6.111 TB Schardl
Project Proposal Nick Harrington
Voice-Driven Joystick October 31, 2008

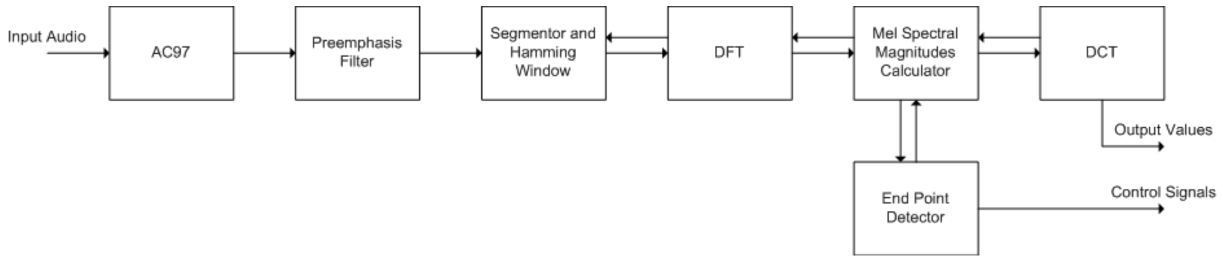

Figure 1: Feature Vector Sequence Generator block diagram.

1 Introduction

We propose designing a simple isolated word speech recognition system in Verilog. Our design is naturally divided into two modules. The first module takes audio input and processes it through a sequence of steps to generate a sequence of feature vectors. These feature vectors are then passed to the second module along with some control signals which dictate whether the incoming data should be stored as a sample or matched against an existing sample. We plan to use this system to drive signals on a serial port (such as USB) such that it may be attached to a personal computer and act as a voice-controlled joystick.

We have implemented a simulation of our speech recognition system in Matlab and verified that it can handle simple test cases. We have also identified a number of simple points of improvement with our design that should allow our algorithm to work effectively with more difficult inputs. Our next challenge is to finalize the specifications of modules and implement a simple form of our algorithm in Verilog.

2 Feature Vector Sequence Generation

This module takes incoming audio signals and converts them to feature vectors that are used to match words. It is also responsible for identifying the beginning and end of incoming words. For the individual module descriptions there is an unstated input of a reset signal and a clock signal.

2.1 Preemphasis Filter

Input Incoming audio signal from the microphone input on the AC97 Audio Codec chip.

Output Filtered audio signal.

The preemphasis filter is used to spectrally flatten the signal. This is because the energy of various frequency channels emitted by the human vocal cords decrease with frequency. That is to say in human speech lower frequencies have a higher amplitude giving them greater weight. This filter evens the energy distributed to the various frequency channels. The FIR filter used has the form $H(z) = 1 - \alpha z^{-1}$. α is dependent upon the sampling rate chosen. The code for this has already been implemented in Lab 4. The filter has minimal memory requirements.

2.2 Segmenter

Input Filtered audio signal.

Output Segmented audio signals; Signal for the end of the block; RFS (*from the DFT*)

The segmenter takes incoming audio and divides it into equal length discrete samples. These samples are on the order of 10-40 ms. The actual length will depend on the final sampling rate chosen and constraints to the input of the DFT module. Also, in this module the segmented audio signals will be scaled by a Hamming window function in order to reduce the problem of DFT leakage because we will be sending it packets of audio data. The windowed audio signal is loaded into a FIFO in preparation for being sent to the DFT. Once the block is loaded into the FIFO and the DFT is asserting RFS high, the signal for the end is fed into the START signal of the DFT and the FIFO's unloading of the windowed audio signal to the DFT. The memory requirement for the FIFO is 8xNS bits where NS is the number of samples.

2.3 DFT Module

Input *The inputs listed here are the non fixed inputs to the Xilinx DFT architecture* START; XN_RE (real input); UNLOAD

Output XK_RE; XK_IM; DONE; RFS

The discrete fourier transform module has already been implemented by the Xilinx tools. Conveniently enough the latest DFT implemented by Xilinx has 4 different available architectures for use where the speed of the implementation is directly proportional to the resource requirement. Documentation can be found at: http://www.xilinx.com/support/documentation/ip$_d$*ocumenta*

2.4 Mel Spectral Magnitude Calculator

Input XK_RE; XK_IM; DONE; RFS; EPRFS

Output UNLOAD; MEL_COEF

The Mel Spectral Magnitude calculator takes the incoming XK_RE and XK_IM signals and first converts these two values into the magnitude of half of the incoming FFT signal. When asked to identify intervals with the same difference in frequency, a human will have a non linear response. For two high pitched frequencies to be perceived as having the same frequency difference as two low pitch frequencies there would need to be greater difference in the actual frequencies. The Mel scale is a representation of the distance between these equally perceived signals. Mapping from actual frequency to the Mel scale is useful in speech recognition because we are trying to imitate a human's perception.

This module basically skews the FFT. It also reduces the number of frequency data points in the FFT from something on the order of 256 to 17. Each of the magnitudes associated with our small, skewed FFT are calculated with piecewise multiplication of the magnitudes produced by the FFT by triangular window functions centered at frequencies corresponding to our Mel scale frequency data points. These Mel scale frequency data points are separated by a frequency that is equal on the Mel scale. This is a fairly computationally intensive module that requires piecewise multiplication of all of the outputs of the FFT with numbers stored in system memory. The memory requirements are scaled by how large the FFT is. The size of the FFT is to be decided at a future design meeting. Once the Mel scale spectral magnitudes are calculated the logarithm of each value is taken and then squared. These values are passed to the DCT and the EPD once the DCT has asserted RFS high and the EPD has asserted EPRFS high as well.

2.5 End Point Detection (EPD)

Input MEL_SM

Output detecting-word; EPRFS

This module is a simple 2 state FSM. The two states are DETECTING and its negation NOT_DETECTING. For purposes of determining state transitions this module also determines the total energy of the incoming spectral magnitudes. Transitions from NOT_DETECTING to DETECTING occur when the total energy crosses a threshold. Transitions from DETECTING to NOT_DETECTING occur when the total energy drops below that threshold energy and stays below it for an arbitrary number of incoming sets of MEL_SMS. detecting-word is passed to the Word Recognition Control module.

2.6 Discrete Cosine Transformation Module (DCT)

Input *The inputs listed here are the non fixed inputs to the Xilinx DFT architecture* START; MEL_SM (real input); UNLOAD

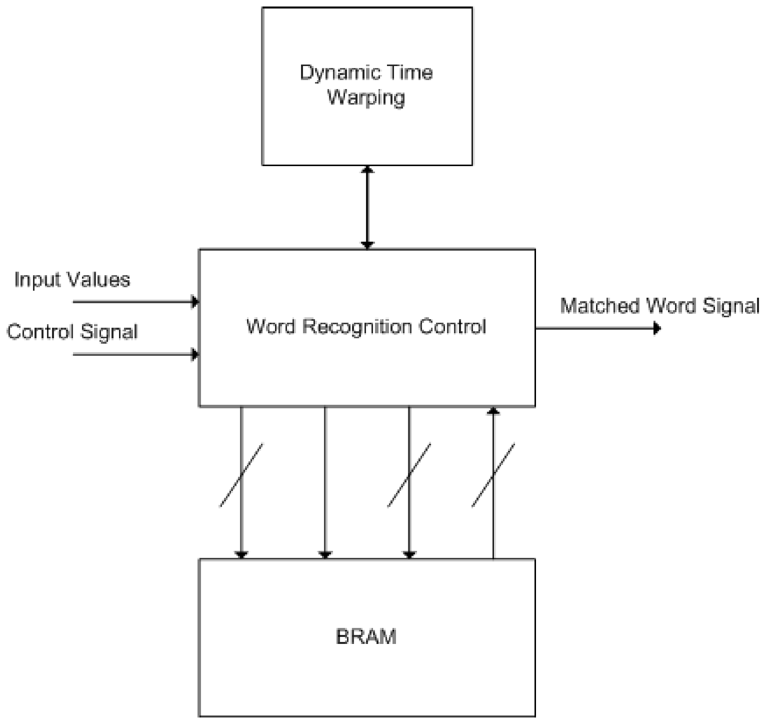

Figure 2: Word Recognition Module block diagram.

Output XK_RE; DONE; RFS

This is an inverse FFT where we only look at the real output of the data. This can be implemented using the Xilinx FFT module. Implementation of that is discussed above. The output is the Mel-frequency cepstral coefficients (MFCCs). The first coefficient is not useful for speech recognition so it is discarded and the 16 MFCCs (XK_RE) are passed to the Word Recognition control module.

3 Word Recognition

This module in effect implements the main behavior of the speech recognition system. This module takes the input values and control signals received from the Feature Vector Sequence Generator along with some additional control signals which will dictate the behavioral mode of the system. It will implement the distinct “matching” and “training” behaviors of the speech recognizer as well as the final word recognition signal in the event of a matched word.

3.1 Word Recognition Control

Input A sequence of feature vectors corresponding to a sample, and a set of control signals.

Output Signal for a matched word, if a word is matched.

Additional Interfaces Connections to BRAM; buses for sending two feature vector sequences to DTW block; buses for receiving data from DTW block

The Word Recognition Control block will be the main FSM driving the Word Recognition module. Depending on the received control signal, this block will either save the incoming sequence of feature vectors to memory as a sample word, or attempt to match these input values with an existing word sample stored in memory. If the input sequence was simply a new sample to store, this module simply stores it.

In the case that a match is requested, this module will buffer the input sequence and pass that buffered input sequence with each possible stored sample to the Dynamic Time Warping module to perform the comparison. Once this algorithm returns for the given input data this control module will compare the returned values with stored threshold values for these matches to determine whether a match was found and if so which match was the best. This block will then return the appropriate match signal to indicate a match in the event that one was found.

The functionality of this block will require a number of control signals to be passed between it an several other blocks, including the Dynamic Time Warping block and the Feature Vector Sequence Generator module. In particular this module will receive new input values serially and must be able to process them appropriately. To accommodate this a communication protocol will need to be established between this block and attached blocks, especially particular the final block of the Feature Vector Sequence Generator. The exact nature of this communication protocol and the control signals this FSM will recognize will be determined in a future design refinement.

3.2 Word Data Storage

Input Memory address, write-enable signal, data

Output Data stored at the given address

Training data samples will be stored in a BRAM attached to the Word Recognition Control block. Each sample will be stored with at least one associated signal such that 1) this sequences can be easily and quickly identified, and 2) if that sample is ever matched by some input sequence in the future the Word Recognition Control FSM will know which signal it should output. In the "training" mode this BRAM will write sequences of input vectors into memory as it receives them, while in the "matching" mode this BRAM will simply send requested sequences of feature vectors associated with the request.

This BRAM imposes several constraints on our speech recognition system. First of all, sequences of feature vectors must have some definite maximum length so they may be stored in a finite RAM. Second, the size of the feature vectors must be fixed in order to allow accurate sizing of this block during compilation. Third, it imposes one particular communication protocol with the Word Recognition Control block, restricting performance to what it can handle. A more detailed specification of the sizing and use of this BRAM will be determined in a future design refinement, although our current estimates put the size of a feature vector at 16 bits.

3.3 Dynamic Time Warping

Input Two sequences of feature vectors

Output Distance information for the two given sequences

When the Word Recognition module attempts to match an input sequence with some sample, the input sequence will be paired with each stored input sample and a Dynamic Time Warping algorithm will be run on each such pair. This algorithm uses dynamic programming to find the minimum cost path through a matrix of distance measurements, where each entry M_{ij} in the matrix stores the distance between the i^{th} vector in one input and the j^{th} vector in the other. This minimum cost path occurs approximately when the contour of the two sequences match up to some local rescaling in time, and thus this algorithm is useful in simple speech recognition systems to match distinct utterances of the same word.

This algorithm therefore requires analyzing $n * m$ entries in a table to find a minimum path. Each analysis can be performed using addition and comparisons, while the table generation itself will require addition and some multiplication. The exact details of this table's implementation will be worked out in a future design refinement.

The effectiveness of this algorithm is closely tied to the effectiveness of the speech recognition system as a whole. Significant research has been conducted into increasing the effectiveness of this algorithm, in particular through manipulations of the distance function used. We may draw upon this research in the future in order to improve the effectiveness of the speech recognition system as necessary.

4 System Output

We plan to add an additional module that will convert recognition signals from our speech recognizer into a serial format familiar to personal computers. This should allow our speech recognizer to act as a HID that can be connected to any computer, similar to a mouse or keyboard, and thus allow us to effectively implement a voice-controlled joystick of sorts.

Furthermore we plan for debugging purposes to have our speech recognition system transmit additional data to the VGA port. We should be able to conveniently display the results of the DFT, the Mel Spectral Magnitudes, state information from the Word Recognition Control block, and various control signals, such as when we are receiving a word and what how that word was matched. This data should give us a reasonable ability to watch our speech recognition system in action and confirm that it is working as expected.

5 Conclusion

We have the beginnings of a design for an isolated word speech recognition system that we plan to implement on an FPGA. We have presented a rough breakdown of this system into modules and have initial sketches of block diagrams for these modules. Additional work is needed to further specify this design and ensure that it is realizable on an FPGA. We have also implemented a prototype of our overall speech recognition algorithm in Matlab and demonstrated its operation on simple test cases. Finally, with this Matlab simulation, we have identified several simple points of improvement to our initial naive algorithm, so the prospect of implementing a successful system in hardware seems tenable.