Formal architectural modelling

Opportunities and challenges

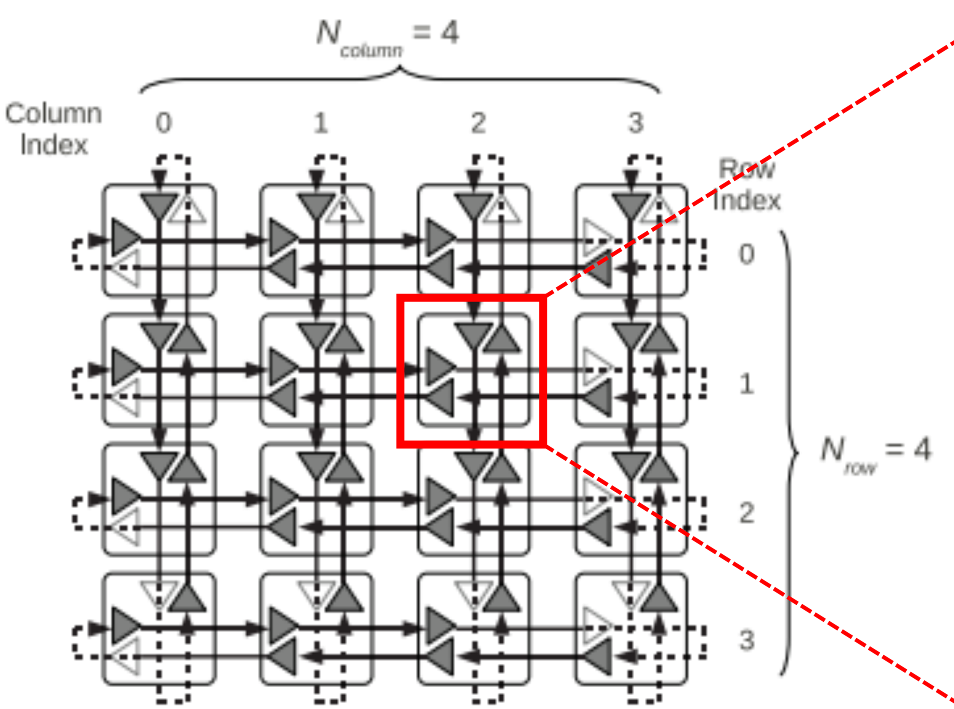
Julien Schmaltz

joint work with
Perry van Wesel and Alexander Fedotov

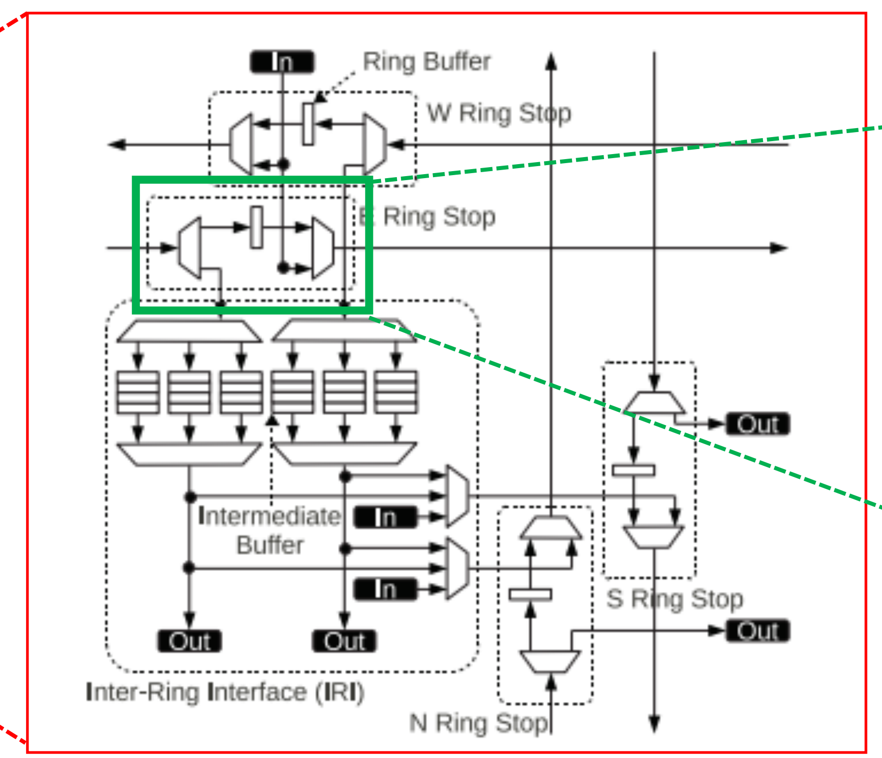
Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

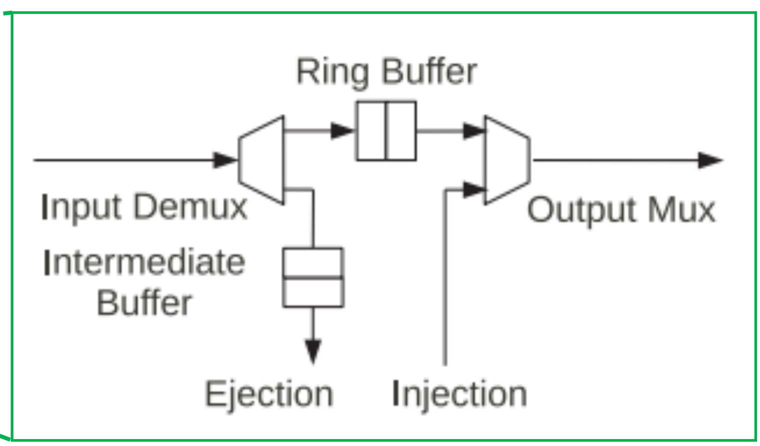
Architectural model example: TornadoNoC (1)



Network



Router

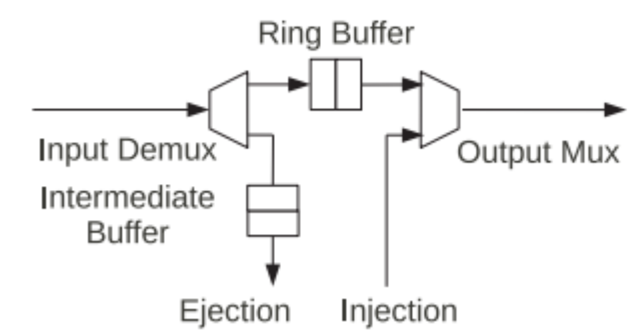


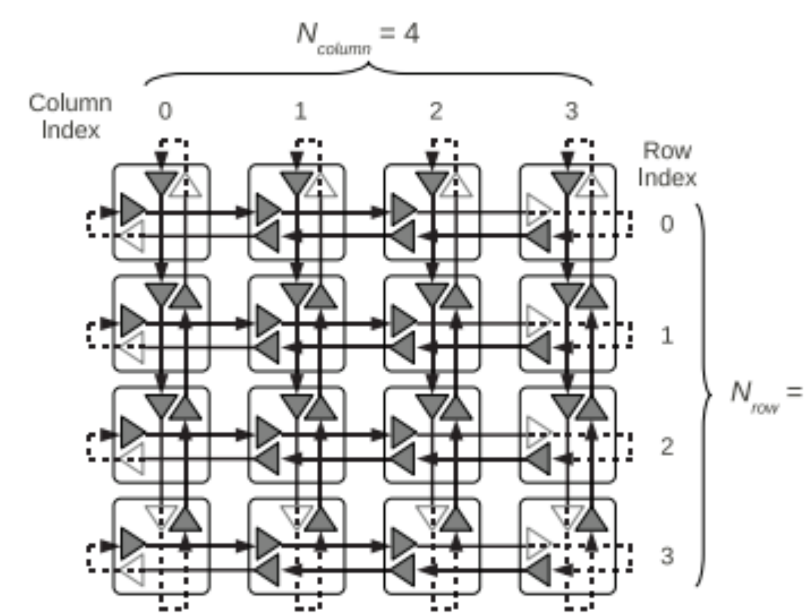
Ring stop

Architectural model example: TornadoNoC (II)

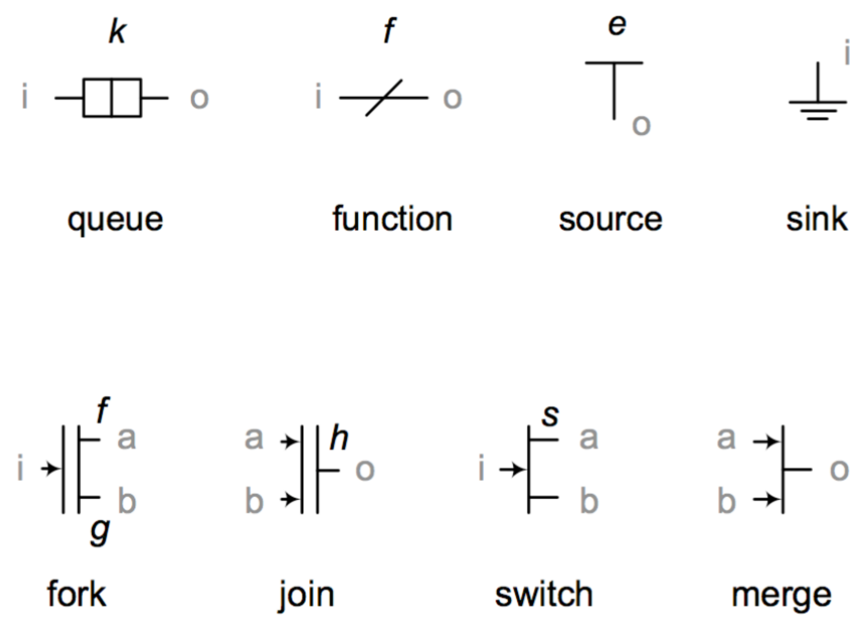
Additional mechanisms:

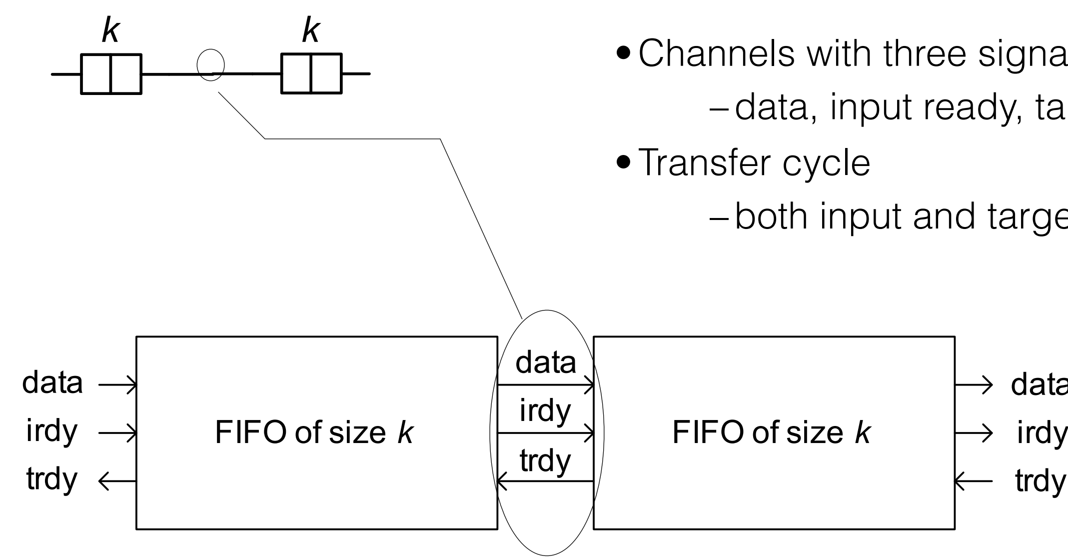
- Spillover – Travel an additional lap to avoid blocking other traffic.
- Sequence numbering – Limit the number of laps for one packet.
- Injection constraint – Limit number of packets in a ring.
- Inflation mechanism - Avoid protocol deadlock and starvation.



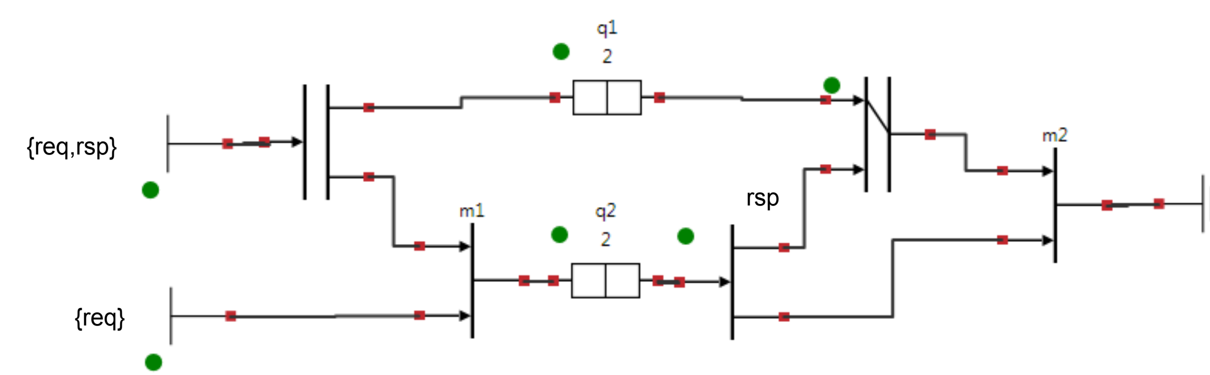


MaDL basic principles - Originates from Intel’s xMAS





- Channels with three signals
  - data, input ready, target ready
- Transfer cycle
  - both input and target are "true"



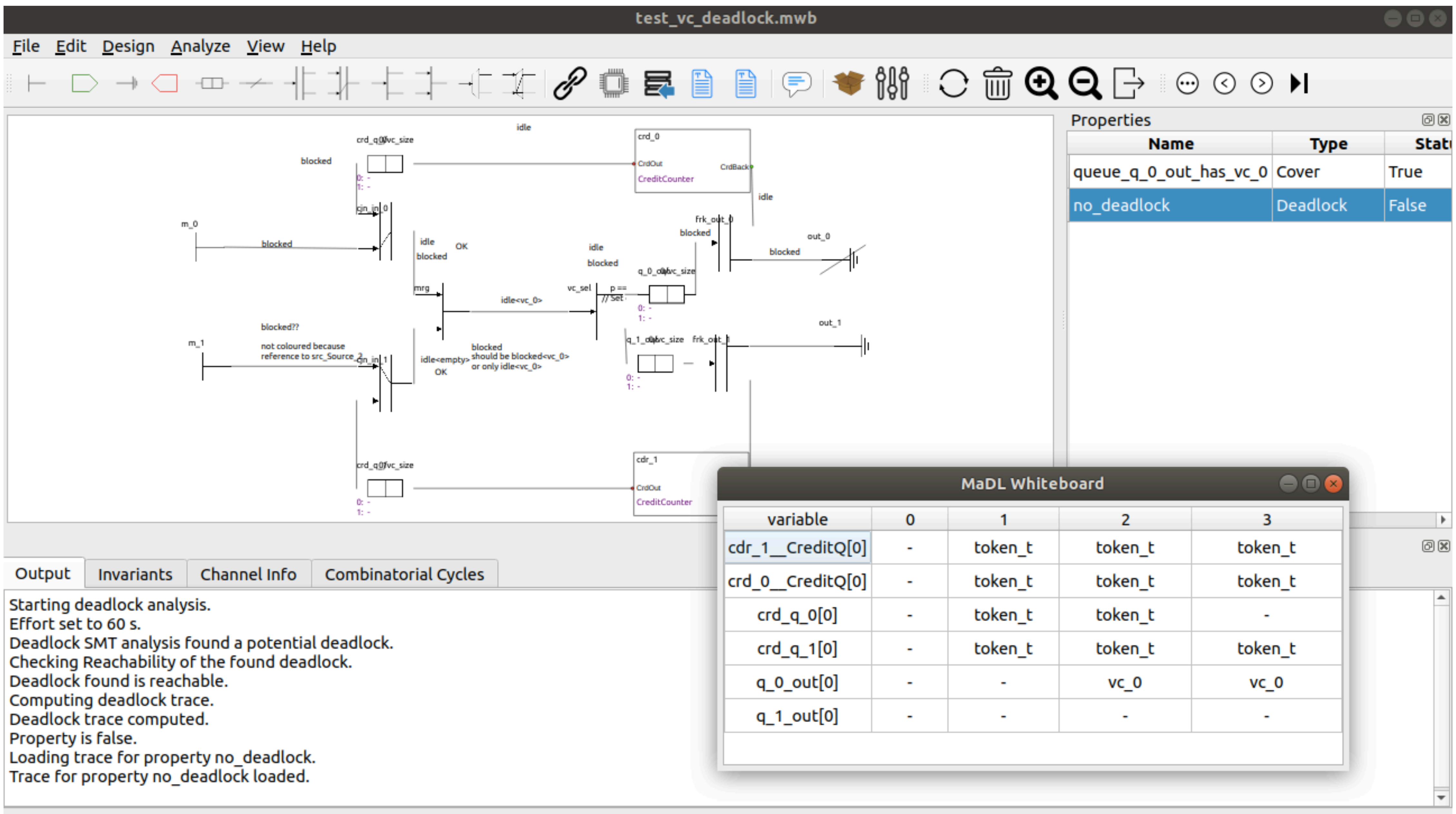
Notion of a “dead” channel: F (c.irdy & G ! c.trdy)

MaDL Design and Verification Environment

textual input

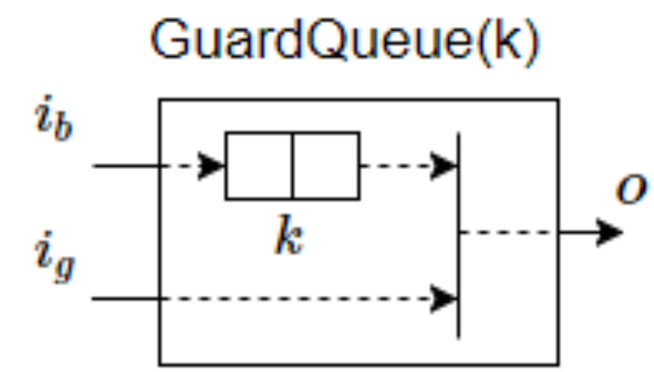
```
param int QSIZE;
const pkt;
chan src := Source(pkt);
chan one, two := Fork(src);
chan long := Queue(QSIZE,Queue(QSIZE, one));
chan short := Queue(QSIZE, two);
Sink(CtrlJoin(long,short));
```

https://github.com/MaDL-DVT/madl-dvt



MaDL Ring Extension - GuardQueue primitive

Modelling the injection constraint with the GuardQueue:
'Only inject if the local buffer is empty'



$i_b.trdy = \neg Full(gq) \wedge sel == 0$

$i_g.trdy = o.trdy \wedge Empty(gq) \wedge sel == 1$

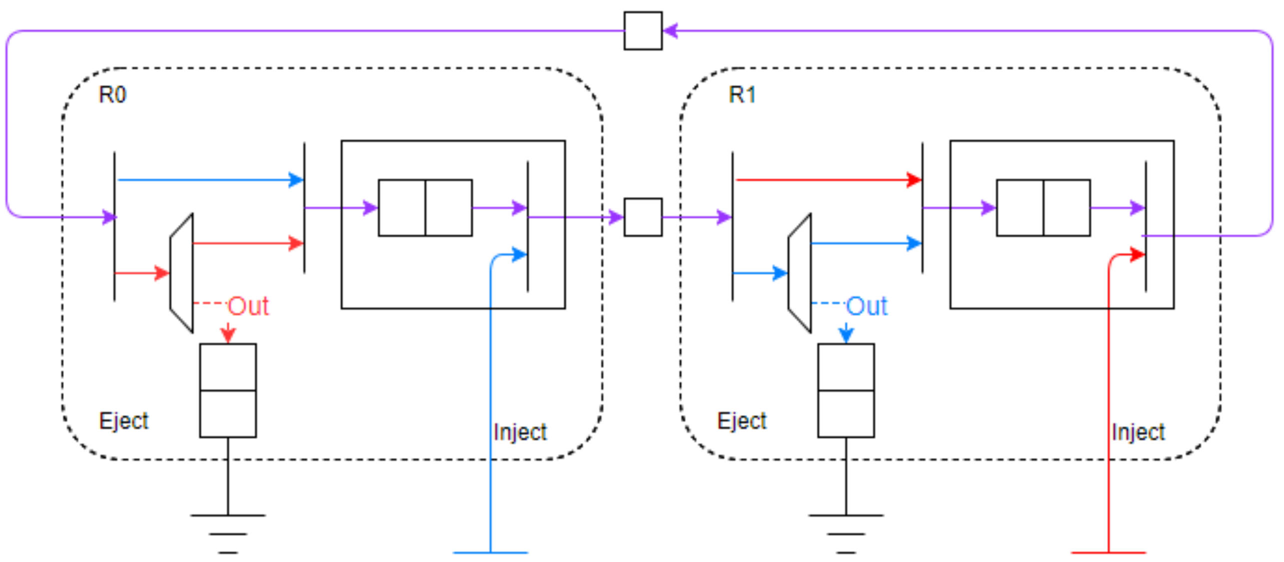
$o.irdy = \big(sel == 0 \wedge \neg Empty(gq)\big) \vee (Empty(gq) \wedge i_g.irdy \wedge sel == 1)$

$$\begin{aligned} block(i_b, c) &= Full(gq) \wedge block(o, gq.head) \\ block\big(i_g, c\big) &= block(o, c) \vee \big(sel == 0 \wedge Full(gq) \wedge block(o, gq.head)\big) \\ &\quad \vee never\ empty(gq) \\ idle(o, c) &= \big(idle(i_b, c) \wedge idle\big(i_g, c\big)\big) \vee \big(idle(i_b, c) \wedge never\ empty(gq)\big) \\ &\quad \vee\ \exists_{c' \in C \backslash c}\, block(o, c') \end{aligned}$$

MaDL Ring Extension - Ring invariants

Ring R = *(D,X,I,O)*

- D: data colours
- X: Channels
- I: Inputs
- O: Outputs



Ring invariant for ring R:

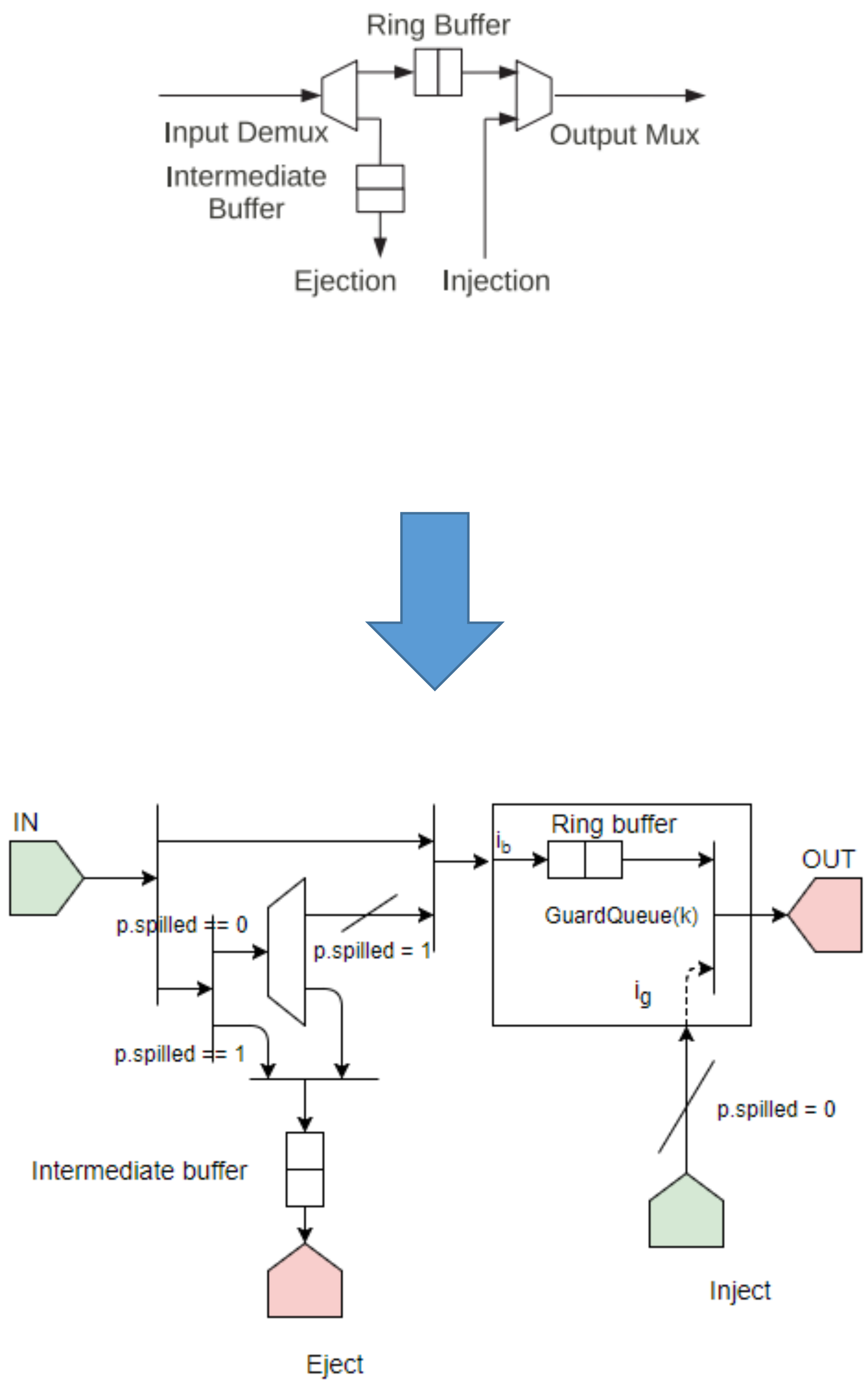
occ(R) <= cap(R) - x

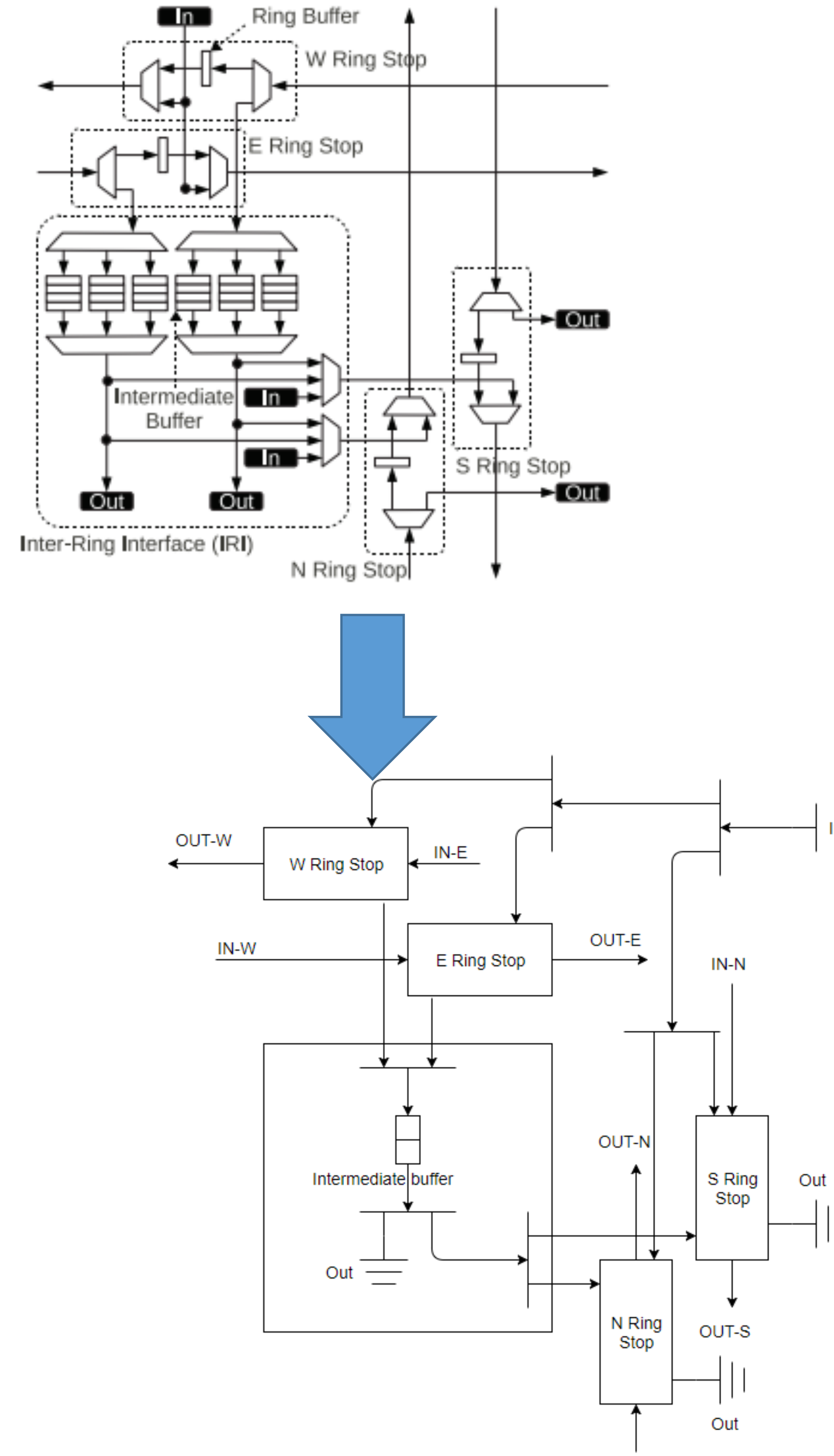
where x is the smallest capacity over all guard queues in the ring.

({red}, {*∪*}, {R1_Inject}, {R0_Out})
({blue}, {*∪*}, {R0_Inject}, {R1_Out})

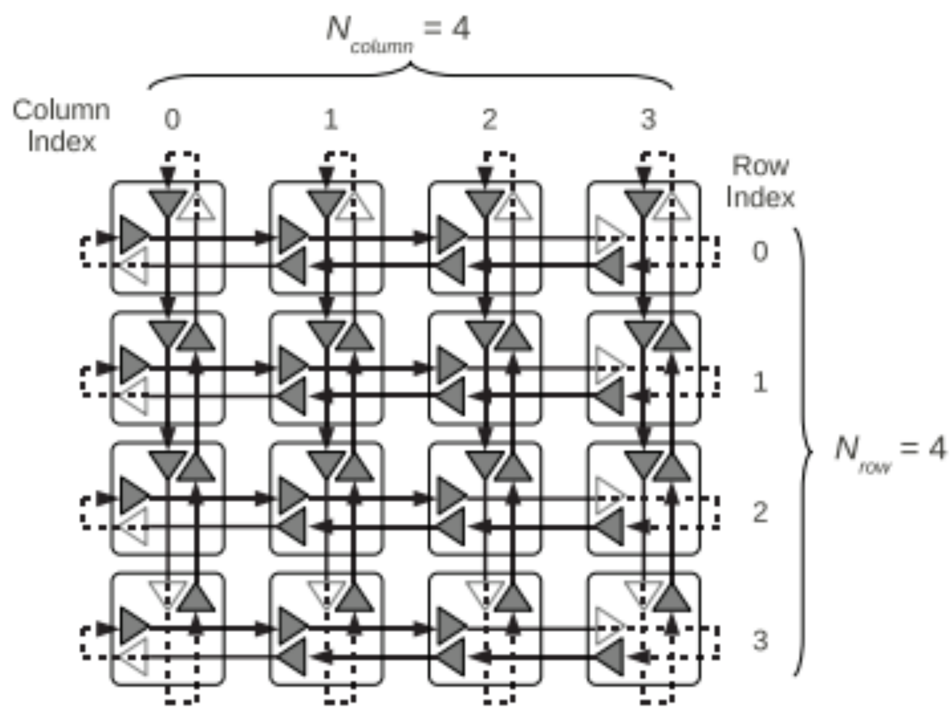
({red,blue}, {*∪ *∪*}, {R0_Inject, R1_Inject}, {R0_Out,R1_Out})

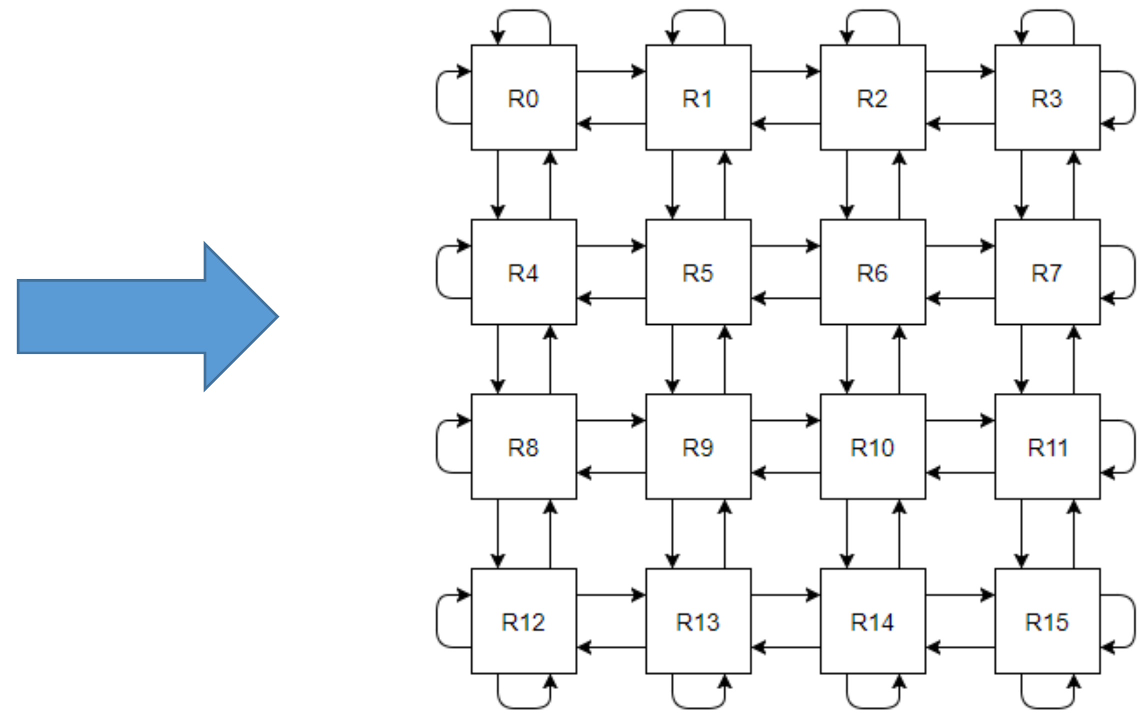
MaDL model of the TornadoNoC blocks



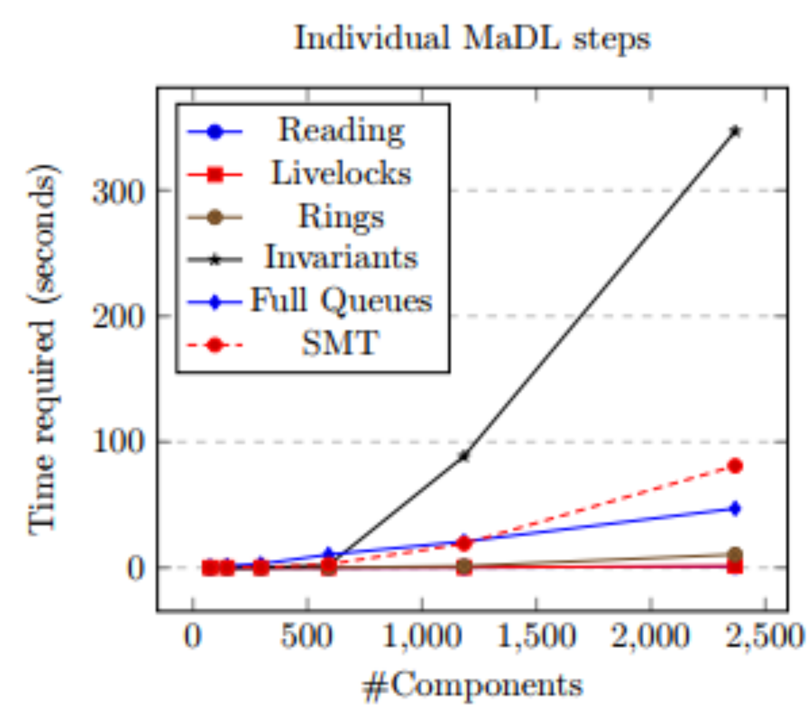


MaDL model and analysis results





Layout	#Components	Rings	Deadlock	Verification time
2×1	98	1	No	1.4s
2×2	196	4	No	2.8s
4×2	392	6	No	8.0s
4×4	784	8	No	59.7s
4×8	1568	12	No	970s
8×8	3136	16	No	3691s



MaDL model and analysis results - Discussion

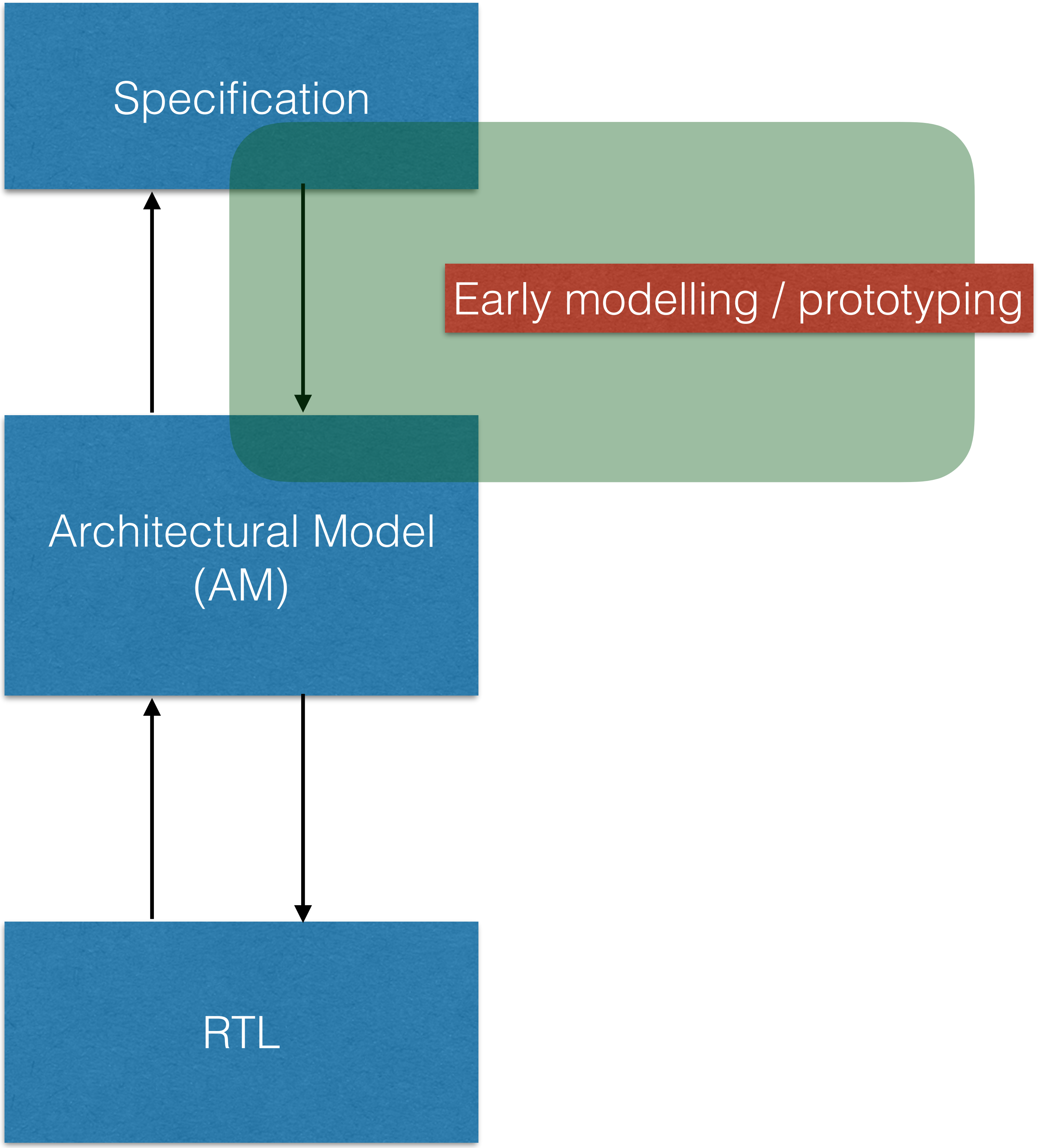
Conclusion:

- Ring networks can now be analysed without finding the trivial all-queues-are-full deadlock.
- TornadoNoC contains a potential deadlock when using credit-flow control
- The MaDL GuardQueue avoids this deadlock.

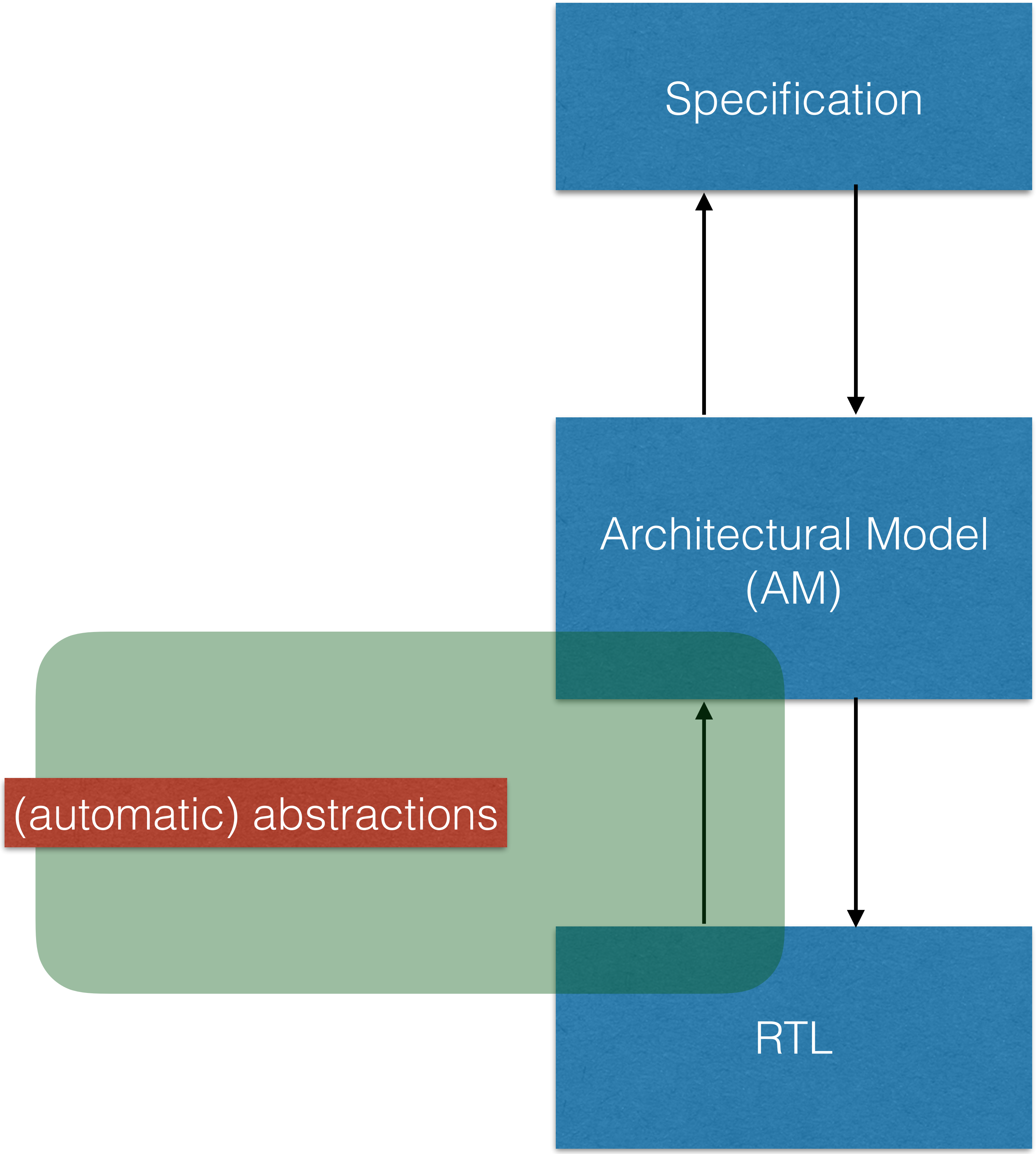
Future Work

- More detailed model of TornadoNOC, including Inflation Mechanism
- Proof of other properties like message ordering, starvation and livelock freedom, etc.

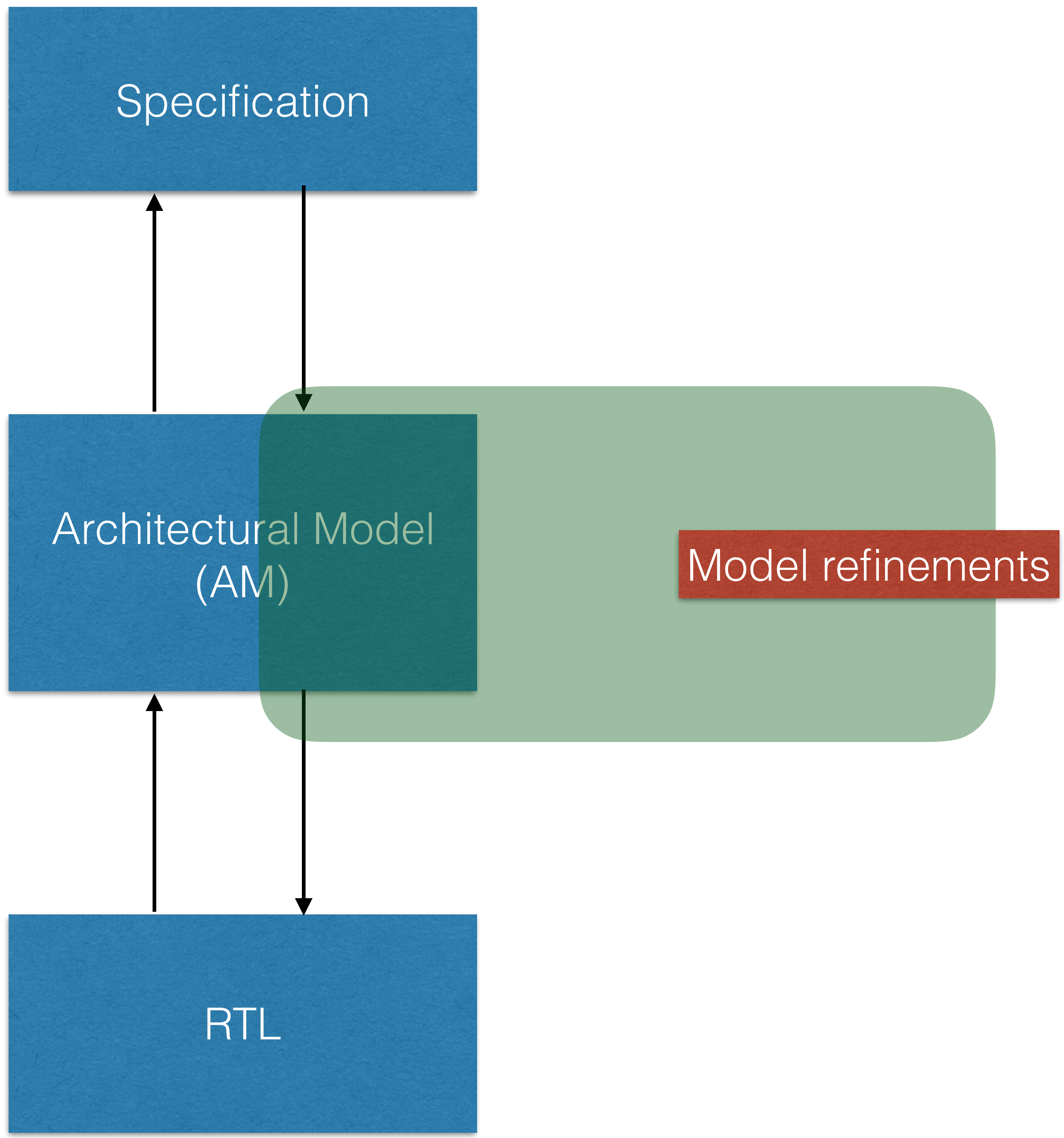
Architectural modelling and analysis - Reflection



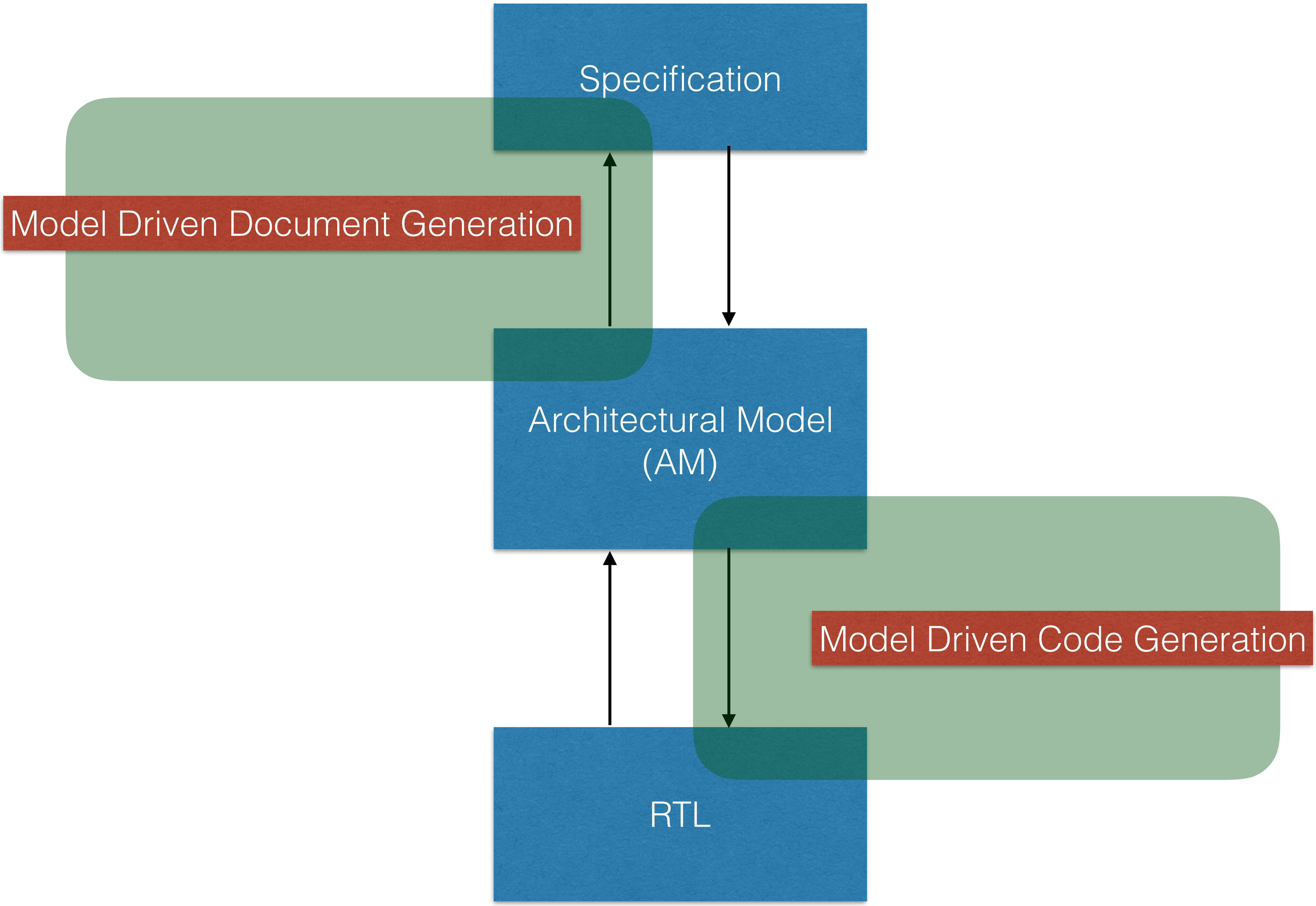
Architectural modelling and analysis - Reflection



Architectural modelling and analysis - Reflection



Architectural modelling and analysis - Reflection



Thanks!