Computing In-Situ and In-Transit

Lizy K. John
The University of Texas at Austin

ENERGY COST OF MOVING DATA

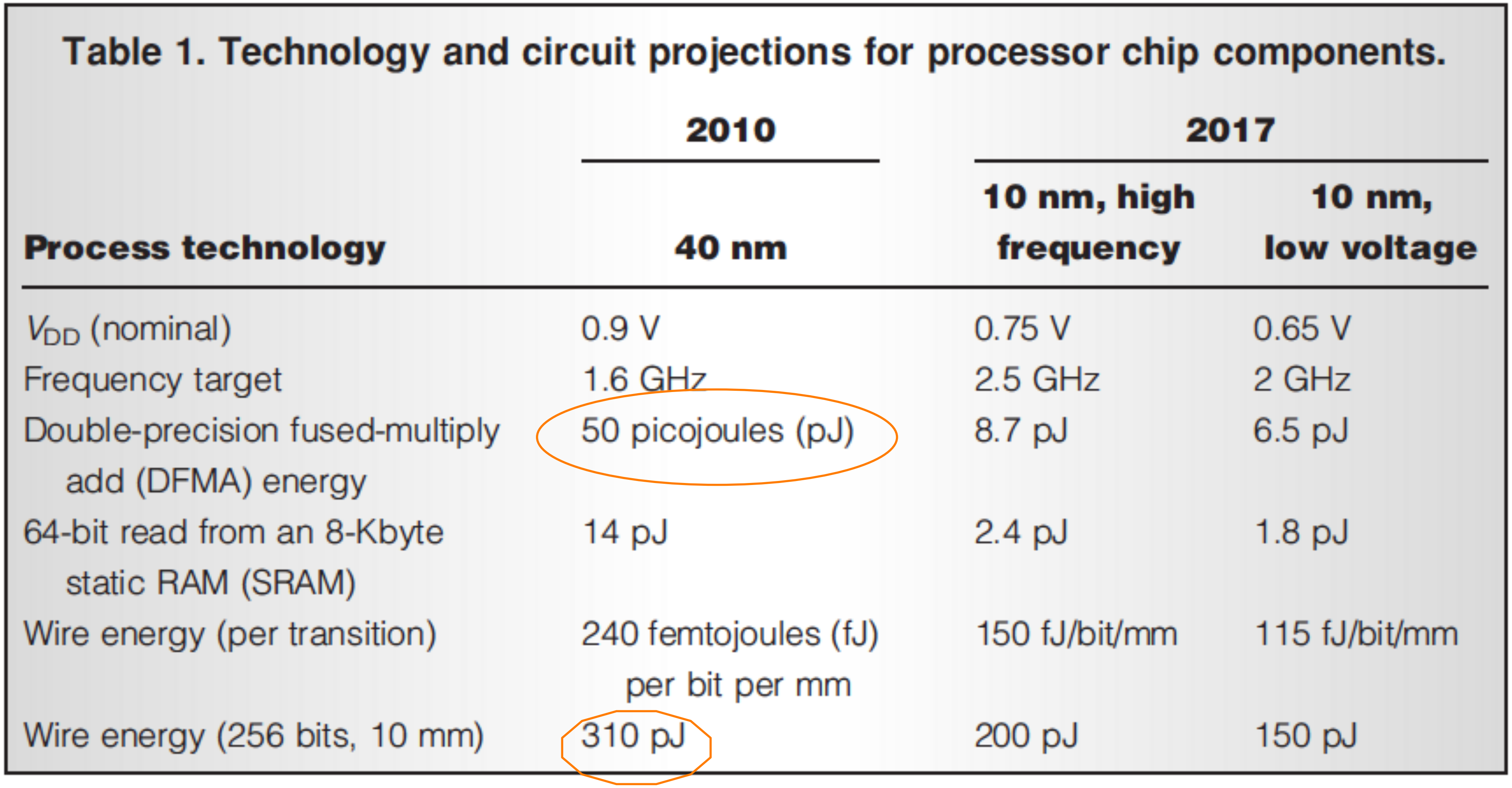


Table 1. Technology and circuit projections for processor chip components.

	2010	2017	
Process technology	40 nm	10 nm, high frequency	10 nm, low voltage
V_{DD} (nominal)	0.9 V	0.75 V	0.65 V
Frequency target	1.6 GHz	2.5 GHz	2 GHz
Double-precision fused-multiply add (DFMA) energy	50 picojoules (pJ)	8.7 pJ	6.5 pJ
64-bit read from an 8-Kbyte static RAM (SRAM)	14 pJ	2.4 pJ	1.8 pJ
Wire energy (per transition)	240 femtojoules (fJ) per bit per mm	150 fJ/bit/mm	115 fJ/bit/mm
Wire energy (256 bits, 10 mm)	310 pJ	200 pJ	150 pJ

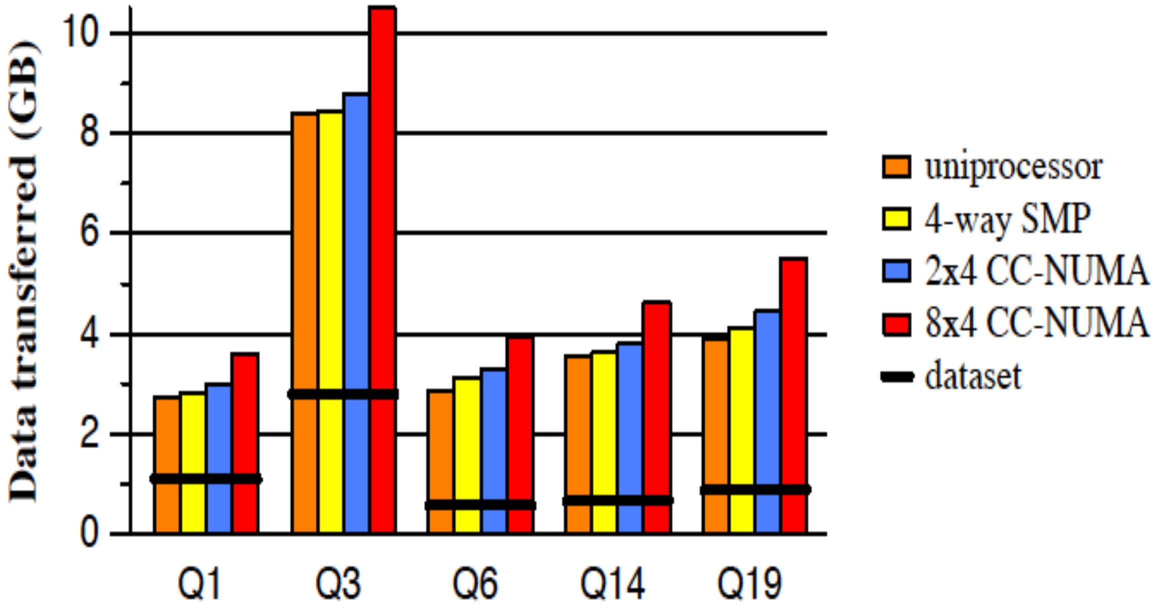
Keckler et al (Nvidia) IEEE Micro Sept 2011

Moving Data is Expensive but We Often Move Lots of Data

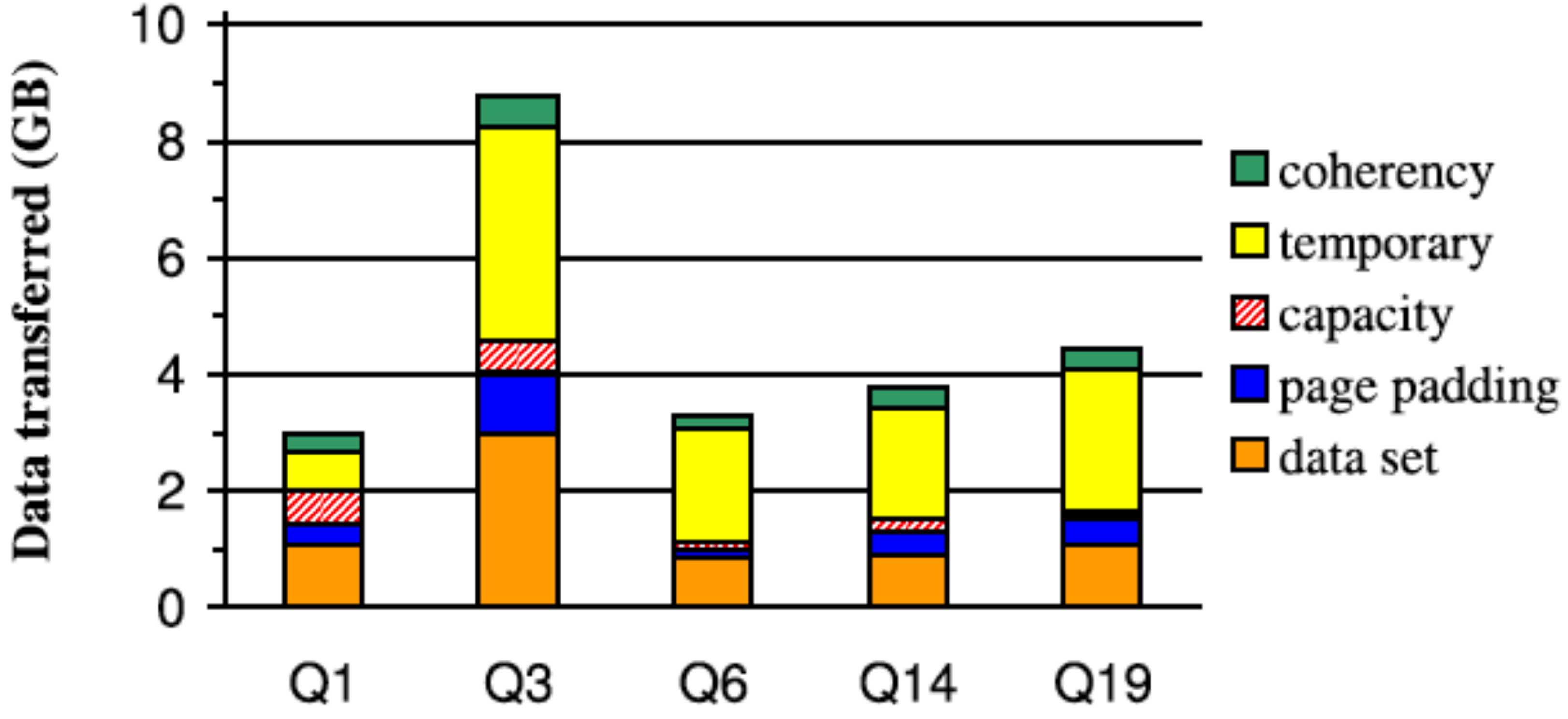
Amount of Data Moved Versus Amount of Data to be Processed

JUAN RUBIO's Ph. D Dissertation, 2005 (UT ECE)

Query	Name	Data set size
Q1	Pricing Summary Report	1.1 GB
Q3	Shipping Priority	2.8 GB
Q6	Forecasting Revenue Change	585 MB
Q14	Promotion Effect	686 MB
Q19	Discounted Revenue	902 MB



Classification of the Data Transferred (Rubio)



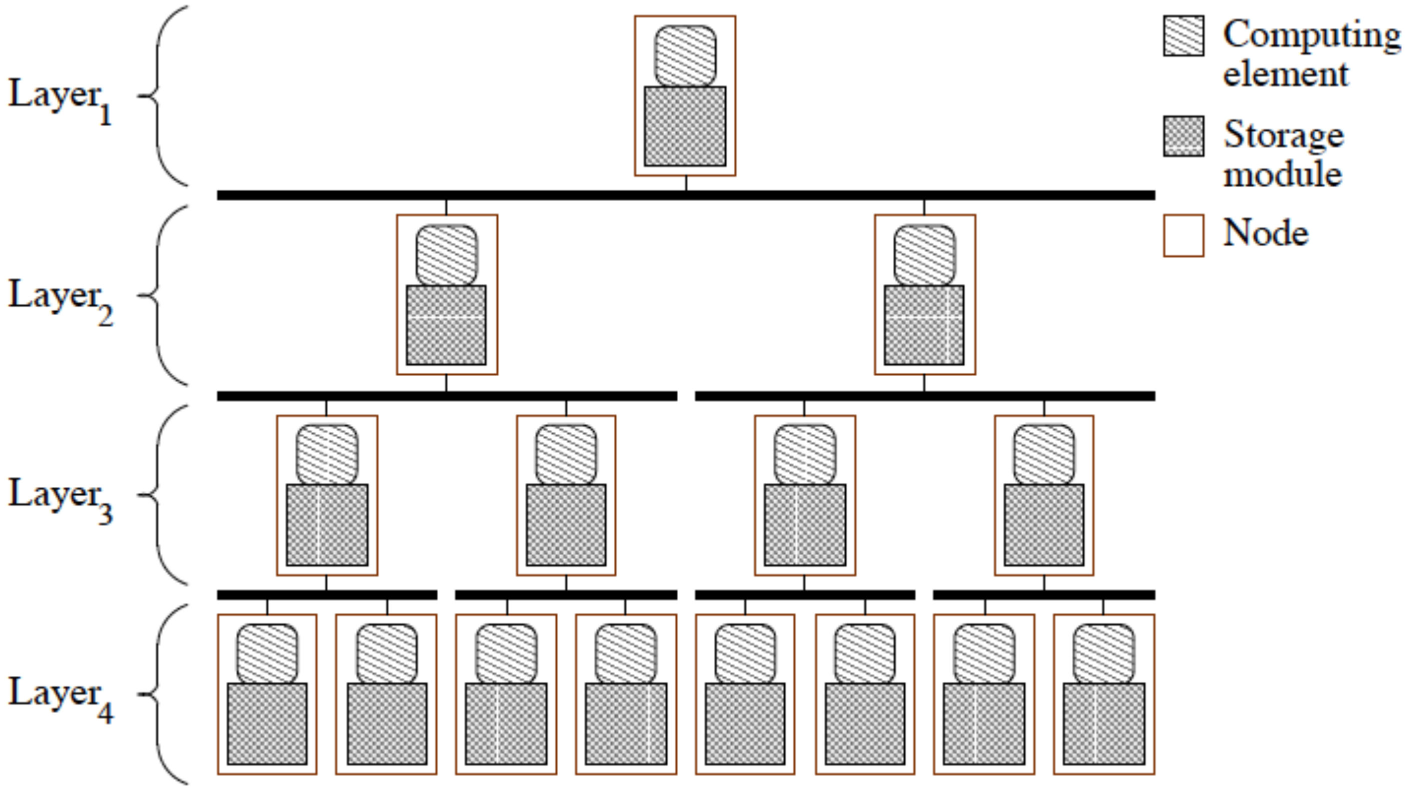
COMPUTE IN-SITU

- **Compute in the Caches**
- **Compute in the DRAM**
- **Compute in the Disk**
- **Compute in the core the data is (heterogeneous cores)**
- **Compute where the data is**
- **Compute in the CPU or the accelerator where the data is?**

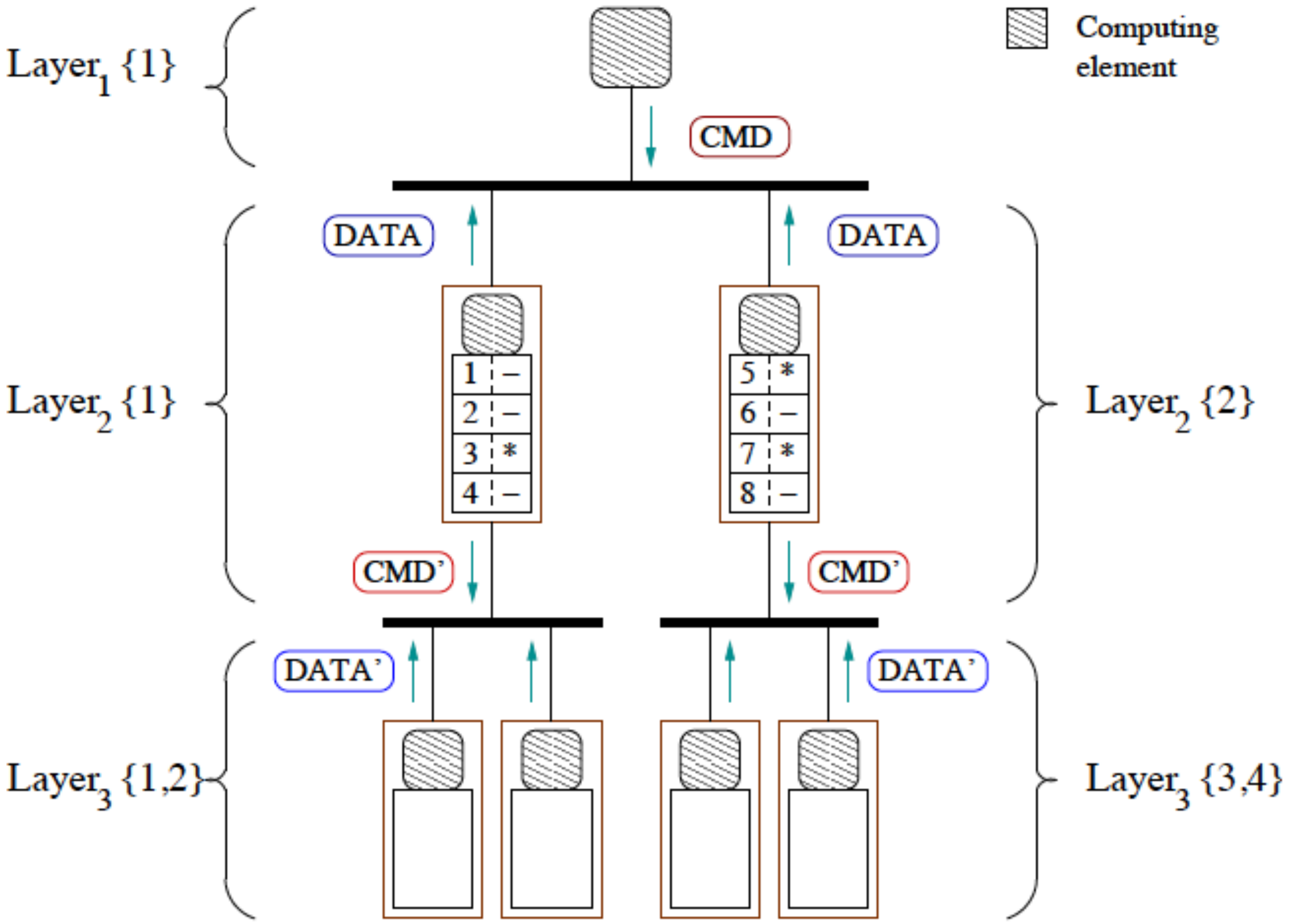
COMPUTE IN-TRANSIT

- **Compute while moving data from memory to core**
- **Compute in the Memory Controllers**
- **Compute while moving from CPU to GPU or vice versa**
- **Compute while moving data to accelerators**

Hierarchical Computing (Rubio Dissertation)



Hierarchical Computing – Task Decomposition



Primitives

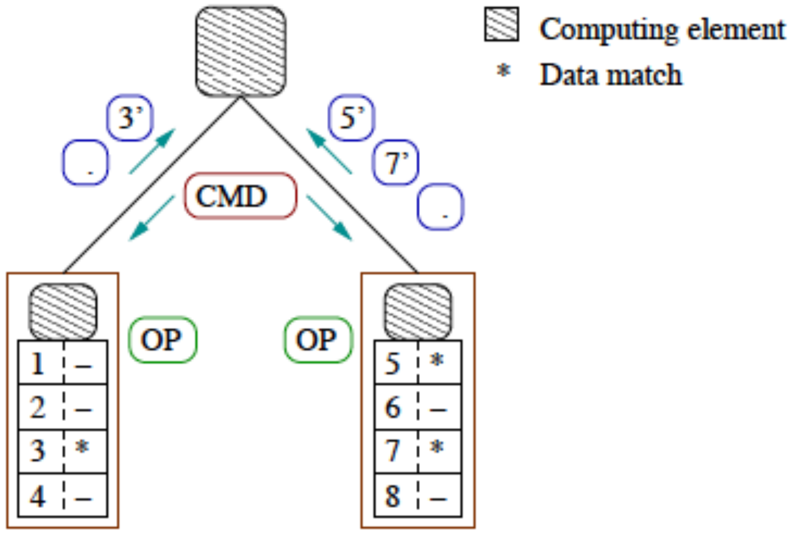


Figure 3.3: Individual Primitive.

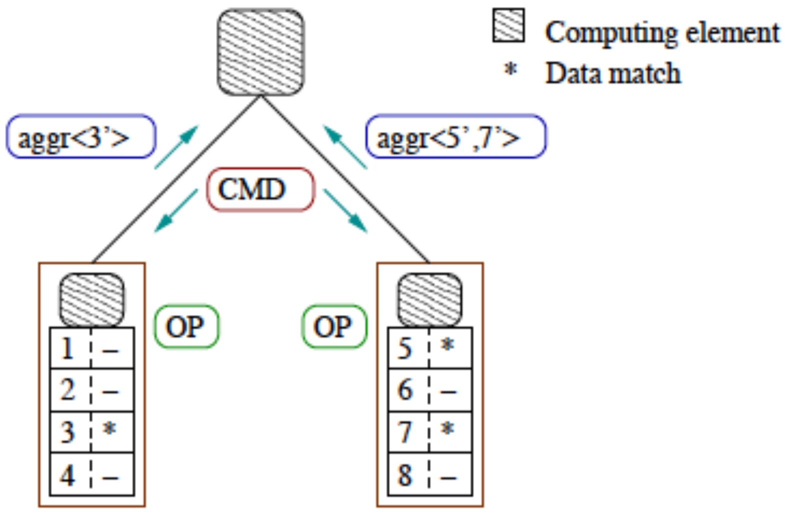
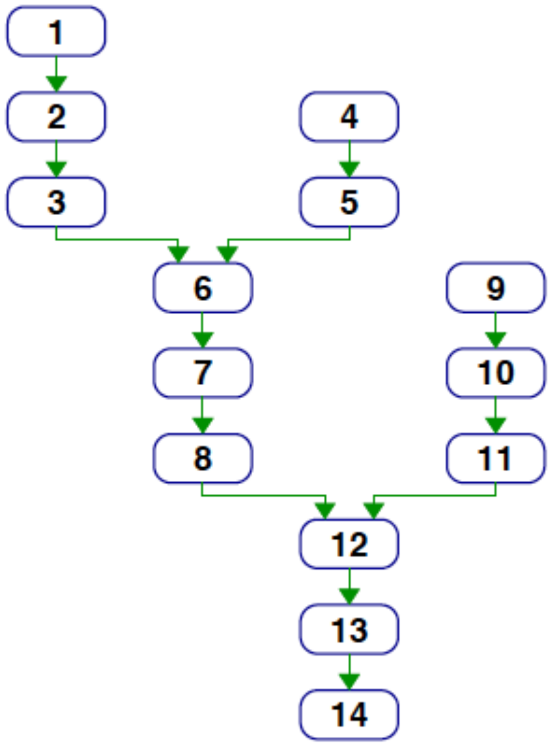
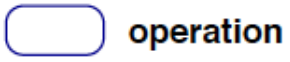


Figure 3.4: Aggregate Primitive.

Task Mapping



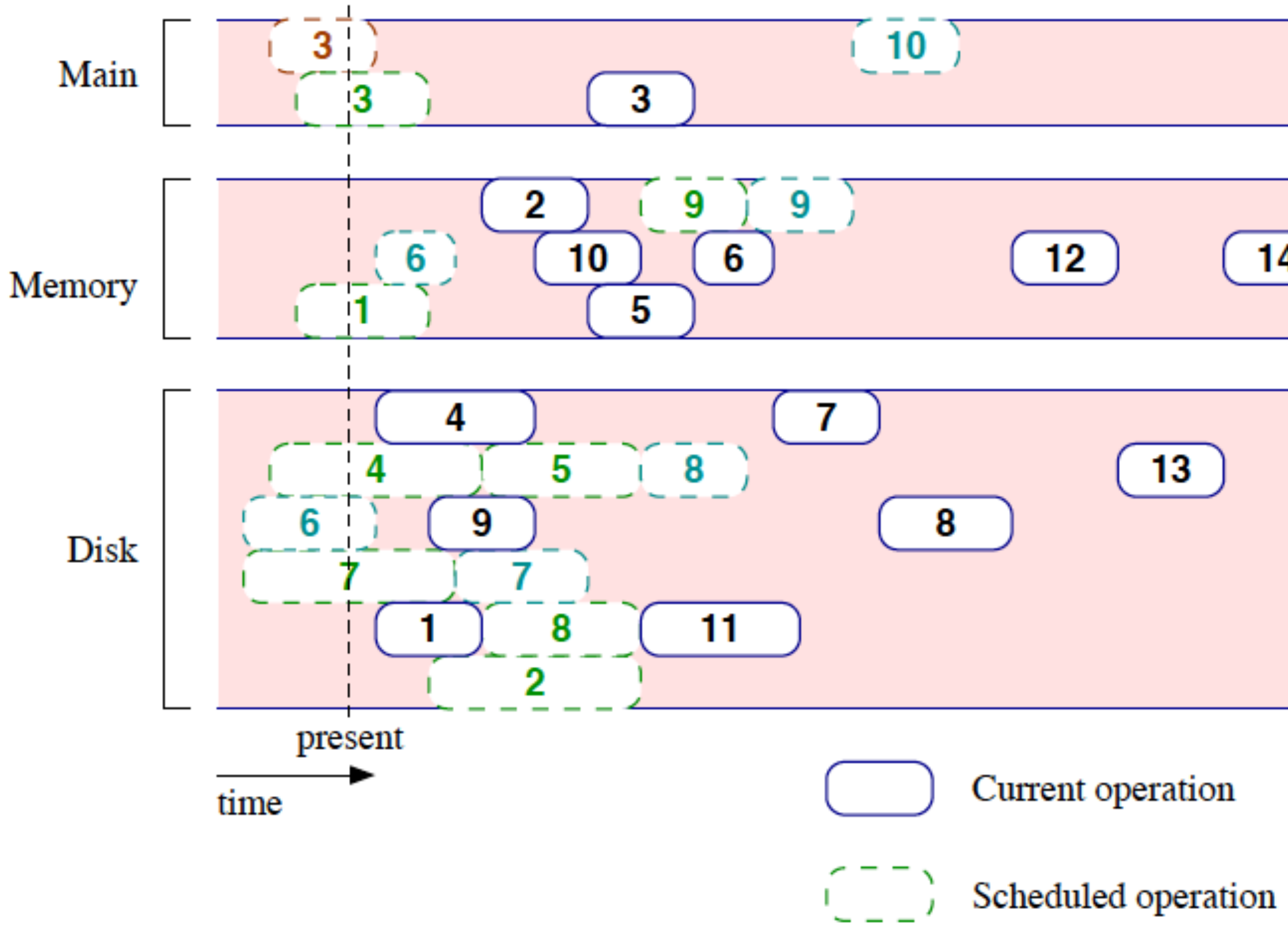
(a)



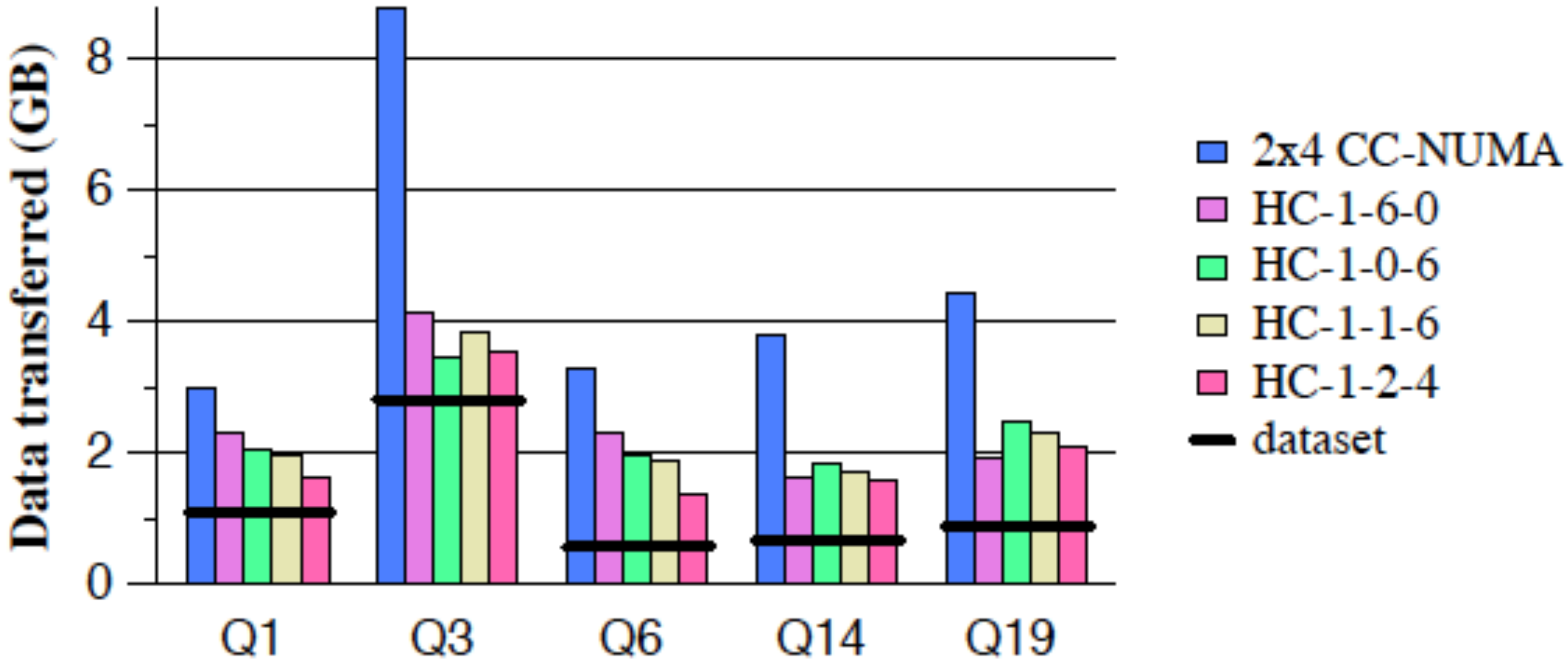
```
operations: 14

operation: "1"
type: table scan
input1: "table orders"
input2: ""
output: "buffer b1"

operation: "2"
type: sort
input1: "buffer b1"
input2: ""
output: "buffer b2"
...

operation: "6"
type: merge join
input1: "buffer b3"
input2: "buffer b5"
output: "buffer b6"
```

(b)

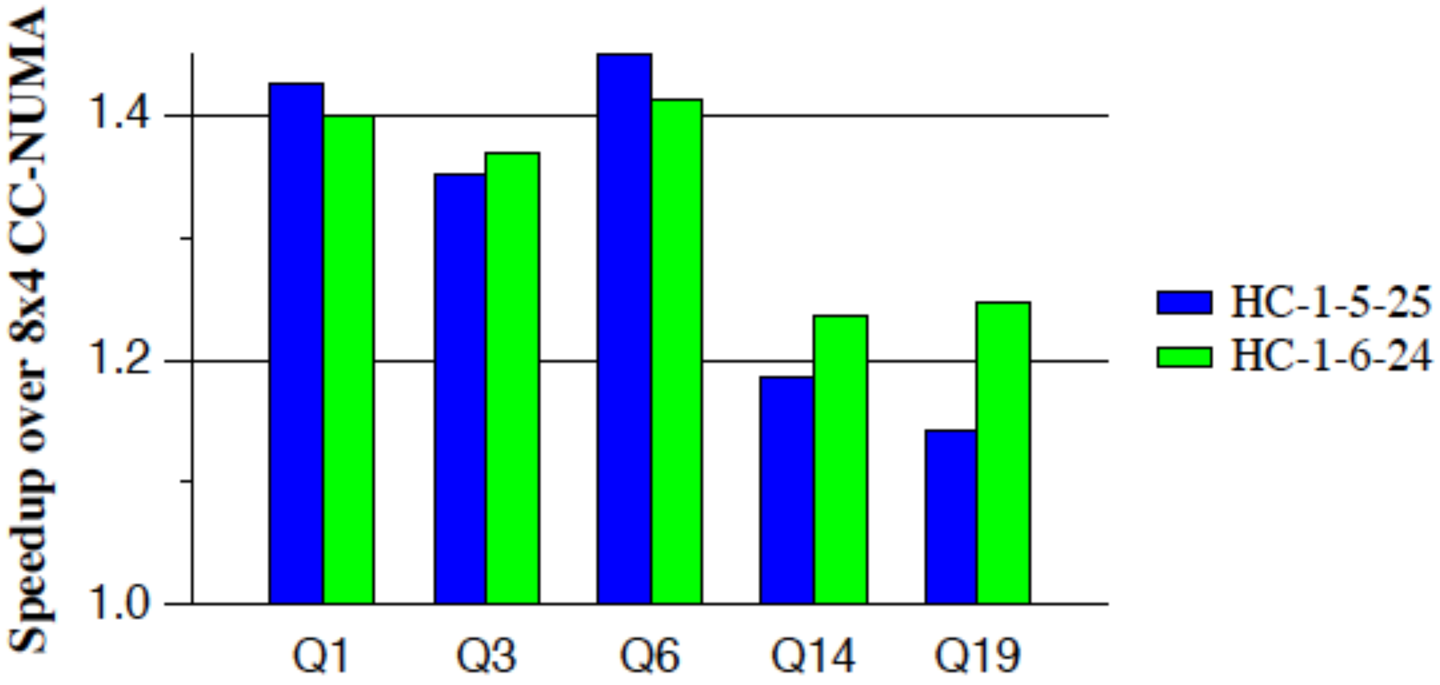
Tasks mapped to various layers in the hierarchical system



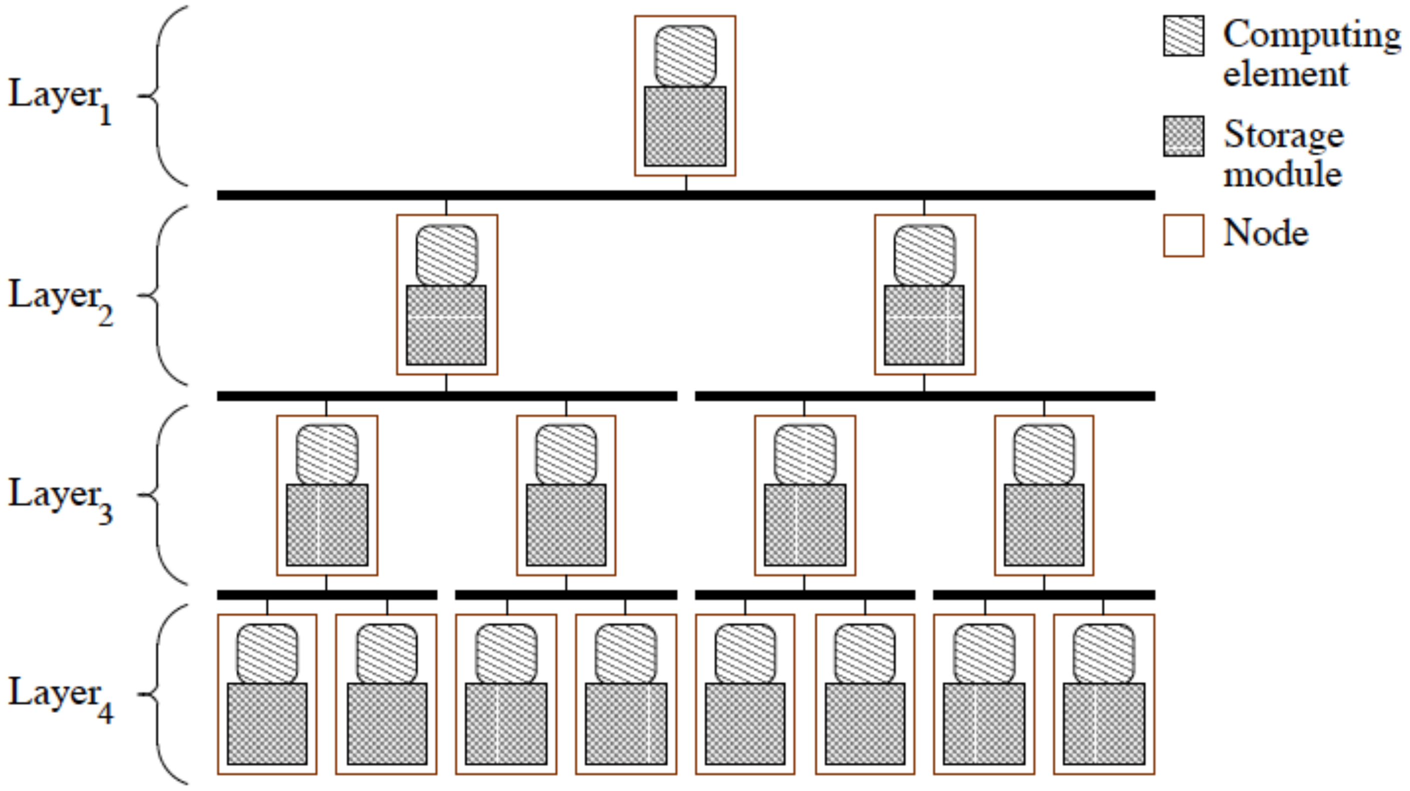
Data Transfer in the Hierarchical Computing

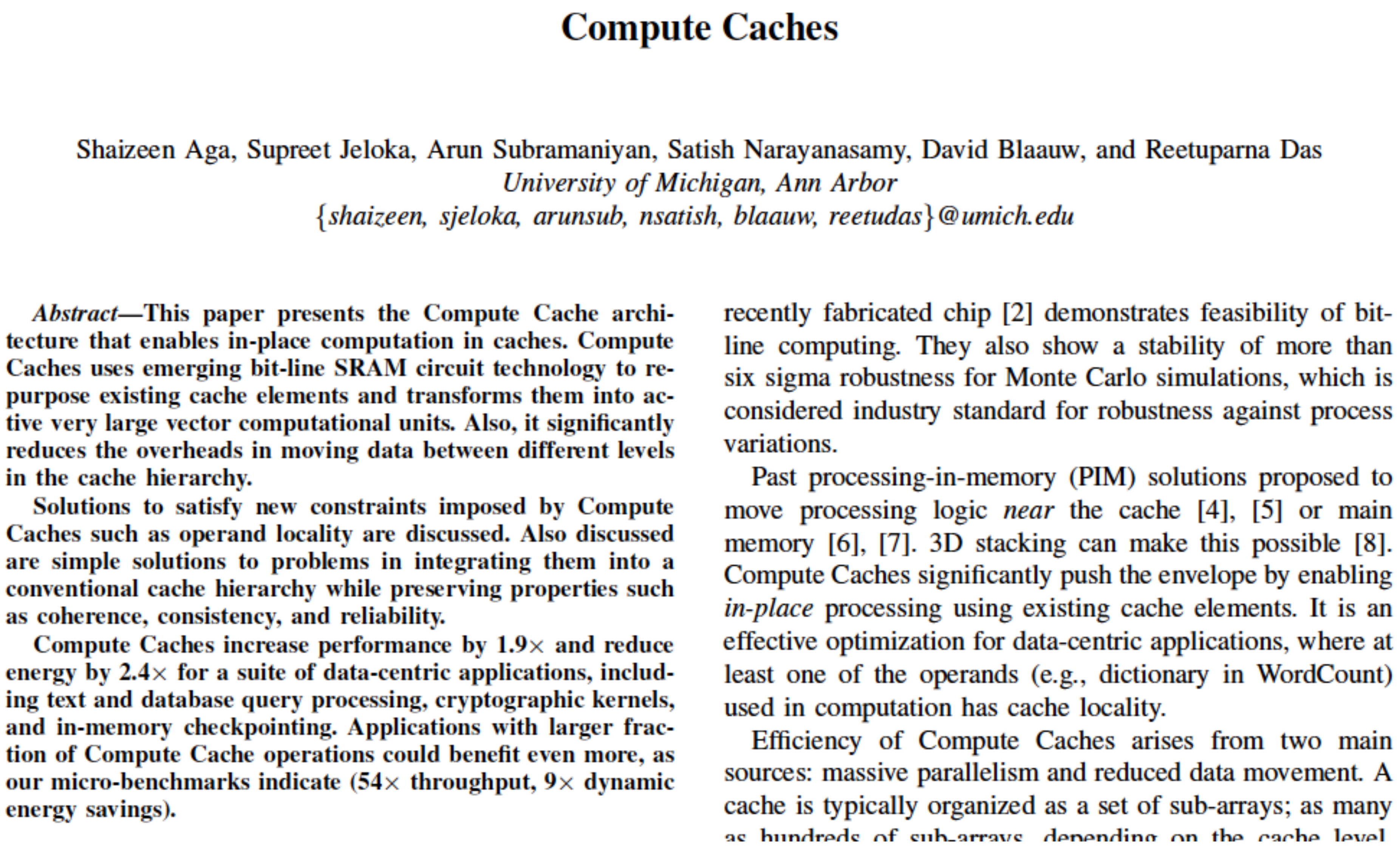


Speedup of Hierarchical Computing



Technology Developments Better Aligned Now





Compute Caches

Shaizeen Aga, Supreet Jeloka, Arun Subramaniyan, Satish Narayanasamy, David Blaauw, and Reetuparna Das
University of Michigan, Ann Arbor
{shaizeen, sjeloka, arunsub, nsatish, blaauw, reetudas}@umich.edu

***Abstract*—This paper presents the Compute Cache architecture that enables in-place computation in caches. Compute Caches uses emerging bit-line SRAM circuit technology to re-purpose existing cache elements and transforms them into active very large vector computational units. Also, it significantly reduces the overheads in moving data between different levels in the cache hierarchy.**

Solutions to satisfy new constraints imposed by Compute Caches such as operand locality are discussed. Also discussed are simple solutions to problems in integrating them into a conventional cache hierarchy while preserving properties such as coherence, consistency, and reliability.

Compute Caches increase performance by 1.9× and reduce energy by 2.4× for a suite of data-centric applications, including text and database query processing, cryptographic kernels, and in-memory checkpointing. Applications with larger fraction of Compute Cache operations could benefit even more, as our micro-benchmarks indicate (54× throughput, 9× dynamic energy savings).

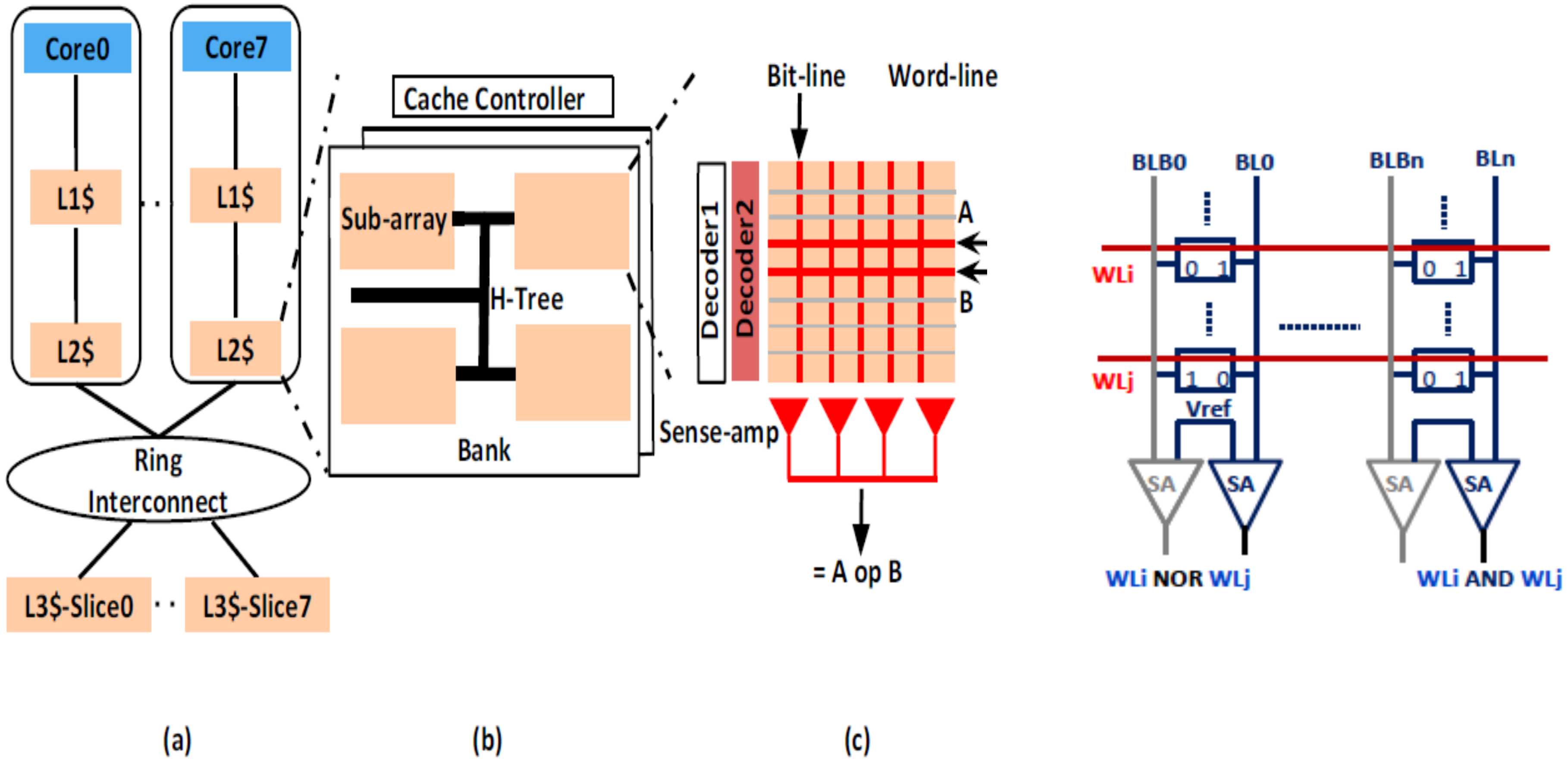
recently fabricated chip [2] demonstrates feasibility of bit-line computing. They also show a stability of more than six sigma robustness for Monte Carlo simulations, which is considered industry standard for robustness against process variations.

Past processing-in-memory (PIM) solutions proposed to move processing logic *near* the cache [4], [5] or main memory [6], [7]. 3D stacking can make this possible [8]. Compute Caches significantly push the envelope by enabling *in-place* processing using existing cache elements. It is an effective optimization for data-centric applications, where at least one of the operands (e.g., dictionary in WordCount) used in computation has cache locality.

Efficiency of Compute Caches arises from two main sources: massive parallelism and reduced data movement. A cache is typically organized as a set of sub-arrays; as many as hundreds of sub-arrays, depending on the cache level

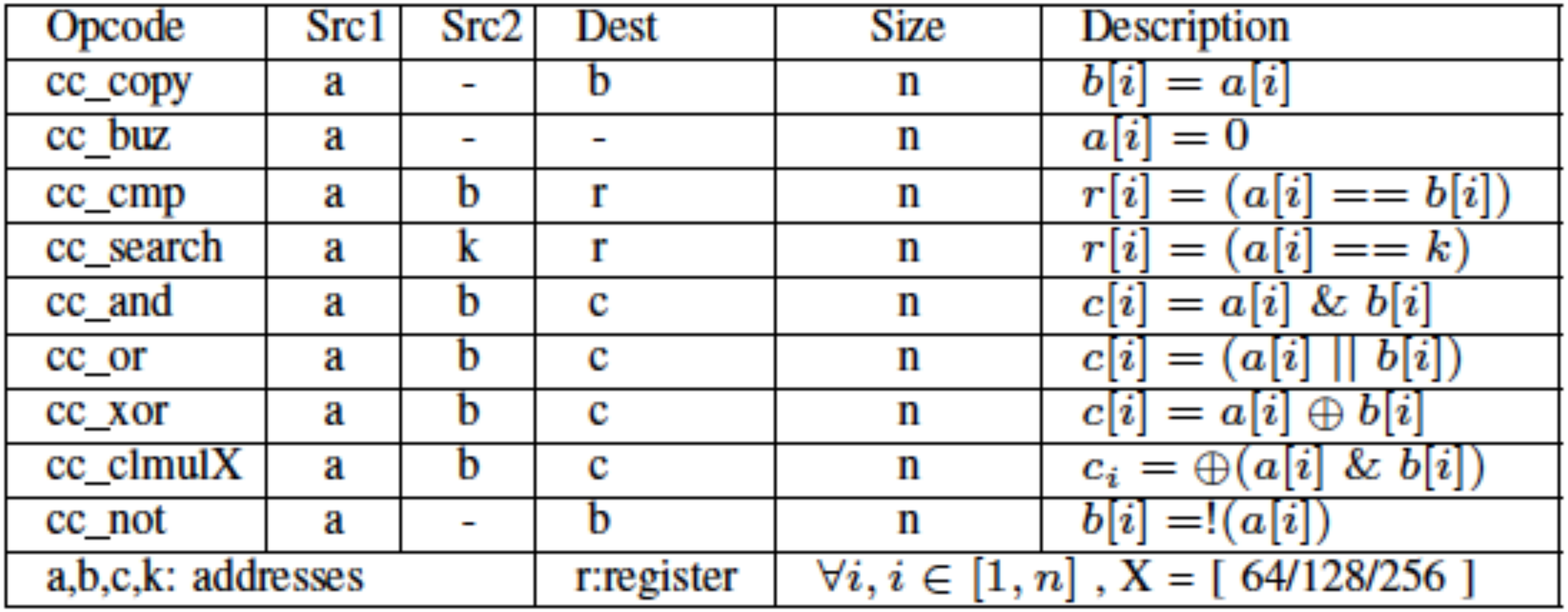
HPCA 2017

Compute Cache Overview



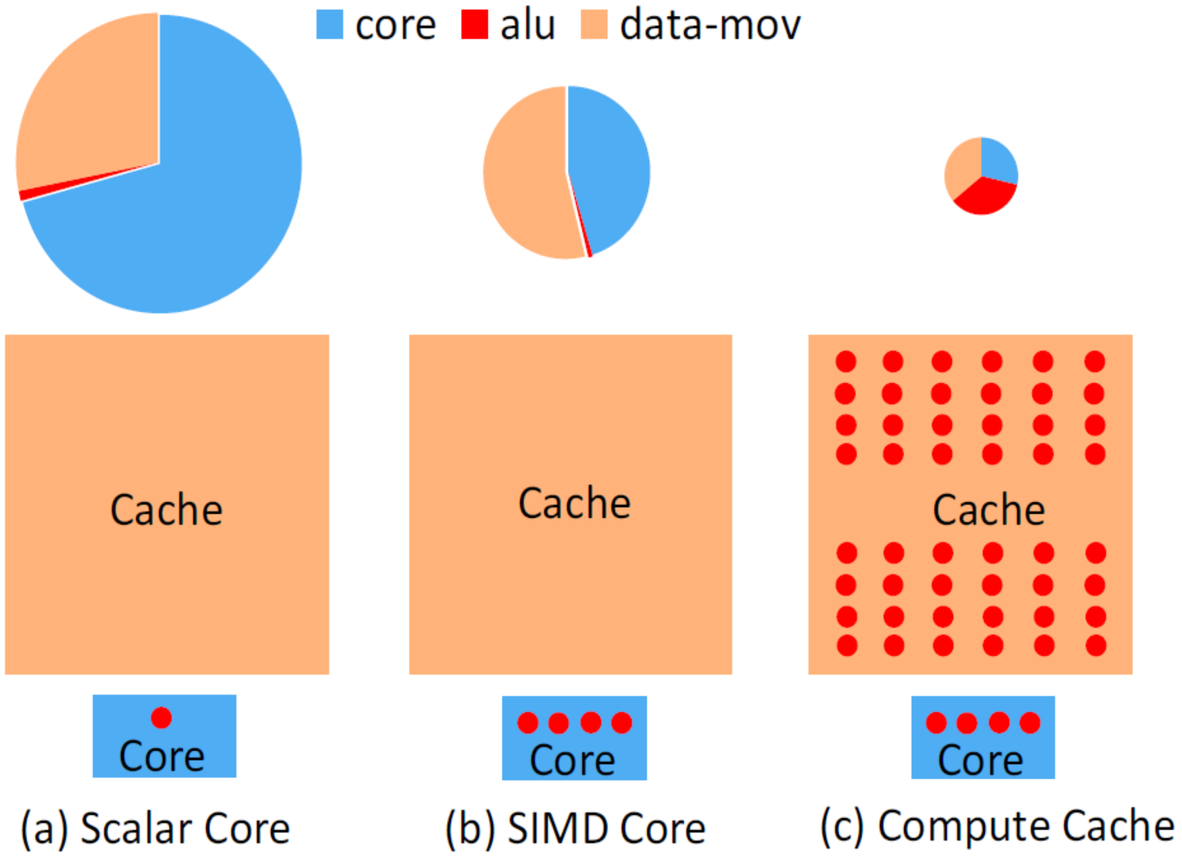
(a) Cache hierarchy. (b) Cache Geometry (c) In-place compute in a sub-array.

Compute Cache ISA



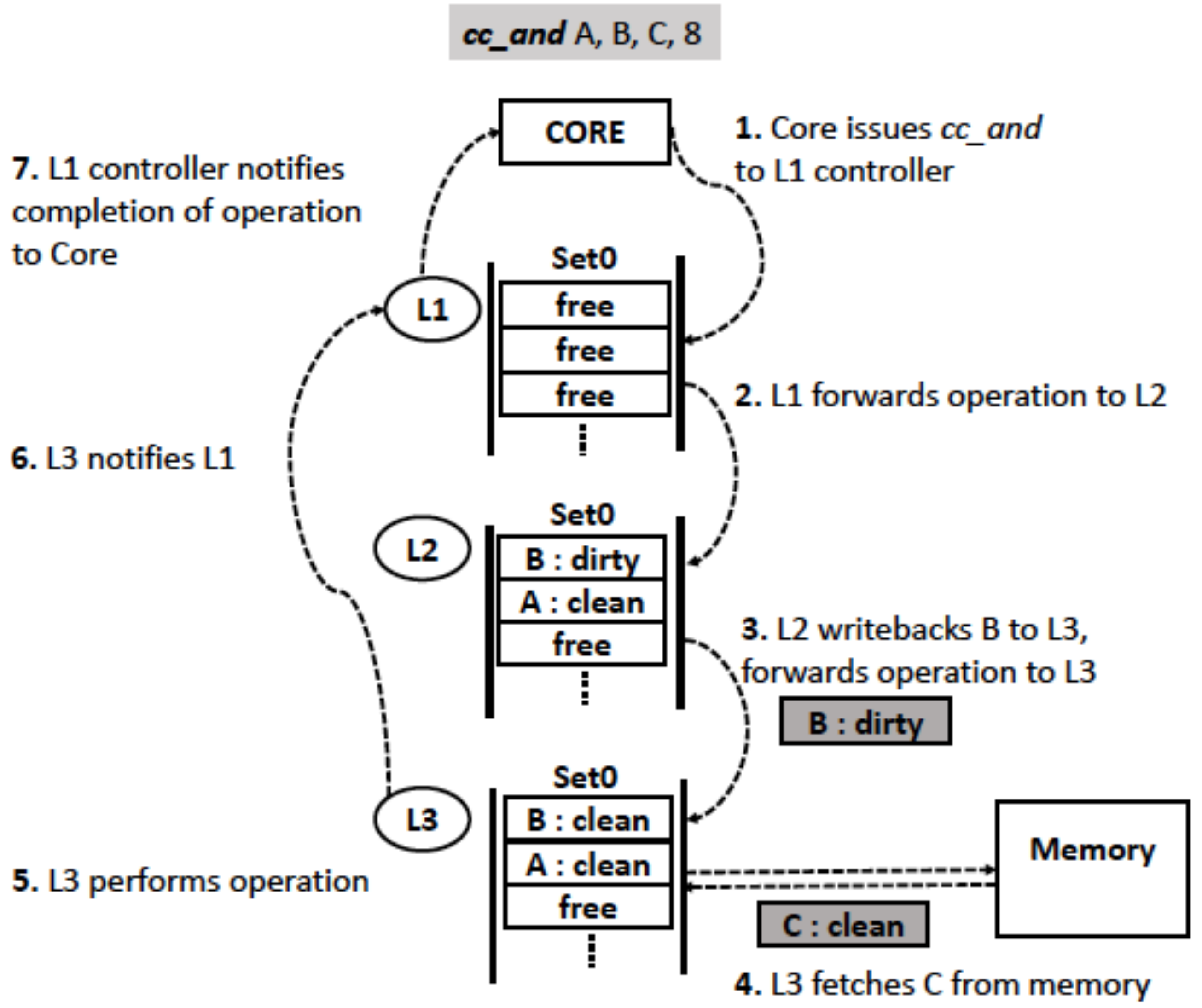
Opcode	Src1	Src2	Dest	Size	Description
cc_copy	a	-	b	n	$b[i] = a[i]$
cc_buz	a	-	-	n	$a[i] = 0$
cc_cmp	a	b	r	n	$r[i] = (a[i] == b[i])$
cc_search	a	k	r	n	$r[i] = (a[i] == k)$
cc_and	a	b	c	n	$c[i] = a[i] \;\&\; b[i]$
cc_or	a	b	c	n	$c[i] = (a[i] \;\|\|\; b[i])$
cc_xor	a	b	c	n	$c[i] = a[i] \oplus b[i]$
cc_clmulX	a	b	c	n	$c_i = \oplus(a[i] \;\&\; b[i])$
cc_not	a	-	b	n	$b[i] = !(a[i])$
a,b,c,k: addresses			r:register	$\forall i, i \in [1, n]$, X = [64/128/256]	

Proportion of energy (top) for bulk comparison operation and area (bottom).



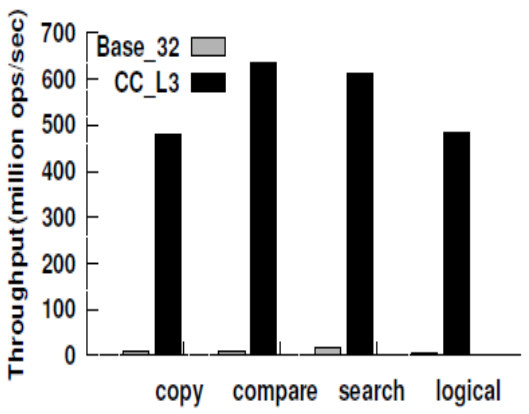
Red dot depicts logic capability.

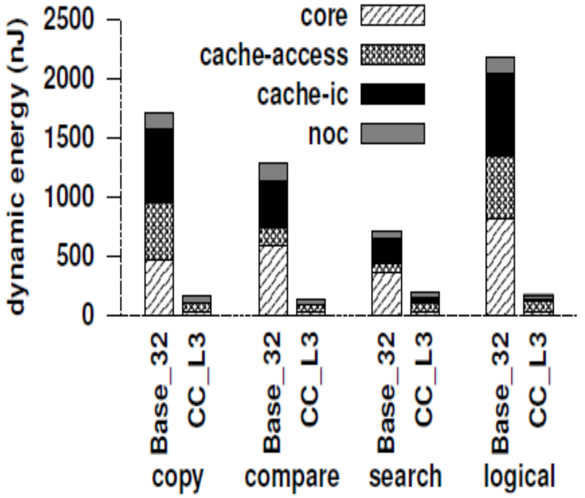
Compute Caches in operation

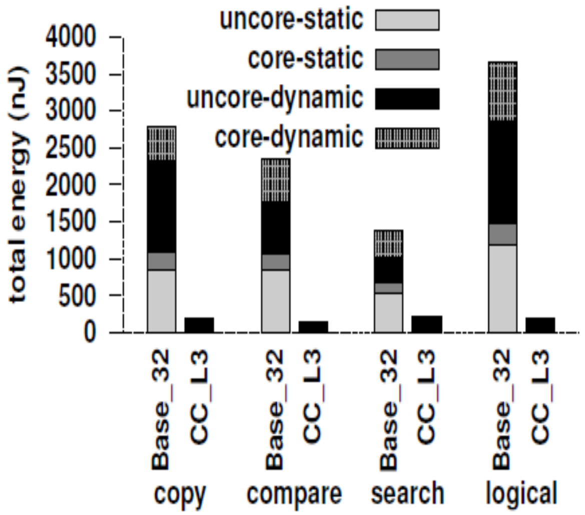


cc_and A, B, C, 8
CORE
1. Core issues *cc_and* to L1 controller
7. L1 controller notifies completion of operation to Core
Set0
L1
free
free
free
2. L1 forwards operation to L2
6. L3 notifies L1
Set0
L2
B : dirty
A : clean
free
3. L2 writebacks B to L3, forwards operation to L3
B : dirty
Set0
L3
B : clean
A : clean
free
Memory
5. L3 performs operation
C : clean
4. L3 fetches C from memory

Benefit of compute cache



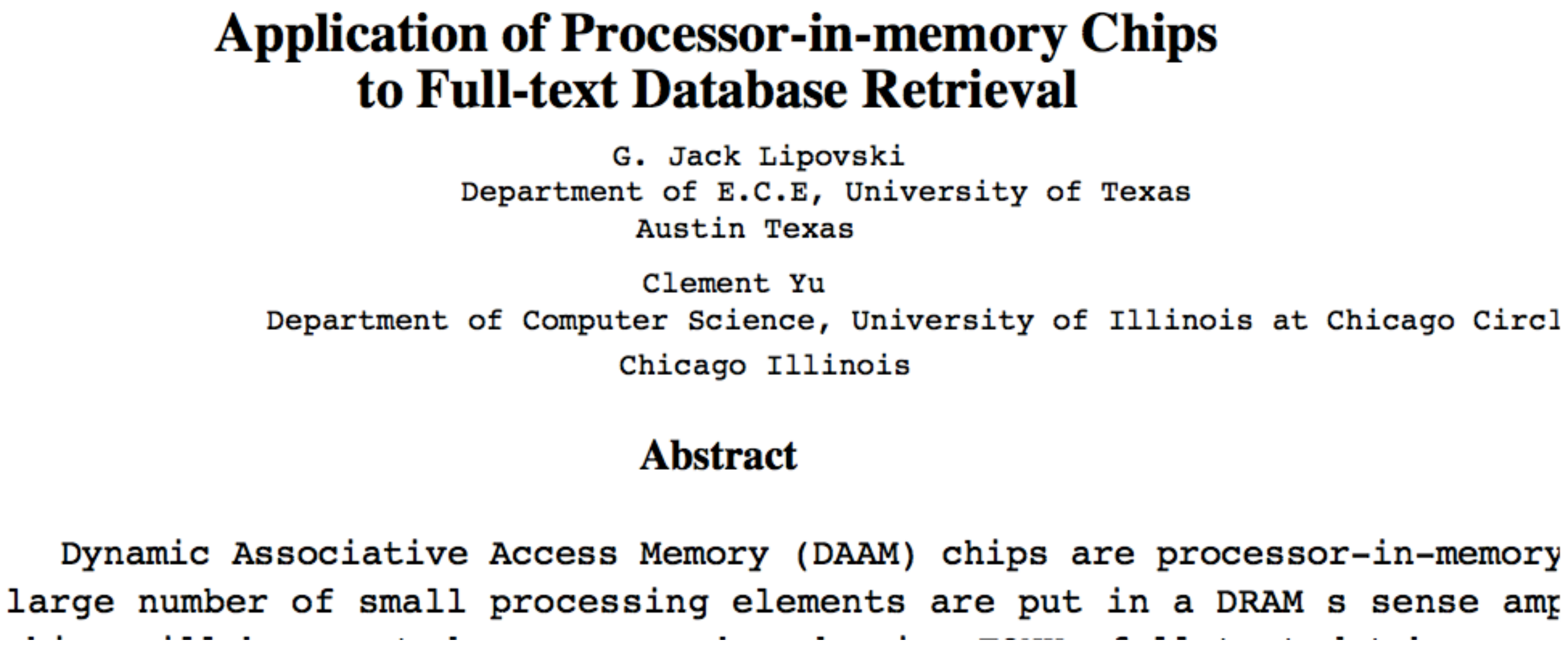




a) Throughput b) Dynamic energy c) Total energy

Base_32: supports 32-byte SIMD loads and stores
CC: compute cache

Dynamic Associative Memory - Lipovski



Application of Processor-in-memory Chips to Full-text Database Retrieval

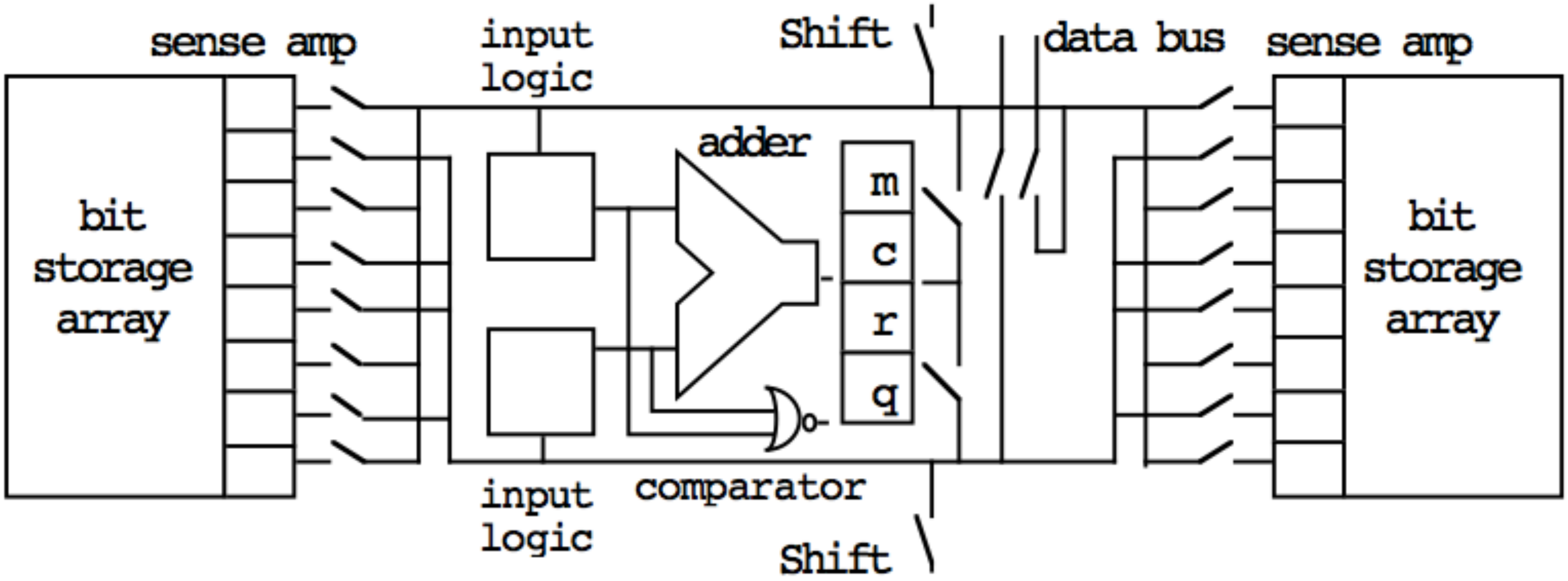
G. Jack Lipovski
Department of E.C.E, University of Texas
Austin Texas

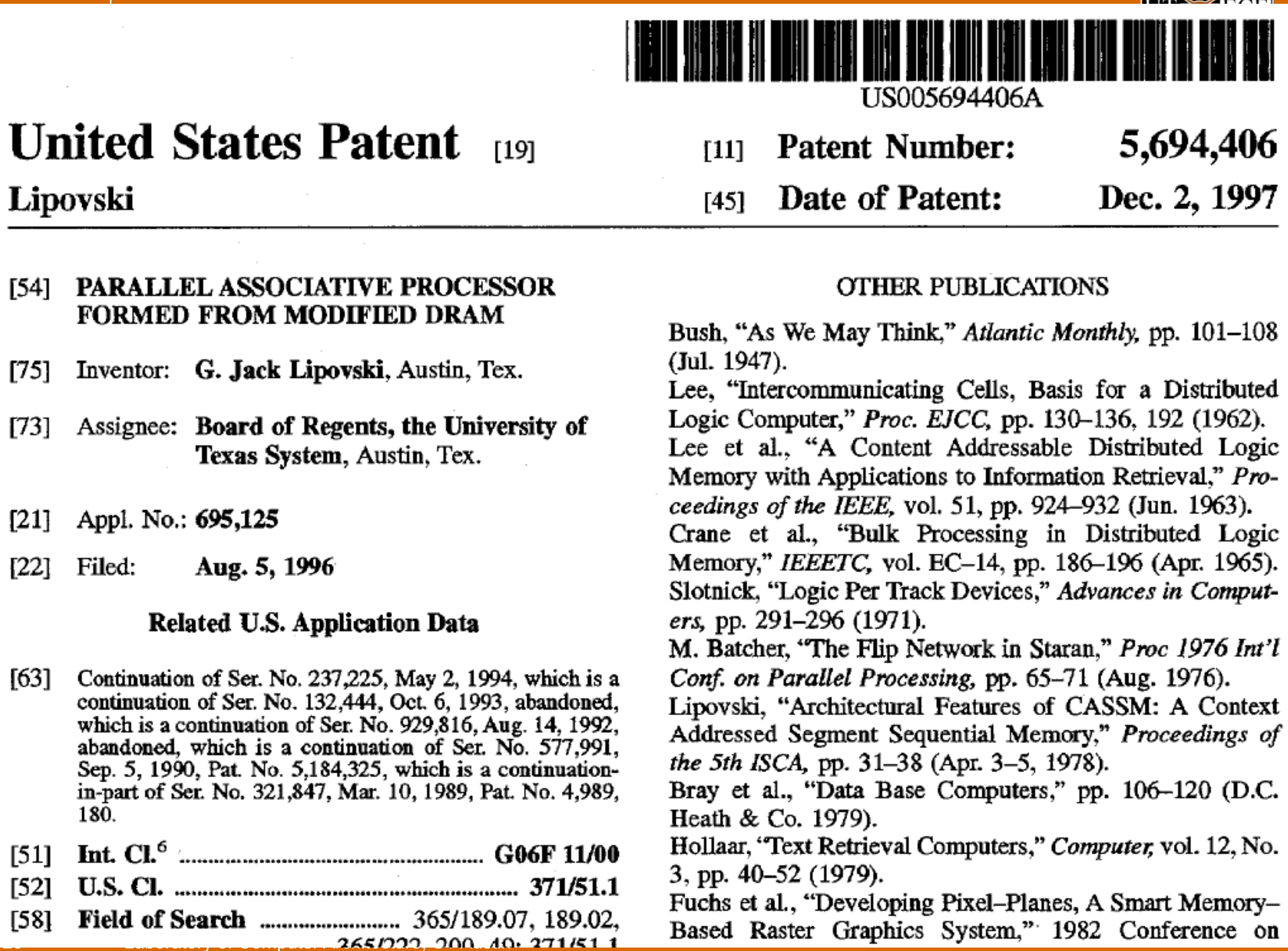
Clement Yu
Department of Computer Science, University of Illinois at Chicago Circl
Chicago Illinois

Abstract

Dynamic Associative Access Memory (DAAM) chips are processor-in-memory
large number of small processing elements are put in a DRAM s sense amp

DAAM Chips – Dynamic Associative Memory - Lipovski





US005694406A

United States Patent [19]

Lipovski

[11] Patent Number: 5,694,406

[45] Date of Patent: Dec. 2, 1997

[54] **PARALLEL ASSOCIATIVE PROCESSOR FORMED FROM MODIFIED DRAM**

[75] Inventor: **G. Jack Lipovski**, Austin, Tex.

[73] Assignee: **Board of Regents, the University of Texas System**, Austin, Tex.

[21] Appl. No.: **695,125**

[22] Filed: **Aug. 5, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 237,225, May 2, 1994, which is a continuation of Ser. No. 132,444, Oct. 6, 1993, abandoned, which is a continuation of Ser. No. 929,816, Aug. 14, 1992, abandoned, which is a continuation of Ser. No. 577,991, Sep. 5, 1990, Pat. No. 5,184,325, which is a continuation-in-part of Ser. No. 321,847, Mar. 10, 1989, Pat. No. 4,989,180.

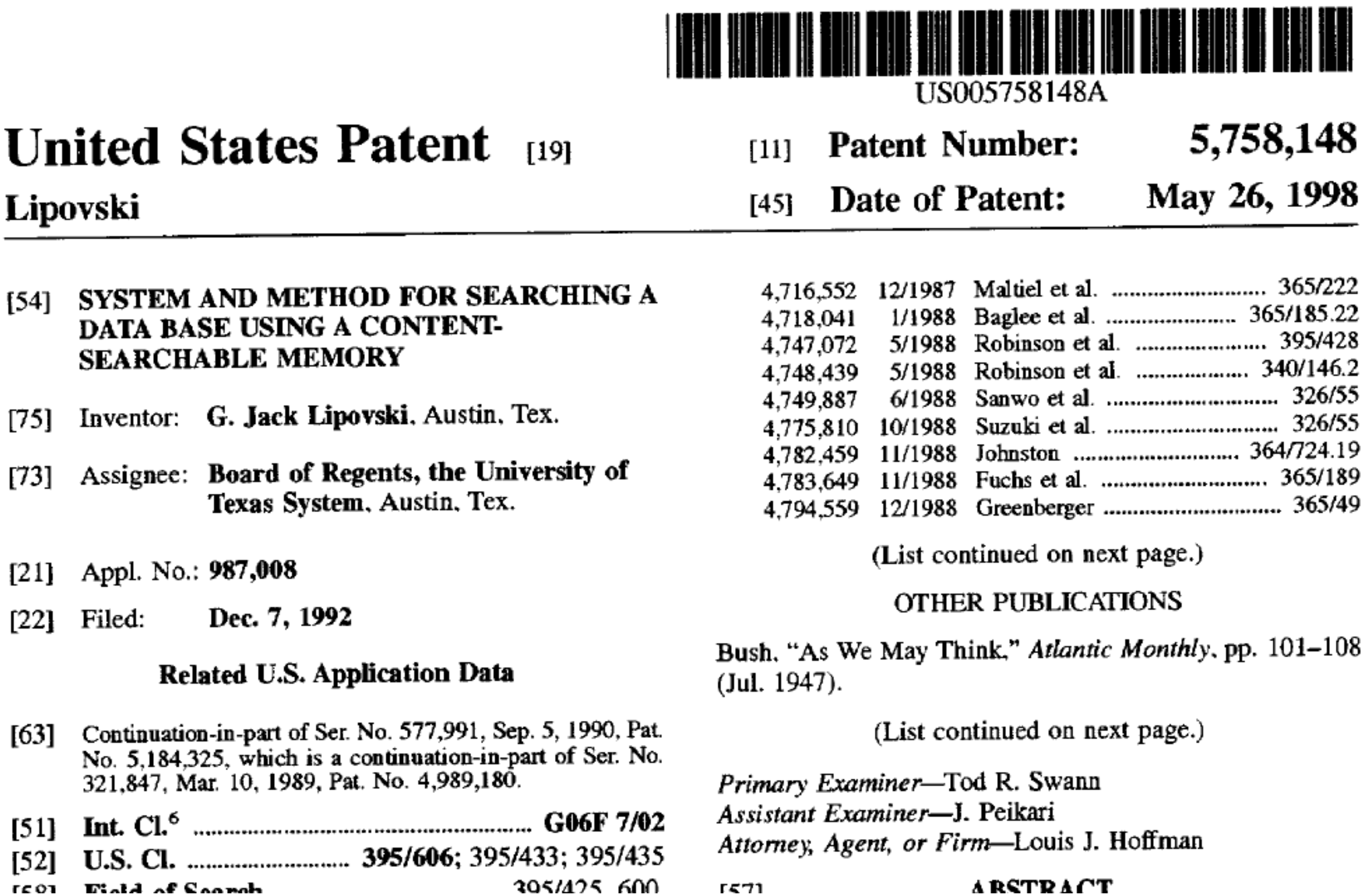
[51] **Int. Cl.**^{6} **G06F 11/00**

[52] **U.S. Cl.** **371/51.1**

[58] **Field of Search** 365/189.07, 189.02, 365/222, 200.49; 371/51.1

OTHER PUBLICATIONS

Bush, "As We May Think," *Atlantic Monthly*, pp. 101–108 (Jul. 1947).
Lee, "Intercommunicating Cells, Basis for a Distributed Logic Computer," *Proc. EJCC*, pp. 130–136, 192 (1962).
Lee et al., "A Content Addressable Distributed Logic Memory with Applications to Information Retrieval," *Proceedings of the IEEE*, vol. 51, pp. 924–932 (Jun. 1963).
Crane et al., "Bulk Processing in Distributed Logic Memory," *IEEETC*, vol. EC–14, pp. 186–196 (Apr. 1965).
Slotnick, "Logic Per Track Devices," *Advances in Computers*, pp. 291–296 (1971).
M. Batcher, "The Flip Network in Staran," *Proc 1976 Int'l Conf. on Parallel Processing*, pp. 65–71 (Aug. 1976).
Lipovski, "Architectural Features of CASSM: A Context Addressed Segment Sequential Memory," *Proceedings of the 5th ISCA*, pp. 31–38 (Apr. 3–5, 1978).
Bray et al., "Data Base Computers," pp. 106–120 (D.C. Heath & Co. 1979).
Hollaar, "Text Retrieval Computers," *Computer*, vol. 12, No. 3, pp. 40–52 (1979).
Fuchs et al., "Developing Pixel–Planes, A Smart Memory–Based Raster Graphics System," 1982 Conference on



US005758148A

United States Patent [19]
Lipovski

[11] Patent Number: 5,758,148
[45] Date of Patent: May 26, 1998

[54] SYSTEM AND METHOD FOR SEARCHING A DATA BASE USING A CONTENT-SEARCHABLE MEMORY

[75] Inventor: **G. Jack Lipovski**, Austin, Tex.

[73] Assignee: **Board of Regents, the University of Texas System**, Austin, Tex.

[21] Appl. No.: **987,008**

[22] Filed: **Dec. 7, 1992**

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 577,991, Sep. 5, 1990, Pat. No. 5,184,325, which is a continuation-in-part of Ser. No. 321,847, Mar. 10, 1989, Pat. No. 4,989,180.

[51] **Int. Cl.**^{6} **G06F 7/02**
[52] **U.S. Cl.** **395/606**; 395/433; 395/435
[58] **Field of Search** 395/425, 600

4,716,552	12/1987	Maltiel et al.	365/222
4,718,041	1/1988	Baglee et al.	365/185.22
4,747,072	5/1988	Robinson et al.	395/428
4,748,439	5/1988	Robinson et al.	340/146.2
4,749,887	6/1988	Sanwo et al.	326/55
4,775,810	10/1988	Suzuki et al.	326/55
4,782,459	11/1988	Johnston	364/724.19
4,783,649	11/1988	Fuchs et al.	365/189
4,794,559	12/1988	Greenberger	365/49

(List continued on next page.)

OTHER PUBLICATIONS

Bush, "As We May Think," *Atlantic Monthly*, pp. 101–108 (Jul. 1947).

(List continued on next page.)

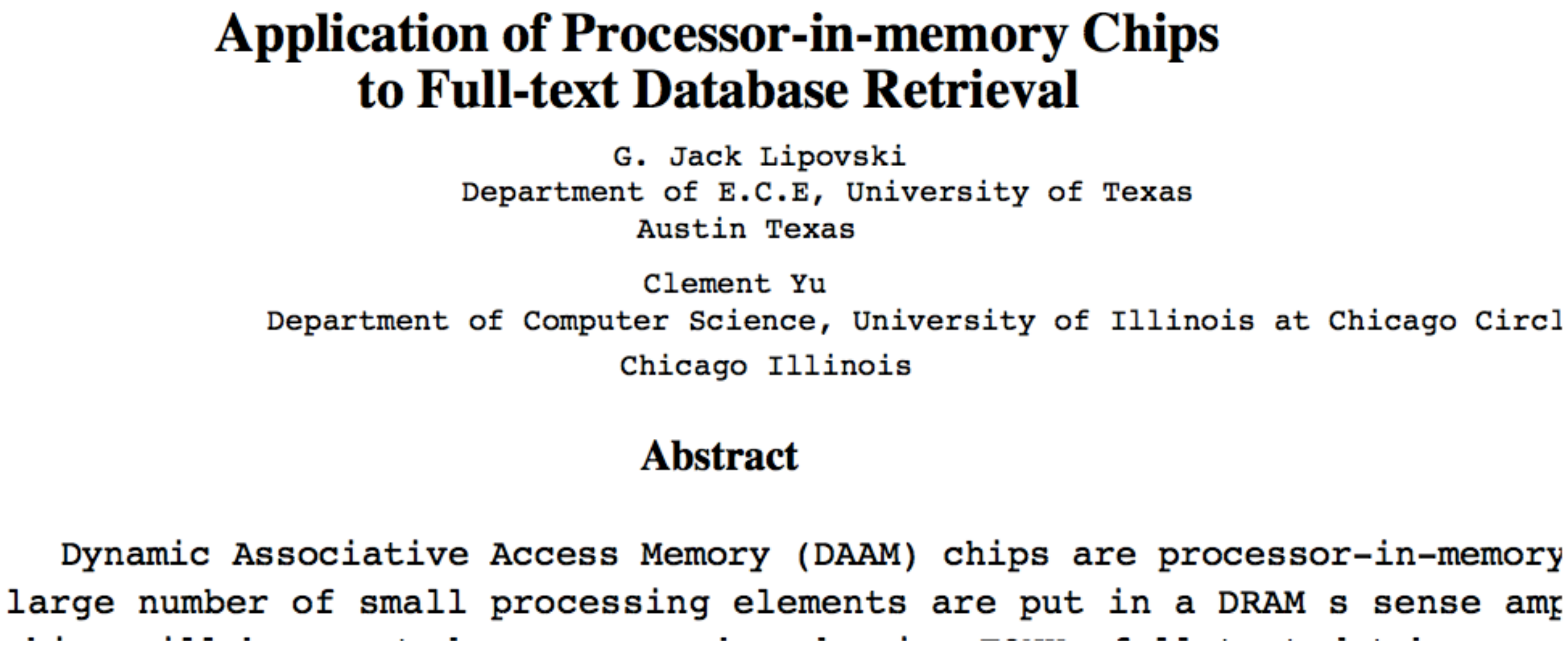
Primary Examiner—Tod R. Swann
Assistant Examiner—J. Peikari
Attorney, Agent, or Firm—Louis J. Hoffman

[57] ABSTRACT

Lipovski Patents

- G. J. Lipovski, System and Method for Searching a Data Base Using a Content Searchable Memory, May 1998
- K. Liu, G. J. Lipovski and C. Yu, “Efficient Processing of Queries in Full-text Search Using Associative Memory”
- G. J. Lipovski, “Parallel Computer Within Dynamic Random Access Memory”, June 1998
- G. J. Lipovski, “Dynamic Systolic Associative Memory Chip”, International Symposium on Application Specific Array Processors, pp. 481-492, 1990, Sep
- G. J. Lipovski, “Dynamic Associative Memory with Login-In-Refresh, January 1991

Dynamic Associative Memory - Lipovski



Application of Processor-in-memory Chips
to Full-text Database Retrieval

G. Jack Lipovski
Department of E.C.E, University of Texas
Austin Texas

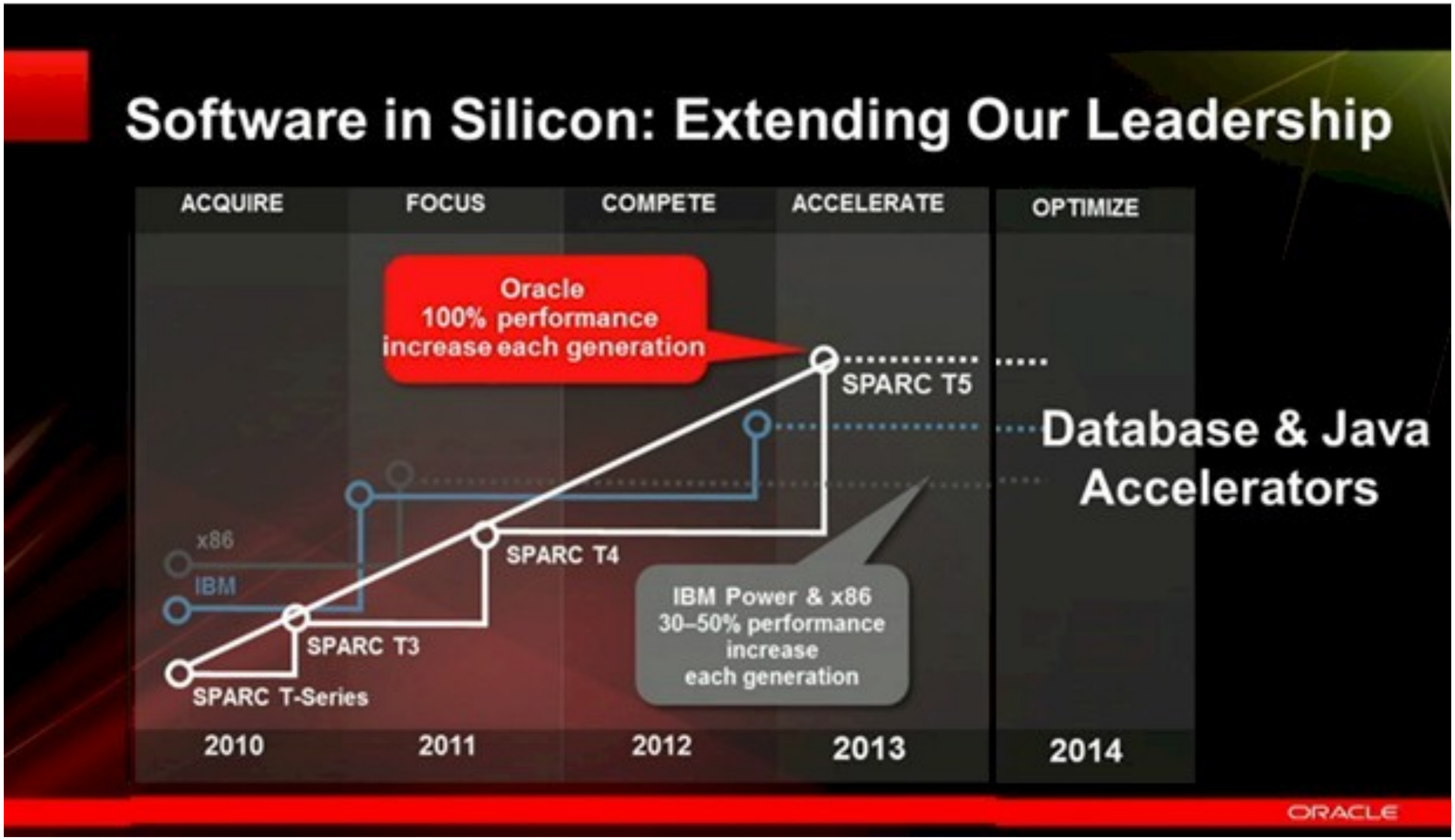
Clement Yu
Department of Computer Science, University of Illinois at Chicago Circl
Chicago Illinois

Abstract

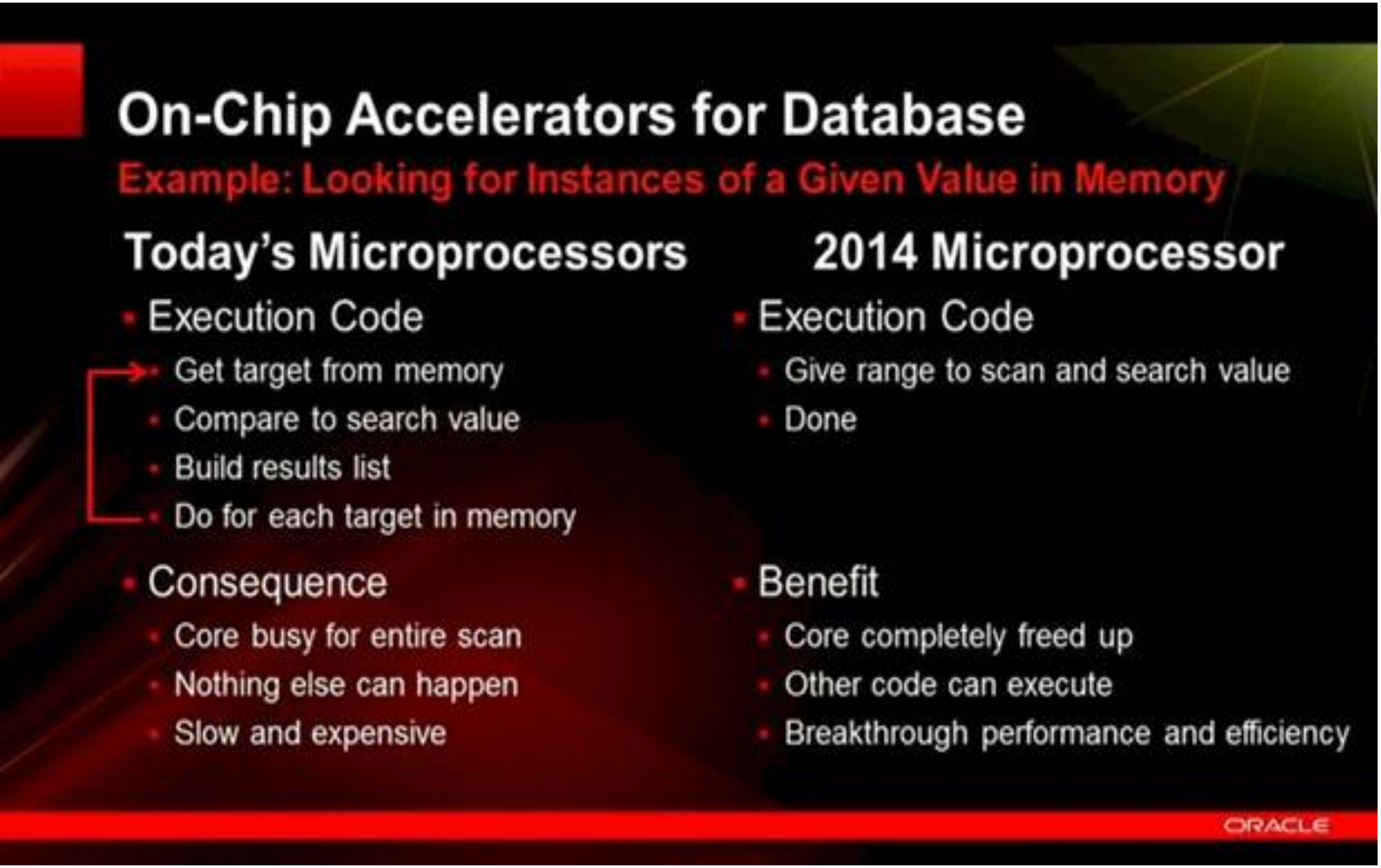
Dynamic Associative Access Memory (DAAM) chips are processor-in-memory
large number of small processing elements are put in a DRAM s sense amp

1999 IEEE International Workshop on Memory Technology, Design and Testing

News from Oracle at the Hot Chips Symposium ‘13



Oracle at Hot Chips 2013 Conference



On-Chip Accelerators for Database

Example: Looking for Instances of a Given Value in Memory

Today's Microprocessors

- Execution Code
  - Get target from memory
  - Compare to search value
  - Build results list
  - Do for each target in memory
- Consequence
  - Core busy for entire scan
  - Nothing else can happen
  - Slow and expensive

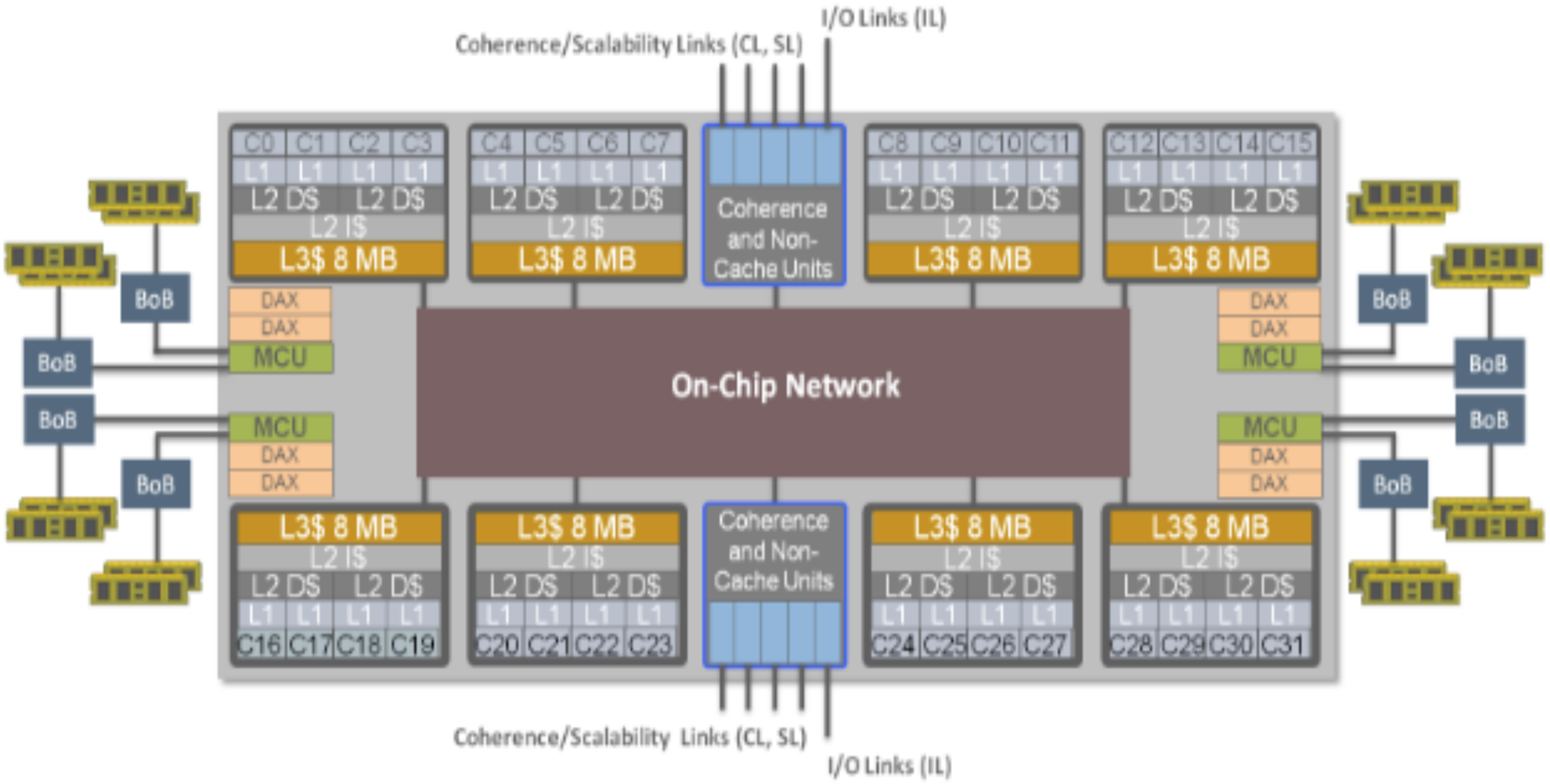
2014 Microprocessor

- Execution Code
  - Give range to scan and search value
  - Done
- Benefit
  - Core completely freed up
  - Other code can execute
  - Breakthrough performance and efficiency

ORACLE

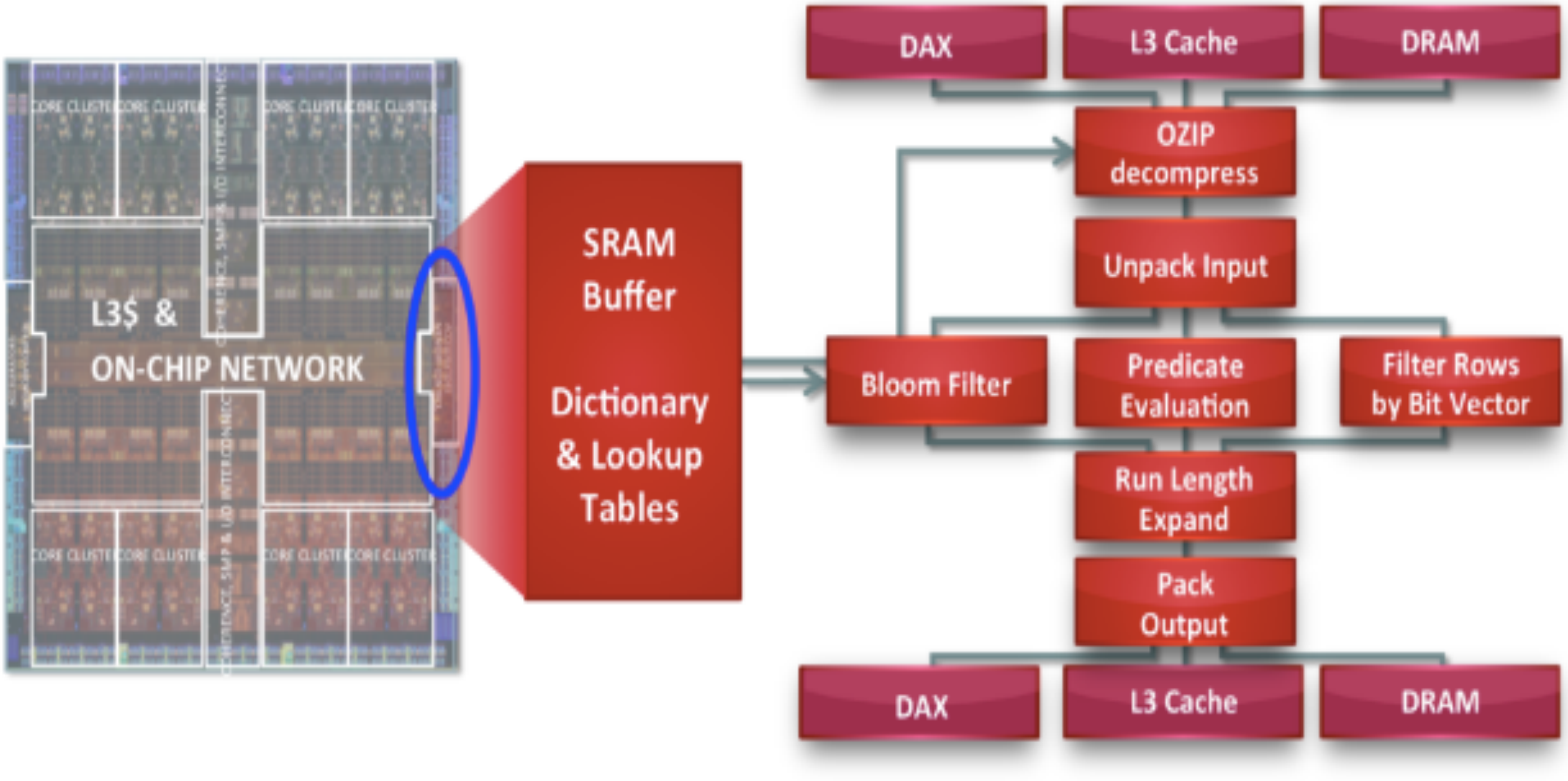
SPARC M7 Data Accelerator

- **SWIS (Software in Silicon)**



SPARC M7 Data Accelerator

- **SWIS (Software in Silicon)**



SPARC DAX Accelerator Pipeline

The first stage is decompression.

Uncompressed data is then unpacked (extract)

Next joins (called Bloom Filter); evaluates conditions such as less than, greater than, or equal to; or performs matches based on an existing result of bit vectors.

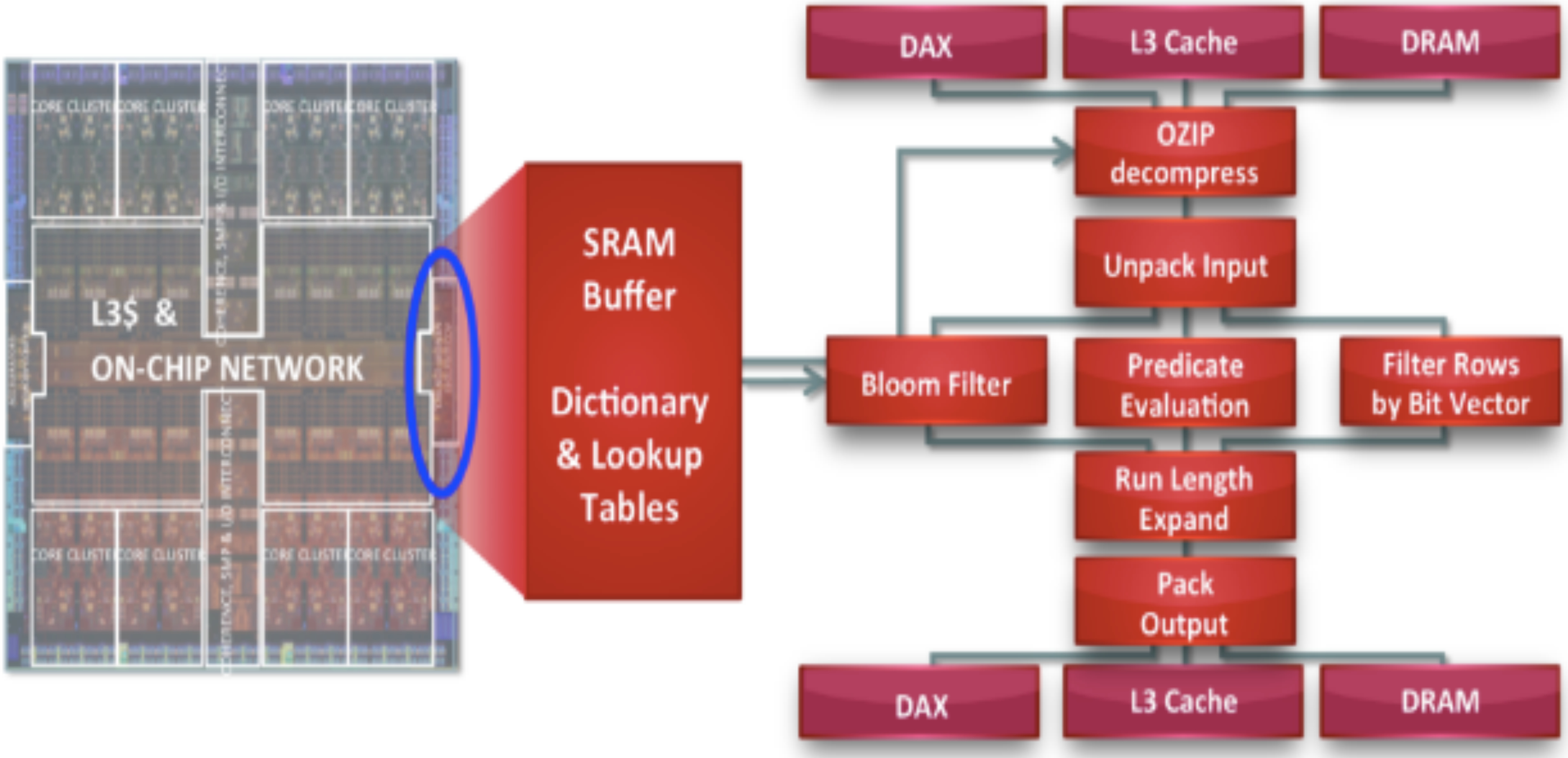
Resulting data is expanded via repeating decompressions.

Then output is packed (if it was expanded or unpacked initially) can also be pipelined either into another DAX / L3 cache /DRAM

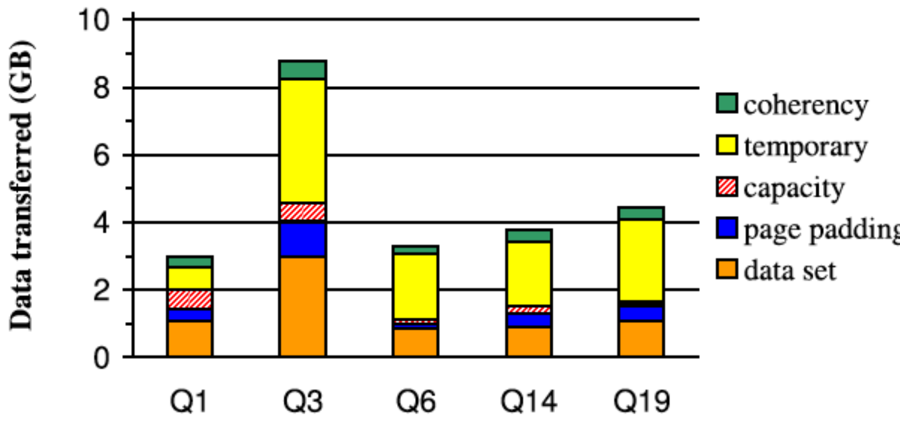
Very specialized pipeline; equivalent to having 32 extra cores for queries and 64 extra cores for decompression; very cost effective.

SPARC M7 Data Accelerator

- **SWIS (Software in Silicon)**

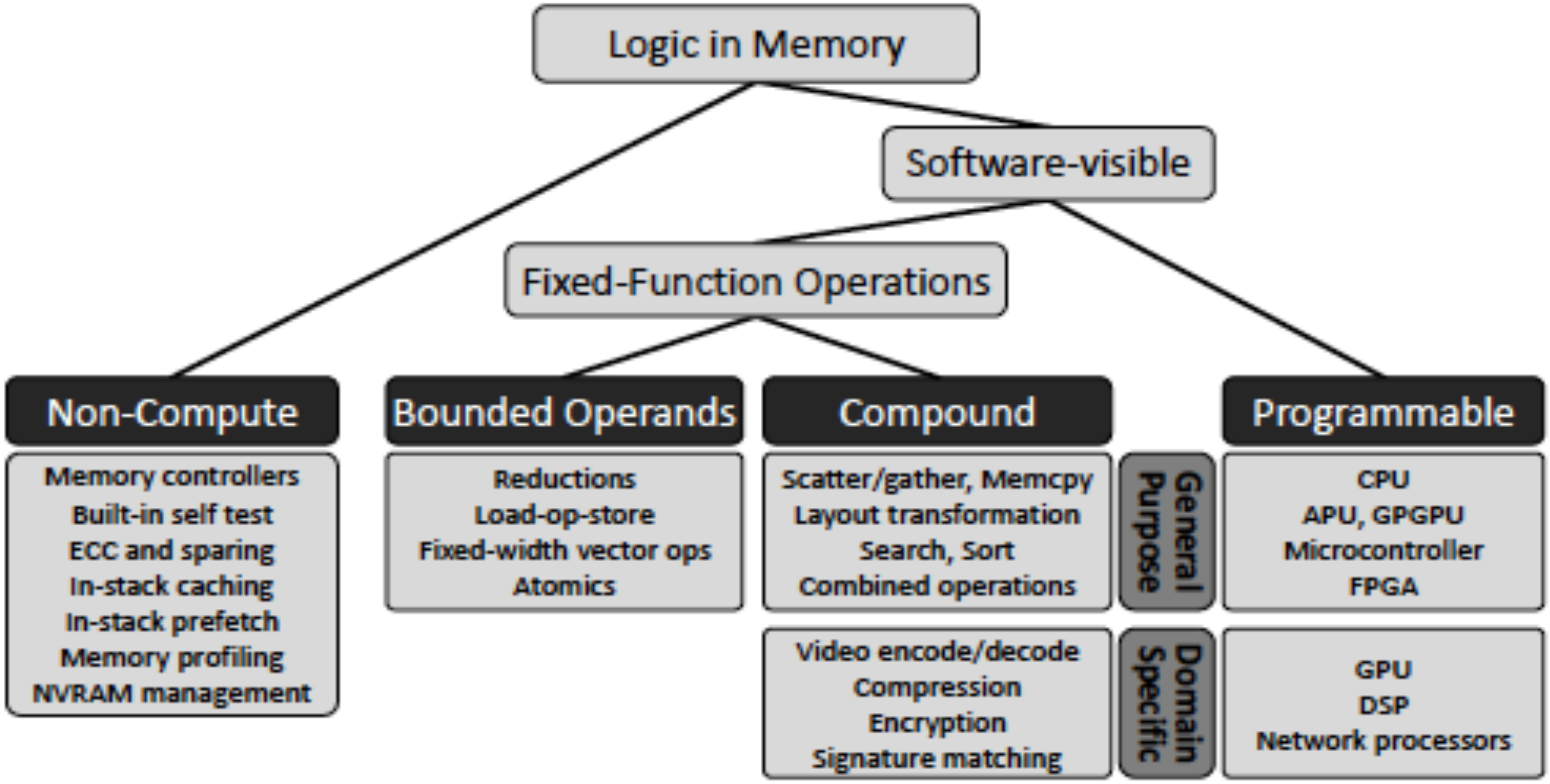


COMPUTE IN-TRANSIT



- **Avoid Allocating and St in DRAM**
- **Simply move data to the Consuming Element**
- **Systolic Processing Concepts**
- **Perform an Operation as part of data transfer**
- **Fixed function or limited programmable**
- **Scatter-gather, Address Mapping Transformations, Reductions, Projections**
- **Simple integer units or small overhead hardware**
- **Programming models to support**

PIM Taxonomy



A Processing-in-Memory Taxonomy and a Case for Studying Fixed-function PIM

Gabriel H. Loh Nuwan Jayasena Mark H. Oskin Mark Nutter David Roberts Mitesh Meswani Dong Ping Zhang Mike Ignatowski
AMD Research – Advanced Micro Devices, Inc.
{gabriel.loh, nuwan.jayasena, mark.oskin, mark.nutter, david.roberts, mitesh.meswani, dongping.zhang, mike.ignatowski}@amd.com

CHIPS THAT REMEMBER AND COMPUTE (ISCA 97 ISCA 98)

IRAM (Berkeley)

EXECUBE (Kogge)

FlexRAM (Torrellas)
Yan Solihin's HPCA 2001 Paper

DAAM Memory Chips (Lipovski)

Active Pages (Oskin, ISCA 1998)

PIM Enabling Instructions (PEIs) (Onur Mutlu ISCA 2015)

Active Disk (ASPLOS 1998)

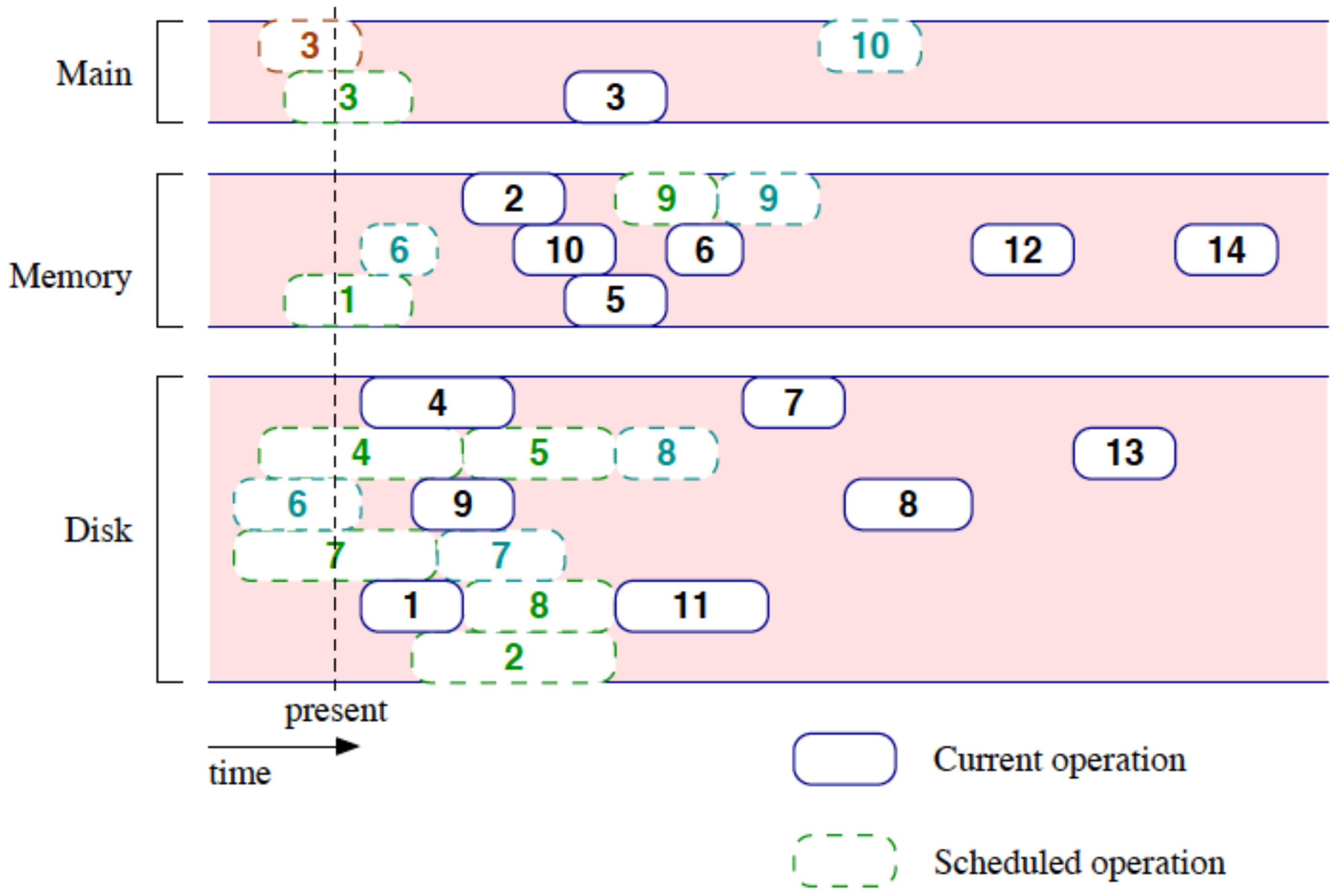
Challenges for PIMs

- **Locality and Data Reuse is the main reason why computing in memory often does not yield performance benefits**
- **Memory is still slow**
- **Die-stacked DRAM is still DRAM and slow like DRAM**
- **If data reuse, temporary creation and reuse, challenging to get performance from integrating compute with slow memory**

Cache energy (pJ) per cache-block (64-byte) (Compute Cache Paper)

Cache	Write	Read	Cmp	Copy	Search	Not	Logic
L3	2852	2452	840	1340	3692	1340	1672
L2	1154	802	242	608	1396	608	704
L1	375	295	186	324	561	324	387

Task Mapping Challenges



CONCLUDING REMARKS

- Computing in Situ offers quite some potential
- Quite a lot of Passion went into PIM in its earlier waves
- Several technology Challenges and Market factors
- Computing in Transit is interesting
- Proliferation of Accelerators
- Instead of Move as little as possible, focus on move what benefits from the move
- Be in the right place at the right time
- Software Support is key for Success

Thank You! Questions?

Laboratory for Computer Architecture (LCA)
The University of Texas at Austin
lca.ece.utexas.edu