March 6, 2007

Instructor Lakshman One
School of Engineering Science
Simon Fraser University
Burnaby, BC
V5A 1S6

Re: ENSC 440 Design specification for a Voice Bandwidth Saving System

Dear Instructor One,

Please find the attached design specification for the Voice Bandwidth Saving System (VBSS), a communication system that reduces the bandwidth usage of VOIP applications. This document will outline our algorithm and the hardware on which it is implemented. As well, it will include testing requirements to ensure the functional requirements outlined in the functional specification document are met. Implementation of the current design should be finished on a prototype by mid-April, 2007.

Cheri Perception is composed of four SFU undergraduate engineering students with a range of technical skills and experience in software system development: Bryan Cua, Cathy Zhang, Tilson Chung, and Hubert Pan. If you have any questions or concerns regarding our proposal, please contact me by phone at 604.619.0841 or by email at ensc440-cheri@sfu.ca.

Sincerely,

Bryan Cua

Bryan Cua
CEO of Cheri Perception

Enclosure: Design Specification for a Voice over IP Bandwidth Saving System.

CHERI

VBSS Design Specification

VoIP Bandwidth Saving System (VBSS)

Design Specification

Project Team: Hubert Pan
Tilson Chung
Cathy Zhang
Bryan Cua

Contact Person: Bryan Cua
bcua@sfu.ca

Submitted to:	Steve Whitmore	ENSC305
	Lakshman One	ENSC440
	Ash Parameswaran	ENSC440

Issued date: March 9, 2007

Revision: 1.0

Executive Summary

By the end of 2005, about two-thirds of Fortune 2000 companies have certain level of VoIP equipment installed. Some of the companies have implemented a full VoIP phone network while others may use this advanced technology for long distance communication between different branches in their company.

A majority of companies already have an existing network infrastructure available for connection to the internet nowadays. VoIP takes the advantage of this pre-existing infrastructure and rides on it with easily installed software which is compatible with most affordable and modern personal computers. Companies can apply VoIP systems with desired features while the costs of hardware, software, and maintaining are low.

On the other hand, VoIP has its disadvantage that hinders remaining companies from joining this trend, which is the large bandwidth usage. The only applied solution is to upgrade their internet data transfer allowance from the internet service provider; however this is a costly solution. Cheri Perception is a new idea and solution which allows companies to take the advantage of VoIP system while remaining low cost.

Development of our system will occur in three phases. The first stage is the prototype stage in which we will attempt to prove our concept and attain the following:

1. Constrain our budget to be less than $1500
2. Simulate an internet environment
3. Default application specific for G711-compliant phones running in *SipX* v3.4
4. Simulate running with phones.
5. Save bandwidth by at least 45%.

The second stage of development concerns production for which our goals are:

1. Increase compliance to other standards G726 G729 and ITU-T standards
2. Increase compatibility with other VoIP servers
3. Dual internet ports as per design
4. Mass production, price for each unit is within $50
5. Include smart arbitration for a mass network

The aims of our third stage are:

1. Partner with current networking companies
2. Realign current design to integrate with current routers and switches from partnered companies

Revision History

Date	Version	Description
03/03/2007	0.1	Initial Draft
03/04/2007	0.2	Second Draft - Change in design
03/05/2007	0.3	Addition of Introduction and Conclusion
03/07/2007	0.4	Third Draft - Minor correction in design
03/08/2007	1.0	Final Revision

List of Tables and Figures

Table of Contents

Glossary

Term	Definition
Bandwidth	A measure for communication capacity of a channel. Channel, here, primarily refers to the internet connection. A larger bandwidth means more information can be carried through that channel.
Channel	Medium for transfer of information. We will mostly refer to the "internet channel" as the "channel", for which we mean the lines or wires that carry information.
CODEC	**Co**der-**Dec**oder A method of encoding and decoding information to save space while transferring. Analogous to using abbreviations. The writer will encode a word into an abbreviation while the reader decodes it to the full word.
cRTP	**C**ompressed **RTP** Protocols use some bandwidth to store header information. cRTP compresses the header information to save bandwidth.
G.711	An audio CODEC standard used readily in VoIP phones. It utilizes PCM to compress information.
G.7xx	A group of audio CODEC standards that are used in VoIP applications. Each CODEC uses a different way to compress information.
Header Information	Information added onto the data in a protocol, such as the RTP protocol, which directs the data to its destination.
IP	**I**nternet **P**rotocol A protocol for moving data from one location to another. There are two types of information – header and data information.
Layer 2	Data-Link Layer The second layer of networking is the data-link layer, which provides a means to transfer data between elements in a network.
Layer 1	Physical Layer The first layer of networking is the physical layer, which only provides a means to transfer bits of data rather than packets
Peripheral Network	The Ethernet Network that is connected to our VBSS System
Packet	Pieces of information passing through the Ethernet Network are commonly known as packets

Term	Definition
Receiving System	System receiving information (data and header)
RTP	**R**eal **T**ime **P**rotocol A type of internet protocol that deals with data that cannot be delayed.
Transmitting System	System transmitting information (data and header)
VoIP	**V**oice **o**ver **I**nternet **P**rotocol The transfer of voice data over the internet is commonly referred to as VoIP
VoIP Phones	Phones that send information through the internet
µ-law algorithm	A compression algorithm used by G.711. G.711 standard has two algorithms. One is µ-law, and the other is A-law. While µ-law is used in North America and Japan, A-law is used in Europe.

Introduction

VBSS is designed to save the bandwidth usage of VoIP systems, and can be maintained by local staff. It is a transparent and readily installable utility. By utilizing physical differences between the voiced speech of any two different people, VBSS is able to join these two distinguished voices together without spectrum spreading or time multiplexing.

1. Scope

This document describes the design specifications for the VBSS system. It explains the technical details of the design and implementations strategies. Both hardware and software architectures are outlined. This document will guide our implementation and testing process as our VBSS develops.

2. Intended Audience

This design specification is intended to be a guideline for all members of Cheri Perception. Members of the design team will use it to ensure the design progress and detail throughout the development phase. Design engineers will use this document to ensure the expected results are reached.

System Overview

Our conceptual system can be broken down into five different components:

- Channel Simulator
- Packet Management
- Information Retrieval Service
- Voice Processor
- Master controller

as shown in **Error! Reference source not found.** (the Ethernet interface is a physical interface). Those components which only apply to the prototype are prefixed by the word “simulated”, while the remainder applies to both prototype and production. Each component will only communicate with its neighbors.

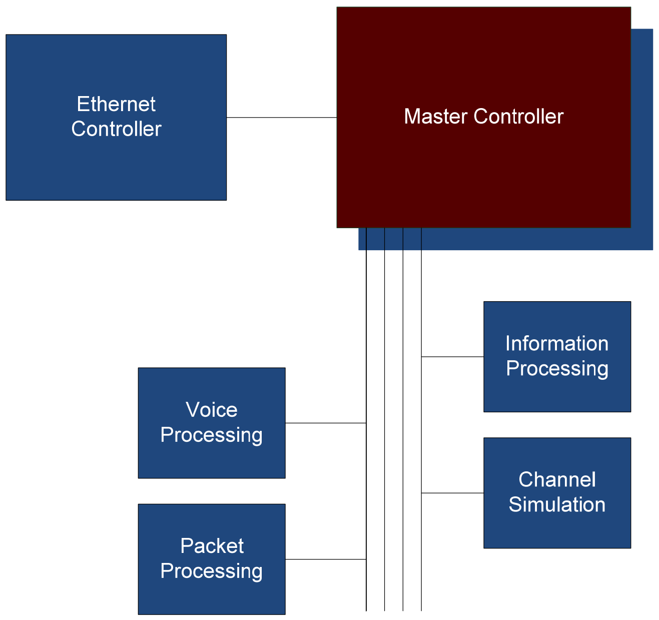

Figure 1: System components (from left to right)

The channel simulator’s function is to simulate an internet channel, and equally, the Ethernet interface simulator’s function is to simulate an Ethernet interface. The packet processing layer inspects the packet for voice information, while the information processing layer is responsible for extracting certain attributes from the packet, such as destination. Finally, the voice processing layer will process the packet voice data.

In this section, the main flow of the system will be presented. Latter sections will discuss these components in more detail.

1. Master Processor

Our main process will be executed on a PowerPC® as per Figure 2.

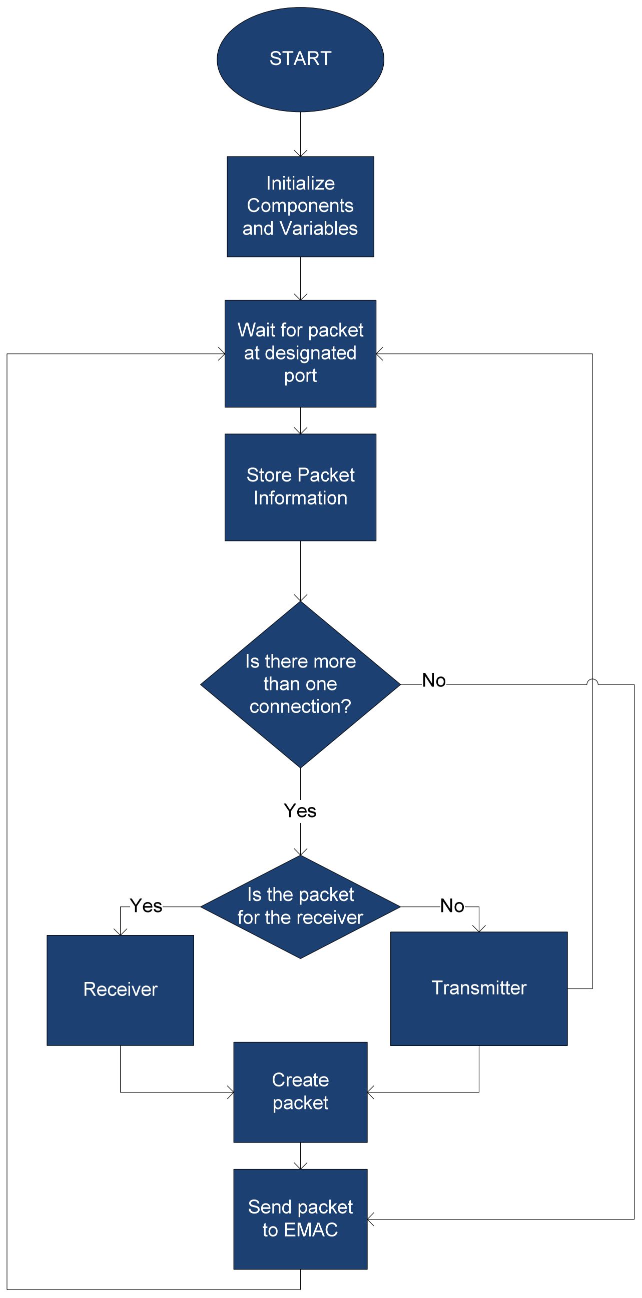
START
Initialize Components and Variables
Wait for packet at designated port
Store Packet Information
Is there more than one connection?
No
Yes
Is the packet for the receiver
Yes
No
Receiver
Transmitter
Create packet
Send packet to EMAC

Figure 2: Main flow

In this diagram, information and voice processing both occur in the receiver and the transmitter. These two stages contain our main algorithm. The transmitter sends a combination packet to a receiver, which splits it back into two packets.

2. Combination Packet

The combination packet which exits the transmitter contains more information than a regular voice packet, as shown in Figure 3. A crucial component of this packet is the inclusion of the speaker's fundamental frequency and their respective destination/origins, which will be included in the modified header information.

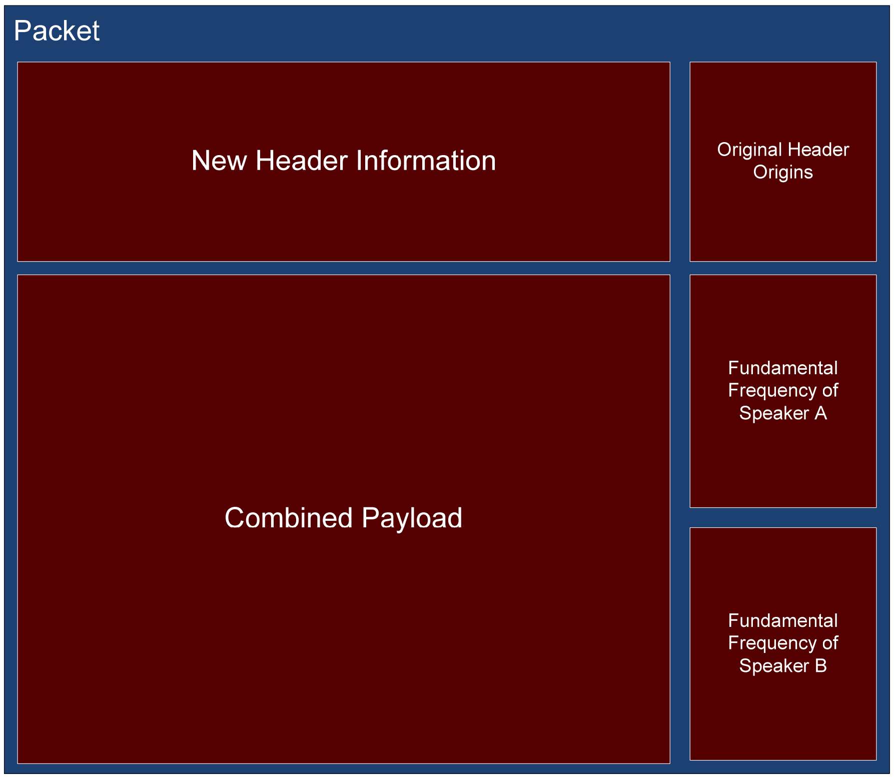
Packet
New Header Information
Original Header Origins
Combined Payload
Fundamental Frequency of Speaker A
Fundamental Frequency of Speaker B

Figure 3: Combination packet

In essence, the algorithm uses differences in the speakers' voices to split the signal into its respective parts. By finding this distinguishing information and sending it in the packet prior to combination, errors induced through other speaker's interference is avoided. This piece of information is the fundamental frequency of the speakers, which each take up two bytes of bandwidth.

The "original header origins" contains the header information of the original two packets. Assume Andrew is calling from Toronto and Edward from Vancouver with fundamental frequencies F_A and F_E, respectively. If this portion of the packet is "Toronto Vancouver", then the fundamental frequencies will also be ordered "F_A F_E". In other words, the order of the headers is the same as the fundamental frequencies.

Hence, the size of a combination packet will be larger than a normal packet. This factor will help distinguish between the two for the receiver.

3. Simulation System

The prototype system has a few differences from the system ready for production. Among this multitude, two drastically change the way this system will be built

- Only one Ethernet port is available on the prototype system
- The whole prototype has only one VBSS

The first problem changes our coding structure, because only one Ethernet MAC is used. Hence, the system is no longer single flow (in either one direction or the other). The second problem, however, forces a change in our flowchart (Figure 4).

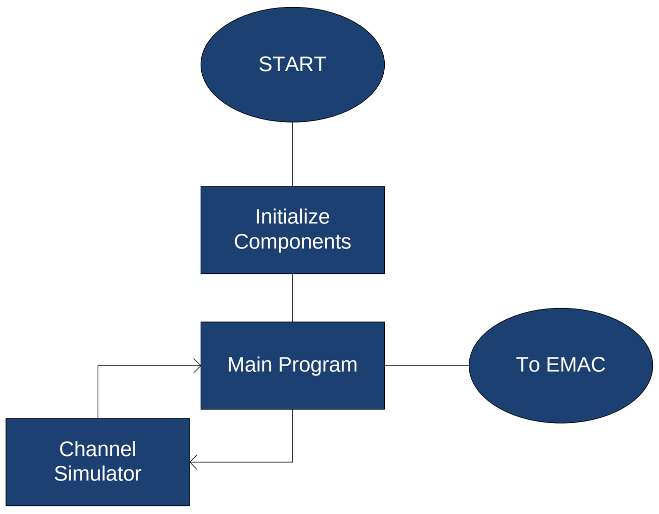

Figure 4: Flowchart with Simulator

Instead of sending all packets to the Ethernet MAC, packets that come from the Transmitter should be rerouted through a channel simulator and back into the main program (Receiver).

Architecture Overview

In this section, we will be focusing on the design for our main system architecture, which is broken down into three main modules: external peripherals, hardware and software. External peripherals refer to the devices that can only be switched on or off but the actual functionalities cannot be changed; this includes external DDR memory and Ethernet Interface. On the other hand, hardware includes any subsystems that are synthesize within a FPGA chip and also can be modified through VHDL. Finally, software is compiled to be run on a hardware processor core, and can be written using either assembly language or high-level language. For our prototype, we will be implementing these modules on a XUP-V2P development board. Figure 5 below shows the big picture of our architecture.

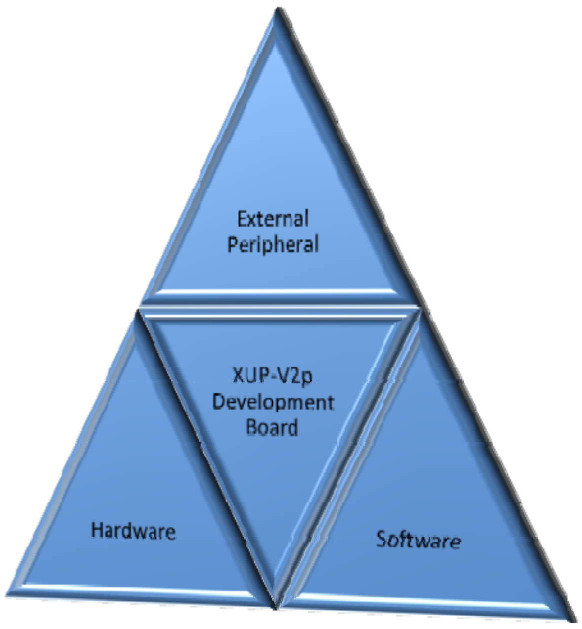

Figure 5: Architecture of VBSS Prototype

Not only XUP-V2P development board is very affordable to students, it also carries some of the key features that are necessary to implement VBSS, including:

- Virtex-II Pro XC2VP30 containing 13,969 slices, 428 Kb of Distributed RAM, 136 Multiplier block.
- 2 PowerPC 405 RISC Cores that can be activated within Virtex-II chip.
- Fast Ethernet Interface, which supports full duplex operation at 10 Mb/s and 100 Mb/s.
- DDR RAM module that can support up to 2GB of DDR RAM.
- Four user LEDs, four user switches, and five user push-buttons.

A top view of XUP-V2P is shown in Figure 6.

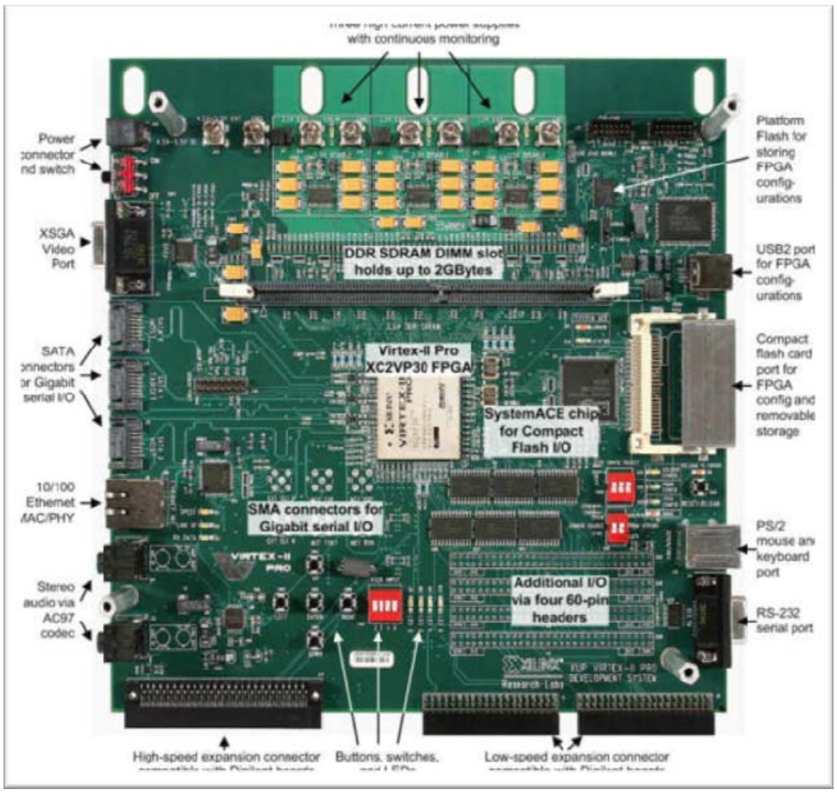
with continuous monitoring
Power connector nd switch
XSGA Video Port
SATA onnectors or Gigabit serial I/O
10/100 Ethernet MAC/PHY
Stereo audio via AC97 codec
DDR SDRAM DIMM slot holds up to 2GBytes
Virtex-II Pro XC2VP30 FPGA
SystemACE chip for Compact Flash I/O
SMA connectors for Gigabit serial I/O
Additional I/O via four 60-pin headers
Platform Flash for storing FPGA config-urations
USB2 port for FPGA config-urations
Compact flash card port for FPGA config and removable storage
PS/2 mouse and keyboard port
RS-232 serial port
High-speed expansion connector
Buttons, switches,
Low-speed expansion connector

Figure 6: A Top View of XUP-V2P

It is possible to implement our whole system with only the hardware and the external peripheral. However, one major advantage of using software is the short development time. Since we have a tight schedule to implement our prototype, we have allocated a larger portion of VBSS system logic into software, while only leaving time-crucial component to the hardware.

1. Hardware and External Peripherals

Our hardware architecture features a single PowerPC 405 processor clocked at 300MHz, a high-frequency Processor Local Bus (PLB), and a low-frequency On-chip Peripheral Bus (OPB). Figure 7 shows the overview of our hardware and external peripherals. Note that we place fast function blocks such as FFT/IFFT, G.711 Codec, memory and EMAC controller on the fast PLB. On the other hand, we have General Purpose Input/Output (GPIO) block for accessing user LEDs and switches on the much slower OPB.

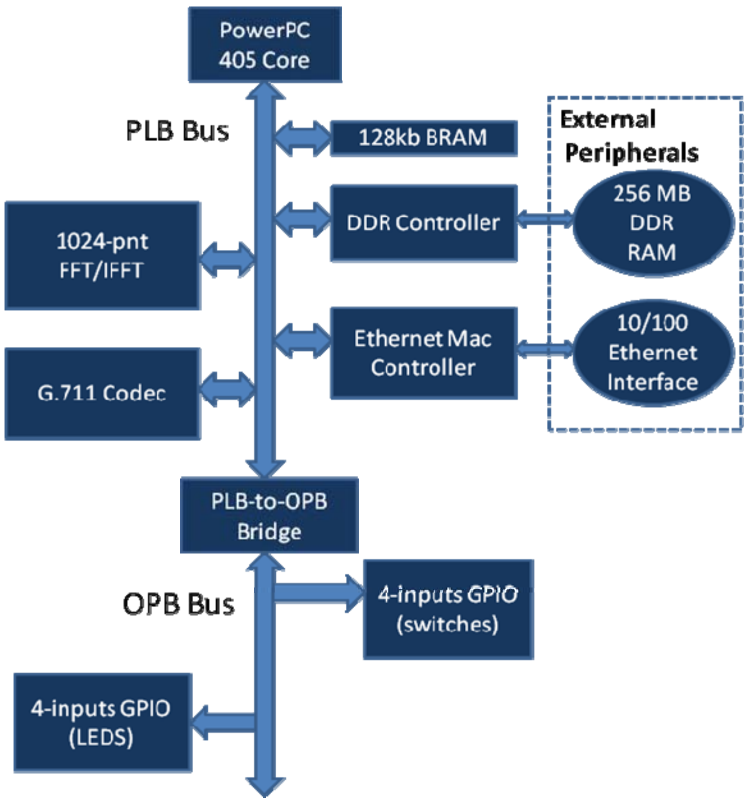

Figure 7: Overview of Hardware and Software

2. Software

The main functionality of our software system is to place control on the VBSS. The software algorithm will be written in high level language and then is compiled to be run on top of the PowerPC 405 processor. The main algorithm is a cycle that goes through five stages as shown in Figure 8.

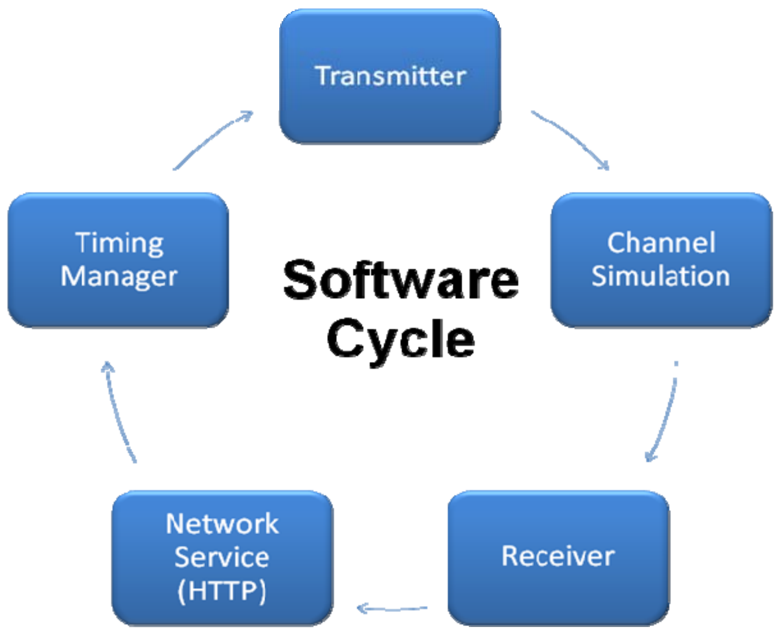

Figure 8: Software Cycle of VBSS

Transmitter detects the incoming phone packets and combines pairs of packets when necessary. It then sends packets to the Channel Simulate stage, where four hazardous environment properties would occur: delay, corruption, jittering and loss. Finally, packets arrive at the Receiver and then are forwarded to the external phones.

Network services include our HTTP server that handles user request from our user-interface webpage. The last component, Timing Manager simply calculates the time elapsed for the current cycle, which is especially useful for the stages that require synchronization. We will be discussing the transmitter, receiver and network service in greater details in the later sections.

Receiver

When a voice packet enters the receiver, the receiver must distinguish if it is a combination packet or a regular voice packet. If a regular voice packet enters, no processing is necessary, and the packet is routed to the next stage for packet processing.

However, if the packet is a combination packet, a series of computations occurs, as shown in Figure 9.

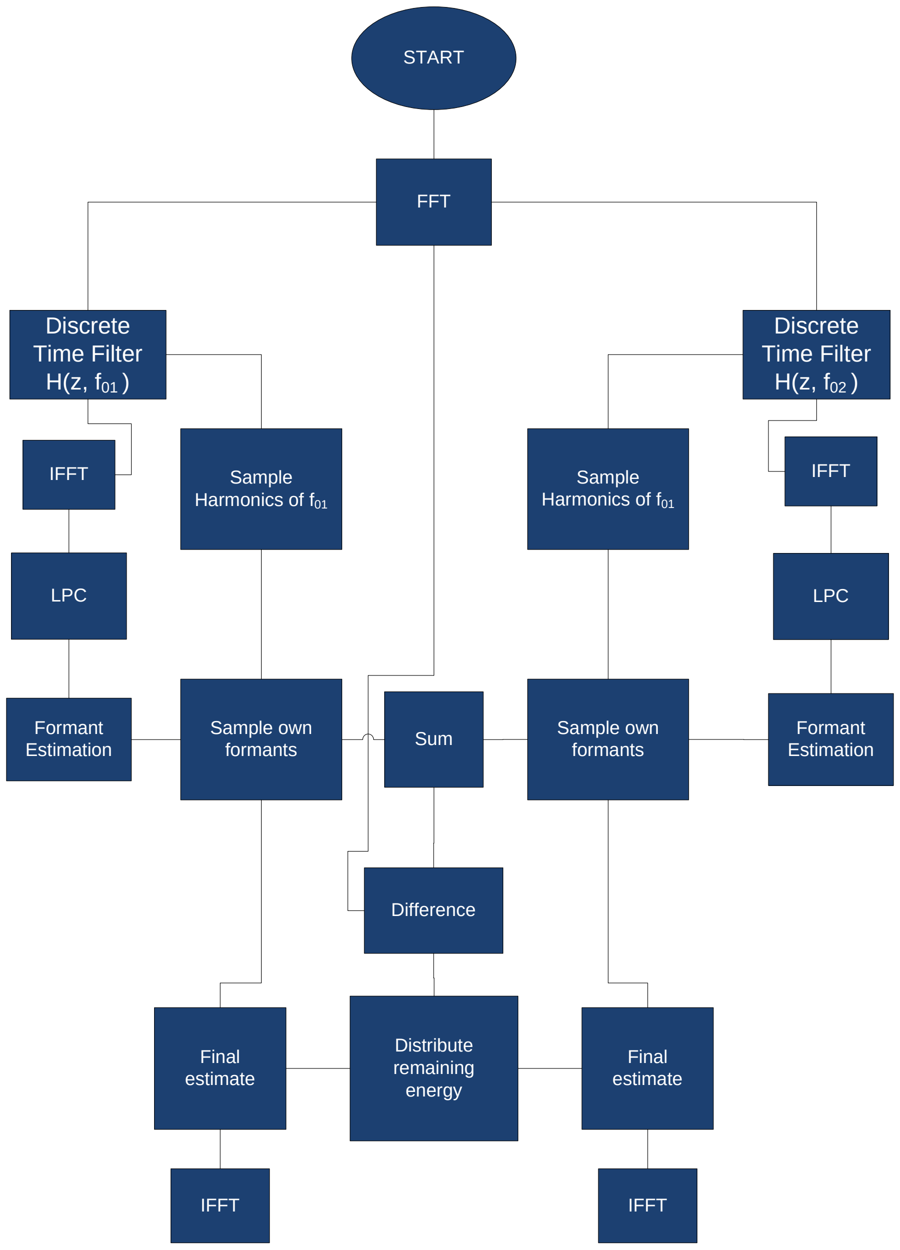
START
FFT
Discrete Time Filter H(z, f_{01})
Discrete Time Filter H(z, f_{02})
IFFT
Sample Harmonics of f_{01}
Sample Harmonics of f_{01}
IFFT
LPC
LPC
Formant Estimation
Sample own formants
Sum
Sample own formants
Formant Estimation
Difference
Final estimate
Distribute remaining energy
Final estimate
IFFT
IFFT

Figure 9: Receiver Flow

This diagram is essentially symmetric, as is the case when both voices have a strong fundamental. Packets coming from the transmitter, however, must have at least one speaker voiced. In that case, only one side of the flowchart will be performed, and remaining energy will be allocated to the other unvoiced speaker.

3. Discrete Time Filter

This block represents a discrete filter described by

$$H(z) = \left(1 + \alpha z^{-f}\right)/(1 \cdot \tag{1}$$

This discrete filter [1] needs to be evaluated at

$$z = e^{-j\omega} = \cos(\omega) - j\sin \tag{2}$$

where *f* is the fundamental frequency of the speaker, and *α* is a scaling factor. For this equation, a lookup table is needed for the sinusoidal real and imaginary components. Hence, a lookup table for a normalized sinusoidal function will be used.

This lookup table will contain values for *sin(ϑ)* between *0* and *2π* on increments of every *π/1024*. This resolution is required because the normalized function will need to be scaled for larger frequencies, as shown in Figure 10.

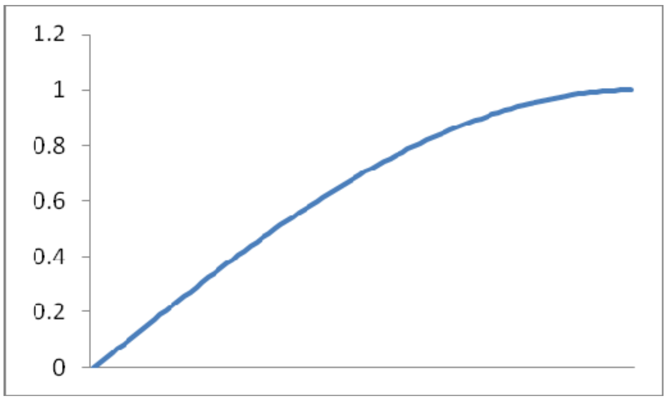

Figure 10: Sine (quarter period)

4. LPC

A 10-order LPC will be implemented based on Schur's algorithm [2] for finding reflection coefficients. The algorithm (Figure 11) is recursive and has a primarily quick hardware implementation.

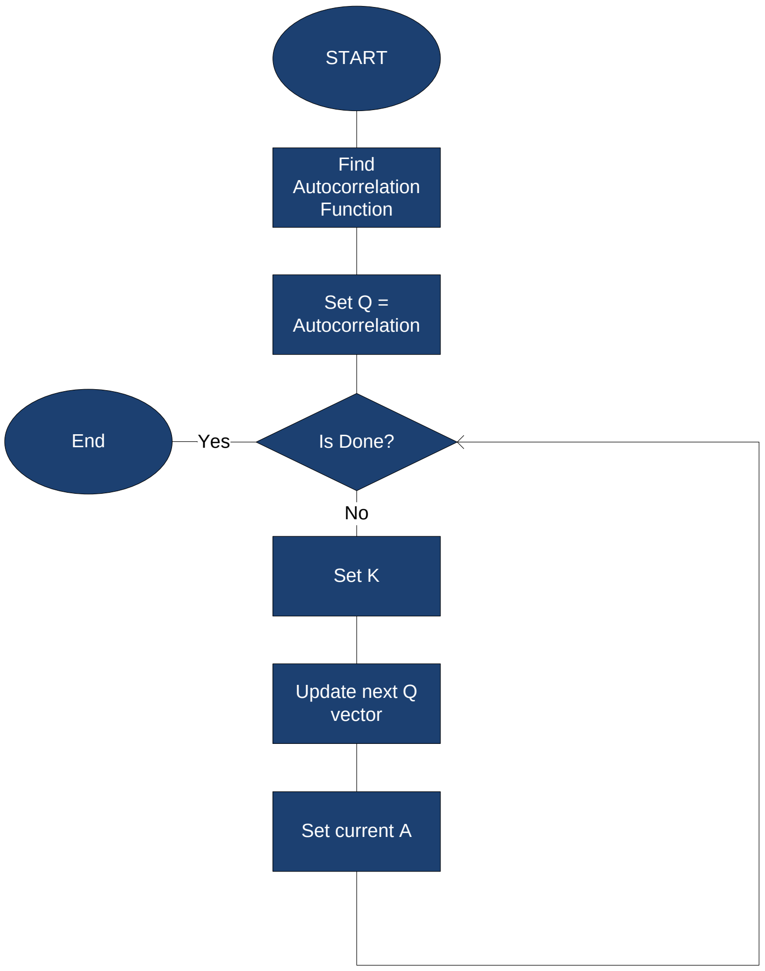

Figure 11: LPC Algorithm

The one drawback to this algorithm is its use of division on every cycle. For an LPC of order 10, 10 divisions would take place. Divisions are computationally expensive, and as a result, become the bottleneck for the speed of this algorithm. However, by normalizing the autocorrelation function, all divisions become fixed point, and therefore, less computationally expensive.

5. Formant Estimation

Formant information is present in the LPC as the roots of the equation. However, root solving is too computationally expensive. Fortunately, formants are also presented as peaks of the frequency response of the LPC derived filter, which use the coefficients as

$$\mathrm{LPC}(z) = \sum a \qquad (3)$$

In this way, a pole-zero pair is a formant, and will show as a peak in the frequency response. In order to magnify peaks, the pole-zero pair must be moved closer to the unit circle by a factor as

$$z^{*} = 0.99z \qquad (4)$$

Transmitter

The transmitter is responsible for handling incoming phone packets and performing compression before forwarding them to the respective destination. Figure 12 shows a general system diagram for the transmitter. The incoming voice packets are estimated to have regular 20ms intervals. Depending on number of incoming caller, transmitter will be creating several packet buffers that stores two previous packets from the same caller. The pair-wise synchronizer tries to synchronize pairs of caller by delaying one caller's packet for at most 10ms. Later, two independent voices, each with 60ms duration, will be passed into the Maximum Likelihood Pitch Detector (MLPD) to calculate the fundamental frequency.

If the MLPD output a non-zero value, the voice segment is considered as "Voiced", which means it can be characterized by a low frequency between 50Hz to 200Hz. On the other hand, a zero value represent "Unvoiced", which means most of the energy in the signal in distributed among the higher frequencies. If the pair contains one or two "Unvoiced", they will not be combined. For other cases, we will be combining packets and send the result to the output buffer. The algorithm for MLPD is shown in Figure 13.

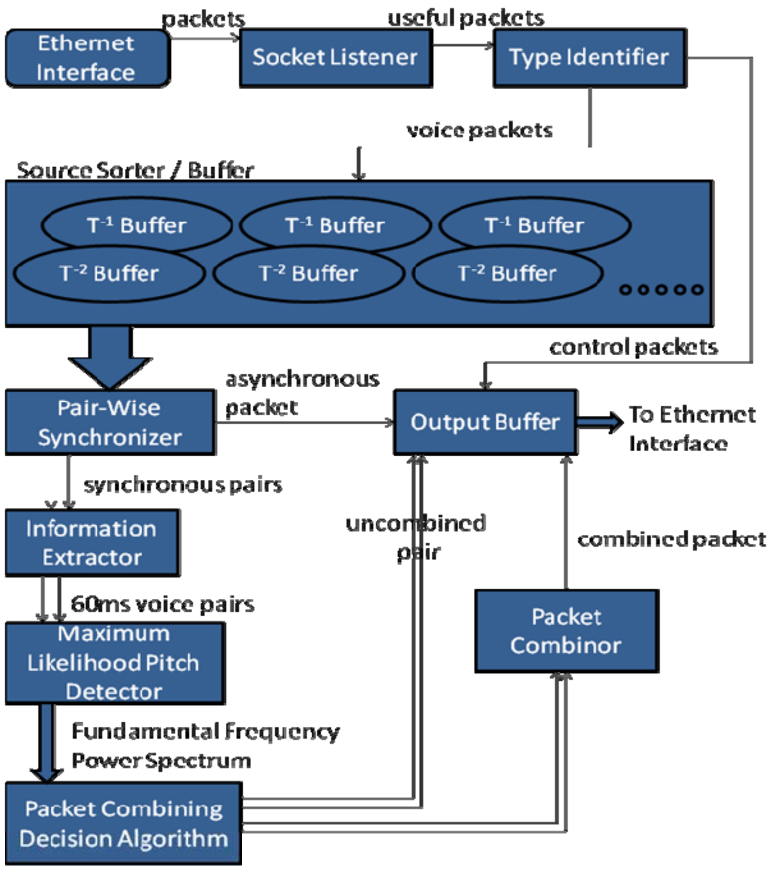

Figure 12: System diagram for transmitter

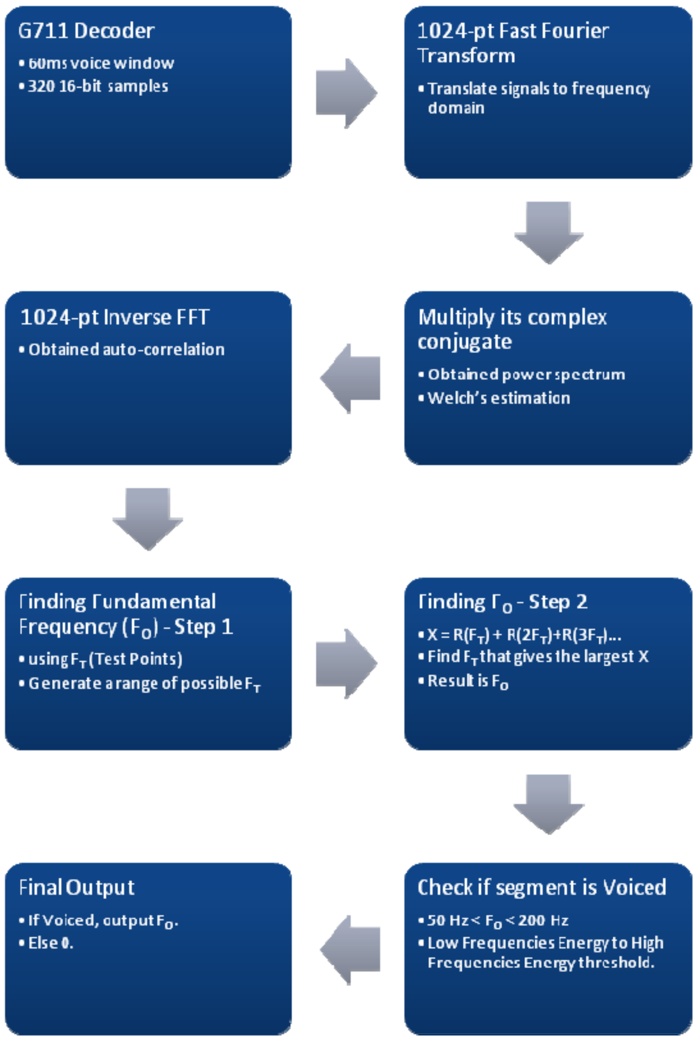
G711 Decoder
• 60ms voice window
• 320 16-bit samples
1024-pt Fast Fourier Transform
• Translate signals to frequency domain
Multiply its complex conjugate
• Obtained power spectrum
• Welch's estimation
1024-pt Inverse FFT
• Obtained auto-correlation
Finding Fundamental Frequency (F_0) - Step 1
• using F_T (Test Points)
• Generate a range of possible F_T
Finding F_0 - Step 2
• $X = R(F_T) + R(2F_T) + R(3F_T)$...
• Find F_T that gives the largest X
• Result is F_0
Check if segment is Voiced
• 50 Hz < F_0 < 200 Hz
• Low Frequencies Energy to High Frequencies Energy threshold.
Final Output
• If Voiced, output F_0.
• Else 0.

Figure 13: Maximum Likelihood Pitch Detection Algorithm

Network

The VBSS System is in constant communication with a vast network of different components ranging from phones, to servers, to switches, to routers. For the production system, HTTP and VOIP ports, and a VOIP control server are required. For the prototype, an additional ARP poisoning mechanism is required.

1. HTTP Server

The HTTP server was designed specifically for the Web, and employs a protocol that supports sending documents from the server to a browser, as well as sending complex data from the client back to the server. Our project uses the FPGA as an HTTP server for configuration and diagnosis purposes. Example pages are shown below (Figure 14)

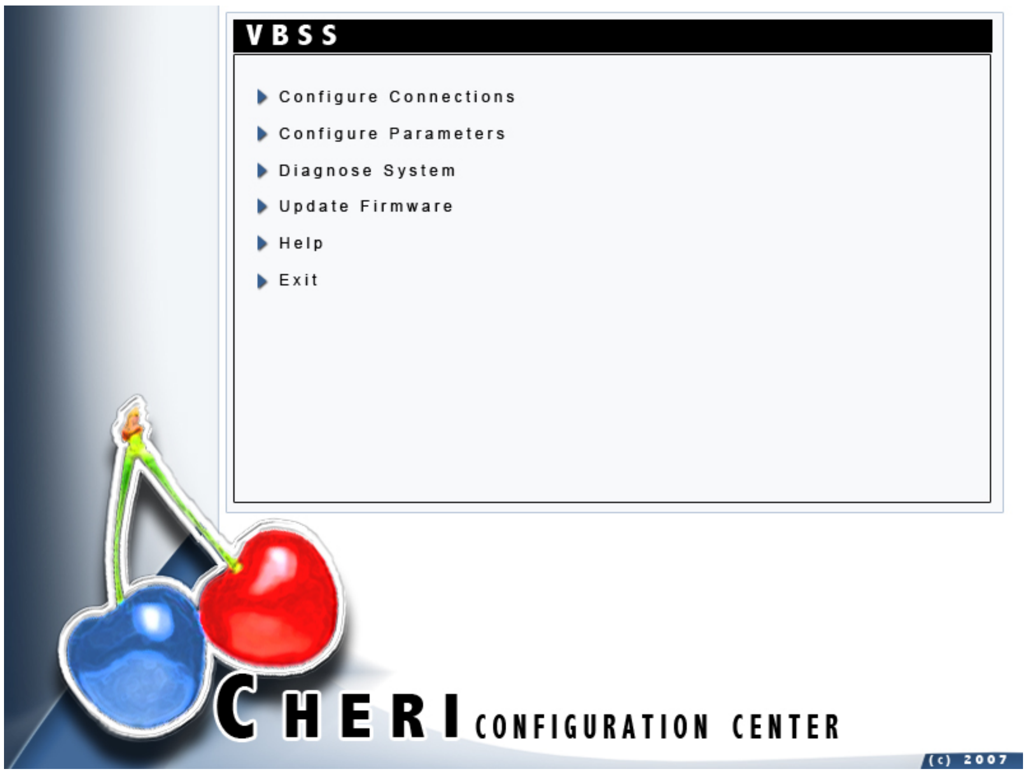

Figure 14: HTTP Entry Page

2. TFTP Server

TFTP transfers files to and from a remote computer. Many files need to be loaded onto the RAM open startup of our system. These would include HTTP pages and images as well as different configuration files. Note that in a real production environment, TFTP will not be required. However, since no ROM is available on-chip, TFTP will be required for the prototype.

3. Linux running SipX

SipX will be installed on a system running *Fedora Core*. *SipX* is responsible for controlling communication between VoIP phones on the network, while *Fedora* is responsible for IP and DNS. To simplify procedures, the IP address and domain name of each of the four phones, as well as the server and FPGA board, will remain static.

4. ARP

Arp-Poisoning server must be able to perform Man-In-The-Middle (MITM) attack between any two of the phones for our project. The Arp-Poisoning server is necessary to allow us to implement our system on the one-Ethernet-port FPGA board.

Normally, a phone will contact another phone directly. However, Arp-Poisoning will confuse each phone to call the FGPA board. This board will then act as a tunnel to another phone (connection is hardcoded). Hence, each phone for the prototype will be linked to another phone.

Test Plan

Our team is currently divided into two sections.

1. The first section sets up runtime environments
2. The second develops code for different sections of the project

Hence, two different test categories must be executed, one pertaining to each group above. Unit tests of each category will be executed concurrently with development. Additionally, these tests only correspond to the prototype system.

1. Runtime Environment

The runtime environment includes *Fedora Core, SipX* Server, *Sip* compliant phones, and the packet routing process.

Runtime Test Case #1

Requirement:

Sip-X server must run smoothly on the Linux system, and it shall allow external phones to register with the server.

Procedure:

1. Inspect the status of each service under *Sip-X* server.
2. Connect four Grandstream phones and the *sip-X* server box to the 8-port switch.
3. Ping other devices on the network from the sip-X server box.
4. Using the Sip-X management interface, add four users and assign each of them a unique phone number.
5. Modify the phone profiles with correct IP of related servers.
6. Restart the Grandstream phones manually.
7. Wait for the phones to restart, and check whether they have downloaded new profile information from *Sip-X* server.
8. Test the phone by checking if it could accept incoming calls and also could make outgoing calls.

Expected Result:

After *Sip-X* server is started up, executing "/sbin/service sipxpbx" should display 'Ok' for each sub-services under *Sip-X*. User should be able to ping all external devices from *Sip-X* server through the 8-port switch. Grandstream phones should be able to download new profiles from *Sip-X* server, and are able to register with the *Sip-X* server. Each Grandstream phone should be equipped with a unique phone number, and calls can be made by all phones to any remaining phones.

Runtime Test Case #2

Requirement:

Arp-Poisoning server must be able to perform Man-In-The-Middle (MITM) attack between any two of the Grandstream phones. The Arp-Poisoning server is necessary to allow us to implement our protype system on a one-Ethernet-port FPGA board.

Procedure:

1. Connect four Grandstream phones and a *Sip-X* server to the 8-port switch.
2. Let the phones register with the *Sip-X* server
3. Check whether the Grandstream phones can receive and make calls.
4. Now connect the *Arp-Poisoning* server to the 8-port switch.
5. Arp-poison one pair of phones.
6. For the pair selected at step (5), make a call from one of the pair to another of the same pair.
7. From the *Arp-poisoning* monitor, check the traffic status and see if it is able to capture the voice packet.
8. Un-poison the phones.
9. Repeat step 6 to see if phones can function normally.
10. Go back to step 5 with another new pair.

Expected Result:

The *Arp-poisoning* server should be able to perform MITM attack and intercept the voice data between any two phones. From the monitor screen, user should find a complete session of recorded voice conversation. The phones should also function normally after the *Arp-poisoning* server release the poison from the phones.

2. VBSS System

The VBSS system has many components. Each component is described in the **System Overview**.

System Test Case #1

Requirement:

FPGA must be able to connect to the network as an HTTP server. FPGA must be able to send and receive packets. HTTP server operations should not be hindered by, or hinder, its operation as a compressor/expander.

Procedure:

1. Add the FPGA into the DNS list and create static IP
2. Program the board with example code from the development CD
3. Run the program on the board
4. Ping the FPGA, awaiting a return packet
5. Enter IP address or domain name of the board into all web browsers
6. Navigate through all pages
7. Call from a phone while navigating through pages
8. Ascertain that additional delay does not occur

Expected Result:

Pages should appear on the web browser as created.

System Test Case #2

Requirement:

Components must be able to communicate with on-chip microprocessors

Procedure:

1. Add VHDL code for G711 Codec
2. Modify HTTP Server to communicate with the peripheral
3. Modify HTTP Server to send textbox information from the browser page into the peripheral
4. Modify HTTP Server to show the output of the peripheral in the browser page
5. Run the modified HTTP Server
6. Input different test variables

Expected Result:

A dynamic output appears on the web browser

System Test Case #3

Requirement:

Microprocessor should send information to the correct component

Procedure:

1. Create a shell for the peripherals
2. Create internals that set the output to equal the input
3. Modify microprocessor to communicate with each of the peripherals in an order
4. Modify HTTP Server pages to receive more than one input for different required parameters for different conditions
5. Modify HTTP Server to output the signals between the different peripherals
6. Run the microprocessor
7. Input different test variables

Expected Result:

The components are stringed together in a specific order. This order is dependent on the type of packet that enters the microprocessor. If the packet is a non-voice packet, it should be sent through immediately. If the packet is a voice packet, the microprocessor should send it to packet processing. Then, concurrently, the voice data will be passed to voice processing, and the information data will be passed to information passing. When finished, both pieces of information will be passed back to packet processing, and back to the microprocessor.

Conclusion

Cheri Perception healthily growing with this innovative and helpful product. Our product helps many companies to achieve the goal of having a complete and convenient communication system without restrictions over long distance range and at the same time maintain a low cost.

We believe by April 2007, a demonstration of our product will be able to perform the designed functionality.

Sources and References

[1] *Maximum Likelihood Pitch Estimation.* **Wise, James D.** 5, s.l. : IEEE Transactions on Acoustics, Speech, and Signal Processing, 1976, Vols. ASSP-24.
[2] *Cochannel Speaker Separation by Harmonic Enhancement and Suppression.* **Morgan, P. David, et al.** 5, s.l. : IEEE Transactions on Speech and Audio Processing, 1997, Vol. 5.
[3] **Parsons, Thomas.** *Voice and Speech Processing.* New York : McGraw-Hill Inc, 1987.
[4] Deloitte Touche Tohmatsu. [Online] 2004. www.deloitte.com/dtt/research/0,1015,cid=64018&pv=Y,00.html.
[5] **Davidson, Jonathon, et al.** *Voice over IP Fundamentals.* Indianapolis : Cisco Press, 2007.
[6] **Markel, J.** *Linear Prediction of Speech.* New York : Springer-Verlag Inc, 1976.
[7] *IEEE Xplore.* [Online] IEEE, 2006. http://ieeexplore.ieee.org/.
[8] G.7xx Audio (Voice) Compression Protocols (CODEC) Overview. *Javvin network management and security.* [Online] Javvin Technologies Inc., 2005. http://www.javvin.com/protocolG7xx.html.
[9] G.711. *Wikipedia: The Free Encyclopedia.* [Online] Wikimedia Foundation, Inc., 2007. http://en.wikipedia.org/wiki/G.711.