Towards General-Purpose Neural Network Computing

Schuyler Eldridge[1] Amos Waterland[2] Margo Seltzer[2]
Jonathan Appavoo[3] Ajay Joshi[1]

[1]Boston University Department of Electrical and Computer Engineering

[2]Harvard University School of Engineering and Applies Sciences

[3]Boston University Department of Computer Science

24th International Conference on Parallel Architectures and Compilation Techniques

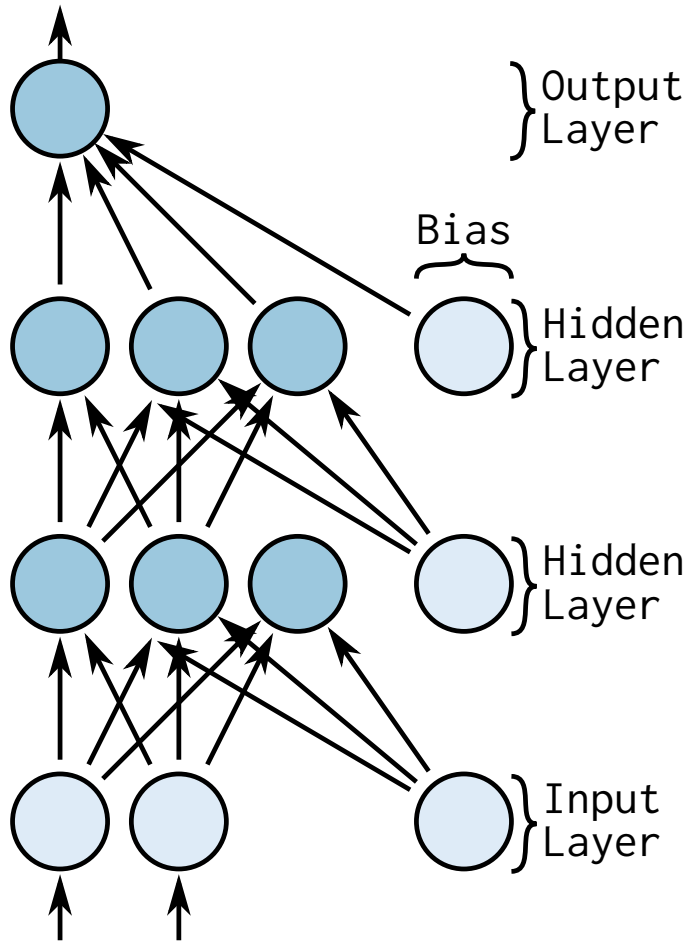
Why Do We Care About Neural Networks?

"Good" solutions for hard problems

- Capable of learning

Neural networks, again?

- The neural network hype cycle has been a bumpy ride
- Modern, resurgent interest in neural networks is driven by:
  - Big, real-world data sets
  - "Free" availability of transistors
  - Use of accelerators
  - The need for continued performance improvements



Neural Network Computing is Hot (Again)

Existing approaches

- Dedicated neural network/vector processors from the 1990s [1]
- Ongoing NPU work for approximate computing [2, 3, 4]
- High performance deep neural network architectures [5, 6]

Neural networks as primitives

- We treat neural networks as an application primitive

[1] J. Wawrzynek et al., "Spert-II: a vector microprocessor system," *Computer*, vol. 29, no. 3, pp. 79–86, Mar 1996.

[2] H. Esmaeilzadeh et al., "Neural acceleration for general-purpose approximate programs," in *MICRO*, 2012.

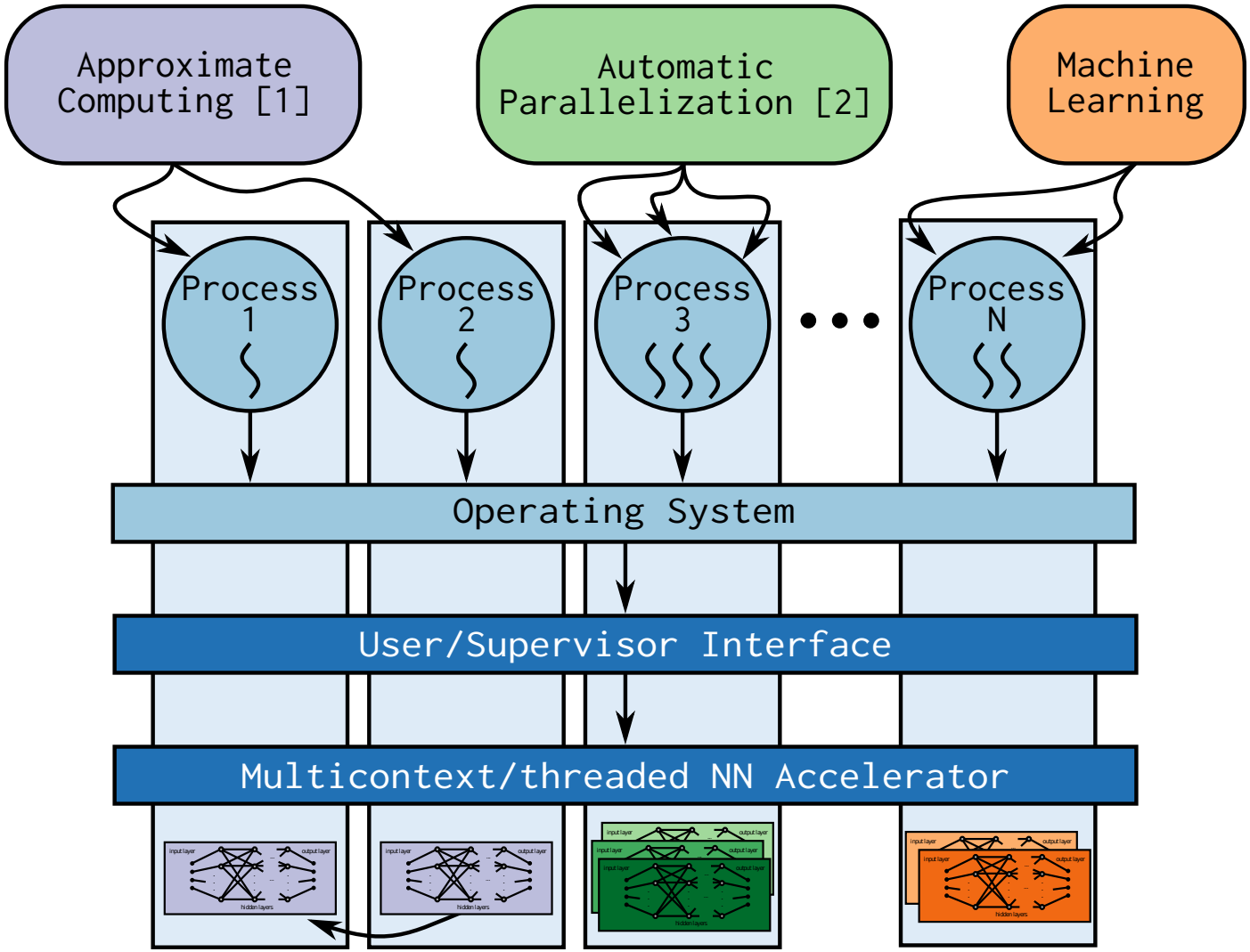
[3] R. St. Amant, et al., "General-purpose code acceleration with limited-precision analog computation," in *ISCA*, 2014.

[4] T. Moreau, et al., "Snnap: Approximate computing on programmable socs via neural acceleration," in *HPCA*, 2015.

[5] T. Chen, et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *ASPLOS*, 2014.

[6] Z. Du, et al., "Shidiannao: shifting vision processing closer to the sensor," in *ISCA*, 2015.

Our Vision of the Future of Neural Network Computing



[1] H. Esmaeilzadeh et al., "Neural acceleration for general-purpose approximate programs," in *MICRO*, 2012.

[2] A. Waterland et al. "Asc: Automatically scalable computation," in *ASPLOS*, 2014.

Our Contributions Towards this Vision

X-FILES: Hardware/Software Extensions

E**x**tensions **f**or the **I**ntegration of Machine **L**earning in **E**veryday **S**ystems

- A defined user and supervisor interface for neural networks
- This includes supervisor architectural state (hardware)

DANA: A Possible Multi-Transaction Accelerator

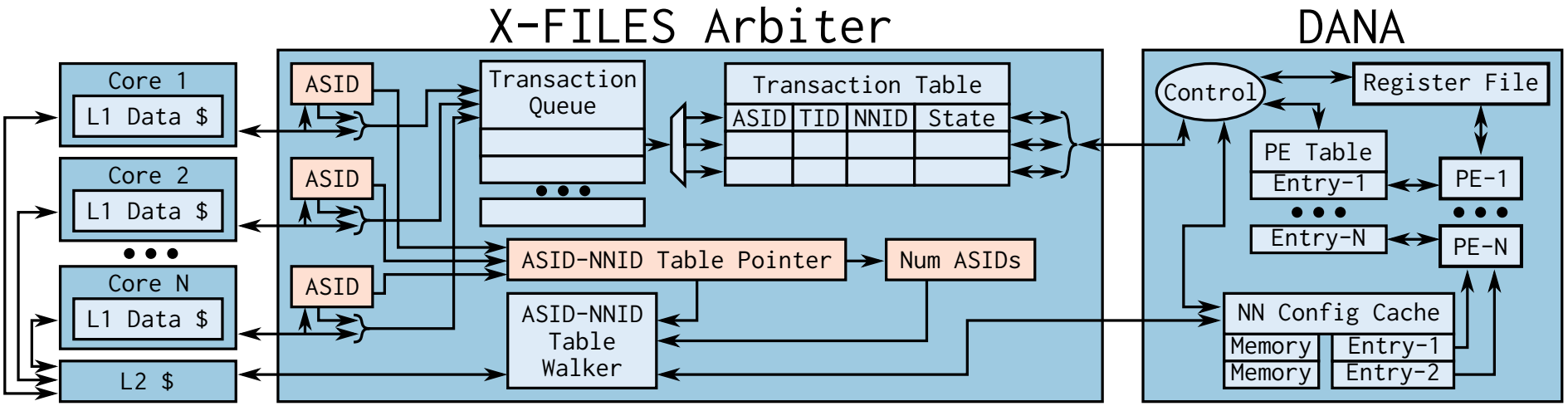
Dynamically **A**llocated **N**eural Network **A**ccelerator

- An accelerator aligning with our multi transaction vision

I apologize for the names

- There is no association with files or filesystems
- X-FILES is plural (like extensions)

An Overview of X-FILES/DANA Hardware



Components

- General purpose cores
- Transaction storage
- A backend accelerator that “executes” transactions
- Supervisor resources for memory safety
- Dedicated memory interface

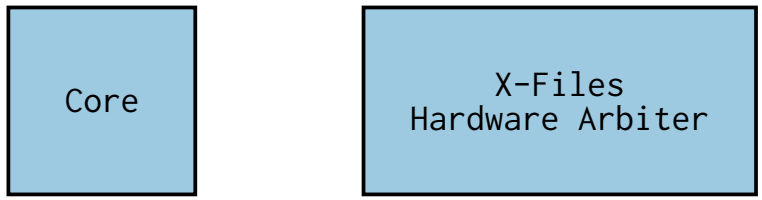
At the User Level We Deal With “Transactions”

Neural Network Transactions

A transaction encapsulates a request by a process to compute the output of a specific neural network for a provided input

User Transaction API:

- `newWriteRequest`
- `writeData`
- `readDataPoll`



Identifiers

`NNID`: Neural Network ID
`TID`: Transaction ID

Core/Accelerator Interface

- We use the RoCC interface of the Rocket RISC-V microprocessor [1, 2]

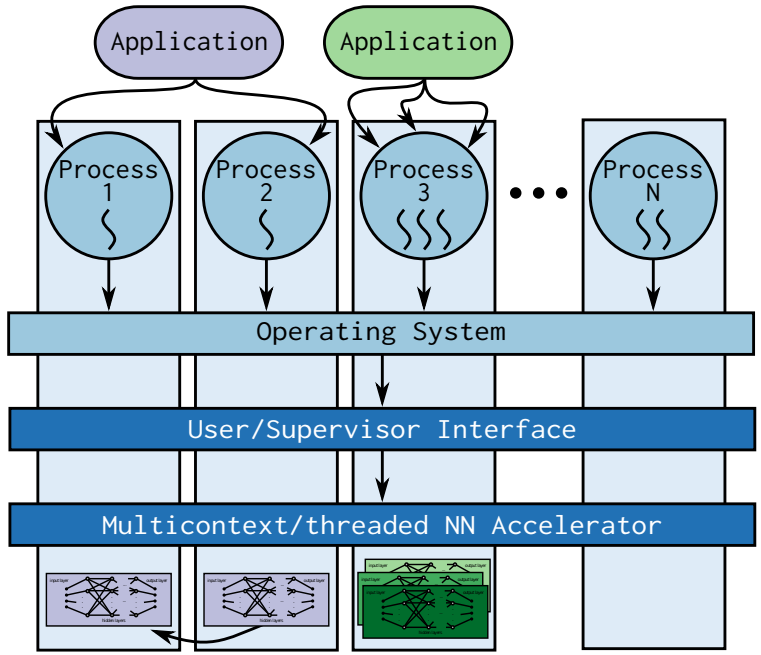
[1] A. Waterman et al., “The risc-v instruction set manual, volume i: User-level isa, version 2.0,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, May 2014.

[2] A. Waterman et al.,, “The risc-v instruction set manual volume ii: Privileged architecture version 1.7,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-49, May 2015.

At the Supervisor Level We Deal With Address Spaces

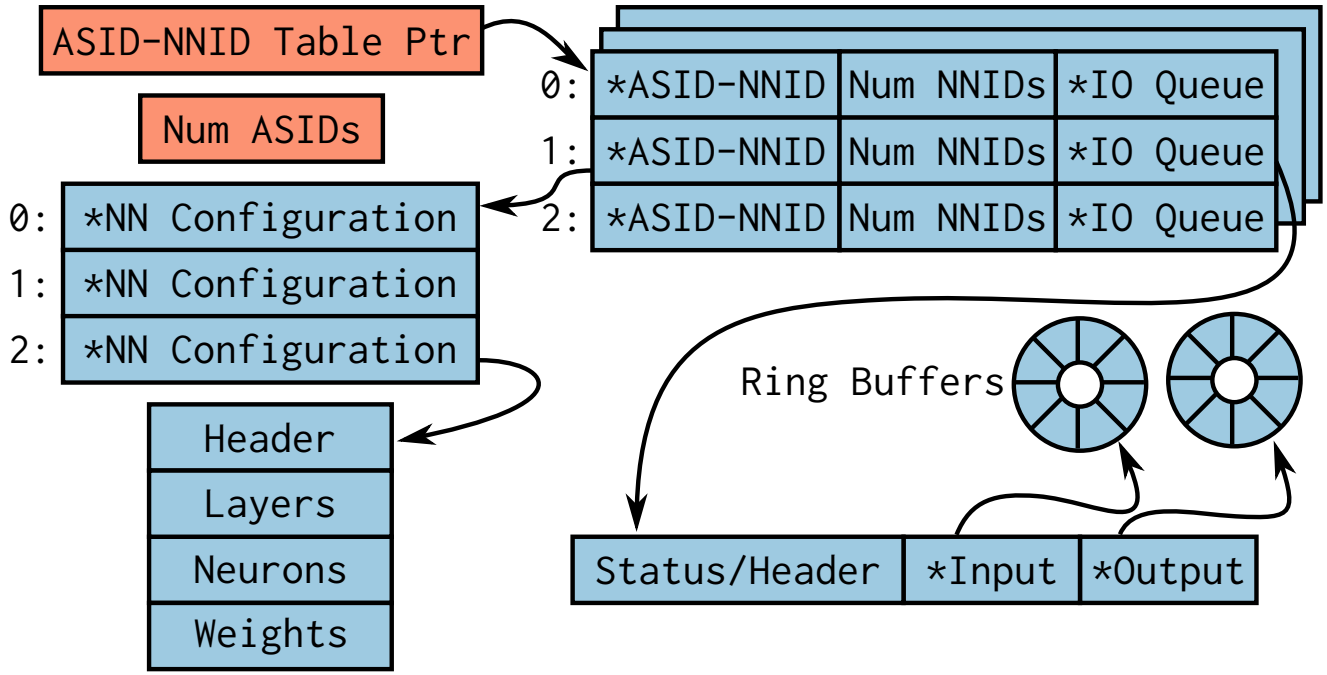
Use cases:

- Single transaction
- Multiple transactions
- Sharing of networks
- Multiple networks



- We maintain the state of executing transactions
- We group transactions into Address Spaces
- Address Spaces are identified by an OS-managed `ASID`
- Each `ASID` defines the set of accessible networks
- Networks can be shared transparently if the OS allows this

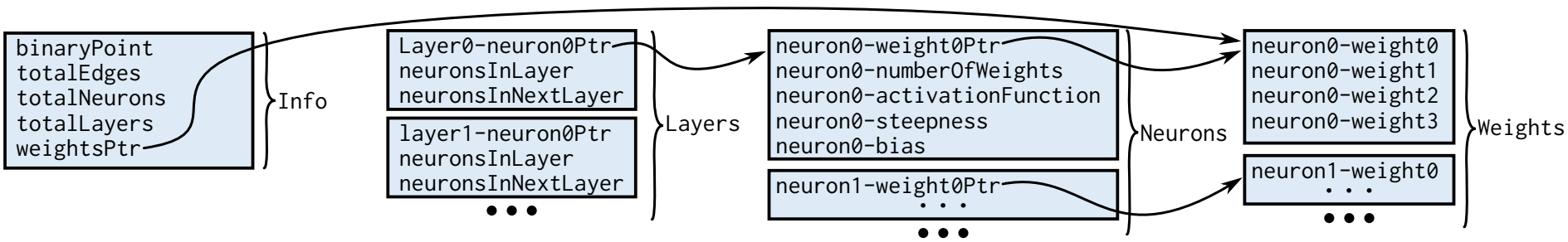
An ASID–NNID Table Enables NNID Dereferencing



ASID–NNID Table

- The OS establishes and maintains the ASID–NNID Table
- We assign ASIDs and NNIDs sequentially
- The ASID–NNID Table contains an optional asynchronous memory interface

A Compact Binary Neural Network Configuration

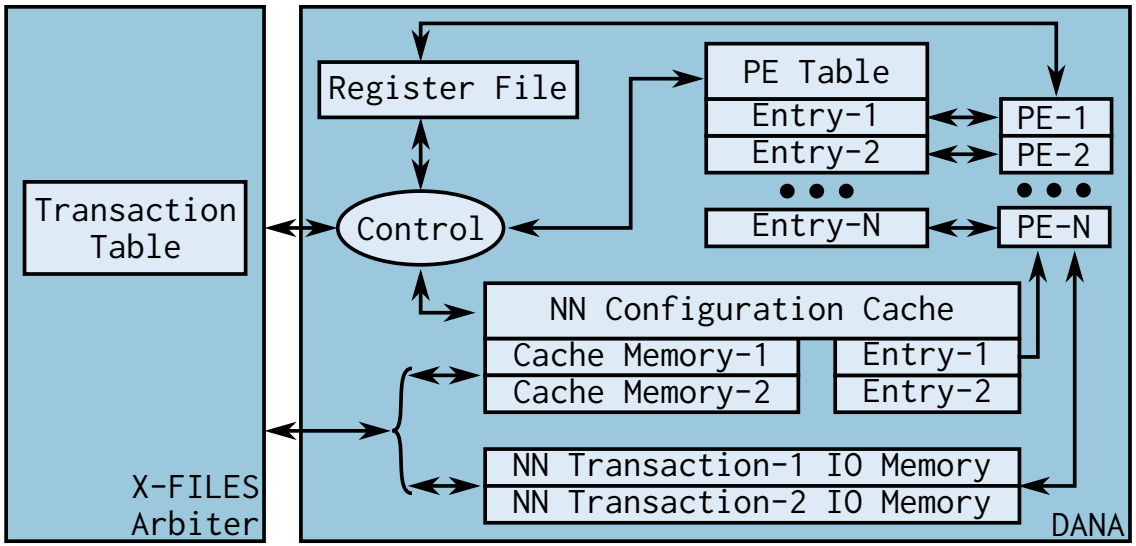


We condense the normal FANN neural network data structure

- We use a reduced configuration from the Fast Artificial Neural Network (FANN) library [1] containing:
  - Global information
  - Per-layer information
  - Per-neuron information
  - Per-neuron weights

[1] S. Nissen, “Implementation of a fast artificial neural network library (fann),” Department of Computer Science University of Copenhagen (DIKU), Tech. Rep., 2003.

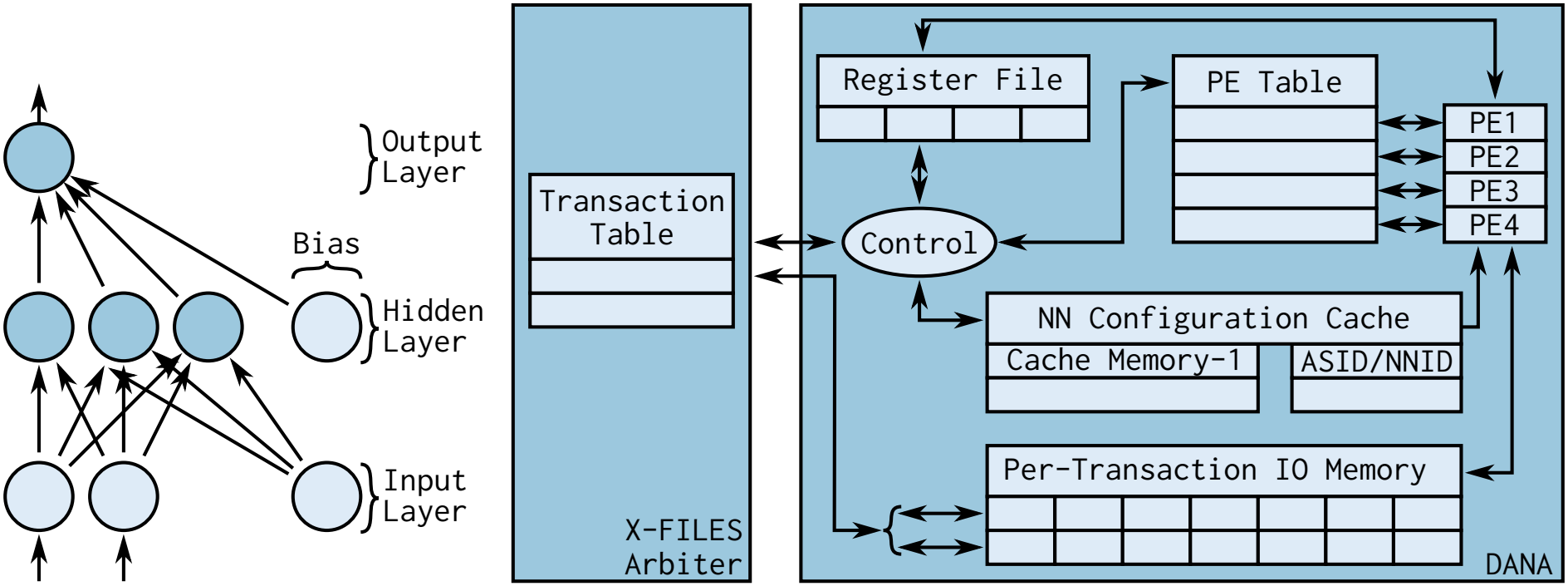
DANA: An Example Multi-Transaction Accelerator



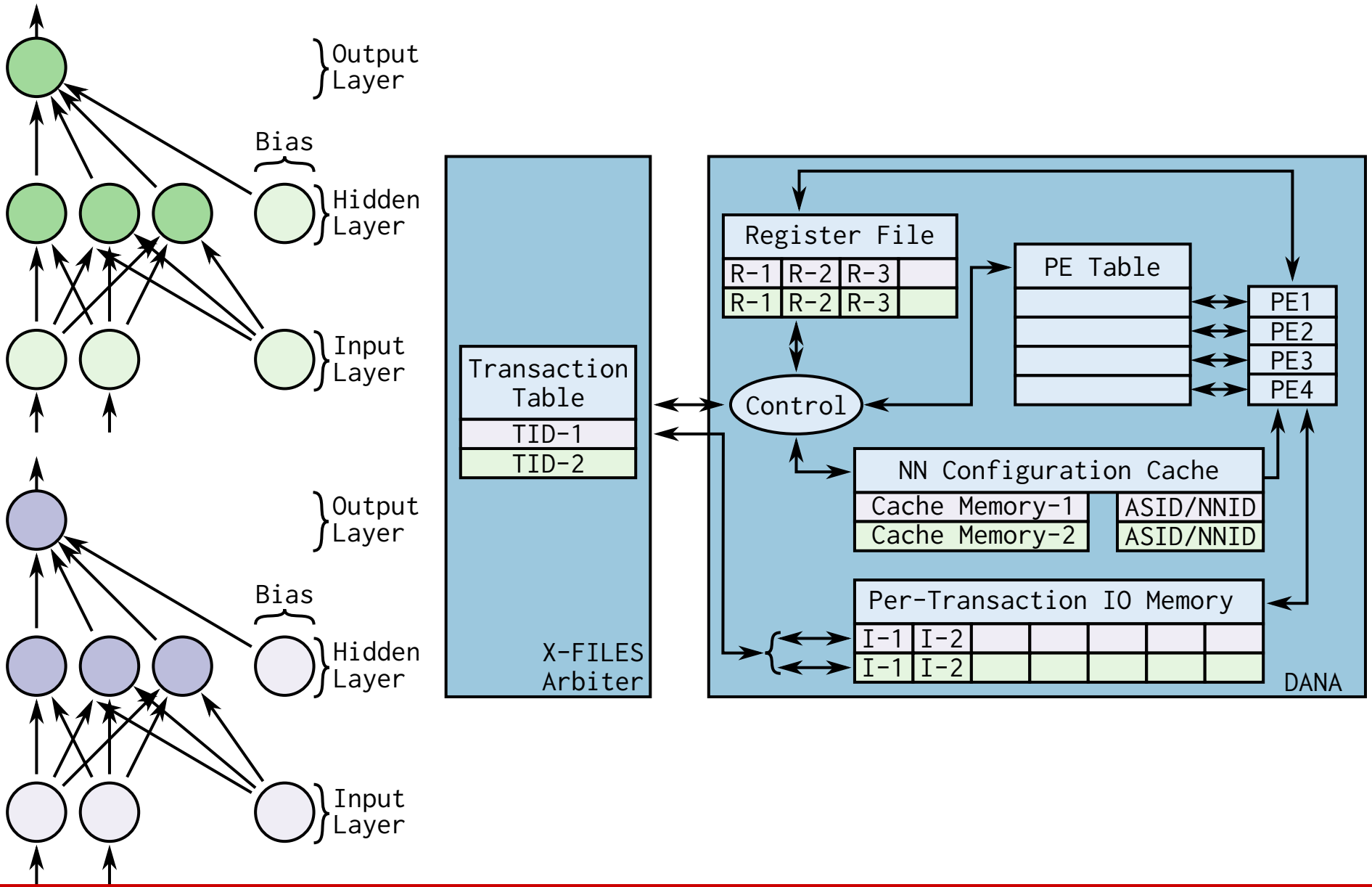
Components

- **Control logic determines actions given transaction state**
- **Network configurations are stored in a Configuration Cache**
- **Per-transaction IO Memory stores inputs and outputs**
- **A Register File stores intermediate outputs**
- **Logical neurons are mapped to Processing Elements**

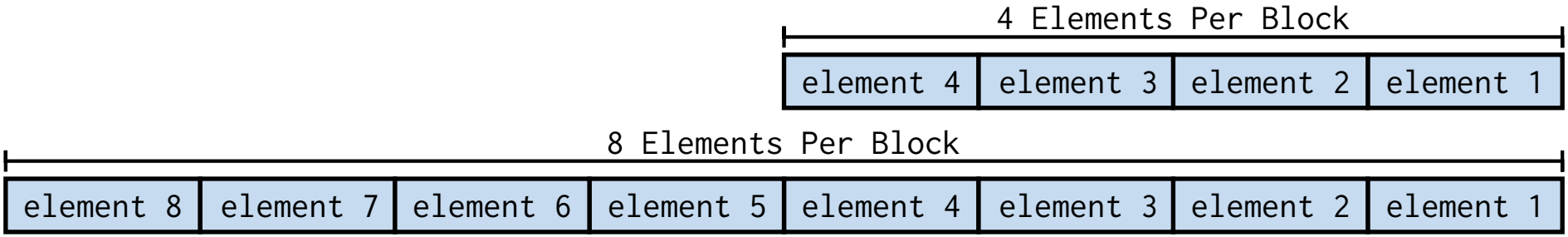
DANA: Single Transaction Execution



DANA: Multi-Transaction Execution



We Organize All Data in *Blocks* of *Elements*



Blocks for temporal locality

- We exploit neural network temporal locality of data
- Here, data refers to *inputs* or *weights*
- Larger block widths reduce inter-module communication
- Block width is an architectural parameter

Evaluation Networks

Area	Application	Configuration	Size
ASC [1]	`3sum`	$85 \times 16 \times 85$	large
	`collatz`	$40 \times 16 \times 40$	large
	`ll`	$144 \times 16 \times 144$	large
	`rsa`	$30 \times 30 \times 30$	large
Approximate Computing [2, 3, 4]	`blackscholes`	$6 \times 8 \times 8 \times 1$	small
	`fft`	$1 \times 4 \times 4 \times 2$	small
	`inversek2j`	$2 \times 8 \times 2$	small
	`jmeint`	$18 \times 16 \times 2$	medium
	`jpeg`	$64 \times 16 \times 64$	large
	`kmeans`	$6 \times 16 \times 16 \times 1$	medium
	`sobel`	$9 \times 8 \times 1$	small
Physics [5]	`edip`	$192 \times 16 \times 1$	large

[1] A. Waterland et al. “Asc: Automatically scalable computation,” in *ASPLOS*, 2014.

[2] H. Esmaeilzadeh et al., “Neural acceleration for general-purpose approximate programs,” in *MICRO*, 2012.

[3] R. St. Amant, et al., “General-purpose code acceleration with limited-precision analog computation,” in *ISCA*, 2014.

[4] T. Moreau, et al., “Snnap: Approximate computing on programmable socs via neural acceleration,” in *HPCA*, 2015.

[5] J. F. Justo et al., “Interatomic potential for silicon defects and disordered phases,” *Physical Review B*, vol. 58, pp. 2539–2550, Aug 1998.

Evaluation Methodology

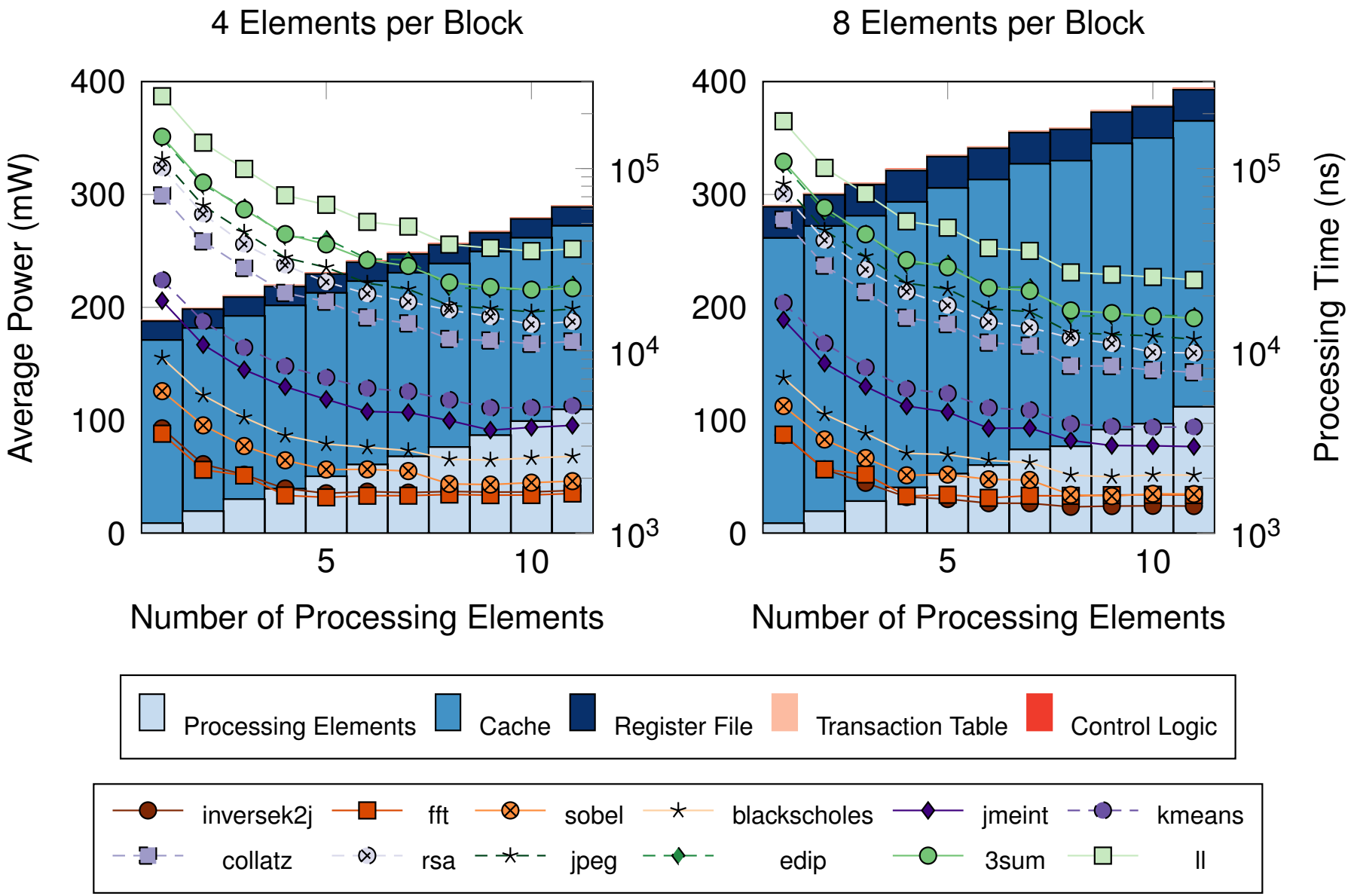
Implementation

- X-FILES Arbiter and DANA implemented in System Verilog
- Free parameters include:
  - Elements per block
  - The number of Processing Elements
  - Internal table widths and storage sizes

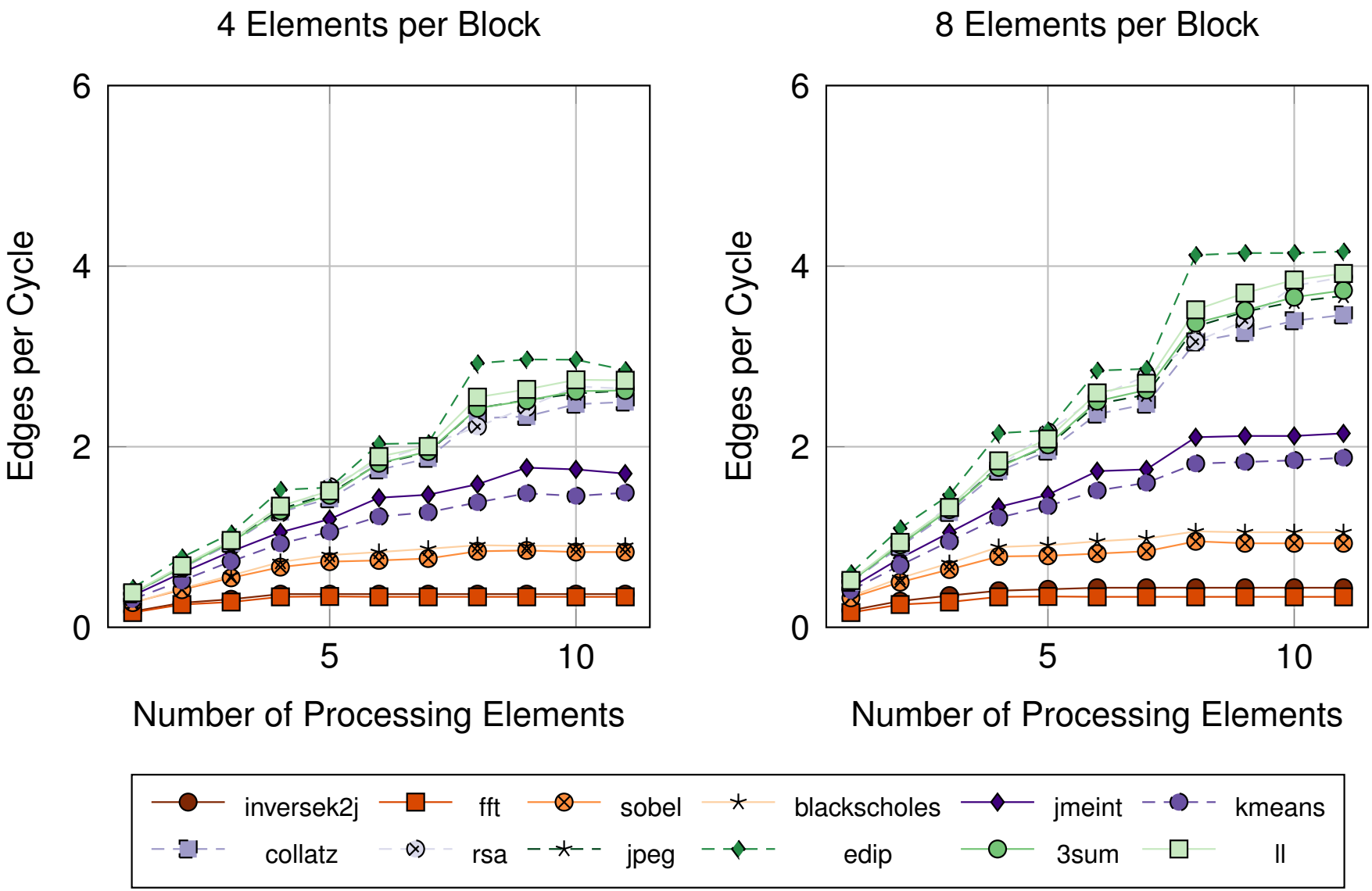
Evaluation

- We compute average power with Cadence SOC Encounter in a 45nm GlobalFoundries process
- We compute operating frequency using Cadence SOC Encounter
- We compute performance by running System Verilog testbenches at the computed operating frequency

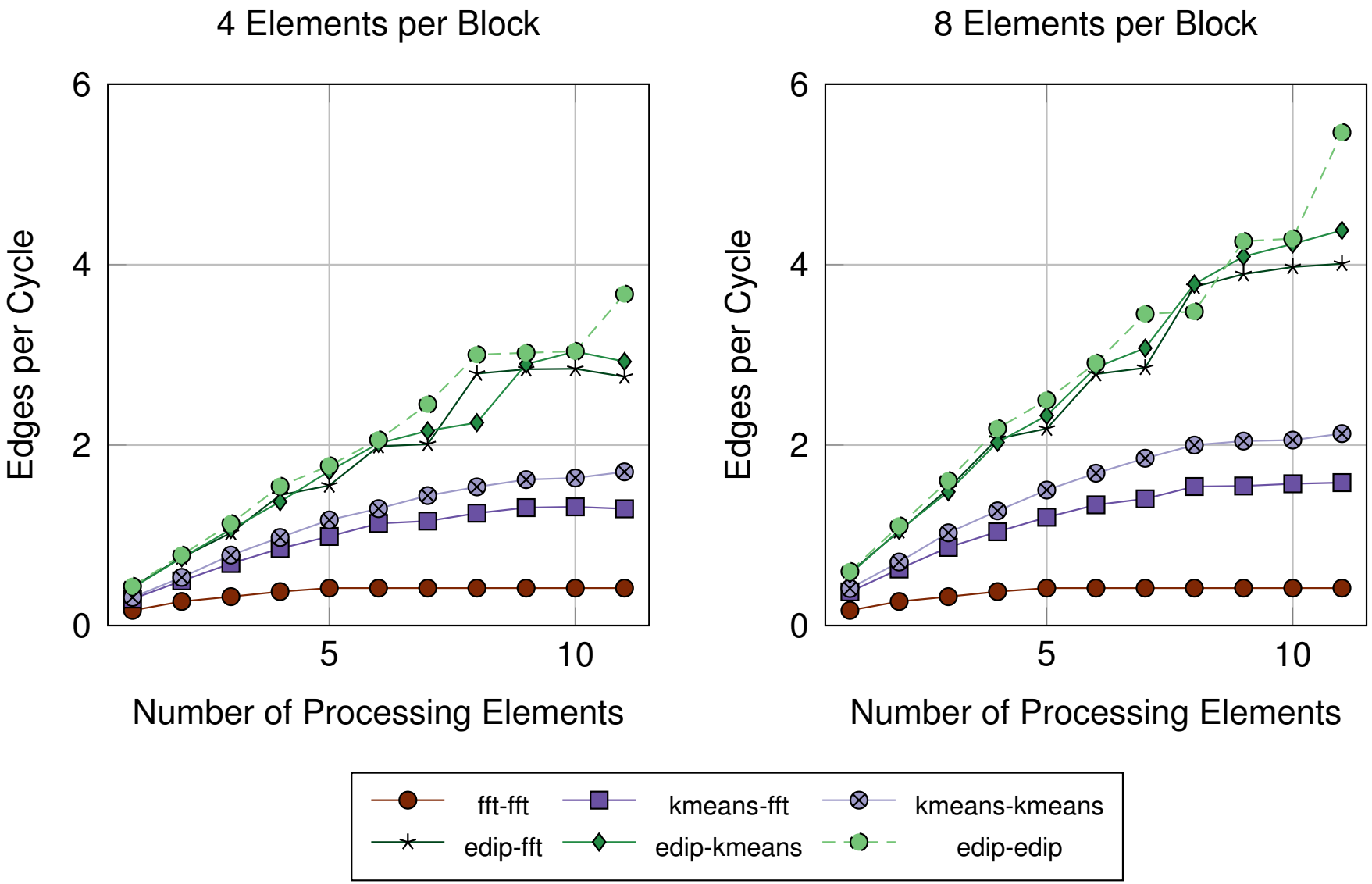
Power and Performance



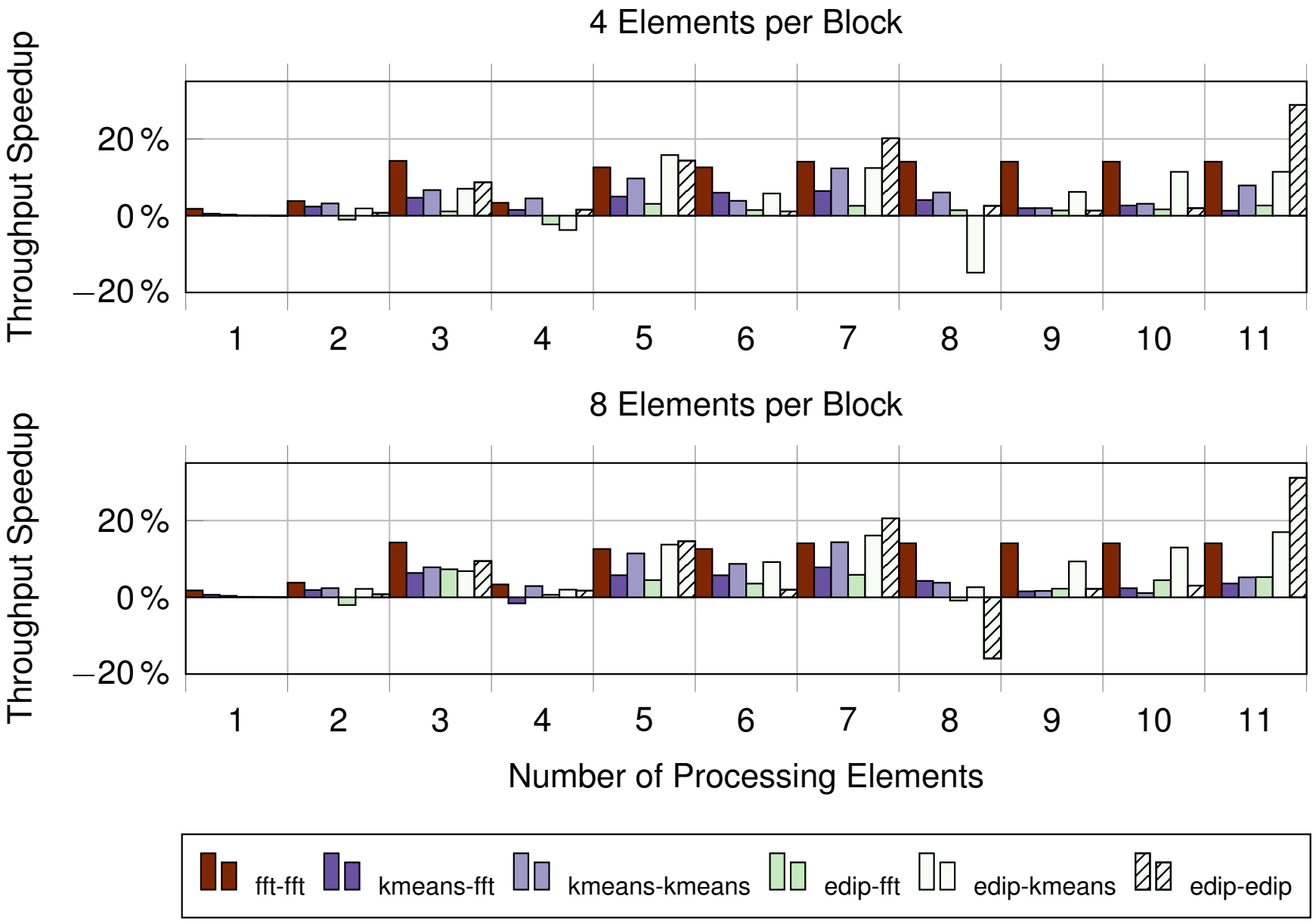
Single Transaction Throughput



Multi-Transaction Throughput



Multi-Transaction Speedup



Software Comparison

NN	Energy	Delay	EDP
`3sum`	7×	95×	664×
`collatz`	8×	106×	826×
`ll`	6×	88×	569×
`rsa`	6×	88×	566×

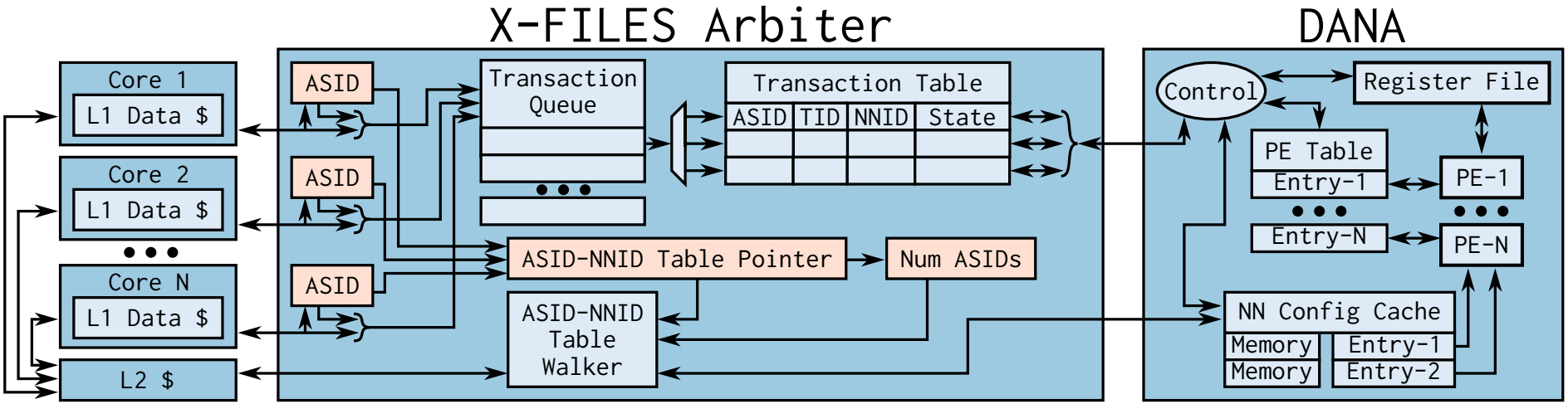
Methodology and comments

- Comparison against a single core Intel SCC
  - Performance and power computed using gem5 [1] and McPAT [2]

[1] N. Binkert et al., "The gem5 simulator," *SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[2] S. Li et al., "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

Summary and Acknowledgments



This work was supported by the following:

- A NASA Space Technology Research Fellowship
- An NSF Graduate Research Fellowship
- NSF CAREER awards
- A Google Faculty Research Award