A 16-nm Always-On DNN Processor With Adaptive Clocking and Multi-Cycle Banked SRAMs

Sae Kyu Lee, *Member, IEEE*, Paul N. Whatmough, *Member, IEEE*, David Brooks, *Fellow, IEEE*, and Gu-Yeon Wei, *Senior Member, IEEE*

***Abstract*— Always-on subsystems in mobile/Internet of Things (IoT) SoCs process a variety of real-time sensor data deep neural network (DNN) classification workloads in a heavily constrained energy budget. This can be achieved with robust, low-voltage circuits, and specialized hardware accelerators. We present a 16-nm always-on DNN processor, which consists primarily of a microcontroller and a DNN accelerator with on-chip SRAM for the model weights. The design operates robustly from 0.4 to 1-V, with calibration-free automatic voltage/frequency tuning provided by tracking small non-zero razor timing error rates. A novel timing error-driven synchronization-free adaptive clocking scheme significantly reduces the adaptation latency to provide resilience to fast on-chip supply noise and reduce margins. To accommodate the tight energy constraints of always-on IoT workloads, we implement a multi-cycle SRAM read scheme that allows the memory voltage to scale at iso-throughput, improving energy efficiency across the entire operating range. The wide operating range allows for high performance at 1.36 GHz, low-power consumption downs to 750 μW, and state-of-the-art raw efficiency at 16-bit precision of 750 GOPS/W dense or 1.81 TOPS/W sparse.**

***Index Terms*— Adaptive clocking, deep neural networks (DNNs), hardware accelerators, Internet of Things (IoT), machine learning (ML), razor, system-on-chip (SoC).**

I. Introduction

ALWAYS-ON intelligence represents a sea-change in the utility of mobile and Internet of Things (IoT) devices, both in consumer and industrial applications. Two big drivers for always-on intelligence are user interfaces (UIs) and sensor data classification. Emerging UIs on small form-factor devices are focused on speech-driven interfaces and require tasks such as audio keyword spotting, gaze detection, speaker identification, and face detection/matching. Sensor data analysis, on the other hand, encompasses a huge variety of potential applications, from industrial environmental monitoring to health monitoring applications such as human activity recognition. In all these cases, there is a common requirement for ultra-low-energy operation and robust autonomous classifier hardware to run these tasks continuously.

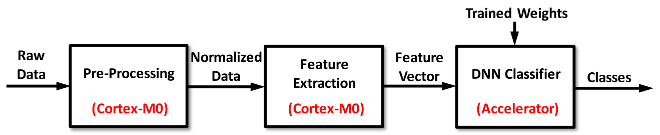


Fig. 1. Overview of a typical always-on sensor classification workload, showing the workload split between the microcontroller and accelerator hardware.

A typical sensor data classification workload (Fig. 1) includes stages for preprocessing, feature extraction, and classification using a machine learning (ML) model, such as a deep neural network (DNN). To handle this workload pipeline within an energy budget that allows for always-on operation is a severe challenge and requires different hardwares for different stages. Preprocessing and feature extraction (as well as general control tasks) are well handled by a microcontroller that offers software programmability. DNN inference demands a significantly larger compute and memory footprint, which is best served by a specialized hardware accelerator.

Always-on operation typically requires hardware that is able to run continuously and autonomously. As part of a larger SoC, the always-on subsystem operates while the rest of the SoC is in a sleep state, and ideally power-gated to remove even leakage power. Therefore, the always-on subsystem must be self-contained and is not able to assume the presence of a larger CPU and memory system. Hence, as well as efficient hardware acceleration of the algorithm computation itself, the hardware must also be self-contained and consider memory and input/output (IO), and provide programmability for simple control tasks. Finally, for energy efficiency reasons, it is essential to avoid the use of off-chip SRAM or DRAM, which incurs at least an order-of-magnitude increase in energy per access compared to on-chip memory [1].

Aggressive supply voltage scaling is essential to reach energy efficiency levels that are realistic for always-on

Manuscript received November 2, 2018; revised February 18, 2019; accepted April 9, 2019. Date of publication May 15, 2019; date of current version June 26, 2019. This paper was approved by Guest Editor Stefan Rusu. This work was supported in part by the U.S. Government through DARPA CRAFT and DARPA PERFECT Programs, in part by NSF under Award 1551044 and Award 1718160, and in part by Intel Corporation. *(Corresponding author: Sae Kyu Lee.)*

S. K. Lee was with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA. He is now with the IBM T. J. Watson Research Center, New York, NY 10598 USA (e-mail: saekyu.lee@ibm.com).

P. N. Whatmough is with Arm Research, Boston, MA 02451 USA, and also with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: paul.whatmough@arm.com).

D. Brooks and G.-Y. Wei are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: dbrooks@eecs.harvard.edu; gywei@g.harvard.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2019.2913098

applications. However, from a circuits perspective, low-voltage operation of digital circuits brings a number of challenges. First, dense SRAM memories operate with much lower noise margins compared to canonical static CMOS logic, due to the use of ratioed logic and dynamic logic. Hence, the voltage scaling of commercial 6 T SRAM arrays is significantly more limited than that of logic cells and flip-flops. Second, it is necessary to be able to set the supply voltage automatically to suit the clock frequency demanded by the application throughput constraint. This is ideally done adaptively, to minimize excess margins otherwise required to account for process, voltage, temperature, and aging (PVTA) [2]–[4]. Finally, it is also essential that circuit operation at low voltage is robust to supply voltage noise induced either directly or through a shared ground, due to the activity of local logic or even activity of another component on the SoC, which can cause supply resonance [5].

A large body of work has accumulated over the past couple of years on the topic of hardware accelerators for DNNs. Of these, a few published chips have focused on the features required for low-energy IoT applications. Bang *et al.* [6] introduced an on-chip nonuniform memory architecture. Whatmough *et al.* [7] employed on-chip memory only and demonstrated circuit-level error tolerance in DNNs. Bankman *et al.* [8] demonstrated a mixed-signal approach to convolutional neural networks, with all memory on-chip. Zhang *et al.* [9] described an in-memory ensemble weak classifier.

This work [10] presents an autonomous wide dynamic range DNN subsystem for always-on IoT applications. Addressing the challenges of low-voltage circuit design in this application, we describe the following.

1) *DNN Processor:* Hardware architecture for an always-on subsystem incorporating a microcontroller and a DNN accelerator with all memory on-chip. Timing-error detection and mitigation circuits (also known as “razor” [4]) are used to automatically tune supply voltage, dynamically eliminating margins without requiring calibration.
2) *Multi-Cycle SRAMs:* The memory bandwidth demands of DNNs are a bottleneck for performance and energy efficiency. In addition to this, SRAM voltage scaling performance is inferior to that of logic. To circumvent both of these issues, we demonstrate an arrangement that allows us to increase the effective cycle time for SRAM reads, without reducing the bandwidth, while using only standard commercial 6 T SRAM arrays.
3) *Adaptive Clocking:* A fast implementation of discrete adaptive clocking driven from timing error detection circuits. Inside an automatic timing error-driven supply voltage scaling loop, fast-adaptive clocking provides a safety-net to allow for operation with minimal margins.

The remainder of this paper is organized as follows. Section II gives an overview of the always-on DNN processor architecture, multi-cycle SRAM read, and adaptive clocking schemes. Section III describes the test chip implementation and Section IV presents the measurement results. Finally, Section V concludes this paper.

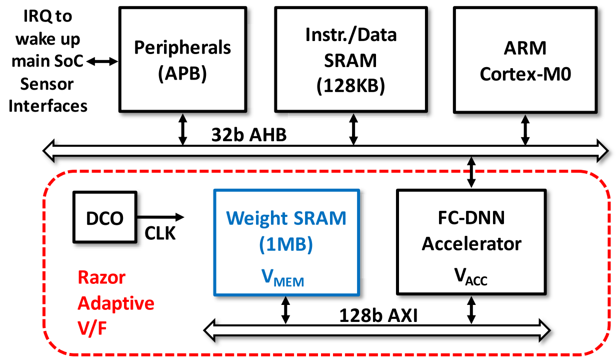


Fig. 2. Block diagram of always-on DNN processor for Mobile/IoT applications. The FC-DNN accelerator includes smaller SRAMs for activations and sparse node list (shown in Fig. 3).

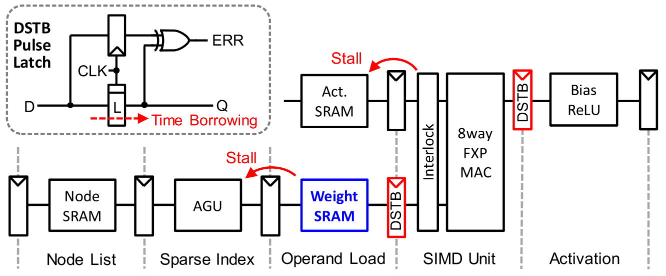


Fig. 3. Block diagram of sparse FC-DNN pipeline with DSTB razor latches (inset) and dynamic sequencing for sparse activations. Both the activation load and the weight address generation are stallable from an interlock preceding the MAC unit.

II. Always-On DNN Processor

A. Architecture Overview

Fig. 2 shows the block diagram of the always-on DNN processor. The major components include an Arm Cortex-M0 microcontroller with 128 KB of instruction and data SRAM, a 32-bit AHB interconnect, low-bandwidth peripherals on an APB interconnect, and a DNN accelerator with a 1-MB weight SRAM on a high-bandwidth 128-bit AXI interconnect. The microcontroller provides programmable system management tasks, as well as pre-processing and feature extraction as required by the application (Fig. 1). The fully connected DNN (FC-DNN) accelerator and weight SRAM are contained in separate voltage islands, clocked by a dedicated digitally controlled oscillator (DCO). The 1-MB banked weight SRAM stores the model weights, allowing the accelerator to operate autonomously without incurring overheads of costly off-chip memory access.

Fig. 3 shows the FC-DNN accelerator pipeline that includes the accelerator and the 1-MB weight SRAM. The FC-DNN accelerator is a second-generation design derived from [7], with a five-stage pipeline, eight-way fixed-point single instruction, multiple data (SIMD) datapath, and optimizations for 16b/8b operands. The accelerator pipeline also includes smaller SRAMs to store activations and sparse node list. The FC-DNN architecture exploits activation data sparsity to significantly reduce both the number of multiply-accumulates (MACs) and the number of memory accesses [1]. During the activation stage, output activations are compared to a programmable threshold to eliminate small neurons from downstream processing, achieved by storing a list of active neurons in the node SRAM.

A razor [4] timing error detection-based adaptive voltage and frequency scheme is used to set the operating condition for the accelerator and weight SRAM. Timing error detection is performed using the double-sampling method with time-borrowing (DSTB) in the datapath [2], [11]. In the accelerator pipeline (Fig. 3), the weight SRAM load and SIMD datapath unit have DSTB latches at critical timing end-points. A DSTB register [11], shown in Fig. 3, is composed of a flip-flop and latch, with the D input feeding into both in parallel. The latch Q output is used in the datapath. An error signal (ERR) is derived from the XOR of the flip-flop and latch outputs. A pulsed clock drives the CLK signal, such that the latch behaves as a pulse latch, while the flip-flop behaves as a normal hard-edge register. The DSTB hence provides TB in the datapath due to the transparency of the latch in the clock pulse phase. Meanwhile, the ERR signal is asserted if the output of the flip-flop and pulse latch differs, which will be true if a transition on D occurs after the clock edge, but before the clock pulse falls. The DNN processor tunes the operating condition of the accelerator and weight SRAM by monitoring the ERR signal. Similar to some previous razor accelerators, timing error detection is used without explicit replay-based correction [1], [12], [13], that is, timing errors are intentionally left uncorrected in the datapath with some quality impact at the algorithm level.

In this paper, we focus on the low-voltage circuit performance of this design in 16-nm FinFET technology, by describing a banked weight SRAM, and a fast-adaptive clocking mechanism. These are described in detail in Sections II-B and II-C.

B. Multi-Cycle Banked SRAM Read

The forward calculation of FC-DNN layers is memory bandwidth limited, in contrast to CNN layers, which are typically compute bound. Exploiting abundant data sparsity significantly reduces memory bandwidth [7], but access to the model parameters (weights) is still ultimately the limitation on performance. In addition to this, SRAM is generally also the bottleneck limitation on the minimum operating voltage (V_{MIN}). To address both of these issues, we describe a banked multi-cycle scheme that relaxes the clock-frequency requirement of the SRAM without degrading the average throughput. We do this using an optimized accelerator micro-architecture with unmodified commercial 6-T SRAMs, which is in contrast to previous work that uses custom 8-T arrays optimized specifically for low-voltage operation [6].

The multi-cycle banked SRAM micro-architecture is shown in Fig. 4, along with the clocking waveforms. The 1-MB weight SRAM is split across four autonomous banks, using the LSBs from the address to determine the bank. Therefore, read requests to sequentially addressed words are serviced from consecutive banks, in a round-robin fashion. A decode and backpressure unit generates an enable for each bank based on

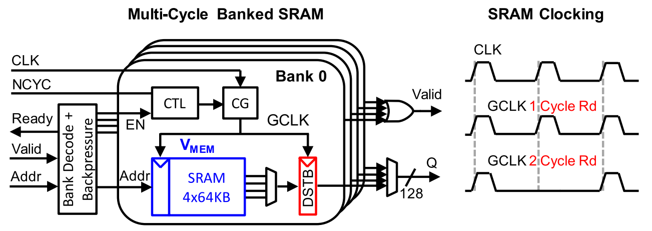


Fig. 4. Multi-cycle banked SRAM microarchitecture and associated waveforms for clocking.

the address received on each transaction on the 128-bit AXI bus. Given that each bank now has an activity factor of one access in every four address requests, the effective throughput of any given bank is reduced to one-fourth, while the overall throughput is unchanged at one 128-bit data access per AXI transaction per cycle. Simply stated, this means that the read access time of the SRAM in each bank has been relaxed by a factor of 4×.

We exploit this read access time reduction by using a counter and clock gating (CG) logic to enable a configurable one or two cycle read latency for the banked SRAMs. This approach allows us to simply use a single clock for the accelerator and the SRAM, avoiding the need to generate and calibrate another clock source, which is an important consideration for realistic product scenarios. Although each bank may now take multiple clock cycles to complete each read, there is no change in throughput for access to sequential addresses; the latency increases by one cycle which has a negligible impact on overall latency per inference. However, for non-sequential accesses that decode to the same bank in consecutive cycles, there is a bank collision, where the SRAM will be completing the second cycle of the first read when it receives a new read request.

The FC-DNN accelerator allows for both conventional dense processing of FC layers, as well as a sparse mode, where only activations larger than a programmable threshold are stored and processed. The sparse mode will often result in non-sequential addresses, which introduces occasional random bank contention and pipeline bubbles. Fig. 5 shows simplified timing diagrams for both dense accesses (which generate sequential addresses) and sparse (which introduces occasional contention resulting in stalls). Contention in sparse mode is handled by stalling the bus for a cycle, which is implemented by the backpressure unit. The accelerator pipeline micro-architecture is redesigned to allow stalling on weight memory bank collisions, as shown in Fig. 3, where the interlock logic before the MAC unit handles pipeline backpressure. Fig. 6 shows an analysis of the impact of stall cycles against network sparsity by sweeping the activation threshold during sparse operation when processing an MNIST model. In sparse mode at iso-accuracy, the stall cycle penalty is very low at less than 1%, which is a negligible impact on throughput. This is due to the high correlation in the input images, which mainly consist of large runs of white background. At the extreme of 90% sparsity, the SRAM reads incur <2.6% additional stall cycles, at which point the classification accuracy is degraded due to overly aggressive pruning of the activation nodes.

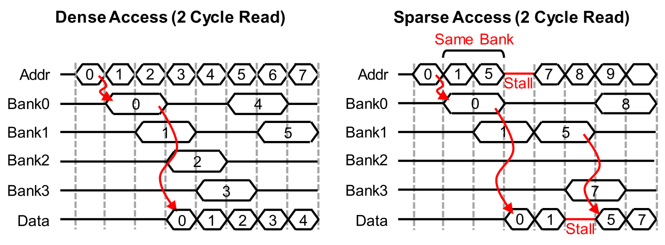


Fig. 5. Bank timing for two-cycle read operation during dense access (left) and sparse access (right). Reads in dense mode are sequentially striped across the banks and therefore multi-cycle read does not result in contention. Sparse mode causes occasional bank collisions that result in stall cycles.

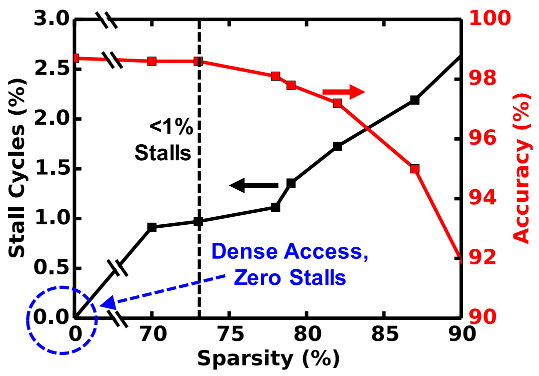


Fig. 6. Analysis of the impact of data sparsity on the percentage of stall cycles and MNIST classification accuracy. Dense access results in zero stalls, but sparse operation introduces occasional stall cycles. Sparse operation at iso-accuracy results in a stall cycle penalty of <1%, which is negligible.

C. Low-Latency Timing-Driven Adaptive Clocking

DNN algorithms are robust to small amounts of noise in the computation, which have minimal impact on classification accuracy. This has previously been exploited to dynamically remove PVTA margins and operate at a small non-zero timing error rate [7]. However, operating in this regime exposes the circuits to fast voltage droops. Although worst case *di/dt* voltage noises are infrequent in occurrence [14], [15], they can introduce large bursts of timing violations that may not be tolerable, and also risk corrupting the control plane logic, thus requiring additional voltage or frequency guardbands. Prior works have demonstrated adaptive clocking techniques to mitigate the impact of fast voltage droops [3], [5], [16]–[21]. In this paper, we propose a fast-adaptive clocking scheme specifically designed for error-tolerant accelerators. We extend the *in situ* razor timing error detection scheme to also trigger the fast-adaptive clocking mechanism.

Previous work has shown that benefit from adaptive clocking is fundamentally limited by voltage droop detection latency [22]. This is due to the fixed latency through the clock generation oscillator and the clock distribution and is typically limited by synchronization latency. Previously proposed voltage-triggered schemes often require complex calibration [3], [17], [18]. However, the use of *in situ* razor circuits to trigger adaptive clocking is ideal since they can be distributed to capture fast local variation and do not require calibration or significant margin. Fig. 7 shows an illustration

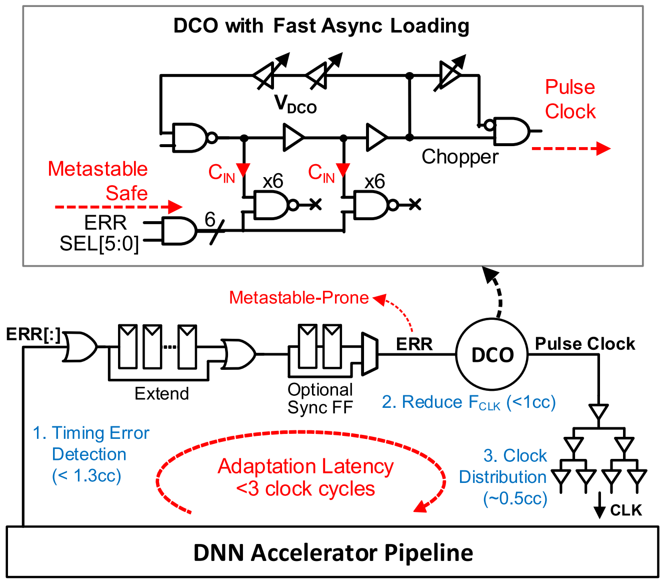

DCO with Fast Async Loading
V_{DCO}
Pulse Clock
Metastable Safe
C_{IN}
x6
Chopper
ERR
SEL[5:0]
6
Metastable-Prone
ERR[:]
Extend
Optional Sync FF
ERR
DCO
Pulse Clock
2. Reduce F_{CLK} (<1cc)
1. Timing Error Detection (< 1.3cc)
Adaptation Latency <3 clock cycles
3. Clock Distribution (~0.5cc)
CLK
DNN Accelerator Pipeline

Fig. 7. Razor timing error-driven fast-adaptive clocking. Double-sampling with TB (DSTB) pulse latches provides timing violation information, which is combined and used to initiate a frequency reduction at the DCO, using a metastability-safe fast loading stage, which removes sync flop latency.

of the razor-driven adaptive clocking scheme. Unfortunately, using razor to trigger discrete[1] adaptive clocking has a similar limitation to other approaches, due to the fact that the output trigger signal (ERR) is asynchronous to the clock (F_{CLK}) and prone to metastability. The ERR signal must be synchronized before being used to adapt the clock, to avoid the possibility of glitches and slow edges in the DCO output. A typical design solution is to use a flip-flop synchronizer to minimize the chance of metastability down to a tolerable risk. However, the synchronizer increases the adaptation latency, which is a critical requirement for adaptive clocking.

Fig. 7 shows a simplified schematic of the proposed scheme, which allows the requirement for the synchronizer to be relaxed without sacrificing metastability performance. DSTB pulse latches at timing critical end points in the accelerator pipeline and the weight SRAM detect timing errors due to fast voltage droops (Fig. 3). Timing error outputs from all DSTB latches in the system are OR reduced to a single ERR control signal, which is then used to drive the metastable-safe control input on the DCO to trigger a reduction in the clock frequency (F_{CLK}). The initial timing errors that trigger adaptation are masked with TB in the DSTB pulse latches. The duration of the F_{CLK} reduction is programmable via the Extend block, to allow it to last until the voltage droop event subsides. A conventional synthesized clock tree then distributes the clock output from the DCO to the flops in the design [23]. The metastable-safe frequency adaptation mechanism in the DCO allows the use of asynchronous ERR signal without the need for a flip-flop synchronizer stage, reducing the adaptation latency. As a result, the worst case overall droop response latency of this scheme, from timing error detection

[1]A discrete-step adaptive clocking scheme triggers a fixed change in the clock period, as opposed to a continuously adaptive clock period.

TABLE I
TEST CHIP SUMMARY

Process Technology	TSMC 16nm FinFET 1P9M
Die Size	2.5mm x 2.5mm
Package	165 pin BGA
Model Type	Fully-Connected DNN
On-chip Model Size	1MB
DSTB latches / Total FF	896 / 8460
V_{DD}	0.8V (Nominal) 0.4 – 1.0V (operational)
F_{MAX}	1.36GHz @1.0V (Max Freq) 1GHz @0.8V (Nominal) 100MHz @0.44V (Min Energy) 57MHz @0.4V (Min Voltage)
Power (16-bit)	135mW @1.0V 59.5mW @0.8V 2.42mW @0.44V 1.1mW @0.4V
Leakage	1.7mW (@0.8V)
Peak Efficiency (16-bit)	750 GOPS/W
MNIST Energy/Accuracy	151nJ / 98.51%
Resilience Features	Razor + Adaptive Clocking

to clock distribution as shown in Fig. 7, is less than 2.8 clock cycles (cc).

The simplified block diagram of the DCO shown in Fig. 7 illustrates the metastability-safe frequency adaptation mechanism. The DCO is an all-digital, single-ended design with muxed delay stages for conventional coarse-grained frequency tuning, and achieves fast metastability-safe frequency reduction via binary varactor loads before the ring oscillator output. The varactors use the two inputs of NAND2 gates: the input closest to the output in the NAND2 stack directly loads the delay path (C_{IN} in Fig. 7), while the other input modulates the effective capacitance of C_{IN}. Hence, any asynchronous events that result in a metastable signal on the NAND2 ERR input cannot cause glitches on F_{CLK}, and hence a synchronizer is not required. Multiple NAND2 stages, close to the DCO output, minimize charge injection and adaptation latency. The magnitude of the F_{CLK} reduction is programmable via the varactor select input in the DCO.

III. TEST CHIP IMPLEMENTATION

The DNN processor (Fig. 2) was implemented in a Taiwan Semiconductor Manufacturing Company (TSMC) 16-nm FinFET technology, and flip-chip bonded to a 165-pin ball grid array (BGA) package. Table I gives a summary of the test chip and Fig. 8 shows the die photograph. Physical implementation was carefully constrained to achieve a wide dynamic range (24×) in F_{MAX}, from 1.36 GHz at 1.0-V down to 57 MHz at near-threshold, 0.4-V. This was achieved with a commercial standard cell library and 6-T SRAM compiler. In all cases, razor error-rate counters are used to automatically find the optimal V_{DD}/F_{CLK} settings at the first error point, without calibration [4], [7]. The DSTB pulse latches placed on critical timing end points (Fig. 3) are implemented using standard cells only, with 2.03× area overhead of a regular flip-flop.

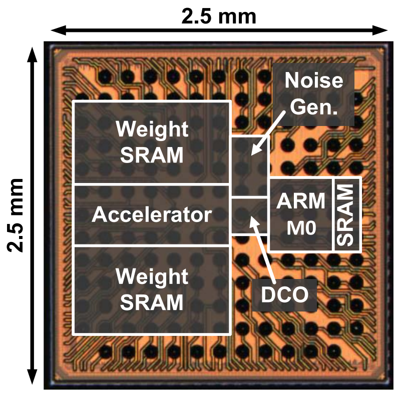


Fig. 8. Annotated die photograph of flip-chip packaged 16-nm test chip.

The area overhead of all DSTB latches accounts for <2.0% of the overall standard cell area in the design. All other paths are constrained with 30% extra timing margin to avoid functional failures.

Validating and characterizing implementations of adaptive clocking schemes are particularly challenging. We used flip-chip packaging and carefully designed Printed Circuit Board (PCB) supply coupling, in order to achieve a predictable supply impedance. A number of circuit instrumentation blocks were incorporated in the design to allow generation and measurement of supply voltage noise. We use an on-chip noise generator to emulate the supply noise generated on a common on-chip ground plane by large blocks in a full SoC. Previous implementations of noise generation circuits have used large devices to short the supply rails briefly to generate supply droops [24]–[26]. However, this approach requires custom layout and carefully validated circuits to ensure that the shorting devices behave as intended and will not inadvertently engage, which would be catastrophic. An alternative approach of using a large ring oscillator [14] has the advantage that it can be implemented using standard-cells and automatic place and route tools. However, ring oscillators can take significant silicon area for a large voltage droop range.

In this paper, we use a novel approach to short the supply rails using only conventional CMOS standard cells. Fig. 9 shows an overview of the on-chip noise generation circuits, which consist of an array of shorted inverter pairs that induce V_{DD} droops by driving one of the inputs with an on-chip pattern generator, while the other input of the pair is tied low. This causes short-circuit current between V_{DD} and V_{SS} when the two inputs differ, i.e., in this condition, one inverter pulls the intermediate node high, while the other pulls low, shorting the supply. The pattern generator unit is able to implement impulses, steps, and other droop waveforms. The test chip implements 256 inverter pairs, enabled by an 8-bit select code. The measured noise generator current is given in Fig. 9(b).

Fig. 9(b) also plots the impedance profile of the power delivery network (PDN) of the test chip that includes an off-chip linear regulator, PCB board, socket, and the flip-chip package, characterized using the on-chip noise generator. The voltage response is measured using a fast oscilloscope

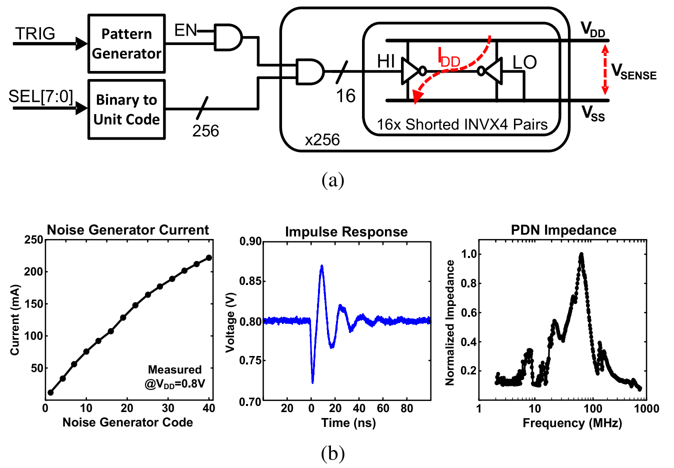


Fig. 9. All-digital on-chip noise generation circuits. (a) Block diagram. (b) Measured current versus noise generator code (left), measured PDN impedance response (middle), and impedance profile (right).

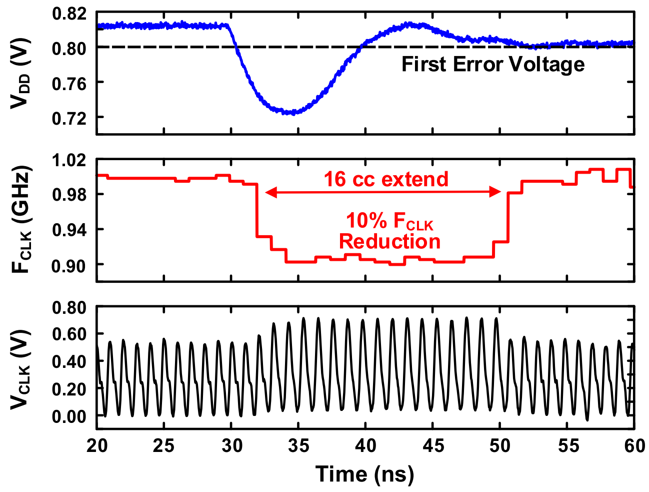


Fig. 10. Measured trace of V_{DD} droop waveform demonstrating adaptive clocking response.

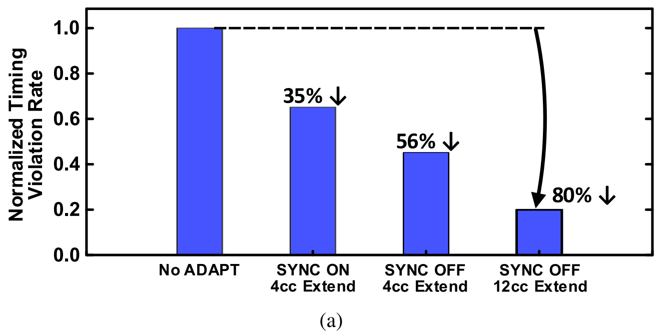
attached to dedicated sense pins on the package. The resonant frequency of the PDN is ~68 MHz for the DNN accelerator. The measured impulse response of the PDN is also shown. The droop magnitude is also calibrated using the same external oscilloscope and voltage sense pins.

At last, the fast-adaptive clock modulation was implemented on the test chip using an open-loop DCO. The required clock frequency is found incrementally, based on the current DCO output frequency, which is observed by dividing down on-chip and then measuring externally using an oscilloscope. The DCO can be embedded in a control loop without significant modification, as the fast modulation is well beyond typical loop bandwidths.

IV. Measurement Results

A. Fast Razor Adaptive Clocking

Measured scope traces shown in Fig. 10 confirm proper operation of the razor adaptive clocking architecture. A step current is generated by the noise generator, which induces a



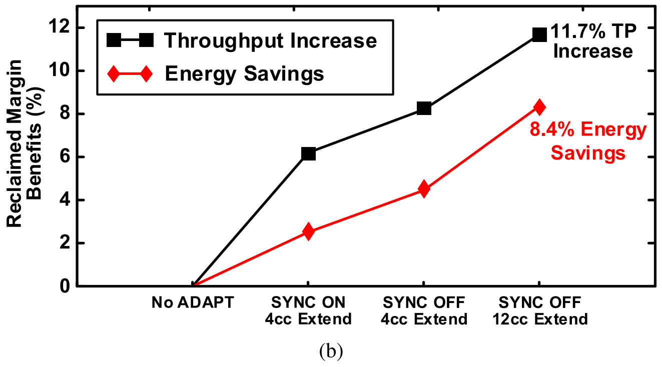


Fig. 11. Measurements of adaptive clocking system benefits (a) reduction in timing violation rate via reducing adaptation latency by removing the synchronization flop, and optimizing the F_{CLK} reduction extend results and (b) resulting throughput increase or energy savings due to reclaimed margin.

V_{DD} droop. Timing violations occur when V_{DD} falls below the "first error" voltage, which then triggers a 10% F_{CLK} reduction in the DCO within 3 clock cycles (cc). F_{CLK} reduction lasts for a configurable number of cycles via the Extend block (shown in Fig. 7) by holding the ERR signal high (16 cc in Fig. 10), allowing it to last until the fast droop event passes.

Fig. 11 summarizes the benefits of the proposed metastability-safe adaptive clocking across a range of adaptation latencies and F_{CLK} reduction cycles. The experimental setup induces a periodic voltage noise on-chip at the resonant frequency with a magnitude of 10% V_{DD} at the nominal V_{DD} = 0.8-V. DSTB latches at timing critical stages monitor and count timing violations due to this V_{DD} noise, which are aggregated and captured using on-chip performance counters. Fig. 11(a) shows the improvements in timing violation rates, normalized to the baseline error rate with the adaptive clocking system disabled. Enabling the adaptive clocking system, with the optional sync flops enabled (shown in Fig. 7) and 4 cc of F_{CLK} reduction, reduces timing violations by 35%. Without synchronization, adaptation latency reduces from 5 to 3 cc and timing violations now decrease by 56%. For the voltage noise scenario presented, extending the F_{CLK} reduction to an optimal setting of 12 cc further reduces timing violations, down by 80% overall. This reduction translates to 11.7% throughput improvement (by increasing the baseline clock frequency) or 8.4% energy savings (by reducing the baseline voltage) by reclaiming some of the timing/voltage guardbands, as shown in Fig. 11(b).

Fig. 12 shows the classification accuracy observed over the whole 10-K vector MNIST [27] test set versus voltage droop

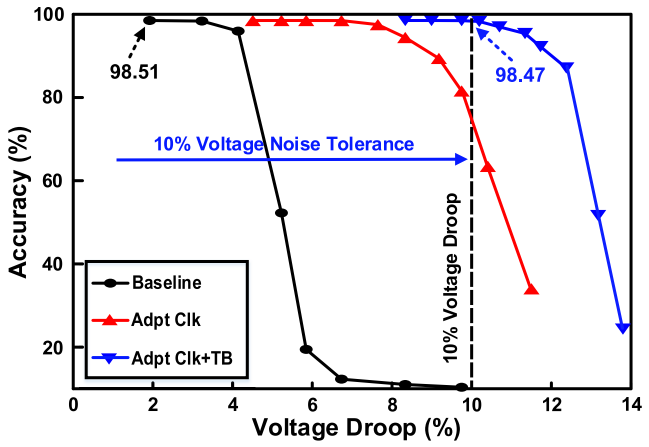
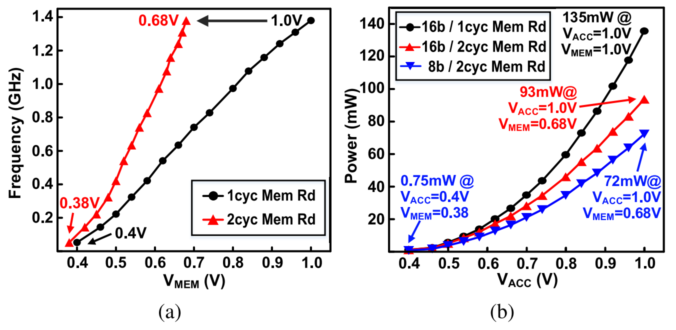


Fig. 12. MNIST classification accuracy versus voltage droop, demonstrating adaptive clocking with TB provides ~10% droop tolerance.

magnitude for resonant supply voltage noise. Without adaptive clocking, classification accuracy plummets as voltage droop magnitude increases beyond 4%. However, adaptive clocking enabled with TB to mask the initial trigger errors can tolerate up to 10% voltage droop with a negligible drop in accuracy. Since the adaptive clocking is driven by *in situ* timing error detection, the calibration is not required. This is in contrast to schemes that use a global canary circuit to trigger clock adaptation.

B. Multi-Cycle Banked SRAM

Although the DNN processor avoids costly off-chip memory accesses by storing all model parameters (weights) in the on-chip weight SRAM, memory power is still a significant portion of total power due to the size of the SRAM and the memory bandwidth required to support the eight-way SIMD unit. The proposed banked weight SRAM architecture, shown in Fig. 4, allows for a two-cycle SRAM read latency, which not only relaxes timing constraints for SRAM access but enables aggressive V_{MEM} scaling at iso-throughput to provide significant power savings across the entire operating range. Fig. 13(a) shows the measurement results from the multi-cycle banked SRAM voltage scaling. At one-cycle memory read latency, the weight SRAM (V_{MEM}) operates at the same voltage as the DNN accelerator (V_{ACC}). By allowing two-cycle read latency for the SRAM, V_{MEM} can scale independently of V_{ACC} without sacrificing throughput. For example, V_{MEM} can scale from 1 to 0.68-V with V_{ACC} still at 1-V, with no loss in performance. This results in significant power savings as shown in Fig. 13(b), which plots the aggregate power consumption of the DNN accelerator and weight SRAM for different operation modes. For example, at $V_{\mathrm{ACC}} = 1$-V, two-cycle read latency reduces overall power consumption from 135 to 93 mW for 16-bit weights with additional reduction down to 72 mW by switching to 8-bit weights. At the other end of the voltage range, the DNN accelerator and weight SRAM achieve 750 μW at $V_{\mathrm{ACC}} = 0.4$-V and $V_{\mathrm{MEM}} = 0.38$-V using two-cycle SRAM read latency and 8-bit weights. Fig. 13(c) summarizes the resulting energy improvements for three operating points: maximum frequency, nominal voltage, and minimum energy. Multi-cycle read latency provides 31%, 24%, and 10% energy improvements, respectively, at the three operating points for 16-bit weights. Overall, this scheme improves energy efficiency across the entire operating range. Finally, Fig. 14 shows a breakdown of power consumption split between the DNN accelerator and the weight SRAM at the nominal voltage (0.8 V). Enabling two-cycle SRAM read and switching to 8-bit weights reduce the SRAM access power, thereby increasing the compute to memory access power ratio and improving the overall energy efficiency.



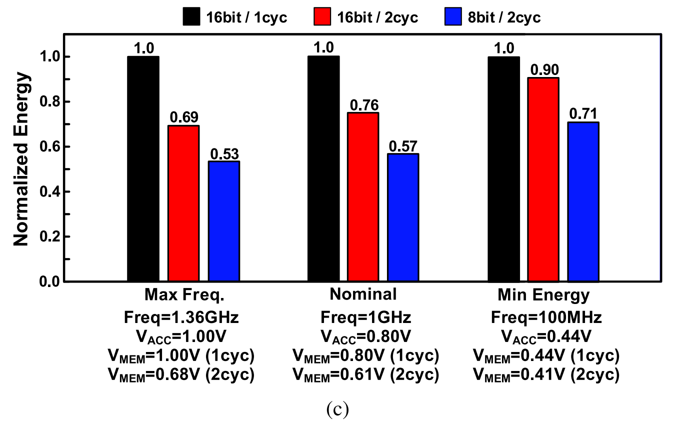


Fig. 13. Measured benefits of multi-cycle banked SRAM read. (a) SRAM operating voltage scaling at iso-throughput. (b) FC-DNN accelerator and weight SRAM power savings. (c) Energy savings for the MEP, nominal voltage, and maximum frequency operating points.

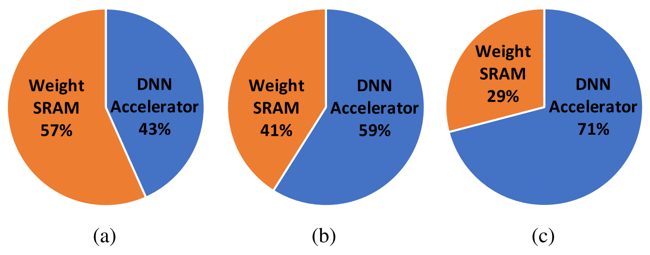


Fig. 14. Measured power breakdown of weight SRAM and DNN accelerator at nominal voltage 0.8-V for (a) one-cycle SRAM read with 16-bit weights, (b) two-cycle SRAM read with 16-bit weights, and (c) two-cycle SRAM read with 8-bit weights.

C. Energy and Throughput

Five always-on sensor classification workloads [28] were mapped onto the DNN processor (Table II). The workloads used consist of: binary face matching (*FACE*) [29], keyword

TABLE II
ALWAYS-ON SENSOR WORKLOADS

Name	Application	Dataset	Topology
FACE	Face Pair Verification	Labeled Faces in the Wild	512-128-128-128-2
KWS	Audio Keyword Spotting	Command	403-200-200-12
HAR1	Human Activity Recognition	Opportunity	924-56-56-56-56-18
HAR2	Human Activity Recognition	Smartphone Raw	384-72-72-72-72-13
DIGIT	Handwritten Digit Classification	MNIST	784-256-256-256-10

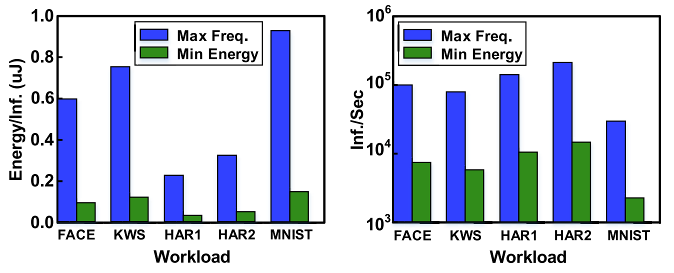


Fig. 15. Energy/inference (left) and inferences/second (right) for IoT workloads at maximum frequency and MEPs.

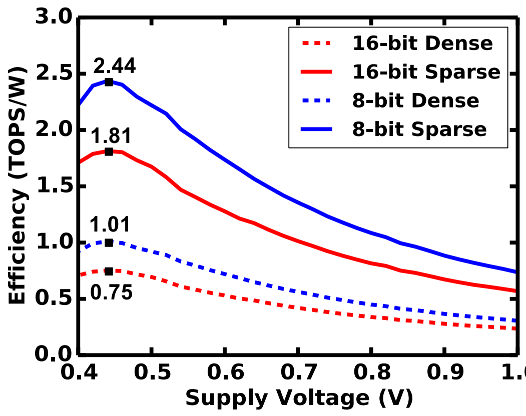


Fig. 16. Measured raw efficiency over the operating voltage range for dense and sparse operations.

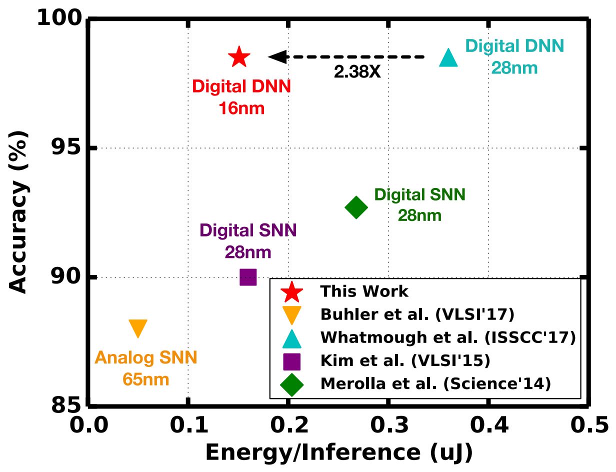


Fig. 17. Energy versus accuracy comparison for published MNIST results measured in silicon. Data are from the following references: Buhler *et al.* (VLSI'17) [36], Whatmough *et al.* (ISSCC'17) [7], Kim *et al.* (VLSI'15) [37], and Merolla *et al.* (Science'14) [38].

spotting (*KWS*) [30], two different human activity recognition scenarios (*HAR1* [31] and *HAR2* [32]), and handwritten digit classification (MNIST) [27]. Fig. 15 reports the measured energy efficiency (energy per inference) and throughput (inferences per second). Two operating points are given for each workload, one at F_{MAX} and one at the minimum energy point (MEP). All five workloads run at well below μJ.

Raw energy efficiency is reported in Fig. 16, as a function of the supply voltage. With 16-bit data and no sparsity optimization, the measured energy efficiency tops out at 750 GOPS/W at the MEP, which includes all memory power. This is the highest reported raw GOPS/W to date at this precision, even amongst CNN accelerators with very high data reuse [33]–[35] not achievable with FC layers. Compared to other work that similarly does not rely on off-chip DRAM [1], this is a 2× improvement on the state of the art. For 8-bit operands with sparse data prediction, the measured efficiency increases to 2.44 TOPS/W at the MEP, which yields the lowest reported MNIST energy per inference of 151 nJ at 98.51% accuracy, which is 2.3× lower than [1] and [7].

D. Comparison With State of the Art

Table III gives a comparison of adaptive clocking schemes published in silicon to date. Compared to prior works, the proposed adaptive clocking scheme does not require any complex calibration, which is a significant advantage. Removing the flip-flop synchronizers from the control path achieves compelling speed-up in adaptation latency, with droop detection and F_{CLK} adjustment each taking ~1 cc. Hashimoto *et al.* [18] achieved comparable results by also removing synchronizers from the droop detection and frequency modulation control path. In addition to the fast adaptation latency, the use of distributed razor DSTB latches allows the capture of fast local droops unlike the global detection circuit used in [18]. The DNN inference accelerator has the unique property that the workload is error tolerant to a degree. This is not the case for the adaptive clocking schemes published for CPUs.

Finally, Fig. 17 shows the comparison of the measurements of our DNN processor with other published accelerator application-specific integrated circuit (ASIC) results on the MNIST benchmark. This is arguably the most worthwhile approach, since simpler operations per second per watt (OPS/W) metrics become confusing with different models, operand bitwidths, storage requirements, and so on.

Fig. 17 includes both DNNs and spiking neural networks (SNNs). Implementations of SNNs [36]–[38] seem to occupy a frontier that is at least an order of magnitude greater in energy and generally offer poor accuracy. The best reported convolutional neural network result for MNIST [34] has too high energy/inference to include on our plot.

TABLE III
COMPARISON OF ADAPTIVE CLOCKING SCHEMES

	This work ESSCIRC'18 [10]	**Floyd ISSCC'17 [16]**	**Bowman JSSC'16 [3]**	**Wilcox JSSC'15 [17]**	**Hashimoto JSSC'18 [18]**
Technology	16nm FinFET	14nm FinFET	16nm FinFET	28nm Bulk	20nm CMOS
Design	SIMD DNN Accelerator, 1MB Banked SRAM	4 CPU cores and cache	Floating-point MAC unit	2 CPU cores 2MB cache	3 CPU cores
F_{CLK} used	1GHz	4GHz	2.5GHz	2GHz – 4GHz	5GHz
Droop Detection Method	DSTB pulse latch. Local – multiple timing monitors	Analog S&H, ref, comparator. Local – multiple timing monitors	Calibrated delay line Global – single detector circuit	Digital DLL. Global – single detector circuit	Time-to-digital Converter Global – single detector circuit
F_{CLK} Adjustment Method	Metastable safe oscillator loading (frequency shift)	PLL code change (frequency shift)	Divide-by-2 (frequency shift)	DLL and phase selector logic (phase shift)	PLL code change (frequency shift)
Calibration?	NO	YES	YES, automatic	YES	YES
Total Latency	$<$3cc	6ns	Not reported	6cc CDC effect	Not reported
Detection + Routing	$<$1.3cc	3ns	1cc	1cc	1cc
Sync + Adjustment	$<$1cc	1.8ns	1–2 cc	1.5cc	1cc
Clock Tree	$\sim$0.5ns	0.9ns	Not reported	Not reported	Not reported
Margin Reclaimed	11.7% throughput or 8.4% energy @10% droop	8% power or 3.5% performance	10% throughput @10% droop	3–6% lower voltage, 7–19% power reduction	7.5% frequency increase or 5% lower voltage
Error Tolerant	YES	NO	NO	NO	NO

The analog SNN implementation in [36] reports impressive energy/inference, although accuracy is very limited. Zhang *et al.* [9] presented an implementation of an MNIST classifier where the computation is performed in a standard 6-T SRAM. Results show very low energy efficiency of 630 pJ/inference, but poor accuracy of 90% on a heavily cut-down version of MNIST (9 $\times$ 9 pixel images instead of 28$\times$28). A related publication on an in-SRAM implementation by Ando *et al.* [39] does not report energy/inference, but again demonstrates poor accuracy of 90.1% with 22 $\times$ 22 images. An in-SRAM CNN design by Biswas and Chandrakasan [40] reports much better accuracy of 96%, but does not report energy/inference. Analog and in-memory architectures are also generally less robust to aggressive voltage scaling and require challenging analog circuit design and layout.

V. CONCLUSION

This paper described a 16-nm always-on DNN processor that achieves state-of-the-art efficiency on IoT DNN inference tasks, which is a key enabling technology for always-on mobile and IoT devices. The fabricated test chip demonstrated a number of novel techniques, including razor timing error rate-driven automatic calibration-free voltage/frequency scaling, multi-cycle banked SRAM scheme to relax the SRAM read cycle time, and a fast-adaptive clocking scheme for robustness. Measurement results show that the DNN processor has a wide dynamic range and can operate at F_{MAX} of up to 1.36 GHz, and achieves low-power operation all the way down to 750 μW at the minimum voltage point. We also achieve state-of-the-art raw efficiency at 16-bit precision of 750 GOPS/W dense or 1.81 TOPS/W sparse.

ACKNOWLEDGMENT

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The authors would like to thank ARM for providing physical IP.

REFERENCES

[1] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018.

[2] J. Tschanz *et al.*, "A 45 nm resilient and adaptive microprocessor core for dynamic variation tolerance," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 282–283.

[3] K. A. Bowman *et al.*, "A 16 nm all-digital auto-calibrating adaptive clock distribution for supply voltage droop tolerance across a wide operating range," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 8–17, Jan. 2016.

[4] D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, and D. Blaauw, "A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 284–285.

[5] N. Kurd *et al.*, "Next generation Intel core micro-architecture (Nehalem) clocking," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1121–1129, Apr. 2009.

[6] S. Bang *et al.*, "14.7 A 288 μW programmable deep-learning processor with 270 KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 250–251.

[7] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "14.3 A 28 nm SoC with a 1.2 GHz 568nJ/prediction sparse deep-neural-network engine with$>$ 0.1 timing error rate tolerance for IoT applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 242–243.

[8] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μj/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 222–224.

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

[10] S. K. Lee, P. N. Whatmough, N. Mulholland, P. Hansen, D. Brooks, and G.-Y. Wei, "A wide dynamic range sparse FC-DNN processor with multi-cycle banked SRAM read and adaptive clocking in 16 nm FinFET," in *Proc. IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2018, pp. 158–161.

[11] K. A. Bowman *et al.*, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 194–208, Jan. 2011.

[12] S. Das, G. S. Dasika, K. Shivashankar, and D. Bull, "A 1 GHz hardware loop-accelerator with razor-based dynamic adaptation for energy-efficient operation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 8, pp. 2290–2298, Aug. 2014.

[13] P. N. Whatmough, S. Das, D. M. Bull, and I. Darwazeh, "Circuit-level timing error tolerance for low-power DSP filters and transforms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 6, pp. 989–999, Jun. 2013.

[14] P. N. Whatmough, S. Das, Z. Hadjilambrou, and D. M. Bull, "Power integrity analysis of a 28 nm dual-core ARM cortex-A57 cluster using an all-digital power delivery monitor," *IEEE J. Solid-State Circuits*, vol. 52, no. 6, pp. 1643–1654, Jun. 2017.

[15] V. J. Reddi *et al.*, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling," in *Proc. Int. Symp. Microarchitecture (MICRO)*, 2010, pp. 77–88.

[16] M. S. Floyd *et al.*, "26.5 Adaptive clocking in the POWER9 processor for voltage droop protection," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 444–445.

[17] K. Wilcox *et al.*, "Steamroller module and adaptive clocking system in 28 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 24–34, Jan. 2015.

[18] T. Hashimoto *et al.*, "An adaptive-clocking-control circuit with 7.5% frequency gain for SPARC processors," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 1028–1037, Apr. 2018.

[19] X. Zhang *et al.*, "A fully integrated battery-powered system-on-chip in 40-nm CMOS for closed-loop control of insect-scale pico-aerial vehicle," *IEEE J. Solid-State Circuits*, vol. 52, no. 9, pp. 2374–2387, Sep. 2017.

[20] S. K. Lee, T. Tong, X. Zhang, D. Brooks, and G.-Y. Wei, "A 16-core voltage-stacked system with adaptive clocking and an integrated switched-capacitor DC–DC converter," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1271–1284, Apr. 2017.

[21] C. R. Lefurgy *et al.*, "Active management of timing guardband to save energy in POWER7," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, New York, NY, USA, Dec. 2011, pp. 1–11.

[22] P. N. Whatmough, S. Das, and D. M. Bull, "Analysis of adaptive clocking technique for resonant supply voltage noise mitigation," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2015, pp. 128–133.

[23] L. Ravezzi and H. Partovi, "Clock and synchronization networks for a 3 GHz 64 bit ARMv8 8-Core SoC," *IEEE J. Solid-State Circuits*, vol. 50, no. 7, pp. 1702–1710, Jul. 2015.

[24] S. K. Lee, D. Brooks, and G.-Y. Wei, "Evaluation of voltage stacking for near-threshold multicore computing," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, Jul. 2012, pp. 373–378.

[25] T. Tong, S. K. Lee, X. Zhang, D. Brooks, and G. Y. Wei, "A fully integrated reconfigurable switched-capacitor DC-DC converter with four stacked output channels for voltage stacking applications," *IEEE J. Solid-State Circuits*, vol. 51, no. 9, pp. 2142–2152, Sep. 2016.

[26] E. Alon and M. Horowitz, "Integrated regulation for energy-efficient digital circuits," *IEEE J. Solid-State Circuits*, vol. 43, no. 8, pp. 1795–1807, Aug. 2008.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[28] S. Kodali, P. Hansen, N. Mulholland, P. Whatmough, D. Brooks, and G.-Y. Wei, "Applications of deep neural networks for ultra low power IoT," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 589–592.

[29] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Univ. Massachusetts, Amherst, MA, USA, Tech. Rep. 07-49, Oct. 2007.

[30] P. Price, W. M. Fisher, J. Bernstein, and D. S. Pallett, "The DARPA 1000-word resource management database for continuous speech recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1, Apr. 1988, pp. 651–654.

[31] R. Chavarriaga *et al.*, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognit. Lett.*, vol. 34, no. 15, pp. 2033–2042, Jan. 2009.

[32] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, Jan. 2016.

[33] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 246–247.

[34] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *Proc. IEEE Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2016, pp. 1–2.

[35] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2016, pp. 262–263.

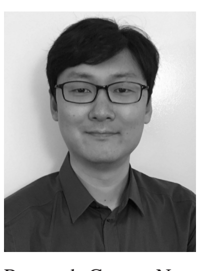
[36] F. N. Buhler, P. Brown, J. Li, T. Chen, Z. Zhang, and M. P. Flynn, "A 3.43 TOPS/W 48.9 pJ/pixel 50.1 nJ/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40 nm CMOS," in *IEEE Symp. VLSI Circuits Dig. Tech. Papers*, Kyoto, Japan, Jun. 2017, pp. C30–C31.

[37] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 640M pixel/s 3.65 mW sparse event-driven neuromorphic object recognition processor with on-chip learning," in *IEEE Symp. VLSI Circuits Dig. Tech. Papers*, Kyoto, Japan, Jun. 2015, pp. C50–C51.

[38] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[39] K. Ando *et al.*, "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 Tops at 0.6 W," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.

[40] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–490.



Sae Kyu Lee (M'10) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2006, the M.S. degree in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2008, and the Ph.D. degree from Harvard University, Cambridge, MA, USA, in 2016.

He was with Intel Corporation, Austin, TX, USA, and Advanced Micro Devices, Boxborough, MA, USA, where he worked on mobile microprocessor design. He is currently with IBM T. J. Watson Research Center, New York, NY, USA. His research interests include energy-efficient accelerator design for machine learning applications and very large scale integration (VLSI) design for efficient on-chip power delivery solutions.

Paul N. Whatmough (M'09) received the B.Eng. degree (Hons.) in electronic communications engineering from the University of Lancaster, Lancaster, U.K., in 2003, the M.Sc. degree (Hons.) in communications systems and signal processing from the University of Bristol, Bristol, U.K., in 2004, and the Ph.D. degree in electronic engineering from University College London, London, U.K., in 2012.

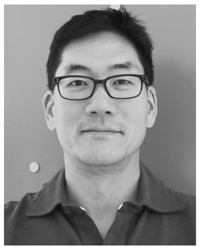
From 2005 to 2008, he was a Research Scientist with Philips/NXP Research Labs, Cambridge, U.K., where he working on hardware architecture and signal processing for software defined radio. From 2008 to 2015, he was with the Silicon R&D Department, ARM Ltd., Cambridge, where he working on hardware accelerators, digital signal processing (DSP), variation tolerance, and supply voltage noise and circuits and systems for emerging IoT applications. From 2015 to 2017, he was a Research Associate with Harvard University, Cambridge, MA, USA, where he leading inter-disciplinary research on machine learning (ML). He is currently with Arm ML Research Group, Boston, MA, USA, where he leads research on hardware for ML. He is also a part-time Associate with Harvard University. He coauthored the book *Deep Learning for Computer Architects*ž (Morgan & Claypool, 2017).

Dr. Whatmough is a member of IEEE. He was a recipient of the IET Student Project Award in 2003, the IEEE Communications Chapter Award in 2004, the European Wireless Technology Conference (EuWiT) Young Engineering Prize in 2008, and the IEEE/ACM International Symposium on Low-Power Electronic Design (ISLPED) Best Paper Award in 2015. He has served on the Technical Program Committee for numerous conferences, including the International Symposium for Computer Architecture (ISCA), the International Symposium on Microarchitecture (MICRO), the International Symposium for High-Performance Computer Architecture (HPCA), the ISLPED, and Design Automation Conference (DAC).

David Brooks (F'16) received the B.S. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1997, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 1999 and 2001, respectively.

He is currently the Haley Family Professor of computer science with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. His current research interests include resilient and power-efficient computer hardware and software design for high-performance and embedded systems.

Dr. Brooks was a recipient of several honors and awards including the ACM Maurice Wilkes Award and ISCA Influential Paper Award.



Gu-Yeon Wei (SM'00) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1994, 1997, and 2001, respectively.

He is currently a Robert and Suzanne Case Professor of electrical engineering and computer science with the Paulson School of Engineering and Applied Sciences (SEAS), Harvard University, Cambridge, MA, USA. His research interests include multiple layers of a computing system: mixed-signal integrated circuits, computer architecture, design tools for efficient hardware, and identifying synergistic opportunities across these layers to develop energy-efficient solutions for a broad range of systems from flapping-wing microrobots to machine learning hardware for IoT devices to large-scale servers.