Pre-processing Speech Signals in FPGAs

Jun Xu, Aladdin Ariyaeeinia, Reza Sotudeh and Zaki Ahmad

School of Electronic, Communication and Electrical Engineering,
University of Hertsfordshire,
College Lane, Hatfield, Hertsfordshire, UK.
x.jun@herts.ac.uk

Abstract:

Algorithms are traditionally developed on off-the-shelf digital signal processors. Recent silicon technology advance has made FPGAs feasible as an alternative solution. However, most of signal processing work in the past has been focused on implementing on software, which has led to a wide gap between algorithms and hardware. This paper is aimed to explore algorithm hardware-migration technologies, in particular, speech signal pre-processing algorithms into FPGAs.

1. Introduction

Automatic Speaker Recognition has been the subject of extensive research over the last two decades [1-3]. This process can be defined as the extraction of the personal identity information from a presented sample utterance using signal measurement techniques. To enhance the accuracy and efficiency of the extraction processes, speech signals are normally pre-processed before features are extracted. Speech signal pre-processing typically includes speech signal detection, pre-emphasis and segmentation with a window technology.

Traditionally, these algorithms are developed on off-the-shelf microprocessors/DSPs. Recent silicon technology advance has made FPGAs feasible as an alternative solution. Reconfigurable device-based systems have been used in diverse applications, and in every case allowing them to improve the flexibility and reduce the cost and the time to market [4-6]. However, most of signal processing work in the past has been focused on implementing on software, which results in a wide gap between the existing algorithms and hardware. The gap may be bridged by developing hardware friendly algorithms or algorithm hardware-migration technologies.

In this paper, the feasibility of migrating existing algorithms, in particular, the speech signal pre-processing algorithms, into FPGAs are explored. The rest of paper is organised as follows: algorithms are described in Section 2; the functionality of each constituent component in the speech signal pre-processing as well as its hardware implementation will be described in Section 3; summary and conclusions are presented in Section 4.

0-7803-9210-8/05/$20.00 ©2005 IEEE

2. Algorithms

2.1. *Speech Signal Detection*

In order to gain a high classification rate, in a speaker recognition system, only the features from valid speech signals (speech utterances) are normally extracted. As a matter of fact, however, there is always an interval between any two speech utterances. Speech Signal Detector is therefore designed to judge when a speech utterance begins and ends. The detection is achieved by short-time energy calculation, and speech energy threshold given by Eq.1. As a result, only the signals whose energies cross the threshold are regarded as valid speech signals.

$$E_{average} = \sum_{n=0}^{N-1} S(n)^2 / N \qquad (1)$$

2.2. *Pre-emphasis*

In speech analysis, it is desirable if a measured spectrum could have a similar dynamic range across the entire frequency band. As a nature of speech, however, there is an overall -6 dB/octave trend in its spectrum as frequency increases. Pre-emphasis is therefore introduced to compensate for the -6 dB/octave roll-off (giving a +6 dB/octave lift). Pre-emphasis in this paper is achieved by a high-pass digital filter cutting off 3 dB frequency somewhere between 100 Hz and 1 kHz. The high-pass filter can be described by Eq.2.

$$S_{pp}[n] = S_{bp}[n] - \alpha S_{bp}[n-1] \qquad (2)$$

Where $S_{pp}[n]$ denotes the current output signal of the filter, $S_{bp}[n]$ is the current input signal and $S_{bp}[n-1]$ is the previous input signal. α is a constant between 0.9 and 1.

2.3. *Segmentation with Hamming Window*

Due to the fluctuating nature of speech waveforms, speech signals are normally segmented into short-time (10-30ms) using windowing techniques before being featured. One popular windowing technique is known as Hamming Window, described in Eq.3, in which amplitude gradually falls to a minimum at each end of the segment.

$$W n = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) & if \quad 0 \le n \le N-1 \\ 0 & otherwise \end{cases} \qquad (3)$$

However if the samples lying near the ends of such a window represent a significant speech feature, their low

weighting will preclude them from being effectively featured in the segment analysis. Two adjacent segments are therefore overlapped by 50% and the signals receiving low weighting at one segment can be highly weighted in the other segment in a complementary form as presented in Fig.1.

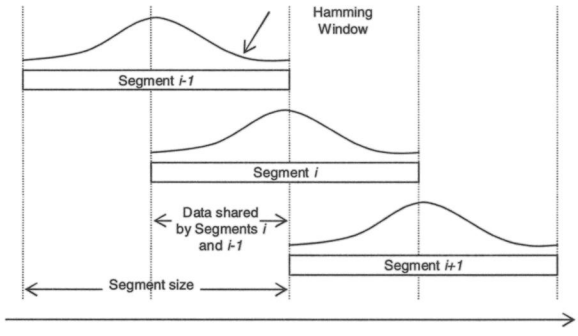

Figure 1 Segmentation with Hamming Window Technique

3. Hardware Architecture and Implementation

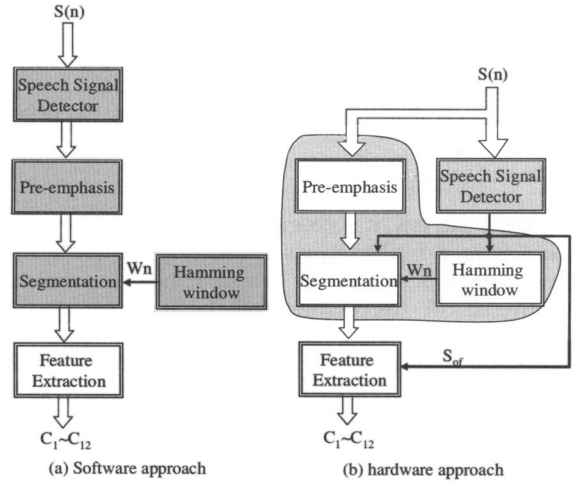

Figure 2 Speech Signal Pre-processing

In software, the constituent components are operated in sequence as shown in Fig.2a. In order to reduce the implementation complexity and improve the hardware performance, we re-examined the relationship between different components at top level. It is found that the relationship between Pre-emphasis, Segmentation and Feature Extraction are tightly coherent with each other while Speech Signal Detection is rather loosely coupled with the rest of pre-processing system. This means that the architecture of the hardware system can be optimised (in comparison with the software approach) at a system level by running Speech Signal Detect logic in parallel with the rest of system without changing the functionality of the pre-processing system. The hardware architecture of pre-processing is shown in Fig.2b.

3.1. *Speech Signal Detection Logic*

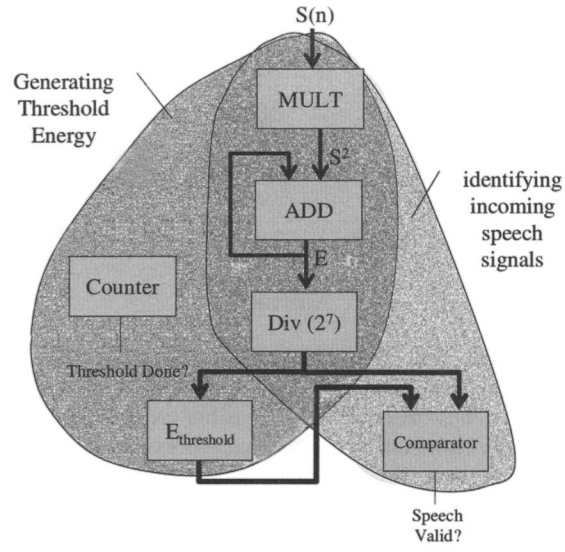

Figure 3 Speech Signal Detection

The operation of Speech Signal Detector in hardware is categorised into two stages: generating threshold energy and identifying speech signals. Generating threshold energy is only conducted during the initial stage of a system--a few seconds after a speaker recognition system is switched on. In this stage, background signals are sampled and threshold energy is decided. As shown in Fig.3, incoming signals are first transformed to energy at Multiplier and then accrued recursively at ADD; energy threshold (mean energy) is finally obtained from the divider. This hardware feature can make a speaker (pre-) processing system more self-adaptable to every environment with different background noise. The rule of thumb is that the noisier environment, the higher threshold is expected.

Once the threshold is generated, Speech Signal Detector moves to the second stage: identifying valid speech signals. Speech Signal Detector will stay in this stage as long as it possible until the system is switched off. Identifying valid speech signals is achieved by comparing the mean energy of each frame with the threshold energy. To reduce the circuit size, the mean energy of each frame is obtained by using the same logic as for generating threshold energy. If the mean energy of a frame is greater than threshold energy, the speech signals in the frame are regarded as valid speech signals; otherwise, the signals are regarded as invalid speech signals.

Unlike its software counterpart, Speech Signal Detector in hardware also outputs the validity of incoming speech signals via a flag (S_{of} in Fig.2b) to the rest of system. In an effort to reduce the power consumption on the chip, most of the system can be shut down on the basis of the information provided by S_{of}.

3.2. *Pre-emphasis Logic*

The overview of digital pre-emphasis filter is illustrated in Fig.4, in which *D* is a register.

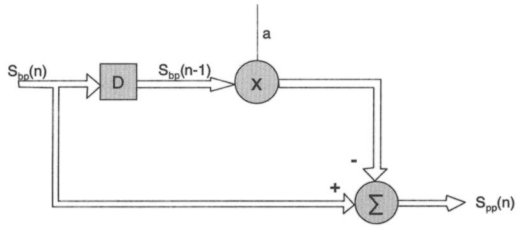

Figure 4 Overview of Pre-emphasis

The first concern in the pre-emphasis logic design is how to implement α. Although α can be any value between 0.9 and 1, assigning an arbitrary value to α can increase implementation cost and processing latency. The cheapest multiplication/division logic would be a shifter if α is equal to 2 to the *n*th power (*n* is integer). Having considered that, in this paper, 0.96875 is assigned to α. This results in Eq.2 being evolved into Eq.4

$$S_{pp}[n] = S_{bp}[n] - 0.96875 S_{bp}[n-1]$$
$$= S_{bp}[n] - S_{bp}[n-1] + S_{bp}[n-1]/2^5 \quad (4)$$

The corresponding pre-emphasis logic therefore can be built as presented in Fig.5. Every newly arrived speech signal is buffered at Reg1; at the same clock cycle, the existing signal in Reg1 is moved to Reg2 and Reg3. In particular, the signal moved to Reg2 is shifted 5-bits rightwards to fulfil the division as described in Eq.3. The newly arrived signal at Reg1 is then subtracted by the one from Reg3 at the next clock cycle; the pre-emphasis of a signal is completed after the result of the subtraction is added on by the signal from Reg2 at the third clock cycle.

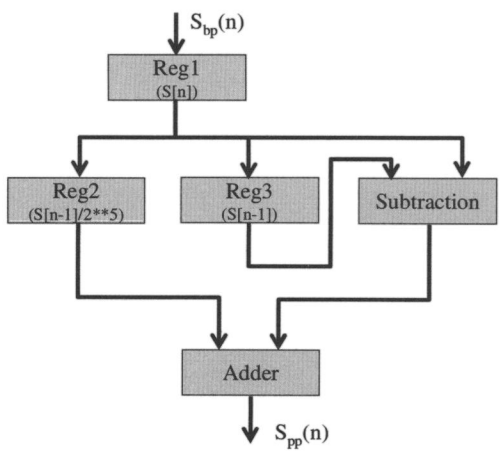

Figure 5 Implementation of Pre-emphasis

3.3. *Segmentation Logic*

One challenge in transferring Hamming-window technique to hardware is to translate the trigonometric identity, i.e., cosine, into circuitry. Two approaches have been considered in our research: one is to use series expansion techniques to approximate the identity. However, to precise the approximation to the third digit after fraction (in decimal), the identity has to be expanded to its 40th order or even higher, which is very costly for hardware implementation. An alternative is to implement a lookup table to record all cosine values that will be used. Given that each segment contains 128 samples and each window value (ranging from 1.000 to 0.080) costs 16-bits, the lookup table only takes 2k-bits memory. As presented in Fig.6a, a counter is deployed to record the sequence of incoming speech signals; its output is converted to a Lookup Table address, by which means the associated Hamming Window value is obtained. Fig.6b shows two complementary speech signals are produced simultaneously after a window value is retrieved. Compared with series expansion techniques, the lookup table is much less complicated and has less resource demand.

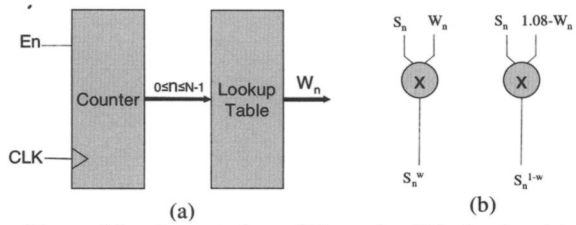

Figure 6 Implementation of Hamming Windowing (a) and Segmentation (b)

4. Summary and Conclusions

This paper aims to explore the implementation of speech signal pre-processing algorithms on FPGAs. Compared to the conventional (software) approach, the hardware solution has taken the advantage of parallism throughout the whole signal processing and signal processing efficiency has been exploited by employing task-specific logic. The FPGA-based hardware solution can be an optimal and alternative solution to the conventional DSPs for speech signal pre-processing technology.

Acknowledgements

The project is funded by DTI UK. The authors would like to thank their financial support.

References:

[1] B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", J. Acoust. Soc. Am., Vol. 50, No. 2, Pt. 2, pp. 637-655, Aug. 1971.

[2] M. J. Carey and E. S. Parris, "Speaker Verification", Proc. IOA, Vol. 18, pp. 99-106, 1996.

[3] H. Gish and M. Schmitdt, "Text-independent Speaker Identification", IEEE Signal Processing Magazine, Oct. 1994, PP. 18-32.

[4] S. Hauck, T. Fry, M. Hosler, J. Kao, "The Chimaera Reconfigurable Functional Unit", Proceedings of the 5th IEEE symposium on FPGA Based Custom Computing Machines, pp. 97-103, 1997.

[5] D. Lewis, D. Galloway, M. van Ierssel, J. Rose, P. Chow, "The Transmogrifier-2: A 1 Million Gate Rapid Prototyping System," in FPGA `97, ACM Symp. on FPGAs, Feb 1997, pp. 53-61.
[6] http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=microblaze.