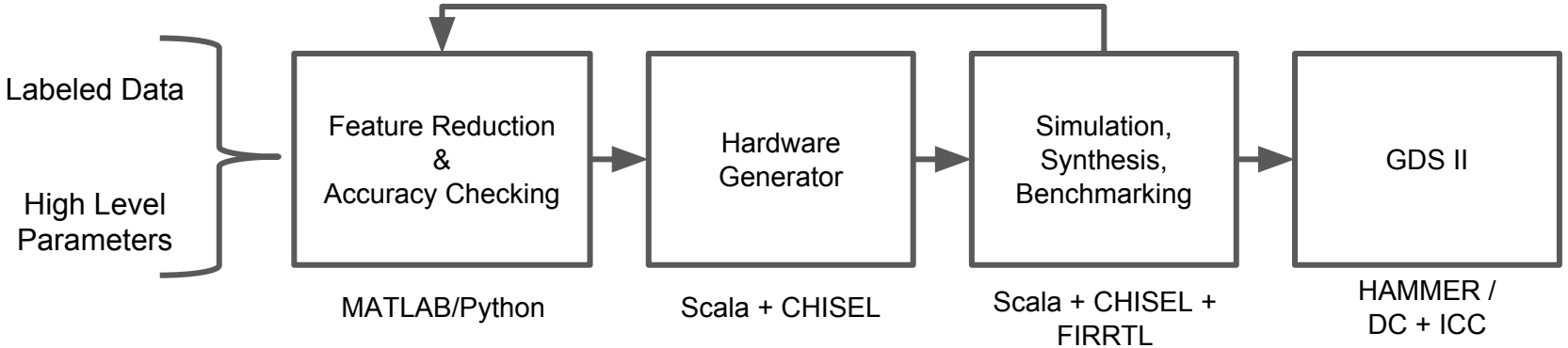
ExG DSP/Classifier Generator in Chisel

Adelson Chua, Justin Doong, Ryan Kaveh, Cem Yalcin, and Rachel Zoll

EE 290C Fall 2018

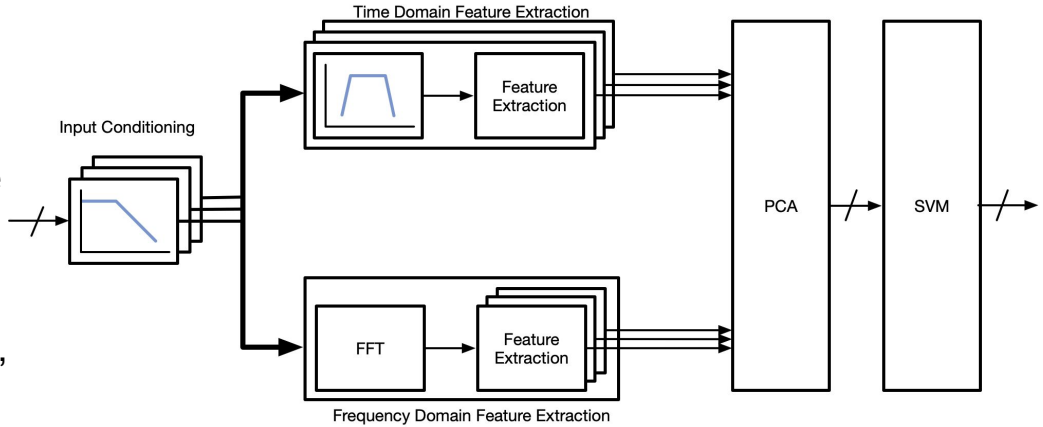
Wellness Monitor - Motivation

- Software tools/libraries exist for everything:
  - EEGLAB in Matlab
  - Biomedical Genomics Library, WFDB in Python
- An expressive, type generic hardware generator can be easily integrated into an automated design exploration
- Block architectures (*e.g.*, classifiers) can be switched out to perform holistic benchmarks



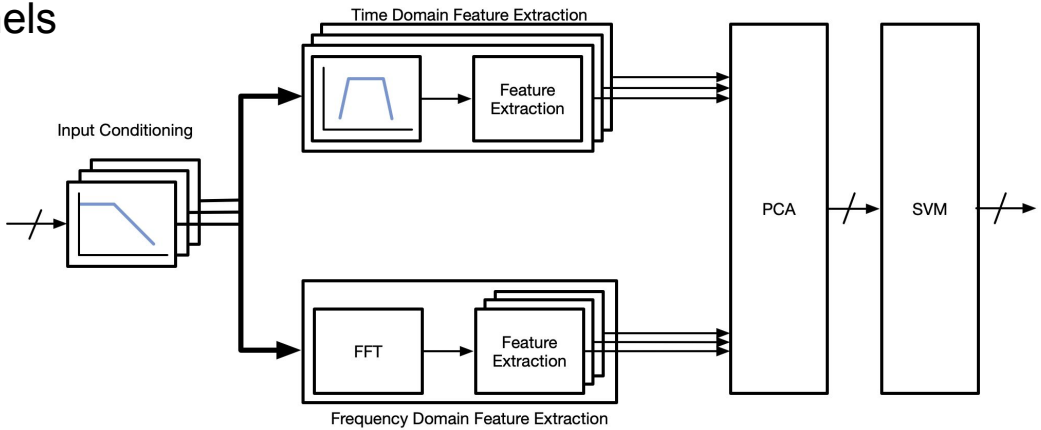
Wellness Monitor - Overview

- **Ingest** different data streams (EEG, ECG, EMG, etc.)
- **Extract** pertinent features
- **Classify**
  - Neurological states (*e.g.*, seizure detection, sleep)
    - Bandpower, line length, etc...
  - Cardiac states (*e.g.,* arrhythmias, bradycardia)
    - DWT, peak variability, etc...
  - General health states (*e.g.,* auscultatory sensing of blood clots)
    - Peak variability, threshold detection, etc...



Wellness Monitor - Overview

- There can be multiple input channels
- Two types of feature extraction:
  - Time Domain
  - Frequency Domain
- Every *feature* has a *datapath* associated with it
- Each *datapath* consists of blocks with similar I/O structures



Module Generators Overview

- The following generators were implemented as building blocks:
  - FIR / IIR filter
  - Bandpower Extractor
  - Line Length Extractor
  - Principal Component Analysis (PCA)
    - Offline eigendecomposition, online dimensionality reduction
  - Support Vector Machine (SVM)
    - Offline training, online classification
  - Some other supplementary modules (memories, stream-to-parallel converters…)
- FFT generator was imported from ucb-art repo

Feature Extractors: Bandpower & Line Length

- Bandpower: finds average power in a frequency range
  - Calculates the following sum, frequency components acquired from FFT block
  - Configurable frequency bands and number of bins

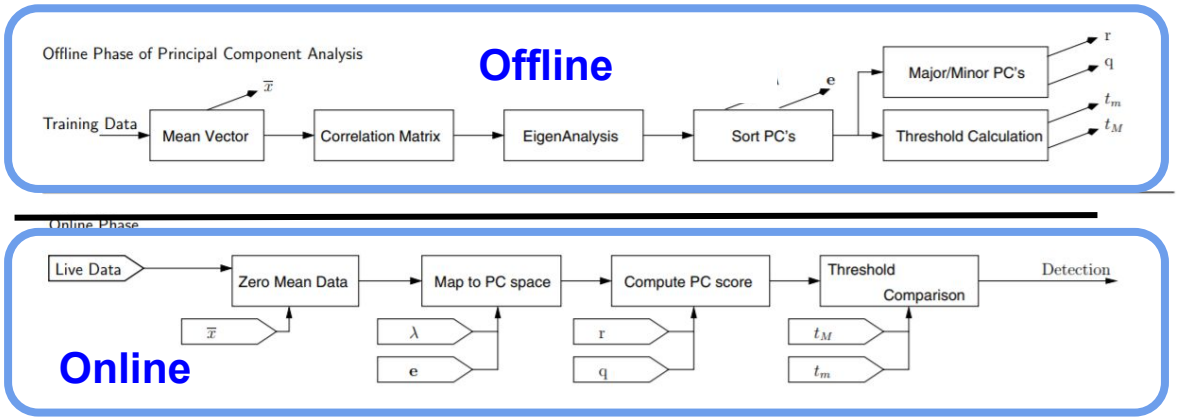
$$\sum \left(\frac{|X(f)|}{N} \right)^2$$

- Line Length: integral of the derivative within a time window
  - Calculates the following sum
  - Configurable window size

$$\sum_{i=2}^{N} |x(i) - x(i-1)|$$

Principal Component Analysis (PCA)

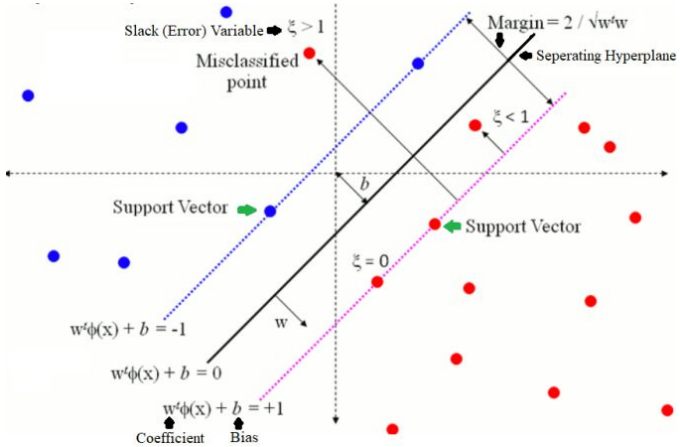
- Reduce dimensionality of feature space to minimize classifier complexity
  - New set of dimensions will be a linear combination of the original features
  - Transformation matrix is acquired from offline training
  - Dot product for dimensionality reduction done in hardware
  - Configurable number of features and output dimensions

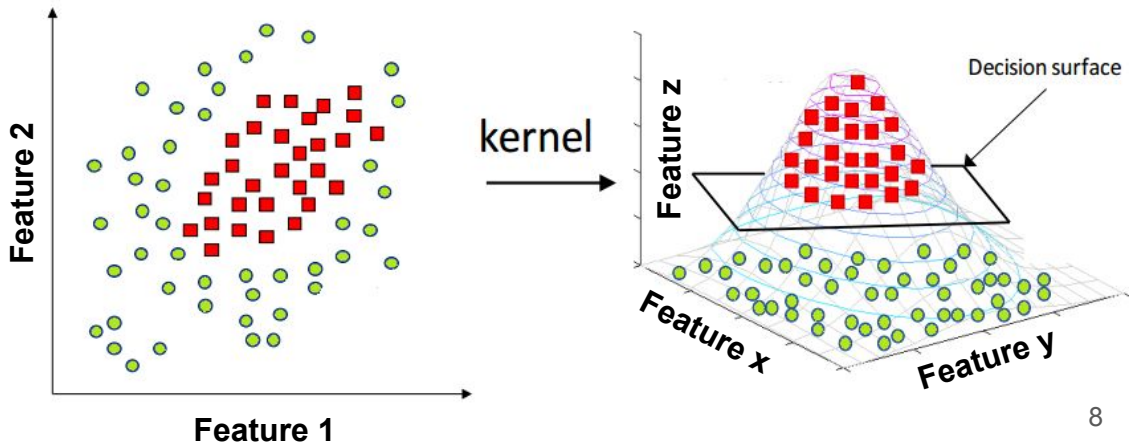


Das, 2008.

Support Vector Machine (SVM)

- Divides positive and negative classes through a separating hyperplane
- Support vectors are the points (actual data from training) that define the hyperplane
- Can also use ‘kernel trick’ which transforms the data projection into higher dimensional space to curve the separating hyperplane in lower dimensional space





SVM Classifier

- Classification is computed using the formula: $\sum_{i=1}^{m} \alpha_i y^{(i)} K\left(x^{(i)}, x\right) + b$
  - Alpha term is the support vector weights
  - y(i) is the support vector class (1 or -1)
  - x(i) is the support vector features, x is the input feature set
  - b is the intercept
  - K(x(i),x) is kernel / the similarity function between the support vector and input
    - Some common kernels:

Polynomial

$$K(\mathbf{q}, \mathbf{q}') = (1 + \mathbf{q} \cdot \mathbf{q}')^k$$

Gaussian radial basis function (RBF)

$$K(\mathbf{q}, \mathbf{q}') = \exp(-\|\mathbf{q} - \mathbf{q}'\|^2/\sigma^2)$$

  - **The sign of the resulting answer decides which class the new point belongs to**

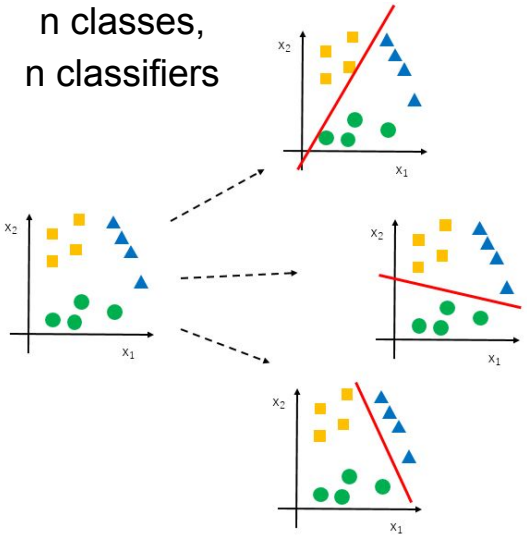
More info: https://github.com/ucberkeley-ee290c/fa18-wellness-monitor/blob/master/doc/svm_tutorial.md

SVM Classifier

- Classification can also be extended to multiple classes
  - Involves creating more than one classifier which vote for the final class

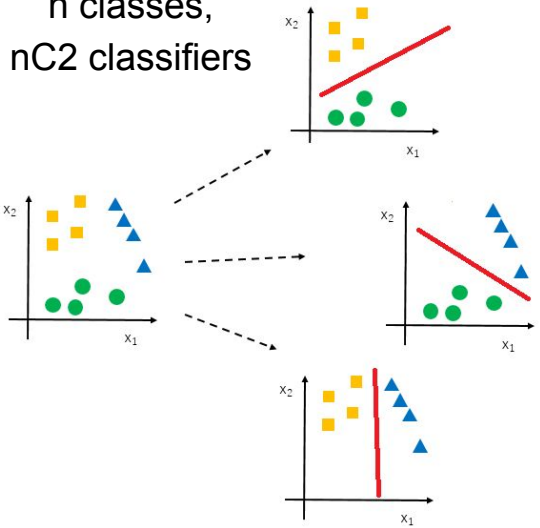
- **One vs Rest classification:**

n classes,
n classifiers



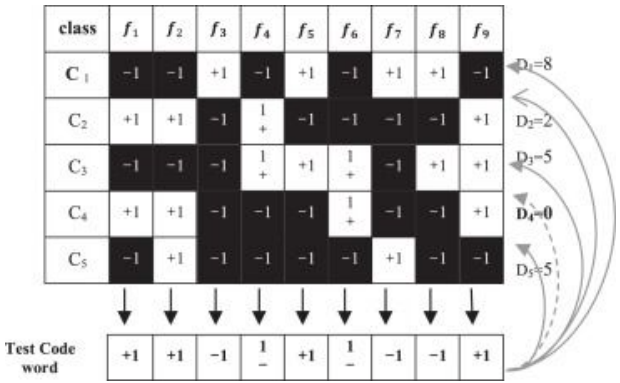
- **One vs One classification:**

n classes,
nC2 classifiers



- **Error-Correcting Output Codes**

n classes,
m classifiers, m>n

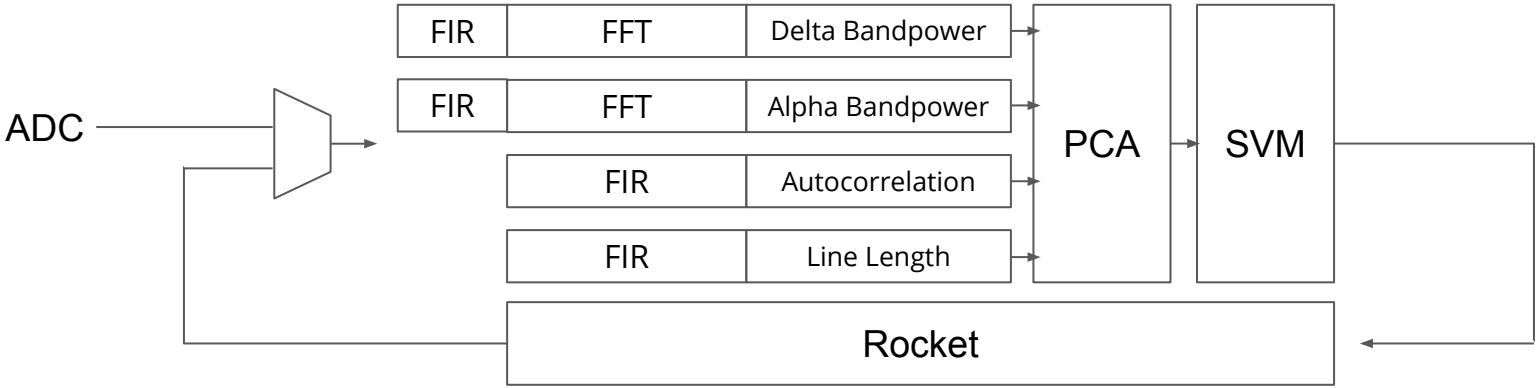


Hamming distance
for classification

More info: https://github.com/ucberkeley-ee290c/fa18-wellness-monitor/blob/master/doc/svm_tutorial.md

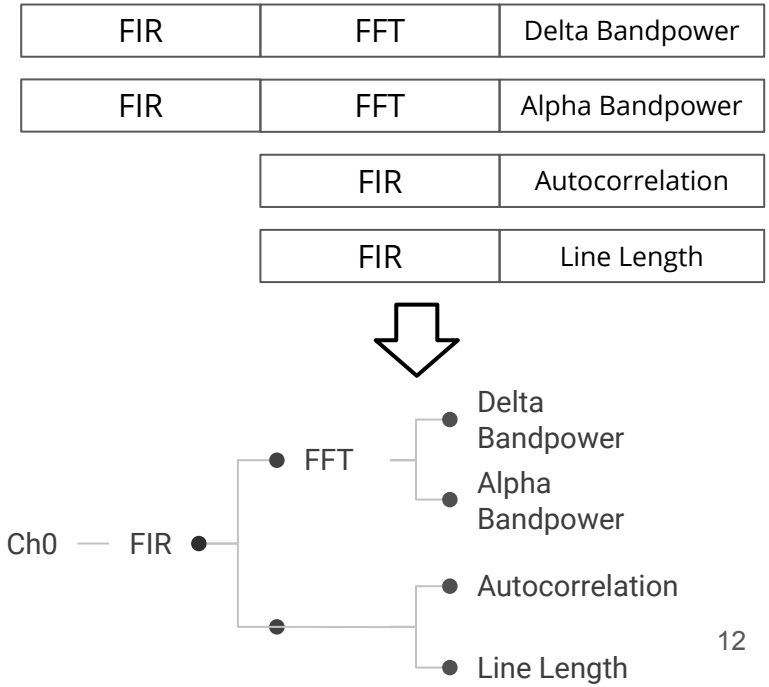
Wellness Generator Overview

- Takes specific features and generates a wellness monitor in hardware
- Currently the generator:
  - Creates collection of desired features
  - Decide on the “datapath” that computes the feature
  - Organizes datapaths in human readable data-structure
  - Generates modules
  - Automatically wires everything and connects monitor to a RISC-V core



Wellness Generator Next Steps

- Generator should not duplicate blocks - it should reuse them
  - Organize datapaths into a tree structure to reuse blocks
  - Generate the modules (as nodes in tree)
  - Connect the nodes
- Finish ‘wrapper’ app that parses user input
- Integrate into automated design flow



Testing

- Two main golden models:
  - ScalaTest
  - Python
- ScalaTest:
  - Unit tests
  - Generatable integration test
- Python
  - Functionally equivalent python model for a specific design
  - Used to confirm seizure detector could actually detect seizures

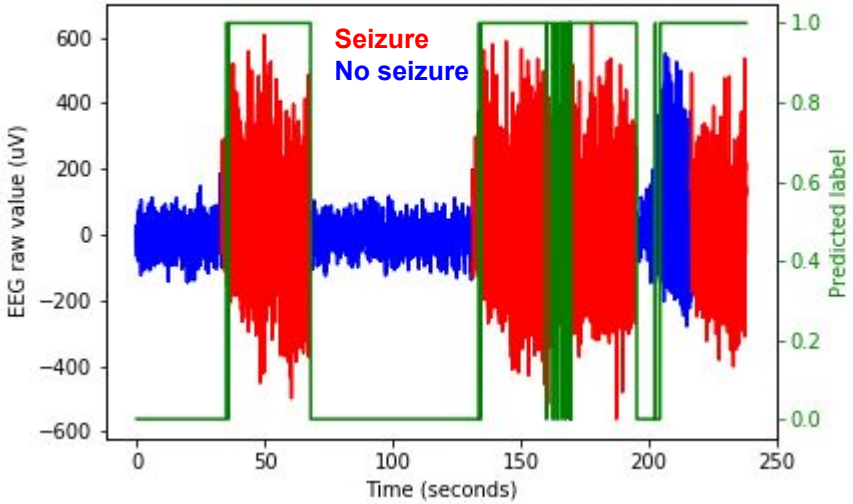
Application-Specific Training-Testing Setup

- Test framework for application-specific testing from raw data
- Starts from SVM training to C tests in a simulated RISC-V environment

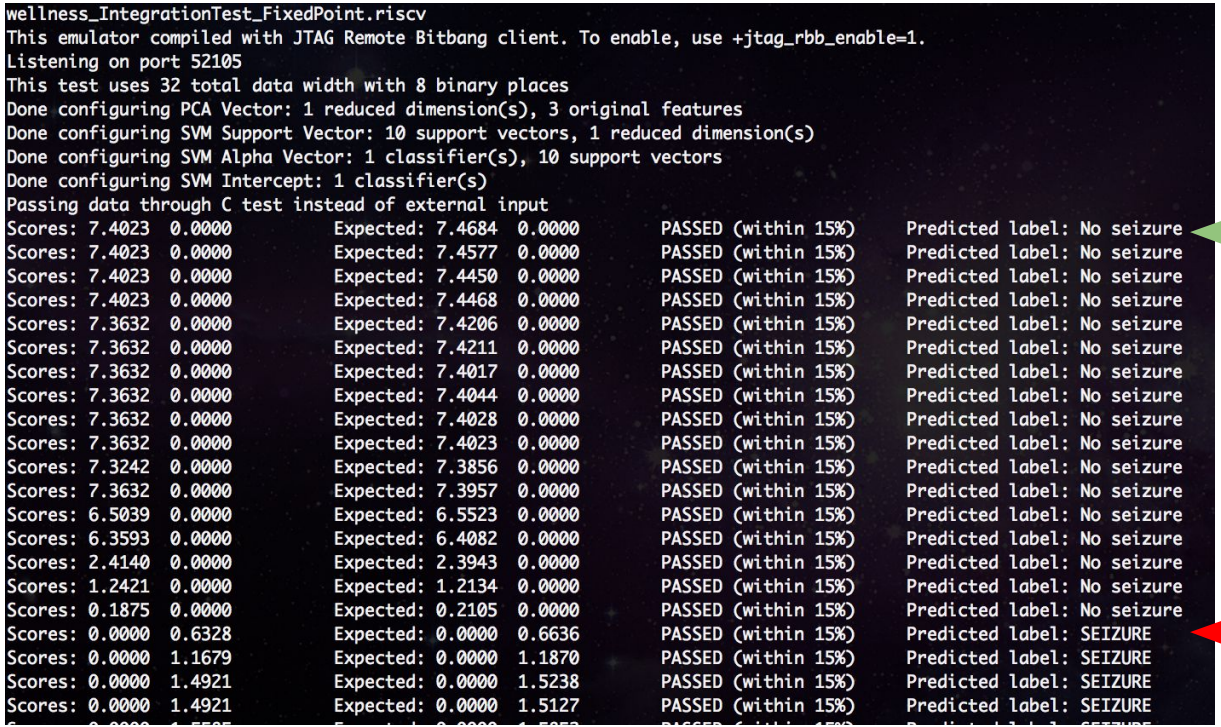
1. Python script creates SVM model given the data, writes out all configuration matrices and input vectors to CSV files
2. Scala tester reads CSV to get parameters, uses it to test Chisel datapath, writes out a C header file (.h) containing config matrices and I/O vectors
3. Header file is used by the C program to test the accelerator connected to the RISC-V core

* All generator parameters are set in the Python script, no need to change parameters in the Scala or C code

Trying out the framework on seizure data



Sample Execution (Seizure Detection)



No seizure

Seizure

Chisel HW Output

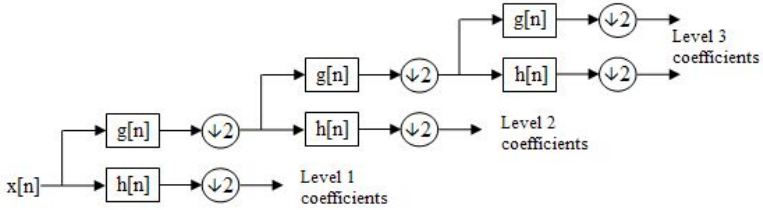
Golden Tester Output

Conclusion and Next Steps

- Flexible and scalable wellness monitor generator implemented in Chisel
- Seizure detector functionality confirmed through comparisons with Python model

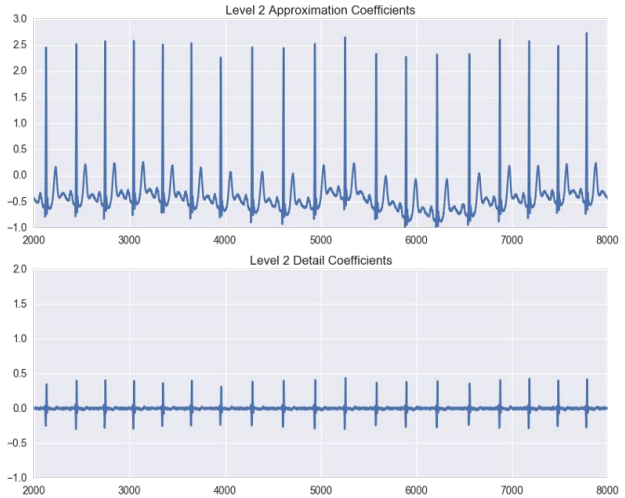
- Block optimizations/modularity (SVM and PCA)
- Optimize datapath generation
- Include new health feature extractors
- GUI for painless design space exploration

DWT Generator



Capture both frequency and temporal information about the signal

- Pass signal through lowpass & highpass filters
- Downsample
- Configurable number of levels, filter coefficients and length



Feature Extractor:

- DWT used to detect QRS waveform
- Tachycardia, bradycardia

https://medium.com/@andrewtan_36013/electrocardiograms-qrs-detection-using-wavelet-analysis-a1070505efee