Developing multimedia components for faster integration in multimedia frameworks

Viraj Karandikar, Sonal Gupta
Presenter – Rohit Bhuva VP Business Development

rohit@ittiam.com 972 202 0744

Ittiam Systems
Contents

- Need for Media Frameworks
- Challenges in component integration
- Approach for faster integration of components
- Case studies on integrating components in
  - OpenCORE™
  - GStreamer
  - DirectShow™
- Conclusion
Introduction to Ittiam Systems

**Our Customers**

- Customer: Electronics OEM Company
- Customer: SC Company

**What We Do**

- Embedded Software IP
- Silicon IP

**Who We Are**

- Digital Signal Processing (DSP) Systems Company
- Embedded Applications in Media Processing & Communication
- Business Model of IP Licensing

**Our History**

- Founded in 2001; venture capital of $12.5M raised till date.
- Rated the world’s most preferred DSP IP supplier four years in a row.
- 190+ customers worldwide, ~80% OEM and ~20% Semiconductor.
- Red Herring 100 Asia Award 2005.
- Red Herring 100 Global finalist 2007.
- 225 people on board in 2009.
- 40+ patents filed till date.
- 6 Patents awarded.
Ittiam Offerings

- Production Quality System Design
- Reference Design
- Application Software
- Framework / Middleware
- Basic Codecs + Components
- OS + BSP
- HW Platform (DSP, SoC and/or ARM + HW Accelerator)

Ittiam is supporting key customers with Reference Design/Application.

Ittiam is supporting a worldwide customer base on a wide range of digital media & communication applications with this offering.

These are critical components for system quality and performance. One need to be expert to maximize Silicon capability by optimizing Algorithm. Application specific codecs also enables competitive advantages.
# Ittiam Video Codecs on ARM

<table>
<thead>
<tr>
<th></th>
<th>ARM9E</th>
<th>ARM11</th>
<th>Cortex-A8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DIVX</strong></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td><strong>H.263 Baseline</strong></td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td><strong>H.264 BP</strong></td>
<td></td>
<td></td>
<td>D, E</td>
</tr>
<tr>
<td><strong>H.264 MP</strong></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td><strong>H.264 BP / MP Multi-channel</strong></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td><strong>H.264 MP / HP</strong></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td><strong>JPEG Baseline /MJPEG</strong></td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
</tr>
<tr>
<td><strong>MPEG2 MP @ ML</strong></td>
<td></td>
<td></td>
<td>In Dev</td>
</tr>
<tr>
<td><strong>MPEG4 ASP</strong></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td><strong>MPEG4 SP</strong></td>
<td>D</td>
<td>D</td>
<td>D, E</td>
</tr>
<tr>
<td><strong>MPEG4 SP Multi-channel</strong></td>
<td>D</td>
<td></td>
<td>D, E</td>
</tr>
<tr>
<td><strong>WMV9 / VC1 SP/MP</strong></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td><strong>WMV9 / VC1 SP / MP / AP</strong></td>
<td></td>
<td></td>
<td>In Dev</td>
</tr>
</tbody>
</table>
Ittiam Audio Codecs on ARM

<table>
<thead>
<tr>
<th>Codec Type</th>
<th>ARM9E</th>
<th>ARM11</th>
<th>Cortex-M3</th>
<th>Cortex-A8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC-LC</td>
<td>D, E</td>
<td>D, E</td>
<td>D</td>
<td>D, E</td>
</tr>
<tr>
<td>AAC-LD</td>
<td>D, E</td>
<td>D, E</td>
<td></td>
<td>D, E</td>
</tr>
<tr>
<td>HE-AAC v1, v2 (eAAC+)</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
</tr>
<tr>
<td>MP3</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
</tr>
<tr>
<td>BSAC</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>FLAC</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Ogg Vorbis</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>WMA</td>
<td>D, E</td>
<td>D, E</td>
<td>D</td>
<td>D, E</td>
</tr>
<tr>
<td>WMA lossless</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>WMA Pro LBR</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>MP2 5.1</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>AAC-LC 5.1</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
</tr>
<tr>
<td>Dolby/AC-3 5.1 *</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
<td></td>
</tr>
<tr>
<td>Enhanced AAC+ 5.1</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
<td></td>
</tr>
<tr>
<td>WMA Pro 5.1/7.1</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

* Dolby/AC-3 5.1 Encoder is Consumer Encoder
Need for Media Framework

- Wide range of multimedia products
- Applications demand support for multiple codecs, parsers, and other system components
- Media framework abstracts these components and helps in application development
Features of Media Framework

- Assists modular design of applications
- Provides standard application interface
- Enables easy component integration using Component and Resource manager
- Reduces development time
Multiple Media Frameworks

- Multiple frameworks in use for different applications, on different platforms
- OpenCORE™ on Android
- GStreamer on Linux
- DirectShow™ on Windows®
- Media Device Framework on Symbian®
- Ittiam Media System Framework
Challenges in component integration

- Need for supporting components on multiple frameworks

- Different Media Frameworks have different
  - Plug-in interface
  - Resource management

- Multithreaded environment demand stable, robust components

- Integrating components in short time is a challenge
Approach for faster component integration

- Implement a common interface across different components
- Test component for robustness and re-entrant implementation
- Implement framework specific plug-in
Common component interface

- `component_init()`
  - Provides flexible resource management through callbacks
  - Allocates resources required for the component

- `component_deinit()`
  - De-allocates all the resources

- `component_process()`
  - Processes input to generate output

- `component_control()`
  - Provides interface to get/set configuration parameters
Robust & Re-entrant components

- Debugging in final application is difficult
- Use stand-alone test bench to validate components

Robustness
- Test for erroneous bit-streams by introducing errors in the bit-stream with different error rates
- Test component over long duration with different bit-streams
- Test for memory corruption using tools like electric fence, valgrind etc.

Re-entrant implementation
- Test in simulated multi-instance, multi-threaded environment
Framework specific plug-in

- Develop a framework compatible plug-in
  - Plug-in exposes framework specific interface functions
  - Plug-in calls the component using common component interface

- Handle resource allocation for the component

- Implement a buffering mechanism for different input payload scenarios
Case studies on integrating components

- OpenCORE™ - PacketVideo®’s media framework used in Android
- GStreamer - Open source framework for creating media applications on Linux
- DirectShow™ - Microsoft®’s media framework used on the Windows platform
OpenCORE Framework

- Uses engine and node-based architecture
- Engine creates and manages the graph of nodes
- Nodes interact with components using OpenMAX™ IL interface

Flow graph for Player Engine
OpenCORE and OpenMAX IL interface

- PacketVideo (PV) OpenMAX IL - Implementation of OpenMAX IL in OpenCORE
Integrating components in OpenCORE

- Developing a OpenCORE plug-in
  - Use the base class implementation for the component (audio/video/base) to derive specific OpenMAX component.

- Implement the following methods
  - ComponentInit()
  - ProcessData()
  - ComponentDeinit()

- Over-ride the base class methods like SetParameter, GetParameter for component specifics if required

- Handle buffering of input data if required
- Register the component to OpenMAX IL core
Integrating components in OpenCORE

- Mapping from OpenCORE component plug-in methods to common component interface
Case studies on integrating components

- **OpenCORE** - PacketVideo’s media framework used in Android

- **GStreamer** - Open source framework for creating media applications on Linux

- **DirectShow** - Microsoft’s media framework used on the Windows platform
- Component (elements) are linked via pads and arranged in a pipeline
- Pads are input (sink) and output (source) ports for the component

GStreamer pipeline
Integrating components in GStreamer

- **Method 1 - Using GStreamer plug-in**
  - Implement a plug-in using component interface functions
  - Implement buffering logic if required

- **Method 2 - Using gst-openmax plug-in**
  - gst-openmax – Generic GStreamer plug-in encapsulating OpenMAX IL APIs
  - Existing OpenMAX IL components can be integrated with minimal effort
Method 1 - Using GStreamer plug-in

GStreamer plug-in

- Constructor functions
  - plugin_base_init()
  - plugin_class_init()
  - plugin_init()

- set_property() and get_property() functions for configuring parameters

- change_state() function for component specific processing

- plugin_chain() function for processing the data

Component Library

- component_control()
- component_init() / component_deinit()
- component_process()
Method 2 - Using gst-openmax plug-in

- GstOmxBaseFilter classes translate interface between GStreamer framework and OpenMAX IL
- Derive gst-openmax plug-in from GstOmxBaseFilter class
- Implement these functions
  - Constructors
  - `component_get_type()`
Method 2 - Using gst-openmax plug-in

- gst-openmax translates GStreamer plug-in interface to OpenMAX IL interface

- gst-openmax implements GstOmxBaseFilter classes and utility functions

- Component plug-in derives from GstOmxBaseFilter and implements these functions
  - Constructors
  - component_get_type() for component registration
Case studies on integrating components

- **OpenCORE** - PacketVideo’s media framework used in Android

- **GStreamer** - Open source framework for creating media applications on Linux

- **DirectShow** - Microsoft’s media framework used on the Windows platform
DirectShow Framework

- Modular architecture based on software components called Filters and Filter Graph Manager
- Filters connect using I/O pins

DirectShow Flow Graph
Integrating components in DirectShow

- Implement component filter using DirectShow transform filter class
- Over-ride the associated pin classes if required
- Implement these methods for the filter plug-in
  - CheckInputType(), GetMediaType() and CheckTransform() to support media type negotiation
  - DecideOutputBufferSize() to set output pin’s buffer requirements
  - CompleteConnect() to allocate resources to the component
  - Transform() function to process the incoming data
  - BreakConnect() function to release the resources allocated to the component
Integrating components in DirectShow

- Mapping from Transform filter to common component interface
Case studies – Summary

- Approach for faster component integration is demonstrated

- Can be further applied to other media frameworks

- Time for integrating pilot component in a framework reduced by 50%

- Time for integrating subsequent components reduced by 75%
Emerging multimedia market demands availability of components in multiple media frameworks in short time

Implementing common component interface makes plug-in development faster

Plug-in developed for one component can be used as a template for new component

Time-to-market is further reduced by having robust components
Questions?

Thank You

Rohit Bhuva
VP Business Development
214 212 0104
rohit@ittiam.com