electronics

MDPI

Article

A 34.7 µW Speech Keyword Spotting IC Based on Subband Energy Feature Extraction

Gexuan Wu, Jianlong Wei, Shuai Wang, Guangshun Wei and Bing Li *

State Key Laboratory of Radio Frequency Heterogeneous Integration, Shenzhen University, Shenzhen 518060, China

* Correspondence: bli@szu.edu.cn; Tel.: +86-755-26536198

Abstract: In the era of the Internet of Things (IoT), voice control has enhanced human–machine interaction and the accuracy of keyword spotting (KWS) algorithms has reached 97%; however, the high power consumption of KWS algorithms caused by their huge computing and storage requirements has limited their application in Artificial Intelligence of Things (AIoT) devices. In this study, voice features are extracted by utilizing the fast discrete cosine transform (FDCT) for frequency-domain transformation and to shorten the process of calculating the logarithmic spectrum and cepstrum. The designed KWS system is a two-stage wake-up system, with a sound detection (SD) awakening KWS. The inference process of the KWS network is achieved using time-division computation, reducing the KWS clock to an ultra-low frequency of 24 kHz.At the same time, the implementation of a depthwise separable convolution neural network (DSCNN) greatly reduces the parameter quantity and computation. Under the GSMC 0.11 µm technology, post-layout simulation results show that the total synthesized area of the entire system circuit is 0.58 mm^2, the power consumption is 34.7 µW, and the F1-score of the KWS is 0.89 with 10 dB noise, which makes it suitable as a KWS system in AIoT devices.

Keywords: deep learning; feature extraction; keyword spotting; low-power circuit

check for updates

Citation: Wu, G.; Wei, J.; Wang, S.; Wei, G.; Li, B. A 34.7 µW Speech Keyword Spotting IC Based on Subband Energy Feature Extraction. *Electronics* **2023**, *12*, 3287. https://doi.org/10.3390/electronics12153287

Academic Editors: Spyridon Nikolaidis and Ronald Tetzlaff

Received: 30 June 2023
Revised: 21 July 2023
Accepted: 24 July 2023
Published: 31 July 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the era of the IoT, efficient interaction between humans and IoT devices has become one of the research hotspots in recent years. Intelligent voice control, as a natural and convenient method of human–machine interaction, is extensively used in consumer electronic products. In the past decade, the application of KWS in intelligent terminals has greatly promoted research on KWS algorithms, among which the accuracy rate of KWS algorithms based on deep learning has reached about 97% [1–3], but they are not suitable for AIoT devices due to their large demand for computing and storage resources.

Power consumption is a significant bottleneck for AIoT devices. Battery capacity is limited in many small IoT devices and sensors. Voice control uses a cascade control method to wake up the control modules in stages and satisfy the power consumption restrictions of small IoT devices and sensors. Cascade control primarily includes SD to detect sound presence or absence and KWS to determine whether the keywords within the speech segment match. As an entry point for voice control, the SD module must maintain an "Always On" state to trigger subsequent voice control functionalities. Therefore, it must satisfy the requirements of extremely low operating power consumption and ultra-low silent power consumption in silent environments, which prolongs the device's battery life.

Numerous researchers have explored low-power KWS circuits. Approaches like fixed-point quantization compression of the detection algorithm and ultra-low-power design with a low-voltage threshold have been used, among others. A KWS system circuit was custom-designed based on a fixed-point neural network [4], which achieved a recognition circuit for ten keywords with 200 kB network parameters at 5-bit precision, with a high

system power consumption of 3.33 mW. A fixed-point neural network was used to develop a reconfigurable KWS circuit [5] with a large size of 730 kB SRAM; the power consumption of the circuit in the task of recognizing 11 keywords was 172 µW. A KWS circuit was studied based on binary-weighted convolutional networks with power consumption within 100 µW [6,7]. However, the KWS system is an "Always On" primary module, which will still cause higher system standby power consumption in practical applications. In a 10 dB signal-to-noise ratio (SNR) environment, although the power consumption of [7] is optimized to 15.1 µW, the recognition accuracy rate dropped by approximately 8% compared with [6]. A two-stage structure of SD and KWS was used to implement a KWS chip [8], with an area of 2.56 mm^2, using 32 kB of on-chip storage and power consumption of 10.6 µW at a voltage of 0.6 V. However, the feature extraction (FEx) process is relatively complex. Furthermore, a low-voltage-threshold design method was used to design an ultra-low-power KWS chip with only 510 nW in a 28 nm process [9]. However, this chip can only recognize one to two keywords and cannot meet the command requirements of daily life schemes. At the same time, there have been many attempts by researchers to perform FEx in the analog domain to achieve low power consumption. However, due to less information being contained in the features captured in the analog domain, many schemes only achieve voice activity detection; that is, only they identify speech or non-speech, but cannot realize KWS [10–12]. A KWS circuit using a ring-oscillator-based time-domain processing technique for its analog FEx was proposed with an area of 2.03 mm^2 and power consumption of 23 µW, including analog FEx and digital neural network classifier [13]. The system successfully realizes the low-power implementation of KWS by using the features extracted from the analog domain, but the lack of an SD module to judge the presence of sound may lead to the waste of power consumption in a silent environment.

Some of these studies used Mel-frequency cepstrum coefficient (MFCC) features as voice features, which required complex calculations such as fast Fourier transform (FFT), Mel filtering, logarithmic spectrum calculation, and cepstrum, which consumed significant logic resources and power during feature extraction. Traditional FEx algorithms use FFT to convert time-domain audio signals to frequency-domain representation. The sound signal is a real signal, and the conjugative symmetry of FFT results in half of the data redundancy when using FFT for time-frequency conversion, and the butterfly operation of FFT requires complex multiplication. The complex multiplier faces severe challenges in low-power hardware logic implementation.

This study proposes the use of the FDCT for frequency-domain transformation and to reduce the process of calculating the logarithmic spectrum and cepstrum to extract voice features. All multiplication operations are real-number operations, reducing hardware costs. Based on the extracted voice features, KWS is implemented using a 4-bit fixed-point DSCNN and reduces the clock frequency of the neural network module to 24 kHz through time-sharing calculations, reducing the computing memory of the neural network computation unit from 10.7 kB to 2.75 kB.

The rest of the paper is organized as follows: Section 2 introduces the proposed voice control algorithm, including SD, FEx, and KWS. Hardware implementation is discussed in Section 3. The behavioral simulations of the algorithm are presented in Section 4. Post-layout simulation results and the performance comparison are presented in Section 5. Finally, the paper is concluded in Section 6.

2. Algorithm

2.1. Sound Detection

As the "Always On" component of the system, the SD module must be designed with minimal computational requirements and extremely low power consumption. The SD module is algorithmically simple, using the short-term average amplitude feature, which requires minimal computation, to achieve ultra-low power consumption in a silent environment. The accumulated sum is processed through averaging, as depicted in Equation (1), to avoid excessively large values. Given the 32 ms frame length and 16 ms frame shift,

coupled with rectangular window framing, extracting the short-term average amplitude feature requires only 512 addition operations and a single division operation.

$$\bar{M}_n = \left(\sum_{m=0}^{511} |x_n(m)| \right) / 512 \tag{1}$$

Because the divisor 512 is equivalent to 2^9, the division logic can be implemented by shifting nine places to the right. Because the frameshift is precisely half the frame length, the hardware implementation can reduce the computational load of the SD module by half by reusing the calculation results from the frameshift section, ensuring ultra-low power consumption. Finally, the calculated feature value $\bar{M}_n$ is compared with a preset threshold M_{th}, if $\bar{M}_n > M_{th}$, the FEx module is activated; otherwise, it remains idle.

2.2. Feature Extraction

In the FEx module, the frame length is 32 ms, and the frameshift is 16 ms. The typical computation of MFCC requires several complex calculations, such as FFT, Mel filtering, logarithm computation, and discrete cosine transform (DCT) transformation. Hence, this study proposes a sub-band energy feature based on the DCT transform calculations and a Mel-filterbank. DCT transformation is a real-number operation. If we use the fast butterfly algorithm, as depicted in Figure 1, the multiplication computational complexity is $(Nlog_2N)/2$, which is consistent with the computational complexity of FFT [14]. However, FFT transformation is a complex-number operation, and complex multipliers are unfriendly in hardware implementation. Typically, one complex multiplier requires four real-number multipliers to implement, as depicted in Equation (2).

$$z = (a + jb) \times (c + jd) = (ac - bd) + j(ad + bc) \tag{2}$$

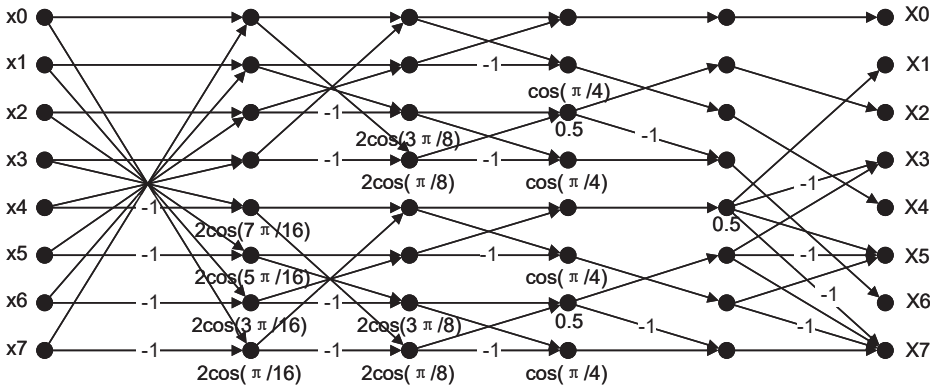

Figure 1. Schematic of 8-point DCT fast butterfly algorithm.

After the DCT transformation, the time-domain audio signal is transformed into a frequency-domain signal. Then, the Mel-filterbank will filter the transformed frequency-domain signal. In the calculation stage of the Mel-filterbank, only the high 4-bit of the transformed spectrum is used in conjunction with the Mel-filterbank coefficient weights to reduce the computational power consumption of this module. Figure 2 illustrates the Mel-filterbank coefficients. The computational expression for the Mel-filterbank portion is depicted in Equation (3).

$$m(l) = \sum_{k=0}^{N-1} melpara_l(k) \bullet |X(k)|, \quad l = 0, 1, \ldots, 31 \tag{3}$$

where l represents the channel number of the filterbank, N is the number of sequences used by the DCT transformation, 512, k is the frequency point after the transformation,

$mel para_l(k)$ is the coefficient weight of the Mel-filterbank channel, $X(k)$ is the frequency-domain value after the DCT, and $m(l)$ is the speech feature data used by the KWS. Each frame of speech, after feature extraction, results in 32 feature values. Figure 3 is a feature spectrum consisting of sub-band energy feature values of 32 frames of speech.

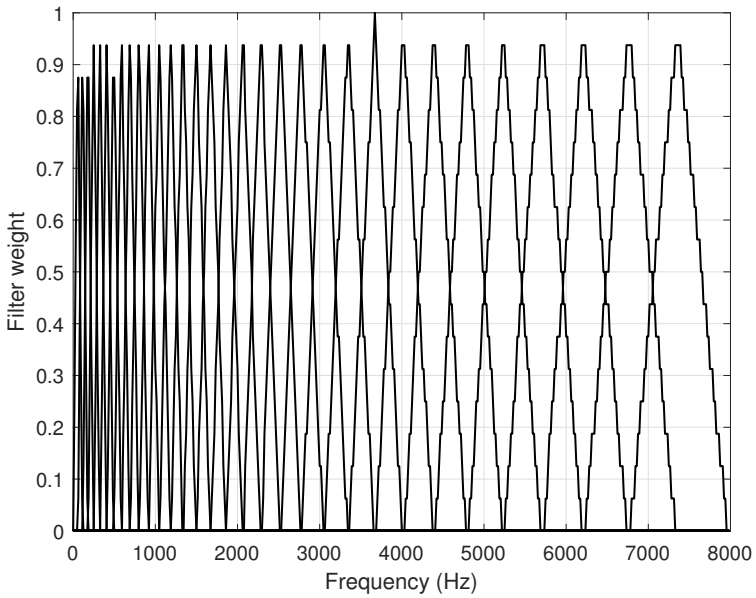

Figure 2. Mel-filterbank coefficients.

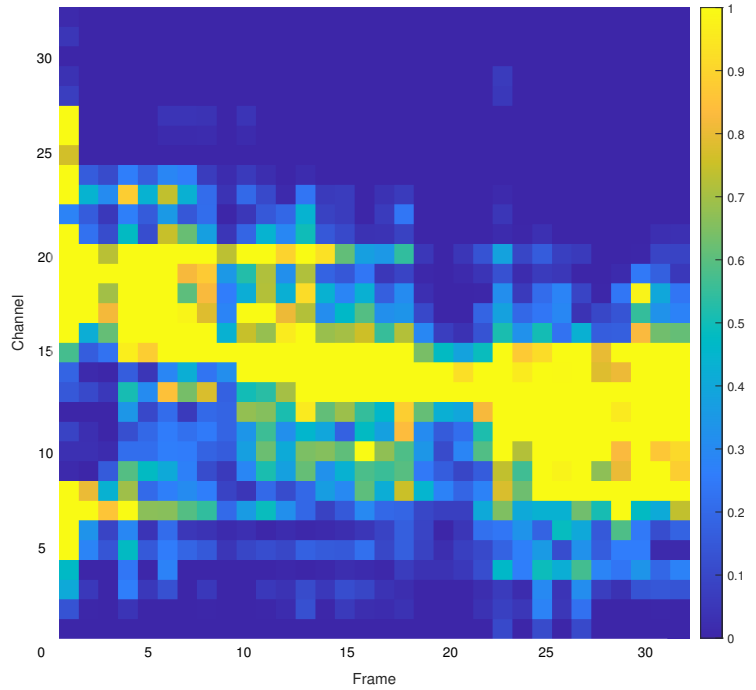

Figure 3. Speech feature spectrum.

Table 1 compares the computational quantity of sub-band energy features under 512 sample points and 32 Mel-filterbanks with the computational quantity of MFCC parameters. Based on the preceding analysis and Table 1, the MFCC parameter computation requires at least 10,324 (1108 + 9216) real-number multiplication operations and 257 logarithm computations, significantly more than the computational quantity required for FEx in this study.

Table 1. Comparison of feature computation.

	This Study	MFCC
Real-number multiplication	3328	1108
Complex-number multiplication	0	2304 *
Logarithm computation	0	257

* equal to 9216 real-number multiplication operations.

2.3. Keyword Spotting

The KWS module uses a DSCNN, an advanced variant of the traditional convolutional neural network (CNN). This DSCNN demonstrates substantial reductions in parameter volume and computational load compared with standard CNNs, deep neural networks (DNNs), and long short-term memory (LSTM) networks, with these reductions becoming particularly noticeable as more attributes are extracted. These reductions in computational and parameter demands can contribute to minimized memory usage and power consumption during the implementation stage in logic circuits.

The structure of the neural network used in the KWS module is illustrated in Figure 4. The initial layer is a conventional convolution layer, succeeded by three depthwise separable convolution (DSC) layers, culminating in a fully connected output layer. The convolution kernel of the first layer is $4 \times 4 \times 1$, with a stride of 2 and 32 convolution kernels. Each of the three intermediate DSC convolution layers uses a $3 \times 3 \times 32$ depthwise convolution kernel, a $1 \times 1 \times 32$ pointwise convolution kernel, and a stride of 2. The Rectified Linear Unit (ReLU) function serves as the activation function.

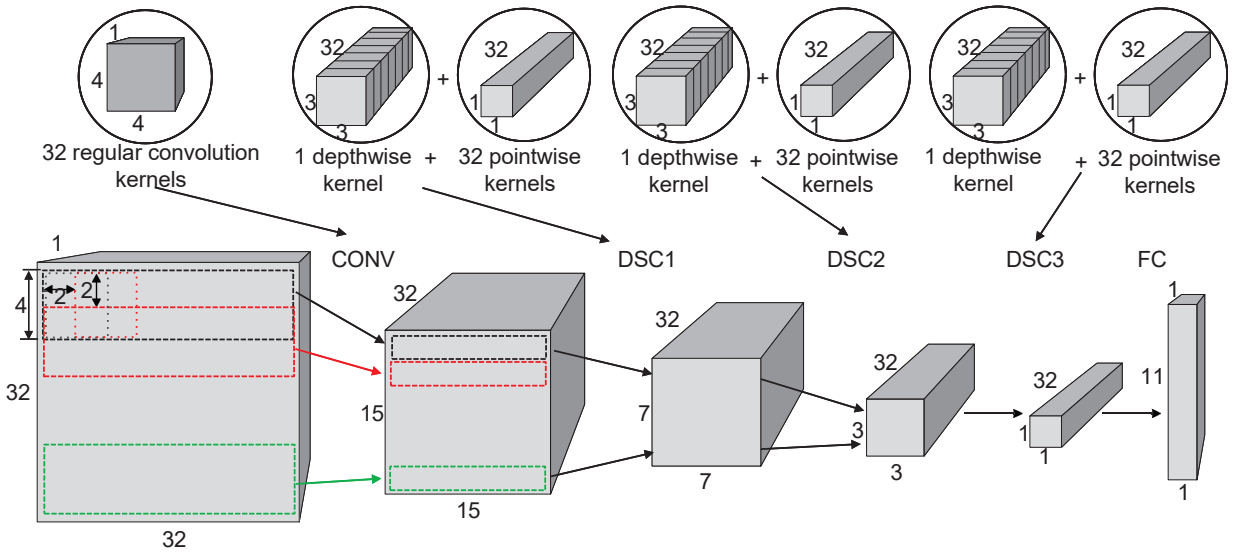

Figure 4. KWS neural network, different colored boxes represent different stages of the operation.

Figure 5 visually represents the computation process of the first DSC layer within the KWS neural network. As depicted, the DSC layer comprises a depthwise convolution layer and a pointwise convolution layer, featuring a collective total of 1345 parameters. A regular convolution layer of the same scale as in Figure 5 is depicted in Figure 6, with a parameter count of 9248. A comparison of these figures reveals that the DSC convolution layer possesses a significantly lower parameter count than a standard convolution layer.

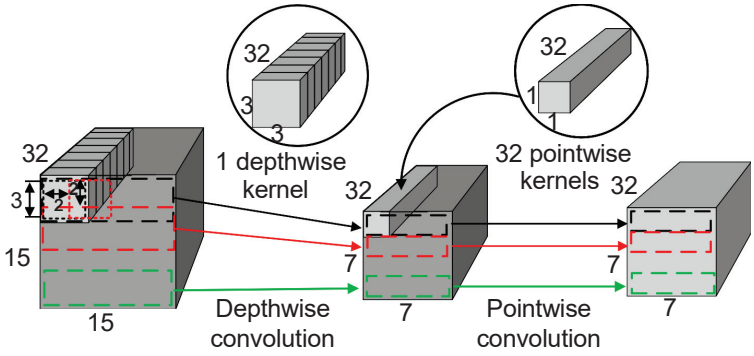

Figure 5. DSC convolution layer calculation diagram, different colored boxes represent different stages of the operation.

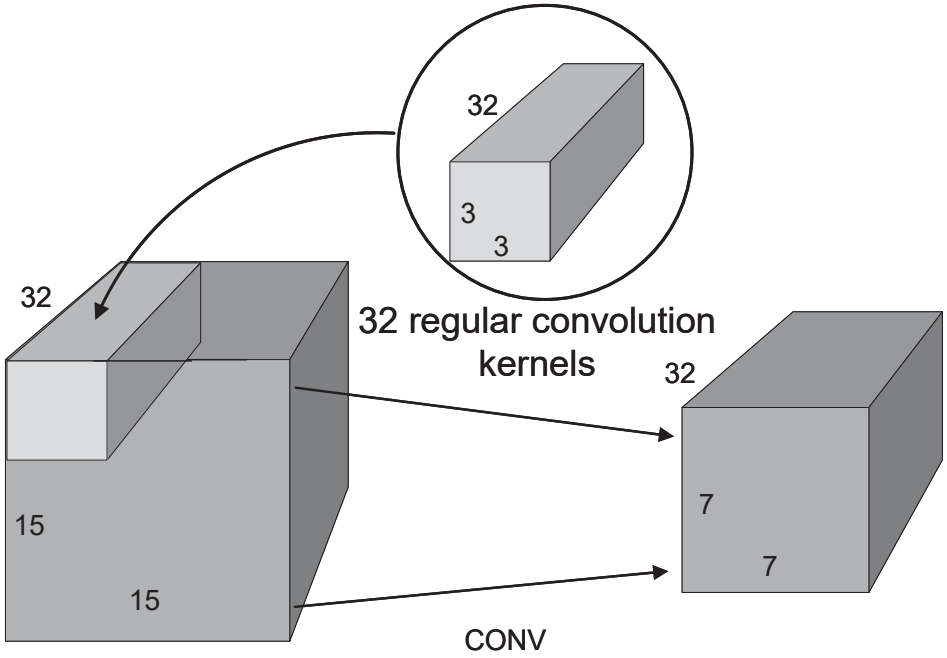

Figure 6. Conventional convolution layer calculation diagram.

All network parameters are set at 4-bit width, with all inputs and features of the intermediate layer maintained at 8-bit width, to reduce power usage during the KWS network's inference phase. The fixed-point quantization method mainly refers to the open-source code Qkeras on GitHub [15]. Table 2 compares DSCNN and CNN resources under this study's design scale constraints. The data demonstrate that the DSCNN's total parameter count amounts to 5035, with a total computational volume of 142,560. The overall parameter count in a CNN of an identical network scale is 28,651, with a computational volume of 608,896. Concerning a standard CNN, the DSCNN achieves an 82.43% reduction in parameter volume and a 76.59% decrease in computational multiplication volume. Consequently, using DSCNN instead of CNN can enable substantial savings in terms of SRAM resources and computation volume per inference, thus optimizing the power consumption of the KWS inference segment at a system algorithm level.

Table 2. Comparison of DSCNN and CNN resources.

	Parameter Quantity	**Multiplication Operation**
CNN	28,651	608,896
DSCNN	5035	142,560

3. Implementation Consideration

3.1. Top-Level System Architecture

Figure 7 displays the top-level framework of the system circuit. When the SD module does not detect any sound, meaning the sound signal is not flipped, the downstream modules are all in a waiting-to-be-triggered state. The trigger signal pulse is generated by the enable signal control module and is then sent to each submodule to trigger its operation. The done signal from all function modules is sent to the enable signal control module to generate the trigger signal for the next level module.

For a low-power implementation of the system circuit, we have divided the entire system circuit into three clock domains based on the computational complexity of each algorithm. The SD module operates under a 16 kHz clock, the voice FEx module operates under a 275 kHz clock, and the KWS neural network and neural network computing unit operate under a 24 kHz clock.

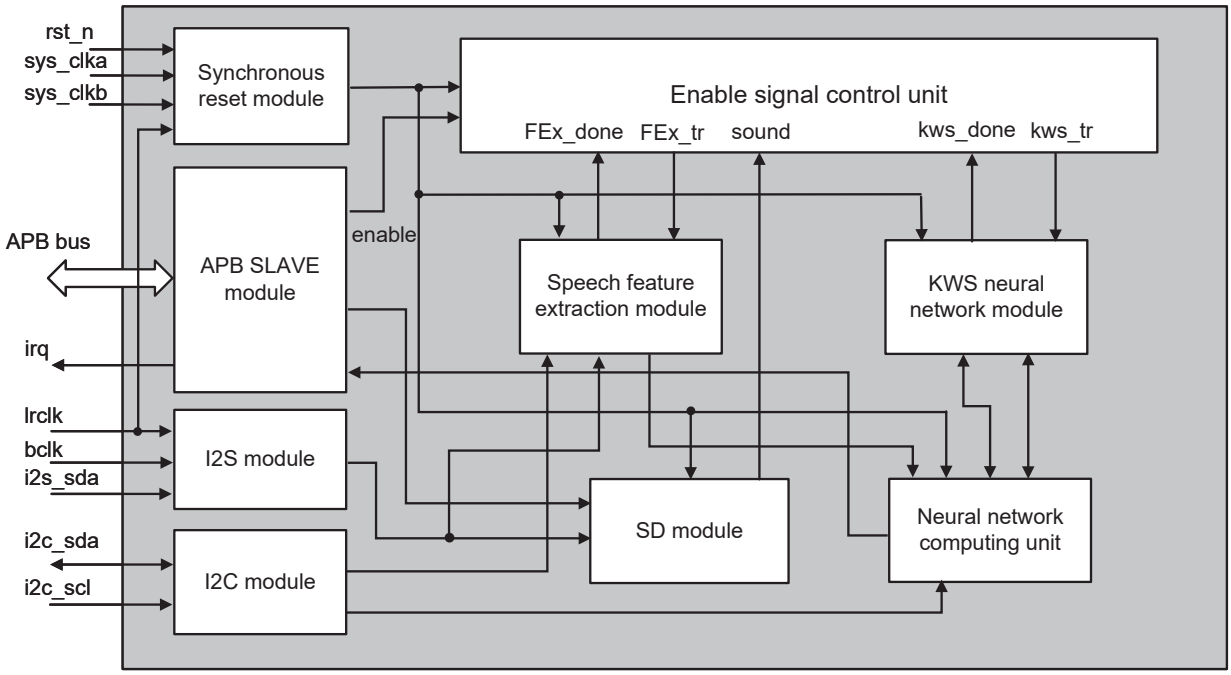

Figure 7. System top-level architecture.

3.2. Sound Detection

Figure 8 is the circuit schematic of the SD module, which only has two adders, two 20-bit registers, three 2-to-1 multiplexers, and a 1-to-2 demultiplexer. The operations involve only abs, addition, selection, bit shifting, and comparison logic.

Because the I2S audio data are in binary complement format, with positive and negative values, the absolute value must be taken before calculating the short-term amplitude. The frameshift designed in this study is 16 ms, which is precisely half of the frame length of 32 ms, so, for each half frame, 256 samples can be accumulated and stored in the register. Accordingly, calculating the short-term average amplitude for each frame can reuse the accumulated result of the frameshift part, reducing the total number of samples added up by the computation from 512 to 256, saving 50% of the computation cost. Dividing by 512 can be achieved by shifting right by 9 bits, avoiding the need for complex division logic. Finally, the short-term average amplitude $\bar{M}_n$ obtained after the shift is compared with the preset threshold M_{th}. If $\bar{M}_n > M_{th}$, it is deduced that there is a sound activity in the current environment, and the generated sound signal is sent to the enable control unit and APB SLAVE for the next level of processing.

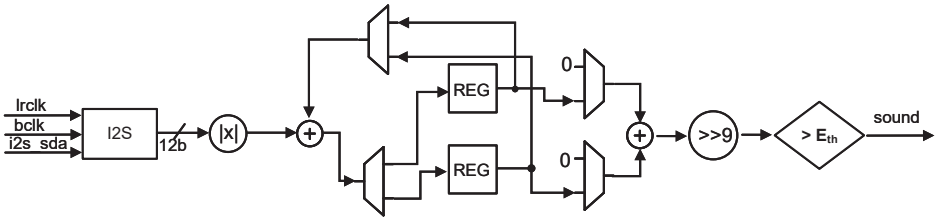

Figure 8. SD circuit schematic.

3.3. Feature Extraction

The structure of the FEx module is depicted in Figure 9. This module principally comprises the FDCT unit, DCT memory, Mel-filterbank unit, and feature memory. The FDCT component performs the computation for the DCT, facilitating the conversion of time-domain audio signals into the frequency domain. The Mel-filterbank unit executes Mel filtering calculations to extract the final audio features. The DCT memory stores the FDCT's intermediate data, whereas the feature memory stores the final feature values, i.e., the results from the Mel-filterbank calculations.

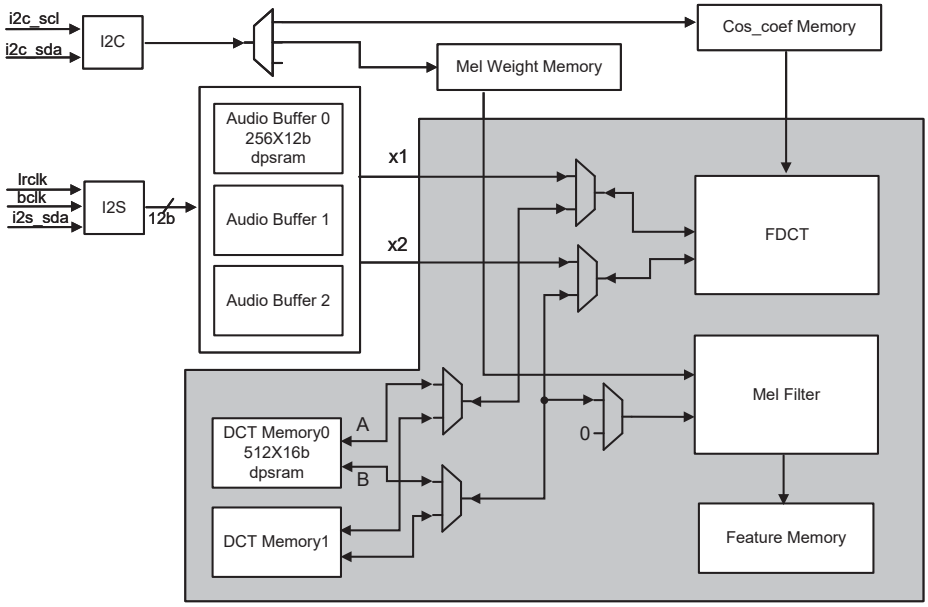

Figure 9. Schematic of FEx module.

3.3.1. Implementation of FDCT

In each cycle, the FDCT retrieves two pieces of data necessary for butterfly computations from either the Audio Buffer or DCT memory, and then writes them back to DCT memory post computation. As the FDCT module performs transformations on datasets with a length of 512 in each instance, a total of 16 computational layers are needed for the entire transformation according to the optimized rapid algorithm discussed in this study, with butterfly operations constituting the first nine layers.

The circuit for the butterfly operation unit is demonstrated in Figure 10. Both x1 and x2 are 12-bit numbers, and cos_coef is an 8-bit number. The results of the butterfly computations must undergo left-shift and right-shift operations to maintain consistency in the data formats for y1 and y2. As each layer's y_values are written into the DCT memory, the maximum y_value (y_max) for the current layer must be stored in the register. The y_max value guides the saturation or truncation processing before the following layer retrieves values from the memory for computation. This ensures the input operands for the butterfly operation unit consistently maintain a 12-bit width, preventing data distortion and overflow that could occur following multiple stages of computation.

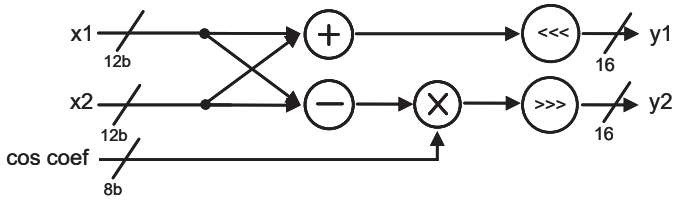

Figure 10. Circuit of the butterfly operation unit.

3.3.2. Implementation of Mel-Filterbank

Post-DCT, the frequency-domain points derived from time-domain audio data must undergo filtration processing via the Mel-filterbank. This study uses triangular Mel filters, with a group of 32 such filters forming the Mel-filterbank, as depicted by the coefficient curve in Figure 2. Some overlap of non-zero coefficients occurs between two neighboring filters, but no overlap is observed between the $k-1$-th and $k+1$-th filters adjacent to the k-th filter. Based on these non-zero coefficient features of the Mel-filterbank, this study proposes an optimized storage method for the entire coefficient matrix, as depicted in Figure 11. The storage matrix measures 512 × 10 bits, with each address precisely corresponding to a frequency point post-DCT transformation.

In this case, $W_{k,m}$ denotes the m-th non-zero coefficient of the k-th filter. A 1-bit flag precedes each coefficient to differentiate between various filter channels in the storage matrix. The upper 5 bits of data represent the coefficients and their corresponding flags of the even-numbered filters, while the lower 5 bits represent the coefficients and corresponding flags of the odd-numbered filters. Taking the upper 5 bits of the coefficient storage matrix as an example, $W_{0,m}$ is the first non-zero coefficient of the first filter, numbered 0. When the flag preceding $W_{0,m}$ switches from 0 to 1, it signifies that the filter coefficient corresponding to flag 1 is from the second filter, numbered 2. When 1 switches back to 0, the corresponding coefficient then belongs to the fourth filter, numbered 4. Before optimization, the storage space needed for the coefficient matrix was 8 kB. However, in storing the coefficient matrix according to the format illustrated in Figure 11, the storage requirement is reduced to 512 × 10 bits, namely 640 B, which reduces the storage space by about 92% compared with before optimization. Furthermore, the flag for the filter number in this study only requires 2 bits, a 60% reduction compared with Giraldo's 5-bit [8]. Furthermore, this method of coefficient storage significantly facilitates the design of the Mel-filterbank computation circuit, reducing the expenditure of logic resources in the circuitry.

0	$W_{0,0}$	0	0
0	$W_{0,1}$	0	$W_{1,0}$
0	$W_{0,2}$	0	$W_{1,1}$
0	$W_{0,3}$	0	$W_{1,2}$
...	...	...	...
1	$W_{2,0}$	0	$W_{1,n}$
1	$W_{2,1}$	0	$W_{1,n+1}$
...	...	...	...
1	$W_{2,n}$	1	$W_{3,0}$
1	$W_{2,n+1}$	1	$W_{3,1}$
...	...	...	...
0	$W_{4,0}$	1	$W_{3,n}$
0	$W_{4,1}$	1	$W_{3,n+1}$
...	...	...	...

Figure 11. Format for storing Mel-filterbank coefficients.

Figure 12 illustrates the Mel-filterbank computation unit circuit. The 1-bit flag in Figure 11 controls the start and termination of accumulation and resets the accumulation register. After data accumulation for each channel, the flag writes the accumulated value (the feature value) into the feature memory.

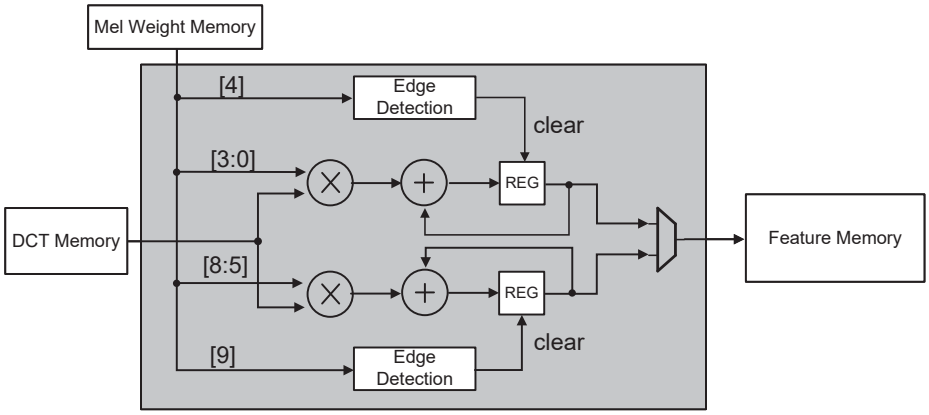

Figure 12. Circuit of the Mel-filterbank computation unit.

3.4. Neural Network Engine

The KWS chip proposed in this study encompasses three neural network frameworks: conventional convolution layers, DSC layers, and fully-connected layers. The DSC layer comprises a depthwise convolution layer, and a pointwise convolution layer. The neural network engine is divided into a network control module and a neural network computation unit. The network control module generates the data addresses and control signals during the neural network computation. It fetches the corresponding data from memory according to the address signals and sends them to the neural network computation unit for inference calculation. As depicted in Figure 13, the neural network computation unit primarily consists of 32 multiply–accumulate (MAC) logic units, a ReLU activation function logic, a quantization function logic, and compute memory. The compute memory comprises 256-bit-width SRAM, where one address can store 32 8-bit data pieces. The neural network computation unit can calculate up to 32 data groups simultaneously, with network parameters being 4 bits and feature values 8 bits. The 32 MAC computation units significantly accelerate neural network computation while maintaining an ultra-low clock frequency of 24 kHz. Post MAC, the multi-path design endows the Neural Network Computational Unit with considerable flexibility, enabling it to expedite the computations of three neural network frameworks.

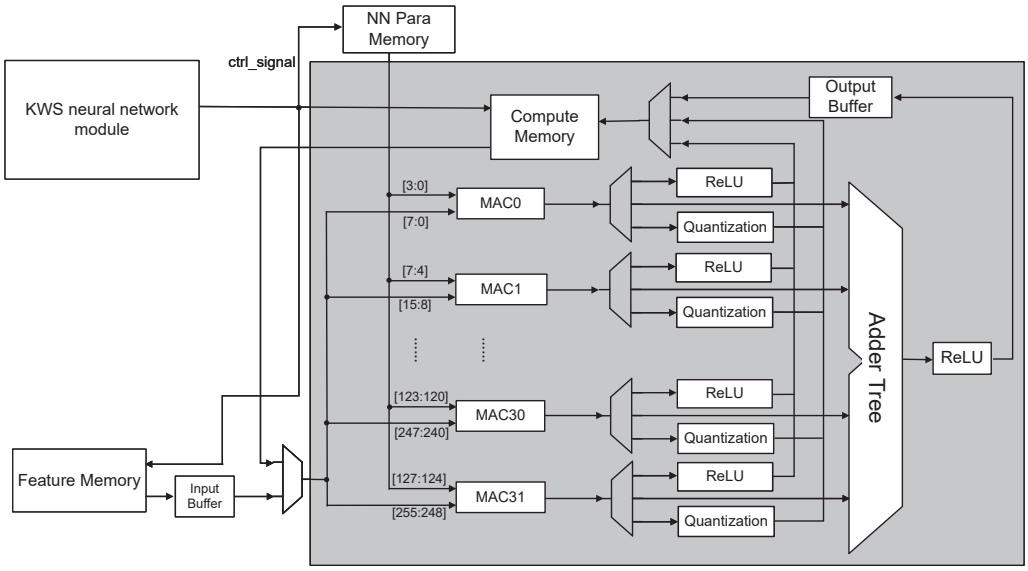

Figure 13. Neural network computation unit.

Figure 14 illustrates the calculation of the first conventional convolution layer in the KWS neural network, while Figure 15 illustrates the calculation of the DSC layer in the KWS neural network. The KWS neural network adopts time-division computation to lower the clock frequency of the convolution neural network computation unit and reduce the computation memory. Given the ordered mapping between the neuron data of the previous and following layers in the DSCNN network structure, and the real-time collection characteristic of audio data, each convolution layer is assigned a computation window, the height of which corresponds to the convolution kernel height.

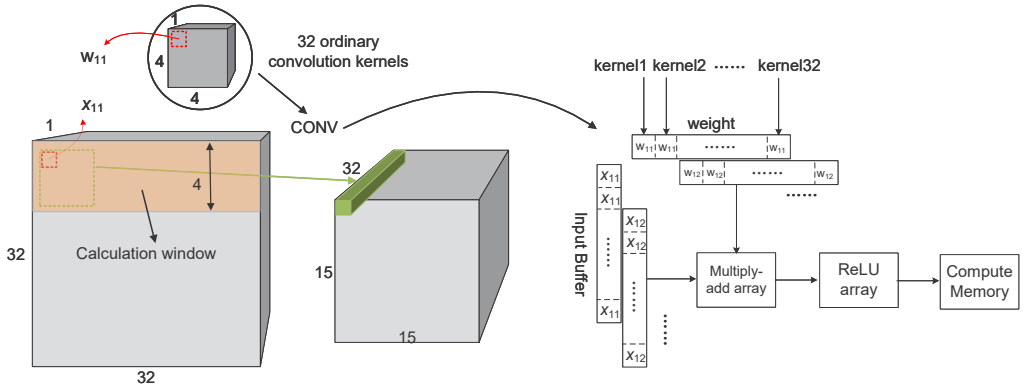

Figure 14. Schematic of KWS conventional convolution layer computation.

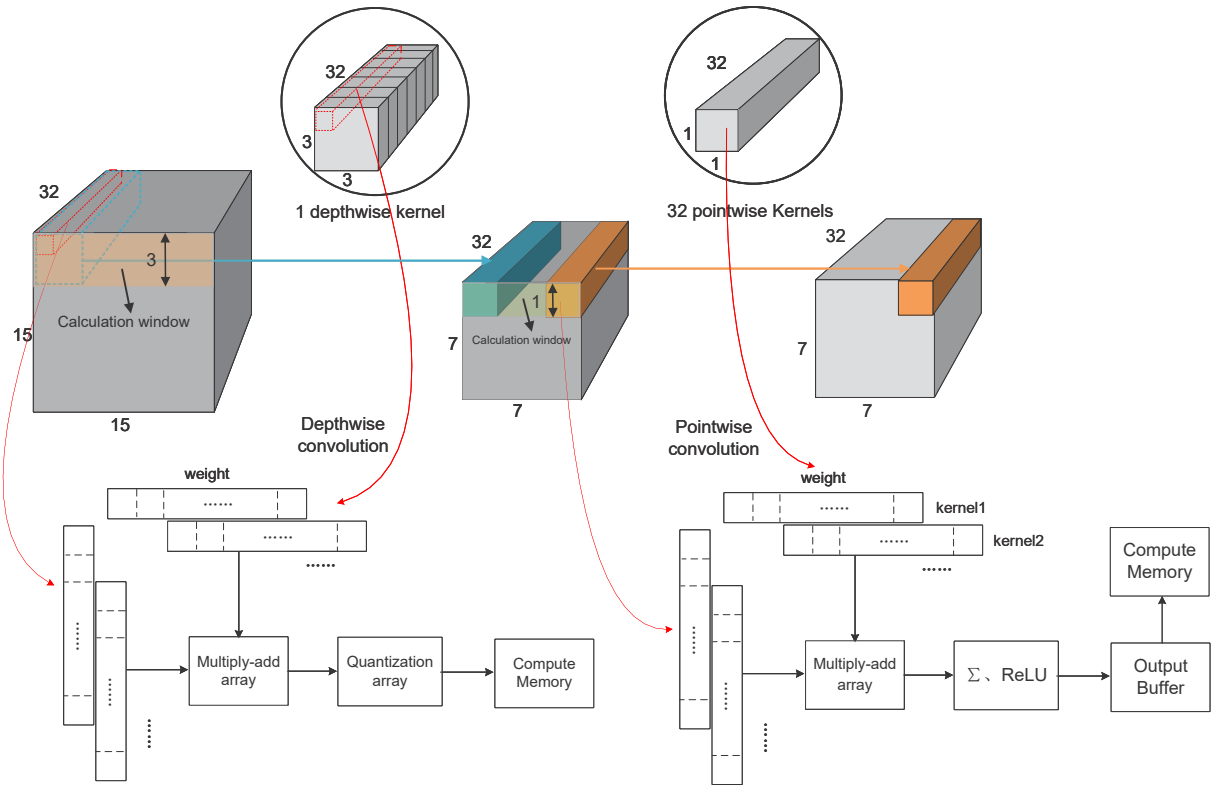

Figure 15. Schematic of KWS DSC layer computation.

Because both conventional and depthwise convolutions have a stride of 2, the computation window must only shift down by two units after each calculation. Therefore, after the first and second frames, the first convolution layer must compute every two frames, the second and third convolution layers compute every four frames, the fourth and fifth convolution layers compute every eight frames, the sixth and seventh convolution

layers compute every 16 frames, and the output layer only computes once. Accordingly, the computation load of the KWS neural network is dispersed throughout 32 frames, reducing the working clock of the neural network module to an ultra-low frequency of 24 kHz and ensuring the low-power logic implementation of the module. Based on the characteristic of time-division computation, the compute memory must only store the data volume corresponding to the computation window. Thus, the capacity of the compute memory of the neural network computation unit only requires 2.75 kB, a reduction of 74.3% of the storage resource expenditure compared with the 10.7 kB of memory required for regular convolution computation.

4. Behavioral Simulation Results of the Proposed Algorithm

The training dataset for KWS consists of commonly used command words in the IoT from the Google Speech Command Dataset (GSCD) and other non-keywords that were randomly selected [16]. The ten classes of command keywords are sequentially numbered from 0–9, with an average of 1500 audio files per command keyword class and 4000 audio files for non-keywords. A real-world environment was simulated by subjecting selected audio files to noise0addition processing at a standard SNR of 10 dB, overlaying three different noise types. The noise data came from the DEMAND dataset [17]. During the KWS neural network training, the audio data were randomly split into test and training sets at a ratio of 3:7.

The KWS algorithm trained in this study is a 10-keyword detection neural network. These 10 keywords are “down”, “up”, “stop”, “go”, “left”, “right”, “no”, “yes”, “on”, and “off”, each having a decimal encoding in hardware implementation from “0” to “9”. Moreover, there is an output for non-keywords, encoded as “10”. The neural network can be trained on any set of keywords and the trained parameters are used in the inference of the neural network, hence the system can be configured to recognize any set of keywords. This study conducts joint debugging of all modules and chooses an audio test stimulus composed of “down”, “stop”, and “Marvin” that lasts for 3 s for testing and verification to intuitively demonstrate whether the function of the entire system is correct, as depicted in Figure 16. The first two words, “down” and “stop”, are speech under 10 dB of white noise, while “Marvin” is speech under 0 dB of white noise. The red box in Figure 16 represents the detection results of the SD algorithm. The bar within the box indicates the part where the audio exceeds the threshold, where SD determines there is sound. SD in Figure 16 has one frame of data below the threshold at “1” and noise misjudgment at “2” and “3”. At this time, the SD threshold register data are configured to 0x4A.

Figure 17 is the simulation waveform of the system circuit after inputting the same audio stimulus. As shown in the simulation waveform in Figure 17, KWS started three times, corresponding to the three keywords in the audio. The encoding of “down” is “0”, “stop” is “2”, and “Marvin” is not one of the preset keywords, with a decimal encoding of “10”. The “kwd num” of the detection system is the detection result. The encoding of the keyword determined the first time is “0”, the second time is “2”, and the third time is “a”, which is the hexadecimal representation of “10”, which is consistent with the encoding “0”, “2”, “10” of the three keywords in the audio test stimulus. Based on this analysis, the function of the keyword spotting system circuit in this study can correctly identify different keywords and non-keywords.

Table 3 presents the test results of the KWS algorithm under different types of noise at a 10 dB SNR. The fixed-point KWS algorithm performs optimally on the audio dataset overlaid with 10 dB of office noise, with an average precision of keyword spotting of 91%, a recall rate of 87%, and an F1 score of 0.89. In a 10 dB SNR environment, the KWS algorithm’s accuracy is consistently around 86%.

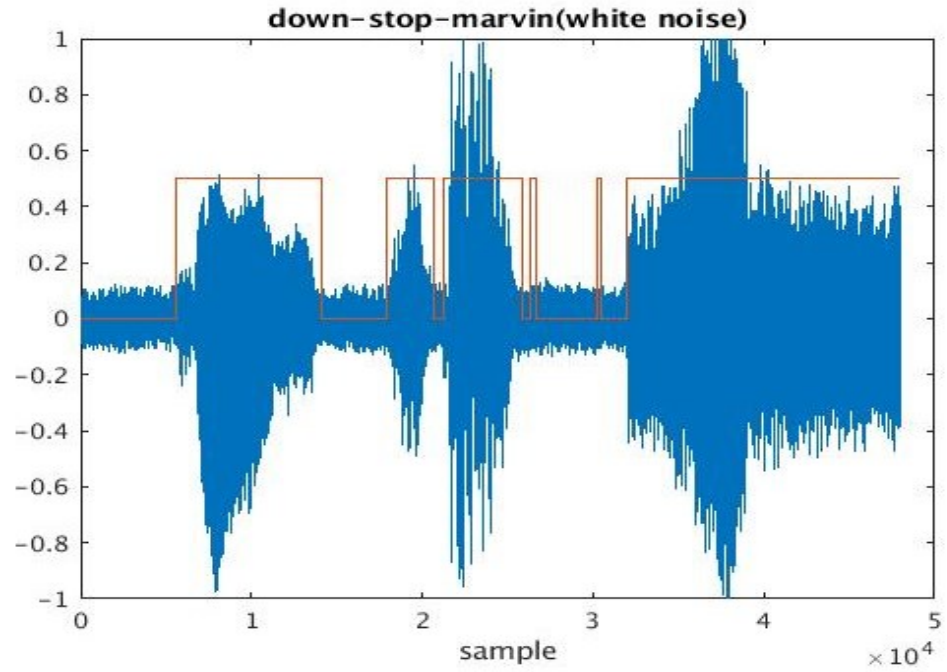

Figure 16. Test audio waveform.

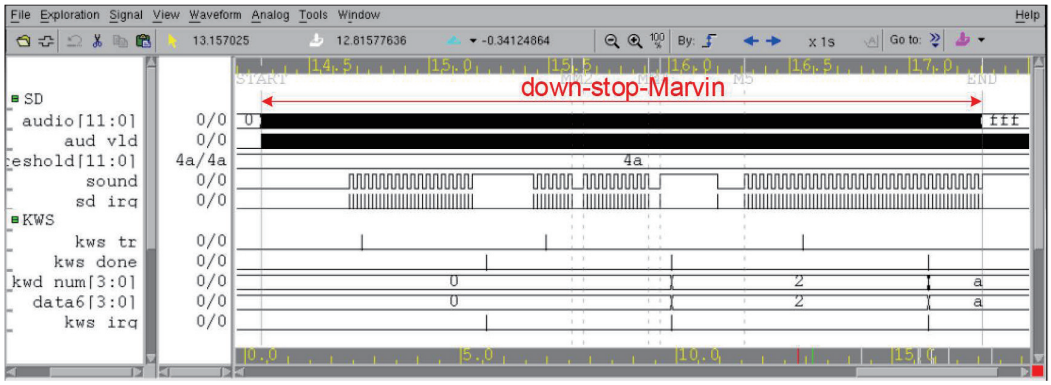

Figure 17. Simulation waveform of system circuit.

Table 3. Test results of the KWS algorithm.

	10 dB White Noise	10 dB Metro Noise	10 dB Office Noise
Accuracy	0.86	0.85	0.88
Macro Precision	0.89	0.88	0.91
Macro Recall	0.85	0.84	0.87
Macro F1	0.87	0.86	0.89

5. Circuit Implementation and Performance Comparison

5.1. Circuit Implementation

The proposed algorithm was implemented in GSMC 0.11 µm CMOS technology using a 1.2 V supply voltage for digital logic circuit unit and a 1.5 V supply voltage for SRAM. The layout of the implemented chip, as depicted in Figure 18, has an overall area of 0.58 mm^2, with SRAM occupying approximately 80% of that area. Figure 19 depicts the distribution of the SRAM in the designed circuit, which altogether requires around 10.2 kB of SRAM resources.

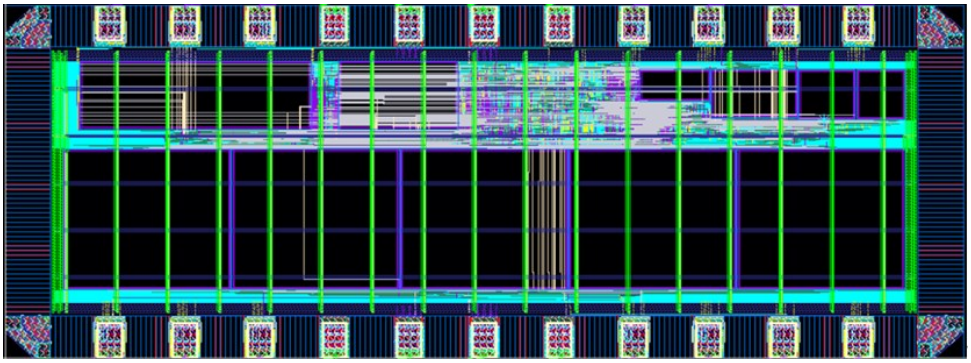

Figure 18. Layout of proposed KWS system.

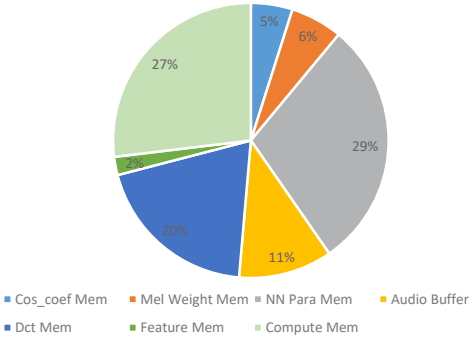

Figure 19. Distribution of SRAM.

5.2. Power Performance

We simulated the power consumption under three process, voltage, and temperature (PVT) conditions. The PVT information and power consumption simulation data are presented in Tables 4 and 5, where the PVT 2 condition is a typical value. Based on Table 5, the lowest average system power consumption in SD mode occurs under typical PVT conditions, with a value of just 1.65 µW, where dynamic power consumption is the main component, accounting for 91% of the total power consumption.

Table 4. PVT information.

	Std Cell Voltage (V)	SRAM Voltage (V)	Temperature (°C)	Process
PVT 1	1.08	1.35	125	slow
PVT 2	1.2	1.5	25	typical
PVT 3	1.32	1.65	−40	fast

Figure 20 illustrates the system power consumption distribution under PVT2 conditions in SD + KWS mode. From this, the FEx module has the highest power consumption in the system, contributing to 85% of the total power consumption. Following this, the KWS classifier part accounts for 8%, with the remaining modules contributing approximately 7% to the total power consumption. This power consumption distribution is reasonable because the FEx module operates at the highest clock frequency and has the greatest computational load, necessitating frequent memory read and write operations, leading to the highest average power consumption for this module. Even though the computation load of the KWS part is also substantial, its total computation time is 32 frames. Averaging this over each frame shows that its computational load is much less than that of feature extraction. Furthermore, the working clock frequency of the neural network module is only

one-tenth that of the FEx module, thus confirming that the power consumption distribution of the entire system circuit is reasonable.

Table 5. The results of power consumption simulation.

	System Power Consumption (µW)	SD Mode	SD + KWS Mode
PVT 1	Static power consumption	3.07	3.12
	Dynamic power consumption	1.2	28
	Total power consumption	4.27	31.12
PVT 2	Static power consumption	0.15	0.18
	Dynamic power consumption	1.5	34.52
	Total power consumption	1.65	34.7
PVT 3	Static power consumption	0.1	0.13
	Dynamic power consumption	1.8	42.67
	Total power consumption	1.9	42.8

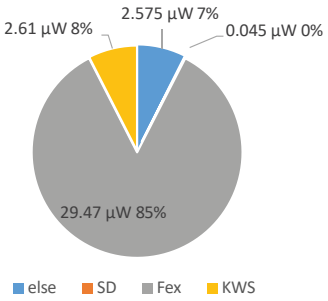

Figure 20. System power consumption distribution.

5.3. Performance Comparison

Table 6 compares the KWS performance of this circuit with other works. The data used in the analysis and comparison below are the power consumption data under the typical PVT 2 condition. However, all the data in this study are post-layout-simulated, and the data in other comparison articles are chip-measured.

Table 6. Performance comparison with other KWS systems.

	This Study	ISSCC 2017 [5]	JSSC 2020 [8]	JSSC 2021 [18]	TCAS-I 2020 [7]	JSSC 2022 [13]
Technology (nm)	110	65	65	65	22	65
Area (mm^2)	**0.58**	13.17	2.56	4.13	0.6	2.03
Voltage (V)	1.2/1.5	0.6	0.6	1	0.6	0.5/0.75
On-chip SRAM (kB)	**10.2**	730	32	38	11	27
Dataset	GSCD	WSJ	GSCD	GSCD	GSCD	GSCD
Classifier	DSCNN	DNN + HMM	LSTM	RAM	CNN	RNN
Number of keywords	10	11	12	7	10	10
Recognition accuracy	0.88@10 dB	0.96@N/A	0.90@N/A	0.90@N/A	0.84@10 dB	0.86@N/A
Latency	16 ms	10 ms	16 ms	40 µs	16 ms	12.4 ms
Frequency	275 kHz/24 kHz	3 MHz	250 kHz	1.9 MHz	250 kHz	250 kHz
Power (µW)	34.7	172	10.6	N/A	15.1	23

The 275 kHz and 24 kHz in Table 6 are the clock frequencies of the FEx module and the neural network part. The supply voltages of the digital logic circuit unit and

SRAM are 1.2 and 1.5 V, which are outdated and higher than other research. However, because of the characteristics of DSCNN's fewer parameters and smaller computation amount, it still has a significant advantage in power consumption and area compared with the literature [5]. Studies [7,8] use 22 nm and 65 nm technology, which are better in terms of power consumption than this circuit. Under the same circuit performance, the more advanced the process, the smaller the voltage, current, and corresponding power consumption. The power consumption of a digital circuit is directly proportional to the square of the working voltage, and the supply voltage of the process used in this study is about twice that in the reference literature. Thus, the influence of voltage on power consumption is four times higher. Although the power consumption in the literature [7] is lower, it uses a binary weight convolutional network, so there is a significant loss in KWS recognition performance, which is only 84%. This study uses a 4-bit fixed-point DSCNN network with a smaller loss in recognition performance, with an accuracy of 88% under a 10 dB SNR. The DSCNN network has significantly reduced parameter quantity and computational load compared with DNN, LSTM, and CNN networks, so the on-chip SRAM resources it uses are also smaller than in other studies.

6. Conclusions

In this study, we designed a low-power KWS chip based on deep learning. First, optimization was performed on the system algorithm level for the KWS circuit, and a low-precision fixed-point quantization FEx and detection algorithm was proposed. In feature extraction, to avoid the complex multiplication operation of FFT, this study uses the DCT for frequency-domain transformation and reduces the process of calculating the logarithmic spectrum and cepstrum. The extracted audio features are further captured and classified by the neural network. Optimization was performed at the system structure level, and a two-level triggering system structure based on SD-KWS was proposed to reduce the system's average power consumption. In circuit implementation, the time-sharing calculation method of the KWS neural network reduces the clock frequency of the entire neural network module to 24 kHz and the compute memory of the neural network computation unit from 10.7 kB to 2.75 kB.

Under the GSMC 0.11 µm technology, the total synthesized area of the entire system circuit is 0.58 mm^2, the power consumption under the system's low-power work mode is only 1.65 µW, and the average power consumption during keyword spotting is 34.7 µW. Under an SNR of 10 dB, the F1-score of KWS is 0.89.

Author Contributions: Conceptualization, G.W. (Gexuan Wu); Software, G.W. (Gexuan Wu) and J.W.; Formal analysis, S.W.; Data curation, G.W. (Guangshun Wei); Writing – original draft, J.W.; Writing – review and editing, G.W. (Guangshun Wei) and B.L.; Visualization, S.W.; Project administration, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Shenzhen Science and Technology Development Funds under Grant (JCYJ20190808115001775, ZDSYS20220527171402005) and the Guangdong Natural Science Foundation of China (2023A1515011275).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, G.; Parada, C.; Heigold, G. Small-footprint keyword spotting using deep neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 4087–4091.
2. Shan, C.; Zhang, J.; Wang, Y.; Xie, L. Attention-based end-to-end models for small-footprint keyword spotting. In Proceedings of the Interspeech 2018, Hyderabad, India, 2–6 September 2018.
3. Mittermaier, S.; Kurzinger, L.; Waschneck, B.; Rigoll, G. Small-footprint keyword spotting on raw audio data with Sinc-Convolutions. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 7454–7458.

4. Shah, M.; Wang, J.; Blaauw, D.; Sylvester, D.; Kim, H.-S.; Chakrabarti, C. A fixed-point neural network for keyword detection on resource constrained hardware. *IEEE Signal Process. Syst.* **2015**, *90*, 727–741. [CrossRef]
5. Price, M.; Glass, J.; Chandrakasan, A. 14.4 A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating. In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 5–9 February 2017; pp. 244–245.
6. Liu, B.; Wang, Z.; Fan, H.; Yang, J.; Liu, B.; Zhu, W.; Huang, L.; Gong, Y.; Ge, W.; Shi, L. EERA-KWS: A 163 TOPS/W always-on keyword spotting accelerator in 28 nm CMOS using binary weight network and precision self-adaptive approximate computing. *IEEE Access* **2019**, *7*, 82453–82465 . [CrossRef]
7. Liu, B.; Cai, H.; Wang, Z.; Sun, Y.; Shen, Z.; Zhu, W.; Li, Y.; Gong, Y.; Ge, W.; Yang, J.; et al. A 22 nm, 10.8 µW/15.1 µW dual computing modes high power-performance-area efficiency domained background noise aware keyword-spotting processor. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 4733–4746. [CrossRef]
8. Giraldo, J.; Lauwereins, S.; Badami, K.; Verhelst, M. Vocell: A 65-nm speech-triggered wake-up SoC for 10-µW keyword spotting and speaker verification. *IEEE J. Solid-State Circuits* **2020**, *55*, 868–878. [CrossRef]
9. Shan, W.; Yang, M.; Xu, J.; Lu, Y.; Zhang, S.; Wang, T.; Yang, J.; Shi, L.; Seok, M. 14.1 A 510 nW 0.41 V low-memory low-computation keyword-spotting chip using serial FFT-based MFCC and binarized depthwise separable convolutional neural network in 28 nm CMOS. In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 16–20 February 2020; pp. 230–232.
10. Faghani, M.; Rezaee-Dehsorkh, H.; Ravanshad, N.; Aminzadeh, H. Ultra-Low-Power Voice Activity Detection System Using Level-Crossing Sampling. *Electronics* **2023**, *12*, 795. [CrossRef]
11. Gutierrez, E.; Perez, C.; Hernandez, F.; Hernandez, L. Time-Encoding-Based Ultra-Low Power Features Extraction Circuit for Speech Recognition Tasks. *Electronics* **2020**, *9*, 418. [CrossRef]
12. Yang, M.; Yeh, C.; Zhou, Y.; Cerqueira, J.; Lazar, A.; Seok, M. A 1 µW Voice Activity Detector Using Analog Feature Extraction and Digital Deep Neural Network. In Proceedings of the 2018 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 11–15 February 2018; pp. 346–348.
13. Kim, K.; Gao, C.; Graça, R.; Kiselev, I.; Yoo, H.-J.; Delbruck, T.; Liu, S.-C. A 23-µW Keyword Spotting IC with Ring-Oscillator-Based Time-Domain Feature Extraction. *IEEE J. Solid-State Circuits* **2022**, *57*, 3298–3311. [CrossRef]
14. Wang, Z. Fast algorithms for the discrete W transform and for the discrete Fourier transform. *IEEE Trans. Acoust. Speech Signal Process.* **2003**, *32*, 803–816. [CrossRef]
15. A Quantization Deep Learning Library for Tensorflow Keras. Available online: https://github.com/google/qkeras (accessed on 18 July 2023).
16. Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv* **2018**, arXiv:1804.03209.
17. Thiemann, J.; Ito, N.; Vincent, E. The diverse environments multi-channel acoustic noise database (DEMAND): A database of multichannel environmental noise recordings. *J. Acoust. Soc. Am.* **2013**, *133*, 3591. [CrossRef]
18. Dbouk, H.; Gonugondla, S.; Sakr, C.; Shanbhag, N.R. A 0.44-µJ/dec, 39.9-µs/dec, recurrent attention in-memory processor for keyword spotting. *IEEE J. Solid-State Circuits* **2021**, *56*, 2234–2244. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.