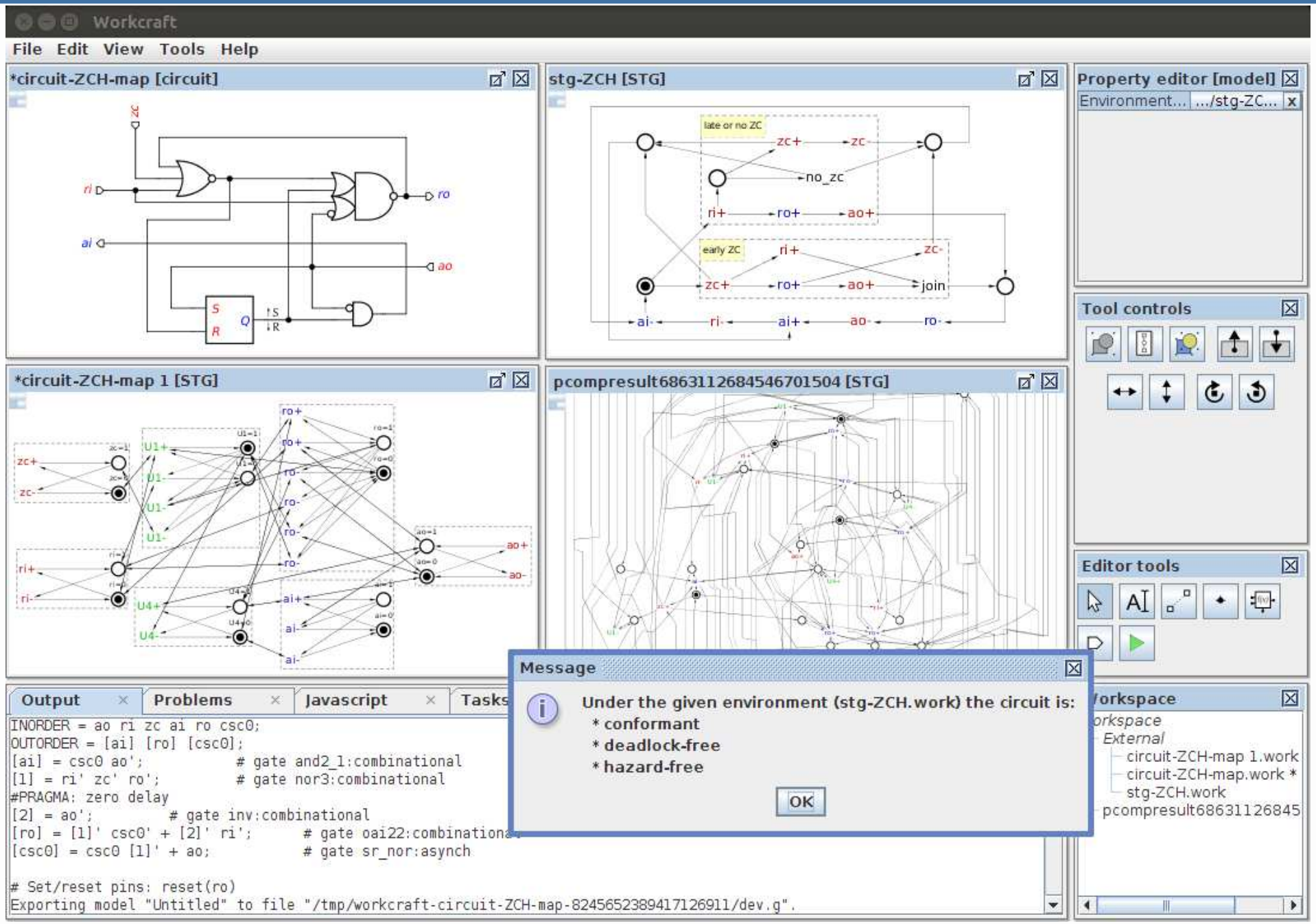
1. Specification of desired circuit behaviour with an STG model
2. Verification of the STG model
   (a) Standard implementability properties:
   consistency, deadlock freeness, output persistency
   (b) Design–specific custom properties
3. Resolution of complete state coding (CSC) conflicts
4. Circuit synthesis in one of the supported design styles
5. Manual tweaking and optimisation of the circuit
6. Verification of circuit against the initial specification
   (a) Synthesis tools are complicated and may have bugs
   (b) Manual editing is error-prone
7. Exporting the circuit as a Verilog netlist for conventional EDA backend

What is hidden from the user?

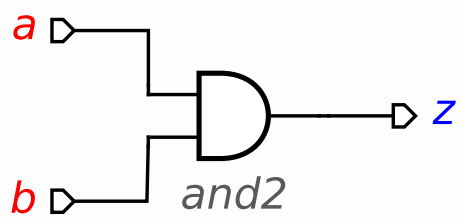
Verification that the circuit conforms to its specification

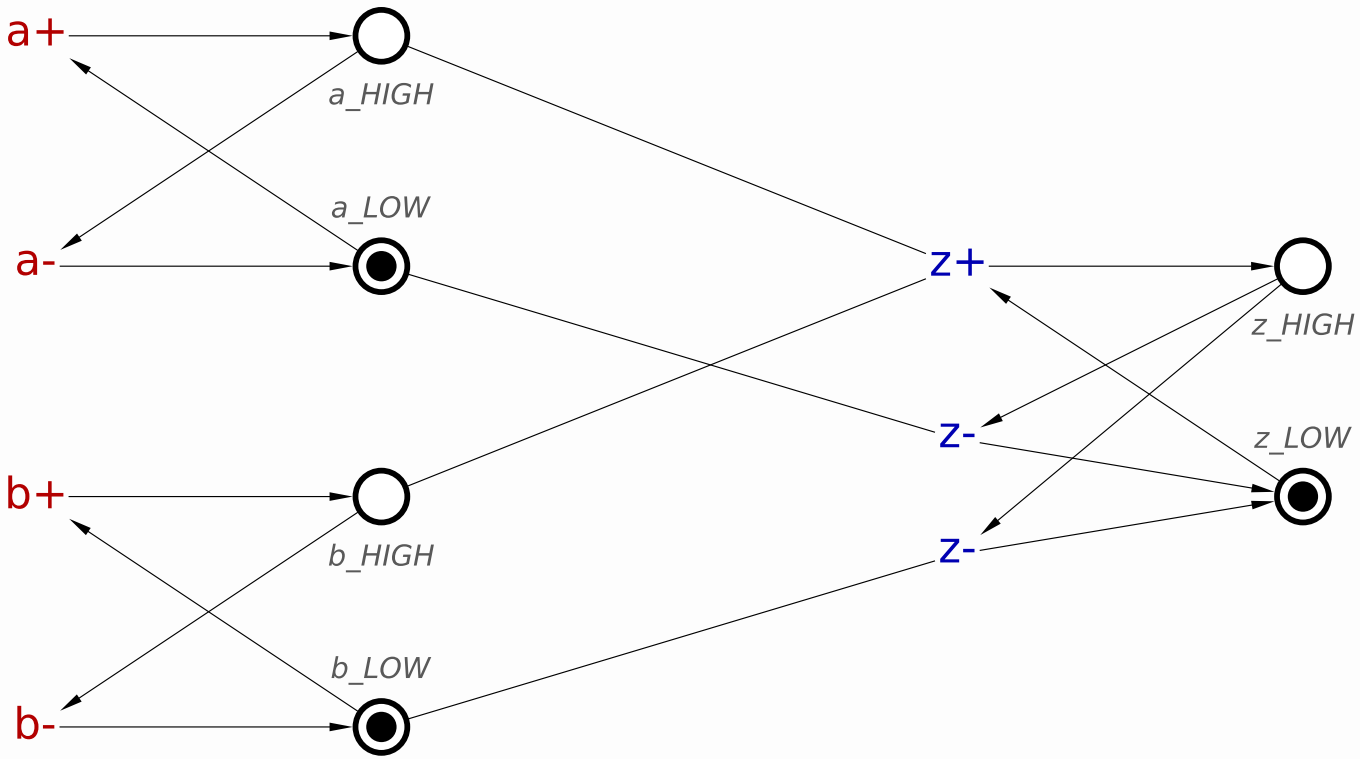
1. Circuit is converted to an equivalent STG – circuit STG
2. Internal signal transitions in the environment STG (contract between the circuit and its environment) are replaced by dummies
3. Circuit STG and environment STG are composed by PCOMP back-end
4. Conformation property is expressed in REACH language
5. Composed STG is unfolded by calling PUNF back-end
6. Unfolding prefix and REACH expression are passed to MPSAT back-end
7. Verification results are parsed by the front-end
8. Violation trace is projected to the circuit for simulation and debugging

Circuit design example

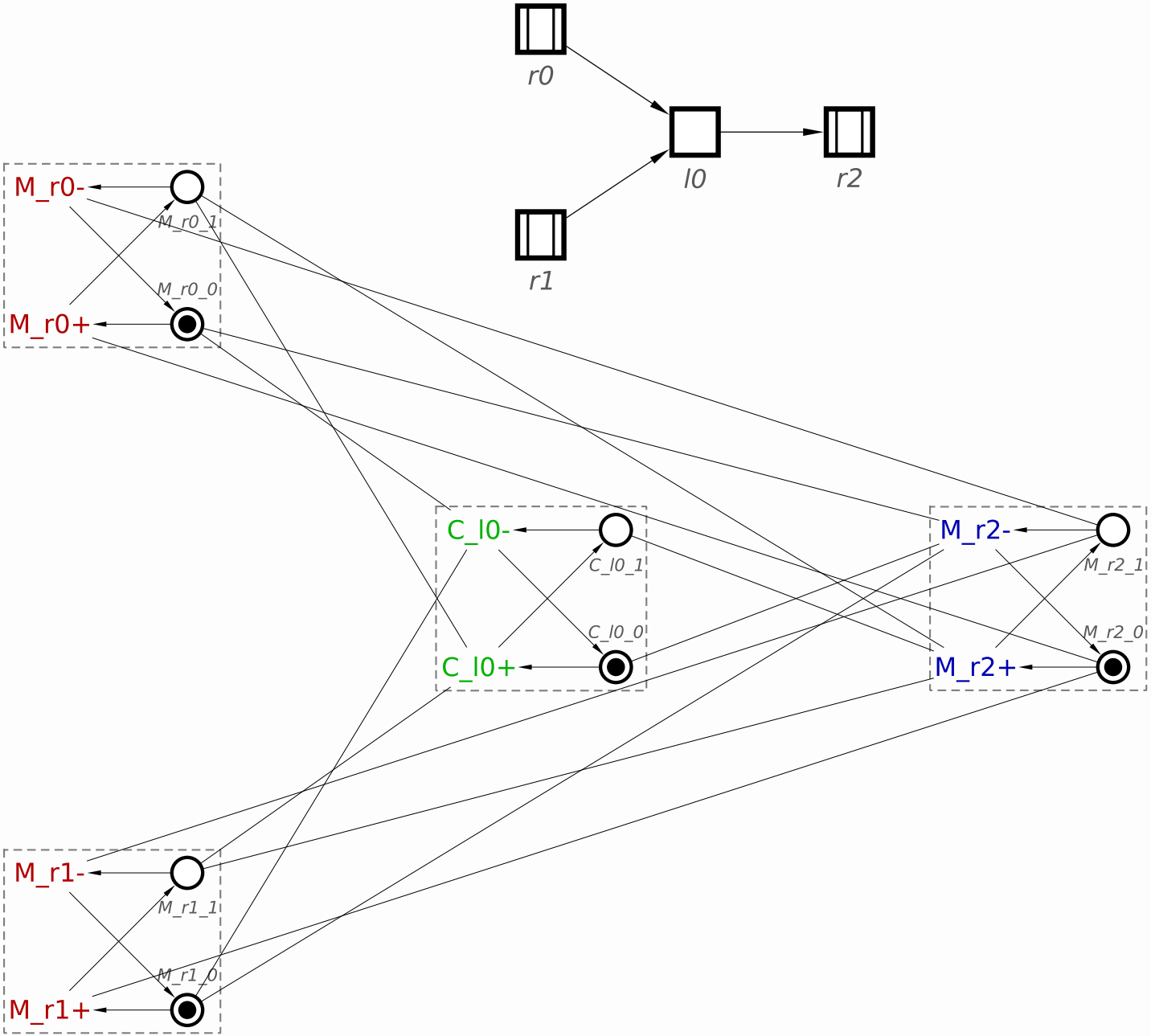


Circuit Petri nets as assembly language

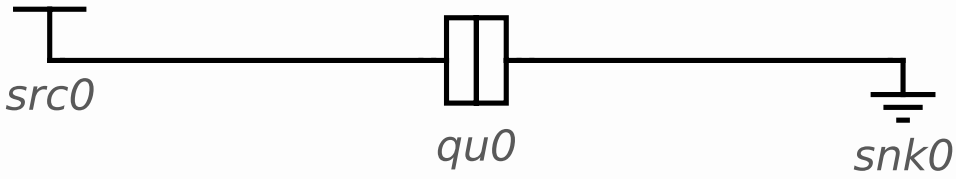


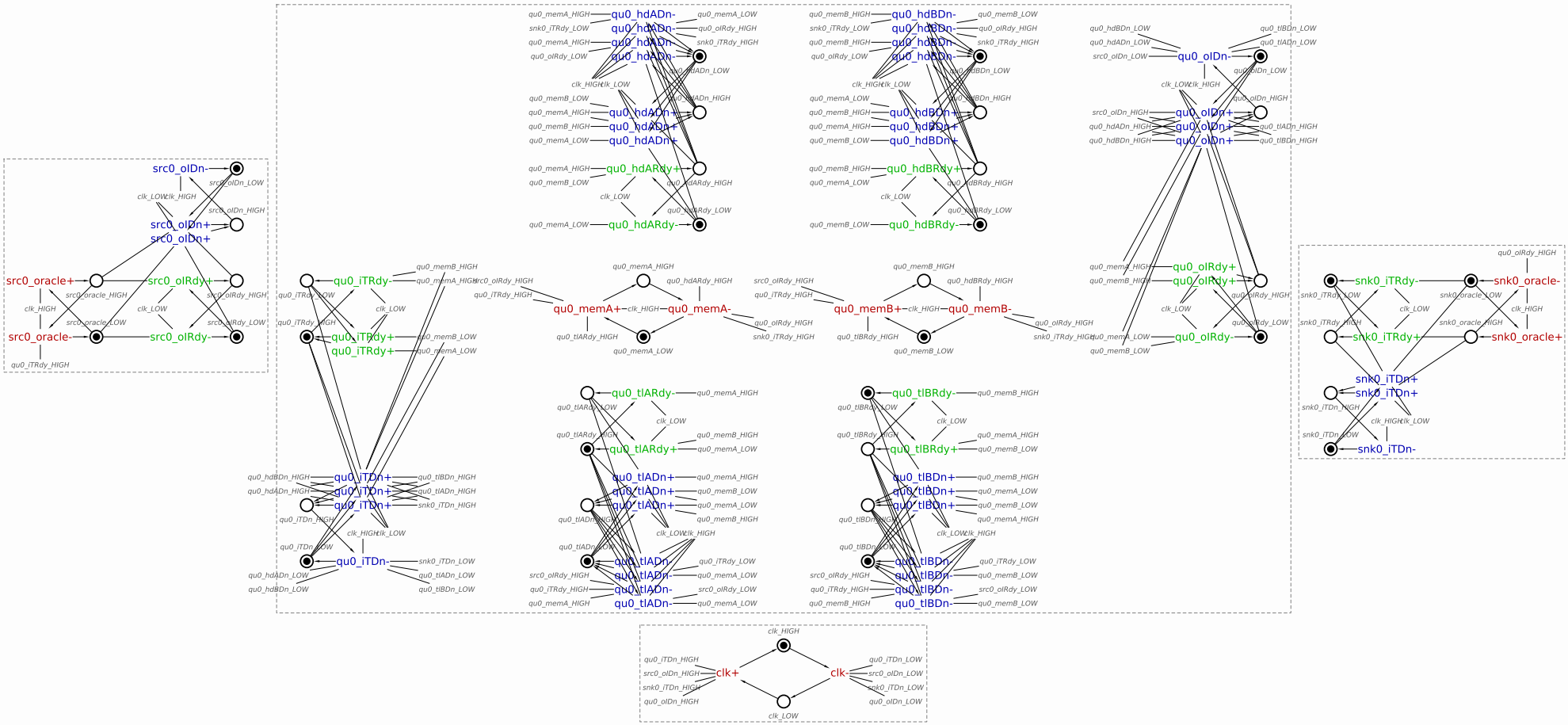


Circuit Petri nets: Dataflow pipelines



Circuit Petri nets: xMAS circuits





Plan for the day

- Morning practical – User interface and basic functionality (90 min)
  - Modelling concurrent vending machine
  - Dining philosophers problem
- Lunch and Learn (90 min)
  - Carving the Perfect Engineer, by Ian Phillips
- Afternoon practical – Design of asynchronous circuits (4 hours)
  - C-element (basic circuit, detailed explanation)
  - Buck controller (medium complexity with some hints)
  - VME bus controller (advanced material for individual work)
  - Analysis and optimisation of asynchronous pipelines
- Demonstration – Applications outside electronics (60 min)
  - Investigation of crime and accident scenes
  - Modelling biological systems

What formalisms will be covered?