NCPU: An Embedded Neural CPU Architecture on Resource-Constrained Low Power Devices for Real-time End-to-End Performance

Tianyu Jia, Yuhao Ju, Russ Joseph, Jie Gu

Northwestern Univeristy, Evanston, IL

{tianyujia2015, yuhaoju2017}@u.northwestern.edu, rjoseph@eecs.northwestern.edu, jgu@northwestern.edu

Abstract— **Machine learning inference has become an essential task for embedded edge devices requiring the deployment of costly deep neural network accelerators onto extremely resource-constrained hardware. Although many optimization strategies have been proposed to improve the efficiency of standalone accelerators, the optimization for end-to-end performance of a computing device with heterogeneous cores is still challenging and often overlooked, especially for low power devices. In this paper, we propose a unified reconfigurable architecture, referred as Neural CPU (NCPU), for low-cost embedded systems. The proposed architecture is built on a binary neural network accelerator with the capability to emulate an in-order RISC-V CPU pipeline. The NCPU supports flexible programmability of RISC-V and maintains data locally to avoid costly core-to-core data transfer. A two-core NCPU SoC is designed and fabricated in a 65nm CMOS process. Compared with the conventional heterogeneous architecture, a single NCPU achieves 35% area reduction and 12% energy saving at 0.4V, which is suitable for low power and low-cost embedded edge devices. The NCPU design also features the capability of smooth switching between general-purpose CPU operation and a binary neural network inference to realize full utilization of the cores. The implemented two-core NCPU SoC achieves an end-to-end performance speed-up of 43% or an equivalent 74% energy saving based on use cases of real-time image classification and motion detection.**

Keywords— Reconfigurable architecture, embedded systems, ultra-low power device, binary neural network, RISC-V, end-to-end performance, SoC silicon validation.

I. INTRODUCTION

A growing gap can be seen between computing demands and the availability of hardware resources in low power embedded systems. On one hand, the immense impact of ubiquitous computing is unfortunately confined by the power and cost constraints of hardware devices. Many emerging applications in the Internet-of-Things (IoT) [1-2] and in sensor networks [3] rely on energy-scavenging devices with extremely limited and unpredictable power sources to perform critical measurement, computation, and communication tasks. Wearables and biomedical devices also are constrained by very tight energy consumption budgets and form factors [4-5]. On the other hand, machine learning (ML) tasks, such as deep neural network (DNN), have become a widely deployed and often exert essential workload on systems from datacenters all the way down to low power mobile devices [6-8]. Unfortunately, the resource demand from such ML applications is often prohibitively high for low-power low-cost embedded systems.

To support the heavy workload of ML tasks, recent embedded SoCs normally adopt a heterogeneous architecture which consists of both general-purpose processors, e.g. CPUs, and domain-specific accelerators, such as DNN accelerator [9-13], as shown in Fig. 1. However, for resource-constrained embedded systems, e.g. smart sensors, micro drones, which have limited form factors and power budget, deploying a DNN accelerator core is often too costly due to high consumption of power and area overheads. As reported in many industry-grade embedded or mobile SoCs, the neural network accelerator engines could consume a few times more area and power than a CPU core due to the large number of MAC units and the large SRAM storage space required by ML accelerators [10-11].

In addition to the power and area cost, the heterogeneous architecture is often bottlenecked by the expensive ML task offloading and CPU/accelerator under-utilization. To accelerate computation of intensive tasks for ML, these tasks are commonly offloaded from CPU to accelerators [23-24]. However, the unbalanced workload between CPU and accelerators could significantly degrade the core utilization rate and impact end-to-end execution latency. For instance, the CPU data pre-processing could take more than 60% runtime in various heterogenous SoCs, as shown in Fig. 1. Unfortunately, although many schemes have been explored to improve the efficiency of standalone DNN accelerator [17-20], there has been very limited optimization for end-to-end performance of heterogeneous CPU + accelerator cores [21-23]. In [15, 23], the accelerator coherency port (ACP) has been utilized to allow accelerators to directly request data from CPU's LLC and reduce the data transfer cost. In [14], the RoCC interface has been deployed for flexible and tight on-chip communication between RISC-V cores and the customized BNN accelerator cores.

As a result of the significant design cost and utilization challenges for the DNN accelerators, a recent survey shows that the majority of mobile edge devices on the market are still relying on CPUs to perform machine learning inference, even though there are dedicated accelerators or GPUs designed within the SoC [24]. For resource-constrained devices, it is worthwhile

978-1-7281-7383-2/20/$31.00 ©2020 IEEE
DOI 10.1109/MICRO50266.2020.00091

to evaluate the tradeoff between performance benefit and the design cost for a dedicated neural network accelerator. It is also critical to develop a new edge SoC architecture which could offer both general-purpose CPU operation and ML inference efficiently with high core utilization and sufficient flexibility for programming.

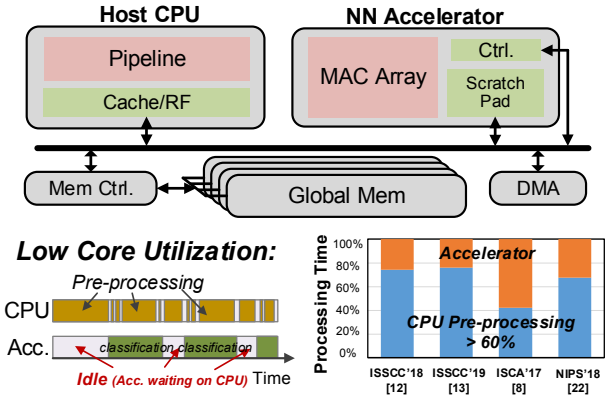


Fig. 1. Low core utilization in the conventional heterogeneous SoC.

In this paper, we propose a novel reconfigurable architecture for resource constrained systems aiming to close the significant design gap between general-purpose CPUs and neural network accelerators. Compared to the previous designs targeting to improve the latency of core-to-accelerator communication [14-15], our work has explored a new architecture solution to realize both the general-purpose CPU and BNN operations in a single core obviating the need for a complex heterogeneous interface. The benefit of such an architecture is to obtain not only cost reduction for edge devices but also to ensure higher core utilization in order to improve end-to-end performance. The proposed design has been fabricated using a 65nm CMOS process and fully verified with the operating voltage ranging from 1V down to the ultra-low power regime of 0.4V. There are two major benefits that have been obtained by NCPU as compared to baseline heterogeneous architecture including separate CPU and BNN accelerators. First, a single NCPU core achieves a 35% area reduction and 12% energy saving while maintaining the functionality and efficiency of both baseline cores, rendering significant cost advantages in resource-constrained edge devices. Second, with less than a 3% area overhead, two NCPU cores SoC achieve full core utilization while avoiding the ML task offloading. With that, a 43% end-to-end performance or equivalent 74% energy improvement is obtained in real-time image and motion detection use cases.

The contributions of this work are summarized as below:

- For resource-constrained edge devices, a novel NCPU architecture is proposed to leverage the existing logic and memories inside a neural network accelerator to recover the capability of conventional CPU pipeline operations. As a result, a NCPU core supports both ML inference and general-purpose CPU computing with efficiency similar to the respective architectures.
- The proposed NCPU architecture is designed to fully support the 32-bit RISC-V Base ISA. A customized RISC-V instruction set extension is developed to incorporate BNN operations, data transferring and mode switching.
- A special zero-latency transition scheme is developed to support seamless switching between CPU and BNN modes by essentially pipelining the reconfiguration. Data can remain in place while the core is reconfiguring thereby eliminating transfer between CPU and accelerator.
- A two NCPU core SoC chip is designed and fabricated using 65nm CMOS technology. Measured performance is compared with baseline conventional heterogeneous design. Real-life use cases for image classification and motion detection are used to demonstrate the energy and end-to-end benefits of the proposed architecture in the embedded system.

II. Motivation and Design Method

A. Heterogeneous SoC for Edge Devices

As described above, the resource constrained embedded SoC has extremely low power and cost budgets for various applications, such as IoT devices. In these types of devices, the total power budget for the computing activities is tens of *mW* or less [53-54]. Conventionally, a CPU is implemented in microcontrollers which can be programmed for various embedded applications [53-56]. With the recent workload requirements for machine learning applications, the support of DNN operation becomes critical for embedded systems, with various industrial products [10-13, 25-26].

To evaluate the benefits of the dedicated neural network accelerator, we built a real-time use case for human motion detection using the sensor data from Ninapro database [59]. In this task, the CPU extracts features from the sensor data and then perform a BNN inference with a 74% classification accuracy. This use case has been tested on either an in-house designed standalone RISC-V CPU or a heterogeneous design including both CPU with a dedicated BNN accelerator.

The measured results from our prototype chip are listed in Table 1. For the real-time application, there is a stringent real-time latency requirement, i.e. 5ms, for each motion detection. It is observed that standalone CPU requires 32ms for feature extraction and BNN inference for a single motion detection and fails to meet the latency requirement. With the help of the ASIC BNN accelerator, the real-time latency and energy-efficiency are improved by 59X to 0.54ms and 36X to 0.58uJ, respectively. Therefore, for some real-time applications, a ML accelerator is indispensable to improve performance as the standalone CPU fails to meet the latency requirement.

TABLE I. Measured Latency and Energy Consumption for a Motion Detection Task with the Real-Time Latency Requirement of 5ms.

	Latency (ms)	Energy (uJ)
Standalone CPU	32	21.12
CPU w/ BNN acc.	0.54	0.58

B. Design Challenges for Resource Constrained SoC

There are several design challenges for heterogeneous SoCs including both CPU and ML accelerator for the resource constrained embedded devices. First, the DNN accelerators are expensive in many respects. For instance, in Intel's 16nm ultra-low power embedded SoC [10], the neural accelerator engine consumes about the same area as the CPU core, delivering up to 1TOPS performance for the neural network. In Samsung's 8nm mobile SoC [11], the DNN accelerator has more than 2X larger

area than the host CPU and consumes 39mW even at 0.5V. For the resource constrained embedded SoCs, the area and power of DNN accelerators become the limiting factor for their adoption.

Second, CPU often dominates end-to-end performance causing the DNN accelerator to be under-utilized. For example, in Intel's IoT edge SoC [12], the CPU operation for data pre-processing takes about 70ms, while the classification of each feature in CNN accelerator only takes 5ms. As a result, the CNN accelerator utilization rate approaches 24%, while remaining idle during the rest of the time, as shown in Fig. 1. Similarly, Microsoft's study also shows that the CPU data pre-processing time could take 67% of runtime, significantly impacting the end-to-end performance [21]. In addition, the costly task offloading from CPU to accelerator often becomes a performance bottleneck limiting the overall improvement from accelerator [23-24]. As implied by Amdahl's Law, the optimizations focusing on standalone DNN accelerator is insufficient to improve the performance of the whole system [27-28]. A novel architecture design is needed to improve the end-to-end performance for the resource-constrained devices.

C. Architecture Design Methodology in This Work

Recently, to close the architecture design gap between CPU and ML accelerators, there are some CPU developments particularly for ML tasks, such as the AMX unit in recent Apple's A13 CPUs [57], or other special in-pipeline support for ML [58]. Some special heterogeneous interfaces, e.g. RISC-V RoCC [14] or ACP [15], have been deployed to the SoC to provide flexibility to accelerators for data transfer. With these interfaces, the accelerators can directly transfer data to/from the CPU caches. Therefore, the data transfer latency is much shorter compared with the conventional DMA access to DDRs. In [16], a flexible and efficient coherence interface is also proposed for heterogeneous cores. However, such interfaces increase the design complexity and consume additional area and power due to complex asynchronous logic and communication protocols.

Different from the above approaches, to meet the low cost and low power requirements of edge devices, this work proposes a design approach that migrates CPU functionalities into an ASIC accelerator design. We used a BNN accelerator as starting baseline architecture which maintains the highest efficiency for neural network inference. The CPU instruction support is then added into the accelerator design to recover the general-purpose computing with very small overhead. The benefit of such an architecture is that it maintains the efficiency of accelerator while still support CPU operation leading to a low cost and high throughput architecture for embedded devices. Compared with the previous SoC design with heterogeneous interfaces [14-15], the NCPU saves significant area cost for resource-constrained devices by both reducing the required cores and avoiding heterogeneous interface design.

It is worth mentioning that the challenges of bridging the architecture design gap between CPU and accelerator vary with the complexity of the baseline architectures. There is a wide open architecture exploration space ranging from simple low-end microprocessor and accelerator to more complicated multi-thread CPUs and neural engines. Depending on different baseline architectures, different architecture design methodology might be needed to couple the heterogeneous cores tightly, such as the RoCC interface used previously [14]. In this work, partially due to the high cost of chip fabrication, we only explored one extreme reconfigurable design for very low cost energy scavenging edge devices. A large design space is yet to be studied for many different application domains. We leave the further exploration of the design space to our future work. Despite of the limitation, to our best knowledge, this work is the first silicon implementation for a reconfigurable architecture which can efficiently operate and smoothly switch between general-purpose CPU and neural network inference.

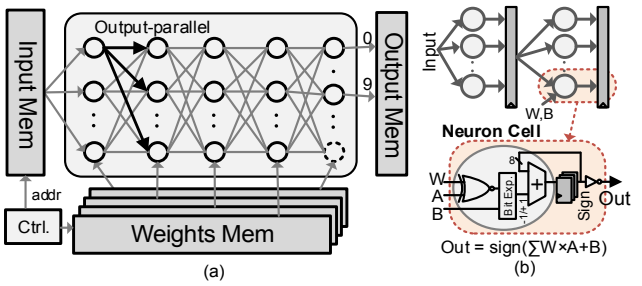


Fig. 2. (a) The baseline BNN accelerator architecture, (b) the hardware implementation of multi-layer BNN, and the detailed design of neuron cell.

III. Binary Neural Network on Edge

The binarized neural network (BNN) has been proposed as an attractive low-cost solution for low power embedded applications. BNN constrains the weights and inputs to be only +1 and -1, which significantly reduces the weight memory cost with marginal degradation in accuracy [39]. In addition, the multi-bit multiplier is replaced by a simple XNOR gate rendering significantly smaller neuron cell design. Due to its advantages in low cost and power consumption, BNN accelerators attract tremendous enthusiasm in chip demonstrations for various embedded applications, such as image classification [40-41], voice detection [42], etc. BNN has also been implemented into low-power industrial products. For instance, BittWare's product is an industry example of using BNN on FPGA showing 100X saving [43]. As BNN only has small accuracy loss for simple tasks, e.g. 3% loss for MINIST, but brings more than 10-100X lower cost and power compared with DNN, it is a good compromise for ultra-low power IoT or edge AI devices. Hence, we select BNN as the baseline of our neural CPU architecture.

Fig. 2(a) shows the architecture of BNN accelerator used in this work. Similar to the previous designs [40, 44], the input and weight values are fetched directly from SRAM memory and sent to the XNOR neurons. The output value of a single neuron is shared with all the neurons at the following layer to increase the data parallelism. The pipelined multi-layer BNN is built to propagate the input from left to right through all the layers and generate the final classification result at the last layer. The registers in each neuron cell, as shown in Fig. 2(b), store the intermediate input values for the next network layer. This is analogous with conventional CPU pipeline operation, which propagates different instructions in sequence all the way to the last pipeline layer to commit. As a result, the depicted BNN architecture resembles the existing pipelined CPU architecture and brings about an opportunity to reconcile the two distinct computing models. The selection of the neuron numbers of the BNN accelerator needs to consider both the BNN model accuracy and the size compatibility with the CPU core. In this

work, a 4-layer BNN accelerator with 100 neurons in each layer is selected with consideration of the targeted embedded applications and the RISC-V CPU pipeline constructions. The selection of the baseline BNN topology will be further discussed in Section 8.1.

IV. Neural CPU Architecture

A. Neural CPU: A CPU Pipeline Emulator

For the target CPU model, we used open-source RISC-V ISA, i.e. 32-bit Base Integer RV32I, which is highly suitable for ultra-low power embedded edge devices due to its amenability to a simple pipelined implementation, low power consumption and high portability [45]. The proposed NCPU architecture recovers the full functionality of the CPU pipeline on top of the baseline BNN accelerator by reconfiguring the existing logic and memory banks with small hardware overhead. As shown in Fig. 3, a four-layer neural network is built as the baseline BNN accelerator to intentionally match a 5-stage in-order RISC-V CPU pipeline. For more than four layers of BNN, the output layer results can be wrapped back to the first layer for the deeper neural network. An in-house designed 5-stage in-order pipeline, which is similar to the RISC-V Rocket core [46], is fused into the BNN accelerator by modifying the data path and binary neuron of every layer. As a result, a neural pipeline is built as a hardware emulator of the RISC-V pipeline operation. The detailed implementation at each neuron layer and their reconfiguration capabilities are explained as follows.

Stage 1 - NeuroPC: Program Counter

Similar to the conventional CPU pipeline, the first neural stage is used to perform the program counter (PC) for fetching the incoming instruction. In most cases, the PC stage is only performing "+4" operation, which is an *ADD* operation. Therefore, 4 neuron cells are connected in series with the self-feedback at the last neuron to realize "+4". The existing adder inside the neuron cell MAC is reused, with each neuron generating 8 bits of the PC. For supporting branch address generated from the following *Execution* stage, a mux is added for branch taken operation.

Stage 2 - NeuroIF: Instruction Fetch

Part of the first neural layer is also reconfigured to emulate the CPU's IF stage. As the PC address is sent to the instruction cache, the instructions are read out and stored. Therefore, the neuron cells are reconfigured as bypass cells, which pass the incoming values directly to the output. The registers inside the neuron cells at the *NeuroIF* stage are reused to store the fetched instructions, with only one additional mux to select the register data source.

Stage 3 - NeuroID: Instruction Decode

The ID stage decodes instructions into partial codes such as opcode, function code, register sources and destinations, etc. The decode field is connected to the XNOR gate with the original weights. To decode particular information, such as the opcode, a group of neuron cells is combined to XNOR the instruction field with the weights of neural network. As a result, a mapping between the instruction ISA and decoded opcodes, e.g. *ADD*, *SUB* is established using neural network operation. Both the adder and registers inside the neuron cells are reused. In addition, the ID stage also readout the operand values from the register file and store them, which is similar to the bypass cell at the *NeuroIF* stage.

Stage 4 - NeuroEX: Execution

The *NeuroEX* stage emulates different arithmetic or boolean operations as a conventional ALU. Since only an adder and XNOR gate exists inside the original BNN neuron cell, hardware is added to recover the rest of ALU operations including *AND*, *OR*, etc. The CPU operations that require similar resources, e.g. *ADD*/*SUB*, *LW*/*SW*, are grouped and mapped into the same neuron cells, to reduce unnecessary activation of unused neurons. In addition, a multiplier is also realized at the Execution stages based on existing "adders" inside neurons. For some special CPU control data paths, such as branch resolution and operand forwarding, these data paths are implemented by the conventional digital design to minimize the area overhead.

Stage 5 - NeuroMEM: Memory Access

The functionality of the *NeuroMEM* stages is mainly reading or writing the data from/to the data cache. Hence, the neuron cells propagate results similar to the *NeuroIF* bypass neurons. Based on the opcode type, the read/write enable signals are sent to the data cache for the memory operations, which will be discussed in the next section. Following the *NeuroMEM* stage, the computation results are written back to the register file and the instruction is committed.

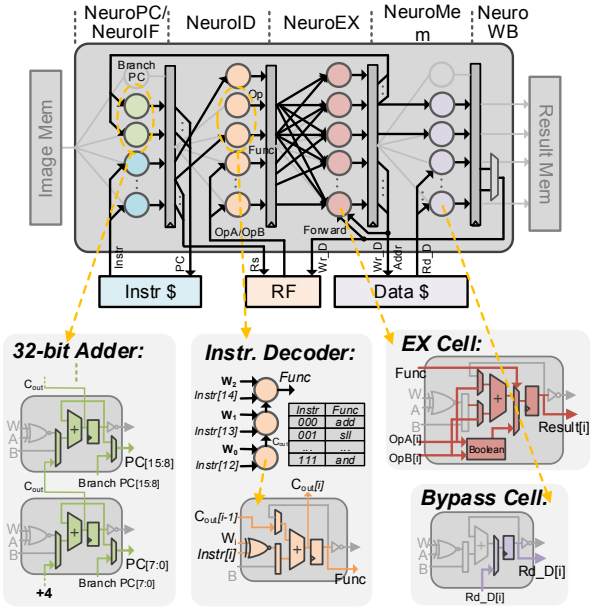


Fig. 3. Top-level architecture of the Neural CPU with its data path at each pipeline stage.

37 RISC-V base instructions and 5 additional customized instructions for BNN modes are supported in the NCPU. For each supported instruction, we recover the required logic in the baseline BNN accelerator stage by stage. For example, to support the decoding of *ADD* (i.e. funct code *000*) in RV32I, the function field Instr[14:12] is XNOR-ed with an inversed neuron weights W[2:0] value of *111* and summed together (i.e. $Instr[14] \odot W[2] + Instr[13] \odot W[1] + Instr[12] \odot W[0]$) in a three neuron cell group, as shown in Fig. 3(a). The output of this

decode neuron group will be asserted to low whenever there is *ADD* decoded leading to addition operations at the following *NeuroEX* stage. In NCPU, each instruction has its corresponding decode neuron group to decode to the target ALU operations. By reusing the existing XNOR and adders in the neuron cell, the hardware overhead has been greatly limited. Besides, the weight values are statically set in the CPU mode for different decoding strategies and hence will not incur any power overhead from memory access.

To support the control instructions, such as conditional branch, some data paths which cross pipeline stages are also recovered. For example, the branch target PC calculated at the *NeuroEX* stage is wired back to the *NeuroIF* stage, as shown in Fig. 3(a). If the branch is taken, the branch PC will be used as the next fetch address. To resolve register data dependencies, the data forwarding paths have also been added between *NeuroEX* and its earlier stages. Recovering these crossing stage data paths only requires simple digital multiplexers with low overhead. The complete data paths under different operation modes are shown in Fig. 4 (a). In NCPU, the *NeuroEX* stage needs the most design modifications, i.e. most area overhead, to recover the full functionality of the CPU ALU. The rest pipeline stages can better reuse the existing BNN accelerator logic and incurs less overhead. The unused neuron cells in the CPU operation mode are clock gated to reduce the power overhead.

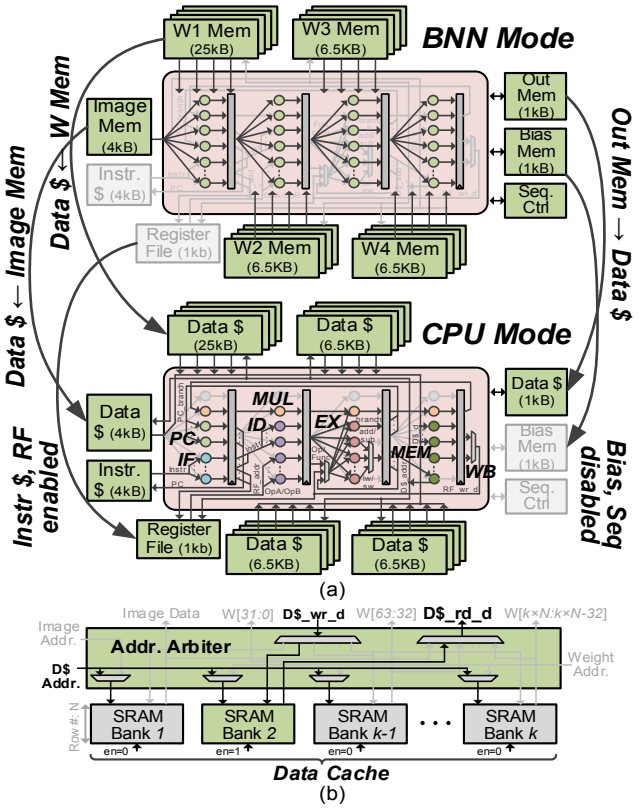


Fig. 4. (a) Memory and data path configuration schemes under BNN and CPU mode, (b) the address arbiter design to support the memory reconfigurations.

B. Memory Reuse Scheme for Neural CPU

Beyond the reconfiguration of the core data path, to save the memory area, the on-chip SRAM memory for the BNN accelerator is also designed to be reused as data cache during CPU operation. Fig. 4(a) shows the memory configuration during both operation modes. In BNN mode, multiple SRAM banks with 32-bit wide port are used to store all the BNN weights and its input image data. Each neuron layer has a group of memory banks to store all the weights and support wide memory bandwidth. There is one dedicated image memory to store the input values and one output memory to store the classification results. The top-level sequence controller is used to generate the unified SRAM address to all the weight memory banks and the sequence control configurations to manage the neural network process. During BNN mode, all the weight memory banks are enabled to support the max image classification throughput.

When switched to CPU mode, all weight memories and the input/output image memories are reused as the data cache. Separate instruction cache and register file are used to store the instruction programs and some initial data. For the data cache operation, an address arbiter is used to enable only one SRAM bank out of all memory banks based on the target address for the read or write operation, as shown in Fig. 4(b). The rest of the unused memory are clock gated to remove the power overhead from weight banks in BNN operation.

The reuse of SRAM banks for both operation modes can not only significantly reduce the total memory capacity requirement for dual operations but also allows CPU/BNN output data to be stored locally without data transfer between the cores, which make the task offloading much simpler. As a result, the cost of data transfer among heterogeneous cores in the conventional design is eliminated.

V. Zero-Latency Switching and ISA Support

A. Zero-latency Switching Between Operation Modes

To establish the zero-latency mode transition between RISC-V CPU operation and the BNN inference operation, a special mode transition sequence with a series of customized instructions is developed for the NCPU, as shown in Fig. 5. At the beginning of task operation, the NCPU stays in CPU operation mode to perform general-purpose computation, such as data pre-processing, configuration calculation, etc. The processed image data is stored into local image memory (reconfigurable as the data cache), and the configurations for accelerator operation stores into special designed transition neuron cells. After completion of CPU pre-processing, the NCPU switches to BNN mode and directly reads the processed data from the image memory and proceed to the classification. Similarly, after the image classification from BNN operation, the NCPU can switch to CPU mode with the classification results directly from the output memory reconfigured as data cache.

For CPU to BNN mode transition, a customized RISC-V instruction *Trans_BNN* is used to switch core operation mode into BNN inference, as Fig. 5(b). To avoid the latency for BNN inference, the weight values of the first neuron layer always reside at one of the weight memory banks. Hence the image inference can start immediately with the layer1 weights after the mode switching, while the weights for the following neuron layers are continuously loaded from the global memory to the local weight SRAM at the same time.

For the BNN to CPU mode transition, the CPU initial data is pre-loaded into data cache before the mode transition happens. While the last image/task in BNN is being inferenced, the DMA engine has acknowledged the upcoming completion of the BNN task (as neural layer 1 is freed) and starts loading CPU initial data from L2. Hence there is no additional latency incurred upon switching to CPU mode. After the NCPU core switches back to the CPU mode, fetch at the current PC resumes to enable the post-processing of the image classification results.

To simplify the cache coherency for resource-constrained devices, a simple software managed data transfer policy is adopted for the NCPU. The developed software run code is responsible for explicitly defining the cache lines that the accelerator is going to read from and/or write to. For small size neural network models, the weights only utilize a portion of weight memory, with the unused weight memory and image memory serve as the data cache for CPU. Hence, no dynamic reconfiguration of SRAM is needed. For large size BNN models with weights fully occupying the whole weight memory, the data cache needs to be dynamically reconfigured into weight cache due to the large model size. As a result, the weights need to be loaded before transitioning into the BNN mode. For such situations, the zero-latency switching scheme is developed to hide the data transfer latency and performance impact during the mode switching.

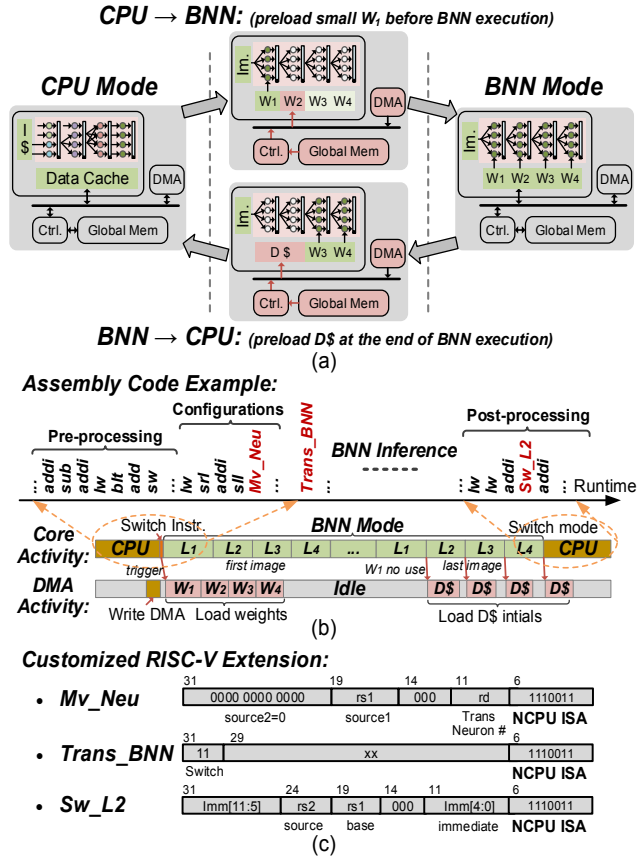


Fig. 5. (a) Operation mode switching with special data pre-loading sequences to achieve zero latency, (b) the assembly code example during the workload runtime, (c) the example of customized RISC-V instruction extension to support NCPU.

B. Customized RISC-V Extension

RISC-V is suitable for ultra-low power embedded edge devices and also supports flexible customized instruction extensions for special purposes. In our work, a series of customized instructions are developed and embedded inside the RISC-V ISA to support special NCPU operations and the mixed mode programming. In general, the last 7 bits of the instruction field are modified to indicate the customized NCPU instructions. Some instructions are introduced as following and in Fig. 5(c).

1. *Mv_Neu*: move the designated register file values to the special design transition neuron located in each neural layer. The transition neurons can store the configurations, e.g. model size, for the neural network operations.
2. *Trans_BNN*: trigger the operation mode of the NCPU core from CPU mode to BNN mode. The instruction will send a special trigger signal to the bus controller, which contains the core mode state.
3. *Sw_L2, Lw_L2*: two special write-through instructions for the data to be stored or loaded directly between the NCPU core and the global L2 memory.
4. *Trigger_BNN*: a special instruction used to trigger the BNN accelerator core operation. This instruction is designed to operate as the conventional heterogeneous architecture for evaluation purposes.

In addition to the above instruction extensions, there are several special transition neuron cells built at each neural layer to support temporal data storage for the operation mode switching. The instruction *Mv_Neu* can store the calculated configuration values e.g. run cycles of each neural network layer, to these transition neuron cells during CPU mode. After the operation mode switched to the BNN inference, the transition neuron values are directly taken as the neural network configurations. This transition neuron cell design enables flexible management using CPU instructions for the following BNN operations.

VI. NCPU Chip Implementation and Performance Evaluation

A. Neural CPU Chip Design

To demonstrate the benefits of both single-core and dual-core configurations, we implemented a two-core design of NCPU, as the top-level SoC architecture shown in Fig. 6. A global memory is shared by two NCPU cores. The memory can serve functionally as an incoherent L2. Each core can access the L2 memory via new customized RISC-V load/store instructions which perform write-through behavior for stores. A DMA engine is designed to manage the data communication between the NCPU cores and the L2 memory. During the workload operations, these two NCPU cores can operate independently for different workload tasks, e.g. CPU programs or classify different images, or operate cooperatively, e.g. form a deeper neural network accelerator by connecting these two NCPU cores in series. We compared the NCPU design with baseline conventional CPU + BNN architecture showing two major benefits: (1) a single NCPU can obtain 35% area reduction and 12% energy saving at 0.4V, (2) two NCPU cores scheme can achieve 43% performance improvement or equivalent 74% energy saving by maintaining both cores at full utilization.

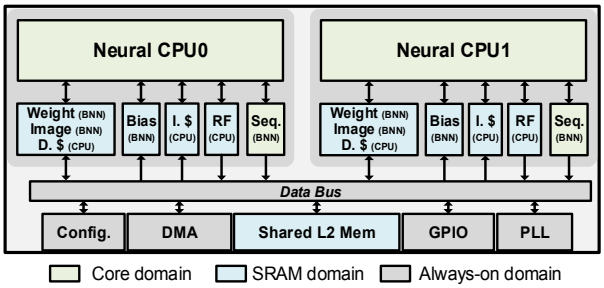
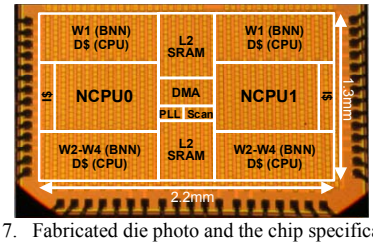
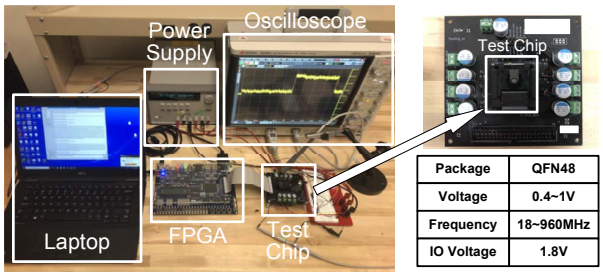


Fig. 6. The top-level architecture of the NCPU chip.



Technology	65nm CMOS
Die Area	2.8mm^2
Frequency	960MHz
Nom Vdd	1V
SRAM	128KB
CPU Power	112mW
BNN Power	241mW
BNN Power (2core)	446mW

Fig. 7. Fabricated die photo and the chip specifications.



Package	QFN48
Voltage	0.4~1V
Frequency	18~960MHz
IO Voltage	1.8V

Fig. 8. Chip measurement setup for the use cases.

B. *Chip Specifications and Measured Performance*

The two-core NCPU test chip has been fabricated in TSMC 65nm GP CMOS technology. The chip die photo and the design specifications are shown in Fig. 7. Two identical NCPU cores have been implemented on chip. The nominal operating frequency for the NCPU core is 960MHz at 1V. The overall active die area is 2.86mm2. The test chip measurement setup is illustrated in Fig. 8. The test chip is packaged by a QFN package and mounted on a PCB board. Altera DE-2 FPGA which communicate with a laptop is used as the interface to chip for the data communication. During the workload operations, the data is first transferred from FPGA to the on-chip global L2 cache, which is then accessed by the cores. The transient power traces for each core were measured to monitor the core activities.

To apply NCPU design for ultra-low power applications, the chip performance and energy consumption for different modes have been measured across a wide supply voltage range down to 0.4V, as shown in Fig. 9. The chip's functionality has been verified by reading out all internal RF and memories after the benchmark run. At 0.4V, the NCPU core can operate correctly at frequency 18MHz, with the power consumption of only 1.2mW for BNN inference and 0.8mW for CPU operations, which is significantly lower than that in Intel's Movidius [10] and other commercial microcontrollers [53-56], as shown in Table 2. The minimum energy point (MEP) for CPU mode is observed at 0.5V, with the leakage power dominating below it. Due to the larger portion of dynamic power for the BNN inference, its MEP point is not observed before a malfunction is observed below 0.4V. The computing efficiency for BNN across voltages is also measured showing 1.6TOPS/W at 1V and a peak efficiency of 6.0TOPS/W at the voltage of 0.4V.

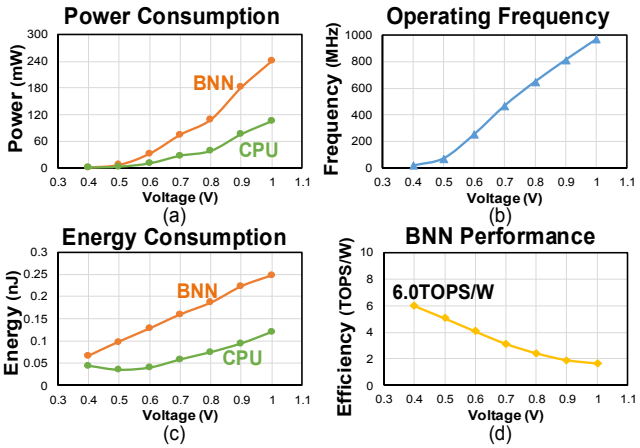


Fig. 9. The measured (a) power, (b) frequency, (c) energy consumption, and (d) power efficiency versus supply voltage for both BNN and CPU modes.

TABLE II. COMPARISON WITH COMMERCIAL MICROCONTROLLERS [53-56].

	[53] Microchip	[54] TI	[55] Microchip	[56] SiFive	NCPU (CPU mode)
Datapath	8b	32b	32b	32b	32b
CPU	RISC	ARM	ARM	RISC-V	RISC-V
Pipe Stage	2	3	8	5	5
Vol. (V)	3	3	1.26	1	0.4-1
Freq (MHz)	64	48	600	250	18-960
Power (mW)	37.2	22.8	229	150	106 (1V) 0.8 (0.4V)
Performance (DMIPS/MHz)	0.25	1.22	1.57	1.61	0.86
Efficiency (DMIPS/mW)	0.43	2.57	4.11	2.68	8.26

TABLE III. COMPARISON WITH STATE-OF-THE-ART ML ACCELERATORS.

	[2] ISSCC'17	[44] ISSCC'19	[40] JSSC'18	[41] ISSCC'18	NCPU (BNN mode)
Process	28nm	65nm	65nm	28nm	65nm
NN Model	FC	FC	FC	Conv.	FC
Datapath	8b	8b	1b	1b	1b
Dataset	MNIST	MNIST	MNIST	CIFAR-10	MNIST
Accuracy	98.36	98.06	90.1	86.05	94.8
Vol. (V)	0.9	0.8	1	0.8	0.4-1
Freq (MHz)	667	20	400	10	18-960
Power (mW)	33.7	23.6	0.6	0.9	241 (1V) 1.2 (0.4V)
Efficiency (TOPs/W)	1.2	3.42	6.0	532	1.6 (1V) 6.0 (0.4V)

Table. 2 compares the performance of RISC-V CPU mode in NCPU with other commonly used commercial embedded microprocessors. The NCPU is able to run at 18MHz at 0.4V for ultra-low power applications, which is in line with other embedded microcontrollers. We believe the performance difference with [56] comes from the different number of computation resources. It can be speculated by SiFive core has 1/4 frequency (250MHz) of NCPU but burns 50% more power than our design. In fact, the efficiency of our design is higher than the designs for standard *Dhrystone* benchmark run. Table. 3 shows the comparison with several previous state-of-the-art ML inference accelerators. The work in [2] is conventional

heterogeneous architecture which requires both the host ARM CPU and a specialized DNN accelerator to support the fully connected neural network. Work [44] is adopting similar neuromorphic accelerator architecture with our work using the fixed 8-bit data path. Comparing with the state-of-the-art standalone BNN accelerators [40-41], the BNN operation mode of NCPU achieves a peak 6.0TOPS/W power efficiency at 0.4V, which is similar to digital based BNN accelerator design [40].

C. *Benefit and Overhead Evaluation*

To evaluate the design overhead for the proposed NCPU, we compare the area, power, speed between the proposed NCPU core with a standalone RISC-V core and a standalone BNN accelerator. The baseline standalone CPU core is an in-house designed single-cycle 5-stage RISC-V pipeline, which has the identical functionality and IPC as the CPU operation in NCPU. The baseline BNN accelerator has 4 layers (100 neurons/layer), which has the same number of neurons as the NCPU. The PPA of the standalone CPU/accelerator cores are evaluated using the final physical design after logic synthesis and place & route.

Fig. 10 shows the area overhead of the NCPU core excluding SRAM and the overhead of the whole NCPU including SRAM. Comparing with the baseline BNN core, the area cost for realizing the NCPU pipeline stages excluding memory is 13.1%. Including both core and the SRAM memories (i.e. including the register file, instruction cache, weight memory, etc.), the overall area overhead of the NCPU design is only 2.7%. Comparing with a standalone BNN or a CPU core, the performance of the NCPU, i.e. maximum operation frequency, degrades only 4.1% and 5.2% for the two operating modes, respectively.

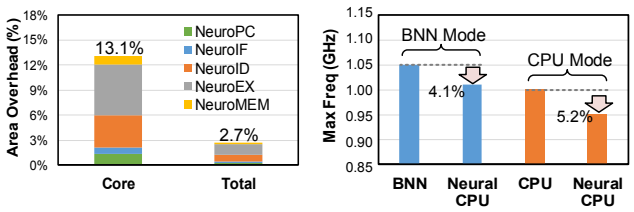


Fig. 10. The area and performance overhead of NCPU comparing with standalone CPU or BNN accelerator core.

The power overhead evaluation is reported from dynamic power analysis by PrimeTime based on the cycle-by-cycle gate level simulation. The power consumption of the proposed NCPU is compared with a single standalone BNN accelerator, or a 5-stage RISC-V pipeline core design. Fig. 11(a) shows the comparison of power consumptions. During BNN operation mode, the proposed NCPU consumes 5.8% more power than the standalone BNN accelerator for MNIST dataset inference due to the extra inserted CPU logic. For CPU operations, multiple embedded programs from the MiBench benchmark suite have been tested [47], which show about 15% more power cost than a single CPU core. Fig. 11(b) also compared the power consumption for all supported RISC-V base instructions individually. An average of 14.7% more power is consumed for various instructions. The extra power cost mainly comes from the dynamic power of some ungated original logic inside each neuron cell.

Although there are small hardware overheads between the NCPU and the standalone CPU or BNN core, the NCPU can achieve the area saving and energy reduction benefits when comparing with the conventional heterogeneous architecture. Fig. 12 compares the area among standalone RISC-V CPU core, standalone BNN accelerator and the NCPU. Comparing with the heterogeneous architecture including both CPU and BNN accelerator, a NCPU achieves 35.7% area reduction while maintaining the same functionality. The area saving is just one benefit of our reconfigurable NCPU architecture. As shown later, significant end-to-end performance gain (or equivalent energy saving) can also be obtained, addressing the often overlooked core under-utilization issue in conventional heterogeneous architecture.

As shown in Fig. 12(b), the reconfigurable design leads to 7.2% energy overhead of NCPU comparing with baseline heterogeneous cores at 1V for a MNIST inference task. However, as the leakage energy starts to dominate total energy consumption at ultra-low voltages, the area saving starts to convert to an energy saving below 0.6V and achieves 12.6% energy saving at 0.4V for the image inference task. It worth to mention that the NCPU core and SRAMs are scaled down together from nominal 1V to 0.55V. For the range below, we only scale NCPU core voltage with SRAM stays at 0.55V due to the Vmin limitation of SRAM. For the baselines, we did the same voltage scale approach to have a fair comparison.

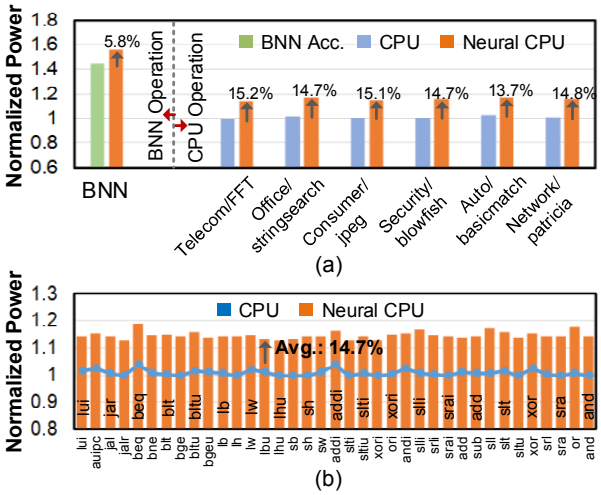


Fig. 11. (a) Power consumption comparison for both BNN acceleration mode the CPU operation mode, (b) power consumption overhead for the supported RV32I instructions.

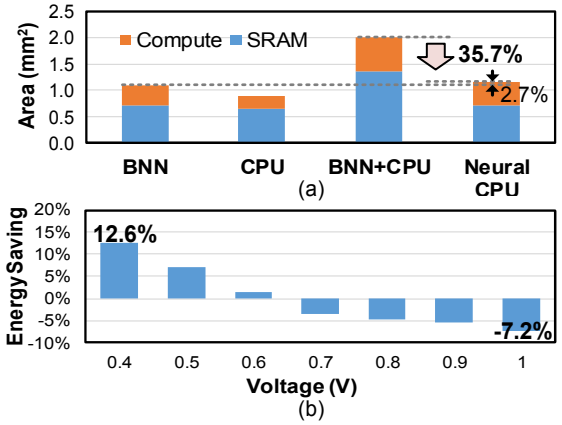


Fig. 12. (a) Area reduction and (b) energy saving benefit of the Neural CPU.

VII. End-to-End Improvement of Real-time Use Cases

A. Benefit of Maintaining Full Utilization of Cores

The proposed NCPU architecture can maintain full utilization of the cores by smoothly switching the operating modes with zero-latency. Hence the end-to-end performance is improved by eliminating any idle time within the cores. Fig. 13 illustrates the end-to-end performance improvement by maintaining full utilization of cores for an image classification use case. During the experiment, the execution latency of the image BNN inference maintains the same. The fraction of the CPU workload, i.e. the CPU run cycles over the sum of CPU and BNN run cycles, is adjusted by changing the complexity of the image data pre-processing algorithms. For the workload with a high fraction of CPU operations, e.g. 70%, the NCPU architecture improves the overall end-to-end performance by 41.2% compared to a baseline heterogeneous architecture. For a well-balanced workload between CPU and BNN accelerator, e.g. CPU workload fraction of 40%, the NCPU still shows an improvement of 28.5%. Therefore, significant end-to-end performance improvement can be obtained by resolving the core under-utilization issue.

The end-to-end performance benefit has also been further evaluated with different batch sizes, e.g. the number of images, under the CPU workload fraction of 70%, as shown in Fig. 14. Large batch sizes help heterogeneous architecture to hide the data transfer latency and obtain higher end-to-end performance. Therefore, NCPU gains a little less end-to-end benefit with larger image batch size while still maintaining above 37% latency improvement with the batch size of 100.

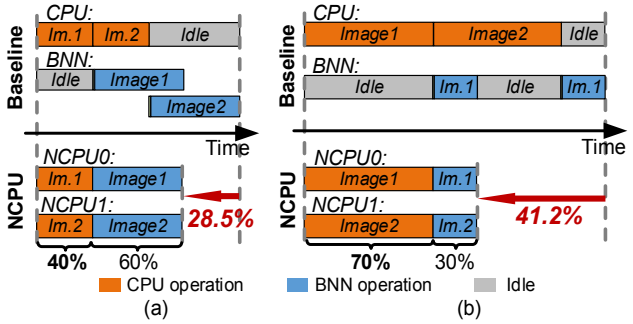


Fig. 13. The core utilizations during runtime under the CPU workload fraction of (a) 40% and (b) 70%.

Fig. 14. The end-to-end performance benefit with sweeping of image batch size under the CPU workload fraction of 70%.

B. Real-Time Use Cases

To evaluate the real end-to-end performance gain for embedded applications, two real-time use cases have been evaluated on the test chip. There is a lack of established benchmarks for evaluating real-time ML end-to-end performance in low power devices. Consequently, we constructed two real-life embedded use cases, including image classification and human motion detection.

Fig. 15 shows the workload breakdown of the CPU processing and BNN inference in the test cases. For the image classification use case, the CPU operation takes a batch of raw image pixel data (size: 224x224x3) and conducts several image processing functions, including image resizing, grayscale filtering, and data normalization [48-49]. These are common image processing algorithms required before the neural network classification [50]. After the raw image data has been processed, the NCPU core switches from CPU mode to the BNN mode to perform the image inference. The BNN model is trained using MNIST dataset achieving the classification accuracy of 94.8%. The CPUs operations take about 76% of the total runtime, which is similar to the study case from Intel's IoT SoC [12]. It is worth mentioning that the data pre-processing tasks are normally managed by separate image signal processor (ISP) in modern mobile SoCs [51-52]. However, in ultra-low power embedded SoC, ISP is not an indispensable IP and CPU processes the majority of the general-purpose computing tasks.

For the human motion detection test case, the BNN model is trained using the recorded accelerometer sensor data from Ninapro database [59], with 74% classification accuracy for simple motion activity detection. Six out of twelve channels of the accelerometer sensor signals were used in our study. Three time-domain features including mean and histogram for each channel were used for BNN classification [60]. The CPU operation is dominated by the feature extraction tasks and shows a workload fraction of 68%.

As our design target is for resource-constrained low power embedded applications, the use cases have been tested on chip with the operating frequency of 50MHz, which is in line with other commercial microcontrollers' speed [53-54]. For the human motion detection use case, only a single human gesture is detected and classified due to the slow human motion time scale in real-time applications.

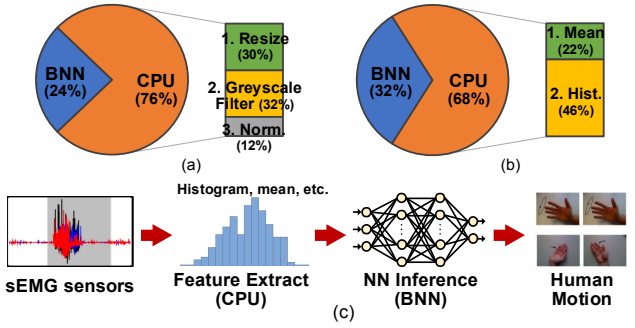


Fig. 15. Runtime CPU and BNN workload breakdown for the use cases of (a) image classification, (b) motion detection, and (c) the real-time task operation sequence for the motion detection case.

C. End-to-end Improvement Results

To evaluate the end-to-end performance improvement, the use cases were tested on both baseline heterogeneous architecture, i.e. CPU core with BNN accelerator, and our developed two-core NCPU SoC. Fig. 16 shows the measured power traces during the runtime of the image classification use case under the NCPU peak performance. For the baseline

heterogeneous architecture, the BNN core stayed idle at the beginning and waited for the CPU data pre-processing tasks to complete. The BNN accelerator was launched for the image inference only after received the processed data, and the CPU continued to process the pixel data for the second image. For the NCPU two-core configuration, both NCPU cores were reconfigured into CPU mode at the beginning to process two images simultaneously. After image processing completed, both NCPUs were switched for BNN inference. As the core utilization rate summarized in Table 4, comparing with the baseline configuration, i.e. CPU+BNN, NCPU SoC can maintain utilization more than 99% runtime. Therefore, for the same image classification task, the real-time end-to-end performance achieves a 43% speed up due to the reconfiguration of NCPU maintaining core full utilization.

Fig. 17 shows the end-to-end performance improvement for the two use cases. With the significant improvement of the core utilization, i.e. realizing both cores at almost 100% utilization, the end-to-end performance of two-core NCPU scheme shows 35% to 43% improvement comparing with the conventional heterogeneous architecture. Comparing the baseline heterogeneous architecture, the area of single NCPU is saved by 35%, while the overall end-to-end performance is only degraded by 13.8%. Therefore, a single NCPU core can achieve significant area saving benefit with small end-to-end performance loss. The obtained end-to-end performance improvement can be equivalently converted to energy saving, which is in high demand for low power embedded applications. To obtain the energy saving, the supply voltage was scaled down while maintaining the same total execution latency. The achieved performance improvement for the image use case is converted into up to 74% energy saving under lower supply voltage. Note that the power tracing was performed at 1V so that the core activities can be easier distinguished due to high power consumption. The same performance benefits can be observed at 0.5V.

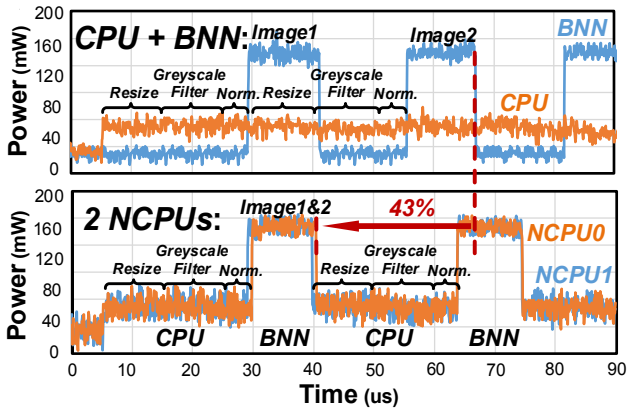


Fig. 16. Measured power traces for image classification use case.

TABLE IV. THE CORE UTILIZATION RATES AT THE BASELINE HETEROGENEOUS SOC OR NCPU SOC CONFIGURATIONS.

Mode/Utilization	NCPU 0	NCPU 1
Baseline	CPU / 80.2%	BNN / 39.4%
2 NCPUs	NCPU / 99.3%	NCPU / 99.3%

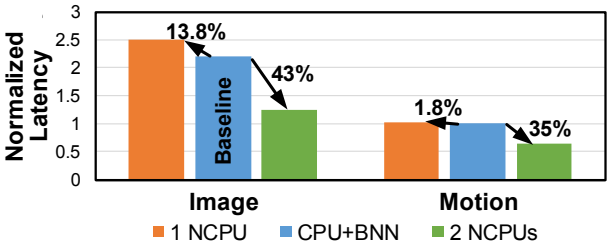


Fig. 17. Fig. 16 End-to-end performance improvement for two use cases.

VIII. DISCUSSION AND RELATED WORK

A. Baseline Accelerator Selection

In this work, the neuromorphic type BNN accelerator design [40, 44] is selected as the baseline architecture. The neuron cell design and their connections are analogous to the conventional CPU pipeline, which makes the reconfigurable NCPU design natural and low overhead. BNN architecture is chosen as it is a good compromise for low-power applications and matches well with the low-cost CPU. The selection of baseline BNN accelerator size needs to consider both the BNN model accuracy and the size compatibility with the CPU core. More neural network layers or neuron cells in each layer will increase both the model accuracy and area. Fig. 18 shows the area saving benefit and BNN inference accuracy with different neuron cells per layer. For different neural network sizes, i.e. neuron cell per layer varying from 50 to 400, the classification accuracy of the MNIST dataset can change from 88.6% to 97.2%. The area saving benefit of the NCPU compared with the conventional heterogeneous architecture reduces to 22.5% with the use of 400 neurons. The tradeoff between the BNN inference accuracy and the area saving benefit leads to our design choice of 100 neuron cells per layer with a moderate BNN accuracy of 94% for MNIST database. In our NCPU SoC, deeper BNN with more layers can be supported by rolling back the BNN operation or connecting two cores in series. Smaller BNNs are supported by configuring NCPU layers using the developed ISA.

One of the main targets for this work is to explore a new computing architecture to close the significant performance gap between neural network accelerator and conventional processor design for ultra-low power embedded devices. There is a significant architecture design space that can be further explored. Supporting multi-bit and complex DNN is definitely a future research direction. In fact, it may be easier to reconfigure a larger DNN accelerator into CPU pipelines due to the abundant hardware resources. But to better utilize the hardware resources of DNN, the reconfigured CPU pipeline will also be preferably scaled up into vector processing, SIMD, superscalar pipeline or multi-core. We hope our preliminary work here can inspire more development in this direction in the future.

It is worth mentioning that similar reconfigurable NCPU architecture also can be implemented on FPGAs. However, due to the large resource consumption of FPGA, it will entail orders of magnitude greater power overhead than the ASIC implementation in this work. For the resource-constrained edge devices, our NCPU design can provide lower cost and better power consumption than FPGA solutions.

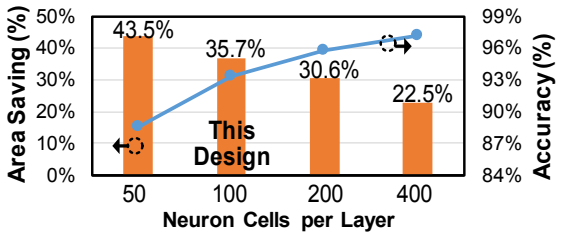


Fig. 18. Area saving with different accelerator size.

B. Prior Related Work

The motivation of this work is to explore a new architecture solution to reconcile the difference between the conventional CPU microprocessor and the popular neural network accelerator in cost and power constrained systems. Previously, in-memory computing is utilized to bring neural network computation inside the memory storage to reduce the data transfer cost [29-30]. A more recent in-memory computing scheme further supports flexible ALU instructions, e.g. addition, subtraction and the multiplications, in SRAM [31]. However, the support of highly programmable general-purpose computing is still challenging for those in-memory computing schemes.

To improve the heterogeneous design, several CPU-accelerator interfaces have been studied and adopted in heterogeneous SoC to couple the cores tightly and reduce the workload offloading cost [14-16]. For example, the RoCC interface is adopted for communicating data between the RISC-V CPU caches and the BNN accelerator. Compared with [14], our work proposes a different reconfigurable architecture solution to solve the offloading challenge and avoid the complicated heterogeneous interface designs. Our evaluations do not consider the area/power overheads of the RoCC interface which will add to the overheads of the CPU+BNN baseline.

There is also a research effort by Google trying to use the neural network to perform CPU operations. "Neural Turing Machine" is a representative work to use neural network to perform CPU operations, e.g. copy, sorting or complex query problems [32-33]. Similar memory augmented neural networks have been studied for cognitive operations [34-35]. A "neural arithmetic logic units (NALU)" architecture is proposed by Google in which a neural network is trained to perform some basic arithmetic operation, e.g. *ADD*, *SUB*, or *MUL* [36]. The above designs still remain at concept level with software implementation to date. As illustrated by our experiment in following, the cost of NALU is prohibitively high comparing with conventional digital design approach and hence is not suitable for a resource constrained SoC.

C. NALU Architecture Experiment

Previously, the concept of "Neural ALU" was proposed to train a neural network to learn and perform several basic arithmetic operations, e.g. *ADD* or *MUL* [36]. However, the prior study was only at conceptual level (software) and did not fully consider hardware cost. We evaluated the NALU concept from a hardware perspective and compare it with the conventional digital approach in term of power, area, etc. The NALU is a two layers fully-connected neural network, which can be trained using back propagation for 8-bit ALU operations, such as *ADD*, *SUB*, *AND*, *XOR*, with the mean squared error (MSE) as a cost function for back propagation. MSE error is scaled relative to a random initialized model, i.e. 100% is equivalent to random and 0% is perfect accuracy. As shown in Fig. 19(a), the NALU architecture performs well to learn *ADD*/*SUB* operation, while it suffers from significant error in Boolean operation *AND*/*XOR*. When realizing both *ADD* and *SUB* simultaneously, NALU output almost becomes random for these ALU operations, leading to training failure. Further experiments show that the output error can be gradually reduced with more hidden layers or neurons. However, given that the single hidden layer architecture already causes significant design overhead as will be shown in the following, adding more hidden layers or neurons will make hardware implementation infeasible. Therefore, results with only single hidden layer architecture will be shown here, which is same as [36].

The area and power cost of the implemented NALU are compared with the conventional digital implementation approach using TSMC 65nm process, as shown in Fig. 19(b). The post-layout result shows the NALU implementation for *ADD* cost about 17X area than a digital adder. The significant NALU area cost is due to the multiplication tasks for simple ALU operations such as *ADD* or *AND* in NALU architecture, which is very area and power costly. The neural network also requires large memory capacity to store all the model weights. All these large area costs translate to significant cost in power consumption. Based on NALU experiment results, we understand that a straightforward implementation of neural network for CPU operations is not realistic for energy efficient implementation. However, these observations lead to the design approach of our NCPU architecture.

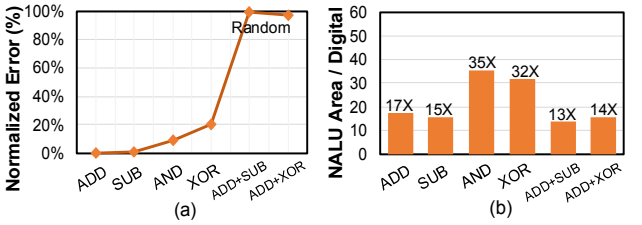


Fig. 19. (a) The normalized error when NALU learns different ALU operations, and (b) NALU area cost comparing with the conventional digital design approach.

IX. Conclusion

In this paper, we propose a reconfigurable Neural CPU, which is a novel architecture suitable for ultra-low power applications with significant cost reduction and performance improvement. The proposed NCPU core supports the flexible programmability of RISC-V CPU as well as BNN inference acceleration. A NCPU SoC test chip was fabricated in a 65nm CMOS process. Compared to a conventional heterogeneous design, a single NCPU core achieves 35% area reduction and 12% energy saving. With two NCPUs, the overall end-to-end performance achieves up to 43% performance improvement for two real-time use cases demonstrated on the test chip.

Acknowledgment

We wish to thank the anonymous reviewers for their helpful feedback. We thank Kendall Kuzminskas for her contribution on preparing some testing programs. This paper is supported in part by the National Science Foundation under grant number CCF-1908488.

REFERENCES

[1] D. Blaauw, D. Sylvester, P. Dutta, Y. Lee, I. Lee, S. Bang, Y. Kim, G. Kim, P. Pannuto, Y.-S. Kuo, D. Yoon, W. Jung, Z. Foo, Y.-P. Chen, S. Oh, S. Jeong, and M. Choi, "IoT Design Space Challenges: Circuits and Systems", *Symposium on VLSI Technology*, 2014.

[2] P. Whatmough, S. Lee, H. Lee, S. Rama, D. Brooks, and G. Wei, "A 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications", *International Solid-State Circuits Conference (ISSCC)*, pp. 242-243, Feb. 2017.

[3] M. Hempstead, N. Tripathi, P. Mauro, G. Wei, and D. Brooks, "An ultra low power system architecture for sensor network applications", *International Symposium on Computer Architecture (ISCA)*, pp. 208-219, 2005.

[4] S. Sridhara, M. DiRenzo, S. Lingam, S. Lee, R. Blazquez, J. Maxey, S. Ghanem, Y. Lee, R. Abdallah, P. Singh, and M. Goel, "Microwatt embedded processor platform for medical system-on-chip applications", *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 46, no. 4, pp. 721-730, Apr. 2011.

[5] Y. Shi, M. Choi, Z. Li, G. Kim, Z. Foo, H. Kim, D. Wentzloff, and D. Blaauw, "A 10mm^3 syringe-implantable near-field radio system on glass substrate", *International Solid-State Circuits Conference (ISSCC)*, pp. 448-449, Feb. 2016.

[6] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen and O. Temam, "DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning", *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 269-284, Mar. 2014.

[7] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks", *International Symposium on Computer Architecture (ISCA)*, pp. 368-379, Jun. 2016.

[8] N. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. Yoon, "In-datacenter performance analysis of a Tensor Processing Unit", *International Symposium on Computer Architecture (ISCA)*, Jun. 2017.

[9] G. Desoli, N. Chawla, T. Boesch, S. Singh, E. Guidetti, F. Ambroggi, T. Majo, P. Zambotti, M. Ayodhyawasi, H. Singh, and N. Aggarwal, "A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems", *International Solid-State Circuits Conference (ISSCC)*, pp. 238-239, Feb. 2017.

[10] Online resource, Intel, "Neural compute engine: hardware based acceleration for deep neural networks", https://www.movidius.com/MyriadX.

[11] J. Song, Y. Cho, J. Park, J. Jang, S. Lee, J. Song, J. Lee, and I. Kang, "An 11.5TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC", *International Solid-State Circuits Conference (ISSCC)*, pp. 130-131, Feb. 2019.

[12] T. Karnik, D. Kurian, P. Aseron, R. Dorrance, E. Alpman, A. Nicoara, R. Popov, L. Azarenkov, M. Moiseev, L. Zhao, S. Ghosh, R. Misoczki, A. Gupta, A. M, S. Muthukumar, S. Bhandari, Y. Satish, K. Jain, R. Flory, C. Kanthapanit, E. Quijano, B. Jackson, H. Luo, S. Kim, V. Vaidya, A. Elsherbini, R. Liu, F. Sheikh, O. Tickoo, I. Klotchkov, M. Sastry, S. Sun, M. Bhartiya, A. Srinivasan, Y. Hoskote, H. Wang, and V. De, "A cm-scale self-powered intelligent and secure IoT edge mote featuring an ultra-low-power SoC in 14nm tri-gate CMOS", *International Solid-State Circuits Conference (ISSCC)*, pp. 46-47, Feb. 2018.

[13] V. Honkote, D. Kurian, S. Muthukumar, D. Ghosh, S. Yada, K. Jain, B. Jackson, I. Klotchkov, M. R. Nimmagadda, S. Dattawadkar, P. Deshmukh, A. Gupta, J. Timbadiya, R. Pali, K. Narayanan, S. Soni, S. Chhabra, P. Dhama, N. Sreenivasulu, J. Kollikunnel, S. Kadavakollu, V. D. Sivaraj, P. Aseron, L. Azarenkov, N. Robinson, A. Radhakrishnan, M. Moiseev, G. Nandakumar, A. Madhukumar, R. Popov, K. P. Sahu, R. Peguvandla, A. Ruiz, M. Bhartiya, A. Srinivasan, and V. De, "A distributed autonomous and collaborative multi-robot system featuring a low-power robot SoC in 22nm CMOS for integrated battery-powered minibots", *International Solid-State Circuits Conference (ISSCC)*, pp. 48-49, Feb. 2019.

[14] S. Davidson, S. Xie, C. Torng, K. Al-Hawai, A. Rovinski, T. Ajayi, L. Vega, C. Zhao, R. Zhao, S. Dai, A. Amarnath, B. Veluri, P. Gao, A. Rao, G. Liu, R. Gupta, Z. Zhang, R. Dreslinski, C. Batten, and M. Taylor, "The Celerity open-source 511-core RISC-V tiered accelerator fabric", *IEEE Micro*, vol. 38, issue. 2, pp. 30-41, Apr. 2018.

[15] P. Whatmough, S. Lee, M. Donato, H. Hsueh, S. Xi, U. Gupta, L. Pentecost, G. Ko, D. Brooks, G. Wei, "A 16nm 25mm^2 SoC with a 54.5x flexibility-efficiency range from dual-core Arm Cortex-A53 to eFPGA and cache-coherent accelerators", *IEEE Symposium on VLSI Circuits (VLSI)*, pp. 34-35, Jun. 2019.

[16] J. Alsop, M. Sinclair, and S. Adve, "Spandex: A Flexible Interface for Efficient Heterogeneous Coherence", *International Symposium on Computer Architecture (ISCA)*, pp. 261-274, Jun. 2018.

[17] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. Horowitz, and W. Dally, "EIE: efficient inference engine on compressed deep neural network", *International Symposium on Computer Architecture (ISCA)*, pp. 243-254, Jun. 2016.

[18] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. Keckler, and W. Dally, "SCNN: an accelerator for compressed-sparse convolutional neural networks", *International Symposium on Computer Architecture (ISCA)*, pp. 27-40, Jun. 2017.

[19] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI", *International Solid-State Circuits Conference (ISSCC)*, pp. 246-247, Feb. 2017.

[20] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS", *International Solid-State Circuits Conference (ISSCC)*, pp. 216-218, Feb. 2018.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv: 1603.04467*, 2016.

[22] D. Narayanan, K. Santhanam, A. Phanishayee, and M. Zaharia, "Accelerating deep learning workloads through efficient multi-model execution", *NeurIPS Workshop on Systems for Machine Learning*, Dec. 2018.

[23] S. Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G. Wei, D. Brooks, "SMAUG: end-to-end full-stack simulation infrastructure for deep learning workloads," *arXiv preprint arXiv*: 1912.04481, 2019.

[24] C. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, T. Leyvand, H. Lu, Y. Lu, L. Qiao, B. Reagen, J. Spisak, F. Sun, A. Tulloch, P. Vajda, X. Wang, Y. Wang, B. Wasti, Y. Wu, R. Xian, S. Yoo, and P. Zhang, "Machine learning at Facebook: understanding inference at the edge", *High Performance Computer Architecture (HPCA)*, pp. 331-344, Feb. 2019.

[25] Online resource, Nvidia, "Embedded systems for next-generation autonomous machines", https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/.

[26] Online resource, Google, "Edge TPU", https://cloud.google.com/edge-tpu/.

[27] M. Hill, M. Marty, "Amdahl's law in the multicore era", *Computer*, vol. 41, no. 7, pp. 33-38, Jul. 2008.

[28] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling", *International Symposium on Computer Architecture (ISCA)*, pp. 365-376, Jun. 2011.

[29] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array", *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 4, pp. 915-924, Apr. 2017.

[30] Z. Jiang, S. Yin, M. Seok, and J. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks", *IEEE Symposium on VLSI Technology (VLSI)*, pp. 173-174, Jun. 2018.
[31] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural Cache: bit-serial in-cache acceleration of deep neural networks", *International Symposium on Computer Architecture (ISCA)*, pp. 383-396, Jun. 2018.
[32] A. Graves, G. Wayne, I. Danihelka, "Neural turing machines," *arXiv preprint arXiv: 1410.5401*, 2014.
[33] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory", *Nature*, vol. 538, pp. 471-476, Oct. 2016.
[34] H. Jang, J. Kim, J. Jo, J. Lee, and J. Kim, "MnnFast: a fast and scalable system architecture for memory-augmented neural networks", *International Symposium on Computer Architecture (ISCA)*, pp. 250-263, Jun. 2019.
[35] J. Stevens, A. Ranjan, D. Das, B. Kaul, and A. Raghunathan, "Manna: an accelerator for memory-augmented neural networks", *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 794-806, Oct. 2019.
[36] A. Trask, F. Hill, S. Reed, J. Rae, C. Dyer, and P. Blunsom, "Neural arithmetic logic units", *Advances in Neural Information Processing Systems (NIPS)*, pp. 8035-8044, Dec. 2018.
[37] C. Chen, H. Peng, X. Liu, H. Ding, and C. Shi, "Exploring the programmability for deep learning processors: from architecture to tensorization", *Design Automation Conference (DAC)*, Jun. 2018.
[38] M. Putic, S. Venkataramani, S. Eldridge, A. Buyuktosunoglu, P. Bose, and M. Stan, "Dyhard-DNN: even more DNN acceleration with dynamic hardware reconfiguration", *Design Automation Conference (DAC)*, Jun. 2018.
[39] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, "Binarized neural networks", *Advances in Neural Information Processing Systems (NIPS)*, pp. 4107-4115, Dec. 2016.
[40] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, T. Kuroda, and M. Motomura, "BRein memory: a single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W", *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 53, no. 4, pp. 983-994, Apr. 2018.
[41] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS", *International Solid-State Circuits Conference (ISSCC)*, pp. 222-223, Feb. 2018.
[42] M. Yang, C. Yeh, Y. Zhou, J. Cerqueira, A. Lazar, and M. Seok, "A 1μW voice activity detector using analog feature extraction and digital deep neural network", *International Solid-State Circuits Conference (ISSCC)*, pp. 346-347, Feb. 2018.
[43] Online resource, BittWare, "FPGA acceleration of binary weighted neural network inference", https://www.bittware.com/resources/bwnn/
[44] J. Park, J. Lee, and D. Jeon, "A 65nm 236.5nJ/classification neuromorphic processor with 7.5% energy overhead on-chip learning using direct spike-only feedback", *International Solid-State Circuits Conference (ISSCC)*, pp. 140-141, Feb. 2019.
[45] A. Waterman, and K. Asanovic, "The RISC-V instruction set manual, volume I: user-level ISA, document version 2.2", *RISC-V Foundation*, May 2017.
[46] B. Keller, M. Cochet, B. Zimmer, J. Kwak, A. Puggelli, Y. Lee, M. Blagojevi´c, S. Bailey, P. Chiu, P. Dabbelt, C. Schmidt, E. Alon, K. Asanovi´c, and B. Nikoli´c, "A RISC-V processor SoC with integrated power management at submicrosecond timescales in 28 nm FD-SOI", *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 7, pp. 1863-1875, Jul. 2017.
[47] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "MiBench: a free, commercially representative embedded benchmark suite", *IEEE International Workshop on Workload Characterization*, pp. 3-14, 2001.
[48] K. Mishiba, and T. Yoshitome, "Image resizing with SIFT feature preservation", *IEEE International Conference on Image Processing (ICIP)*, pp. 991-995, 2013.
[49] N. Mitianoudis, and N. Papamarkos, "Multi-spectral document image binarization using image fusion and background subtraction techniques", *IEEE International Conference on Image Processing (ICIP)*, pp. 5172-5176, 2014.
[50] D. Demirović. E. Skejić, and A. Šerifović–Trbalić, "Performance of some image processing algorithms in Tensorflow", *International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 1-4, 2018.
[51] N. Nachiappan, H. Zhang, J. Ryoo, N. Soundararajan, A. Sivasubramaniam, M. Kandemir, R. Iyer, and C. Das, "VIP: virtualizing IP chains on handheld platforms", *International Symposium on Computer Architecture (ISCA)*, pp. 655–667, Jun. 2015.
[52] Y. Zhu, A. Samajdar, M. Mattina, and P. Whatmough, "Euphrates: algorithm-SoC co-design for low-power mobile continuous vision", *International Symposium on Computer Architecture (ISCA)*, pp. 547–560, Jun. 2018.
[53] Online resource, Microchip, "PIC18F13K22", https://www.microchip.com/wwwproducts/en/PIC18F13K22
[54] Online resource, Texas Instruments, "MSP432P401R", https://www.ti.com/product/MSP432P401R
[55] Online resource, Microchip, "ATSAMA5D44", https://www.microchip.com/wwwproducts/en/ATsama5d44
[56] Online resource, SiFive, "E31", https://www.sifive.com/cores/e31
[57] Online resource, Wikipedia, "Apple A13", https://en.wikipedia.org/wiki/Apple_A13
[58] Z. Azad, M. S. Louis, L. Delshadtehrani, A. Ducimo, S. Gupta, P. Warden, V. J. Reddi, and A. Joshi, "An end-to-end RISC-V solution for ML on the edge using in-pipeline support", *Boston area Architecture (BARC) Workshop*, 2020.
[59] M. Atzori, A. Gijsberts, S. Heynen, A. Mittaz Hager, O. Deriaz, P. Van der Smagt, C. Castellini, B. Caputo, and H. Müller, "Building the Ninapro database: A resource for the biorobotics community", *IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 1258-1265, Jun. 2012.
[60] K. Otseidu, T. Jia, J. Bryne, L. Hargrove, and J. Gu, "Design and optimization of edge computing distributed neural processor for biomedical rehabilitation with sensor fusion", *International Conference on Computer-Aided Design (ICCAD)*, Nov. 2018.