electronics

MDPI

Article

A Resource-Efficient Keyword Spotting System Based on a One-Dimensional Binary Convolutional Neural Network

Jinsung Yoon, Neungyun Kim, Donghyun Lee, Su-Jung Lee, Gil-Ho Kwak and Tae-Hwan Kim *

School of Electronics and Information Engineering, Korea Aerospace University, 76, Hanggongdaehak-ro, Deogyang-gu, Goyang-si 10540, Gyeonggi-do, Republic of Korea
* Correspondence: taehwan.kim@kau.kr

Abstract: This paper proposes a resource-efficient keyword spotting (KWS) system based on a convolutional neural network (CNN). The end-to-end KWS process is performed based solely on 1D-CNN inference, where features are first extracted from a few convolutional blocks, and then the keywords are classified using a few fully connected blocks. The 1D-CNN model is binarized to reduce resource usage, and its inference is executed by employing a dedicated engine. This engine is designed to skip redundant operations, enabling high inference speed despite its low complexity. The proposed system is implemented using 6895 ALUTs in an Intel Cyclone V FPGA by integrating the essential components for performing the KWS process. In the system, the latency required to process a frame is 22 ms, and the spotting accuracy is 91.80% in an environment where the signal-to-noise ratio is 10 dB for Google speech commands dataset version 2.

Keywords: keyword spotting; convolutional neural networks; binarized neural networks; inference; processor; field-programmable gate arrays

Citation: Yoon, J.; Kim, N.; Lee, D.; Lee, S.-J.; Kwak, G.-H.; Kim, T.-H. A Resource-Efficient Keyword Spotting System Based on a One-Dimensional Binary Convolutional Neural Network. *Electronics* **2023**, *12*, 3964. https://doi.org/10.3390/electronics12183964

Academic Editor: Valeri Mladenov

Received: 23 August 2023
Revised: 19 September 2023
Accepted: 19 September 2023
Published: 20 September 2023

1. Introduction

Keyword spotting (KWS) refers to the task of detecting predefined keywords in an audio stream. This plays a foundational role in voice assistant services by triggering the services with a single keyword [1]. KWS systems are embedded in devices, such as smartphones and smart speakers, and involve considerable complexity with respect to spotting the keyword spoken by unspecified speakers. Such complexity becomes a limiting factor in miniaturized embedded devices with tight constraints on the available resources, thus necessitating the creation of an efficient KWS system with low resource usage. Recent works have shown that feature extraction using Mel-frequency cepstral coefficients (MFCCs) [2] and classification via deep neural networks (DNNs) [3] can contribute to achieving high spotting accuracy, even in environments with background noise. However, these techniques increase the complexity of the system [4].

To reduce the complexity of the KWS system, various techniques have been proposed to optimize the feature extraction and classification process. K. He [5] and Y. S. Chong [6] attempted to reduce the complexity of calculating the MFCCs. In the work of K. He [5], the discrete cosine transformation and windowing procedure were removed. In the work of Y. S. Chong [6], a rectangular filter was introduced instead of the conventional triangular filter to reduce computational and memory requirements. Convolutional neural networks (CNNs) have been successfully employed to perform classic speech recognition tasks [7], which is closely related to KWS. Several studies employing CNNs or their variants to implement KWS systems have been conducted [8]. In the work of S. Bae [9], the memory required for storing model parameters was reduced by employing a depthwise separable CNN (DS-CNN) while achieving high spotting accuracy based on ternary quantization. In the work of W. Shan [10], a DS-CNN-based model was also employed to reduce system complexity with a low memory footprint. In the work of B. Liu [11], a low-complexity

KWS system was effectively implemented by eliminating the MFCC calculations with a one-dimensional convolutional recurrent neural network (1D-CRNN) model.

In addition, studies on achieving a real-time KWS system with high spotting accuracy have been conducted. In the work of E. Ceolini [12], the processing latency of a KWS system was reduced by introducing rate-based machine learning networks. In the work of Yan [13], an event-driven pipeline was utilized to reduce the processing latency of KWS systems. In the work of R. Prabhavalkar [14], a robust KWS system was developed to achieve high spotting accuracy even in noisy environments on the basis of the novel technique of automatic gain control. In the work of Y. A. Hwang [15], a voice enhancement algorithm with dual microphones was employed to implement a noise-robust KWS system.

This work proposes a resource-efficient KWS system based on a 1D binary convolutional neural network (1D-BCNN) and validates the proposed system with the implementation results. This is the extension of our preliminary work presented in [16]. The contributions can be summarized as follows:

- The proposed system relies on a single 1D-CNN model for the purposes of feature extraction and classification. Relatively few resources are used, as no additional hardware is explicitly required to perform feature extraction.
- Resource requirement is further reduced by binarizing the 1D-CNN model based on XNOR-Net [17], and the inference engine [18] is optimized to handle a 1D audio stream. This engine uses threshold- and pooling-based skipping techniques to skip redundant operations, thereby achieving a high inference speed.
- The proposed system is implemented using 6895 ALUTs in an Intel Cyclone V FPGA. The latency of processing a frame is as short as 22 ms. In an environment with a signal-to-noise ratio (SNR) of 10 dB, the spotting accuracy for Google speech command dataset version 2 (GSCD v2) [19] is 91.80%.

The remainder of this paper is organized as follows. Section 2 briefly reviews the conventional KWS system. Section 3 details the proposed system. Section 4 presents the implementation results of the proposed system and compares its performance with previous results. Section 5 presents our conclusions.

2. Conventional KWS System

The KWS process is conventionally performed in two stages: feature extraction and classification. The audio samples are first processed by emphasizing the human-audible frequency range and grouped into a fixed-length frame for the subsequent KWS process. The features with the characteristics of the predefined keywords are extracted in the feature extraction stage. The extracted features are used to determine whether they correspond to the predefined keyword by computing the probability distribution for the keywords in the classification stage.

MFCCs are widely used to represent the audio features [2–6,9,10,12,14,20–22]. The calculation process for the MFCCs can be described as follows. The frequency spectrum for each of the short ranges in a frame is first calculated based on the Fourier transform. This spectrum is then processed by using a Mel-scale filter bank, allowing for the extraction of energy at the specific frequency range, which approximates the human auditory system's response. After extracting the energy, the logarithmic functions are applied to these values to accentuate the magnitude difference. Finally, the discrete cosine transform is applied to this log-scaled spectrum. The resulting coefficients are then used as features for the subsequent KWS process. The procedure for calculating the MFCCs described above is similar to the mechanism that occurs in the human cochlea. Hence, the MFCC-based features can be efficaciously used to implement the KWS process to achieve high spotting accuracy.

DNNs are usually employed to perform classification. The DNNs have structures with several blocks connected in series, where each block is composed of numerous neurons. The neurons in adjacent blocks are connected via non-linear activation, and the DNNs can be trained to find non-linear relationships between the features and keywords. Through

this mechanism, the DNNs can be employed to successfully achieve highly accurate classification. Various kinds of DNNs have been employed for classification in the KWS systems. The CNN model presented in [22] was used to perform classification with MFCC-based feature maps. The 1D-CNN model presented in [5] was used to find features to recognize the specific part of a keyword. The long short-term memory (LSTM) model presented in [21] was trained to capture long-term information in an audio stream.

Even though the MFCCs and DNNs are effective in achieving high spotting accuracy, they involve considerable complexity, thus entailing many resources for hardware implementation. The MFCC-based features are calculated based on complicated operations such as Fourier and cosine transforms. It has been reported that about 40% of hardware resources are used to implement the MFCC calculations [4]. The DNN-based classification demands high memory bandwidth and computational complexity. It has been reported that about 50% of the hardware resources are used to implement the DNN-based classification [4].

3. Proposed KWS System

This section presents the design and implementation of the proposed KWS system. The proposed system performs the KWS process based only on the 1D-CNN model without involving complicated processing. The 1D-CNN model is binarized to reduce resource usage, and a dedicated hardware engine is employed to achieve a high inference speed.

3.1. 1D-CNN-Based KWS Process

In the proposed KWS system, the overall processing is performed in two stages, as shown in Figure 1. The audio samples are first grouped into frames and then classified into keywords through 1D-CNN inference. This inference fulfills not only feature extraction but also classification. The inference process is composed of operations through convolutional blocks and fully connected blocks. The features are extracted from the convolutional blocks, whose weights have been trained to map the temporal patterns to the keywords. The 1D-CNN operation is performed in the time domain, where the convolution can be considered as the cross-correlation between the audio samples and filters with the trained coefficients. The keywords are classified in the fully connected blocks with the extracted features.

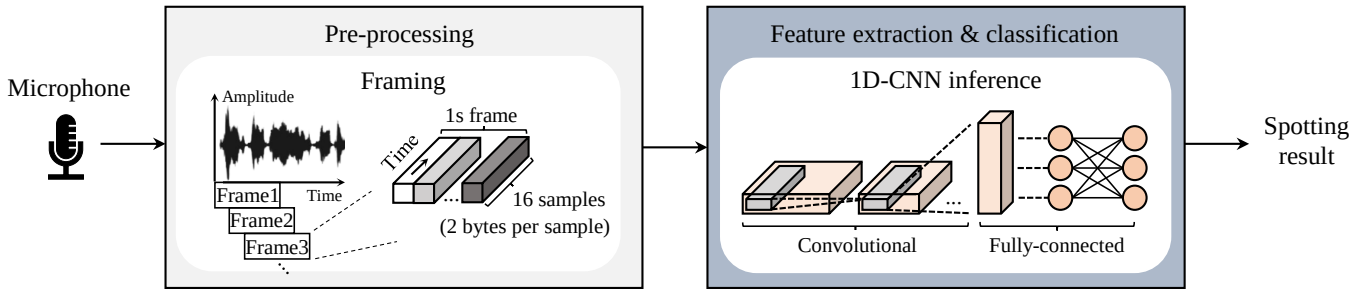

Figure 1. Overall process flow in the proposed KWS system.

The proposed system tries to find a keyword within each of the frames. The frames are formed from the stream data with a constant shift I, as illustrated in Figure 2, where I denotes the inter-frame shift. The inter-frame shift can be considered by the time budget given for the system to perform the KWS process for a frame before the next frame. Since I is set to be less than the length of each frame, the frames partially overlap each other, as illustrated in the figure. However, these frames may not correspond exactly to the real audio frame containing the keyword, which causes jitter between them. In the figure, the amount of the jitter is denoted by J. J is proportional to I and not greater than $I/2$ since the frames are formed as described above. Because the jitter contains unrelated information about a certain keyword, it can have a negative effect on spotting accuracy. It is, therefore, desirable to minimize the jitter by minimizing the inter-frame shift. However, this leads to a shorter time budget for the KWS process.

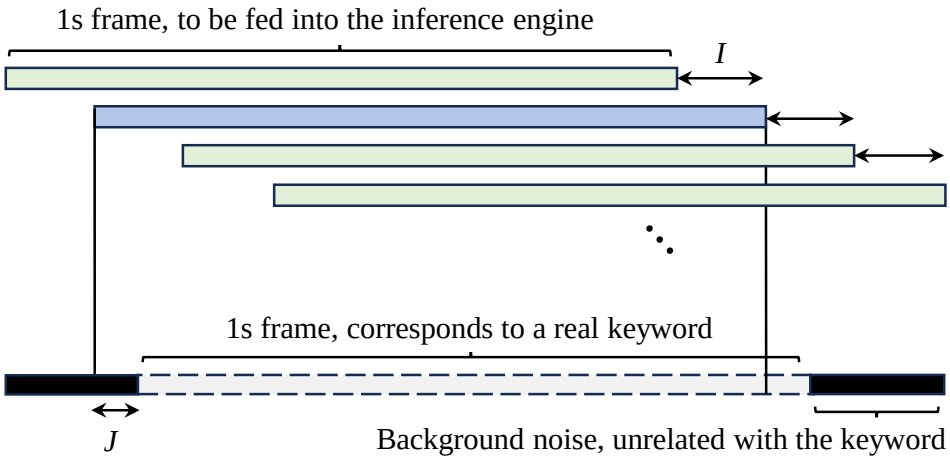

Figure 2. Framing in the proposed system, where *I* and *J* stand for the inter-frame shift and jitter, respectively.

3.2. Overall System Architecture and Processing Mechanism

The proposed system is designed based on the architecture shown in Figure 3 by including all the essential components needed to perform the KWS process. The components are integrated based on the 32-bit wide Avalon memory-mapped interconnect. The MCU serves as the host processor. We employed the Nios-II processor, a proprietary soft processor that is available in the FPGA we used for the implementation. The MCU contains the two master interfaces, which correspond to the instruction and data memories, respectively. The MCU controls the overall flow by orchestrating the other components. The 1D-CNN inference engine is composed of an inference core and scratch-pad buffers. The inference core inside the inference engine performs the inference process on a block-by-block basis. The scratch-pad buffers store features (**I**/**O**), weights (**W**), and thresholds (**T**), accessed by the core to perform the inference process. The audio CODEC converts analog to digital, where the audio stream is sampled as 16-bit data at a rate of 16 kHz. The performance counter precisely measures the processing time, taking the operating frequency into account, which is used to measure the latency of the KWS process.

The operating mechanism of the proposed KWS system is illustrated in Figure 4. The audio stream is sampled by the audio CODEC. The continuous data are efficiently grouped into frames based on the circular buffer, as shown in the figure. A frame is composed of 1024 segments, where each segment is composed of sixteen 16-bit samples and the sampling rate is 16 kHz; hence, a frame is composed of 16384 samples in 1.024 s. Each frame is copied into the scratch-pad buffer in the inference engine. The engine loads the parameters from the MCU's data memory to the scratch-pad buffer every time a block is computed. The core in the engine processes each block. The output feature map of each block is stored in the scratch-pad buffer in the engine and becomes the input feature map for processing the next block, without any MCU interventions.

3.3. Inference Engine

The 1D-CNN model is binarized based on XNOR-Net [17] and thus is called the 1D-BCNN model in this work. The 1D-BCNN model has a single dimension for each channel, as illustrated in Figure 4. As the weight and feature elements of the binarized model are ± 1, the memory footprint for their storage can be significantly reduced compared to that of the non-binarized models. In addition, the dot-product operations for the convolution can be performed based on the XNORPopcount (XPOP) operation. The dot-product operation involves a number of multiply-accumulate operations. Since each of the operands for the multiplications in the binarized model is represented by a single bit (either of ± 1), the multiplications can be performed on the basis of single-bit logical operations (XNOR-based) instead of complicated multi-bit arithmetic operations. In addition, because each multiplication result is also represented by a single bit, the accumulation can be

performed on the basis of counting the population. This XNOR-based multiplication, followed by the population counting operations, is fused into the so-called XNORPopcount (XPOP) operation.

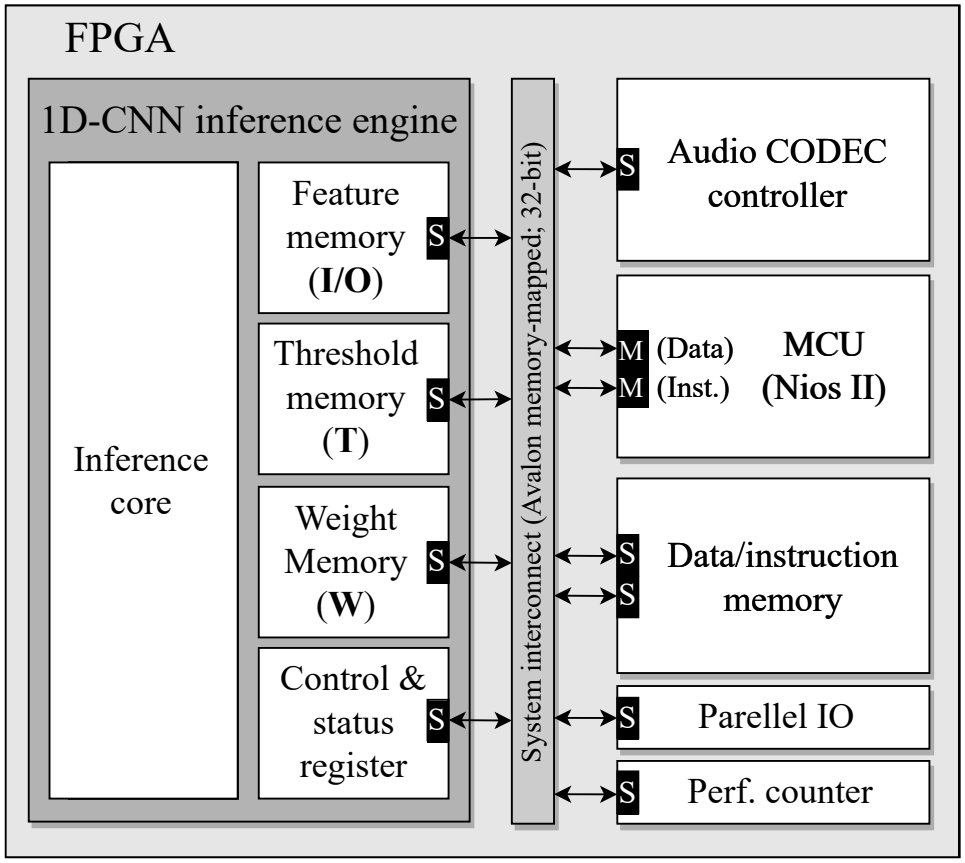

Figure 3. Overall architecture of the proposed system, where M and S stand for the master and slave interfaces, respectively.

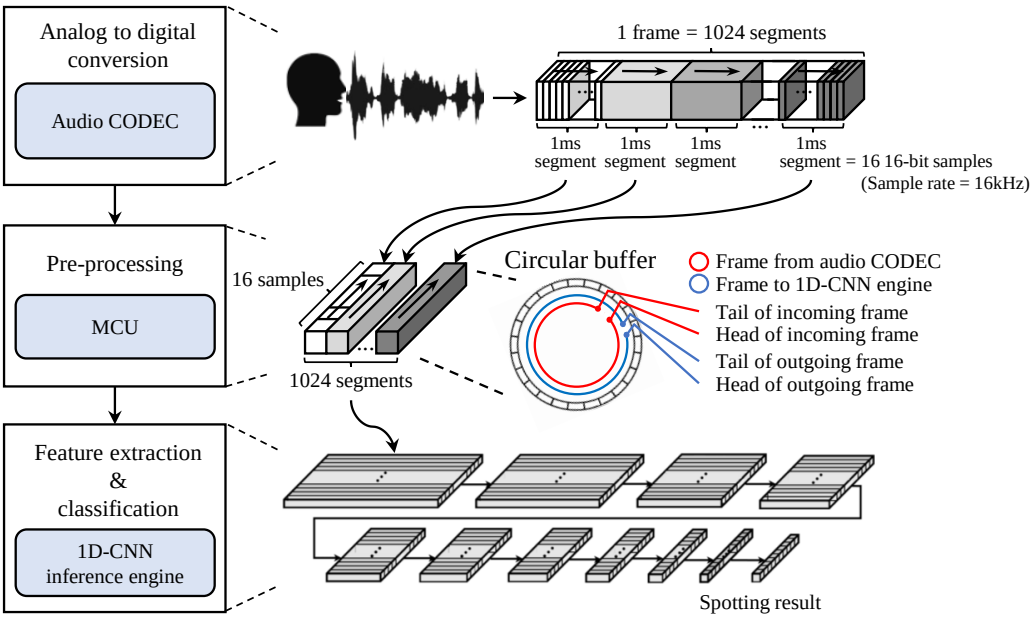

Figure 4. Processing procedure in the proposed system.

A dedicated inference engine is employed to perform the 1D-BCNN inference process. The 1D-BCNN inference engine is optimized to handle 1D audio data, based on the hardware architecture developed in our previous work [18]. The architecture in [18] was designed to effectively process 2D images using the operation-skipping technique inherent to images. The original architecture was modified by simplifying the convolution and pooling for 1D processing to reduce complexity. The convolution, which extracts 2D spatial features from images, was dimensionally reduced to extract 1D temporal features. The pooling, which downsamples 2D spatial features, was modified to downsample 1D temporal features. These simplifications for 1D processing can significantly reduce resources for implementation. This is because the counters for the induction variables used in iterating through dimensions can be eliminated, along with their control logic.

The inference engine processes each of the blocks that make up the model. Figure 5a shows the overall inference procedure, where the blocks share the same compute structure, as expressed below:

$$\mathbf{O} = \mathrm{Pool}(\mathrm{Sign}(\mathbf{I} \bullet \mathbf{W}, \mathbf{T})) \quad (1)$$

Here, **I** and **O** denote the input and output features, respectively, and **W** and **T** denote the weights and thresholds, respectively. Pool(·) and Sign(·) stand for the pooling and sign-activation functions, respectively, and • represents the convolution for the convolutional blocks or multiplication for the fully connected blocks. The thresholds can be precalculated with the batch normalization parameters and convolutional weights. They can also be precomputed with the parameters involved in the batch normalization layers and the average magnitude of the weight parameters, as presented in [18]. The features are extracted from the first few convolutional blocks, followed by keyword classification performed by the remaining fully connected blocks. Each pair of the 128-bit wide vectors is loaded from **I** and **W**, the XPOP operation is performed with them, and the result is accumulated. This processing is devised based on the output-oriented and output stationary dataflow [23], as shown in Figure 5b. It is processed sequentially and compared to a precalculated threshold, skipping subsequent tasks if true. In the pooling block, if the output element is 1, the remaining elements are skipped. This operation-skipping technique proposed in previous work [18] is employed to achieve high resource efficiency.

The microarchitecture of the inference engine is designed for the processing in Figure 5b. Figure 6 shows the microarchitecture that has a simple linear pipeline. In the first stage, addresses are generated with the counter values for accessing the memories. In the second stage, the data required for the XPOP operation are loaded from the scratch-pad buffers (corresponding to the **I** and **F** memories). In the third stage, the XPOP is computed with the weight and feature elements. In the fourth stage, the partial sum register is initialized with the threshold and updated by accumulating the XPOP result. The most significant bit (MSB) of the cumulative result is selected to implement the sign-activation function, as shown in Figure 5. The MSB is stored in the shift register. In the fifth stage, the output elements aggregated in the shift register are finally stored in the scratch-pad buffer (**O**). The entire pipeline is controlled by considering the possibilities of skipping operations according to the techniques described in Figure 5. The operation-skipping techniques can effectively increase inference speed by skipping 10.72% of the XPOP computations (during the entire KWS process).

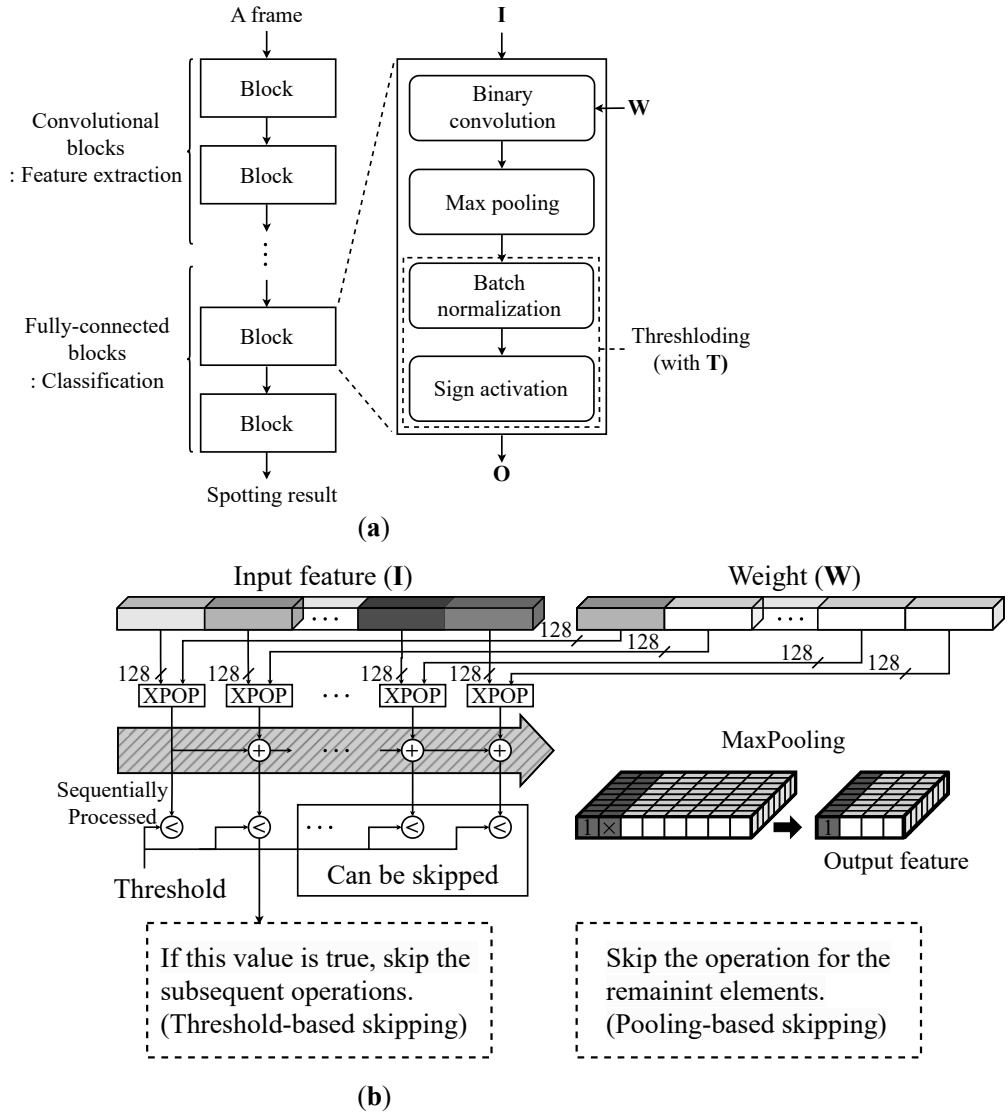

Figure 5. (**a**) BCNN model structure and (**b**) processing procedure in the inference engine incorporated in the proposed system.

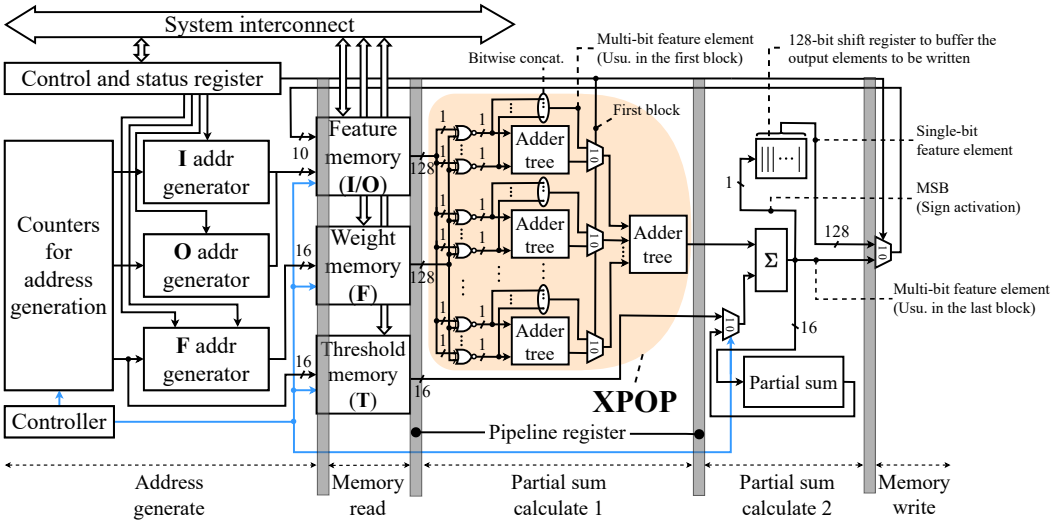

Figure 6. Microarchitecture of the inference engine in the proposed system.

4. Implementation Results and Discussion

The 1D-BCNN model used in the proposed system was evaluated by training for GSCD v2. GSCD v2 is composed of 65,000 frames, where the sampling rate is 16 kHz with 16-bit samples. The proposed KWS system considered the keywords "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", and "Unknown" for spotting. A KWS task with ten + unknown classes was also adopted in previous studies, such as [20], ensuring a fair comparison between other results and those of the proposed system.

The 1D-BCNN model used in the proposed system was designed based on the structure shown in Table 1. By binarizing the parameters, it only required 506 Kbits of memory to store the parameters. This was small enough to fit into the on-chip memory The model was trained using the ADAM optimizer, with the exponential moving average coefficient of the momentum estimated in the beta tuple set to 0.9 and the exponential moving average coefficient of the squared slope set to 0.999. The model was trained for 600 epochs with a batch size of 256, and the training curve is shown in Figure 7. The initial learning rate was set to 0.01, and if the validation accuracy did not improve for 50 epochs, the learning rate was reduced by a factor of 1/10 to optimize the model's performance.

Table 1. BCNN model structure designed for GSCD v2.

Block [a]	Input Channel	Input Size	Kernel	Pooling Stride
CV_1	1024	16	3	2
CV_2	512	128	3	2
CV_3	256	128	3	2
CV_4	128	128	3	2
CV_5	64	128	3	2
CV_6	32	128	3	2
CV_7	16	128	3	2
CV_8	8	128	3	2
FC_9	512	1	1	1
FC_{10}	128	1	1	1
FC_{11}	128	1	1	1

[a] CV*n* and FC*n* stand for the *n*-th convolutional and fully connected blocks, respectively.

To achieve high spotting accuracy even in a noisy environment, noise was added to the training dataset, resulting in an SNR between −5 dB and 10 dB. The test accuracy was measured by adding noise of varying SNRs to the test data, and the results are summarized in Table 2. The spotting accuracy was 92.94% when the SNR was 20 dB and 90.33% when the SNR was 0 dB. The confusion matrix in Figure 8 shows that the proposed system achieved high spotting accuracy even in noisy environments. Additionally, the test accuracy was measured by considering the jitter, and the results are summarized in Table 2. As shown in the table, the spotting accuracy decreased with a jitter of over 20 ms, based on which the inter-frame shift was determined to be 40 ms.

The proposed system was synthesized using Quartus Prime v18.1 targeting an Intel Cyclone V FPGA whose part name was 5CSEMA5F31C6N. The Terasic DE1-SoC board was employed for the implementation. We employed this board because the board is equipped with the components necessary for the realization of the KWS process (e.g., audio CODEC). The resource efficiency and real-time operation achieved by the system were not solely due to the merits of the board or device. The overall system was implemented with 6895 ALUTs, 1957 Kbits of on-chip memory, and 0 digital signal processor (DSP). Among these, the MCU and inference engine used 1569 Kbits and 388 Kbits of the on-chip memory, respectively, where the MCU's data memory was used for the storage of the parameters for inference. The latency required to perform the KWS process for a frame was 22 ms at an operating

frequency of 50 MHz, which was within the 40 ms inter-frame shift. Figure 9 illustrates the environment used to verify the functionality of the proposed system.

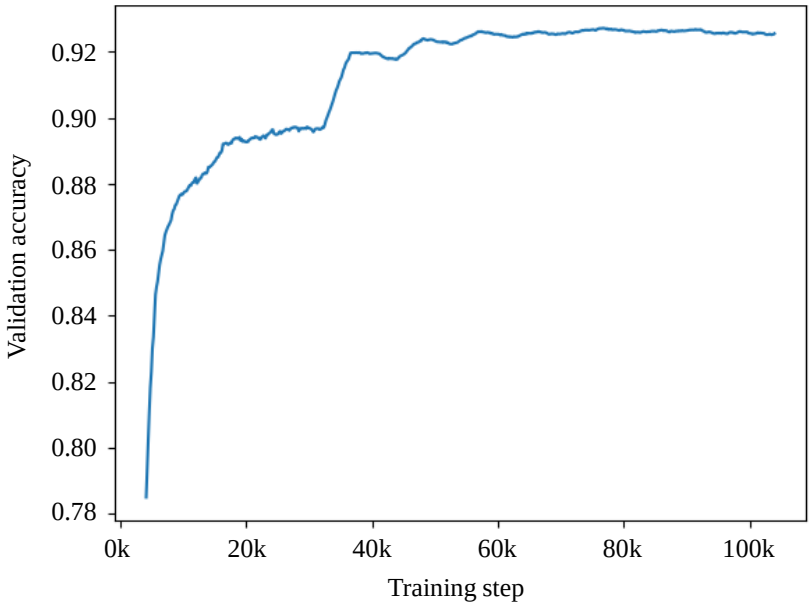

Figure 7. Training curve of the BCNN model for GSCD v2.

Table 2. Spotting accuracy for jitter and SNR.

Max Jitter/SNR	−5 dB	0 dB	10 dB	20 dB
0 ms	86.12%	90.33%	91.80%	92.94%
20 ms	80.70%	84.73%	86.51%	87.56%
50 ms	67.16%	72.85%	74.06%	74.21%
100 ms	64.81%	65.66%	66.33%	66.43%
500 ms	62.41%	62.86%	63.38%	63.46%

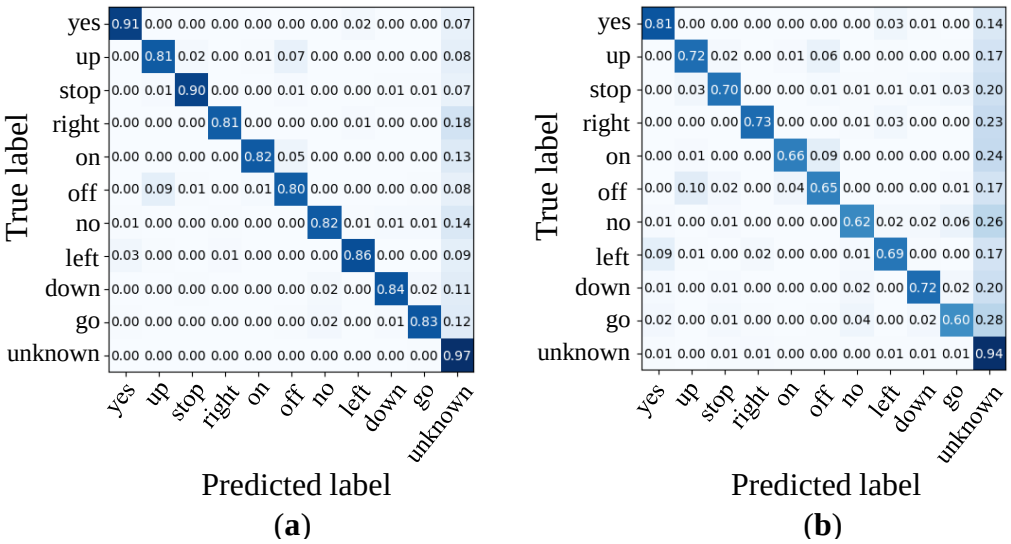

Figure 8. KWS results for the test set in GSCD v2, where the results were obtained under SNRs of (**a**) 20 dB and (**b**) −5 dB by the model trained with random noise.

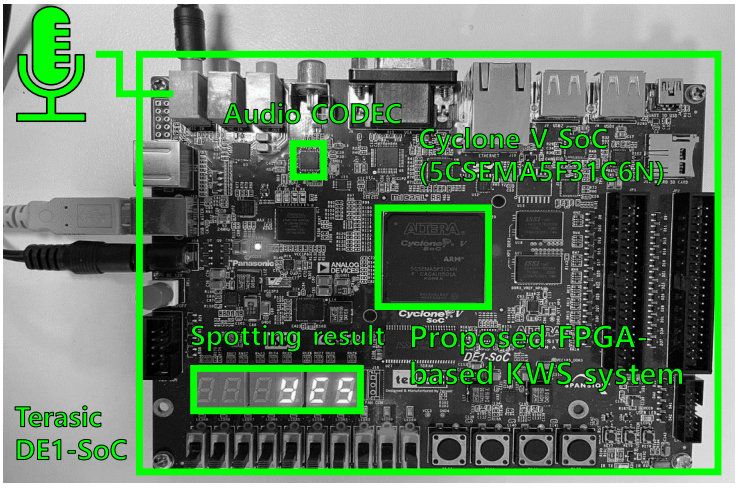

Figure 9. Demonstration environment setup. The demonstration video can be seen at https://abit.ly/kws (accessed on 18 September 2023).

The implementation results of the proposed system are compared with the previous results in Table 3. The KWS system in [5] demonstrated high spotting accuracy in an environment where the SNR was 10 dB, based on 1D-CNN-based classification with MFCC-based features. However, compared to the proposed system, it exhibited high resource usage, especially in the use of DSP. Assuming that one DSP is equivalent to 200 LUTs [24], the proposed system consumes about 60% fewer LUTs than the system in [5]. The KWS system proposed in [9] also showed high resource usage due to the MFCC calculation. It is noteworthy that such low resource usage in the proposed system may lead to low-leakage power consumption. The systems proposed in [6,10,11] were implemented in ASIC and consumed very low power, but their spotting accuracies were inferior to that of the proposed system but with longer latencies.

Table 3. Implementation results of the KWS systems.

KWS System	**This Work**	**[5]**	**[9]**	**[11]**	**[10]**	**[6]**
Implementation technology	Cyclone V (28 nm FPGA)	Zynq 7000 (28 nm FPGA)	Zynq UltraScale+ MP (40 nm FPGA)	22 nm CMOS (ASIC)	28 nm CMOS (ASIC)	40 nm CMOS (ASIC)
Processing	1D BCNN	MFCC 1D CNN	MFCC DS CNN	1D CRNN	MFCC DS CNN	MFCC LSTM
Latency	22 ms	81 ms	22 ms	30 ms	64 ms	32 ms
GSCD v2 acc. @ SNR	91.8% @ 10 dB 90.3% @ 0 dB	92.4% @ 10 dB 84.5% @ 0 dB	90.5% @ n.a.	91.7% @ 10 dB 88.2% @ 0 dB	91.7% @ n.a.	90.6% @ n.a.
# of keywords	11	12	10	5	4	10
Operating freq.	50 MHz	50 MHz	170 MHz	250 kHz	40 kHz	400 kHz
Power cons.	159.9 mW [a]	209.0 mW	n.a.	1.4 µW	0.51 µW	2.5 µW
Resource usage [b]	6895 (2384) LUTs, 0 (0) DSPs	n.a. (5575) LUTs, n.a. (64) DSPs [c]	27,315 (11,270) LUTs, 31 (26) DSPs	0.788 mm^2	0.23 mm^2	0.16 mm^2

[a] This is the power consumption of the entire system. [b] The number inside the parentheses corresponds to the usage of the inference engine and the number outside the parentheses corresponds to that of the entire system. [c] One DSP is known to be comparable to 200 LUTs [23].

Compared to previous works, the merits of the proposed system are reflected in the resource efficiency, as shown in Table 3. This high resource efficiency can be attributed to a few reasons. First, complicated processing for feature extraction, such as the MFCC-based features, was effectively eliminated as the convolutional blocks were trained to replace them. Second, the 1D-CNN model was binarized into a 1D-BCNN model to reduce the memory footprint required to store features and parameters and to efficiently implement the multiply-accumulate operations through XPOP operations. Third, some of the redundant

operations in the 1D-BCNN inference process were skipped by an efficient inference engine, which was modified from our previous work. Each of these techniques, contributing to the high resource efficiency, originated from previous studies and were employed independently; however, we applied them in practice for designing and implementing the KWS system, successfully demonstrating their validity.

5. Conclusions

This paper proposes a resource-efficient KWS system. The proposed system performs the entire KWS process relying on a single 1D-BCNN inference process. A resource-efficient engine executes the inference process with the operation-skipping techniques. As a result, the proposed system demonstrates 60% less resource usage than the previous state-of-the-art work, while its processing latency is as short as 22 ms per frame. The proposed system achieves a spotting accuracy of 91.80% for GSCD v2 in an environment with an SNR of 10 dB.

In future research, the proposed system can be extended to handle larger-scale tasks with more classes. To achieve high spotting accuracy for such tasks, it would be necessary to employ a more advanced DNN model with a complex data flow and/or multi-bit quantization instead of the linear data flow and binarization utilized in this work. This may require further improvement of the inference engine to support these enhancements. In addition, power consumption could be reduced by detecting non-active situations and disabling all or part of the system when no spoken keywords are detected.

Author Contributions: Conceptualization, J.Y., N.K., D.L., and T.-H.K.; methodology, J.Y., N.K., D.L., and T.-H.K.; software, J.Y., N.K., D.L., S.-J.L., G.-H.K., and T.-H.K.; validation, J.Y., N.K., D.L., and T.-H.K.; formal analysis, J.Y. and T.-H.K.; investigation, J.Y., N.K., D.L., and T.-H.K.; writing—original draft preparation, J.Y. and T.-H.K.; writing—review and editing, J.Y. and T.-H.K.; visualization, J.Y. and T.-H.K.; supervision, T.-H.K.; project administration, T.-H.K.; funding acquisition, T.-H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by ABOV Semiconductor (202200840001, Study of a micro NPU for a tiny MCU (Institute of Information and Communications Technology Planning and Evaluation (IITP)) grant) funded by the Korean government (MSIT) (2017-0-00528, The Basic Research Lab for Intelligent Semiconductor Working for the Multi-Band Smart Radar), and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korean government (MSIT) (2021R1F1A1059536). The EDA tools were supported by the IC Design Education Center.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. López-Espejo, I.; Tan, Z.H.; Hansen, J.H.; Jensen, J. Deep spoken keyword spotting: An overview. *IEEE Access* **2021**, *10*, 4169–4199. [CrossRef]
2. Han, W.; Chan, C.F.; Choy, C.S.; Pun, K.P. An efficient MFCC extraction method in speech recognition. In Proceedings of the IEEE International Symposium on Circuits and Systems, Kos, Greece, 21–24 May 2006; IEEE: Toulouse, France, 2006; p. 4.
3. Giraldo, J.S.P.; Verhelst, M. Laika: A 5 µW programmable LSTM accelerator for always-on keyword spotting in 65 nm CMOS. In Proceedings of the European Solid State Circuits Conference 2018—IEEE 44th, Dresden, Germany, 3–6 September 2018; IEEE: Toulouse, France, 2018; pp. 166–169.
4. Shan, W.; Qian, J.; Zhu, L.; Yang, J.; Huang, C.; Cai, H. AAD-KWS: A sub-µW keyword spotting chip With an acoustic activity detector embedded in MFCC and a tunable detection window in 28-nm CMOS. *IEEE J. Solid-State Circuits* **2022**, *58*, 867–876. [CrossRef]
5. He, K.; Chen, D.; Su, T. A configurable accelerator for keyword spotting based on small-footprint temporal efficient neural network. *Electronics* **2022**, *11*, 2571. [CrossRef]
6. Chong, Y.S.; Goh, W.L.; Nambiar, V.P.; Do, A.T. A 2.5 μW KWS engine with pruned LSTM and embedded MFCC for IoT applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *69*, 1662–1666. [CrossRef]

7. Krichen, M. Convolutional neural networks: A survey. *Computers* **2023**, *12*, 151. [CrossRef]
8. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6999–7019. [CrossRef]
9. Bae, S.; Kim, H.; Lee, S.; Jung, Y. FPGA implementation of keyword spotting system using depthwise separable binarized and ternarized neural networks. *Sensors* **2023**, *23*, 5701. [CrossRef] [PubMed]
10. Shan, W.; Yang, M.; Wang, T.; Lu, Y.; Cai, H.; Zhu, L.; Xu, J.; Wu, C.; Shi, L.; Yang, J. A 510-nW wake-up keyword-spotting chip using serial-FFT-based MFCC and binarized depthwise separable CNN in 28-nm CMOS. *IEEE J. Solid-State Circuits* **2020**, *56*, 151–164. [CrossRef]
11. Liu, B.; Cai, H.; Zhang, Z.; Ding, X.; Wang, Z.; Gong, Y.; Liu, W.; Yang, J.; Wang, Z.; Yang, J. More is less: Domain-specific speech recognition microprocessor using one-dimensional convolutional recurrent neural network. *IEEE Trans. Circuits Syst. Regul. Pap.* **2021**, *69*, 1571–1582. [CrossRef]
12. Yan, Y.; Stewart, T.C.; Choo, X.; Vogginger, B.; Partzsch, J.; Höppner, S.; Kelber, F.; Eliasmith, C.; Furber, S.; Mayr, C. Comparing Loihi with a SpiNNaker 2 prototype on low-latency keyword spotting and adaptive robotic control. *Neuromorphic Comput. Eng.* **2021**, *1*, 014002. [CrossRef]
13. Ceolini, E.; Anumula, J.; Braun, S.; Liu, S.C. Event-driven pipeline for low-latency low-compute keyword spotting and speaker verification system. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, UK, 12–17 May 2019; IEEE: Toulouse, France, 2019; pp. 7953–7957.
14. Prabhavalkar, R.; Alvarez, R.; Parada, C.; Nakkiran, P.; Sainath, T.N. Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, South Brisbane, QLD, Australia, 19–24 April 2015; IEEE: Toulouse, France, 2015; pp. 4704–4708.
15. Huang, Y.A.; Shabestary, T.Z.; Gruenstein, A. Hotword cleaner: Dual-microphone adaptive noise cancellation with deferred filter coefficients for robust keyword spotting. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, UK, 12–17 May 2019; IEEE: Toulouse, France, 2019; pp. 6346–6350.
16. Yoon, J.; Lee, D.; Kim, N.; Lee, S.J.; Kwak, G.H.; Kim, T.H. A real-time keyword spotting system based on an end-to-end binary convolutional neural network in FPGA. In Proceedings of the IEEE Symposium on Low-Power and High-Speed Chips, Tokyo, Japan, 19–21 April 2023; IEEE: Toulouse, France, 2023; pp. 1–3.
17. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: Imagenet classification using binary convolutional neural networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 525–542.
18. Lee, S.J.; Kwak, G.H.; Kim, T.H. TORRES: A resource-efficient inference processor for binary convolutional neural networks based on locality-aware operation skipping. *Electronics* **2022**, *11*, 3534. [CrossRef]
19. Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv* **2018**, arXiv:1804.03209.
20. López-Espejo, I.; Tan, Z.H.; Jensen, J. Exploring filterbank learning for keyword spotting. In Proceedings of the European Signal Processing Conference, Amsterdam, The Netherlands, 18–21 January 2021; IEEE: Toulouse, France, 2021; pp. 331–335.
21. Wang, Y.; Chong, Y.S.; Goh, W.L.; Do, A.T. Noise-aware and lightweight LSTM for keyword spotting applications. In Proceedings of the International SoC Design Conference, Gangneung-si, Republic of Korea, 19–22 October 2022; IEEE: Toulouse, France, 2022; pp. 135–136.
22. Mohanty, P.; Nayak, A.K. CNN based keyword spotting: An application for context based voiced Odia words. *Int. J. Inf. Technol.* **2022**, *14*, 3647–3658. [CrossRef]
23. Sim, J.; Lee, S.; Kim, L.S. An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm CMOS. *IEEE Trans. Very Large Scale Integr. Syst.* **2019**, *28*, 87–100. [CrossRef]
24. Chang, S.E.; Li, Y.; Sun, M.; Shi, R.; So, H.K.H.; Qian, X.; Wang, Y.; Lin, X. Mix and match: A novel FPGA-centric deep neural network quantization framework. In Proceedings of the IEEE International Symposium on High-Performance Computer Architecture, Seoul, Republic of Korea, 27 February–3 March 2021; IEEE: Toulouse, France, 2021; pp. 208–220.