IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 2, 2019, accepted December 15, 2019, date of publication December 19, 2019,
date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960948

An Ultra-Low Power Always-On Keyword Spotting Accelerator Using Quantized Convolutional Neural Network and Voltage-Domain Analog Switching Network-Based Approximate Computing

BO LIU[1], (Member, IEEE), ZHEN WANG[2], WENTAO ZHU[1], YUHAO SUN[1], ZEYU SHEN[1], LEPENG HUANG[1], YAN LI[1], YU GONG[1], AND WEI GE[1]

[1]National ASIC System Engineering Technology Research Center, Southeast University, Nanjing 210096, China
[2]Nanjing Prochip Electronic Technology Company Ltd., Nanjing 210001, China

Corresponding author: Bo Liu (liubo_cnasic@seu.edu.cn)

This work was supported in part by the National Science and Technology Major Project under Grant 2018ZX01031101-005, and in part by the National Natural Science Foundation of China under Grant 61404028 and Grant 61574033.

ABSTRACT An ultra-low power always-on keyword spotting (KWS) accelerator is implemented in 22nm CMOS technology, which is based on an optimized convolutional neural network (CNN). To reduce the power consumption while maintaining the system recognition accuracy, we first perform a bit-width quantization method on the proposed CNN to reduce the data/weight bit width required by the hardware computing unit without reducing the recognition accuracy. Then, we propose an approximate computing architecture for the quantized CNN using voltage-domain analog switching network based multiplication and addition unit. Implementation results show that this accelerator can support 10 keywords real time recognition under different noise types and SNRs, while the power consumption can be significantly reduced to 52μW.

INDEX TERMS Keyword spotting, approximate computing, bit-width quantization.

I. INTRODUCTION

The keyword spotting (KWS) system is a very widely used always-on speech interface which is becoming prevailing in human-machine interaction, especially for wearable devices, the Internet of Things, etc. Requirements of ultra-low power and real-time processing are critical for those battery-powered devices. In the past decades, deep neural networks (DNN) have been shown to outperform traditional models (i.e., Hidden Markov models and Gaussian mixture models) on a variety of speech recognition benchmarks by a large margin, but its massive parameters and computation produce too much power consumption. To overcome the challenge, many DNN accelerators for ultra-low power speech recognition have been proposed in recent years. Shah M., et al. proposed an energy-efficient DNN accelerator for KWS implemented under 40nm TSMC technology [1]. This work can support 10 keywords detection and the power consumption is 11.12 mW. Price M., et al. presented an ultra-low power speech recognizer for both KWS and complex speech recognition tasks, where the adopted DNN is made up of three fully-connected (FC) layers and the bit width of data and weight are both 16 bits. This work can reduce the power to 7.78 mW @40MHz with WER of 8.78% under TSMC 65nm low-power logic process [2]. Bang S., et al. proposed a DNN accelerator which can support voice wake-up function (one keyword recognition) with power consumption of 321 μW [3]. In Giraldo's work [4], they proposed an optimized DNN accelerator for near-microphone KWS. This work is implemented in 65nm

The associate editor coordinating the review of this manuscript and approving it for publication was Gian Domenico Licciardo.

logic process and can support 10 keywords recognition with power consumption of 18.3 μW. Yin S., et al. proposed a CGRA named Thinker, which can support variant bit widths computing of DNNs, and achieved 1.27 TOPS/W in energy efficiency [5]. In Yin's work [6], they first presented an optimized Binary Neural Network (BNN) for speech recognition, where the bit width of data and weight are both 1 bit. In the computation of this BNN, 99% of operations are additions, and the multiplication operations are almost eliminated. To further reduce the power consumption, an ultra-low power DNN accelerator with approximate addition units is proposed to process the calculation of each layer in the BNN. For low background noise (SNR $\geq$ 5dB), this work can support KWS with ultra-low power consumption of 141 μW. However, limited by the low recognition accuracy of BNN, this work can only support one keyword recognition with low background noise. In practical applications, changes of background noise, and even small changes in the distance between the speaker and the microphone, can cause the SNR of input speech change dynamically. Therefore, the robustness of the KWS system under various background noise is a very important evaluation criterion [7].

In this paper, we propose an ultra-low power KWS accelerator based on an optimized CNN with quantized data/weight bit width, which is trained through the Google's Speech Commands database deployed with different types of noise and SNRs. The proposed KWS system can support 10 keywords recognition under different noise types and SNRs. To accelerate the CNN and make it energy efficient, we first perform a bit-width quantization method on the proposed CNN, in order to reduce the data/weight bit width required by the hardware computing unit without reducing the recognition accuracy. Secondly, we propose an approximate computing architecture for the quantized CNN using voltage-domain analog switching network based multiplication and addition units. Implementation results show that this accelerator can support 10 keywords (''yes'', ''no'', ''up'', ''down'', ''left'', ''right'', ''on'', ''off'', ''stop'', ''go'', along with ''silence'' and ''unknown'') real time recognition under different noise types (babble, white, pink, etc.) and SNRs ($-$5dB, 0dB, 5dB, 10dB, etc.), while the power consumption can be significantly reduced from 583μW to 52μW. Compared to the state-of-the-art KWS architectures, our work can achieve high energy efficiency (52μW for low power consumption), while maintaining high system capability (10 keywords for KWS) and high system adaptability (SNR $\geq -$5dB for supporting high background noise).

The rest part of this paper is organized as follows. Some related preliminary works are briefly discussed in section II. Section III describes the KWS prototype system and the optimized CNN with bit width quantization. In section IV, we propose the energy-efficient approximate computing approach for the CNN, including the voltage-domain analog switching network based multiplication and addition units. Finally, implementation results are analyzed in section V and the paper is concluded in section VI.

II. PRELIMINARIES

A. NETWORK OPTIMIZATION APPROACHES FOR LOW POWER KWS SYSTEM

For low-power speech recognition systems, the adopted DNN for feature classification should be firstly optimized to reduce the power consumption of data access and computation. The conventional DNN optimization methods are pruning, encoding and quantization, which are discussed in work [8]–[10]. In our previous work [11] and [12], we proposed several compression methods with hybrid bit-width weights scheme, which can save the memory storage of the typical DNN networks, LeNet, AlexNet and EESEN by 7x$\sim$8x. However, for KWS systems, where the adopted DNNs are typically compact networks customized for specific scenarios, these conventional network compression approaches with pruning and encoding, are likely to cause great accuracy loss. In Yin's work [6], they proposed a BNN for KWS, where the bit width of data and weight are both 1 bit. Compared to typical DNNs with 16 bit data/weight bit width, this BNN can significantly reduce the data/weight memory size and the load/store power consumption. However, this BNN can only support one keyword recognition with low background noise and is too simple for complex speech recognition applications.

In our previous work [13], we have proposed a Binary Weight Network (BWN) for KWS, where the reconfigurable data bit width is 4/8/16 bits, and the weight bit width is 1 bit. Compared to BNN, this BWN network can support 10 keywords recognition under high background noise. However, to support high recognition accuracy, we had to add a lot of network layers and filters to the BWN network. In our previous work [13], the BWN consists of 6 convolution layers and 3 fully-connected layers. For these 6 convolution layers, two of the convolution layers require up to 64 convolution kernels, and two convolution layers require 32 convolution kernels. Therefore, for the BWN, the hardware need to process more calculations, which in turn causes extra power consumption. In summary, for low-power speech recognition systems, there are three advantages to quantize the data and weight bit width of the DNNs: firstly, it can effectively reduce the memory size and the data/weight access power consumption [14]; secondly, the reduced data/weight bit width can also effectively reduce the hardware resources and power consumption of the computing units [15]; thirdly, for the voltage-domain analog computing circuit, the analog noise mismatches can also be reduced. For example, 6-bit data can be encoded within 64 (2^6) voltage values, while 16-bit data requires 65536 (2^{16}) voltage values for encoding. Therefore, the DNN accelerator using voltage-domain analog computing with 6-bit data encoding can achieve much more accuracy than that with 16-bit data encoding.

In this work, we propose a bit-width quantization method to reduce the data/weight bit width without reducing the recognition accuracy. With the reduced data/weight bit width, the hardware resources used to implement the CNN will be also significantly reduced. This method can quantize the CNN data and weight bit width bit-by-bit respectively, so that

the optimal data and weight bit width can be obtained within a limited precision loss for specific application scenarios. The optimized CNN consists of only 3 convolution layers and 2 fully-connected layers, and the data/weight bit width can be quantized to 8/7 bits respectively, while the customized CNN can support 10 keywords recognition under very low background noise (SNR ≥ -5dB). The number of convolution kernels for each layer is 32, 24 and 12. Therefore, compared to the BWN proposed in our previous work [13], the optimized CNN for KWS requires much less calculations, and can be much more energy efficient.

B. ENERGY EFFICIENT APPROXIMATE COMPUTING FOR CUSTOMIZED DNNS

In a typical DNN, the operation numbers of additions and multiplications are almost the equal, however the power consumption of multiplications can account for 96% of all [16]. Thus, a convincing idea to reduce power consumption for processing DNNs is to improve the energy efficiency of multiplication operations. In our previous work [17], we have tried to replace most multiplication operations with addition operations in the convolution layers. This approach can significantly reduce the energy consumption of multiplication operations in convolution layers for image recognition applications with low accuracy requirements. However, for speech recognition applications, especially for KWS with high background noise, this approach is not suitable and may cause a great recognition accuracy loss. Despite of their high accuracy, standard Wallace-Tree based multiplication units have problems in reducing area and energy consumption. Thus approximate multiplication units are required to be adopted in DNN processing because they can significantly improve energy efficient with little cost in accuracy loss. In our previous work [18] and [11], we have proposed two digital approximate multiplication unit architectures to reduce the DNN computing power consumption. These two approximate multiplication units are customized for DNNs based on the iterative logarithmic multiplication principle [19]. Comparison results show that these approximate multiplication units can reduce the power consumption by about 50% with negligible loss of recognition accuracy.

In this work, we propose a voltage-domain based analog multiplication architecture to further reduce the power consumption of the DNN processing. To the best of our knowledge, this is the first voltage-domain approximate computing architecture customized for low power KWS system. Compared to the digital approximate multiplication architectures, this work can significantly improve the energy efficiency of the DNN with low data/weight bit width (8/7 bits respectively), and reduce the power consumption from 583μW to 52μW.

III. TOP ARCHITECTURE OF KWS SYSTEM

A. SYSTEM ARCHITECTURE OVERVIEW

The KWS process adopted in our work mainly consists of two parts: the input speech feature extraction based on MFCC and the keywords classification based on CNN. The feature extraction module is used for extracting the feature values of the input speech. The output of feature extraction module is 26 Mel-scale Frequency Cepstral Coefficients (MFCC). The speech classification module classifies the 26 MFCC output by the feature extraction and determines which keyword it is (or an unknown word). The feature extraction mainly includes the following approaches: MFCC, linear prediction coding coefficient (LPCC) [20], perceptual linear production (PLP) [21] and rasta-plp [22]. In Veton's work [23], the advantages and disadvantages of these approaches (MFCC, LPCC, PLP, rasta-plp and other digital feature extraction approaches) are evaluated by experimental comparative analysis. Experimental results and comparisons show that MFCC is a good choice when the background noise changes greatly or the SNR is low, because of its high robustness and low computational complexity.

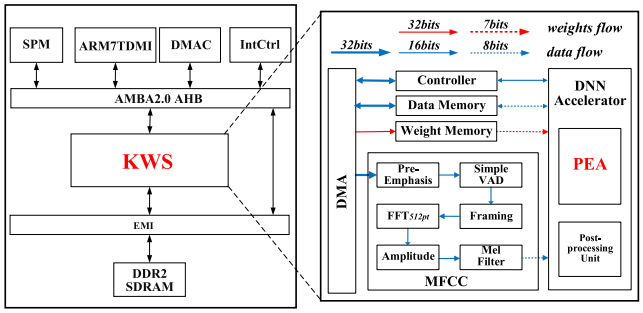


FIGURE 1. Top Architecture of the KWS Prototype System.

In this work, we use a customized MFCC as the feature extraction module. The feature extraction module consists of a Pre-emphasis unit, an energy-based simple Voice Activity Detector (VAD) unit, a framing unit, a 512-point FFT unit, a 16-stage pipeline CORDIC based Amplitude unit and a Mel Filtering unit. The top architecture of the prototype KWS system with the DNN accelerator integrated is as shown in Figure 1. The top-level architecture consists of a system controller implemented with ARM7TDMI, a KWS processor, an 8Kbytes SRAM as system memory and several assistant modules for system scheduling. All modules are AMBA2.0-AHB-compatible and connected to a 32-bit AHB bus module, used as the system bus. The KWS processor consists of a MFCC module, a DNN accelerator, the controller, and the data/weight/configure memory, which are 14/26/4Kbytes SRAMs, respectively. The MFCC module is used to process the feature extraction of the input speech, which consists of a Pre-Emphasis module, a Framing module, a Mel filter, and an Amplitude module. The DNN accelerator can be reconfigured to process different layers of the CNN for the keywords classification. The input speech signal is sampled at 16KHz, and both modules in Figure 1 operate on frames of 40ms with 20ms step size.

This paper also trains a CNN for KWS. As shown in Figure 2, the CNN is composed by three convolution (CONV) layers and two fully-connected (FC) layers,

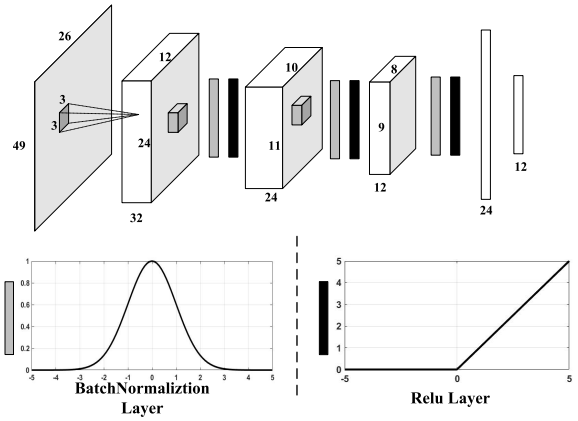


FIGURE 2. CNN Topology for KWS System.

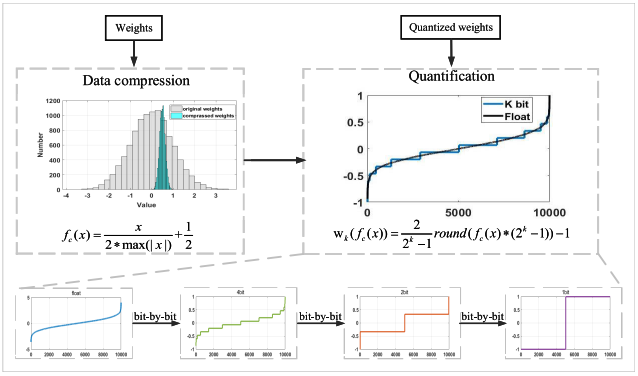


FIGURE 3. CNN Training Process with Data/Weight Bit width Quantization.

several activation (ACT) layers and batch normalization (BN) layers. The convolution kernel size of each CONV layer is 3×3, while the number of convolution kernels is 32, 24 and 12, respectively. The strides (X,Y) (X/Y represents the convolution strides of the speech feature in the frequency/time domain, respectively) are as follows: (2, 2) for CONV layer 1; (1, 2) for CONV layer 2; (1, 1) for CONV layer 3. Each CONV layer is followed by an ACT/BN Layer. The layers used in our CNN are denoted as follows: **CONV layer:** the input filter is a $3 \times 3 \times 32$ three-dimensional matrix. The value of each output neuron is $y = \sum_{j=1}^{32} \sum_{i=1}^{24} \omega_{ji} \cdot x_{ji} + b_j$. **FC layer:** the input multi-dimensional matrix graph is expanded into one-dimensional feature vectors by row or column, then calculated with a matrix multiplication followed by a bias offset to get the value of each output neuron. The formula is:$y_j = \sum_{i=1}^{n} x_i \cdot \omega_{ji} + b_j$. **BN Layer:** this operation is to reduce the problem of slow convergence speed or ''gradient explosion'' in training. The formula is:$y = \gamma \frac{x-\mu}{\sqrt{\varepsilon+\sigma^2}} + \beta$. Four parameters are represented respectively: meaning, μ; variance, σ; scale, γ; offset, β. **ACT Layer:** we use ReLU as the activation function of output neurons in each BN layer. The formula is: $y = max(0, x)$.

B. BIT-WIDTH QUANTIZATION APPROACH FOR CNN BASED KWS SYSTEM

The quantization of weights and activation values is very important for hardware implementation. Traditionally, the trained weights and activation values are mostly fixed to 8bit/16bit, but this data compression method will inevitably lead to the decrease of recognition accuracy. Based on the principle of XNOR-Net quantization framework where the bit width of data and weight are both 1 bit [24], we present the CNN training and quantization methods to quantize the weight and data bit width bit-by-bit, while avoiding recognition accuracy loss. The proposed CNN parameter quantization method is as follows:

$$quantize_k(x_i) = \frac{1}{2^k - 1} round(x_i * (2^k - 1)) \tag{1}$$

$$f(x) = \frac{tanh(x)}{2 * max(|tanh(x)|)} + \frac{1}{2} \tag{2}$$

$$w_q = 2 * quantize_k(f(w_i^n)) - 1 \tag{3}$$

$$x_q = 2 * quantize_k(f(x_i^n)) - 1 \tag{4}$$

where w_i and x_i are the *i-th* layer weight and activation value parameters, k is the data bit width taken, $quantize_k(.)$ and $f(.)$ represent the quantization function and compression function, and w_q and x_q are the corresponding quantization results. Therefore, for x_i, w_i of any layer in network, there is a quantized output real value (b_i is the original floating point offset):

$$z_q = x_q * w_q + b_i \tag{5}$$

$$z_q = \left\{2 * quantize_k(f(x_i^n)) - 1\right\} * \left\{2 * quantize_k(f(w_i^n)) - 1\right\} + b_i \tag{6}$$

Figure 3 shows the quantization method of the CNN adopted in this paper. At k bit width ($k > 1$), both the input layer and the BN layer will be quantized simultaneously. In fact, since the BN layer contains data compression processing, the activation value quantification of the *tanh* function can be discarded, and thus the compression function $f_c(.)$ can be optimized as follows:

$$f_c(.) = \frac{w}{2 * max(|w|)} + \frac{1}{2} \tag{7}$$

Throughout the quantification process, the input weights are first compressed to the range of 0 to 1. The compressed data is subjected to quantization process of the equations (1) and (3). The weights are transformed to non-destructive fixed-point numbers between $[-1, 1]$. In order to enable the quantized weights to better approach the ideal value during the training process, the proposed quantization method can be adopted with a bit-by-bit mode. The first time of training does not directly use low bit width quantization, but instead chooses high bit width quantization. The high bit width weights are saved for re-training, and the quantization bit width is reduced bit-by-bit during next training steps. For example, the quantization bit width can be pre-quantized from 8 bits, 4 bits to 2 bits and then finally quantized to

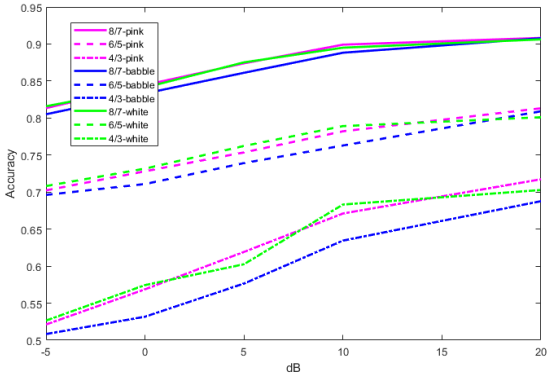


FIGURE 4. KWS Accuracy with Different Date/Weight Width.

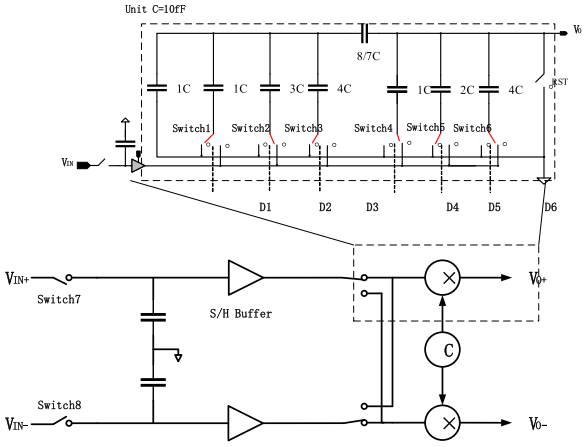


FIGURE 5. Voltage-domain Analog Switching Network based Multiplication Unit.

1 bit. In this way, the most advantageous point of the network training can be quickly found, and the bit-by-bit quantization can be performed at the most advantageous point, which can improve both the accuracy of the training and the reliability of the quantized weights.

The advantage of this method is that it can quantize the CNN data and weight bit width bit-by-bit respectively, so that the optimal data and weight bit width can be obtained within a limited precision loss for a specific application scenario. With the proposed quantization method, we can use the 8/7 bits for data/weight bit width in the proposed KWS system, while maintaining the system recognition accuracy. Figure 4 shows the KWS recognition accuracy with different data/weight bit width combinations under different background noises. The comparison results show that compared with the data/weight bit width of 6/5 bits and 4/3 bits, we choose the data/weight bit width of 8/7 bits, which can make the KWS recognition accuracy not significantly decrease. Because in the analog multiplication operation based on voltage-domain signal, the input digital data needs to be converted into the corresponding analog voltage signal first. Besides, the bit width required for the input signal directly determines the precision of the analog multiplication computing. Therefore, by reducing the data/weight bit width to 8/7 bits, we can maintain the computational accuracy of the voltage-domain analog switching network based multiplication which is proposed in the next section.

IV. VOLTAGE-DOMAIN ANALOG SWITCHING NETWORK BASED APPROXIMATE COMPUTING

CNN mainly consists of CONV layers, FC layers, ACT layers and BN layers. All of these network layers require a lot of multiplication and addition. However, the traditional standard multipliers and adders have high latency and high power consumption, which cannot meet the requirements of low power consumption and high energy efficiency in the design of CNN accelerator. However, DNNs have been proven to be naturally fault-tolerant, and the calculation accuracy requirements for various application scenarios, such as the KWS systems, are also in large variations [25]. Therefore, we can use approximate computing units with reduced power consumption to replace the traditional standard computing units adopted in DNNs. In this section, we propose a voltage-domain analog switching network based approximate computing architecture to process the CNN efficiently.

A. DESIGN OF VOLTAGE-DOMAIN ANALOG SWITCHING NETWORK BASED APPROXIMATE COMPUTING

As shown in Figure 5, the input voltage is passed through the Sample/Hold (S/H) Buffer, and the weight value in CONV/FC layers (or the coefficient factor value in ACT/BN layers) is represented by a 7-bit coefficient ($D_7D_6D_5D_4D_3D_2D_1$), where D_7 is the sign bit. This 7-bit coefficient is used to control the switching signal in the circuit to obtain the output voltage. The calculation process is as shown in Equation (8).

$$V_O = V_{IN}\frac{2^5D_6+2^4D_5+2^3D_4+2^2D_3+2^1D_2+2^0D_1}{2^6} \tag{8}$$

When the input voltage of the adaptive analog multiplication calculation array enters the analog multiplication calculation unit, the calculation unit control module combines the convolution kernel size and the weight data to configure a 1-bit control signal and a 6-bit multiplication coefficient value. The 1-bit control signal controls the operating mode of each analog multiplication unit (the forward process mode, ot the reverse process mode). The 6-bit multiplication coefficient value controls the switches 1 to 6 to further adjust the value of the coefficient. After selecting the operating mode, the input voltage is stabilized by the sampling and holding buffer circuit. After the input voltage is stabilized, it is used as the input voltage of the six parallel switch branches, and each switch has an independent branch. The switch on each branch is connected in series with a capacitor corresponding to the value of the multiplication factor. If the corresponding bit is 1, the switch is closed, and the corresponding capacitor is charged; if the corresponding bit is 0, the switch is open, and the corresponding capacitor is discharged.

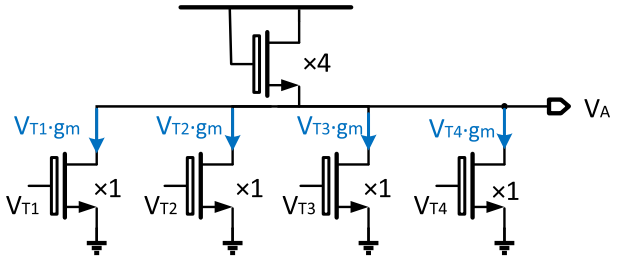


FIGURE 6. Voltage-domain Analog Switching Network based Addition Unit.

The analog multiplication unit uses a discrete time switched capacitor circuit. Adjustable high-order narrow-bandwidth programmable filter is based on the switching circuit. The output voltage of the analog multiplication unit can be obtained by controlling the closing of the six switches by the digital circuit and superimposing the branch voltage generated when the charging capacitor is charged on the switching circuit. For example, when the input data is 51, the digital signal is converted to an analog voltage value $V_{in} = 0.804$V after passing through the DAC module. When the working mode control signal $D_7 = 0$, the working mode is forward working. At the same time, if $D_6D_5D_4D_3D_2D_1 = 010101$, the equivalent capacitance between input and output is 126C/43, the grounding capacitance of output is 53C/9, the hardware coefficient is 0.332, and the corresponding equation coefficient is 0.328. The accurate output voltage is 0.267V, and the output voltage calculated by equation (8) is 0.264V. After passing through the ADC module, the digital output will be 17, which is the same as the result calculated by equation (8).

As shown in Figure 6, to explain the implementation of the analog adder in the current field, the filter size of 2×2 is taken as an example: the voltage signal generated by the DAC is extracted ($V_{T1} \sim V_{T4}$) into a current signal through the G_m unit (NMOS). The change in the input voltage causes the current on the branch to change, and the current varying in each branch is summed at the output node. Finally, the total current is converted to an output voltage signal (V_A) by the output impedance. The diode-connected load is four times larger than the input transistor to maintain the output DC voltage. The analog addition is used to add the input voltage variation. When the four coefficients are all 1, the gain of the adder is 1/4. The load resistance can be increased by adding a current source at the load to increase the gain of the addition.

Implemented and evaluated on TSMC 22nm technology, with the threshold voltages of the NMOS and PMOS transistors as 0.36V and −0.48V, the simulation results of voltage-domain multiplication unit with fixed coefficient/input data are shown in Table 1 and Table 2 (at 25°C TT corner). The comparisons of the computing results with the proposed voltage-domain analog switching network based approximate multiplication units and the computing results with standard multiplication units are shown in Figure 7 (for different input voltages, i.e. the input feature data of each CNN layer) and Figure 8 (for different coefficients, i.e. the network weights of each CNN layer), respectively. From experimental results, it can be seen that the computing results of the voltage-domain approximate multiplication units and the standard multiplication units fit well, and the calculation error is within 0.57%. As shown in Figure 9, the variation

TABLE 1. Simulation Results of Voltage-domain Multiplication Unit With Fixed Coefficient.

Coefficient for Weight ($D_6D_5D_4D_3D_2D_1$)	V_{IN}(V)	V_{OUT}(V)
101, 110	1.000	0.717
101, 110	0.800	0.574
101, 110	0.600	0.430
101, 110	0.500	0.358
101, 110	0.400	0.286
101, 110	0.875	0.628
101, 110	0.571	0.410
...	...	...
100, 000	1.000	0.500
100, 000	0.800	0.397
100, 000	0.600	0.299
100, 000	0.500	0.248
100, 000	0.400	0.199
100, 000	0.875	0.437
100, 000	0.571	0.290
...	...	...
011, 010	1.000	0.405
011, 010	0.800	0.324
011, 010	0.600	0.243
011, 010	0.500	0.203
011, 010	0.400	0.162
011, 010	0.875	0.355
011, 010	0.571	0.231

TABLE 2. Simulation Results of Voltage-domain Multiplication Unit With Fixed Input Voltage.

V_{IN}(V) for Input Data	Coefficient for Weight ($D_6D_5D_4D_3D_2D_1$)	V_{OUT}(V)
0.875	100000	0.429
0.875	110000	0.645
0.875	111000	0.756
0.875	111100	0.848
0.875	111110	0.842
0.875	111111	0.854
0.875	101110	0.628
0.875	011010	0.355
0.875	100101	0.506
...	...	...
0.800	100000	0.397
0.800	110000	0.589
0.800	111000	0.691
0.800	111100	0.744
0.800	111110	0.769
0.800	111111	0.780
0.800	101110	0.574
0.800	011010	0.324
0.800	100101	0.462
...	...	...
0.571	100000	0.290
0.571	110000	0.420
0.571	111000	0.494
0.571	111100	0.530
0.571	111110	0.549
0.571	111111	0.557
0.571	101110	0.410
0.571	011010	0.231
0.571	100101	0.330

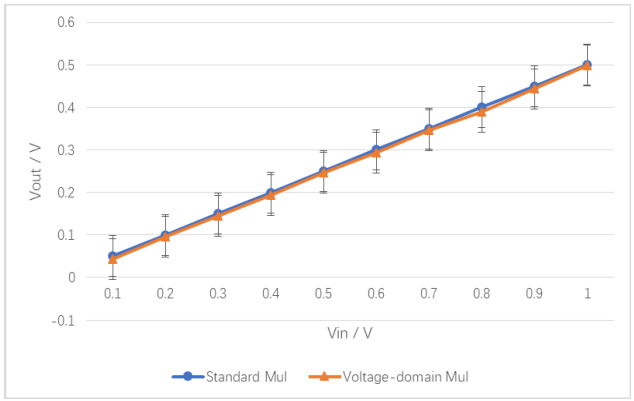


FIGURE 7. Comparisons of Voltage-domain Approximate Multiplication and Standard Multiplication (With Fixed Coefficients).

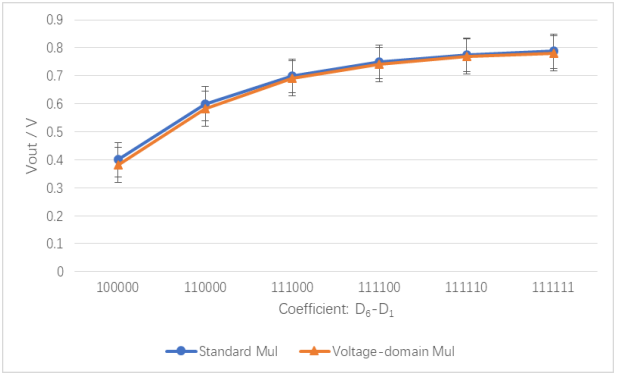


FIGURE 8. Comparisons of Voltage-domain Approximate Multiplication and Standard Multiplication (With Fixed Input).

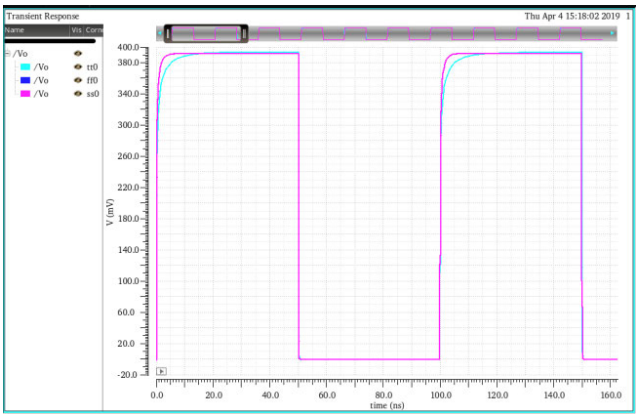


FIGURE 9. Effect of Different Process Corners on Variations of Voltage-domain Multiplication Outputs.

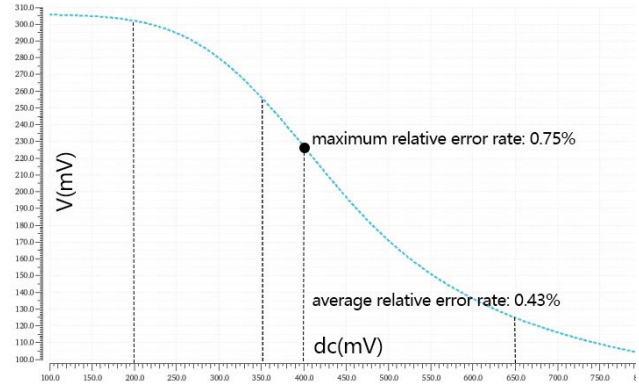


FIGURE 10. Relationship Curve for Voltage-domain Addition Unit and Standard Addition Unit.

of process corners has little effects on the outputs of the proposed voltage-domain multiplication units. There is a non-linear corresponding relationship between the output voltage of the proposed voltage-domain analog switching network based addition unit and the computing result of the standard addition unit. The relationship curve for voltage-domain addition unit and standard addition unit is shown in Figure 10. The output voltage is divided into 2^6 segments, and the voltage value in each segment corresponds to the computing result of the standard addition unit. Since any voltage value in one segment corresponds to the same computing result, there may be some mismatches for the proposed voltage-domain analog switching network based addition unit. As shown in Figure 10, the maximum relative error rate, which is 0.75%, is at the maximum slope point of the curve where the output voltage is 0.4V. Similarly, there are mismatches in other segments, but all of them are much smaller than the maximum value of 0.75%, and the average relative error rate for all segments is only 0.43%.

B. CUSTOMIZED DAC/ADC FOR PROPOSED APPROXIMATE COMPUTING UNITS

The digital-to-analog converter (DAC) circuit designed in this work is as shown in Figure 11. The input data (X_{IN}) is fed into the column DAC, which precharges the output signal (GRBL) terminal to analog voltage (V_A). The DAC consists of a cascaded PMOS constant current source composed of three PMOS transistors and one NMOS transistor. The GRBL terminal charging current duration is t_{ON}, and the current duration is proportional to the input value X_{IN}. In order to maintain a linear correlation between t_{ON} with X_{IN}, there should be only one ON pulse in the circuit, avoiding multiple charging phases for each input. Therefore, we use the following design methods: as shown in Figure 12(a), when the input data is 6 bits, the three upper MSBs of the input data X_{IN} are used to select the first half of the charging pulse width, while the three lower LSBs are to determine the second half of the charging pulse width. An 8:1 multiplexer with 8 timing signals is shared to reduce the area overhead and signal routing. This design method can generate an ON pulse for each X_{IN}. The 8:1 multiplexer shown in Figure 12(b) determines the pulse width of the output t_{ON} by the input value. The lower 3 bits of the 6-bit input value indicate the pulse width of 0-7 t_{ON}. According to the values of the upper 3 MSBs and the lower 3 LSBs, the corresponding output charging pulse is obtained, and then transmitted to the DAC, thereby realizing the conversion from the digital signal to the analog voltage signal.

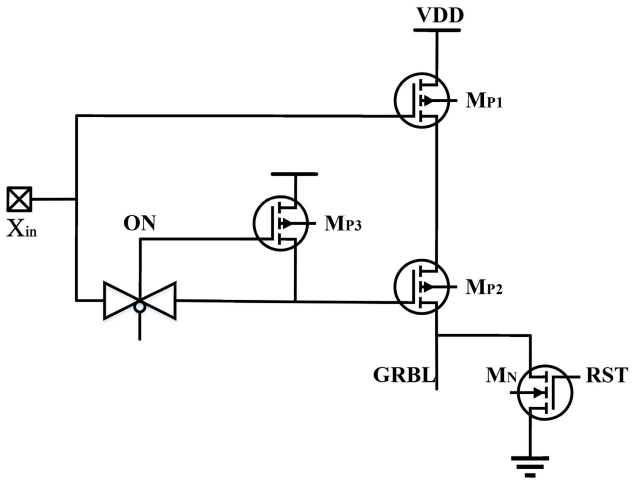


FIGURE 11. Customized DAC for Voltage-domain Approximate Computing.

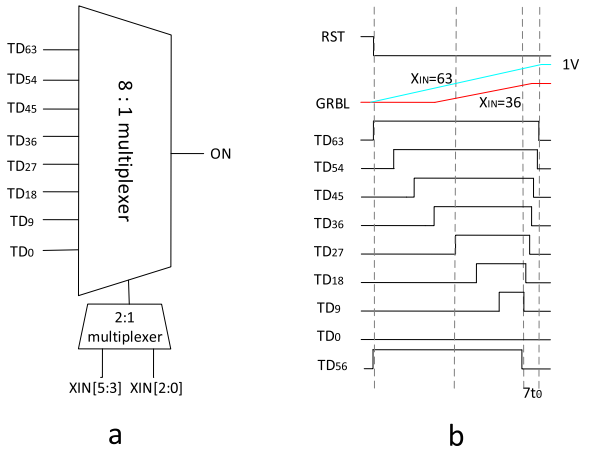


FIGURE 12. Output Pulse Width Control for Voltage-domain Computing.

TABLE 3. Encoding Method for the DAC.

Digital Inputs	Encoding ($X_6X_5X_4X_3X_2X_1$)	Analog Voltage(V)
63	111,111	1
62	111,110	0.982
61	111,101	0.964
...	...	...
56	111,000	0.875
55	110,111	0.875
...	...	...
2	000,010	0.036
1	000,001	0.0180
0	000,000	0

The Encoding Methods for DAC and Coefficients are shown in Table 3 and Table 4, respectively. The decimal digit input is converted to binary code $X_6X_5X_4X_3X_2X_1$, then it is divided into ($X_6X_5X_4$, $X_3X_2X_1$) and converted to decimal (Y_2, Y_1), The output analog voltage V can be calculated as follows:

$$V = \frac{7Y_2 + Y_1}{56} \tag{9}$$

TABLE 4. Encoding Method for the Coefficients.

($D_6D_5D_4D_3D_2D_1$)	Value of coefficient with equation (8)	Value of coefficient with proposed circuits
000,000	0	0
000,001	0.0156(1/64)	0.014
000,010	0.0313(1/32)	0.0468
000,100	0.0625	0.062
...	...	...
010,101	0.328(21/64)	0.332
...	...	...
011,111	0.48(31/64)	0.500
100,000	0.5(1/2)	0.499
...	...	...
111,110	0.984(63/64)	0.9375
111,111	1	1

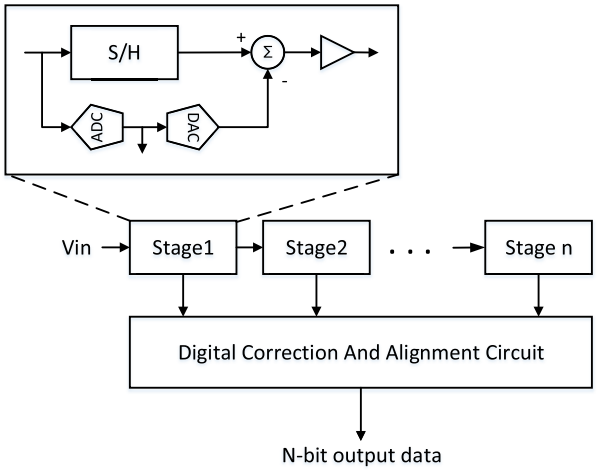


FIGURE 13. ADC for Voltage-domain Approximate Computing.

As shown in Table 3, when the input data is 56 and 55, the output voltage is the same. This is because when the high-order digital Y_2 is reduced, the low-control number Y_1 is correspondingly increased. Since the relative error in this case is very small, it does not affect the recognition accuracy of the neural network for KWS.

In this work, we present a pipelined analog-to-digital converter (ADC) with low-power and small-area overhead. As shown in Figure 13, it mainly consists of multiple cascaded circuits, each of which includes a sample/hold (S/H) amplifier, a low precision ADC, a DAC and a summing circuit. The input analog signal is converted to a 3-bit digital value by a 3-bit precision ADC. The digital value is the upper 3 bits of the output data, and it is then converted into an analog signal by the DAC. The S/H amplifier samples the input analog signal and performs addition or subtraction operations. The result is amplified and sent to the next stage circuit for processing, thereby obtaining the lower 3 bits of the output data.

The Voltage-domain analog switching network based multiplication/addition unit and the DAC/ADC of the DNN accelerator are customized with Cadence Virtuoso Tool. The layouts of voltage-domain analog switching network based

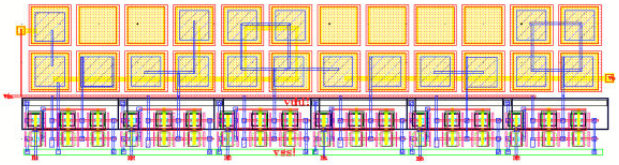


FIGURE 14. Layout of Voltage-domain Approximate Multiplication Unit.

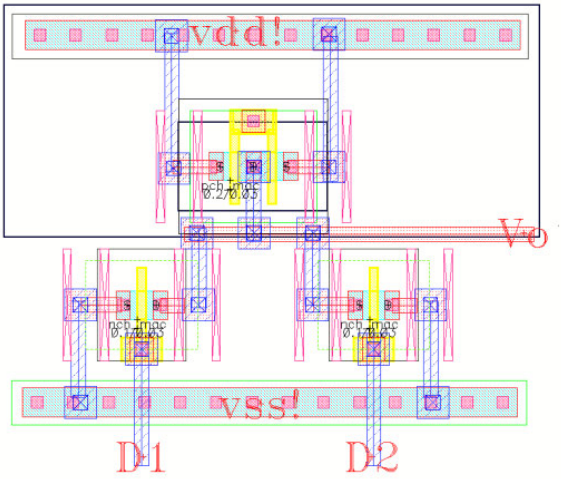


FIGURE 15. Layout of Voltage-domain Approximate Addition Unit.

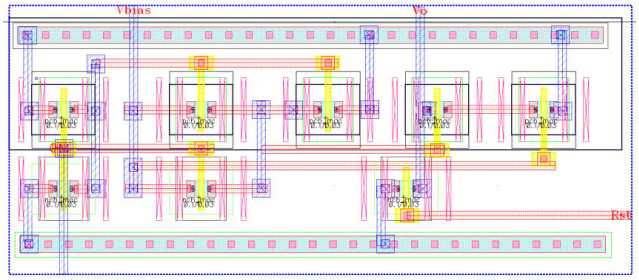


FIGURE 16. Layout of Proposed DAC.

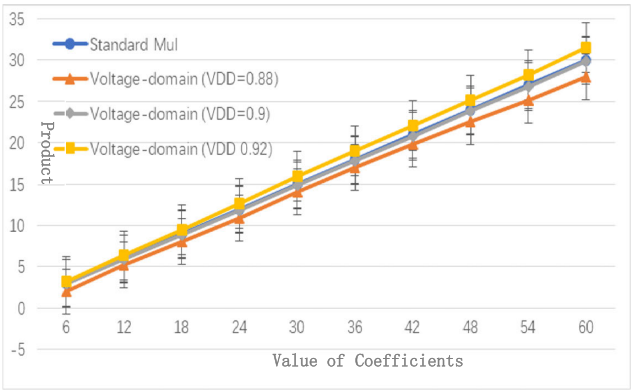


FIGURE 17. Mismatch of Voltage-domain Multiplication Caused by Voltage Fluctuation.

multiplication/additon unit and DAC are shown in Figure 14, Figure 15 and Figure 16, respectively. In this work, the proposed voltage-domain analog multiplication unit and the DAC/ADC unit are all customized with fixed circuit structures and design parameters. These customized units can only support multiplication and addition operations with fixed data/weight bit width, For example, if the bit width of input data is 5 bits, which are smaller than the bit width of the voltage-domain analog multiplication unit, then the missing high-order data bits must be padded with bit '0'. In this case, the computing accuracy of the voltage-domain analog multiplication unit remains the same. If the bit width of input data is greater than the bit width of the proposed voltage-domain analog multiplication unit, the input data cannot be converted to the appropriate voltage value by the customized DAC encoding. Analog circuits, especially the DACs/ADCs, are susceptible to voltage fluctuations (such as voltage fluctuations caused by device temperature variations). In this paper, we analyze the effects of voltage fluctuations by testing the process of input signals through DACs, analog multipliers, and ADCs. When the power supply voltage of these analog circuits is 0.9V, considering the influence on the circuit when the voltage fluctuation range is 2% (in practical applications, the voltage fluctuation is rarely more than 2%), the voltage fluctuation range is 0.88V $\sim$ 0.92V. Figure 17 shows the mismatches of the computing results with voltage-domain multiplication caused by voltage fluctuations. The voltage fluctuation causes the mismatches of the computing results to increase, and the larger the value of the calculation results, the larger the mismatches. However, since the data distribution of DNNs basically obeys to the normal distributions: the density of the data near the smaller value is very large, while the density of the data with a large value is very small. In another word, the data with a large value has less influence on the mismatches of the DNNs. The experimental results show that the average accuracy loss is only 2.68%. Since the CNN is inherently fault-tolerant, the calculation results are mainly used to distinguish the differences between different inputs, so the proposed voltage-domain computing approach can fully meet the computational requirements of the CNN for proposed KWS system.

C. APPROXIMATE PROCESSING ELEMENT ARRAY

As shown in Figure 18, the processing element array (PEA) mainly includes a digital computing unit (DCU), an analog computing unit (ACU), a DAC, an ADC and a global buffer. The input data from MFCC, the weight data and the computing temporary/result data are all stored in the SRAM. When the system controller enables the PEA, the global buffer reads the input data and the weight data from the SRAM accordingly. The DCU is composed of 4 $\times$ 4 Digital Process Elements (DPEs), each of which can perform digital multiplication and addition, and include a FIFO storage array. Input data can be temporarily stored in the FIFO for data reuse. When the DPE receives the input data, it performs a corresponding multiply-and-accumulate operation. The output is transmitted to the right and down DPEs in the next cycle, and the final result is passed to the global buffer. The ACU is composed

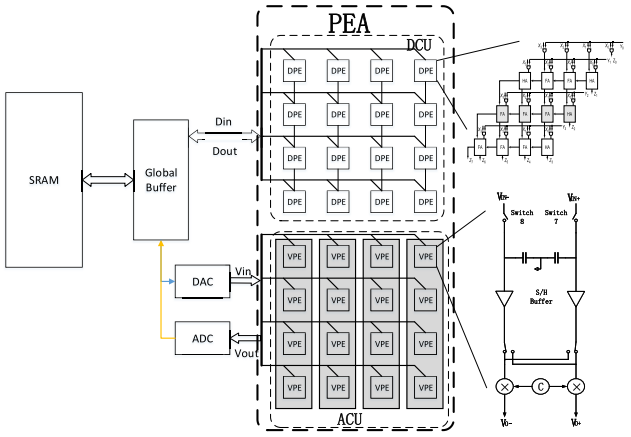


FIGURE 18. PEA with Proposed Voltage-domain Approximate Computing Units.

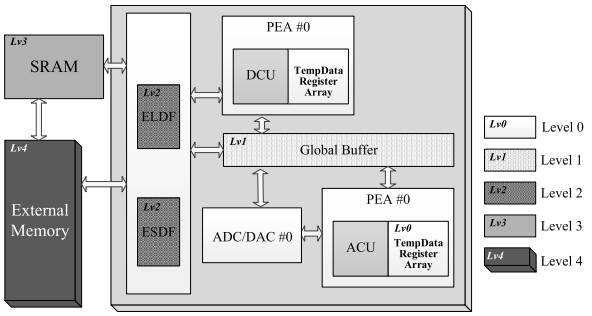


FIGURE 19. Hierarchical Data Storage Structure for Proposed KWS Accelerator.

of 4 × 4 Voltage Process Elements (VPEs) for performing the approximate computing based on voltage-domain analog switching network.

When the PEA is configured to perform the approximate computing, the input data is loaded from the buffer and transferred to the FIFO array of the DAC in ACU. Then the converted voltage from DAC is directly transmitted to the VPE. The weights of CONV/FC layers (or the coefficient factors for ACT/BN layers) are read from the shared memory and stored in the FIFO array of the VPE. Each VPE contains a voltage-domain analog switching network based multiplication/addition unit. The output voltage from VPE is passed to the ADC module to obtain the output digital data, and then the data is transferred to the global buffer. The PEA module can dynamically configure the working mode of the computing unit according to different calculation requirements. Normally, the convolution kernel size is 3 × 3, in which case the PEA will use 3 × 3 computational units DPEs or VPEs in the array to complete the calculation. When the convolution kernel size is 2 × 2, the calculation unit module can be configured to perform four sets of calculations in parallel. When the convolution kernel size is greater than 4 × 4, the PEA module can be configured to complete the calculation in several times.

The processing of convolution neural network is a data-access-sensitive task that requires frequent access to memory and there is a large amount of data interactions during convolution operations. In order to reduce the bandwidth requirements and memory access delay, and improve the stability of the data stream, a suitable data storage structure design is needed. In this work, we use a 5-level hierarchical data storage structure as shown in Figure 19. The Level4 memory is the external main memory which is a DDR SDRAM, and level3 memory is the pre-fetch buffer which is a on-chip SRAM. Level2 memory is used as a data buffer between the DNN accelerator and the external memory. It adopts a FIFO structure and is composed of an External Data Storage FIFO (ESDF) and an External Load Data FIFO (ELDF). The ESDF is the data buffer when the output of the DNN accelerator needs to be transferred to the external memory, and the ELDF is used for the accelerator to read the input data, which is then transmitted by the ELDF to each internal storage structure of the accelerator. Level1 memory is used to store the internal data of the PEA, including the input data, the weights and the output data. Level0 memory is the data register in the PEA. The level0 data register is tightly coupled to the processing units DCU, ACU, and DAC/ADC. There are 4 × 4 register files in each DCU and ACU for the data routing of DPEs and VPEs, respectively. Besides, the DAC and ADC modules each contain a temporary register file.

V. IMPLEMENTATION RESULTS

The prototype system as shown in Figure 1 is implemented and evaluated on TSMC 22nm ULL HVT transistor process technology. The Voltage-domain analog switching network based multiplication/addition unit and the DAC/ADC of the CNN accelerator are customized with Cadence Virtuoso Tool (version: IC6.1.7-64b.78), while the other digital modules are described with Verilog HDL language and synthesized by Synopsys Design Compiler (DC, version: J-2014.09-SP3). The SRAM blocks and other digital modules are functional with the logic supply voltage of 0.55V (with the working frequency of 250KHz). The VPEs in ACU with voltage-domain analog switching network based approximate computing are functional with the logic supply voltage of 0.9V (with the working frequency of 2.5MHz). The die layout of the prototype system is shown in Figure 20. The area of ACU and DCU macro are 0.48×0.29 mm^2, 0.49×0.27 mm^2 (without memory), and the whole prototype accelerator is 0.75 mm^2. For the layout of ACU, the area of PEA is 0.18×0.18 mm^2, which accounts for 20% of the total area of ACU, the DAC/ADC and other rest modules accounts for 80% of the total area of ACU.

To evaluate the power consumption and recognition accuracy with the proposed approximate computing units, a reference design with standard multipliers and adders in DCU module is also implemented. When the DCU module in PEA is enable, the proposed CNN is working on standard computing mode; when the ACU module is enable, the CNN is working on approximate computing mode with proposed voltage-domain analog switching network based multiplication and addition units. The timing and

FIGURE 20. Die Layout of Proposed Prototype KWS System.

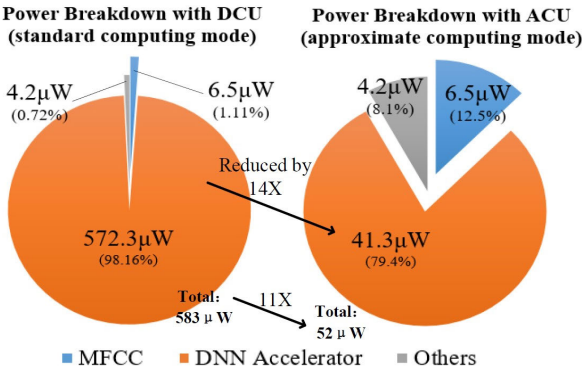


FIGURE 21. Power Breakdown in Standard/Approximate Computing Mode.

power consumption are evaluated with Synopses HSIM (version: K-2015.06) at 25°C TT corner. The power breakdown of the proposed KWS accelerator in standard/approximate computing mode is shown in Figure 21. The power consumption of the KWS prototype system is 52μW in the approximate computing mode (with ACU) and 583μW in the standard computing mode (with DCU). In the approximate computing mode, the power consumption of DNN accelerator with ACU is 41.3μW, of which the power consumption of the VPEs accounts for 18%, and the power consumption of the DAC/ADC and other rest modules accounts for 82%. With the proposed approximate computing, the power consumption of the prototype system and the DNN accelerator is reduced by 11X and 14X, respectively.

We use Google's Speech Commands [26] as our training and evaluating database. The chosen keywords are "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", along with "silence" and "unknown". Table 5 shows the recognition accuracy of proposed CNN based KWS system under different background noise and computing approaches, including: the software simulation based on MATLAB using floating data/weight bit width, the KWS

TABLE 5. Recognition Accuracy Comparisons with Proposed Approximate Computing.

Noise type	SNR	Software Simulation data/weight:float	Standard Computing data/weight:8/7bits	Approximate Computing data/weight:8/7bits
Pink	-5dB	83.69%	81.34%	81.12%
	0dB	86.32%	84.46%	84.19%
	5dB	88.68%	87.39%	87.11%
	10dB	90.31%	89.91%	89.78%
	20dB	91.26%	90.82%	90.34%
Babble	-5dB	82.83%	80.51%	80.19%
	0dB	84.71%	83.32%	82.96%
	5dB	87.17%	86.11%	86.92%
	10dB	89.89%	88.82%	88.51%
	20dB	91.11%	90.78%	90.47%
White	-5dB	83.92%	81.59%	81.21%
	0dB	86.31%	84.12%	83.62%
	5dB	88.58%	87.52%	87.11%
	10dB	90.79%	89.50%	89.13%
	20dB	91.03%	90.62%	90.29%
Clear Speech	∞	91.29%	90.82%	90.51%

prototype system using standard computing units, and the prototype KWS system using proposed voltage-domain analog switching network based approximate computing units. The noise types include white, babble and pink, while the SNR ratios include -5db, 0dB, 5dB, 10dB, 20dB and clear (without background noise). As shown in Table 5, with the proposed CNN data/weight quantization method, we can reduce the data/weight bit width of the CNN from float to 8/7 bits, while the KWS recognition accuracy is reduced by only less than 3%. Compared to the reference design with standard computing units, the proposed approximate computing units can significantly reduce the KWS accelerator power consumption by 11X, while the loss of recognition accuracy is less than 1%.

Comparisons with other state-of-the-art KWS architectures based on DNNs are shown in Table 6. In *Giraldo's work (published in VLSI'19) [4]*, the DNN accelerator is proposed for near-microphone KWS, where the background noise is very low and can be ignored. In *Yin's work (published in VLSI'18) [6]* and our work, the DNNs adopted for KWS contain both FC and CONV layers. The CONV layers can effectively improve the recognition accuracy of KWS under low data/weight bit width. In *Yin's work [6]*, the proposed architecture is customized for a BNN where the bit width of data and weights are both 1 bit. To further reduce the energy consumption of the addition units, a digital approximate addition architecture is also proposed. Benefiting from the BNN network and the approximate addition architecture, the power consumption of *Yin's work [6]* can be reduced to 141μW. However, this work can only support one keyword recognition under low background noise (SNR $\geq$ 5dB). In our work, we use the CNN for KWS system with data/weight quantized as 8/7bits. Compared to *Yin's work [6]*, our work can support 10 keywords recognition under high background noise (SNR ≥ -5dB), while the power consumption can be significantly reduced to 52μW.

For each frame of the speech input, the accelerator proposed in *Giraldo's work (VLSI'19) [4]/ Shah's work (JSPS'18) [1]/ Yin's work (VLSI'18) [6]/ OurWork* needs to process 115,380/ 2,103,255/ 11,074,048/ 2,101,824

TABLE 6. Comparisons with other KWS architectures.

	VLSI'19 [4]	JSPS'18 [1]	VLSI'18 [6]	This work
Technology (nm)	65	40	28	22
Frequency (MHz)	0.25	50	2.5	0.25
Latency (ms)	16	10	25	20
Voltage (V)	0.6	0.6	0.57	0.55
DNN Structure	LSTM+FC	FC	CONV+FC	CONV+FC
Bit Width (Weight)	4/8	6	1	7
Bit Width (Data)	10	16	1	8
Computing Circuits	Standard Computing		Approximate Computing (ADD)	Approximate Computing (MULT/ADD)
Number of Keywords	10	10	1	10
Background Noise	SNR$\approx\infty$ (no noise)	NA	SNR$\geq$ 5dB	SNR$\geq$ -5dB
Power	18.3μW	11.2mW	141μW	52μW
DNN Operations	115,380	2,103,255	11,074,048	2,101,824
Power Consumption per Operation	0.159nW	5.325nW	0.013nW	0.025nW
Normalized Energy Efficiency	1X	0.03X	12.23X	6.36X

multiplication and addition operations, respectively. The power consumption of *Giraldo's work [4]/ Shah's work [1]/ Yin's work [6]/ OurWork* is 18.3μW/ 11.2mW/ 141μW/ 52μW, respectively. Therefore, the power consumption per operation (for each DNN operation) of *Giraldo's work [4]/ Shah's work [1]/ Yin's work [6]/ OurWork* is 0.159nW/ 5.325nW/ 0.013nW/ 0.025nW, respectively. The energy efficiency is 1/(power consumption per operation). With the energy efficiency of accelerator proposed in *Giraldo's work [4]* as the normalization value 1X, the normalized energy efficiency of *Shah's work [1]/ Yin's work [6]/ OurWork* can be calculated, which is 0.03X/ 12.23X/ 6.36X, respectively.

Compared to the DNN used in our work, which consists of 3 CONV layers (32/24/12 kernels of each layer) and 2 FC layers (2,101,824 operations for each input speech frame), the accelerator in *Giraldo's work [4]* requires a much smaller amount of computation (115,380 operations for each input speech frame) and the hardware power consumption is low (18.3μW). However, since the DNN used is very simple, the KWS accelerator proposed in *Giraldo's work [4]* can only work in near microphone cases, where the background noise can be ignored (SNR$\approx\infty$). Compared with *Giraldo's work [4]*, the DNN used in our work has higher robustness and fault tolerance (thus requiring much more computation), and therefore can support high recognition accuracy with very high background noise (even the SNR is $-$5dB). Compared with the DNN used in *Giraldo's work [4]*, the operations of the DNN used in our work is over 18X of the former, however the power consumption of our work is only 2.8X of the former, and the energy efficiency of our work is 6.36X of the former. In *Yin's work [6]*, the DNN consists of 4 CONV layers (64/32/64/32 kernels of each layer) and 2 FC layers. For each input speech frame, the accelerator should process 11,074,048 DNN operations. The energy efficiency of the accelerator in *Yin's work [6]* is 1.92X better than our work. In *Yin's work [6]*, the bit width of data and weights are both only 1 bit, which can greatly reduce the power consumption per operation, and therefore can obtain the highest energy efficiency of all these works in Table 6. However, the DNN with data/weight bit width of 1/1 bit will greatly reduce the DNN robustness and fault tolerance for KWS, and therefore it can support only one key word recognition. Experimental results show that our work can achieve high energy efficiency (52μW for low power consumption), while maintaining high system capability (10 keywords for KWS) and adaptability (SNR ≥ -5dB for supporting high background noise).

VI. CONCLUSION

This paper proposed an energy-efficient DNN accelerator for keyword spotting using convolution neural network (CNN) and approximate computing. To accelerate the CNN and make it energy efficient, we presented a bit width quantization method to reduce the data/weight bit width required for the CNN, and an approximate computing architecture for the quantified CNN based on voltage-domain analog switching network. This approximate computing can contribute a significant decrease in energy consumption compared to those with standard computing units. Implemented under TSMC 22nm CMOS technology, our work can support 10 keywords real-time keywords recognition under different noise types and SNRs with the power consumption of 52μW. Experimental results show that our work can achieve high energy efficiency, while maintaining high system capability and adaptability.

REFERENCES

[1] M. Shah, S. Arunachalam, J. Wang, D. Blaauw, D. Sylvester, H.-S. Kim, J.-S. Seo, and C. Chakrabarti, "A fixed-point neural network architecture for speech applications on resource constrained hardware," *J. Signal Process. Syst.*, vol. 90, no. 5, pp. 727–741, 2018, doi: 10.1007/s11265-016-1202-x.

[2] 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*, San Francisco, CA, USA, Feb. 2017, pp. 244–245, doi: 10.1109/ISSCC.2017.7870352.

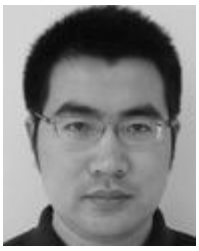
[3] S. Bang, J. Wang, Z. Li, C. Gao, Y. Kim, Q. Dong, Y.-P. Chen, L. Fick, X. Sun, R. Dreslinski, T. Mudge, H. S. Kim, D. Blaauw, and D. Sylvester, "A 288 μW programmable deep-learning processor with 270kb on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 250–251, doi: 10.1109/ISSCC.2017.7870355.

[4] J. S. P. Giraldo, S. Lauwereins, K. Badami, H. Van Hamme, and M. Verhelst "18 μW SoC for near-microphone keyword spotting and speaker verification," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2019, pp. C52–C53, doi: 10.23919/VLSIC.2019.8777994.

[5] S. Yin, P. Ouyang, S. Tang, F. Tu, X. Li, S. Zheng, T. Lu, J. Gu, L. Liu, and S. Wei, "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, Apr. 2018, doi: 10.1109/JSSC.2017.2778281.

[6] S. Yin, P. Ouyang, S. Zheng, D. Song, X. Li, L. Liu, and S. Wei, "A 141 μW, 2.46 PJ/neuron binarized convolutional neural network based self-learning speech recognition processor in 28nm CMOS," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2018, pp. 139–140, doi: 10.1109/VLSIC.2018.8502309.

[7] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP*, May 2013, pp. 7398–7402, doi: 10.1109/ICASSP.2013.6639100.
[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. ICLR*, 2016, pp. 3–7.
[9] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ISCA*, Jun. 2016, pp. 243–254, doi: 10.1109/ISCA.2016.30.
[10] H. Salehinejad and S. Valaee, "Ising-dropout: A regularization method for training and compression of deep neural networks," in *Proc. ICASSP*, May 2019, pp. 3602–3606, doi: 10.1109/ICASSP.2019.8682914.
[11] Z. Wang, M. Xia, B. Liu, X. Ruan, Y. Gong, J. Yang, W. Ge, and J. Yang, "EERA-DNN: An energy-efficient reconfigurable architecture for DNNs with hybrid bit-width and logarithmic multiplier," *IEICE Electron. Express*, vol. 15, no. 8, pp. 1–10, 2018, doi: 10.1587/elex.15.20180212.
[12] B. Liu, X. Ruan, M. Xia, Y. Gong, J. Yang, W. Ge, and J. Yang, "An energy-efficient accelerator for hybrid bit-width DNNs," in *Proc. IEEE Symp. Comput. Intell.*, Nov./Dec. 2018, pp. 1–8, doi: 10.1109/SSCI.2017.8280940.
[13] B. Liu, Z. Wang, H. Fan, J. Yang, B. Liu, W. Zhu, L. Huang, Y. Gong, W. Ge, and L. Shi, "EERA-KWS: A 163 TOPS/W always-on keyword spotting accelerator in 28nm CMOS using binary weight network and precision self-adaptive approximate computing," *IEEE Access*, vol. 7, pp. 82453–82465, 2019, doi: 10.1109/ACCESS.2019.2924340.
[14] P. Yin, S. Zhang, Y. Qi, and J. Xin, "Quantization and training of low bit-width convolutional neural networks for object detection," *J. Comput. Math.*, vol. 37, no. 3, pp. 349–359, 2018, doi: 10.4208/jcm.1803-m2017-0301.
[15] M. Pietras, "Error analysis in the hardware neural networks applications using reduced floating-point numbers representation," in *Proc. AIP Conf. Proc.*, vol. 1648, 2015, pp. 239–255, doi: 10.1063/1.4912881.
[16] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14, doi: 10.1109/ISSCC.2014.6757323.
[17] Y. Gong, B. Liu, W. Ge, and L. Shi, "ARA: Cross-Layer approximate computing framework based reconfigurable architecture for CNNs," *Microelectron. J.*, vol. 87, pp. 33–44, May 2019, doi: 10.1016/j.mejo.2019.03.011.
[18] B. Liu, W. Dong, T. Xu, Y. Gong, W. Ge, J. Yang, and L. Shi, "E-ERA: An energy-efficient reconfigurable architecture for RNNs using dynamically adaptive approximate computing," *IEICE Electron. Express*, vol. 14, no. 15, pp. 1–11, 2017, doi: 10.1587/elex.14.20170637.
[19] Z. Babić, A. Avramović, and P. Bulić, "An iterative logarithmic multiplier," *Microprocess. Microsyst.*, vol. 35, no. 1, pp. 23–33, Feb. 2011, doi: 10.1016/j.micpro.2010.07.001.
[20] H. Gupta and D. Gupta, "LPC and LPCC method of feature extraction in speech recognition system," in *Proc. 6th Int. Conf.-Cloud Syst. Big Data Eng. (Confluence)*, 2016, pp. 498–502, doi: 10.1109/CONFLUENCE.2016.7508171.
[21] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Amer.*, vol. 87, no. 4, pp. 1738–1752, 1990, doi: 10.1121/1.399423.
[22] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "The challenge of inverse-E: The RASTA-PLP method," in *Proc. ACSSC*, Nov. 1991, pp. 800–804, doi: 10.1109/ACSSC.1991.186557.
[23] Z. K. Veton and A. E. Hussien, "Robust speech recognition system using conventional and hybrid features of MFCC, LPCC, PLP, RASTA-PLP and hidden Markov model classifier in noisy conditions," *J. Comput. Chem. Commun.*, vol. 3, no. 6, pp. 1–9, 2015, doi: 10.4236/jcc.2015.36001.
[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542, doi: 10.1007/978-3-319-46493-0_32.
[25] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2017, pp. 1606–1610.
[26] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*. [Online]. Available: https://arxiv.org/abs/1804.03209

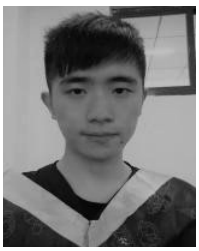


BO LIU (M'19) was born in Taizhou, Jiangsu, China, in 1984. He received the B.S. and Ph.D. degrees in electronic science and engineering from Southeast University, in 2006 and 2013, respectively.

He is currently a Lecturer with the National ASIC System Engineering Research Center, Southeast University. His research interests include chip architecture design, reconfigurable computing, approximate computing, and related VLSI designs. He has received more than ten National Circuit Design Award and holds 30 patents. He has coauthored more than 40 academic articles in the above research fields in the conferences and journals, such as the IEEE TVLSI, the IEEE SiPS, IEEE Access, the IEEE DSP, the IEEE FPT, the IEEE Trustcom, the IEEE ICUWB, the IEEE IPDPS, the IEEE NANOARCH, the IEICE T Inf&Syst, the IEICE ELEX, and *Microelectronics Journal*. His research was supported by the National Natural Science Foundation, the National Science and Technology Major Project, and the National Key Research and Development Program. He serves as the Session Chair of the IEEE NANOARCH 2019.

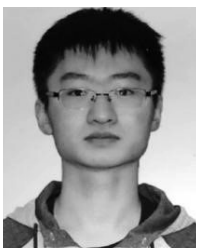
ZHEN WANG received the B.S. degree in radio and the M.S. and Ph.D. degrees in electronic science and engineering from Southeast University, in 2002, 2005, and 2018, respectively.

He is currently working with Nanjing Prochip Electronic technology Company Ltd. His research interests include calculation in memory, low power circuit, and the AI voice circuit.



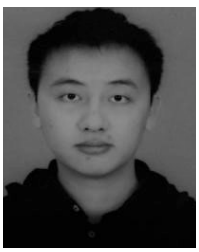
WENTAO ZHU received the B.S. degree in electronic science and technology from the Chongqing University of Technology, Chongqing, China, in 2017. He is currently pursuing the M.S. degree in integrated circuit engineering from Southeast University, Nanjing, China.

His current research interests include speech recognition and low voltage circuits.



YUHAO SUN received the B.S. degree from Soochow University, Suzhou, China, in 2017. He is currently pursuing the M.S. degree with the School of Department of Electronic Engineering, Southeast University, Nanjing, China.

His current research interests include digital application specific integrated circuit design and neural network chip.



ZEYU SHEN received the B.S. degree from Southwest Jiaotong University, Chendu, China, in 2018. He is currently pursuing the M.S. degree with the School of Electronic Engineering, Southeast University, Nanjing, China.

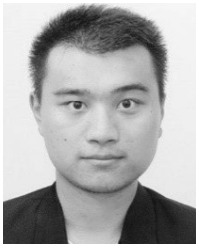
His current research interests include digital IC design and neural network chip.

LEPENG HUANG received the B.S. degree from Soochow University, Suzhou, China, in 2017. He is currently pursuing the M.S. degree with the School of Electronic Engineering, Southeast University, Nanjing, China.

His current research interests include digital application specific integrated circuit design and neural network chip.

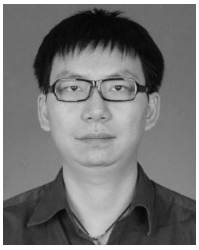
YAN LI received the B.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2018. He is currently pursuing the M.S degree in digital IC design and neural network accelerator design based on FPGA with Southeast University, Nanjing, China.

His current research interests include digital application specific integrated circuit design and neural network chip.



YU GONG received the B.S. degree in mathematics and the M.S. degree in integrated circuits from Southeast University, in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree in electronic science and engineering.

His research interests including approximate computing, reconfigurable computing, deep learning accelerator, and related VLSI design.



WEI GE received the B.S. and Ph.D. degrees in electronic science and engineering from Southeast University, in 2006 and 2015, respectively.

He is currently an Assistant Researcher with the Electrical Engineering Department, Southeast University. His research mainly focuses on SoC design technology, reconfigurable computing, and related VLSI design.