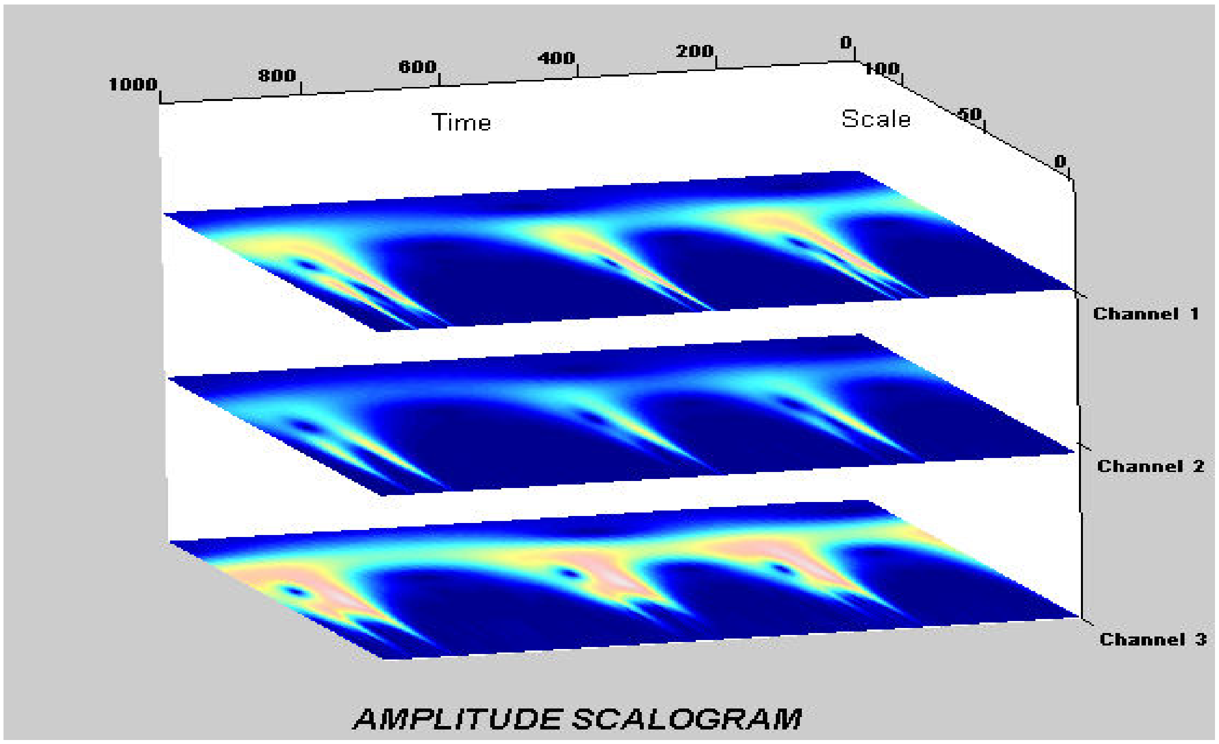
# ECE 496 Design Project: Hardware/Software Dynamic Visualisation of the EEG



Diane Kostka: 993662522
Rohan Karkhanis: 993866385
Supervisor: Berj L. Bardakjian
Administrator: Hans Kunov
Group #: 2008019
March 26, 2009

# ECE496Y Final Report Individual Evaluation Form (one sheet per student)

| | | | |
|---|---|---|---|
| Student | **Project ID:** 2008019 | **Project Title:** Hardware/Software Dynamic Visualization of the EEG | |
| | **Student Name:** Diane Kostka | | **Supervisor:** Berj L. Bardakjian |
| | **Section:** 2 | **Administrator:** Hans Kunov | |

| **Administrator's Evaluation**<br>Provide a rating for each section. Circle to indicate problem areas. | Excellent | Good | Adequate | Marginal | Poor/unclear | **Comments** *(also see comments in report)* |
|---|---|---|---|---|---|---|
| **Document** | | | | | | |
| **Introduction:** clear background, motivation, goals, requirements | | | | | | |
| **Final Design:** system diagram, system and module level descriptions, assessment of strengths and weaknesses | | | | | | |
| **Testing and Verification:** adequate documentation, discussion of results, comparison of results to requirements | | | | | | |
| **Summary and Conclusions:** summary of accomplishments and challenges, success in achieving project goals | | | | | | |
| **Presentation:** clear writing, grammar, organization, use of tables, diagrams, figures | | | | | | |
| **Project** | | | | | | |
| **Final outcome:** success in achieving stated project goals, technical complexity | | | | | | |
| Individual effort and contributions | | | | | | |

| | | |
|---|---|---|
| **Administrator's grade (/10):** | **Section average (/10):** | **Administrator's signature:** |

**General Comments:**

Note: Supervisor final evaluations are done online. See Supervisor > Final Evaluation on the course menu at http://int.ece.utoronto.ca/ece496.xxyy/index.html, where xxyy are the years (ex 0809 or 0910)

# ECE496Y Final Report Individual Evaluation Form (one sheet per student)

| | | |
|---|---|---|
| Student | **Project ID:** 2008019 | **Project Title:** Hardware/Software Dynamic Visualization of the EEG |
| | **Student Name:** Rohan Karkhanis | **Supervisor:** Berj L. Bardakjian |
| | **Section:** 2 | **Administrator:** Hans Kunov |

## Administrator's Evaluation

Provide a rating for each section. Circle to indicate problem areas.

| | Excellent | Good | Adequate | Marginal | Poor/unclear | **Comments** *(also see comments in report)* |
|---|---|---|---|---|---|---|
| **Document** | | | | | | |
| **Introduction:** clear background, motivation, goals, requirements | | | | | | |
| **Final Design:** system diagram, system and module level descriptions, assessment of strengths and weaknesses | | | | | | |
| **Testing and Verification:** adequate documentation, discussion of results, comparison of results to requirements | | | | | | |
| **Summary and Conclusions:** summary of accomplishments and challenges, success in achieving project goals | | | | | | |
| **Presentation:** clear writing, grammar, organization, use of tables, diagrams, figures | | | | | | |
| **Project** | | | | | | |
| **Final outcome:** success in achieving stated project goals, technical complexity | | | | | | |
| Individual effort and contributions | | | | | | |

| | | |
|---|---|---|
| **Administrator's grade (/10):** | **Section average (/10):** | **Administrator's signature:** |

**General Comments:**

Note: Supervisor final evaluations are done online. See Supervisor > Final Evaluation on the course menu at http://int.ece.utoronto.ca/ece496.xxyy/index.html, where xxyy are the years (ex 0809 or 0910)

**The Edward S. Rogers Sr. Department of Electrical and Computer Engineering**

**University of Toronto**

**ECE496Y Design Project Course**

**Group Final Report**

| | | |
|---|---|---|
| Title: **Hardware/Software Dynamic Visualization of the EEG** | | |
| Project I.D.#: | 2008019 | |
| Team members: (Select one member to be the main contact. Mark with '*') | Name: Diane Kostka* | Email: diane_koskta@hotmail.com |
| | Rohan Karkhanis | r.karkhanis@utoronto.ca |
| | | |
| | | |
| Supervisor: | Berj L. Bardakjian | |
| Section #: | 2 | |
| Administrator: | Hans Kunov | |
| Submission Date: | March 26, 2009 | |

# Group Final Report Attribution Table

This table should be filled out to accurately reflect who contributed to each section of the report and what they contributed. Provide a **column** for each student, a **row** for each major section of the report, and the appropriate codes (e.g. 'RD, MR') in each of the necessary **cells** in the table. You may expand the table, inserting rows as needed, but you should not require more than two pages. The original completed and signed form must be included in the hardcopies of the final report. Please make a copy of it for your own reference.

| **Section** | **Student Names** | |
|---|---|---|
| | Diane Kostka | Rohan Karkhanis |
| Executive Summary | RD | ET |
| Summary of Groups Progress | RD | ET |
| Background and Motivation | RD RS | ET |
| Project Goal | RD RS | ET RS |
| Project Requirements | RD | ET RS |
| Technical Design: Software Subsystem | RD RS | ET |
| Technical Design: Interface Subsystem | ET | RD RS |
| Technical Design: Hardware Subsystem | ET | RD RS |
| Assessment of Final Design | ET | RD |
| Testing and Verification | ET RS | RD RS |
| Conclusion | ET | RD |
| Appendix | RD ET | RD |
| All | CM FP | CM FP |

## Abbreviation Codes:

Fill in abbreviations for roles for each of the required content elements. You do not have to fill in every cell. The "**All**" row refers to the complete report and should indicate who was responsible for the final compilation and final read through of the completed document.

RS – responsible for research of information
RD – wrote the first draft
MR – responsible for major revision
ET – edited for grammar, spelling, and expression
OR – other
"All" row abbreviations:
FP – final read through of complete document for flow and consistency
CM – responsible for compiling the elements into the complete document
OR - other

## Signatures

By signing below, you verify that you have read the attribution table and agree that it accurately reflects your contribution to this document.

| | | | | | |
|---|---|---|---|---|---|
| **Name** | **Diane Kostka** | **Signature** | | **Date:** | **March 26, 2009** |
| **Name** | **Rohan Karkhanis** | **Signature** | | **Date:** | **March 26, 2009** |

# Voluntary Document Release Consent Form[1]

To all ECE496 students:

To better help future students, we would like to provide examples that are drawn from excerpts of past student reports. The examples will be used to illustrate general communication principles as well as how the document guidelines can be applied to a variety of categories of design projects (e.g. electronics, computer, software, networking, research).

Any material chosen for the examples will be altered so that all names are removed. In addition, where possible, much of the technical details will also be removed so that the structure or presentation style are highlighted rather than the original technical content. These examples will be made available to students on the course website, and in general may be accessible by the public. The original reports will not be released but will be accessible only to the course instructors and administrative staff.

Participation is completely voluntary and students may refuse to participate or may withdraw their permission at any time. Reports will only be used with the signed consent of all team members. Participating will have no influence on the grading of your work and there is no penalty for not taking part.

If your group agrees to take part, please have all members sign the bottom of this form. The original completed and signed form should be included in the hardcopies of the final report.

Sincerely,
Phil Anderson
ECE496Y Course Coordinator

## Consent Statement

We verify that we have read the above letter and are giving permission for the ECE496 course coordinator to use our reports as outlined above.

Project ID: 2008019 Project Title: Hardware/Software Dynamic Visualization of the EEG

Supervisor: Berj L. Bardakjian Administrator: Hans Kunov

| | | | | | |
|---|---|---|---|---|---|
| **Name** | **Diane Kostka** | **Signature** | | **Date:** | **March 26, 2009** |
| **Name** | **Rohan Karkhanis** | **Signature** | | **Date:** | **March 26, 2009** |

[1] This form will be detached from the hardcopy of the final report. Please make sure you have nothing printed on the back page.

Attribution Table, ECE496
Team # 2008019 Project title: Hardware/Software Dynamic Visualization of the EEG

Team members: Diane Kostka & Rohan Karkhanis

| Task | Student name | | Completion |
|---|---|---|---|
| | Diane Kostka | Rohan Karkhanis | |
| Defining the core problem | J | J | OS |
| Motivation and background | R | | OS |
| Matlab research | R | A | OS |
| Matlab code to extract data | R | | OS |
| Plot spectral components of data | R | | D10 |
| Stacking of scalograms | R | | D10 |
| Validation and testing using EEG | R | | D15 |
| FPGA research | | R | OS |
| GPU research | | R | OS |
| Evaluating hardware | | R | D5 |
| Obtain hardware | | R | D10 |
| Simulation of design | | R | D15 |
| Embedded M-code | R | | D20 |
| Synthesize sample design | | R | D20 |
| Hardware synthesis | J | J | I |
| Validation and testing of hardware | J | J | I |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

R = responsible; A = assisting; J = joint work

OS = on schedule
I = incomplete
DX = delayed X days according to Gantt chart located in Appendix A

Signature and date of all members of the team:

| | | |
|---|---|---|
| Diane Kostka | Signature: | March 26, 2009 |
| Rohan Karkhanis | Signature: | March 26, 2009 |

## **Executive Summary** (author: D. Kostka)

Active research is being carried out to develop computer algorithms that can automatically detect the origin of a seizure and track its progression through time and space. This project will act as a visual tool for researchers to help identify features associated with abnormal electrical activity in the brain, consequently, using the information to develop algorithms to track these abnormalities.

The goal of the project is to create a software implementation of the time-frequency-space and power spectral strength visualisation of EEG signals and then optimize the implementation through the use of hardware. Wavelet transform is required to extract the time-frequency and power spectral components of the signal.

In the software component, the wavelet analysis was performed by using a log Gabor filter of varying scale and was successfully implemented to produce the required visualisation. The log Gabor filter allowed a simpler translation of software code into Verilog code for hardware implementation. The interface between software and hardware was created by converting the Matlab M-code into Embedded Matlab Code and through the use of Hardware in the Loop and HDL Import DSP Builder blocks.

The hardware component of the wavelet analysis consisted of a DSP Builder built-in FFT block coupled in a loop with the scaled wavelet. While the individual sub-functions perform, the module as a whole fails to compile due to syntax and functional errors during integration.

Testing of the visualisation produced was done by creating an active comparison with that produced by built-in Matlab functions. Furthermore, each individual Verilog sub-function was tested in Quartus using an input waveform.

In conclusion, the hardware implementation failed to meet requirements due to time constraints while the software implementation successfully created a visual of stacked plots produced from each electrode.

**Summary of Groups Progress** (author: D. Kostka & R. Karkhanis)

Some of the key accomplishments during the period since the design review have been:

1. Development of software implementation of the visualization.
2. Testing and verification of software implementation method with other alternatives.
3. Translation of the software code into Embedded Matlab code for the creation of the Simulink interface between the hardware and software components.
4. Utilizing DSP Builder to synthesize designs on to the FPGA.
5. Testing of the DSP built-in FFT block through the use of HIL (Hardware in the Loop)
6. Coding of the hardware implementation of the wavelet analysis in Verilog.
7. Importing Verilog into Simulink through the use of HDL Import to integrate the software and hardware.

The software design has been successfully implemented and tests were performed by comparing the resulting visualisation to those produced by alternative solutions.

The interface between the software and hardware components has been successfully created. However, the hardware implementation has failed to be successfully completed. The individual modules for the wavelet analysis method compiled but the integration of the modules to perform the wavelet transform failed to compile.

**Individual Progress and Contributions: D. Kostka** (author: D. Kostka)

My main contribution to the project thus far has been the coding of the software implementation of the EEG visualization using Matlab. This included the design choices concerning the mode of displaying the data, the choice of signal analysis methods to be used on the EEG data as well as the testing of the final implementation using clinical data. A sizeable portion of my time was also spent in constructing the Verilog code for the wavelet analysis.

| | **Task Title** | **Status** |
|---|---|---|
| i | Developed Matlab code to plot frequency, time and power spectral components (scalograms) of EEG data | Completed |
| ii | Stacked scalograms produced from each electrode in order to display the spatial progression of the EEG signal | Completed |
| iii | Coded the hardware implementation of the wavelet analysis in Verilog HDL. | This step has not been successfully completed due to compilation errors caused in the integration of sub functions. |
| iv | Tested two methods of software implementation; namely log Gabor and Daubechi implementations of wavelet transform; and accessed them in terms of ease of implementation in hardware and accuracy in | Complete |

| | | |
|---|---|---|
| | capturing high frequency data. | |
| v | Translated Matlab M-code into Embedded Matlab code required for the hardware – software interface in Simulink. | Completed |

**Individual Progress and Contributions: R. Karkhanis** (author: R. Karkhanis)

My main contribution to the project thus far has been the testing of Verilog code and synthesis of the DSP Builder blocks onto hardware to optimize the software implementation of the EEG visualization in Simulink. This included the design choices concerning the choice of hardware (FPGA vs. GPU), manufacturer (Altera vs. Xilinx) and the choice of modes in hardware (burst mode vs. frame mode). The interface design required a high level approach in order for the hardware and software to cooperate. I ran a lot of tutorials and read a lot of documents to understand the complexity of hardware.

| **Task #** | **Task Title** | **Status** |
|---|---|---|
| i | Tested hardware tools, the hardware – software interface and the DSP Builder blocks. | Completed |
| ii | Tested the interface between Simulink and FPGA board using a frequency sweep function. | Completed |
| iii | Demonstrated optimized algorithm speed when run on hardware as opposed to software | Completed |
| iv | Designed Verilog code for the implementation of the | Incomplete |

| | | |
|---|---|---|
| | Wavelet Analysis | |
| v | Hardware Synthesis – Integrate the Hardware and Software blocks, using the interface to communicate | Incomplete |

**Acknowledgements** (author: R. Karkhanis)

This Design Project would not have been, in the current state, without the encouragement and commitment from our contributors. We would like to thank our supervisor Berj L. Bardakjian and administrator Hans Kunov for continuous assistance and encouragement. In addition, we appreciate all the help from Professor Bardakjian’s graduate students Dave Stanley, Josh Dian, and Marija Cotic. We would also like to thank our fellow student Raihan Masroor for spending time to explain to us the FFT design in Simulink.

Thanks to Phil Anderson for helping us understand the motivation of the project and to Peter Weiss for emphasizing the importance of silence and pause during a presentation.

**Table of Contents**

## 1. Introduction

This report summarizes the motivation, design, implementation and testing of the four dimensional visualization of electroencephalographic (EEG) signals as part of our final year design project course, ECE496. The report concludes with suggestions of improvements and future work.

### 1.1 Background and Motivation (author: D. Kostka)

The monitoring of epileptic patients relies on the identification of parts of the brain where abnormal electrical activity occurs and the various features associated with these abnormalities. This monitoring is made possible through the use EEG signals obtained by the placement of extracranial and intracranial electrodes on the patient. Seizure activity is identified by the frequency components and the spectral strength of the EEG signals, in particular the high frequency components.

To accurately recognize changes in brain activity from the EEG signals captured from multiple electrodes, various visualization technologies have been developed. One such technology, the Ray Tracing Technique, maps electrode position and interpolates scalp potential onto morphological data acquired from CT and MRI devices [1]. Butterfly plots employ an organization of data similar to the conventional single channel EEG representation except that the signals from all electrodes are superimposed [2].

These existing solutions suffer from long computational times and the inability to effectively track the progress of a seizure activity on a single plot since most do not simultaneously display the time, frequency, spatial and power spectral components of the EEG signal. The goal of this design project is to create a software implementation for the visualization of the power spectral, time, frequency and spatial components of EEG signals. The processing time for the visualization will be further optimised by performed the signal analysis on a hardware platform.

The method of data interpretation and pattern recognition chosen for this project is visualization due to the fact that the human visual system is a pattern seeker of enormous power and subtlety. The eye and the visual cortex of the brain form a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers, hence, making the words *understanding* and *seeing* synonymous [3].

This design would serve as a visual tool for researchers to help identify features associated with seizure activity and consequently develop algorithms that can effectively recognise and track the spread of seizure activity in the brain. A graphical representation of stacked plots employed in the design allows the researchers to easily compare and track frequency activity from different electrodes as it travels in time and space.

Currently researchers can only analyse the time, frequency and spectral components of data plotted from a single channel at a time, making the task where a large number of electrodes are involved, extremely rigorous. As the number of patients increase, clinicians need quick computerized methods that will enable them to establish the point of origin of a seizure and track its movement in time and space. Our design will give researchers and designers a tool to make that need a reality.

Since EEG signals vary both in time and frequency, a Fast Fourier Transform (FFT) is disadvantageous as it has only frequency resolution and no time resolution. To overcome this problem a method of signal decomposition called the Short Term Fourier

Transform (STFT) was implemented. In STFT, the signal is divided into small enough segments, where these segments (portions) of the signal can be assumed to be stationary. For this purpose, a window function "w" is chosen. The width of this window must be equal to the segment of the signal where its stationarity is valid. The problem with STFT lies in what is known as the Heisenberg Uncertainty Principle. This principle states that one cannot know the exact time-frequency representation of a signal, i.e., one cannot know what spectral components exist at what instances of times but one can know the time intervals in which certain bands of frequencies exist.

This problem arises from the choice of width of the window function that is used. Since in STFT, the window is of a finite length, only a portion of the signal is covered hence causing poorer frequency resolution. If a window of infinite length is used, the FT is calculated, which gives perfect frequency resolution, but no time information. The Wavelet Transform (WT) was developed to overcome the resolution problem of the STFT. The wavelet analysis is done in a similar way to the STFT analysis, in the sense that the signal is multiplied with a function i.e. the wavelet similar to the window function in the STFT, and the transform is computed separately for different segments of the time-domain signal. However, the main difference between the STFT and the WT is the fact that the width of the window is changed as the transform is computed for every single spectral component, hence offering both good time and frequency resolution. Thus the wavelet transform method forms the basis for signal analysis in this project, specifically the Discrete Wavelet Transform (DWT) which reduces computational time by performing analysis on a sampled version of the signal [4].

The DWT analyzes the signal at different frequency bands with different resolutions by decomposing the signal into a coarse approximation and detail information. DWT employs two sets of functions, which are associated with low pass and high pass filters. The decomposition of the signal into different frequency bands is simply obtained by successive high pass and low pass filtering of the time domain signal. The original signal x[n] is first passed through a halfband high pass filter g[n] and a low pass filter h[n]. After the filtering, half of the samples can be eliminated according to the Nyquist’s rule, and the signal can therefore be sub-sampled by 2, simply by discarding every other sample as demonstrated in Figure 1.

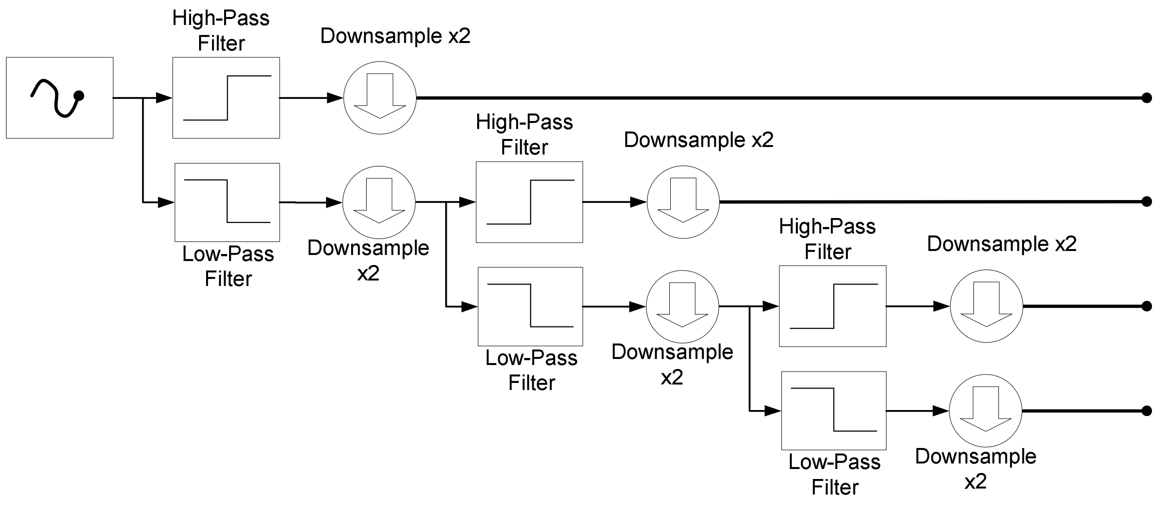


Figure 1. Discrete Wavelet Transform Breakdown

In order to optimize our project, the design consisted of a hardware implementation. Many high level languages (C++, Java, and Matlab) do not use the hardware they are run on efficiently. In order to increase performance of the calculations, a lower language must be used to optimize the usage of hardware. Field Programmable Gate Array (FPGA) is a customizable chip used to optimize software designs. Through the use of Verilog or another hardware descriptive language (HDL), the customizable chip can be designed to the designer's specification. FPGAs are formidable in their use of parallelization, which decrease processing times, especially when used for convolution and FFTs [5].

## 1.2 Project Goal

The goal of this project is to improve the diagnostic tools available for epileptic patients by designing a software and hardware implementation of an amplitude-time-frequency-space visualization of the EEG readings. This would enable researchers to study the features associated with seizure activity by creating a mapping of the brain activity that is both easy to interpret as well as does not require extreme computational time.

## 1.3 Project Requirements

### *1.3.1 Functional Requirements*

1. The software implementation of the visualization shall take in raw EEG data and output a four dimensional (power spectral-time-space-frequency) mapping of the EEG signal.
2. The hardware implementation of the wavelet analysis shall take in raw EEG data and output the time-frequency and power spectral components of the signal.
3. The hardware implementation of the signal transform shall take in a signal and output its time-frequency spectrum.
4. The hardware implementation of the visualization shall be proved to be faster than the software implementation.

5. The EEG signals shall be processed in order to extract the power spectral, temporal and frequency components.
6. The frequency-time-spectral information from each EEG electrode shall be stacked to add the spatial component to the final visualization.
7. The hardware and software components should integrate.

### *1.3.2 Constraints*

1. The final visualization shall display all four aspects of the EEG signal i.e. the frequency, temporal, spectral and spatial components.
2. The visualization must differentiate between EEG signals coming from each electrode.
3. The visualization shall be implemented on a hardware interface to increase computational speed.

### *1.3.3 Objectives*

1. The visualization shall be easy to interpret by the user and data from each electrode should be easily identifiable.
2. The design should be scalable so that the design could be miniaturized and optimized in the future.

## 2. Technical Design

## 2.1 System-Level Overview (author: R. Karkhanis & D. Kostka)

The entire system consists of three main components namely the software, hardware and interface, as demonstrated in Figure 2. The software component forms the initial proof of concept basis of the design, while the hardware component acts as a design optimization in order to reduce computational time.

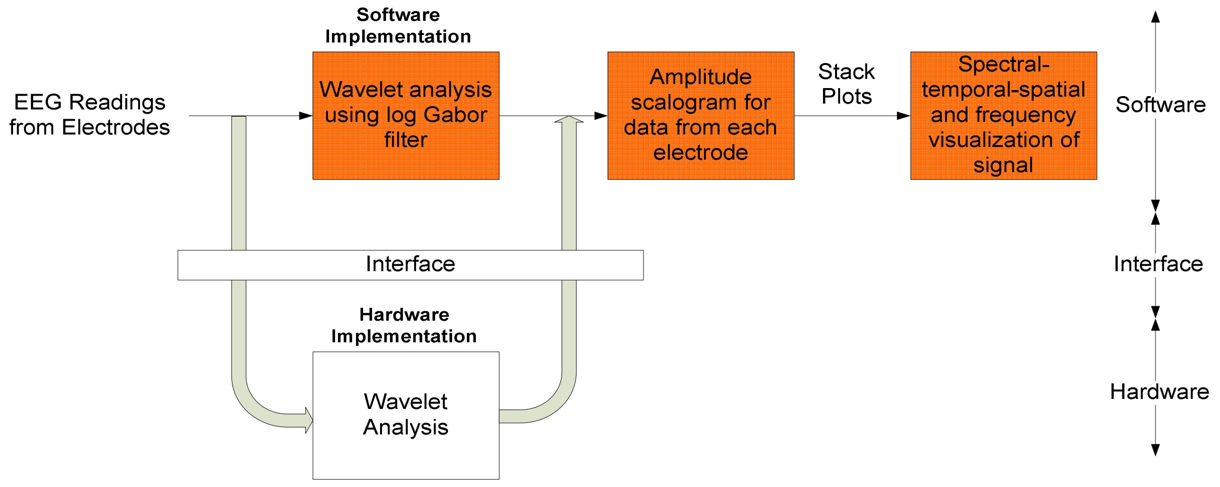

Software Implementation
EEG Readings from Electrodes
Wavelet analysis using log Gabor filter
Amplitude scalogram for data from each electrode
Stack Plots
Spectral-temporal-spatial and frequency visualization of signal
Software
Interface
Interface
Hardware Implementation
Wavelet Analysis
Hardware

Figure 2. System Level Overview

For the 4-D visualisation, the EEG signal from each electrode is first read into Matlab in the form of a .MAT file. This signal is then broken done into the time-frequency and power spectral components using Wavelet Analysis methods. This is visualised via a scalogram, which is a time-frequency graphical representation in which each pixel is color coded with an intensity that corresponds to the power spectral value at that point. This signal decomposition was first implemented in software and later in hardware so as to increase speed. The deconstructed data from each channel/electrode is then fed into the plotting algorithm which stacks scalogram produced by each channel, so as to add the spatial dimension to the visualisation.

To allow for the interface between the Matlab coded software implementation of the plotting function and the Verilog coded hardware implementation of wavelet analysis, several Simulink blocks were used. Simulink allows the creation and synthesis of design blocks that recognise Embedded Matlab code and Verilog HDL code. Since Simulink does not recognise Matlab M-code, a majority of the software modules had to re-coded and translated into Embedded Matlab.

The hardware subsystem takes advantage of Altera DSP Builder design blocks to synthesize the design onto the FPGA. The wavelet analysis required the use of a FFT block, which is a Megacore function in the DSP Builder blockset. The details of each subsystem are discussed below.

## 2.2 Software Subsystem (author: D. Kostka)

### *2.2.1 Data Acquisition*

Data is loaded onto Matlab using the ‘load’ command. The data was inputted in the form of a text file where each column corresponded to a separate channel/electrode placement.

### *2.2.2 Signal Analysis*

The signal was decomposed using discrete wavelet transforms via the log Gabor filter method. This method was chosen over using built-in discrete wavelet transform analysis as it would be easier to implement in hardware it terms of coding complexity.

### *2.2.3 Wavelet construction*

The Log-Gabor filter has a response that is Gaussian when viewed on a logarithmic frequency scale instead of a linear one. This allows more information to be captured in the high frequency areas and also has desirable high pass characteristics. The log Gabor filter is defined as,

$$G(f) = \exp\left[-\frac{\log\left(\frac{f}{k}\right)}{2\log\left(\frac{\sigma}{k}\right)}\right]$$

where k = [*u*0 *v*0 *w*0 ···]*T* is the centre frequency of the sinusoid and s is a scaling factor of the bandwidth [6].

For the design, the Gabor wavelet was built using pre-existing code and is described as follows:

```
 radius =  [0:fix(ndata/2)]/fix(ndata/2)/2;
radius(1) = 1;

fo = 1.0/wavelength;      % Centre frequency of filter.

% Construct filter
logGabor(1:ndata/2+1) = exp((-(log(radius/fo)).^2) / (2 *log(sigmaOnf)^2));
logGabor(1) = 0;
```

where ‘ndata’ is the size of the data array being analysed.

### *2.2.4 Wavelet Transform*

For each scale of the wavelet, a convolution between the fast Fourier transform of the signal and the wavelet filter was determined. The inverse Fourier transform was taken to obtain the spectral amplitude of the signal. At each iteration, the wavelength of the filter was changed by a pre determined factor. This method of implementation was equivalent to performing a discrete wavelet transform on the signal. This step was repeated to extract the power spectral components for data from each channel.

The following code demonstrates the wavelet analysis procedure used:

```
for row = 1:nscales
  % Centre frequency of filter.
  fo = 1.0/wavelength;

  % Construct filter
  logGabor(1:ndata/2+1)=exp((-(log(radius/fo)).^2)/(2 * log(sigmaOnf)^2));
  logGabor(1) = 0;

  % Multiply filter and FFT of signal, then take inverse FFT.
  EO = ifft(signalfft .* logGabor);
  % Record the amplitude of the result
  channel(i).amplitude(row,:) = abs(EO);
  % Increment the filter wavelength.
  wavelength = wavelength * mult;
end
```

(For entire function please refer to Appendix C)

### *2.2.5 Plotting*

The method of displaying data to the end user was evaluated in term of ease of readability and interpretation. We chose to display the time, frequency, electrode location and power spectral data in the form of stacked scalograms as opposed to a running movie of scalograms produced per electrode. This choice was based on the fact that using a movie method of visualisation would require the researcher to remember data from an earlier plot in order to track changes. However, in the stacking method changes in EEG readings can be traced visually as they travel in both time and space.

Using the Matlab ‘surface’ function, the power spectral data extracted from the signals were plotted on their corresponding scalograms. The decision to display the data in the form of a scalogram (where spectral parameters are determined using wavelet analysis) as opposed to a spectrogram was based on the fact that the time resolution of the scalogram is superior, especially at lower scales where contours are well defined in time.

### *2.2.6 Channel Stacking*

The 'surface' command was used to create four surfaces at different elevations in the z-axis:

```
surface(imgstruct(k).x_axis, imgstruct(k).y_axis,3*(k -2)*ones(sizeimg(k,1),sizeimg(k,2)),
imgstruct(k).ccfs, 'EdgeColor','none');
```

Each surface represents the scalogram produced for a single electrode (Ref Figure 4). To allow for easy visualisation of data at each level of the stacked plot alpha mapping was used to set the transparencies of the different scalograms with the bottommost layer being the most opaque. Alpha mapping is a Matlab property by which each element in the alpha data maps to a transparency value in the plot. (For complete code refer to Appendix D).

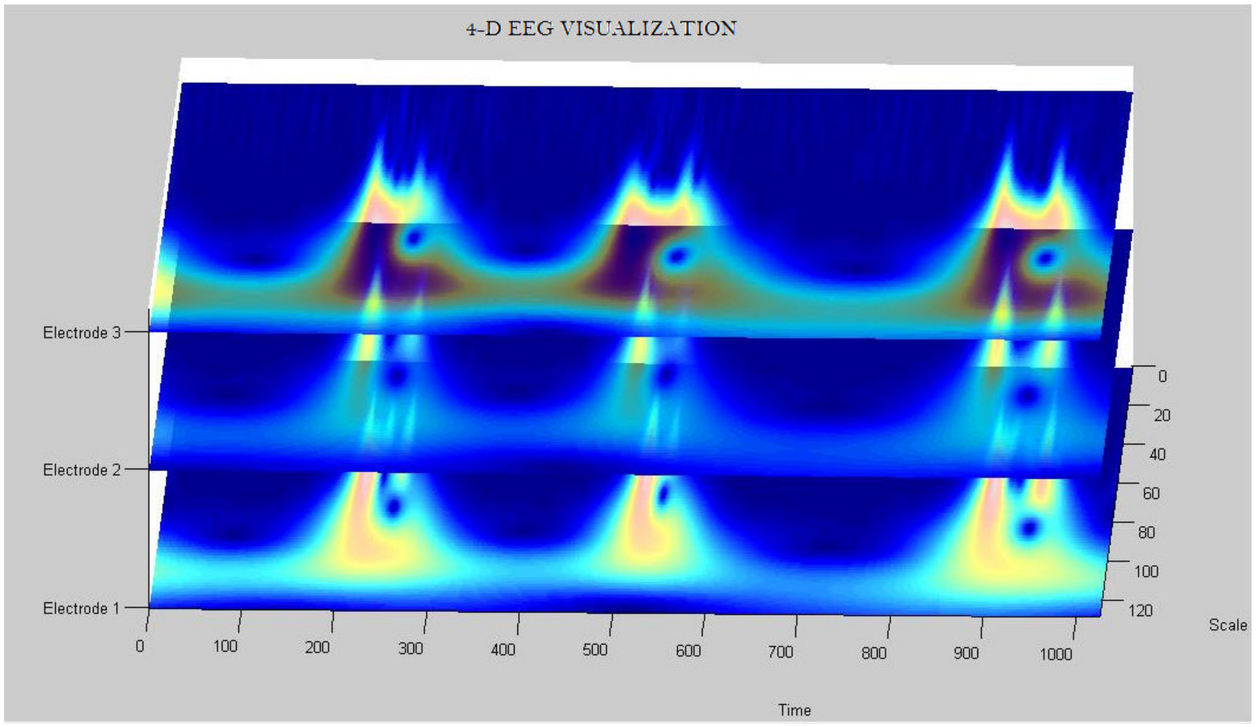


Figure 4 Result - Four Dimensional Visualization

## 2.3 Interface Subsystem

To increase computational speed, the wavelet analysis was implemented in Verilog and run on an FPGA board. However, data acquisition and plotting of the visualization were carried out in software. To allow for this interaction between the hardware and software components of the design, Simulink blocks were used.

Simulink is an environment for multi-domain simulation and Model-Based Design of dynamic and embedded systems. The core components used in Simulink to enable this interface were Hardware in the Loop (HIL), Hardware Descriptive Language (HDL) Import, and Embedded Matlab blocks.

*<u>2.3.1 Hardware In the Loop</u>* (author: R Karkhanis)

The HIL block's function was to incorporate the hardware into the design. The visualization was done using Matlab code and needed to integrate with the hardware. Simulink blocks were used to synthesize functions onto the hardware. Incorporating HIL block into the Simulink model allowed us to co-simulate a software Matlab design with the FPGA board on which the design was synthesized. Through the use of HIL, we were capable of optimizing the design and obtained faster simulation times.

The Altera DSP Builder blockset provides blocks that can be synthesized on to a FPGA. Once a block is used, it is compiled and a Quartus project file is created. The HIL block uses this project file to implement the design.

The following picture, Figure 5, from Altera's DSP Builder User Guide demonstrates the system level flow of the HIL block [7].

**Figure 5–1.** System-Level Design Flow

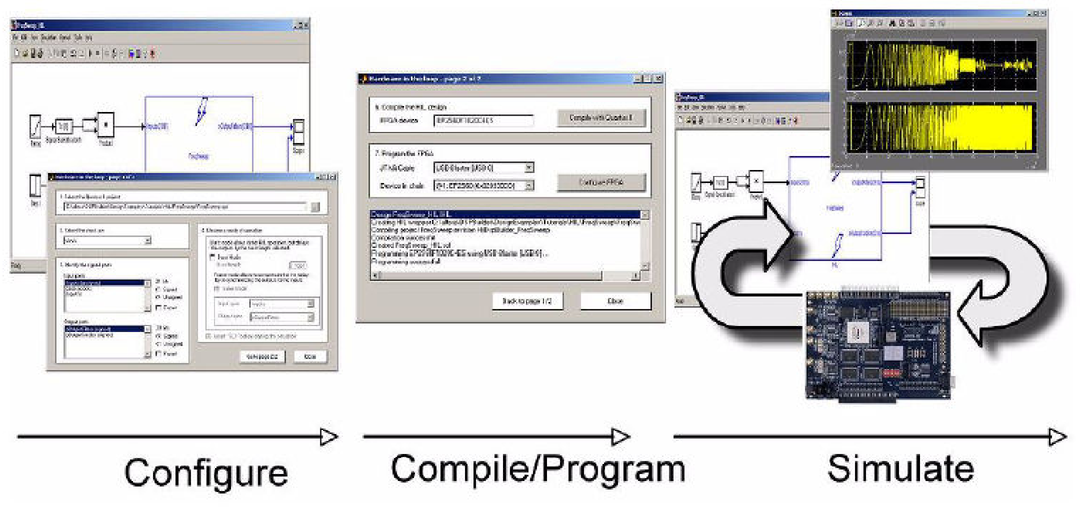


Figure 5. Co-simulation of software and hardware

The FFT HIL block shown below in Figure 6, has the input and output pins displayed for simple hookup.

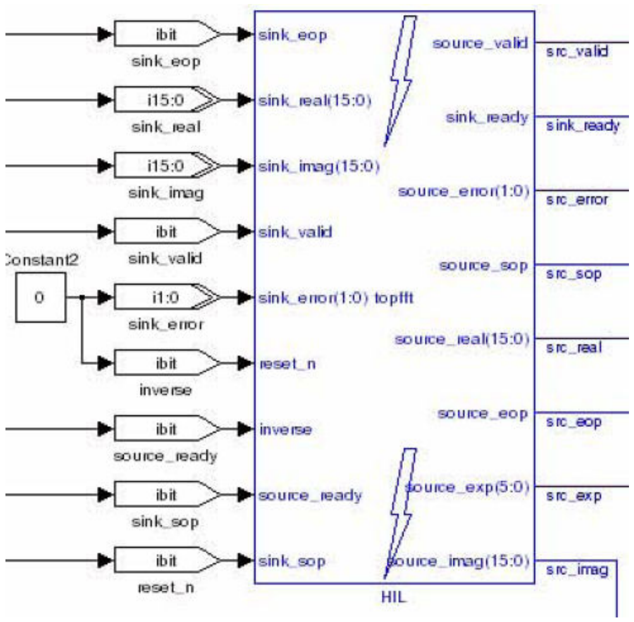


Figure 6. HIL of FFT

### *<u>2.3.2 HDL Import</u>* (author: R. Karkhanis)

HDL import is a Simulink block that is used to incorporate hardware designs into Simulink. HDL Import allowed the integration of the Verilog design into Simulink. Thus the Verilog design of the wavelet analysis is imported into a black box.

Through the use of HDL Import and HIL we were able to co-simulate Verilog design and DSP Builder existing hardware blocks with other simulation blocks. However, the code developed in Matlab cannot directly interact with the hardware since hardware involves the use of bits, buses, and wires. In this case, Embedded Matlab code is required.

### *<u>2.3.3 Embedded Matlab Blocks</u>* (author: D.Kostka)

To include Matlab algorithms in Simulink models for efficient system simulation and code generation, embedded Matlab code was used. The Simulink layout of the interface in shown in Figure 6 below. A Simulink 'From File' block was used to read EEG data from a .MAT file. The data from each channel was then sent to a data processing blocks to extract a portion of the EEG signal to be analysed. This data was sent in parallel to a signal analysis block, coded in Embedded Matlab that performed the wavelet decomposition of the signal. [8]

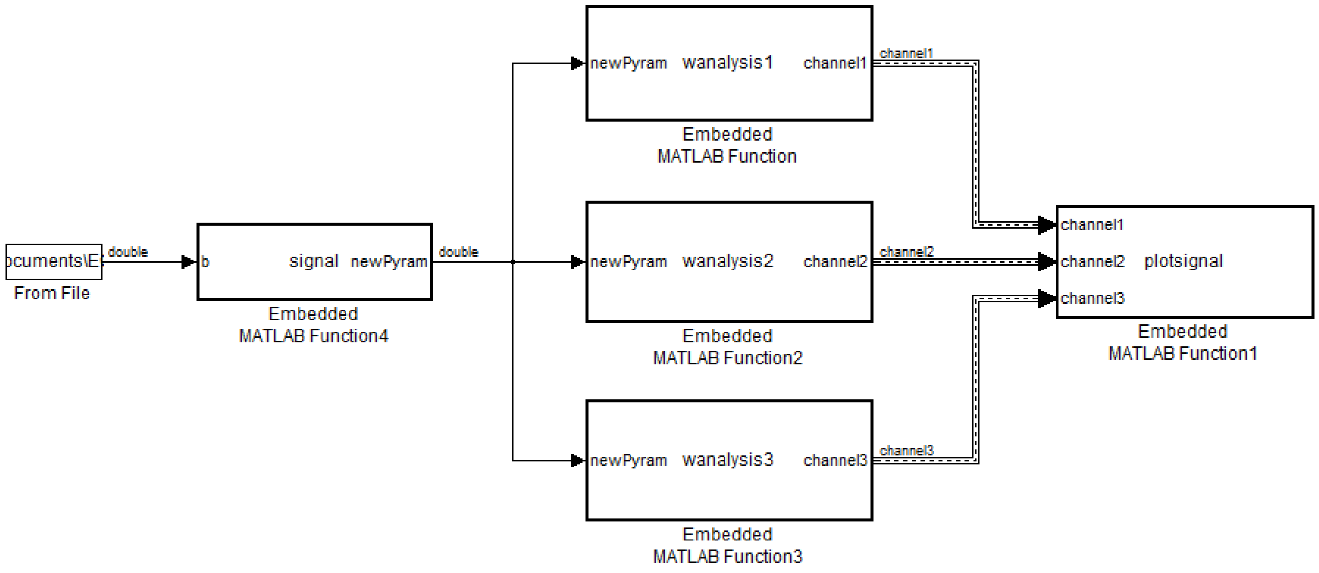

ocuments\E
From File
double
b
signal newPyram
Embedded MATLAB Function4
double
newPyram wanalysis1 channel1
Embedded MATLAB Function
channel1
newPyram wanalysis2 channel2
Embedded MATLAB Function2
channel2
newPyram wanalysis3 channel3
Embedded MATLAB Function3
channel3
channel1
channel2 plotsignal
channel3
Embedded MATLAB Function1

Figure 6. Simulink representation of Interface

## 2.4 Hardware Sub-system (author: R. Karkhanis)

In order to expedite the visualization process, the system was implemented using a hardware design. This optimization was done 'off-board' on the CycloneII FPGA using the DE-2 board from Altera. FPGA being a programmable platform allows the flexibility to modify code without much delay or processing time. Debugging and verification becomes easier through the use of such a re-configurable processor.

The core of the hardware subsystem was the wavelet analysis. The top level understanding of the wavelet analysis came from the software implementation. The analysis method consisted of taking the Fast Fourier Transform of the signal and convoluting it with scaled versions of the log Gabor wavelet to produce a matrix of coefficients. The greater the similarity between the transformed signal and the wavelet, the higher is the coefficient obtained at that point. The following diagram, Figure 7, details the structure of the hardware system:

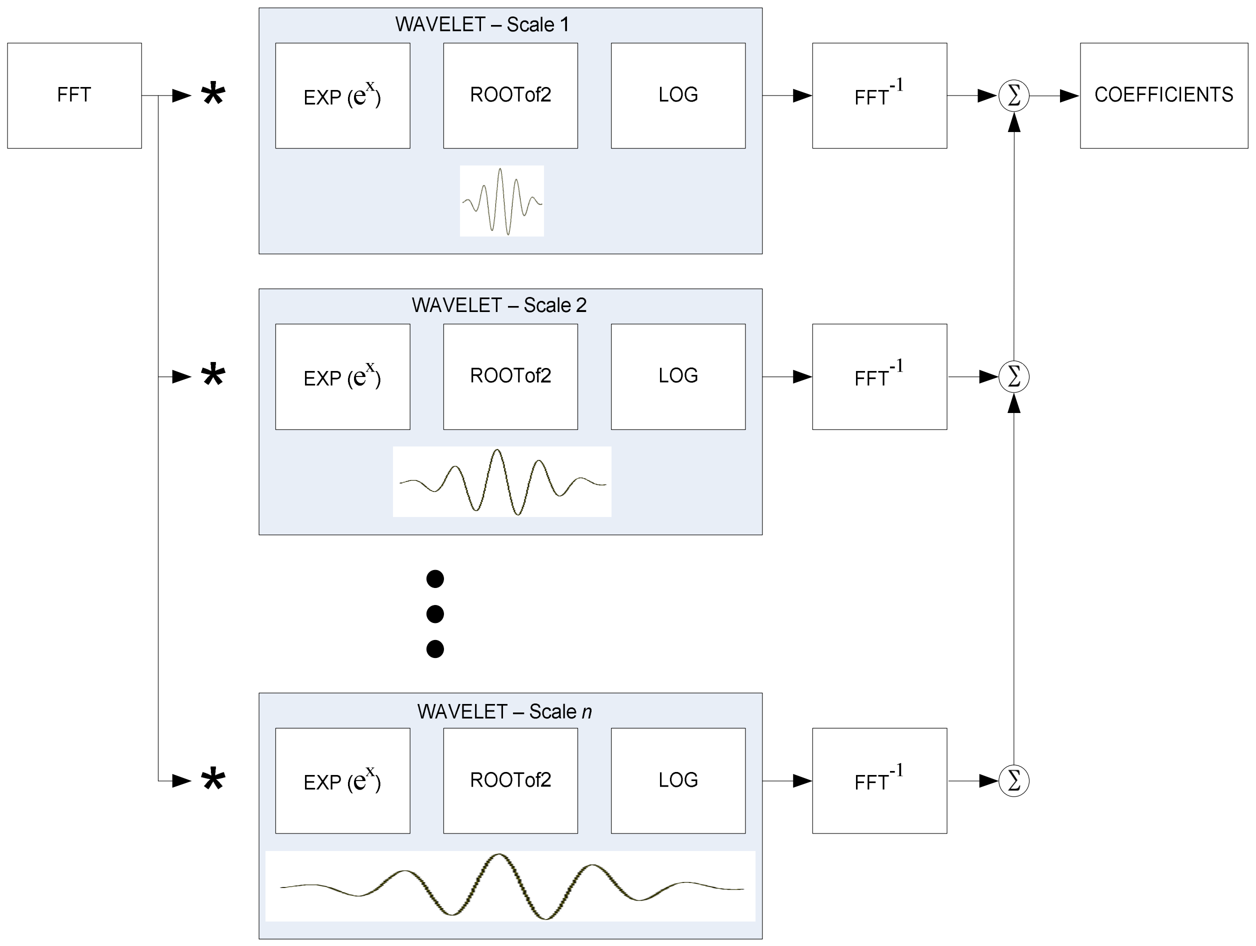


Figure 7. Hardware Implementation of Wavelet Analysis

*2.4.1 Log-Gabor Wavelet* (author D. Kostka)

To implement the log Gabor wavelet in hardware, a direct conversion of Matlab M-code was used to create Verilog code as follows.

```
for (i = 0; i < 513; i = i +1)
        %determine the radius of the Gabor wavelet
        begin
                assign radius[i] = i/ndata;
        end
        assign radius[1] = 1;
        for (i = 0; i < 513; i = i +1)
        begin
                assign radius[i] = i/ndata;
        end
end
 % create the log Gabor wavelet
for (i = 0; i < 513; i = i +1)
  begin
        assign d = radius[i]/fo;
        log stage0 (d,B[i]);
        log stage1 (sigmaOnf, m);
        assign f = (-B[i])^2 / m;
        expo stage2 (f,logGabor[i]);
  end
```

This required the coding of exponential, logarithmic and root functions as Quartus does not support built-in definitions of these. Pre-existing code was used for these functions however; modifications to the code were required in order to translate it from System Verilog to Synthesizable Verilog.

*2.4.2 Exponential Function* (author D. Kostka)

The base-two exponential (antilogarithm) function, 2x, is computed by examining the bits of the argument, and for those bits of the argument that are 1, multiplying the result by the corresponding power of a base very close to one. The encoded function assumes 23 bits of accuracy. (Refer to Appendix E)

### *2.4.3 Log Function* (author D. Kostka)

The logarithm function prints an error message for arguments less than or equal to zero because the real-valued logarithm is not defined for such arguments. The loop here requires an argument greater than or equal to one. For arguments between zero and one, this code uses the identity ln(1/x) = -ln(x). (Refer to Appendix E)

### *2.4.4 FFT Megacore Function* (author R. Karkhanis)

The FFT Megacore function is a built-in Altera DSP Builder block that creates the time-frequency spectrum of the signal. The parameterization and compilation of the FFT block were required in order for it to be implemented in a Quartus project.

## 3. **Assessment of Final Design** (author: R Karkhanis & D.Kostka)

Unfortunately due to the timing constraints and integration errors in the Verilog code, the project wasn't able to meet all the requirements set out at the beginning of the year. The following subsections will give a more thorough analysis of the level of completion reached for each of the target tasks as well as an evaluation of any design decisions involved

**3.1 Software Component** (author: D. Kostka)

The software version of the design was successfully implemented in M-code using Matlab. All design goals and requirements were met and a four dimensional visualisation of the EEG data was obtained.

A major roadblock faced during the coding stage of the software was the choice of wavelet analysis method to be implemented. Through data testing and comparison of scalograms produced by the two options, we decided to implement the DWT through an FFT using a varying log-Gabor wavelet. The log Gabor method on comparison with the DWT was slightly less precise due to the fact that the FFT performed on the signal smoothed out the curve leading to a loss of some data in the higher frequencies. However this loss was minimal and hence, this method while slightly less accurate than using DWT, would allow for easier implementation and translation of the wavelet analysis method into hardware.

**3.2 Hardware Component** (author: R. Karkhanis)

In order to optimize our design, the hardware had to meet the following requirements:

- easy to use
- easy to re-design
- inexpensive

It was determined and clarified by background research that the FPGA implementation would be more suitable for our application than a GPU implementation. Due to the delays in acquiring the hardware, testing of the hardware was delayed.

The hardware implementation of the wavelet analysis failed to compile as a whole even though the individual sub functions compiled. This could be attributed to syntax errors that arose on integration on the function.

The complexity of creating the software-hardware interface was unanticipated and hence pushed our time line over the initial plan. Ultimately, however, the interface was successfully created in Simulink by the conversion of the Matlab M-code into Embedded Matlab code.

Overall, the hardware component of the design was not successful due to un-accounted complications that arose and time constraints.

## 4. **Testing and Verification** (author: D. Kostka & R. Karkhanis)

**Requirement (# and title):**

*Task #1* - The software implementation of the visualization shall take in raw EEG data and output a four dimensional (amplitude-time-space-frequency) mapping of the EEG signal.

**Final Result:** A four dimensional visualization was successfully obtained with time as the x-axis, frequency /scale as the y-axis, space as the z-axis and power spectral strength encoded by colour.

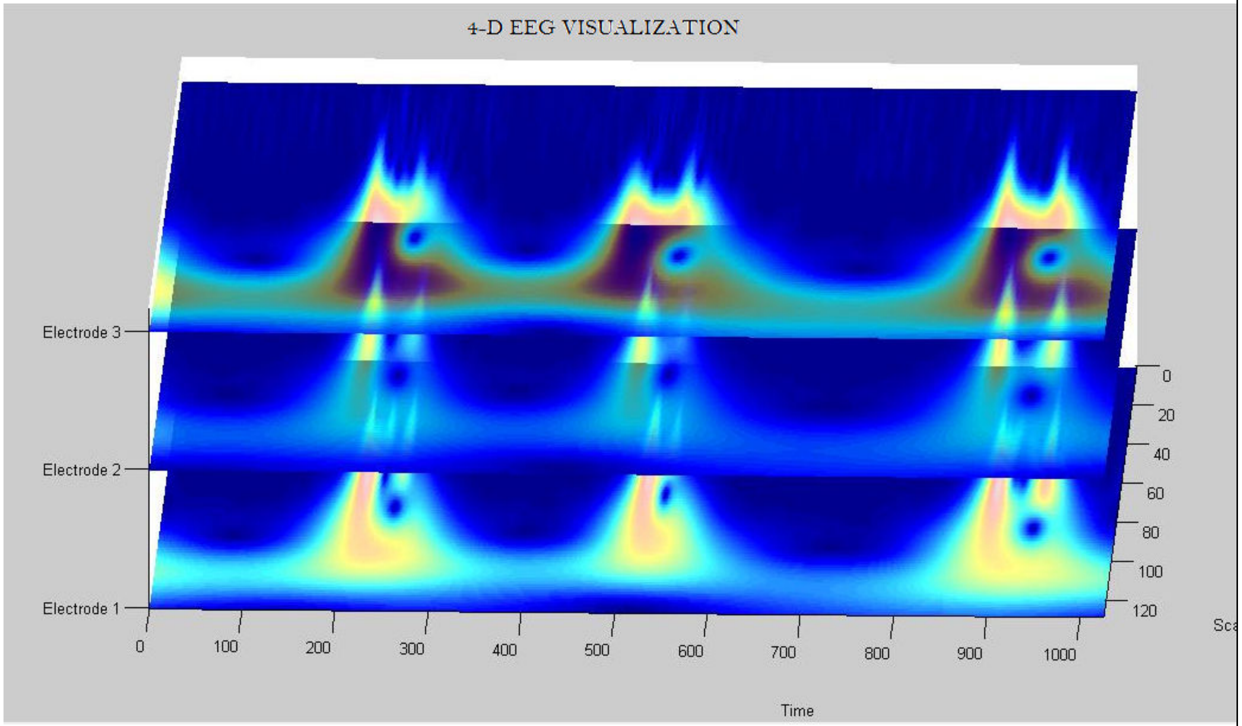
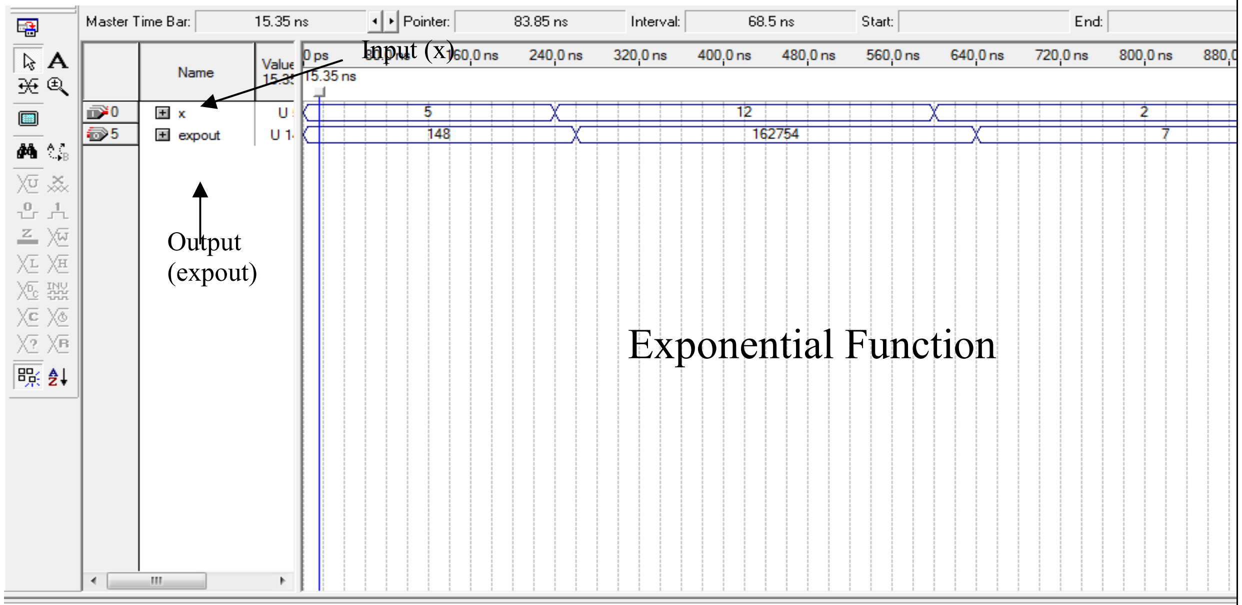
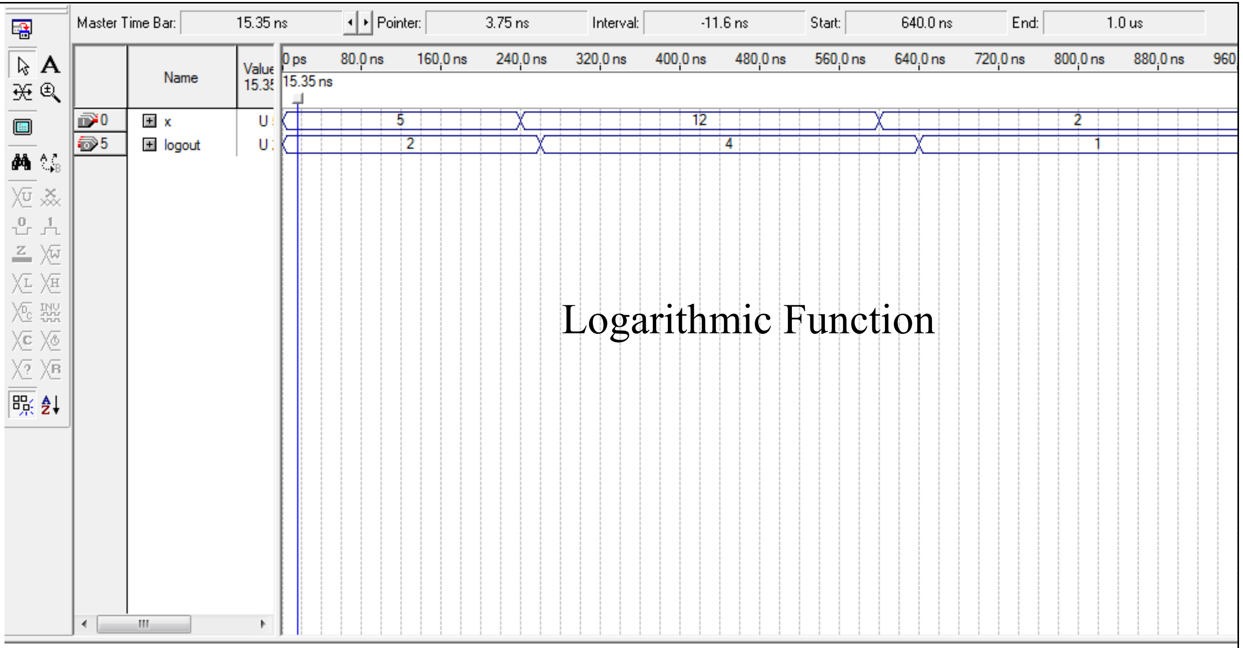


Figure 8. 4-D Visualization showing alpha mapping

**Compliance (Pass/Fail):** Pass

| |
|---|
| **Requirement (# and title):** *Task #2* - The hardware implementation of the wavelet analysis shall take in raw EEG data and output the time-frequency and power spectral components of the signal. |
| **Final Result:** The overall function failed to compile as a whole. |
| **Compliance (Pass/Fail):** Fail |
| **Comments:** Failure to compile was due to errors created during the integration of necessary sub-functions. However the exponential and logarithmic sub functions were successfully implemented as follows:   Master Time Bar: 15.35 ns<br>Pointer: 83.85 ns<br>Interval: 68.5 ns<br>Start:<br>End:<br>Name<br>Value 15.35<br>0 ps<br>160,0 ns<br>240,0 ns<br>320,0 ns<br>400,0 ns<br>480,0 ns<br>560,0 ns<br>640,0 ns<br>720,0 ns<br>800,0 ns<br>15.35 ns<br>x<br>expout<br>5<br>12<br>2<br>148<br>162754<br>7<br>Input (x)<br>Output (expout)<br>Exponential Function Figure 9. Waveform results of (i)Exponential and (ii)Logarithmic function simulation |



Logarithmic Function

**Requirement (# and title):**

*Task #3* - The hardware implementation of the signal transform shall take in a signal and output its time-frequency spectrum.

**Final Result:**

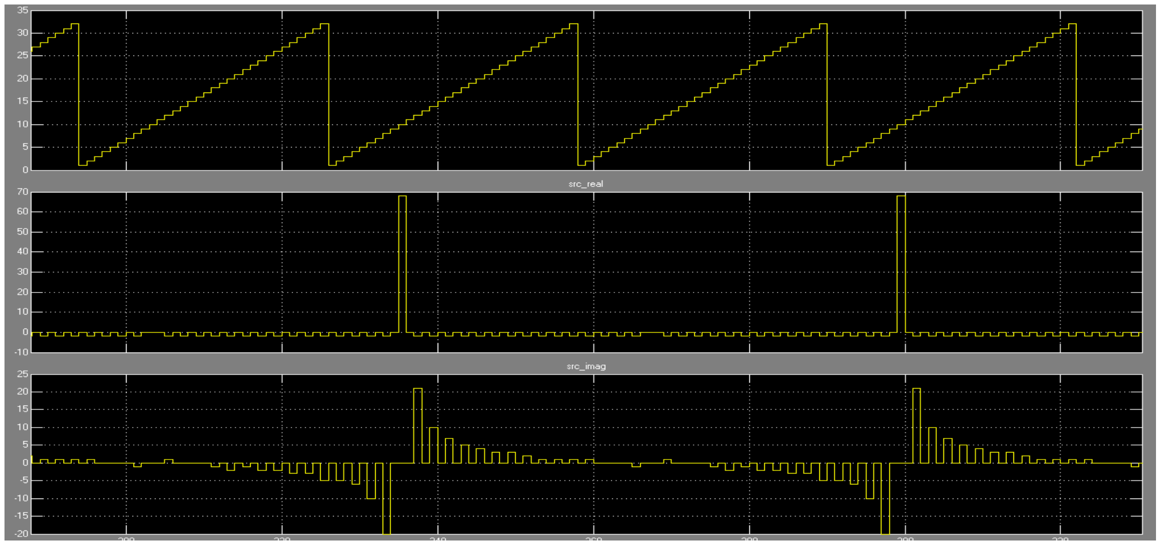
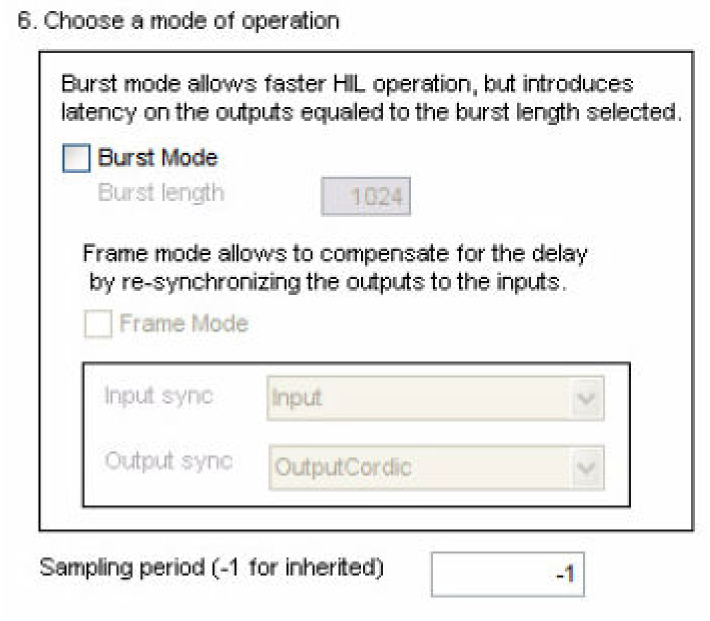


Figure 10. FFT results

| |
|---|
| The first graph shows the input of 0 and 1 connected by a ramp. The second graph produces a jump in the frequency showing the frequency component and its amplitude. The third graph shows the phase delay as the imaginary component. |
| **Compliance (Pass/Fail):** Pass |

| |
|---|
| **Requirement (# and title):** *Task #4* - The hardware implementation of the visualization shall be proved to be faster than the software implementation. |
| **Final Result:** Since the hardware implementation wasn't completed in time, we were not able to prove this. |
| **Compliance (Pass/Fail):** Fail |
| **Comments:** Demonstrating the speed of the hardware would indicate that the optimization of our design could indeed be done. Through the use of a tutorial in DSP Builder User Guide, we synthesized a design on the FPGA. In performing the synthesis of the design on the FPGA using HIL, some options must be selected. Through testing it was discovered that \`Burst Mode\`, showed an increase in performance. In the test that we ran we could see a 10 fold decrease in processing time. This test proved the concept that an FPGA design works faster than a software design. |


6. Choose a mode of operation

Burst mode allows faster HIL operation, but introduces latency on the outputs equaled to the burst length selected.

[ ] Burst Mode

Burst length: 1024

Frame mode allows to compensate for the delay by re-synchronizing the outputs to the inputs.

[ ] Frame Mode

Input sync: Input

Output sync: OutputCordic

Sampling period (-1 for inherited): -1

Figure 11. Frame mode vs. Burst Mode

The above is a screenshot of the section of the HIL that had to be selected in order to control the way the input and output is handled by the FPGA.

| **Requirement (# and title):** *Task #5* - The EEG signals shall be processed in order to extract the power spectral, temporal and frequency components. |
|---|
| **Final Result:** The wavelet analysis algorithm was successfully implemented using the log Gabor filter. Wavelet coefficients produced determined the level of similarity between the Gabor wavelet and the EEG signal at a particular time and frequency for a point in space. The coefficients are captured in the matrix channel.cfd() The scalogram produced for a single channel is as shown below. |

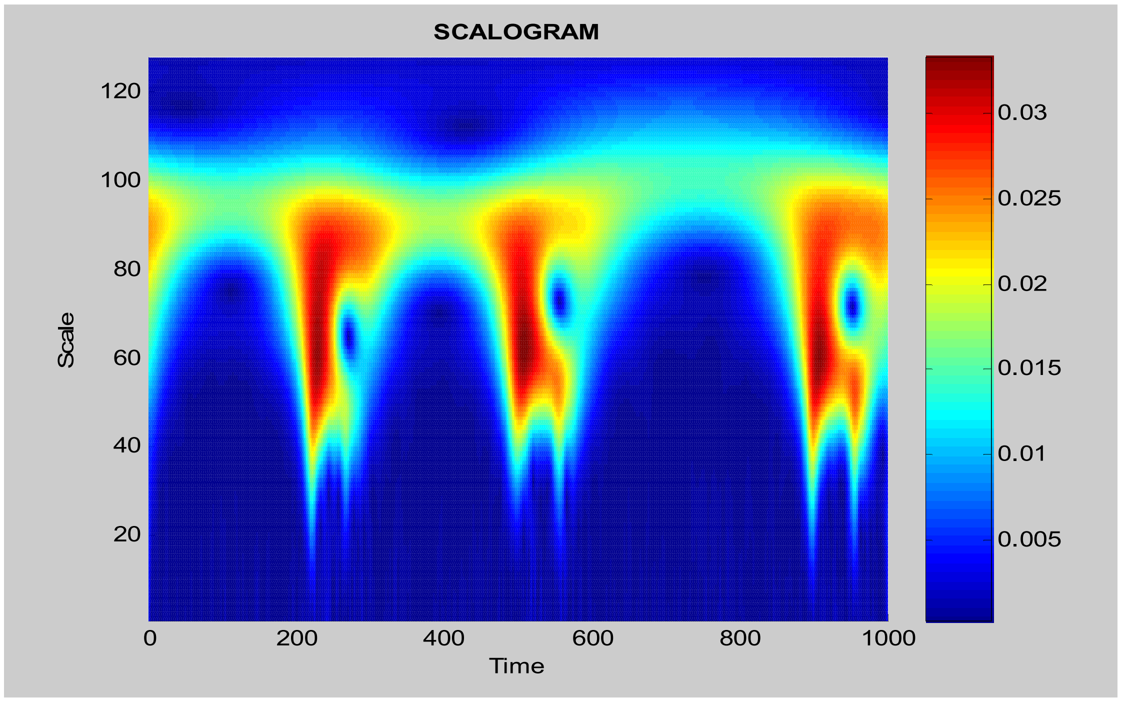


Figure 12. Scalogram produced by one electrode

**Compliance (Pass/Fail):** Pass

**Comments:** The coefficients obtained using this analysis method were compared to those obtained using the built-in Matlab DWT function to access degree of similarity. The coefficients were found to be successfully computed.

**Requirement (# and title):**

*Task #6* - The frequency-time-spectral information from each EEG electrode shall be stacked to add the spatial component to the final visualization.

**Final Result:** The frequency-time and spectral information from each electrode is plotted as a surface at different elevations in the z-axis as follows:

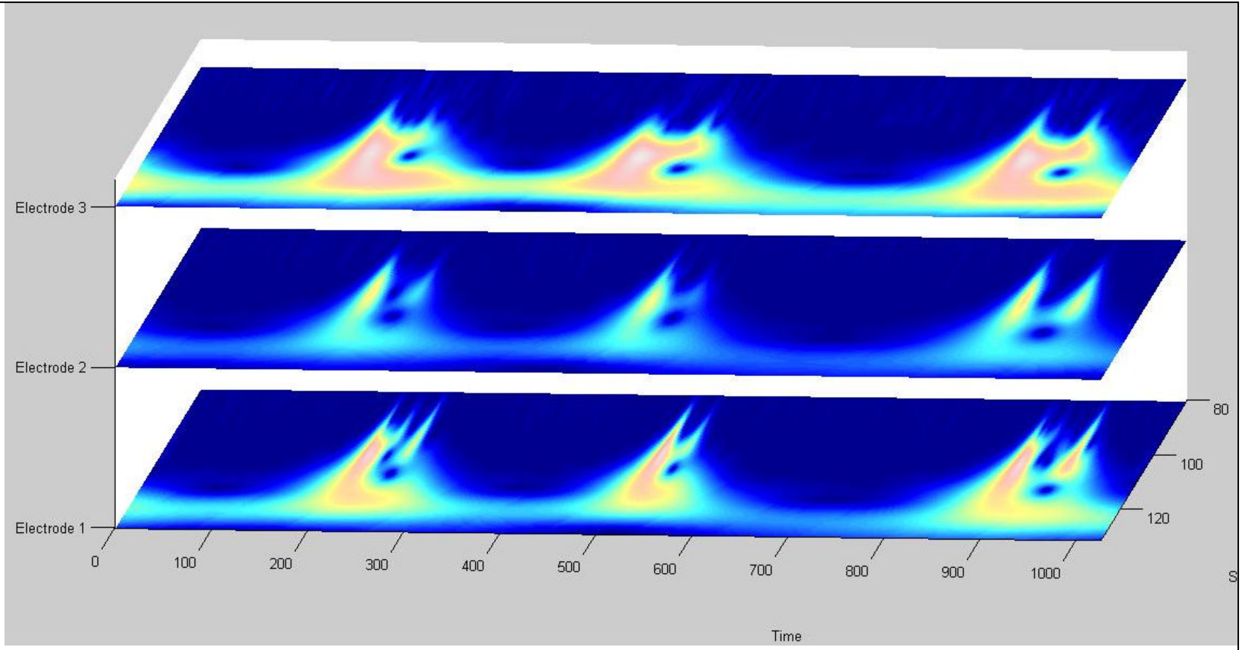


Figure 13. Stacking scalograms to produce visualisation

**Compliance (Pass/Fail):** Pass

**Requirement (# and title):**

*Task #7* - The hardware and software components should integrate.

**Final Result:** Due to failure in implementing and integrating the hardware component into the system, this test could not be run. However the HIL, HDL and Embedded Matlab components of the interface were designed as compliant.

**Compliance (Pass/Fail):** Fail

**Comments:** The complete integration between the hardware and software components required the implementation of the following steps:

<u>Hardware in the Loop</u>

Hardware in the Loop was tested through the use of a tutorial found in DSP Builder User Guide [1]. The synthesized design was then simulated through dual operations of the computer and the FPGA. The following is a Simulink picture of the design. The test was performed on a frequency sweep design.

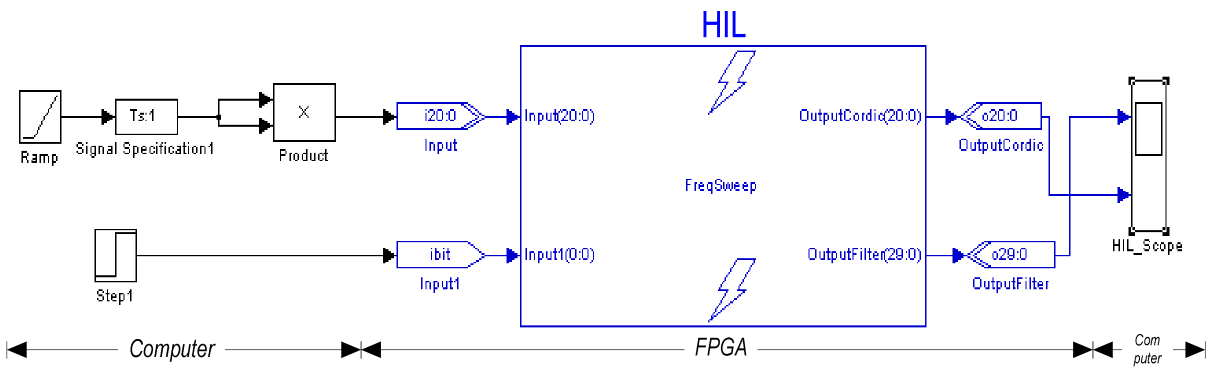


Figure 14. Frequency sweep HIL implementation

Observe that the first set of Computer blocks are Simulink blocks from the blockset library that performs the simulation on the computer. The HIL block is the one that has been synthesized in hardware along with the hardware bit stream interface marked by FPGA. The output is then sent to the scope, which again is present on the computer. The output of this test displayed (shown below) the same simulation results as the one done just by computer.

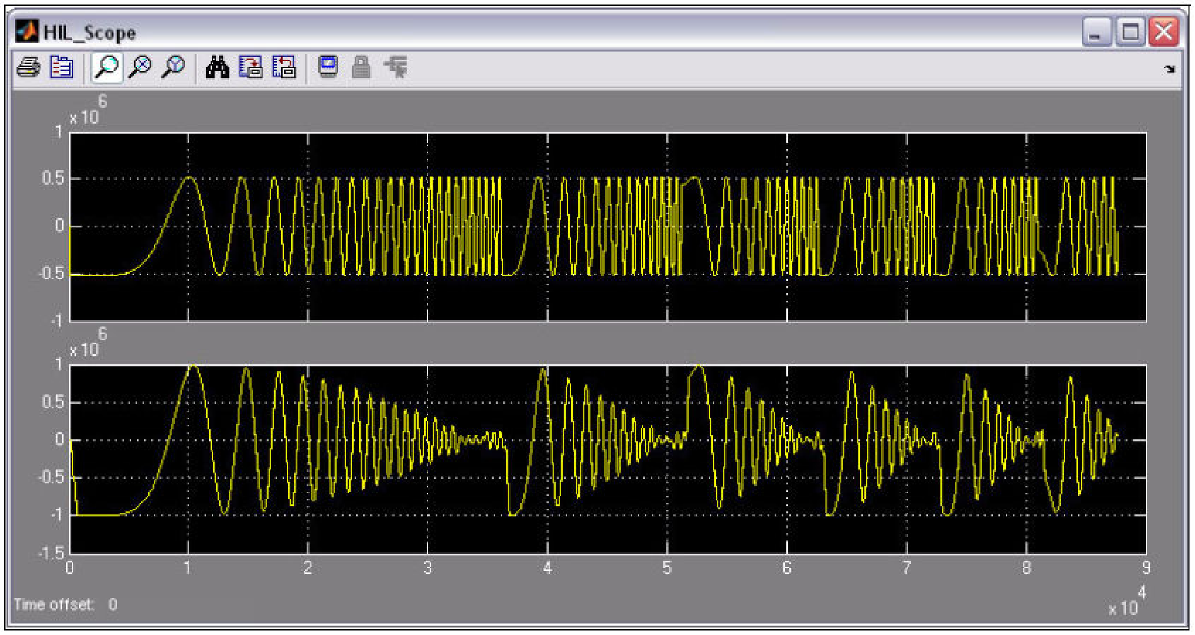


Figure 15. Frequency sweep output

Performing this test was a major milestone that showed that the interface would work between Simulink blocks and the FPGA. This model was further used in testing the speed of the hardware (more on this later on).

<u>HDL Import test</u>

Hardware in the Loop was tested through the use of a tutorial found in DSP Builder User Guide. Before starting to code in Verilog, we had to verify that we could use the Verilog code in Simulink. The ROOTof2 function from Verilog was imported into the Simulink model.

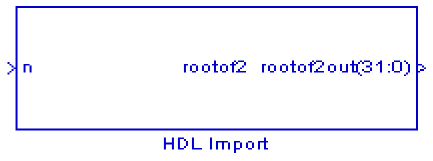


Fig 16. HDL Import Test

This block is now treated as a black box by the Signal compiler as described in the

System Design.

The complete test included a run through from the tutorial and then running the simulation in Simulink. The result showed that Verilog code can indeed be incorporated into Simulink. Simulink does not require Verilog to be designed in any particular way, which makes the import easier.

Embedded Matlab

The Matlab M-code was successfully translated into Embedded Matlab code and the four dimensional visualization was produced. This visualization was compared to the one produced by the software implementation and was found to be accurate.

## 5. Summary and Conclusion

The four dimensional visualization was successfully executed in software using Matlab. However, the hardware component was not completed due to time delays and complications that arose during integration of the code. As demonstrated by our validation and acceptance tests, the four dimensional mapping of the EEG signal was successful in representing the power spectral-time-frequency-space components.

The design project opened us to the world of signal processing and Matlab and through it, we have learnt and gained a number of tools for signal analysis and modification.

The choice of implementation method for wavelet analysis was evaluated in terms of accuracy and ease of implementation in hardware. A scalable log Gabor wavelet was convoluted with the FFT of the signal in order to reproduce the DWT. Overall, this choice made the project a better learning process since it required a deeper understanding of the wavelet transform in order to design for it, as opposed to using the built-in wavelet analysis functions.

Furthermore, dividing the project into a number of sub functions and modules, helped in the testing and evaluation of each step of the design.

A prime application of the 4-D EEG visualization is to serve as a visual tool for researchers to help identify features associated with seizure activity or other abnormal electric brain activity. The graphical representation of stacked plots employed in the design allows researchers to easily compare and track frequency activity from different electrodes as it travels in time and space.

Future work would entail the completion of the hardware implementation of the wavelet analysis so as to optimize the visualization. Additionally, by implementing the visualization itself on hardware it would greatly decrease the processing time of the design. In essence, the optimal design would be involve the processing of the EEG signals in real-time.

## References

[1]: L. Senhadji, J-L. Dillenseger, F. Wendling, C. Rocha, and A. Kinie, "Wavelet analysis of EEG for three-dimensional mapping of epileptic events", France: HAL Author Manuscripts, Sep–Oct 1995

[2]: M. ten Caat, N. M. Maurtis, and J. B. T. M. Roerdink, "Tiled Parallel Coordinates for the Visualization of Time-Varying Multichannel EEG Data", IEEE VGTC Symposium on Visualization, Netherlands, 2005

[3]: Morgan Kauffman , "Information Visualization: Perception.", Ware, C. (2000).

[4]: Robi Polikar , "The Engineers Ultimate Guide to the Wavelet Tutorial."

[5]: B. Cope, "Implementation of 2D Convolution on FPGA, GPU and CPU", Imperial College of London, 2004

[6]: Kevin W. Bowyer, Kyong Chang, and Patrick Flynn, "A survey of approaches and challenges
in 3D and multi-modal 3D + 2D face recognition," *Computer Vision and Image Understanding*,
vol. 101, no. 1, pp. 1–15, 2006, TY - JOUR.

[7]: "DSP Builder User Guide," Altera Corporation. Online: http://www.altera.com/literature/ug/ug_dsp_builder.pdf. Nov., 2008. [Accessed 2009-01-04]

[8]: Mathworks:
http://www.mathworks.com/products/featured/embeddedMatlab/capabilities.html,
"Using Embedded Matlab Code," Mathworks, Boston, 2009

## APPENDIX A: Gantt charts

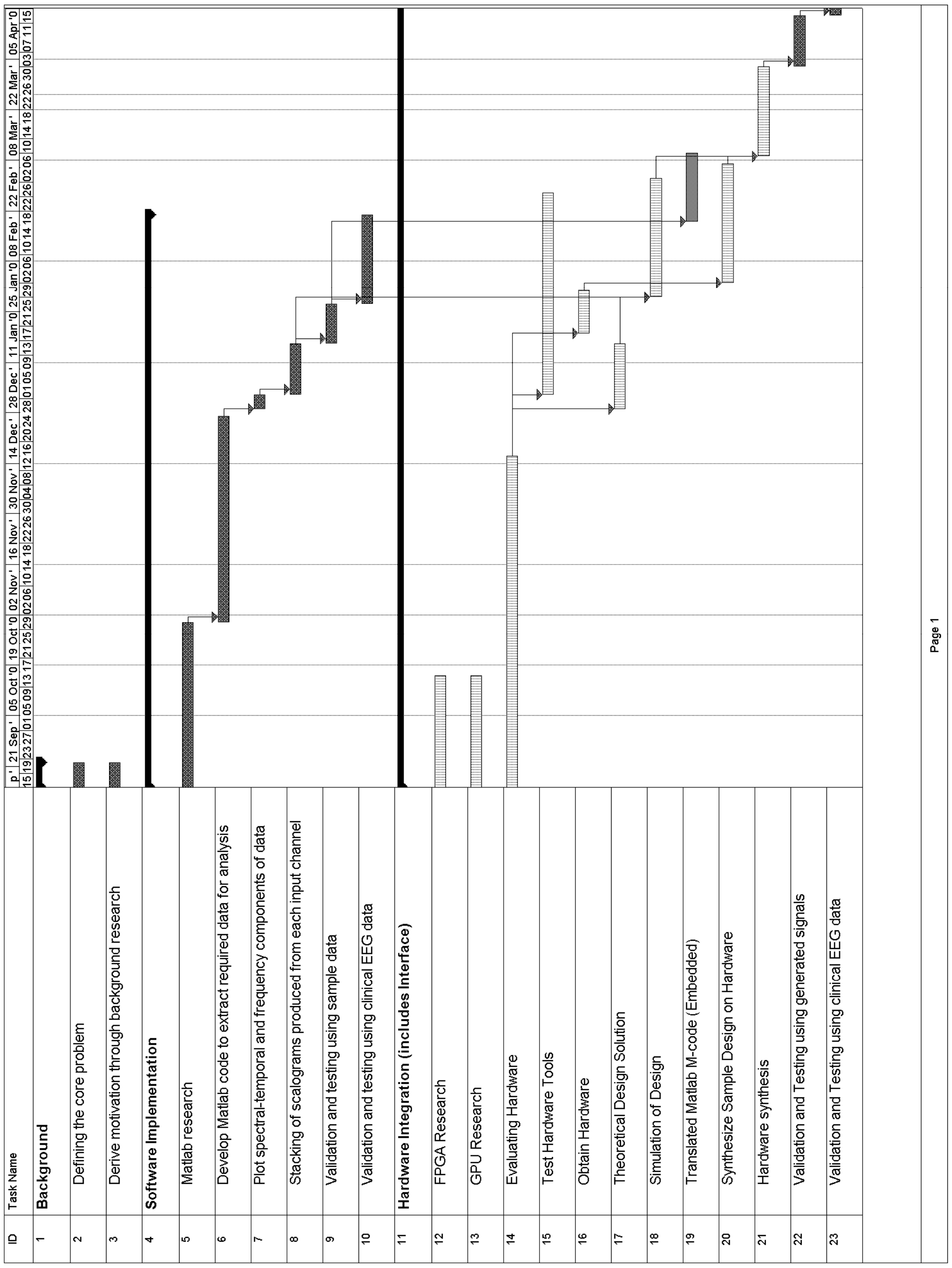
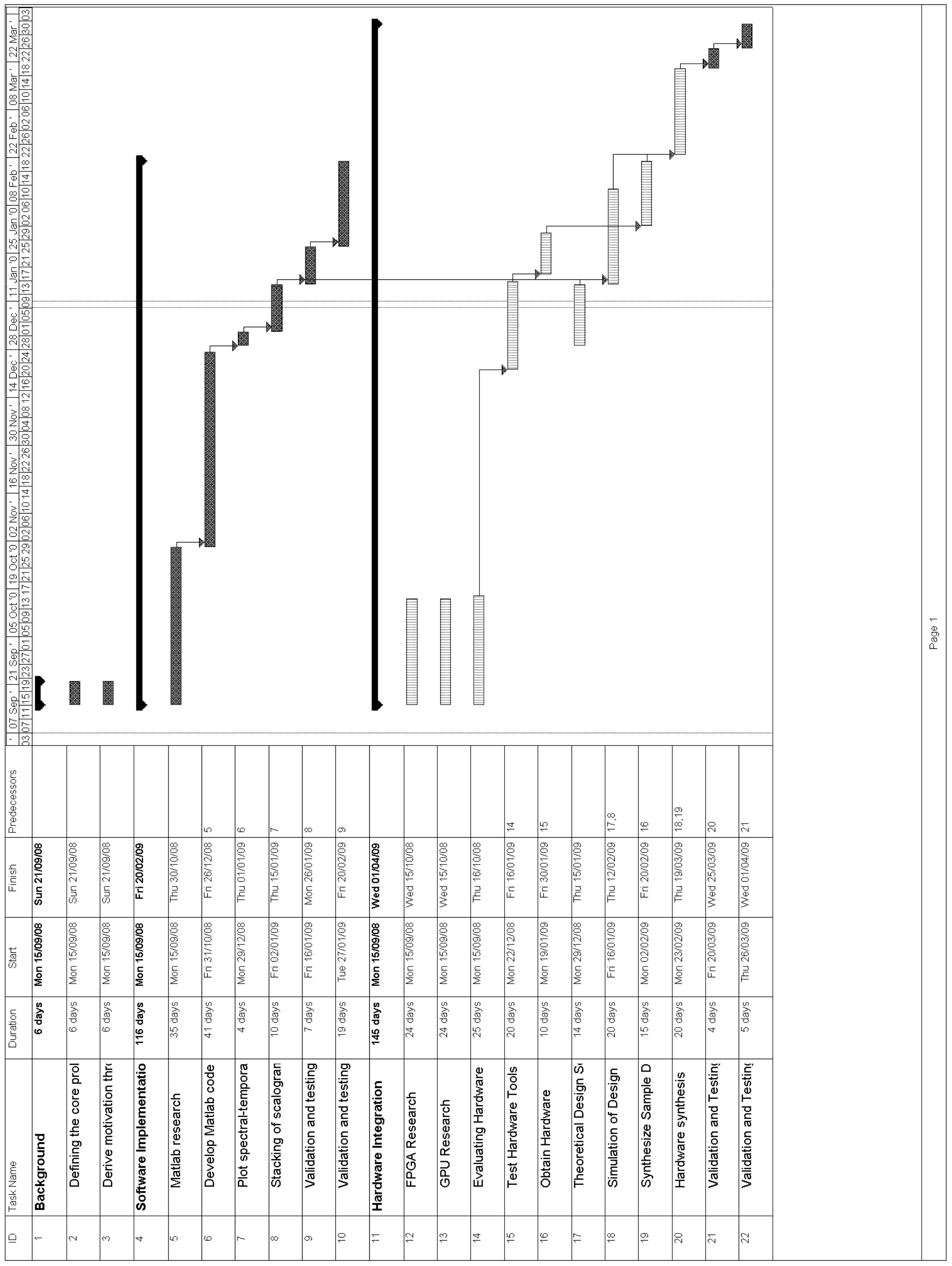


**Figure 1. Gantt chart – FINAL – Rotated to Fit Page – Diane (in dark shade), Rohan (in light shade)**

The Gantt chart contains one more added task (#19). The rest of the tasks stayed the same, but their depth and level of involvement increased, causing delays in our projected deadlines.

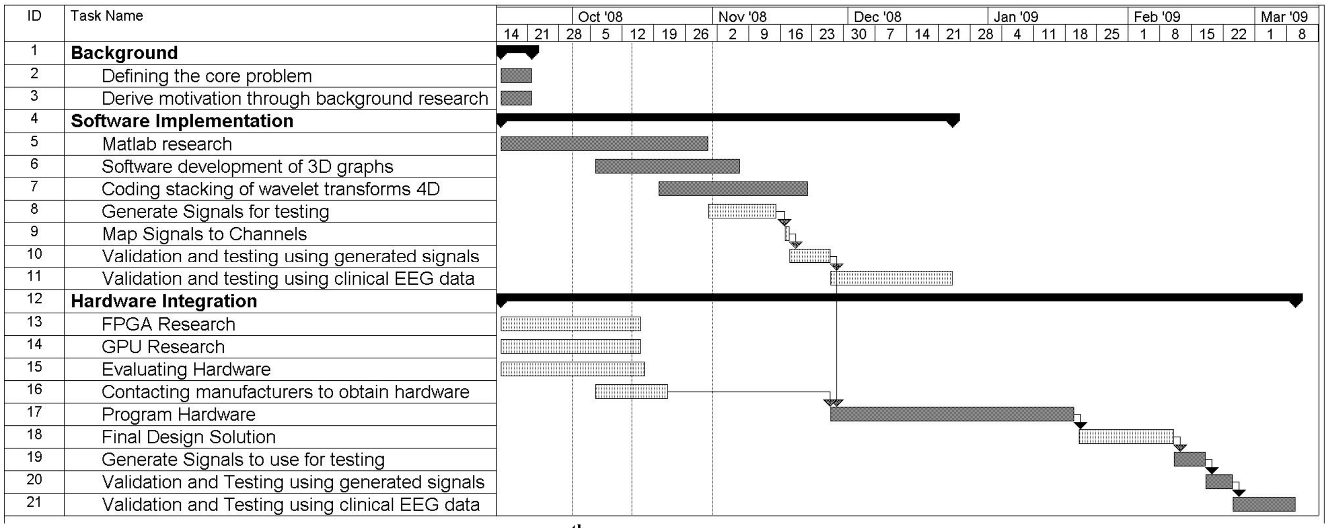
Individual Progress Report – Gantt chart



| ID | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 1 | **Background** | **6 days** | **Mon 15/09/08** | **Sun 21/09/08** | |
| 2 | Defining the core prol | 6 days | Mon 15/09/08 | Sun 21/09/08 | |
| 3 | Derive motivation thr | 6 days | Mon 15/09/08 | Sun 21/09/08 | |
| 4 | **Software Implementatio** | **116 days** | **Mon 15/09/08** | **Fri 20/02/09** | |
| 5 | Matlab research | 35 days | Mon 15/09/08 | Thu 30/10/08 | |
| 6 | Develop Matlab code | 41 days | Fri 31/10/08 | Fri 26/12/08 | 5 |
| 7 | Plot spectral-tempora | 4 days | Mon 29/12/08 | Thu 01/01/09 | 6 |
| 8 | Stacking of scalogran | 10 days | Fri 02/01/09 | Thu 15/01/09 | 7 |
| 9 | Validation and testing | 7 days | Fri 16/01/09 | Mon 26/01/09 | 8 |
| 10 | Validation and testing | 19 days | Tue 27/01/09 | Fri 20/02/09 | 9 |
| 11 | **Hardware Integration** | **145 days** | **Mon 15/09/08** | **Wed 01/04/09** | |
| 12 | FPGA Research | 24 days | Mon 15/09/08 | Wed 15/10/08 | |
| 13 | GPU Research | 24 days | Mon 15/09/08 | Wed 15/10/08 | |
| 14 | Evaluating Hardware | 25 days | Mon 15/09/08 | Thu 16/10/08 | |
| 15 | Test Hardware Tools | 20 days | Mon 22/12/08 | Fri 16/01/09 | 14 |
| 16 | Obtain Hardware | 10 days | Mon 19/01/09 | Fri 30/01/09 | 15 |
| 17 | Theoretical Design S | 14 days | Mon 29/12/08 | Thu 15/01/09 | |
| 18 | Simulation of Design | 20 days | Fri 16/01/09 | Thu 12/02/09 | 17,8 |
| 19 | Synthesize Sample D | 15 days | Mon 02/02/09 | Fri 20/02/09 | 16 |
| 20 | Hardware synthesis | 20 days | Mon 23/02/09 | Thu 19/03/09 | 18,19 |
| 21 | Validation and Testin | 4 days | Fri 20/03/09 | Wed 25/03/09 | 20 |
| 22 | Validation and Testin | 5 days | Thu 26/03/09 | Wed 01/04/09 | 21 |

**Figure 2. Gantt Chart Rotated to Fit Page – Diane (in dark shade), Rohan (in light shade)**

## Proposal – Gantt chart



| ID | Task Name |
|---|---|
| 1 | **Background** |
| 2 | Defining the core problem |
| 3 | Derive motivation through background research |
| 4 | **Software Implementation** |
| 5 | Matlab research |
| 6 | Software development of 3D graphs |
| 7 | Coding stacking of wavelet transforms 4D |
| 8 | Generate Signals for testing |
| 9 | Map Signals to Channels |
| 10 | Validation and testing using generated signals |
| 11 | Validation and testing using clinical EEG data |
| 12 | **Hardware Integration** |
| 13 | FPGA Research |
| 14 | GPU Research |
| 15 | Evaluating Hardware |
| 16 | Contacting manufacturers to obtain hardware |
| 17 | Program Hardware |
| 18 | Final Design Solution |
| 19 | Generate Signals to use for testing |
| 20 | Validation and Testing using generated signals |
| 21 | Validation and Testing using clinical EEG data |

**Figure 3. Gantt chart from Proposal – October 16th 2008**

## APPENDIX B: Testing and Validation from Proposal

Validation and Acceptance Testing

- Initially, computer generated signals with time varying frequencies will be used as input to verify the software and hardware implementation of the visualization. A four dimensional mapping of the signals should be produced representing their power spectral-time-frequency-space components.
- Finally, pre-recorded EEG signals will be used as input to verify the software and hardware implementation of the visualization.

## APPENDIX C

```
function channel = wanalysis( Pyram, tstart, tint, tstop)
%This function takes in the EEG data and using the log Gabor function for wavelet analysis,
%computes the amplitude spectrum of the signal for each channel.
format compact;
clock1 = clock;
Pyram = Pyram (tstart:tint:tstop,:);
% Get sizes of data to be analysed
sizexy = size(Pyram);
maxchannels = sizexy(1,2); % maxchannels - the columns in the input data corresponding
% ...to electrode readings
for i = 2 : maxchannels-1
        channel(i).data = Pyram(:,i)';
        numcoefs = 9;
        sigmaOnf = .55; % sigmaOnf - Shape factor of log Gabor filter

        % ...controlling bandwidth: .55 - bandwidth of approx 2 ocatves
        mult = 1.05; % mult - scaling factor between successive filters
        nscales = 128; % nscales - No of filtering scales to use
        wavelength = 4; % minwavelength - wavelength of smallest scale filter to use
        lv = length(channel(i).data);

        if mod(lv,2) == 1 % If there is an odd No of data points
                ndata = lv-1; % throw away the last one.
                channel(i).data = channel(i).data(1:ndata);
        end
        signalfft = fft(channel(i).data); % Take FFT of signal
        channel(i).amplitude = zeros(nscales,ndata); % Pre-allocate memory for speed
        channel(i).phase = zeros(nscales,ndata);
        logGabor = zeros(1,ndata);
        EO = zeros(1,ndata);
        radius = [0:fix(ndata/2)]/fix(ndata/2)/2; % Frequency values 0 - 0.5
        radius(1) = 1;
        for row = 1:nscales
                fo = 1.0/wavelength; % Centre frequency of filter.
                 % Construct log Gabor filter
                 logGabor(1:ndata/2+1) = exp((-(log(radius/fo)).^2) / (2 * log(sigmaOnf)^2));
                 logGabor(1) = 0; % Set value at zero frequency to 0
                 % Multiply filter and FFT of signal, then take inverse FFT.
                 EO = ifft(signalfft .* logGabor);
                 channel(i).amplitude(row,:) = abs(EO); % Record the amplitude of the result
                 channel(i).phase(row,:) = angle(EO); % .. and the phase.
                 wavelength = wavelength * mult; % Increment the filter wavelength.
        end
       channel(i).x_axis = 1:size(channel(i).amplitude,2)
       channel(i).y_axis = 1:size(channel(i).amplitude,1)
end
```

## APPENDIX D

The following snippet of code from the function plotsginal() shows how the data is plotted using the Matlab define ‘surface’ function.

```
%surface() is the low-level function for creating surface graphics objects. %Surfaces are plots of
matrix data created using the row and column indices of %each element as the x- and y-
coordinates and the value of each element as the %z-coordinate. surface(X,Y,Z,C) plots the
parametric surface specified by X, Y, %and Z, with color specified by C plot scalograms and stack
scalograms from %different channels
        k=2
        handles.a=surface(channel(k).x_axis,channel(k).y_axis,3*(k-
        2)*ones(sizeimg(k,1),sizeimg(k,2)),channel(k).amplitude,'EdgeColor','none');
        hold on
        k=3
        handles.b=surface(channel(k).x_axis,channel(k).y_axis,3*(k-
        2)*ones(sizeimg(k,1),sizeimg(k,2)),channel(k).amplitude,'EdgeColor','none');
        k=4
        handles.c=surface(channel(k).x_axis,channel(k).y_axis,3*(k-
        2)*ones(sizeimg(k,1),sizeimg(k,2)),channel(k).amplitude,'EdgeColor','none');
        k=5
        handles.d=surface(channel(k).x_axis,channel(k).y_axis,3*(k-
        2)*ones(sizeimg(k,1),sizeimg(k,2)),channel(k).amplitude,'EdgeColor','none');
        handles.e=surface(x,y,z5)
        hold off
        %define transparency parameters using alpha blending
        alpha('color')
        alphamap('rampdown') % Create a linear alphamap with decreasing opacity
        alphamap('increase',.1) % Modify the alphamap making it more opaque (default delta
        % is .1, which is added to the current values).
```

## APPENDIX E

Logarithminc Function

```
module log(x,logout);
input x;
reg re,log2;
output logout;
reg logout;
integer i;

always @(x or logout or re or log2)
begin
  if (x = 0.0)
  begin
     $display("log illegal argument:");
     $stop;
     assign logout = 0;
  end
  else
  begin
     if (x<1.0)
        assign re = 1.0/x;
     else
        assign re = x;
     assign log2 = 0.0;
     for (i=7; i>=-23; i=i-1)
        begin
           if (re > rootof2(i))
           begin
                assign re = re/rootof2(i);
                assign log2 = 2.0*log2 + 1.0;
           end
           else
                assign log2 = log2*2;
         end
         if (x < 1.0)
             assign logout = -log2/12102203.16;
         else
         assign logout = log2/12102203.16;
  end
end
```

## Exponential Function

```
module exp(expout);
parameter x=23 ;
reg x1,power,prod;
output expout;
reg expout;
integer i;
reg inter;
reg power2;
integer m;

always @(x1 or x or prod or power or expout or i or inter)
begin
    x1 = (x)*1.44269504;
    if (x1 > 255.0)
    begin
            expout = 0.0;
            if (x>0.0)
            begin
                  $display("exp illegal argument:");
                  $stop;
            end
    end
else
begin
    prod = 1.0;
    power = 128.0;
   for (i=7; i>=-23; i=i-1)
   begin
            if (x1 > power)
            begin
                 prod = prod * root2(i);
                 x1 = x1 - power;
            end
            power = power / 2.0;
   end
   if (x < 0)
            expout = 1.0/prod;
  else
             expout = prod;
end
endmodule
```