Formal Analysis of a Fault-Tolerant Routing Algorithm for a Network-on-Chip

Zhen Zhang[1], Wendelin Serwe[2], Jian Wu[3], Tomohiro Yoneda[4], Hao Zheng[5], and Chris Myers[1]

[1]Dept. of Elec. & Comp. Eng., Univ. of Utah, Salt Lake City, UT 84112, USA

[2]Inria & Univ. Grenoble Alpes, LIG Inovallée 38334 St-Ismier Cedex, France

[3]Marvell Technology Group Ltd., Santa Clara, CA 95054, USA

[4]National Institute of Informatics, Tokyo, Japan

[5]Dept. of Comp. Sci. and Eng., Univ. of S. Florida, Tampa, FL 33620, USA

Motivating Application: Cyber-Physical Systems

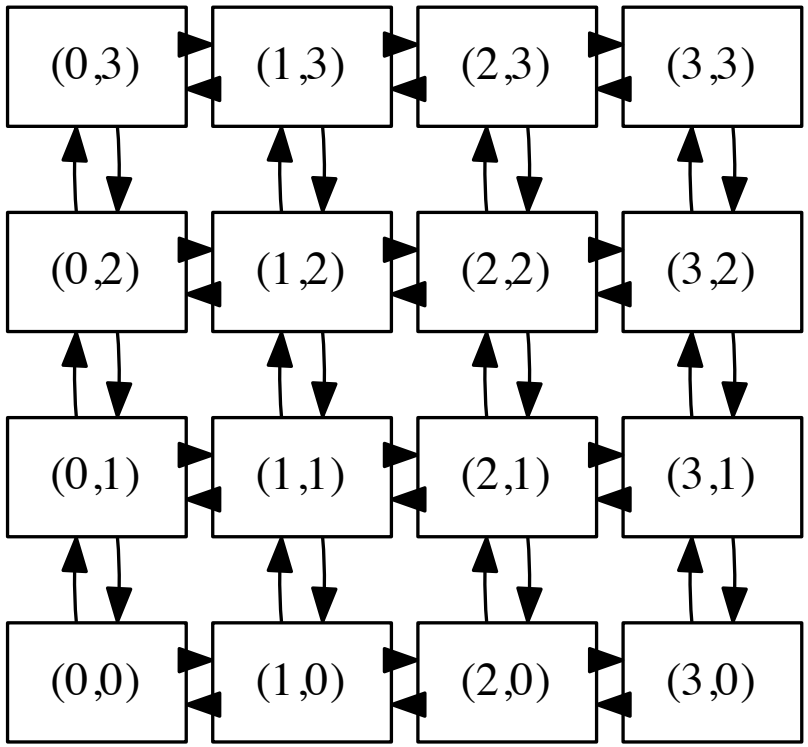
- A *Cyber-Physical System* (CPS) has tight interaction between computation and physical processes.
- Example: *Electronic control units* (ECUs) control everything (engine, brakes, drive-train, etc.) in automotive systems.
- An ECU uses some sensors and actuators to control a part of the physical system through a feedback loop.
- ECUs statically tied to a processor cannot share computing power and are subject to faults.
- Prof. Tomohiro Yoneda's research group at National Institute of Informatics (Japan) proposed a Network-on-Chip (NoC) approach for flexible mapping of ECUs onto the available processors.
- In this talk: experience report about the verification of different routing models with CADP.

Related Work

- precomputed route tables: not adaptive, only permanent faults
- Glass/Ni algorithm: faults of complete routers rather than single links
- GeNoC: formal proofs with ACL2
- ANOC/CHP: verified with CADP, but no fault-tolerance

Network-on-Chip Topology

- Two-dimensional mesh
- Nodes labeled by their position "(x, y)"
- Multi-flit wormhole routing



Properties for Verification

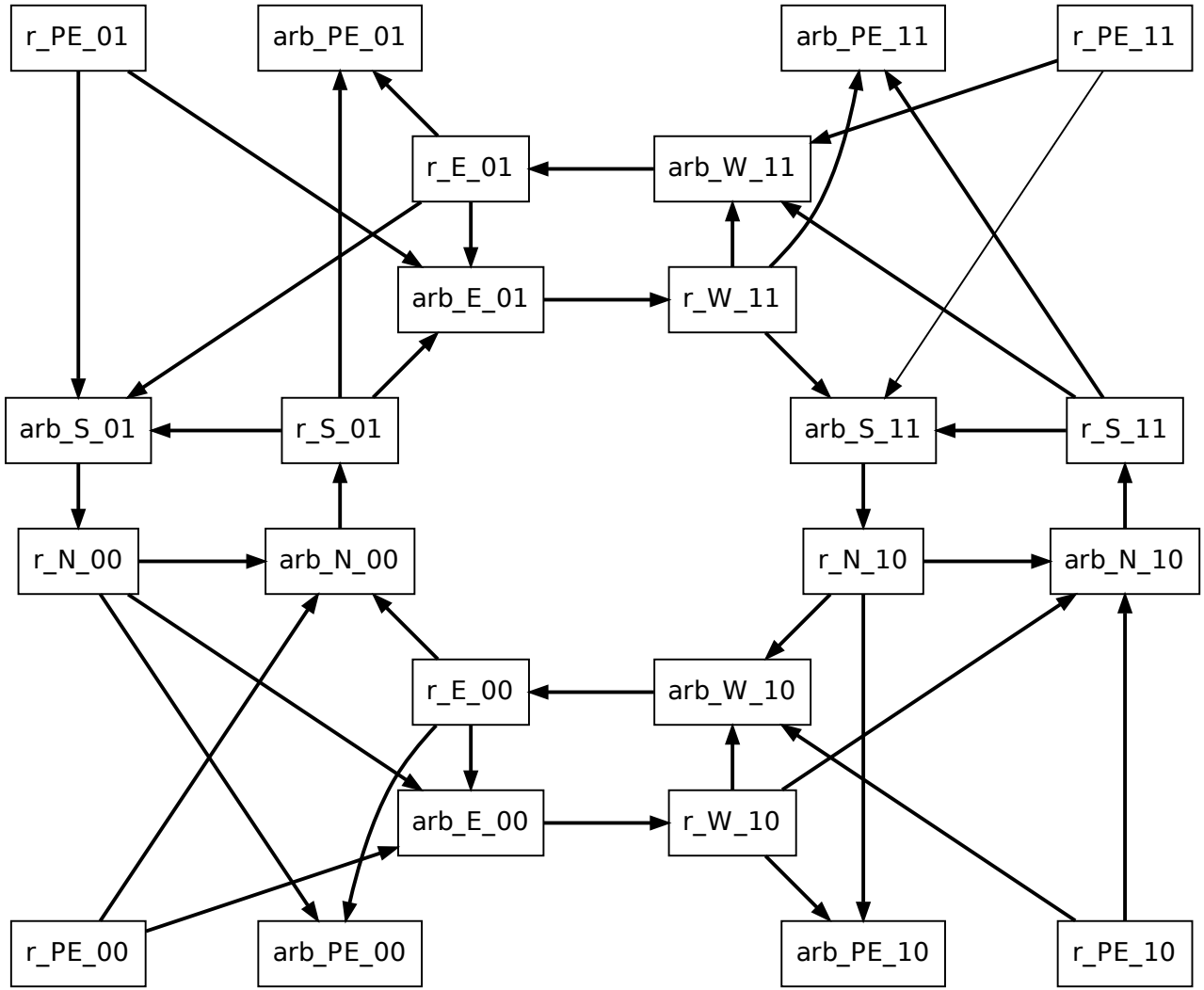
- The link-fault tolerant routing algorithm is free of deadlocks.
- Given at most one failure link, it is never the case that a router is unable to route a packet.
- Given at most one failure link, a packet never gets dropped when it is the only one packet in the network.

Plan of the Presentation

- Two-By-Two Mesh Architecture
- Principles of the Routing Algorithm
- Counter-Clockwise Routing Model
- Structural Abstraction to Reduce the State Space
- Data-Abstractions to Reduce the State Space
- Verification Results
- Conclusion

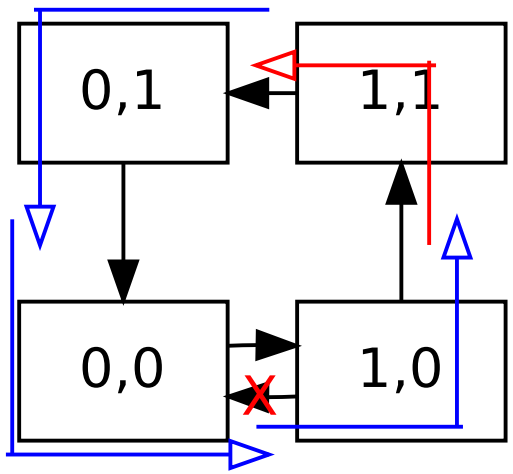
Two-by-Two Mesh Architecture

- 4 *nodes*
- each node with
  - 3 *arbiters* arb_*D*_*xy*
  - 3 *routers* r_*D*_*xy*
- *D*: direction (North, East, South, West, PE)
- *xy*: coordinates

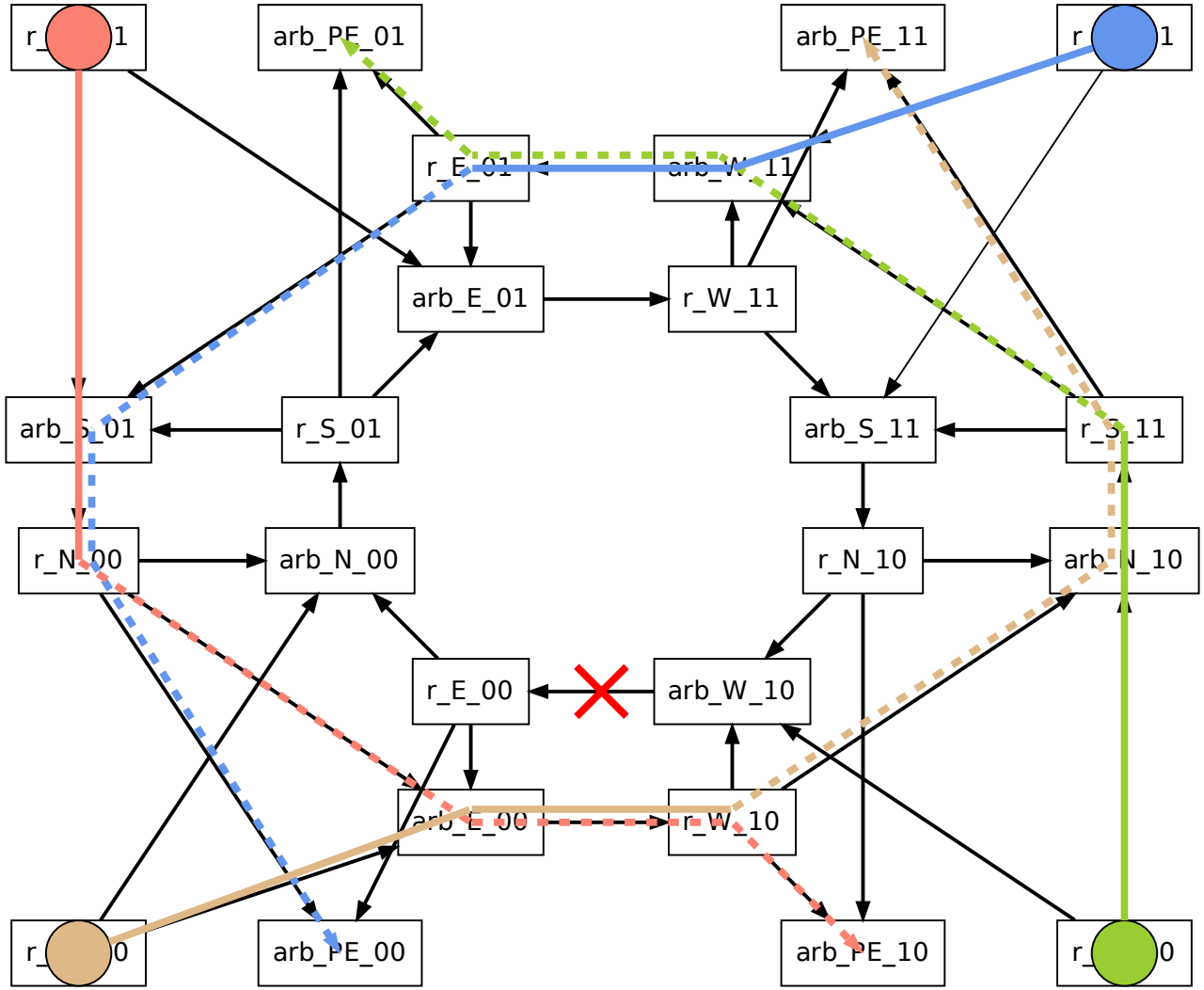


Principles of the Routing Algorithm

- Two-phase routing
  1. negative phase: towards south/west
  2. positive phase: towards north/east
- *Illegal turn*: switch from the positive phase back into the negative phase
- Tolerance of link faults: divert packets to take illegal turns
- Deadlock avoidance by dropping packets attempting illegal turns (if the path is occupied)

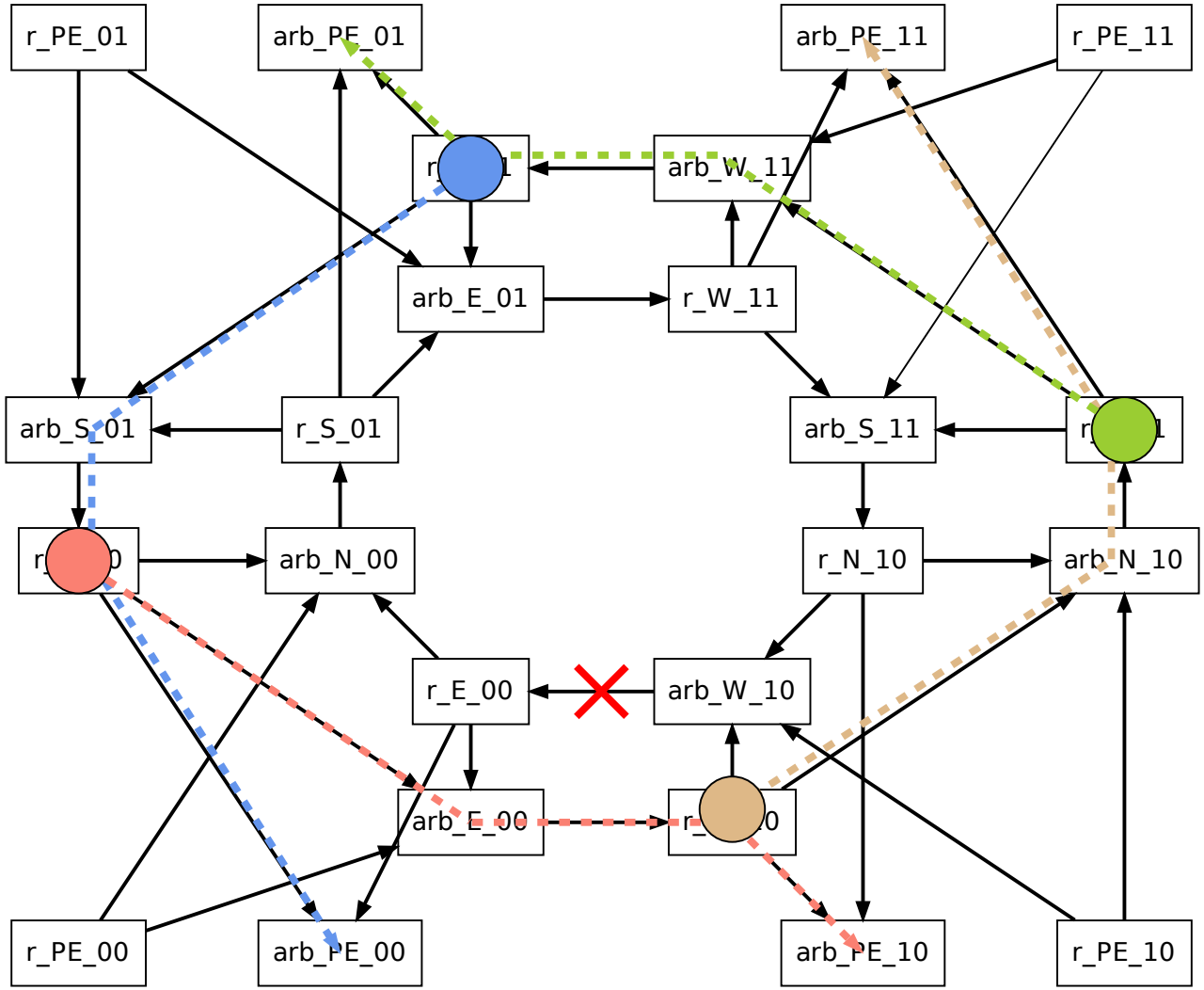


An Example of Cyclic Deadlock



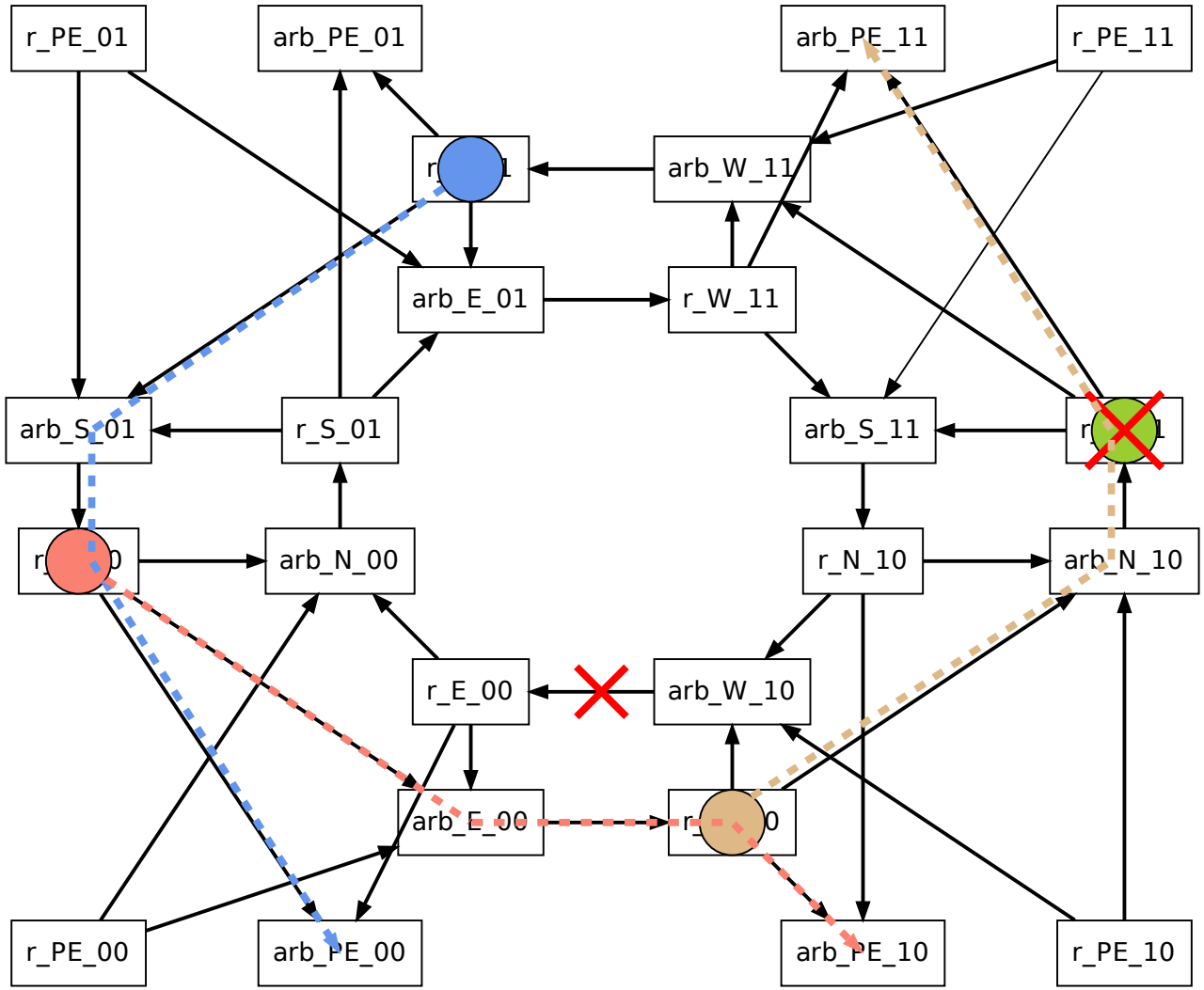
link failure “arb_W_10 -> r_E_00” implies diversion & illegal turn for green packet from r_ip_10 to arb_ip_01

An Example of Cyclic Deadlock



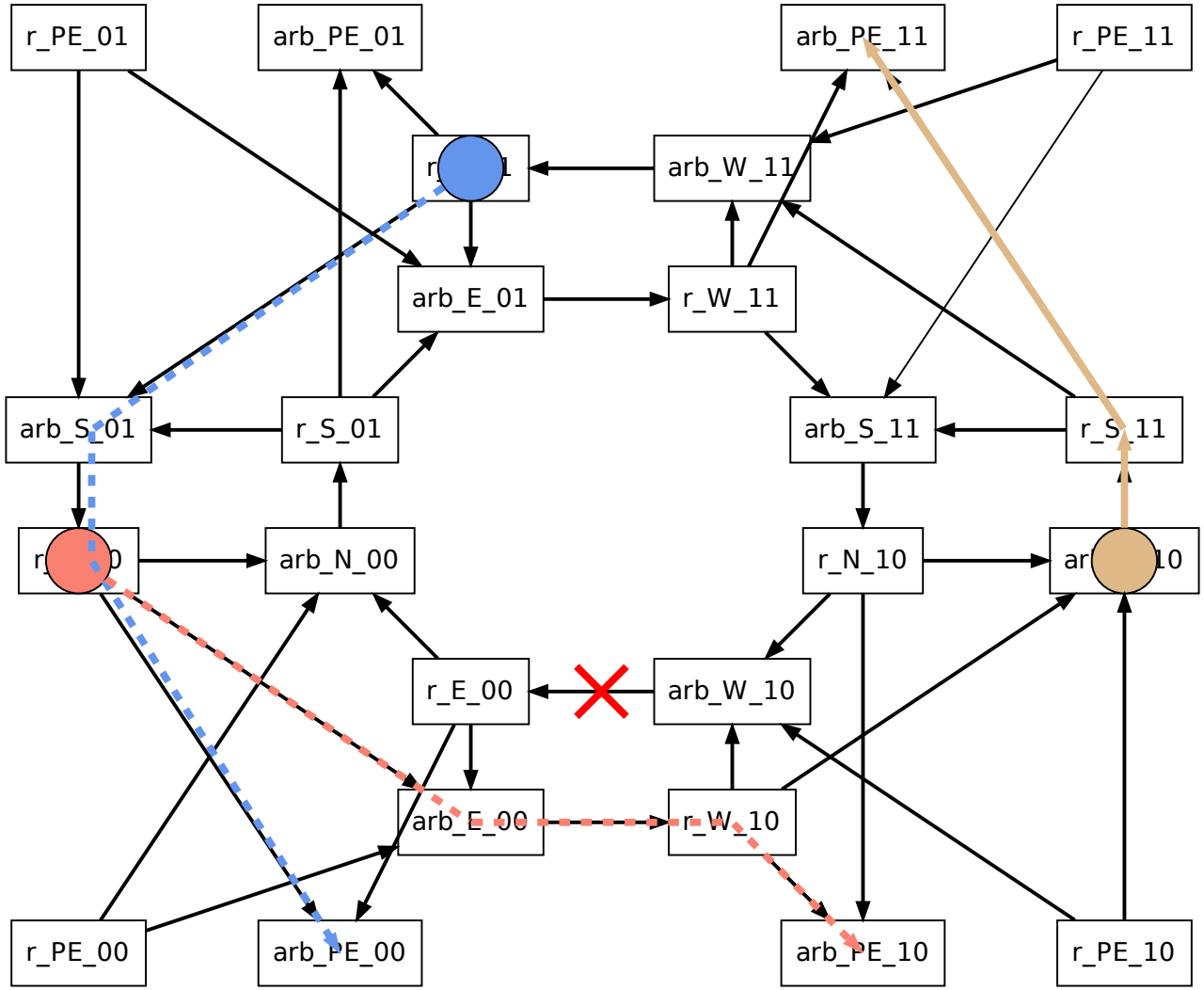
link failure “arb_W_10 -> r_E_00” implies diversion & illegal turn for green packet from r_ip_10 to arb_ip_01

Deadlock Avoidance: Example

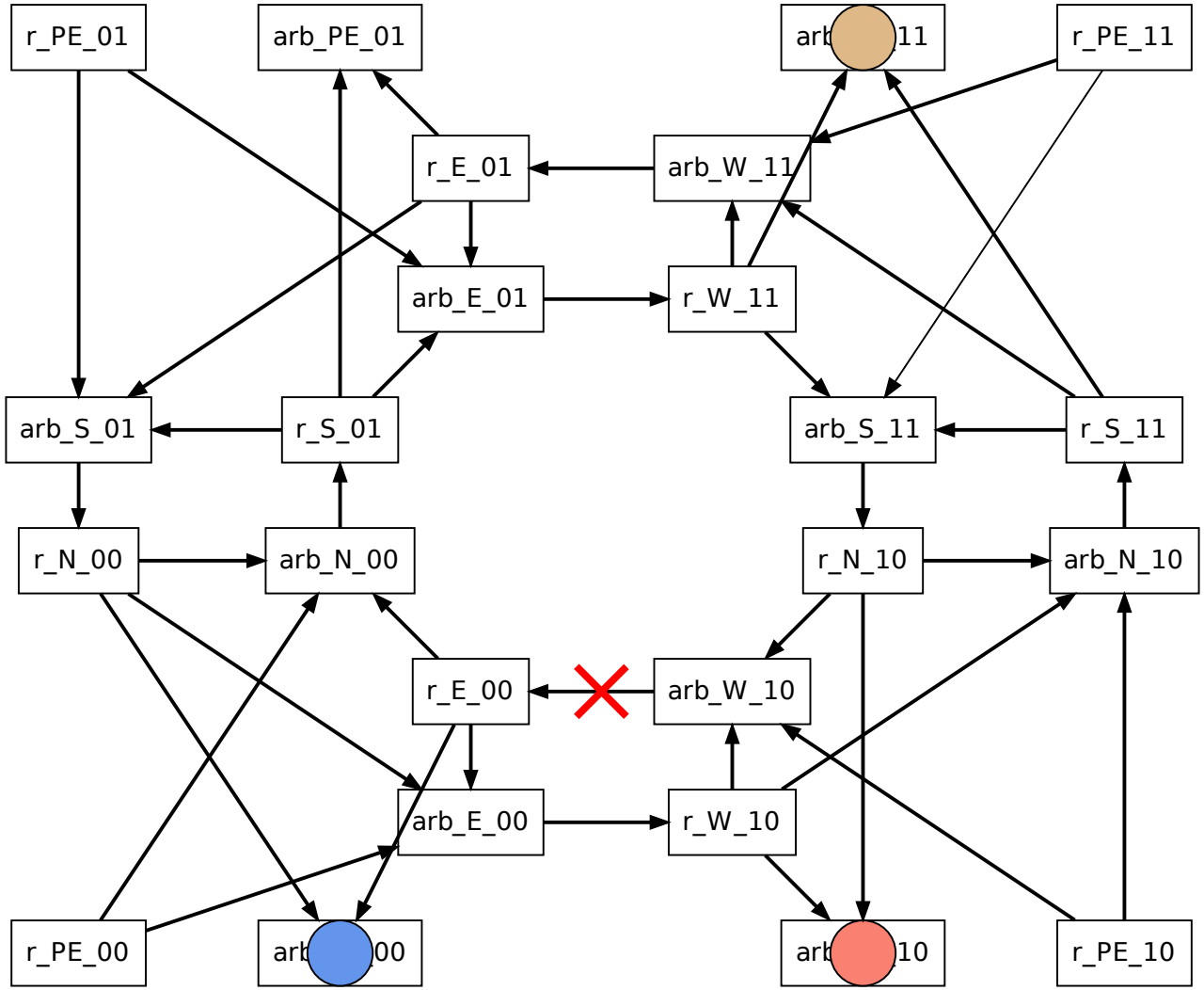


to avoid deadlock, drop green packet attempting an illegal turn

Deadlock Avoidance: Example



Deadlock Avoidance: Example



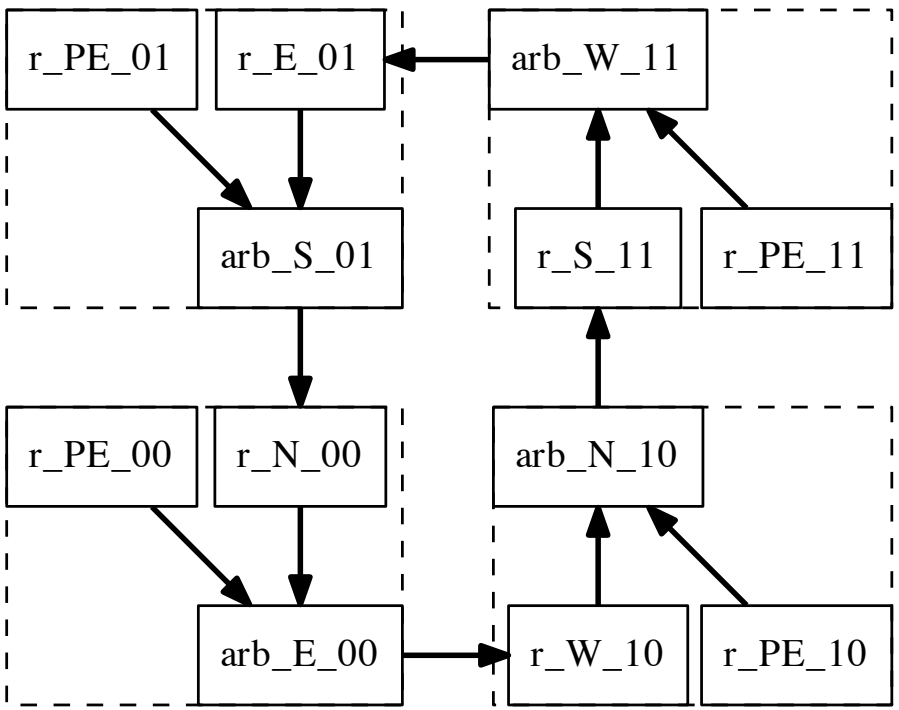
CADP (Construction and Analysis of Distributed Systems)

- Toolbox for the design of *asynchronous* systems
- Formal approach rooted in *concurrency theory*:
  process calculi, labeled transition systems, temporal logic
- Many verification techniques:
  simulation, model- and equivalence checking, compositional and distributed verification, test-case generation, performance evaluation, rapid prototyping
- Convenient languages for
  - system modeling (*LNT*),
  - temporal logic properties (*MCL*), and
  - execution of verification scenarios (*SVL*)
- More than 150 published case-studies and 70 third-party tools

For more information: `http://cadp.inria.fr`

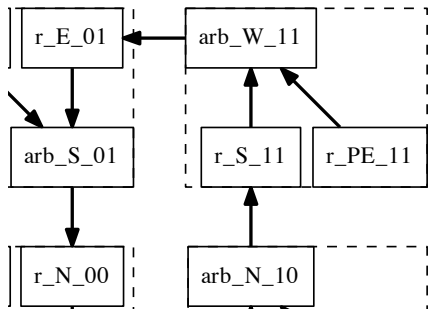
A Counterclockwise Routing Model

- Each PE router only generates one single-flit packet destined to the node in its diagonal direction.
- Only the counterclockwise routing direction is available.
  - No packet forwarding computation in the routers.
  - Packets take the north-west illegal turn in node (1,1).



LNT Model of the West Arbiter of Node (1,1) — arb_W_11

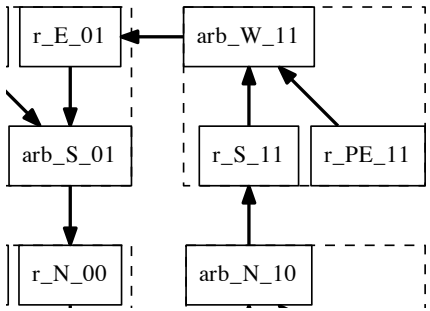
```
process arbiter_nack [ca, ra, ar : any] is
  loop
    var one_flit : Nat in
      select
        ar;                 -- Router r_E_01 is ready to accept packet
        select
          ca(?one_flit);    -- Receive packet from r_ip_11
          ar(one_flit)      -- Send packet to r_E_01
        []
          ra(true);         -- Ready to accept from r_S_11
          ra(?one_flit);    -- Receive packet from r_S_11
          ar(one_flit)      -- Send packet to r_E_01
        end select
      []
        ra(false)           -- Send negative acknowledgment to r_S_11
      end select
    end var
  end loop
end process

arbiter_nack [r_ip_11, r_S_11, r_E_01]   -- arb_W_11
```



LNT Model of the South Router of Node (1,1) — r_S_11

```
process router_drop_pkt [ar, ra : any] is
  loop
    var status : Bool, one_flit : Nat in
      ar;               -- Ready to accept packet from arb_N_10
      ar(?one_flit);    -- Receive packet from arb_N_10
      ra(?status);      -- Request arb_W_11's status
      if status then
        ra(one_flit)    -- Send packet to arb_W_11 ONLY on TRUE status
      end if
    end var
  end loop
end process
```

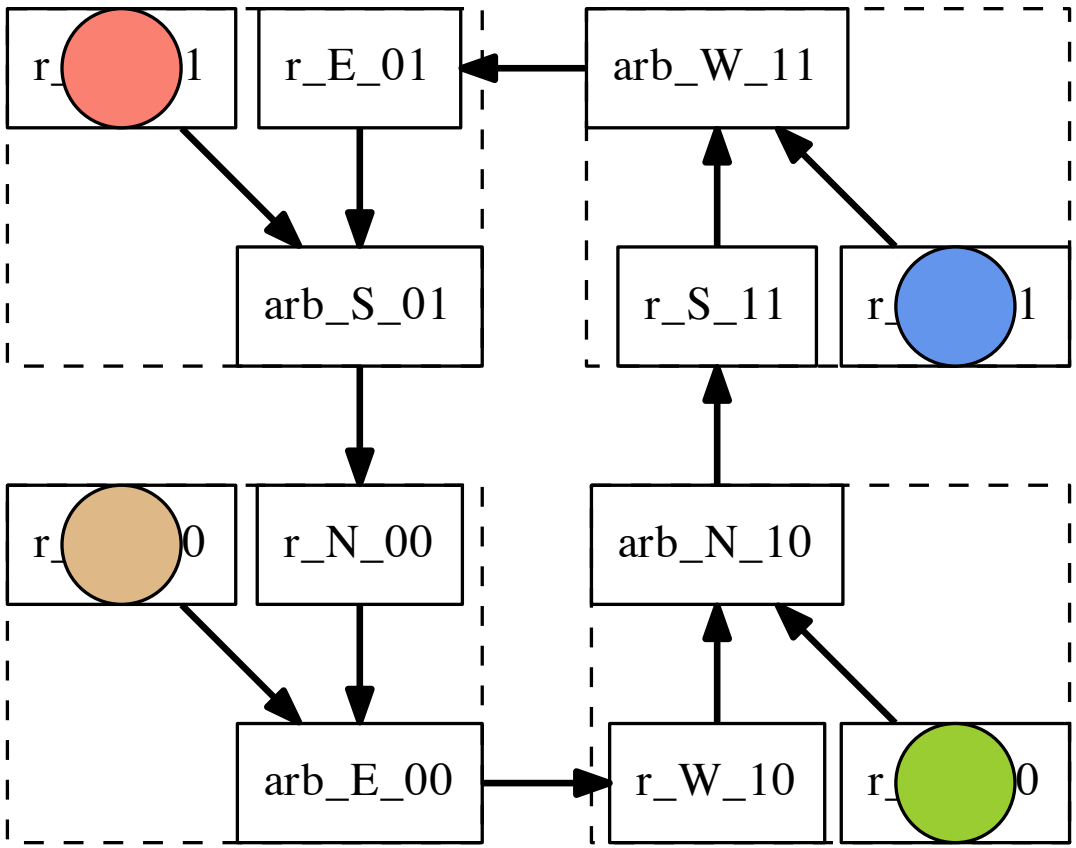
```
router_drop_pkt [arb_N_10, arb_W_11]   -- r_S_11
```



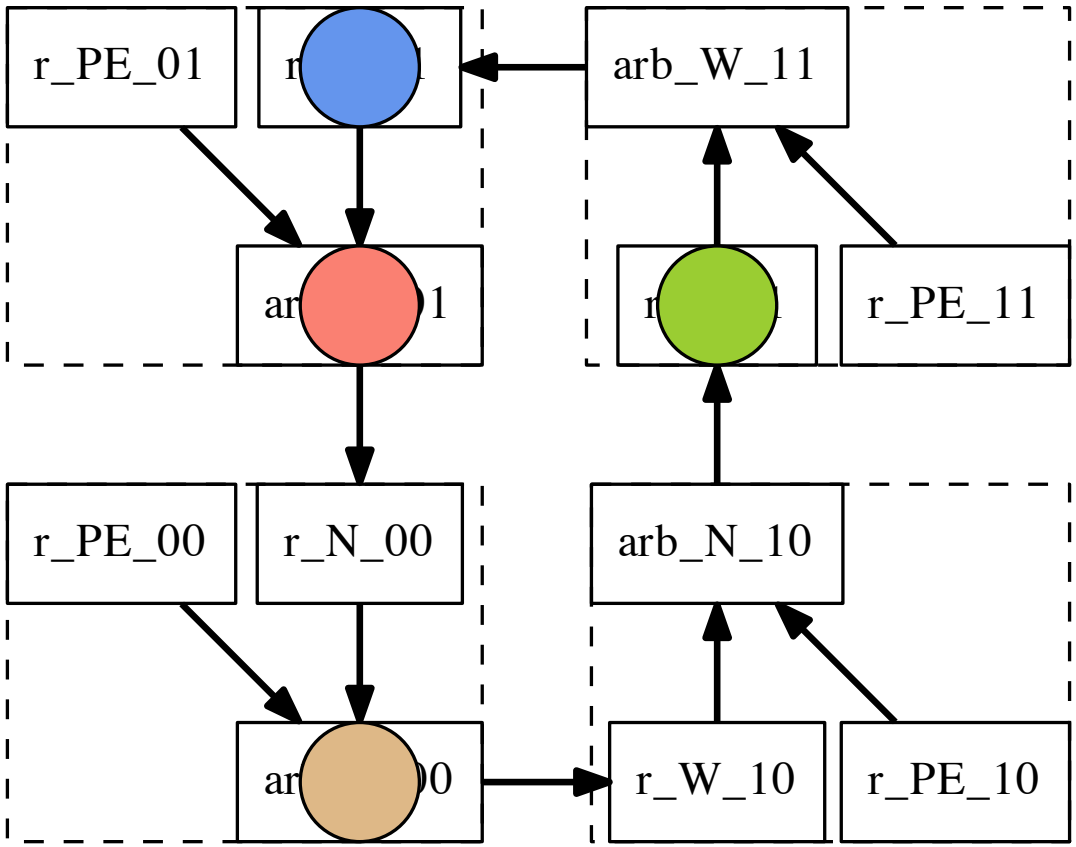
Verification Results for the Counterclockwise Routing Model

- No reduction techniques are required for state space generation.
- Deadlock avoidance mechanism drops packets making an illegal turn.
- Expected worst case:
  - three packets make illegal turns and get dropped;
  - at least one packet remains cycling in the network.
- *Reachability analysis finds deadlocks!*
- Diagnostic sequence (“packet leakage path”):
  All four packets get dropped due to deadlock avoidance.

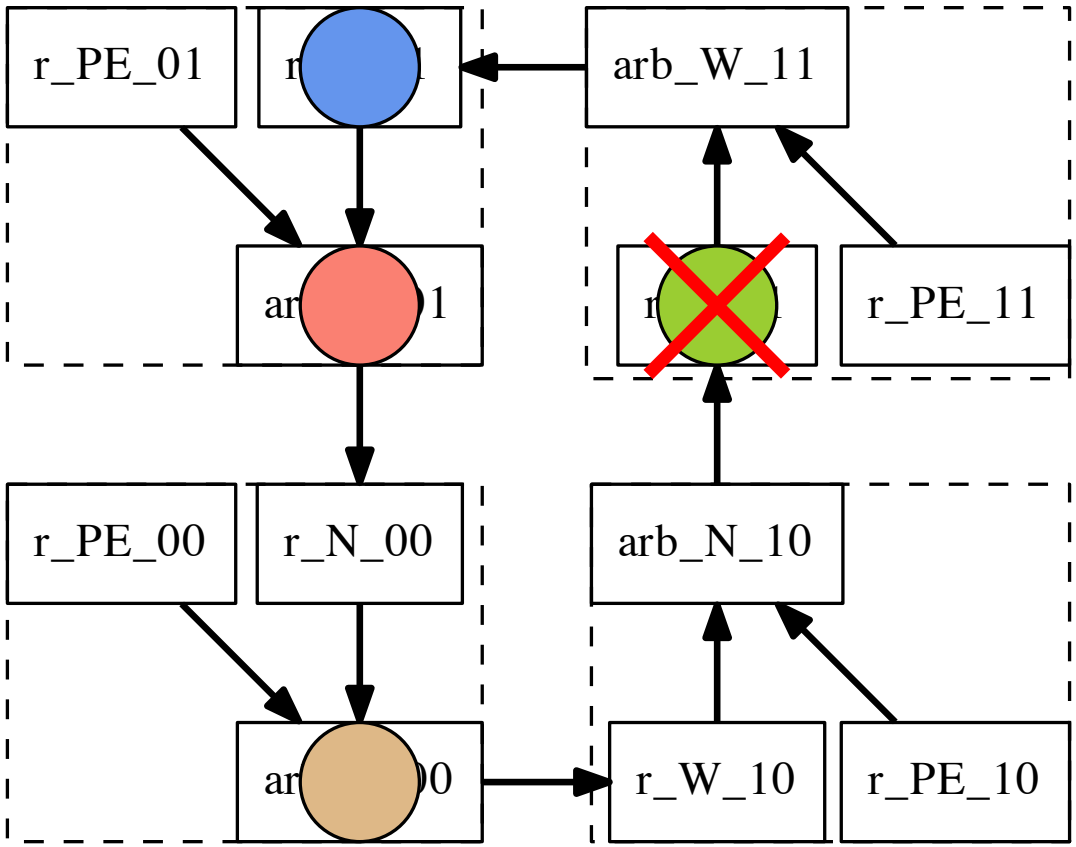
Packet Leakage Transition Sequence



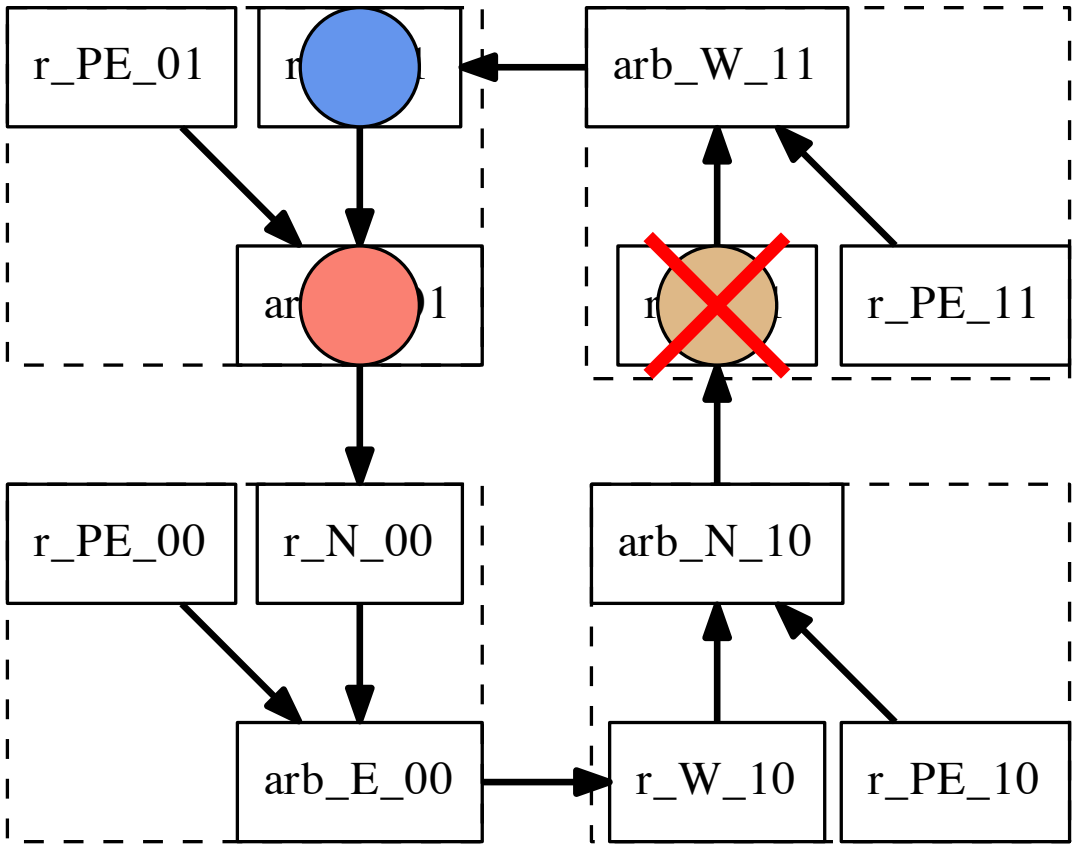
Packet Leakage Transition Sequence Example



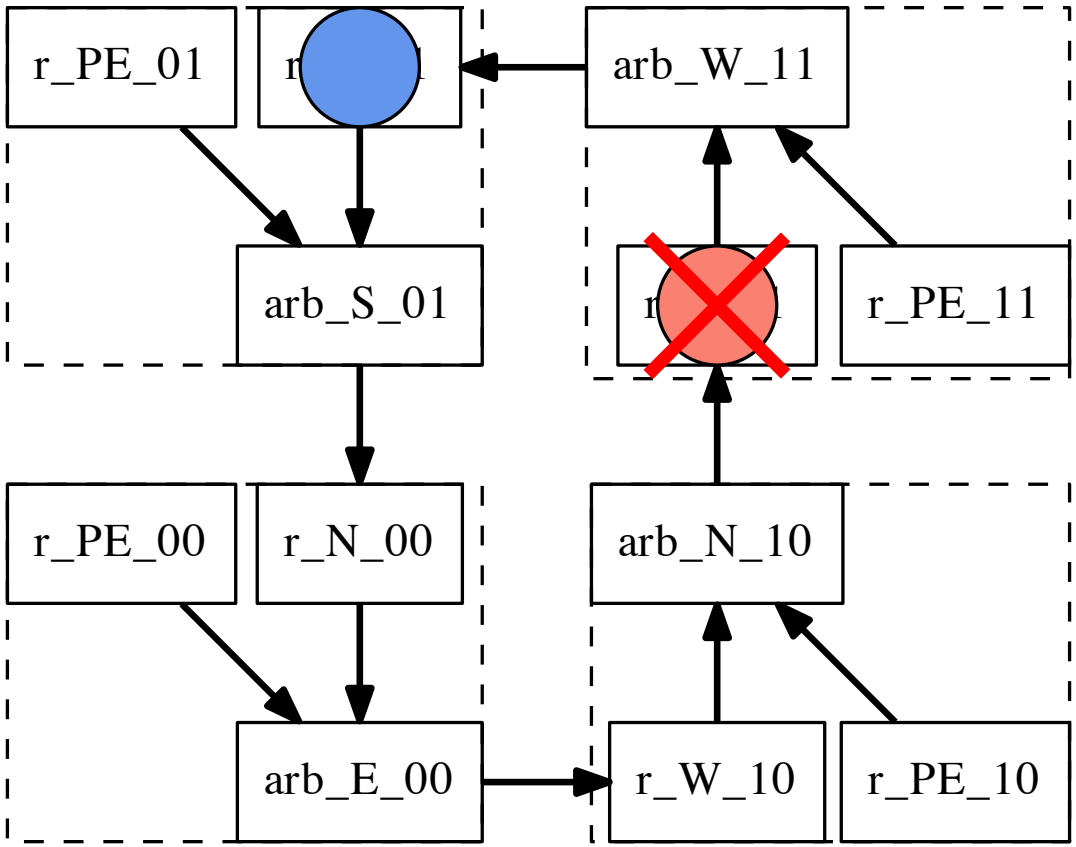
Packet Leakage Transition Sequence Example



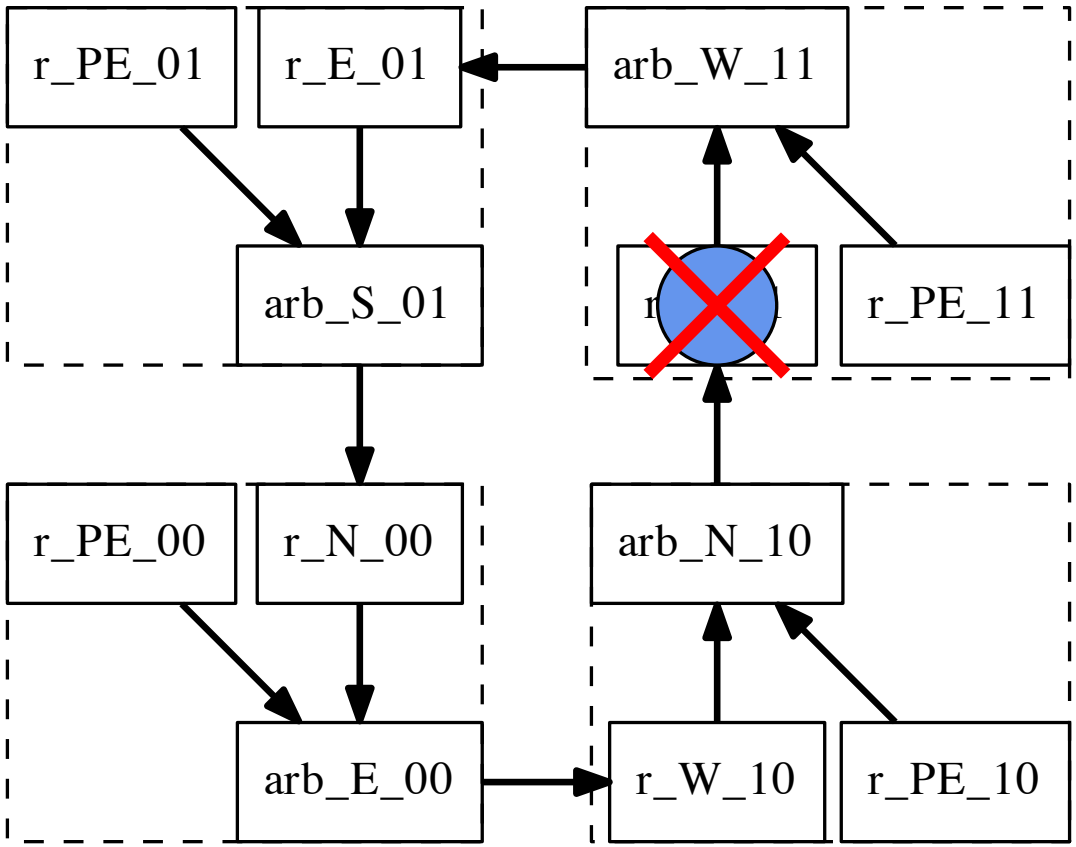
Packet Leakage Transition Sequence Example



Packet Leakage Transition Sequence Example

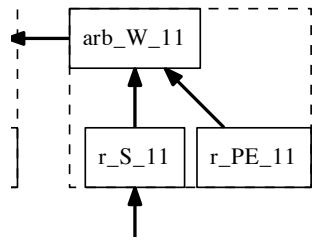


Packet Leakage Transition Sequence Example



Reason for Packet Leakage in the West Arbiter of Node (1,1)

```
process arbiter_nack [ca, ra, ar : any] is
  loop
    var one_flit : Nat in
      select
        ar;                  -- Router r_E_01 is ready to accept packet
        select
          ca(?one_flit);     -- Receive packet from r_ip_11
          ar(one_flit)       -- Send packet to r_E_01
        []
          ra(true);          -- Ready to accept from r_S_11
          ra(?one_flit);     -- Receive packet from r_S_11
          ar(one_flit)       -- Send packet to r_E_01
        end select
      []
        ra(false)            -- Send negative acknowledgment to r_S_11
      end select
    end var
  end loop
end process

arbiter_nack [r_ip_11, r_S_11, r_E_01]   -- arb_W_11
```



Sending a negative acknoweldegment is *always* enabled!

Packet Leakage in the West Arbiter of Node (1,1)

- Arbiter arb_W_11: nondeterministic choice to send to r_S_11
  - a positive acknowledgement
  - a negative acknowledgement
- Negative acknowledgment always possible, regardless of potential deadlocks:
  a packet leakage path exists!

Possible solution: *prioritized choice: sending a positive acknowledgement with higher priority*

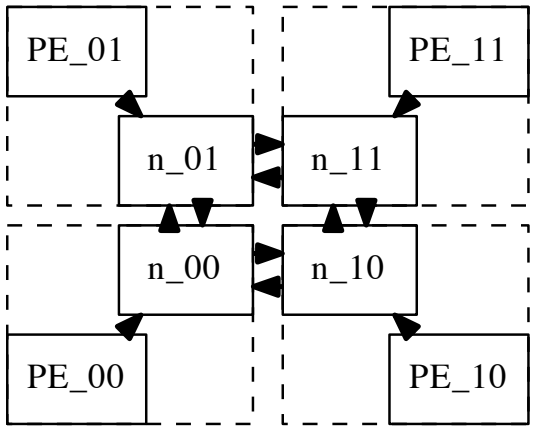
Drawbacks of the priority-based approach:

- LNT implementation of the priority choice requires additional processes, leading to possible state explosion.
- Pruning the unwanted execution paths from the generated state space using the priority operator in EXP.OPEN/SVL.
- The divergence-sensitive branching bisimulation is not a congruence for the priority operator.

Structural Abstraction to Reduce the State Space

- State explosion becomes a problem once all components are built for the two-by-two mesh.
- One major reason: interleavings of gate rendezvous between connected routers and arbiters.
- Idea: Merge routers and arbiters of one node into one process, completely removing rendezvous internal to a node.

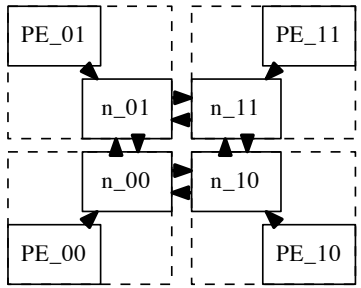
- Drawbacks of this simplification:
  - It removes the possibility of multiple packets passing through a single node at the same time.
  - It removes the buffering capacity of each arbiter, which *causes deadlock*.



Example of Deadlock

```
process n00 [ip_00, ...] is
  loop
    var one_flit : Nat in
      select
          ip_00(?one_flit);
          if one_flit == 1 then
            n00_n01(one_flit)
          else
             ...
          end if
      []
          n01_n00(?one_flit);
          ...
      []
          n10_n00(?one_flit);
          ...
      end select
    end var
  end loop
end process
```

```
process n01 [ip_01, ...] is
  loop
    var one_flit : Nat in
      select
          ip_01(?one_flit);
          if one_flit == 0 then
            n01_n00(one_flit)
          else
           ...
          end if
      []
          n00_n01(?one_flit);
          ...
      []
          n11_n01(?one_flit);
          ...
      end select
    end var
  end loop
end process
```



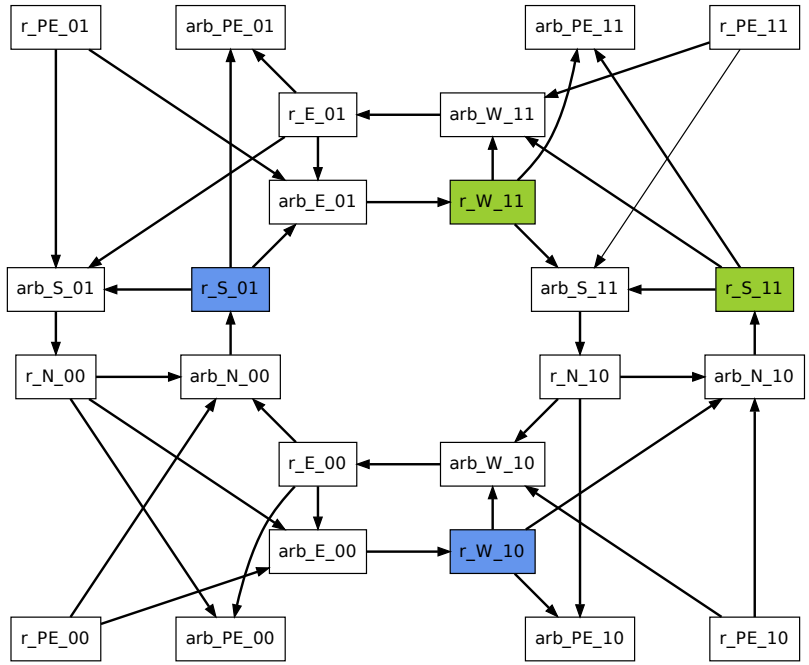

```
par
    n00_n01, n01_n00, ... -> n00 [...]
|| n00_n01, n01_n00, ... -> n01 [...]
end par
```

Finding Proper Data Abstractions

- The counter-clockwise routing experiment reveals the packet leakage in certain arbiter's design.
- The experiment with bufferless arbiters shows the necessity of arbiter's buffering ability.
- Data abstraction is required to keep the state space manageable.
- The packet's content is *only* checked by routers to precisely determine the next forwarding direction.
- Each router may potentially send a randomly destined packet to any possible forwarding direction.
- Abstract the precise forwarding direction decision with *nondeterministic choice*.
- This abstraction results in manageable state space.

Finding Proper Data Abstractions

- Constraints:
  - Fault-tolerance provides alternative route(s) for all packets.
  - Router's nondeterministic choice should allow all forwarding directions for a randomly destined packet.
  - Illegal turns are never the preferred choice unless all forwarding routes of a router are illegal.
- Routers are classified into three types:
  - RI2 can make only one illegal turn: r_S_11 and r_W_11;
  - RI1 can make one illegal turn and one legal turn: r_W_10 and r_S_01;
  - RI0 makes no illegal turn: all other routers.



Packets Without Data

- Extreme abstraction: eliminate the packet entirely, use pure synchronization
- The packet leakage still exists at RI2 (router allowing only illegal turns):
  - RI2 has two choices for a packet: to drop it or forward it on an illegal turn.
  - With no packet content, the nondeterministic choice may always take the illegal turn.
  - An illegal turn leads to packet drop regardless of deadlock avoidance.
- A packet takes an illegal turn *only* after an unsuccessful attempt to take route due to a failure on the route.
- Stated otherwise: When a packet makes an illegal turn, it must have been diverted at least once before.
- Use a single-bit Boolean variable to model packet data: true iff packet has been diverted before.

The LNT Process for RI2

```
process router_two_illegal [input, out_arb_ip, out1_illegal,
                            out2_illegal, drop : any] is
  var one_flit, arb_status : Bool in loop
    input(?one_flit);
    select
      out_arb_ip(one_flit)
    []
      if one_flit == true then
        -- priority to out1_illegal
        out1_illegal(?arb_status);
        if arb_status == true then out1_illegal(one_flit)
        else
          out2_illegal(?arb_status);
          if arb_status == true then out2_illegal(one_flit)
          else drop -- both illegal turns impossible
          end if
        end if
      else stop end if
    end select
  end loop end var
end process
```

The LNT Process for The Arbiter Corresponding to RI2

```
process arbiter_nack_2 [in_ip_router, in1_illegal,
                        in2_illegal, arb_out : any] is
var one_flit : Bool in
   loop select
      in_ip_router(true); in_ip_router(?one_flit);
      loop L1 in select
         arb_out(one_flit); break L1
      [] in1_illegal(false)
      [] in2_illegal(false)
      end select end loop -- L1
   [] in1_illegal(true); in1_illegal(?one_flit);
      loop L2 in select
         arb_out(one_flit); break L2
      [] in1_illegal(false)
      [] in2_illegal(false)
      end select end loop -- L2
   [] in2_illegal(true); in2_illegal(?one_flit);
      loop L3 in select
         arb_out(one_flit); break L3
      [] in1_illegal(false)
      [] in2_illegal(false)
      end select end loop -- L3
   end select end loop
end var end process -- arbiter_nack_2
```

Compositional LTS generation for Two-by-Two NoCs

- Nine scenarios: without failed link plus all possible single link failures
- SVL scripts to automate compositional LTS generation and reduction
- No deadlocks found in any of the generated LTSs
- All but route-failure and drop gates hidden:
  - No route failures: no corresponding label in the LTSs
  - No packet drop for a single packet in the NoC
  - Packet drop possible whenever more than one packet in the NoC

Failure	Interm. LTS Size		Final LTS		Performance		Labels
Link	States	Transitions	St.	Tr.	RAM	Time	
none	6,295,773	83,386,208	1	1	32,945	5,976	i
01 $\rightarrow$ 00	20,340	193,726	41	224	111	83	i, drop_Sr_11, drop_Wr_11
01 $\rightarrow$ 11	1,369,068	18,221,153	1	3	4,039	499	i, drop_Sr_01, drop_Sr_11
00 $\rightarrow$ 10	6,560	50,688	21	104	111	80	i, drop_Sr_11, drop_Wr_11
00 $\rightarrow$ 01	6,560	50,688	21	104	111	81	i, drop_Wr_11, drop_Sr_11
10 $\rightarrow$ 11	122,724	1,269,981	1	3	111	89	i, drop_Wr_10, drop_Wr_11
10 $\rightarrow$ 00	20,340	193,726	41	224	111	80	i, drop_Wr_11, drop_Sr_11
11 $\rightarrow$ 01	367,200	4,172,652	1	3	111	106	i, drop_Sr_11, drop_Wr_11
11 $\rightarrow$ 10	367,200	4,172,652	1	3	111	105	i, drop_Sr_11, drop_Wr_11

Conclusion & Future Work

- Formal Analysis facilitates detection of design flaws (packet leakage)
- Gain deeper understanding of the routing algorithm (necessary buffering capacity of the arbiters)

- Extension of the LNT model to larger networks (three-by-three, four-by-four, ...), where nodes differ in the number of connections
- Study packet delivery guarantees
- Refine abstractions
- Can the discovered symmetries help in fighting the state space explosion?
- Enrich the model to enable performance analysis