IMPLEMENTATION OF VOICE PROCESSING ALGORITHMS IN FPGA HARDWARE

José L. Gómez-Cipriano [1, 2], Roger P. Nunes [1], Dante A.C. Barone [1]

[1] Informatics Institute
Federal University of Rio Grande do Sul, Porto Alegre – Brazil

[2] Technological and Exact Sciences Institute
Feevale College, Novo Hamburgo-Brazil

ABSTRACT

Some speech recognition applications, like speaker verification, dialog recognition or speech to text transcription could require real time processing and a good precision. Other applications such as toys, automotive vehicles or portable machines still could aggregate portability and low-power requirements, in addition to physical compactness. A specific hardware could be a solution for this problem.

The current work proposes an architecture using hardware based in FPGAs, optimizing the pre-processing and parameter extraction for performing efficient speech recognition.

IMPLEMENTATION OF VOICE PROCESSIG ALGORITHMS IN FPGA HARDWARE

José L. Gómez-Cipriano [1, 2], Roger P. Nunes [1], Dante A.C. Barone [1]

[1] Informatics Institute
Federal University of Rio Grande do Sul, Porto Alegre – Brazil

[2] Technological and Exact Sciences Institute
Feevale College, Novo Hamburgo-Brazil

ABSTRACT

Some speech recognition applications, like speaker verification, dialog recognition or speech to text transcription could require real time processing and a good precision. Other applications such as toys, automotive vehicles or portable machines still could aggregate portability and low-power requirements, in addition to physical compactness. A specific hardware could be a solution for this problem.

The current work proposes an architecture using hardware based in FPGAs, optimizing the pre-processing and parameter extraction for performing efficient speech recognition.

1. INTRODUCTION

Most of existing speech recognition systems (SRS) is based in computer software. However, the requirement of a personal computer and a software system could be a disadvantage in some applications. For example, a full-computerized system is not economically viable for a washing machine or a TV device control. In such cases, specific hardware may be a solution.

Moreover, a hardware speech recognition system, with a datapath specification based in speech recognition algorithms, could be faster than a general-purpose processors implementation [1] [2].

The goal of this work is to implement functional blocks for a portable speech recognition system of isolated words, using FPGAs [3]. In the next sections each system component, is shown.

2. PRE-PROCESSING SYSTEM

Pre-processing has a pre-emphasis block, a frame division function and a windowing block.

The input is the voice signal, filtered by an anti-aliasing filter and converted to digital format. Voice signal was recorded in 16 bits RAW format and 11025Hz sampling rate.

2.1 Pre-emphasis Function

Digital voice signal S(n) is filtered by a pre-emphasis filter:

$$\tilde{S}(n) = S(n) - a_{pre}.S(n-1) \quad (1)$$

where $\tilde{S}(n)$ is the filter output [6]. The rational number 15/16 was used for pre-emphasis coefficient, apre, because simplicity of division by 16 which reduces hardware complexity.

Equation 1 was modified for fixed point implementation:

$$\tilde{S}(n) = S(n) - \frac{15}{16}.S(n-1) = S(n) - S(n-1) + \frac{S(n-1)}{16} \quad (2)$$

In figure 1, the pre-emphasis datapath is shown. A division circuit, a 20 bits adder, a 16 bits subtractor, an accumulator and two registers were used.

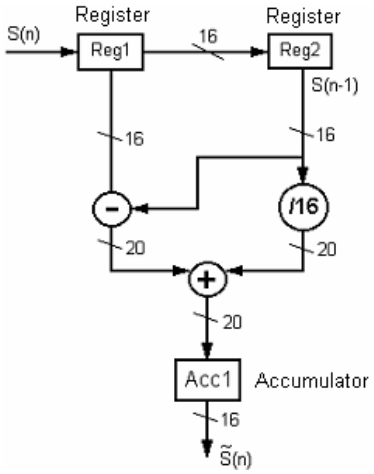

Figure 1. Pre-emphasis datapath.

For external data, 16-bits were used. However, internal operations use a 20-bits datapath in order to get a

better precision. Division by 16 was obtained making a 4-bits shift right and introducing four zeros in more significant bits position, to allow one clock pulse division.

2.2 Frame Separation Function

After pre-emphasis, voice signal is separated in overlapping frames i=0 ... T-1, where T is the number of frames that voice signal could be divided. Each frame is formed by 252 voice samples (22ms aprox.) to guarantee stationarity [4]. Moreover, the samples number (252) allows a frame composition by three equal size blocks (84 samples, each block), necessary for frame overlapping pipeline design.

Because of the frame size and overlap, each sample falls into the first third of current analysis frame, the second third of previous analysis frame and the final third of the two previous analysis frames. After 84 samples, when one of the three frames is completed, the relationship of the three analyzed frames rotates cyclically.

An algorithm that allows optimal hardware implementation of frame separation and overlapping was developed. The algorithm operates in the next manner: In order to obtain the frames, samples after pre-emphasis are segmented in blocks j=1...T, where T is the number of blocks that could be formed from the voice signal. Figure 2 shows the blocks segmentation. Each block j has 84 samples, with non-overlap between blocks. Three successive blocks (252 samples) compose each frame

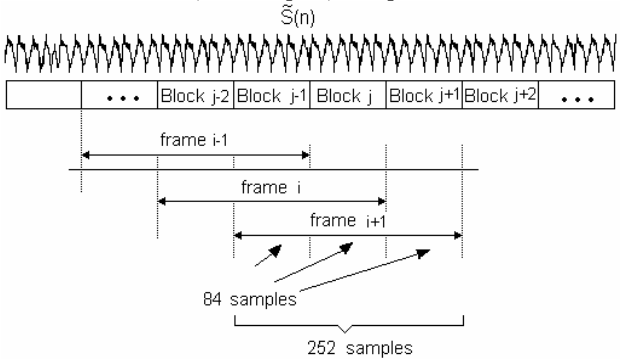

Figure 2. Overlap between frames.

An internal dual-port RAM memory, that allows read/write parallel operations, was used for temporal storage of 252 samples. This memory was divided in three memory segments Mi, i=0...2, where each segment has 84 samples.

Figure 3 shows frame separation operation. Each new block j begins to be stored in the segment memory Mi where the samples of the two previous blocks are stored. Each block is multiplied by the correspondent values of Hamming window, w(n), n=0...251, before storing in memory. Finally, the frame multiplied by Hamming window is stored in an output register.

In each time period, only 28 samples of the new block are stored. It represents the third part of the 84 samples of the block. In parallel, the content of the segment memory that has the samples of the last block is read. In the next time period, the second third of the new block is read (28 samples) and the content of the next to the last block is also read. In the next period time, the final 28 samples of the new block are stored. At the same time, the segment memory that corresponds to the last block is also read.

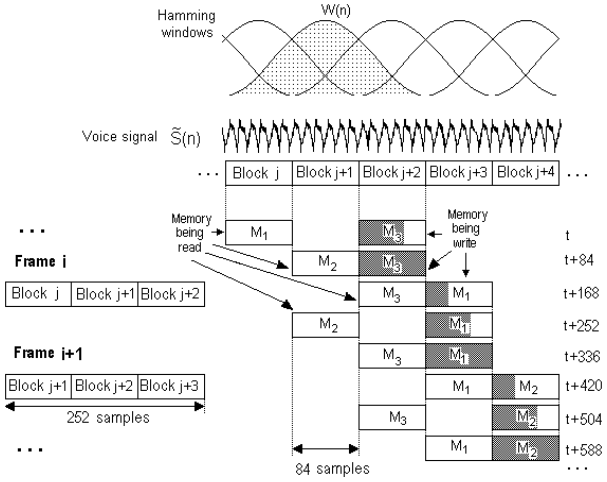

Figure 3. Frame separation using voice signal blocks.

2.3 Windowing Function

The Hamming window has the equation [6]:

$$w(n) = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2\pi \cdot n}{Ns - 1}\right), 0 \le n \le Ns - 1 \quad (3)$$

where Ns is the samples number of each window and n is the sample being evaluated. The window size is 252 samples (22ms aprox.), in order to simplify the Hamming window and the frame separation implementation. In figure 4, the windowing datapath join with the frame division block is shown.

A ROM memory stores the first 126 values of Hamming window (n= 0...125), corresponding to the first two quads of the cosine function. The window values for other points are calculated using cosine function periodicity properties. An up/down binary counter makes hamming memory addressing.

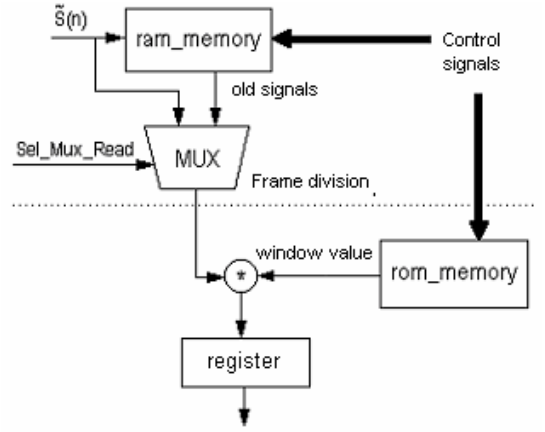

Figure 4. Datapaths of windowing and frame division.

3. MEL-CEPSTRA PARAMETERS EXTRACTION SYSTEM

In figure 5, a diagram for the mel-cepstra parameters extraction is shown. First, spectral energy is computed for windowed sequential frame, using a FFT processor [2].After this, the energy in each of the 27 channels of a triangular bandpass filters set is calculated. Finally, the discrete cosine transform (DCT) of the log energy is computed.

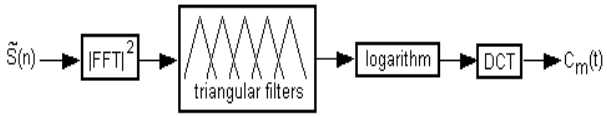

Figure 5. Extraction of Mel-cepstra parameters

In figure 6, hardware for triangular filters function is shown. The triangular filters values are stored in a ROM. A three-stage pipeline is used. The outputs of this bank are 27 energy values.

A logarithm processor using the CORDIC algorithm [5] was implemented in order to obtain the log energy value required for this system. Figure 7 shows the logarithm processor implementation, formed by a scaling circuit, a hardware implementation of the CORDIC algorithm and a control unit. The logarithm processor uses 20 clock pulses for the calculus of a logarithm value.

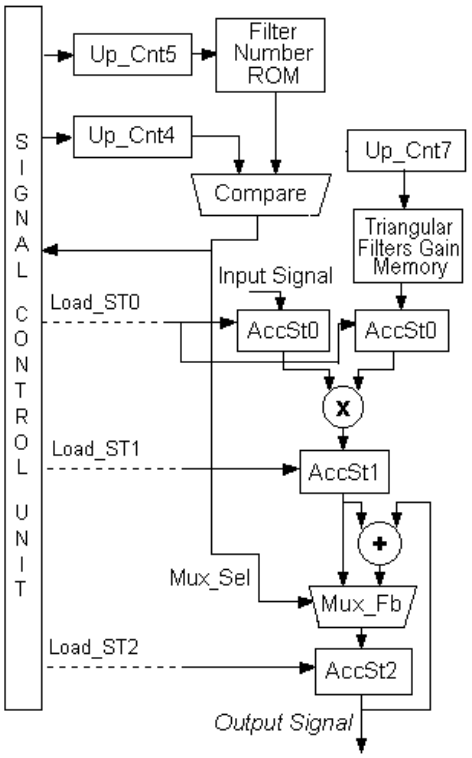

Figure 6. Triangular filters architecture.

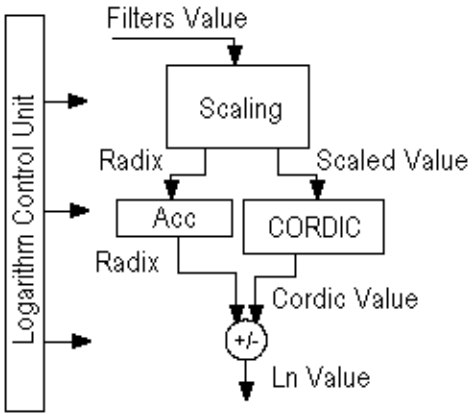

Figure 7. Logarithm processor.

The outputs of the triangular filters (Filters_Value signal) are scaled in order to obtain numbers between 0,5 and 1 that allows the calculus of the CORDIC algorithm for logarithm function. The radix obtained by the scaling is used to get the final result of the logarithm operation.

A DCT pipelined memory optimized algorithm [6] was also implemented. In figure 8, the DCT architecture is shown. A ROM stores cosine values for the DCT calculus. A three stages pipeline was used.

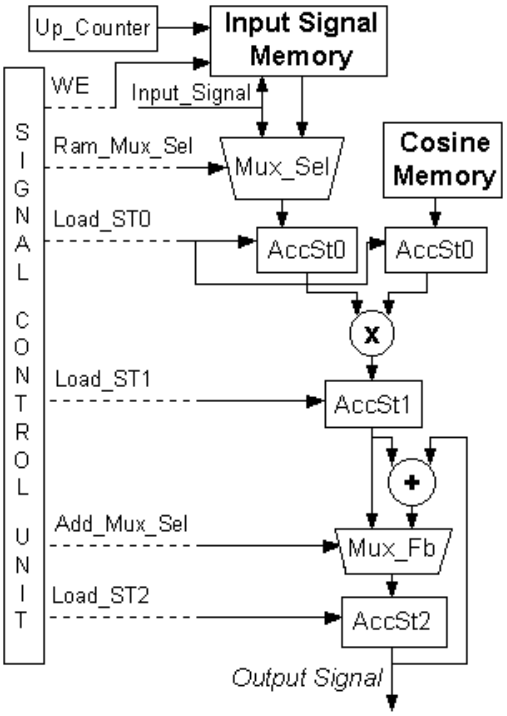

Figure 8. DCT architecture.

4. VECTOR QUANTIZATION

The vector quantization stage (which makes the link between the Mel-Cepstra extraction block and the Viterbi processor) was designed too.

The codebook size is 64. Each vector has 27 mel-cepstra parameters. Figure 9 shows the vector quantization datapath. It uses a RAM memory for temporary storage of the mel-cepstra parameters. A ROM has the centroids obtained after training. Training is made offline.

5. VITERBI PROCESSOR

A Viterbi decoder for speech recognition was also implemented [7]. The goal was the use in left-right Hidden Markov models.

In figure 10, the datapath for the Viterbi processor is shown. The addressing circuit for the Viterbi processor is shown in figure 11.

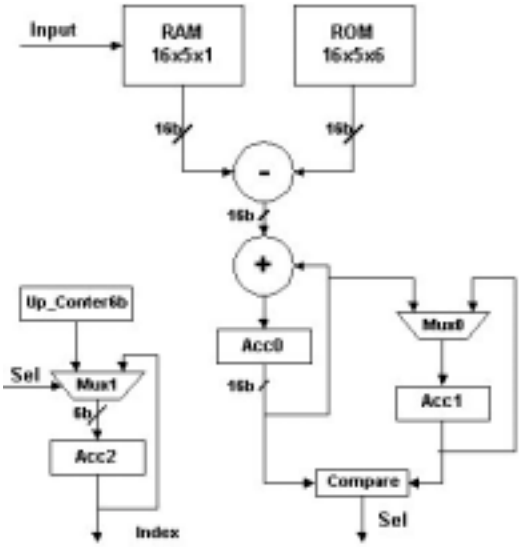

Figure 9. Vector Quantization datapath.

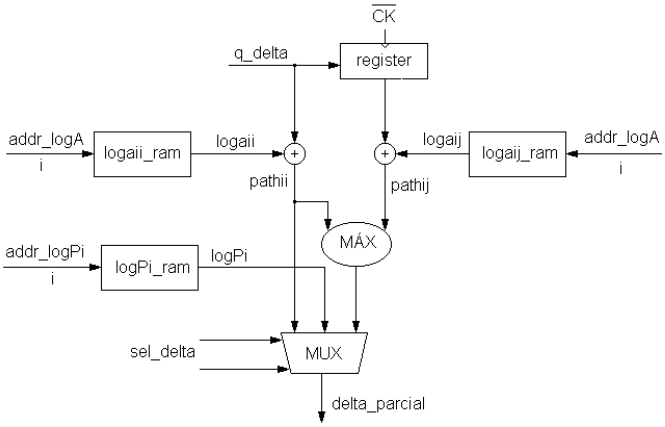

Figure 10. Viterbi processor datapath.

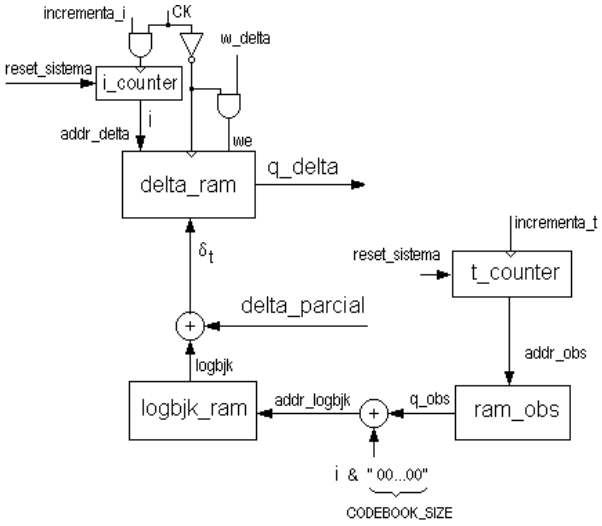

Figure 11. Addressing circuit for Viterbi processor.

6. EXPERIMENTAL RESULTS

The functions explained above were implemented using the Maxplus II tool, in order to use them with FPGAs. They implementation in Matlab and in C was also made, in order to compare behavioral simulations with hardware results. Tests were made with an isolated word small vocabulary for industrial control of elevators.
In table 1, a comparison between the implementation using speech recognition in hardware vs. software implementation is shown.

	HW	SW
Stage	Time(μs)	Time(μs)
Parameter Extraction and Pre-processing	2085	110000
Vector Quantization	173	440000
Recognition with Viterbi decoding	3,75	50000

Table 1. Hardware vs. software results.

8. CONCLUSIONS

The speed characteristics of a dedicated circuit, the physical space, the flexibility of the VHDL description and the potential of FPGA systems make this design usable for the development of new applications. Hardware description language specification opens the possibility to synthesize such systems in different circuit technologies, generating application specific integrated circuits. A system with these features could be used in the experimentation of different stages of speech recognition technology, reducing the physical space used by speech recognition software running in a PC.
The system proposed would be used in problems that require a small vocabulary and a limited speaker number.

Many optimizations were made, to obtain a low latency and a low use of memory.

9. REFERENCES

[1] Brown, M. K. et. al. "The DTWP: An LPC-Based Dynamic Time-Warping Processor for Isolated Word Recognition". AT&T Bell Laboratories Technical Journal, v.63 n.3: 441-457, 1984.

[2] Lapsley, P. et. al. DSP Processor Fundamentals: architectures and features. New Jersey: IEEE Press, 1997. 210p.

[3] Gómez-Cipriano, J. L. et. al. Functional Blocks for Speech Recognition Systems. In: Symposium on Integrated Circuits, SBCCI, 2001

[4] Vergin, R.; O'Shaughnessy, D. Generalized Mel Frequency Cepstral Coefficients for Large-Vocabulary Speaker-Independent Continuous-Speech Recognition. IEEE Transactions on Speech, and Audio Processing, v.7, n.5, p. 525-532, 1999.

[5] Andraka, R. "A survey of CORDIC algorithms for FPGA based computers". In: Proc. of the 1998 ACM/SIGDA Sixth International Symposium on FPGAS. 1998

[6] Aggarwal, G.; Gajski, D. Exploring DCT Implementations. Tech. report, University of California, Irvine. 1998

[7]Gómez-Cipriano, J. L. et. al. FPGA Hardware for speech Recognition Using Hidden Markov Models, ICSLP, 2002.