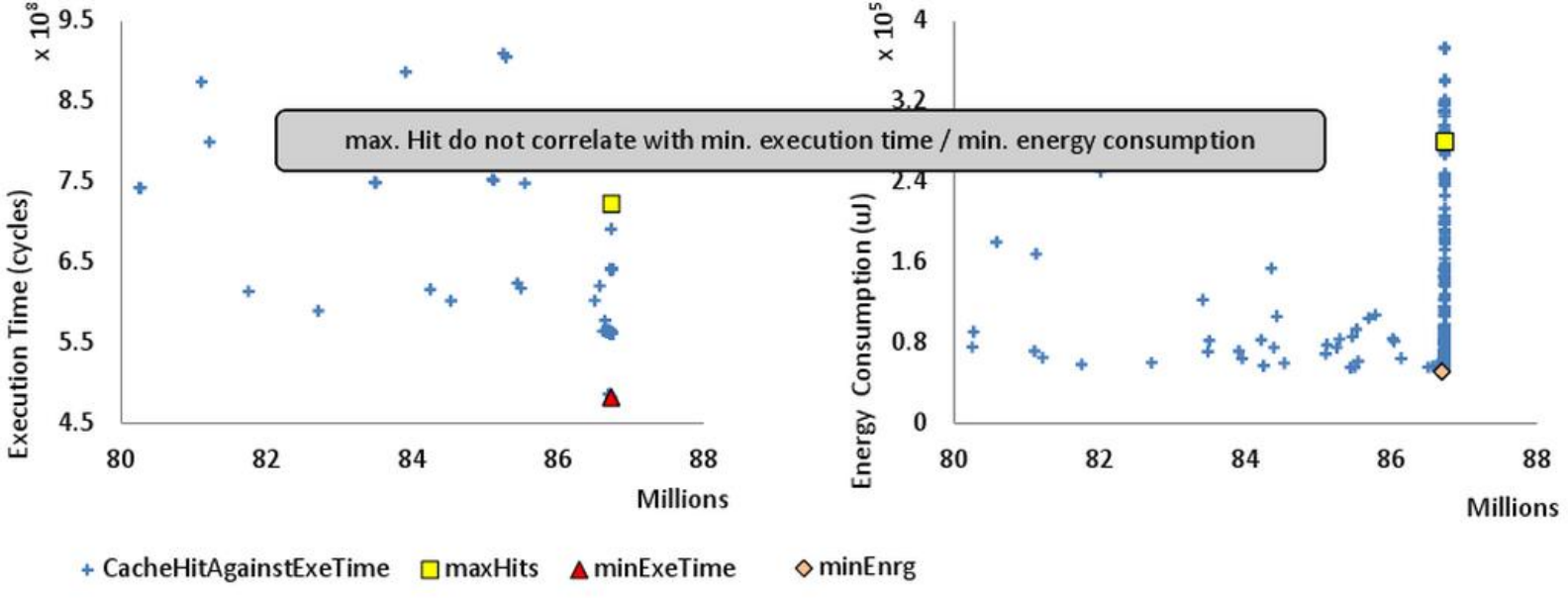
A Scorchingly Fast FPGA-Based Precise L1 LRU Cache Simulator

Josef Schneider, Jorgen Peddersen, Sri Parameswaran

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

Why simulate a cache?

To find optimal configuration for a specific application.



Why simulate a cache?

- For many modern processors, the cache configuration can be chosen at design time (ARM, Xtensa, NIOS II, etc.)
- Optimal cache configuration can be found through cache simulation.
- Current cache simulator solutions require lengthy simulation times.

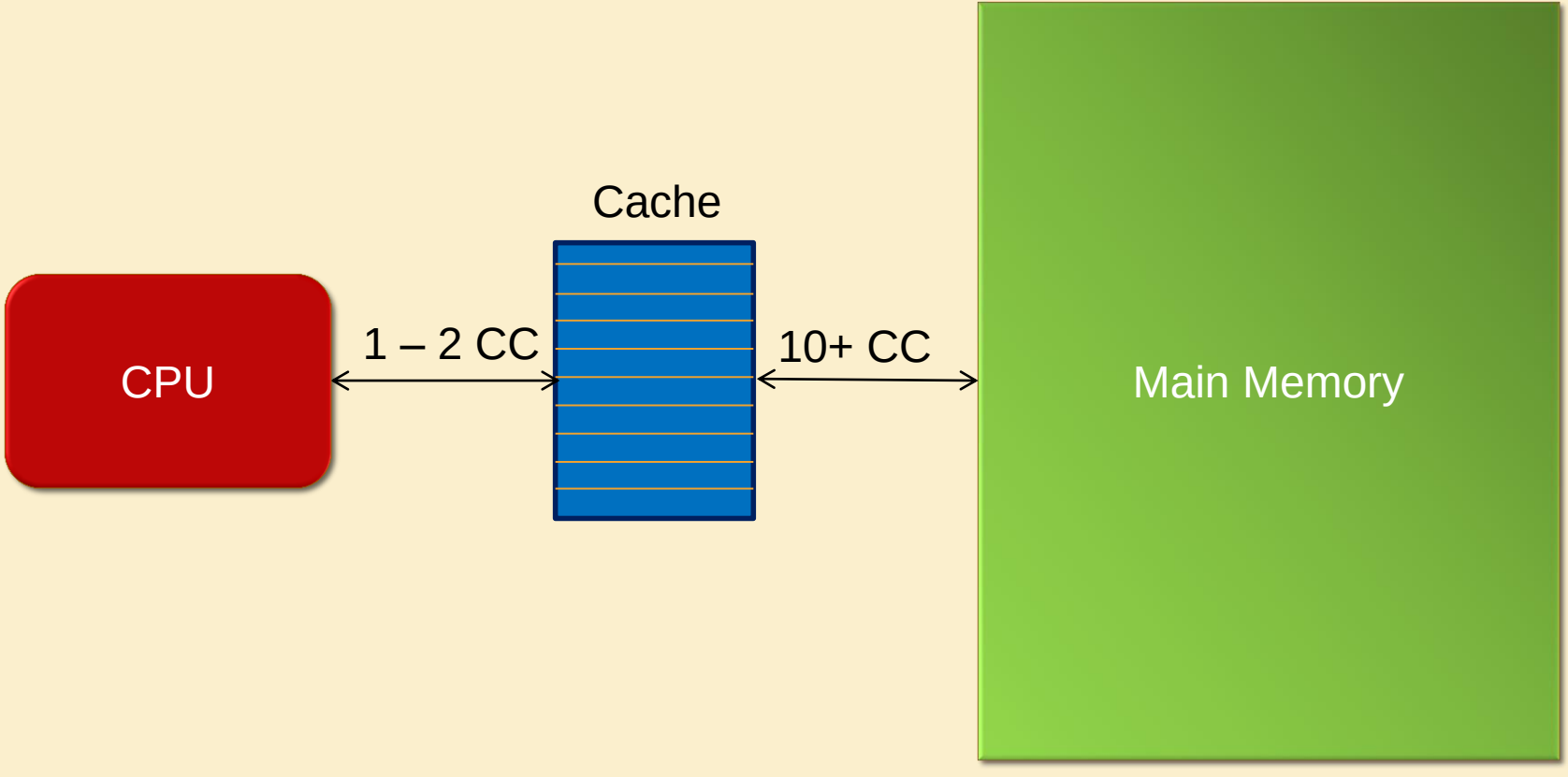
Contributions

- FPGA-based Multiple Cache Simulator exhibiting:
  - 100MHz memory trace consumption rate.
  - Simulates 44 caches concurrently.
  - Ability to implement in-system for real-time cache simulation.

Presentation Contents

- What is a cache?
- Existing simulator shortcomings
- Cache simulator design
- Performance comparison
- Future work

What is a cache?



The Accountant's Desk Analogy

A cache is like a desk

Picking up a d
the desk takes

When you get a document from the filing cabinet, you are likely to reuse it, so you put it on the desk for faster access

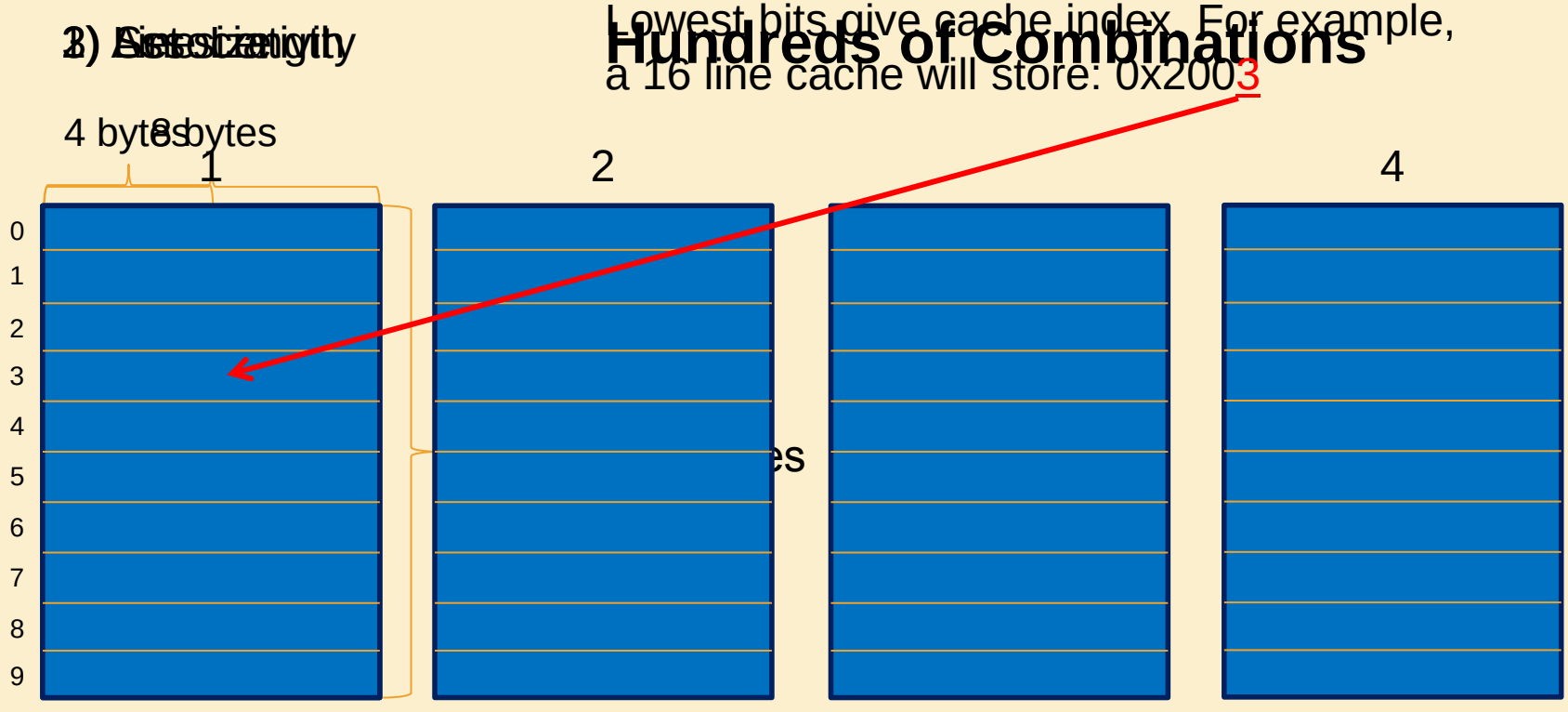
Getting it from the filing cabinet takes a lot longer

Desk Considerations

The bigger the desk is, the more documents it can hold, but:

- It will cost more
- It will take longer to find this document on the desk

Three Factors, Many Combinations



Notable Existing LRU Cache Sims

- DineroIV (J. Edler and M. Hill) for LRU, FIFO and Random cache replacement policies.
- Cheetah (Sugumar et al.).
- “Finding optimal L1 cache configuration for embedded systems” (A. Janapsatya et al.) for LRU policy.
- CRCB Algorithm (Tojo et al.).
- SuSeSim (M. Haque et al.).

Existing Cache Sim. Shortcomings

- All are implemented in software and require lengthy computations for advanced applications

  (Using Cheetah: simulating an MPEG2 encoder for 24 video frames took over 65 minutes for 44 cache configurations.)

- Simulations are *usually* based on static traces which take a very long time to generate

  (On the Xtensa processor simulator: creating the trace for the MPEG2 encoding application took over 70 hours.)

- Application inputs are hard-coded

  (Simulated inputs are contrived and may not represent realistic application input.)

Our Solution: Cache Sim. In Hardware

Our hardware-based cache simulator can be configured on an FPGA. By exploiting two LRU cache inclusion properties, many caches can be simulated concurrently and efficiently.

In software, inclusion properties allow for faster simulation (fewer searches). In hardware, the same properties are used to minimise hardware utilisation.

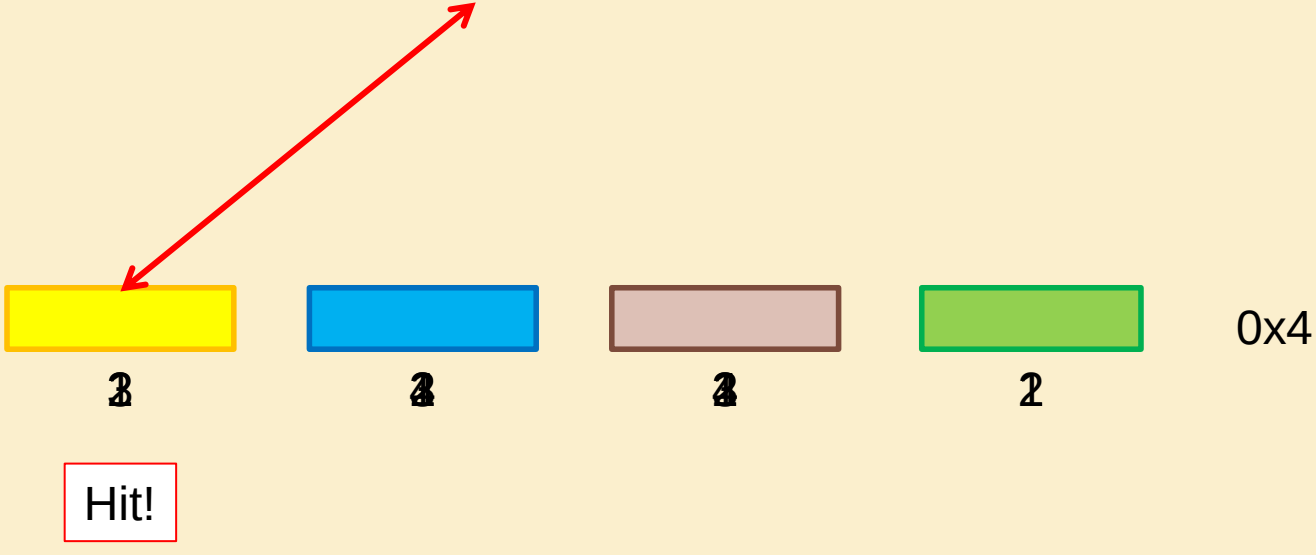
Inclusion Property 1

An LRU cache of set size s, line length l and associativity a is a subset of an LRU cache of set size s, line length l and associativity larger than a.

(Mattson et al. 1970)

LRU Replacement Example

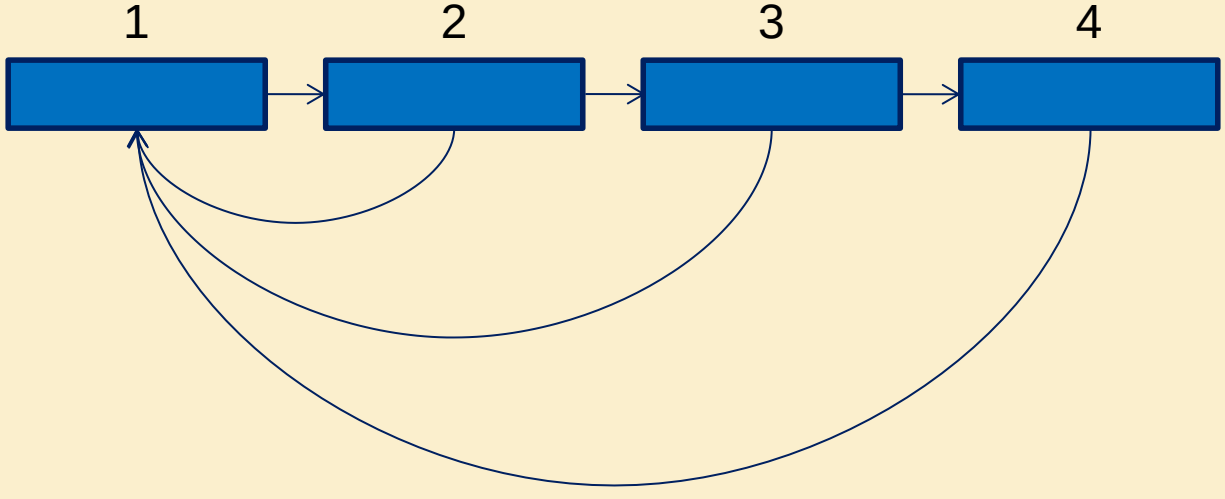
Access Address: 0x4



Hit!

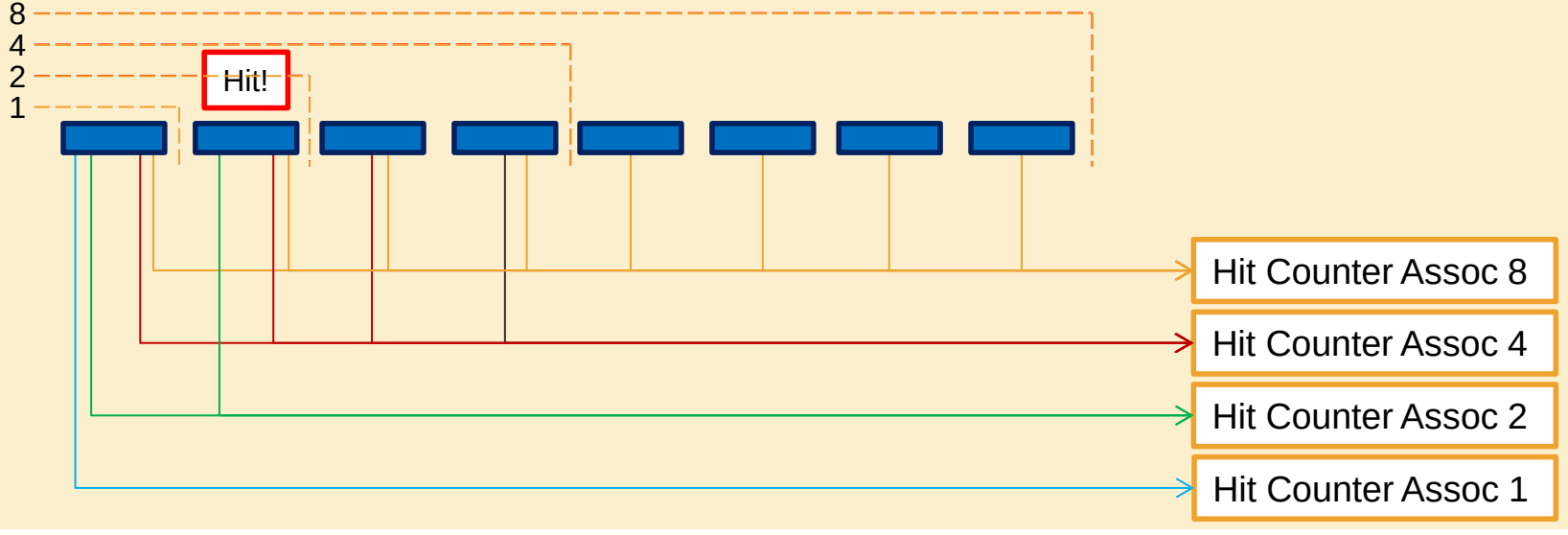
LRU Replacement Example

The same behaviour is exhibited by an LRU shift register.

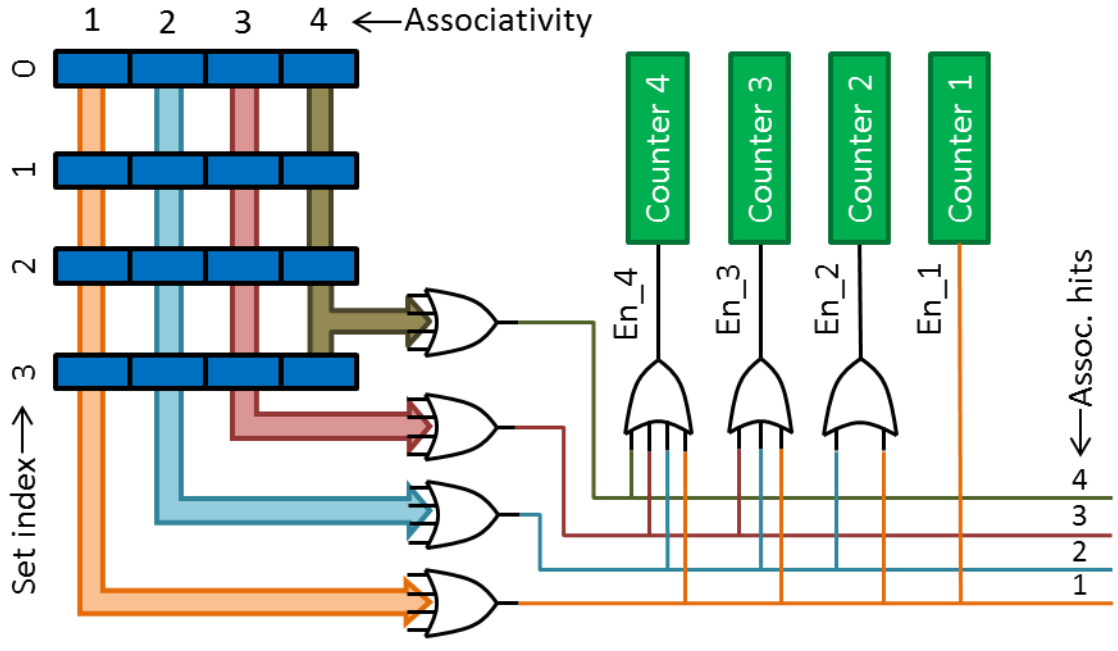


Making use of Inclusion Property 1

Thanks to Inclusion Property 1, cache sets of associativity a and smaller can be simulated using the same hardware.



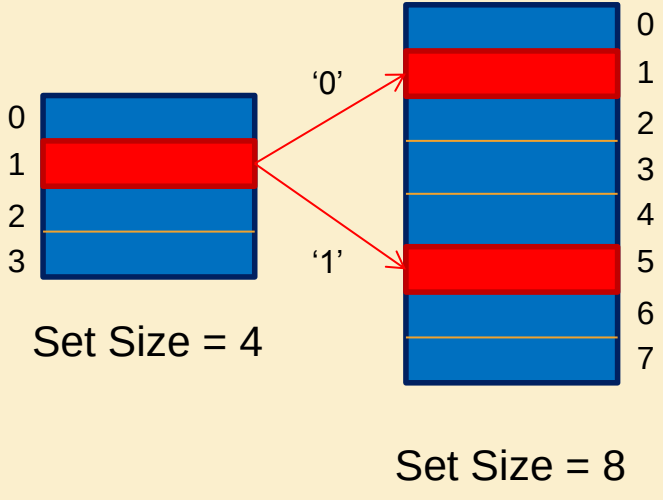
Top Level Cache Sim. Design



Enables the simulation of caches of set size 4 and associativities 4 and smaller.

Inclusion Property 2

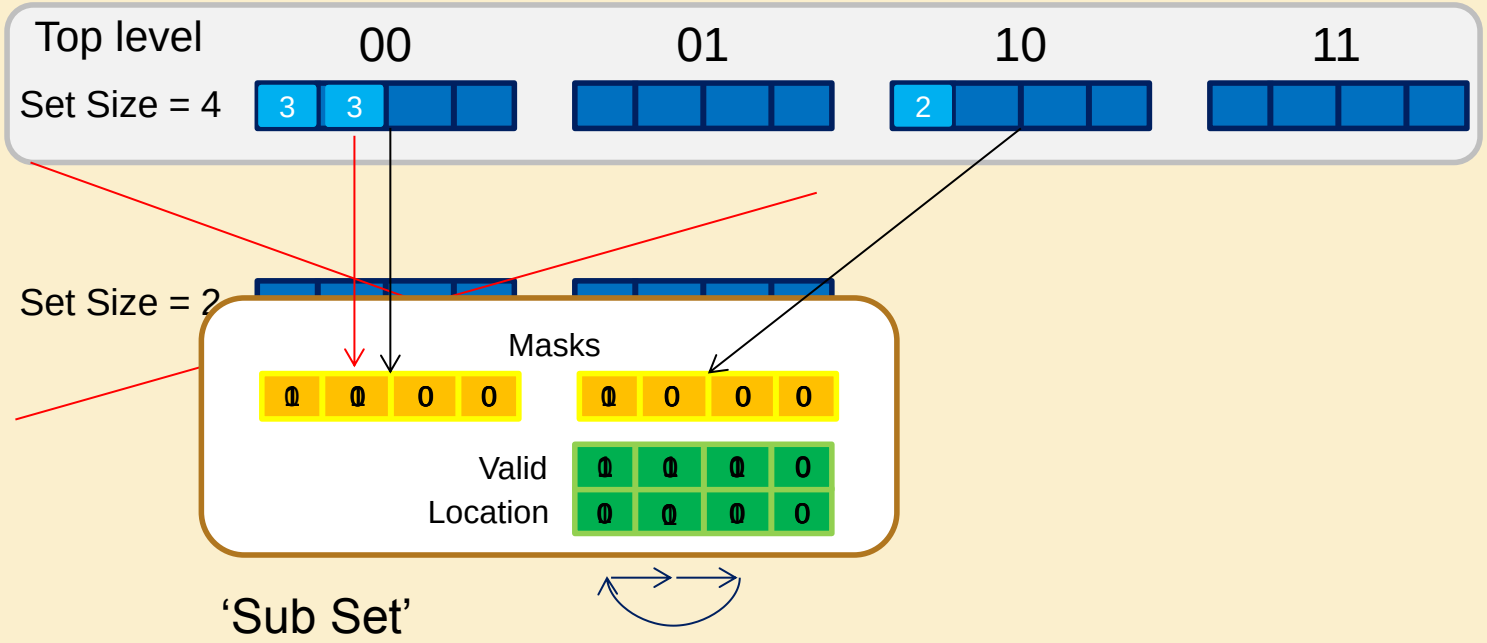
A cache of line length l, associativity a and set size s is always a subset of a cache of line length l, associativity a and set size larger than s. (Mattson et al. 1970)



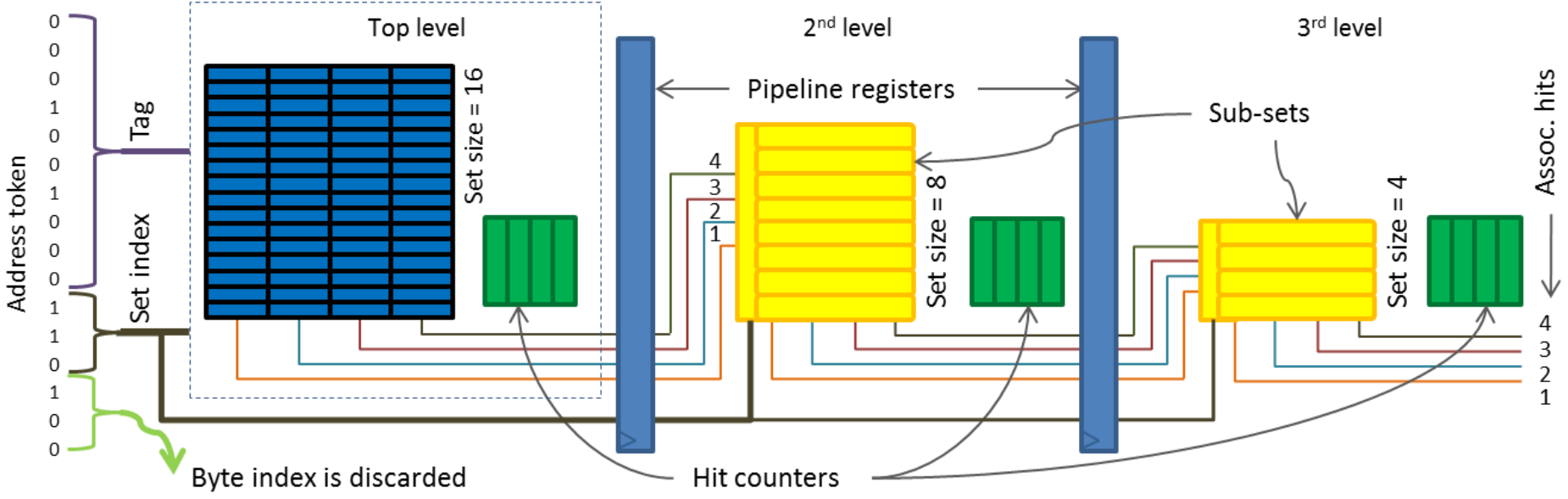
Making use of Inclusion Property 2

This can be used to our advantage

Addr:
1)0b101000
2)0b101010
3)0b011000
4)0b101000



Combining Top Level with Sub Sets



Enables the simulation of caches of set sizes 16 to 4, and associativities 4 and smaller. Total caches simulated: 12

Simulating Multiple Line Sizes

Different line lengths = shift line index in address

Set index for a set of size 16 lines, line length = 1 byte

Address = 0b00010110100

Set index for a set of size 16 lines, line length = 2 bytes

Disadvantage: need to re-run simulation for each different line size.

Implementation in Altera Stratix IV FPGA

addr width	max assoc.	Set count max					
		32		256		1024	
		LUTs	Registers	LUTs	Registers	LUTs	Registers
	4	4932	3702	25085	15326	79431	46077
	8	10529	7163	56859	29935		
17	16	21434	14084				
	4	5517	4342	26541	20446	87626	66557
	8	11644	8443	60321	40175		
22	16	23672	16644				
	4	5997	4982	31990	25566	105037	87037
	8	12655	9723	70216	50415		
27	16	26203	19204				
	4	6410	5622	35313	30686	126235	107517
	8	13500	11003	77732	60655		
32	16	27829	21764				

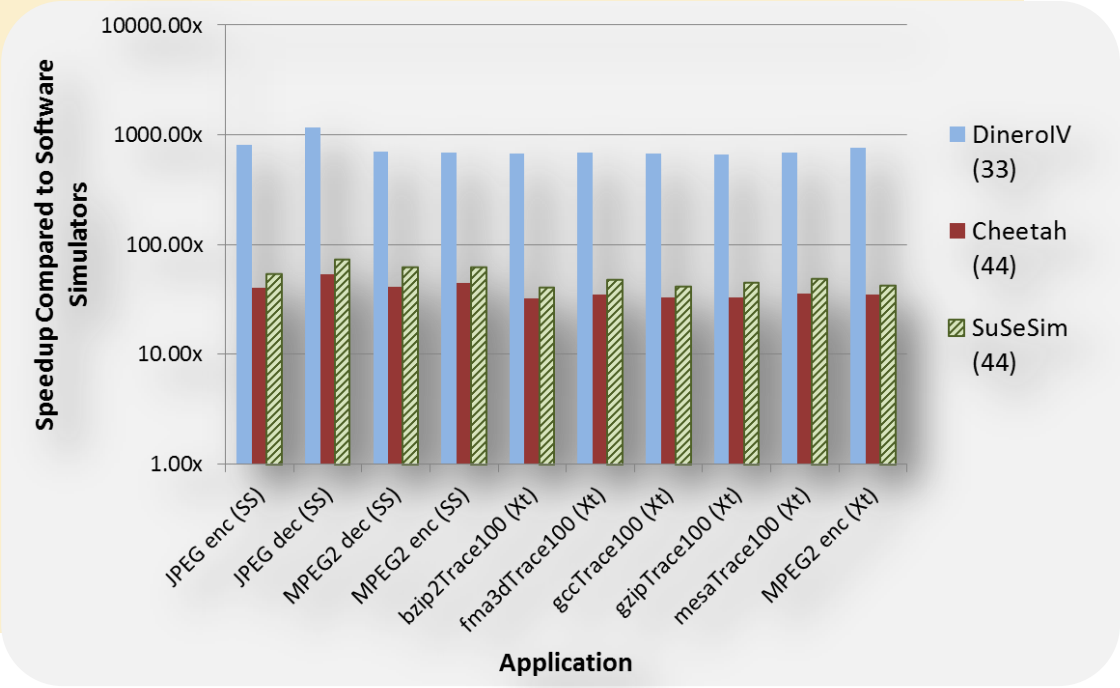
Simulator Characteristics on Altera Stratix IV FPGA

- Capable of simulating 308 cache configurations, of which it can simulate 44 concurrently.
- Address consumption rate of 100MHz.
- Fully written in VHDL and easily parameterisable.
- Two buses – one for trace input, another for control.

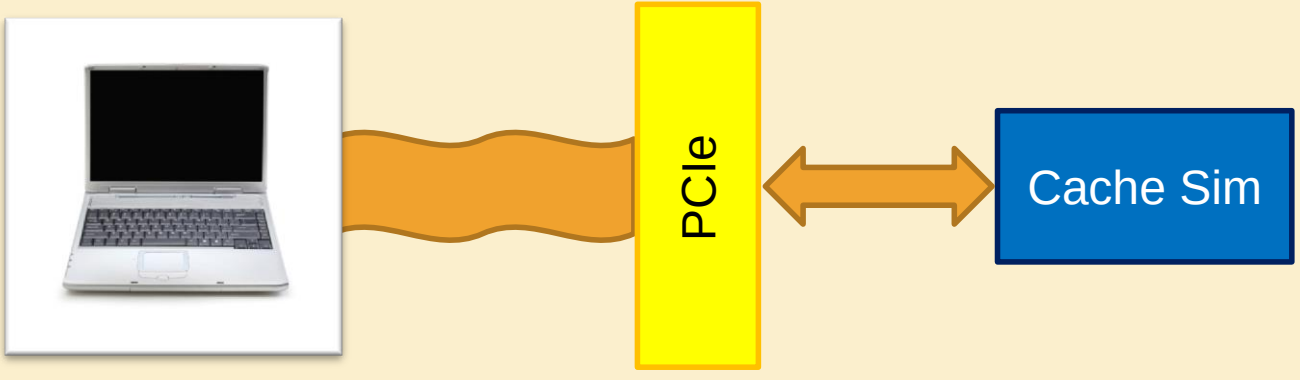
Performance Comparison

At the time of writing the paper, no direct performance comparison was possible. Simulator throughput was compared with:

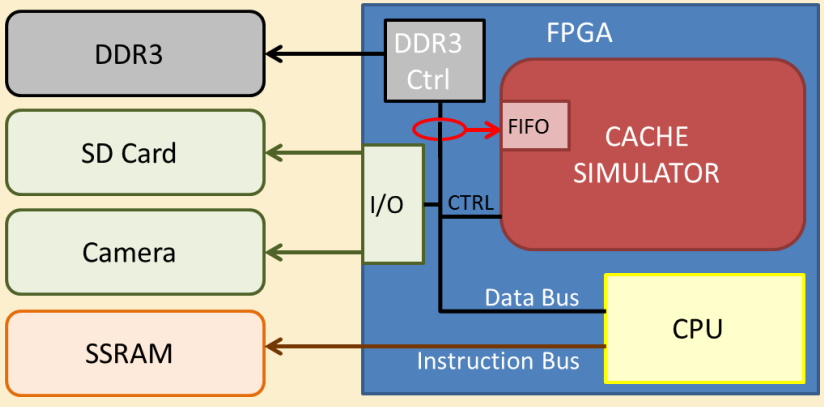
- DineroIV
- Cheetah
- SuSeSim

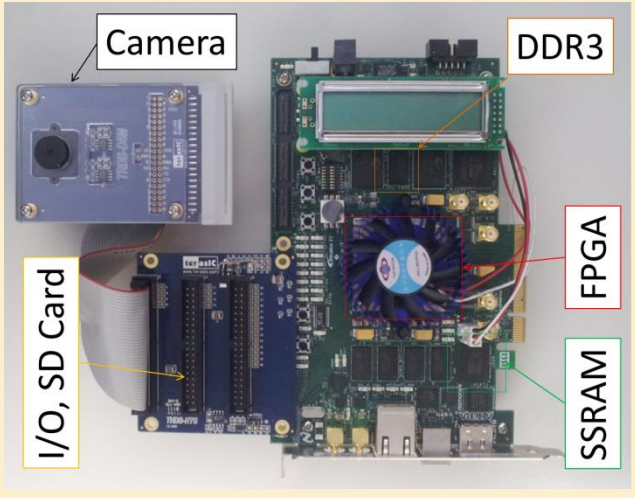


Example Use: Static Trace Simulation



Example Use: In-System Implementation

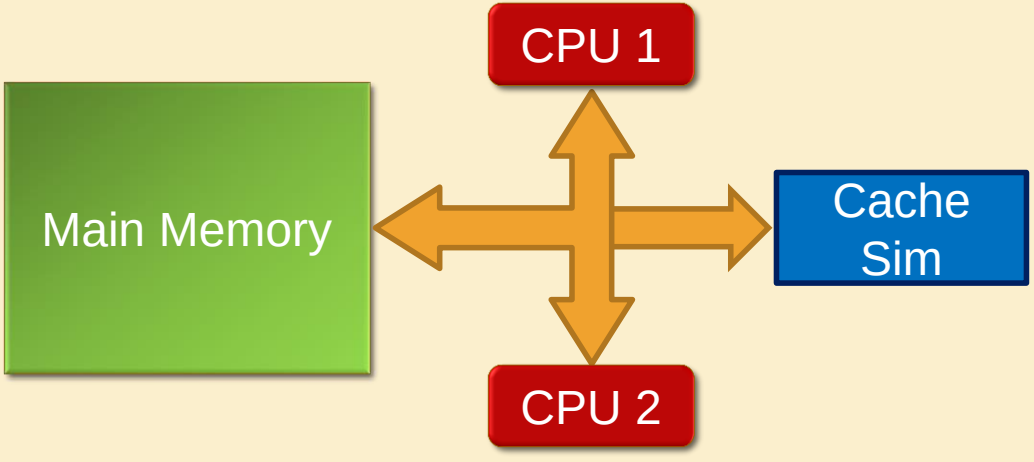




Directly connected to soft-core processor

- Real time, real-input cache simulation
- Executes as fast as the system

Future Work



Multi-processor cache simulator.

Conclusions

- First hardware-based multiple cache simulator.
- Reduces hardware usage by utilising cache inclusion properties.
- Allows for real-time, in-system cache simulation.
- Throughput up to 53x faster than one of the fastest software-based cache simulators.

Thank You