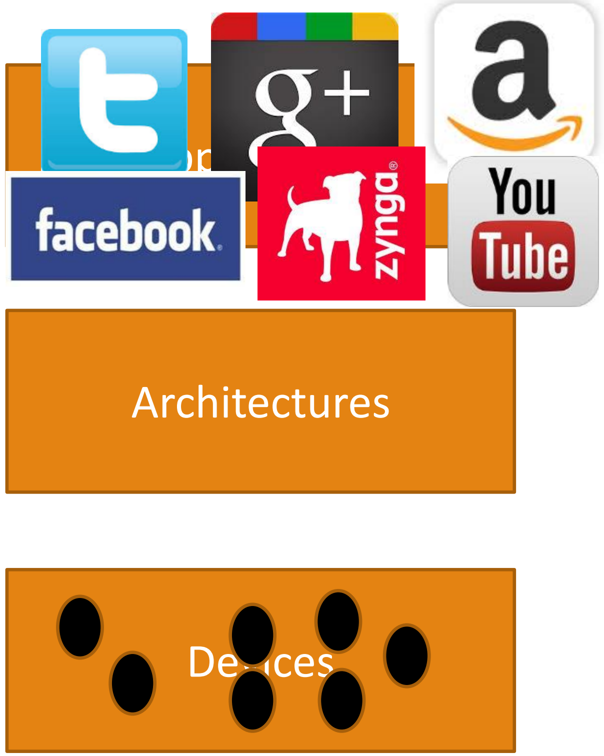
An FPGA-based In-line Accelerator for Memcached

MAYSAM LAVASANI, HARI ANGEPAT, AND DEREK CHIOU

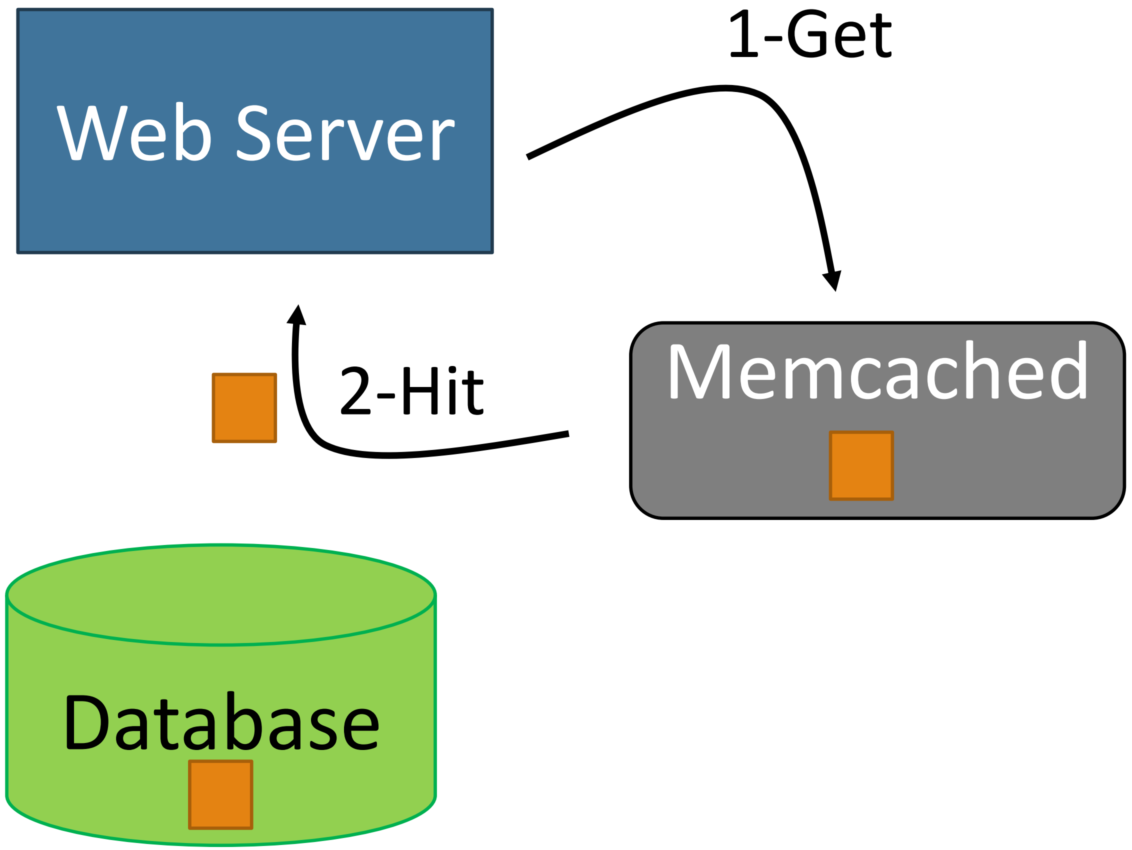
THE UNIVERSITY OF TEXAS AT AUSTIN

Challenges for Server Processors

- Workload changes
  - Social networking
  - Cloud applications
  - Big data applications
- Power wall
  - Dark silicon
    - More and more cores insufficient

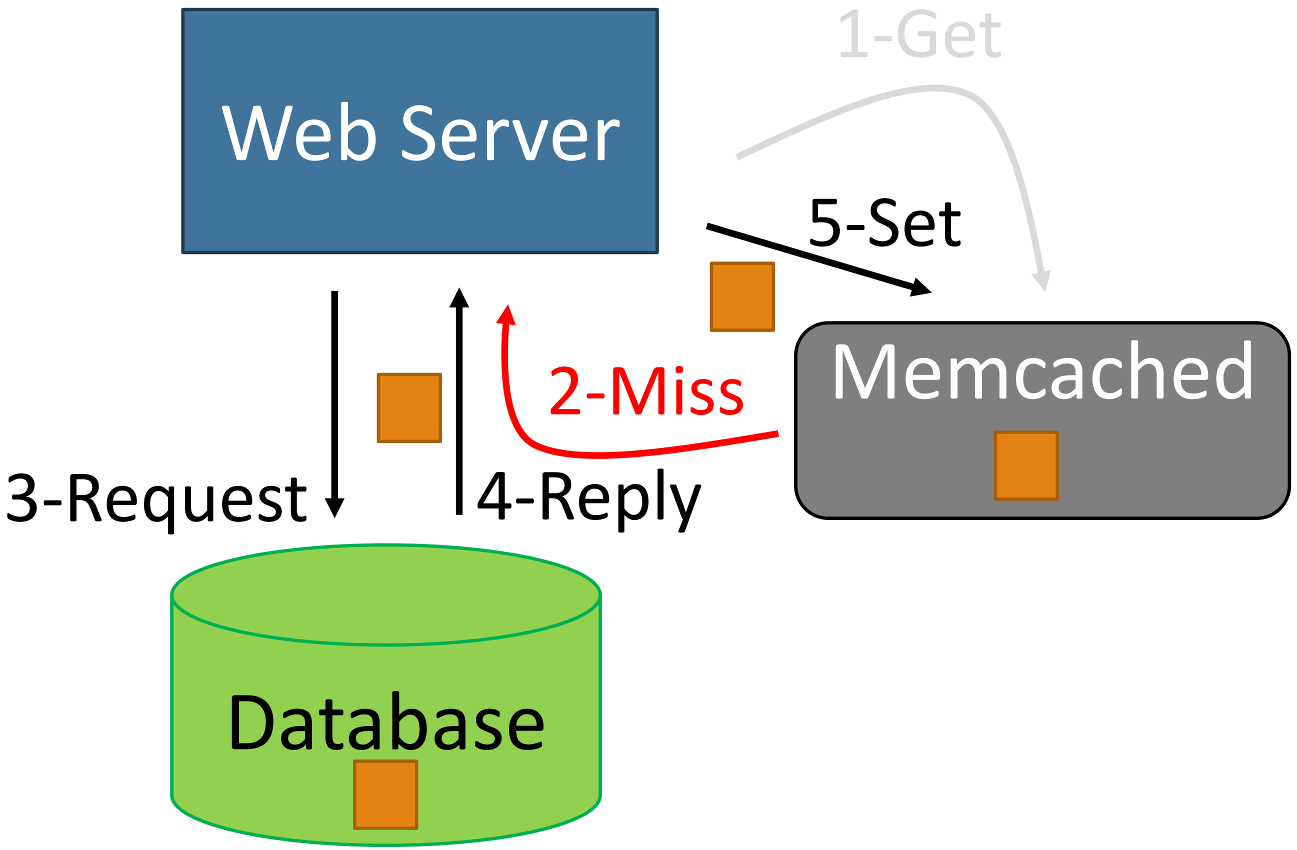


Memcached: A Highly-Used Server Application



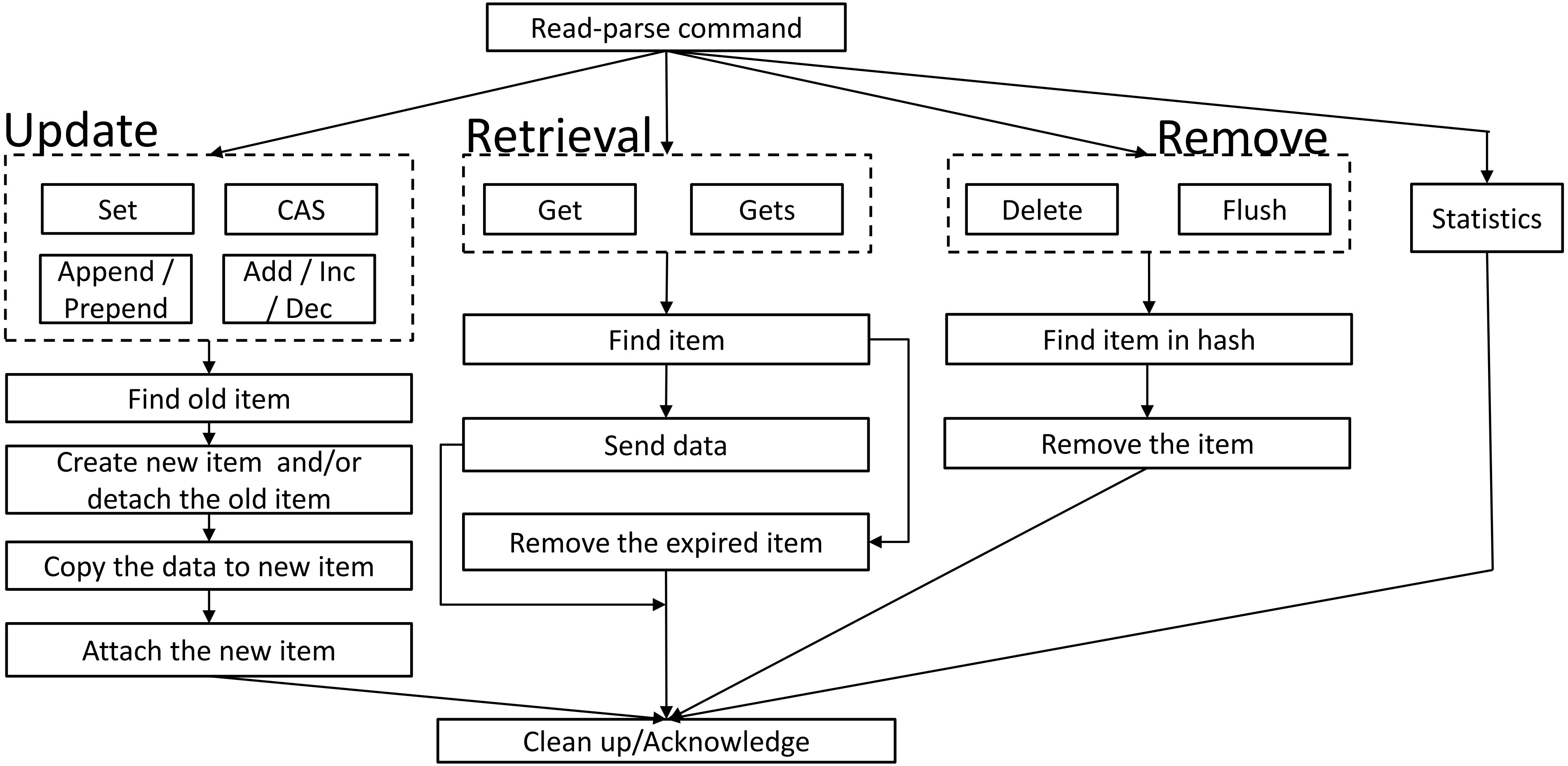
- An application level cache
  - Database queries
  - Server computations
- Used in social networking sites
  - Facebook
  - YouTube
  - Twitter
  - Reddit

Memcached: A Highly-Used Server Application

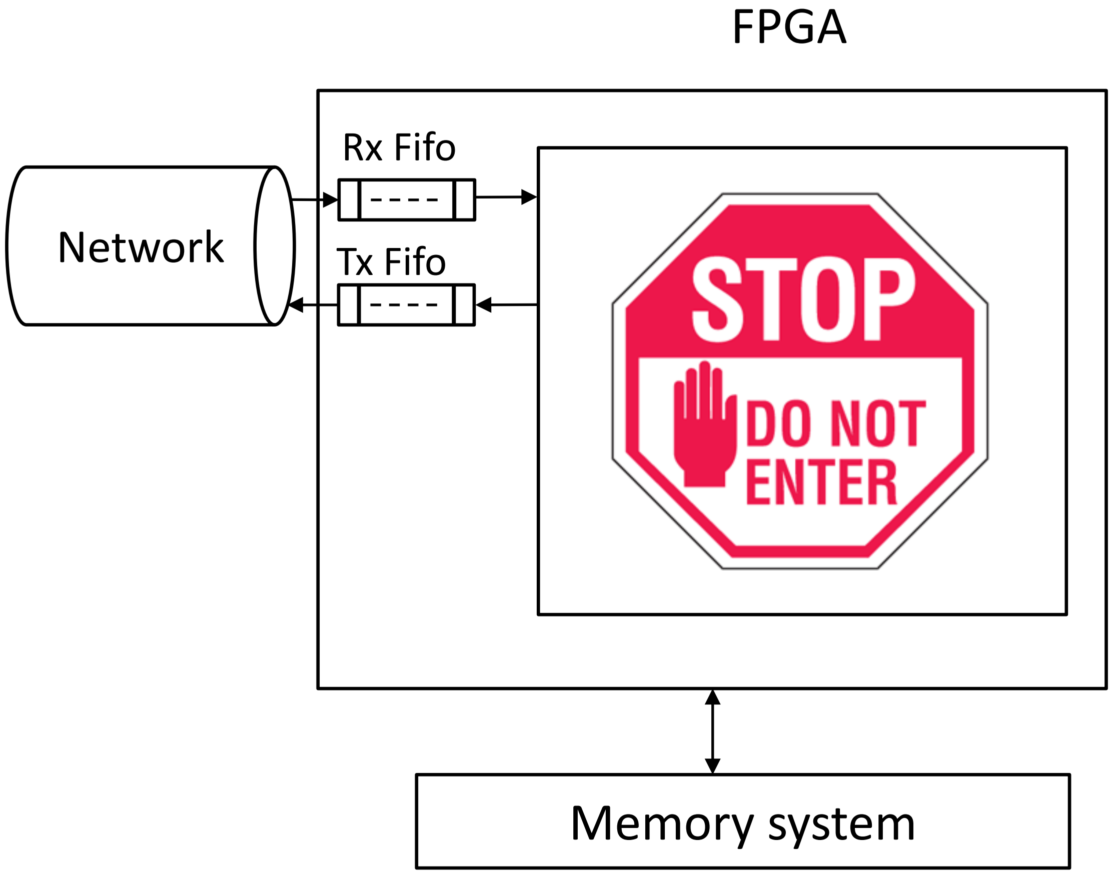


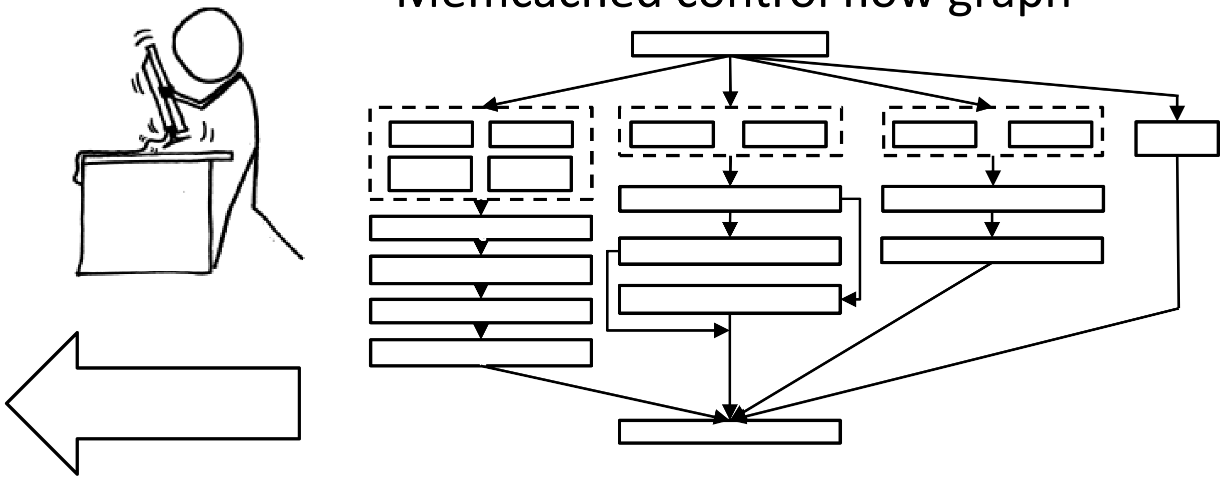
- On Memcached Miss
  - Access database
  - Update Memcached (Set)

Simplified Memcached Control Flow



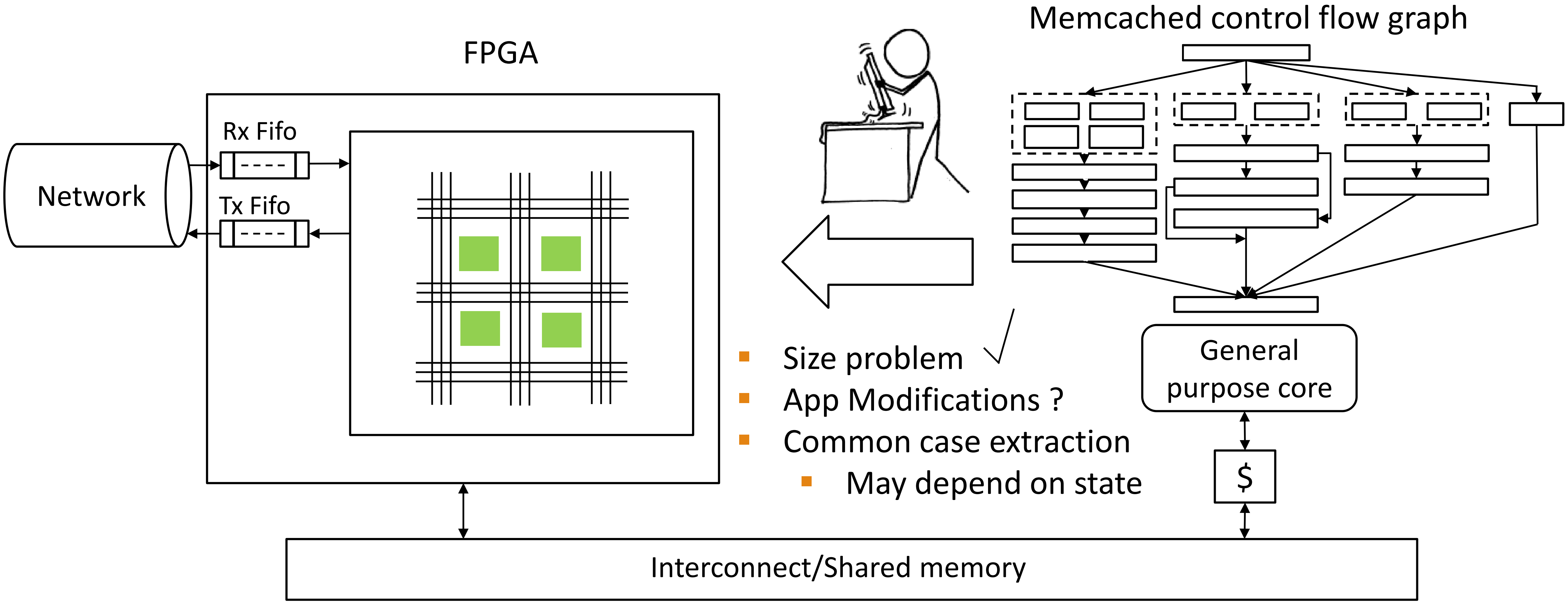
Memcached on an FPGA?





- Pros: Specialization benefit
- Cons: Complex to be implemented as hardware
  - Memcached entirely on an FPGA
    - Waste of hardware resources
  - Deal with application modifications

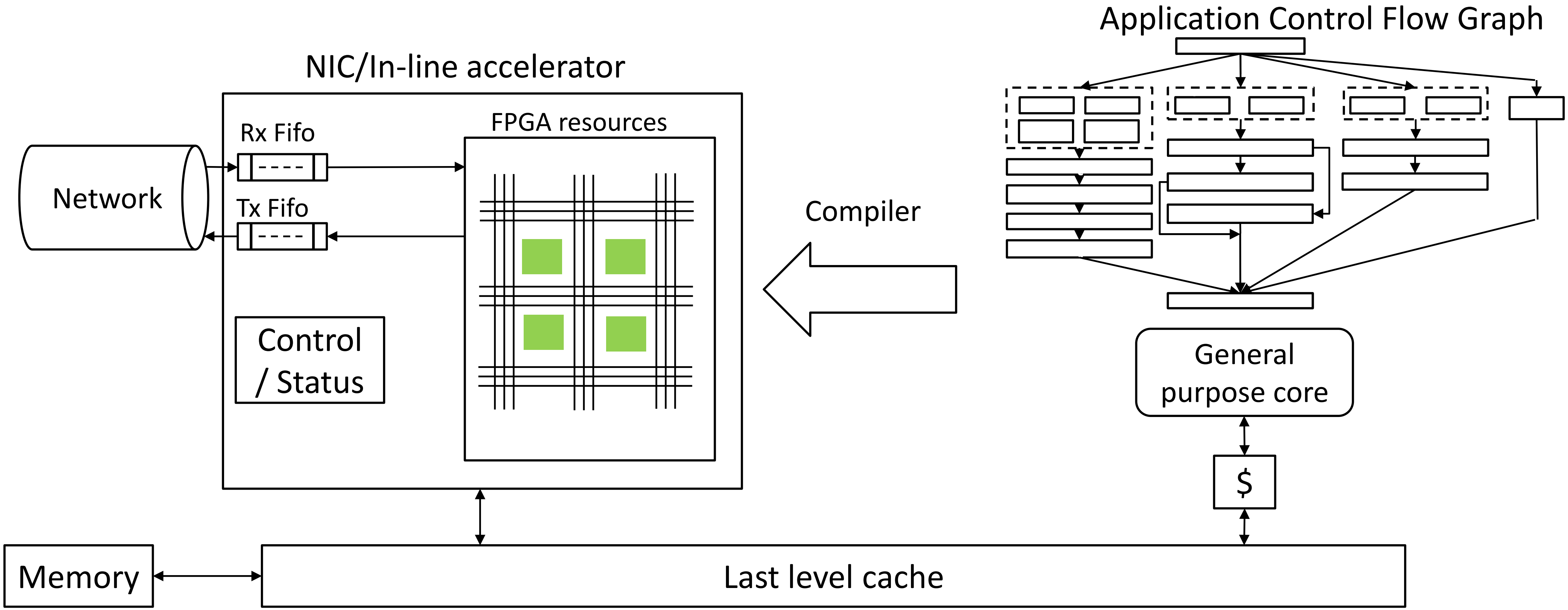
Hybrid Architecture: Optimize the common case



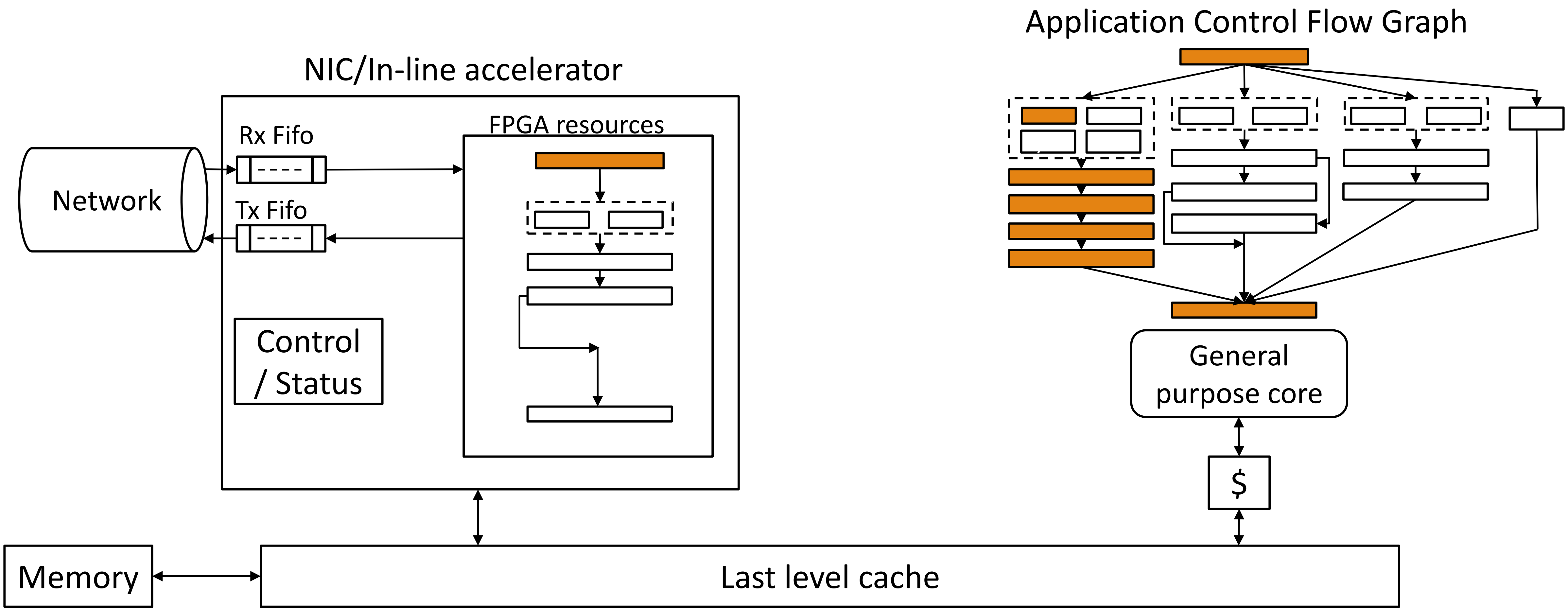
- Size problem ✓
- App Modifications ?
- Common case extraction
  - May depend on state

In-Line Acceleration

In-line Acceleration – Fast Path: Hot Trace



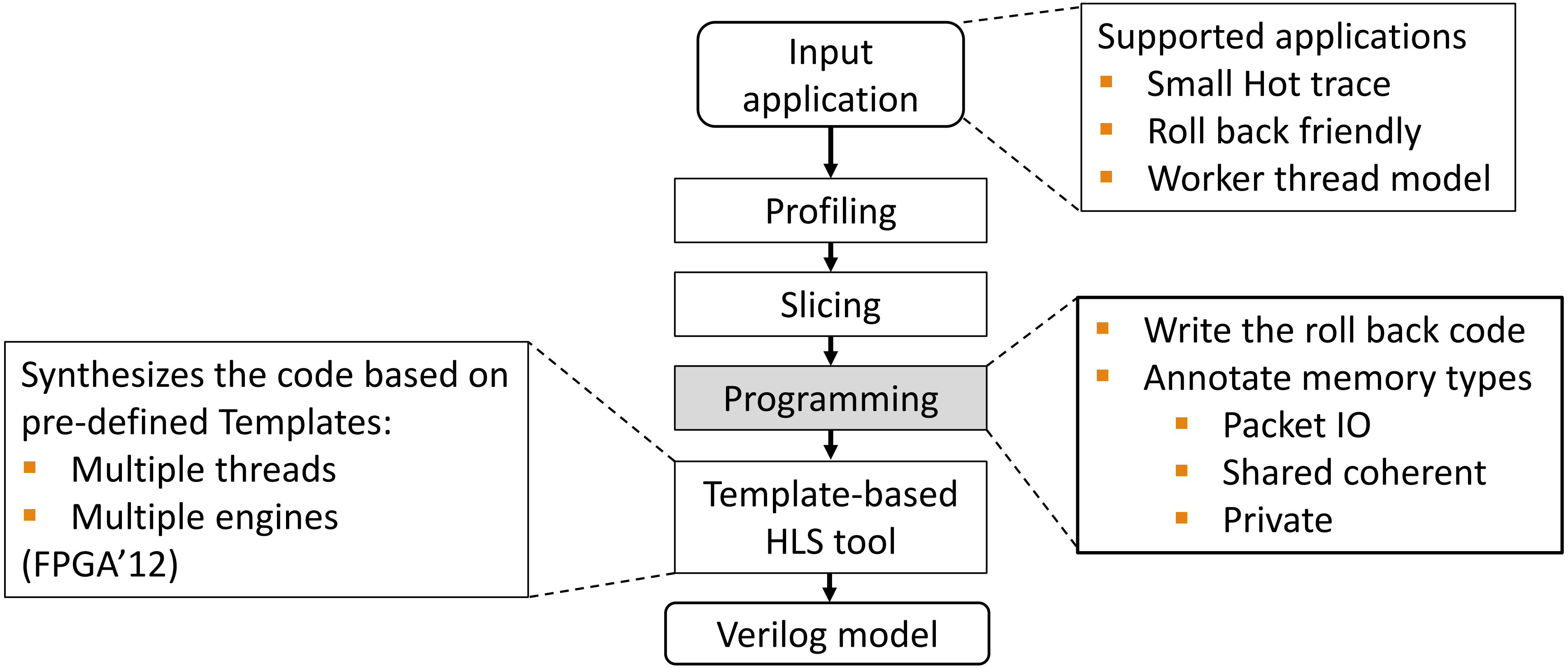
In-line Acceleration – Slow Path: Original App



Bail out Issue

- Problem:
  - Fast path may modify some global data and later decides to bail out
  - Transfer the computation to the slow path without causing any inconsistency
- Solution:
  - Roll back global updates
    - User-defined roll back routine
- Many server applications are roll back friendly
  - Rollback code is already available in transactional applications (i.e. databases)
  - Partial updates are isolated to provide atomicity
  - Memcached bail out : 30 Lines of Code

In-line Accelerator Generation Process



Input application

Profiling

Slicing

Programming

Template-based HLS tool

Verilog model

Supported applications

- Small Hot trace
- Roll back friendly
- Worker thread model

- Write the roll back code
- Annotate memory types
  - Packet IO
  - Shared coherent
  - Private

Synthesizes the code based on pre-defined Templates:

- Multiple threads
- Multiple engines

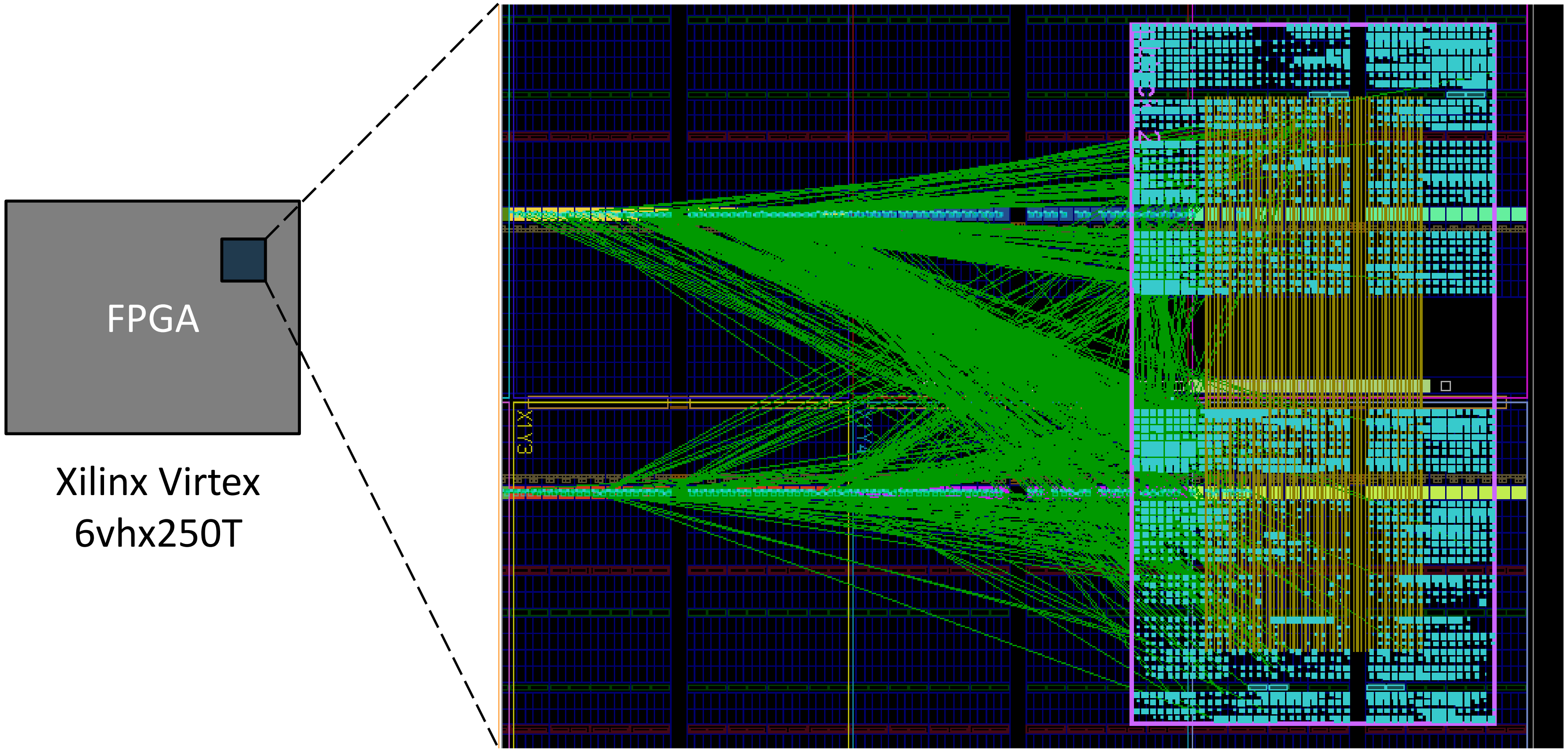
(FPGA’12)

Required programming << Hardware design efforts

Memcached Results

The Memcached In-line Accelerator

Single engine/Single thread - 5% of FPGA's LUTs

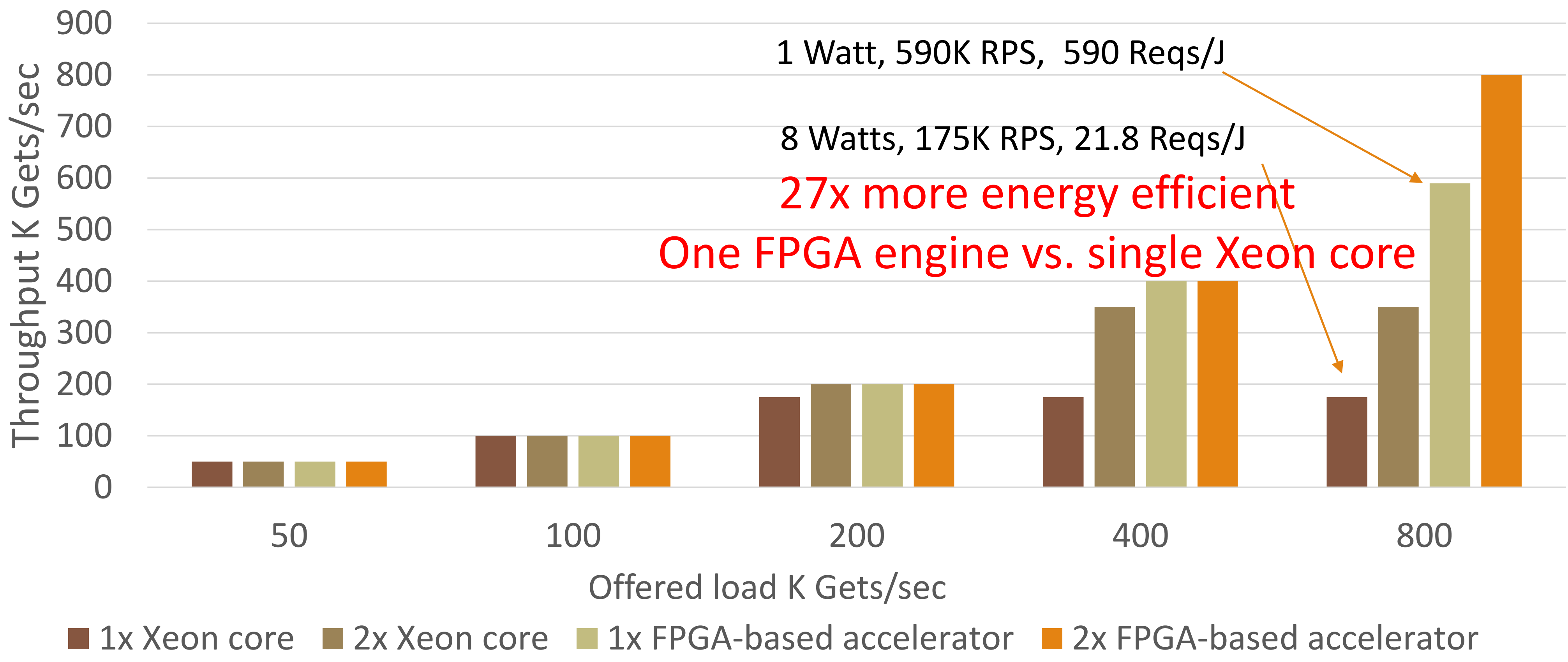


Xilinx Virtex
6vhx250T

Slice registers: 6570
Slice LUTs: 8739
Clock cycle: 7.30 ns

Assumes accelerator has dedicated last level cache port

Fast Path Performance/Power



1 Watt, 590K RPS, 590 Reqs/J

8 Watts, 175K RPS, 21.8 Reqs/J

27x more energy efficient

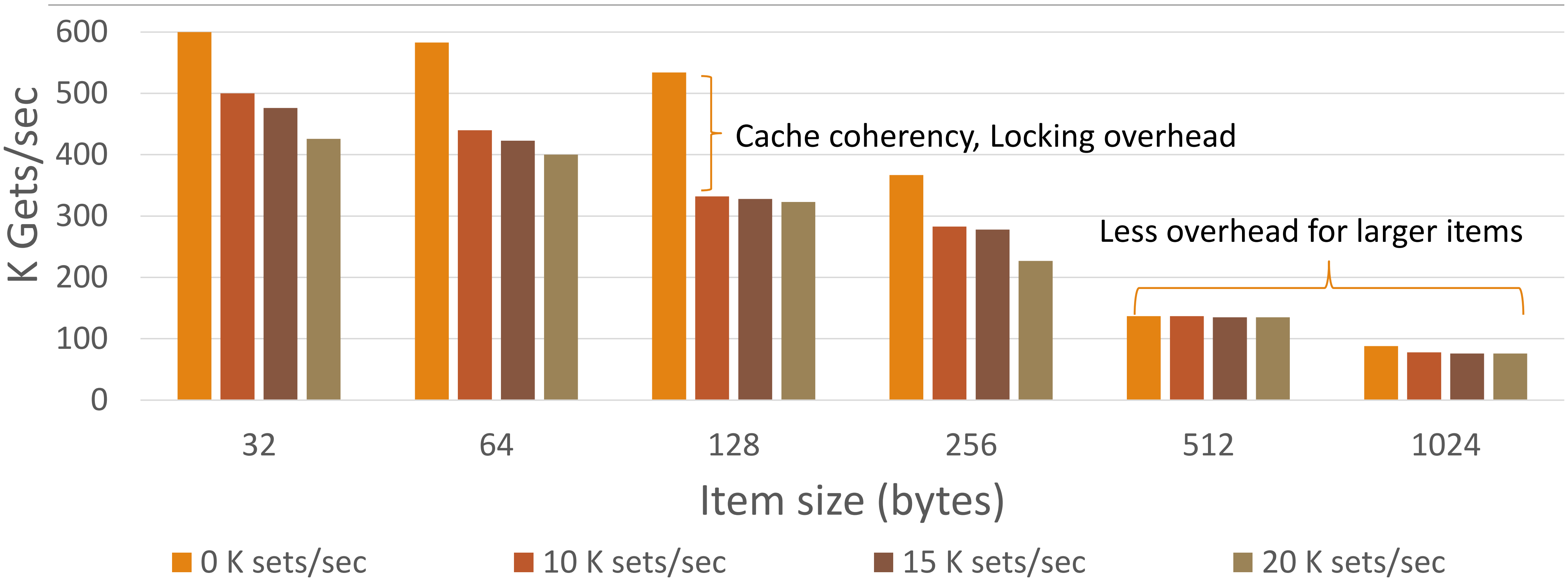
One FPGA engine vs. single Xeon core

Xeon frequency: 2.2 Ghz

FPGA frequency: 100 Mhz

Assumes accelerator has dedicated last level cache port

Fast Path/Slow path Interaction Cost



Full system stack - Gem5 simulation of Client and server

Server: LLC-connected FPGA-based accelerator (single engine fast path) + one Alpha core (slow path)

Assumes each engine has dedicated last level cache port

Projecting the Hybrid Architecture Performance

	Active Xeon cores	Accelerator engines	Processor + FPGA power (watts)	Performance (requests/sec)	Energy efficiency (requests/J)
Xeon processor	8	0	92	1.4 M	15.2 K
Xeon processor + In-line accelerator	1	4	33	1.6 M	48 K

Conclusion/Future work

- Conclusion
  - Accelerated Memcached
  - Performance/energy efficiency: Important
  - Software-like programmability: Also important
- Future work
  - In-line acceleration of other applications
  - Tradeoffs in the selection of fast path
  - Alternative bail out solutions
  - Reduce cache port assumptions

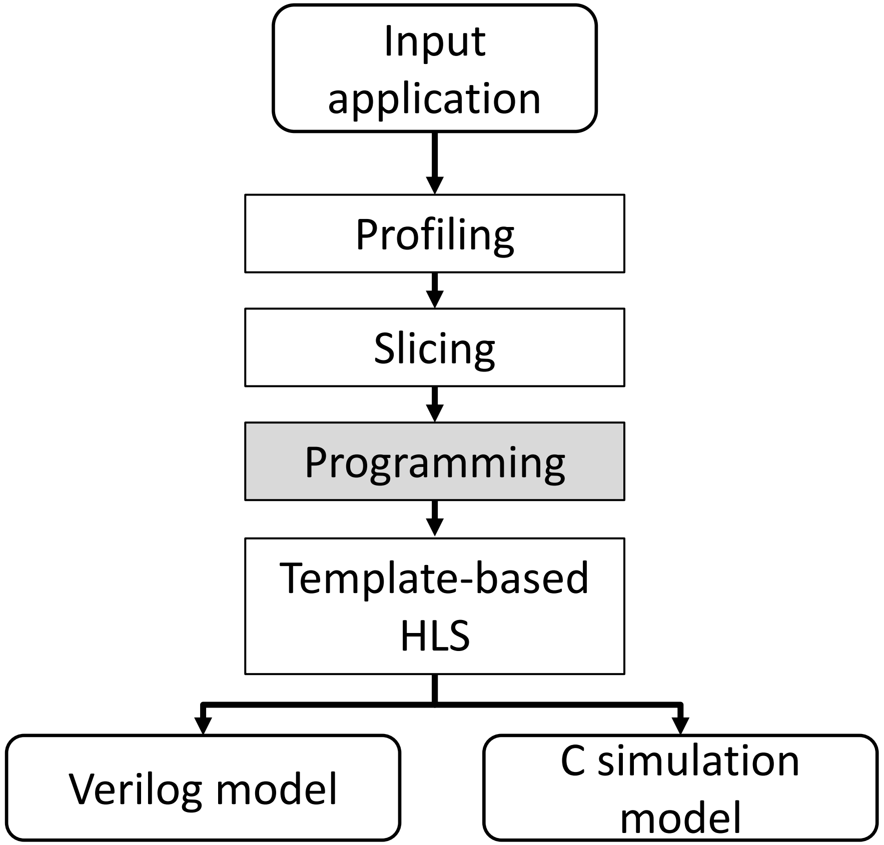
Thank you

This material is based upon work supported in part by the National Science Foundation under grants 0747438 and 0917158.

Recent work

Reference	Platform	Main contribution
Berezecki et al. , IGCC 2011	Tilera	Many-small-cores are more efficient
Hetherington et al., ISPASS 2012	GPU	GPU can improve the performance
Chalamalasetti et al., FPGA 2012	FPGA	Get requests on FPGA
Lim et al., ISCA 2013	ASIC/FPGA + General purpose cores	Get on FPGA, the rest on CPU
Blott et al., HotClouds 2013	FPGA + General purpose cores	Improving the throughput
Lavasani et al., CAL 2013	FPGA + General purpose cores	Automatic slicing and HW generation

Generating Memcached In-line Accelerator



Memcached 1.4.10

10687 LOCs

User-level working thread: Libevent

Profiled and sliced

Hot trace: Almost all get operations on UDP

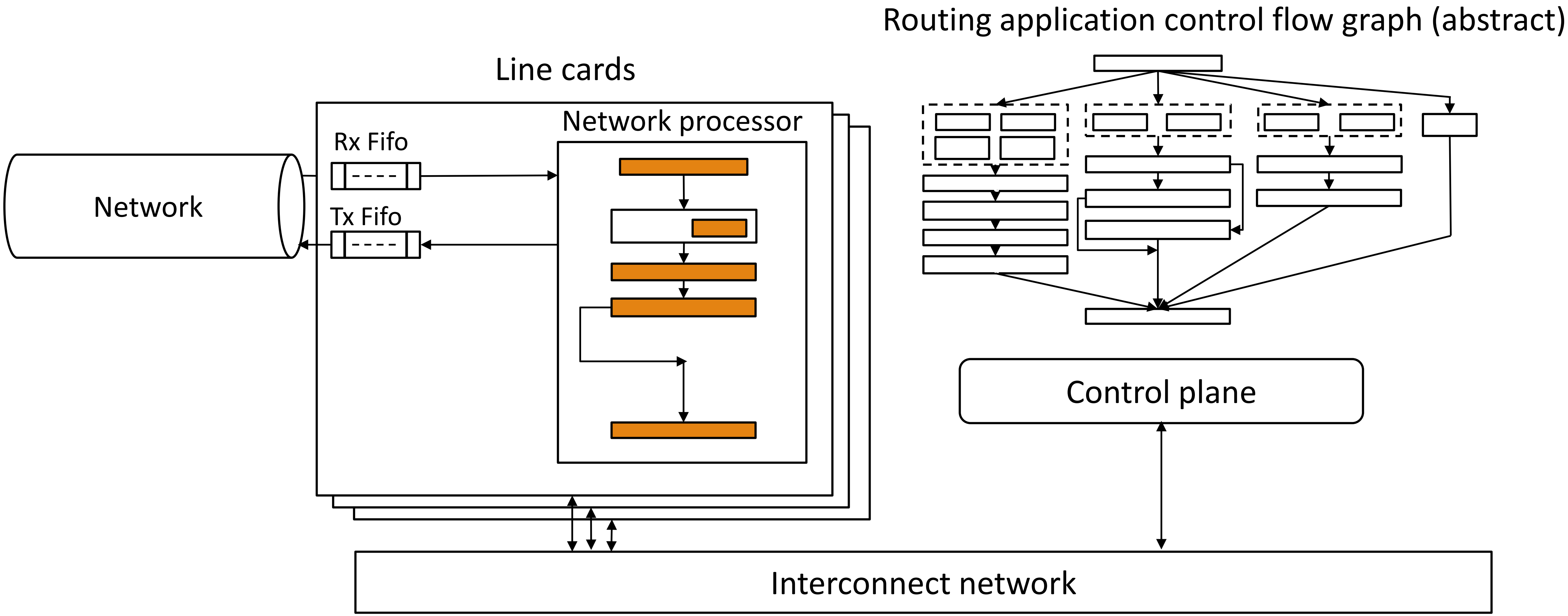
Hot routines LOCs: 963

Hot instructions: <2000

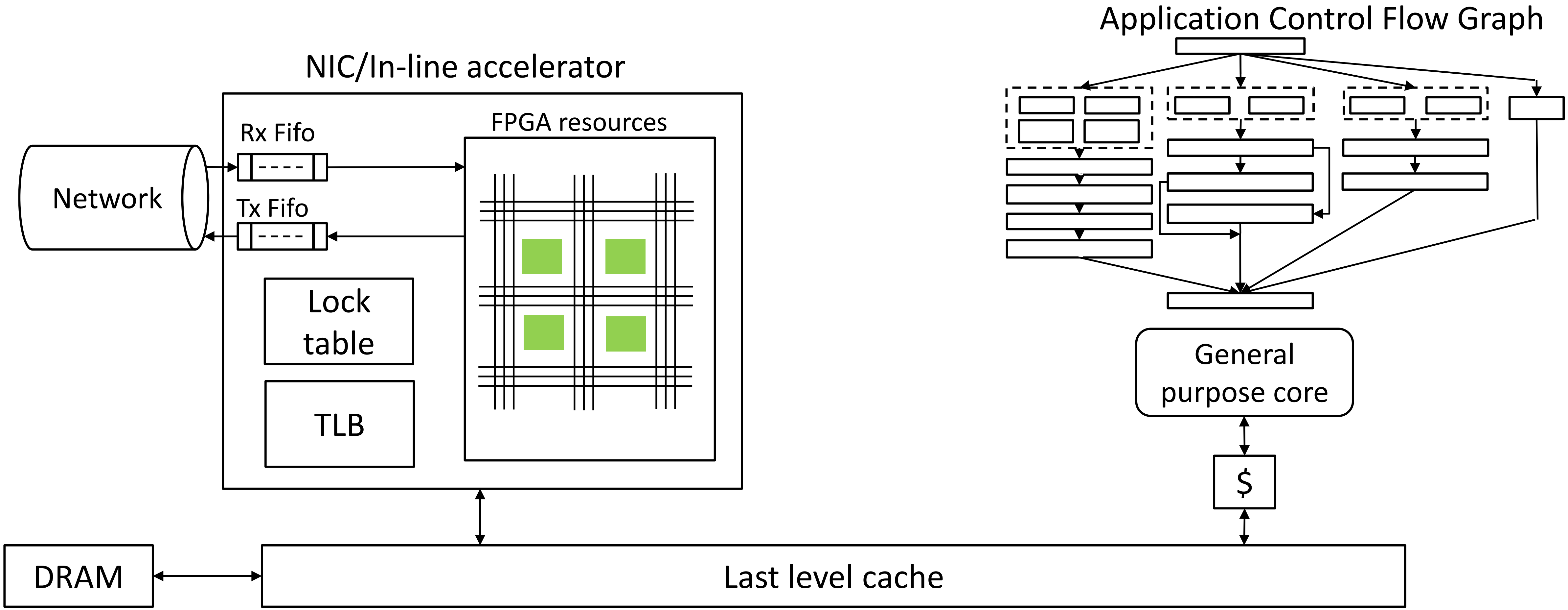
Bail-out code is 30 lines of code

Memory type annotation

IP Routers Are Similar



Synchronization/Address Translation


Application Control Flow Graph
NIC/In-line accelerator
FPGA resources
Rx Fifo
Tx Fifo
Network
Lock table
TLB
General purpose core
$
DRAM
Last level cache