2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP) | 979-8-3503-0245-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSP58490.2023.10248542

# Design and implementation of a speech recognition module based on RISC-V embedded processor

Can Ding[1,a]
[1] School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China
[a]e-mail: 202022172013166@gs.zzu.edu.cn

Yunfei Zhu[2,b]
[2] School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China
[b] e-mail: 202022172013312@gs.zzu.edu.cn

Rongcai Zhao[3,c*]
[3] School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China
* [c] Corresponding author: rczhao126@126.com

Xiao Zhang[4,d]
[4] School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China
[d]e-mail: 202011171010161@gs.zzu.edu.cn

***Abstract*- Due to the increasing demand for speech recognition in terminal devices, in order to speed up the speech recognition process in embedded processors and improve the speech recognition accuracy, and considering the disadvantages of solidity, high cost and inflexibility of application specific integrated circuit implementation, this paper proposes a speech recognition module based on RISC-V embedded processor, which is integrated into Hummingbird E203 processor, controlled by four custom instructions, and streamlined execution, taking advantage of the FPGA's strengths in table look-up and IP cores to speed up the speech recognition execution process. The recognition module operates at 16MHz with a total power consumption of 0.4W and a recognition time of 18ms, which is 2-3 times faster than similar designs and has been tested to achieve a 90% recognition accuracy rate.**

***Keywords*- RISC-V; speech recognition; MFCC; convolutional neural network; FPGA**

## I. INTRODUCTION

Speech as an important way to convey information, human understanding of speech is based on accumulated knowledge and experience, and how to make machines recognize and understand speech accurately has been a popular topic of research by experts and scholars, for which many theories and algorithms have been proposed, among which the mainstream view is to first extract the parameters reflecting speech features, and then use speech recognition models to make predictions. The existing methods of feature parameter extraction mainly include Linear Predictive Cepstral Coefficient (LPCC), Mel Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Cepstral Coefficients (PLP) [1], The Merle cepstral coefficients are used to simulate the non-linear perception of sound by the human ear, and have been used with good results. Speech recognition models include the well-established Dynamic Time Warping (DTW), Hidden Markov Model (HMM) and the rapidly developing neural network models that are still on the rise in recent years [2] .

Although speech recognition technology has been available for a long time, the original implementation method has high requirements for processor performance due to its inherently large computational volume, which is generally computed by high-performance processors. With the popularity of smart terminal devices such as cell phones and watches [3], it is difficult for low-frequency processors to support the original computational approach, and the demand for real-time processing of speech makes it necessary to make improvements in computing MFCC parameter methods and neural network inference for the embedded domain. Shan et al. implemented by application specific integrated circuit (ASIC) [4-6], but the development cycle is long, the investment cost is high, and the product iteration is slow, Wang Z et al. implemented by means of FPGA [7-10], but there is a lack of flexibility, programmability drawbacks, so the choice of general-purpose processors on the Extending a dedicated processor to achieve hardware acceleration, a more flexible approach, can be a good solution to the above problems [11, 12]. In this paper, we propose a speech recognition method based on RISC-V processors to make up for the lack of arithmetic power of embedded processors by adding hardware acceleration modules, while also retaining the flexibility of the processors to complement the development of speech feature extraction for RISC-V platforms.

## II. PRELIMINARIES

### *A. RISC-V instruction sets*

Because of the patent limitations of the existing mainstream MIPS and ARM architectures, and because they had become redundant over the years, Professor Krste Asanovic proposed the RISC-V architecture in 2010, a fifth-generation open-source instruction set architecture based on the Reduced Instruction Set (RISC) [13], which has gained widespread support in academia and industry because of its design for modularity, scalability and open interoperability features. RISC-V improves on the strengths and weaknesses of existing instruction sets by mandating the implementation of only the basic integer instruction subset RV32I as shown in Table 1, with the rest being optional extension modules M/A/F/D/C, which can be freely combined as required, making RISC-V suitable for any field of microprocessors. It also has the advantage of configurable general-purpose register banks, a regular number of instructions with only a few dozen, dedicated Load/Store instruction accesses, accesses to one element address at a time without self-incrementing, simplified branch jumps, static prediction, and no instruction condition code branch delay slots.

The Hummingbird E203 processor is an open source RISC-V processor developed by Corelai Technology [14], which also

has the advantages of the RISC-V architecture and can be configured for the RV32IMAC architecture, with area power consumption and performance comparable to ARM's Cortex-M0+ processor cores. The processor implements general-purpose processor functions, but also has a reserved co-processor interface that allows for easy functional expansion, and this paper is about implementing a speech recognition acceleration module through a co-processor.

TABLE 1. MODULAR INSTRUCTION SET FOR RISC-V

| Instruction sets | Description |
|---|---|
| RV32I | Base Integer Instruction Set,32-bit |
| RV32E | Base Integer Instruction Set(embedded),32-bit,16registers |
| RV64I | Base Integer Instruction Set,64-bit |
| RV128I | Base Integer Instruction Set,128-bit |
| Extension | |
| M | Standard Extension for Integer Multiplication and Division |
| A | Standard Extension for atomic Instructions |
| F | Standard Extension for Single-Precision Floating-Point |
| D | Standard Extension for Double-Precision Floating-Point |
| C | Standard Extension for Compressed Instructions |
| V | Standard Extension for Vector Operations |

*B. MFCC parameters extraction*

MFCC parameter extraction is the process of processing frame by frame from the input continuous audio signal and finally outputting the corresponding acoustic feature parameter values. As shown in Fig.1, the input speech signal is first pre-emphasised to compensate for the high frequency loss; the short time signal has the characteristic of smoothness, so the signal is framed; to avoid causing frequency leakage, the Hamming window function is added for smoothing; the time domain signal is converted to frequency domain value by FFT transform; the filter group value is obtained by Mel filter transform; the logarithmic operation is performed on the group value to compress the data; finally, the discrete cosine transform is performed The MFCC eigenvalue data is obtained and finally fed into a convolutional neural network for prediction.

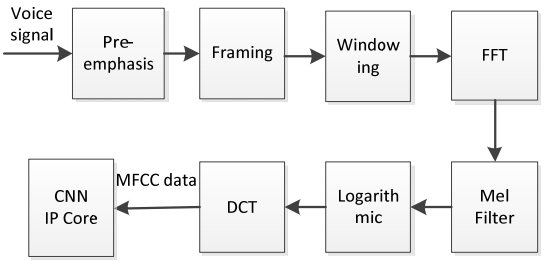


Figure 1. MFCC flow chart.

*1)Pre-emphasis:*The high frequency component of the sound is heavily attenuated by human physiology, so pre-emphasis is required to enhance the high frequency component of the signal to reduce its impact. Pre-emphasis is essentially a high-pass filter, with a transfer function expressed in the time domain as in (1).

$$s(t) = x(t) - \alpha * x(t-1) \tag{1}$$

Where α is the pre-emphasis coefficient and takes the value range [0.9, 1], x(t) is the tth data of the original speech signal and x(t-1) the t-1th data of the original speech signal.

*2) Splitting frames, adding windows:*Framing is the division of the speech signal into segments of data at a specific time, which is generally considered stable for a short period of time, while there is an overlap of 1/3 to 1/2 between each frame for continuity. The frame length and overlap length can be adjusted as required to obtain the best results.

Windowing is the multiplication of the framed data by a windowing function to eliminate the effects of endpoint discontinuities caused by framing, the formula for windowing is shown in (2).

$$x(k) = s(k) * \omega(k), (0 \le n \le N-1) \tag{2}$$

Where N is the frame length, x(k) is the frame data, ω(k) is the window function, and the window functions are mainly rectangular windows, Hamming windows and Hanning windows [15]. Among them, the Hamming window is used in this paper because of its better effect, and its equation is as in (3).

$$\omega(k) = 0.54 - 0.46cos\left(\frac{2\pi k}{N-1}\right), (0 \le k \le N-1) \tag{3}$$

*3) FFT:*The Fast Fourier Transform (FFT) is an efficient implementation of the Discrete Fourier Transform (DFT) [16] and is widely used in science and engineering. the FFT is an important part of extracting the MFCC parameters by which the time domain signal is quickly and efficiently converted to frequency domain data. A data sequence of length N, x(n), with a discrete Fourier transform X(k) can be expressed in (4).

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \tag{4}$$

where x(n) is the time domain data and N is the number of sampling points,$W_N^{kn} = e^{-j\frac{2\pi}{N}nk}$. If calculated directly using (4), the time complexity is $N^2$, while the FFT takes advantage of the symmetry and periodicity of the DFT and uses butterfly operations [17]to reduce the time complexity to NlogN.

*4) Mel filter:* The Mel filter transform chooses M filter banks to filter frequencies, which mimics the human auditory characteristic of being sensitive to low frequencies [18], so the filter is dense at low frequencies and sparse at high frequencies. mel frequency is transformed with frequency f as in(5), in Hz.

$$Mel(f) = \lg(1 + \frac{f}{700}) \tag{5}$$

The Mel filter is calculated as shown in (6).

$$s(m) = \sum_{k=0}^{N-1} |X(k)|^2 H_m(k)\,, \quad (0 \le k \le M) \tag{6}$$

where X(k) denotes the FFT transformed data, m denotes the Mel filter bank number, Hm (k) is the filter bank and its transfer function is as in (7).

$$H_m(k) = \begin{cases} 0 \quad , k < f(m-1), k > f(m+1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)} , f(m-1) \le k < f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)}, f(m) \le k \le f(m+1) \end{cases} \quad (7)$$

*5) Taking logarithms and discrete cosine transforms:* The Mel filter results are logarithmically manipulated to compress the data, and the formula is shown in (8).

$$l(m) = \ln\left(\sum_{k=0}^{N-1}|X(k)|^2 H_m(k)\right), (0 \le k \le M) \quad (8)$$

Subsequently, in order to obtain low frequency data and data dimensionality reduction and data volume reduction, DCT transformation was performed to improve the speech recognition accuracy, and the formula is shown in (9).

$$M(n) = \sum_{m=0}^{N-1} l(m) \cos\left(\frac{\pi n(m-0.5)}{M}\right), n = 1,2,3,\dots,L \quad (9)$$

where M (n) is the MFCC eigenvalue and L is the order of the DCT.

### *C. Convolutional neural network models*

The Convolutional Neural Network (CNN) is a special neural network structure, mainly used for image and video data processing, but also widely used in the field of speech recognition, where the convolutional, activation and pooling layers have similar roles to those in image processing, but with certain adaptations for the characteristics of speech signals.

In speech recognition, the input signal is usually a segment of the speech waveform in the time domain, which usually needs to be transformed into a form such as a spectrogram or Mel frequency cepstrum coefficients in order to extract the frequency domain features of the speech. Therefore, in CNNs, the convolution layer usually uses one-dimensional convolution (i.e. convolution in the time dimension) to perform convolution operations on feature maps such as spectrograms or MFCCs to extract local features in the time domain.

The activation and pooling layers serve similar roles as in image processing, where commonly used activation functions include ReLU, sigmoid, tanh, etc. The pooling layer usually employs maximum pooling or average pooling, etc. for feature dimensionality reduction.

The above are the three basic components of CNN. By stacking and combining different convolutional, activation and pooling layers, convolutional neural networks with different number of layers and different structures can be constructed. Long Short-Term Memory (LSTM), which can effectively handle the temporal relationships of speech signals. More complex speech recognition models, such as Deep Convolutional Recurrent Neural Network (DCRNN), can also be constructed by connecting multiple CNN and RNN network structures.

## III SOFTWARE AND HARDWARE IMPLEMENTATION

### *A. Software implementation*

The Nuclei Software Platform (NSP) includes the Nuclei Microcontroller Software Interface Standard (NMSIS) and a software development kit for the Nuclei processor core ( NMSIS is based on ARM's open source CMSIS framework and includes three major components - Core, DSP and NN - that support floating point, fixed point and neural networks. reusability and shorten the development cycle.

The C program for calculating MFCC parameters and neural network inference is written on the NSP platform, compiled into machine instructions, uploaded into the program, and then executed using the arithmetic power of the Hummingbird processor, each step of the calculation needs to go through a complex execution process, but due to insufficient hardware performance and limited software optimisation, it cannot be designed as a pipeline to parallel processing, so the running speed is very slow, processing 1 It takes more than 3 seconds to process 1 second of speech data, which is not enough to meet the real-time requirements. Therefore, this paper chooses the software and hardware collaborative approach, using the software to achieve process control, receiving the speech data into memory, then calling the MFCC feature parameter calculation module, and finally sending it to the neural network module to predict, and finally outputting the prediction results.

### *B. Coprocessor acceleration*

*1) General Architecture:* The MFCC parameter calculation module and neural network inference are implemented within the co-processor of the Hummingbird E203 processor, with data accessed and results returned via the co-processor EAI interface. Using the custom instructions mentioned above, the E203 processor controls the switching on and off of the modules and the overall system architecture is shown in Fig.2. The data is in fixed-point numbers and the modules are calculated with a hierarchical quantization of the data according to the size of the calculation. The Mel filter transform has a multiplicative accumulation calculation and the data can overflow, so it is widened to 32 bits and 16 bits for the remaining modules. Although floating point numbers can represent a greater range and accuracy, but considering MFCC involves a large number of addition, subtraction, multiplication and division operations, in the FPGA operation, can only use IEEE754 floating point numbers, can not be directly operated, is not conducive to optimization, and change to fixed point numbers can be directly added and subtracted and the use of shift to replace part of the advantages of multiplication and division calculation, the calculation speed will be accelerated.

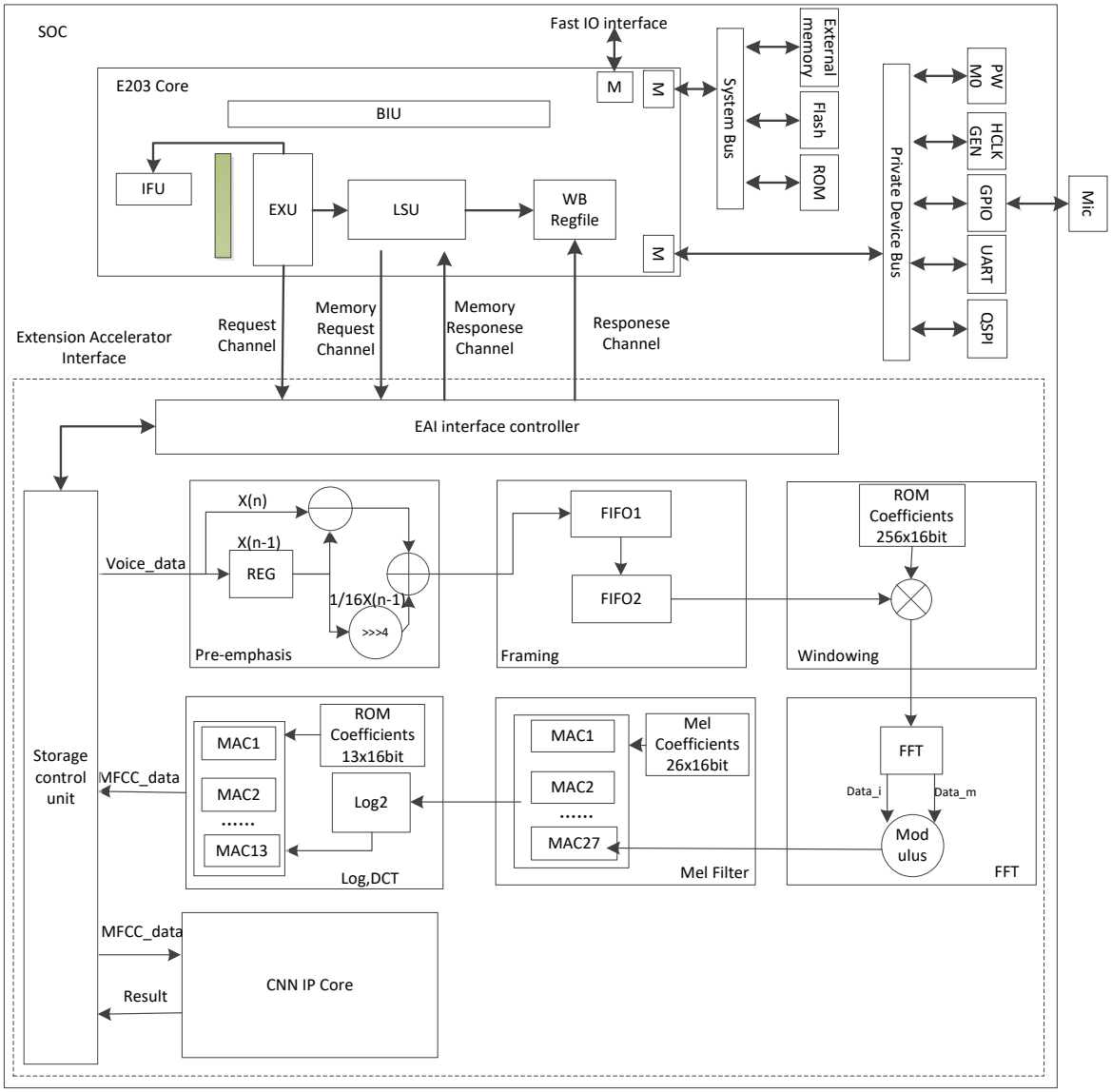


Figure 2. Top architecture of the system.

*2) Custom instructions:* Four custom instructions are designed to be extended according to the Custom instructions reserved for the RISC-V architecture, and when found to be extended instructions when executed by the processor EXU, they are dispatched directly to the co-processor for decoding and execution. The specific instruction coding format is shown in Fig.3. The lower 7 bits of the instruction are the opcode code field, with 4 sets of reserved combinations corresponding to the 4 Custom instructions. funct3 consists of xs1, sx2 and xd, which control whether the source registers rs1, rs2 need to be read and the destination register rd needs to be written respectively. funct7 field can be used as an additional coding extension to customise more instruction entries.

This article uses Custom3 instruction with opcode 111011 as shown in Fig.3. Custom MFCC_DADDR instruction is used to specify the starting address and length of voice data, xs1 and xs2 are 1 and need to read operands rs1 and rs2, xd is 0 and does not need to write back to rd, funct7 is specified as 0001000.

The custom MFCC_DSTORE instruction is used to specify the starting address and length of the MFCC data to be stored. xs1 and xs2 are 1, indicating that the operands rs1 and rs2 are read, xd is 0, no need to write back to rd, and funct7 is specified as 0010000.

The custom MFCC_START instruction is used to read the speech data in memory according to the start address, then send it to the MFCC module interface to execute the calculation and wait for the result to be written to memory when the execution is completed. xd is 1, which requires writing back the instruction execution result, xs1 and xs2 are 0, do not read operands rs1 and rs2, and funct7 is specified as 0100000.

The custom CNN_START instruction is used to read the MFCC data in memory according to the start address, then send it to the CNN module interface to execute the calculation and wait for the result to be written to memory when the execution is complete. xd is 1, the result of the instruction execution needs to be written back, xs1 and xs2 are 0, do not read the operands rs1 and rs2, and funct7 is specified as 1000000.

The Nuclei SDK software platform uses asm inline assembly for the execution of custom instructions. Pseudo instructions can be represented as ".insn r opcode,func3,func7,rd,rs1,rs2"; of course it is also possible to modify the GCC compiler toolchain's Binutils can also be modified to support the compilation of custom instructions, encapsulating functions into library functions to be called and executed.

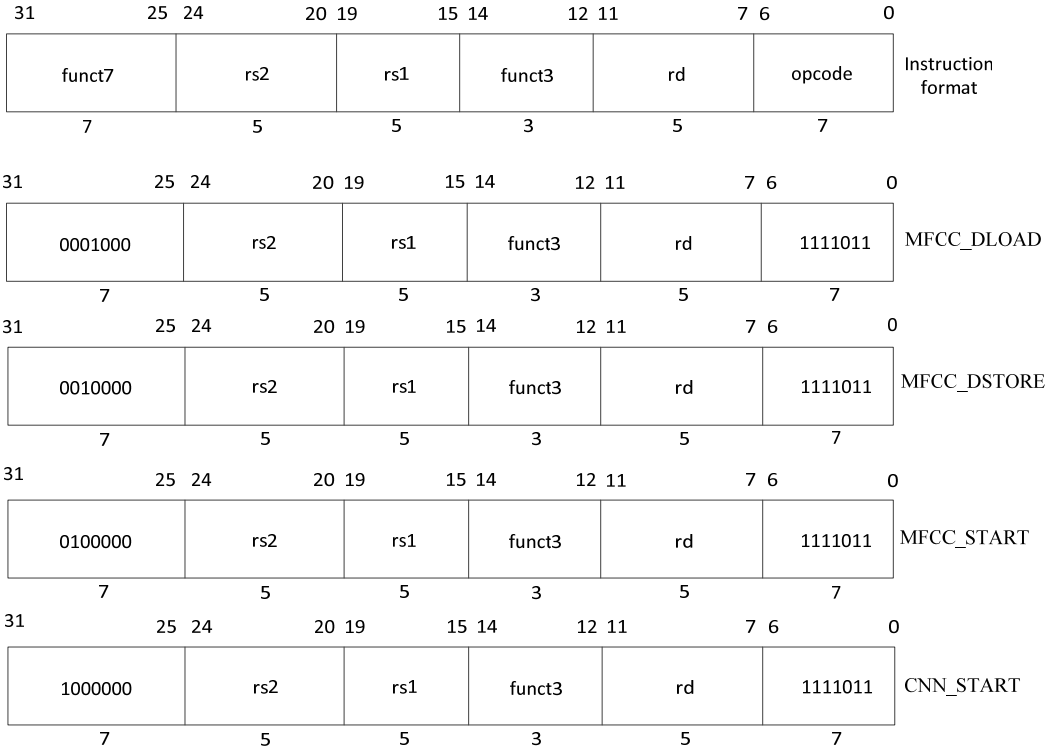


Figure 3. Custom extended commands.

*3) MFCC parameter extraction specific design:* According to the (1) can be seen, pre-emphasis is to let similar two signal values to do subtraction, so that the pre-emphasis coefficient α = 0.9375 = 15/16, then the original formula can be converted to formula (10), now only adder and shift register can be used to achieve, reduce the complexity of the calculation.

$$S_{(n)} = x(n) - x(n-1) + \frac{1}{16}x(n-1) \tag{10}$$

The thesis uses a frame length of 512, a frame shift of 256 and an overlap of 1/2 between frames. When the length of the first frame reaches 512, the data is read out into FIFO2, and later, when 256 points of data are received each time, it can be read out into FIFO2 and sent to the next level.

Window addition is mainly a multiplication of a frame of data by a Hamming window. Since the Hamming window is fixed after the frame length has been determined, the window coefficients can be calculated in advance by (3), and because of the symmetry of the Hamming window, the 256 window coefficients are stored in ROM and multiplied directly with the frame data by counter control. Since no calculation operation is done for splitting the frame, it is possible to move the addition of the window forward to be implemented with the pre-emphasis and then split the frame with no effect on the result.

For efficiency and security reasons, the FFT is calculated using the Fast Fourier Transform V7.1 IP core developed by Xilinx, as shown in Fig.4, with a transform length of 512 and a 32-bit input data pin s_axis_data_tdata, where the low 16-bit real part is the data after adding the window, and the high 16-bit imaginary part is directly assigned to 0. The FFT result output pin m_axis_data_tdata is also 32 bits wide, with the real and imaginary numbers in the high and low 16 bits respectively. As the result of the FFT calculation is a complex number, the result of the FFT calculation is modulo, squared and summed to the short time energy of the frame.


Figure 4. FFT transformation IP core.

The Mel filter transform essentially multiplies the 26 Mel filter banks by the FFT transformed spectral data and accumulates them. after the frame length is determined, the filter banks can also be calculated in advance according to (7), the filter bank data is stored in ROM, and the 26 MAC units are used to do parallel multiplication and accumulation calculations through counter control, and the 27th MAC unit calculates the frame energy separately. the MAC is the multiplication and accumulation The units are shown in Fig.5, data_a is the FFT data, data_b is the data stored in ROM, and acc is the accumulation register.

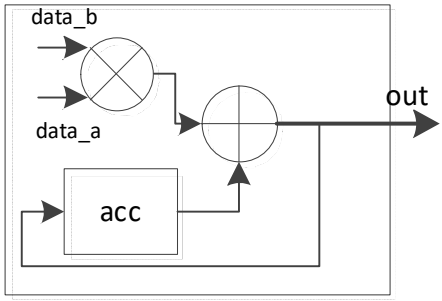


Figure 5. Multiply and accumulate unit.

The logarithm is taken in order to limit the data. If the logarithm with e as the base is used, the CORIC (Coordinate Rotation Digital Computer) algorithm needs to be used, which is highly accurate but computationally intensive. In order to simplify the circuit, the logarithmic operation with 2 as the base is used, which transforms the original calculation into an operation to find the first bit as 1 from the highest bit, using the dichotomous lookup.

The discrete cosine transform is a transfer of frequency domain data back to the time domain, which is essentially also logarithmic data multiplied by cosine data and accumulated, and the cosine data is also related to the frame length. The data is calculated in advance according to (9), stored in ROM, and calculated using 13 MAC units controlled by counters to do parallel multiplication and accumulation, and after experiments, the DCT 1st order data has very little effect on speech recognition, which is replaced by frame energy, and the MFCC feature parameters are finally output.

Since the MFCC calculation process is calculated independently between each module on a frame by frame basis, with dependencies on the higher level only at the interface, the MFCC parameter extraction process is designed as a pipeline approach as shown in Fig.6 [19], and as the amount of processed data increases, the advantages of increasing the pipeline extraction of MFCC parameters in this paper will be more obvious, which is not available in the E203 processor.

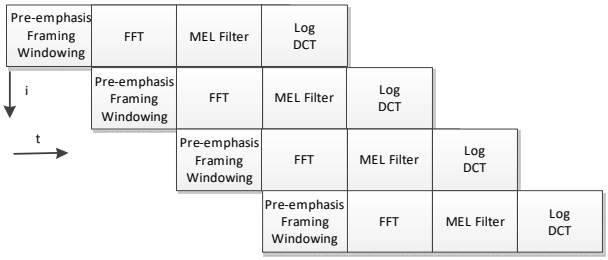


Figure 6. MFCC parameter extraction pipeline.

*4) Convolutional neural network specific implementation:* In this paper, it is found that even the simplest convolutional neural network can achieve good results by extracting speech features through MFCC, and the recognition accuracy can reach more than 90%, if we want to improve the accuracy, we need to stack convolution or match other models, which will increase the computational complexity and is not conducive to the deployment of the model. So this paper builds a simplified convolutional neural network model, mainly consisting of two convolutional layers, Relu activation function layer and maximum pooling layer, and one fully connected layer, the specific structure is shown in Table 2.

TABLE 2. CONVOLUTIONAL NEURAL NETWORK STRUCTURE

| Layers | Name | Output size |
|---|---|---|
| 1 | Conv(5*5,4) | (28*28,4) |
| 2 | Relu | (28*28,4) |
| 3 | MaxPool(2*2) | (14*14,4) |
| 4 | Conv(5*5,8) | (14*14,8) |
| 5 | Relu | (14*14,8) |
| 6 | MaxPool | (7*7,8) |
| 7 | FullConnect | (6) |
| 8 | SoftMax | 1 |

Because the model weights and biases are fixed values after the neural network is trained, each prediction is performed as forward inference, which involves a large number of multiplicative accumulation calculations that are difficult for the Hummingbird E203 processor to meet. Moreover, due to the natural parallelism of convolutional computing, FPGAs are suitable for parallel and pipelined operations, which makes it possible for FPGAs to accelerate the inference process of convolutional neural networks.

In this paper, the forward inference process of the neural network is implemented in C. High-level synthesis is performed

using the Vitis HLS tool, as shown in Fig.7. The design flow finally generates the convolutional IP cores, and HLS performs parallel optimisation of the code loops when generating the IP cores. The generated IP core is mounted to a co-processor, which is controlled by the E203 processor through configuration parameter instructions and start-up instructions, feeding the MFCC feature parameters into the IP core for internal processing and finally returning the prediction results.

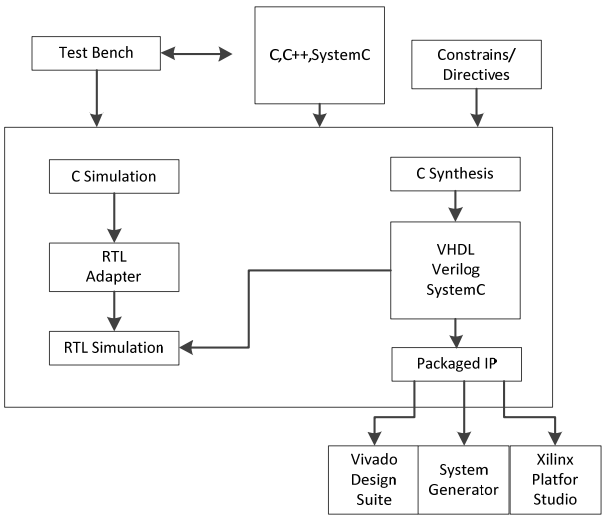


Figure 7. HLS design flow

## IV Results and Evaluation

### *A. Accuracy test*

In order to test the effect of speech recognition, this paper used the audio file dataset provided by Google Speech Command. 6 short words from this dataset were selected and a total of 709 data were tested. The results are shown in Table 3, where the rows represent the true tag values and the columns represent the model predictions.

TABLE 3. Speech recognition test results

| | down | left | right | up | yes | no |
|---|---|---|---|---|---|---|
| down | 101 | 1 | 3 | 4 | 2 | 7 |
| left | 1 | 104 | 2 | 3 | 6 | 2 |
| right | 0 | 6 | 107 | 3 | 2 | 0 |
| up | 1 | 5 | 2 | 105 | 1 | 4 |
| yes | 0 | 2 | 0 | 0 | 105 | 1 |
| no | 9 | 3 | 1 | 1 | 0 | 115 |

Although the accuracy of this paper is reduced in the implementation of the MFCC extraction module data using a fixed-point approach, the neural network model is fault-tolerant and has less impact on the recognition results. From Table 2 it can be concluded that the number of successful predictions equals a value of 637 on the diagonal, and dividing the number of successful predictions by the total number of tests yields an average word recognition accuracy of approximately 90%.

### *B. Resource and performance analysis*

The Vivado2021 platform was simulated and synthesised and compared with work from previous papers to assess factors such as the overall resources and performance of the system. Where the system resource consumption is shown in Table 4.

TABLE 4. System resource consumption table

| Resource modules | E203 processor | This paper |
|---|---|---|
| LUT | 10900 | 32010 |
| FF | 7082 | 20283 |

The table above shows that the co-processor consumes 74% of the overall LUT resources and 74% of the overall FF register resources due to the need to use the cache to store temporary data during the calculation process. Compared to the E203 processor, the co-processor takes up more resources, but it is impossible to achieve real-time speech recognition simply by relying on a low-frequency processor like the E203. The co-processor increases the processing efficiency to the millisecond level, which has obvious advantages in the speech recognition process, and the amount of its increased resources is within the acceptable range.

This paper was also analysed in comparison with the work done in the literature 6, 10, 20 [20] and the results are shown in Table 5.

TABLE 5. comparison table for different implementations

| Architecture | literature 20 | literature 6 | literature 10 | This paper |
|---|---|---|---|---|
| Platform | STM32F7 | ASIC | Zynq-7000 | Xc7k325t |
| Core | Contex-M7 | - | - | E203 |
| Frequencey (MHz) | 216 | 16 | 100 | 16 |
| Accuracy (%) | 87 | 95.6 | 96.67 | 90 |
| Latency (ms) | 680 | ⩽50 | 40 | 18 |
| Power(w) | - | 0.0264 | - | 0.401 |

As we can see from the table above, this paper increases the speech recognition speed to 18ms with limited additional resources and power consumption. Comparing to literature 20, the speech recognition speed is increased by 37 times and the accuracy is increased by 3% to 90% with the E203 itself not performing as well as the M7 processor. Comparing to literature 6,10, it also improves the speed by a factor of 2-3 with limited loss of accuracy and retains the flexibility and versatility of the system, while meeting the needs of embedded speech recognition for low power consumption and real time.

## V CONCLUSIONS

This paper describes the implementation of MFCC speech feature extraction and convolutional neural network prediction using a Hummingbird E203 processor and co-processor acceleration. To improve computational efficiency, the MFCC part adds hardware circuit support for operations based on its algorithmic features, including adding windows, Mel filter banks and discrete cosine transforms, and is implemented by table look-up and parallel computing, and the convolutional neural network part adds an IP core for forward inference. Due to the pipelined execution, the acceleration effect is stable even when processing large amounts of data, which meets the real-time processing requirements for speech recognition in embedded systems. Moreover, the generality of the speech recognition module makes it possible to mount it on other processors to achieve the acceleration effect as well, and the accuracy can be subsequently improved by optimising the convolutional neural network model.

## REFERENCES


[1] Këpuska, V. Z., Elharati, H. A. (2015) Robust speech recognition system using conventional and hybrid features of MFCC, LPCC, PLP, RASTA-PLP and hidden Markov model classifier in noisy conditions. Journal of Computer and Communications, 3: 1-9.

[2] Xiang, L., Lu, S., Wang, X., Liu, H., Pang, W., Yu, H. (2019) Implementation of LSTM accelerator for speech keywords recognition. In 2019 IEEE 4th International Conference on Integrated Circuits and Microsystems (ICICM) . Beijing. pp. 195-198.

[3] Schiavone, P. D., Rossi, D., Di Mauro, A., Gürkaynak, F. K., Saxe, T., Wang, M., ... Benini, L. (2021) Arnold: An eFPGA-augmented RISC-V SoC for flexible and low-power IoT end nodes. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29:677-690.

[4] Shan, W., Yang, M., Xu, J., Lu, Y., Zhang, S., Wang, T., ... Seok, M. (2020, February). 14.1 A 510nW 0.41 V low-memory low-computation keyword-spotting chip using serial FFT-based MFCC and binarized depthwise separable convolutional neural network in 28nm CMOS. In 2020 IEEE International Solid-State Circuits Conference-(ISSCC). San Francisco.pp. 230-232.

[5] Nguyen, T. C., Pham, L. D., Nguyen, H. M., Bui, B. G., Ngo, D. T., Hoang, T. (2016) A High Performance Dynamic ASIC-Based Audio Signal Feature Extraction (MFCC). In 2016 International Conference on Advanced Computing and Applications (ACOMP) . Can Tho.pp. 113-120.

[6] Wu, L., Wang, Z., Zhao, M., Hu, W., Cai, Y., Huang, R. (2021) A high accuracy multiple-command speech recognition asic based on configurable one-dimension convolutional neural network. In 2021 IEEE International Symposium on Circuits and Systems (ISCAS) . Daegu .pp. 1-4.

[7] Jo, J., Yoo, H., Park, I. C. (2015) Energy-efficient floating-point MFCC extraction architecture for speech recognition systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 24: 754-758.

[8] Glittas, A. X., Gopalakrishnan, L. (2021) A low latency modular-level deeply integrated MFCC feature extraction architecture for speech recognition. Integration, 76: 69-75.

[9] Wang, Z., Zha, W., Chai, J., Liu, Y., Xiao, Z. (2021) Lightweight Implementation of FPGA-Based Environmental Sound Recognition System. In 2021 International Conference on UK-China Emerging Technologies (UCET) . Chengdu.pp. 59-66.

[10] Hu Z. China.(2021) The design and FPGA verification of speech command recognition algorithm for assisted driving.https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD202201&filename=1022468751.nh

[11] Lee, S. Y., Hung, Y. W., Chang, Y. T., Lin, C. C., Shieh, G. S. (2021) RISC-V CNN coprocessor for real-time epilepsy detection in wearable application. IEEE transactions on biomedical circuits and systems, 15: 679-691.

[12] Tang, W.,Zhang, P. (2022) GPGCN: A General-Purpose Graph Convolution Neural Network Accelerator Based on RISC-V ISA Extension. Electronics, 11:3833.

[13] Waterman, A., Lee, Y., Patterson, D. A., Asanovi, K. (2014) The risc-v instruction set manual. volume 1: User-level isa, version 2.0. California Univ Berkeley Dept of Electrical Engineering and Computer Sciences. http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html.

[14] Song, Y., Jia, B., Yang, B., Zhang, P. (2022) Hummingbird E203 RISC-V processor core-based traffic flow detection system design. In 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA).Dalian.pp. 823-826.

[15] Hannah, A. A., Agordzo, G. K. (2020) A Design of a low-pass FIR filter using Hamming Window Functions in Matlab. Comput. Eng. Intell. Syst, 11: 24-30.

[16] Tiwari, A., Pandey, S. (2016) Implementation of Fast Fourier Transform in Verilog. International Journal of Engineering and Management Research (IJEMR), 6: 35-40.

[17] Yang, C., Xie, Y. Z., Chen, L., Chen, H., Deng, Y. (2015) Design of a configurable fixed-point FFT processor.In:IET International Radar Conference 2015. Hangzhou.pp. 1-4.

[18] Upadhya, S. S., Cheeran, A. N., Nirmal, J. H. (2019) Discriminating Parkinson diseased and healthy people using modified MFCC filter bank approach. International Journal of Speech Technology,22: 1021-1029.

[19] Li, Z., Hu, W., Chen, S. (2019) Design and implementation of CNN custom processor based on RISC-V architecture. In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Zhangjiajie. pp. 1945-1950.

[20] Xu,Y., Huang,L. (2019) Design of embedded off-line speech recognition system based on deep learning.Cyber Security And Data Governance, 38: 67-70.