A Systolic Neural CPU Processor Combining Deep Learning and General-Purpose Computing With Enhanced Data Locality and End-to-End Performance

Yuhao Ju, *Student Member, IEEE*, and Jie Gu, *Senior Member, IEEE*

***Abstract*— While neural network (NN) accelerators are being significantly developed in recent years, CPU is still essential for data management and pre-/post-processing of accelerators in a commonly used heterogeneous architecture, which usually contains an NN accelerator and a processor core with data transfer performed by direct memory access (DMA) engine. This work presents a special neural processor, referred to as a systolic neural CPU processor (SNCPU), which is a unified architecture combining deep learning and general-purpose computing for fifth-generation of reduced instruction set computer (RISC-V) to improve end-to-end performance for machine learning (ML) tasks compared with a common heterogeneous architecture with CPU and accelerator. With 64%–80% processing elements (PEs) logic reuse and 10% area overhead, SNCPU can be configured into ten RISC-V CPU cores. Special bi-directional dataflow and four different working modes are developed to enhance the utilization of deep NN (DNN) accelerator and eliminate the expensive data transfer between CPU and DNN accelerator in existing heterogeneous architecture. A 65-nm test chip was fabricated demonstrating a 39%–64% performance improvement on end-to-end image classification tasks for ImageNet, Cifar10, and MNIST datasets with over 95% PE utilization and up to 1.8TOPs/W power efficiency.**

***Index Terms*— Bi-directional dataflow, CPU, deep neural network (DNN) accelerator, end-to-end performance, heterogeneous architecture, machine learning (ML), general-purpose computing for fifth-generation of reduced instruction set computer (RISC-V).**

I. Introduction

ACCELERATORS designed for machine learning (ML) tasks, especially for DNN, are being rapidly developed in recent years. Because of the broad application space of DNN and its tremendous computing workload, improving energy efficiency for DNN accelerator has become a dominant effort for accelerator design. Cross-layer approaches have been explored at the software level such as quantization [1], architecture level such as flexible dataflows [2], and bit-precision [3], [4], micro-architecture level such as adaptive clock [5], and circuit level such as compute-in-memory techniques [7], [8], [9], [10]. However, very few works have focused on improving the end-to-end performance of deep learning tasks for the system-on-chip (SoC) level considering the cooperation of the accelerator and other digital modules.

In an end-to-end ML task, besides DNN computing using the accelerator, pre-processing, and post-processing also consume significant latency and power [11], [12], especially for edge computing on Internet-of-Things (IoT) devices with real-time processing applications where data streaming is performed from sensory device to the digital backend. For example, in [11], the DNN inference accelerator is served as an "edge-gateway-cloud" which achieves on-the-fly visual recognition and classification of insect blobs. Preprocessing in [11], including image capture, segmentation, extraction, and loading DNN configuration settings, takes over 70% of run-time for the end-to-end procedure. In [12], the SoC is implemented for a vision artificial intelligence (AI) solution with the DNN accelerator and image sensor on one single chip. Neural network (NN) results need to pass through a digital signal processing (DSP) sub-system, a result-selector, and interfaces for post-processing. Before DNN processing, image readout and image signal processing (ISP) are necessary for the accelerator as preprocessing work. Overall, pre/post-processing and data management take around 60% of the total run time.

Even for commonly used neural processing units (NPUs), accelerators are only in charge of the multiplication-accumulation (MAC) operations for different layers of NNs. As for inter-layer data preparation such as padding, batch normalization, data alignment, and duplication, the accelerator needs cohesive cooperation with another CPU core and highly efficient core-to-core communication. Fig. 1 illustrates a typical heterogeneous configuration that has a DNN accelerator and a CPU pipeline core that handles the data preparation and pre-/post-processing [6], [11], [12], [13], [14], [15]. Direct memory access (DMA) engine is developed to perform data transfer between the CPU core and the DNN accelerator. Interface and global MEM provide the capability for data streaming with off-chip DRAMs. Within the DNN accelerator, the systolic array is a popular architecture for 2-D convolution with data being piped through PE units for easier dataflow management and less SRAM bandwidth requirements.

Manuscript received 29 May 2022; revised 25 August 2022; accepted 30 September 2022. Date of publication 27 October 2022; date of current version 28 December 2022. This article was approved by Associate Editor Sophia Shao. This work was supported by the National Science Foundation under Grant CCF-2008906. *(Corresponding author: Yuhao Ju.)*

The authors are with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA (e-mail: yuhaoju2017@u.northwestern.edu; jgu@northwestern.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2022.3214170.

Digital Object Identifier 10.1109/JSSC.2022.3214170

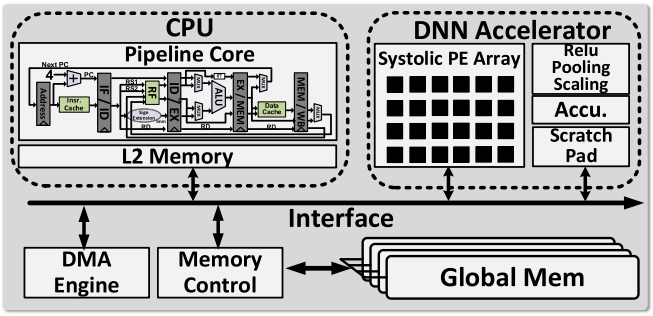


Fig. 1. Conventional heterogeneous architecture with a CPU pipeline core and a DNN accelerator.

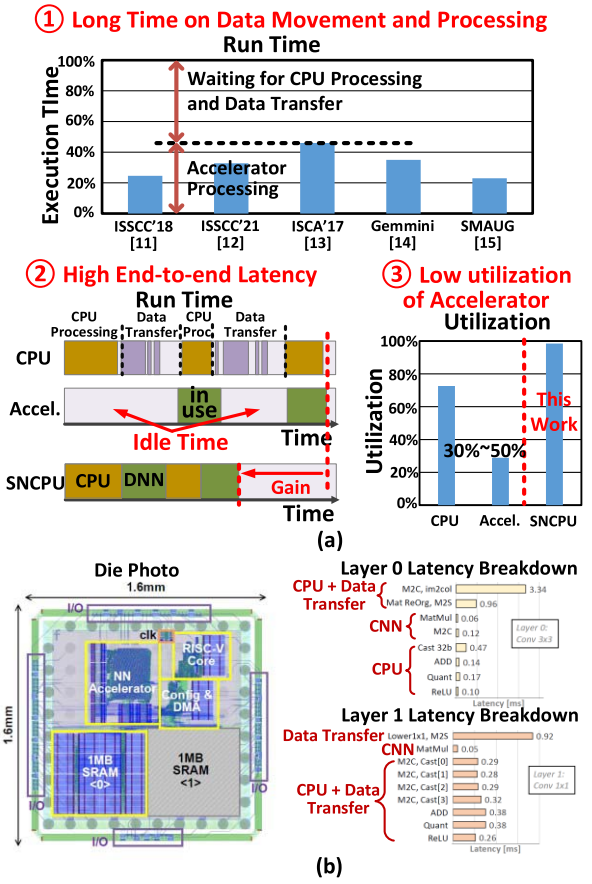


Fig. 2. (a) Challenges for heterogeneous architecture of the CPU and the accelerator. (b) Example of workload breakdown from the recent demonstration from meta [6].

There are several critical challenges for the existing heterogeneous architecture of CPU and accelerator to process end-to-end ML tasks. The challenges include: 1) large run time cost due to data movement across CPU and accelerators; 2) high latency for data preparation and pre-/post-processing using CPU; and 3) low utilization of accelerator due to imbalance of workload and waiting time on CPU and data transfer across processing cores [6], [11], [12], [13], [14], [15]. As shown in Fig. 2(a), for an end-to-end ML task, the accelerator is utilized for only 30%–50% of the total run time. The rest of the time is to wait for CPU processing and data movement between CPU and accelerator cores, which causes the stall of the accelerator. Only 46.2% of utilization is reported from the tensor processing unit (TPU) [13].

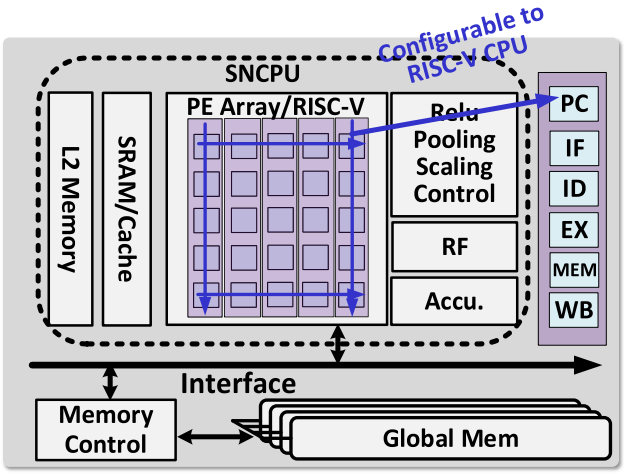


Fig. 3. Block diagram for the architecture of the proposed SNCPU.

Another detailed example is given in Fig. 2(b) for a test chip, which is recently published by Meta on AR/VR applications [6]. The SoC contains an NN accelerator with an general-purpose computing for fifth-generation of reduced instruction set computer (RISC-V) CPU pipeline core and a powerful DMA engine for data movement. For the first layer of the convolutional NN (CNN) model for eye gaze tracking in this SoC, the total latency is 5.36 ms with only 0.18 ms used by accelerator computing. Data movement to different cores and initial data preparation using im2col take more than 60% of the execution (EX) time. A similar story is for the second layer of CNN. Without initial image preprocessing, the data movement still takes more than 1 ms in terms of 3.16 ms total latency, which is over 30% of the total run time. NN accelerator only takes 0.05 ms for CNN processing. The rest of the EX time is for CPU processing such as casting, ReLU, batch normalization, and quantization.

To improve the efficiency of core-to-core communication, prior works have considered compressing data, accelerating data transfer, and increasing MEM bandwidth. As an example of [16], an accelerator coherency port (ACP) was designed to request data directly from the last level cache of the CPU instead of using the DMA engine to control the data transfer.

To address the challenges discussed above, in this work, we developed a new architecture, which is shown in Fig. 3, referred to as a systolic neural CPU processor (SNCPU), which combines an RISC-V CPU and a systolic CNN accelerator in one unified core. The contributions of this work include: 1) the flexible configuration for a multi-core RISC-V CPU or a systolic DNN accelerator with over 95% PE utilization for the end-to-end operation; 2) significant throughput improvement for CPU work because of ten-core CPU reconfiguration with 10% area overhead compared with a heterogeneous architecture for one CPU and one accelerator; 3) avoidance of expensive data movement across cores by using a special bi-directional dataflow for latency reduction; and 4) the demonstration of SNCPU through a 65-nm test chip with the

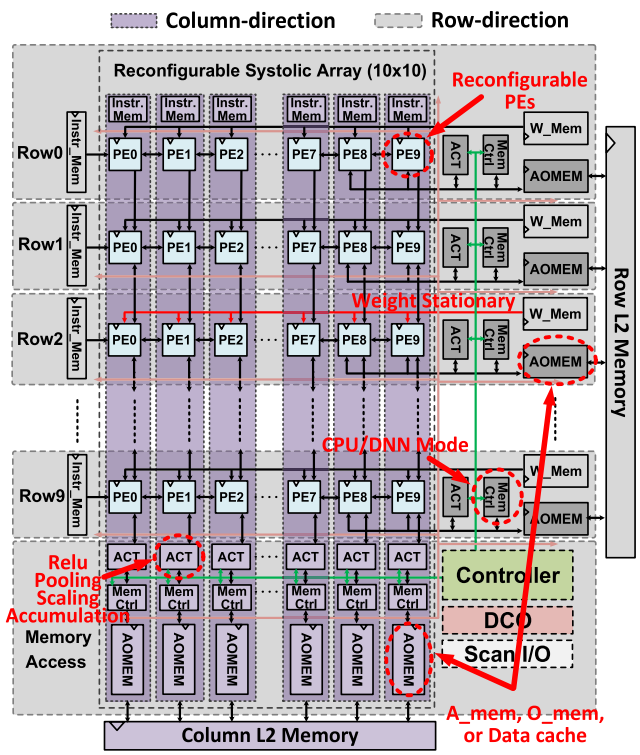


Fig. 4. Top-level architecture of SNCPU.

39%–64% latency improvement and the 0.65–1.8TOPS/W energy efficiency for end-to-end image-classification tasks.

The rest of this article is organized as follows. The overview of the top-level architecture of SNCPU is discussed in Section II. Section III shows the details for PE logic reuse and MEM reconfiguration supporting both CPU and accelerator functions. Special bi-directional dataflow is introduced in Section IV. The implementation and measurement results obtained by the test chip are shown in Section V, which also includes the test cases. Conclusions are in Section VI. This article is a detailed extension of the conference publication in ISSCC 2022 [22].

II. Top-Level Architecture of SNCPU

A. Design Overview

Fig. 4 shows the top-level architecture of SNCPU. A reconfigurable 10 × 10 PE array serves as the central computing tiles. This 2-D systolic array can be utilized as a baseline DNN accelerator or a ten-core five-stage RISC-V processor by reusing PE logic and MEM.

As shown in Fig. 4, the baseline DNN accelerator supports INT8 MAC operations with weight stationary dataflow. Memories are evenly separated and mapped to each row or column of the accelerator to enhance the flexibility and convenience of CPU mode reconfiguration. Each row or column of ten PEs has one reconfigurable "activation, output and data cache memory" (AOMEM), which can be reconfigured as an activation (ACT) SRAM bank, an output SRAM bank, or a data cache in different reconfigurations or dataflows. In addition, the ACT module is implemented for each row or column to deal with

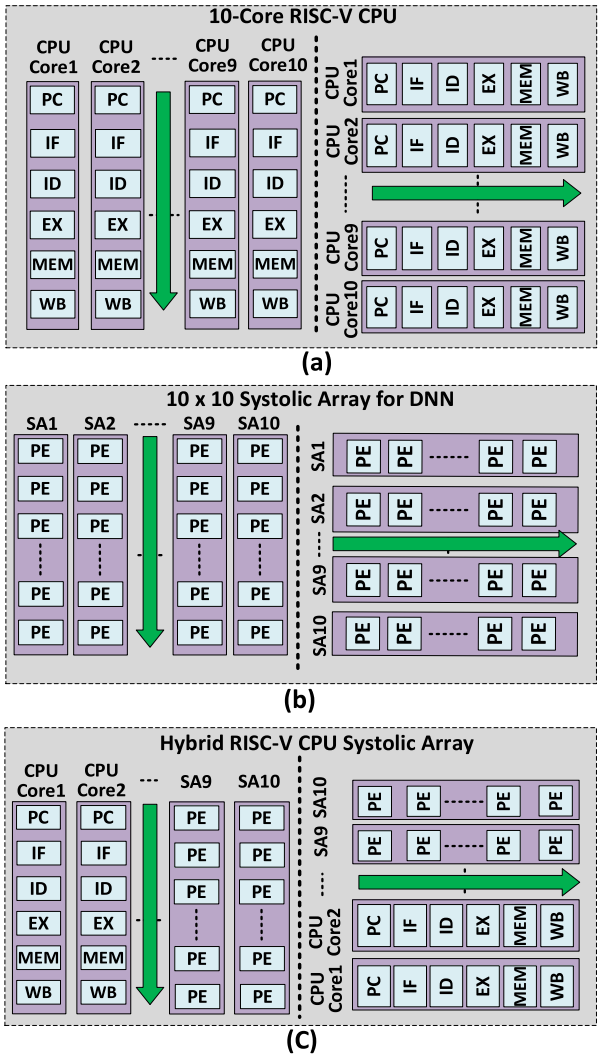


Fig. 5. Different configuration modes of SNCPU. (a) Multi-core CPU mode. (b) DNN mode. (c) Hybrid mode.

simple processing functions that DNN needs, such as partial sum accumulation, pooling, ReLU, and scaling. Furthermore, each row/column of 10 PEs owns a small controller to control the SRAM data arbitration and cohesive cooperation with different reconfiguration modes (DNN/CPU) and dataflows. A global digital controller oscillator (DCO) provides a tunable global clock. A top-level controller is designed to control the mode switching and dataflow switching. Each PE in the 2-D array is reconfigurable to satisfy the requirement of typical DNN operation and CPU operation. Additional SRAM banks are used to support multi-core CPU functionalities, such as the instruction SRAM bank of each row/column and two L2 SRAM banks for core-to-core communication in CPU mode.

B. Reconfiguration Modes and Dataflow Overview

Reconfiguration modes are shown in Fig. 5 where each lane of the PE array, i.e., each row or each column of PEs, can be configured as either systolic MAC operations for the DNN accelerator or CPU pipeline stages. Associated SRAM banks are also reconfigured for both purposes. Each mode contains

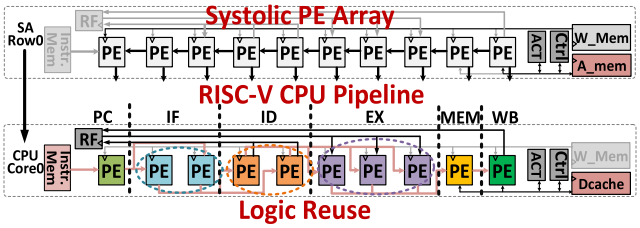


Fig. 6. Reconfiguration of rows of PE array for a single RISC-V pipeline core.

bi-directional dataflows and will be introduced in Section IV in detail. Fig. 5(a) illustrates two multi-core CPU modes as each row/column is utilized as one five-stage RISC-V pipeline core. Fig. 5(b) demonstrates DNN modes supporting DNN using MAC operation for convolutions. A special hybrid mode is also supported for imbalanced workload for end-to-end ML tasks between the CPU and the accelerator. In the hybrid mode, as shown in Fig. 5(c), half number of PEs are configured as DNN accelerators while the rest are configured into a five-core RISC-V CPU providing additional flexibility in workload assignment between the DNN accelerator and the CPU.

III. PE Logic Reuse and Memory Reconfiguration

A. PE Logic Reuse for CPU Reconfiguration

Fig. 6 shows the construction of a 32-b five-stage RISC-V CPU pipeline core from a lane of systolic PE array. Similar to a typical accelerator design, each PE in SNCPU contains a simple pipelined MAC unit with 8-b input and weight and maximum 32-b accumulation output. Fig. 6 also shows the logic reuse methodology for one row of ten PEs to realize pipeline functions. One column of ten PEs also supports CPU mode reconfiguration by using the same strategy of logic reuse. Based on the CPU mode reconfiguration for one row and one column, each PE needs to support two different pipeline stages of CPU mode, which means PE design is not unified in the 2-D PE array. While the design complexity has increased for the PE unit, the additional design efforts were managed by creating templates of pipeline stages and re-distribute them to the specific PE based on its location in the 2-D array. Besides one row/column PE reconfiguration, additional CPU-only SRAM banks were added to support complete CPU functions such as instruction caches and register files (RFs). CPU-only SRAM banks are gated for DNN modes.

Details of PE logic reuse for row reconfiguration are shown in Fig. 7, the very first PE in a row or column reuses the MAC's adder and 32-b registers as program counter (PC) to provide the instruction cache access for each pipeline core. Two PEs are used as the instruction fetch (IF) stage for instruction fetch with a reuse of the internal 32-b register and 8-b input registers. Two PEs are reconfigured into the instruction decoding (ID) stage, where the logic in the 8-b multiplier and 32-b adder are reconstructed to generate computing control signals by performing numerical/logical operations with the op-code or func-code of instructions. The next three PEs are combined into the EX stage, including one PE serving as an arithmetic logic unit (ALU) with additional

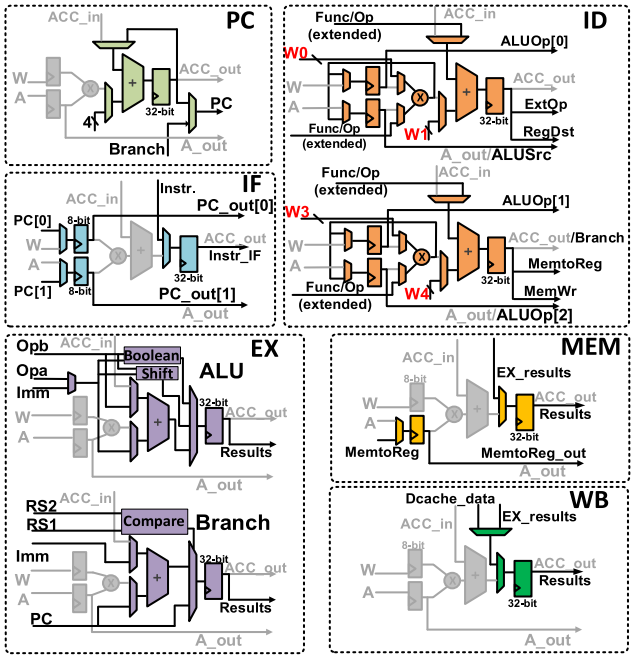


Fig. 7. PE logic reuse details of five pipeline stages for RISC-V ISA of CPU mode reconfiguration.

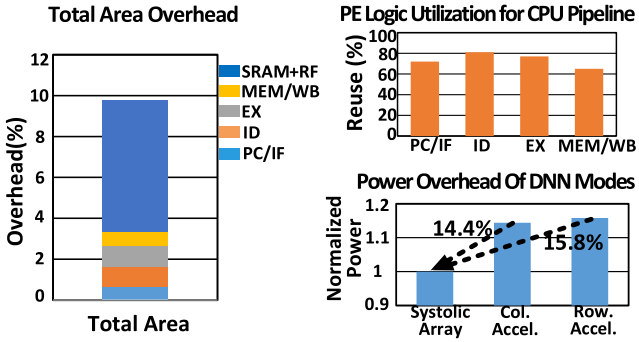


Fig. 8. Area and power overhead for CPU support and PE logic reuse rate.

logic for Boolean operations and a shifter, one PE supporting ALU branch calculation for new address from the instruction cache, and one PE used only for the registers to pass the EX results. The next PE is reconfigured into the CPU MEM stage for sending the ALU results to the data cache (Dcache) or bypassing the Dcache read/write. The last PE is utilized as the write-back (WB) stage to fetch the readout data from Dcache and send it to the RF by reusing registers with additional MUX logic. Several forwarding paths are also implemented to support CPU data dependency. Multipliers are also reused to realize multiplication instructions for RISC-V instruction set architecture (ISA). The multiplication instruction may take several cycles to finish in CPU modes.

B. Area, Power Overhead, and Utilization

As shown in Fig. 8, with an emphasis on logic sharing, the pipeline core reconfiguration utilizes 64%–80% of the original PE logic for CPU construction including the reconfiguration for multiplication instruction. The logic reuse rate is 69% at

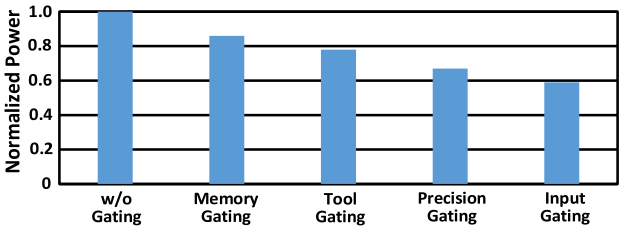


Fig. 9. Different implemented power gating techniques and their power benefits on DNN modes.

the PC and IF stages, 80% at the ID stage, 77% at the EX stage, and 64% at MEM/WB stage. The logic reuse rates are the average values for both row and column pipeline reconfigurations which differ slightly. For example, adders in PE are not reused in MEM/WB stage for row reconfiguration but are reused in column pipeline reconfiguration. Compared with the original systolic DNN accelerator, the area overhead to include CPU functions is 3.4% in the PE-array, 6.4% in the MEM, e.g., instruction and RF, and overall 9.8% for the whole SNCPU processor. Extensive clock gating is implemented to eliminate redundant power consumption from the additional logic and unused memories for DNN modes. As shown in Fig. 8, in two different accelerator modes of SNCPU, power overhead is about 15% compared with the baseline systolic array DNN accelerator.

Fig. 9 shows the improvement of different clock gating techniques for the DNN modes. Gating unused SRAM banks for the DNN modes such as instruction caches, L2 caches, and RFs can achieve 15% power saving. RTL-level optimization for better support of the gating function from the synthesis tool provides another 8% power saving. CPU reconfiguration with RV32I has 32-b precision while the DNN configuration only needs 26-b precision at most without overflow for the accumulator of each PE. Hence, unused flip-flops are gated for DNN modes achieving another 11% power saving with two different reconfigured precisions (20 b, 26 b) for different models. Special input gating is also used to eliminate the power consumption of unused CPU-only combinational logic leading to a 6% power saving. Overall, nearly 40% of energy saving for DNN modes is achieved by power gating.

Fig. 10 shows the power overhead for different CPU instructions from the reconfigurable PEs implemented in this work in comparison with baseline RISC-V pipelines excluding MEM power. The CPU operation from the reconfigurable PE array incurs around 14% power overhead compared with the scalar baseline RISC-V design (RV32I).

C. Customized Extension of RISC-V ISA for SNCPU

Customized instructions added into baseline RISC-V ISA as extension instructions are developed to support a smooth mode switching between CPU and DNN modes as shown in Fig. 11(a). Several control and status registers (CSRs) are used to store and configure the parameters required by the DNN accelerator mode processing. Accelerator setup instruction

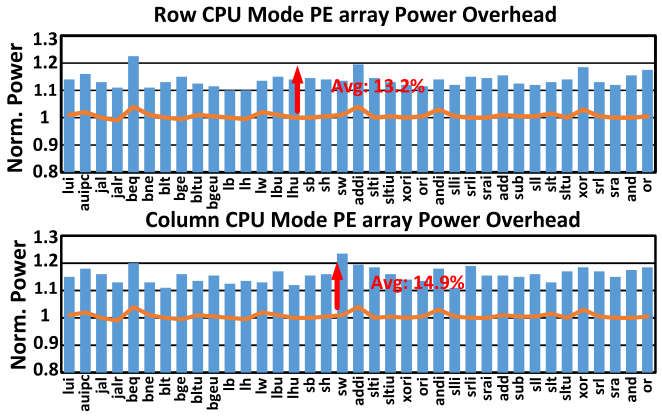


Fig. 10. Simulated power overhead from the reconfigurable PE array for RV32I instructions.

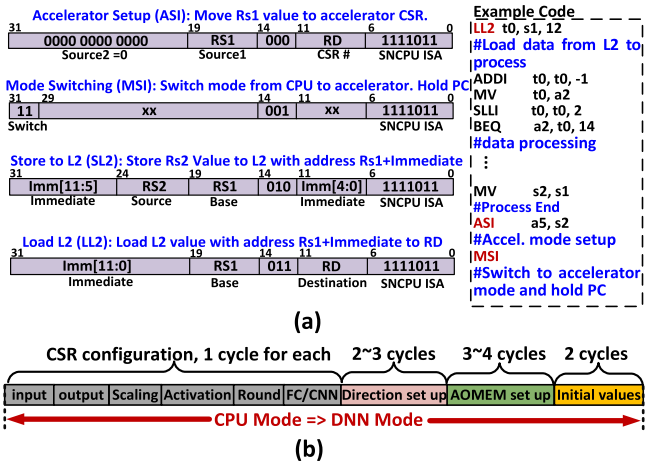


Fig. 11. (a) Customized RISC-V ISA extension for CSR, mode switching, and multi-core communication for CPU modes. (b) Detailed mode switching process from CPU to accelerator.

(ASI) is utilized to save values to those CSRs before the DNN accelerator starts to run. Mode switching instruction (MSI) is used to switch from CPU mode to DNN accelerator mode with holding the PC for future processing. Store L2 instruction (SL2) and load L2 instruction (LL2) provide a direct communication method to store/reload data from L2 for multi-core CPU data sharing.

The switching process from CPU to DNN accelerator is shown in Fig. 11(b). CSR configuration takes 6 cycles to save required configuration parameters for DNN running, such as input vector numbers, output vector numbers, scaling factor, and convolution/fully-connected (FC) calculation selection, 2–3 cycles are needed to set up the flow direction, i.e., row-based or column-based flow. The 3–4 cycles are used for "AOMEM" SRAM bank selection and configuration. It also takes two additional cycles to set up initial values for the DNN controller and initialize input gating. The total switching time is 13–15 clock cycles for switching from CPU to DNN configuration. On the other hand, when the DNN accelerator finishes the work, a trigger signal is generated to switch back into CPU modes from accelerator modes consuming only 3 cycles.

D. Reconfiguration Modes and Memory Reuse

The SNCPU architecture allows the majority of data to be retained inside the SRAM banks of the processor core even when operation modes change. It can eliminate the expensive data movement and the DMA engine in the heterogeneous architecture. To achieve high data locality, a special dual-mode bi-directional dataflow is developed resulting in four different reconfiguration modes. The four reconfiguration modes include two different operations, i.e., CPU and DNN, two directional dataflows, i.e., row-based dataflow and column-based dataflow.

Fig. 12 shows the detailed four different operation modes with activated modules highlighted in this figure. As shown in Fig. 12(a), column accelerator mode is the typical accelerator mode with weight stationary dataflow. The "AOMEM" SRAM bank in each row is used as input MEM to provide input data from right to left. Bottom "AOMEM" SRAM banks are used for output MEM for the accumulation results from top to bottom. Instruction caches and L2 SRAM banks are gated in this mode. As for the row CPU mode, which is shown in Fig. 12(b), each row of ten PEs can be configured as a CPU pipelined core, with each "AOMEM" SRAM bank on the right serving as a data cache for each CPU core. Instructions caches on the left are activated to pass instructions through every pipeline stage from left to right.

In the other directional scenario, as shown in Fig. 12(c), the PEs in row accelerator mode receives the input data from the bottom AOMEM banks and store the results in the right AOMEM banks. Bottom "AOMEM" banks are reused as input MEM for each column. Right "AOMEM" banks are utilized as output MEM. The accumulation dataflow for row accelerator mode is from left to right, which is an orthogonal direction compared to the column accelerator mode. All CPU-only SRAM banks are gated in this mode. The last mode, column CPU mode, is introduced in Fig. 12(d). It receives the instructions from the top instruction caches and the bottom AOMEM banks are reconfigured to data caches, which allows one column of ten PEs to be reconfigured to one RISC-V pipeline core.

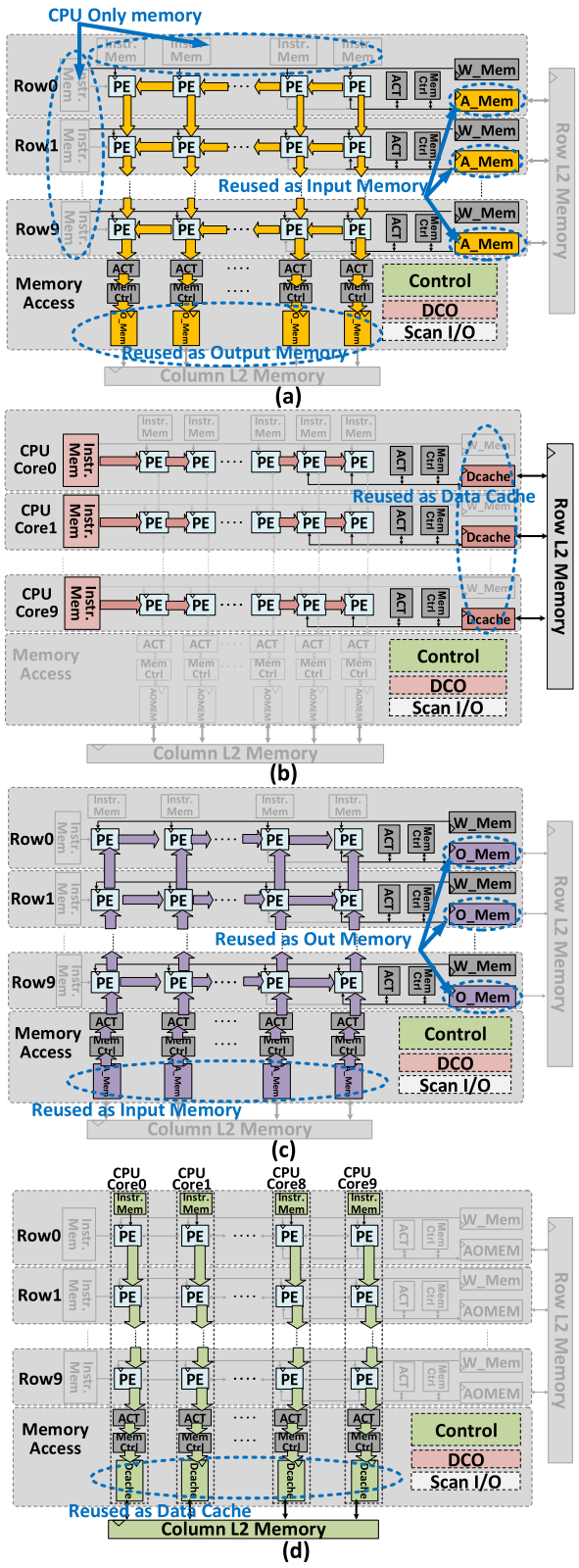


Fig. 12. Four different reconfiguration modes for two directions of SNCPU. (a) Column accelerator mode. (b) Row CPU mode. (c) Row accelerator mode. (d) Column CPU mode.

IV. Dual-Mode Bi-Directional Dataflow for Enhanced Data Locality

Fig. 13(a) shows the diagram of a special four-step dataflow with the four different configurations from Fig. 12 using the image classification task as an example. The four-step dataflow is designed to avoid data movement across MEM banks at different operation phases. In step 1 and step 3, different CPU modes are responsible for the pre/post-processing and inter-layer data preparation. Step 2 and Step 4 are used for DNN operations. In image classification tasks, preprocessing includes reshape, grayscale, rotation, and normalization. The inter-layer data processing for some commonly used DNN model contains data alignment, padding, duplication, and batch normalization.

Details are shown in Fig. 13(b). First, the SNCPU operates in row CPU mode to perform image preprocessing to generate the input data for the DNN operation of the first layer of the DNN model. Finished input data is stored in the "AOMEM" SRAM banks on the right, which are utilized as data caches

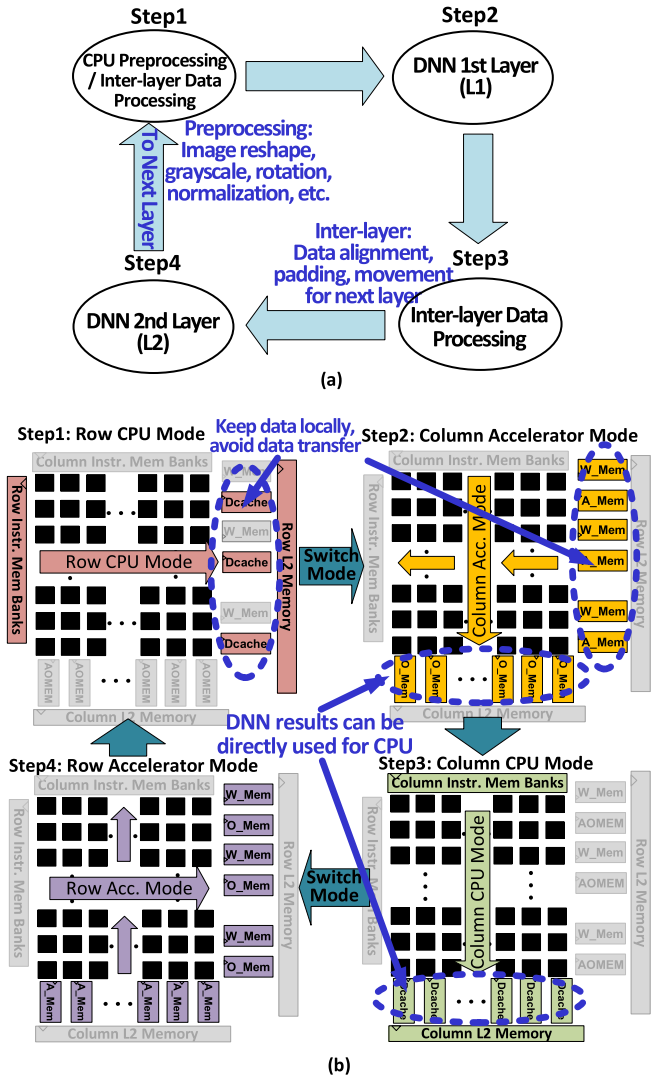
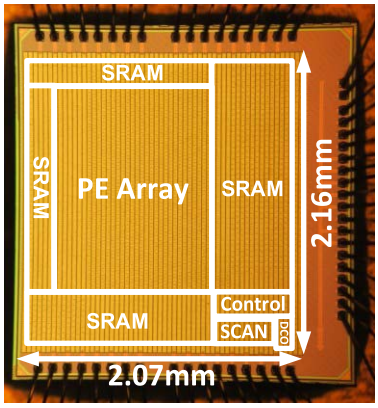


Fig. 13. (a) Block diagram and (b) architecture level illustration for four-step dual-mode bi-directional dataflow in SNCPU using image classification as an example.

for row CPU mode. Second, the SNCPU operates in column accelerator mode which reuses the right "AOMEM" banks as input MEM for the DNN accelerator. The DNN accelerator can directly use the data from the right "AOMEM" SRAM banks as input data, which is also the result of row CPU mode. Considering the same scenario in heterogeneous architecture, the DMA engine has to be engaged to transfer preprocessed data from the CPU data cache to the input scratchpad of the accelerator. Moreover, after the column accelerator finishes the entire first layer of the DNN model, the SNCPU is reconfigured to column CPU mode to perform data preparation work such as data alignment, padding, duplication, and batch normalization by directly using the data in the output MEM from the previous accelerator mode. Finally, the SNCPU switches to row accelerator mode to process the second layer of the CNN by directly using the data cache from the previous CPU mode as input MEM. The four-step operation repeats until all CNN layers are finished. All the data can stay either in the bottom "AOMEM" SRAM banks or right



Technology	65nm CMOS
Accelerator Area	4.47mm²
PE Number	100
CPU Power	589mW (10-Core)
DNN Power	116mW
Bit Precision	INT8
Frequency	400MHz
Supply Vdd	0.5~1V
SRAM	150KB
Efficiency (TOPS/W)	0.66 (8b, 1V) 1.82 (8b, 0.5V)

Fig. 14. Chip micrograph and specifications.

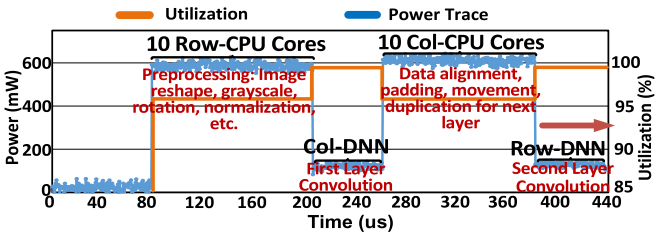


Fig. 15. Measured power trace and PE utilization during processing of the VGG16 model for CIFAR10 dataset.

"AOMEM" SRAM banks without any data transfer between CPU modes and DNN modes. Multi-core CPU improves the CPU throughput because of the parallel processing.

V. Chip Implementation, Measurement Results, and Case Study

A. Chip Implementation

A 2-D 10 × 10 SNCPU processor was designed and fabricated using a 65-nm CMOS process. The chip micrograph and implementation details are shown in Fig. 14. The active die area is 4.47 mm^2 (2.07 × 2.16 mm) with 1.0-V nominal supply voltage and 400-MHz operating frequency. The chip can support 8-b integer bit precision for DNN accelerator modes and RISC-V 32-b integer ISA for CPU modes with several customized ISA extensions. The chip was tested with a supply voltage scaled down to 0.5 V. The total on-chip SRAM is around 150 kB. This chip provides a scan IO interface to load and read out all on-chip SRAM content. A field-programmable gate array (FPGA) board is engaged in chip testing for data streaming in and out of the test chip through scan IO ports for verification and measurement.

B. Performance Measurement Results

Fig. 15 shows the measured first a few hundred microseconds of power trace on different modes using an image classification task. The row CPU mode starts to work at around 80 μs for ten cores running together to perform preprocessing work, such as image reshape, rotation, and normalization. After that, the SNCPU switches to column accelerator mode

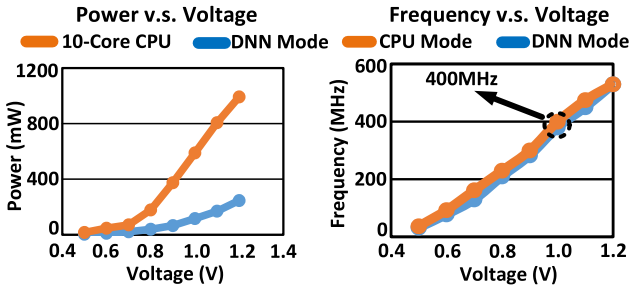


Fig. 16. Measured power and frequency with voltage scaling.

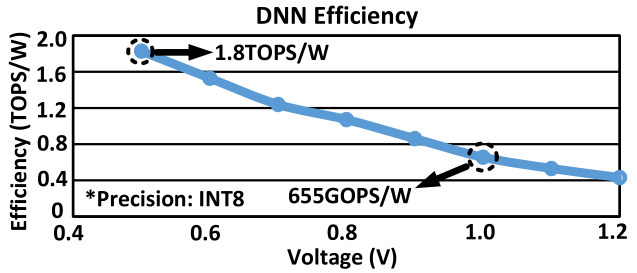


Fig. 17. Energy efficiency of SNCPU for DNN operation with voltage scaling.

to process the first layer of the DNN model for about 60 μs. Then the SNCPU switches to column CPU mode to prepare the input data for the next layer with padding and data alignment. Starting from 380 μs, the SNCPU does the DNN operations of the second layer with row accelerator mode for ten cores. Only the starting of the second layer is shown for the row accelerator mode in Fig. 15.

The PE utilization of each mode is also shown in Fig. 15. The average PE utilization for DNN modes is around 99% and the average PE utilization for CPU modes is around 96%. Total average PE utilization is 97% which is significantly higher than the conventional heterogeneous architectures [11], [12].

Fig. 16 shows the measured power and frequency with the voltage scaled down to 0.5 V. The nominal supply voltage for both DNN modes and CPU modes is 1.0 V with 589-mW ten-core CPU power and 116-mW DNN accelerator power at 400-MHz frequency. Fig. 17 shows the energy efficiency with a scaled supply voltage down to 0.5 V. The results are based on the INT8 bit precision. The DNN modes achieve 655GOPS/W at 1.0 V and increase to 1.8TOPS/W at 0.5 V.

As shown in Fig. 18, there are three reasons for elevated power in CPU modes compared with DNN modes. First, there are several CPU-only memories such as instruction caches, L2 caches, and RFs which added significantly more power consumption. Second, although extensive power gating for DNN modes was performed, the same level of optimization for CPU modes was not carried out due to the stringent tapeout time. Many unused SRAM banks and logic are ungated in CPU mode causing a power waste of about 45% in CPU mode. Third, additional power saving was achieved in DNN mode through programmable clock gating in the design by adjusting the required bit precision. For example, the accumulator in each PE only utilizes 20-b precision (based on the VGG16

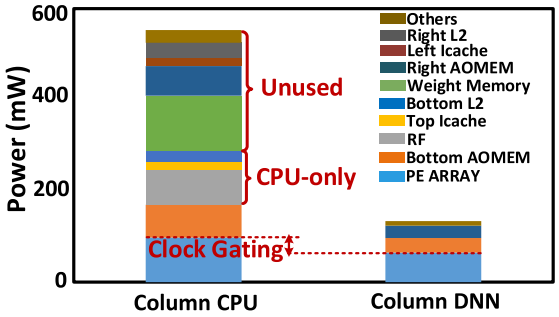


Fig. 18. Power breakdown for column CPU and DNN.

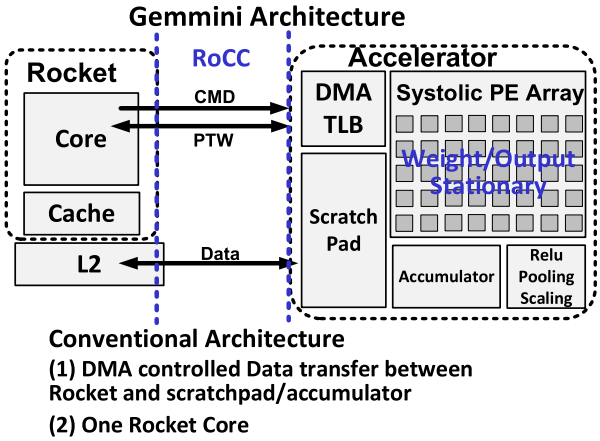


Fig. 19. Description of Gemmini architecture [14].

model evaluated) out of 32-b available logics for DNN modes. Hence, unused bits are gated off through the chip global setting.

C. Benchmark Results

The open-source RISC-V-based heterogeneous architecture, Gemmini [14], is utilized as the reference for comparison with SNCPU. Image classification tasks are evaluated with three different datasets, MNIST, CIFAR10, and ImageNet, using three different NN models, i.e., VGG16, ResNet18, and a simple ELU network. The ELU network with the MNIST database is a much smaller workload which helps to evaluate the performance of our design under a different model setting.

Fig. 19 shows a brief description of Gemmini SoC [14] for comparison. The Rocket core is one open-source in-order scalar processor with RISC-V ISA and five pipeline stages. Here, we use a Rocket core with a two-level MEM system with 32-b RISC-V ISA. RoCC is a RISC-V-based interface connecting the Rocket core with extensional modules. As shown in Fig. 19, by utilizing the RoCC interface, a 2-D systolic PE array with scratchpads and an accumulator is physically connected to the Rocket CPU core as a DNN accelerator. DMA engine is used to control the data transfer between the caches of the Rocket core and the scratchpads or accumulators of the accelerator.

We implemented end-to-end image classification operation using an 8-b quantized model for all three datasets and models.

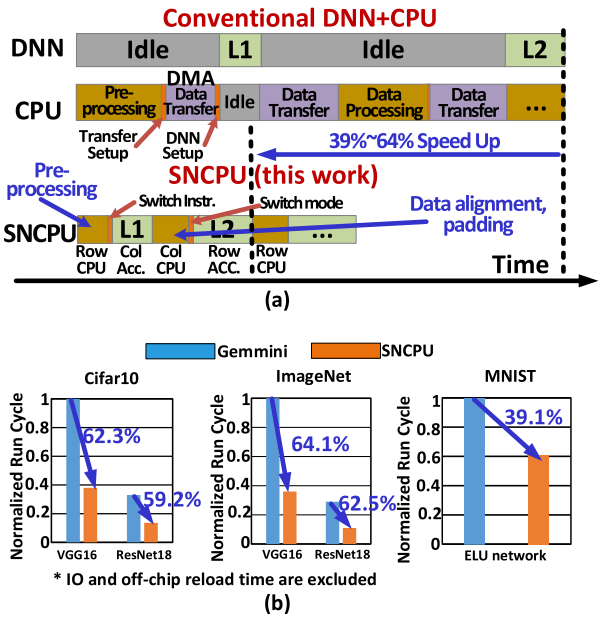


Fig. 20. Performance improvement compared with Gemmini [14]. (a) Explanation for the reasons of benefit for SNCPU. (b) Improvement details for CIFAR10, ImageNet, and MNIST datasets using VGG16, ResNet18, and simple ELU models.

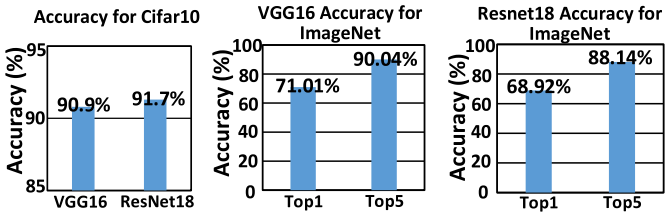


Fig. 21. Benchmark accuracy in this work.

As shown in Fig. 20(a), for each model, certain preprocessing works such as image processing, normalization, and inter-layer data preparation work between layer 1 processing (L1) and layer 2 processing (L2), are needed to be performed by CPU. For conventional heterogeneous architecture, there are two different parts of data transfer. First of all, input data prepared by the CPU needs to be transferred to the input scratchpad of the accelerator. In addition, finished data from the accelerator needs to be sent back to the CPU for general-purpose computation such as batch normalization, data alignment, and padding. The data transfer is controlled by the DMA engine with extra power consumption and takes around 30% of the run cycles in total. It also needs to pay extra run cycles for setting up the data transfer and the initialization of the accelerator before the work starts. Data transfer between different cores and the imbalance of CPU workload and accelerator workload cause the idle time for both the accelerator and CPU pipeline core. Especially for the DNN accelerator, it can reduce the core utilization rate and sacrifice the end-to-end latency of the image classification tasks.

In SNCPU, dual-mode bi-directional dataflow can keep data locally for both CPU and DNN processing avoiding data transfer effort compared with conventional architecture. Also, the unified architecture of the CPU and DNN accelerator eliminates the idle time for the SNCPU core leading to improvement of the core utilization. Because of the ten-core configuration, SNCPU can provide parallel processing capability for CPU work, which is much faster than the single-core CPU in Gemmini. As shown in Fig. 20, a 39%–64% total latency improvement was observed throughout image classification tasks. By processing identical programs for the same DNN model and the same dataset for both SNCPU and Gemmini design, the detailed improvement for image classification is shown in Fig. 20(b). As for the Cifar10 dataset, SNCPU shows 62.3% latency improvement for VGG16 and 59.2% for ResNet18. The SNCPU achieves 64.1% latency improvement for VGG16 and 62.5% for ResNet18 for ImageNet. The average latency benefit is around 61% due to 30% from the removal of data transfer and 31% from parallel processing by multi-core CPU. The workload of CPU processing impacts the latency improvement from SNCPU. Benefits drop from 61% to 39.1% with the three-layer FC ELU network on the simple MNIST dataset. The dropped benefit is due to the less CPU workload for preprocessing and less inter-layer data movement on the FC layers and in the simpler MNIST dataset. Because of the limitation of the test chip, MB level on-chip SRAM size which is enough for one layer processing of well-known DNN model such as VGG cannot be afforded. The benefit results assume that SNCPU and the heterogeneous architecture have the same SRAM size and the data reload latency from off-chip MEM such as DRAM is ignored. The reason is that off-chip data reload is necessary and similar for both architectures if the on-chip SRAM size is the same. SNCPU can only save on-chip data transfer as well as CPU processing acceleration. If considering off-chip data reload time, it may dominate the end-to-end latency for both architectures which is beyond the scope of chip design and make the discussion really complicated.

As shown in Fig. 21, the VGG16 and ResNet18 for the Cifar10 dataset can achieve around 91% accuracy with INT8 bit precision by using SNCPU. VGG16 for ImageNet dataset has 71.01% top 1 accuracy and 90.04% top 5 accuracy. ResNet18 can reach 68.92% top 1 accuracy and 88.14% top 5 accuracy on the ImageNet dataset on SNCPU.

D. Comparison Results

The comparison results are shown in Table I. Compared with prior reconfigurable binary NN (BNN) accelerator-based design [17], SNCPU achieves much higher throughput and can support the most commonly used DNN models. Compared with regular accelerator design [2], [3], SNCPU can support general-purpose computing which is important for real-time embedded applications. Table I also shows the comparison with three prior RISC-V-based heterogeneous SoC works, SamurAI [23] (1 × RISC-V + 1 × accelerator), Vega [24] (9 × RISC-V + 1 × accelerator) and Dustin [25] (16 × RISC-V). Compared with those SoC works, the DNN efficiency for 8 b is compatible with existing SoC demonstration at 0.66GOPS/W at 1 V and 1.8TOPS/W at 0.5 V. Compared with the 16-core RISC-V design in Dustin, the

TABLE I
COMPARISON WITH PRIOR WORK

	[17] NCPU	[2] Eyeriss	[3] DNPU	[24] SamurAI	[25] Vega	[26] Dustin	SNCPU
Process	CMOS 65nm	CMOS 65nm	CMOS 65nm	CMOS 28nm	CMOS 22nm	CMOS 65nm	CMOS 65nm
Area (mm2)	2.86	12.25	16	4.5	12	10	4.47
Architecture	CPU/BNN	CNN	CNN,FC,RNN	Hete. CPU+DNN	Hete. CPU+DNN	Hete. CPU	CPU/DNN
CPU	1x RISC-V	-	-	1x RISC-V	9x RISC-V	16x RISC-V	10x RISC-V
Application	IoT GP BNN	DNN	DNN	IoT GP DNN	IoT GP DNN	IoT GP DNN	IoT GP DNN
DNN Power	241mW	278mW	279mW	96mW (total power)	49.4mW (total power)	-	116mW
CPU Power	106mW	-	-	24.5mW (1x)	-	156mW (16x)	589mW (10x)
Int Precision	2,32	16	16	8,16,32	8,16,32	2,4,8,16,32	8,32
Max Freq.	960MHz	250MHz	200MHz	350MHz	450MHz	205MHz	400MHz
Supply Voltage	0.5-1.0V	0.82-1.17V	0.77-1.1V	0.45-0.9V	0.5-0.8V	0.8-1.2V	0.5-1.0V
SRAM	55.5KB	181KB	290KB	464KB	128KB (L1)	208KB	150KB
CPU Perf. (GOPS)	0.96 @1.0V (8b)	-	-	1.5 @ 0.9V (8b)	15.6 @ 0.8V (8b)	15 @ 1.2V (8b)	16 @ 1.0V (8b)
DNN Efficiency (TOPS/W)	1.6 @ 1V(1b) 6.0 @ 0.4V(1b)	0.24 @ 1V(16b)	1.0 @ 1.1V(16b) 2.1 @ 0.77V(16b)	0.38 @ 0.9V (8b) 1.3 @ 0.5V (8b)	0.9 @ 0.8V(8b) 1.35 @ 0.6V (8b)	0.303 @ 0.8V (8b) 1.15 @ 0.8V(2b)	0.66 @ 1V(8b) 1.8 @ 0.5V(8b)

TABLE II
COMPARISON WITH EXISTING CPUS

	[18] Microchip	[19] TI	[20] Microchip	[21] SiFive	SNCPU (CPU)
Datapath	**32b**	**32b**	**8b**	**32b**	**32b**
CPU	**ARM**	**ARM**	**RISC-V**	**RISC-V**	**RISC-V**
Pipe Stage	**8**	**3**	**2**	**5**	**5**
Voltage (V)	**1.26**	**3**	**3**	**1**	**1**
Freq (MHz)	**600**	**48**	**64**	**250**	**400**
Power (mW)	**229**	**22.8**	**37.2**	**150**	**58.9**
Performance (DMIPS/MHz)	**1.57**	**1.22**	**0.25**	**1.61**	**0.93**
Efficiency (DMIPS/mW)	**4.11**	**2.57**	**0.43**	**2.68**	**6.31**

performance of CPU mode of SNCPU achieves 16GOPS at 1 V which is similar to the performance of Dustin. The power of SNCPU is higher than Dustin due to the lack of optimization in CPU mode in SNCPU.

Since one CPU core in CPU modes of SNCPU is an in-order scalar processor, Table II lists the comparison of the CPU performance of SNCPU with four similar types of low-cost embedded commercial processors [18], [19], [20], [21]. In order to support the related general-purpose computing for DNN, complicated MEM system and pipeline optimizations are not very critical compared with commercial processors, which also provide competitive performance and power efficiency for CPU modes of SNCPU.

VI. CONCLUSION

This article presented a unified architecture, i.e., SNCPU, combining the systolic DNN accelerator and the RISC-V CPU core. SNCPU is designed based on a 10 × 10 2-D systolic array which can be reconfigured to a ten-core 32 b in-order RISC-V CPU. The design achieves over 95% PE utilization for ML tasks. It is implemented by using logic sharing inside the PE array and MEM reuse with 9.8% area overhead and 15% power overhead for DNN modes. A test chip was fabricated using 65-nm CMOS technology under 1.0-V supply voltage and 400-MHz frequency. The SNCPU achieves an energy efficiency of 655GOPS/W to 1.8TOPS/W from 1.0 to 0.5 V. Based on the evaluation using popular DNN datasets and models, the SNCPU achieves a 39%–64% speedup for end-to-end image classification tasks due to the enhanced data locality and parallel data processing.

REFERENCES

[1] K. Ueyoshi et al., "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 160–161.

[2] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2016.

[3] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 240–241.

[4] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[5] T. Jia, Y. Ju, and J. Gu, "31.3 A compute-adaptive elastic clock-chain technique with dynamic timing enhancement for 2D PE-array-based accelerators," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 482–484.

[6] H. E. Sumbul et al., "System-level design and integration of a prototype AR/VR hardware featuring a custom low-power DNN accelerator chip in 7nm technology for codec avatars," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2022, pp. 1–8.

[7] Y.-D. Chih et al., "16.4 an 89TOPS/W and 16.3TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 252–254.

[8] J. Yue et al., "14.3 A 65nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 234–236.

[9] J. Wang et al., "A 28-nm compute SRAM with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 76–86, Jan. 2020.
[10] X. Si et al., "15.5 A 28nm 64Kb 6T SRAM computing-in-memory macro with 8b MAC operation for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 246–248.
[11] T. Karnik et al., "A cm-scale self-powered intelligent and secure IoT edge mote featuring an ultra-low-power SoC in 14nm tri-gate CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 46–48.
[12] R. Eki et al., "9.6 A 1/2.3inch 12.3Mpixel with on-chip 4.97TOPS/W CNN processor back-illuminated stacked CMOS image sensor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 154–156.
[13] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.
[14] H. Genc et al., "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 769–774.
[15] S. L. Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, and D. Brooks, "SMAUG: End-to-end full-stack simulation infrastructure for deep learning workloads," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 4, pp. 1–26, Nov. 2020.
[16] P. N. Whatmough et al., "A 16nm 25 mm2 SoC with a 54.5x flexibility-efficiency range from dual-core arm cortex-A53 to eFPGA and cache-coherent accelerators," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C34–C35.
[17] T. Jia, Y. Ju, R. Joseph, and J. Gu, "NCPU: An embedded neural CPU architecture on resource-constrained low power devices for real-time end-to-end performance," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 1097–1109.
[18] Online Resource, Microchip. *ATSAMA5D44*. Accessed: Mar. 23, 2022. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATsama5d44
[19] Online Resource, Texas Instruments. *MSP432P401R*. Accessed: Mar. 23, 2022. [Online]. Available: https://www.ti.com/product/MSP432P401R
[20] Online Resource, Microchip. *PIC18F13K22*. Accessed: Mar. 23, 2022. [Online]. Available: https://www.microchip.com/wwwproducts/en/PIC18F13K22
[21] Online Resource, SiFive. *E31*. Accessed: Mar. 23, 2022. [Online]. Available: https://www.sifive.com/cores/e31
[22] Y. Ju and J. Gu, "A 65nm systolic neural CPU processor for combined deep learning and general-purpose computing with 95% PE utilization, high data locality and enhanced end-to-end performance," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 1–3.
[23] I. Miro-Panades et al., "SamurAI: A 1.7MOPS-36GOPS adaptive versatile IoT node with 15,000× peak-to-idle power reduction, 207ns wake-up time and 1.3TOPS/W ML efficiency," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2020, pp. 1–2.
[24] D. Rossi et al., "Vega: A ten-core SoC for IoT endnodes with DNN acceleration and cognitive wake-up from MRAM-based state-retentive sleep mode," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, Jan. 2022.
[25] A. Garofalo et al., "A 1.15 TOPS/W, 16-cores parallel ultra-low power cluster with 2b-to-32b fully flexible bit-precision and vector lockstep execution mode," in *Proc. IEEE 47th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2021, pp. 267–270.

Yuhao Ju (Student Member, IEEE) received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2017, and the M.S. degree from Northwestern University, Evanston, IL, USA, in 2019, where he is currently pursuing the Ph.D. degree in computer engineering.

His current research interests include computer architecture and machine learning accelerator design.

Jie Gu (Senior Member, IEEE) received the B.S. degree from Tsinghua University, Beijing, China, in 2001, the M.S. degree from Texas A&M University, College Station, TX, USA, in 2003, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, USA, in 2008.

He was an IC Design Engineer with Texas Instruments, Dallas, TX, USA, from 2008 to 2010, with a focus on ultralow-voltage mobile processor design and integrated power management techniques. He was a Senior Staff Engineer with Maxlinear, Inc., Carlsbad, CA, USA from 2011 to 2014, focusing on low-power mixed-signal broadband SoC design. He is currently an Associate Professor with Northwestern University, Evanston, IL, USA. His current research interests include novel circuits and architectures for emerging applications.

Dr. Gu was a recipient of the NSF CAREER Award. He has served as program committee and conference co-chair for numerous low-power design conferences and journals, such as ISPLED, DAC, ICCAD, and ICCD.