X-FILES/DANA: RISC-V Hardware/Software for Neural Networks

Schuyler Eldridge[1] (schuye@bu.edu) Han Dong[2] Thomas Unger[2] Marcia Sahaya Louis[1]
Leila Delshad Tehrani[1] Jonathan Appavoo[2] Ajay Joshi[1]

Boston University Departments of Electrical and Computer Engineering[1] and Computer Science[2]

Neural Networks are "Hot Hot Hot!!!"

- Neural networks are layered structures consisting of neurons useful for:
  - Classification/regression
  - Soft computing
- Broad use in high tech, e.g., Google [1]
- Enable alternative computing methodologies
  - Automatic parallelization, e.g., ASC [2]
  - Approximate computing, e.g., NPU [3]

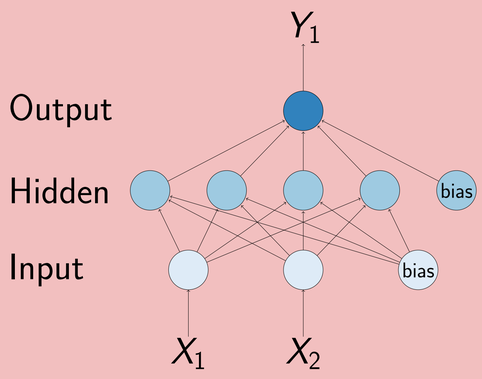

Figure 1: An example neural network

[1] Google TensorFlow, https://github.com/tensorflow/tensorflow
[2] A. Waterland, E. Angelino *et al.*, "Asc: Automatically scalable computation," in *Proc. ASPLOS*, 2014.
[3] H. Esmaeilzadeh, A. Sampson *et al.*, "Neural acceleration for general-purpose approximate programs," in *Proc. MICRO*, 2012.

Our Vision of General-Purpose Neural Network Computing

1. Treat neural networks as *functional primitives* backed by accelerators [1]
2. Follow a transaction model for neural network computation
3. Exploit anticipated sharing of neural networks across applications

- We break this into two distinct contributions:
  - X-FILES – Software/Hardware extensions for transaction management
  - DANA – One possible backend accelerator that interfaces with X-FILES

[1] S. Eldridge, A. Waterland *et al.*, "Towards general-purpose neural network computing," in *Proc. PACT*, 2015.

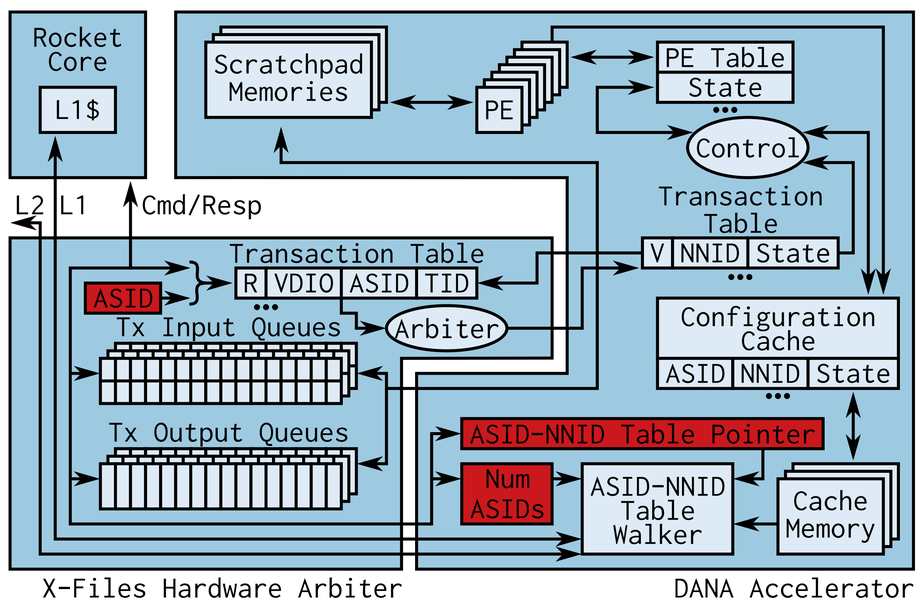

Figure 2: The X-FILES/DANA hardware architecture. Neural network transactions are managed by the X-FILES hardware arbiter and executed on the backend accelerator, DANA.

Table 1: X-FILES user, supervisor, and Proxy Kernel systemcall API

User/Supervisor	Function

supervisor	`old_asid = set_asid(new_asid)`
	`old_antp = set_antp(*asid_nnid_table_entry, size)`
	`csr = xf_read_csr(csr_index)`

user	`tid = new_write_request(nnid, learning_type, num_output)`
	`error_code = write_data(tid, *inputs, num_input)`
	`error_code = read_data_spinlock(tid, *output, num_output)`
	`id = xfiles_dana_id(flag_print)`
	`error_code = kill_transaction(tid)`

user (PK only)	`old_asid = pk_syscall_set_asid(new_asid)`
	`old_antp = pk_syscall_set_antp(new_antp)`
	`asid_nnid_table_create(**table, num_asids, num_configs)`
	`asid_nnid_table_destroy(**table)`
	`attach_nn_configuration(**table, asid, *nn_config)`
	`attach_garbage(**table, asid)`

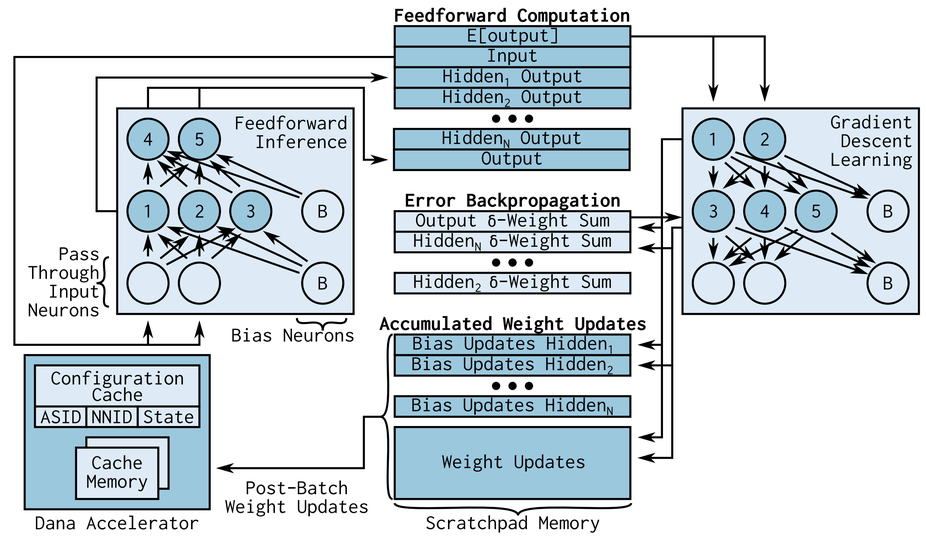

Figure 3: Processing Element (PE) ordering and the interactions of PEs with one of DANA's per-transaction scratchpad memories for feedforward and learning transactions

Safe Multi-Transaction Management using ASIDs

- An ASID defines an address space of neural networks, identified with NNIDs
- An OS-managed ASID–NNID Table enables NNID dereferencing (Figure 4)
- All transparently handled by the OS

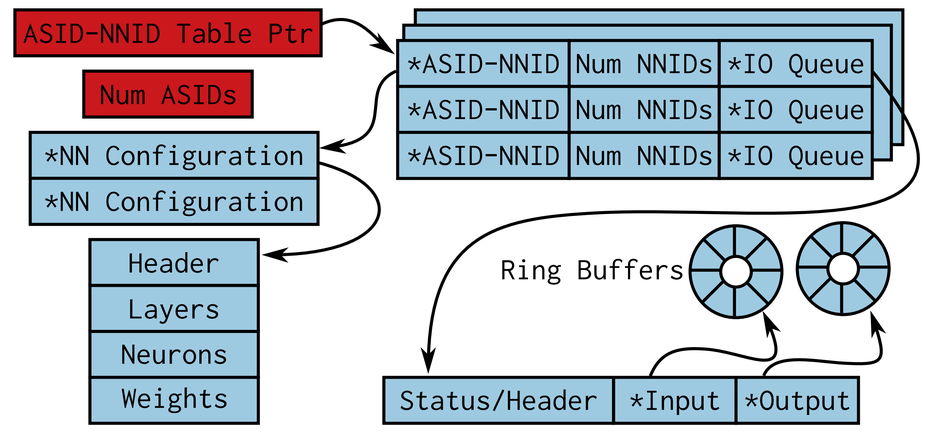

Figure 4: An ASID–NNID Table for dereferencing neural network configurations from an ASID and an NNID.

Usage with a RISC-V Microprocessor

1. Grab a Rocket Chip RISC-V Microprocessor [1]
2. Build a RISC-V toolchain
3. Grab a copy of our X-FILES/DANA accelerator [2]
4. Build Rocket + X-FILES/DANA for FPGA
5. User processes can throw *transactions* at X-FILES

[1] Rocket Chip git repository, UC Berkeley, Online: github.com/ucb-bar/rocket-chip
[2] X-FILES/DANA git repository, Boston University, Online: github.com/bu-icsg/xfiles-dana

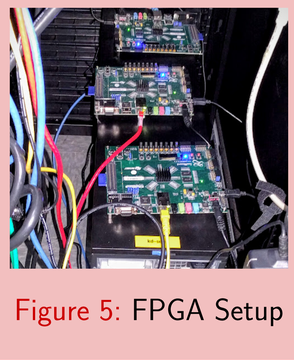

Figure 5: FPGA Setup

Initial Evaluation

- Comparison against a RISC-V microprocessor running FANN [1]
  - Figure 6 shows a 30× improvement over software for networks in Table 2

[1] S. Nissen, "Implementation of a fast artificial neural network library (fann)," DIKU, Tech. Rep., 2003, http://fann.sf.net.

Open Source Availability and Acknowledgments

- **On GitHub**: github.com/bu-icsg/xfiles-dana
- This work was supported by the following:
  - A NASA Space Technology Research Fellowship, NSF CAREER Awards

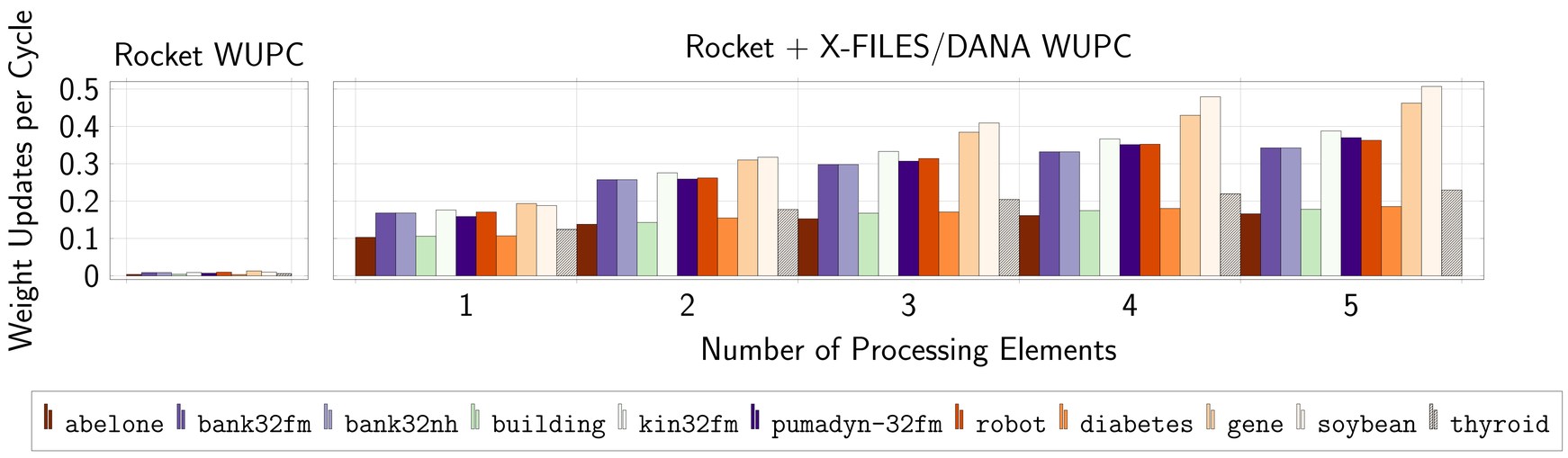

Figure 6: The weight updates per cycle (WUPC) when executing batch learning requests for different NN configurations

Table 2: Evaluated FANN-provided datasets

Type	Name	Topology
Regression	abelone	$10 \times 8 \times 1$
	bank32fm	$32 \times 16 \times 1$
	bank32nh	$32 \times 16 \times 1$
	building	$14 \times 8 \times 3$
	kin32fm	$32 \times 20 \times 1$
	pumadyn_32fm	$32 \times 16 \times 8 \times 4 \times 1$
	robot	$48 \times 16 \times 3$
Classification	diabetes	$8 \times 10 \times 2$
	gene	$120 \times 20 \times 3$
	soybean	$82 \times 32 \times 19$
	thyroid	$21 \times 10 \times 3$