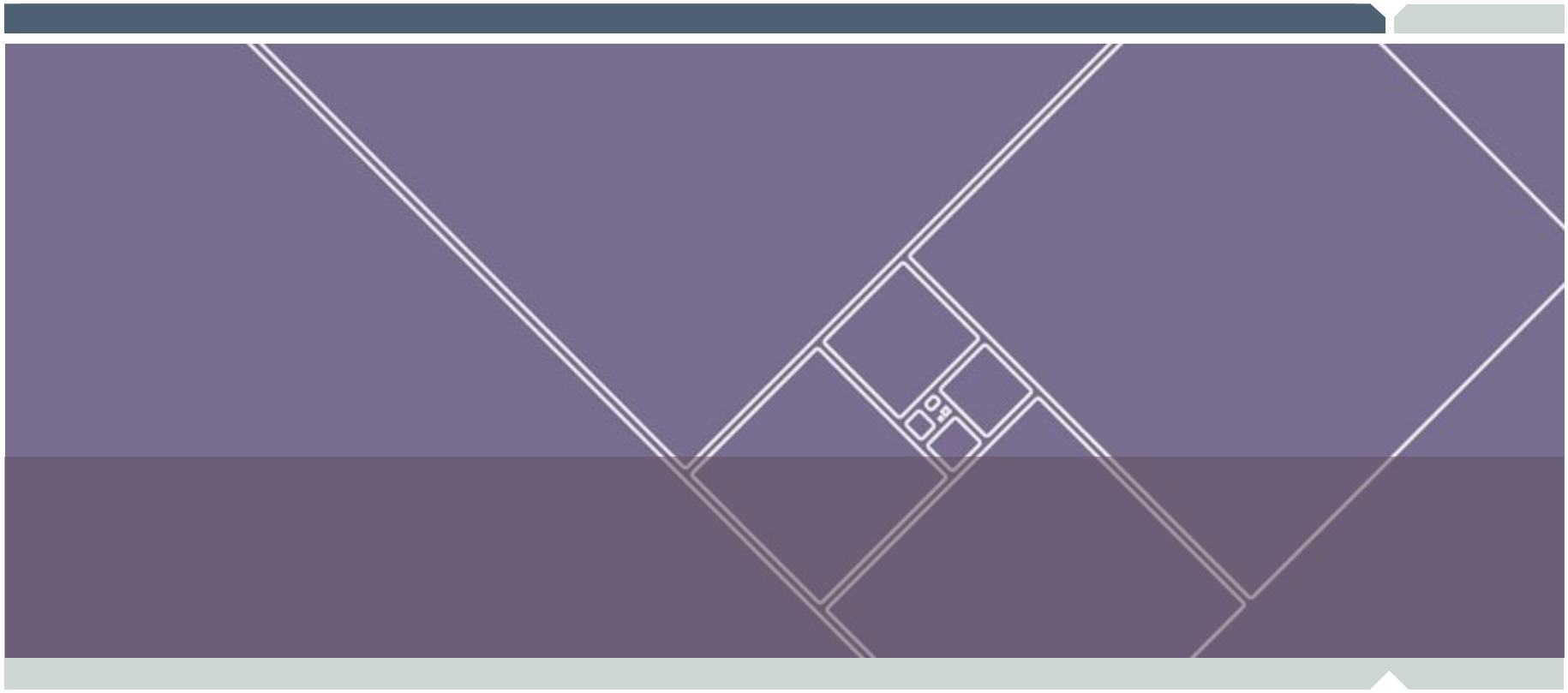
May, 2011

i.MX53 Video Processing Unit

Agenda

- i.MX53 Video Processing Unit Overview
- List of Supported Media Types
- Installing the MX53 Codec Package
- Overview of Gstreamer Framework
- Examples of VPU usage: video playback , video encoding, audio playback, streaming, audio encoding)
- Q & A



i.MX53 Video Processing Unit Overview

i.MX53 VPU Block Diagram

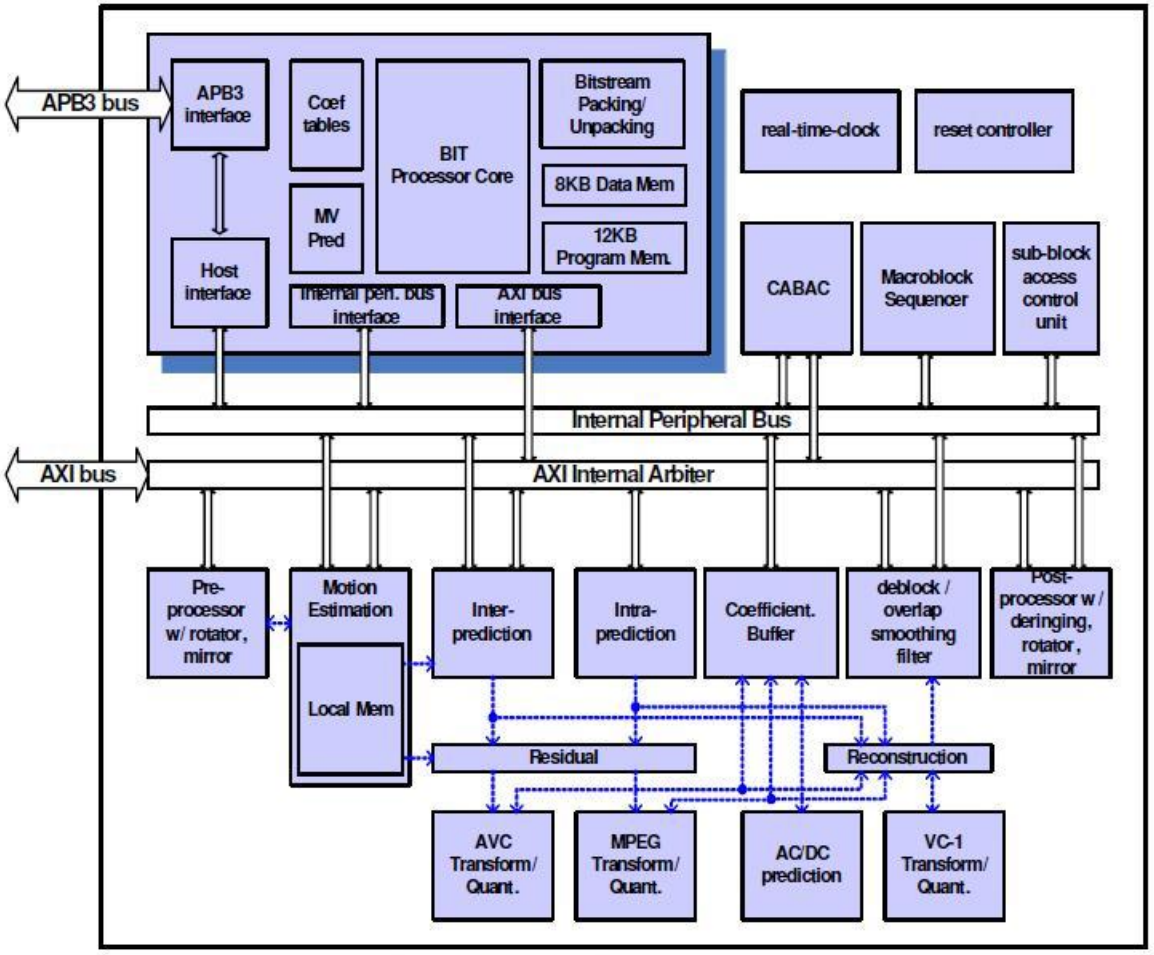


Figure 1. i.MX5x VPU Block Diagram

i.MX53 VPU Decoding Support

- • **H.264**
- — Fully compatible with the ITU-T Recommendation H.264 specification in BP/MP and HP.
- — Supports CABAC/CAVLC
- — Variable block size (16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4)
- — Error detection, concealment and error resilience tools
- • **VC1**
- — Supports all VC-1 profile features – SMPTE “Proposed SMPTE Standard for Television: VC-1 Compressed Video Bitstream format and Decoding Process”
- — Supports Simple/Main/Advanced Profile
- — Multi-resolution (Dynamic resolution) is not processed inside of video decoder
- • **MPEG-4**
- — Supports Simple/Advanced Simple profile except GMC.
- — Supports H.263 Baseline Profile
- — Support Divx from ver3.x to ver6.x
- — Supports Xvid

i.MX53 VPU Decoding Support (cont.)

- • **MPEG-2**
- Fully compatible with ISO/IEC 13182-2 MPEG2 specification in Main Profile.
- Support I,P and B frame
- Support field coded picture (interlaced) and frame coded picture
- • **MJPEG**
- — Baseline ISO/IEC 10918-1 JPEG compliance
- — Supports JFIF 1.02 input format with up to 3 components
- — 8 bit samples for each component
- — Support up to 4:4:4 in decoding
- • **RV-8/9/10** (Real Video)
- *(Only for licensees)*

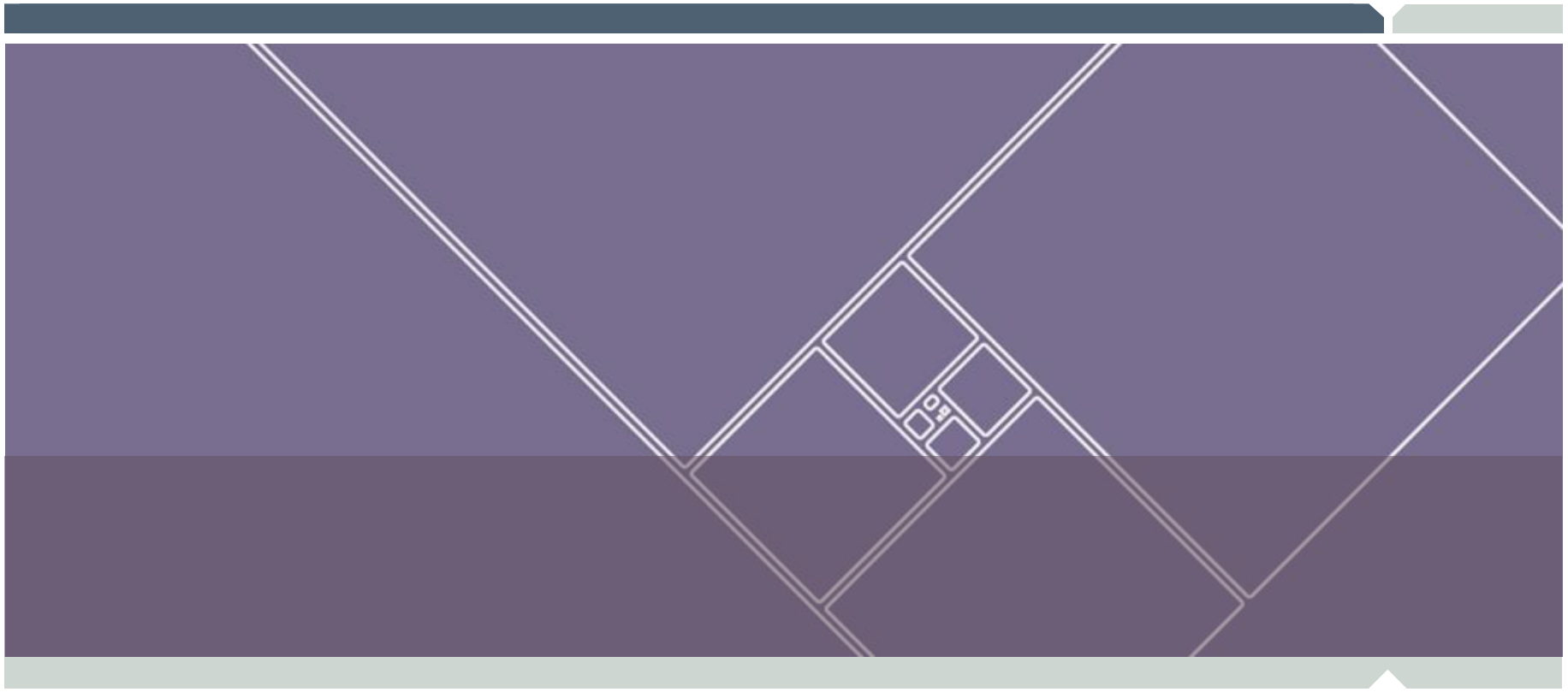
i.MX53 VPU Encoding Support

- **MPEG-4** AC/DC prediction
- **H.264/AVC** intra-prediction
- **H.263** Annex J, K (RS=0 and ASO=0), and T are supported.
- • Error resilience tools
- — MPEG-4 resync marker & data-partitioning with RVLC (Fixed number of bits/macroblocks between macroblocks)
- — CIR (Cyclic Intra Refresh)/AIR (Adaptive Intra Refresh)
- — Bit-rate control (CBR & VBR)

- **MJPEG** encoder supports up to 4:2:2 format

i.MX53 VPU Performance

- • All video decoder standards support up to 1920x1088 @ 30 fps at 133 MHz
- • All video encoder standards support up to 1280x720 @ 30 fps (720x576 @ 25 fps) at 66 MHz
- • MJPEG decoder supports 32 M pixel per second and the image size is up to 8196 x 8196 @ 133 MHz.
- • MJPEG encoder supports 64 M pixel per second and the image size is up to 8196 x 8196 @ 133 MHz.



VPU Software Structure

MX53 VPU Software

- The VPU software can be divided into two parts: the kernel driver and the user-space library as well as the application in user space. The kernel driver takes responsibility for system control and reserving resources(memory/IRQ). It provides an IOCTL interface for the application layer in user-space as a path to accesssystem resources. The application in user-space calls related IOCTLs and codec library functions to implement a complex codec system.

Source Code Structure (Kernel Driver)

- The table below lists the kernel space source files available in the following directories:
- <ltib_dir>/rpm/BUILD/linux/arch/arm/plat-mxc/include/mach/
- <ltib_dir>/rpm/BUILD/linux/drivers/mxc/vpu/

File	Description
mxc_vpu.h	Header file defining IOCTLs and memory structures
mxc_vpu.c	Device management and file operation interface implementation

Source Code Structure (Userspace)

- The table below lists the user-space library source files available in the
- <ltib_dir>/rpm/BUILD/imx-lib-11.03.00/vpu directory:

File	Description
vpu_io.c	Interfaces with the kernel driver for opening the VPU device and allocating memory
vpu_io.h	Header file for IOCTLs
vpu_lib.c	Core codec implementation in user space
vpu_lib.h	Header file of the codec
vpu_reg.h	Register definition of VPU
vpu_util.c	File implementing common utilities used by the codec
vpu_util.h	Header file

Source Code Structure (cont.)

- The table below lists the firmware files available in the following directories:
- <ltib_dir>/rpm/BUILD/firmware-imx-11.03.00/lib/firmware/vpu/vpu_fw_imx53.bin

VPU unit test

- Linux BSP provides some unit test examples for the VPU
- To extract the source code:

./ltib –p imx-test –m prep

- Source code will be available at:

rpm/BUILD/imxt-test-11.01/test/mxc_vpu_test

- Examples for VPU decode/encode/loopback are presented (dec.c , enc.c , loopback.c)

VPU unit test (cont.)

- Example

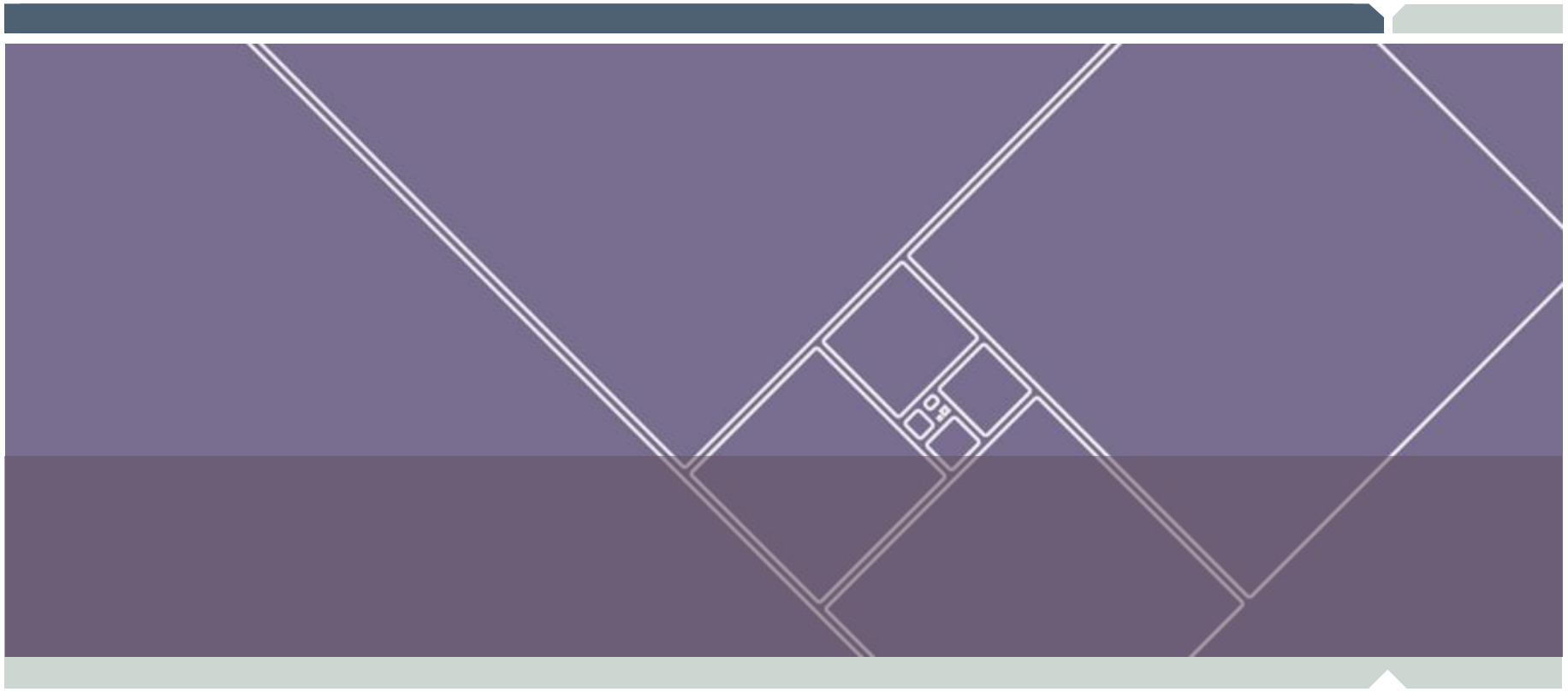
./mxc_vpu_test.out -D "-i /home/raw.h264 -f 2“

,where

-D: tell the VPU to perform a decoder operation

-i : Specify the input file

-f 2: Pass the format to the VPU: 2 means H.264



Installing i.MX53 Codec Package

MX53QSB Software Setup

- Download the 11.03 BSP:
  - www.freescale.com/imxquickstart _Click on Downloads tab and then download the L2.6.35_11.03_ER_SOURCE file

- Download the codec package (Gstreamer framework) :
  - www.freescale.com/imxquickstart , Click on Downloads tab and then download the IMX53_11_03_LINUX_MMCODECS file

- Documentation:

Extract
Linux_Multimedia_Framework_Docs_MX53Ubuntu_1.9.8.tar.gz file and take a look at:
Linux_Multimedia_Framework_Docs_MX53Ubuntu_1.9.8\docs
Linux_Multimedia_Framework_User_Guide.pdf

MX53QSB Bootloader Setup

- U-boot is the bootloader used in Linux for MX53QSB
- Display Selection options
- 1. video=mxcdi1fb:BGR24,XGA di1_primary tve
- 2.video=mxcdi0fb:RGB565,CLAA-WVGA
- 3. video=mxcdi0fb:RGB24,SEIKO-WVGA
- 4. video=mxcdi0fb:RGB24,1024x768M@60
- 5. video=mxcdi1fb: YUV444, 1080P30 di1_primary tve

MX53QSB Bootloader Setup (cont.)

- Kernel is retrieved via TFTP
- Root file system is mounted via NFS
- bootcmd = run bootcmd_net

(bootcmd_net does the NFS settings. If booting from MMC then

bootcmd = run bootcmd_mmc)

Installing the i.MX53 Codec Package

- After installation of MX53 11.03 BSP, please install the 11.03 Multimedia following the instructions from Linux_Multimedia_Framework_Docs_MX53Ubuntu_1.9.8/docs/**Linux_Multimedia_Framework_User_Guide.pdf**

- Extract the Multimedia package (IMX_ER_1103_LINUX_MMCODECS) content package and copy the **gst-fsl-plugin-1.9.8.tar.gz** and **fsl-mm-codeclib-1.9.8.tar.gz** files to /opt/freescale/pkgs

Installing the i.MX53 Codec Package (cont.)

- Selecting the Multimedia Package from LTIB

./ltib -c

Package list --->

Freescale Multimedia Plugins/Codecs --->

--- fsl-mm-codec-libs

[*] gstreamer-fsl-plugins

and also select

[*] gstreamer-plugins-good

Testing the codec package installation

- In order to check whether the installation of the codecs were succesful:

```
root@freescale ~$ gst-inspect | grep mfw
mfw_v4lsink:  mfw_v4lsink: Freescale: V4L Sink
mfw_deinterlacer:  mfw_deinterlacer: Mfw De-interlace
mfw_mpeg4aspdecoder:  mfw_mpeg4aspdecoder: Freescale MPEG4 Decoder
mfw_vpudecoder:  mfw_vpudecoder: Freescale: Hardware (VPU) Decoder
mfw_mp3decoder:  mfw_mp3decoder: freescale mp3 decoder
mfw_ipucsc:  mfw_ipucsc: Freescale IPU Color Space Converter
mfw_aacdecoder:  mfw_aacdecoder: Freescale AAC Decoder Plugin
mfw_mpeg2decoder:  mfw_mpeg2decoder: Freescale MPEG2 Decoder
mfw_v4lsrc:  mfw_v4lsrc: Freescale Video Source plug-in
mfw_vpuencoder:  mfw_vpuencoder: Freescale: Hardware (VPU) Encoder
mfw_vorbisdecoder:  mfw_vorbisdecoder: Freescale vorbis Decoder Plugin
mfw_isink:  mfw_isink: Freescale: i_sink
mfw_h264decoder:  mfw_h264decoder: Freescale    H264 decoder
mfw_audio_pp:  mfw_audio_pp: Freescale Audio Post-process Filter
```

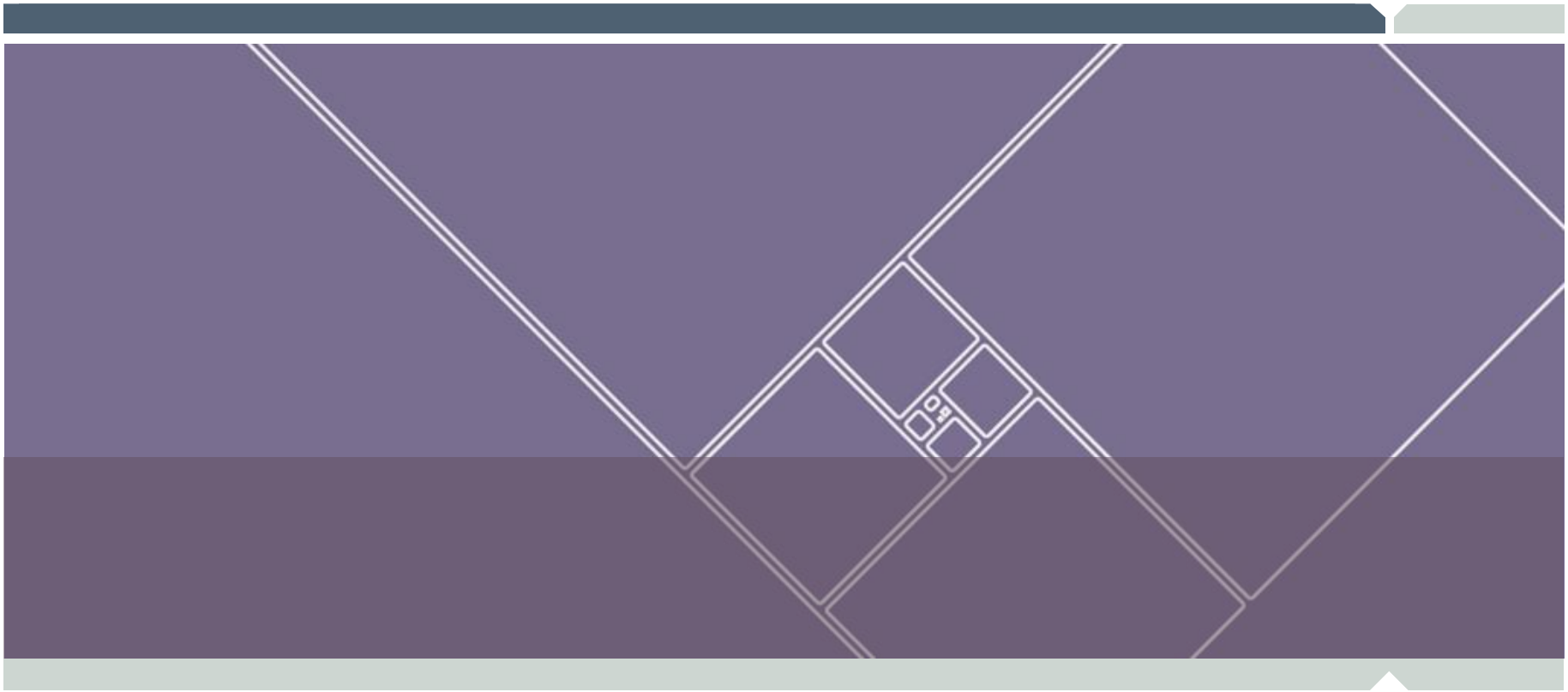
Testing audio in Gstreamer

- In order to test audio in Gstreamer:

- Connect a headphone to xxx jack of MX53QSB and run the following pipeline:

gst-launch audiotestsrc ! alsasink

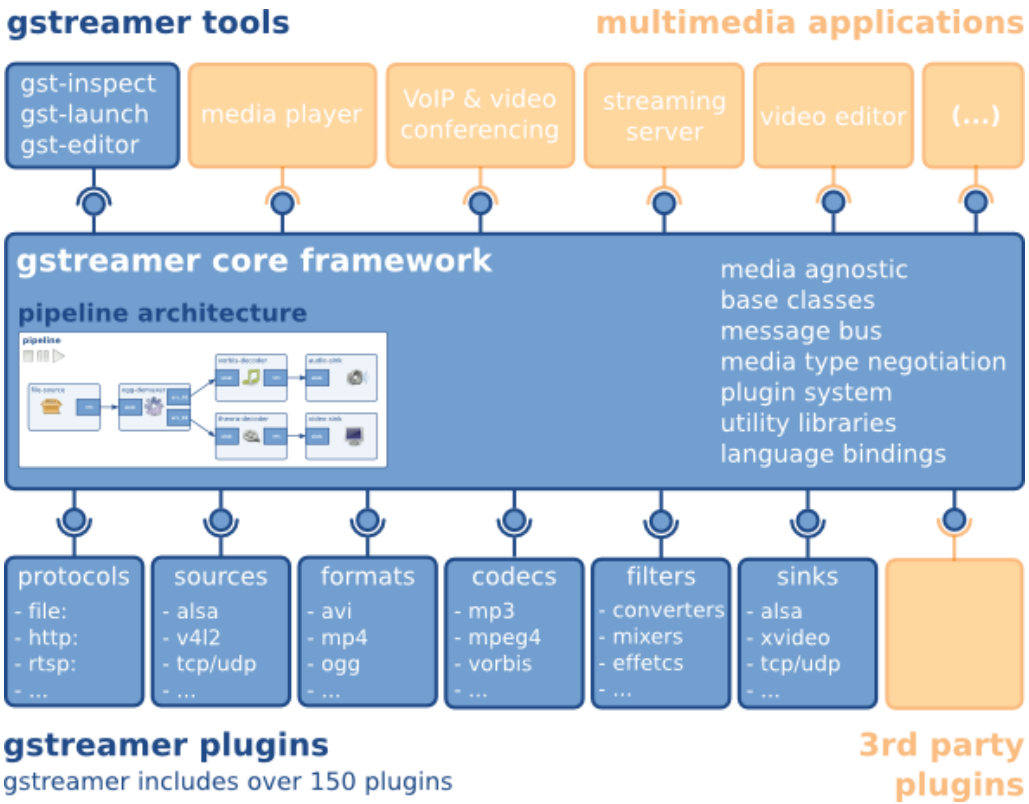
- Tone should be heard



Introduction to Gstreamer

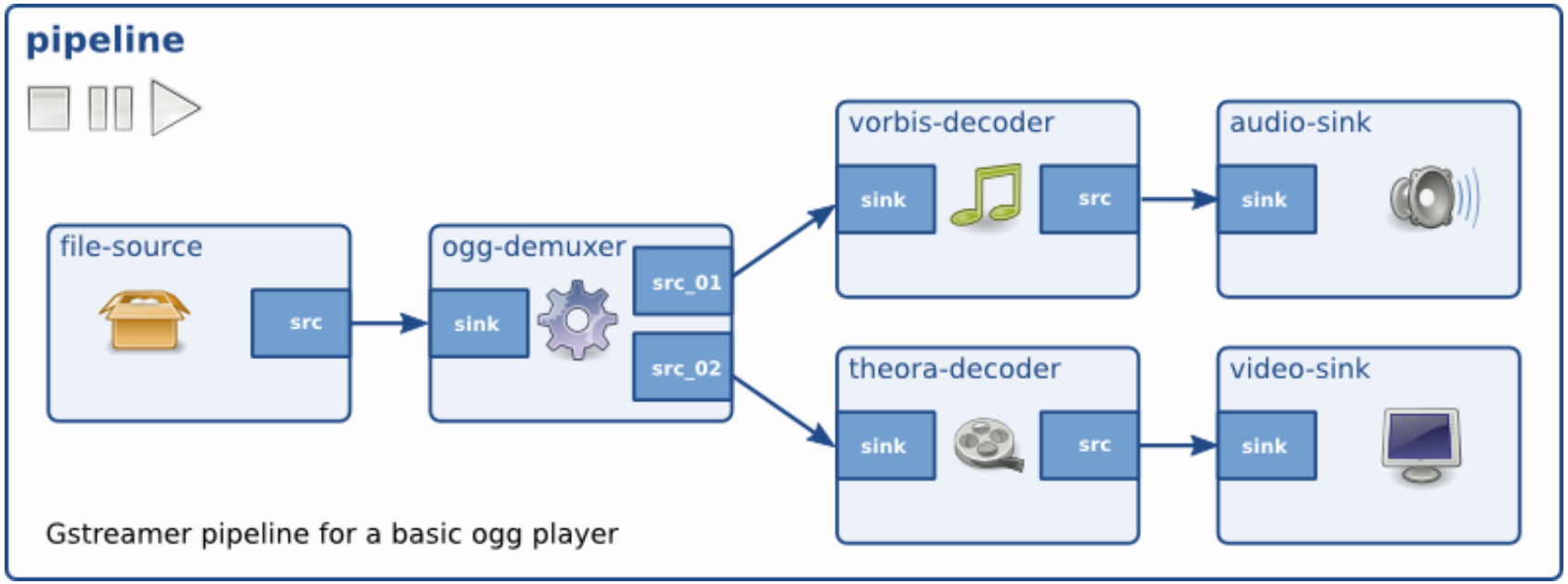
Gstreamer Overview

- GStreamer is a framework for creating streaming media applications



Gstreamer Pipeline Concept

- Example of a Gstreamer pipeline



Inspecting the properties of a Gstreamer plugin

- **gst-inspect audiotestsrc**

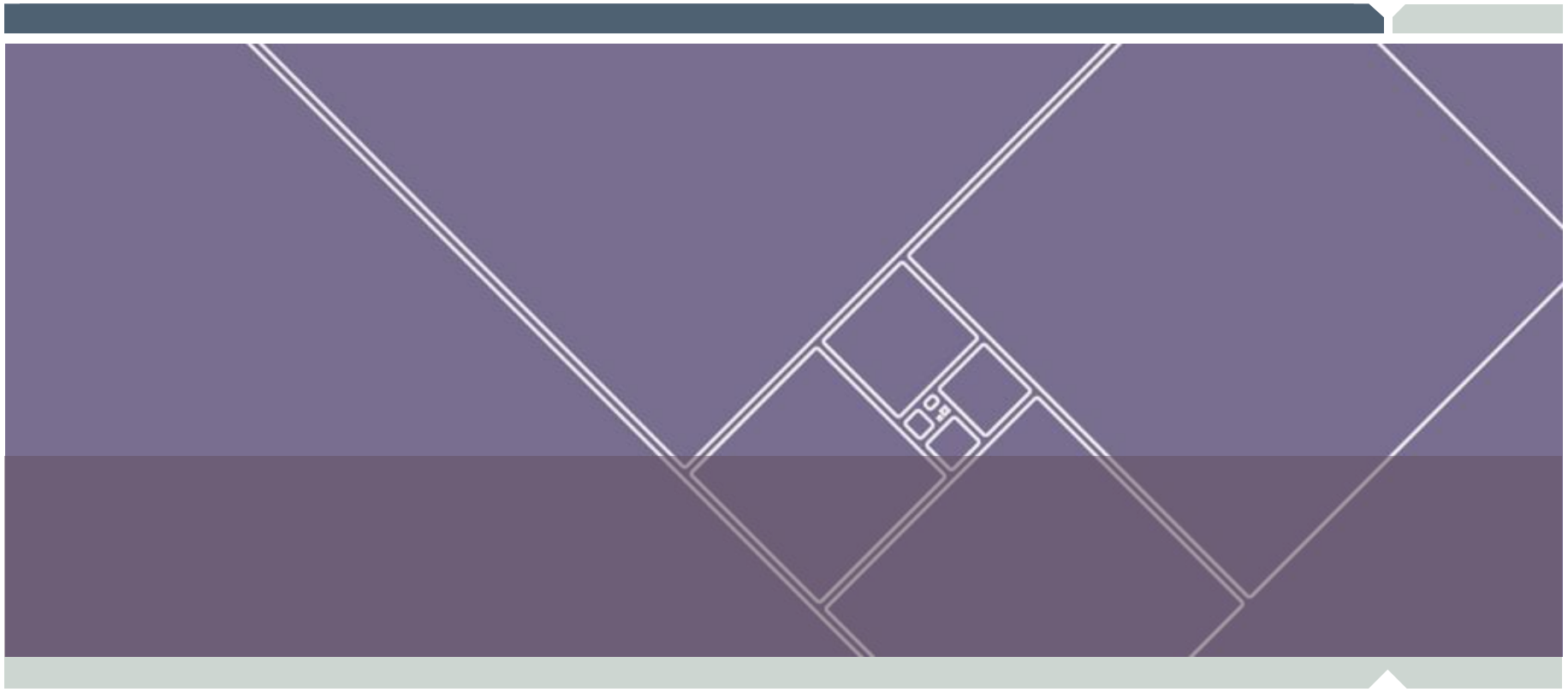
```
gst-inspect-0.10:2077): GStreamer-WARNING **: Failed to load plugin '/usr/lib/F
Factory Details:
  Long name:    Audio test source
  Class:        Source/Audio
  Description:  Creates audio test signals of given frequency and volume
  Author(s):    Stefan Kost <ensonic@users.sf.net>
  Rank:         none (0)

Plugin Details:
  Name:                audiotestsrc
  Description:         Creates audio test signals of given frequency and volume
  Filename:            /usr/lib/gstreamer-0.10/libgstaudiotestsrc.so
  Version:             0.10.25
  License:             LGPL
  Source module:       gst-plugins-base
  Binary package:      GStreamer Base Plug-ins source release
  Origin URL:          Unknown package origin

GObject
 +----GstObject
       +----GstElement
             +----GstBaseSrc
                   +----GstAudioTestSrc
```

Inspecting the properties of a Gstreamer plugin (cont.)

```
…
wave            : Oscillator waveform
                  flags: readable, writable, controllable
                  Enum "GstAudioTestSrcWave" Default: 0, "sine" Current: "
                     (0): sine            - Sine
                     (1): square          - Square
                     (2): saw             - Saw
                     (3): triangle        - Triangle
                     (4): silence         - Silence
                     (5): white-noise     - White uniform noise
                     (6): pink-noise      - Pink noise
                     (7): sine-table      - Sine table
                     (8): ticks           - Periodic Ticks
                     (9): gaussian-noise  - White Gaussian noise
  freq          : Frequency of test signal
                  flags: readable, writable, controllable
                  Double. Range:               0 -          20000 Defaul0
  volume           : Volume of test signal
                  flags: readable, writable, controllable
                  Double. Range:
….
```



Basic Gstreamer Examples

Testing Video in Gstreamer

- In order to test the video output on Gstreamer:
- **gst-launch videotestsrc ! autovideosink**
- Note: in order to avoid the LCD timeout you can add the following line to rootfs/etc/profile:

```
echo -e -n '\033[9]' > /dev/tty0
```

- Or in case the LCD has already shutdown it is possible to turn it on:

```
echo 0 > /sys/class/graphics/fb0/blank
```

Changing the video output resolution

- In order to change the video output resolution:

- **gst-launch videotestsrc ! mfw_v4lsink ! disp-width=640 disp-height=480**

Video-Only Playback

- In order to play a H264 (in avi format) file locally:

- **gst-launch filesrc location=file.avi ! avidemux ! mfw_vpudecoder ! mfw_v4lsink**

- In order to play a MP4 (in mp4 format) file locally:

- **gst-launch filesrc location=file.mp4 ! qtdemux ! mfw_vpudecoder ! mfw_v4lsink**

Video and Audio Playback

- In order to play a MP4(MPEG4+AAC) file:

- **gst-launch filesrc location=test.mp4 ! mfw_mp4demuxer name=demux demux. ! queue max-size-buffers=0 max-size-time=0 ! mfw_vpudecoder ! mfw_v4lsink demux. ! queue max-size-buffers=0 max-size-time=0 ! mfw_aacplusdecoder ! audioconvert ! 'audio/x-raw-int, channels=2' ! alsasink**

- **There should be a simpler way to do it! ;-)**

Video and Audio Playback (cont.)

- The playbin plugin does all the magic for you:
  - **gst-launch playbin2 uri=file:///home/file.mp4**

A Gstreamer application example

- Freescale provides a Gstreamer based application called ‘gplay’ and its usage is very straightforward::

- **gplay file.mp4**

```
FSL_PLAYER_01.00_LINUX build on Jun 30 2010 16:14:33
    [h]display the operation Help
    [p]Play
    [s]Stop
    [e]Seek
    [a]Pause when playing, play when paused
    [v]Volume
    [m]Switch to mute or not
    [>]Play next file
    [<]Play previous file
    [r]Switch to repeated mode or not
    [f]Set full screen or not
    [z]resize the width and height
    [t]Rotate
    [i]Display the metadata
    [x]eXit
```

A Gstreamer application example (cont.)

- *gplay* does use *playbin2* and allow user control inputs, such as play/stop/seek/volume,next, fullscreen, etc
- Its source is available at: **ltib/rpm/BUILD/gst-fsl-plugin-1.9.8/tools/gplay**
- Reference to understand how Gstreamer application programming works

USB Webcam Loopback

- A UVC compliant webcam is required. To check the UVC supported webcams: http://www.ideasonboard.org/uvc/#devices

- Connect a UVC USB camera to the MX53QSB

- The kernel should recognize it and display some messages such as:

```
usb 1-1: new high speed USB device using fsl-ehci and address 3
usb 1-1: device v0ac8 p332d is not supported
uvcvideo: Found UVC 1.00 device Vimicro USB2.0 Camrera (0ac8:332d)
input: Vimicro USB2.0 Camrera as /devices/platform/fsl-ehci.0/usb1/1-1/1-1:1.0/input/input5
```

Run the pipeline: **gst-launch v4l2src ! autovideosink**

Encoding from camera to a file

- Capture from the camera and encode it as a H.264 D1 file

```
gst-launch v4l2src always-copy=false queue-size=16 num-buffers=900 ! queue !
capsfilter caps='video/x-raw-yuv,format=(fourcc)YUY2, width=640, height=480,
framerate=30/1' ! ffmpegcolorspace ! capsfilter caps='video/x-raw-
yuv,format=(fourcc)I420, width=640, height=480' ! mfw_vpuencoder ! avimux!
filesink location= /home/test.avi
```

- Play the resultant file:

- **gplay test.avi**

Streaming over the Network

- Capture image from the camera, encode it in MPEG4, send it to the network via the internal FEC to a remote PC:

- export HOST=10.29.240.182 (This is the IP address of the remote PC that will receive and decode the video stream sent by the MX53QSB)

- **gst-launch-0.10 -v mfw_v4lsrc capture-width=640 capture-height=480 ! mfw_vpuencoder width=640 height=480 codec-type=std_mpeg4 ! rtpmp4vpay send-config=true ! udpsink host=$HOST port=5434**

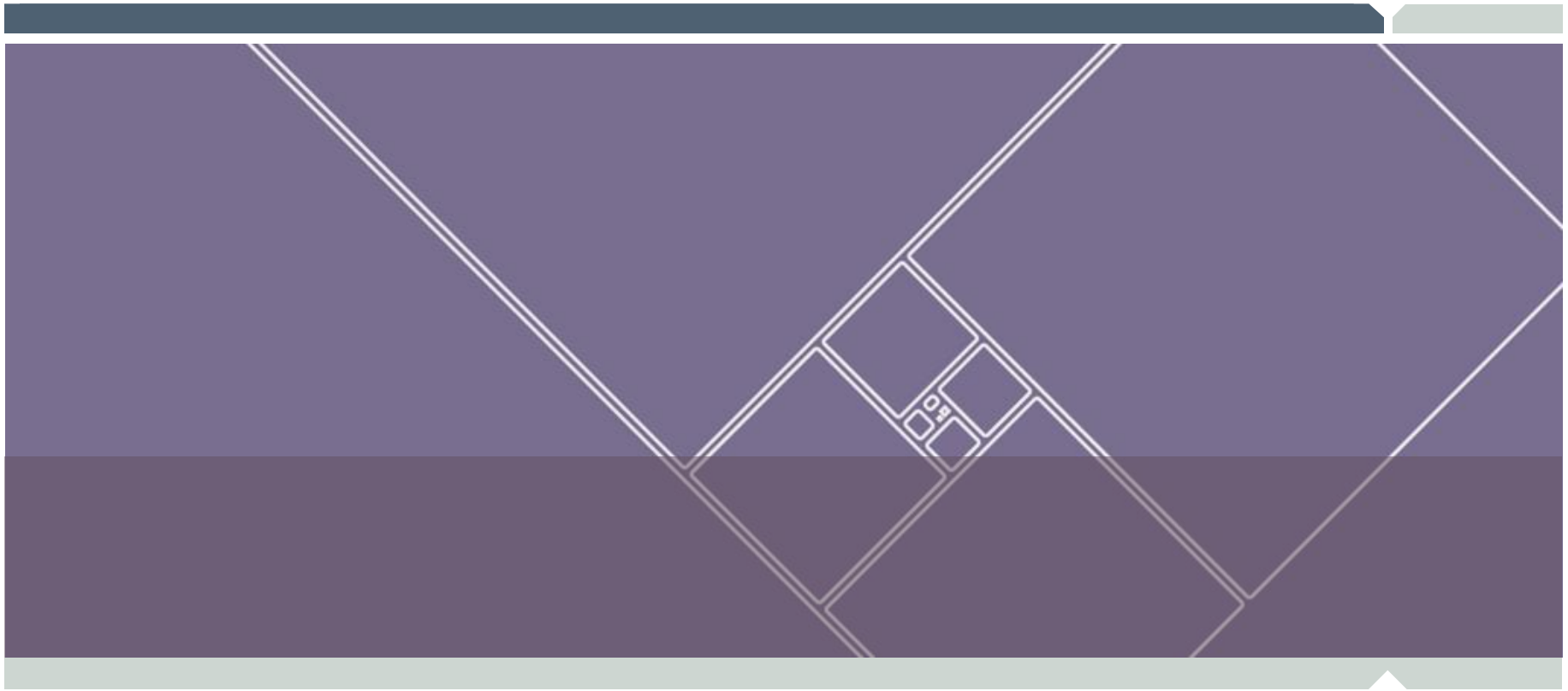
Streaming over the Network (cont.)

- It is assumed that VLC player is installed in the remote Linux PC

- Double click on the mpeg4.sdp file. Just copy the content below to a text file and save it as streaming.sdp

```
v=0
m=video 5434 RTP/AVP 96
b=AS:63
c=IN IP4 10.29.240.190
a=rtpmap:96 MP4V-ES/90000
a=fmtp:96 profile-level
   id=4;config=000001b004000001b5911300000100000001200oc888800f514043c14
   103;
```

- The PC will receive and decode the stream



Performance

Software Decoding versus Hardware Decoding

- Software Decoding

gst-launch filesrc location=/home/Fast_and_Furious.mp4 ! qtdemux ! ffdec_h264 ! mfw_v4lsink

PID PPID USER STAT VSZ %MEM %CPU COMMAND

2277 2271 root S 44760 11% **97%** /usr/bin/gst-launch-0.10 filesrc

- Hardware Decoding via VPU

gst-launch filesrc location=/home/Fast_and_Furious.mp4 ! qtdemux ! mfw_vpudecoder ! mfw_v4lsink

top:

PID PPID USER STAT VSZ %MEM %CPU COMMAND

2270 2262 root S 28324 7% **4%** /usr/bin/gst-launch-0.10 filesrc locat

1080p Decoding

- Instead of placing the 1080p in the NFS rootfs (where network throughput can affect the performance) , let s place it into a DDR buffer directly:
  - mkdir /tmp/shm
  - mount –t tmpfs /dev/shm /tmp/shm
  - cp file1080p.mov /tmp/shm

- Hardware Decoding via VPU
  - gst-launch filesrc location=/mnt/shm/file080p.mov ! qtdemux ! mfw_vpudecoder ! mfw_v4lsink
  - CPU Usage: 5%

1080p Decoding at 160MHz

- Set the processor frequency at 160 MHz
  - echo 160000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
  - gst-launch filesrc location=/mnt/shm/file1080p.mov ! qtdemux ! mfw_vpudecoder ! mfw_v4lsink
  - CPU Usage: 17%

720p Encoding

- MX53SMD board has a OV5642 camera that is capable of capturing at 1280x720 at 30fps
- 720p Capture and H264 Encode to a File

```
gst-launch mfw_v4lsrc -v --gst-debug=2 capture-mode=4 capture-width=1280 capture-
    height=720  num-buffers=2200 ! mfw_vpuencoder codec-type=std_avc width=1280
    height=720 loopback=true ! filesink location=video.h264
```

CPU usage: 1%

- 720p Capture/Encode/Decode/Display out

```
gst-launch mfw_v4lsrc -v --gst-debug=2 capture-mode=4 capture-width=1280 capture-
    height=720  num-buffers=2200 ! mfw_vpuencoder codec-type=std_avc width=1280
    height=720 loopback=true ! mfw_vpudecoder parser=false loopback=true ! mfw_v4lsink
```

CPU usage: 3%

720p Encoding at Lower Frequency

- Set the CPU frequency at 160MHz:

```
echo 160000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

- Capture at 720p and encode it to a file:

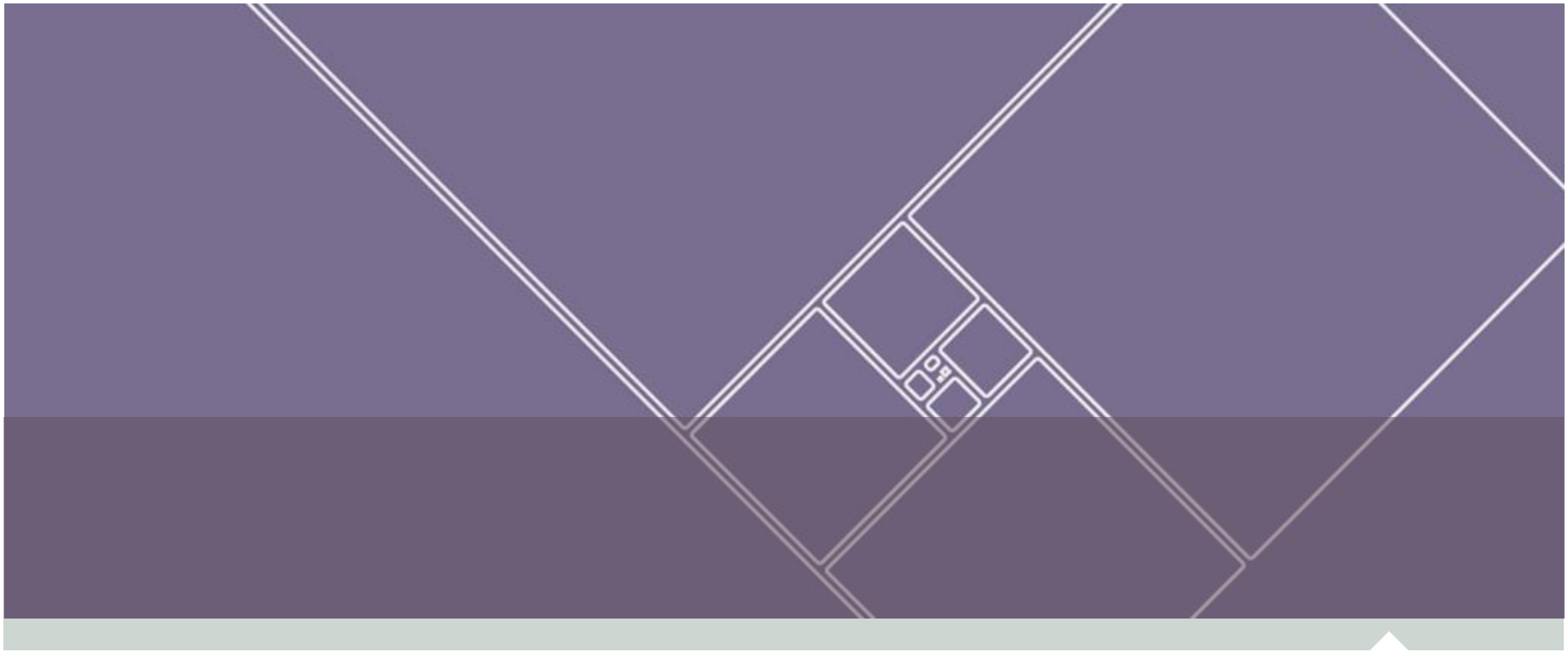
```
gst-launch mfw_v4lsrc capture-mode=4 capture-width=1280 capture-height=720  num-buffers=2200 ! mfw_vpuencoder codec-type=std_avc width=1280 height=720 loopback=true ! filesink location=video.h264
```

CPU usage: 6% at 160MHz

- Capture at 720p/encode/decode/display output:

```
gst-launch mfw_v4lsrc capture-mode=4 capture-width=1280 capture-height=720  num-buffers=2200 ! mfw_vpuencoder codec-type=std_avc width=1280 height=720 loopback=true ! mfw_vpudecoder parser=false loopback=true ! mfw_v4lsink
```

CPU usage: 18% at 160 MHz



Q & A