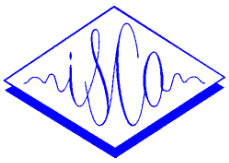

4th European Conference on
Speech Communication and Technology
EUROSPEECH '95
Madrid, Spain, September 18-21, 1995

HARDWARE DESIGN OF LPC CODING FOR SPEECH FEATURE EXTRACTION

M Li and J T Proudfoot
Department of Electrical & Electronic Engineering,
University College of Swansea, SWANSEA, SA2 8PP, UK
N Ireland Bio-Engineering Centre,
University of Ulster at Jordanstown, Co Antrim BT37 0BQ, N Ireland

ABSTRACT

This paper describes the high-level design and specification of an Application-Specific Integrated Circuit (ASIC) to implement a speech feature extraction algorithm based on Linear Predictive Coding (LPC). The ASIC uses a fully synchronous top-down design methodology with a hardware description language -- ELLA. To suit the hardware design, the LPC algorithm is reformulated. A register-transfer level structure is used to meet requirements for suitability and efficiency of hardware, and a hierarchical design methodology used to implement the algorithm in low cost ASIC form. The approach can be applied to a wide range of digital signal processing functions. The goal of this effort is to develop a speaker recognition system based on the LPC algorithm which can be used as an alternative to the currently used software system.

1. INTRODUCTION

Achieving the performance required from a modern digital signal processing (DSP) system often necessitates the real time application of reliable numerical algorithms for least squares estimation, distance measuring, solving linear equations, performing singular value decomposition and so on. In order to perform such computations at a required data rate, it is sometimes necessary to design a dedicated processor which can be implemented using advanced VLSI technology. This paper presents the design of such a chip to implement the LPC algorithm for speech feature extraction at high level using ELLA tools: ELLA hardware description language and simulator.

The LPC algorithm is a very important spectral estimation technique in speech processing because it can model the vocal tract and has complete mathematical fundamentals [1]. It has been extensively used in applications of speech coding, speech and speaker recognition. In this project, it is used to extract features especially for speaker recognition. It is a medium bit rate, good quality speech coding technique. The LPC algorithm assumes speech can be modelled as an all-pole filter. The feature extraction is to determine the all-pole filter coefficients which are evaluated by using the Least-Square Error Algorithm (Autocorrelation Method and Durbin Recursive Procedure). This is very complex algorithm which requires several different types of processing. If there is method to derive a computation scheme to simplify the equations and corresponding architectures, it would be beneficial to the hardware design. For this reason this paper presents some techniques to reformulate linear algebra computations such as matrix multiplication, triangularisation and back substitution which are developed to map onto very efficient architectures.

Raw speech is pre-processed by a low-pass filter after A/D conversion at sampling rate 10 kHz, then segmented into 25.6 msec (256 samples) frames with 50% overlap. The segmented speech is multiplied by a Hamming window prior to the LPC analysis using autocorrelation method.

2. REFORMULATING THE LPC ALGORITHM

Assuming there is a speech signal s(n), the LPC coefficients, a_i, based on the autocorrelation method and Levinson-Durbin [1] can be calculated as follows:

$$R(i) = \sum_{n=0}^{N-1-i} s(n)s(n+i) \qquad (i=0,1,...,p) \qquad (1)$$

where $R(i)$ is the autocorrelation coefficients, N is the length of a frame of speech data and p is the number of the LPC coefficients which is chosen as 12 in this project. Based on the autocorrelation coefficients, the Durbin's recursion is used to calculate a set of PARCOR coefficients k_i:

$$k_i = \frac{R'(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R'(i-j)}{E^{(i-1)}} \qquad (2)$$

where $R'(i) = R(i)/R(0)$ is a normalised autocorrelation coefficients and

$$E^{(i)} = (1 - k_i^2)E^{(i-1)} \tag{3}$$

where $E^{(0)} = R'(0)$,

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}, \tag{4}$$

$$(j = 0,1,...,i\text{-}1;\ i = 1,2,...,12;\ j \neq i),$$

$$a_i^{(i)} = k_i \qquad (when \quad j = i) \tag{5}$$

Therefore, the LPC coefficients are

$$a_j = a_j^{(12)} \qquad (j = 1, 2, ..., 12) \tag{6}$$

From the above equations, it can be seen that the calculations are irregular-loop recursive and requires several steps each with several different operations to obtain the LPC coefficients. If there is a method to derive a computation scheme to simplify the equations and corresponding architectures, it would be beneficial to the hardware design. There is a way in which Equation (2) can be rewritten as an entire recurrence (a repeated process or iteration) as follows:

$$-k_i = \frac{\sum_{j=0}^{i-1} a_j^{(i-1)} R'(i-j)}{E^{(i-1)}} \tag{7}$$

where $a_0^{(i-1)}$ is created and always set to -1. It can be seen that equation (7) can simplify not only the architecture of the function calculating k_i significantly but also the downstream structure. For example, the formulas (4) and (5) can be combined to one formula:

$$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}, \qquad (j = 0, 1, 2, ..., i-1;\ \ i = 1, 2, ..., 12) \tag{8}$$

where the subtraction operation in equation (4) becomes an addition and $a_0^{(i-1)}$ is always -1. The formula (5) can be deduced to $a_i^{(i)} = -k_i$, from formula (8), since $a_{i-j}^{(i-1)} = a_0^{(i-1)} = -1$ and $a_j^{(i-1)} = 0$ when all of $a_j^{(i-1)}$ except $a_0^{(i-1)}$ are set to zero before the entire one-vector recursive calculation begins. As a result, it is identified that there exist four different sets of recurrence, that is, the formulae (1), (3), (7) and (8). Based on the four equations, Figure 1 shows the partition of the reformulated LPC algorithm.

3. ASIC DESIGN METHODOLOGY

This project uses a fully synchronous top-down design methodology with the ELLA design tools. ELLA [2] is an integrated high-level CAD environment for the design and simulation of an architecture described in a hardware description language. It models architectures in a modular way, and its support environment is used to simulate different design alternatives. The hierarchical design methodology inherent in ELLA enables various degrees of parallelism and pipeline to be conveniently exploited to achieve an efficient ASIC implementation. The ASIC implementation of the LPC algorithm utilises both parallelism and pipeline to form an optimised architecture which performs the desired task.

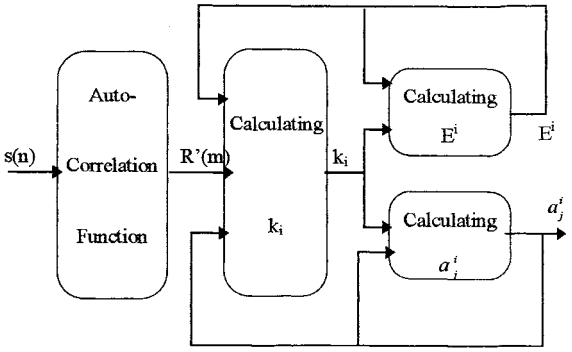

Figure 1 Partition of the LPC Algorithm

4. ARCHITECTURE

A bit-parallel processor architecture [3] [4] without parallel multipliers is used for the design. Figure 2 shows the block diagram of the system, which represents both its functional flow diagram and its logical and physical structure. This is a pipelined, multiplexed and multiprocessor arrangement [5]. The realisation of the circuit from this level of description only requires few operator module designs implementing the desired functions with mutually compatible interfaces and a controller for the timing.

From Figure 1, it can be seen that the communication among processes in calculating k_i , E^i and a_j is very complicated. Thus, it is better to consider these three parts together. The architecture combines the two parts that calculate k_i and a_j to share one module, that is multiplier-accumulator. This is because the functionality is similar in a multiplier-accumulator and a multiplier-adder. Through simulation it was found that the architectural arrangement is more efficient than that for two modules to calculate k_i and a_j. respectively This is a very complicated recursive system and the basic functionality of the reformulated LPC algorithm design, see Figure 2, is discussed as follows.

1) Storage

The system has three individual memory blocks. Since the twelve autocorrelation coefficients *R'(i)* are repeatedly accessed during a frame time, they are first loaded into a input buffer and kept until a vector of LPC coefficients is

obtained. The working area requires two buffers, each with p+1 elements, to store the current intermediate results of a_j^i and the previous intermediate data for the current recursive cycle calculation.

2) Iteration Processor

The iteration processor is a multiplier-accumulator or a multiplier-adder to perform recursive functions in both equations (7) and (8). Their functionality is similar and they work at different times since their computations are dependent on each other. Thus the two processes can share the same recursive processor and be multiplexed to execute in sequence.

3) Multiplexers

The multiplexers are used to switch correct data sources at an appropriate time to the inputs of the recursive processor to work for either multiplication-accumulation or multiplication-addition as discussed above.

3) Divider

The divider follows the iteration processor to realise the division in equation (6).

4) E Processor

The E processor computes equation (3) and sends the output to the iteration processor for calculating the next k_i and to itself for calculating the next E.

5) Registers

The dependent steps can be isolated in some cases, by delaying stages of computation by one sample cycle, using registers. This is the basis of pipelining, through which the now independent stages may be implemented concurrently. The K-REG register isolates two calculating steps and the stores previous E(i-1) for current calculation of E(i).

6) Data Busses

The system use two data busses: one is special for the input buffer of *R'(i)* to provide data to the iteration processor which frequently accesses the buffer and the other is a common bus for all other communications between processors and buffers.

32-bit floating point arithmetic is used throughout the data-buses and buffers, and 64-bit floating point arithmetic is exploited within all the arithmetic units, but their outputs are rounded off to 32 bits to keep bus size, real estate usage and memory size low [5].

5. SIMULATION RESULTS

High level functional simulations were carrieded out during development of the ASIC architecture to verify circuit functionality, timing and performance. Test vectors were generated by extracting parameters from real speech. Figure 3 shows two sets of LPC coefficients which are extracted from utterance a. The left one is obtained from software and the right is obtained from ASIC simulation. Comparing these two graphics they are essentially identical.

6. CONCLUSIONS

The research of this paper presents the design of a speech feature extractor based on Linear Predictive Coding (LPC) at high level architecture using ELLA tools: the ELLA hardware description language and simulator. Starting from the algorithm-level description, it is ended with the architectural level implementation of the LPC algorithm. The application has shown that properly reformulating an algorithm before mapping into an architecture makes significant contribution to hardware design and efficient architecture. At high level design the ELLA environment enables the designer to gradually add detail to the modules, establishing the partitioning, structure and communication requirements which will apply in layout. In addition ELLA can provide useful feedback on some design performance at a very early stage in the design process so that the system architecture can be perfected rapidly and efficiently.

REFERENCES

[1] L. R. Rabiner, *Digital Processing of speech signals*, Prentice Hall, 1978, pp 396-412.

[2] *The ELLA User Manual, chapter 7 and 8*, Praxis Electronic Design Ltd, 1990.

[3] David Green, *Modern Logic Design*, Addison Wesley Publishing Company, Inc. 1986.

[4] De Man, *Cathedral II: A synthesis and Module Generation System for Multiprocessor Systems on a Chip*, Design Systems for VLSI Circuits---Logic synthesis and silicon complication, 1987.

[5] S. P. Pope, *Automatic Generation of Digital Signal Processing Circuits*, Design Systems for VLSI Circuits---Logic Synthesis and Silicon Compilation, Edited by G. De Micheli, Martinus Nijhoff Publishers, 1987, pp541-569.

[6] P.A. Subrahmanyam, *Synthesizing VLSI Circuits from Behavioral Specifications: A Very High Level Silicon Compiler Theoretical Basis*, Design Systems for VLSI Circuits---Logic Synthesis and Silicon Compilation, Edited by G. De Micheli, Martinus Nijhoff Publishers, 1983, pp175.

[7] A.C.Woo and L.E.Peppard, *System-Level Modelling in VHDL*, Microelectronics Journal, 23, 1992, pp. 223-230.

[8] Franklin P. Prosser, *The art of Digital Design --An Introduction to Top-Down Design*, Prentice-Hall, 1987.

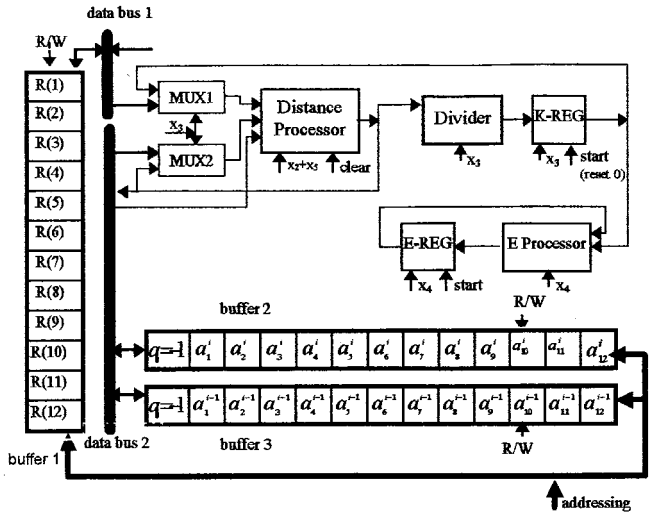

Figure 2 Block Diagram of Architecture of the LPC

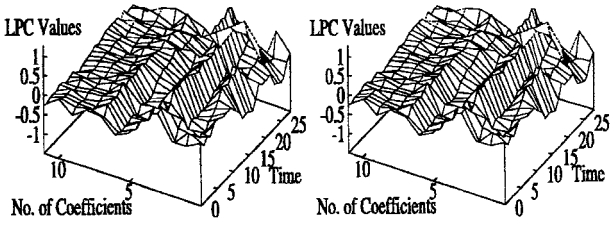

Figure 3 LPC coefficients extracted from utterance a: left: obtained from the software, right: obtained from the ASIC simulation