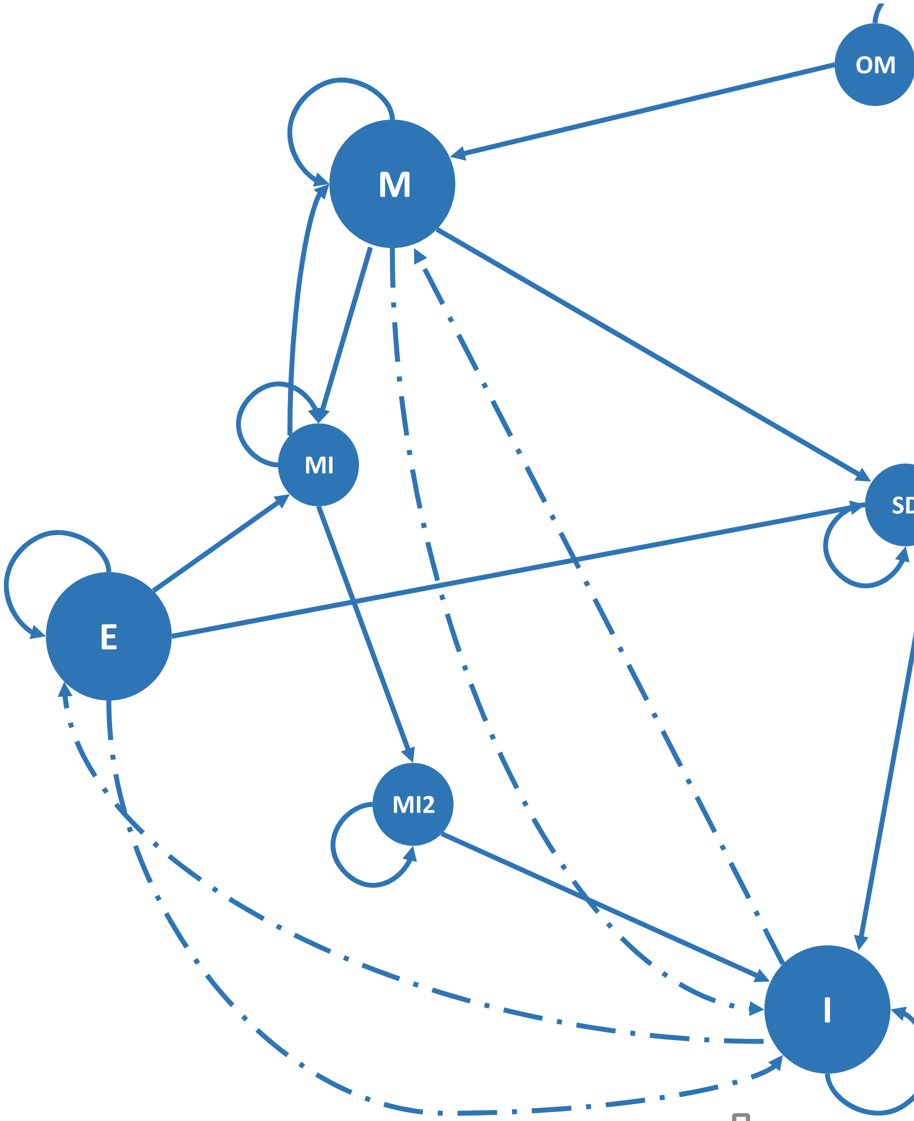
PVCoherence

Zhang, Bringham, Erickson, & Sorin

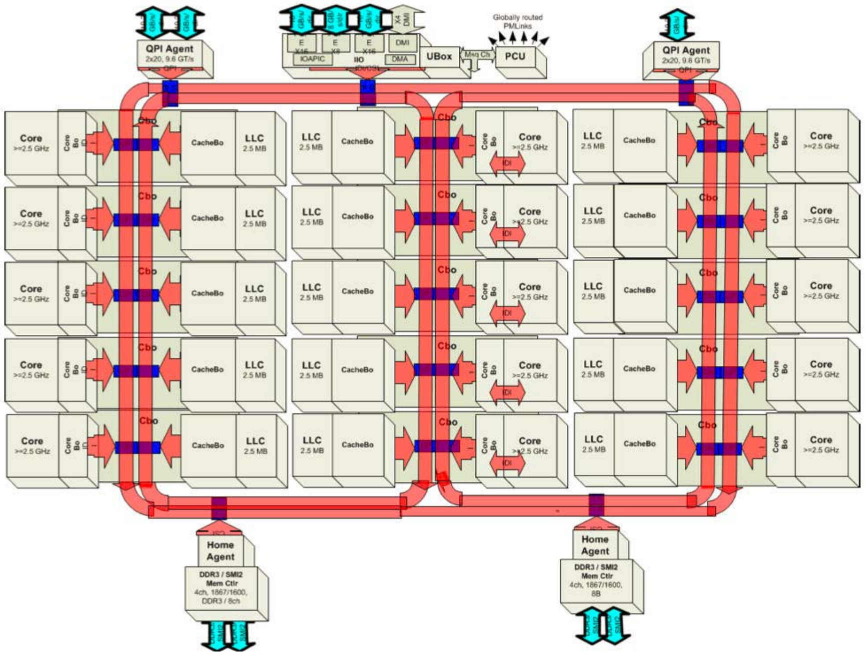
Amlan Nayak & Jay Zhang

Overview

- Motivation
- Background
- Parametric Verification
- Design Guidelines
- PV-MOESI vs OP-MOESI
- Results
- Conclusion



Issue with **Coherence Protocols**



Difficult to automatically verify for many core systems

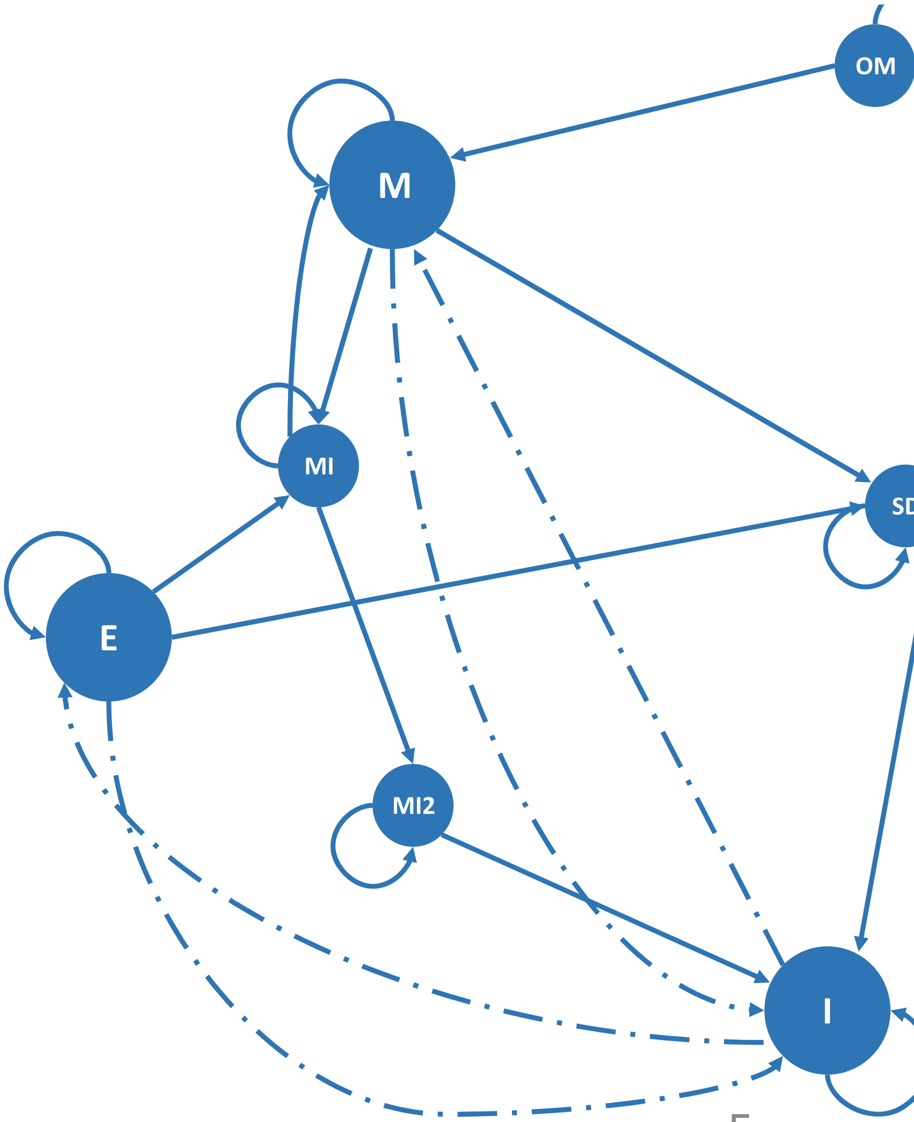
Better performance
⇒ Complex protocols →<span style="color:red">**Difficult to formally verify**</span>

GOAL

Architect arbitrarily large flat protocols such that they can be verified using a mostly-automated methodology

Overview

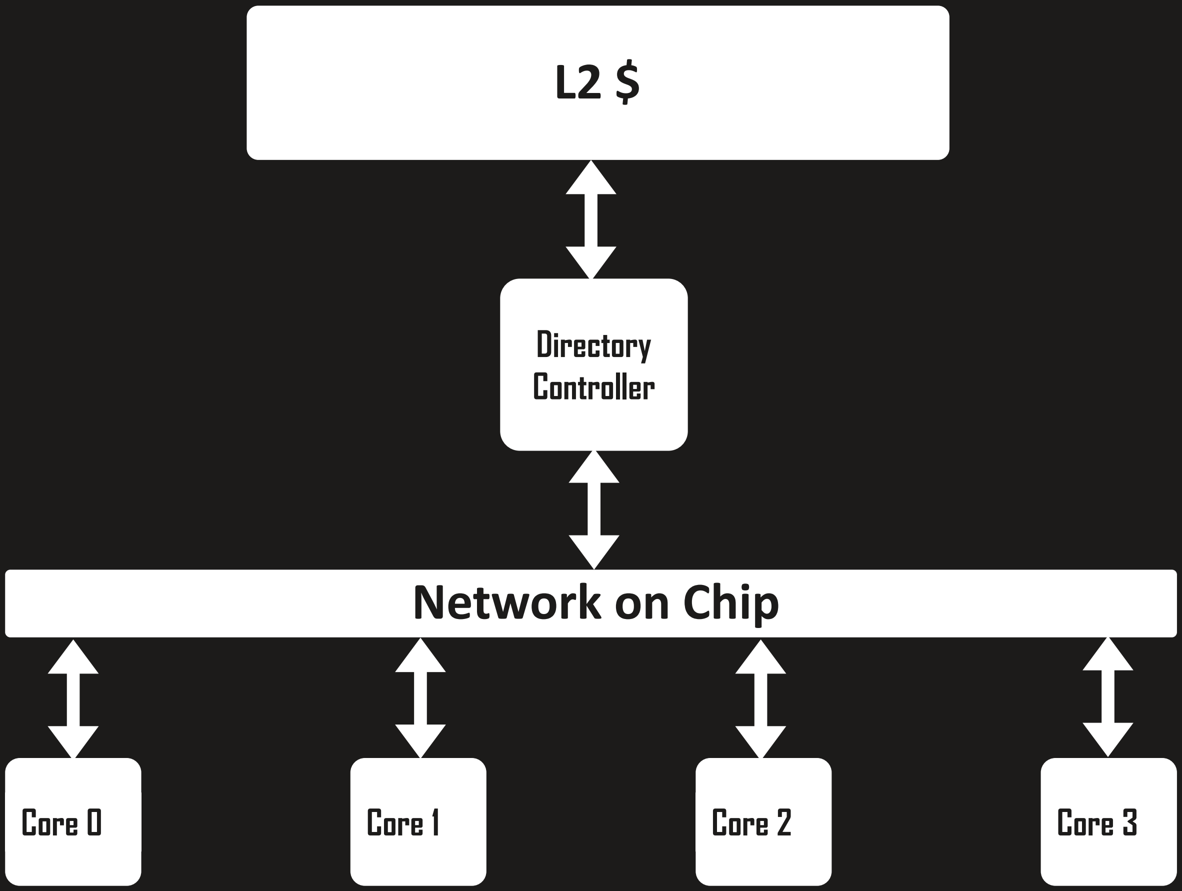
- Motivation
- **Background**
- Parametric Verification
- Design Guidelines
- PV-MOESI vs OP-MOESI
- Results
- Conclusion



Coherence Protocols

Two Primary Categories: **Snooping** & **Directory-based**

(ex. MSI, MESI, MOESI, MESIF)



L2 $

Directory Controller

Network on Chip

Core 0

Core 1

Core 2

Core 3

State Space Exploration

Formally verifying a coherence protocol

Theorem Proving

State Space Exploration

(with Murphi)

GOAL

Architect arbitrarily large flat protocols such that they can be verified using a mostly-automated methodology

GOAL

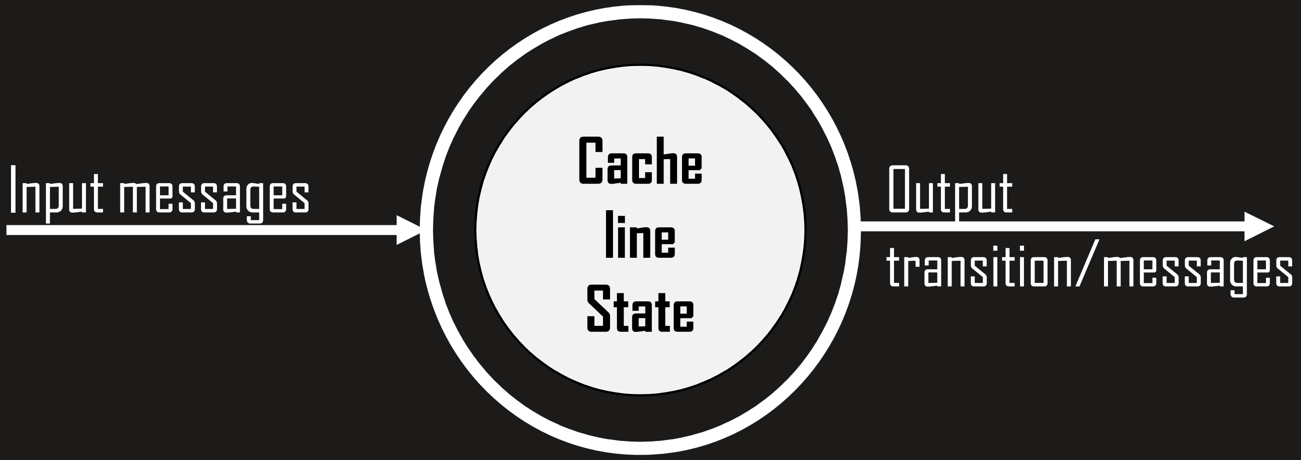
Architect arbitrarily large flat protocols such that they can be verified using a mostly-automated methodology

Model Protocol in Murphi
Check invariants

Model Protocol in Murphi

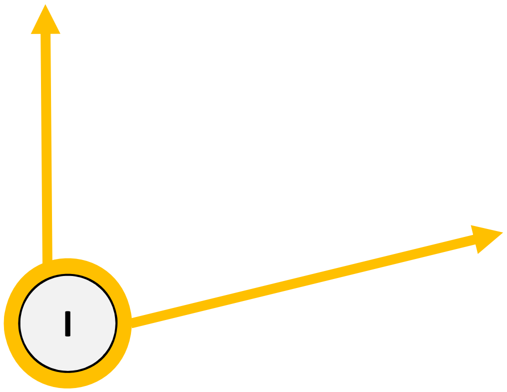
Check invariants

Murphi Processor Node State Definition

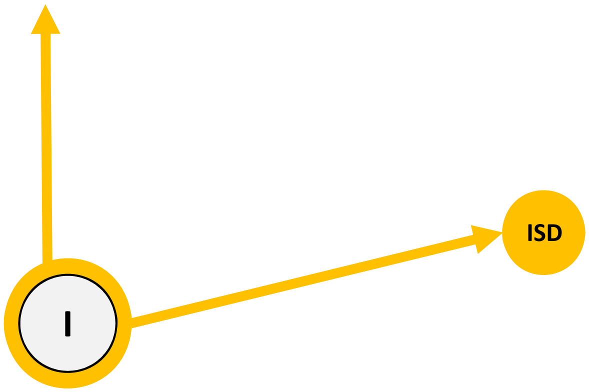


MOESI protocol
Processor
Cache Controller

MOESI protocol
Processor
Cache Controller



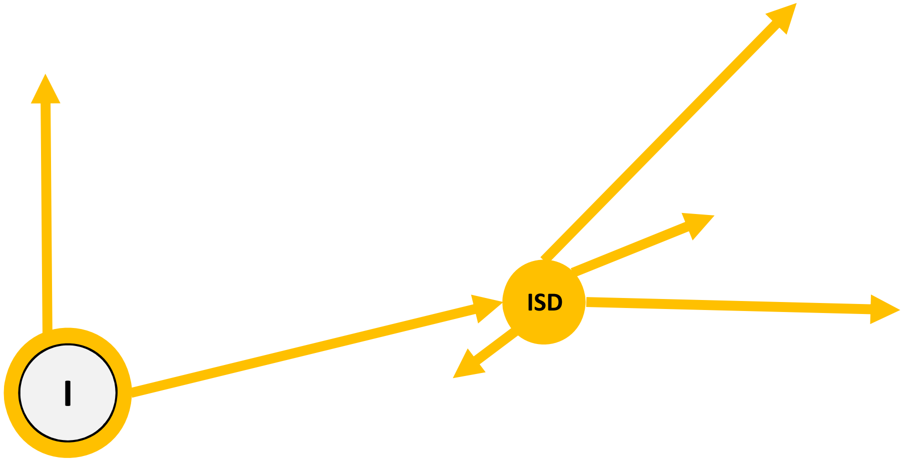
MOESI protocol
Processor
Cache Controller



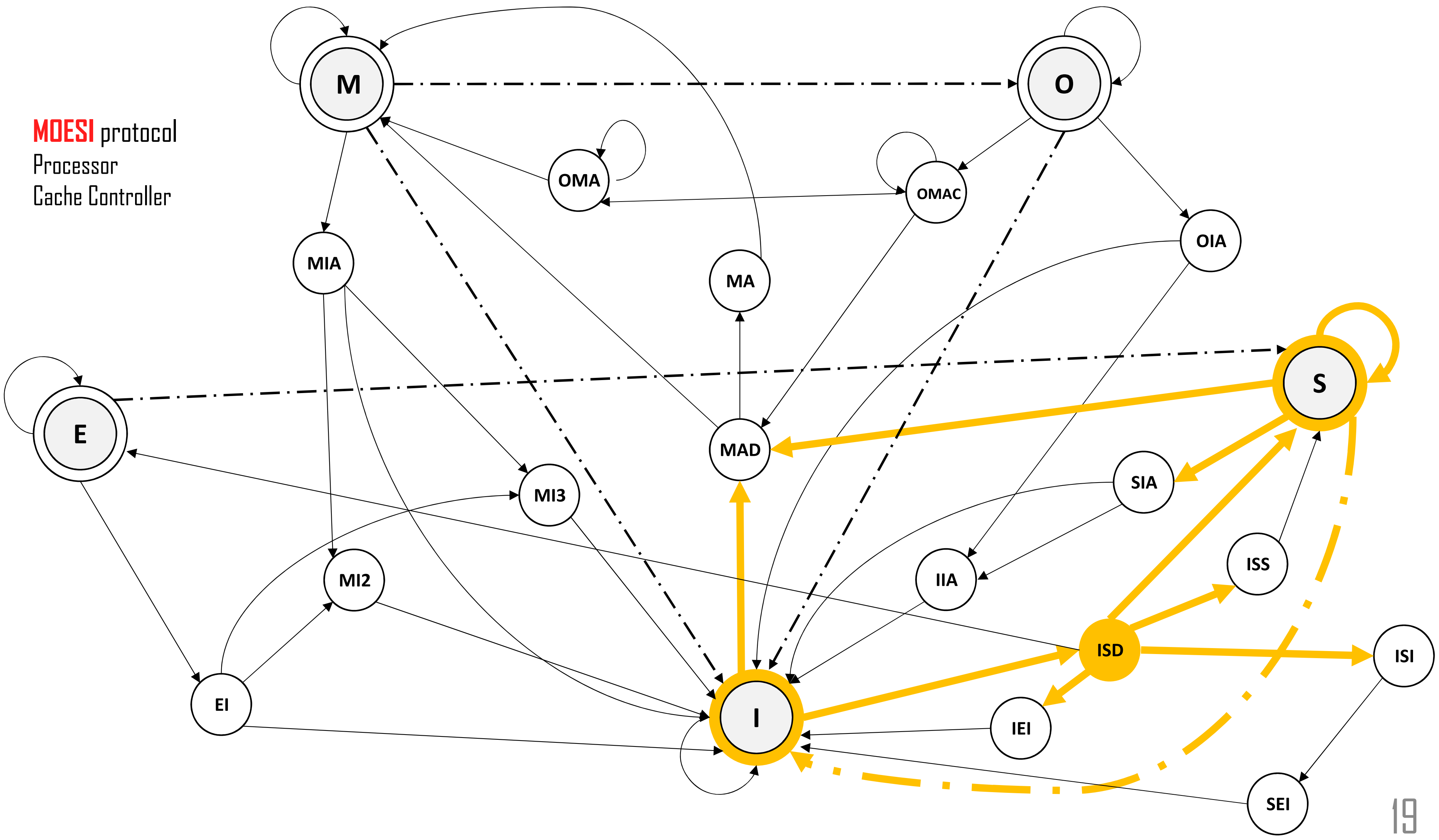
MOESI protocol

Processor

Cache Controller



MOESI protocol
Processor
Cache Controller



Model Protocol in Murphi

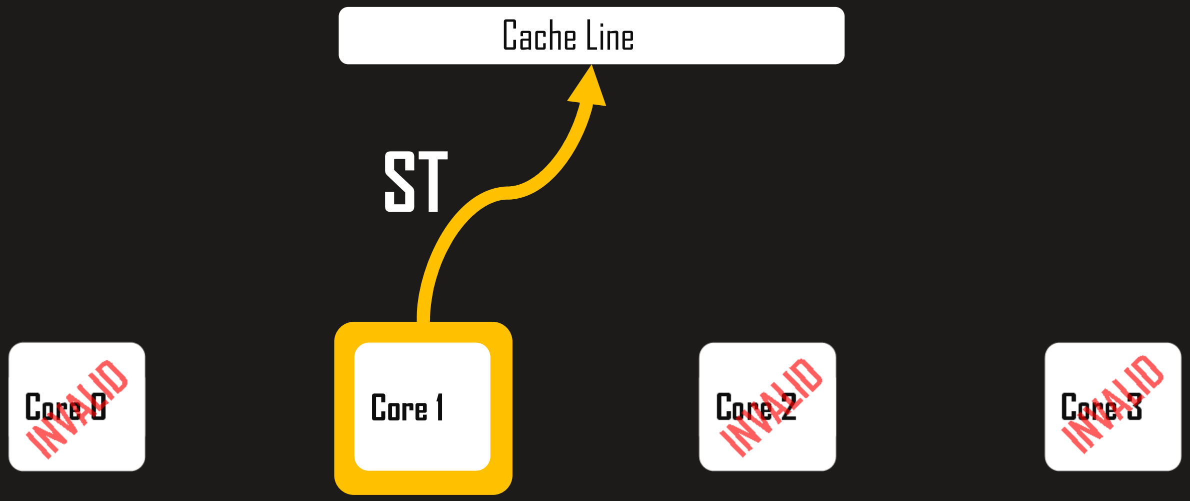
Check invariants

Check invariants

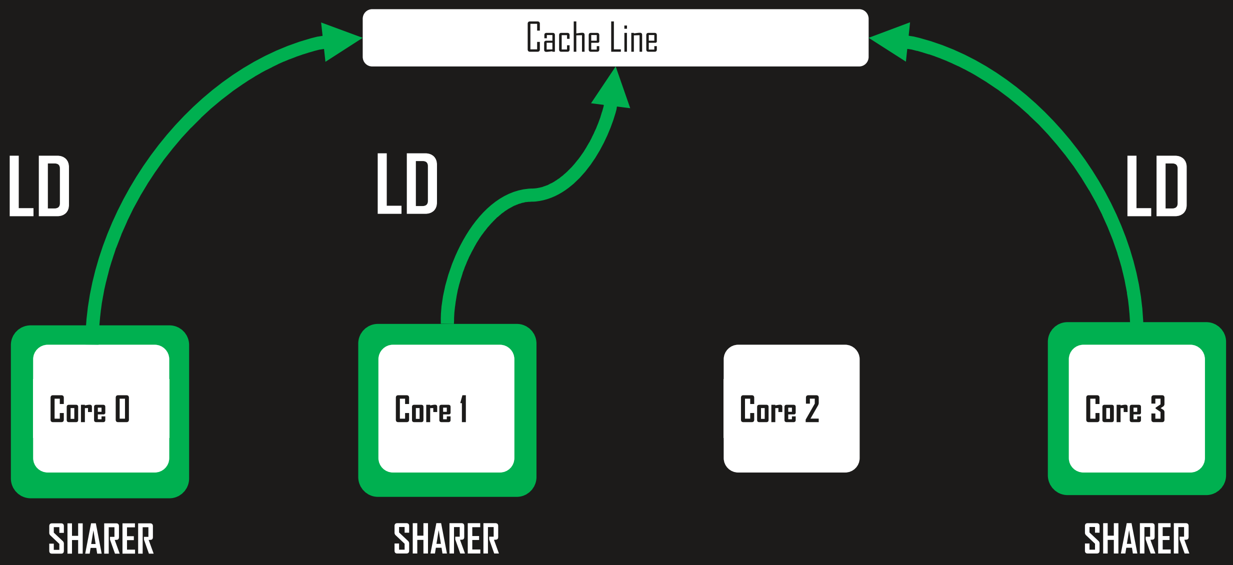
1. **Permission Invariant:** Single-Writer, Multiple-Reader
2. **Data Invariant:** Read returns value of last write

Check invariants

1. **Permission Invariant:** Single-Writer, Multiple-Reader
2. **Data Invariant:** Read returns value of last write



SINGLE Writer



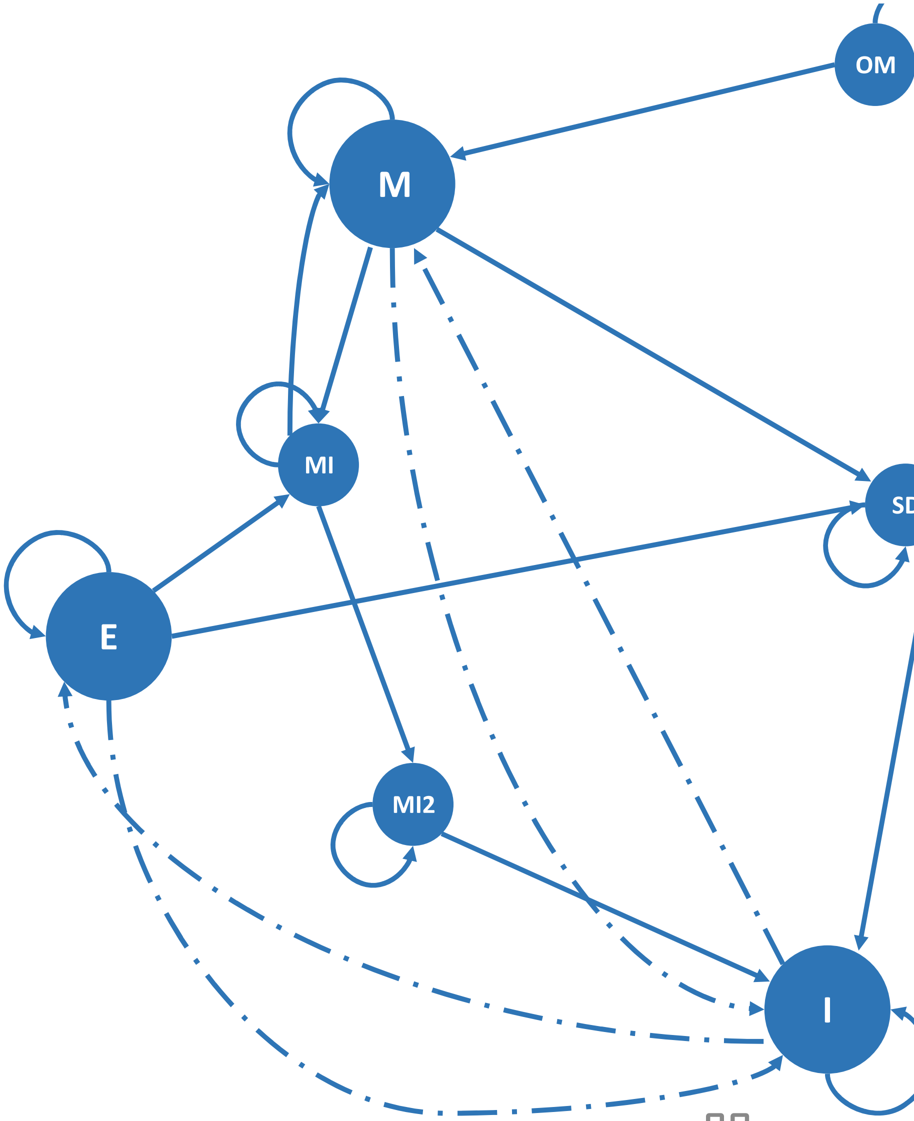
Multiple Reader

Check invariants

1. Permission Invariant: Single-Writer, Multiple-Reader
2. **Data Invariant:** Read returns value of last write

Overview

- Motivation
- Background
- **Parametric Verification**
- Design Guidelines
- PV-MOESI vs OP-MOESI
- Results
- Conclusion



PARAMETRIC VERIFICATION (PV)

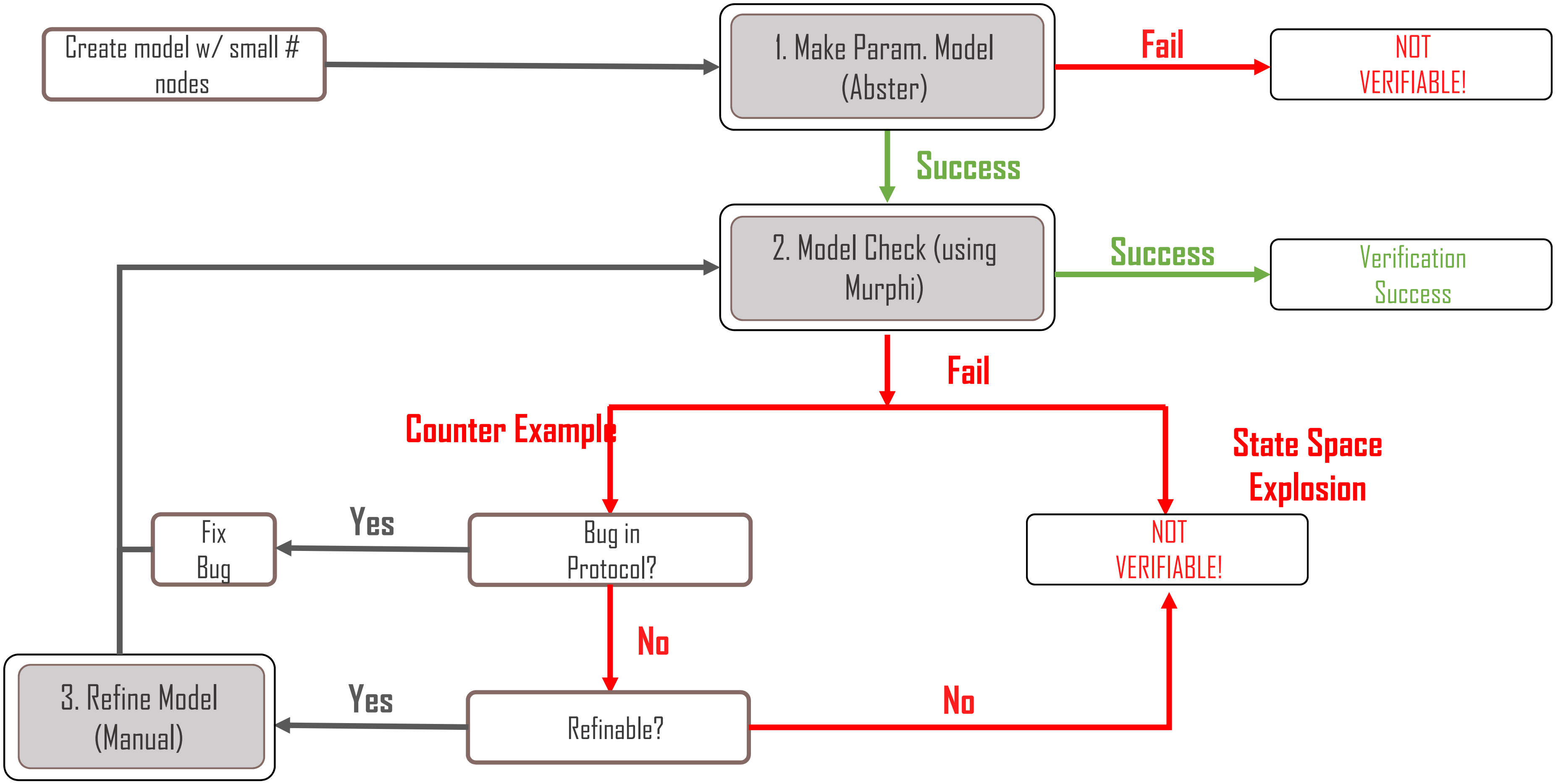
Treat number of ***nodes*** as a **parameter** **(using Abster tool)**

Prove properties of design agnostic to **parameter size**

This process scales to many nodes and is almost fully automatic

Simple-PV Process Flow

How to design a readily verifiable Coherence Protocol



Create model w/ small # nodes

1. Make Param. Model (Abster)

Fail

NOT VERIFIABLE!

Success

2. Model Check (using Murphi)

Success

Verification Success

Fail

Counter Example

State Space Explosion

Fix Bug

Yes

Bug in Protocol?

NOT VERIFIABLE!

No

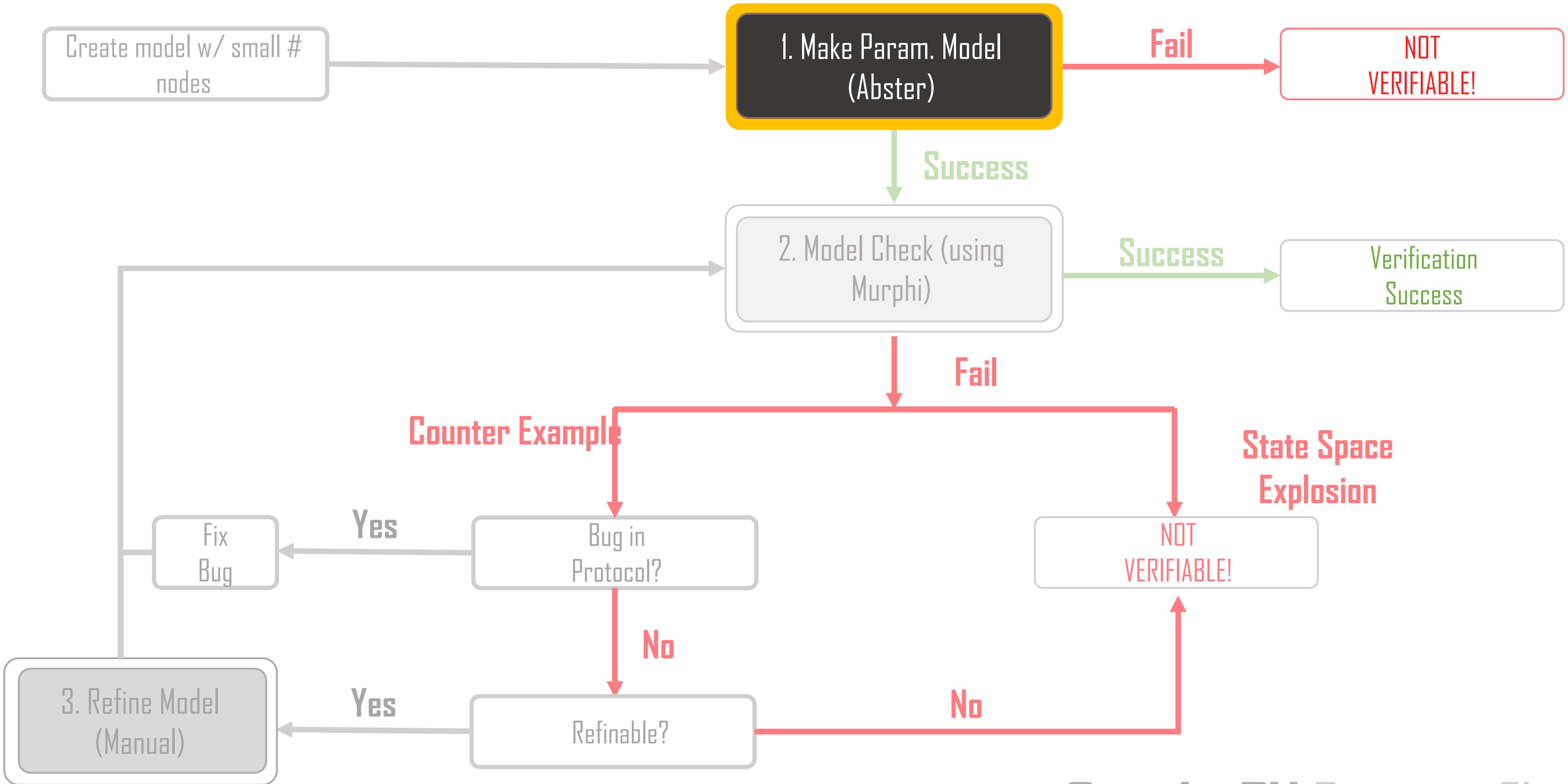
3. Refine Model (Manual)

Yes

Refinable?

No

Simple-PV Process Flow



Create model w/ small # nodes

1. Make Param. Model (Abster)

Fail

NOT VERIFIABLE!

Success

2. Model Check (using Murphi)

Success

Verification Success

Fail

Counter Example

State Space Explosion

Fix Bug

Yes

Bug in Protocol?

NOT VERIFIABLE!

No

3. Refine Model (Manual)

Yes

Refinable?

No

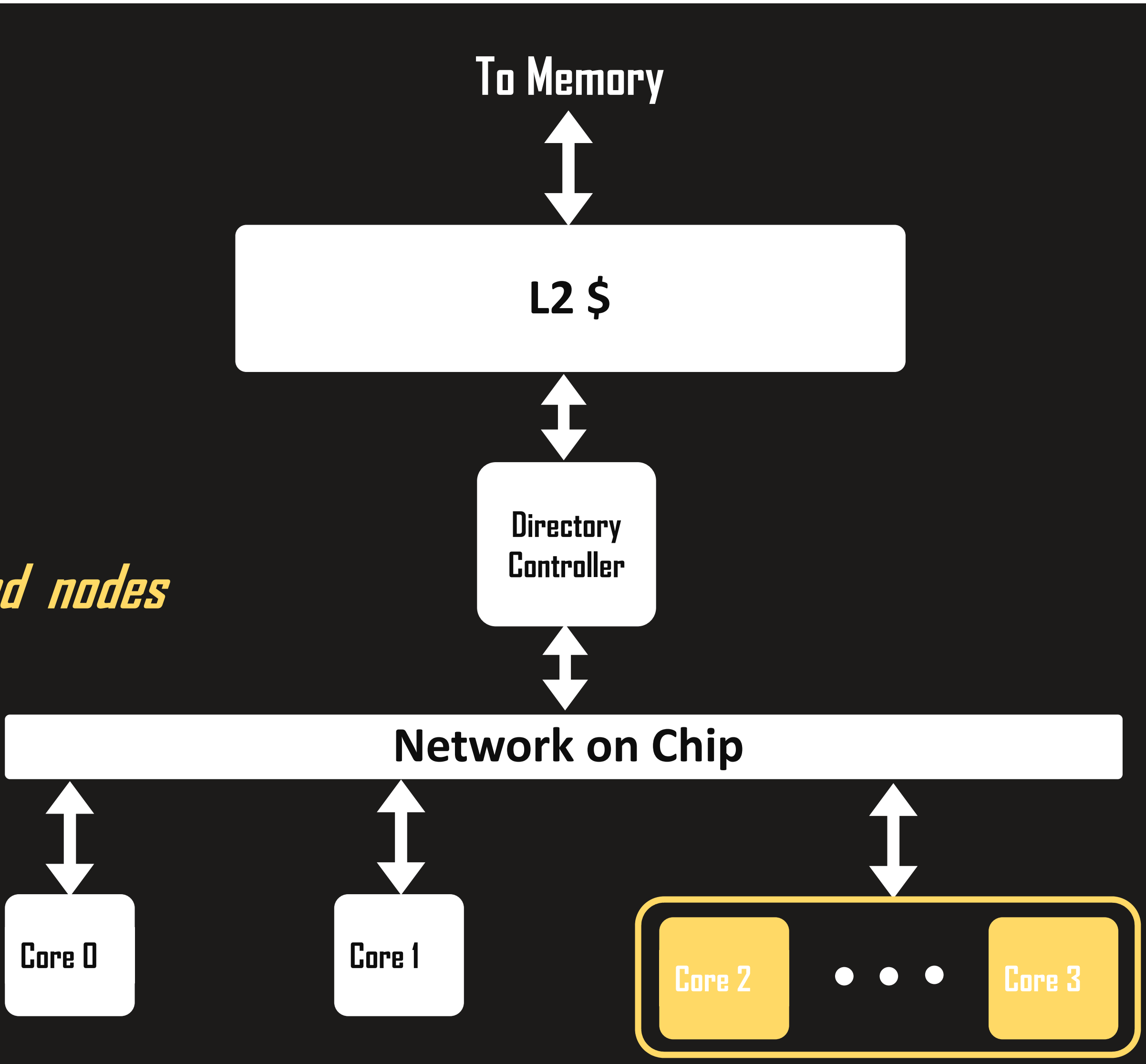
Simple-PV Process Flow

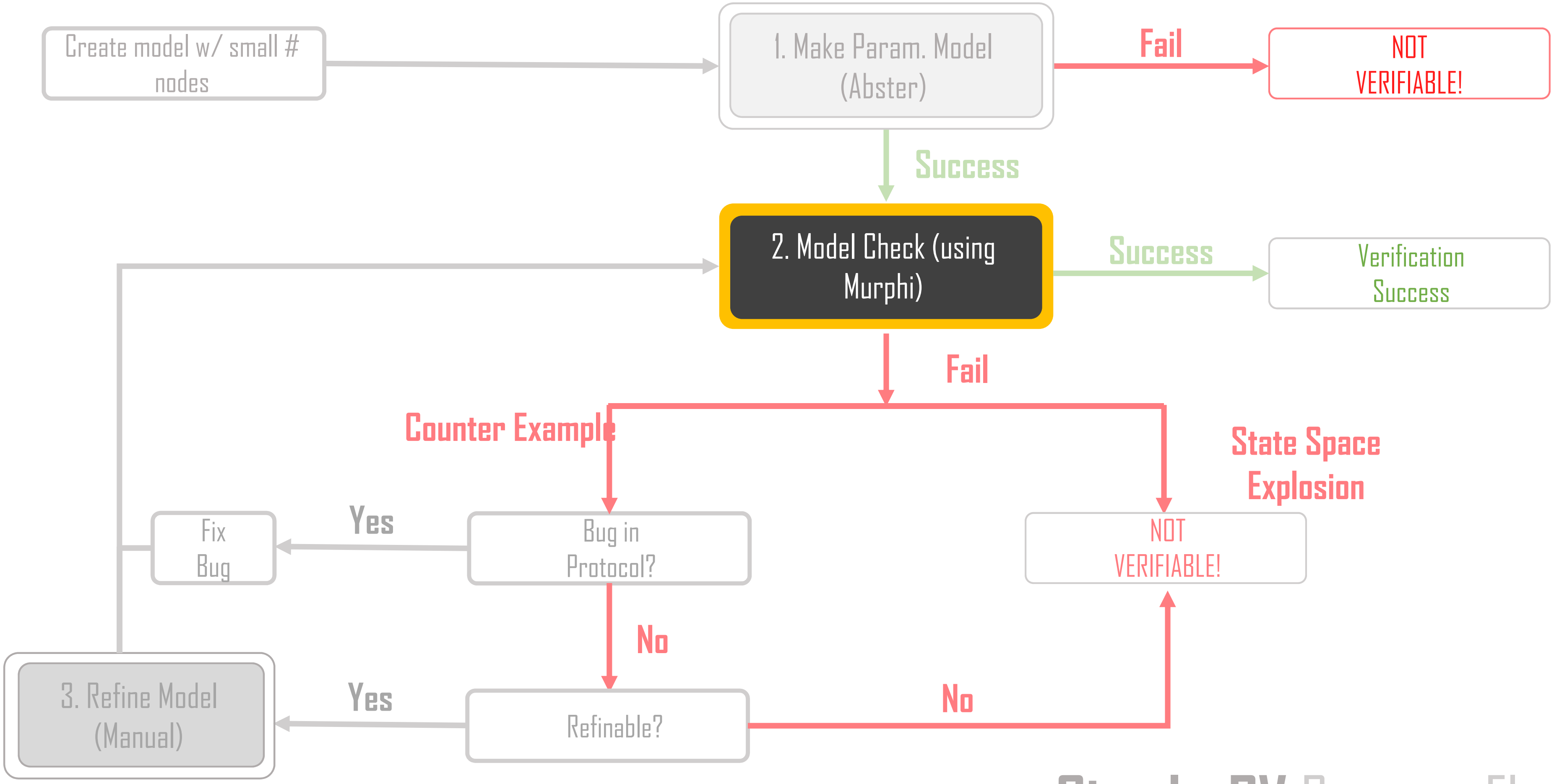
Automatically Create Parametric Model

- Create parametric model from non-parametric model
  - N concrete nodes -> 2 concrete nodes + “Other Node” (N-2)
  - Abster automatic abstraction tool
- Abster over-approximates the behavior of the N-2 nodes
- If Abster fails, modify protocol until it is compatible

N-2

parameterized nodes





Create model w/ small # nodes

1. Make Param. Model (Abster)

Fail

NOT VERIFIABLE!

Success

2. Model Check (using Murphi)

Success

Verification Success

Fail

Counter Example

State Space Explosion

Fix Bug

Yes

Bug in Protocol?

NOT VERIFIABLE!

No

3. Refine Model (Manual)

Yes

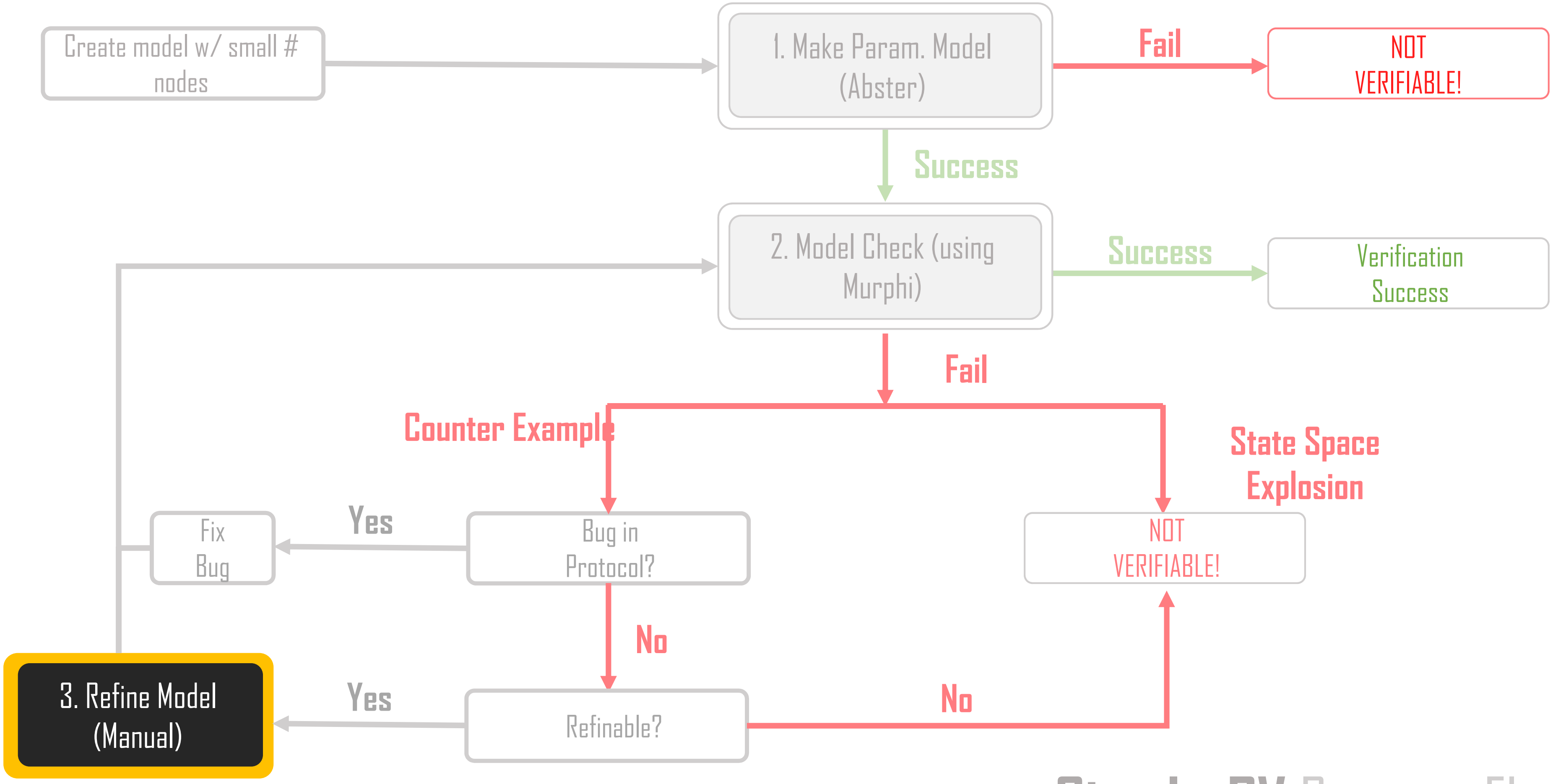
Refinable?

No

Simple-PV Process Flow

Automatically Model Check the model

(MURPHI)



Simple-PV Process Flow

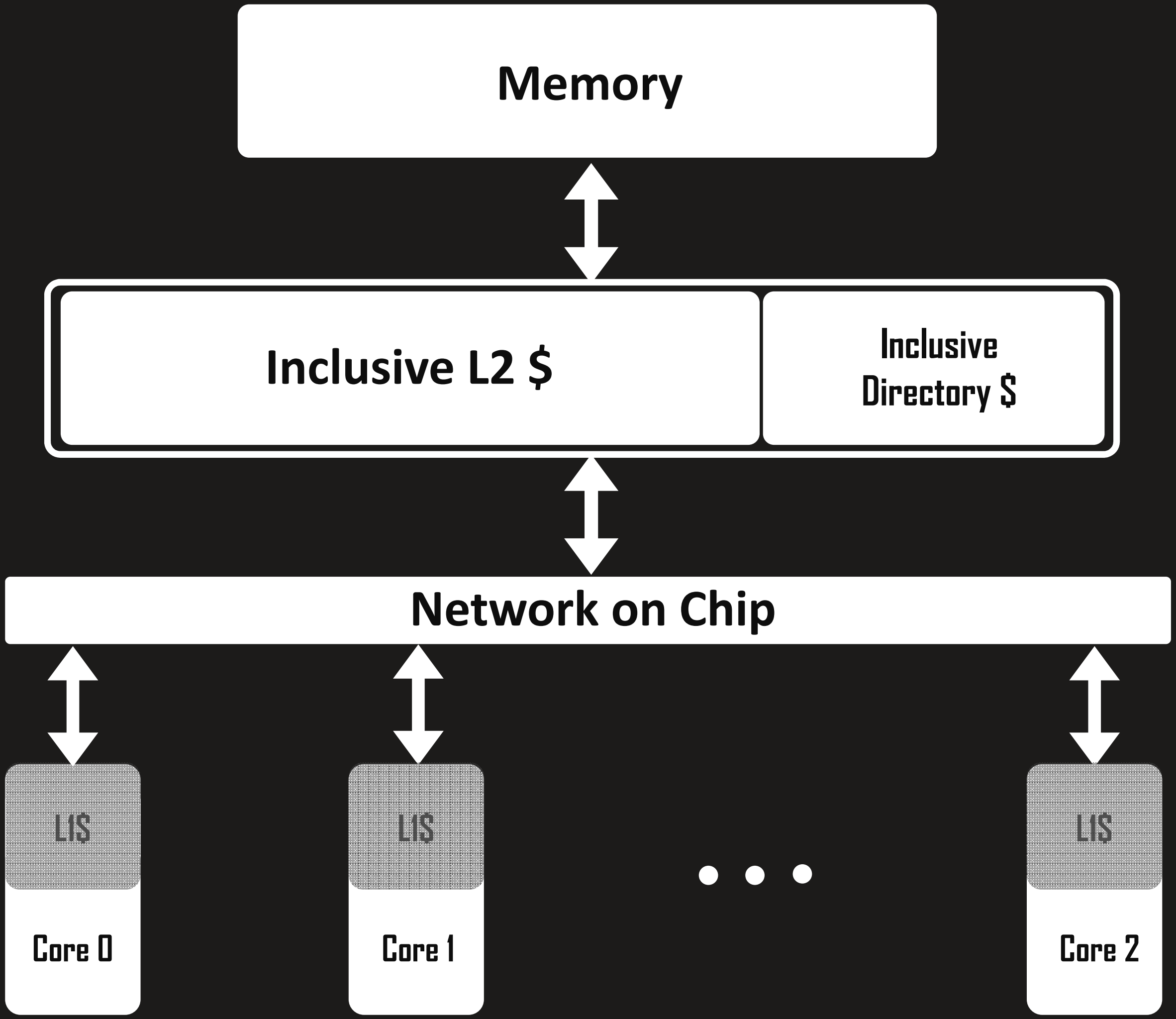
Manually Refine the Model

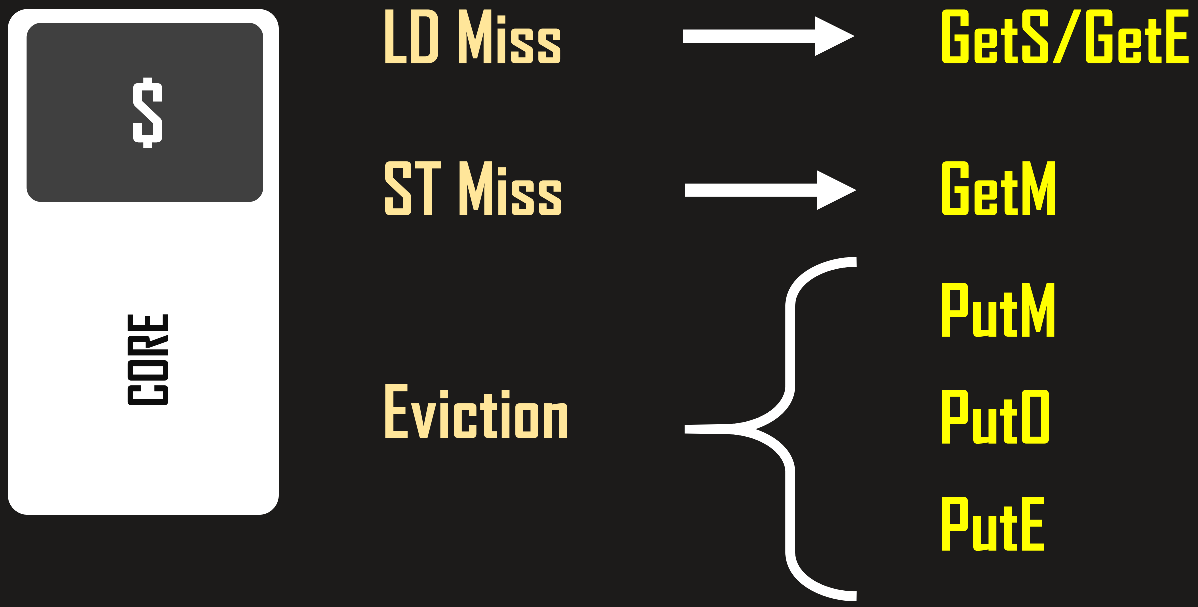
- Over-approximation leads to spurious invariant violations

- Must modify behavior of "Other node"

- **KEY:** Add constraint and check their validity
  - Add invariant **(*lemma*)** – must be true for non-abstracted model
  - Check on the concrete nodes

System Architecture
for PVCoherence

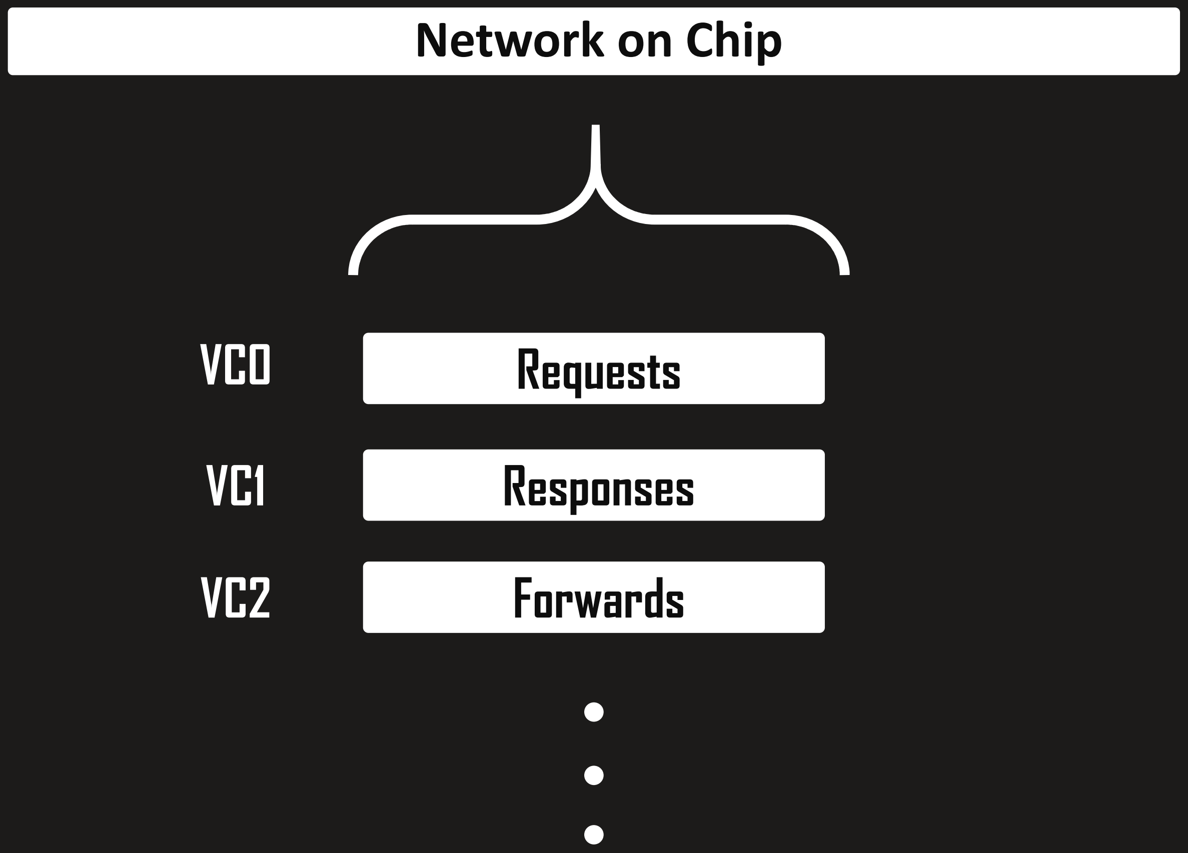

Memory
Inclusive L2 $
Inclusive Directory $
Network on Chip
L1$
L1$
L1$
Core 0
Core 1
• • •
Core 2



LD Miss → GetS/GetE

ST Miss → GetM

Eviction → PutM, PutO, PutE



Network on Chip

VC0 Requests

VC1 Responses

VC2 Forwards

Objective

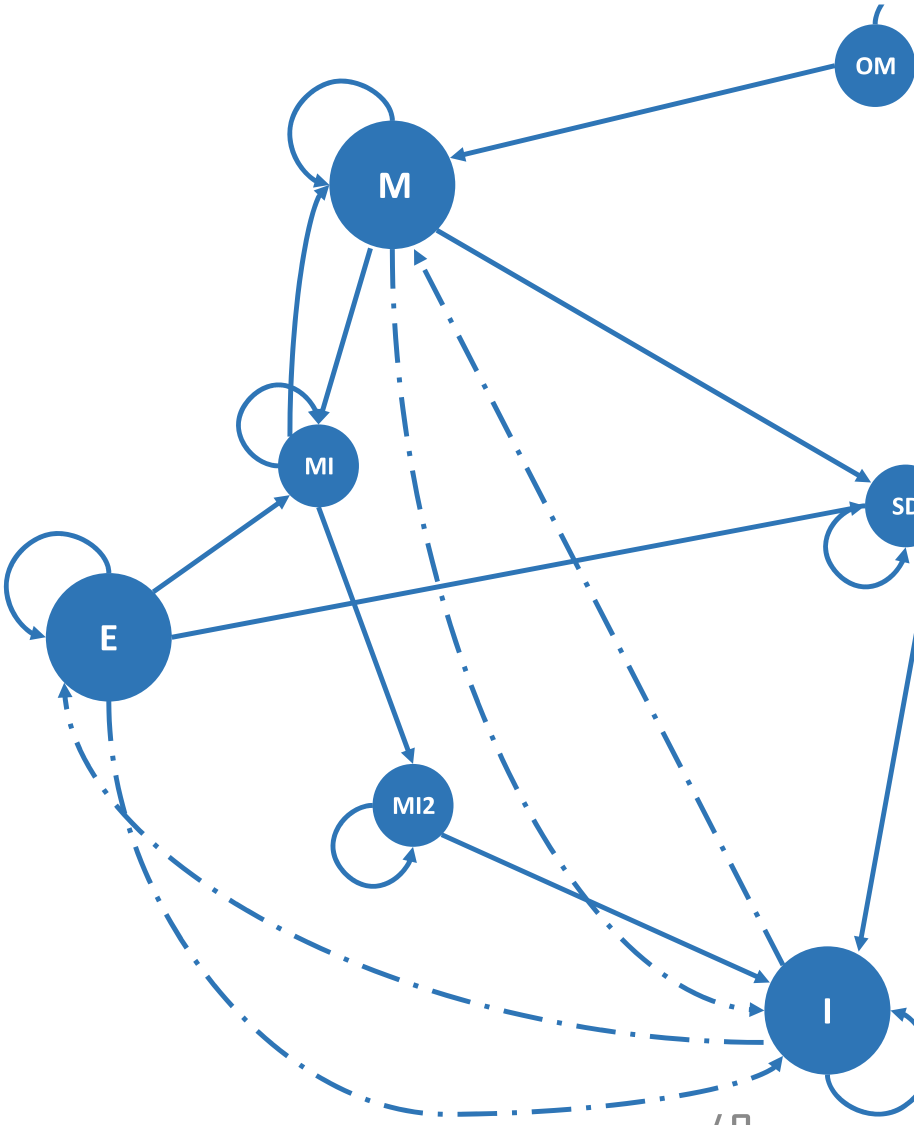
HUMAN EFFORT

AUTOMATION

during **protocol design**

Overview

- Motivation
- Background
- Parametric Verification
- **Design Guidelines**
- PV-MOESI vs OP-MOESI
- Results
- Conclusion



Guidelines

for Simple-PV compliance

Guidelines

for **Simple-PV** compliance

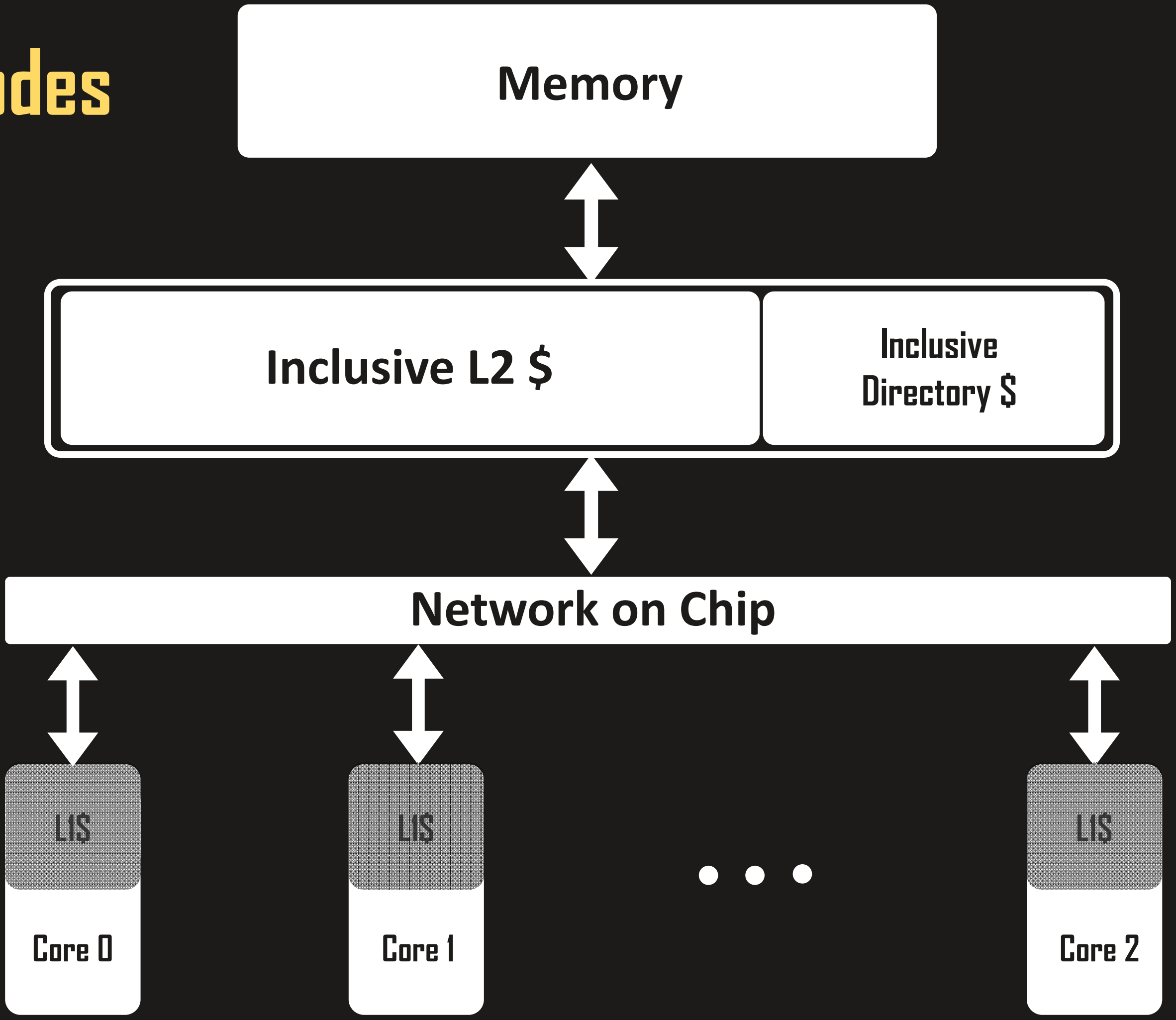
#1: Identical Nodes

#2: No variables must depend on number of Nodes

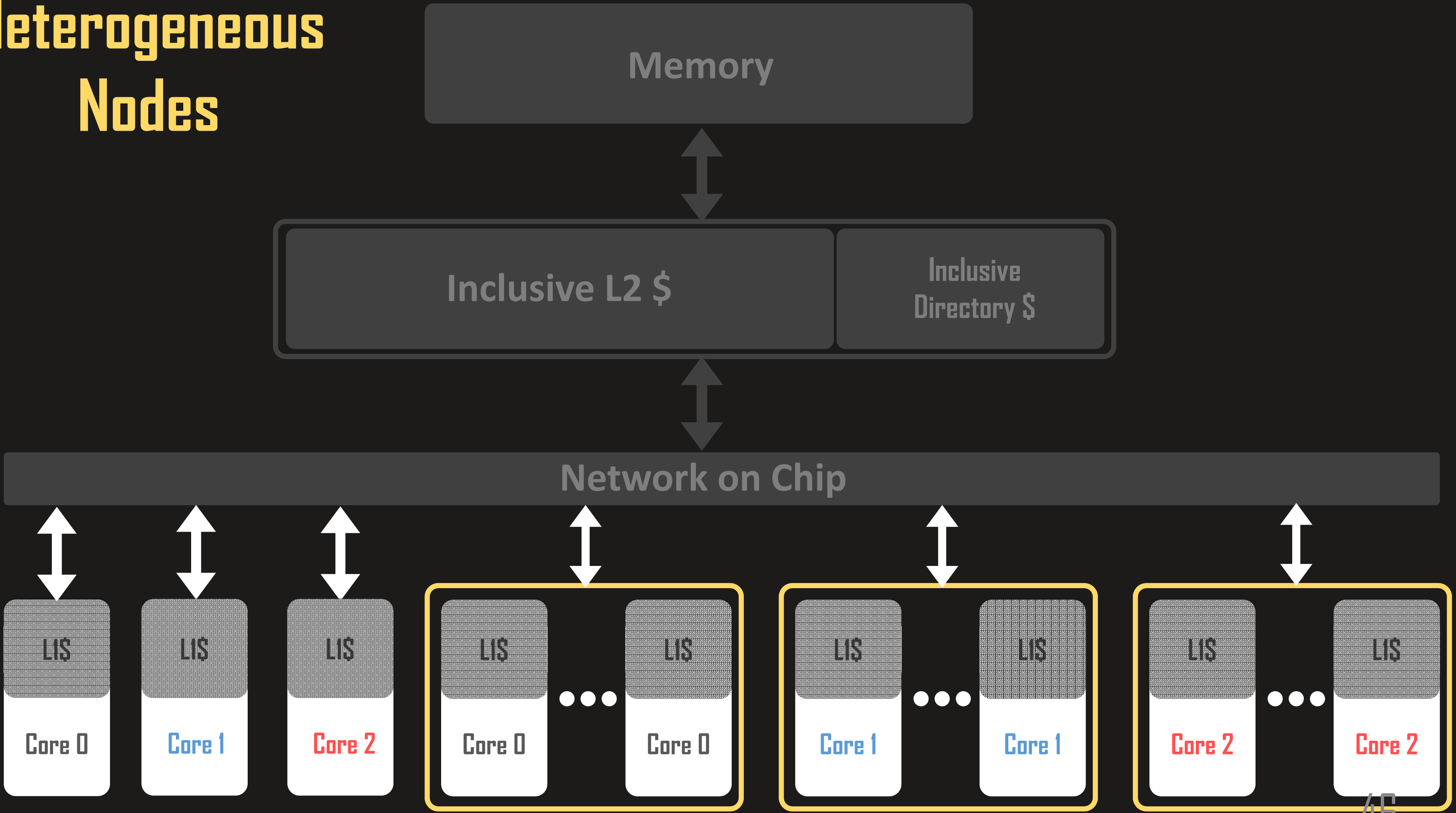
#3: No ordering over list/queue sized by number of nodes

#4: Should not parameterize buffers/queues in more than 1-dim.

Identical Nodes



Heterogeneous Nodes



Guidelines

for Simple-PV compliance

#1: Identical Nodes

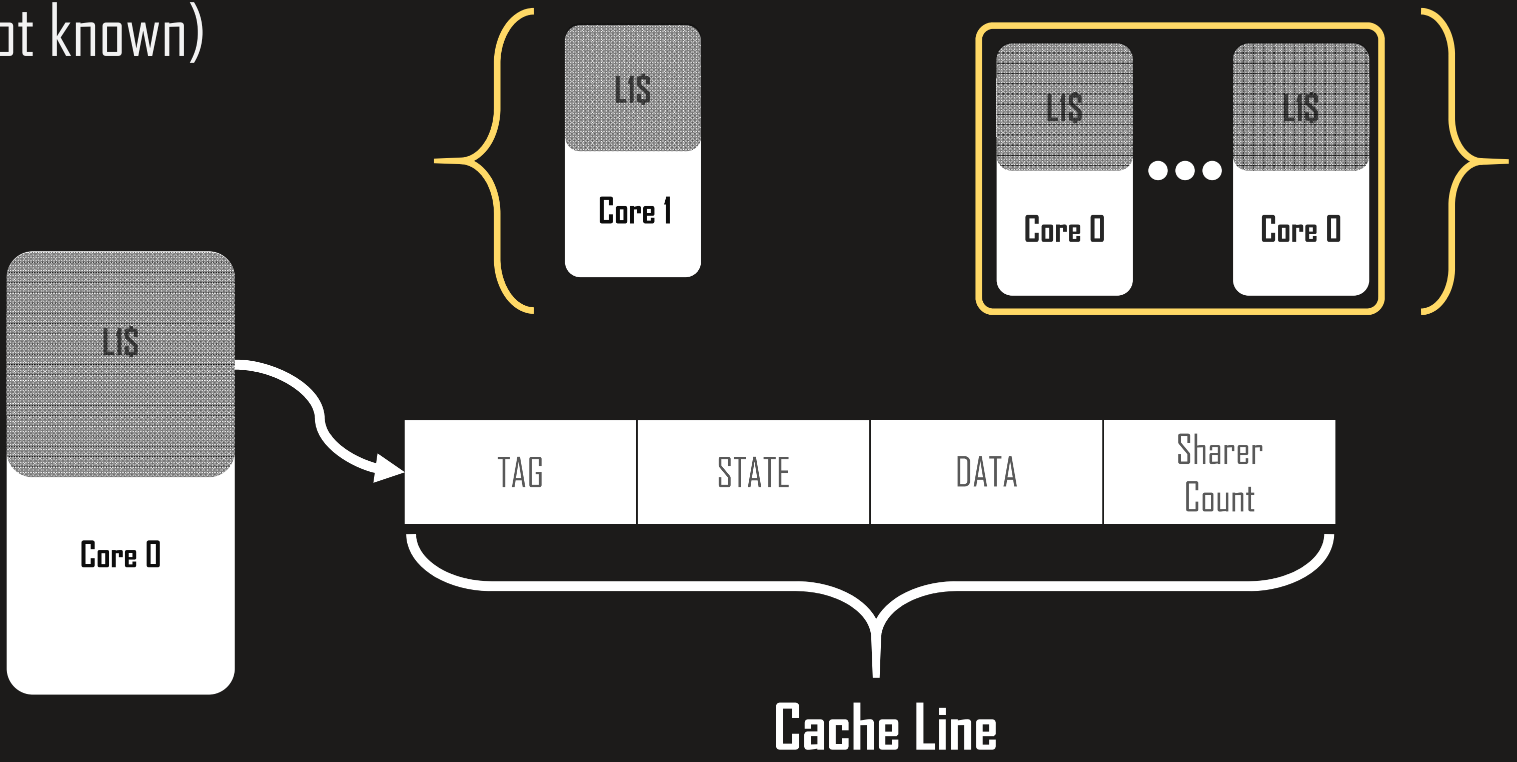
#2: No variables must depend on number of Nodes

#3: No ordering over list/queue sized by number of nodes

#4: Should not parameterize buffers/queues in more than 1-dim.

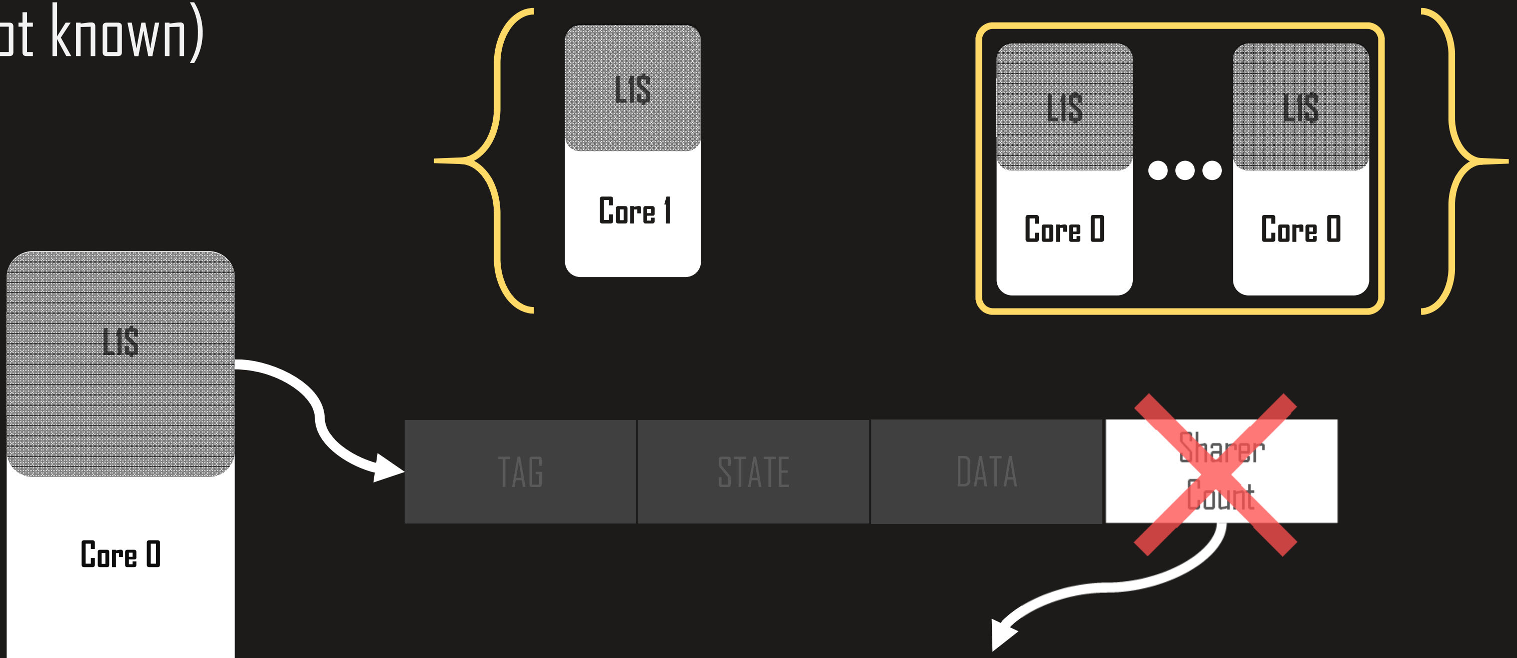
N-1 Nodes to track

(N not known)



N-1 Nodes to track

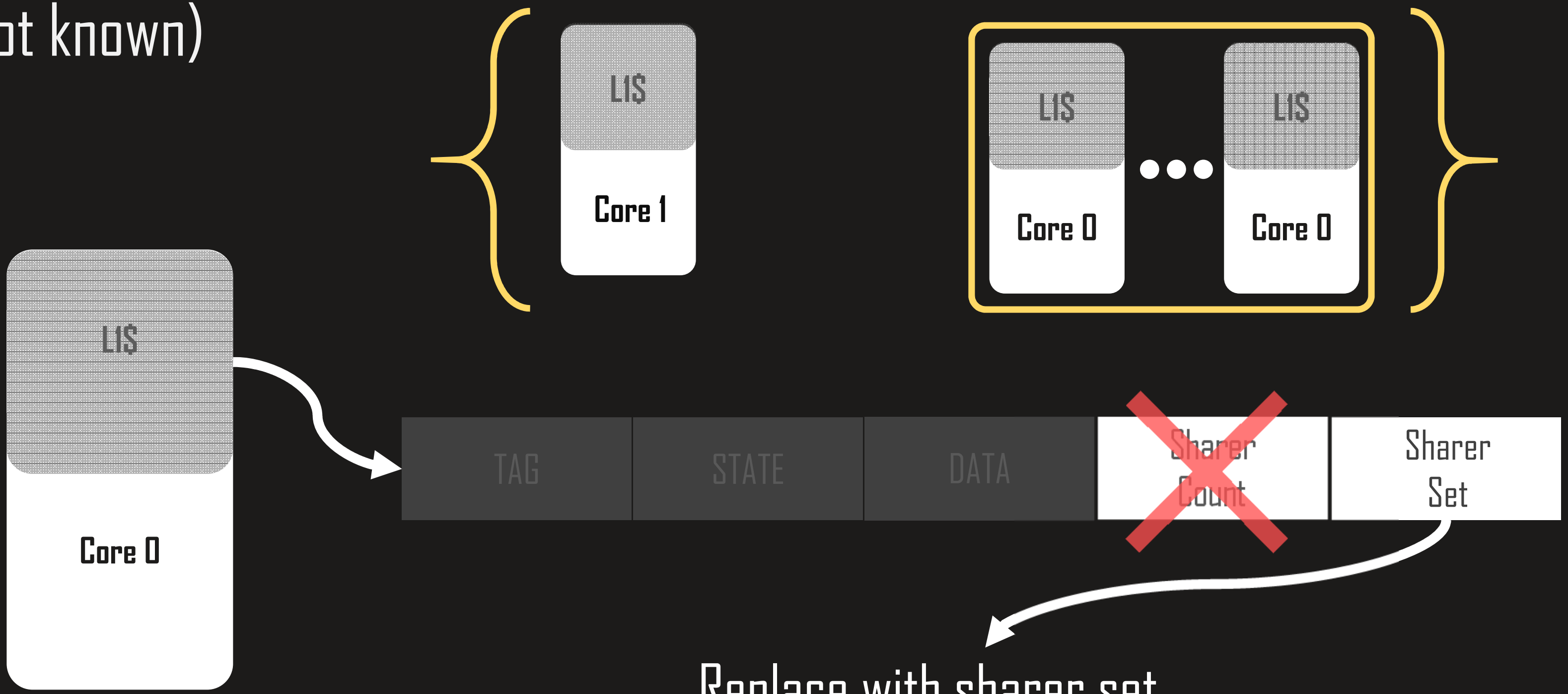
(N not known)



Cannot compare with **parameterized** value or carry out math operations

N-1 Nodes to track

(N not known)



Replace with sharer set
(bit vector)

Guidelines

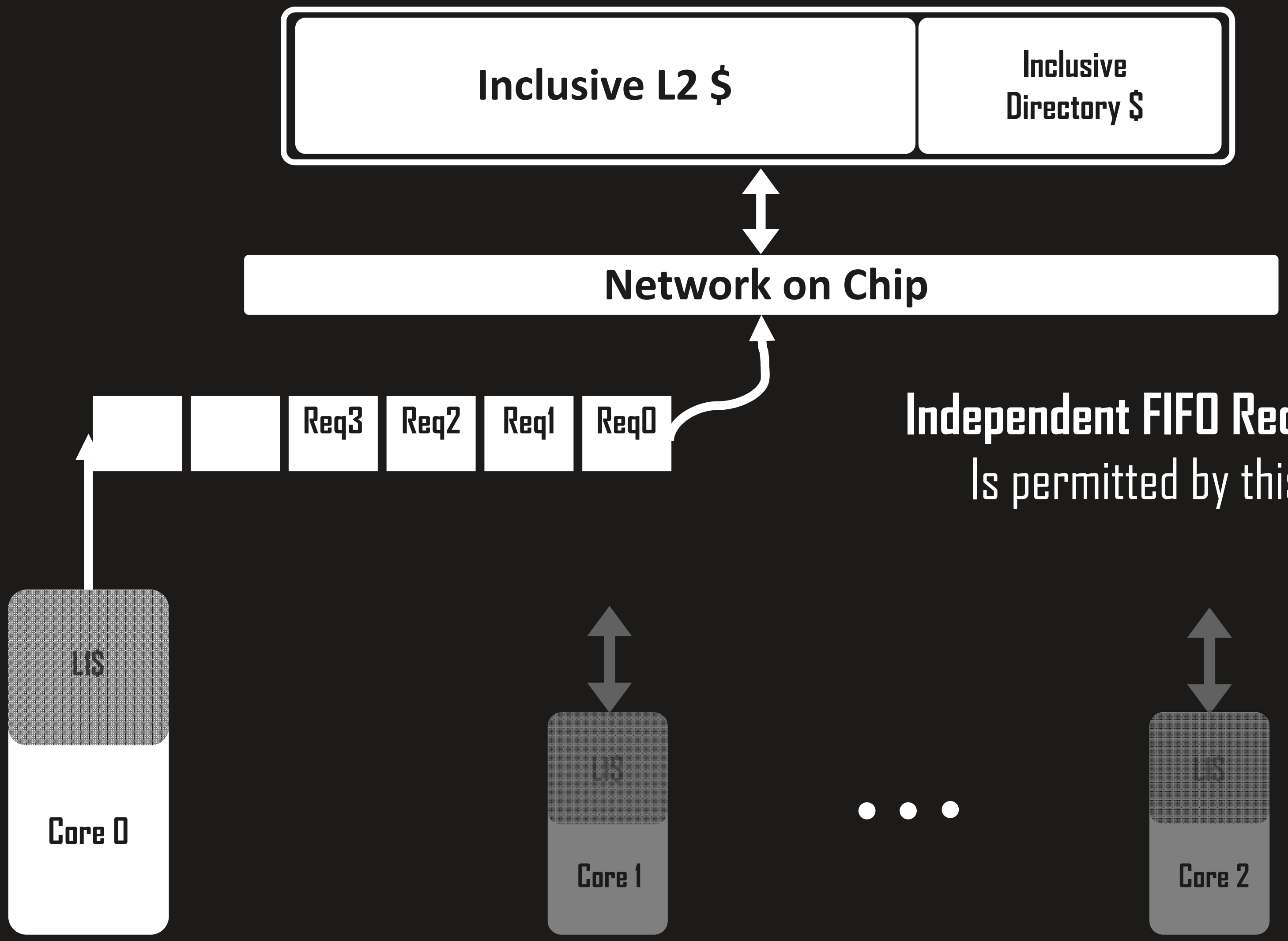
for Simple-PV compliance

#1: Identical Nodes

#2: No variables must depend on number of Nodes

#3: No ordering over list/queue sized by number of nodes

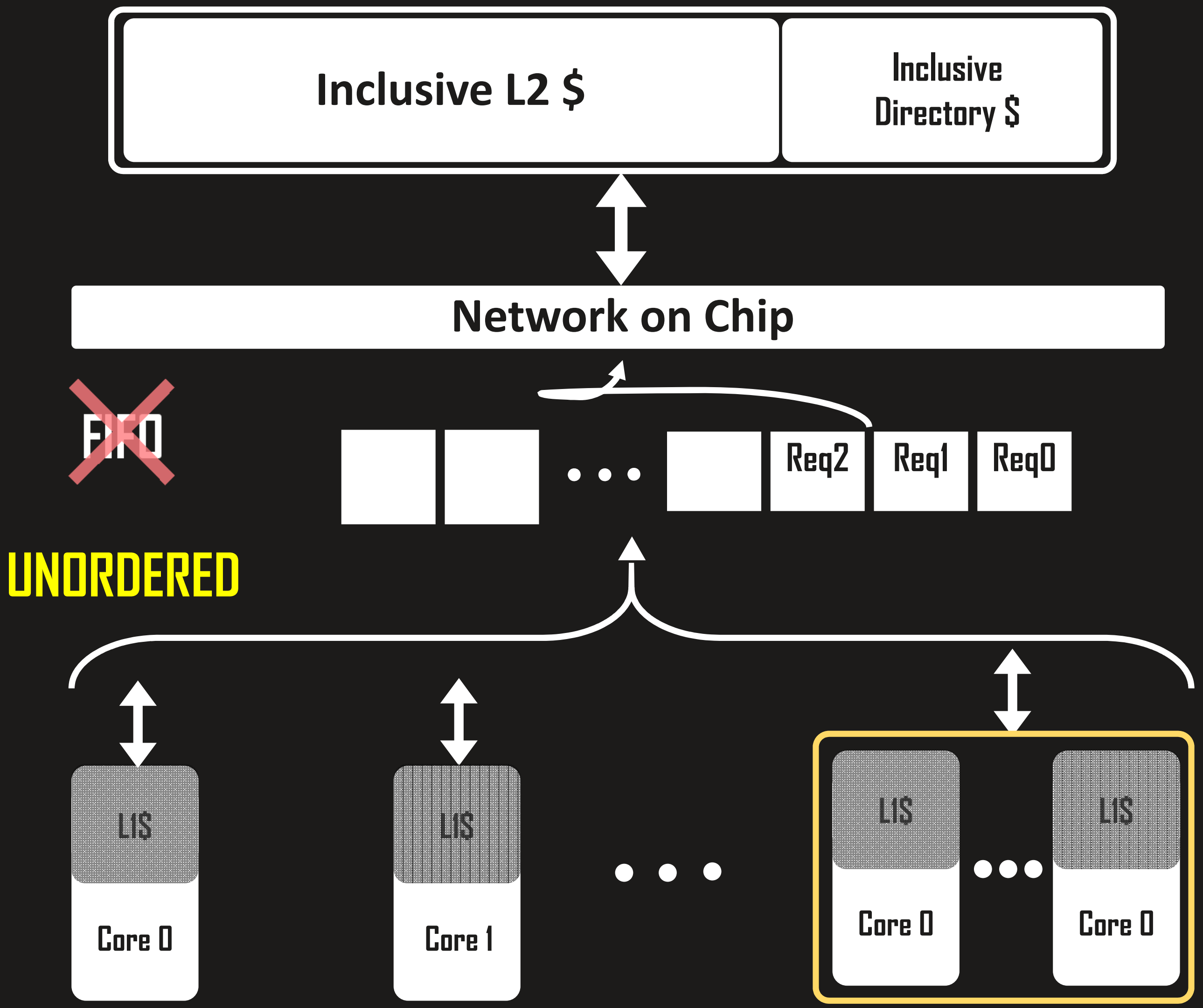
#4: Should not parameterize buffers/queues in more than 1-dim.



Independent FIFO Request Queue

Is permitted by this method

Shared Request Queue



~~FIFO~~

UNORDERED

Guidelines

for Simple-PV compliance

#1: Identical Nodes

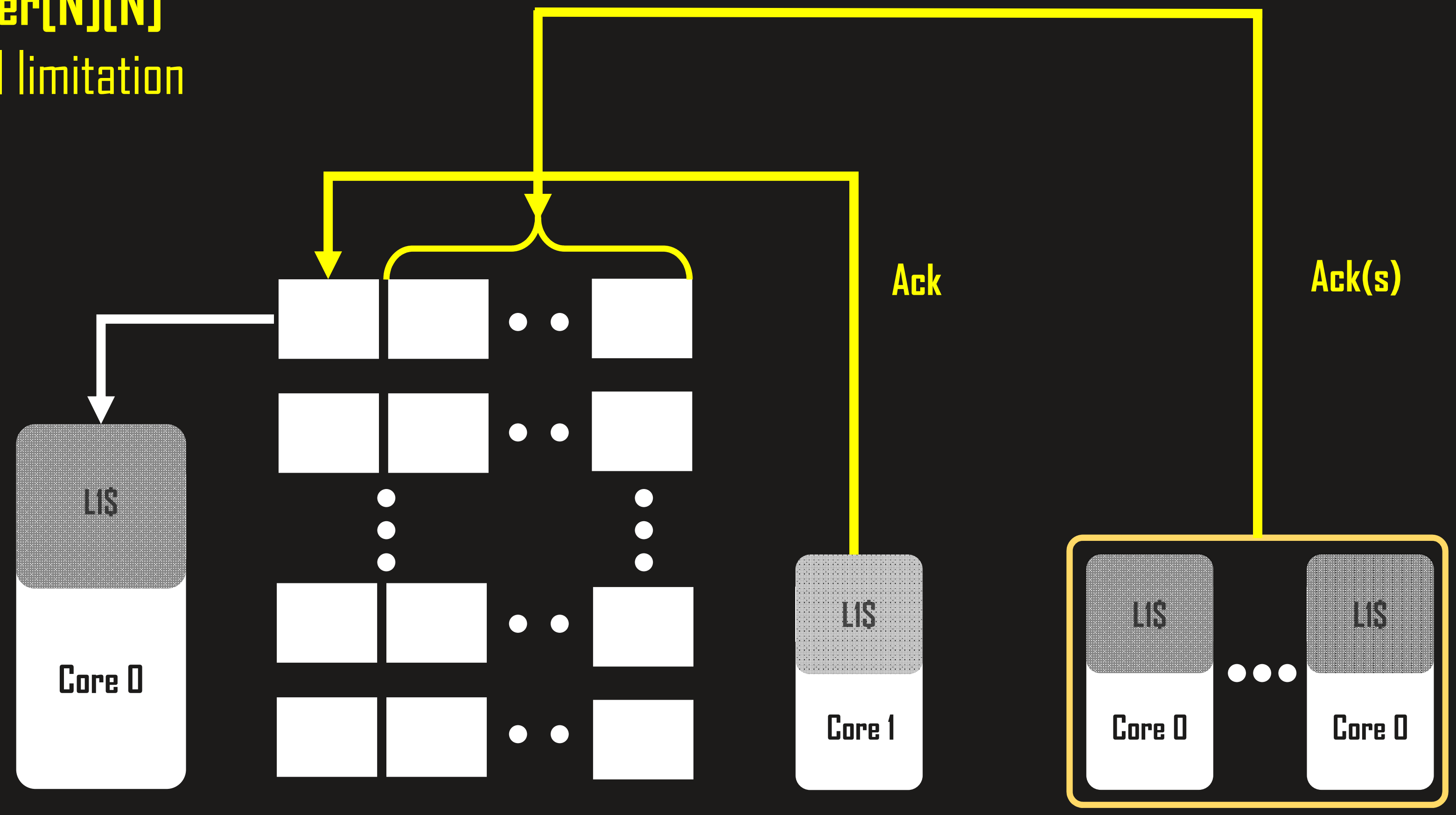
#2: No variables must depend on number of Nodes

#3: No ordering over list/queue sized by number of nodes

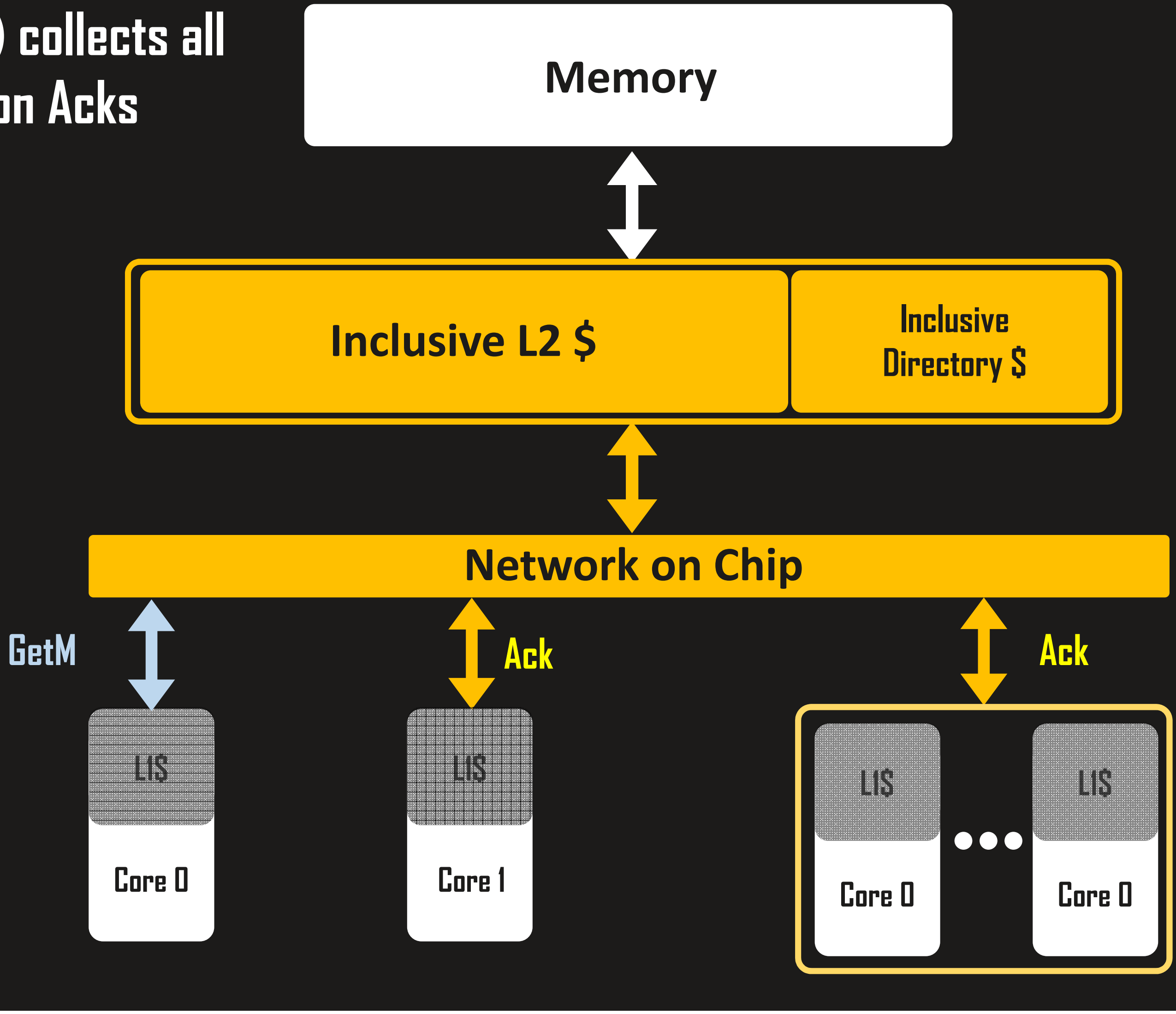
#4: Should not parameterize buffers/queues in more than 1-dim.

NO Buffer[N][N]

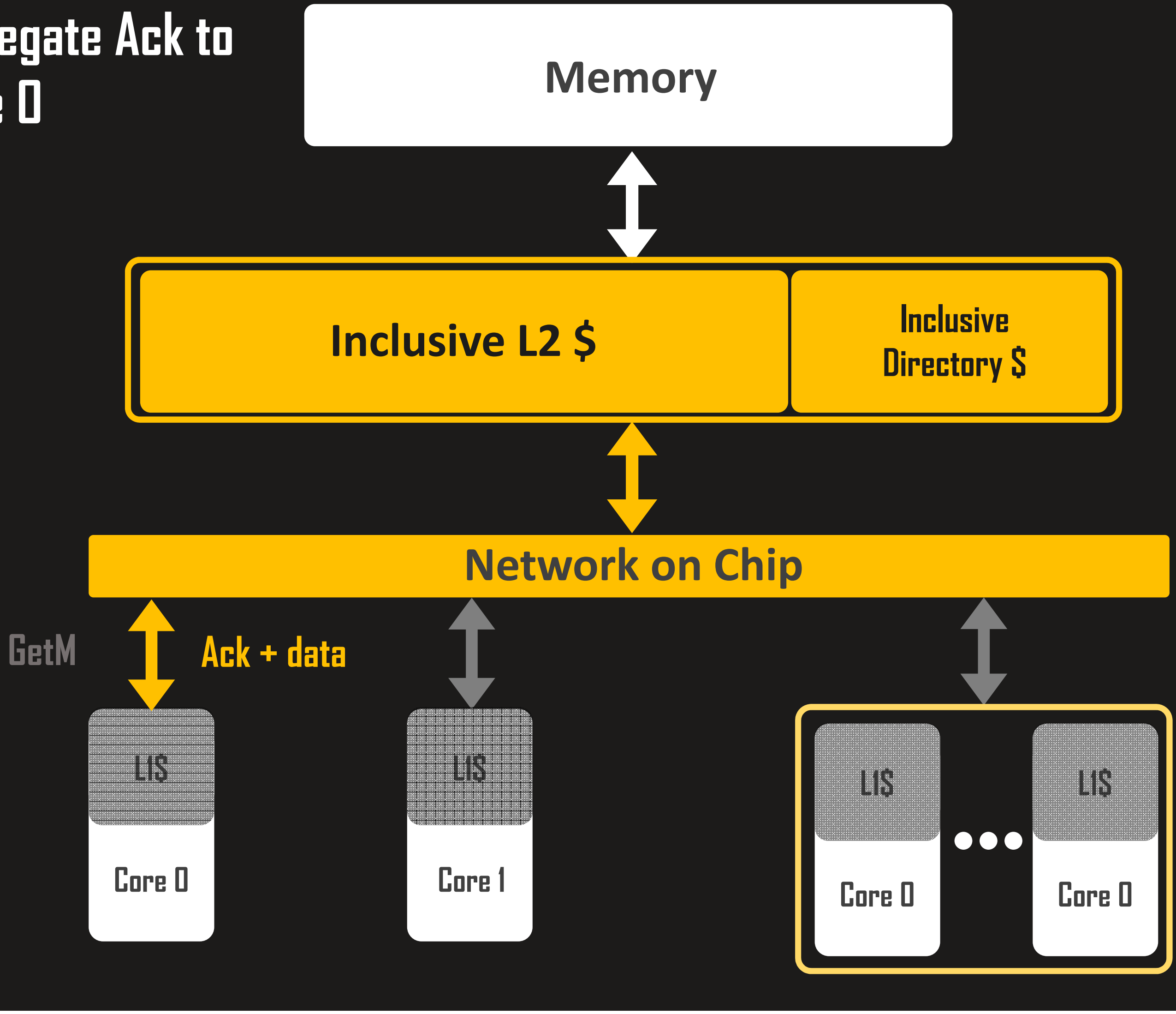
Practical limitation



L2 (Directory) collects all invalidation Acks

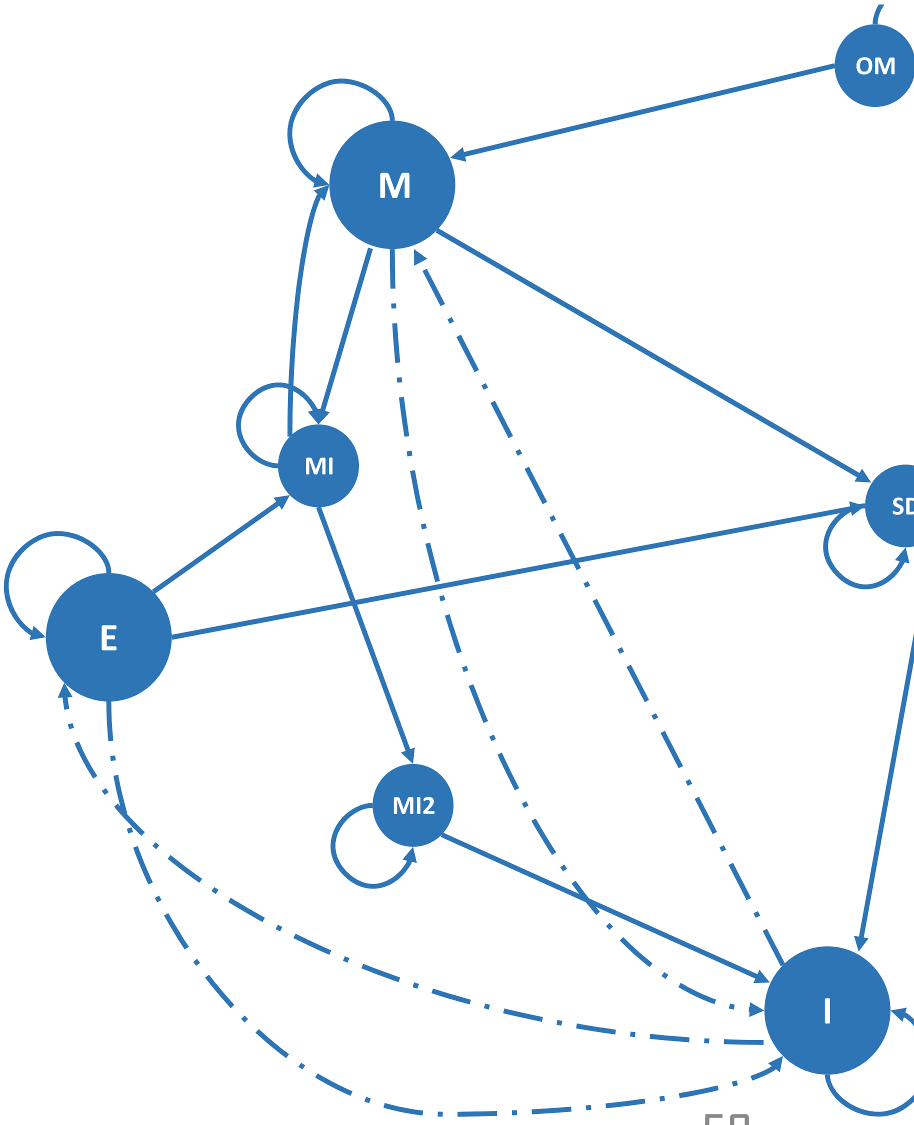


L2 Sends aggregate Ack to Core 0



Overview

- Motivation
- Background
- Parametric Verification
- Design Guidelines
- **PV-MOESI vs OP-MOESI**
- Results
- Conclusion



Optimizations

(OP-MOESI)

Optimization	Compatible with SimplePV?	Impact
Adding **Exclusive** State	Y	**NO IMPACT**
Adding **Owned** state	Y	Add 2 lemmas
Adding **Self-Upgrade**	Y	Add lemma
Adding **Silent Evictions**	Y	Add lemma
Removing the completion messages for **GetS** when data response comes from L2$	Y	Add lemma
Removing the completion messages for **GetM**	N	--

OP-MOESI to **PV-MOESI**

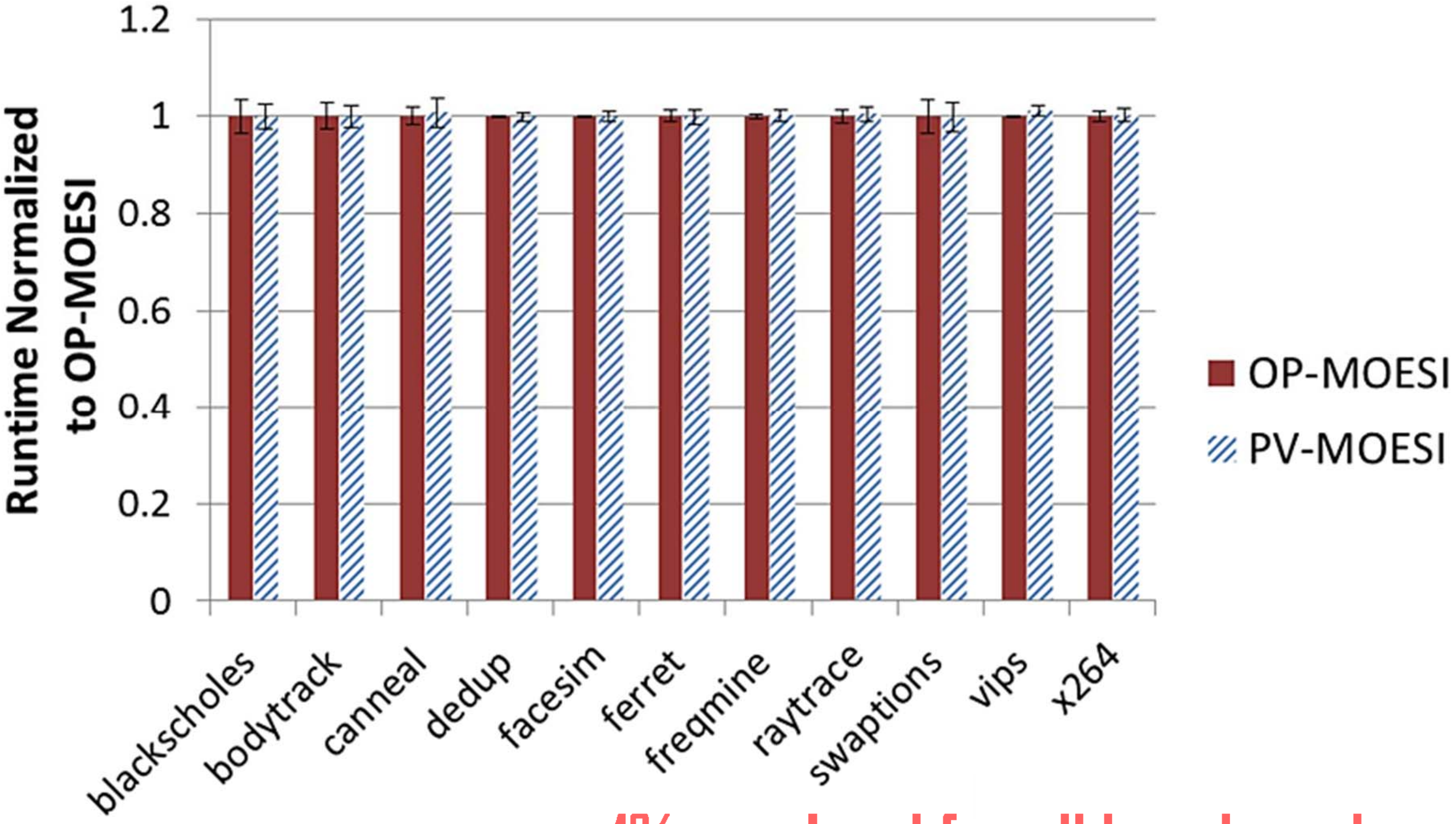
- To ensure successful abstraction by Abster…
  - **For GetM**
    - Replace response **counter** with **sharer set**
    - Let **L2** collect **Acks** and send **aggregated Ack** to requesting **L1**
  - Remove **point-to-point ordering** in all VCs and avoid races by **adding extra messages or transient states** but without blocking **L1**

OP-MOESI to PV-MOESI

- To ensure successful verification by Murphi…
  - **Problem**: multiple in-flight GetM requests without ordering
  - **Solution**: L2 blocks other subsequent requests whenever it receives a GetM request until it receives a Completion message from the requesting L1
  - **Impact**: performance decrease due to blocking at L2

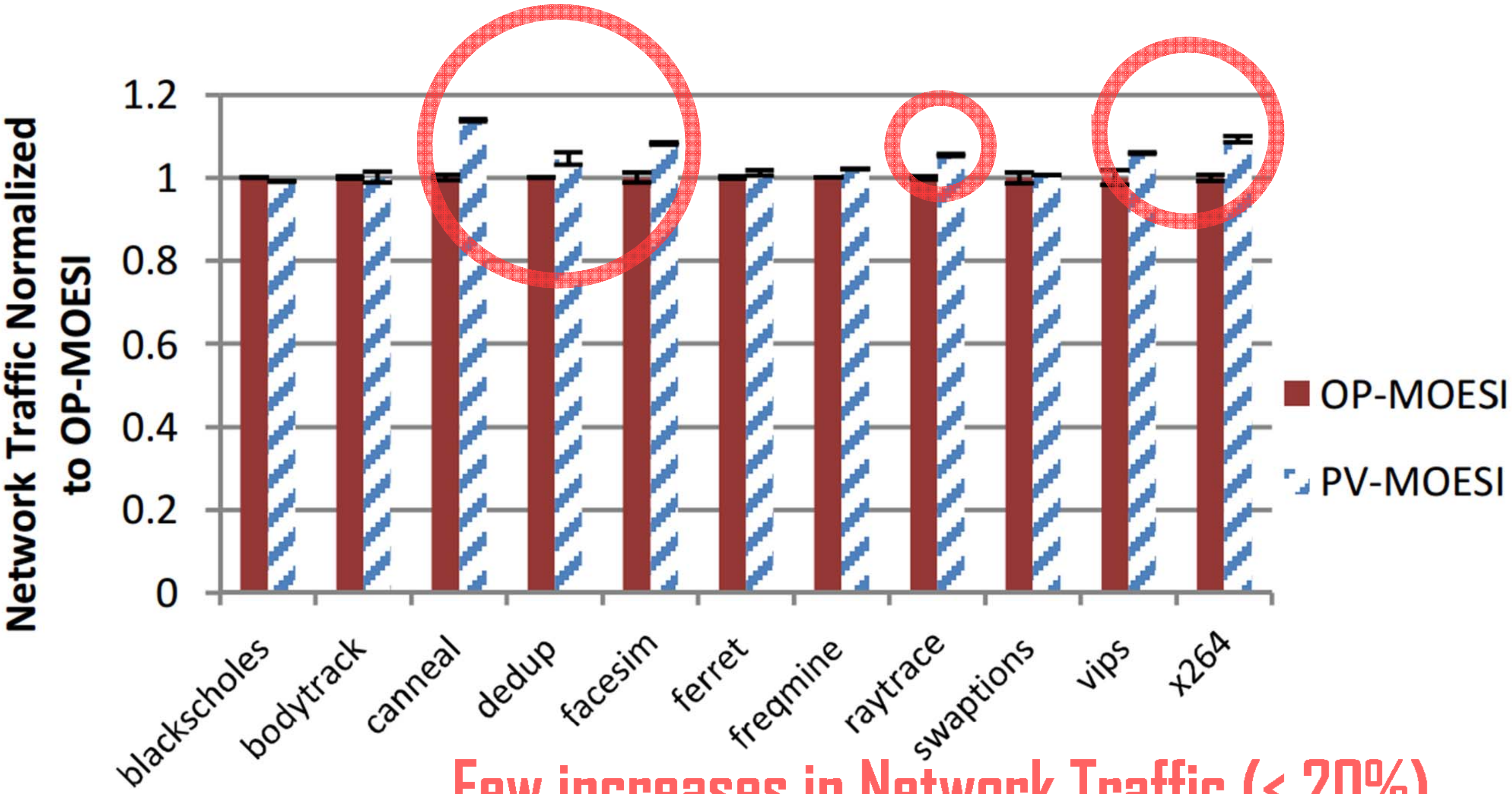
Evaluation:
OP-MOESI vs. PV-MOESI

Runtime



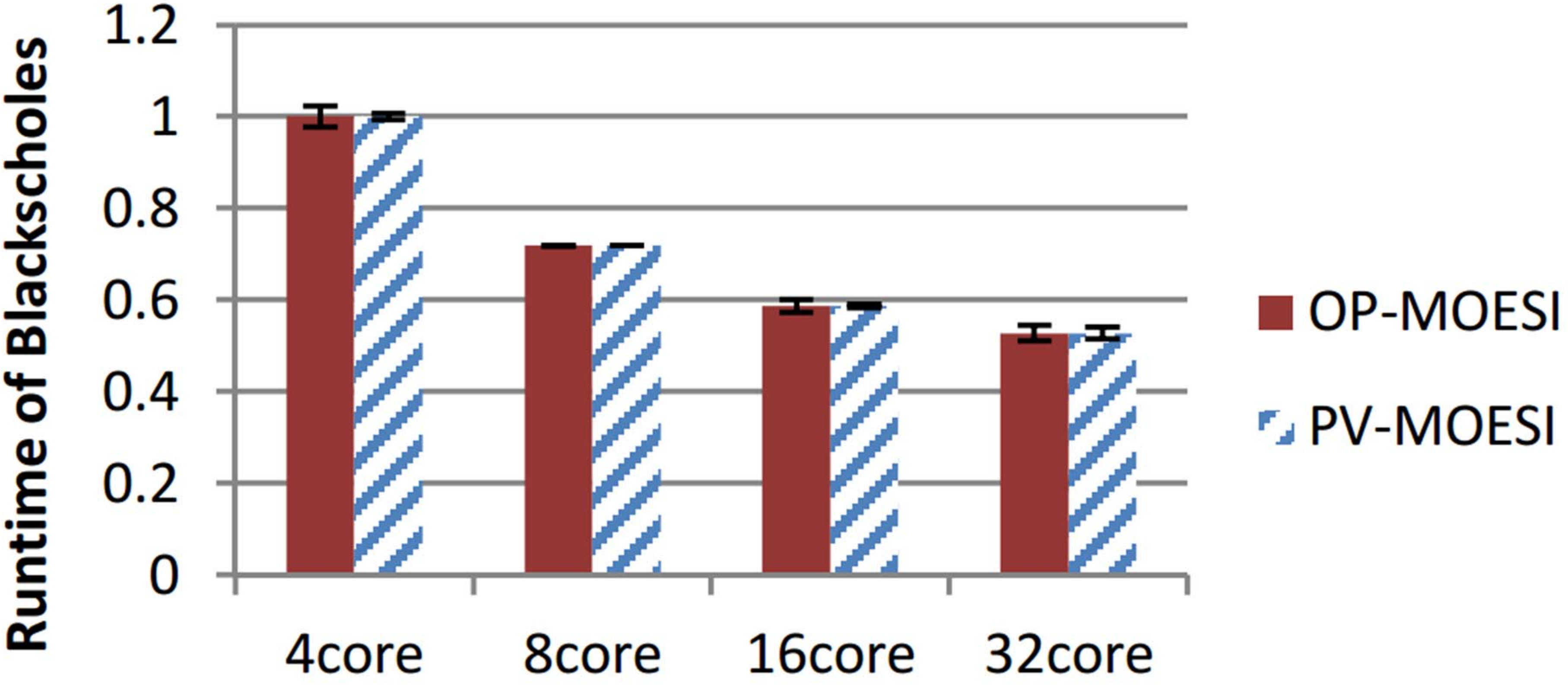
< 1% overhead for all benchmarks

Network Traffic Overhead



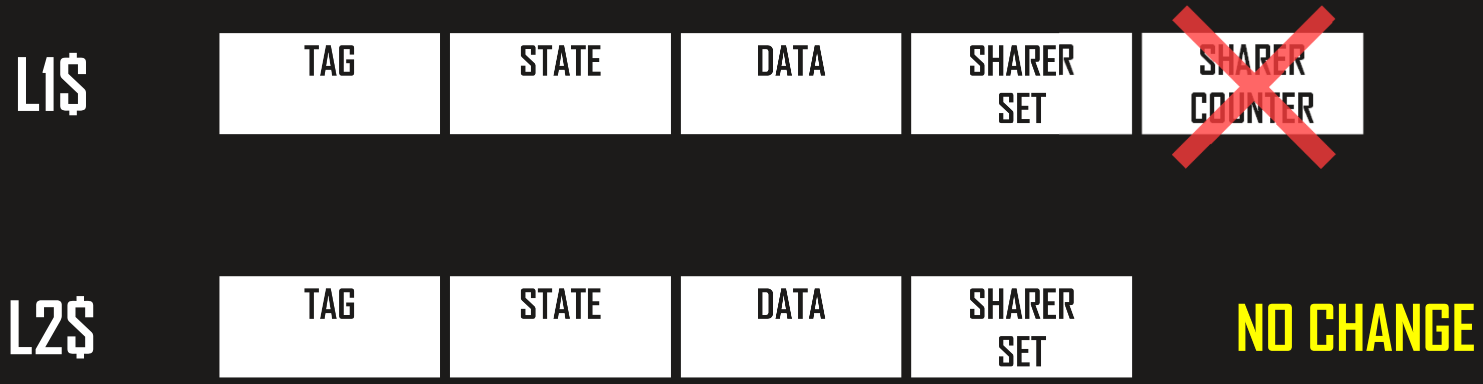
Few increases in Network Traffic (< 20%)

Performance Scalability



Scalable in both directions (up & down)

Storage Overhead



L1$

TAG | STATE | DATA | SHARER SET | ~~SHARER COUNTER~~

L2$

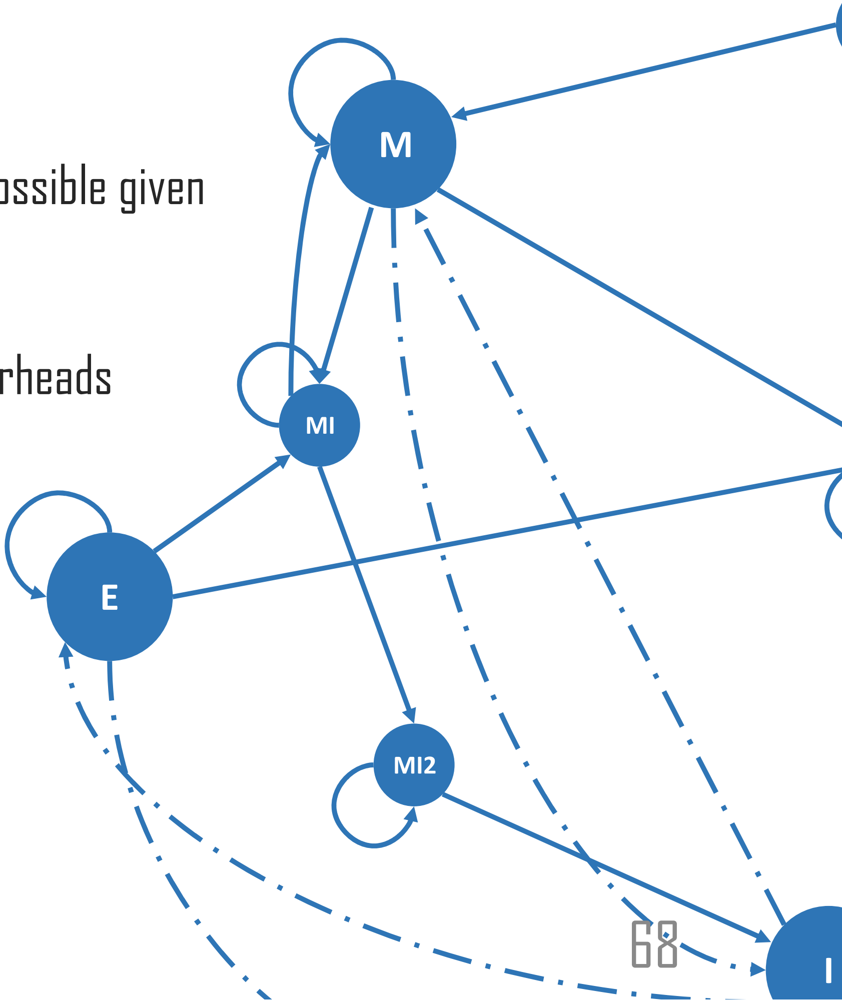
TAG | STATE | DATA | SHARER SET

NO CHANGE

Overhead is generally **negligible**

CONCLUSION

- Design of **parametrically verifiable** coherence protocols is possible given that the **guidelines** introduced here are adhered to
- There is no significant performance drawbacks or storage overheads
- **Automation** is key



Debate

- Is it necessary for a protocol to be parametrically verifiable? There are a few design corners that are cut to make such PV-compliant protocols work. Is this worth it?