International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue (4) July, Technovision-2014, ISSN 2249–071X

Technovision-2014: 1st International Conference at SITS, Narhe, Pune on April 5-6, 2014

Speech Compression using Analysis by Synthesis

Minal Mulye
M.E. Student, Department of E & TC Engineering,
Smt. Kashibai Navale College of Engineering, Pune
(MS) India- 411041
Email: minalmulye@gmail.com

Prof. Sonal K. Jagtap
Department of E & TC Engineering,
Smt. Kashibai Navale College of Engineering, Pune
(MS) India-411041.
Email: skjagtap.skncoe@sinhgad.edu

***Abstract* – Linear prediction plays a fundamental role in all aspects of speech. Its use seems natural and obvious since for a speech signal the value of its current sample can be well modeled as a linear combination of its past values. Calculation for predictor coefficients with the help of automatic code generation gives the solution for early and efficient computing. Automatic code generation is a fast paced technology with lots of new capabilities and customer experiences. In this, Simulink is the leading environment for modeling and simulating a broad range of dynamic systems. It is the foundation for Model-Based Design, with support for multi-domain modeling and simulation, automatic code generation for production use and real-time testing, and a range of capabilities for verification and validation. With its open architecture and APIs, the integration of models from other tools or languages such as C\C++ is possible. Also embedded coder with Simulink can generate C code for target specific applications. The proposed system implements a model based design by using the linear prediction coefficients of the encoded speech data and prove to be the promising method for speech compression.**

***Keywords* – Simulink, Model Based Design, Embedded Coder.**

I. INTRODUCTION

In communication systems, service providers are constantly met up with the challenge of accommodating more users within a limited allocated bandwidth. Hence speech coder that provides good quality speech at low bit rates is needed. Speech coding is a method of reducing the amount of information desirable to represent a speech signal. The purpose of all speech coding systems is to transmit speech with the highest possible quality using the least possible channel capacity. This means speech coding provides a solution to the handling of the huge and increasing volume of information that needs to be carried from one end to another. Furthermore, not only communication but voice storage and multimedia applications also require digital speech coding. So to deal with this scenario a large number of speech coding paradigms have been proposed. Among them Linear Predictive Analysis-by-synthesis speech coder is one of the most effective modern coder. They use most of the available information from the speech signal to improve the quality and reduce the bit-rate. In particular, they make use of acoustic noise masking, acoustic frequency resolution, acoustic phase insensitivity, syllabic energy variation, the long term vocal tract properties, and pitch information in the coding process. This makes these coders among the best-quality, lowest bit-rate, and most computationally demanding of speech coding techniques. In general, there is always a positive correlation between coder bit-rate efficiency and the algorithmic complexity required to attain it. The processing delay and the implementation cost both are directly proportional to the complexity of algorithm. The more complex an algorithm, the more is processing delay and cost of implementation. [1] To overcome these problems Model-based Design could be of great use. Basically Model-Based Design is used to model complex systems and simulate them on the desktop environment for analysis and design purposes. It supports a broad range of C/C++ code generation applications that include objective simulation, rapid control prototyping, hardware-in-the loop testing, and production or embedded code deployment.

A model represents a dynamic system whose response at any time is a mathematical function based on its inputs, current state, and current time. With specified competitive temporal and cost constraints, developing a system on time and within budget requires an organized approach to design and realization. This ensures that the final system meets the initial requirements and lets engineering teams with different specializations work together and communicate between stages in the overall process. In addition, this approach also ensures that the design process and the final artifact are documented for maintenance and future development. By using models in the early design stages, executable specifications can be created that enable to immediately validate and verify specifications aligned with the requirements. This approach allows detecting errors earlier. Models can be used to communicate between engineering teams with different specializations, allowing them to work together and to communicate between stages in the overall process. Moreover, initial design models can be incrementally extended to include increasing implementation detail. Thus, model-based design allows experimenting with different design alternatives, even in very early conceptual design stages, while having an executable specification and taking detailed implementation effects into account. This is in contrast to a document centered approach where each of the design stages generates new models of the same system under design from the specification of the previous design stage. [2]

Even more sophisticated is the use of model transformation to generate different representations of the

same system, which further minimizes the effort to move from one design stage to another. In particular, the use of automatic code generation technology and hardware-in-the-loop testing improve errors introduced during manual implementation and realization tasks and shortens the path to product delivery by generating code for testing, calibration, and the final production. An important benefit of this model-based design paradigm is the traceability of design decisions all the way down to the implementation. So test results can be directly interpreted as high-level design decisions. Finally, even though electronic models are easier to navigate than paper documents, the formal system design process still requires detailed documentation. Advanced tools allow automatic generation of this documentation from a model while model-based design forces the design process as well as the final product to be documented for maintenance and future developments. [3]

II. Literature Survey

Jeremy Bradbury in his work asserts that Linear Predictive Coding is an analysis/synthesis technique to lossy speech compression that attempts to model the human production of sound instead of transmitting an estimate of the sound wave. Linear predictive coding achieves a bit rate of 2400 bits/second which makes it ideal for use in secure telephone systems. Secure telephone systems are more concerned that the content and meaning of speech, rather than the quality of speech, be preserved. The trade off for LPC's low bit rate is that it does have some difficulty with certain sounds and it produces speech that sound synthetic. [1]

J. Benesty, J.Chen and Y.Huang have detailed the most important results in linear prediction for speech. They have explained the principle of forward linear prediction and have shown that the optimal prediction error signal tends to be a white signal. They extended the principle of forward linear prediction to backward linear prediction and derived the Cholesky factorization of the inverse correlation matrix. The classical Levinson– Durbin algorithm, which is a very efficient way to solve the Wiener–Hopf equations for the forward and backward prediction coefficients, was explained. They have enlightened the idea behind the lattice predictor and shown how the spectrum of a speech signal can easily be estimated thanks to the prediction coefficients. They have given some notions of linear interpolation and line spectrum pair polynomials with demonstration on how the condition number of the correlation matrix is related to the optimal interpolators. They have generalized some of these ideas to the multichannel case. [2]

Wei Li, Anu Sridhar and Tina Teng in their thesis "Comparison of Speech Coding Algorithms: ADPCM, CELP and VSELP" state that speech coders differ broadly in their approaches to achieve signal compression. Based on the constraints by which compression is achieved they are classified as: a waveform coders or analysis-by-synthesis coders. Waveform coders essentially strive to reproduce the time waveform of the speech signal as closely as possible. They have the advantage of being robust for a wide range of speech characteristics and for noisy environments. All these advantages are preserved with minimal complexity, and in general these coders achieve only moderate economy in transmission bit rate. Analysis-by-synthesis speech coders are among the newest and most effective of modern speech coders. They make use of aural noise-masking, aural frequency resolution, aural phase insensitivity, syllabic energy variation, the long-term vocal tract properties, and pitch information in the coding process.

Various attempts have been made to encode the residue signal in an efficient way. The most successful methods use a codebook, a table of typical residue signals, which is set up by the system designers. In operation, the analyzer compares the residue to all the entries in the codebook, chooses the entry which is the closest match, and just sends the code for that entry. The synthesizer receives this code, retrieves the corresponding residue from the codebook, and uses that to excite the formant filter. Schemes of this kind are called Code Excited Linear Prediction. For CELP to work well, the codebook must be big enough to include all the various kinds of residues. But if the codebook is too big, it will be time consuming to search through, and it will require large codes to specify the desired residue. The biggest problem is that such a system would require a different code for every frequency of the source (pitch of the voice), which would make the codebook extremely large. According to Feng Luo and Zhihui Huang, the custom embedded target enables the user to generate embedded C codes from Simulink/Stateflow models and download them into the micro-controllers conveniently. Through the tests in visual C++ environment, they have concluded that automatic generated code is accurate and efficient. However, when the generated code was compiled in CodeWarrior and downloaded into micro-controller through BDM, the computing results are correct but the execution time was slower than the code written by hand. The main reason is: the code generated has lots of function callings, additional codes and extern variables, and the expressions are not concision. But these disadvantages could be reduced as the code length increase. Except for the code efficiency, designer also should take readability and the ability to adapt fast development into consideration. That's why the application of auto code generation plays a more and more important role in today's development process of embedded systems. [6]

III. Methodology

Proposed methodology gives a thought on Model- Based Design. It is used throughout the system development life

cycle because it focuses on producing design flows that lead toward continuous verification and validation of requirements, designs, and implementations. This approach is important for formal software processes. The flow chart for the model based design is as follows including all the steps [4]:

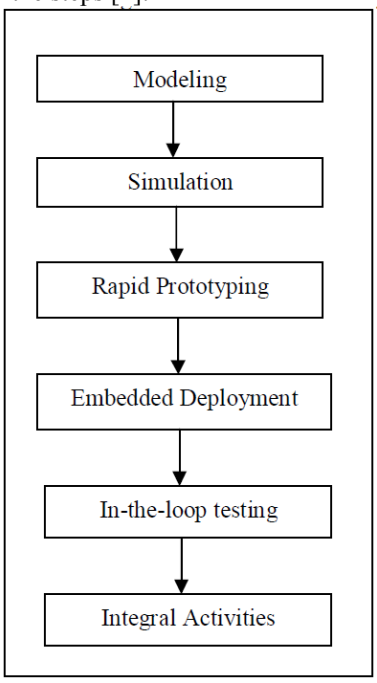

Fig.1. Flow chart for model based design

A block diagram model of a dynamic system is represented schematically as a collection of blocks interconnected by lines that represent signals. The signals are the inputs, outputs, and states of the blocks to be processed. Simulation can be accomplished in two ways. One way is to use an in-memory representation of the systems and execute the simulation in an interpretive mode. The other way is to generate code from the model and execute the code using a technique of simulation through code generation. In bypass rapid prototyping, code is generated from the controller or algorithm model. The code is then cross-compiled and the success is declared when performance requirements are met, proving that the new algorithm is feasible. A detailed software design activity is often undertaken to convert the controller model to a detailed, executable software specification. Embedded code is then generated for the detailed controller model and downloaded to the actual embedded microprocessor as part of the production software build. Model testing involves a formal approach to the creation and execution of test cases. Special blocks, such as signal builders and assertions, facilitate this type of formal test procedure. Model-Based Design environments automate the generation of documentation from models. Documentation is done in template form letting users specify the content of each documentation section.

IV. MODEL OF THE SYSTEM

Following figure shows the general flow of proposed system. Speech signal is processed with LPC technique and coefficients are obtained. This output is given as input to the Real-time platform by using Embedded C code generated with the help of Real Time Workshop-Embedded Coder tool in MATLAB.

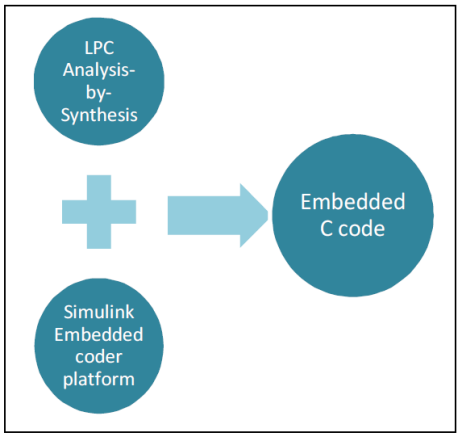

Fig.2. Model diagram of the system

LPC starts with the assumption that the speech signal is produced by a buzzer at the end of a tube. The glottis (the space between the vocal cords) produces the buzz, which is characterized by its intensity (loudness) and frequency (pitch). The vocal tract (the throat and mouth) forms the tube, which is characterized by its resonances, which are called formants. It analyzes the speech signal by estimating the formants, removing their effects from the speechsignal, and estimating the intensity and frequency of the remaining buzz. The process of removing the formants is called inverse filtering, and the remaining signal is called the residue. The numbers which describe the formants and the residue can be stored or transmitted somewhere else. LPC synthesizes the speech signal by reversing the process: use the residue to create a source signal, use the formants to create a filter (which represents the tube), and run the source through the filter, resulting in speech. Because speech signals vary with time, this process is done on short chunks of the speech signal, which are called frames.

Usually 30 to 50 frames per second give intelligible speech with good compression. The basic problem of the LPC system is to determine the formants from the speech signal. This can be depicted from Fig.3, which shows the original plot of word “matlab” uttered by female voice. The basic solution is a difference equation, which expresses each sample of the signal as a linear combination of previous samples. Such an equation is called a linear predictor, which is why this is called Linear Predictive Coding. The coefficients of the difference equation (the prediction coefficients) characterize the formants, so the LPC system needs to estimate these

coefficients. The estimate is done by minimizing the mean-square error between the predicted signal and the actual signal.

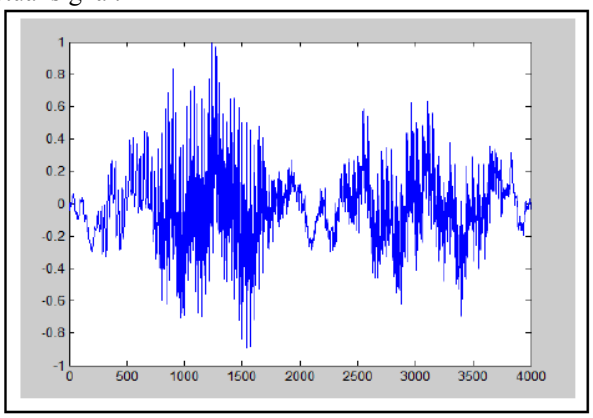

Fig.3. Speech spectrum of word "matlab"

Basically LPC analyses the input speech signal and does the feature selection on the account of speech intelligibility. Pre-processing aims at simplifying subsequent processing operations without losing relevant information. With the help of Simulation, verification of system accountability can be checked. RMS values of filter coefficients and predictor coefficients are easily analyzed with frequency and energy spectrum.

From Simulink library, blocks are decided and dragged into model workspace. Now the parametersare set by exploring each and individual block set to model parameters configuration. The mathematical treatment for finding the reflection coefficients, residual coefficients, etc is given by following equation.

$$H(z) = \frac{G}{\prod_{i=1}^{p}(1-a_i z^{-1})(1-a_i^{*} z)} \quad \ldots\ldots\ldots (1)$$

Here 'G' represents the gain, 'p' is the order of the system and 'ai' denotes the poles.

This equation represents the transfer function of the system. As human auditory system is more sensitive to poles than zeros, equation relies on all-pole model structure. In practice, it involves the computation of a matrix of coefficient values, and the solution of a set of linear equations.

V. SIMULATION STUDY

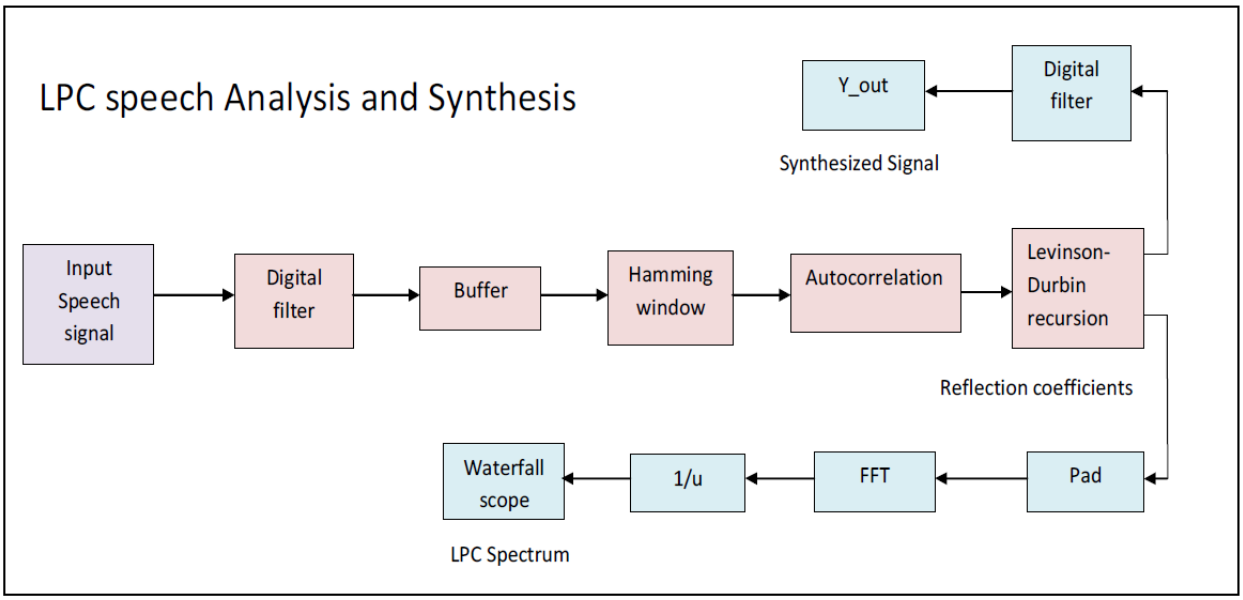

Fig.4. Simulation diagram of the system

The very first step of proposed methodology is simulation of generalized block diagram. In this simulation, the speech signal is divided into frames of size 20 ms (160 samples), with an overlap of 10 ms (80 samples). Each frame is windowed using a Hamming window. Eleventh-order autocorrelation coefficients are found, and then the reflection coefficients are calculated from the autocorrelation coefficients using the Levinson-Durbin algorithm. The original speech signal is passed through an analysis filter, which is an all-zero filter with coefficients as the reflection coefficients obtained above. The output of the filter is the residual signal. This residual signal is passed through a synthesis filter which is the inverse of the analysis filter. The output of the synthesis filter is the original signal. Digital filter is used because it independently filters each channel of the input over time using a specified digital filter implementation. We can specify filter coefficients using either tunable mask dialog

parameters or separate input ports, which are useful for time-varying coefficients.

Buffer converts scalar samples to a frame output at a lower sample rate. We can also convert a frame to a smaller or larger size with optional overlap. Hamming window block generates a window function and/or applies a window function to an input signal. For some fixed-point modes, the fraction length or slope of the window values is automatically set for the best possible precision given the real-world values and word length of the window values. This is equivalent to the "Best Precision: Matrixwise" scaling option used in some Simulink fixedpoint blocks. Autocorrelation block computes the autocorrelation along the first dimension of an N-D sample-based input or along each column of a frame based input. When "Compute all non-negative lags" is selected, it computes using lags in the range [0, size (input, 1)-1]. Otherwise, it computes using lags in the range [0, maxLag], where you specify the value of maxLag in the "Maximum non-negative lag" parameter.

Levinson- Durbin block solves Hermitian Toeplitz system of equations using the Levinson-Durbin recursion. Input is typically a vector of autocorrelation coefficients with lag 0 as the first element. Outputs polynomial coefficients A, reflection coefficients K, and/or the prediction error power P. Padding is required as it appends or prepends a constant value to the input along the specified dimensions. Truncation occurs when the specified output dimensions are shorter than the corresponding input dimensions. After this, we can analyze the frequency spectrum obtained with Fast Fourier algorithm.

VI. Experimental Results

First step of the simulation is the Initialization. Here we initialize some of the variables like the frame size and file name and also instantiate the System objects used in processing. These objects also pre-compute any necessary variables or tables resulting in efficient processing calls later inside a loop. Create a System object to read from an audio file and determine the file's audio sampling rate. Next step is to create an FIR digital filter System object used for pre-emphasis. Create a buffer System object and set its properties such that an output will be twice the length of the frame size with an overlap length of frame Size. Create an autocorrelation System object and set its properties to compute the lags in the range [0:12] scaled by the length of input. Create a System object which computes the reflection coefficients from auto-correlation function using the Levinson-Durbin recursion. We configure it to output both polynomial coefficients and reflection coefficients. The polynomial coefficients are used to compute and plot the LPC spectrum. Create an FIR digital filter system object used for analysis.

Similarly at the synthesis section, a system of digital filter is designed. Table 1 shows the standard bit allocation for LPC- 10 having a bit rate of 2.4 kbps. When compared to this standard, it is realized that the proposed system also achieves the same frame rate.

Table 1: Bit allocation for standard 2.4kbps LPC-10 speech coder

Sample rate	8 kHz
Frame size	180 samples
Frame rate	44.44 frames/second
Pitch	7 bits
Spectrum(5,5,5,5,4,4,4,4,3,2)	41 bits
Gain	5 bits
Spare	1 bit
Total	54 bits/frame
Bit Rate	54*44.44=2.4kbps

Table 2 shows the calculation for the proposed system. Though there is difference in number of allocated bits, overall but rate of 2.4 kbps is achieved.

Table 2: Bit allocation for proposed system

Sample rate	8 kHz
Frame size	160 samples
Frame rate	50 frames/second
Pitch	7 bits
Spectrum(3,4,4,4,4,3,3,3,3,3)	34 bits
Gain	7 bits
Total	48 bits/frame
Bit Rate	50*48=2.4kbps

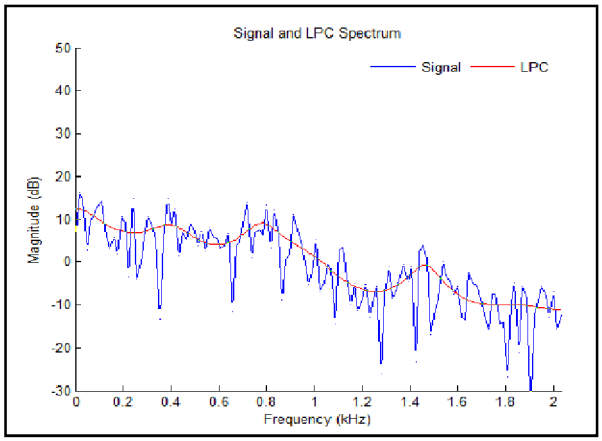

Fig.5. Graph for the word “matlab”

As can be seen from fig.5, LPC analysis produces an estimate smoothed spectrum, in which much of the influence in the excitation is removed. This holds true for the short frame of the signal.

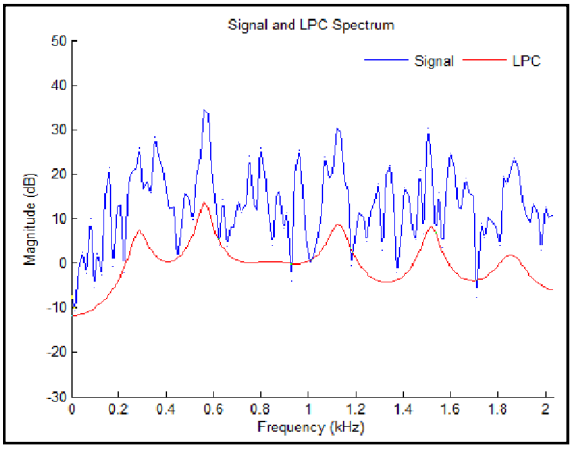

Fig 6: Graph for a music tune

For musical tune, linear predictive coding gives the results which are near to original but not exactly the same as original.

VII. Conclusion

Efficient speech compression can be achieved with the help of linear predictive coding (a type of analysis by synthesis coder). When combined with embedded coder technology, it generates the code that is easy to read, trace, and customize for required production environment. There are some optimizations inherent in the Embedded-C code format. It makes use of real time model data structure optimizes memory usage specifically for given model. Simplified calling interface reduces overhead and easily incorporate the generated code into hand-written application code.

Future Scope

When the generated code is compiled and checked it gives the correct computing results but the execution time is slower than the code written by hand. The main reason is: the code generated has lots of function callings, additional codes and extern variables, and the expressions are not concision. But these disadvantages could be reduced as the code length increase. Attempts can be made for the perfect, flawless yet fast algorithm development and hence the system.

Acknowledgement

I sincerely thank Prof. S.K Jagtap, Assistant professor, Smt. Kashibai Navale College of Engineering, Pune for her valuable guidance for all my research endeavors. The authors are indebted to the reviewers for their detailed review, valuable comments and constructive suggestions.

References

[1] Patrick A. Naylor, Anastasis Kounoudes, Jon Gudnason, and Mike Brookes. Estimation of glottal closure instants in voiced speech using the dypsa algorithm. Audio, Speech and Language Processing, IEEE Transactions, 15(1):34–43, 2007.

[2] Tim Fingscheidt, Suhadi, Sorel Stan, “Environment-Optimized Speech Enhancement”, IEEE transactions on audio, speech, and language processing, vol. 16, no. 4, 2008.

[3] Ren Chuanjun, Jiang Zhiwen,” Research and Application of Real-Time Simulation on Real-Time Workshop Computer Simulation”, vol. 24, no. 8, 2008.

[4] Yang Xiaozhong, An Jinwen, Cui Weng,” The Research on Application of Embedded Auto Code Generation”, Journal of Projectiles, Rockets, Missiles and Guidance, vol. 28, no.3, 2008.

[5] Vinod Reddy, Siva Nadarajah and George Beals, “Fixed- Point Modeling and Code Generation Tips”, by MATLAB Central, 2008.

[6] Feng Luo and Zhihui Huang,” Embedded C Code Generation and Embedded Target Development based on RTW-EC”, 978-1-4244-5540-9/10/© IEEE2010.

[7] Kadam V.K, Dr.R.C.Thool, “Performance Analysis of Optimization Tool for Speech Recognition Using LPC & DSK TMS3206711/13 Using Simulink & Matlab”, International Journal Of Computational Engineering Research (ijceronline.com) Vol. 2 Issue 5, September 2012.