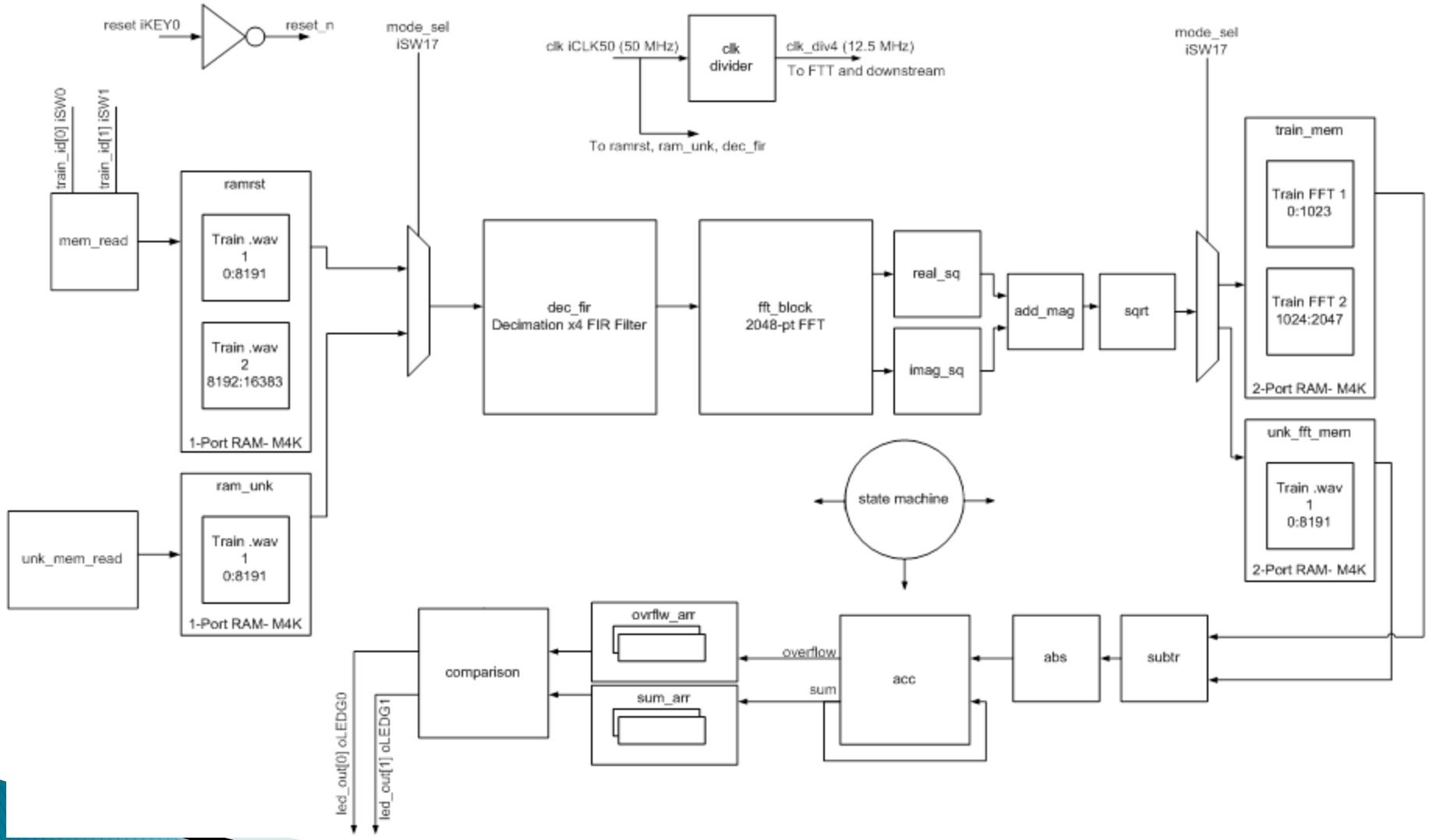
Voice Command Recognition Algorithm on FPGA

Matt De Zaio
ECE 641 Term Project
5/3/12

Overview

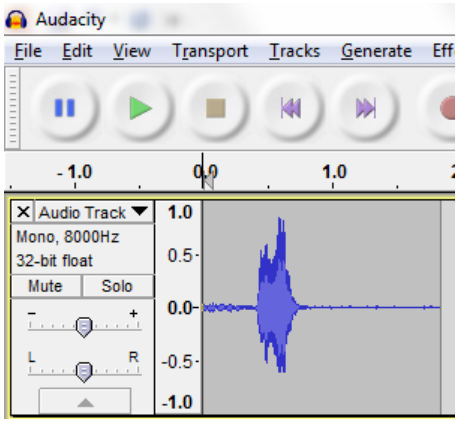
- During training, voice commands are processed to generate training vectors
- In operation, unknown voice commands are read in, processed and a decision is made
- Training vectors are represented by the FFT of the data
- Nearest neighbor algorithm is employed to determine desired command

System Block Diagram



Audio Input

- Concentrated on DSP portion
  - Audio CODEC not implemented
- Audio samples are obtained with Audacity software and exported as .wav files
  - 8 kHz sample rate
  - 16 bit signed samples
- Used Matlab to limit to 8192 samples and format for .hex initialization file
  - MIF caused problem for ModelSim



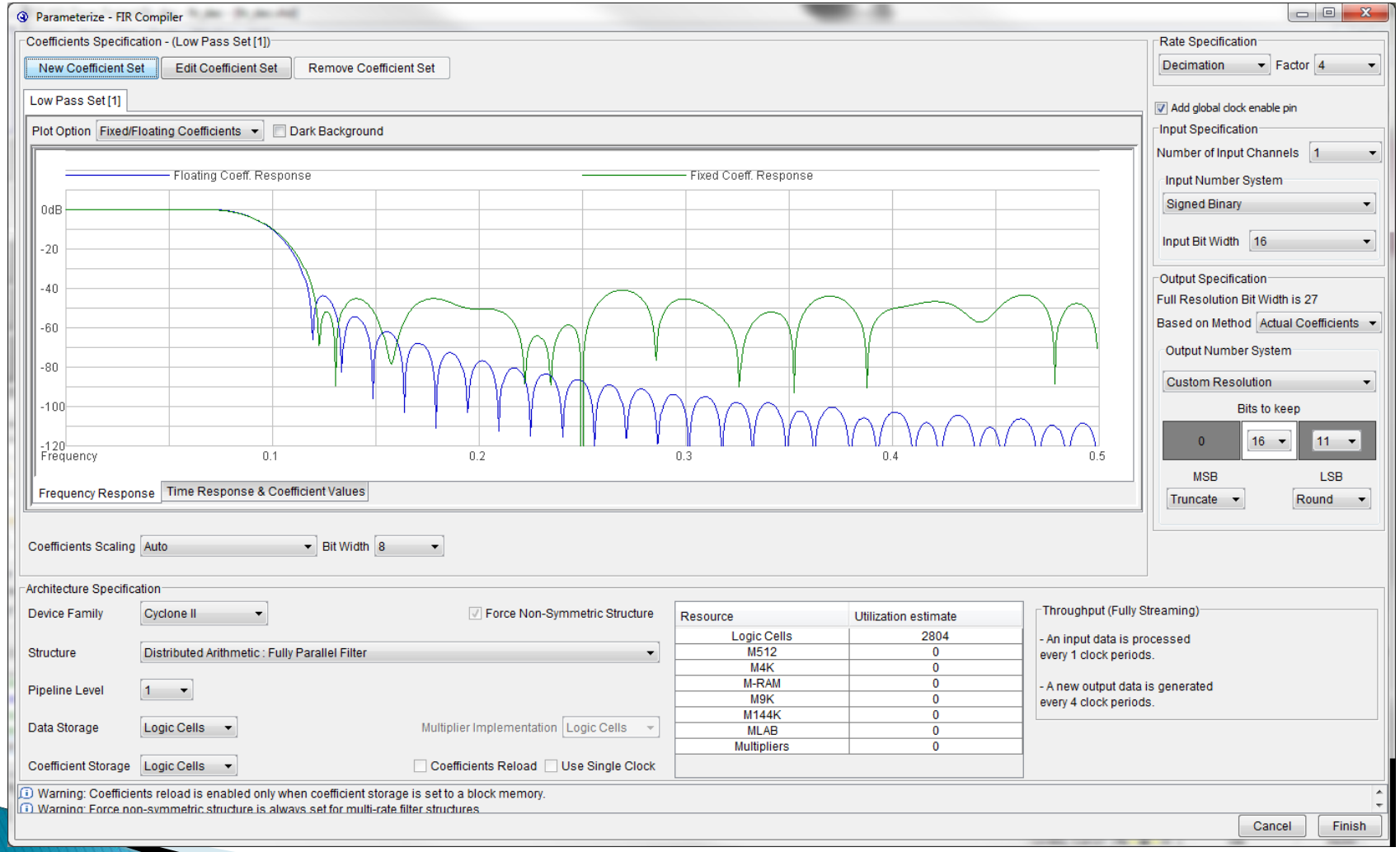
Memory Interfaces

- Concentrated on DSP portion
  - FLASH memory not implemented
- Utilized 1-port RAM with internal M4K for storing input training and unknown data
- Utilized 2-port RAM with internal M4K for FFT magnitude of training and unknown data
- Due to limited memory resources in the device, the scope was reduced to two training vectors

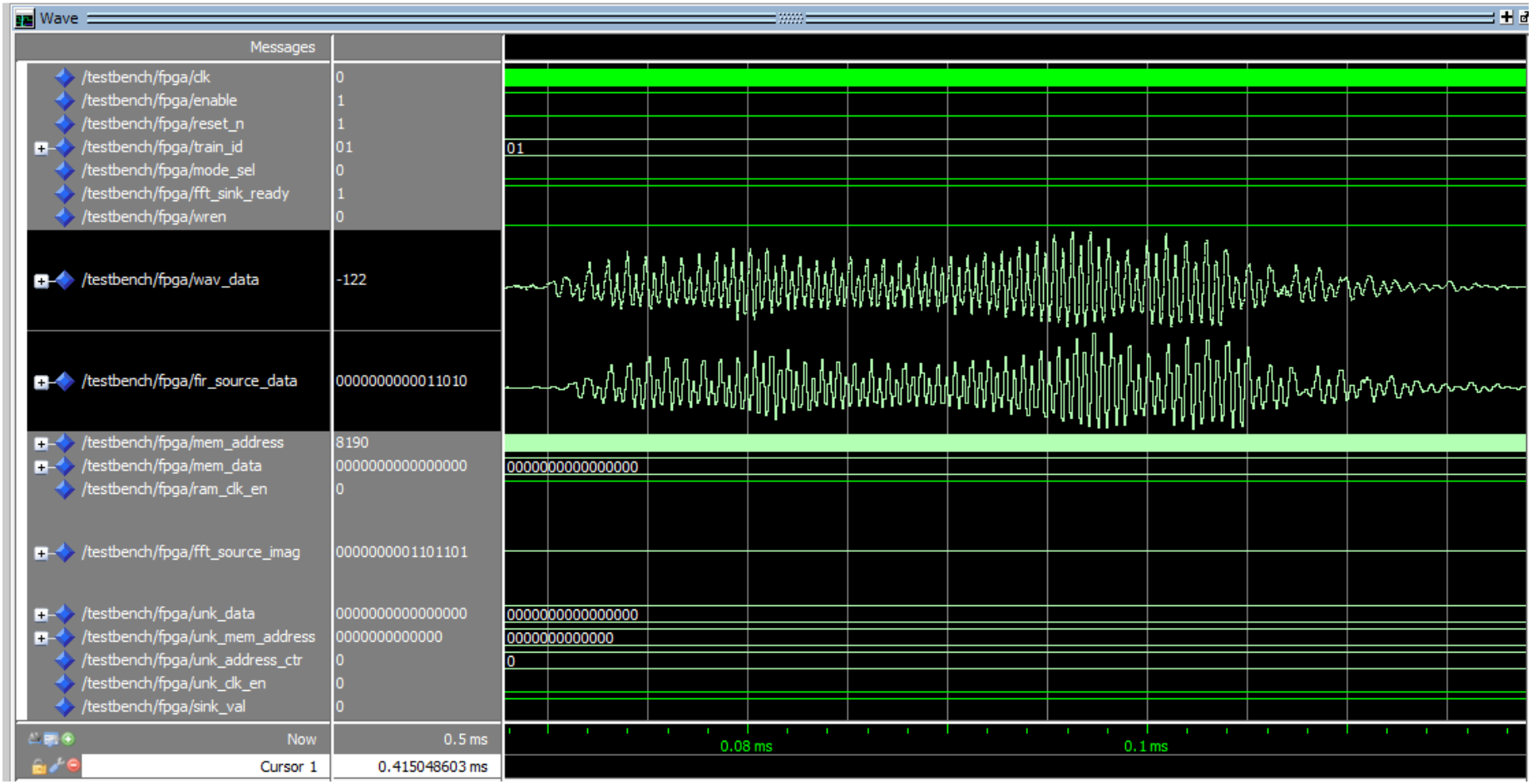
Decimation Filter

- Downsampling used to reduce number of data points for processing
  - Original sample rate: 8 kHz
  - Desired sample rate 2 kHz
- Filter is required to prevent aliasing
  - $f_{new} > 2B$ ($B_{aud} \approx 500$ Hz, design LPF for $B \approx 1$kHz)
- FIR filter is used, built with MegaWizard
  - Length 64 coefficients
  - Cutoff frequency of 750 Hz ($f_{pass}=550$ Hz, $f_{stop}=950$Hz)
- Outputs one sample every 4 clock cycles
- Output truncated to 16 bits (full res. 27 bits)
  - Trade-off between precision and bit-width carried forward in processing chain

FIR Compiler MegaWizard

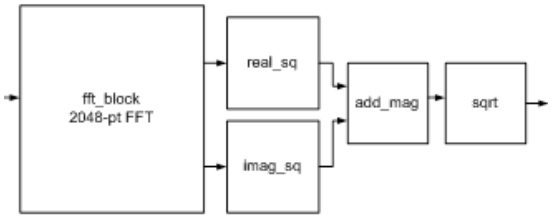


FIR – ModelSim Results

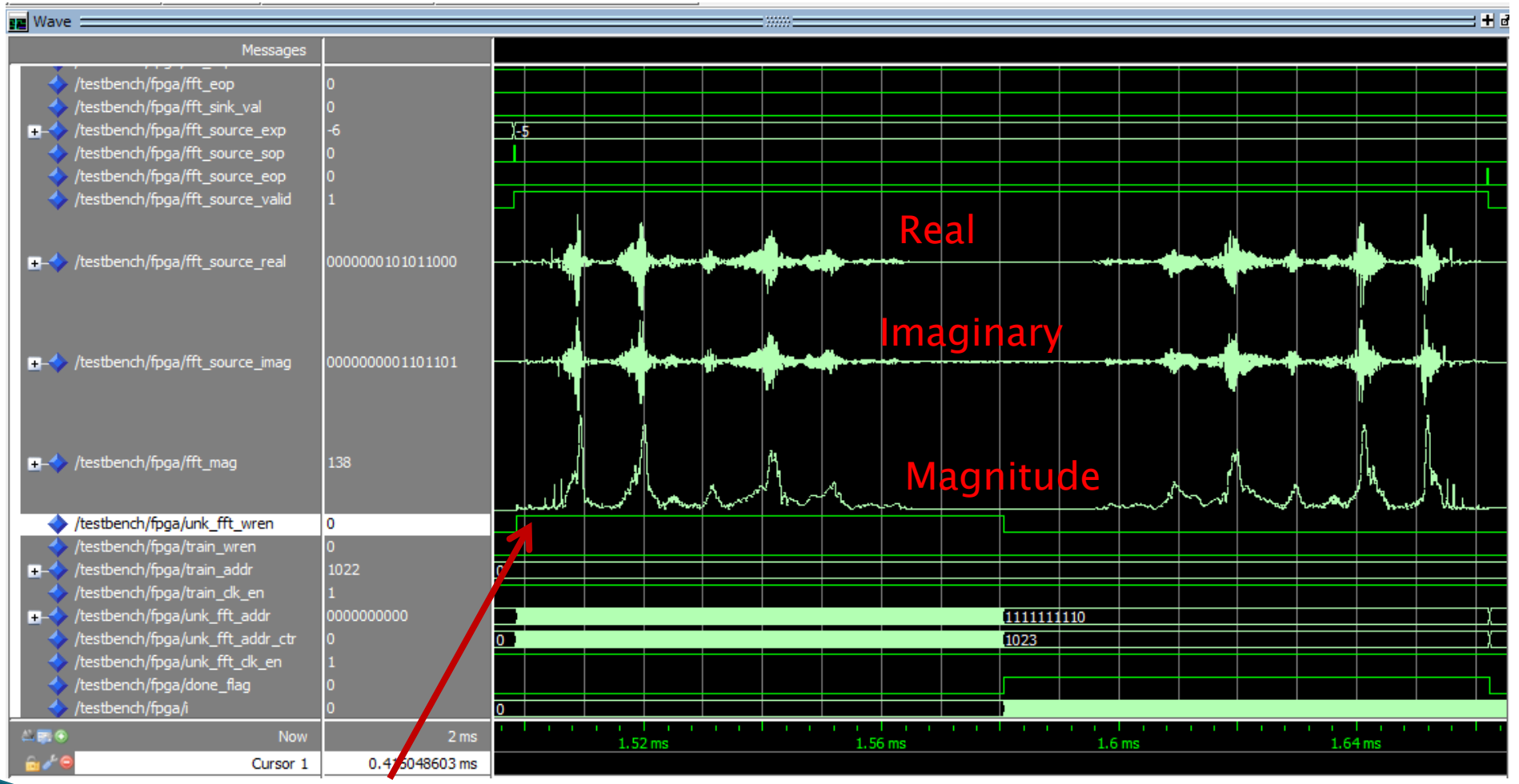


FFT

- Generated 2048 point FFT using MegaWizard
  - Processes created to generate Avalon streaming interface control signals
- FIR output is connected to FFT input
  - FFT clocked at ¼ speed of filter
- Outputs complex data, real and imaginary source outputs
  - Magnitude of the FFT is computed
  - First and second set of L/2 bins are redundant so only ½ of the output bins are stored



FFT – ModelSim Results



wren to write L/2 bins to memory

Algorithm

- Training vectors stored in memory
  - Represented by FFT of recorded voice signal
- Unknown input FFT computed and compared to training vectors
- Original comparison method: Euclidean distance

$$d_{eu} = \sqrt{\sum_{i=0}^{L-1} (x_i - y_i)^2}$$

- Nearest neighbor (smallest distance) determines desired command

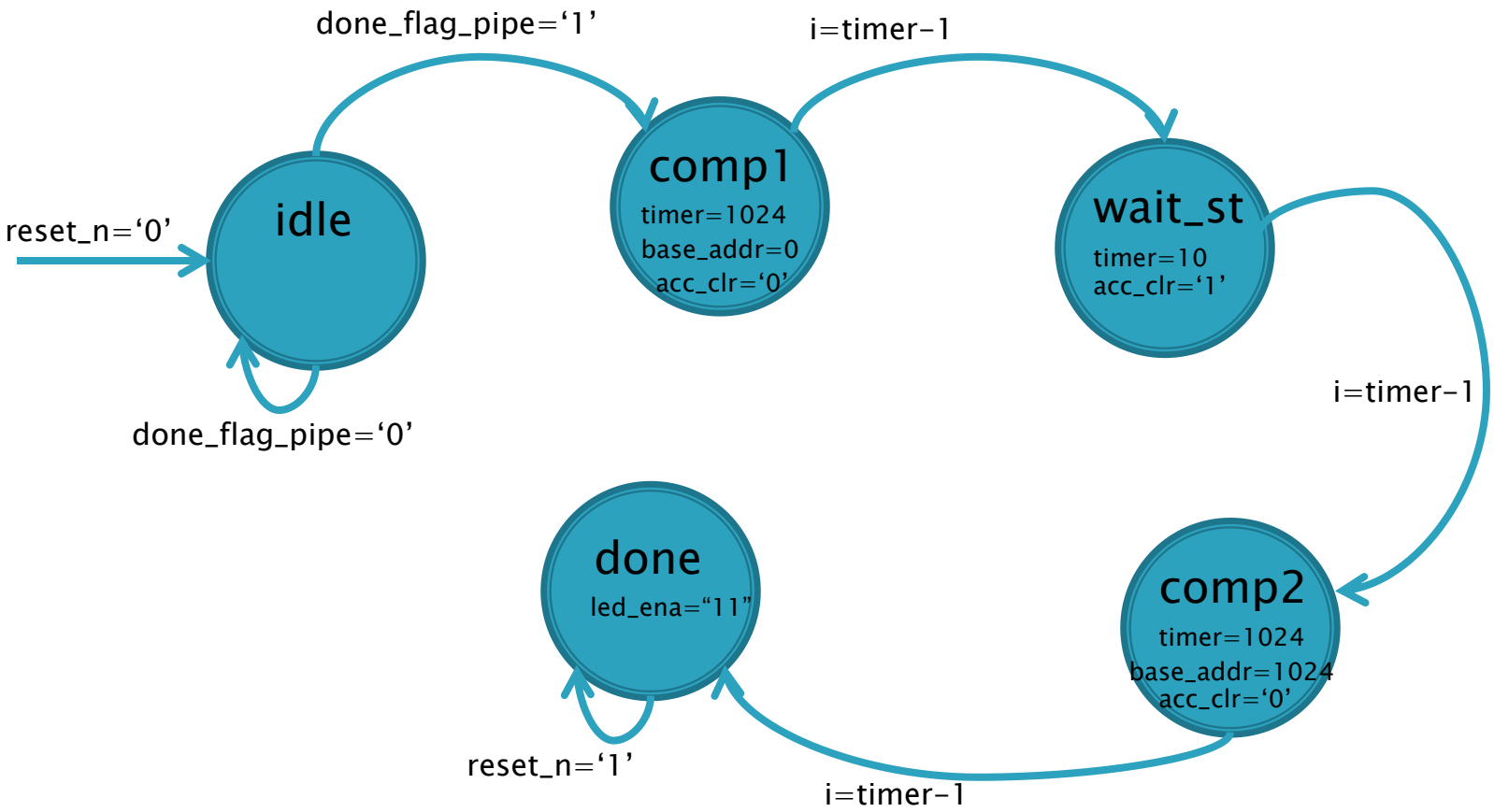
Algorithm

- Euclidean distance involves squaring and summing a large amount of data points
  - Bit growth is a problem
  - Needs multipliers (square) and sqrt
- Implemented comparison method: sum of absolute value of difference between elements
  - No multipliers or sqrt

$$d_{imp} = \sum_{i=0}^{L-1} |x_i - y_i|$$

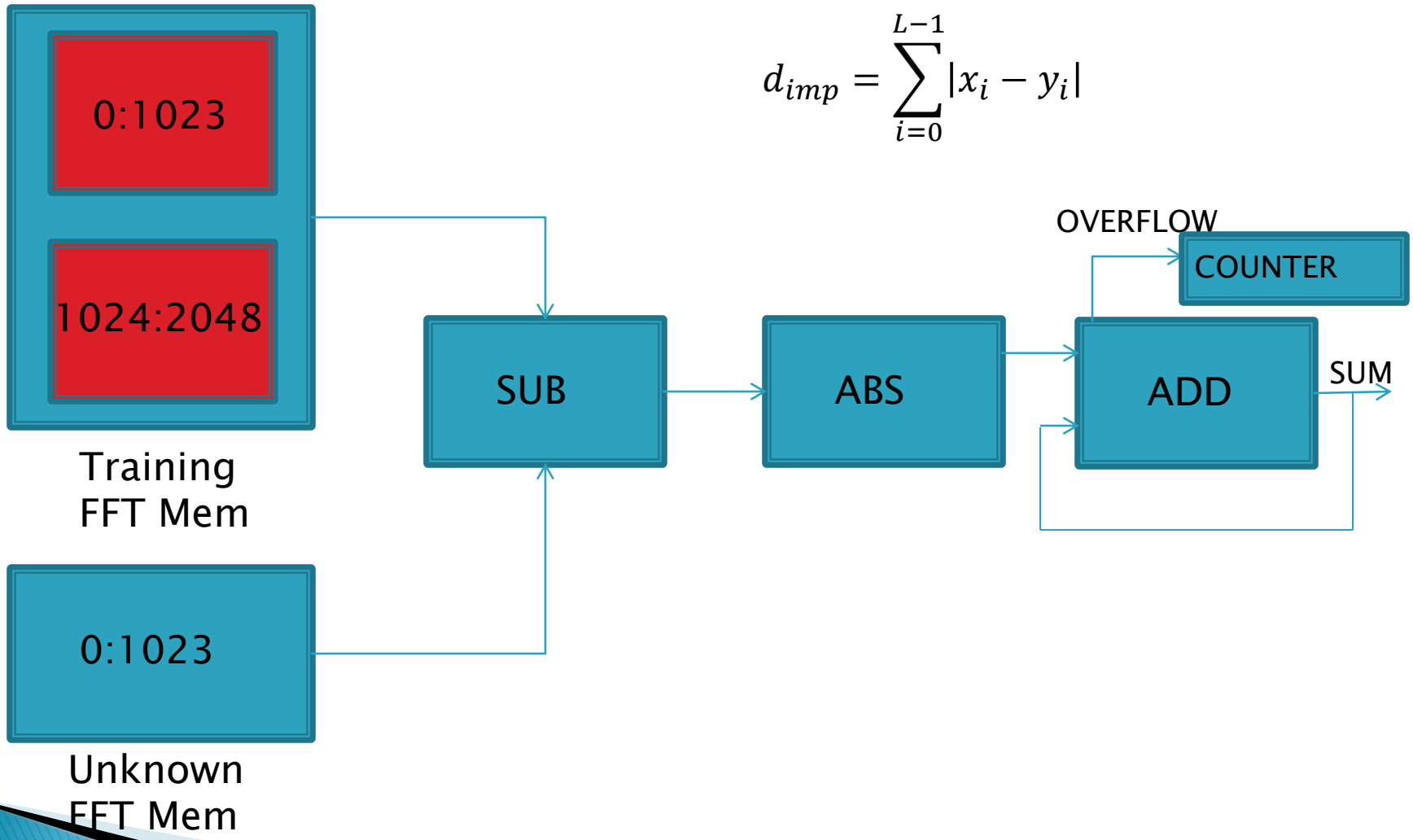
- Nearest neighbor (smallest distance) determines desired command

State Machine for Comparison



Algorithm

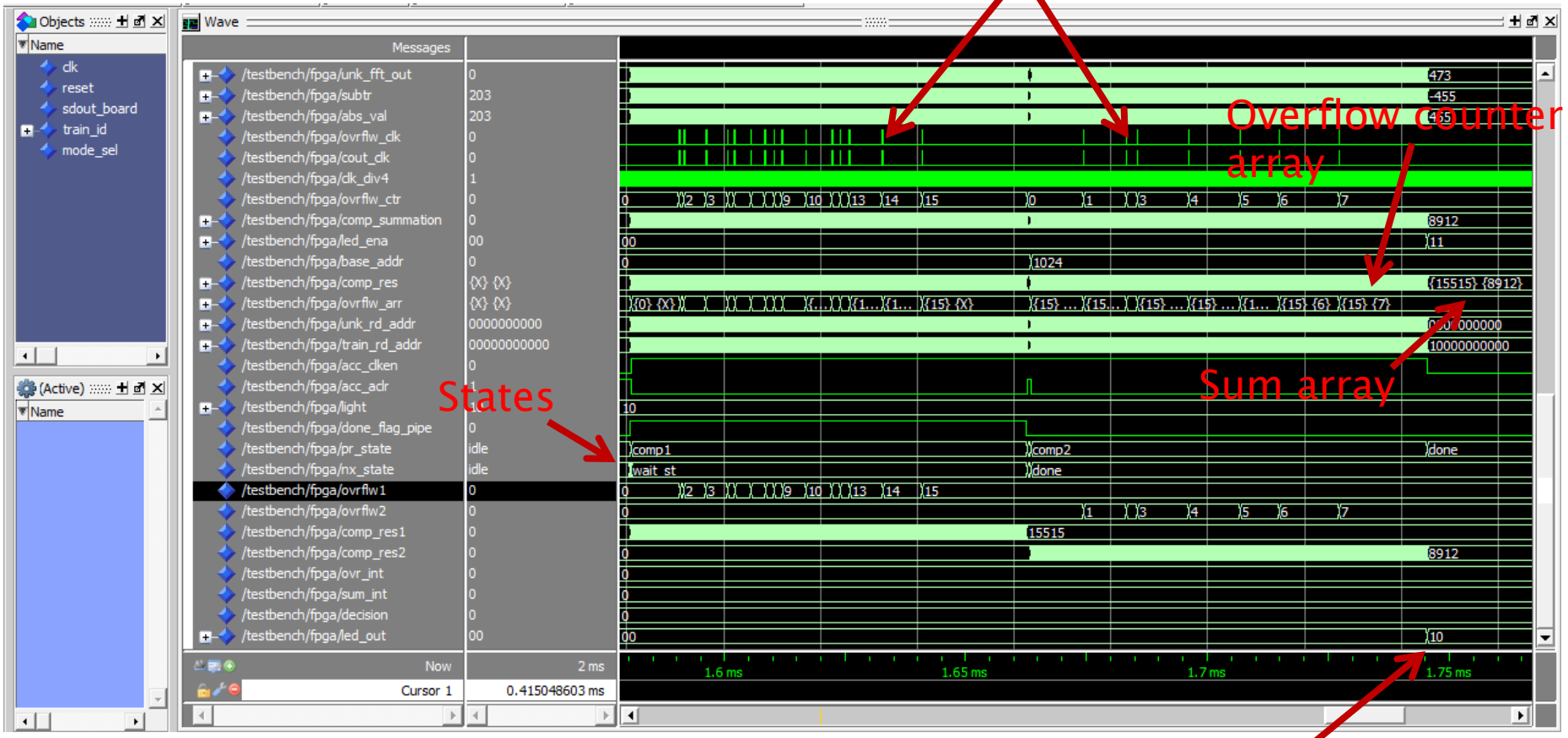
$$d_{imp} = \sum_{i=0}^{L-1} |x_i - y_i|$$



Algorithm

- Decision is made with Nearest Neighbor algorithm
- During comparison, there are two outcomes generated
  - Sum
  - Overflow count – total sum exceeds vector size
- The overflow count is the primary factor
  - If overflow counts for two training vectors are equal, sum is used to decide between the two
- Component implemented to make decision
  - Allows for easy expansion to more training vectors

Algorithm – ModelSim Results



Conclusion

- Successfully simulated design using ModelSim
- Implemented:
  - Decimation FIR filter
  - Streaming 2k FFT
  - FFT magnitude computation
  - Write/read RAM memory interfaces
  - State machine for comparison
  - Nearest neighbor decision algorithm

Areas of Improvement

- Timing errors with Altera-generated FIR core prevented testing on DE2-70 board
- Did not implement audio interface/FLASH memory – concentrated on DSP
  - Audio CODEC would allow real-time operation
  - FLASH memory would allow for more training vectors/not require retraining each time
- Accuracy could potentially be improved
  - Averaging training vectors
  - Implement Euclidean distance
  - Normalize amplitude