Crossing Guard: Mediating Host-Accelerator Coherence Interactions

Lena E. Olson*, Mark D. Hill, David A. Wood

University of Wisconsin-Madison

* Now at Google

ASPLOS 2017

April 10th, 2017

Accelerators are here!

- Complex, programmable accelerators increasingly prevalent
- Many applications: graphics, scientific computing, video encoding, machine learning, etc…
- Accelerators may benefit from cache coherent shared memory
- May be designed by third parties

However…

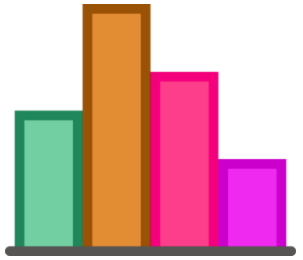
- Host coherence protocols may be proprietary and complex
- Bugs in accelerator implementations might crash host system!
- **Crossing Guard**: coherence interface to safely translate accelerator ↔ host protocol

Outline

Goals

Design

Guarantees



Evaluation

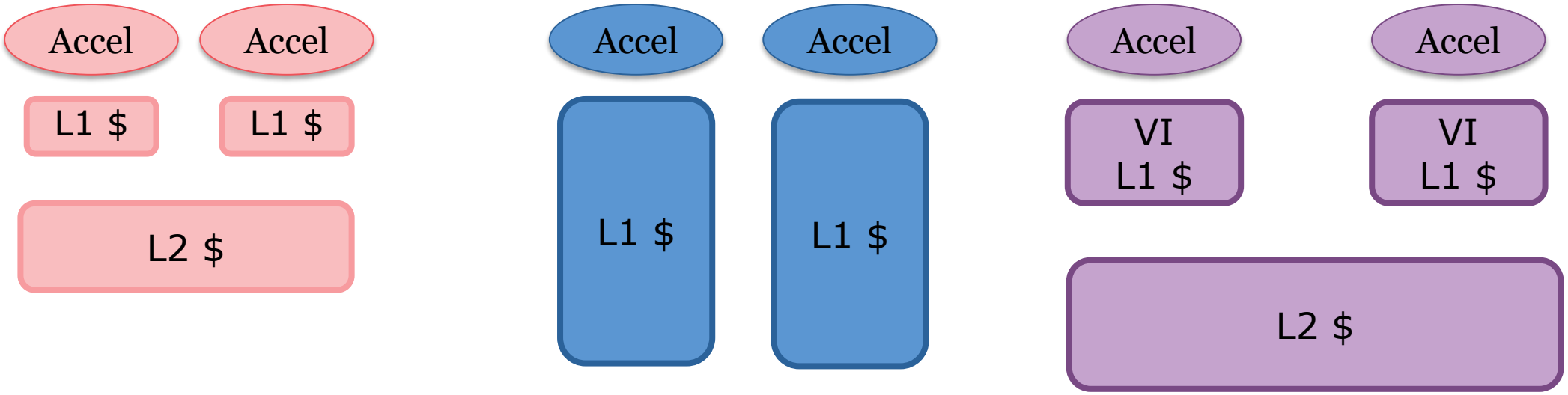
Crossing Guard Goals

When adding accelerators to host coherence protocol:

1. Allow accelerators customized caches
2. Simple, standardized accelerator coherence interface
3. Guarantee safety for the host system

1. Why Customize Caches?

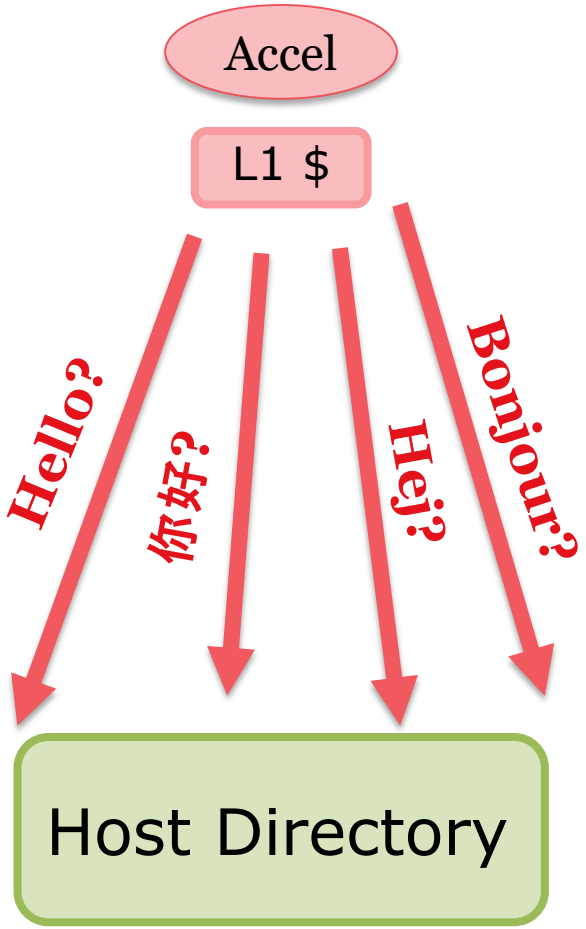
- CPU caches have to work with most types of workloads
- Accelerators may only run some workloads!
  - Optimize caches for likely data access patterns
  - Number of levels, writeback vs. writethrough, MSI vs VI, etc.



2. Why Simple, Standardized Interface?

Host systems speak different protocols…

- Expensive to redesign for each one!
  - Intel, AMD, ARM, IBM, Oracle…
  - CCIX shows industry cares!



2. Why Simple, Standardized Interface?

L1 controller from gem5's MOESI_hammer

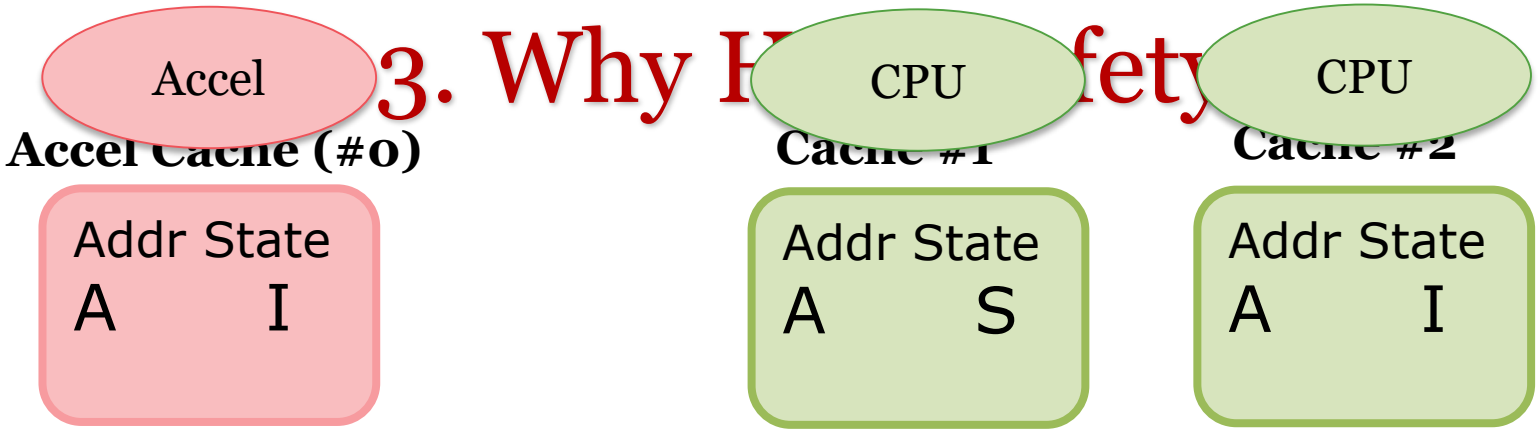
Events

States

	Ifetch	Load	Store	Invalidate	Other GETS only	Other GETS	Merged GETS	Other GETX	Ack	Shared Ack	Data	Shared Data	Exclusive Data	Writeback Ack	Writeback Nack	All acks	All acks no sharers	L2 Replacement	L1 to L2	Trigger L2 to LID	Trigger L2 to LII	Complete L2 to L1
I	i a uim um k / IS	i a udm um k / IS	i b udm um k / IM	fi	fi	fi		fi														
S	h uih k	h udh k	i b udm um k / SM	f cc gr l / I	fi	f l		f cc gr l / I										cc r ka / I	i g v li s	i r l fu s z ll / ST	i r j fu s z ll / ST	
O	h uih k	h udh k	i b udm um k / OM	e cc gr l / I	e l	e l	em l	e cc gr l / I										i d cc r ka / OI	i g v li s	i r l fu s z ll / OT	i r j fu s z ll / OT	
M	h uih k	h udh k	h udh k / MM	e cc gr l / I	e l / O	e l / O	em l / O	e cc gr l / I										i d cc r ka / MI	i g v li s	i r l fu s z ll / MT	i r j fu s z ll / MT	
MM	h uih k	h udh k	h udh k	e cc gr l / I	e l / O	e cc gr l / I	em l / O	e cc gr l / I										i d cc r ka / MI	i g v li s	i r l fu s z ll / MMT	i r j fu s z ll / MMT	
IR	i r a uim um k / IS	i r a udm um k / IS	i r b udm um k / IM	z	z	z	z	z											z			
SR	h uim uh k ka / S	h udm uh k ka / S	i b udm um k / SM	z	z	z	z	z											z			
OR	h uim uh k ka / O	h udm uh k ka / O	i b udm um k / OM	z	z	z	z	z											z			
MR	h uim uh k ka / M	h udm uh k ka / M	h udm uh k ka / MM	z	z	z	z	z											z			
MMR	h uim uh k ka / MM	h udm uh k ka / MM	h udm uh k ka / MM	z	z	z	z	z											z			
IM	z	z	z	fi	fi	fi		fi	m o n		u m o sx n kd / MM^W		u m o sx n kd / MM^W				kk gm sxt s i kd / MM	z	z			
SM	h uih k	h udh k	z	f cc l / IM	f l	f l		f cc l / IM	m o n		v m o n / ISM		v m o n / ISM				kk sxt gm s i kd / MM	z	z			
OM	h uih k	h udh k	z	e cc l / IM	e l	e l	em l	e cc l / IM	m o n		kk m o n		kk m o n			sxt gm s i kd / MM	sxt gm s i kd / MM	z	z			
ISM	h uih k	h udh k	z						m o n		kk m o n		kk m o n				sxt gm s i kd / MM	z	z			
M^W	h uih k	h udh k	h udh k / MM^W						m o n	kk m o n	kk m o n	kk m o n	kk m o n			kk gm s i kd / M	gm s i kd / M	z	z			
MM^W	h uih k	h udh k	h udh k						m o n	kk m o n	kk m o n	kk m o n	kk m o n			kk gm s i kd / MM	gm s i kd / MM	z	z			
IS	z	z	z	fi	fi	fi		fi	m o n	m r o n	u m o hx uo n kd / SS	u r m o hx uo n kd / SS	u m o hx n kd / M^W			kk gs s h i kd / O	kk gs s h i kd / O	z	z			
SS	h uih k	h udh k	z						m o n	m r o n	kk m o n	kk m o n	kk m o n			gs s i kd / S	gs s i kd / S	z	z			
OI	z	z	z	q l / II	sq l	sq l	qm l	q l / II						q s l kd / I	kk s l kd / I			z	z			
MI	z	z	z	q l / II	sq l / OI	sq l / OI	qm l / OI	q l / II						t s l kd / I	kk s l kd / I			z	z			
II	z	z	z	fi	fi	fi		fi						g s l kd / I	s l kd / I			z	z			
ST	z	z	z	z	z	z	z	z										z	z			i kd / SR
OT	z	z	z	z	z	z	z	z										z	z			i kd / OR
MT	z	z	z	z	z	z	z	z										z	z			i kd / MR
MMT	z	z	z	z	z	z	z	z										z	z			i kd / MMR

(Transition table in style of Sorin et al.)

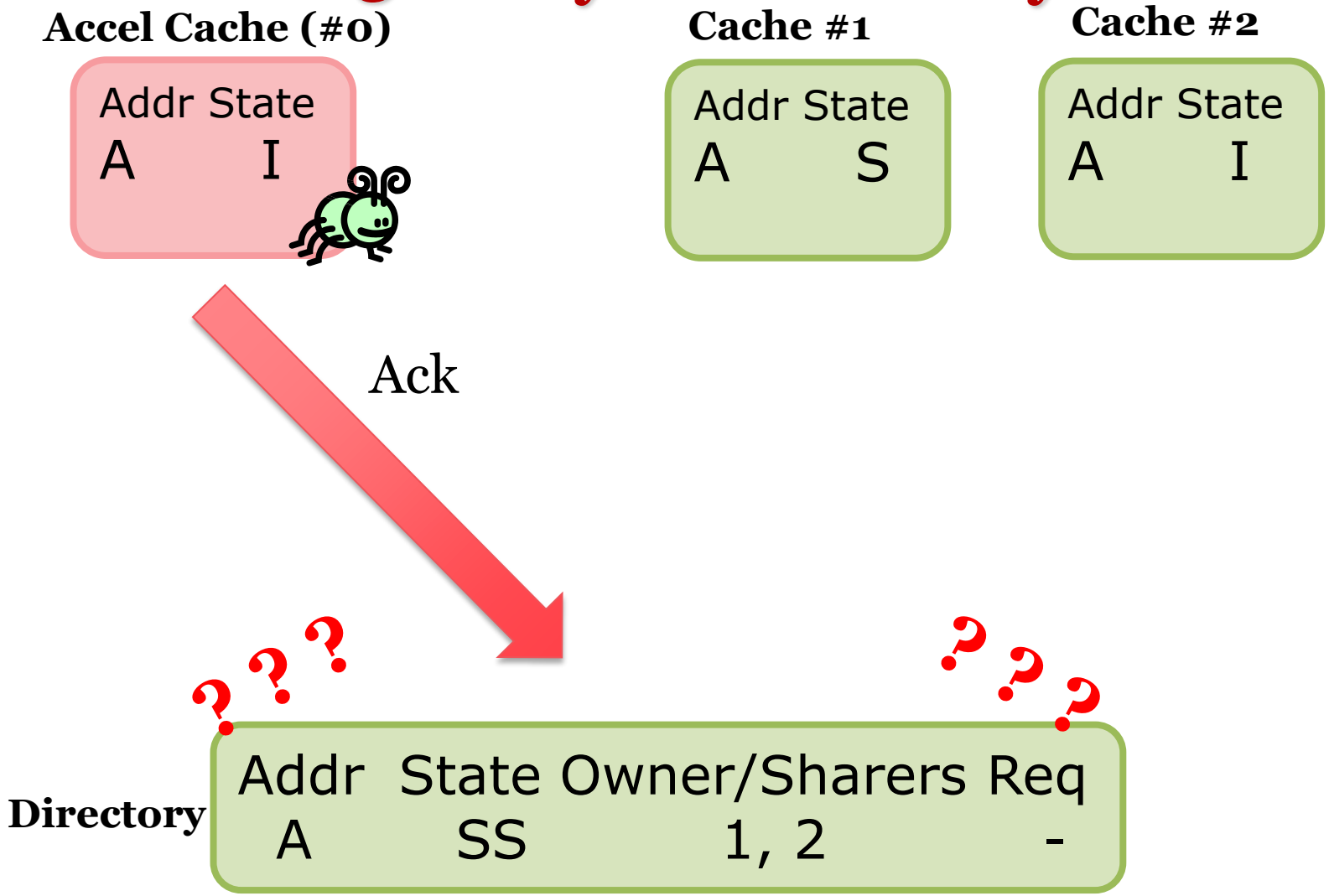
3. Why H… fety



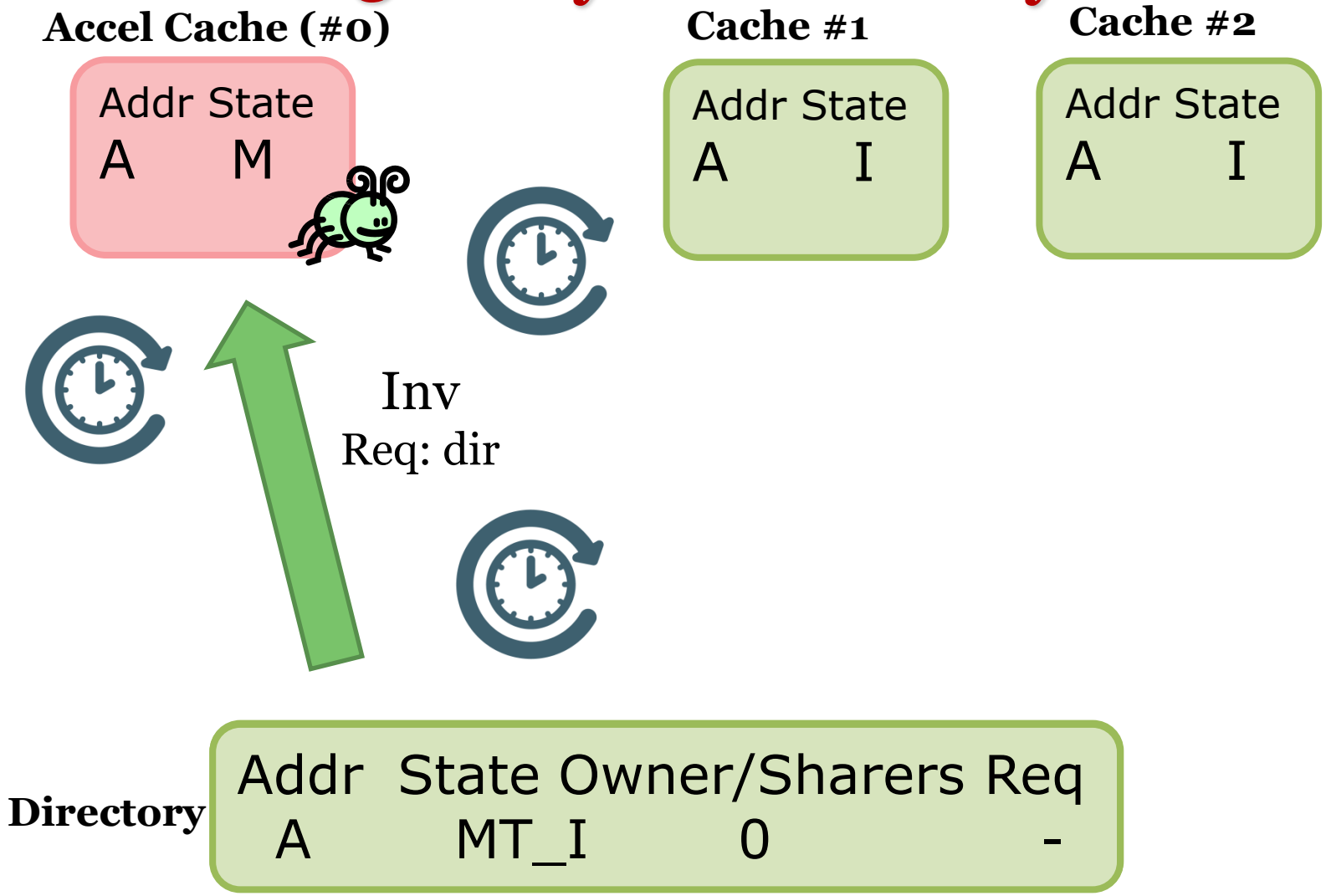
Directory

Addr	State	Owner/Sharers	Req
A	SS	1, 2	-

3. Why Host Safety?



3. Why Host Safety?

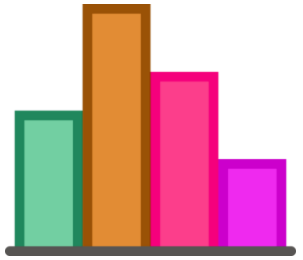


Outline

Goals

Design

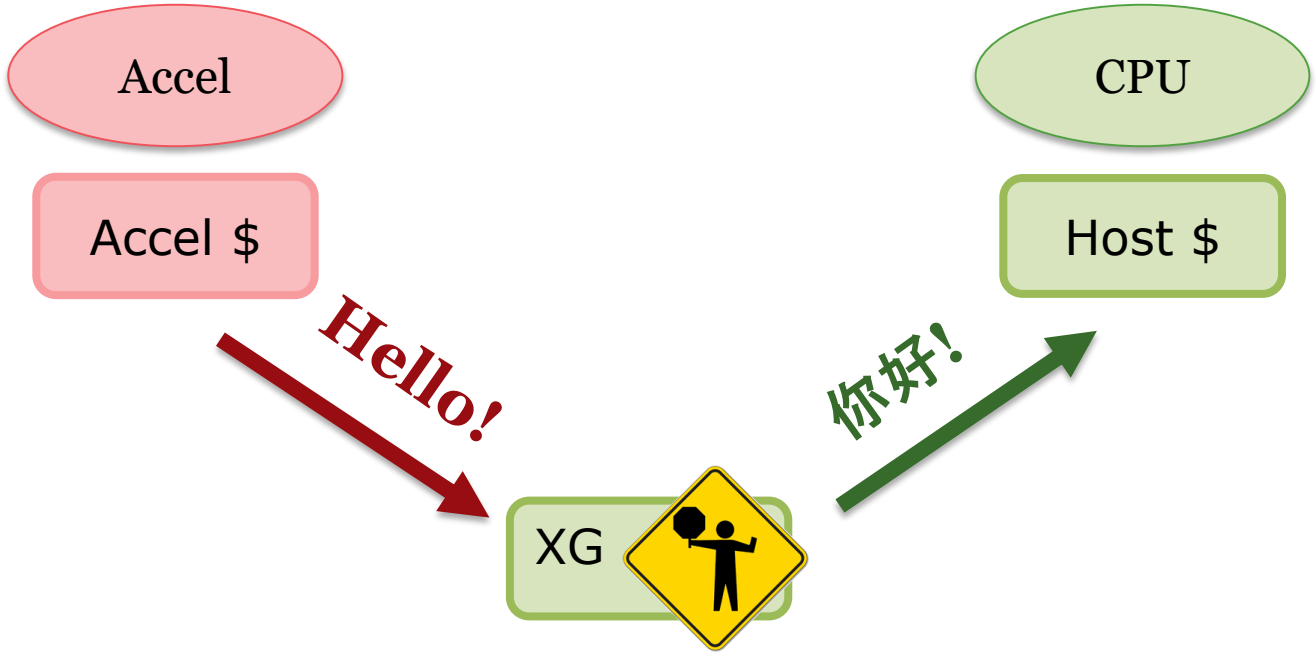
Guarantees



Evaluation

Crossing Guard

- Hardware translating between host and accelerator protocols



- Set of accelerator ↔ host coherence messages (like an API)

Crossing Guard Interface

Accelerator → Host Requests

- GetS, GetM
- PutS, PutE, PutM

Host → Accelerator Requests

- Invalidate

Host → Accelerator Responses

- DataS, DataE, DataM
- Writeback Ack

Accelerator → Host Responses

- InvAck, Clean Writeback, Dirty Writeback

Crossing Guard

- Hides implementation details of host protocol
  - No counting acks, sending unblocks, handling races, etc.
- Moves protocol complexity into Crossing Guard hardware
  - Only implemented once per host system
  - By experts!

Experimental Implementation

- Coherence controllers / protocols implemented in slicc
- Simulations using gem5
- Code and transition tables available online

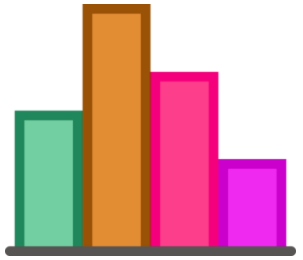
  http://research.cs.wisc.edu/multifacet/xguard/

Outline

Goals

Design

Guarantees

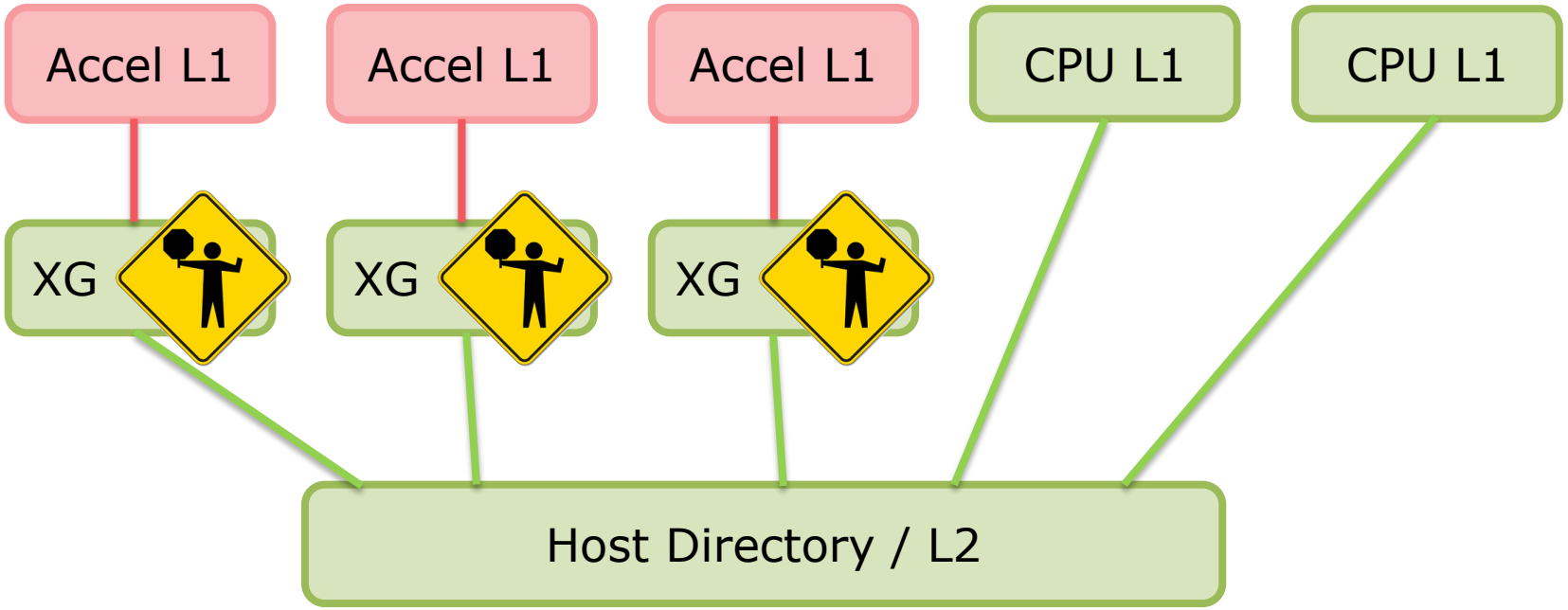


Evaluation

1. Customize Caches ✓

- Designed + implemented two sample systems

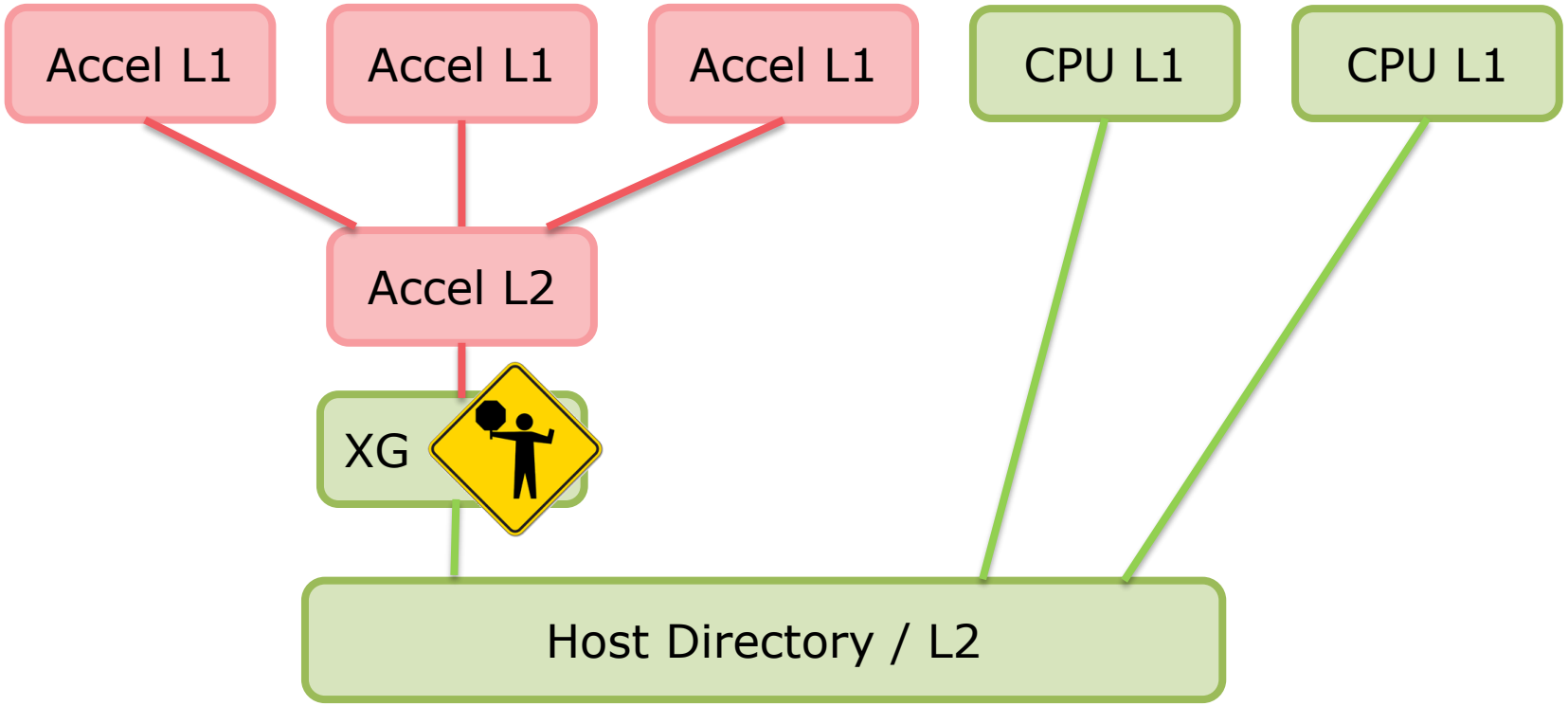
Private Per-Core L1 at Accelerator



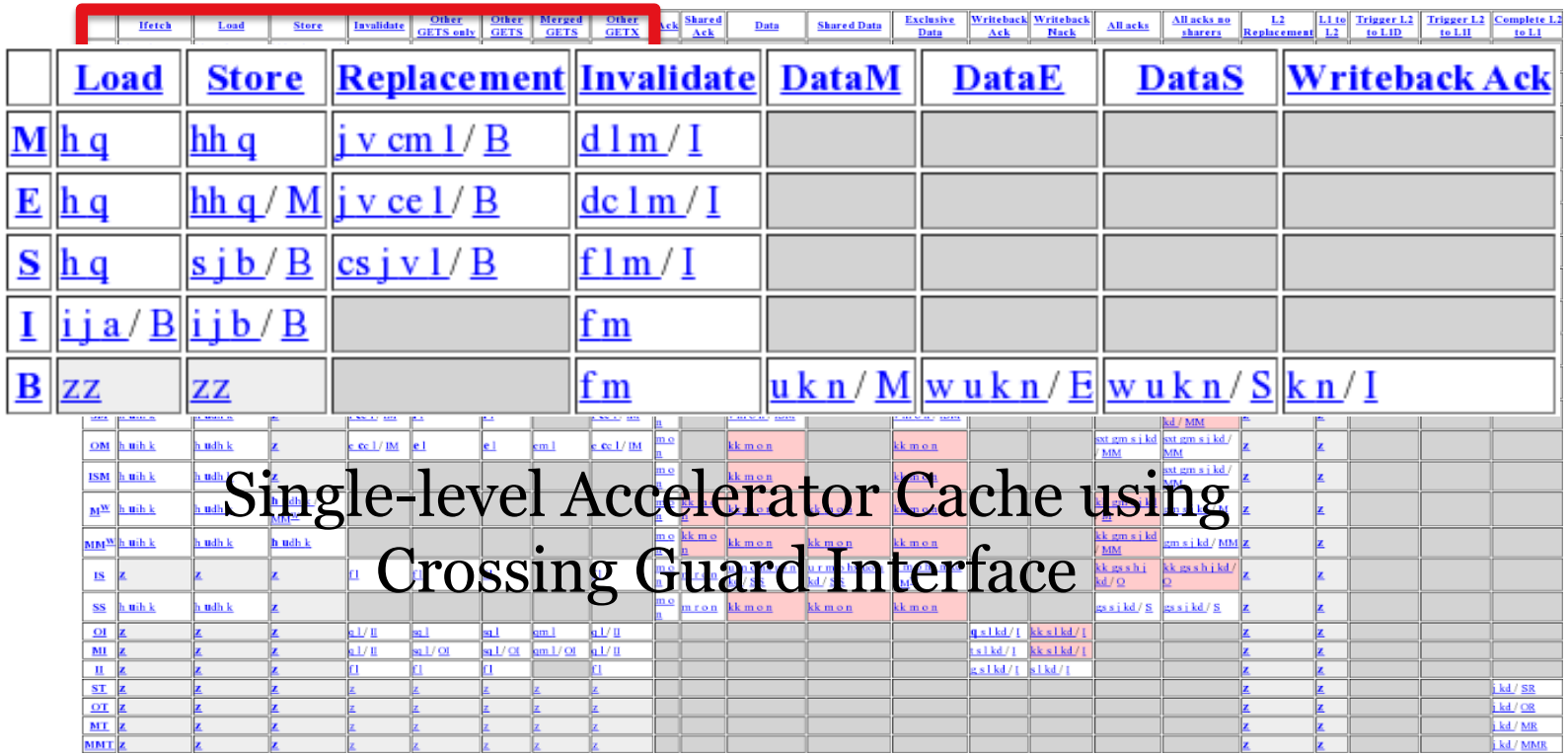
1. Customize Caches ✓

- Designed + implemented two sample systems

Private L1s + Shared L2 at Accelerator



2. Simple, Standardized Interface ✓



	Load	Store	Replacement	Invalidate	DataM	DataE	DataS	Writeback Ack
M	h q	hh q	j v cm l / B	d l m / I				
E	h q	hh q / M	j v ce l / B	dc l m / I				
S	h q	s j b / B	cs j v l / B	f l m / I				
I	i j a / B	i j b / B		f m				
B	zz	zz		f m	u k n / M	w u k n / E	w u k n / S	k n / I

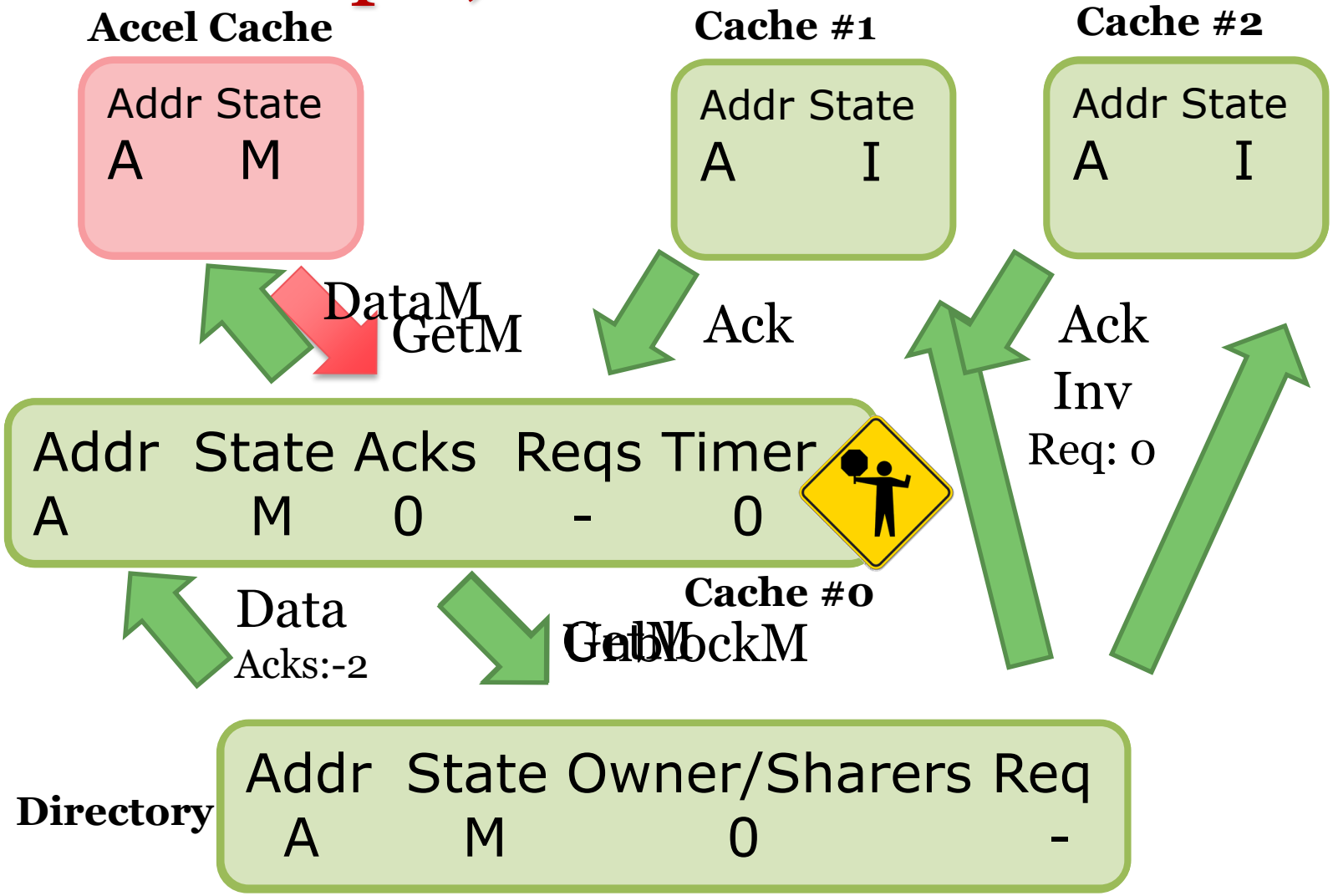
Single-level Accelerator Cache using Crossing Guard Interface

Controller	States	Transitions
AMD Hammer-like Private $$	**24**	**148**

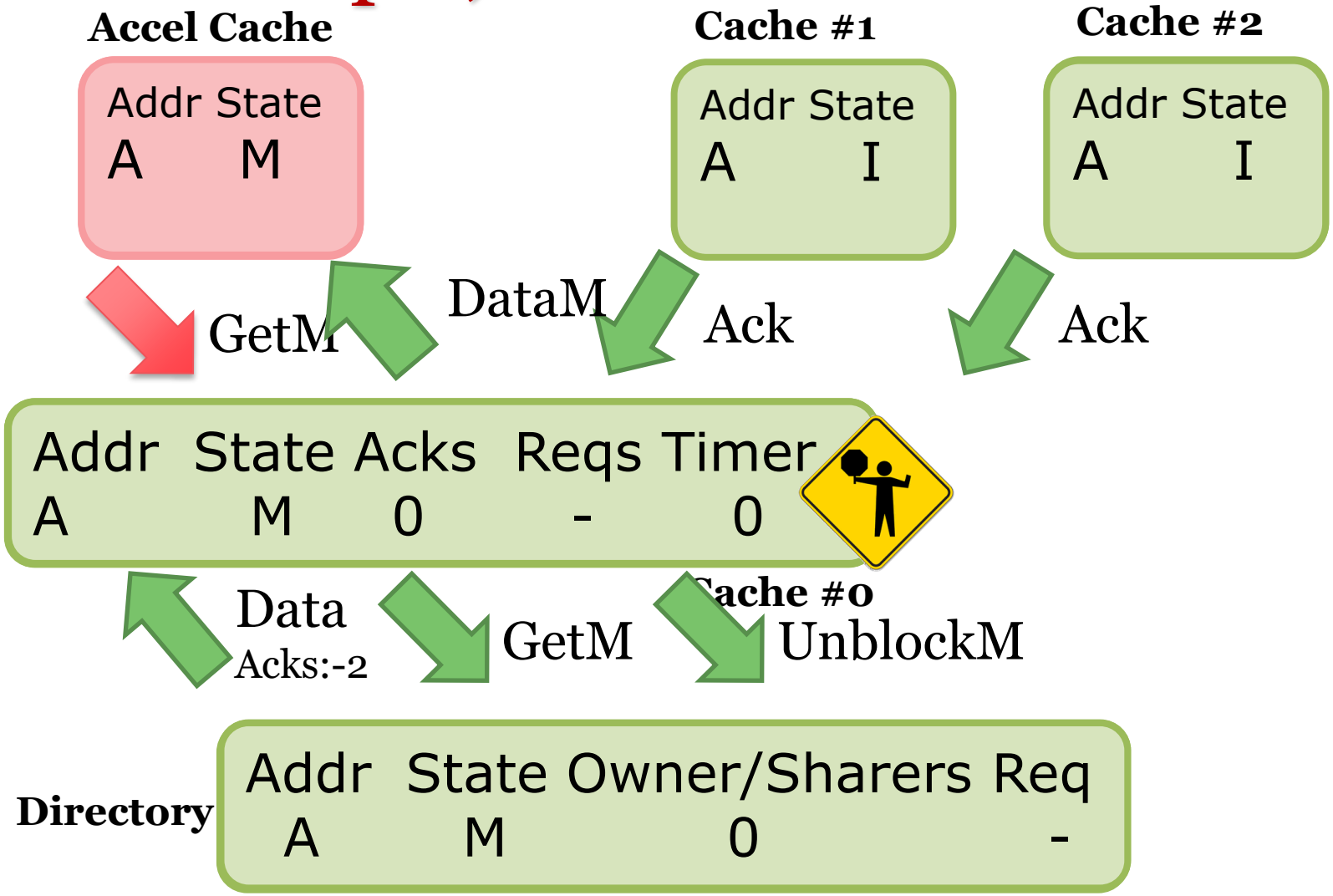
2. Simple, Standardized Interface ✓

- Implemented Crossing Guard controller for two host protocols
  - AMD Hammer-like Exclusive MOESI
  - Two-Level MESI Inclusive
- **Modularity:** Host and Accelerator protocol choice is completely independent

2. Simple, Standardized Interface ✓



2. Simple, Standardized Interface ✓



3. Host Safety ✓

Accel Cache

Addr	State
A	I

Cache #1

Addr	State
A	S

Cache #2

Addr	State
A	I

Ack

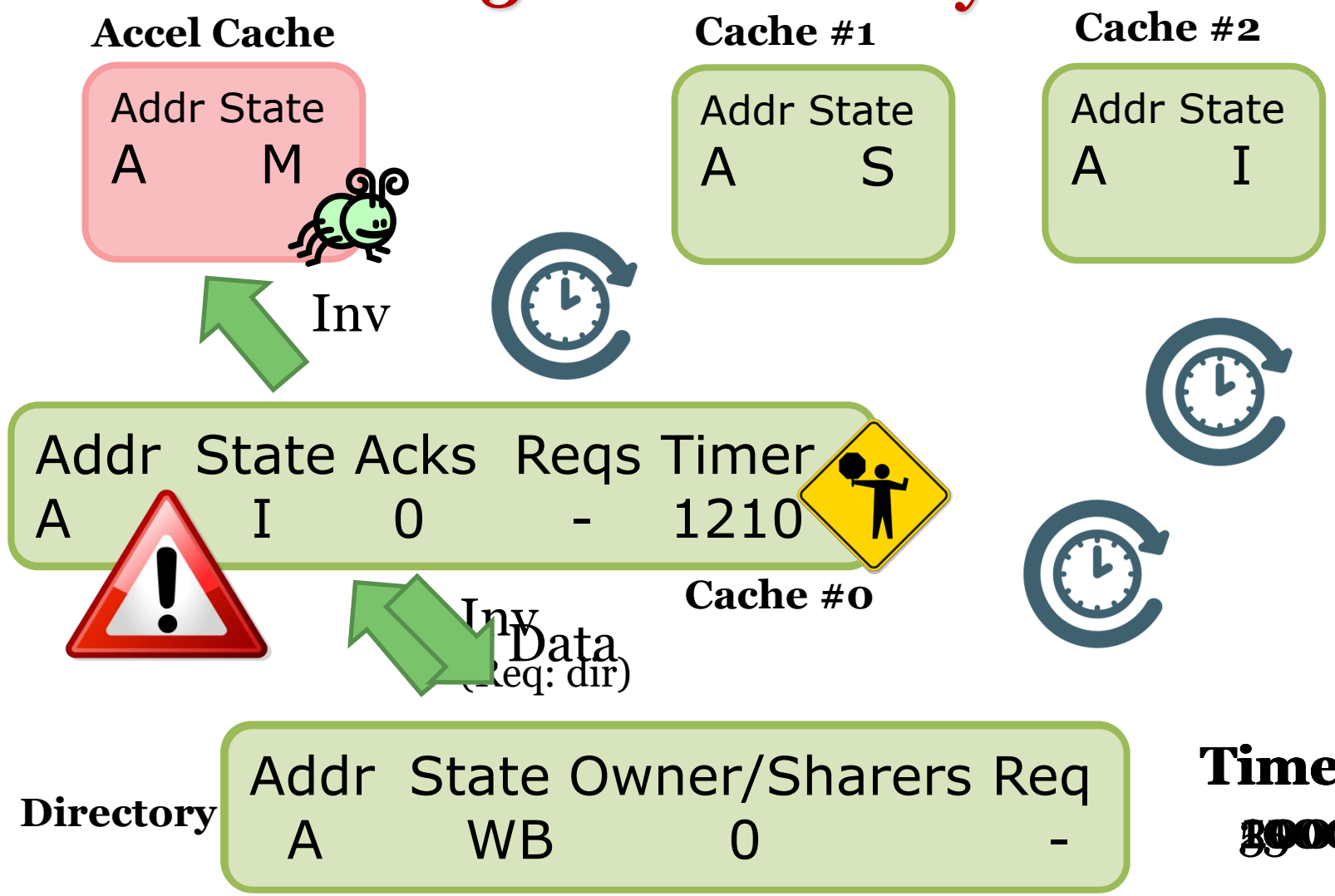
Addr	State	Acks	Reqs	Timer
A	I	0	-	0

Cache #0

Directory

Addr	State	Owner/Sharers	Req
A	SS	1, 2	-

3. Host Safety ✓

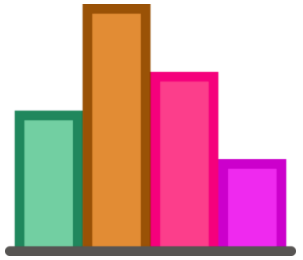


Outline

Goals

Design

Guarantees



Evaluation

Evaluation

I. Does it provide coherence to correct accelerator?

II. Does it provide safety to host?

III. Does it allow high performance?

I. Correctness Testing

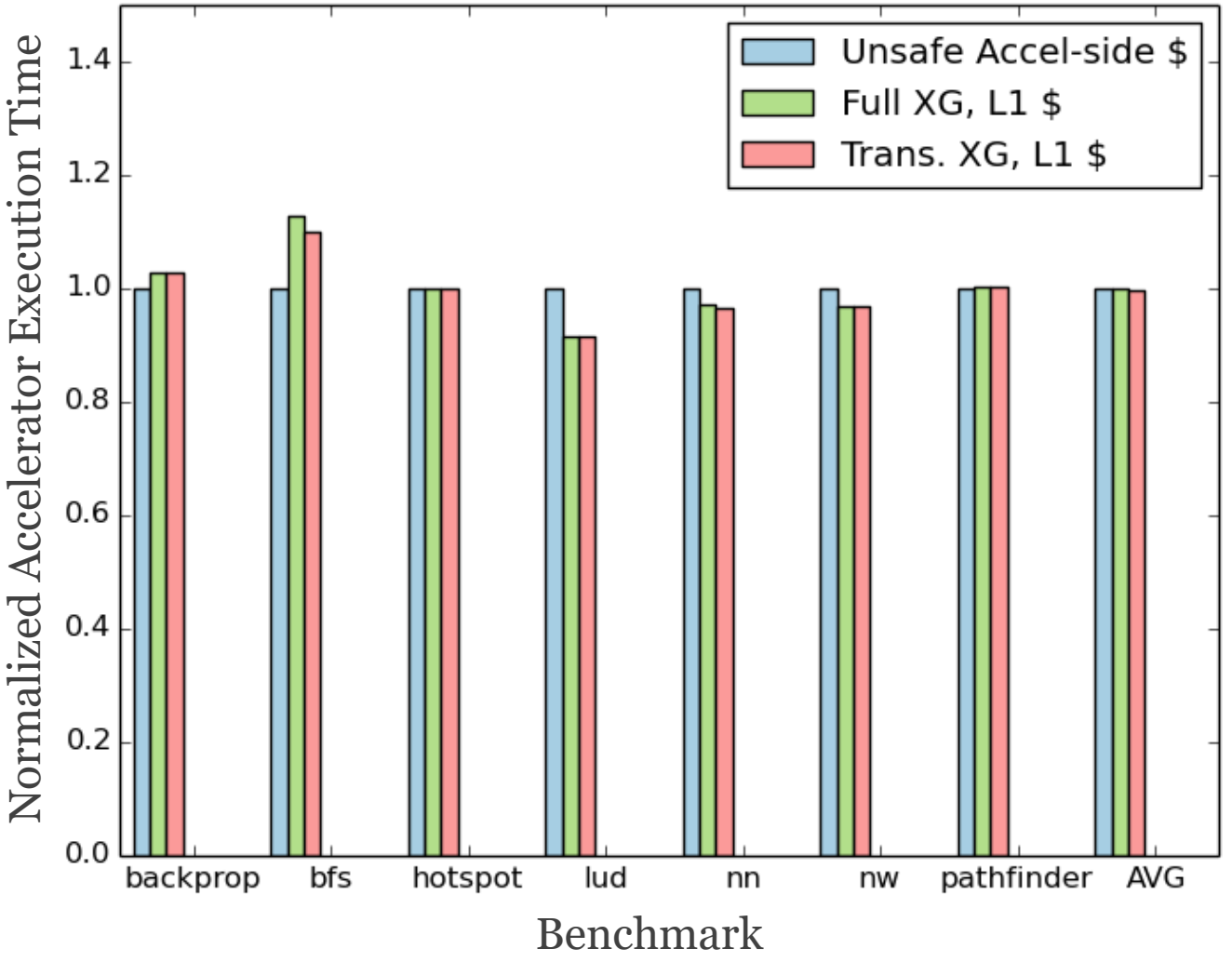
- Are coherence invariants are maintained when accelerator is acting correctly?
- How? Random tester
  - Store-Load pairs to random addresses
  - Check integrity of data
- Ran for 160 billion load/store pairs
- Local coverage: 100% states, 100% events, > 99% transitions

II. Fuzz Testing

- Is host safety maintained when accelerator misbehaves?
- How? Replace accelerator cache with evil controller
  - Generates random coherence messages to random addresses
  - Desired outcome: No deadlocks / crashes
- Ran for 7 billion load/store pairs
- Local Coverage: 100% states, 100% events, > 99% transitions

III. Performance Testing

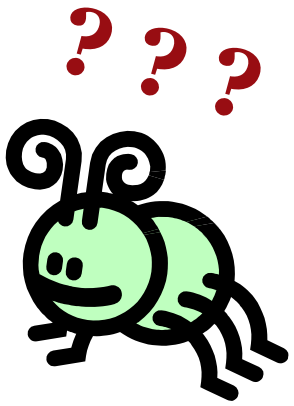
- gem5-gpu
- Rodinia workloads
- MESI Inclusive host protocol



Crossing Guard Summary

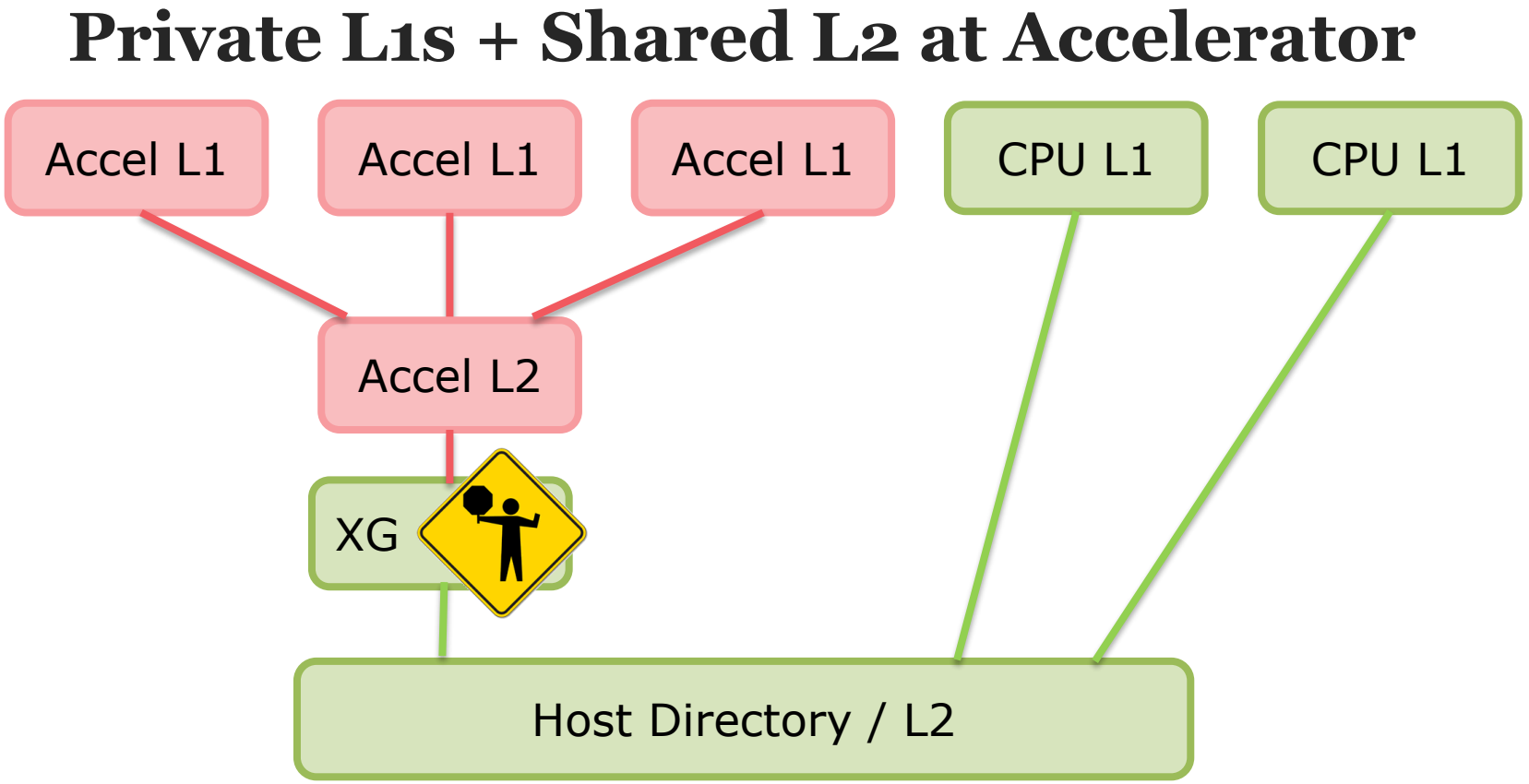
- Provides **simple, standardized interface** to ease accelerator development
- **Correctness** when accelerator is correct
- **Host safety** when accelerator is incorrect
- **Low performance overhead**

Questions?



Backup Follows

Two-Level Accelerator Protocol (1)



Two-Level Accelerator Protocol (2)

L1 Controller (M state contains dirty/clean bit)

	Load	Store	Replacement	Invalidate	DataM	DataS	Writeback Ack
M	h q	hh q	j c v l / MI	d l m / I			
S	h q	j b q / IM	l / I	f l m / I			
I	i j a q / IS	i j b q / IM		f m			
IS	z	z	z	f m / IS I		w u k xxlh n / S	
IM	z	z	z	f m	u k xxsh n / M		
MI	z	z	z	f m			k n / I
IS I	z	z	z	f m		w u xxlh k l n / I	

Two-Level Accelerator Protocol (3)

L2 Controller

(Coordinates Sharing among Accelerator L1s)

	getM	getS	putM	InvAck	Writeback	Inv	DataM	DataS	WBAck	All Acks	L2 Replacement	L2 Replacement Clean
I	a r k qreq1 / IM	b r m qreq1 / IS				d qreqBC						
S	t k i p rrr qreq1 / SM	m fr rrr uu h qreq1				t w i s qreqBC / SI						t w i s / SR
MO	e r g rrr uu h qreq1 / M	m fr rrr uu h qreq1 / MOS				t w c s u qreqBC / I					t w c wm s / MRI	
M	h k qreq1 / MM	h m qreq1 / MMOS	n1p yo j qput1 / MO			t w h s qreqBC / MI					t w h s / MR	
MOS	t w k i p rrr qreq1 / MOSM	m fr uu h rrr qreq1				t w i s qreqBC / MOSI					t w i s / MOSR	t w i s / ER
IM	zz 1r	zz 1r				d qreqBC	n g rrr uu m qrspBC / M				z	z
IS	zz 1r	m qreq1				d qreqBC	n f uu m rrr qrspBC / MOS	n f uu m rrr qrspBC / S			z	z
SM	zz 1r	zz 1r		o p qrsp1		zz BCr				y u a qt / IM	z	z
SI	z	z		o p qrsp1						d u qt / I		
SR	z	z		o p qrsp1		qreqBC / SI				c ws qt / SRI		
SRI	z	z				d qreqBC			u qrspBC / I			
MR	z	z	n1 tp j qput1 / MRi	hr qrsp1	n t yo c wm qrsp1 / MRI	qreqBC / MI						
MRI	z	z				d qreqBC			u qrspBC / I			
MI	z	z	n1 tp j qput1 / MIi	hr qrsp1	n t yo c u qrsp1 / I							
MOSI	z	z		o p qrsp1						c u qt / I		
MOSR	z	z		o p qrsp1		qreqBC / MOSI				c wm qt / MRI		
ER	z	z		o p qrsp1		qreqBC / MOSI				c we qt / MRI		
MM	zz 1r	zz 1r	n1p j qput1 / MMi	hr qrsp1	nw g uu m qrsp1 / M	zz BCr					z	z
MMOS	zz 1r	zz 1r	n1p j qput1 / MMOSi	hr qrsp1	nw f uu m yo qrsp1 / MOS	zz BCr					z	z
MOSM	zz 1r	zz 1r		o p qrsp1		zz BCr				g rrr y u uu m qt / M	z	z
MIi	z	z		yo c u qrsp1 / I								
MRi	z	z		yo c wm qrsp1 / MRI		qreqBC / MIi						
MMi	zz 1r	zz 1r		g uu m qrsp1 / M		zz BCr					z	z
MMOSi	zz 1r	zz 1r		f uu m yo qrsp1 / MOS		zz BCr					z	z

Crossing Guard Invariants

Crossing Guard Guarantees to Host:

1. Accelerator **requests** must be correct
   a) Consistent with block stable state at accelerator
   b) Consistent with block transient state at accelerator

2. Accelerator **responses** must be correct
   a) Consistent with block stable state at accelerator
   b) Consistent with block transient state at accelerator
   c) Received within a reasonable time

(+ Border Control Protections!)

Crossing Guard Variants

- Full State Crossing Guard
  - Inclusive directory of accelerator state
  - + Places few restrictions on host protocol
  - + Can hide all errors
  - - Requires tag + metadata storage for all blocks
- Transactional Crossing Guard
  - Stores only data for in-flight transactions
  - + Small storage
  - + Provides most safety properties
  - - Requires some host tolerance

Single-Level Cache

States	Accelerator Events			XG Requests	XG Responses			
	Load	Store	Replacement	Invalidate	DataM	DataE	DataS	WB Ack
M	hit	hit	issue PutM / B	send Dirty WB / I	-	-	-	-
E	hit	hit / M	issue PutE / B	send Clean WB / I	-	-	-	-
S	hit	issue GetM / B	issue PutS / B	send InvAck / I	-	-	-	-
I	issue GetS / B	issue GetM / B	-	send InvAck	-	-	-	-
B	*stall*	*stall*	*stall*	send InvAck	/ M	/ E	/ S	/ I

Simulation Parameters

CPU		
CPU Cores		1
CPU Frequency		3 GHz
Host Caches		
	Hammer-like	**MESI Inclusive**
L1I & L1D	32kB each	32kB each
L2	128kB private	512kB shared w/ GPGPU

Table 4: CPU simulation configuration details.

GPGPU			
Cores			4
GPU Frequency			700 MHz
GPGPU Caches (Hammer-like)			
	Accel-side	**Host-side / 1-level**	**2-level**
L1	16kB I and D	160kB	32kB
L2	128kB private	-	512kB shared
GPGPU Caches (MESI Inclusive)			
	Accel-side	**Host-side / 1-level**	**2-Level**
L1	32kB I and D	64kB	16kB
L2	-	-	192kB shared

Cache-to-Cache Latency	
Accelerator L1 to L2	10 cycles
Accelerator L2 to XG	200 cycles
XG to Directory/Shared L2	10 cycles
Accelerator to Host-side Cache	210 cycles

Table 5: GPGPU simulation configuration details.

Time Spent Simulating (Random)

Configuration	Time
XG Full + Hammer + 1 Level	5.28 years
XG Full + Hamer + 2 Level	2.51 years
XG Full + MESI Inc + 1 Level	133 days
XG Full + MESI Inc + 2 Level	223 days
XG Trans. + Hammer + 1 Level	3.17 years
XG Trans. + Hammer + 2 Level	1.38 years
XG Trans + Inc + 1 Level	90 days
XG Trans + Inc + 2 Level	103 days
TOTAL	**13.9 years**

Full Coverage %s (Random)

Full State XG	Single-level	Two-level
Hammer-like	99	99.8
MESI Inclusive	100	99.4
Transactional XG	**Single-level**	**Two-level**
Hammer-like	99.3	99.5
MESI Inclusive	100	99.7

Time Spent Simulating (Fuzz)

Configuration	Time
XG Full + Hammer-like	1.62 years
XG Full + MESI Inclusive	287 days
XG Transactional + Hammer-like	5.3 years
XG Transactional + MESI Inclusive	41 days
Total	**7.82 years**

Full Coverage %s (Fuzz)

Full State Crossing Guard	Fuzz Tester
Hammer-like	99.3
MESI Inclusive	99.7
Transactional Crossing Guard	**Fuzz Tester**
Hammer-like	99.7
MESI Inclusive	100

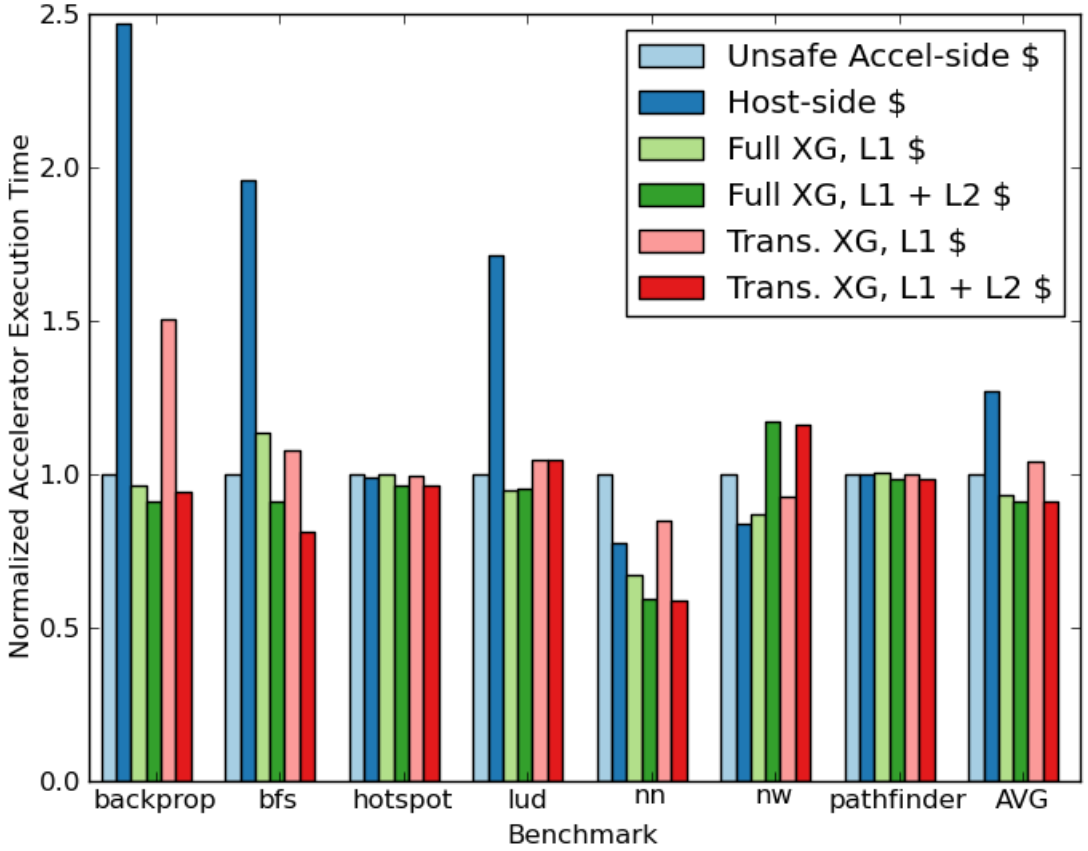
PutS Accelerator Messages

- Why?
  - Some host protocols use them
  - Simplify management of Full State Crossing Guard
  - Cannot implement Transactional Crossing Guard + host protocol with PutS without them
- Bandwidth Impact
  - Carry no data
  - Only between accelerator cache → Crossing Guard, not host system
  - ~1-4% of that bandwidth in experiments.
  - Could be reduced by setting a flag at Crossing Guard.

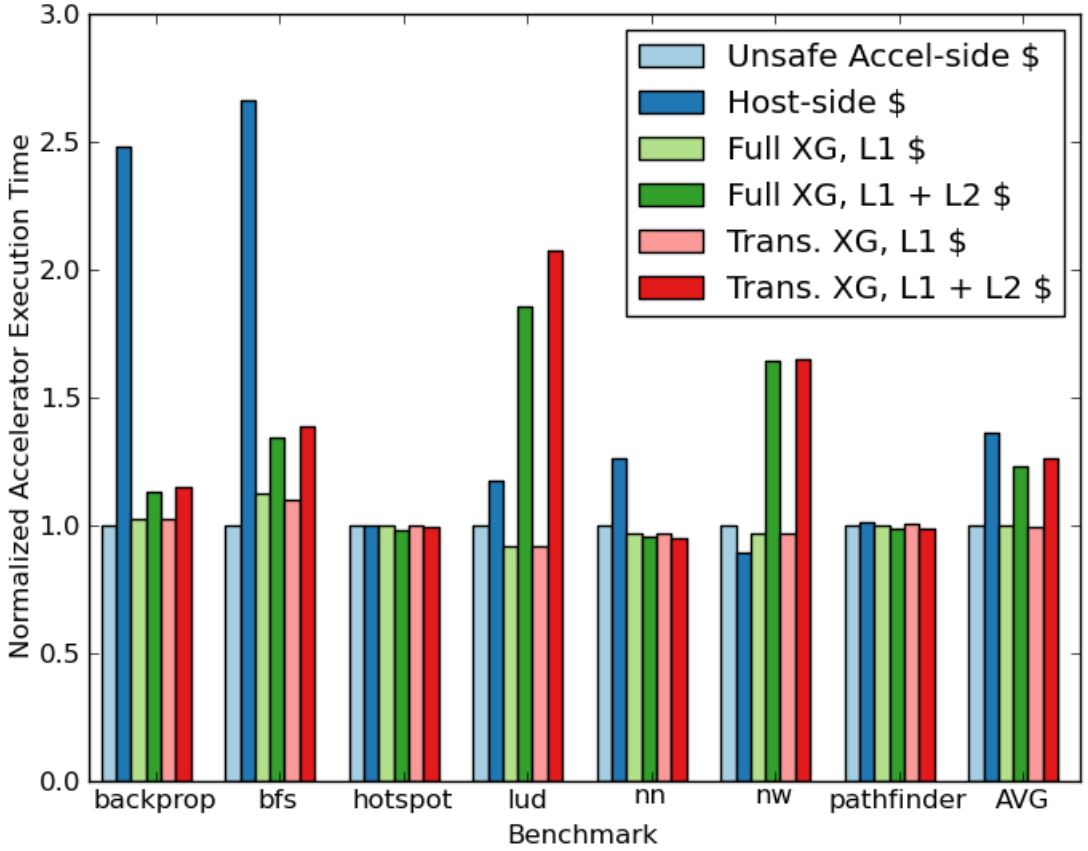
Why not Model Checking?

- Model checking is useful! Industrial implementation of Crossing Guard would use.
- Academic tools have limitations ☹
  - Benefit from symmetry, but Crossing Guard system asymmetric
  - May only work with one block in system
  - Substantial implementation overhead
- This work was a proof of concept
  - Random / Fuzz testing not perfect, but results suggestive.
  - Even models can have mistakes!

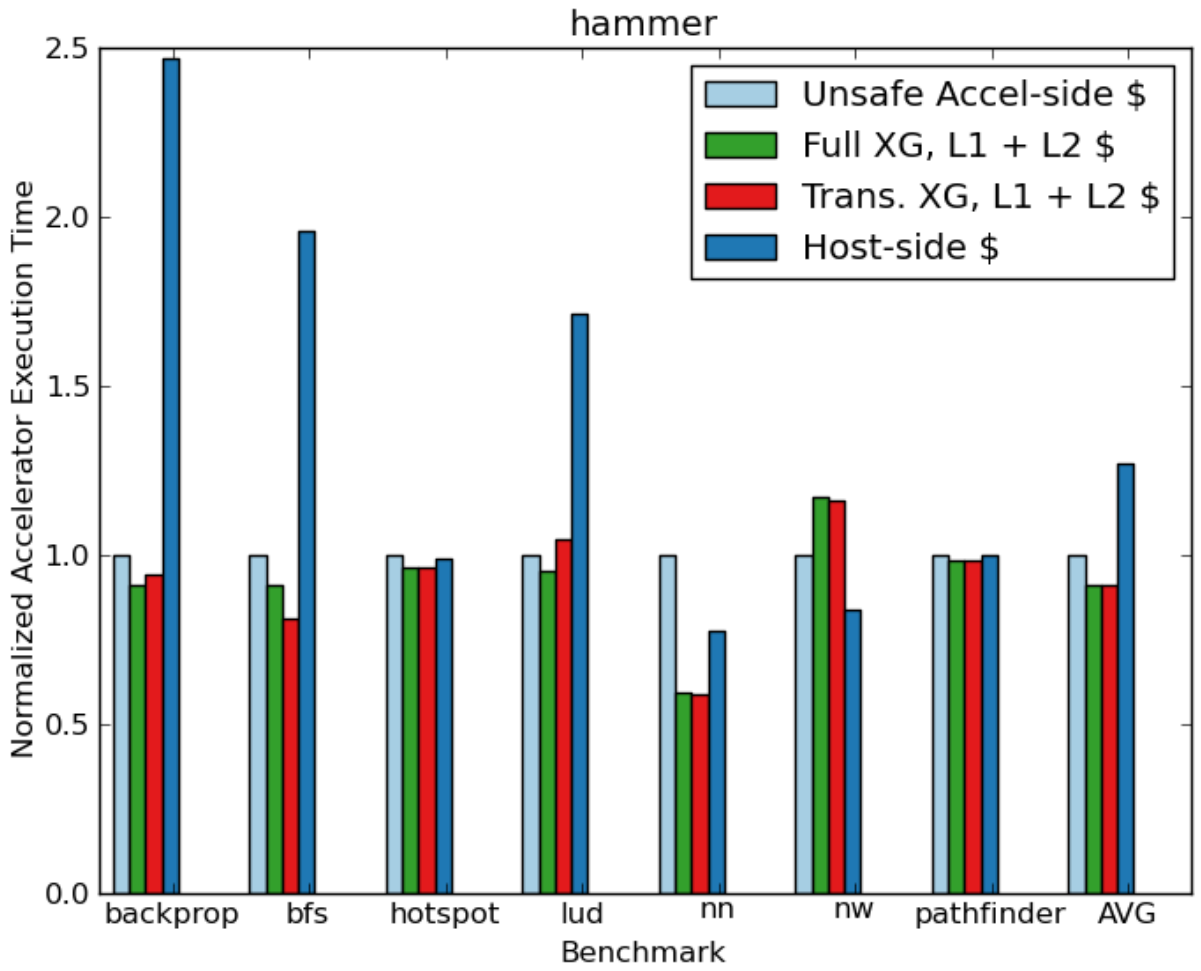
Performance: Hammer-like



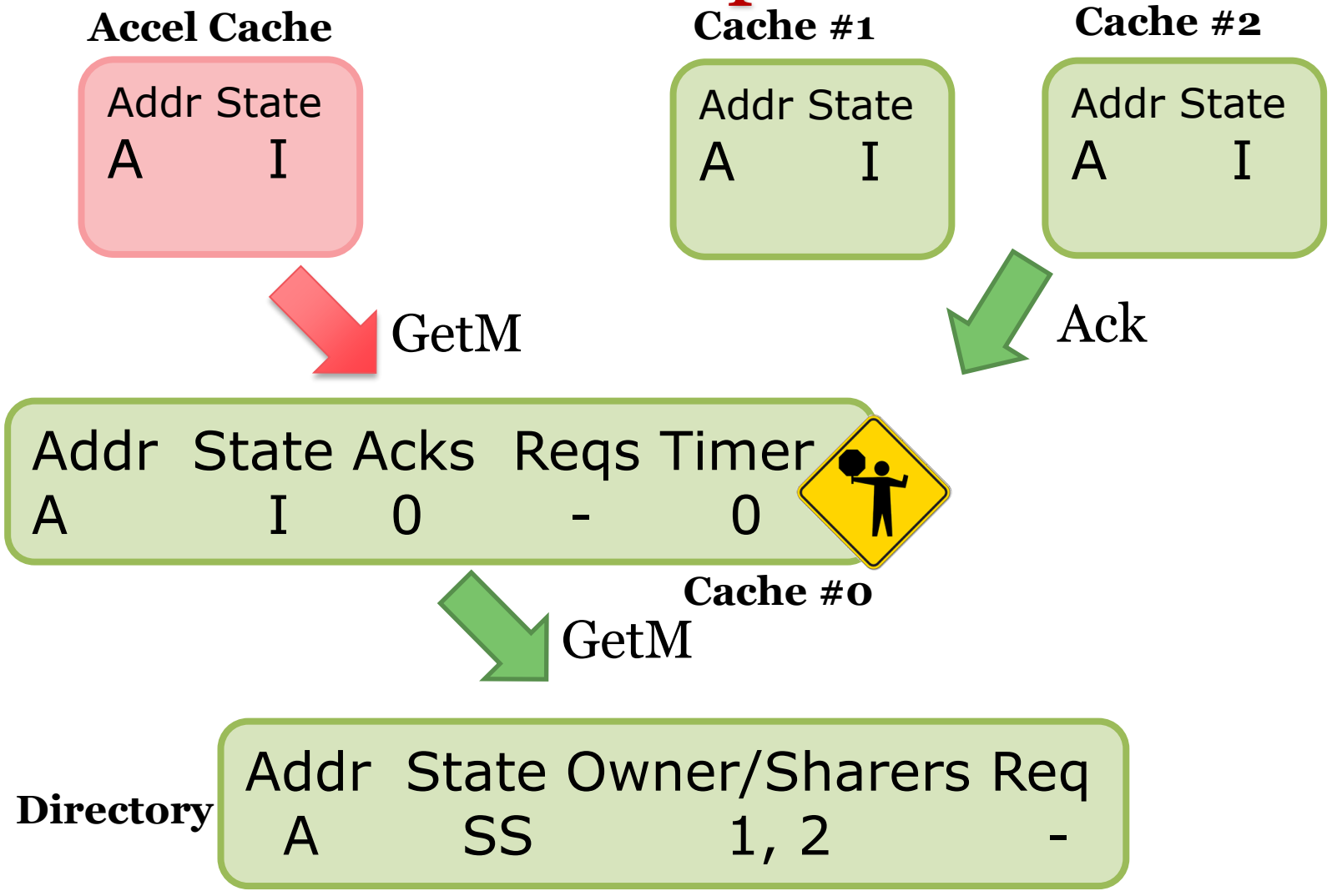
Performance: MESI Inclusive



Performance (Hammer-like)

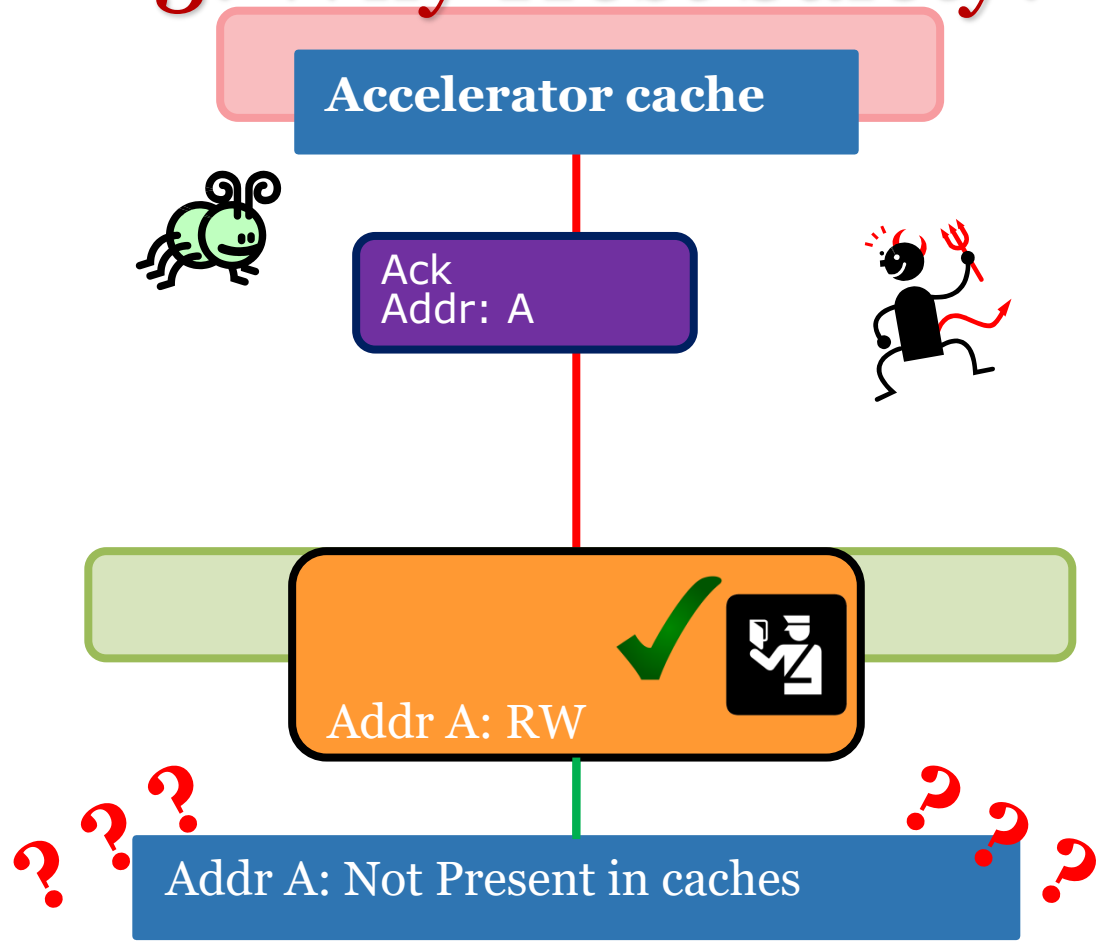


Template

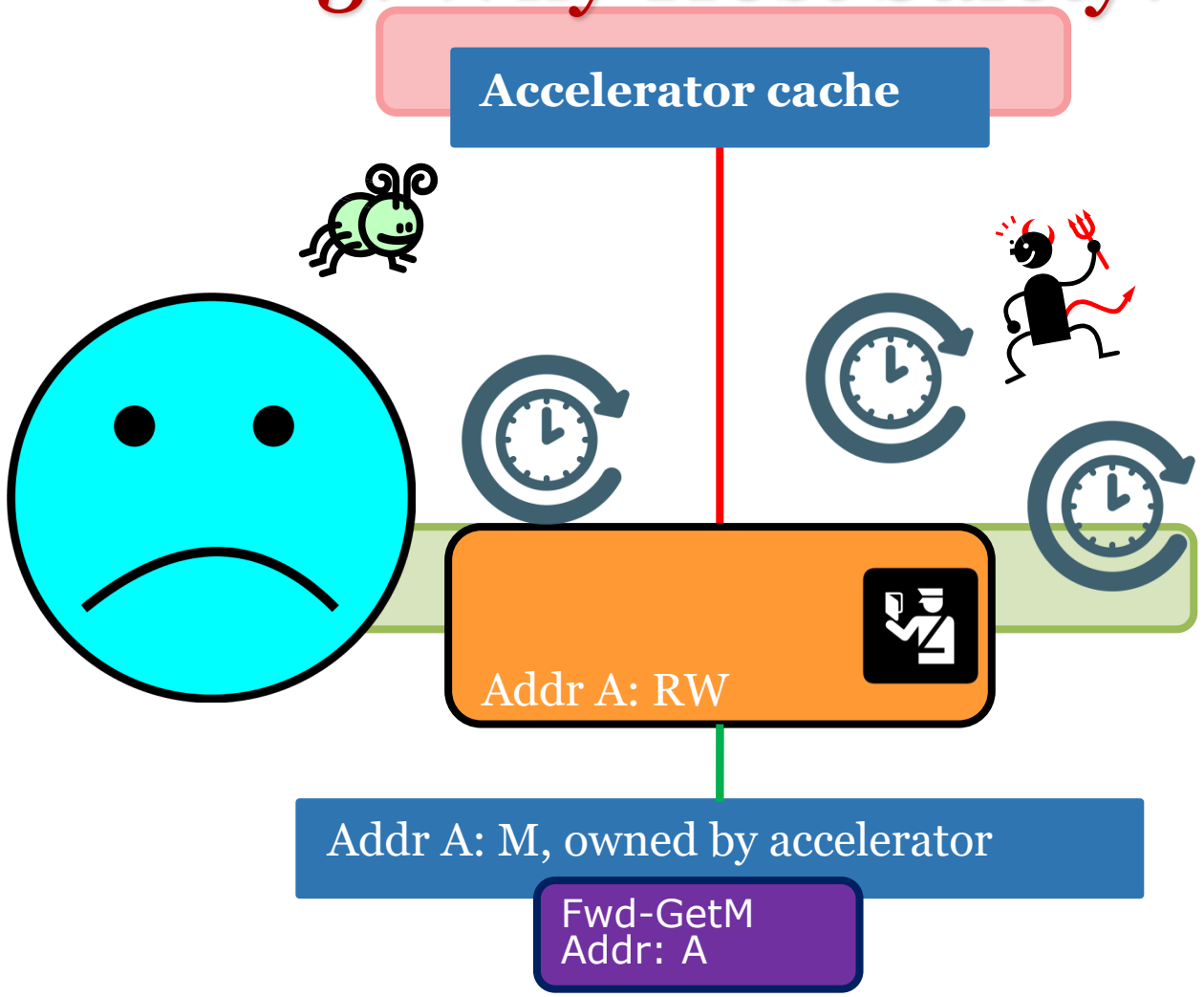

Accel Cache
Addr State
A I
Cache #1
Addr State
A I
Cache #2
Addr State
A I
GetM
Ack
Addr State Acks Reqs Timer
A I 0 - 0
Cache #0
GetM
Directory
Addr State Owner/Sharers Req
A SS 1, 2 -

Old Slides

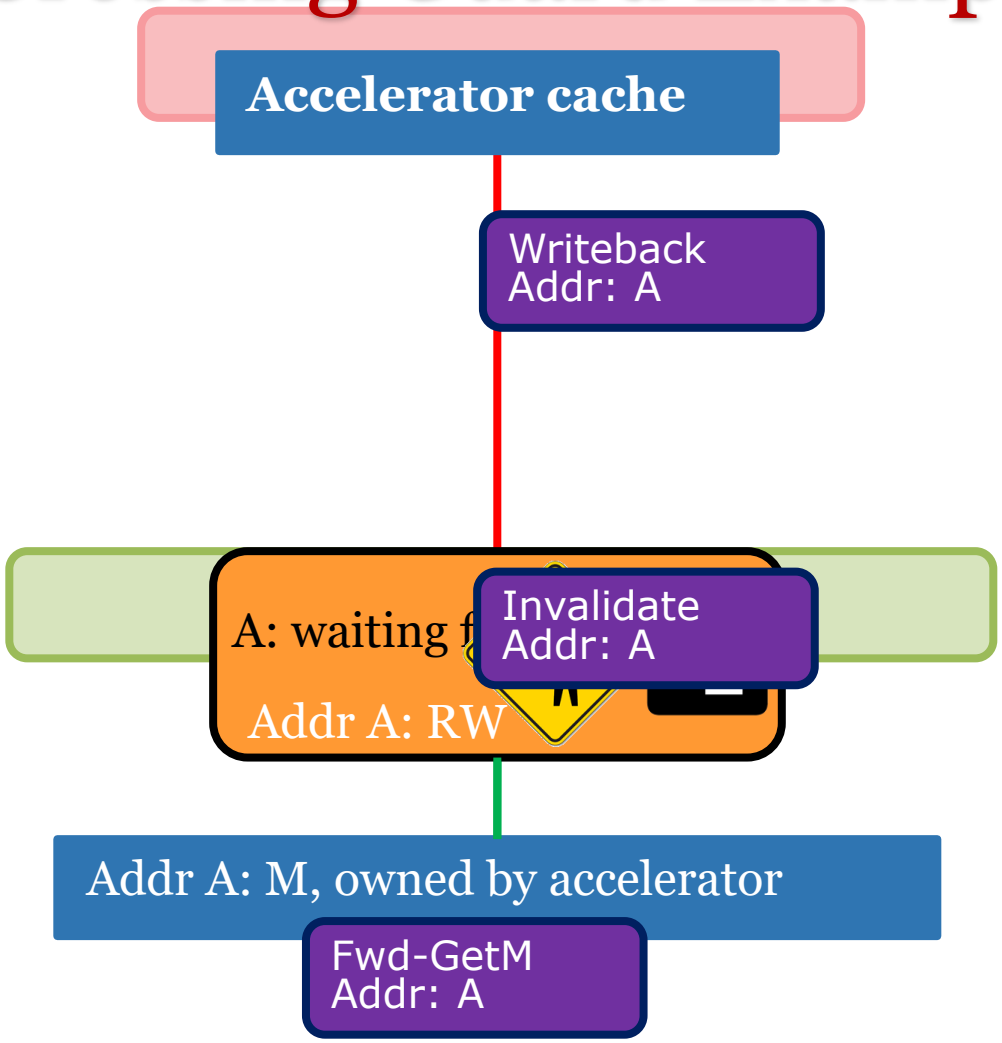
3. Why Host Safety?



3. Why Host Safety?



Crossing Guard Example



Crossing Guard Example