Speech Silicon AM: An FPGA-Based Acoustic Modeling Pipeline for Hidden Markov Model based Speech Recognition

Jeffrey W. Schuster, Kshitij Gupta, Raymond Hoare
Department of Electrical and Computer Engineering
University of Pittsburgh, Pittsburgh, PA, 15233
{jws52, ksg3, hoare}@pitt.edu

Abstract

This paper presents the design of a FPGA-based hardware co-processor, based on the SPHINX 3 speech recognition engine from CMU; capable of performing Acoustic Modeling (AM) for medium sized vocabularies in real-time. By creating an input-driven pipeline for performing the calculations we were able to maximize the throughput of the system while simultaneously minimizing the number of pipeline stalls. Use advanced placement techniques enabled post place-and-route speeds even greater than those necessary for real-time operation while operating at maximum workload. Further, by using input control vectors all FSMs were removed from the design, greatly increasing the flexibility of the design. These results combined with the ability to reprogram the system for different recognition tasks serve to create a system capable of in a vast array of environments. Synthesis to both Xilinx Virtex 4 and Spartan 3 FPGAs helps to further characterize the flexibility of the architecture.

1.0 Introduction

Over the last decade, Hidden Markov Model (HMM) based speech recognition has become increasingly popular and many of today's state-of-the-art software systems rely on the use of HMMs to calculate the probability that a particular audio sample matches a specific acoustic characteristic of a given word [2, 3]. Such systems have been observed to achieve accuracy rates upwards of 95% however; this accuracy comes at the expense of needing to evaluate hundreds of thousands of Gaussian probabilities and results in execution times of up to 10X the real-time requirement [4].

Figure 1 gives a conceptual overview of the Speech Recognition process using HMMs. Words are first broken down into their individual components, called phones, each represented by a large oval in Figure 1. Each phone is then represented by an HMM whose inputs are called senones, the unique sub-phonetic units that make up the language of interest. The scores associated with each HMM represent the probability that a given sequence of senones represents a particular phone.

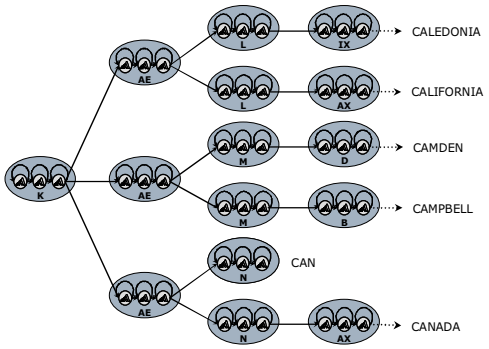

Figure 1: Conceptual Overview of ASR using HMMs

The senone database, or *acoustic model*, is based on multi-dimensional Gaussian probabilities based on features derived from the audio input. For each of the senones that need calculated every 10ms, over three hundred calculations are needed to complete the evaluation. This results in a staggering 60 million calculations per second to calculate just 2,000 senones.

2.0 The Speech Silicon Project

Today's automatic speech recognition (ASR) engines consist largely of four basic components: the Feature Extractor (FE), the Acoustic Modeler (AM), the Phoneme Evaluator (PE), and the Word Modeler (WM), each presenting its own unique problems. Figure 2 shows a conceptual diagram of a traditional ASR system.

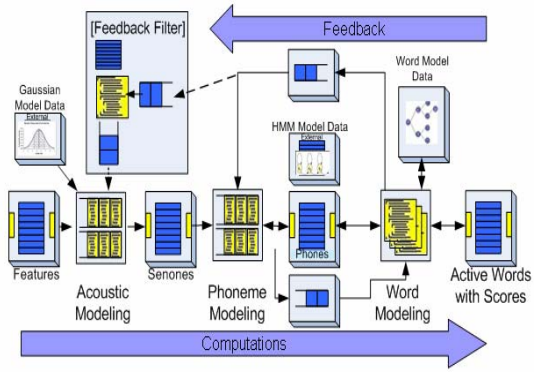

Figure 2: Conceptual Diagram of Architecture

1-4244-0054-6/06/$20.00

It is generally understood by the speech recognition community that FE can be executed in software by DSP processors without a significant performance penalty. The standard FE front-end is described in [9] and consists largely of frequency domain transformations and warping functions. The AM is responsible for evaluating the inputs received from the DSP unit with respect to a database of Gaussian probabilities and producing a set of scores, or senones. It is this phase of the speech recognition process that is the most computationally intensive taking up to 95% of the execution time [3, 5]. The PE associates groups of senones into to HMMs representing the phonetic units, phonemes, allowable in the systems dictionary. The WM uses a tree-based structure to string phonemes together into words based on the sequences defined in the system dictionary.

This paper focuses on the design and simulation of the AM pipeline. The major calculations necessary for AM will be described and the synthesis and post place-and-route results for each unit will be presented to help further its characterization.

3.0 - Acoustic Modeler

AM is the process of relating the Cepstral data received from the feature extractor to statistical models found in the systems database and can account for 70% to 95% of the computational effort in modern HMM-based ASR systems [3, 5]. The major equations necessary to derive a single senone score are shown in Equations 1-3 with a detailed description of given in [8]. Figure 3 illustrates the interaction between these equations noting that some of the input buses supply more than type of data value helping to minimize the number of pipeline stalls in the architecture.

$$P(\overline{X}) = \frac{1}{\sqrt{(2\pi)^D \left|\overline{V}^*\right|}} e^{-\sum_{d=1}^{D} \frac{(X_d - M_d)^2}{2*\sigma_d^2}} \qquad \text{[Eq. 1]}$$

$$S_i(\overline{X}) = \sum_{c=1}^{C} \left[W_{i,c} * P_{i,c}(\overline{X})\right] \qquad \text{[Eq. 2]}$$

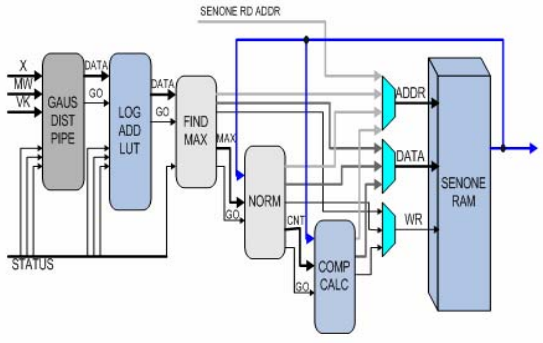

Figure 3: Block Diagram of AM Pipeline

Equation 1 shows the calculation for a single multi-dimensional Gaussian distribution, or component. From here we must combine multiple components with an associated weighting factor to create senones as summarized in Equation 2. At this point in our models it is necessary to define a log-base conversion factor, ψ, in order to stay in line with the SPHINX models used as our baseline [1]. The use of a conversion factor in these equations is useful in transforming Equation 1 into the log-domain for simplification of the calculations but the use of the specific value is unique to the SPHINX system. Equation 3 shows the results of this transformation and presents the final equation necessary to calculate a single senone.

$$\log_\psi[S_i(\overline{X})] = LOG\sum_{c=1}^{C}\left[\log_\psi(W_{i,c}) + \log_\psi(P_{i,c}(\overline{X}))\right] \qquad \text{[Eq. 3]}$$

The values *M, , V, K, & W* relate to specific speech corpus being used and represent the mean, standard deviation, co-variance matrix, scaling constant, and mixture weight respectively. These values are stored in ROMs that are otherwise unassociated with the system and can be replaced or reprogrammed if a new speech corpus is desired. The *X* vector contains the Cepstral coefficient input values provided by the FE block.

Our system is based on the 1000-word RM1 dictionary provided by the LDC [7] and requires over 2.5 Million Floating-Point operations to calculate every senone. By performing a thorough analysis of these calculations it was found that the computationally intensive floating-point calculations required could be replaced with fixed-point calculations with errors remaining on the order of 10^{-4}. The ability to use fixed-point data, allowed for the design of a pipelined acoustic modeling core running at over 125MHz post place-and-route on a Vertix-4 SX35-10. Additionally, through the use of intelligent hand-shaking between blocks and the use of an input control bus all local FSMs needed to control the pipeline were removed.

3.1 – Gaussian Distance Pipe

The Gaussian Distance Pipeline (GDP) is the heart of AM block and is responsible for calculating Equation 1 for each component in the database. Figure 4 shows a Data Flow Graph (DFG) for operations inside the GDP. The GDP must execute Eq. 1 over 15,000 times for each new frame of data and therefore must have the highest throughput of any component in the system. To accommodate this requirement while still trying to minimize the resources consumed by pipeline, the inputs to crucial arithmetic operations are multiplexed, allowing the inputs to the operation to be selected based on the bits of the Status Bus.

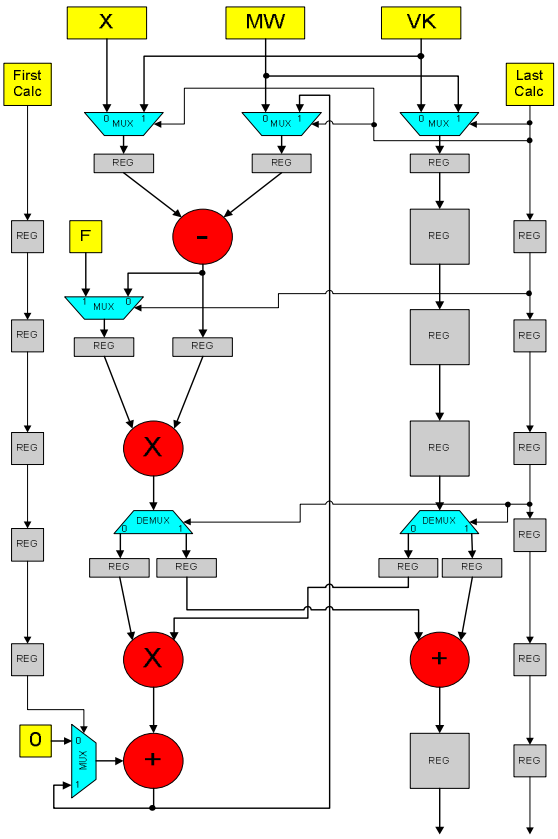

Figure 6: DFG for GDP

In order to help with low power applications, the GDP has a 'pipe freeze' feature included which is not shown in the DFG. If the last calc bit is seen at the end of the GDP before a new first calc bit has been seen the GDP will completely shut down and wait for the presence of a new first calc bit. Internal to the pipe each stage passes a valid bit to the successive stage that serves as a local stall, which will freeze the pipe until the values of the predecessor stage have become valid again.

3.2 – Log-Add LUT

After the GDP completes the scoring for one component, that component is sent to the Log-Add LUT (LL) for use in the senone calculation. The LL accumulates each of the eight partial senone scores and outputs them when the summation is complete. To minimize the complexity of this portion of the design a 20K entry LUT was used to replace the full calculation. In an effort to maximize the speed of the look-up this LUT was divided into small ROMs and the process was pipe-lined over 2 clock cycles wherein the address is de-muxed on the first cycle and the data is fetched and muxed onto the output bus during the second.

Similar to the GDP the LL also has a pipe-freeze function built in. This function performs exactly as the GDP freeze with respect to the first and last calc bits, and will also perform a local stall if the partial log-add has been updated before a new component value is available. The entire LL block takes a minimum of 10 clock cycles to process a single input and return the partial summation for use by the next input, resulting in an 80 cycle minimum per senone.

3.3 – Find Max / Normalizer

Once a senone has been calculated it must first pass through the Find Max block before being written to the senone RAM. This block compares the incoming data to the current best score and overwrites it when the incoming data is better. Since senones are only received every 80 cycles the pipe-freeze function for this block provides a notable power savings for the device during run-time.

When the last raw senone is put into the senone RAM a '*MAX done*' will be set high, signaling to the Normalizer block that it can begin. During the process of normalization the raw senones are read sequentially out of the senone RAM and subtracted from the best senone score found by the Find Max block. The Normalizer block takes 4-cycles per senone, is fully pipelined, and possesses a pipe-freeze capability.

3.4 – Composite Senone Calculation

In the RM1 speech corpus there are two distinctly different types of senones. The first are what can be considered 'normal' or 'base' senones and are calculated via the processes described in sections 3.0-3.3. The second type of senone is a sub-set of the normal senones called Composite Senones (CS). CS are used to represent more difficult or easily confusable sounds, as well as non-verbal anomalies such as silence or coughing. Each CS is pointer to a group of normal senones, and for a given frame the CS takes the value of the best scoring normal senone in its group. By finding each CS value and writing it to its own unique location in the senone RAM the later portion of the design only have to be concerned with one type of senone, greatly simplifying their design.

Like the other blocks of the AM calculation, the CS calculation has the built-in ability for locally stalling during execution and freezing completely when no new data is present at the input. In terms of cycles this means remaining inactive for 650,000 clock cycles while the normal senones are calculated and then running for only 2,200 cycle before going back to sleep.

4.0 – Synthesis and Place-and-Route

Table 1 summarizes the final synthesis and place-and-route numbers for each unit in the acoustic modeler and Figure 11 shows the final floor-plan for the Acoustic Modeling Pipeline. All components for the Acoustic

Modeler were synthesized using the Synplify® synthesis engine allowing for the use of simplified VHDL as well as the pipelining and retiming features built into the tool. These features provided the ability alter the depth of the pipeline in a very simple manner and quickly obtain results on the performance increase gained with each alteration.

Table 1: Synthesis and Place-and-Route Results for Virtex-4 SX35

Component	Synth (MHz)	PAR (MHz)	MISC.
Gaussian Dist. Pipe	**157**	**145**	6 DSP48s, 411 Slices
Log-Add LUT	**164**	**150**	13 BRAMs, 307 Slices
Find Max	**181**	**160**	90 Slices
Normalizer	**197**	**172**	144 Slices
Composite Senone Calc.	**197**	**140**	2 BRAMs, 147 Slices
AM Block (TOTAL)	**164**	**125**	**6 DSP48s, 30 BRAMs, 1328 Slices**

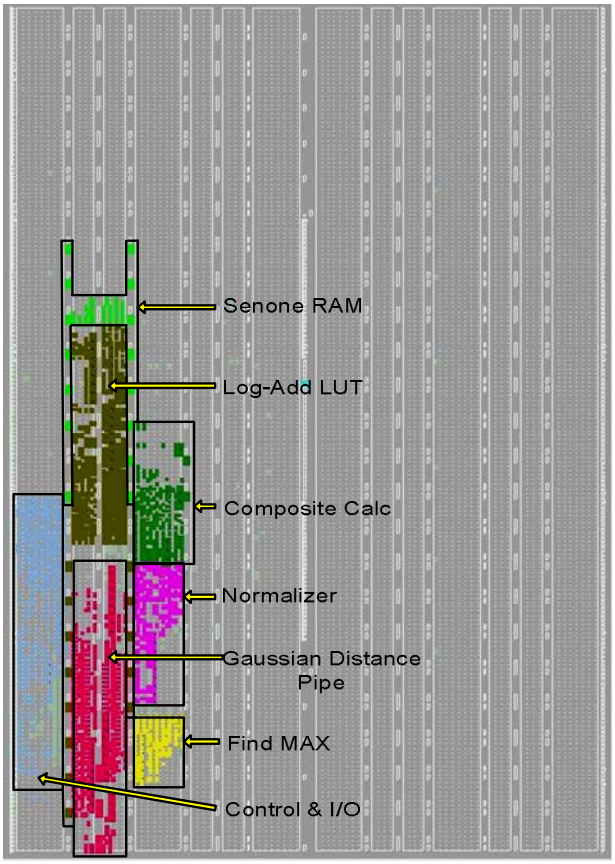

Figure 12: Floor-plan for AM Pipeline on a Xilinx Virtex 4 SX 35-10 FPGA

5.0 – Conclusions

In this work we have introduced the Speech Silicon Acoustic Modeler, an FPGA-based acoustic modeling pipeline that is capable of executing in real-time. Further, this architecture has the ability to freeze the pipelines to minimum power consumption without effecting processing bandwidth. Our system also highlights an architecture built to be driven completely by the data. This enables the system to be reprogrammed for multiple applications and dictionaries simply by altering the input to the system. This research has proven the effectiveness of the proposed design methodology and helped further the development of portable low-power speech recognition system for use in embedded environments.

References

[1] P. Placeway, *et al,* "The 1996 HUB-4 Sphinx-3 System", *Proc. DARPA Speech Recognition Workshop*, Feb. 1997.

[2] K.K. Agaram, S.W. Keckler, D. Burger, "Characterizing the SPHINX Speech Recognition System", University of Texas at Austin, Department of Computer Sciences, *Technical Report TR2001-18*, January 2001.

[3] C. Lai, S.-L. Lu, & Q. Zhao, "Performance Analysis of Speech Recognition Software", *Proc. Fifth Workshop on Computer Architecture Evaluation using Commercial Workloads*, February, 2002.

[4] M. Ravishankar, *et al*, "The 1999 CMU 10X Real Time Broadcast News Transcription System", *Proc. DARPA Workshop on Automatic Transcription of Broadcast News*, Washington DC, May 2000

[5] J. Nouza, "Feature Selection Methods for Hidden Markov Model-Based Speech Recognition," *Proc. International. Conference on Pattern Recognition*, 1996, vol. 2, pp. 186.190.

[6] X. Li & J. Blimes, "Feature Pruning in Likelihood Evaluation of HMM-Based Speech Recognition", Unversity of Washington, 2003.

[7] Linguistic Data Consortium. University of Pennsylvania. 27 Oct. 2004 www.ldc.upenn.edu

[8] "Speech Silicon: A data-driven SoC for Performing Hidden Markov Model based Speech Recognition," R. Hoare, J. Schuster, K. Gupta. *Proc. HPEC 2005,* Lincoln Labs, MIT, Boston, MA.

[9] M.J. Hunt, "Spectral Signal Processing for ASR", Dragon Systems UK Research and Development Ltd., 2000.