宜蘭大學工程學刊(2007)
第三期第 69～86 頁

Structural Design for a 1.6 Kbps GELP Speech Coder

Hwai-Tsu Hu[1]　Yong -Nan Chen[2]

1. Professor, Department of Electronic Engineering, National Ilan University
2. Technical Manager, Terasic Technologies Inc.

Abstract

This paper presents a structural design for hardware implementation of low-bit-rate speech coding. The processing algorithm emerges from the traditional pitch-excited linear prediction vocoder with incorporation of glottal-source properties. To make the coding algorithm realizable, constraints are imposed on the complexity of computational modules as well as the need of arithmetic operations. Nonetheless, the simplified algorithm is still capable of carrying out 1.6 Kbps speech coding with acceptable quality.

We have gone through algorithm modification, fixed-point analysis, hardware design, and circuit synthesis by making use of various design tools. The hardware design, described by using Verilog-HDL, is compiled and verified on an FPGA-based DSP board. Our results confirm that the proposed architecture works quite well. As this architecture needs not consume many hardware resources, it is suitable for single chip implementation.

Keywords：speech coding, VLSI architecture, hardware implementation

1.6Kbps GELP 語音編解碼器之結構設計

胡懷祖[1] **陳泳男**[2]

1. 國立宜蘭大學電子工程學系教授
2. 友晶科技公司技術經理

摘 要

本文針對低位元率語音編碼之硬體實現提出結構性設計，負責程序處理之演算法源自於傳統的脈衝激源線性預測語碼器，另摻入聲源之特性。爲了讓編解碼演算法得以硬體實作，諸如計算模組的複雜度與算數運算的需求均加上許多限制，然而簡化後的演算法仍能執行 1.6Kbps 語音編解碼並輸出可接受之音質。

我們藉由多樣設計工具，經歷了演算修正、定點數分析、硬體設計、電路合成等過程，最後以 Verilog 硬體描述語言設計出之電路硬體是在擁有 FPGA 的 DSP 開發板進行編譯及驗證，所得結果證實這套架構的工作效能。由於該架構不需耗用過多硬體資源，其實是非常適合以單晶片實現。

關鍵詞：語音編解碼、VLSI 構造、硬體實現

1. Introduction

In recent years, low-bit-rate speech coding has drawn much attention due to its wide applications to telecommunications and information appliances. The phrase “low bit rate” is particularly used to signify the reduction of transmission bandwidth and memory storage, which in turn promotes the efficiency of speech-related devices and facilities. Nowadays prevailing products consist of the cellular handset, videophone, VoIP, dialogic system, digital recorder, and digital answering machine, etc. In order to economize the above-mentioned commercial products, a plausible choice would be the exploitation of chip design so that all processing requirements are accomplished by integrated circuits and control logics. Nevertheless, the mapping of a coding algorithm into hardware is a work-intensive process, since the designer must understand how to adjust the underlying algorithm subject to many constraints. Generally encountered constraints include the latency, throughput, timing characteristics and the complexity of the algorithm. For ASIC-based implementation a well-developed algorithm should also be parameterizable. In other words, the algorithm should be coded into modules using a hardware description language like VHDL and Verilog. Furthermore, to make the developed algorithm portable to a structural design with limited hardware resources, it is essential for us to trim the algorithm as much as possible and to take away complicate operations providing the resulting performance is still acceptable.

2. Design procedure

In this paper, apart from the algorithmic issue, the design and implementation of the VLSI architecture is also our concern. The steps leading to hardware implementation of speech coding are as follows. First, the algorithm is developed and verified within the working environment of Matlab. We then translate the algorithm into a fixed-point version using the C++ programming language. A new data type based on the size of bits and the position of the radix point is created to suit the need of binary arithmetic. Hence different types of numeric values are convertible via casting operators. Arithmetic operators are reconstructed accordingly to manage fixed-point operations. As we proceed with the algorithm flow, the radix point of the variables is dynamically allocated to preserve the most significant bit. For some particular variables we have reserved few bits to avoid overflow during intermediary stages.

The foregoing fixed-point analysis ensures the portability of the developed algorithm to ASIC-based implementation. As we attempt to shorten each design cycle, the hardware system is developed based on a DSP development board. This board has an FPGA device integrated with an AD/DA converter responsible for audio sampling and playback. We use the Verilog language to create the required processing components and control units. Subsequent functional simulation and logic synthesis are accomplished by resorting to the ModelSim (which is a Verilog simulator) and the Quartus II (which is a synthesis tools distributed by Altera Inc.). For a complicated system such as our speech coder, timing simulation and analysis is crucial as well. Eventually, the in-circuit evaluation is carried out on the DSP board once the FPGA device is properly configured. The above procedures for algorithm development, computer simulation, and hardware verification are repeated until we reach satisfying results.

Consequently, this paper not only aims at the development of an efficient algorithm but also verifies its feasibility via hardware implementation using an FPGA. The rest of this paper is organized as follows. Following the discussion of the design procedure, both the speech production model and the bit allocation for the proposed coding scheme are described in Section 3. Sections 4 and 6 present the architectures for speech analysis and synthesis respectively, while the spectral quantization is placed in between (namely, section 5). The performance evaluation of the coding algorithm and architecture is discussed in Section 7. Concluding remarks are finally given in Section 8.

3. Speech production model

We adopt a model called glottal excited linear prediction (GELP) [1]-[3] to incorporate glottal properties into the conventional linear prediction (LP) coder. As shown in Fig. 1, this model inherits the fundamental structure of the LP coder [4], which is developed based on the source-filter theory. The

source model switches between the voiced and unvoiced types according to the voicing condition, while the all-pole synthesis filter renders the composite spectral characteristics of the glottal flow, vocal tract transfer function, and lip radiation.

Given that the speech signal is sampled at 8 KHz with 8-bit resolution, we update the analysis frame at a rate of 200 samples with an overlap of 56 samples for consecutive frames. Table I presents the bit allocation for the speech coder.

4. Architecture for speech analysis

Speech analysis with respect to the speech coder is equivalent to extracting modeling parameters such as the pitch, gain, and filter coefficients. Techniques constituting the framework of speech analysis thus involve pitch detection, gain determination, and estimation of filter coefficients.

There are five important modules that constitute the hardware for speech analysis. As shown in Fig. 2, apart from the buffer, one module is designed to detect pitch periods while the remaining three are used to derive spectral parameters.

A. Data Buffer and Memory Requirement

As each analysis frame consists of 256 speech samples in length, we reserve a memory space of 400 bytes to achieve a double-buffer processing of speech data. The reserved memory is split into two parts of equal size. Speech samples captured by the AD converter are assigned to the designated memory in a circulating manner. A buffer is full once it collects 200 samples. The last 56 samples in the alternative buffer and those in the current buffer then form a frame of 256 samples for analysis. Fig. 3 illustrates the double-buffer scheme.

As will be clear later, we impose a 1st-order highpass filter on the speech signal before performing LP analysis. Additional memory space is required if we want to reserve intermediate outcomes. To avoid unnecessary memory consumption, we instead apply a 1st-order filtering operation to the autocorrelation function. Given that the 1st-order filtering process is expressed as

$$y(n) = s(n) - 0.925\, s(n-1) \tag{1}$$

where s(n) denotes the speech signal, and y(n) is the filtered output that is also regarded as the intermediate result. Multiplying each side of Eq. (1) by itself with a different index and taking the time average over the underlying frame result in

$$R_{yy}(k) = R_{ss}(k) - 0.925\big(R_{ss}(k+1) + R_{ss}(k-1)\big) + 0.925^2 R_{ss}(k) \tag{2}$$

where $R_{yy}(k)$ and $R_{ss}(k)$ correspond to the kth-lag autocorrelation function of y(n) and s(n), respectively. It turns out that an extra lag of $R_{ss}(k)'s$ is indispensable in order to compute $R_{yy}(k)'s$. However, since the number of multiplication required by the extra lag autocorrelation function is exactly the same as that required by the 1st-order filtering, the need of an extra autocorrelation function does not mean the necessity of additional computation. Thus, Eq. (2) is considered an alternative approach to acquiring the autocorrelation function of a filtered signal without any demand for memory space to store the intermediate result.

B. Autocorrelation

The most time-consuming computation lies on the computation of the autocorrelation function. To accelerate the processing speed, we deploy ten multiplier-accumulators (MACs) to attain the autocorrelation sequence in parallel. Fig. 4 presents the block diagram for the computation of autocorrelation functions. Since speech samples captured from the AD converter are trimmed down to have 8-bit resolution, the expenditure of hardware resources is still under a reasonable level.

C. Levinson-Durbin Recursion

Once the autocorrelation function is available, a Levinson-Durbin recursion module shown in Fig. 5 is brought in to derive LP coefficients. The involving computational units consist primarily of multiplexers and primitive arithmetic operators, which are designed to cope with data of different resolution. A by-product associated with the Levinson-Durbin recursion module is the easy derivation of the gain factor, which is used to control the intensity of synthetic speech. As suggested by Makhoul [5], we computed the gain, termed G, by referring to the linear prediction relation.

$$G = \sqrt{R_{yy}(0) - \sum_{k=1}^{p} a_k R_{yy}(k)} \tag{3}$$

where a_k denotes the kth LP coefficient, and p stands for the LP order. While the above formula necessitates a square root operation, a fast algorithm

developed by Tommiska [6] is employed to execute the process. The adopted square-root operation needs only n clock cycles for 2n-bit wide numbers. In Eq. (3), the value within the square root operator is just the variable E (variance of prediction errors) appearing in Fig. 5. This particular value is simultaneously available right after completing the linear prediction analysis of speech signals.

D. Conversion from LP Coefficients to LSP Parameters

The forth module focuses on the conversion from the LP coefficients to the line spectral pair (LSP) parameters. Given that the LP coefficients are available through the Levinson-Durbin iteration, two auxiliary polynomials are formed by adding and subtracting the LP transfer function to its reversed system. The angles of the roots derived from these two polynomials are known as the LSP parameters. Unlike many other algorithms that employed iterative procedures to find the roots, we had deliberately reduced the prediction order down to 9, thus allowing us to adopt a closed-form solution to pursuit the roots. More specifically, we utilize a 1st-order highpass filter in combination with an 8th order LP filter to model the spectral characteristics. The 1st-order highpass filter is set as Eq. (1) to emphasize the high-frequency region of spectrum, while the 8th-order LP filter accounts for the formant structure. Assuming that $A(z)$ is the transfer function of the 8th order linear predictor defined as

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_8 z^{-8}. \quad (4)$$

Under such a condition, the auxiliary symmetric and anti-symmetric polynomials, denoted as $P(z)$ and $Q(z)$, become

$$\begin{cases} P(z) = A(z) + z^{-(m+1)} A(z^{-1}) \\ Q(z) = A(z) - z^{-(m+1)} A(z^{-1}) \end{cases}, \quad m = 8. \quad (5)$$

Notice that either $P(z)$ or $Q(z)$ can be rewritten as the following form:

$$\begin{aligned} R(z) &= z^{-9}\left[\left(z^4 + z^{-4}\right) + r_3\left(z^3 + z^{-3}\right) + r_2\left(z^2 + z^{-2}\right) + r_1\left(z + z^{-1}\right) + r_0\right] \\ &= z^{-9}\left[x^4 + r_3 x^3 + \left(r_2 - 4r_4\right)x^2 + \left(r_1 - 3r_3\right)x + r_0 - 2\left(r_2 - r_4\right)\right] \end{aligned} \quad (6)$$

where $x = z + z^{-1} = 2\cos\omega$, and ω denotes the corresponding line spectrum frequency. We rewrite the expression within the brackets on the right hand side of Eq. (6) as

$$x^4 + ax^3 + bx^2 + cx + d = 0 \quad (7)$$

As demonstrated in [7], the above equation can be solved as follows. First, let $E = \frac{a}{2}$, $A = \sqrt{a^2/4 - b + y_1}$, and $B = \frac{Ey_1 - c}{A}$, where y_1 is one of the roots for the resolvent cubic equation

$$y^3 - by^2 + (ac - 4d)y - a^2 d + 4bd - c^2 = 0.$$

Then, the four roots of Eq. (7) become

$$z_{1,2} = \frac{-(A+E) \mp \sqrt{(A+E)^2 - 2(y_1 + B)}}{2} \quad (8)$$

$$z_{3,4} = \frac{(A-E) \mp \sqrt{(A-E)^2 - 2(y_1 + B)}}{2}. \quad (9)$$

Fig. 6 illustrates the required hardware resources for the root-solving procedure.

The above-mentioned root-solving module demands a lot of square-root operations to derive the cosine function of the LSP parameters, termed *clsp*[0] ~ [3] in Fig. 6. Again, the digital circuit devised by Tommiska [6] is employed to fit for the task. In fact, the square-root component has become the major part of the arithmetic unit shown in Fig. 6. Other crucial constituents consist of two adders/subtractors, a multiplier, a divider, a comparator, and a logic circuit for sorting the sequence of parameters. Fig. 7 depicts such a structure.

E. Pitch Detection

The last module for speech analysis is dedicated to pitch detection. For LP-type vocoders, the pitch is often estimated by examining the autocorrelation function of the analysis frame. However, computation involving the correlation function is always burdensome, especially when the range of the time lag extends from 21 to 147 considered in our case. It is definitely unwise to deal with this kind of computation in a parallel manner since the corresponding hardware demands 127 MACs. We therefore resolve the difficulty by treating the autocorrelation function as the comparison between the sign bits extracted from a lowpass filtered signal and a shifted version of themselves. In practice, the sign bits of all samples in a frame can be collected as a lengthy word. The correlation function can be attained by first taking the bit-wise equivalence (or called the exclusive-NOR) operation and then counting the number of 1's. In fact, this kind of approaches was considered useful in early days when the speed of processors is slow, but is obsolete nowadays. Ironically, it suits the VLSI design very

well. The accuracy of the estimated pitch is somewhat degraded due to the degeneration of speech samples, but can be ameliorated a lot by incorporating with supplementary decision logics.

In our coding scheme, the voicing classification and pitch detection is formulated as the following. Let $m_c(k)$ denotes the number of 1's accumulated from the outcome of the bit-wise equivalent operation when the time lag is k. We multiply $m_c(k)$ by a scaling factor $(1+0.002k)$ to render a vale more like the unbiased autocorrelation function, i.e., $\tilde{m}_c(k) = (1+0.002k)m_c(k)$ Supposed that p_c happens to be

$$p_c = \arg\max_k \left\{ \tilde{m}_c(k) \middle| 21 \le k \le 147 \right\}. \quad (10)$$

We regard p_c as the pitch period in case the frame is claimed as voiced speech. However, the analysis frame is categorized as the unvoiced type if either one of the following criteria is met.

(i) $R_{ss}(0) < 240$ (11-1)

(ii) $\left(R_{ss}(1) < 0.3R_{ss}(0)\right)$ & $\left(\tilde{m}_c(p_c) < 185\right)$ (11-2)

(iii) $\left(\left|p_c - p_p\right| > 0.15p_p\right)$ & $\left(\tilde{m}_c(p_c) < 160\right)$ (11-3)

where p_P denotes the pitch period obtained from the previous frame. In Eq. (11), criterion (i) is used to assess the energy level of the speech segment, while criterion (ii) is adopted to examine the spectral tilt. On the other hand, criterion (iii) is aimed at the discrimination of unvoiced conditions against abnormal pitch variations. We have tried out these conditions for a variety of speech utterances. The resulting performance is quite satisfactory.

To summarize this section in brief, our computational efficiency in the analysis phase originates from algorithmic modifications in manifold. Firstly, the speech ensemble is degenerated into a lengthy word, within which each bit represents the sign of a speech sample. As a result, the autocorrelation function necessitated by the pitch detection is substituted by a bit-wise equivalence operation between two words followed by summing up the number of 1's. Secondly, in contrast with most vocoders that employed a 10th-order LP analysis, the proposed coder adopts an 8th-order predictor cascaded by a 1st-order emphasis filter. Because the order directly reflects the computational requirements, such an arrangement not only makes the derivation of LP coefficients easier but also reduces the amount of bits required to encode spectral properties. Moreover, the choice of an 8th-order predictor allows a simple closed-form approach to derive the LSP parameters.

5. Spectral quantization

As indicated by Table I, there are 70% of bits reserved for spectral quantization. It is therefore advantageous for us to devote one section to this issue. The conversion between the LP coefficients and LSP parameters is essential in consideration of fewer bits for spectral quantization. The popularity of LSP parameters for spectral representation results from its superiority in stability check, excellent interpolation properties, and relative insensitivity to quantization errors [8]. Since the cosine functions of LSP parameters are directly applicable to the synthesis filter, we encode them using a 28-bit scalar quantizer with bits allocated according to the sequence, {4,3,4,3,4,3,4,3}. Notice that the predictive vector quantization is not considered here because it demands a great deal of computation and memory space.

To obtain the optimal nonuniform quantizer, the well-known generalized Lloyd algorithm is used [9]. Our training data consist of 68314 speech frames extracted from Mandarin sentences uttered by 10 speakers (5 males and 5 females). Another 22112 sample frames extracted from a different set of speakers and sentences are prepared for verifying the competence of the trained quantizer. Table II presents the results in terms of spectral distortion between the actual and quantized cosine function of LSP parameters. It is shown that the interlacing strategy for bit assignment comes up with an average SD of 0.93 dB. Among the distortion measures from all the training data, only 1.45% of them exceed 2 dB and 0.0029% are beyond 4 dB. In particular, we observe no significant difference for the SD's measured either inside or outside the training set. Apparently, this scalar quantizer achieves a transparent quantization [10] of spectral information for sure.

It ought to be noted that the bit assignment plays an important role in reducing the average spectral distortion. We have attempted a variety of bit allocations other than {4,3,4,3,4,3,4,3}, but ended up with worse results. Such consequences can be best

understood by inspecting Table III, which indicates the SD's contributed by each individual parameter. In the case under study, only one parameter was quantized at a time using either 3 or 4 bits and the other parameters remain intact. It is evident in Table III that not only the SD measures with odd-indices are larger than that with adjacent even indices, but also the improvement is relatively significant for the quantization of odd parameters when the number of bits is increased from 3 to 4. This suggests assigning more bits to quantize odd-indexed cosine function of LSP parameters.

6. Architecture for speech synthesis

Synthetic speech is the result attained by feeding the excitation (either the glottal pulse or random noise) to a synthesis filter. The synthesis of unvoiced speech is straightforward since the excitation is accessible from a random number generator. The synthesis of voiced speech is rather complicated because we have to modulate the pitch period apart from replicating the glottal features.

In light of the speech production model described in Fig. 1, the architecture for hardware implementation is illustrated in Fig. 8. It consists mainly of a white noise generator, a glottal pulse codebook, three interpolators, a gain codebook, a LSP synthesis filter, and a LSP parameter decoder.

The format of the input data utilized in the proposed architecture is consistent with that listed in Table I. For each frame of 25 ms, a packet of 40 bits is converted to modeling parameters before synthesizing into 200 speech samples. The voicing parameter is used to determine whether the excitation should be drawn from the glottal pulse modulator or from the white noise generator. The gain of the excitation is quantized based upon a codebook of a 5-bit size. We multiply the selected excitation source by the decoded gain to adjust the vocal intensity.

As we use 28 bits to encode the cosine functions of eight LSP parameters, an associated decoder is required prior to loading into the synthesis filter. Also, an interpolation scheme is employed in our proposed architecture to smooth the transition of speech synthesis. All the components are illustrated in the following subsections.

A. Glottal Pulse Codebook

The glottal pulse codebook contains a pulse-like waveform with glottal phase characteristics. Steps of establishing the glottal codebook are as follows. First, a prototype of the glottal pulse for one pitch period is extracted from the prediction residual of a sustained vowel /a/. Next, this prototype is thoroughly decolorized by unifying its magnitude discrete Fourier transform (DFT) while leaving the associated phase DFT unchanged. The ensemble reserved in the codebook is obtained by circularly shifting the main excitation to the beginning position. Since the range of pitch period presumably varies from 21 to 147 samples, we adopt an overlap-and-add approach [11] to adjust the pitch period by

$$v(i)=\begin{cases}\frac{1}{N-1}\left((N-1-i)\times w(i)+i\times w(L-N+i)\right), & \text{if } L\ge N\\ w_1(i)+w_2(i), & \text{if } L<N\end{cases}$$

$$\text{for}\quad 0\le i<N \qquad (12\text{-}1)$$

$$w_1(i)=\begin{cases}w(i)\times\frac{L-1-i}{L-1}, & i=0,1,2,\cdots,L-1\\ 0, & i=L,L+1,\cdots,N-1\end{cases} \qquad (12\text{-}2)$$

$$w_2(i)=\begin{cases}0, & i=0,1,2,\cdots,N-L-1\\ w(i)\times\frac{i-N+L}{L-1}, & i=N-L,N-L+2,\cdots,N-1\end{cases} \qquad (12\text{-}3)$$

where $v(i)$ is the resulting excitation, $w(i)$ represents the prototype of the glottal pulse with a length of L, and N denotes the targeted pitch period. In Eqs. (11-2) and (11-3), $w_1(i)$ and $w_2(i)$ can be regarded as the $w(i)$ weighted by forward and backward triangular windows followed by zero-padded at the end and beginning positions, respectively. Fig. 9 illustrates the process for pitch adjustment.

B. Random Noise Generator

The hardware implementation of a pure random noise generator is very difficult. A pseudo random noise generator is commonly used instead. Here we adopt the linear feedback shift register (LFSR) to produce a random sequence. Since each synthesis frame only needs to fetch 200 points, the maximal length sequence does not require a large value to meet the characteristics of randomness. The LFSR

structure thus suits our purpose very well for generating the noise excitation for speech synthesis.

C. Interpolator

The interpolator is employed to smooth the transition across frames. When the synthesis steps forward in a frame, modeling parameters are interpolated according to the current location by

$$\theta^k = \begin{cases} 0.875\phi + 0.125\varphi, & k = 0 \\ 0.625\phi + 0.375\varphi, & k = 1 \\ 0.375\phi + 0.625\varphi, & k = 2 \\ 0.125\phi + 0.875\varphi, & k = 3 \end{cases} \quad (13)$$

where ϕ and φ denote the decoded parameters of the previous and current frames, respectively. θk is the interpolated parameter for synthesis in the kth subframe.

To simplify the complexity of hardware circuitry, we modify the original weighting values of Eq. (12) by using integer manipulation rather than regular arithmetic. That is, we replace the weighting values {0.875, 0.625, 0.375, 0.125} by its fractional counterparts {7/8, 5/8, 3/8, 1/8}. This change will allow the interpolation performable on a simple hardware circuit, since it only requires fundamental operations such as shifting and summation. For instance, the multiplication of 1/8 can be achieved by arithmetically shifting the input value to the right by 3 bits, and the multiplication of 7/8 is obtainable by subtracting such a shifted version from itself.

D. Gain Codebook

The gain codebook used in our proposed speech synthesis allocates 32 ($=2^5$) words in memory. This codebook reflects the power of the synthesized speech. For simplicity, it is implemented by using a lookup table within a ROM.

E. LSP Synthesis Digital Filter

In LSP speech synthesis, a digital filter is constructed based on the LSP parameters [12]. Fig. 10 illustrates the signal flow graph of the LSP synthesis filter of order 8. It is readily seen that such a structure is very suitable for hardware implementation due to its regularity. This particular filter consists of 8 trunk circuits. The z-transformation of each trunk circuit appears to be $1 - c_i z^{-1} + z^{-2}$, where $c_i = 2\cos\omega_i$ and ω_i is the ith LSP parameter. Since this new circuit holds its regularity, the whole LSP computation is accomplishable by using the same hardware structure except for the final summation (see Fig. 10).

7. Performance evaluation

A mean opinion score (MOS) listening test is performed on a total of sixteen utterances gathered from 4 speakers (2 male, 2 female), each delivering four speech sentences. Apart from the synthetic results of the proposed coder, the outcomes from the 2400 bps LPC-10e vocoder (FS-1015) [13], 4800 bps CELP coder (FS-1016) [14], and 2400 bps MELP [15] coder are also provided as three baselines. Since the synthetic speech signals obtained from the other coding schemes are in floating-point format, the Matlab version of the proposed algorithm is adopted for comparison. There are 18 listeners participating in the MOS test. During the test, every listener is equipped with a pair of high-fidelity earphones, which are connected to the sound card on a PC. A program is designed to play back the speech stimuli in random order and to collect the answer picked by the listeners. The results are shown in Table IV. It is to our surprise that the mean opinion scores of the synthesized utterances for the males are inferior to that for the females for all kinds of speech coders. According to the listeners' opinions, this is probably due to the fact that the vocal quality of one male sounds a little boring. As expected, the MELP coder holds the best average score, followed by the CELP coder. The proposed coding scheme is slightly worse than the LPC-10e vocoder. We believe that the faults in our proposed coder are mostly attributable to the occasional errors in pitch detection and the over-simplification in spectral characterization. Although the problems are generally ameliorable by means of more advanced techniques, the resolutions are often accompanied with the increase in computational burden. Since there always exists a tradeoff between algorithmic complexity and synthetic quality in concern with hardware implementation, a wise strategy would be the development of a reconfigurable system inside which each module is designed subject to hardware specifications.

Besides algorithm development, the functional analysis of the architecture for speech coding has been

justified by using Verilog HDL. The obtained outcome of the Verilog simulator does not deviate from the fixed-point analysis based on the C++ program. Such a result evidences the competence of the proposed architecture. Nevertheless, the fixed-point rendition of our speech coder seems to suffer degradation attributable to numerical round-offs and truncation errors. Amelioration of such deficiency is possible at the cost of increasing size in bits. To further examine our proposed architecture, we try it out on a DSP development board mounted with an FPGA device containing approximately 1,500,000 gates counts. After completing the hardware compilation, we find that the proposed architecture required approximately 16,733 logic elements to synthesize the intended functionality, while a single such FPGA offers 51,840 logic elements in totality. About 85% of the spent hardware resources are ascribable to speech analysis.

8. Conclusion

A hardware-oriented algorithm with its hardware implementation for speech coding at 1.6 Kbps has been presented in this paper. To make the developed algorithm suitable for chip design, we have trimmed the algorithm and taken away complicate computation. The performance of the proposed 1.6 Kbps speech coder has been subjectively evaluated. The preliminary results show that our proposed coder is capable of attaining an acceptable quality close to that of LPC-10e.

We have gone through algorithm modification, fixed-point analysis, hardware design, and system verification by making use of several development tools. It is confirmed that the resultant algorithm is realizable using fundamental arithmetic and bit-wise operations. The only exception is the square root function, which is nonetheless transferred into a series of bit-wise shifts and comparisons. The output observed in the simulation of Verilog modules consists with our fixed-point analysis. However, the fixed-point rendition may suffer quality degradation due to round-off errors. Further investigation is required in order to determine how to resize parameters in the module so that the perceived distortion can be effectively reduced.

Since our proposed architecture only requires a moderate amount of hardware resources, it is suited for the ASIC or FPGA implementation. Such an architecture is appropriate for the design of electronic devices which demand voicing interfaces, such as those previously mentioned in the introduction.

Acknowledgment

This work was supported by the National Science Council, Taiwan, ROC, under Grant NSC91-2622-E-197- 004-CC3.

References

[1] D. G. Childers and H. T. Hu, “Speech synthesis by glottal excited linear prediction,” J. Acoust. Soc. Am., vol. 96, no. 4, pp. 2026-2036, 1994.

[2] H. T. Hu and H. T. Wu “A glottal-excited linear prediction (GELP) Model for low-bit-rate speech coding,” Proc. Natl. Sci. Counc. ROC(A)., vol. 24, no. 2, pp. 134-142, 2000.

[3] H. T. Hu, F. J. Kuo, and H. J. Wang, “A pseudo glottal excitation model for the linear prediction vocoder with speech signals coded at 1.6 kbps,” IEICE Trans. Inf. & Syst., vol. 83, no. 8, pp. 1654-1661, 2000.

[4] 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, 637-655, 1971.

[5] J. Makhoul, “Linear prediction: A Tutorial Review,” Proc. IEEE, vol. 63, pp.561-580, 1975.

[6] M. T. Tommiska, “Area-efficient implementation of a fast square root algorithm,” in Proc. the 2000 Third IEEE International Caracas Conference on Devices, Circuits and Systems, 2000, pp. S18/1 -S18/4.

[7] C. H. Wu and J. H. Chen, “A novel two-level method for the computation of the LSP frequencies using a decimation-in-degree algorithm,” IEEE Trans. Speech Audio Process., vol. 5, no. 2, pp. 106-115, 1997.

[8] F. K. Soong and B. H. Juang, “Optimal quantisation of LSP parameters,” IEEE Trans. Speech Audio Process., vol. 1, no. 1, pp. 15-24, 1993.

[9] Y. A. Linde, Y. A. Buzo, and R. M. Gray, “An algorithm for vector quantization design,” IEEE

Trans. Commun., vol. 28, no. 1, pp. 84-95. 1981.

[10] K. Paliwal and B. Atal, “Efficient vector quantization of LPC parameters at 24 bits/frame’, IEEE Trans. Speech Audio Process., vol. 1, no. 1, pp. 3-14, 1993.

[11] E. Moulines and F. Charpentier, “Pitch-synchrounous waveform processing techniques for text-to-speech synthesis using diphones,” Speech Commun., vol. 9, nos. 5/6, pp. 453-467, 1990.

[12] S. Furui, Digital processing, synthesis, and recognition, Marcel Dekker, Inc., New York and Basel, 1989.

[13] T. E. Tremain, “The government standard linear predictive coding algorithm: LPC-10,” Speech Tech. Mag., pp. 40-49, 1982.

[14] J. P. Campbell, T. E. Tremain, and V. C. Welch, “The federal standard 1016 4800 bps CELP Voice Coder,” Digital Signal Process, vol. 1, no. 3, pp. 145-155, 1991.

[15] L. M. Supplee, R. P. Cohn, J. S. Collura, and A. V. McCree, “MELP: the new federal standard at 2400 bps,” Proceedings of ICASSP, 1591-1594, 1997.

Table I : BIT Allocation FOR THE PROPOSED 1.6 KBPS SPEECH CODER

Sampling Rate 8 KHz **Frame Rate: 25 ms (200 samples/frame).**	
Parameter	**bits/frame**
Voicing & Pitch	7
Gain	5
Spectrum	28
Total	40

Table II: Spectral distortion of the scalar quantizer with respect to two different bit allocation strategies

Bit allocation	**Data set**	**Spectral Distortion [dB]**	**Outliers (%)**	
			2-4 dB	> 4 dB
{4,3,4,3,4,3,4,3}	within training	0.934	1.45	0.0029
	out-of-training	0.937	1.38	0.0000
{4,4,4,4,3,3,3,3}	within training	1.005	2.91	0.0498
	out-of-training	1.078	4.18	0.0814

Table III: Influence due to quantization with respect to each individual cosine function of line spectral frequency

Parameter		1^{st}	2^{nd}	3^{rd}	4^{th}
Quantization error (dB)	**3 bits**	0.360	0.279	0.396	0.378
	4 bits	0.181	0.149	0.206	0.193
Improvement due to an extra bit [dB]		0.179	0.130	0.190	0.185
		5^{th}	6^{th}	7^{th}	8^{th}
		0.416	0.357	0.385	0.264
		0.216	0.184	0.200	0.136
		0.200	0.173	0.185	0.128

Table IV: MOS's with respect to the speech synthesized by FOUR different SPEECH coders

	Proposed 1.6 Kbps GELP	**2.4 Kbps LPC-10e**	**4.8 Kbps CELP**	**2.4 Kbps MELP**
Male	1.958	2.035	3.139	3.681
Female	2.493	2.861	3.569	3.771
Overall	2.226	2.448	3.354	3.726

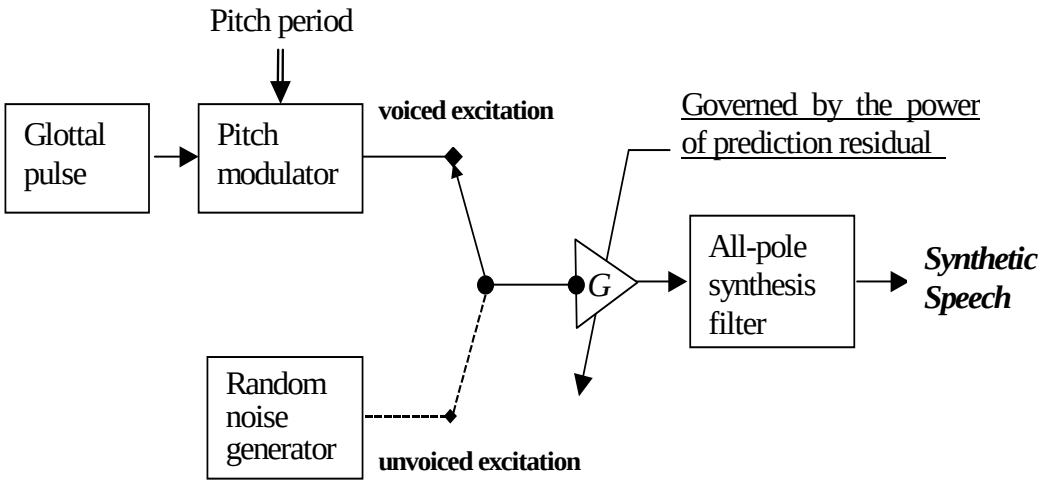

FIG. 1. A SIMPLIFIED MODEL OF SPEECH PRODUCTION MECHANISM.

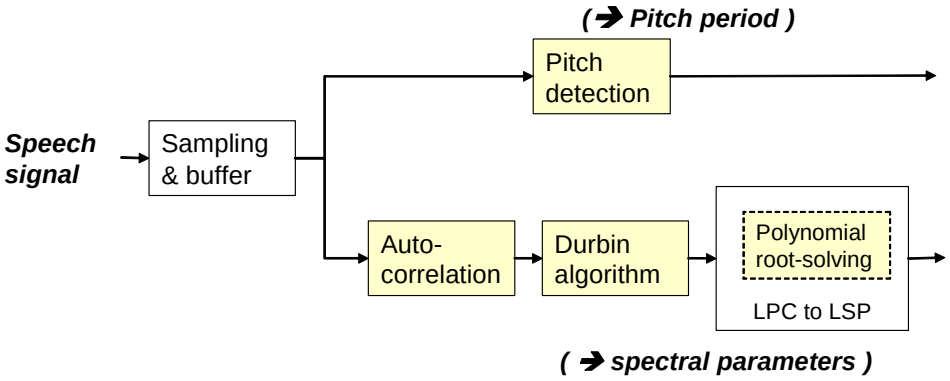

FIG. 2. KEY MODULES FOR ASIC DESIGN OF SPEECH ANALYSIS.

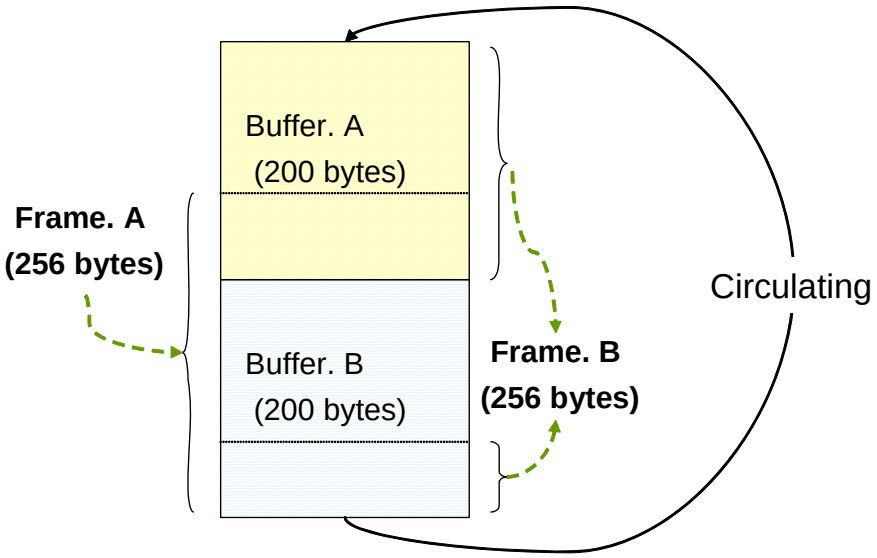

FIG. 3. DOUBLE BUFFERS FOR SPEECH SAMPLING.

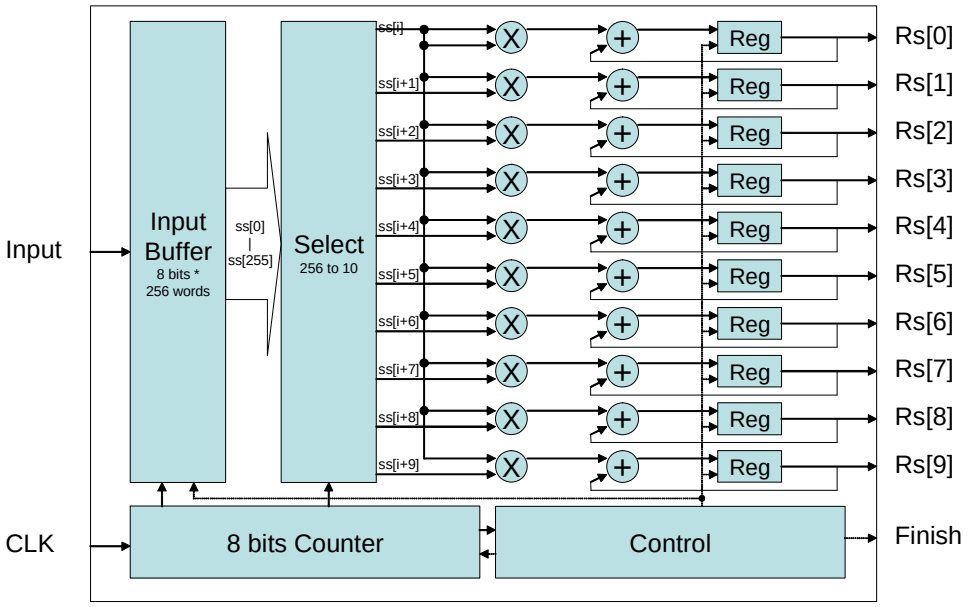

FIG. 4. DIAGRAM FOR COMPUTATION OF AUTOCORRELATION FUNCTION IN PARALLEL.

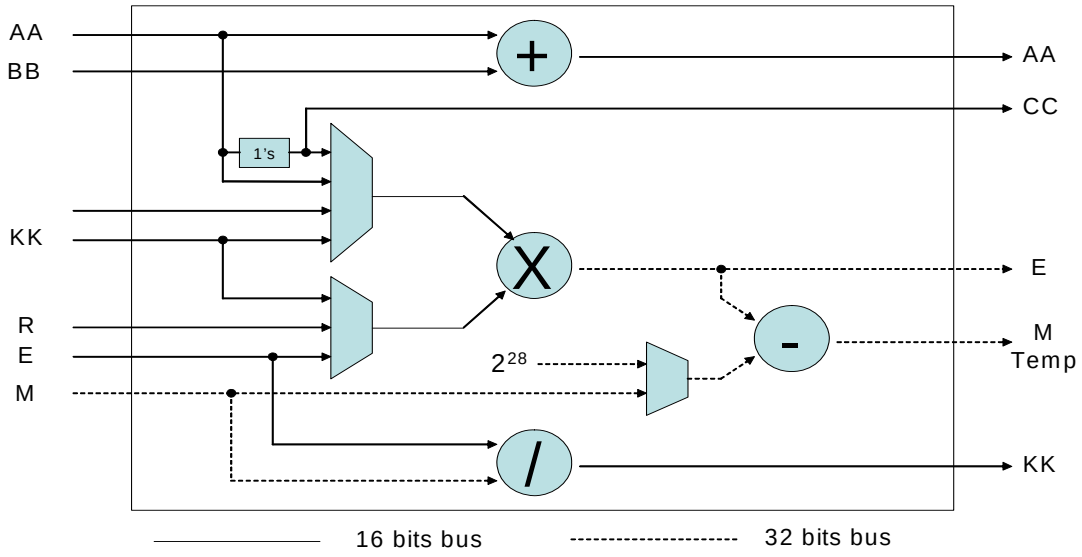

FIG. 5. LEVINSON-DURBIN ITERATIVE MODULE.

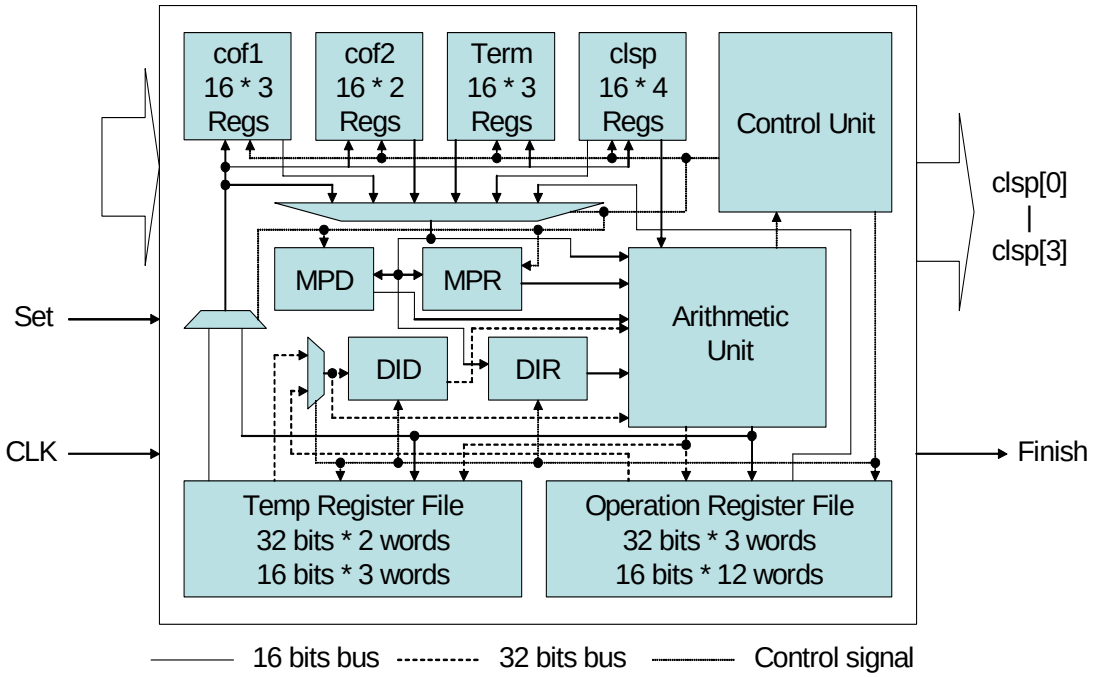

FIG. 6. DIAGRAM FOR ROOT-SOLVING MODULE.

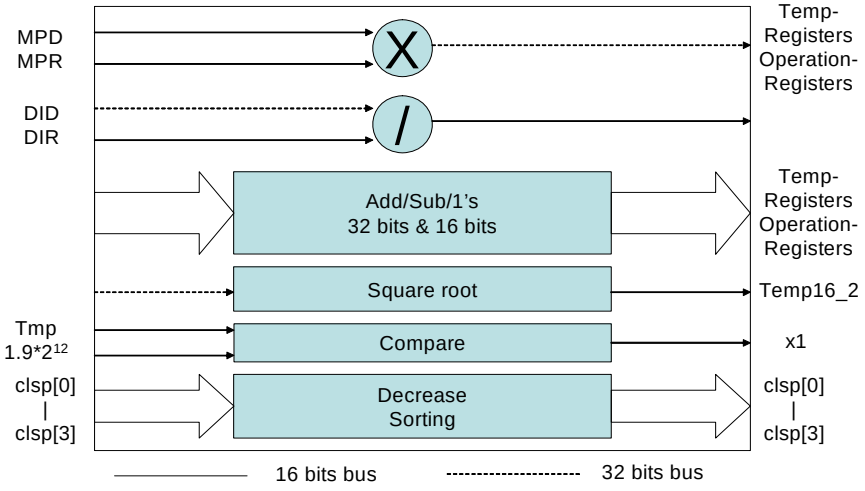

FIG. 7. PROCESSING UNITS THAT CONSTITUTE THE ARITHMETIC UNIT OF THE ROOT-SOLVING MODULE PRESENTED IN FIG. 6.

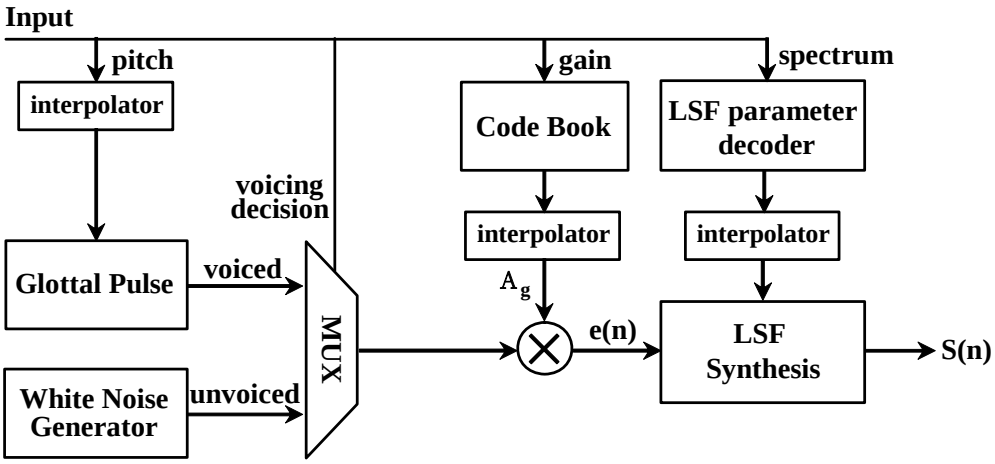

FIG. 8. BLOCK DIAGRAM OF SPEECH SYNTHESIS.

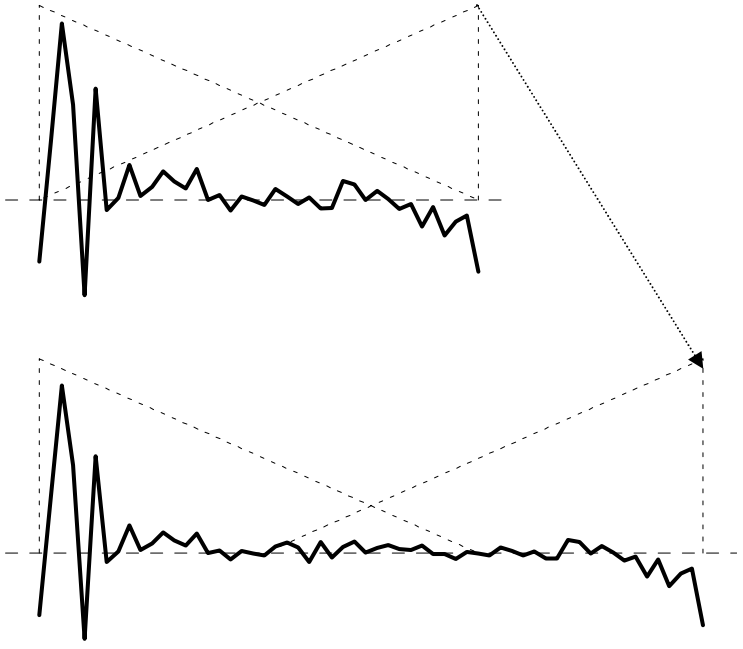

FIG. 9. PITCH ADJUSTMENT FOR THE GLOTTAL PULSE.

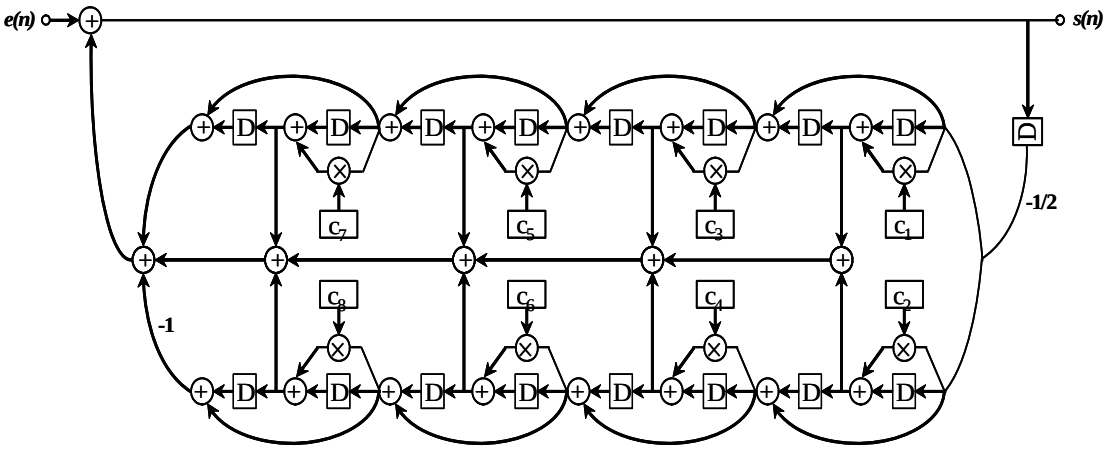

FIG. 10. SIGNAL FLOW GRAPH OF LSP SYNTHESIS DIGITAL FILTER.