Embedded Streaming Media with GStreamer and BeagleBoard

Presented by Todd Fischer
todd.fischer (at) ridgerun.com
Agenda

• BeagleBoard-XM multimedia features
• GStreamer concepts
• GStreamer hands on exercises
• DMAI and GStreamer
• Questions
What's New

- Performance
  - Tuning not as critical
- Streaming media not central to product
- HD – more power for higher resolution
AM3730 Architecture

- Multimedia features:
  - Cortex A8 with Neon
  - C64x+ DSP
  - HD video accelerators
  - How to utilize the hardware features?
AM3730 Architecture

- Cortex A8
  - Neon
  - Super-scalar
  - Ghz clock
AM3730 Architecture

- C64+ DSP
  - HD video encode / decode
AM3730 Architecture

- Graphics Accelerator
  - Dedicated hardware
  - IVA – image, video audio accelerator
  - SGX accelerator
    - Supports OpenGLES
GStreamer

- Streaming media framework – audio and video
- Close to 200 plug-ins available
- Higher level than just input / filters / output
- Networking, audio/video mixed streams, auto data handling
- Various options utilizing hardware accelerators
GStreamer Overview

- **Elements**
  - Source, filters, sinks

- **Bins and Pipelines**
  - Containers, pipeline is the overall bin

- **Pads**
  - Element source / sink connection

- **Caps**
  - Capabilities organized by stream type with a set of properties

- **Plugin**
  - Collection of elements
Hands On Exercise 0

- Double click on GStreamer Class icon

- In terminal window, type `source ./s`

Need the period
Hands On Exercise 0

- Run video pipeline
  v1
- Actual command
  gst-launch videotestsrc ! ffmpegcolorspace ! xvimagesink
- See script contents
  cat v1
- There are lots of scripts
  ls
Simple MP3 Player

- Create dynamically using `gst-launch`
- Source element reads from a file
- Filter element converts MP3 to PWM
- Sink element passes to ALSA output
Simple Audio Player
Source Code

- Create pipeline, source, filter, sink
  - Set element properties
- Build into pipeline
  - Connect src and sink pads
- Setup pipeline event handler
  - End of stream
- Set pipeline state to play
- Run

See source `a_gst.c`
Keeping Plug-ins Organized

- Each known plug-in is added to registry
- Most aspects of plug-in are tracked in the registry
- Registry support run-in pipeline creation and dynamic filter selection
- Use `gst-inspect` to list plug-ins
Hands On Exercise 1

- Using `gst-inspect`, list
  - All plug-ins
  - All video plug-ins
  - Element properties for filesrc plug-in
Hands On Exercise 2

- GStreamer demultiplexing pipelines
  - d5 – flash video
    - First demultiplex into audio and video
      `gst-launch filesrc location=sprc720.flv ! flvdemux name=demux`
    - Second, process audio
      `demux.audio ! queue ! mad ! alsasink`
    - Third, process video
      `demux.video ! queue ! ffdec_vp6f ! omapdmaifbsink`
- Idea is the same
  - source data, filter data, send data to sink
GStreamer Daemon

- Separates audio / video streaming from controlling application
- Uses D-Bus messages to control pipeline
- Simplifies application development
  - No interaction with Gstreamer API
- Simplifies testing
  - Test app just sends D-Bus messages
Performance
Data Passing

- Minimize data copies
- Stream held in buffers with data, timestamp, other info
- When possible, buffer memory allocated by sink pad
- Use hardware when data copy is necessary
Performance
Data Transformation

- Cortex A8 compiler optimization
- NEON
  - Single Instruction Multiple Data
- C64+
  - Video accelerator
- DMA and other data movers
Performance Scheduling

- GStreamer elements may not be tuned for embedded use model
- Decoder may starve output device
  - Noticeable audio clicks
- Adjust buffering to pace entire pipeline
- Adjust thread priority
Davinci Multimedia Application Interface

Cortex Linux Application

GStreamer Libraries
- gst plug-in
- DMAI plug-in
- gst plug-in

Davinci Multimedia Application Interface
- audio
- video
- display
- frame copy

Codec Engine
- VISA
- OSAL
- CMEM

DSP Link

ARM Cortex with NEON

DSP Link

DSP BIOS

Framework Components
- codec
- video codec
- codec

video hardware accelerator
DMAI and GStreamer

- Davinci Multimedia Application Interface
  - Exposes OMAP/Davinci hardware using high level of abstraction
    - Stream audio / video
    - Graphics display
    - Hardware optimized frame/data copy
Sitara Codec Engine

- Isolates users for audio/video codecs from those implementing the codecs
- Codec can run in several places without the calling application being aware
  - Cortex A8, NEON, C64, hardware accelerator
  - Uses DSPLink and DSPBios conventions to support DSP based algorithms dynamically
Convenience Video

• The power of the AM3730
  ◦ Streaming audio / video can be added to most any product
• Example: stream from DM365 Leopard Board 365

v4l2src ! dmaienc_mpeg4 ! rtpmp4vpay ! udpsink

BeagleBoard XM

udpsrc ! rtpmp4vdepay ! ffdec_mpeg4 ! omapdmaifbsink
DSPLink

- DSP/BIOS™ LINK API
- PROCESSOR MANAGER
- GPP
- OS
- ADAPTATION LAYER
- LINK DRIVER
- GPP
- DSP

ARM9

C64
GStreamer in 3 Layers

Your Way Cool Application

GStreamer Media Handling

API for Super Fast Hardware
Backup Slides

• DSPLink presentation
DSPLink
ARM / DSP Communication

See diagrams in DSPLINK Programmers Guide
DSPLink
Communication Modules

- **Notify**
  - Low frequency communication
  - Small messages

- **MSGQ**
  - single reader
  - multiple writers
  - Variable size messages
  - Fixed buffer size

- **MPLIST**
  - multiple readers
  - multiple writers
DSPLink
Communication Modules

- **CHNL**
  - Single reader
  - Single writer
  - Fixed size buffers
  - Legacy SIO
  - Simplified buffer handling

- **RingIO**
  - Single reader
  - Single writer
  - Low reader/writer coupling
  - Variable data creation/consumption
  - Independent execution
DSPLINK
Support Modules

• PROC
  - hardware setup
  - DSP code load and boot
  - ARM/DSP communication
  - DSP shutdown

• POOL
  - Manage shared memory
  - Allocate / free Address translation
  - Cache alignment