# UNIVERSITY OF ALASKA ANCHORAGE

CSCE A470

CAPSTONE PROJECT

# Digital Headend Media Decryption


Author:

**Robert Aschenbrenner**

Supervisor:

**Prof. James "Randy" Moulic, PhD**


Anchorage AK, April 2016


Computer Science & Engineering Department
UNIVERSITY of ALASKA ANCHORAGE

Robert Aschenbrenner

raschenb@alaska.edu

Version 2.7

# Abstract

The aim of this applied software development capstone is produce a new hardware and software design for cable companies during the receive process of the headend band. The application is intended to be used in routine signal operations on a daily basis, where it will assist the cable company's engineering department.

The developed application presents the user with a mosaic comprised of individual titles showing the spectral content of waveforms at the signal. Each title can be viewed individually with higher resolution to have a better view of a shorter time span.

This combination of views permits an analyst or engineer to quickly scan an entire day, week, or month of data, while still allowing a closer look at specific events of interest.

# Acknowledgements

This capstone would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I am highly indebted to Dr. James Moulic, PhD. for his guidance and constant supervision through this project and Dr. Adriano Cavalcanti, PhD. for keeping me on track during Capstone and for pushing me to finish my education.

Several organizations helped with hardware and testing of the Headend software process, without their assistance and insightful feedback, Digital Headend software could not have been successful. I would like to thank Linda Pruett from Tesra Corporation, Altera University Program, Altera Japan Corporation, General Communications Inc., and David Geaous from the Khronos group for his guidance and direction on understanding the changes to OPENCL C++14 guidelines.

Most importantly, I want to thank my capstone partners Kyle and Paul, without their help and long hours, this project would not have been possible.

# Table of Contents

**1 Introduction** ..... **1**
1.1 Introduction ..... 1
1.2 Applications ..... 2
1.3 Functional Specifications ..... 4
1.4 Project development requirements ..... 4

**2 System Integration and Modeling** ..... **7**
2.1 Technology ..... 7
2.2 Design ..... 8
2.2.1 Top-down Design ..... 8
2.2.2 Bottom-up Design ..... 9
2.3 Components Used in Development ..... 10
2.4 Coding Methodology ..... 11
2.4.1 Agile ..... 11

**3 User Interface and Testing Methodology** ..... **12**
3.1 User Interface ..... 12
3.1.1 Configuration ..... 12
3.1.2 Key Display and Master Key ..... 13
3.2 Testing Methodology ..... 15
3.2.1 Correctness Testing ..... 16
3.2.2 Performance Testing ..... 16
3.2.3 Reliability Testing ..... 17
3.2.4 Security Testing ..... 17
3.3 Agile Project Management ..... 18

**4 User Manual** ..... **19**
4.1 Introduction ..... 19
4.2 Configuration ..... 19
4.2.1 Configuration HTML format ..... 20
4.2.2 System Information ..... 21
4.2.3 IP output Configuration Window ..... 23

**5 Conclusion** ..... **26**
5.1 Summary ..... 26
5.2 Future Development ..... 26

**UML** ..... **28**
**Software License BSD-2** ..... **29**
**Source Code (Backend)** ..... **33**

# List of Figures

Figure 1.1: The Denali Tower, North. General Communications Inc., Anchorage, Alaska. ..... 2
Figure 1.2: Cisco Systems© 2006, Signal distribution. ..... 3
Figure 1.3: Analog QAM: measured PAL color bar signal on a vector analyzer screen. ..... 4
Figure 1.4: Method and apparatus for partial response demodulation, Leo Montreuil. ..... 5
Figure 1.5: : System to deliver encrypted access control information to support interoperability between digital information processing/control equipment ..... 6
Figure 2.1: Shows OPENCL Flow chart. ..... 8
Figure 2.2: Headend data flow using Altera V Soc Board. ..... 8
Figure 2.3: An example of threads created with 4 connections to Alter V Soc Board. ..... 9
Figure 2.4: Project Schedule. ..... 10
Figure 2.5: Agile lifecycle. Google Images. ..... 11
Figure 3.1: Configuration tree for Software. ..... 13
Figure 3.2: Power and multiple signals using Altera Quatuars II Software. ..... 15
Figure 3.3: Single Signal using Logic Analyzer. ..... 15
Figure 3.4: Satellite Signal with the range of 1Ghz~2Ghz. ..... 18
Figure 3.5: Satellite Y(x) and Y(y) Signal. ..... 18
Figure 4.1: Main unit connected to Subunit. ..... 21
Figure 4.2: HTML Main System Information. ..... 22
Figure 4.3: System Information. ..... 23
Figure 4.4: System Information Continued Bottom Half. ..... 24
Figure 4.5: Main Output. ..... 25
Figure 4.6: IP and Port Configuration. ..... 25
Figure 4.7: Output Services. ..... 26
Figure 4.8: Output Services list. ..... 26
Figure 4.9: Output Services Select Screen. ..... 27
Figure 5.1: HTML 5 Logo ..... 27

# Chapter 1
# Introduction

## 1.1 Introduction

Cable company's equipment has always been behind technology. Our proposal is to combine the varies equipment process's into what we hope to be a single receiver that will be able to combine the multi-lateral pieces currently in use now, in the process, be allowed to receive the L band signal and decrypt the signal using the standard keys of Biss, FTA (no keys), and PowerVu (key on chip) to forward the signal to the next part of the chain which would be also internal. Breaking up the signals and recombining them into the video industrial standard of MPEG (Moving Picture Experts Group) and audio format of MPEG using MP2 and MP4 layered formats. As the signal is send out to the cable company's network, re-encrypt the signal following the standard format based on PGP (Pretty Good Protocol) and use the QAM (Quadrature Amplitude Modulation) encryption to allow the customers receivers to be able to decrypt the signal and watch their favorite show/channel.

Our project is to take an encrypted L Band signal (1GHZ ~2GHZ) and split it into two quadrature components (L1 and L2 respectively). The L1 and L2 signals are separately and independently of each RF signal and each quadrature component the four down-converted signals are counter-rotated with a respective model phase, correlated with a respective OPENCL C++14/17 code, and then successively summed and dumped over presume intervals substantially coincident with chips of the respective encryption code.

Without knowledge of the encryption-code signs, the effect of encryption-code sign flips is then substantially reduced by selected combinations of the resulting presumes between associated quadrature components for each RF signal, separately and independently for the L1 and L2 signals. The resulting combined presumes are then summed and dumped over longer intervals and further processed to extract amplitude, phase and delay for each RF signal. Precision of the resulting phase and delay values should be close to approximately four times better than that obtained from straight cross-correlation of L1 and L2. This improved method provides the following options: separate and independent tracking of the L1-Y and L2-Y channels; separate and independent measurement of amplitude, phase and delay L1-Y channel; and removal of the half-cycle ambiguity in L1-Y and L2-Y carrier phase.

*Figure 1.1: The Denali Tower, North. General Communications Inc., Anchorage, Alaska.*

## 1.2 Application

With the advent of more advanced FPGAs and embedded processors on those FPGAs the bandwidth requirement to handle the virtualization is now possible and give cable providers the tools to decrease their energy usage and infrastructure requirements by decreasing the number of components needed. After having a meeting with the engineers at GCI, our group believes we have enough knowledge and background to move forward with the project.

We plan on using FPGAs to virtualize a portion of the cable delivery process. FPGAs give us the ability to choose multiple carrier waves and channel selections on those carrier waves by building the receiver unit virtually that is currently being performed on dedicated physical receivers for each channel. The transcoding and encoding of the video signal to meet the distribution requirements will also be included in the project. The final step of recombining into one IP stream will also be handled on the FPGA. We will not work going from IP stream to a RF signal at this time to allow for integration reasons.

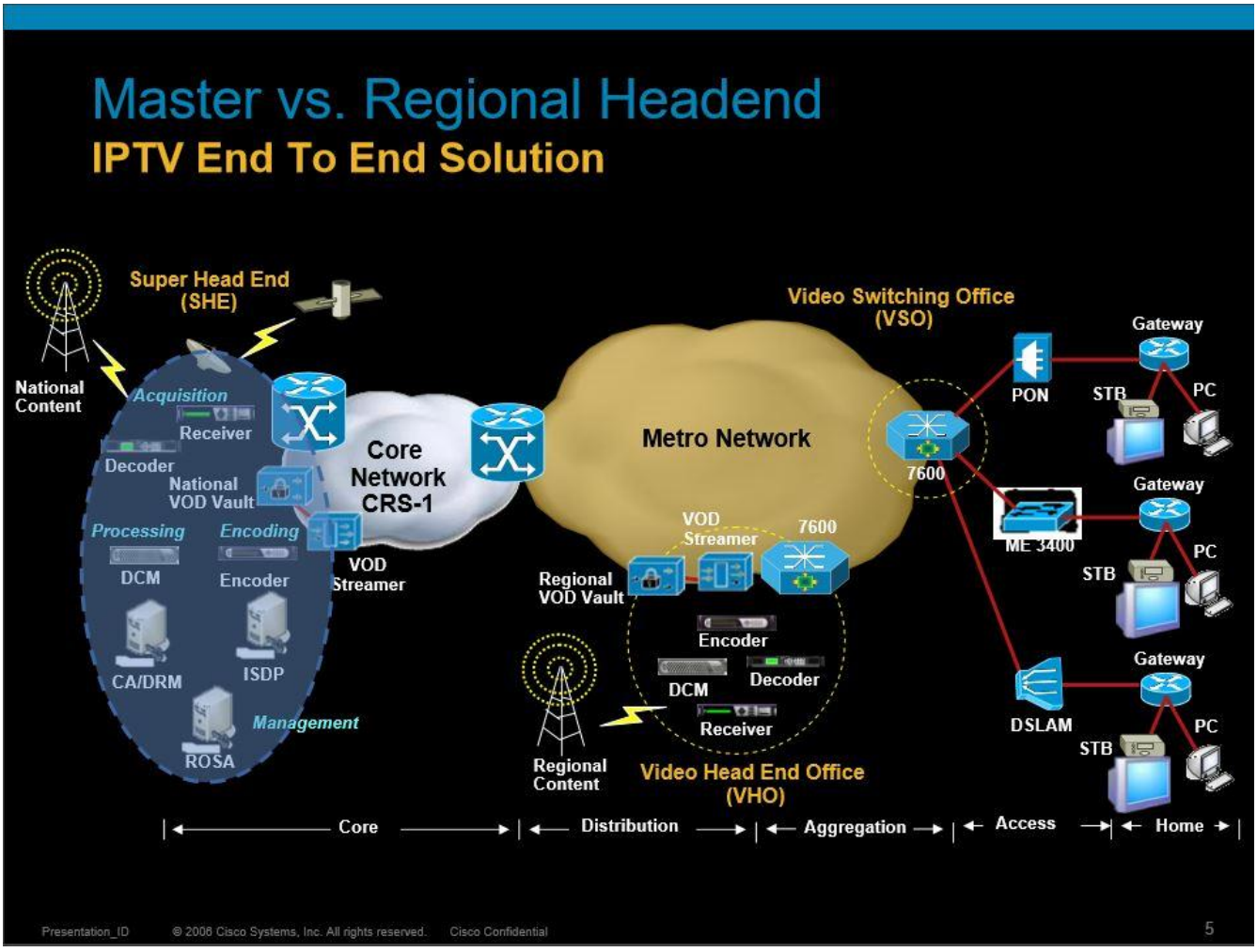



*Figure 2: Cisco Systems© 2006, Signal distribution.*

## 1.3 Motivation

The motivation behind this project is nothing like this has been done before. Technology in hardware and software has been increasing at an alarming rate. We feel technology and our knowledge, that this project can be done. Being the first does bring a kind of fulfillment but mostly, when the project is completed, a glowing set of pride that we were the first to fulfill. My project partner Kyle and Paul is passionate about the hardware side of this project and myself have a passion for coding.

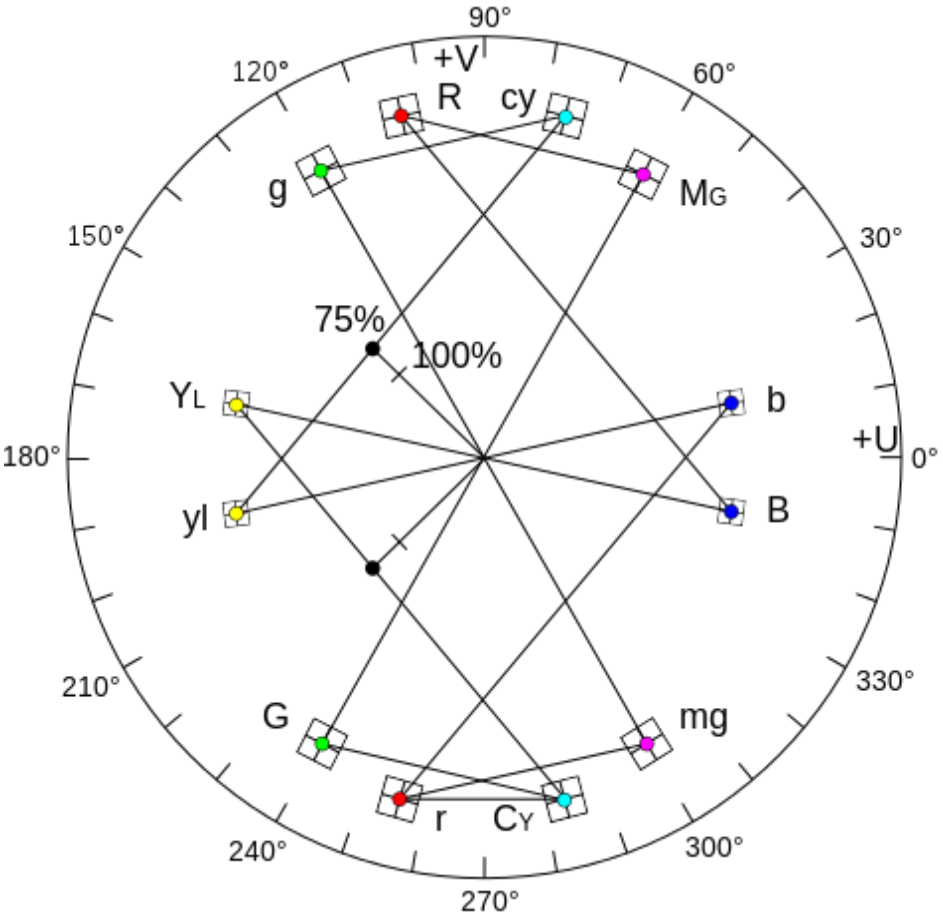



*Figure 3: Analog QAM: measured PAL color bar signal on a vector analyzer screen.*

## 1.4 Recent Developments

Technology, encryption, decryption, and modulation compression scheme we feel has reached a point where we will be able to combine these multi-lateral steps into a single receiver. Major positive technological changes have happened in the field of communication over the past years. Today, when we are talking of human communication, we are thinking of communications through technological innovations like the Internet, cable television, the

telephone, etc. Technological advances in Communication system have been great in recent years. The key technology underlying all modern communication technology is electronics. After the introduction of so many technologies, devices such as BlackBerry were being introduced, which push e-mails to mobile phones. Line-of-business applications are also beginning to combine services. This brings us to the present day where we see the result of different communications technologies creating streams of communication that do not work together as well as they could. This means that a lot of effort has been put into technology to make communication easy but little thought has been applied to making systems work together.

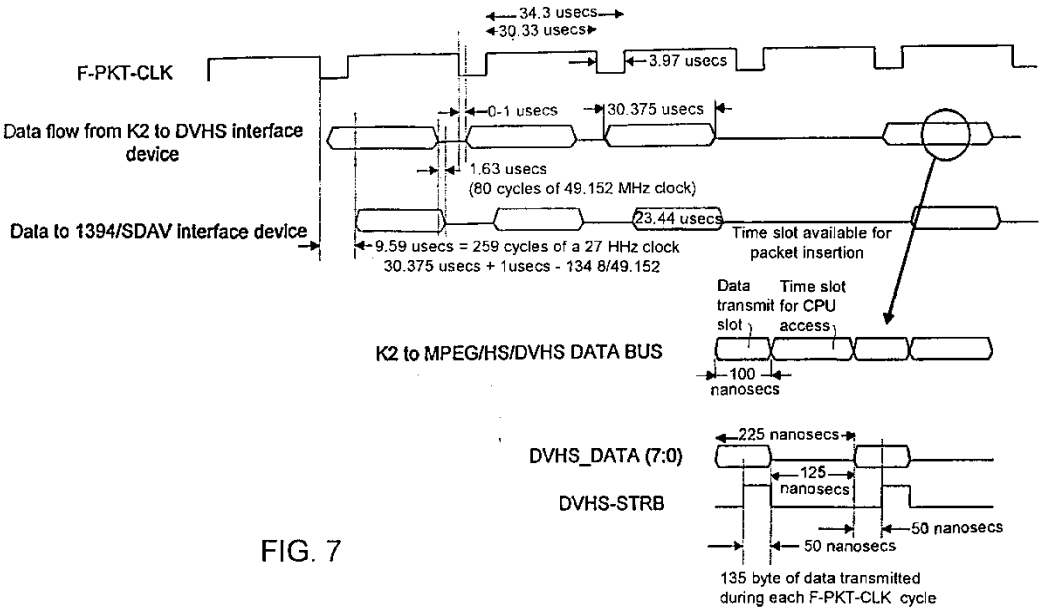



FIG. 7



*Figure 4: Method and apparatus for partial response demodulation, Leo Montreuil.*

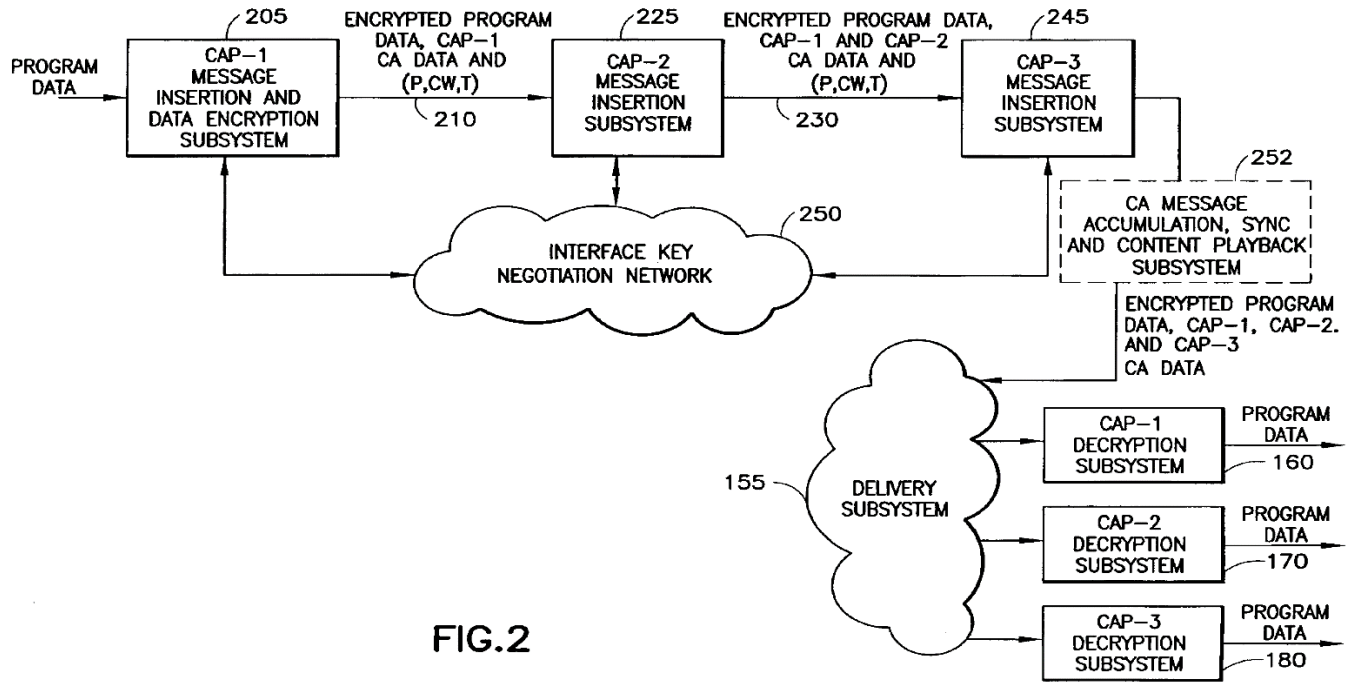



FIG.2

*Figure 5: : System to deliver encrypted access control information to support interoperability between digital information processing/control equipment, US Patent US6898285 B1 George T. Hutchings, Eric Sprunk, Lawrence D. Vince, Richard DiColli, Mark DePietro.*

# Chapter 2

# System Integration and Modeling

---

## 2.1 Technology

Headend Decryption was written in OPENCL 2.1 and complies with Open Computing Language standards released November 2015 and International Standard ISO/IEC 14882:2014(E). The updated release of 2.1 from 2.0 gives the programmer the freedom of coding most of the program using an industry standard language of C++ following the extension 14 guild lines which were published May 15, 2013 (N3690) and March 2, 2014 (N3936) which were accepted and released December 15, 2014. Coding in OPENCL C++14 allows the code to be updated following C++17 guild lines for further improvements and changes as needed.

Our Headend Capstone project relies on the Altera Arria V SoC Development Board programed using Altera Quartus II Software. This software allows either flashing the quad SPI flash memory or loading the software using a standard SD card. Figure 2.1 shows OPENCL flow chart.

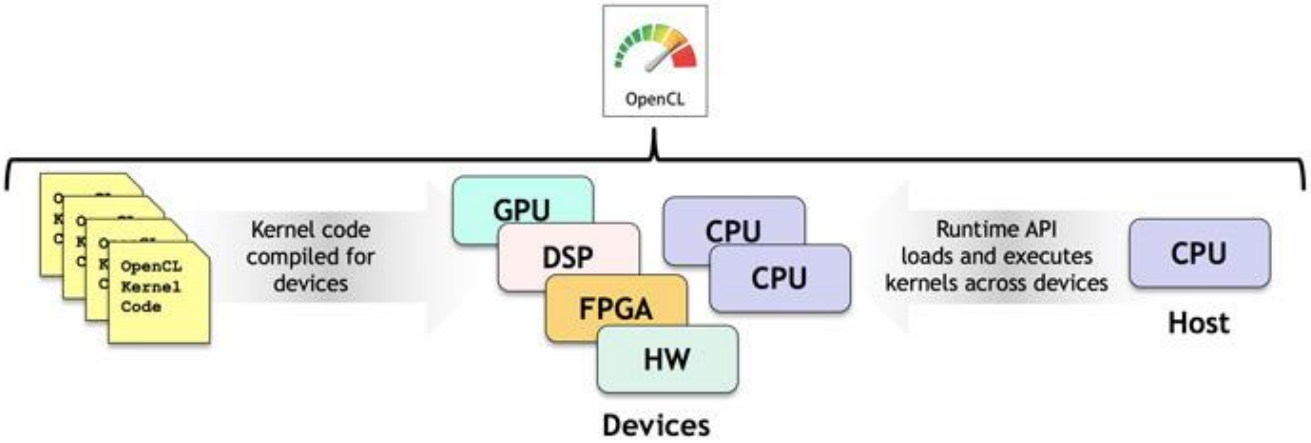



*Figure 2.1: Shows OPENCL Flow chart.*

# 2.2 Design

### 2.2.1 Top-down Design

Headend encryption and decryption software is written in OPENCL based on the C++ standards, which will request the key based on BISS encryption every 5 minutes and verify the top signal key with the lower signal key. The keys should be the same four-digit number, which can be verified on using the Altera V Soc digital read out display. Figure 2.2 shows this data flow.

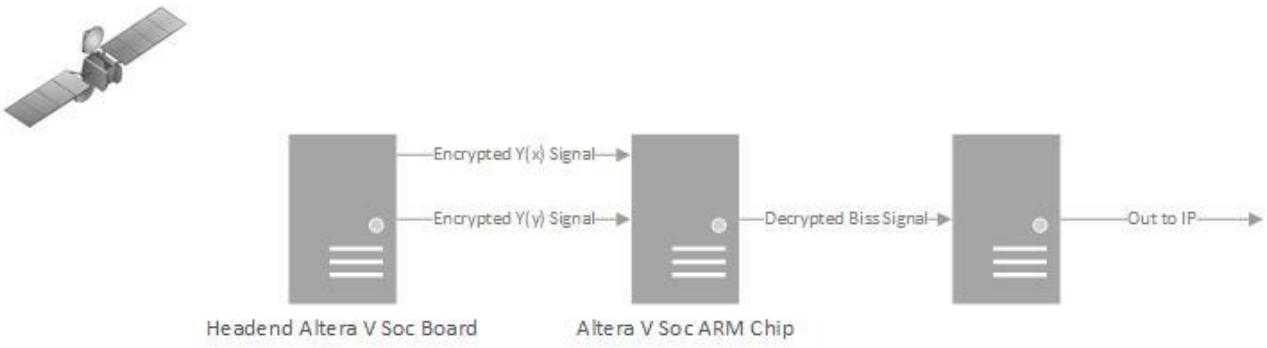



*Figure 2.2: Headend data flow using Altera V Soc Board.*

Altera V Soc Board supports dual functions to support the FPGA side and a Soc ARM chip to allow function programing. Primary configuration of the software is preformed through a USB cable connected to a computer to flash the quad SPI Flash Memory or can be programed using a standard SD card. The software should be simple and make it easy to run

the first time. However, it also must be very flexible to allow the software engineer to adapt changes to varying instrumentation, network design, and user preferences.

### 2.2.2 Bottom-up Design

Digital Headend project is intended to support the production a large number of signals on a continuous basis, each of which will require data to be retrieved from 1Ghz to 2Ghz signal frequency. The keys from the data may be prohibitively slow if multiple keys performed sequentially. The software must be able to process these requests concurrently. However, sending an excessive number of concurrent requests to a single key may impact other clients of that system. To balance these needs, a bounded pool of connections to each ARM chip will be maintained. To ensure keys are produced in a time-ordered sequence, each key request will be added to a synchronized priority queue based of the request or update of the key. Figure 2.3 shows an example of threads created with 4 key requests.

The initial project schedule is shown in Figure 2.3. It is expected to evolve over the course of the project.

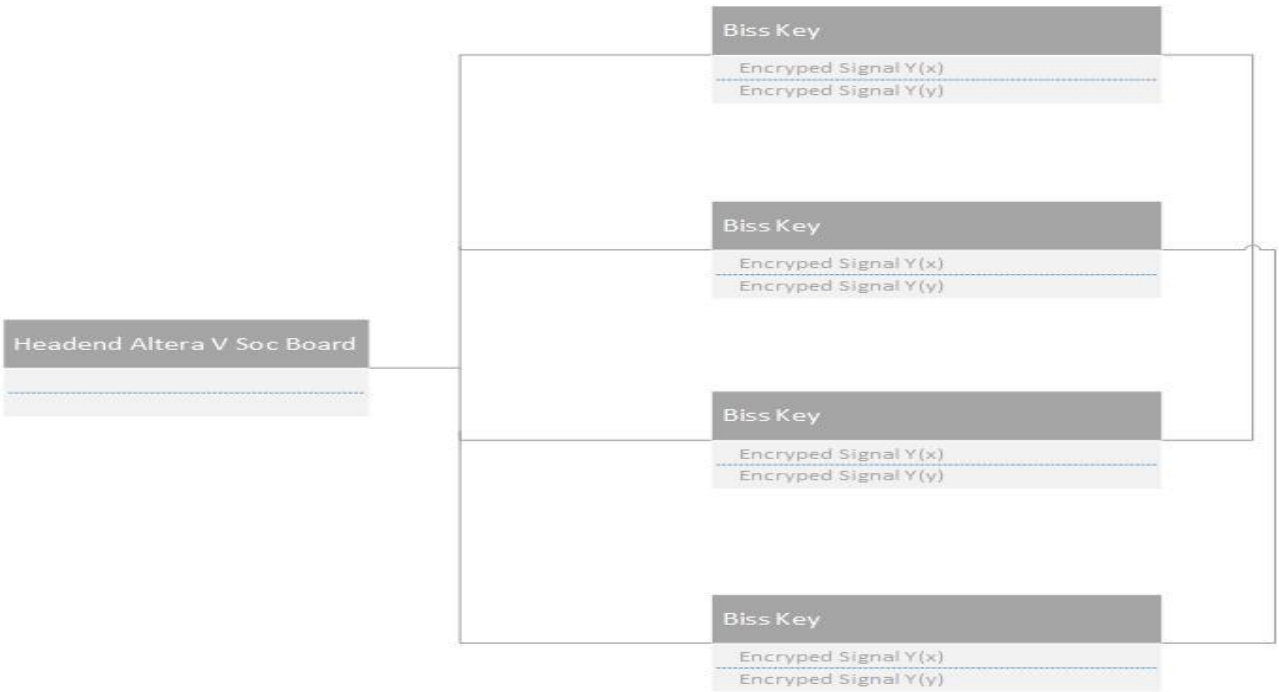



*Figure 2.3: An example of threads created with 4 connections to Alter V Soc Board.*

Headend encryption software will produce an unbounded number of keys. To make archival and navigation of these keys simple, they will be placed into hierarchy based on date in a Single Linked List. The specific structure of the hierarchy will be configurable by the user using C++ strftime function.

FPGA Altera V Soc Encryption and Decryption Software Project

| ID | Task Name | Start | Finish | Actual Finish | Duration | Actual Duration |
|---|---|---|---|---|---|---|
| 1 | Capstone Project Idea | 1/11/2016 | 1/20/2016 | 1/20/2016 | 8d | 8d |
| 2 | ALTERA V Soc Board | 1/21/2016 | 2/18/2016 | 2/18/2016 | 21d | 1d |
| 3 | OPENCL C++14 Software | 2/18/2016 | 2/22/2016 | 2/23/2016 | 3d | 4d |
| 4 | OPENCL Encryption Software | 2/22/2016 | 2/29/2016 | 3/18/2016 | 6d | 19d |
| 5 | OPENCL Decryption Sowtware | 3/1/2016 | 3/21/2016 | 3/22/2016 | 15d | 15d |
| 6 | Software Testing | 3/23/2016 | 4/15/2016 | 4/15/2016 | 18d | 18d |
| 7 | Finial Software Release | 4/18/2016 | 4/25/2016 | 4/25/2016 | 6d | 6d |


*Figure 2.4: Project Schedule.*

## 2.3 Components Used in Development

Development of Digital Headend backend software was performed using Microsoft Visual Studio 2015, a licensed IDE development environment and Altera Quartus II Software, a licensed and subscription Software.

Documentation of the Headend code is created using the freely available Doxygen tool, which parses the declarations and documentation comments in a set of source files and produces a set of HTML pages describing the classes, interfaces, constructors, methods, and fields. Model diagrams were created with Microsoft Visio 2016. Features of Microsoft Visio include multiple-page printing, export to varies formats (EPS, SVG, CGM, PNG, and JPEG), and the ability to use custom shapes created by the user as simple XML descriptions. Microsoft Visio is useful for drawing UML diagrams, network maps, flowcharts, and data flow charts.

In addition to the development tools, server libraries were used in constructing Headend.

## MegaCore® IP Library

MegaCore® contains Altera IP MegaCore functions. This allows the user to simulate behavior of the MegaCore function within your system, verify functionality of your design, and quickly and easily evaluate its size and speed.

## OpenCore Plus

OpenCore Plus hardware evaluation feature is an evaluation tool for prototyping.

## Nios® II Embedded Design Suite (EDS)

A full-featured set of tools that allows you to develop embedded software for the Nios II processor, which the user can include in Altera FPGA designs.

# 2.4 Coding Methodology

## 2.4.1 Agile

Agile coding methodology emphasizes iteration and flexibility over more traditional software development methodologies. Agile grew out of a meeting of developers who felt the current system of writing needed to be standardized and produced The Agile Manifesto to document the four core values which define Agile. Figure 2.5 shows the Agile lifecycle.

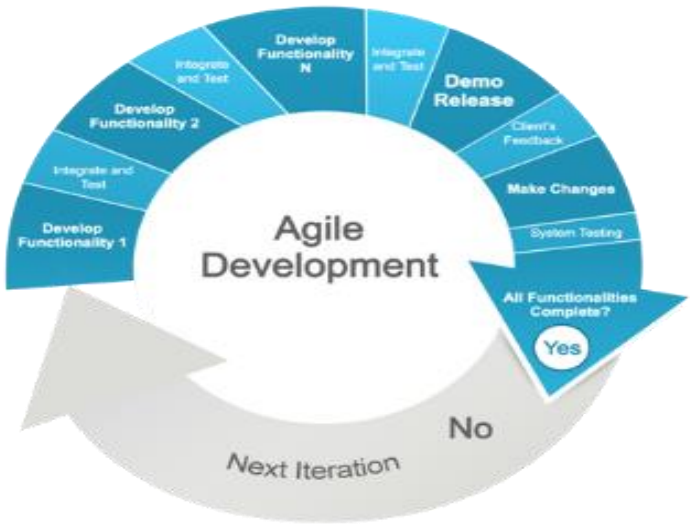



*Figure 2.5: Agile lifecycle. Google Images.*

# Chapter 3
# User Interface and Testing Methodology

---

## 3.1 User Interface

Users will primarily interact with the Headend software application through Altera Quartus II® software and Eclipse® software, an open source software. The complexity of OPENCL C++ configuration will vary with the requirements of its users. A short concise configuration will work for many users, while users desiring a greater degree of customization are also supported. Users will be able to see the keys being interacted through Altera's V Soc digital display. Please Note: This might change do to clients concerns about security, but for testing and demonstrating, this will stay.

Headend software users will view and navigate the logs created through a single page application, viewed in the web browser of their choice. The single page application can be used either locally or remotely.

## 3.1.1 Configuration

Headend is configured through either Altera Quartus II® or Eclipse® software through a Dynamic Link Library (DLL) file. This file is parsed into a tree whose child nodes inherit values from their parent. Figure 3.1 shows the structure of the configuration tree.

Configuration of the Headend software is preformed using a single configuration file named Headendconfig.dll. This file contains a hierarchical set of configuration stanzas Figure 3.1 shows the tree layout. This permits both a very expressive configuration and also a very simple configuration, depending of the users need.

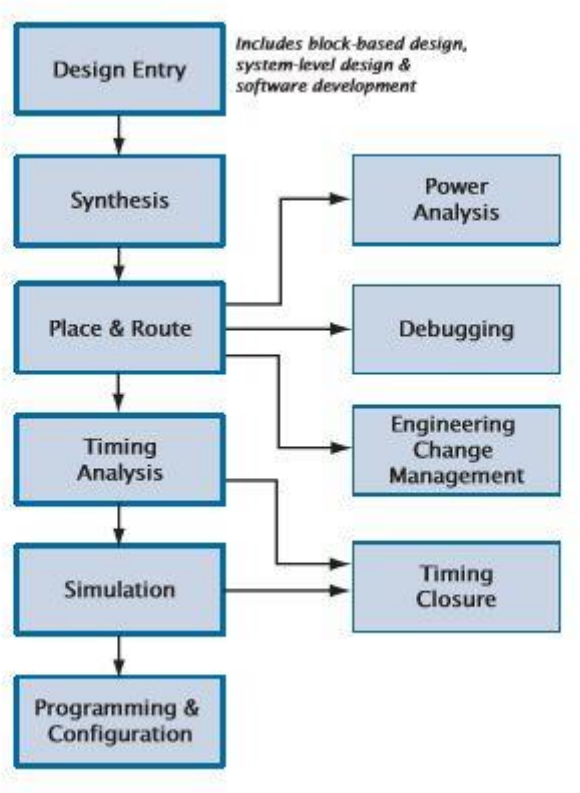



*Figure 3.1: Configuration tree for Software.*

## 3.1.2 Key Display and Master Key

Display and navigation of the Biss keys and configuration menu by Digital headend is done either by a web browser or by Headconfig.dll file. The user will load the Headend HTML page by opening it directly on their workstation or through a URL pointing to a

remote server. The Headend page is divided into three sections. The top bar contains buttons to select the overall state of the display; a middle bar that displays the Biss key from Y(x) and Y(y) signals and the timespan shown between both, along with buttons to move to an adjacent display either spatially or temporally; and the main display of the system currently controlled by Headend network.

The top bar provides a provides controls to set the overall state of the display. Selection lists are provided to choose the network and subnet that is displayed. Following the selection lists are three buttons labeled" Absolute real time"," Y(x) Time", "Y(y) Time".

The Absolute real time box shows the start time and finish time the data is getting transmitted in Nanoseconds. The button will display a modal dialog box with a single text output field. Headend uses dynamic HTML to display keys and respond to users input if needed. At the top of the page is a series of buttons which allows the user to the single window, Headend has two display modes; mosaic and single image. Both modes use a consistent header bar which allows the user to choose a different signal or stream to view, without altering either the display mode or time span displayed. The mosaic display (see Figure 3.2) mode allows the user to quickly scan days of stream data at once. This mode displays the name of the signal at the top of the page with buttons on either side used to switch to the previous or next signal in the list. Below the signal name is the time span displayed. This line has buttons on either side of the timespan allowing the user to move forward or backwards in time, along with a third button which moves to the most recent time span.

Below the time span, there are a series of rows of images. Each row begins with the time of the first sample displayed on the signal and ends with the last signal not shown. This method of displaying the end time was selected to avoid unnecessarily precise time displays and to provide continuity between rows. Between the timestamps is a mosaic of thumbnail images of the signal without axis decoration. These are presented with a thin line between images to help in estimating time within the row. The single signal display (see Figure 3.3) allows the user to take a closer look at a short time span. The header is similar to that of the mosaic display. In this mode, a single image is displayed with fully decorated axes

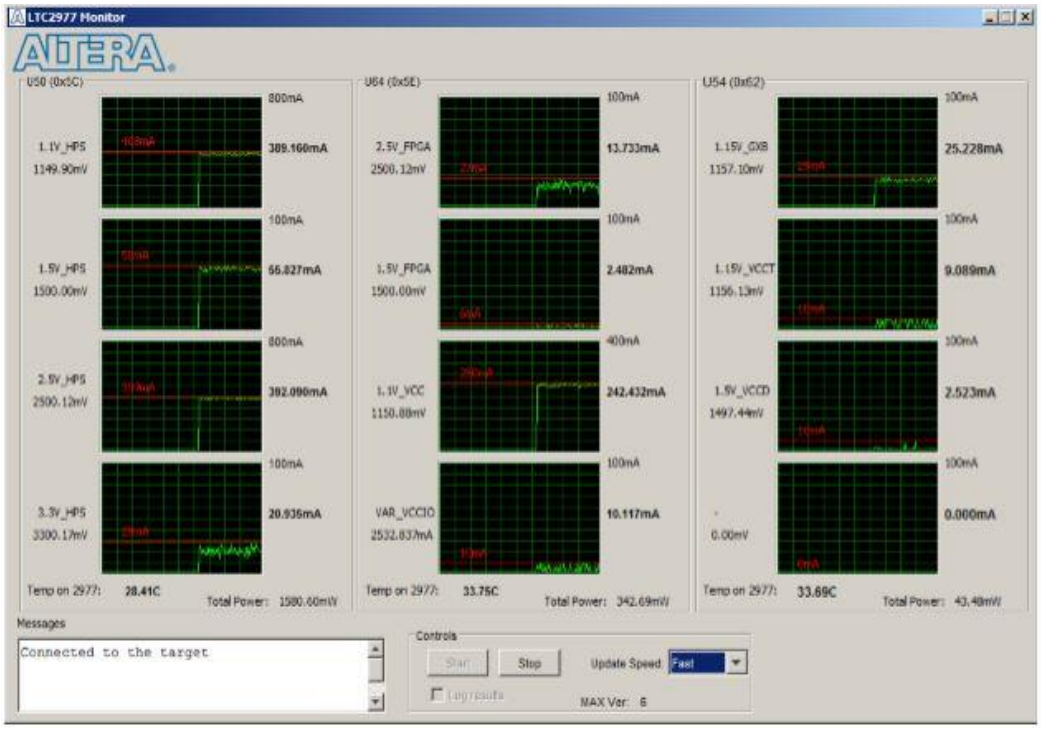



*Figure 6: Power and  multiple signals using Altera Quatuars II Software.*

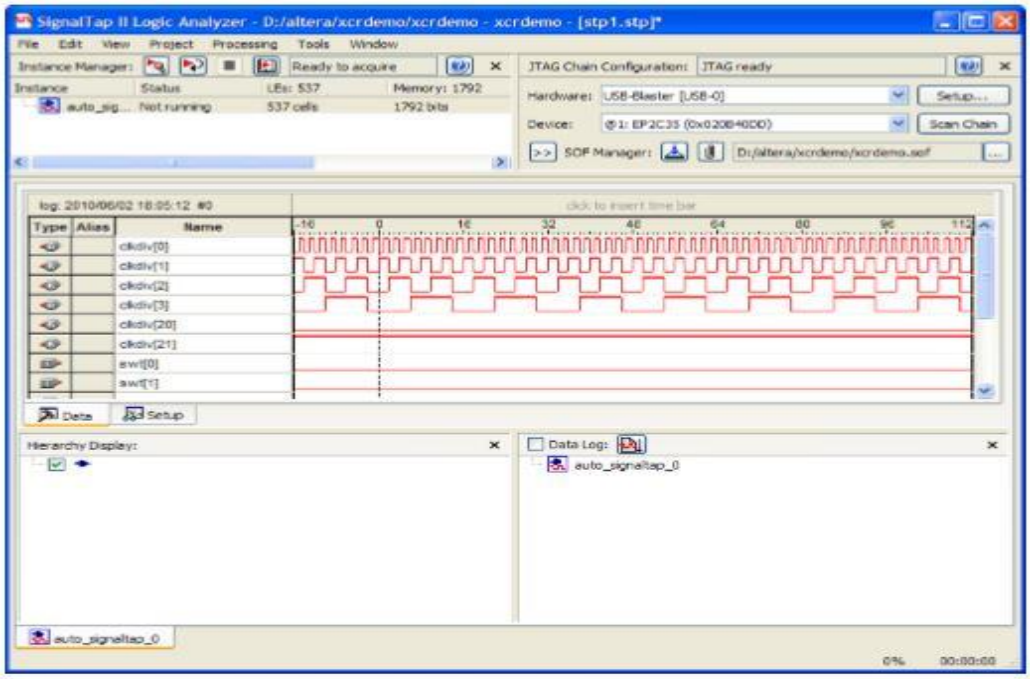



*Figure 3.3: Single Signal using Logic Analyzer.*

## 3.2 Testing Methodology

Testing of Digital Headend backend software was divided into classifications by purpose. These classifications were correctness, performance, reliability, and security.

### 3.2.1 Correctness Testing

Correctness is the minimum requirement of software, the essential purpose of testing. Headend underwent a combination of black-box and white-box correctness testing to verify that the application works as expected with differing configuration scenarios and that it met the needs of software engineers and was useful in monitoring signal activity.

Black-box testing is driven by the applications functional requirements and data, without consideration of the underlying structure of the code. Black-box testing was performed by putting Headend into use at General Communications Inc. Headend was configured to receive signals for every monitored satellite in GCI's Network and allowed to run for several weeks. During this time, software engineers and network administrators regularly used Headend in their daily monitoring activities. Feedback from the users was integrated into the application and testing was completed with no unresolved bug reports.

White-box testing is performed with knowledge of the structure of the code. White-box testing was performed by creating several configuration files, attempting to exercise the full range of configuration options the Headend software and Altera V Soc Board supports. Configurations including invalid or nonsensical directives were included to assess how Headend handles error conditions. While there is room to improve feed back to the user, Headend handled all configurations in an acceptable manner.

### 3.2.2 Performance Testing

Scalability was an important goal for Headend. Users should be able to configure as many signals and streams required to cover the BISS keys they monitor. This goal requires Headend to use system and network resources efficiently. Headend was configured with two networks which had a combined total of 2 subunits, containing signals for 287 channels of continuous video data. This configuration was allowed to run for several days and monitored for both excessive instantaneous resource consumption and increasing resource consumption over time. It is expected that disk space is the only resource need which should increase with time.

A full complement of signals for GCI was configured and run on a modest desktop computer. Headend was started and allowed to run continuously for one week. During this

period the application was profiled using the Altera Quartis II® software. The heap size of the running Quartus II was watched over a period of two weeks. After 8 days' heap consumption had remained nearly constant as shown in Figure 3.4. Thread count remained constant as shown by Figure 3.5

### 3.2.3 Reliability Testing

Reliability testing of Headend focused on how the app behaved when faced with various failures of its network and upstream data source. The app was run continuously for several days as its environment changed and its response was monitored. The app failed gracefully when its upstream data provider became unreachable or unresponsive, and recovered on its own when those conditions cleared up.

### 3.2.4 Security Testing

The portions of Headend vulnerable to attack have intentionally been kept small. Care was taken to make the system robust and all known security issues have been addressed.

Headend will do its best to obey the provided configuration file. This was a deliberate design decision to allow maximum flexability to Headend users. However, this requires that the configuration file be appropriately protected by the underlying operating system. The Headend web application accepts HTTP form variables to set the initial state of the display. These variables are sanitized using a white list permitting only characters appropriate for each field. Specifically, Headend was designed to avoid vulnerability to cross-site scripting attacks and brute force attacks.

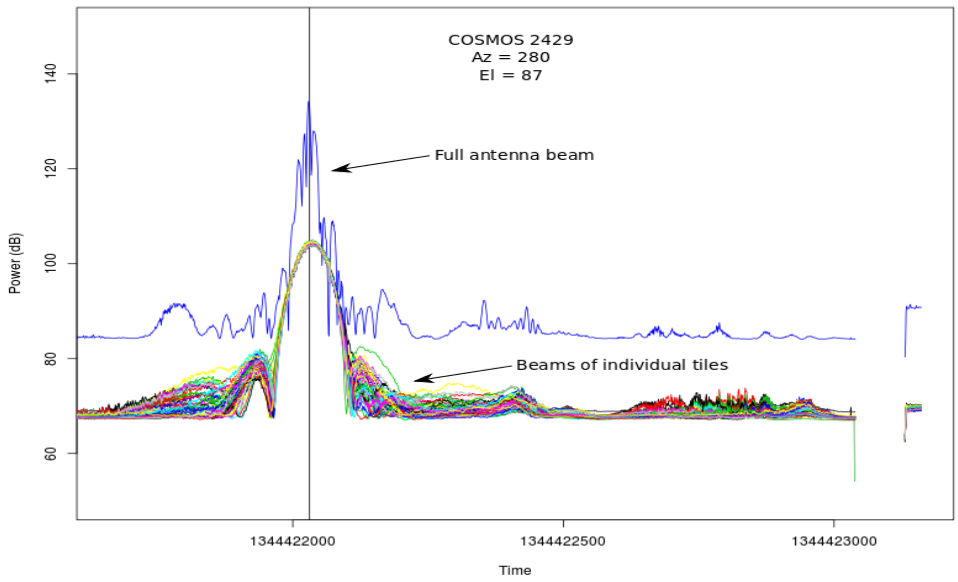



*Figure 3.4: Satellite Signal with the range of 1Ghz~2Ghz.*

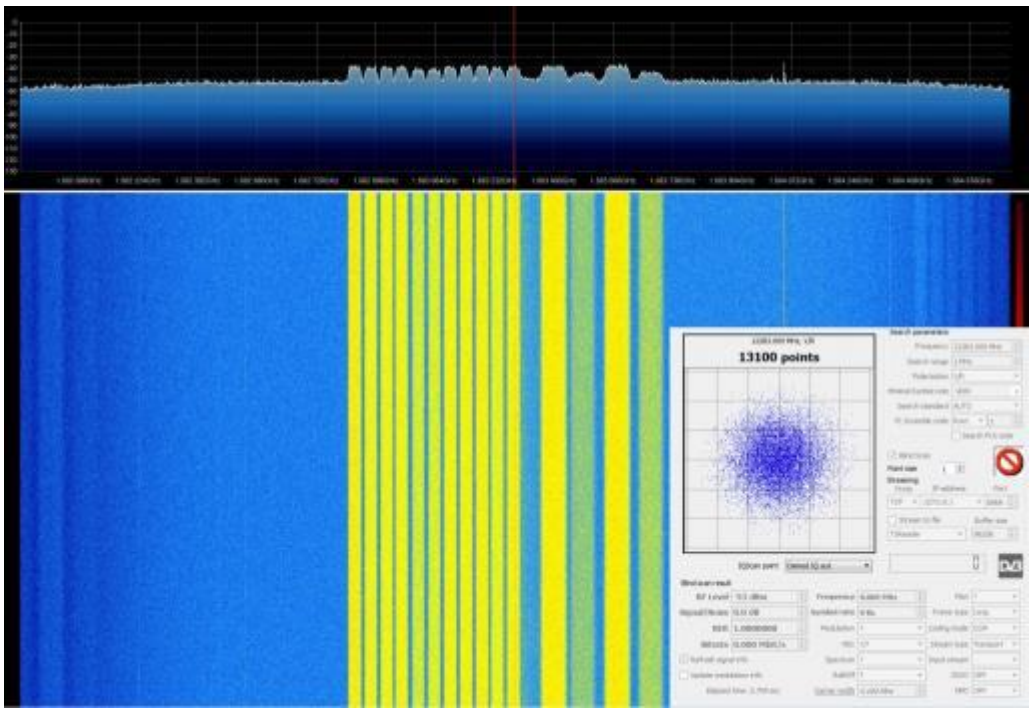



*Figure 3.5: Satellite Y(x) and Y(y) Signal.*

## 3.3 Agile Project Management

Agile project management attempts to integrate the philosophy of Agile development into the project management process. Agile Leadership Network, an organization dedicated to applying agile leadership principles and values. In their Declaration of Interdependence, they describe the goals of agile project management as:

• We increase return on investment by making continuous flow of value our focus.

• We deliver reliable results by engaging customers in frequent interactions and shared ownership.

• We expect uncertainty and manage for it through iterations, anticipation, and adaptation.

• We unleash creativity and innovation by recognizing that individuals are the ultimate source of value, and creating an environment where they can make a difference.

• We boost performance through group accountability for results and shared responsibility for team effectiveness.

• We improve effectiveness and reliability through situationally specific strategies, processes and practices.

This project did not use formal agile project management.

# Chapter 4
# User Manual

## 4.1 Introduction

All incoming signals from input modules initially arrive in the Digital Headend IP-pool. From this pool they can be converted into any output signals you require while simultaneously being fed to several different output modules. The configuration of input and output modules can be readily changed at any time.

The ability of converting any input signal to any output signal makes the Digital headend system flexible, efficient and economical.

It is possible to install up to 12 output modules in a headend unit. Each output module consists of 4 RF channels. Depending on the type of module you can add up to 4 CAM modules on each output module.

In the center part of the output section there are to slot for the installation of one or two auxiliary modules (for future options). For details refer to products that use auxiliary boards.

## 4.2 Configuration

If you want to use the Digital headend system to get and deliver IP services, you must connect the main unit and subunits to a Gigabit network switch. The network switch has to support IGMP version 2 and to have an adequate number of ports to connect to all the Link sockets on the main and subunits.

If your future plans include the handling of IP services, then Digital Headend Project recommends that you use a network switch for connecting the main and subunits even if you do not presently handle IP services.

Connect Link 1 and Link 2 on each unit to a port on the network switch using Cat5e shielded or better cables and SFP transceivers in the Link sockets.

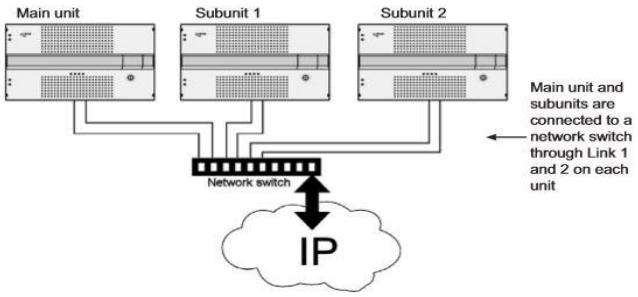



*Figure 4.1: Main unit connected to Subunit.*

When you have connected the main unit and the subunits, you have to set the IDs of the main unit and the subunits using the “ID switch” on the HTML webpage menu. You set the main unit to “3”, subunit1 to “1” and subunit 2 to “2”.

Connect each headend unit to a combiner using RF cables from the RF output socket to the combiner. Next you connect each headend unit to the ground. Then you connect each headend unit to the mains with a power cord from the power socket to the mains installation.

## 4.2.1 Configuration HTML format

In order to set up and configure the modules within the headend system you have to connect your laptop (Windows based) to the RJ45 socket on the main headend unit using a Cat5e shielded cable or better.

The Digital Headend system uses the programs below:

**Minimum requirements:**

Operating system: Windows XP or above.

Browser: Windows Internet Explorer version 6.0 or equivalent.

Additional software: Microsoft© Silverlight Runtime version 3.0 or above.

Altera Quantuas II Software 15.0 or above.

**Note:** Windows 10 has issues with installation with Quantuas II Software, it is recommended setting the installation to Windows 7 or 8/8.1.

The first time you want to access the Digital Headend HTML Service Tool, open your web browser on your laptop and input the following IP address in the web address bar **http://192.168.0.100** and press enter.

**Note:** Your laptop has to use a static IP address in order to access the Digial Headend HTML Service Tool using the above mentioned IP address.

The first time you want to log in to your Digial headend system you have to enter the default password. The default password is: UAA1234 You can change the default password as soon as you have logged in to your headend system.

**Note:** After 20 minutes of inactivity in the Digial Headend Service Tool (you have not displayed or changed anything), you will automatically be logged out unless you have selected the Keep me logged in option.

Every time you set up a new headend system you have to start from scratch in the Digial Headend Service Tool, meaning that you will have change the default password, select which language you want to use in the Digial Headend Service Tool, and change the IP address if your Digial headend system is part of a distribution network.

## 4.2.2 System Information

To get detailed information about the individual units you can click the unit in question in the System information list and open the System information window.

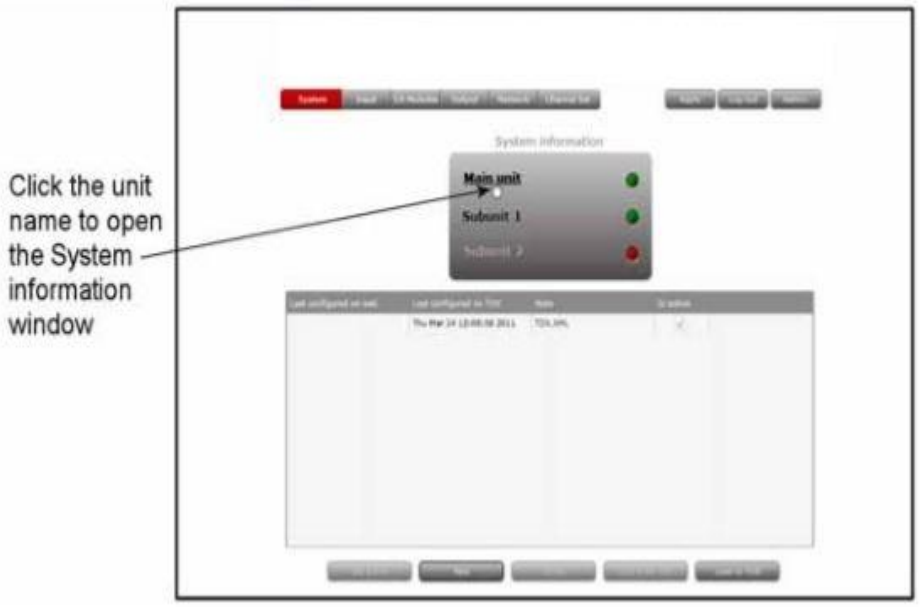



*Figure 4.2: HTML Main System Information.*

In the System information window you get information about system errors, which software version each of the input and output modules uses, the MAC addresses and the current/minimum/maximum temperatures of the unit, power supply and each input/output module in the selected unit.

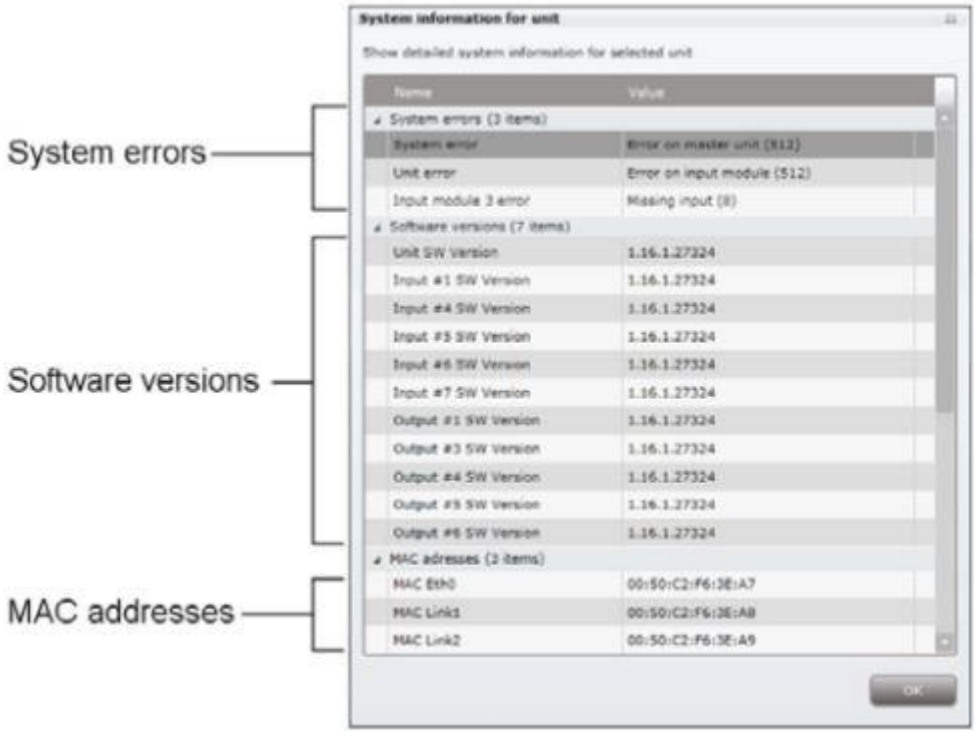



*Figure 4.3: System Information.*

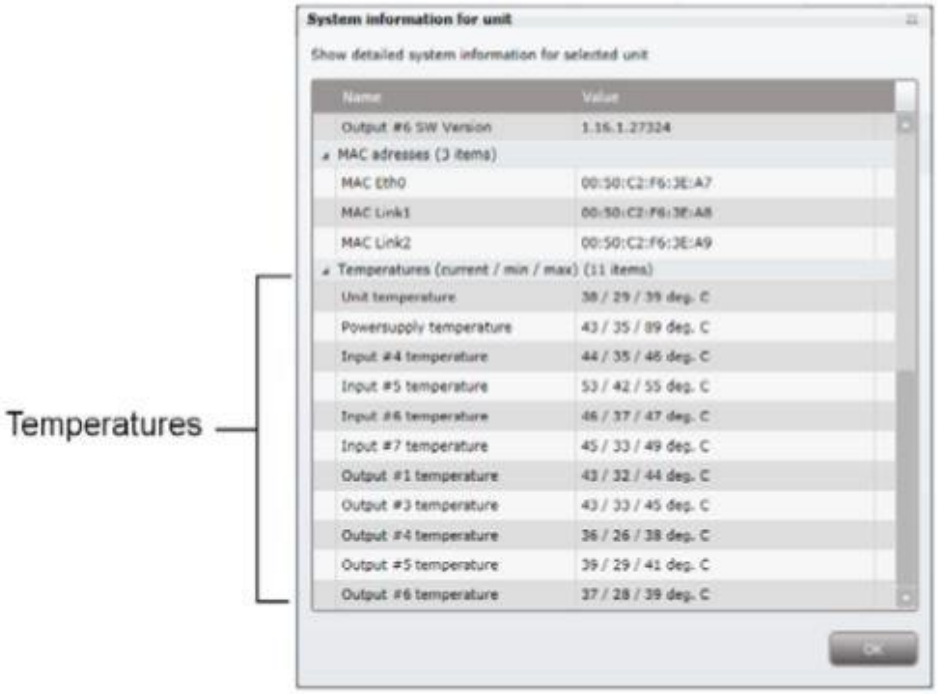



*Figure 4.4: System Information Continued Bottom Half.*

If the temperature of the unit, power supply or a module rises significantly it may indicate that something is wrong.

In connection with errors in your Digital headend system you can use the System information window as starting point for locating the errors.

## 4.2.3 IP output configuration window

The first time the Digial Service Tool displays the Configuration window in a new configuration the fields in the window will either display a default value or be empty.

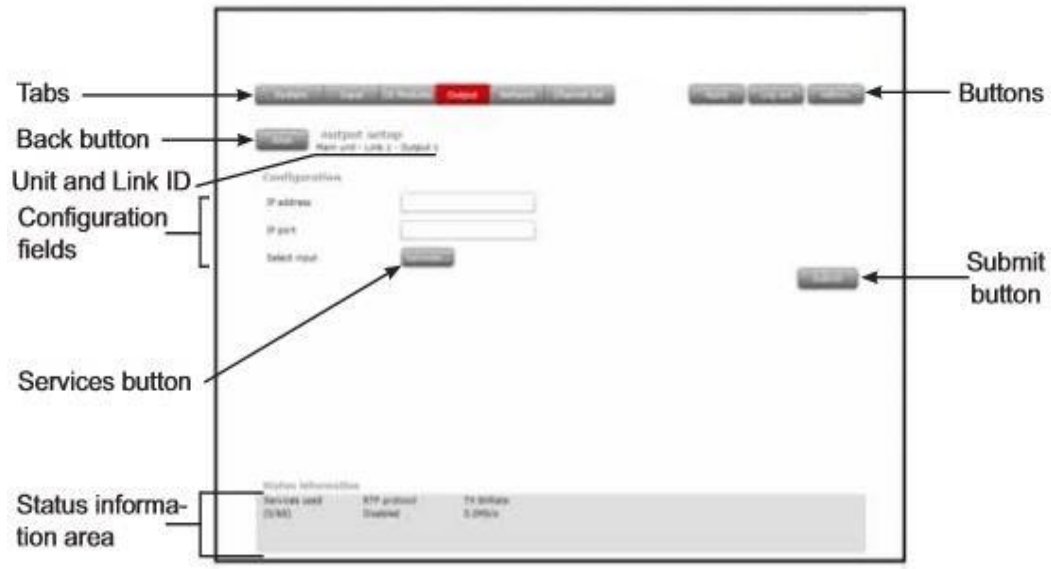



*Figure 4.5: Main Output.*

**IP address:** Enter the desired IP address in the IP address field.

**Port Enter:** the desired IP port number in the Port field.


*Figure 4.6: IP and Port Configuration.*

**Select input:** To select services, click the Services... button next to Select input label to open the Select services window.

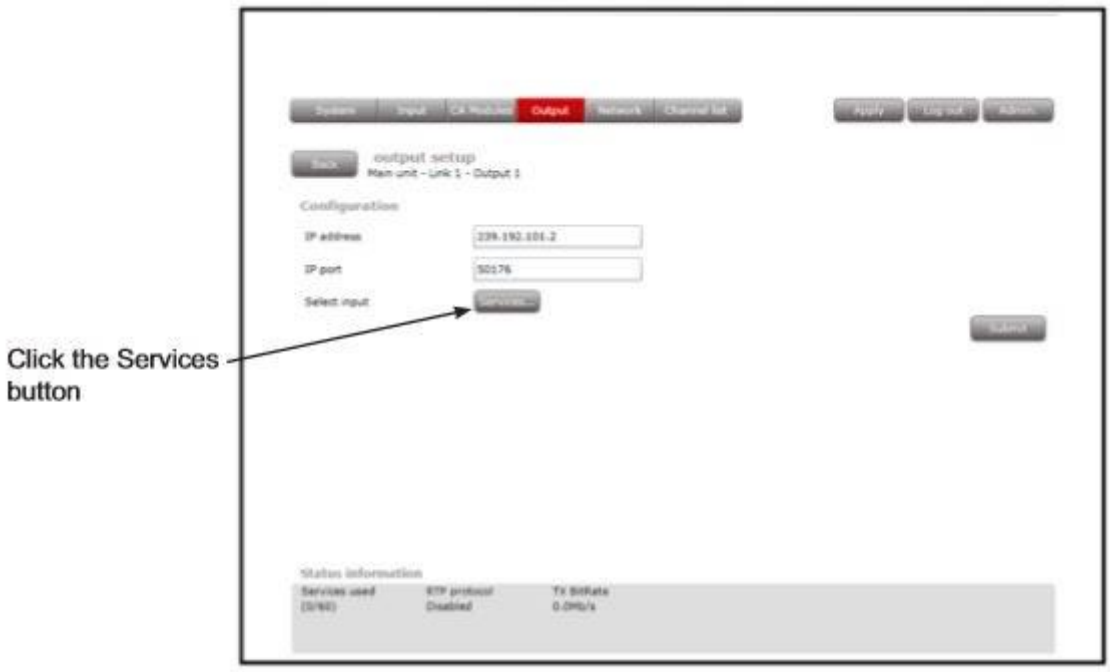



*Figure 4.7: Output Services.*

The Select Services window displays only services from input that has entered the Digital headend system through the same unit whose Link socket(s) you want to use for distributing services.

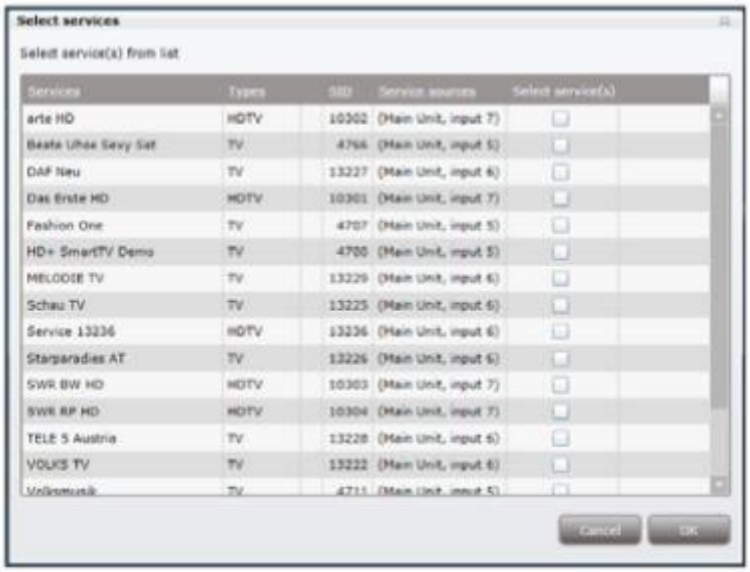



*Figure 4.8: Output Services list.*

In the Select Services window you can select the service(s) that you want to output.

To select one or more services, click the check box (square) to the right of the service you want.

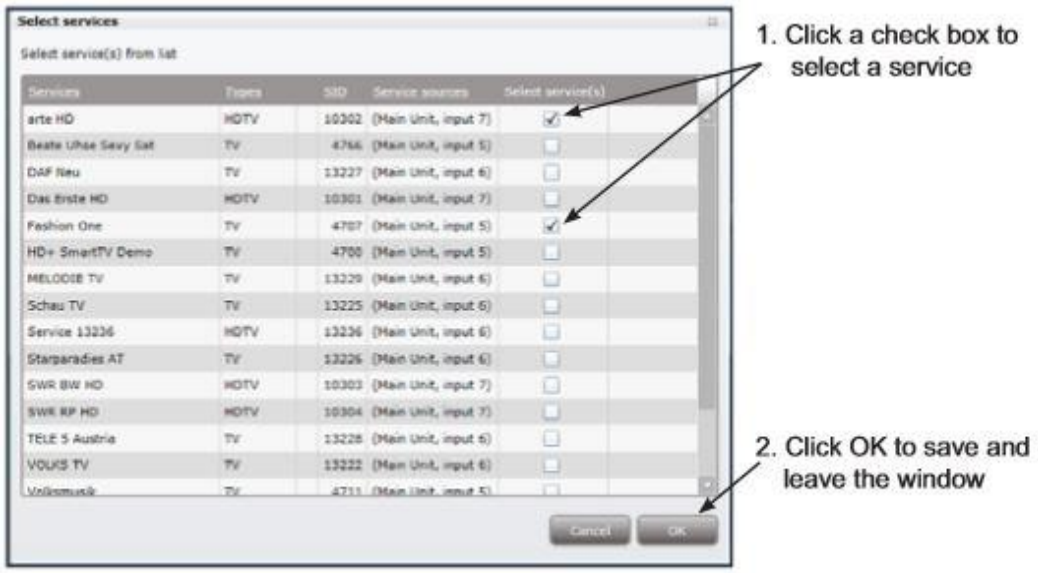



*Figure 4.9: Output Services Select Screen.*

Click OK to return to the Configuration window when you have selected the services you want to output.

**Note:** The services you have selected for one output on a Link will no longer be available in the Digital Headend-pool for other outputs on the same Link.

# Chapter 5

# Conclusion

## 5.1 Summary

We presented the prototyping and development of an OpenCL-to-FPGA decryption program. We showed through two case simulations that the OpenCL-to-FPGA flow is not only feasible, but also effective in designing high-performance circuits. When discussing the initial case studies, we covered major lessons we learnt from this process that helped us shape the architecture of the automated compiler. These lessons were incorporated in the framework, which we implemented and evaluated on a set of applications.

Our work shows OpenCL is well-suited to automatic generation of high-performance circuits for FPGAs. Our framework generated circuits for the benchmark suite that provide high throughput and has been shown to have a wide coverage of the OpenCL language, including synchronization using barriers, support for local memory, as well as floatingpoint operations. In addition, the framework includes a host library to communicate with kernels over PCIe and PC interface. Finally, we have shown the ability to automatically implement complex OpenCL applications that comprise not only an FPGA-based computation engine, but also host processing that interfaces with peripherals.

The circuits generated by our compiler are very different from what a PC interface-based implementation would look like. While at the high-level, the system components are similar, their architecture at the low level accounts for the difference. While on CPUs each processing core performs operations in a SIMD fashion, in our architecture each thread executes a distinct operation on a distinct set of data. Thus in a true sense, this kernel architecture is a multiple-instruction multiple-data style design. This accounts for its size, and also high performance.

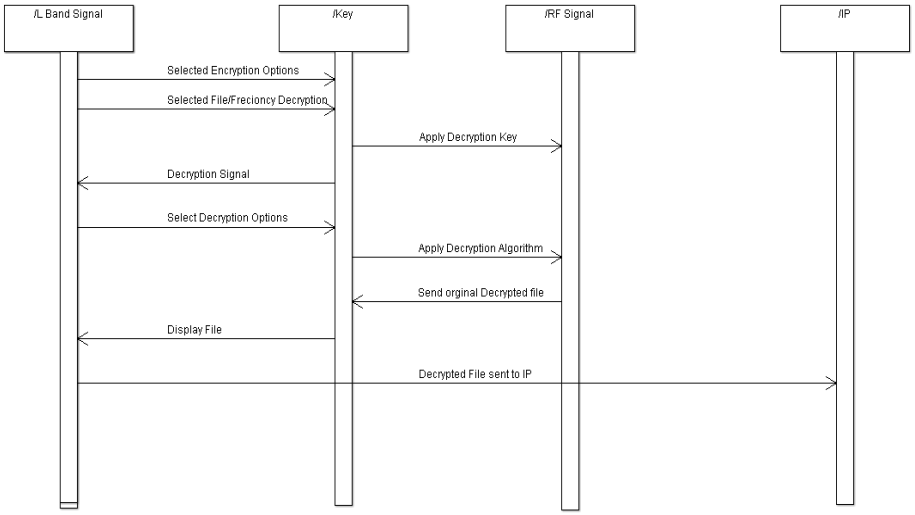

*Figure 5.1: HTML 5 Logo, Bing Images.*

## 5.2 Future Development

Future development of this project can easily be picked up and continued. This is the first part of a multi-part project.

# Appendix A

## UML Sequence Diagram



# Appendix B

# Software License

The *BSD license* is a class of extremely simple and very liberal licenses for computer software that was originally developed at the University of California at Berkeley (UCB). It was first used in 1980 for the *Berkeley Source Distribution* (BSD), also known as BSD UNIX, an enhanced version of the original UNIX operating system that was first written in 1969 by Ken Thompson at Bell Labs.

The only restrictions placed on users of software released under a typical BSD license are that if they redistribute such software in any form, with or without modification, they must include in the redistribution (1) the original copyright notice, (2) a list of two simple restrictions and (3) a disclaimer of liability. These restrictions can be summarized as (1) one should not claim that they wrote the software if they did not write it and (2) one should not sue the developer if the software does not function as expected or as desired. Some BSD licenses additionally include a clause that restricts the use of the name of the project (or the names of its contributors) for endorsing or promoting *derivative works*.

The most basic definition of a derivative work is a product that is based on, or incorporates, one or more already existing works. This can become a complex issue, particularly with regard to software, but the primary indicator that a software program is a derivative of another program is if it includes *source code* from the original program, even if the source code has been modified, including improving, extending, reordering or translating it into another programming language.

Source code is the version of software (usually an application program or an operating system) as it is originally *written* (i.e., typed into a computer) by a human in *plain text* (i.e., human readable alphanumeric characters). Source code can be written in any of hundreds of programming languages, some of the most popular of which are C, C++ and Java.

Due to the extremely minimal restrictions of BSD-style licenses, software released under such licenses can be freely modified and used in *proprietary* (i.e., commercial) software for which the source code is kept secret.

It is possible for a product to be distributed under a BSD-style license and for some other license to apply as well. This was, in fact, the case with very early versions of BSD UNIX, which included both new code written at UCB and code from the original versions of UNIX written at Bell Labs.

BSD-style licenses have been very successful, and they are now widely used for a variety of software. Among the many products released under this class of licenses are all of the major modern descendants of the original BSD UNIX, i.e., FreeBSD, OpenBSD, NetBSD and Darwin (the foundation of the Mac OS X). BSD-licensed software is also commonly included in Linux *distributions* (i.e., versions) and has even been incorporated into some of the Microsoft Windows operating systems.

**BSD Licenses Versus the GPL**

The GPL (*GNU General Public License*) is by far the most widely used license for *free software* (i.e., software whose source code is available at no cost for anyone to use for any purpose). The Linux *kernel* (i.e., the core of the operating system) as well as much of the other software generally included in Linux distributions have been released under the terms of the GPL.

Although far fewer programs are released under BSD-style licenses, this class of licenses is disproportionately important because of the widespread use of BSD-licensed code in both free and proprietary operating systems.

Possibly the biggest difference between the GPL and BSD licenses is the fact that the former is a *copyleft* license and the latter is not. Copyleft is the application of copyright law to permit the free creation of derivative works but requiring that such works be redistributable under the same terms (i.e., the same license) as the original work.

Closely related to this is the fact that, in sharp contrast to the GPL, BSD-style licenses do not require that derivative works based on BSD-licensed software make the source code for such derivative works freely available. This allows the direct incorporation of code from open source projects (i.e., from BSD-licensed software) into closed source projects. The GPL, however, specifically states: "This General Public License does not permit incorporating your program into proprietary programs."

A third difference is that the GPL is a single, copyrighted (by the Free Software Foundation, Inc.) license with no variants. BSD-style licenses, in contrast, are commonly modified for the specific situation.

In many cases, the use of open source code can allow companies to develop products more quickly and with less expense than if they wrote them with entirely original code. The fact that derivative products of BSD-licensed software are not required to be open source can be very useful for developers who want to create commercial products from open source code but who want to keep their modifications and/or extensions secret. Interestingly, companies that initially develop closed source products based on BSD-licensed code tend to be more likely to eventually make their source code publicly available than are companies that develop products that do not incorporate code.

The issue of which license provides greater freedom and does the most to promote the development of improved software is highly controversial. In spite of the seeming simplicity of the licenses, there are no simple answers.

One of the most controversial properties of the GPL is its *viral* nature. This means that once some useful modification or addition to a GPL licensed program has been released, the source code of the modified or extended program must likewise be made freely available. That is, the GPL is a mechanism that deprives developers of the freedom to make their source code secret at some future date, although the developer can still use such code in commercial products. Critics of the GPL claim that this diminishes or destroys the commercial value of software because others can produce products that incorporate the same code.

GPL advocates claim that although the GPL is *contagious* in theory, it is not necessarily so in practice. Rather, they assert, it merely places restrictions on the code's re-use, as do BSD-style licenses.

One thing about both the GPL and the BSD-style licenses for which there is widespread agreement is that both have problems. Neither is perfect, and perhaps no license can be perfect. There is also considerable agreement that there are benefits both to software developers and to society as a whole from the choice provided by the existence of a variety of types of free software licenses, including the GPL and BSD-style licenses.

**The "Advertising Clause"**

The original version of the BSD license contained the so called *advertising clause*, which stated that all advertising materials that mention features of or use of the software must display the acknowledgment: "This product includes software developed by the University of California, Berkeley and its contributors."

One of the problems with this clause arose from the fact that people who made changes to the source code often wanted to have their names added to the acknowledgment. This could easily result in large and cumbersome acknowledgments for products with numerous contributors and for software distributions consisting of multiple individual projects.

A second problem was legal incompatibility with the terms of the GPL. This is because the GPL prohibits the addition of restrictions beyond those that it already imposes. Thus it was necessary to segregate GPL and BSD-licensed software within projects.

Initially, the "obnoxious BSD advertising clause," as it was referred to by GPL advocates, was used only for the BSD UNIX license. That did not cause any major problems because it was only necessary to include a single sentence of acknowledgment in any advertisement.

However, the fact that other software developers did not copy the clause verbatim, but replaced the phrase "University of California" with the name of their own organization or persons involved in it, resulted in a proliferation of slightly different licenses and a consequently serious problem when many such programs were assembled to form a larger work or an operating system. For example, if an operating system or other program required fifty slightly different acknowledgment sentences, each naming a different developer or group of developers, such *advertising* alone might require a full page. Not only would this be very tedious reading, but it could also be costly.

In June 1999, after two years of discussion, the Office of Technology Licensing at UCB finally proclaimed: "Effective immediately, licensees and distributors are no longer required to include the acknowledgment within advertising materials. Accordingly, the foregoing paragraph of those BSD Unix files containing it is hereby deleted in its entirety."

This was clearly very useful. However, it could not eliminate the legacy of the advertising clause, as similar clauses still exist in the licenses of many programs that followed the old BSD license; only the developers of such packages can change them.

(3) A template for a BSD-style license. [YEAR], [COPYRIGHT OWNER] and [LICENSOR] are to be replaced by the actual year of copyright, the owner of the copyright and the licensor.

The copyright owner and licensor may be the same, as in the case of the license for FreeBSD (as shown above).



Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY [LICENSOR] "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

*** Disclaimer:** The above material is presented for reference purposes only, and it is not intended as nor does it constitute legal advice. Neither Bellevue Linux nor any of its content providers shall be liable for any errors or omissions in the content or for any actions taken in reliance thereupon. The author of said material is not an attorney and makes absolutely no claim to have any knowledge about legal matters beyond that of an informed layman. Any questions should be referred to a licensed attorney specializing in copyrights and intellectual property law. Proper legal advice can only be provided by a licensed attorney with reference to the specific facts of a particular situation and to the laws of the relevant jurisdiction.

# Appendix C

**www.github.com/raschenb**

# Code:

```
#include "stdafx.h"
#include<string.h>
#include<stdlib.h>
#include<stdio.h>
#include<iostream>
#include<fstream>
//#include<cl.h> OPENCL C++14 header
#include<stack>
#define SIZE 100000


using namespace std;

int main()
{
	ofstream myfile;
	ofstream myfile1;
	stack s, d;
	s.init();
	stack1 a;
	a.init1();
	cout << "\n\n\n********************//DECRYPTION//******************** \n\n";

	/*cout<<"Loading! Please wait...";
	int dot2,dot22=1;
	for(dot2=0;dot2<dot22;dot2++)
	{
	int waste1;
	int time1=90005799563254;
	int temp23401=0;
	int temp23411=1;
	int temp23421=9;
	for(waste1=0;waste1<time1;waste1++)
	{
	temp23421=temp23401+temp23411;
	temp23401=temp23421;
	temp23421=temp23401+temp23411;
	}
	cout<<".";
	}*/
	cout << "\n";
	//cout<<"Enter the message to be encrypted.\n";
	//cout<<"Note:-Message can be a string of characters,Number and Valid
symbols."<<endl;

	//cout<<"Enter Below:\n";
	//char ch;
	//int i=0;
	int count1 = 0;



	char cut, str[256];
	ifstream is;

	cout << "Enter the name of an existing text file(To be Decrypted): ";
```

```
cin.get(str, 256);

is.open(str);

int fileSize = 0;
if (is.is_open())
{
	is.seekg(0, ios::end);
	fileSize = is.tellg();
}
cout << "File size is " << fileSize << "\n";

is.close();

is.open(str);

char msg[fileSize], msg1[fileSize], chkey[fileSize];
int numkey; char nkey[2];
for (int i = 0; i<fileSize; i++)
{
	cut = is.get();
	msg1[i] = cut;
}


for (int i = 0; i < 2; i++)
{

	cut = msg1[i];
	switch (cut)
	{
	case '0':s.push(0);
		break;

	case '1':s.push(1);
		break;
	case '2':s.push(2);
		break;
	case '3':s.push(3);
		break;
	case '4':s.push(4);
		break;
	case '5':s.push(5);
		break;
	case '6':s.push(6);
		break;
	case '7':s.push(7);
		break;
	case '8':s.push(8);
		break;
	case '9':s.push(9);
		break;
	}

}

int first = s.pop();
int second = s.pop() * 10;
int sumf;
sumf = first + second;
cout << "Length of pwd:" << sumf << endl;

cout << "Pwd:" << endl;
```

```
	int count10 = 0;
	for (int i = 3; i<sumf + 3; i++)//doubt
	{
		cut = msg1[i];
		chkey[count10] = cut;
		count10++;
		cout << cut;
	}
	cout << endl << "Encrytped:" << endl;
	int count11 = 0;
	for (int i = sumf + 4; i<fileSize; i++)
	{
		cut = msg1[i];
		msg[count11] = cut;
		count11++;
		cout << cut;
	}


	is.close();

	fstream is4;
	is4.open(str, std::fstream::out | std::fstream::trunc);
	is4.close();




	count1 = fileSize;




	/*while(ch!='\n')
	{

	ch=getchar();
	msg[i]=ch;
	i++;
	count1++;
	}
	if(msg[0]=='\n')
	{
	cout<<"No Message.Can't Encrypt.Please Try Again with Valid MESSAGE. \n Program
Terminated."<<endl;
	exit (0);
	}



	count1=count1-1;*/

	//cout<<count1<<endl;
	cout << "\n";

	cout << "Enter the security keycode:\n";
	cout << "Note:-Keycode can be a string of characters,Number and Valid symbols."
<< endl;

	cout << "Enter Below:\n";

	char keycopy[100], key[100], ch1;
```

```
	int j = 0;
	int count2 = 0;
	getchar();
	while (ch1 != '\n')
	{

		ch1 = getchar();
		key[j] = ch1;
		j++;
		count2++;
	}
	if (key[0] == '\n')
	{
		cout << "No Key.Can't Encrypt.Please Try Again with Valid KEYCODE. \n
Program Terminated." << endl;
		exit(0);
	}
	count2 = count2 - 1;


	int noofcells = 0;
	noofcells = (count11 / 4);

	if (count11 % 4 != 0)
	{
		noofcells = noofcells + 1;
	}




	int noofcells1 = 0;
	noofcells1 = (count2 / 4);

	if (count2 % 4 != 0)
	{
		noofcells1 = noofcells1 + 1;
	}



	int noofspace;
	noofspace = count11 % 4;
	int x;
	x = noofspace;
	noofspace = 4 - x;

	int noofspace1;
	noofspace1 = count2 % 4;
	int x1;
	x1 = noofspace1;
	noofspace1 = 4 - x1;

	cout << "No of cells in message:" << noofcells << endl;
	cout << "No of space in message:" << noofspace << endl;

	cout << "No of cells in key:" << noofcells1 << endl;
	cout << "No of space in key:" << noofspace1 << endl;


	if (noofspace>0 && noofspace<4)
	{
		for (int q = 0; q<noofspace; q++)
```

```
	{
		msg[count11 + q] = ' ';


	}

	cout << "Message:Space correction,Check!" << endl;
}

else
{
	noofspace = 0;
}
if (noofspace1>0 && noofspace1<4)
{
	for (int t = 0; t<noofspace1; t++)
	{
		key[count2 + t] = ' ';


	}

	cout << "Keycode:Space correction,Check!" << endl;
}
else {
	noofspace1 = 0;
}



char c[noofcells][4];
int m, n;

int count4 = 0;
for (m = 0; m<noofcells; m++)
{
	for (n = 0; n<4; n++)

	{

		c[m][n] = msg[count4];
		cout << c[m][n] << endl;

		count4++;


	}


}

char k[noofcells1][4];
count4 = 0;
for (m = 0; m<noofcells1; m++)
{
	for (n = 0; n<4; n++)

	{
		k[m][n] = key[count4];
		cout << k[m][n] << endl;
		count4++;
	}
}
```

```
int y, z;

for (y = 0; y<noofcells; y++)
{
	for (z = 0; z<4; z++)
	{
		switch (c[y][z])
		{
		case '0':s.push(0);
			break;
		case '1':s.push(1);
			break;
		case '2':s.push(2);
			break;
		case '3':s.push(3);
			break;
		case '4':s.push(4);
			break;
		case '5':s.push(5);
			break;
		case '6':s.push(6);
			break;
		case '7':s.push(7);
			break;
		case '8':s.push(8);
			break;
		case '9':s.push(9);
			break;
		case 'a':s.push(10);
			break;
		case 'b':s.push(11);
			break;
		case 'c':s.push(12);
			break;
		case 'd':s.push(13);
			break;
		case 'e':s.push(14);
			break;
		case 'f':s.push(15);
			break;
		case 'g':s.push(16);
			break;
		case 'h':s.push(17);
			break;
		case 'i':s.push(18);
			break;

		case 'j':s.push(19);
			break;
		case 'k':s.push(20);
			break;
		case 'l':s.push(21);
			break;
		case 'm':s.push(22);
			break;
		case 'n':s.push(23);
			break;
		case 'o':s.push(24);
			break;
		case 'p':s.push(25);
			break;
		case 'q':s.push(26);
```

```
        break;
case 'r':s.push(27);
        break;
case 's':s.push(28);
        break;
case 't':s.push(29);
        break;
case 'u':s.push(30);
        break;
case 'v':s.push(31);
        break;
case 'w':s.push(32);
        break;
case 'x':s.push(33);
        break;
case 'y': s.push(34);//prob
        break;
case 'z':s.push(35);
        break;
case 'A':s.push(36);
        break;
case 'B':s.push(37);
        break;
case 'C':s.push(38);
        break;
case 'D':s.push(39);
        break;
case 'E':s.push(40);
        break;
case 'F':s.push(41);
        break;
case 'G':s.push(42);
        break;
case 'H':s.push(43);
        break;
case 'I':s.push(44);
        break;

case 'J':s.push(45);
        break;
case 'K':s.push(46);
        break;
case 'L':s.push(47);
        break;
case 'M':s.push(48);
        break;
case 'N':s.push(49);
        break;
case 'O':s.push(50);
        break;
case 'P':s.push(51);
        break;
case 'Q':s.push(52);
        break;
case 'R':s.push(53);
        break;
case 'S':s.push(54);
        break;
case 'T':s.push(55);
        break;
case 'U':s.push(56);
        break;
case 'V':s.push(57);
```

```
        break;
case 'W':s.push(58);
        break;
case 'X':s.push(59);
        break;
case 'Y':s.push(60);
        break;
case 'Z':s.push(61);
        break;

case '!':s.push(62);
        break;
case '@':s.push(63);
        break;
case '#':s.push(64);
        break;
case '$':s.push(65);
        break;
case '%':s.push(66);
        break;
case '^':s.push(67);
        break;
case '&':s.push(68);
        break;
case '*':s.push(69);
        break;
case '(':s.push(70);
        break;
case ')':s.push(71);
        break;
case '-':s.push(72);
        break;
case '_':s.push(73);
        break;
case '+':s.push(74);
        break;
case '=':s.push(75);
        break;
case '[':s.push(76);
        break;
case ']':s.push(77);
        break;
case '{':s.push(78);
        break;
case '}':s.push(79);
        break;
case ':':s.push(80);
        break;
case ';':s.push(81);
        break;
case '"':s.push(82);
        break;
case '\'':s.push(83);
        break;
case '<':s.push(84);
        break;
case '>':s.push(85);
        break;
case ',':s.push(86);
        break;
case '.':s.push(87);
        break;
case '/':s.push(88);
```

```
                        break;
                case '?':s.push(89);
                        break;
                case ' ':s.push(90);
                        break;
                case '`':s.push(91);
                        break;
                case '~':s.push(92);
                        break;
                case '|':s.push(93);
                        break;
                case '\\':s.push(94);
                        break;




                }

        }


}

z = 0;
int temp, result[noofcells];
int t[4];
for (y = 0; y<noofcells; y++)
{

        for (z = 0; z<4; z++)
        {

                temp = s.pop();

                t[z] = temp;
        }
        t[0] = t[0] * 1000000;
        t[1] = t[1] * 10000;
        t[2] = t[2] * 100;

        result[y] = t[0] + t[1] + t[3] + t[2];
        result[y] = result[y] + 1100000000;

        cout << result[y] << endl;
}
cout << "Reversing the array:" << endl;
int g, h;
int fresults[noofcells];
for (g = noofcells - 1, h = 0; g >= 0; g--, h++)
{

        fresults[h] = result[g];

}


for (h = 0; h<noofcells; h++)
{
        cout << fresults[h] << endl;
}
d.init();
```

```
for (y = 0; y<noofcells1; y++)
{
	for (z = 0; z<4; z++)
	{
		switch (k[y][z])
		{

		case '0':d.push(0);
			break;

		case '1':d.push(1);
			break;
		case '2':d.push(2);
			break;
		case '3':d.push(3);
			break;
		case '4':d.push(4);
			break;
		case '5':d.push(5);
			break;
		case '6':d.push(6);
			break;
		case '7':d.push(7);
			break;
		case '8':d.push(8);
			break;
		case '9':d.push(9);
			break;
		case 'a':d.push(10);
			break;
		case 'b':d.push(11);
			break;
		case 'c':d.push(12);
			break;
		case 'd':d.push(13);
			break;
		case 'e':d.push(14);
			break;
		case 'f':d.push(15);
			break;
		case 'g':d.push(16);
			break;
		case 'h':d.push(17);
			break;
		case 'i':d.push(18);
			break;

		case 'j':d.push(19);
			break;
		case 'k':d.push(20);
			break;
		case 'l':d.push(21);
			break;
		case 'm':d.push(22);
			break;
		case 'n':d.push(23);
			break;
		case 'o':d.push(24);
			break;
		case 'p':d.push(25);
			break;
		case 'q':d.push(26);
```

```
        break;
case 'r':d.push(27);
        break;
case 's':d.push(28);
        break;
case 't':d.push(29);
        break;
case 'u':d.push(30);
        break;
case 'v':d.push(31);
        break;
case 'w':d.push(32);
        break;
case 'x':d.push(33);
        break;
case 'y':d.push(34);
        break;
case 'z':d.push(35);
        break;
case 'A':d.push(36);
        break;
case 'B':d.push(37);
        break;
case 'C':d.push(38);
        break;
case 'D':d.push(39);
        break;
case 'E':d.push(40);
        break;
case 'F':d.push(41);
        break;
case 'G':d.push(42);
        break;
case 'H':d.push(43);
        break;
case 'I':d.push(44);
        break;

case 'J':d.push(45);
        break;
case 'K':d.push(46);
        break;
case 'L':d.push(47);
        break;
case 'M':d.push(48);
        break;
case 'N':d.push(49);
        break;
case 'O':d.push(50);
        break;
case 'P':d.push(51);
        break;
case 'Q':d.push(52);
        break;
case 'R':d.push(53);
        break;
case 'S':d.push(54);
        break;
case 'T':d.push(55);
        break;
case 'U':d.push(56);
        break;
case 'V':d.push(57);
```

```
        break;
case 'W':d.push(58);
        break;
case 'X':d.push(59);
        break;
case 'Y':d.push(60);
        break;
case 'Z':d.push(61);
        break;

case '!':d.push(62);
        break;
case '@':d.push(63);
        break;
case '#':d.push(64);
        break;
case '$':d.push(65);
        break;
case '%':d.push(66);
        break;
case '^':d.push(67);
        break;
case '&':d.push(68);
        break;
case '*':d.push(69);
        break;
case '(':d.push(70);
        break;
case ')':d.push(71);
        break;
case '-':d.push(72);
        break;
case '_':d.push(73);
        break;
case '+':d.push(74);
        break;
case '=':d.push(75);
        break;
case '[':d.push(76);
        break;
case ']':d.push(77);
        break;
case '{':d.push(78);
        break;
case '}':d.push(79);
        break;
case ':':d.push(80);
        break;
case ';':d.push(81);
        break;
case '"':d.push(82);
        break;
case '\'':d.push(83);
        break;
case '<':d.push(84);
        break;
case '>':d.push(85);
        break;
case ',':d.push(86);
        break;
case '.':d.push(87);
        break;
case '/':d.push(88);
```

```
                        break;
                case '?':d.push(89);
                        break;
                case ' ':d.push(90);
                        break;
                case '`':d.push(91);
                        break;
                case '~':d.push(92);
                        break;
                case '|':d.push(93);
                        break;
                case '\\':d.push(94);
                        break;




                }

        }


}

z = 0;
temp = 0; int result1[noofcells1];
int t1[4];
for (y = 0; y<noofcells1; y++)
{
        for (z = 0; z<4; z++)
        {
                temp = d.pop();

                t1[z] = temp;

        }
        t1[1] = t1[1] * 100;
        t1[2] = t1[2] * 10000;
        t1[3] = t1[3] * 1000000;

        result1[y] = t1[0] + t1[1] + t1[3] + t1[2];
        result1[y] = result1[y] + 1100000000;
        result1[y] = 1194949494 - result1[y];
        result1[y] = 1100000000 + result1[y];
        cout << result1[y] << endl;


}

cout << "Reversing the Array:" << endl;
int fresults1[noofcells1];
cout << "checkpoint\n";
for (g = noofcells1 - 1, h = 0; g >= 0; g--, h++)
{

        fresults1[h] = result1[g];

        cout << fresults1[h] << endl;


}
// VISUAL EFFECTS
```

```
cout << "Encryption in Progress! Please wait...";
int dot, dot1 = 3;
for (dot = 0; dot<dot1; dot++)
{
	int waste;
	int time = 57993254;
	int temp2340 = 0;
	int temp2341 = 1;
	int temp2342 = 9;
	for (waste = 0; waste<time; waste++)
	{
		temp2342 = temp2340 + temp2341;
		temp2340 = temp2342;
		temp2342 = temp2340 + temp2341;
	}
	cout << ".";
}

int multiplier;
if (noofcells1 == 1)
{
	multiplier = fresults1[0];
}

else if (noofcells1 == 2)
{
	if (count2 % 2 == 0)
		multiplier = fresults1[0];
	else
		multiplier = fresults1[1];
}
else
{


	if (count2 % 3 == 0)

	{
		multiplier = fresults1[0];
	}
	else if (count2 % 3 == 1)
	{
		multiplier = fresults1[1];
	}

	else {
		multiplier = fresults1[2];
	}
}
cout << "\n";
cout << "Multiplier:" << multiplier << endl;
cout << "MANUPULATING DATA:" << endl;


for (int ha = 0; ha<noofcells; ha++)
{
	fresults[ha] = 1194949494 - fresults[ha];
	fresults[ha] = 1100000000 + fresults[ha];
	cout << fresults[ha] << endl;
}
cout << "checkpoint 2:" << endl;
int tresults[noofcells]; int da;
```

```
for (h = noofcells - 1, da = 0; h >= 0; h--, da++)
{
	tresults[da] = fresults[h];
	cout << tresults[da] << endl;

}

//for(h=0;h<noofcells;h++)
//{
//  fresults[h]=tresults[h];
//}
//
int count55 = 0;
int twod1[4];
fstream is2;
is2.open(str);
for (h = 0; h<noofcells1; h++)
{


	for (count55 = 0; count55<4; count55++)
	{
		twod1[count55] = fresults1[h] % 100;
		fresults1[h] = fresults1[h] / 100;
		//cout<<twod[count5]<<endl;

		if (twod1[count55] / 10 == 0)
		{
			switch (twod1[count55])
			{



			case 0:cout << "0"; is2 << "0";
				break;

			case 1:cout << "1"; is2 << "1";
				break;
			case 2:cout << "2"; is2 << "2";
				break;
			case 3:cout << "3"; is2 << "3";
				break;
			case 4:cout << "4"; is2 << "4";
				break;
			case 5:cout << "5"; is2 << "5";
				break;
			case 6:cout << "6"; is2 << "6";
				break;
			case 7:cout << "7"; is2 << "7";
				break;
			case 8:cout << "8"; is2 << "8";
				break;
			case 9:cout << "9"; is2 << "9";
				break;



			}
		}
		else
			switch (twod1[count55])
			{
```

```
case 10:cout << "a"; is2 << "a";
       break;
case 11:cout << "b"; is2 << "b";
       break;
case 12:cout << "c"; is2 << "c";
       break;
case 13:cout << "d"; is2 << "d";
       break;
case 14:cout << "e"; is2 << "e";
       break;
case 15:cout << "f"; is2 << "f";
       break;
case 16:cout << "g"; is2 << "g";
       break;
case 17:cout << "h"; is2 << "h";
       break;
case 18:cout << "i"; is2 << "i";
       break;
case 19:cout << "j"; is2 << "j";
       break;
case 20:cout << "k"; is2 << "k";
       break;
case 21:cout << "l"; is2 << "l";
       break;
case 22:cout << "m"; is2 << "m";
       break;
case 23:cout << "n"; is2 << "n";
       break;
case 24:cout << "o"; is2 << "o";
       break;
case 25:cout << "p"; is2 << "p";
       break;
case 26:cout << "q"; is2 << "q";
       break;
case 27:cout << "r"; is2 << "r";
       break;
case 28:cout << "s"; is2 << "s";
       break;

case 29:cout << "t"; is2 << "t";
       break;
case 30:cout << "u"; is2 << "u";
       break;
case 31:cout << "v"; is2 << "v";
       break;
case 32:cout << "w"; is2 << "w";
       break;
case 33:cout << "x"; is2 << "x";
       break;
case 34:cout << "y"; is2 << "y";
       break;
case 35:cout << "z"; is2 << "z";
       break;
case 36:cout << "A"; is2 << "A";
       break;
case 37:cout << "B"; is2 << "B";
       break;
case 38:cout << "C"; is2 << "C";
       break;
case 39:cout << "D"; is2 << "D";
       break;
case 40:cout << "E"; is2 << "E";
```

```
        break;
case 41:cout << "F"; is2 << "F";
        break;
case 42:cout << "G"; is2 << "G";
        break;
case 43:cout << "H"; is2 << "H";
        break;
case 44:cout << "I"; is2 << "I";
        break;
case 45:cout << "J"; is2 << "J";
        break;
case 46:cout << "K"; is2 << "K";
        break;
case 47:cout << "L"; is2 << "L";
        break;
case 48:cout << "M"; is2 << "M";
        break;
case 49:cout << "N"; is2 << "N";
        break;
case 50:cout << "O"; is2 << "O";
        break;
case 51:cout << "P"; is2 << "P";
        break;
case 52:cout << "Q"; is2 << "Q";
        break;
case 53:cout << "R"; is2 << "R";
        break;
case 54:cout << "S"; is2 << "S";
        break;

case 55:cout << "T"; is2 << "T";
        break;
case 56:cout << "U"; is2 << "U";
        break;
case 57:cout << "V"; is2 << "V";
        break;
case 58:cout << "W"; is2 << "W";
        break;
case 59:cout << "X"; is2 << "X";
        break;
case 60:cout << "Y"; is2 << "Y";
        break;
case 61:cout << "Z"; is2 << "Z";
        break;

case 62:cout << "!"; is2 << "!";
        break;
case 63:cout << "@"; is2 << "@";
        break;
case 64:cout << "#"; is2 << "#";
        break;
case 65:cout << "$"; is2 << "$";
        break;
case 66:cout << "%"; is2 << "%";
        break;
case 67:cout << "^"; is2 << "^";
        break;
case 68:cout << "&"; is2 << "&";
        break;
case 69:cout << "*"; is2 << "*";
        break;
case 70:cout << "("; is2 << "(";
        break;
```

```
                        case 71:cout << ")"; is2 << ")";
                               break;
                        case 72: cout << "-"; is2 << "-";
                               break;
                        case 73:cout << "_"; is2 << "_";
                               break;
                        case 74:cout << "+"; is2 << "+";
                               break;
                        case 75:cout << "="; is2 << "=";
                               break;
                        case 76:cout << "["; is2 << "[";
                               break;
                        case 77:cout << "]"; is2 << "]";
                               break;
                        case 78:cout << "{"; is2 << "{";
                               break;
                        case 79:cout << "}"; is2 << "}";
                               break;
                        case 80:cout << ":"; is2 << ":";
                               break;
                        case 81:cout << ";"; is2 << ";";
                               break;
                        case 82:cout << "\""; is2 << "\"";
                               break;
                        case 83:cout << "'"; is2 << "'";
                               break;
                        case 84:cout << "<"; is2 << "<";
                               break;
                        case 85:cout << ">"; is2 << ">";
                               break;
                        case 86:cout << ","; is2 << ",";
                               break;
                        case 87:cout << "."; is2 << ".";
                               break;
                        case 88:cout << "/"; is2 << "/";
                               break;

                        case 89:cout << "?"; is2 << "?";
                               break;

                        case 90:cout << " "; is2 << " ";
                               break;

                        case 91:cout << "`"; is2 << "`";
                               break;
                        case 92:cout << "~"; is2 << "~";
                               break;
                        case 93:cout << "|"; is2 << "|";
                               break;
                        case 94:cout << "\\"; is2 << "\\";
                               break;



                        }

        }
}


is2.close();
```

```
	is2.open(str);

	int fileSize2 = 0;
	if (is2.is_open())
	{
		is2.seekg(0, ios::end);
		fileSize2 = is2.tellg();
	}
	else
	{
		cout << "File not found.\n"; exit(0);
	}

	cout << "File size is:" << fileSize2 << "\n";

	is2.close();
	is2.open(str);
	char cut22;
	char msg22[fileSize2];
	count10 = 0;
	int ina;
	for (ina = 0; ina<fileSize2; ina++)
	{
		cut22 = is2.get();
		msg22[ina] = cut22;
		//cout<<chkey[ina]<<endl;
		//cout<<msg22[ina]<<endl;
		if (msg22[ina] != chkey[count10])
		{

			cout << "Incorrect Password.Decryption Terminated!\n Destroying
Source File..." << endl;
			exit(0);
		}
		count10++;
	}
	is2.close();
	is2.open(str, std::fstream::out | std::fstream::trunc);



	is2.close();

	cout << endl;
	cout << endl;

	cout << "Reverse Decrypted Message(AFTER COLON):";
	myfile.open(str);
	int count5 = 0;
	int twod[4];
	int count14 = 0;
	for (h = 0; h<noofcells; h++)
	{



		for (count5 = 0; count5<4; count5++)
		{
			count14++;
			twod[count5] = tresults[h] % 100;
			tresults[h] = tresults[h] / 100;
			//cout<<twod[count5]<<endl;
```

```
if (twod[count5] / 10 == 0)
{
	switch (twod[count5])
	{



	case 0:cout << "0"; myfile << "0";
		break;

	case 1:cout << "1"; myfile << "1";
		break;
	case 2:cout << "2"; myfile << "2";
		break;
	case 3:cout << "3"; myfile << "3";
		break;
	case 4:cout << "4"; myfile << "4";
		break;
	case 5:cout << "5"; myfile << "5";
		break;
	case 6:cout << "6"; myfile << "6";
		break;
	case 7:cout << "7"; myfile << "7";
		break;
	case 8:cout << "8"; myfile << "8";
		break;
	case 9:cout << "9"; myfile << "9";
		break;

	}
}
else {
	switch (twod[count5])
	{


	case 10:cout << "a"; myfile << "a";
		break;
	case 11:cout << "b"; myfile << "b";
		break;
	case 12:cout << "c"; myfile << "c";
		break;
	case 13:cout << "d"; myfile << "d";
		break;
	case 14:cout << "e"; myfile << "e";
		break;
	case 15:cout << "f"; myfile << "f";
		break;
	case 16:cout << "g"; myfile << "g";
		break;
	case 17:cout << "h"; myfile << "h";
		break;
	case 18:cout << "i"; myfile << "i";
		break;
	case 19:cout << "j"; myfile << "j";
		break;
	case 20:cout << "k"; myfile << "k";
		break;
	case 21:cout << "l"; myfile << "l";
		break;
	case 22:cout << "m"; myfile << "m";
		break;
```

```
case 23:cout << "n"; myfile << "n";
       break;
case 24:cout << "o"; myfile << "o";
       break;
case 25:cout << "p"; myfile << "p";
       break;
case 26:cout << "q"; myfile << "q";
       break;
case 27:cout << "r"; myfile << "r";
       break;
case 28:cout << "s"; myfile << "s";
       break;

case 29:cout << "t"; myfile << "t";
       break;
case 30:cout << "u"; myfile << "u";
       break;
case 31:cout << "v"; myfile << "v";
       break;
case 32:cout << "w"; myfile << "w";
       break;
case 33:cout << "x"; myfile << "x";
       break;
case 34:cout << "y"; myfile << "y";
       break;
case 35:cout << "z"; myfile << "z";
       break;
case 36:cout << "A"; myfile << "A";
       break;
case 37:cout << "B"; myfile << "B";
       break;
case 38:cout << "C"; myfile << "C";
       break;
case 39:cout << "D"; myfile << "D";
       break;
case 40:cout << "E"; myfile << "E";
       break;
case 41:cout << "F"; myfile << "F";
       break;
case 42:cout << "G"; myfile << "G";
       break;
case 43:cout << "H"; myfile << "H";
       break;
case 44:cout << "I"; myfile << "I";
       break;
case 45:cout << "J"; myfile << "J";
       break;
case 46:cout << "K"; myfile << "K";
       break;
case 47:cout << "L"; myfile << "L";
       break;
case 48:cout << "M"; myfile << "M";
       break;
case 49:cout << "N"; myfile << "N";
       break;
case 50:cout << "O"; myfile << "O";
       break;
case 51:cout << "P"; myfile << "P";
       break;
case 52:cout << "Q"; myfile << "Q";
       break;
case 53:cout << "R"; myfile << "R";
       break;
```

```
case 54:cout << "S"; myfile << "S";
       break;

case 55:cout << "T"; myfile << "T";
       break;
case 56:cout << "U"; myfile << "U";
       break;
case 57:cout << "V"; myfile << "V";
       break;
case 58:cout << "W"; myfile << "W";
       break;
case 59:cout << "X"; myfile << "X";
       break;
case 60:cout << "Y"; myfile << "Y";
       break;
case 61:cout << "Z"; myfile << "Z";
       break;

case 62:cout << "!"; myfile << "!";
       break;
case 63:cout << "@"; myfile << "@";
       break;
case 64:cout << "#"; myfile << "#";
       break;
case 65:cout << "$"; myfile << "$";
       break;
case 66:cout << "%"; myfile << "%";
       break;
case 67:cout << "^"; myfile << "^";
       break;
case 68:cout << "&"; myfile << "&";
       break;
case 69:cout << "*"; myfile << "*";
       break;
case 70:cout << "("; myfile << "(";
       break;

case 71:cout << ")"; myfile << ")";
       break;
case 72: cout << "-"; myfile << "-";
       break;
case 73:cout << "_"; myfile << "_";
       break;
case 74:cout << "+"; myfile << "+";
       break;
case 75:cout << "="; myfile << "=";
       break;
case 76:cout << "["; myfile << "[";
       break;
case 77:cout << "]"; myfile << "]";
       break;
case 78:cout << "{"; myfile << "{";
       break;
case 79:cout << "}"; myfile << "}";
       break;
case 80:cout << ":"; myfile << ":";
       break;
case 81:cout << ";"; myfile << ";";
       break;
case 82:cout << "\""; myfile << "\"";
       break;
case 83:cout << "'"; myfile << "'";
       break;
```

```
                    case 84:cout << "<"; myfile << "<";
                           break;
                    case 85:cout << ">"; myfile << ">";
                           break;
                    case 86:cout << ","; myfile << ",";
                           break;
                    case 87:cout << "."; myfile << ".";
                           break;
                    case 88:cout << "/"; myfile << "/";
                           break;

                    case 89:cout << "?"; myfile << "?";
                           break;
                    case 90:cout << " "; myfile << " ";
                           break;
                    case 91:cout << "`"; myfile << "`";
                           break;
                    case 92:cout << "~"; myfile << "~";
                           break;
                    case 93:cout << "|"; myfile << "|";
                           break;
                    case 94:cout << "\\"; myfile << "\\";
                           break;



                    }

             }
       }
}




/*char temp1,temp11[100];
char temp12;
for(h=0;h<noofcells+2;h++)
{
for(count5=0;count5<4;count5++)
{
temp1=a.pop1();
cout<<temp1;
myfile <<temp1;


if (temp1=='|')
{
temp12=a.pop1();
cout/*<<"popped Again:"*//*<<temp12;

myfile <<temp12;

temp11[count5]=temp12;
count5++;

}
temp11[count5]=temp1;
//cout<<temp11[count5];

}
}
```

```
	cout<<temp11[count5+1]<<endl;//cout<<temp11[count5+2];

	*/

	cout << "\n There is your Encrypted Message! \n\n"; cout << "Use RS-Decryption
Program to Decrypt!\n\nProgram will terminate now.\n\nTHANK YOU!\n\n\n\n" << endl;

	cout << count14 << endl;
	myfile.close();

	//
	fstream is1;
	is1.open(str);

	int fileSize1 = 0;
	if (is1.is_open())
	{
		is1.seekg(0, ios::end);
		fileSize1 = is1.tellg();
	}
	else
	{
		cout << "File not found.\n"; exit(0);
	}

	cout << "File size is:" << fileSize1 << "\n";

	is1.close();
	is1.open(str);
	char cut11;
	char msg11[fileSize];

	for (int i = 0; i < fileSize1; i++)
	{
		cut11 = is1.get();
		msg11[i] = cut11;

	}
	is1.close();
	is1.open(str, std::fstream::out | std::fstream::trunc);


	for (int i = fileSize1 - 1; i >= 0; i--)
	{
		cout << msg11[i]; is1.put(msg11[i]);
	}

	is1.close();



	return 0;




}
```

# References


Altera Software Installation and Licensing: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/quartus_install.pdf

Altera Quartus II Handbook: https://www.altera.com/support/literature/lit-qts.html

Chin, S. Alexander, and Paul Chow. "OpenCL memory infrastructure for FPGAs." In *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, pp. 269-270. ACM, 2012.

Chorti, A.; Kanaras, I., "Masked M-QAM OFDM: A simple approach for enhancing the security of OFDM systems," in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on* , vol., no., pp.1682-1686, 13-16 Sept. 2009 doi: 10.1109/PIMRC.2009.5450168

Czajkowski, Tomasz S., David Neto, Michael Kinsner, Utku Aydonat, Jason Wong, Dmitry Denisenko, Peter Yiannacouras, John Freeman, Deshanand P. Singh, and Stephen D. Brown. "OpenCL for FPGAs: Prototyping a compiler." In *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012.

Falcao, Gabriel, Muhsen Owaida, David Novo, Madhura Purnaprajna, Nikolaos Bellas, Christos D. Antonopoulos, Georgios Karakonstantis, Andreas Burg, and Paolo Ienne. "Shortening design time through multiplatform simulations with a portable OpenCL golden-model: the LDPC decoder case." In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pp. 224-231. IEEE, 2012.

Fletcher, Christopher W., Ilia A. Lebedev, Narges B. Asadi, Daniel R. Burke, and John Wawrzynek. "Bridging the GPGPU-FPGA efficiency gap." In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 119-122. ACM, 2011.

Gaster, Benedict, Lee Howes, David R. Kaeli, Perhaad Mistry, and Dana Schaa. *Heterogeneous Computing with OpenCL: Revised OpenCL 1*. Newnes, 2012.

Hanas, O.J.; den Toonder, P.; Pennypacker, F., "An Addressable Satellite Encryption System for Preventing Signal Piracy," in *Consumer Electronics, IEEE Transactions on* , vol.CE-27,no.4,pp.631-636,Nov.1981, doi: 10.1109/TCE.1981.273532

Hill, Kenneth, Stefan Craciun, Alan George, and Herman Lam. "Comparative analysis of OpenCL vs. HDL with image-processing kernels on Stratix-V FPGA." In *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, pp. 189-193. IEEE, 2015.

Hirosaki, B., "An Orthogonally Multiplexed QAM System Using the Discrete Fourier Transform," in *Communications, IEEE Transactions on* , vol.29, no.7, pp.982-989, Jul 1981
doi: 10.1109/TCOM.1981.1095093

Intel Developer OPENCL Zone: https://software.intel.com/en-us/intel-opencl

Khronos Group OPENCL Documentation: https://www.khronos.org/spir

Krommydas, Konstantinos, Muhsen Owaida, Christos D. Antonopoulos, Nikolaos Bellas, and Wu-chun Feng. "On the portability of the OpenCL dwarfs on fixed and reconfigurable parallel platforms." In *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, pp. 432-433. IEEE, 2013.

Kultala, Heikki, Otto Esko, Pekka Jaaskelainen, Vladimir Guzma, Jarmo Takala, Jiao Xianjun, Tommi Zetterman, and Heikki Berg. "Turbo decoding on tailored OpenCL processor." In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pp. 1095-1100. IEEE, 2013.

Long Bao, Yicong Zhou, Image encryption: Generating visually meaningful encrypted images, Information Sciences, Volume 324, 10 December 2015, Pages 197-207, ISSN 0020-0255, http://dx.doi.org/10.1016/j.ins.2015.06.049.

Pratas, Frederico, Joao Andrade, Gabriel Falcao, Vítor Manuel Mendes da Silva, and Leonel Sousa. "Open the Gates: Using High-level Synthesis towards programmable LDPC decoders on FPGAs." In *GlobalSIP*, pp. 1274-1277. 2013.

Ma, Sen, Miaoqing Huang, and David Andrews. "Developing application-specific multiprocessor platforms on fpgas." In *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, pp. 1-6. IEEE, 2012.

Mirian, Vincent, and Paul Chow. "UT-OCL: an OpenCL framework for embedded systems using xilinx FPGAs." In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1-6. IEEE, 2015.

Mirian, Vincent, and Peter Chow. "Using an OpenCL framework to evaluate interconnect implementations on FPGAs." In *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, pp. 1-4. IEEE, 2014.

Rodriguez-Donate, C., G. Botella, C. Garcia, E. Cabal-Yepez, and M. Prieto-Matias. "Early experiences with opencl on fpgas: Convolution case study." In *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, pp. 235-235. IEEE, 2015.

Singh, Desh, and Supervising Principal Engineer. "Higher level programming abstractions for fpgas using opencl." In *Workshop on Design Methods and Tools for FPGA-Based Acceleration of Scientific Computing*. 2011.

Singh, Deshanand. "Implementing FPGA design with the OpenCL standard." *Altera whitepaper* (2011).

T. Pfau, S. Hoffmann, and R. NoÃ©, "Hardware-Efficient Coherent Digital Receiver Concept With Feedforward Carrier Recovery for $M$-QAM Constellations," J. Lightwave Technol. **27**, 989-999 (2009).

Takei, Yasuhiro, Hasitha Muthumala Waidyasooriya, Masanori Hariyama, and Michitaka Kameyama. "Design of an FPGA-Based FDTD Accelerator Using OpenCL." In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

Zhang, Chao, Hong-cen Mei, and Hao Yang. "A Parallel Way to Select the Parameters of SVM Based on the Ant Optimization Algorithm." *arXiv preprint arXiv:1405.4589* (2014).