An Ultra-Low Power Binarized Convolutional Neural Network-Based Speech Recognition Processor With On-Chip Self-Learning

Shixuan Zheng, Peng Ouyang, Dandan Song, Xiudong Li, Leibo Liu, Shaojun Wei, and Shouyi Yin

***Abstract*— Always-on speech interfaces are prevailing in human-machine interaction, especially on wearable devices, Internet of Things, etc., which benefits from the recent breakthroughs in deep learning. For battery-powered devices, ultra-low power and real-time processing are critical. However, the massive memory access and computation of deep neural networks (DNNs) lead to long latency and huge energy consumption, which hinder their further integration in battery-powered devices. Extremely low-bit quantization shows the potential to enhance energy efficiency and speed by orders of magnitudes, but it suffers from the degradation of recognition accuracy. In this paper, we propose a binarized convolutional neural network (BCNN) based speech recognition processor, integrated with on-chip self-learning mechanism to compensate the accuracy loss caused by low-precision. We optimize the BCNN architecture by eliminating the computation redundancy, compressing the weights and tailoring approximate circuits. Fabricated in 28 nm CMOS, this processor supports real time speech recognition with power consumption of 141 μW and energy efficiency of 2.46 pJ/Neuron. Compared with state-of-the-art speech recognition implementations, this processor achieves 2.5× reduction on energy consumption per neuron, and 8.0× energy reduction per speech frame.**

***Index Terms*— Speech recognition, low power, binary convolutional neural networks, frame-level reuse, on-chip learning, approximate computing.**

I. Introduction

AS A convenient and user-friendly human-machine communication fashion, speech interface has become a key feature of smart devices. As it converts human voices into machine-comprehensive commands, users can operate the devices and get services more efficiently.

Fig. 1 shows the components of modern automatic speech recognition (ASR) systems, which mainly consists of voice activity detection (VAD), feature extraction, phoneme classification, and decoding. In recent years, researches on deep learning have made breakthroughs in ASR. Specifically, deep neural networks (DNNs), recurrent neural networks (RNNs), and long-short-term-memory (LSTM) are adopted as the acoustic model, which outperforms the traditional GMM-HMM approach on a variety of speech recognition tasks [1]. Besides these fully-connected models, convolutional neural networks (CNNs) have shown compact model size and decent performance ([2]), which are adequate for ASR systems implemented on power-constrained devices.

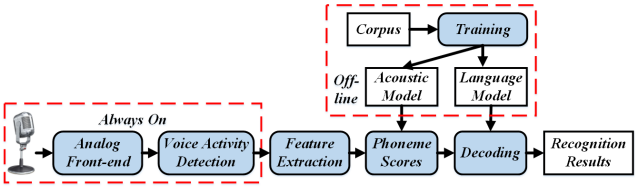


Fig. 1. Block diagram of general speech recognition systems.

An increasing number of battery-powered devices are adopting speech recognition as the human-machine interface, such as mobile phones, smart watches, and Internet-of-Things (IoT) devices. Considering battery lifetime, low power consumption and high energy efficiency are the key objects in the designing of such systems. However, current neural network based ASR cannot meet these requirements, since the state-of-the-art neural networks with a number of hidden layers and massive parameters demand a large number of memory access and arithmetic computations. For example, Google's CLDNN [2] is a combination of CNNs, LSTMs, and DNNs for speech recognition tasks, which executes about 1.5G MAC operations and 13 Million weight in inference, leading to unacceptable compute latency and power consumption. This feature of DNN based ASR prevents their extensive adoption in power-constrained devices.

Previous research works have been proposed to improve energy efficiency of DNN based speech recognition engines. Price *et al.* [3] adopts sparsity encoding to compress DNN weights, and voice-activated power gating to reduce core power and the overhead of clock tree. In [4], TrueNorth chip is configured as an end-to-end audio processing pipeline. Trade-offs between accuracy, power, and performance of this platform are explored. Bang *et al.* [5] design a non-uniform memory access architecture to optimize data movement and storage, consuming low energy on keyword spotting task.

Manuscript received May 7, 2019; revised August 14, 2019; accepted September 13, 2019. This work was supported in part by the National Key Research and Development Project under Grant 2018YFB2202600, the NSFC under Grant 61774094, and the China Major S&T Project under Grant 2018ZX01031101-002. This article was recommended by Associate Editor P. K. Meher. *(Corresponding author: Shouyi Yin.)*

S. Zheng, L. Liu, S. Wei, and S. Yin are with the Institute of Microelectronics, Tsinghua University, Beijing 100084, China (e-mail: yinsy@tsinghua.edu.cn).

P. Ouyang, D. Song, and X. Li are with TsingMicro Tech. Ltd., Beijing 100080, China.

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

Digital Object Identifier 10.1109/TCSI.2019.2942092

Since these works focus only on architectural and circuit level optimization, the energy efficiency of these works are still bounded by the intrinsic complexity of DNNs.

Apart from architectural and circuit level techniques, one promising way to greatly enhance energy efficiency is quantizing network parameters with low bit-width on algorithmic level. Binary convolutional neural networks (BCNNs) have been proposed to reduce the scale of DNN models [6], [7]. In these works, the forward passes (inference) compute with binarized parameters, and the backward passes (training) update the shadow parameters in floating point. In this way, the weights and activations can be quantized to 1 bit, which saves the memory footprint by 32×, enabling purely on-chip weight storage and eliminating expensive off-chip memory access. Moreover, since the resource-consuming multipliers are replaced by XNOR logics, the energy efficiency and performance of DNNs could be further enhanced. Xiang *et al.* [8] have applied binary neural networks in ASR systems. Comparing with floating point DNNs, they accelerate the inference by at most 7.2× and 5.4× on CPU and GPU platforms, respectively. However, although binary neural networks show the above advantages, they degrade the recognizing accuracy. According to [8], on TIMIT phoneme recognition task, the relative word error rate (WER) is increased by about 7%, and on 50-hour Switchboard speech recognition task, the relative increments of WER ranges from 6.7% to 12.9%.

Therefore, to design an ultra-low power BCNN-based ASR processor, two problems need to be addressed. Firstly, a highly energy efficient, low power BCNN architecture is necessary for power-constrained applications. Secondly, a BCNN-oriented on-line learning mechanism is required to compensate accuracy loss due to binarization.

To pursue high energy efficiency, we consider three characteristics of BCNNs in speech processing. (1) Previous BCNN accelerator designs, such as YodaNN [9], FINN [10], and FPGA-based BNN accelerator in [11], focus mainly on the spatial locality of convolution, and design dataflows that can maximally reuse data, thereby minimize the memory access. However, apart from spatial locality, CNN based speech recognition also shows temporal locality, since contiguous speech feature maps are largely overlapped. This chance can be utilized to further reuse input data and improve energy efficiency. In this work, we propose Frame-Level Reuse to exploit temporal data locality and eliminate the redundancy in BCNN computation. (2) In order to optimize the memory access patterns of input data and weights, we elaborately partition the on-chip memory and propose Bit-Level Regularization to compress BCNN weight matrices. Different from previous pruning approach [12], this method focuses on enhancing the regularity of binary weight matrices. (3) Since multipliers are replaced by XNORs, additions become dominant in BCNN computation. To further improve efficiency, we propose a dedicated approximate adder to shorten the critical datapath and reduce power-delay-product (PDP), while maintaining the recognizing ability of BCNN.

To address the second problem, we propose on-chip self-learning to extract speaker-specific features, update the BCNN weights at runtime, and thereby compensate the accuracy loss due to binarization. Until now, network parameters are mostly trained off-line, on general purpose architectures (CPU, GPU) with well chosen and labeled training samples. However, in real-world scenarios, such samples are unavailable, and the supervised learning should be replaced by unsupervised approaches. Speaker adaptation has been studied to adapt a generally trained model to serve specific users or select the most effective samples for network retraining [13]–[17]. In this work, we propose a novel framework for on-line self-learning, which includes runtime training sample selection, gradient back-propagation, and weight updating of the last BCNN layer. With this framework implemented on chip, the recognition performance of this processor is expected to be enhanced incrementally when serving specific users.

In summary, we propose an 141 μW, 2.46 pJ/Neuron BCNN based speech recognition processor (named after "Thinker-S") [18], with the following features:

- A configurable architecture optimized for various BCNN topologies is proposed in this work, which supports flexible sizes of feature maps, convolutional kernels, and number of cascaded layers. Frame-level reuse, which is a spatial-temporal 2-Dimensional BCNN computation optimization, is leveraged to maximize data reuse, and eliminate the redundancy in convolutional computation. To optimize memory access and storage, we tailor the memory layout of BCNN, and propose Bit-level Regularization to compress BCNN weight matrices, along with hybrid-bank memory.
- Approximate circuits are designed to strike the balance between accuracy and energy consumption of BCNN operations. The dedicated approximate adder and linearly approximate softmax function are proposed for BCNN inference and back-propagation respectively.
- We implement runtime self-learning on this chip. A novel framework is proposed to select training samples and generate labels during runtime. By updating weights according to the user-specific speech features, the recognition word error rate (WER) is reduced relatively by 14.37%–51.4%.

The remainder of this paper is organized as follows. Section II introduces the design of our proposed speech recognition framework. Section III proposes the top-level architecture of this speech recognition processor, illustrating its main components, functions, and features. Section IV describes our BCNN optimization techniques and approximate circuit designs in detail. Section V explains the on-chip self-learning framework and its hardware implementation. Section VI analyzes the experimental results of recognition accuracy and hardware energy efficiency. The chip specifications and comparison are also shown. Finally, Section VII concludes the paper.

II. BCNN Based Speech Recognition System

A. BCNN Based Speech Recognition Flow

Fig. 2 shows the proposed BCNN based speech recognition flow. It takes digital speech signals as inputs, and outputs the recognized wakeup word, voice commands, and phoneme scores. Five modules are included, namely VAD, feature extraction, acoustic model, decoding, and self-learning.

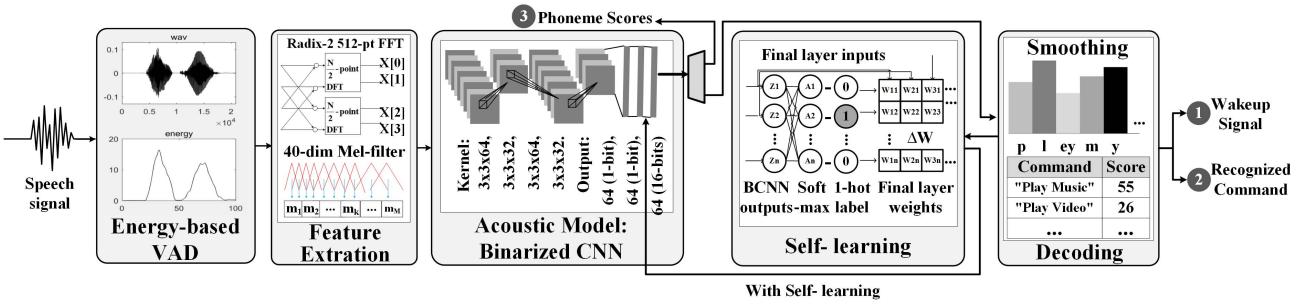


Fig. 2. BCNN based speech recognition flow.

TABLE I
STATISTICS OF PROPOSED BCNN

Neural network	Layer number	Bit-width (weights)	Bit-width (inputs)	Kernel Size	Input Feature ×Channel	Output Feature ×Channel	MAC-ops (MOPS)	Weights (KB)
CONV Layers	1	1	16	3×3	11×40×1	9×38×64	0.20	0.07
	2	1	1	3×3	9×38×64	7×36×32	4.64	2.25
	3	1	1	3×3	7×36×32	5×34×64	3.13	2.25
	4	1	1	3×3	5×34×64	3×32×32	1.77	2.25
FC Layers	5	1	1	–	640	64	0.041	5
	6	1	1	–	64	64	0.0041	0.5
	7	16	1	–	64	64	0.0041	8
Total	–	–	–	–	–	–	9.79	20.32

(1) The digital speech signals are first processed by VAD unit. The length of each speech frame is set as 25 ms at a sampling rate of 16 KHz, which results in 400 points per frame. Two contiguous frames have an overlap of 15 ms. During runtime, only VAD unit needs to be always-on, and the downstream units could be turned off to save power consumption. The design of VAD circuits have been extensively studied [3], [19], [20], which is not the focus of this paper. We adopt an simple energy estimation solution, which calculates the energy by accumulating the signal's amplitude, and compare it with a threshold to activate the downstream units.

(2) For extracting features, we use discrete Fourier transformation and mel-frequency filter bank to convert time domain speech signals to 40-dimensional feature vectors. The 400-point frame is first zero-padded to 512 points, and then processed by radix-2 512-point FFT, transformed into frequency-domain signals, finally squared into power spectrum values. In the subsequent filtering, mel-frequency cepstral coefficients (MFCCs) [21] are adopted. MFCCs are an approximation of anthropic nonlinear frequency sensing characteristic, which represent short-term power spectrum of speech frames.

(3) Then, the filtered feature vectors are sent to acoustic model, where BCNN is adopted to generate phoneme-level classification results. This BCNN consists of 4 convolutional (CONV) layers and 3 fully-connected (FC) layers. Binarized networks conform the similar formula with the floating-point version, except replacing multiplication with 1-bit XNOR. The computation of one output point is expressed in Equation 1:

$$O_{ij} = \sum_{c=1}^{Ch} \sum_{g=0}^{k-1} \sum_{h=0}^{k-1} I[i+g][j+h] \cdot W[g+1][h+1] \quad (1)$$

where I, W, O, Ch, and k represent input feature maps, weights, output feature maps, number of channels, and the kernel size, respectively.

The network topology and statistics on computation demand and memory footprint is listed in Table I. The inputs of first layer are 16-bit speech features, and the weights of the final FC layer are set as 16 bits to improve classification accuracy. The number of MAC operations is mainly contributed by CONV layers, and the majority of weights are from FC layers.

The batch normalization and binarization follow after each layer's convolution. The standard flow consists of additions, multiplications and sign function (comparing with 0). In our design, they are equivalently fused into only one addition and comparison (Equation 2). In the simplified equation, only two coefficients are needed.

$$y = Sign(\lambda \cdot (x-\mu) \cdot i + B) = Sign(\lambda \cdot i \cdot (x + \frac{B}{\lambda \cdot i} - \mu)) \quad (2)$$

where x and y represent the input and output of batch normalization and binarization, λ, μ, i, and B are coefficients. The outputs of BCNN (phoneme scores) could be sent to external decoders for continuous speech recognition task. The detailed operations of BCNN are shown in Fig. 3.

(4) Decoding uses the phoneme scores to search for predefined wakeup words and speech commands. In this work, we adopt smoothing method for the searching module. It is a phoneme histogram based decoding algorithm, which analyzes the occurrence of each phoneme in speech frames, and matches the voice command with the maximum likelihood.

(5) Self-learning is an innovative feature in ASR frameworks. As mentioned before, BCNN decreases the computation complexity, especially eliminates the expensive multiplications. However, binarization causes roughly 10%

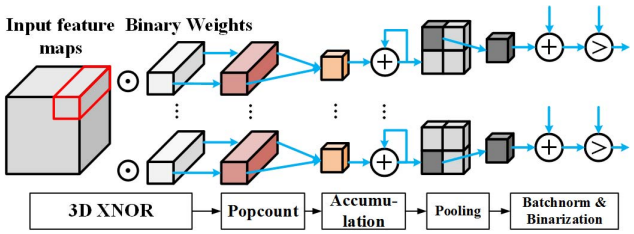


Fig. 3. Operations in each layer of BCNN: 3D XNOR, popcount (1-bit-addition), accumulation (16-bit-addition), pooling (optional), batch-normalization, and binarization. The last two operations are combined into one in our implementation.

accuracy loss. To guarantee satisfactory recognizing performance, we leverage speaker-adaptive network fine-tuning to enhance BCNN based speech recognition framework. This self-learning mechanism is implemented in hardware on this processor. At runtime, it can update the BCNN weights based on users' speech input.

Self-learning first processes the outputs of final FC layer with softmax function, and generates one-hot labels. The effective training samples are selected based on the phoneme-level confidence and decoding results. Then the weight increments are calculated using standard back-propagation algorithm. Finally, 16-bit weights of final FC layer are updated after batches with positive speech frames.

B. Analysis of BCNN and On-Chip Self-Learning

In this recognition flow, BCNN is the most time and energy consuming unit, it is also the bottleneck of recognition accuracy due to binarized parameters. Therefore, in this work we mainly focus on improving the computing latency and energy efficiency of BCNN, as well as enhancing its classification accuracy by implementing on-chip self-learning with minimum overhead, with regard to the following 4 aspects.

Firstly, on-chip self-learning incurs extra hardware overhead and compute latency, which mainly results from complex nonlinear functions and gradients calculation. In this work, we optimize the design of nonlinear functions, reuse hardware resources for weight updating, and hide the computing latency by pipeline scheduling.

Secondly, BCNN for speech recognition also shows temporal locality: the continuous two input feature maps have large portion of overlap. In this network, over 80% BCNN operations are redundant, which provides us the chance to exploit frame-level data reuse.

Thirdly, we study the memory access pattern of BCNN. We develop memory partitioning to reduce the activation access latency. Furthermore, by regularizing the BCNN weight matrices, we compress the BCNN weights to reduce the weight access.

Fourthly, additions (1-bit and 16-bit) become the dominant operations in BCNN. And we observe that nearly 96% of additions in our BCNN are popcount (1-bit), while 16-bit additions take up only 4%. Therefore, adopting accurate popcount and approximate 16-bit addition will further improve energy efficiency.

III. Chip Architecture

A. Overall Architecture

Fig. 4 shows the top-level architecture of the proposed speech recognition processor, which mainly consists of energy-based VAD unit, feature extraction and self-learning unit, BCNN unit, smoothing unit, main controller, and shared memory. The inputs of this processor are 16-bit speech signals, and the outputs include wakeup signals, recognized voice commands, and the phoneme scores. The weights and activations of BCNN are totally stored on chip, using 26 KB weight memory and 4 KB shared data memory, respectively. During runtime, the units on chip are clock-gated, e.g. the feature extraction unit works only when the speech signal passes its previous VAD unit, and so forth.

1) Front-End Units: The digital speech signals enter the *energy-based VAD unit*, after they are processed by 0-mean computing and accumulated to compare with the already set threshold, the result is used to activate downstream units. After that, the speech signals are multiplied with a predefined window function. The *Fbank and Self-learning unit* includes a reconfigurable datapath, softmax function, and mel-frequency filter. When configured for feature extraction, it executes a radix-2 512-pt FFT. Then the frequency domain signals are processed by a mel-frequency filter bank, with the coefficients stored on chip. After the logarithmic function, the 40-dimension speech features are extracted. When configured for self-learning, this datapath receives the gradient values to update the BCNN weights. The softmax unit is used in back-propagation, its outputs are used for training sample selection and gradient calculation.

2) BCNN Unit: This configurable BCNN unit works in a layer-by-layer manner, and supports flexible sizes of input/output feature maps, convolutional kernels, and the number of cascaded layers, which will be further analyzed in Section III-B. Eleven frames of speech features constitute a 11×40 input feature map as the input of BCNN. The 1-bit activations and weights are first multiplied in 3072 XNOR gates, and then processed in 96 32b-popcount units and accumulated into output feature maps. After pooling function, batch normalization and binarization, the outputs of each layer are generated. Considering the relatively fewer operations (see table I), we allocate only 64 XNOR gates to binary FC layers. The computation of FC layers includes MAC (also implemented with XNOR, popcount, and accumulation) and batch normalization. Finally, at most 64-dimension 1-bit phoneme vectors are generated as acoustic model, which are transferred to smoothing unit. The 16-bit outputs and weights of the final FC layer are transferred to self-learning unit.

3) Smoothing Unit: The smoothing unit is built with an index decoder, phoneme buffers, keyword memory, and histogram computing logics under the control of scoring finite state machine (FSM). It first decodes the BCNN output vector into phoneme indexes, then calculates the histogram to detect the wake-up word and recognize voice commands. Based on this histogram, the predefined words with the maximum likelihood will be selected.

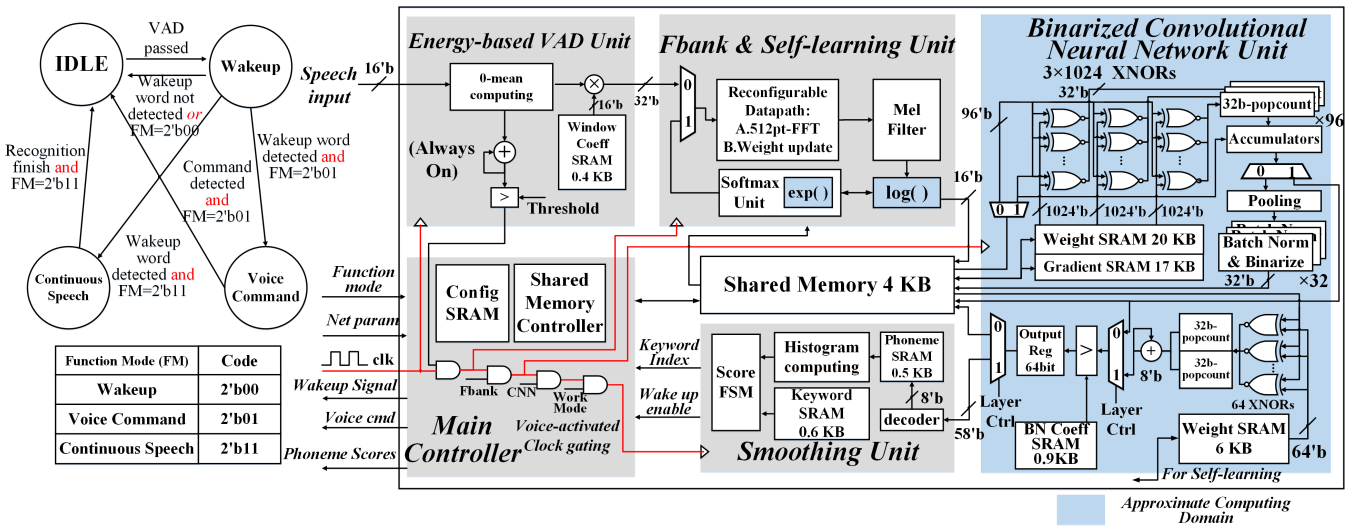


Fig. 4. Architecture of proposed speech recognition processor. The processor works in 3 function modes: wakeup word detection, voice command recognition, and continuous speech (outputs phoneme scores). The nonlinear functions and BCNN adders are implemented with approximate computing logics (shadowed in blue). During runtime, the downstream inactive units are clock gated to save energy (red line).

4) Main Controller: The main controller configures the chip's working states according to the function mode it receives, manages the clock-gating as well as the shared memory, and controls the IO interfaces of this processor. The working states transfer graph is also illustrated in Fig. 4. This chip can be configured into 3 function modes, which are wakeup, voice command, and continuous speech. When the input speech signals pass the VAD unit, the chip enters wakeup mode to detect the wakeup word. If detected, the chip begins to search for voice command or generate phoneme scores for continuous speech recognition according to the function mode.

B. BCNN Computation Flow

The iterative computation flow of BCNN is shown in Fig. 5, which operates as an output-stationary dataflow [22]. In each iteration step, 3 pixels from 32 input feature maps (channels) are selected and sent to XNOR units. Correspondingly, $3\times32\times32$ (3072) bits of weights are fetched from memory to conduct XNOR operation with these input pixels, producing immediate results for 32 output feature maps. In this way, the input pixels are reused, saving the feature access by $32\times$. After XNOR operations, each of 32-bit vectors is summed (popcount) into a partial sum, and these values are accumulated (16-bit addition) to generate output feature maps. For 16-bit operators, their sign bits are firstly sent to XNOR gates to operate with binary weights, and then combined with 96×15 value bits. They bypass the pop-counters and directly go to accumulators, as shown in Fig. 5. In the computation of FC layer, 64 bits of activations and weights are sent to XNOR gates at each cycle.

The iteration information of each layer is pre-calculated. Suppose the size of an output feature map is H×W×C (Height×Width×Channel), and the kernel size is $K_h\times K_w$ ($K_h = K_w = 3$ in this example), the number of iterations would be $H\times K_w\times W\times \lceil\frac{C}{32}\rceil$. For the network layers whose number of channel is not a multiple of 32, we will

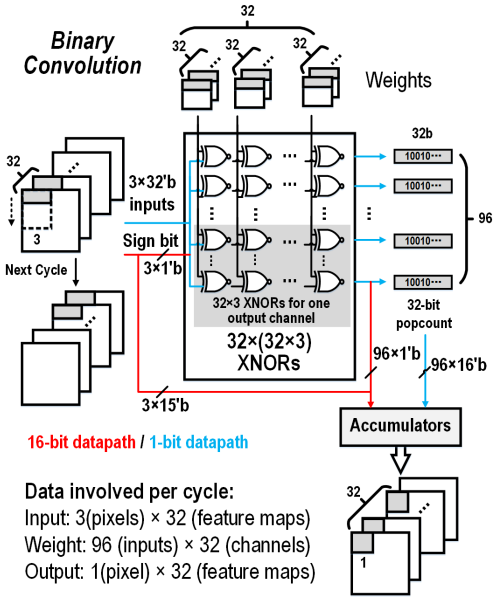


Fig. 5. Detailed dataflow of our BCNN implementation. At most 96 bits activations and 3072 bits weights are involved in 1 cycle.

append 1 for the remainder parts of feature map, and append 1 and 0 (representing -1) alternatively for the remainder parts of weights, making the accumulated results still being correct. These iteration information is loaded to configuration SRAM in the main controller (See Fig.4), and is transmitted to BCNN module under the management of main controller. In this approach, BCNN module can process networks with various topologies.

IV. BCNN Optimization Techniques

In this section, we explain our BCNN performance and energy-efficiency oriented techniques, namely (1) temporal-spatial locality and memory partitioning, (2) weight regularization based compressing, and (3) approximate adder design.

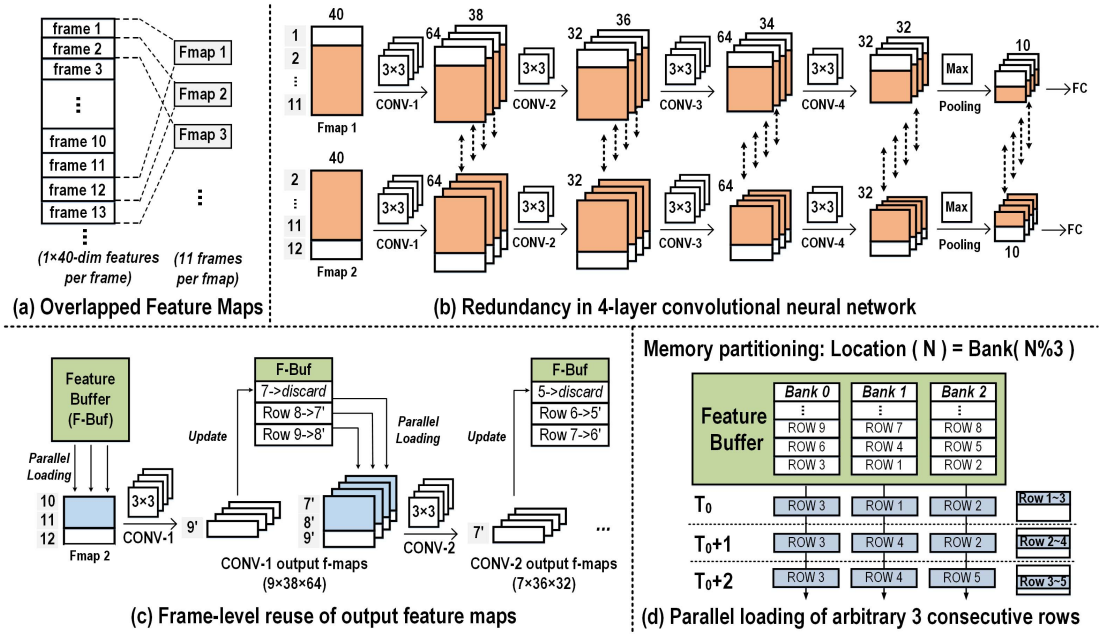


Fig. 6. Frame-level output feature reuse in speech convolutional neural networks and memory partitioning for parallel feature loading.

A. Frame Level Reuse for BCNN Based Speech Processing

When accelerating CNNs, spatial locality is usually exploited for data reuse, e.g. scheduling the computing in a specific stationary dataflow [22]. In the convolution of speech feature maps, apart from spatial locality, temporal locality is also observed, as shown in Fig. 6 (a). Each frame (40-dimension feature vector) is combined with 10 contiguous frames into an 11×40 sized feature map. Therefore, two consecutive feature maps have 10 frames in common (shaded). Fig. 6 (b) illustrates that when they are convolved with 3×3 kernels of the first convolutional layer, the output feature maps will have 8 rows in common (out of 9 rows in total). Similar phenomenon can be observed on all the subsequent layers. Considering all the convolutional layers in our proposed network topology (Table I), the total convolutional results are overlapped by 80.48% on average, leading to serious redundancy and poor energy efficiency. We address this problem by exploiting frame-level data reuse. The immediate data in 4 CONV layers generated from the starting input speech feature map (Fmap 1) are buffered as follows. First 3 layers buffer the last two rows of their output feature map, and the final layer buffers the whole output feature map except the oldest row. For the consequent speech inputs (Fmap 2), only the newest 3 frames (frame 10–12 in Fig.6(c)) of the input feature maps are used to compute the non-overlapping output row. And this row is combined with 2 buffered rows of the first layer to generate the non-overlapping row for next layer. The buffered features are updated by adding the newly generated row and discarding the oldest row of each channel. Each CONV layer repeats the similar approach until the final CONV layer, which sends the total 640-dimension features to FC units after pooling. This method is independent with the parameter bit-width and network topology. With this approach, 80.48% useless operations are eliminated in the convolutional layers of our network, which significantly improves the speed and energy efficiency.

In cooperation with the above frame-level activation reuse, we propose a memory partitioning technique to ensure conflict-free data loading, as illustrated in Fig. 6(d). In the proposed computing data-flow, features from 3 rows of input feature maps (across 32/64 channels) are needed for XNOR computation in one cycle. Therefore, parallel memory access of multiple rows is demanded in BCNN computation. We partition the feature buffer into 3 banks, ensuring the potential to provide multiple data stream. The mapping of data in the memory is decided by its row index modulo 3, e.g. the 4th row is stored in bank $4\%3 = 1$. As shown in the Fig.6(b), in this way, data from arbitrary 3 consecutive rows can be read out without conflict, e.g., rows 3,4,5 are loaded from bank 0,1,2, respectively.

B. Bit-Level Regularization of BCNN Weight Matrices

For the sake of reducing energy consumption of weight accessing, we consider the compression of BCNN weights. In 1-bit precision, weights are restricted in only two values: 1 and -1, and -1 is represented and stored as 0 for XNOR operations. As shown in Fig. 7, the initial distribution of zeros in the 3-D weight matrix is irregular. Continuous same bits at a given position can be compressed to save memory space and reduce the amount of memory access. Previous pruning approach successfully squeezes the CNN weights [12], but the index-based compressing is inefficient for binary weights. Aimed at hardware-friendly regular structures, we propose bit-level regularization to compress BCNN weight matrices. To be specific, in the regularization of the pre-trained BCNN model, we incrementally prune the nonzero bits into zeros starting from one end of weight matrices (see Fig. 7). The pruning stops when the recognition accuracy drops quickly.

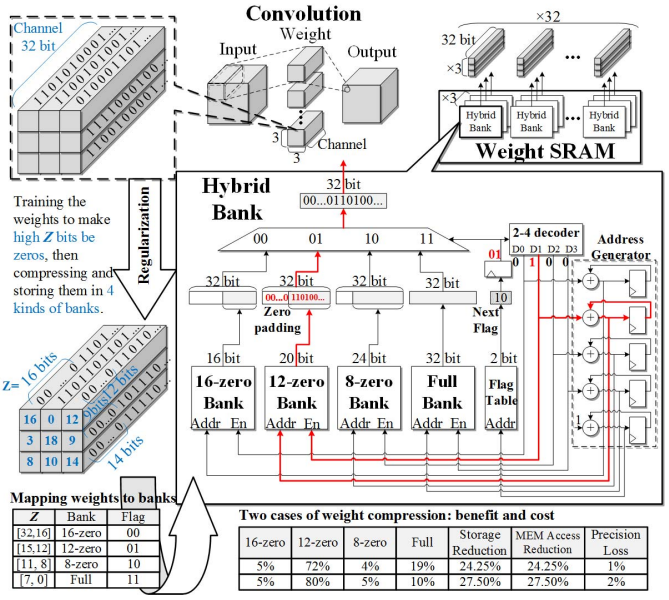


Fig. 7. Weight compressing is based on continuous zeros, which are derived via fine-tuning. The dedicated hybrid weight memory stores 4 kinds of banks, and decodes the weights when access happens.

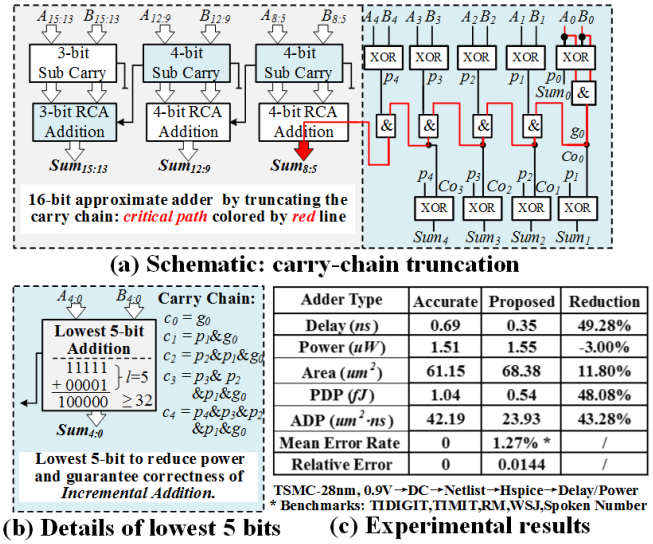


Fig. 8. Dedicated approximate adder [18]. (a) Schematic of this adder. (b) Details of lowest 5 bits. (c) Experimental results.

This approach makes zero bits to appear continuously in one end of the weight vector.

To store and decode the compressed weight matrices, dedicated hybrid weight memory is designed. The weights are stored in 4 kinds of banks: 16-zero, 12-zero, 8-zero, and full bank, where 16-zero bank means that the highest 16 bits of the stored weight vector are all zeros, and so forth. A 2b flag table is designed to record the bank types and direct the accessing of these hybrid banks. Each hybrid bank owns a separate address generator. In each cycle, the 2-4 decoder indicates which bank to read, and the address generator provides the exact address. For example, the current flag "01" indicates that the weights to be read is stored in the 12-zero bank. So the 2-4 decoder sends an enable signal to the 12-zero bank and activates the corresponding generator to increase the address by 1, which will also be sent to the 12-zero bank. Then the weights are read out from the 12-zero bank and attached with 12 zeros at the starting bits. With slight accuracy loss (1–2%), the weight accessing amount and energy are reduced by 24.25–27.5%.

C. Dedicated Approximate Adder for BCNN

From the synthesis report, the adders for popcounting binary products and accumulating partial-sums are on the critical path of the BCNN module. Therefore, the system speed benefits from the optimization of such adders. Since binarization removes the multiplications in BCNNs, additions become dominant. Among all the additions in our proposed network, 95.9% of which are 1-bit additions (popcount), and only 4.1% are 16-bit additions. To reduce latency and energy consumption, we propose a dedicated approximate adder to replace the standard 16-bit adder, introducing only negligible error rate. In Fig. 8, the circuit-level diagram and the experimental results are shown. The 16b approximate adder is designed as the combination of a 3b-Ripple Carry Adder (RCA), two 4b-RCAs and an approximate 5b-RCA. In two cases, the carry output of a 1-bit full-adder will be 1. Firstly, when the two input bits are both 1. Secondly, when one of the two input bits is 1, and the carry input is 1. For popcount operations, only the later case exists. Therefore, the circuit for lowest 5 bits can be optimized for only accurate popcount, by simplifying the carry generation logic into only one AND-gate. Since the number of input channels being concurrently operated is at most 32, the maximum result of popcount is also 32, which takes 5 bits to represent. When optimizing 16-bit addition, the carry inputs of the 3-bit RCA and the following 4-bit RCA are generated only from the sub carry producer, with the carry chain between these stages truncated. Therefore, the 16b approximate adder's critical path is determined by the 5b-RCA's carry chain and 4b-RCA's addition path (marked in red in Fig. 8). It is 49.28% shorter than an normal 16b-RCA's critical path, obtaining 48.08% Power-Delay-Product (PDP) reduction at 0.9V, 50MHz. Since it guarantees accurate increment additions, according to experiments, the mean error rate of additions is only 1.27% tested on public speech dataset. On phoneme-level classification and decoding results, the accuracy drops are only 1.4% and 0.5%, respectively.

V. On-Chip Self-Learning Techniques

In this work, we propose on-chip self-learning to improve recognition accuracy of BCNN. Instead of training off-line with CPU or GPU, we propose self-learning on this processor for runtime weight updating. It is challenging to implement such an idea for mainly three reasons: (1) When fine-tuning BCNN with real-world speech signals from specific users, labeled samples are unavailable, which need to be distinguished automatically with unsupervised approaches. (2) Implementation of back-propagation will inevitably introduce extra hardware overhead. Without being well optimized,

it may impair the low-power and high energy efficiency advantages of BCNN. (3) Training will also bring extra latency. Without careful scheduling, on-chip self-learning will degrade the latency of this speech recognition processor. We analyze these problems and propose our solutions in this section.

A. Algorithmic Design for On-Chip Self-Learning

The complete network training includes the feed forward process and back-propagation [23]. For classification tasks, which is the focus of this paper, the network outputs are firstly converted into probabilistic values via a $softmax$ function:

$$p_i = \frac{\exp(d_i)}{\sum_{j=1}^{N} \exp(d_j)} \tag{3}$$

where p_i represents the probability, and d_i represents one of the N outputs of final FC layer. Then, the loss function (L) is measured with a logistic function, from which the gradient values are derived, as shown in the equations below (4).

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial d_j} \cdot \frac{\partial d_j}{\partial W_{ij}} = \left(\sum_{k=1}^{N} \frac{\partial L}{\partial p_k} \cdot \frac{\partial p_k}{\partial d_j}\right) \cdot \frac{\partial d_j}{\partial W_{ij}} = (p_j - l_j) \cdot x_i \tag{4}$$

where x_i is the i-th input of final FC layer. The gradient is multiplied with learning rate η, and is subtracted from old weight to update it. Moreover, momentum method is usually leveraged to accelerate gradient descent [24], as shown in equation 5.

$$W_{ij}^{(new)} = W_{ij}^{(old)} - \eta \cdot \frac{\partial L}{\partial W_{ij}} + m \cdot \delta W_{ij} \tag{5}$$

where δW represents the previous increments, and m represents the momentum factor.

Previous research has shown that slight modification of an already trained neural network can lead to better adaptation towards specific speakers, and effective training samples can be automatically selected from unlabeled data. In [13], authors use speaker-specific features to partially modify the pre-trained network, transforming speaker-independent models to particular speakers targeted models, thereby reducing the recognition error. In [16], confidence scores of un-transcribed utterances are computed based on both acoustic model and the language model, and the utterances with smallest scores are selected for transcription. Our proposed self-learning mechanism only use the effective samples of one specific speaker for fine-tuning, "overfit" the network parameters to adapt to this speaker, and thereby improve the recognition rate under specific situation.

We choose effective training samples from the actual speech inputs based on both the batch-level voice command recognition and frame-level phoneme classification results at runtime. Firstly, we check the decoding results on each batch of frames (currently set as 300). Since the initial speaker-independent network is pre-trained with general corpus and shows eligible accuracy, the false accept rate is considerably low. Therefore, when one batch is detected as wakeup word or voice command, 300 frames in this batch are highly likely to be positive samples. Secondly, for each frame, after BCNN inference and softmax conversion, the maximum value among 64 softmax outputs indicates the phoneme it belongs to. The higher this value, the more likely the classification is correct. Therefore, we compare the maximum value with a threshold to judge the likelihood level of single frame. If not exceeded, this sample will be abandoned. However, as a special case, silent speech input will disable this strategy, because it is also included in 64 softmax output values and can easily exceeds the judgment threshold. Therefore, we also abandon the silent frames. Constrained by these two conditions, most negative samples are abandoned and the effective speech inputs are used to update the 16-bit weights of final FC layer.

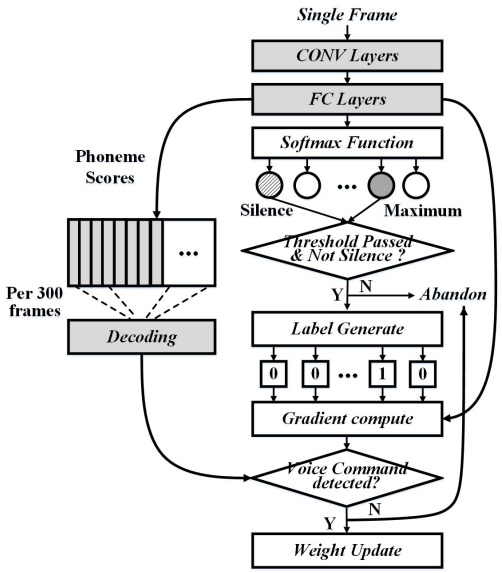


Fig. 9. Work flow of on-chip self-learning.

The labels for self-learning are generated during runtime. We use one-hot labels for gradient computing: setting the output node with the maximum value (probability) as 1, and the other nodes as 0. This strategy adapts the network to the positive samples of a specific user, making the classification more robust. Therefore, even if the utterance is slightly deviated from the standard pattern, it can also be recognized.

In conclusion, the complete work flow of our on-chip self-learning is illustrated in Fig. 9. The outputs of FC network are sent to softmax function, where the probabilities are calculated. Then the maximum and silent value are compared with the threshold. One-hot labels will be generated for the chosen samples, gradients of these frames will also be calculated and used to update the accumulated weight increments stored in the gradient buffer. The weight updating is only executed for batches with positive decoding results.

We adopt HOME corpus [25] and fine-tune the proposed BCNN according to this work flow. The experimental results show that, after 20 self-learning epochs the Wakeup word accuracy is enhanced by at most 3.3% (Section VI-C).

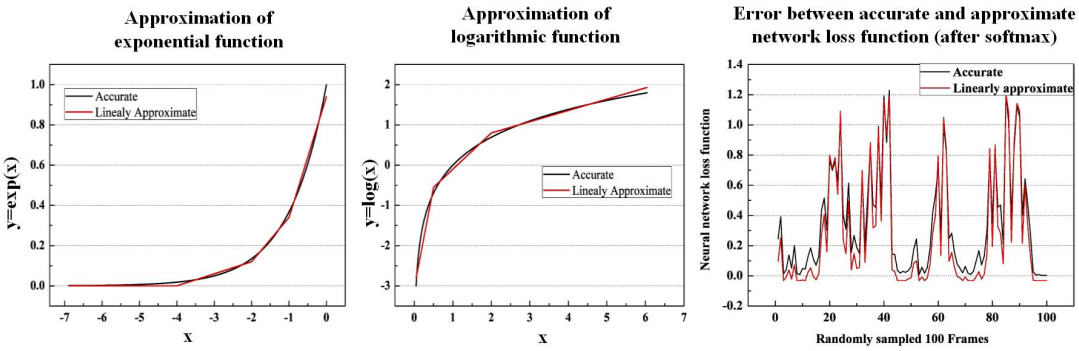


Fig. 10. Software simulation results of exponential function, logarithmic function, and error of neural network final loss function. The black lines are the accurate functions, and the red lines represent the results of linearly approximate functions.

And among 70k training frames, 65.7% of which pass the two conditions and are chosen for weight update. And 95.2% of these selected frames are validated to be positive samples.

B. Implementation of On-Chip Self-Learning

Apart from BCNN unit which produces the feed forward results, we implement softmax function, gradient calculation, weight updating, and extra control logics for back-propagation.

1) Implementation of Nonlinear Functions: In this work, we implement the nonlinear functions with linear approximation. The formula of softmax function is transformed to facilitate hardware implementation, as described below.

$$p_i = \frac{\exp(d_i)}{\sum_{j=1}^{N}\exp(d_j)} = \frac{\exp(d_i - d_{max})}{\sum_{j=1}^{N}\exp(d_j - d_{max})}$$
$$= \exp(d_i' - \log(\sum_{j=1}^{N}\exp(d_j'))) \quad (6)$$

where d_i' represents the d_i subtracted by d_{max}. The maximum value is subtracted from all 64 softmax inputs (16-bit BCNN outputs), which is equivalent to dividing both the numerator and the denominator by $\exp(d_{max})$, and will not change the value of the fraction. This approach reduces the input range of exponential functions to be equal or smaller than zero, restricting the output value to only $[0, 1]$, making it easier to implement exponential function with a piecewise linear function. Moreover, as division consumes too much cycles and area, we remove it by cascading logarithmic and exponential functions. Logarithmic function does not bring extra area overhead because it has already been implemented in Fbank unit.

2) Analysis of Approximate Functions: Obviously, the whole transformation will not change the value of softmax function. The approximation is only from the piece-wise linear functions. The exponential and logarithmic functions are plotted in Fig. 10, as well as the approximation error of neural network loss function among 100 speech samples, which indicates that the errors are tolerable. To measure the influence of the approximation to the final accuracy, we conduct experiments on HOME corpus [25] for fine-tuning tasks, and the influence of linear softmax function on training convergence is illustrated in Fig. 11, which shows

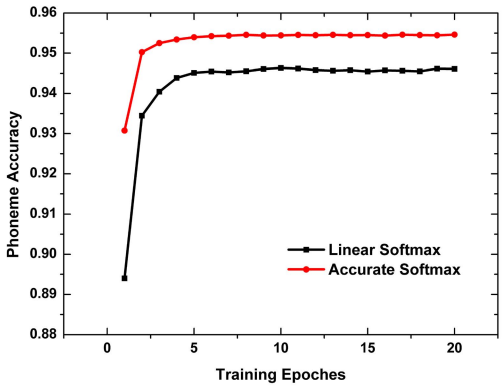


Fig. 11. The influence of linearly approximated softmax on fine-tuning convergence.

only a slight phoneme classification accuracy loss (~1%). And the accuracy drop of the final decoding results due to linear softmax function is only 0.3%, which proves the feasibility of our approximation approach.

3) Hardware Reuse for Weight Updating: We reuse the datapath of FFT calculation for weight updating to reduce the hardware overhead of self-learning. Fig. 12 shows the compute data-path of FFT (red), weight-update data-path (blue), and how they share the same hardware resources. The $a + bi$ and $c + di$ are time-domain complex signals as the FFT input, while $e + fi$ represents the complex coefficient of the butterfly computation. In accord with equation 3-6 in manuscript, p_j, l_j, and x_i represent the softmax output, label, and activation, respectively. While m and $\hat{\mathrm{I}}\cdot$ represent the momentum factor and the learning rate, respectively. The compute units choose the proper input data (VAD Unit or Gradient SRAM) according to the function control signal from the main controller, and send the outputs to the corresponding receivers (Mel filter or Weight SRAM). Fig. 12 also shows the detailed data-path design on the operator level, illustrating how the two functions share resources through multiplexers.

4) Pipeline Scheduling: To hide the latency introduced by self-learning, we propose dedicated pipeline scheduling. Shown in Fig. 13 (b), Fbank of next frame is scheduled in parallel with BCNN of current frame, and self-learning begins at the same time of smoothing. Before adopting this

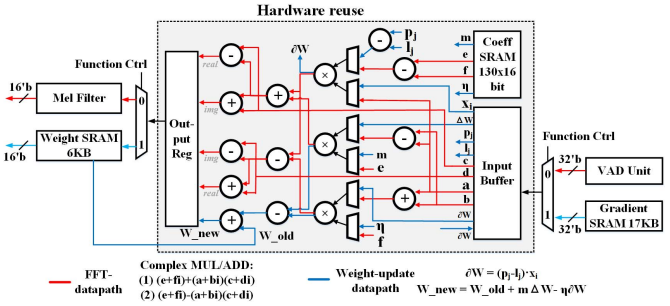


Fig. 12. Hardware reuse for FFT (red line) and weight update (blue line).

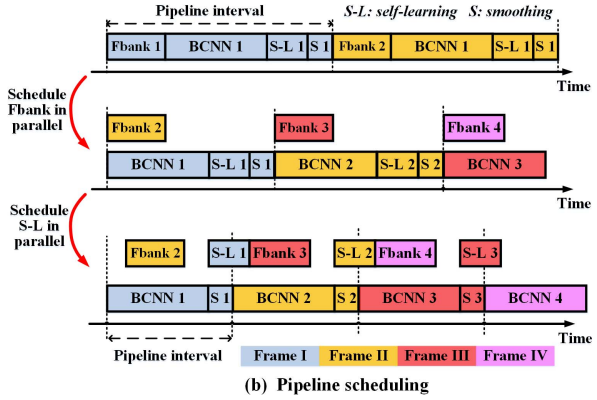
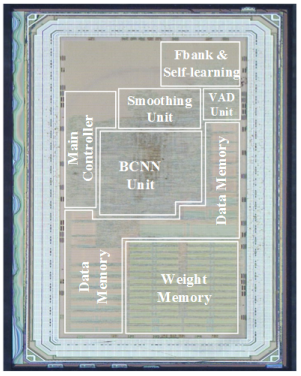


Fig. 13. Pipeline scheduling of on-chip self-learning hardware design.

strategy, the time consumption of one single pipeline stage is the sum of Fbank, BCNN, self-learning, and smoothing. After re-scheduling, it becomes only the sum of BCNN time and smoothing time. Therefore, the parallelized scheduling has shorter pipeline interval than straight forward sequential scheduling. This strategy shortens the latency by 45%. Benefited from shortened latency, the resource utilization is enhanced by 18.9%, as the idle time of each unit becomes shorter.

VI. Chip Implementation and Evaluation

This speech recognition processor is fabricated in 28nm CMOS technology. In the evaluation, we choose TIMIT [26], TIDIGITS [27], and Home Appliances Control Speech (HOME) corpus [25] as benchmarks. TIMIT includes 6300 sentences from 630 speakers. TIDIGITS contains 326 speakers, with each of them pronouncing 77-digit sequences. HOME includes a 300-hour training set and a 50-hour testing set, covering 2 wakeup words, 10 voice commands, and 150 speakers. In order to fairly validate the efficacy of self-learning, same neural network model should be used. However, due to unreleased network models and different training datasets, it is hard to rebuild the exact neural networks with the same weights and neurons of the previous works. Therefore, in this paper, we only use our self-developed model to test the efficacy of self-learning mechanism. The validation and training of our proposed BCNN based speech recognition framework is conducted using Kaldi toolkit [28] with denoised data. The chip specification is shown



Process	28 nm CMOS
Supply voltage	0.57 V - 0.9 V
Frequency	2.5 MHz - 50 MHz
Core size	1.74 × 0.74 mm^2
Die size	2 × 1 mm^2
Logic gates (NAND2)	0.78 million
SRAM	52 KB
Power consumption	141 uW - 2.85 mW
Energy efficiency	90 TOPS / W
Energy per frame	141 nJ / Frame
Neural network efficiency	2.46 pJ / Neuron

Fig. 14. Die photo and chip specifications [18]. An available demo video is in [29].

in section VI-A. The breakdown of chip area and power consumption are shown in section VI-B. And in section VI-C, we evaluate the performance of proposed speech recognition processor. Comparison with state-of-the-art designs is presented in section VI-D.

A. Chip Specification

The die photograph and chip specification are shown in Fig. 14. The core size is 1.29 mm^2, and the die size is 2 mm^2. The supply voltage scales from 0.57 V to 0.9 V. The power consumption reaches 141 μW at 0.57 V, 2.5 MHz, and is up to 2.85 mW at 0.9 V, 50 MHz. With totally 52 KB on-chip memory, all the weights and activations are stored on chip. During test, the host sends the newest speech data of 10 ms long to the chip at each time. These data are combined with previously buffered 15ms overlapping data to form a new frame, which is further combined with previous 10 frames to form a 11×40 feature map. Finally, the chip sends the recognition result back to host. The latency of this procedure (regarded as 11 frames being processed) ranges from 0.5 ms (@50 MHz) to 10 ms (@2.5 MHz). The energy efficiency reaches the peak value of 90 TOPS/W at 0.59 V. At the minimum supply voltage of 0.57 V, the peak neural network efficiency and energy per frame of this processor are only 2.46 pJ/Neuron and 141 nJ, respectively.

B. Area Breakdown, Power Breakdown, and Data Statistics

The area and power breakdowns are shown in Fig. 15 and Fig. 16, respectively. The area breakdown is derived according to area report of placement and routing tools. The power breakdown is calculated by performing post placement and routing simulations using actual workloads. Since apart from XNOR operations, large amount of 32-bit popcount and 16-bit addition are required, BCNN unit becomes the most area and power consuming part, which takes up 36.47% chip area and 38.69% power. IO takes up 27.15% area and 24.56% power. Since the network is binarized to 1-bit, the on-chip memory consumes only 18.74% area. With the memory access reduced by the output stationary dataflow and weight compression, the power dissipation raito of memory is only 9.64% power.

TABLE II
DATA STATISTICS OF EACH SPEECH FRAME ON THIS CHIP

Unit	VAD	FBANK and S-L	BCNN	SMOOTHING	TOTAL
Input data	0.8 KB	0.8 KB	5.94 KB	16 B	7.56 KB
Output data	2 B	0.08 KB	5.38 KB	–	5.46 KB
Weight/Parameter	2 B	1.84 KB	20.32 KB	–	22.2 KB
Memory writing	0.8 KB	11.3 KB	0.80 KB	4 KB	17.0 KB
Memory reading	1.6 KB	16.4 KB	271.87 KB	4 KB	293.9 KB
# of operations	400	3.49K	9.79M	10.3 K	9.8M

TABLE III
BREAKDOWN OF MEMORY ACCESS ON 7 NETWORK LAYERS

Layer	Input data volume (KB)	Memory reading of inputs (KB)	Output data volume (KB)	Memory writing of outputs (KB)	Weight data volume (KB)	Memory reading of weights (KB)
CONV1	0.86	0.47	2.67	0.24	0.070	2.16
CONV2	2.67	1.60	0.98	0.20	2.25	114.84
CONV3	0.98	1.18	1.33	0.27	2.25	85.90
CONV4	1.33	0.80	0.38	0.075	2.25	55.46
FC1	0.078	0.078	0.08	0.008	5	5
FC2	0.008	0.008	0.08	0.008	0.5	0.5
FC3	0.008	0.008	0.08	0.008	8	8
TOTAL	**5.93**	**4.14**	**5.38**	**0.80**	**20.32**	**271.87**

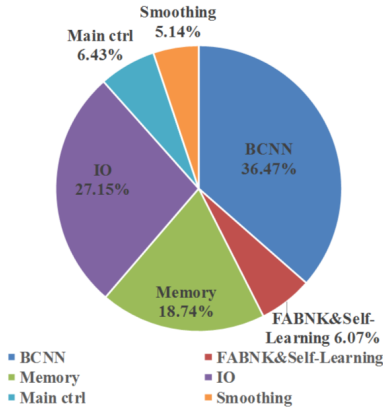


Fig. 15. Area breakdown.

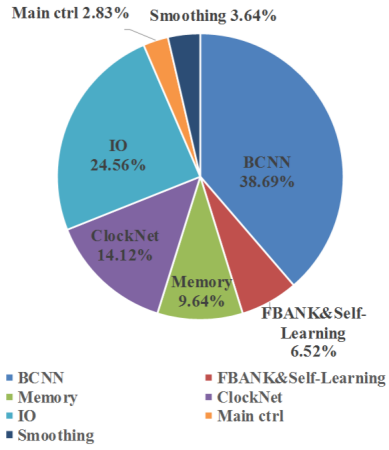


Fig. 16. Power breakdown.

On-chip self-learning brings 17KB SRAM, softmax circuits, and control overhead. However, with the weight updating reusing FFT circuits, self-learning occupies only 7.4% of total area (considering memory and logic). Moreover, the execution cycles of self-learning are hidden by pipeline scheduling.

TABLE IV
RECOGNITION ACCURACY OF WAKEUP WORDS

SNR(db)	Wakeup w/ S-L	Wakeup w/o S-L
5	86.9%	85.3%
10	88.3%	86.5%
15	90.9%	87.6%
20	92.3%	89.9%
25	93.9%	91.0%
30	95.3%	92.4%

TABLE V
RECOGNITION ACCURACY OF VOICE COMMANDS

Benchmark	TIDIGITS		HOME	
SNR (db)	w S-L	w/o S-L	w S-L	w/o S-L
5	95.6%	88.7%	85.1%	82.3%
10	97.0%	90.5%	88.4%	83.9%
15	97.4%	91.9%	93.0%	85.0%
20	97.6%	93.0%	94.0%	85.2%
25	98.2%	93.1%	95.0%	88.0%
30	98.6%	94.1%	96.0%	90.9%

TABLE VI
WORD ERROR RATE

Benchmark	TIMIT	TIDIGITS	HOME
w/o S-L	25.20%	5.18%	3.97%
w S-L	21.58%	3.89%	1.94%
Relative improvement	14.4%	24.9%	51.4%

Table II details the data statistics of the main units of this processor, listing their input data, output data, involved weights or parameters, memory access, and number of operations. BCNN consumes 271.87KB memory reading access because its weights need to be loaded for multiple times. FBANK and self-learning unit contributes 11.3KB memory

TABLE VII
COMPARISON WITH STATE-OF-THE-ART

Metric	ISSCC 2017 [3]	ISSCC 2017 [5]	JSPS 2016 [32]	TC 2017 [4]	This work
Technology	65 nm	40 nm	40 nm	28 nm	28 nm
Area	9.61 mm^2	7.1 mm^2	12 mm^2	4.3 cm^2	1.29 mm^2
Voltage	0.6 V – 1.2 V	0.63 V – 0.9 V	0.6 V	0.8 V	0.57 V – 0.9 V
Frequency	3 – 86 MHz	1.9 – 19.3 MHz	50 MHz	/	2.5 MHz – 50 MHz
Latency	Real-time	6.5 ms	10 ms	Real-time	0.5 ms (@50 MHz) – 10 ms (@ 2.5 MHz)
Power	Speech Recognition (SR): 172 uW – 7.8 mW	Keyword Spotting: 321 μW @3.9MHz 0.65V	Keyword detection: 11.2mW SR: 51.96mW	Keyword spotting: 38.59 mW	Wakeup/Voice Command/ASR: Min: 141 μW @2.5MHz, 0.57V Max: 2.85mW @50MHz, 0.9V
Normalized minimum power[1]	155 uW	247 μW	Keyword: 10.1 mW SR: 46.89 mW	19.59 mW	141 μW
Peak energy efficiency per frame (Normalized to 28nm^2)	/	2086 nJ/Frame (1123 nJ/Frame)	112 mJ/Frame (71mJ/Frame)	/	141 nJ/Frame @0.57V
Peak neural network efficiency (Normalized to 28nm^2)	16 pJ/Neuron (6.22pJ/Neuron)	2.54 nJ/Neuron (1.37nJ/Neuron)	140 nJ/Neuron (88 nJ/Neuron)	/	2.46 pJ/Neuron @0.57V
Energy efficiency (Normalized to 28nm^3)	/	318 GOPS/W (551 GOPS/W)	/	/	90 TOPS/W @0.59V

[1] Normalized power = Absolute power / (C_{abs}/C_{28nm}) / (f_{abs}/f_{28nm}) / (voltage$_{abs}$/voltage$_{28nm}$)^2 = Absolute power / (voltage/0.57V)^2
[2] Normalized energy per frame/neuron = (Absolute energy per frame/neuron) / (technology/28nm) / (voltage/0.57V)^2
[3] Normalized energy efficiency = Absolute energy efficiency $\times$ (technology/28nm) $\times$ (voltage/0.59V)^2

writing access out of 17.0KB in total, since it updates the weights of final FC layer during runtime. The statistics of data volume and memory access of 7 BCNN layers are listed in table III. The theoretical on-chip access for reading inputs is 25.88KB. With frame-level reuse, this number is reduced to 4.14KB, saving 84% reading energy. Similarly, the output writing is reduced from 4.82KB to only 0.80KB. Benefited from both frame-level reuse and bit-level regularization, weight access of BCNN is reduced greatly from 1.76MB to 271.87KB, which validates the effectiveness of our optimization techniques.

C. Evaluation of Speech Recognition With Self-Learning

We conduct experiments of wakeup word, voice commands, and continuous speech recognition. For each task, same BCNN model is tested under various Signal to Noise Ratios (SNRs), in order to evaluate the recognition accuracy of this chip and validate the effectiveness of self-learning. The accuracy is calculated on word-level, considering the influence of on-chip VAD. As shown in Table IV, with self-learning, the wakeup accuracy ranges from 86.9%–95.3%, as the SNR varies from 5db to 30db, which meets the recognition rate demand of mobile and wearable devices. Compared with original model without self-learning, the wake up accuracy is enhanced by 1.6%–3.3%. The measured false alarm rate (FAR) is only 0.01%, which is equivalent to only 2-3 times of false wakeups in the test of 24-hour speech data. Table V illustrates that for voice command recognition of both TIDIGITS and HOME, the absolute accuracy is enhanced by at most 8.8% with self-learning. We also measure the Word Error Rates (WERs) of continuous speech testing on TIMIT, TIDIGITS, and HOME. Shown in Table VI, WERs are reduced relatively by 14.37%, 24.9%, and 51.4%, respectively, which validates the effectiveness of proposed self-learning on continuous speech recognition tasks.

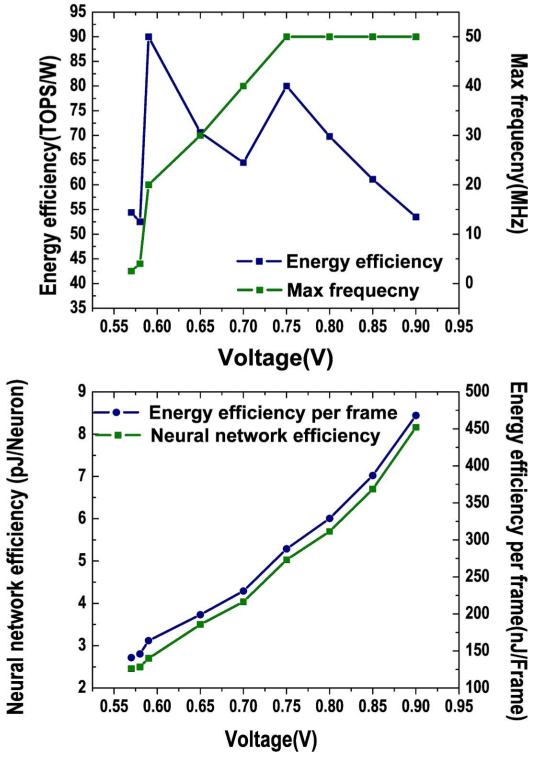


Fig. 17. Frequency and energy efficiency scales along with voltage [18].

D. Voltage Scaling Results and Comparison With State-of-the-Art Designs

As shown in Fig. 17, we measure the energy efficiency, maximum working frequency, energy consumed per frame, and neural network efficiency at a variety of supply voltages. This chip reaches its optimal energy efficiency of 90 TOPS/W at 0.59 V. At the minimum supply voltage of 0.57 V, the energy consumed per frame is 141 nJ, and the neural network efficiency is 2.46 pJ/neuron.

We also compare this processor with state-of-the-art designs in Table VII, scaling their performance and power consumption to 28nm technology. In [3], run-length encoding is adopted to process sparse weight matrices, and the precision of DNN weights ranges from 4-bit to 12-bit. Their recognizer consumes 172 uW to 7.8 mW power for speech recognition, and achieves peak neural network efficiency of 16 pJ/Neuron. The processor presented by [5] adopts a 3-layer FC network to spot keywords, with 6-bit weights, 12-bit activations, and 0.3M MACs, consuming 321 μW at 3.9 MHz and 0.65 V. The energy consumed per neural on this processor is much higher than designs of [3] and ours. This processor achieves 6.5 ms latency, peak neural network efficiency of 2.54 nJ/neuron, and power efficiency of 318 GOPS/W. In [30], an architecture is proposed for speech applications with fixed-point neural network. Their network for keyword detection consists of 2 FC layers with 400 neurons per layer, quantizing the weights to 5-bit. On keyword spotting and speech recognition (without decoder) tasks, its power consumption are 11.2 mW and 51.96 mW, respectively. TrueNorth is fabricated in 28nm and configured for always-on speech recognition in [4], working at 0.8 V. In TIDIGITS task, it consumes 38.59 mW power consumption. Our speech recognition processor aims at battery-powered devices, outperforming works of [30] and [4] by orders of magnitude on power consumption. On the energy consumed per frame and the energy consumed per neuron, our work also outperforms [30] by orders of magnitude. Comparing with [3] and [5] (normalized to 28nm), our processor reduces 9.0% and 42.9% power consumption, respectively. Its peak energy efficiency per frame is 8.0× better than [5], and energy consumed per neuron is 2.5× lower than [3], which benefits from neural network binarization and the proposed techniques.

VII. Conclusion

This paper proposes an ultra-low power speech recognition processor fabricated in 28nm CMOS technology, which achieves real time processing with energy efficiency of 2.46 pJ/Neuron and power consumption of 141 μW. Binary convolutional neural network (BCNN) is utilized and optimized with frame-level reuse, memory partitioning, and weight regularization. Dedicated approximate adder is elaborated to reduce latency with negligible accuracy loss. On-chip self-learning is designed and implemented on this chip to compensate the accuracy loss brought by network binarization, and improve the recognition performance towards an specific user incrementally. Comparing with state-of-the-art designs, this processor reduces 9.0%–42.9% power consumption, 2.5× lower energy consumption per neuron, and 8.0× lower energy per speech frame.

References

[1] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[2] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. Interspeech*, 2015, pp. 1478–1482.

[3] M. Price, J. Glass, and A. P. Chandrakasan, "A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 244–245.

[4] W.-Y. Tsai *et al.*, "Always-on speech recognition using TrueNorth, a reconfigurable, neurosynaptic processor," *IEEE Trans. Comput.*, vol. 66, no. 6, pp. 996–1007, Jun. 2017.

[5] S. Bang *et al.*, "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*, Feb. 2017, pp. 250–251.

[6] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 525–542.

[8] X. Xiang, Y. Qian, and K. Yu, "Binary deep neural networks for speech recognition," in *Proc. INTERSPEECH*, 2017, pp. 533–537.

[9] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An architecture for ultralow power binary-weight CNN acceleration," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 48–60, Jan. 2018.

[10] Y. Umuroglu *et al.*, "Finn: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2017, pp. 65–74.

[11] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*. New York, NY, USA: ACM, 2017, pp. 15–24. doi: 10.1145/3020078.3021741.

[12] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," Feb. 2016, *arXiv:1602.01528*. [Online]. Available: https://arxiv.org/abs/1602.01528

[13] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 12, pp. 1713–1725, Dec. 2014.

[14] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 7947–7951.

[15] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Proc. INTERSPEECH*, 2013, pp. 1248–1252.

[16] D. Hakkani-Tür, G. Riccardi, and A. Gorin, "Active learning for automatic speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 4, May 2002, pp. IV-3904–IV-3907.

[17] M. Mimura and T. Kawahara, "Unsupervised speaker adaptation of DNN-HMM by selecting similar speakers for lecture transcription," in *Proc. Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA), Asia–Pacific*, Dec. 2014, pp. 1–4.

[18] S. Yin *et al.*, "A 141 UW, 2.46 PJ/neuron binarized convolutional neural network based self-learning speech recognition processor in 28 NM CMOS," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 139–140.

[19] A. Raychowdhury, C. Tokunaga, W. Beltman, M. Deisher, J. W. Tschanz, and V. De, "A 2.3 nJ/frame voice activity detector-based audio front-end for context-aware system-on-chip applications in 32-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1963–1969, Aug. 2013.

[20] K. M. H. Badami, S. Lauwereins, W. Meert, and M. Verhelst, "A 90 nm CMOS,6 μW power-proportional acoustic sensing frontend for voice activity detection," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 291–302, Jan. 2016.

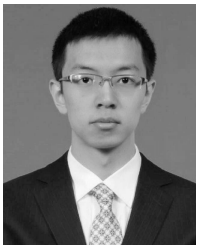
[21] R. Vergin, D. O'Shaughnessy, and A. Farhat, "Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 5, pp. 525–532, Sep. 1999.

[22] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[23] Y. LeCun *et al.*, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 396–404.

[24] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

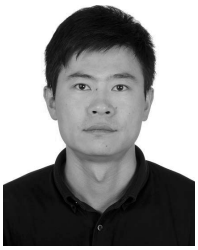
[25] *Tsingmicro Open-Source Speech Corpus*. Accessed: Mar. 2019. [Online]. Available: http://github.com/liguoqianga/tsingmicro.git
[26] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NASA STI/Recon Tech. Rep. N, Feb. 1993, vol. 93.
[27] R. G. Leonard and G. Doddington, *Tidigits Speech Corpus*. Dallas, TX, USA: Texas Instruments, 1993.
[28] D. Povey *et al.*, *The Kaldi Speech Recognition Toolkit*. Martigny, Switzerland: Idiap, 2012.
[29] *Wakeup and Command Recognizing by Thinker-S: An Ultra-Low Power Speech Recognition Processor*. Accessed: Aug. 2018. [Online]. Available: https://youtu.be/6mA2egw8XUk
[30] M. Shah *et al.*, "A fixed-point neural network architecture for speech applications on resource constrained hardware," *J. Signal Process. Syst.*, vol. 90, pp. 727–741, May 2016.



Shixuan Zheng received the B.S. degree from Tsinghua University, Beijing, China, in 2016, where he is currently pursuing the Ph.D. degree with the Institute of Microelectronics. His current research interests include reconfigurable computing architecture and VLSI design.

Peng Ouyang received the B.S. degree in electronic and information technology from Central South University, Changsha, Hunan, China, in 2008, and the Ph.D. degree in electronic science and technology from Tsinghua University in 2014. He held a postdoctoral position with the School of Information, Tsinghua University. He is currently the Chief Technology Officer (CTO) of TsingMicro Intelligent Technology Co. Ltd. His research interests include the embedded deep learning, neuron computing, and reconfigurable computing.

Dandan Song received the B.S. degree in automation from Harbin Engineering University, Harbin, China, in 2014, and the M.S. degree from the Institute of Microelectronics, Tsinghua University, in 2018. Her research interests include speech recognition and machine learning.



Xiudong Li received the B.S. degree in electronic information science and technology from Henan University in 2007 and the M.S. degree in microelectronics from the Harbin Institute of Technology, Shenzhen, China, in 2010. He was with NVIDIA, Shanghai, as an ASIC Physical Design Engineer. He is currently the Vice President of TsingMicro Intelligent Technology Co. Ltd. His research interests include the design and implementation of deep learning accelerator processor, mobile computing, and wireless communication.

Leibo Liu received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Institute of Microelectronics, Tsinghua University, in 2004. He currently serves as a Professor with the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing, and VLSI DSP.

Shaojun Wei was born in Beijing, China, in 1958. He received the Ph.D. degree from the Faculte Polytechnique de Mons, Belgium, in 1991. He became a Professor with the Institute of Microelectronics, Tsinghua University, in 1995. His main research interests include VLSI SoC design, EDA methodology, and communication ASIC design. He is a Senior Member of the Chinese Institute of Electronics (CIE).

Shouyi Yin received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2000, 2002, and 2005, respectively. He was with Imperial College London, London, U.K., as a Research Associate. He is currently an Associate Professor with the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing, and SoC design.