Leveraging an Open-Source VoIP Application to Speed VoIP Product Development

Scott Kurtz
Adaptive Digital Technologies
Agenda

• Overview – What are we trying to accomplish? Leverage Technology
• Brief view of traditional telecom
• Voice-over IP (VoIP)
• The Open Source Business Model
• Introducing G.PAK™: An open source VoIP DSP software application
• Hands-on Examples
Leveraging Technology

• It is difficult enough to keep up with the pace of complexity in technology, let alone make use of it.

• How do we make use of very complex technology?
  – We stand on the shoulders of those who preceded us.
Overview
Semiconductor Complexity

Transistors per IC

Year


1K 10K 100K 1M 10M
Semiconductors: Managing Complexity

• IC Design
  – 1960s and 1970s: transistors
  – 1970s and 1980s: gates
  – 1990s-Present: VHDL, functional blocks, cores, etc.

• Can you imagine designing a DSP or Pentium chip from transistors?
DSP Application Complexity

Lines of Code vs Year

- 1980
- 1990
- 2000
- 2010

- 1K
- 10K
- 100K
- 1M
DSP Application: Managing Complexity

• DSP Software
  ▪ 1970s: Hardware Implementations and Microcode
  ▪ 1980s: Assembly Language Coding
  ▪ 1990s: High-Level Language (“C”) Coding, Libraries, Operating Systems
  ▪ 2000s: Application-Specific Signal Processors (ASSP)

• Can you imagine writing a complete VoIP DSP application in Microcode or assembly language code?
DSP Applications: Today’s Options

• Start From Scratch
• Leverage DSP Libraries
• Leverage Operating System
• Leverage Software Framework
• Leverage ASSPs
• Leverage Open Source Software
Leveraging Proven Algorithms

- Voice Quality – the bottom line
- Field hardened algorithms
- Voice Quality related algorithms:
  - Echo Cancellation
  - Conferencing
  - Automatic Gain Control (AGC)
  - Packet Loss Concealment (PLC)
  - Discontinuous Transmission (DTX)
DSP Applications: Today’s Options

Operating System

Application Software

Algorithms

Drivers

ASSP

TI DSP

TI Developer Conference

Minds in Motion

Technology for Innovators™

Texas Instruments
Introducing Open-Source DSP

• Open-Source DSP lies between the ASSP and custom development.
• With open-source, you can
  ▪ Leverage existing software
  ▪ Focus on your application
  ▪ Keep control of your design
  ▪ Hit the market quickly with your product
Cost and Flexibility

What about opportunity cost?
Brief View of Traditional Telecom
The PSTN
(Public Switched Telephone Network)
Digital Channels

PSTN

10010010111010010001
0010010111010010011
10010010111010010001
0010010111010010011

10010010111010010001
0010010111010010011

10010010111010010001
0010010111010010011

10010010111010010001
0010010111010010011

Technology for Innovators\textsuperscript{TM}

Texas Instruments
Compressed Data

PSTN

01001001011101
00100101110100
1001001011101001001
0010010111010010011
10010010111011
10010010111010
01001001011101
10010010111010

00100101110100
10010010111010

010010010111101001001
0010010111010010011
0010010111010010011

Technology for Innovators™
# A Few Voice Compression Standards

<table>
<thead>
<tr>
<th>Product</th>
<th>MOS</th>
<th>Rate(s)</th>
<th>MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711</td>
<td>4.1</td>
<td>64</td>
<td>0.36</td>
</tr>
<tr>
<td>G.726 (Low Mem)</td>
<td>3.9 (32 kbps)</td>
<td>16, 24, 32, 40</td>
<td>11.0</td>
</tr>
<tr>
<td>G.726 (Low MIPS)</td>
<td>3.9 (32 kbps)</td>
<td>16, 24, 32, 40</td>
<td>9.8</td>
</tr>
<tr>
<td>G.729</td>
<td>3.9</td>
<td>8</td>
<td>21.5</td>
</tr>
<tr>
<td>G.729B</td>
<td>3.9</td>
<td>8</td>
<td>22.2</td>
</tr>
<tr>
<td>G.729A</td>
<td>3.7</td>
<td>8</td>
<td>11.4</td>
</tr>
<tr>
<td>G.729AB</td>
<td>3.7</td>
<td>8</td>
<td>12.2</td>
</tr>
<tr>
<td>G.728</td>
<td>3.7</td>
<td>16</td>
<td>35.0</td>
</tr>
<tr>
<td>G.723.1</td>
<td>3.9/3.6</td>
<td>6.3/5.3</td>
<td>16.9/16.5</td>
</tr>
<tr>
<td>G.722</td>
<td></td>
<td>48, 56, 64</td>
<td>15.6</td>
</tr>
</tbody>
</table>
PSTN Functions

- PSTN Functions
  - Call Setup/Teardown
  - Pass Signaling (next slide)
  - Pass Voice, Fax, Modem Signals
Signaling

- Dual-Tone Multi-Frequency (DTMF) Dialing (or Touch-Tone Dialing)
- Pulse Dialing
- Hook-Switch Signaling
- Ringing
- Hook Flash
- Disconnect
- Many others
Transition to VoIP

• Signaling – must be captured and sent in packets. Why?
• Tonal Signaling – may need to be captured and sent in packets. (Next Slide)
• Modem Signals – may need to be relayed (T.38 Fax Relay)
• Voice – G.711 PCM or further compression
Need for Tone Relay

FFT Before Speech Compression

FFT After G.729A Compression (8 kbps)
PCM = Pulse Code Modulation - 8000 Samples/Sec, 64 kbps

Relay: Tone or Fax
VoIP
VoIP

ATA / Gateway
Voice: Analog
Signaling: DC

Packet Network
Voice: G.711 PCM or Other Packets
Signaling: Packets
Fax: T.38 Packets

ATA / Gateway
Voice: Analog
Signaling: DC

Minds in Motion
Technology for Innovators™
Texas Instruments
Voice Over Data Networks

• Why carry voice over data networks?
• PSTN vs. Data Networks
• Integrating voice into data networks
Traditional Voice And Data Networks

**PSTN**
- Circuit Switched
- Very Low Delay
- Continuous Data Flow

**Data Network**
- Packet Switched
- Higher Delay
- Bursty Data Flow
- Throughput and delay depend upon network activity
Traditional Voice and Data Networks

- Circuit Switching - A circuit is dedicated to a specific communication link for the duration of the call.
- Packet Switching - Packets of data sent from user A to user B are not necessarily carried on the same physical circuit; allows for sharing of circuits.
Voice Over Data Network
Voice Over Data Network

IP PBX/Gateway

Router

Data Network

Router

IP PBX/Gateway

Minds in Motion
Why Carry Voice on Data Networks?

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share Hardware Costs</td>
<td>Network Impairments</td>
</tr>
<tr>
<td>Share Access Costs</td>
<td>Security?</td>
</tr>
<tr>
<td>Reduce Long Distance Charges</td>
<td></td>
</tr>
</tbody>
</table>

Advantages: Share Hardware Costs, Share Access Costs, Reduce Long Distance Charges

Disadvantages: Network Impairments, Security?
Issues With VoIP

• Types of errors
  ▪ Bad packet received
  ▪ Missing packet
  ▪ Packet received out of order

• Recover Mechanisms
  ▪ ACK/NACK
  ▪ Sequence Numbers
  ▪ Packet Retransmission
  ▪ Timers
  ▪ Drop packet on floor
Error Recovery – Why Not?

• By the time an error is detected and recovered, it’s too late to play out the voice.
• RTP (Real-time Transport Protocol) does not implement error recovery.
• RTP does handle re-sequeuing out-of-order packets and detecting lost or late packets.
• RTP may be used with diversity.
Packet Transmission Errors

IP Tx
0 10 20 30 40 50 60 70 80 90

IP Rx
0 10 20 40 60 50 70 80 90

RTP Out
0 10 20 40 50 60 70 80 90

PLC Out
0 10 20 30 40 50 60 70 80 90
Integrating Voice into Data Networks

• Traditionally, voice has been transmitted over circuit switched networks with constant data rate and delay.
• Data networks introduce many undesirable characteristics that need to be addressed:
  - Blocking
  - End-to-end delay
  - Delay jitter
  - Packet Loss
  - Header overhead
  - Throughput
Integrating Voice into Data Networks

• Desired Features for Packet Voice Implementation
  ▪ Compression
  ▪ Silence compression (a.k.a. DTX)
  ▪ Jitter Compensation
  ▪ Packet Re-sequencing
  ▪ Packet Loss Concealment
  ▪ Transmission of Signaling
  ▪ Echo Control
VoIP Summary

• Data networks are not inherently suited to voice transmission. Data networks are packetized and bursty in nature. Voice signals are more continuous in nature.

• Special care must be taken when integrating algorithms to minimize the effects of the network.
The Open Source Business Model

• Open Source does not necessarily mean free software
• GPL (GNU General Public License)
  ▪ The most widely known open-source license.
• Other Open-Source Licenses

  Academic  Microsoft®
  Apache    MIT
  Apple®    Mozilla®
  BSD       NASA
  CPL       Nokia®
  Eclipse™  Python®
  Intel®    RealNetworks
  Lucent    Sun™
  Many More
GPL - The Up-Side for a Vendor

- Market Penetration
- Become the de facto standard
- High Visibility
- Sell additional services
- Sell associated hardware
- License associated software
- Many customers will pay to license software to obtain support and to avoid GPL restrictions
GL - The Down-Side for a Vendor

• Enables competitors
• Exposes intellectual property / trade secrets
• Code Contamination
GPL - The Up-Side for a Vendor

• No need to reinvent the wheel
• Leverage existing code that hopefully already works
• Faster time to market
GPL – The Down-Side for a Developer

• You must open up your source
• Enables competitors
• Exposes intellectual property / trade secrets
GPL - The Up-Side for Users

• Free software is available
GPL - The Down-Side for Users

- Free software comes with no warranty and no support. Proceed at your own risk.
- Free software tends to be light on documentation.
G.PAK Source Licensing

- G.PAK license is not under GPL. Why Not?
  - G.PAK framework source code is not redistributable (protects our IP)
  - Derivative works are therefore not required to be redistributed (protects your IP)

- Unpaid license comes without support
G.PAK Source License

• You **Must**:  
  - Use G.PAK source code in conjunction with Adaptive Digital’s algorithm libraries

• You **May Not**:  
  - Redistribute G.PAK Source Code
G.P.AK: An Open Source VoIP DSP Framework

- G.P.AK Overview
- G.P.AK Channel Types
- Interfacing with G.P.AK via host
- Host supplied routines
- G.P.AK Internals
- Modifying G.P.AK with new Vocoders
G.PAK Features

- Most Standard Vocoders
- Multiple Channel Types (PCM-PKT, PCM-PCM…)
- Conferencing
- Echo Cancellation (PCM and Packet)
- Voice Activity Detection/Comfort Noise
- Tone Detection and Generation (DTMF, MFR1, MFR2 FORWARD, MF2 BACK, CALL PROGRESS)
- RTP/AAL2 Payload Formats
- Self Contained DSP Code
- Host API routines
G.PAK - Overview

• Algorithms vs. Turnkey
  - Algorithms – write your own framework
    Turnkey – No DSP Programming
  - Turnkey – Shorter Time To Market
    Lower NRE
  - Turnkey + Open Source – Increases flexibility and design/product control
Algorithms vs. G.PAK

Customer SW

ADT SW

Control Processor
With API to access DSP

Control

Scheduling

Buffering

DSP

Algorithm A

Algorithm B

Buffering

Multi-Channel Input

Multi-Channel Output
G.PAK vs. Algorithms

Customer SW
ADT SW

Control Processor
With
API to access DSP

DSP
Control
Scheduling

Buffering → Algorithm A → Algorithm B → Buffering

Multi-Channel Input

Multi-Channel Output
G.PAK Channel Types
PCM to PCM (TDM to TDM) Channel

Control

T-1 Framer

McBSP

TI DSP

HPI

T-1 Framer

T-1 Channel Bank

1

24

Minds in Motion

Technology for Innovators™
G.PAK: PCM to PCM Channel

Input Serial Port/Slot 1

8 kHz PCM In

PCM Frame Buffer

PCM Echo Cancel

PCM Frame Buffer

8 kHz PCM Out

Output Serial Port/Slot 2

8 kHz PCM Out

PCM Frame Buffer

PCM Echo Cancel

PCM Frame Buffer

8 kHz PCM In

Input Serial Port/Slot 2
Example: PCM Echo Cancellation
PCM – Packet Channel: VoIP Gateway
G.PAK: PCM to Packet Channel

Serial Port/Slot → 8 kHz PCM In → PCM Frame Buffer → PCM Echo Cancel → AGC → VAD → Silence → Tone Detect → Tone → Packet Out → Host

Serial Port/Slot → 8 kHz PCM In → Packet In Codec Encode → Tone Gen → Tone → Packet In → Host

Serial Port/Slot → 8 kHz PCM Out → PCM Frame Buffer → Packet Echo Cancel → Bulk Delay → CNG → Silence → Packet In Codec Decode → Audio → Packet In → Host
G.PAK: Packet to Packet Channel
G.PAK: PCM to Conference Channel

Input Serial Port/Slot → 8 kHz → PCM In → PCM Frame Buffer → Echo Cancel → AGC → Conference n → PCM In n → PCM of Other Conference Channels → PCM Out n → PCM Frame Buffer → 8 kHz → PCM Out → Output Serial Port/Slot
GPAK: Packet to Conference

Diagram showing the flow of audio and packets from Host to Conference, including stages for Tone Gen, VAD, Packet In/Out, Codec Encode/Decode, Silence, and Echo Cancel. PCM Frame Buffer and AGC are also indicated.
Host Interface

• Configuration Messages
• Payload Transfer
• DSP Download
• CPU Usage
Host Interface: Layers

<table>
<thead>
<tr>
<th>HOST</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host API</td>
<td>SWI_MSG</td>
</tr>
<tr>
<td>HPI Driver</td>
<td>Framing Tasks</td>
</tr>
<tr>
<td></td>
<td>Channel Mux/Demux</td>
</tr>
<tr>
<td></td>
<td>DMA HWI</td>
</tr>
<tr>
<td></td>
<td>Serial Port(s)</td>
</tr>
</tbody>
</table>
# Configuration Messages

<table>
<thead>
<tr>
<th>Functionality</th>
<th>G.PAK API Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure System Parameters</td>
<td>gpakWriteSystemParms</td>
</tr>
<tr>
<td>Configure Serial Ports</td>
<td>gpakConfigurePorts</td>
</tr>
<tr>
<td>Configure Conference</td>
<td>gpakConfigureConference</td>
</tr>
<tr>
<td>Configure Channel</td>
<td>gpakConfigureChannel</td>
</tr>
<tr>
<td>Tear Down Channel</td>
<td>gpakTearDownChannel</td>
</tr>
<tr>
<td>Read System Configuration</td>
<td>gpakGetSystemConfig</td>
</tr>
<tr>
<td>Read System Parameters</td>
<td>gpakReadSystemParms</td>
</tr>
<tr>
<td>Read Channel Status</td>
<td>gpakGetChannelStatus</td>
</tr>
</tbody>
</table>
## Payload Transfers

<table>
<thead>
<tr>
<th>Functionality</th>
<th>G.PAK API Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write payload to DSP</td>
<td>gpakSendPayloadToDsp</td>
</tr>
<tr>
<td>Read payload from DSP</td>
<td>gpakGetPayloadFromDsp</td>
</tr>
</tbody>
</table>
## Other API Calls

<table>
<thead>
<tr>
<th>Functionality</th>
<th>G.PAK API Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write image to DSP</td>
<td>gpakDownloadDsp</td>
</tr>
<tr>
<td>Read frame task statistics</td>
<td>gpakReadCpuUsage</td>
</tr>
</tbody>
</table>
## Host Supplied Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads memory from specified DSP address</td>
<td>gpakReadDspMemory</td>
</tr>
<tr>
<td>Writes memory at specified DSP address</td>
<td>gpakWriteDspMemory</td>
</tr>
<tr>
<td>Delays at fixed time interval before polling for message responses</td>
<td>gpakHostDelay</td>
</tr>
<tr>
<td>Provides host task with exclusive access to DSP’s memory</td>
<td>gpakLockAccess</td>
</tr>
<tr>
<td>Releases exclusive access to DSP’s memory</td>
<td>gpakUnlockAccess</td>
</tr>
<tr>
<td>Reads data from ‘File’ for DSP download</td>
<td>gpakReadFile</td>
</tr>
</tbody>
</table>
1. Use DSP Resource Wizard to estimate resource requirements and select DSP

2. Use G.PAK Configuration Builder to build DSP image (Load and test)

3. Customize G.PAK Source Code (Load and Test)
Hands On #1: DSP Resource Wizard

1. Use DSP Resource Wizard to size application and select DSP

2. Browse to c:\adaptivedigital\tidc
   - Available at http://wizard.adaptivedigital.com

3. Double-click on DSPResourceWiz.htm

4. Select DSP, algorithms, and channel count
Hands On #2: G.PAK Build

1. Use the G.PAK Configuration Builder to configure and build a DSP image

2. Load the DSP image into the DSK board

3. Use the host interface program to set up a phone call
G.PAK Configuration Builder

1. User provides Configuration Information.
2. Configuration Information is used to configure the Program (GUI).
3. Configuration Builder creates Code Generator.
4. Code Generator generates Application Specific Code.
5. Application Specific Code is compiled by Compile/Build Tools.
7. Application SW Image is sent to ASSP.
G.PAK Demo

PC Emulates Host Processor Running G.PAK API

IP Network

PC

JTAG / PCI

C64X DSK/EVM

Hybrid

Phone

Audio Source

Speaker

IP Phone

Minds in Motion

Technology for Innovators™
Hands On #2: Step by Step

1. Start Code Composer Studio

2. Open the G.PAK Project
   - Project -> Open ->
     c:\adaptivedigital\tidc\gpakc64x\gpak64xdsp\gpak6416develop.pjt
     (ignore the missing library warning – we’ll build that later)

3. Build the G.PAK Project
   - Project->Build
     When prompted for a G.PAK Project, click on gpak6416develop.pjt
Hands On #2: Step by Step
Hands on #2: Step by Step

1. Select the G.729AB Vocoder
2. Click Build
3. The program will load when the build is complete
4. Start the program – Debug -> Run
Hands On #2: Step by Step

1. Start the host interface tool
   - Double-click on `c:\adaptivedigital\tidc\Gpak64x\gpakhostinterface.exe`

2. Click on RTDX

3. The host interface tool will appear
Hands On #2: Step by Step

1. Set up the serial port (Click Port Params)
   This screen will come up:

2. Use Defaults

3. Click OK
Hands On #2: Step by Step

1. Set up a TDM to Packet channel. The following screen will come up:

2. Use Defaults

3. Click OK
Hands On #2: Step by Step

1. Start your audio source

2. Listen to the output

3. Other things to try:
   change the vocoder from G.729AB to G.711
   using the channel setup command
Hands On #2: Review

- Use DSP Resource Wizard to check DSP resource requirements
- Configure and build the DSP image
- Load and run the DSP image
- Use the host interface program to configure the DSP software and set up a PCM-Packet channel
Hands-On #2: Review

PC Emulates Host Processor Running G.PAK API

IP Network

PC

C64X DSK/EVM

JTAG / PCI

Hybrid

Phone

Audio Source

Speaker

IP Phone

Minds in Motion

Technology for Innovators™

Texas Instruments
Hands-On #3: Adding an Algorithm

- We will modify the G.PAK source code to add a reverb algorithm to the voice path.
- View reverb.h
Hands-On #3: Adding Reverb
Adding a New Algorithm

✓ Important note about auto-generated files
  - The G.PAK configuration utility modifies some source files.
  - If you are modifying source files, these source files will be overwritten if you re-run the G.PAK configuration utility.
Adding New Algorithms

• Memory Allocation
• Instantiation
• Initialization
• Signal Processing
• Teardown
## Memory Allocation / Instantiation

<table>
<thead>
<tr>
<th>File</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GpakDefs.h</td>
<td>Include file reverb.h</td>
</tr>
<tr>
<td>GPak6416Devel op.c</td>
<td>Instantiate Reverb channels and pooling data</td>
</tr>
<tr>
<td>GpakExts.h</td>
<td>Add references to Reverb instance and pooling information</td>
</tr>
</tbody>
</table>
## Initialization

<table>
<thead>
<tr>
<th>File / Function</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GpakMsg.c / InitGpakInterface</td>
<td>Mark all Reverb channels in pool as available</td>
</tr>
<tr>
<td>GpakFrame.c / initReverbChannel</td>
<td>New Function – Pulls a Reverb channel from pool and initializes it</td>
</tr>
<tr>
<td>Pcm2Pkt.c / SetupPcm2Pkt</td>
<td>Call initReverbChannel</td>
</tr>
<tr>
<td>GpakMsg.c / ProcConfigureChannelMsg</td>
<td>Free Reverb channel to available pool</td>
</tr>
</tbody>
</table>
### Signal Processing

<table>
<thead>
<tr>
<th>File / Function</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pcm2Pkt.c / FrameInPcm2Pkt</td>
<td>Add call to the Reverb function</td>
</tr>
</tbody>
</table>
Teardown

<table>
<thead>
<tr>
<th>File / Function</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GpakFrame.c / DeallocChanReverb</td>
<td>New function to free a Reverb channel to the pool</td>
</tr>
<tr>
<td>GpakMsg.c / ProcTearDownChannelMsg</td>
<td>Call DeallocChanReverb</td>
</tr>
</tbody>
</table>
Hands On # 3: Step by Step

- Disable G.PAK Configuration builder so that it does not overwrite source files
  - From CCS:
    1. Right Click on GPAKC6416Develop project in project pane
    2. Click on Build Options
    3. Click on “General” Tab
    4. Place REM in front of the command that runs the configuration .EXE file
    5. Click OK
Hands On # 3: Step by Step

• Replace gpak6416develop.c with gpak6416develop_handson_3.c
  - File -> Open gpak6416develop_handson_3.c
  - File -> Save As gpak6416develop.c
    • Yes to overwrite warning

• Why did I do this?
  - The G.PAK configuration program rewrites gpak6416develop.c. We need a modified version for this exercise, which has been already created as gpakdevelop_handson_3.c.
Hands On # 3: Step by Step

• Open each of the following files:
  ✓ GPAKDEFS.H
  ✓ GPAK6416DEVELOP.C
  ✓ PCM2PKT.C
  ✓ PKT2PCM.C
  ✓ GPAKFRAME.C

• In Each file, look for the occurrence of the word HANDS_ON to understand the code that the #ifdefs are enabling
Hands On # 3: Step by Step

- Change the project options in each CCS project to include the pre-processor definition HANDS_ON
  - Right click on a project
  - Click on Build Options
  - Click on the Preprocessor category
  - In the “Pre-defined symbols” line, append with “;HANDS_ON”
  - Repeat for all projects
Hands On # 3: Step by Step

• Rebuild the main project (GPAK6416Develop.pjt)
  ▪ From the Project menu, click on Build

• Once the project builds successfully, the executable file will be loaded into the DSP on the DSK board.
1. Start the program
   - From the Debug menu click on Run

2. Switch windows tasks back to the Host Interface program.
Hands On #3: Step by Step
Hands On # 3: Step by Step

• Since the host program was already running, click on “Reconnect”. Click OK on the error messages.

(We reloaded the DSP, causing the host program to become out-of-sync.)
Hands On # 3: Step by Step

1. Set up the serial port (Click Port Params)
   This screen will come up:

2. Use Defaults

3. Click OK
1. Set up a TDM to Packet channel. The following screen will come up:

2. Use Defaults

3. Click OK
Hands On # 3: Step by Step

1. Start your audio source
2. Listen to the output
3. Other things to try:
   change the vocoder from G.729AB to G.711 using the channel setup command
• Use DSP Resource Wizard to check DSP resource requirements
• Configure and build the DSP program
• Modify the source code to add reverb algorithm
• Rebuild the DSP software program
• Load and run the DSP program
• Use the host interface program to configure the DSP software and set-up a PCM-Packet channel
• Listen to the result
Summary

- Importance of leveraging existing technology
- Use of open-source to leverage existing technology
- Open Source Licensing
Leveraging an Open-Source VoIP Application to Speed VoIP Product Development

Scott Kurtz
Adaptive Digital Technologies
610-825-0182 x 117
scott.kurtz@adaptivedigital.com
www.adaptivedigital.com