TI | Developer Conference
February 28-March 2, 2006 • Dallas, TX

NAVI Applications and DaVinci™ Technology

(1) Networked Audio and Video Innovation

Jean-Michel Mercier

Application Manager
ATEME
Jm.mercier@ateme.fr

ateme

Agenda

- Introduction
- DaVinci™ Technology and NAVI Requirements
- Software design for NAVI on DaVinci
- Conclusion

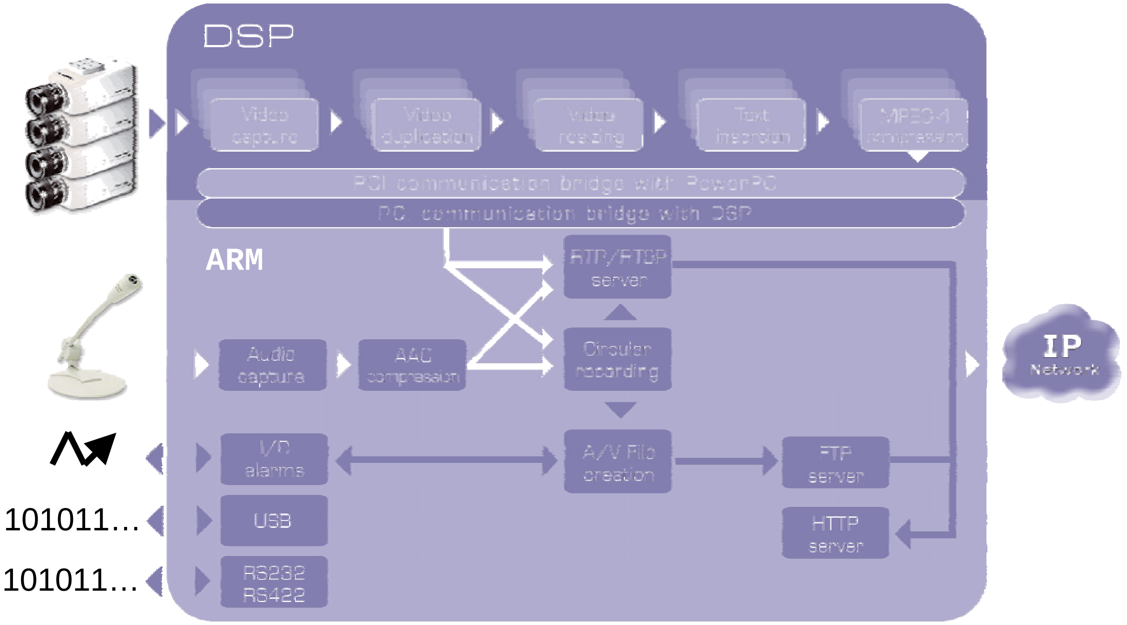
Introduction

- **Market requirements**
  - Good quality and low bandwidth
  - Low latency
  - Support of any type of videos source
  - Standard streaming protocols
  - Intelligence on the device
  - Easy software design
  - Fast track to market

Introduction

- **DaVinci is a perfect chip for NAVI**
  - DSP for video compression and analytics
  - ARM for streaming, recording and user interface

- **Example of existing application on mixed architecture**



Agenda

- Introduction
- DaVinci™ and NAVI Requirements
- Software design for NAVI on DaVinci
- Conclusion

Codec Offer from TI and 3P

Video

H.263 prof. 0	Encoder + Decoder
MPEG-2 MP@ML	Encoder + Decoder
MPEG-4 SP and ASP	Encoder + Decoder
H.264 BP/MP	Encoder + Decoder
WM9V MP	Encoder
WM9V MP/AP/VC-1	Decoder
JPEG/MJPEG	Encoder + Decoder

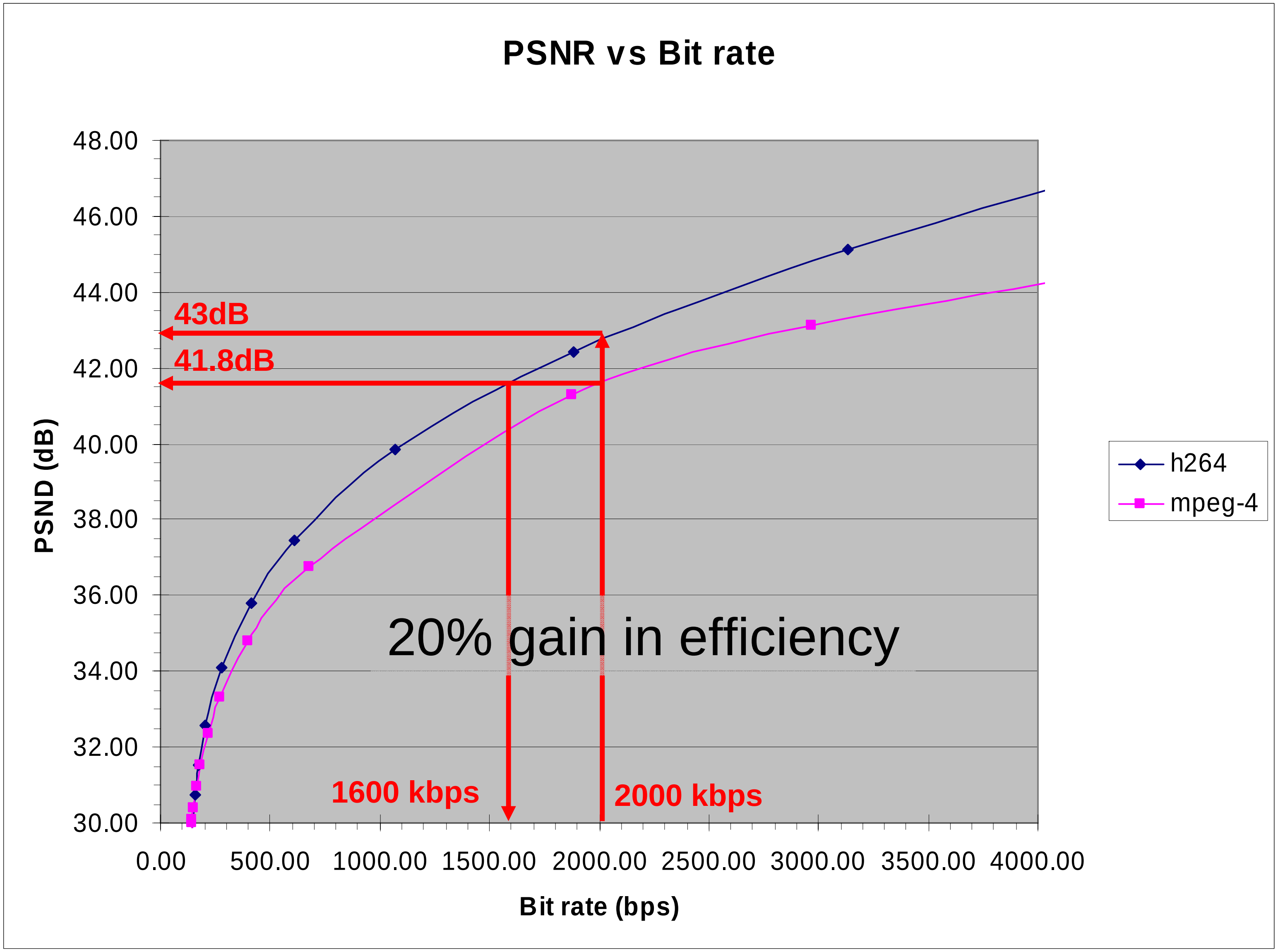
Audio

MPEG Audio Layer 1,2,3	Encoder + Decoder
AAC LC/HE	Encoder + Decoder
WM9A / WM8A	Encoder + Decoder
G.7xx	Encoder + Decoder

Others

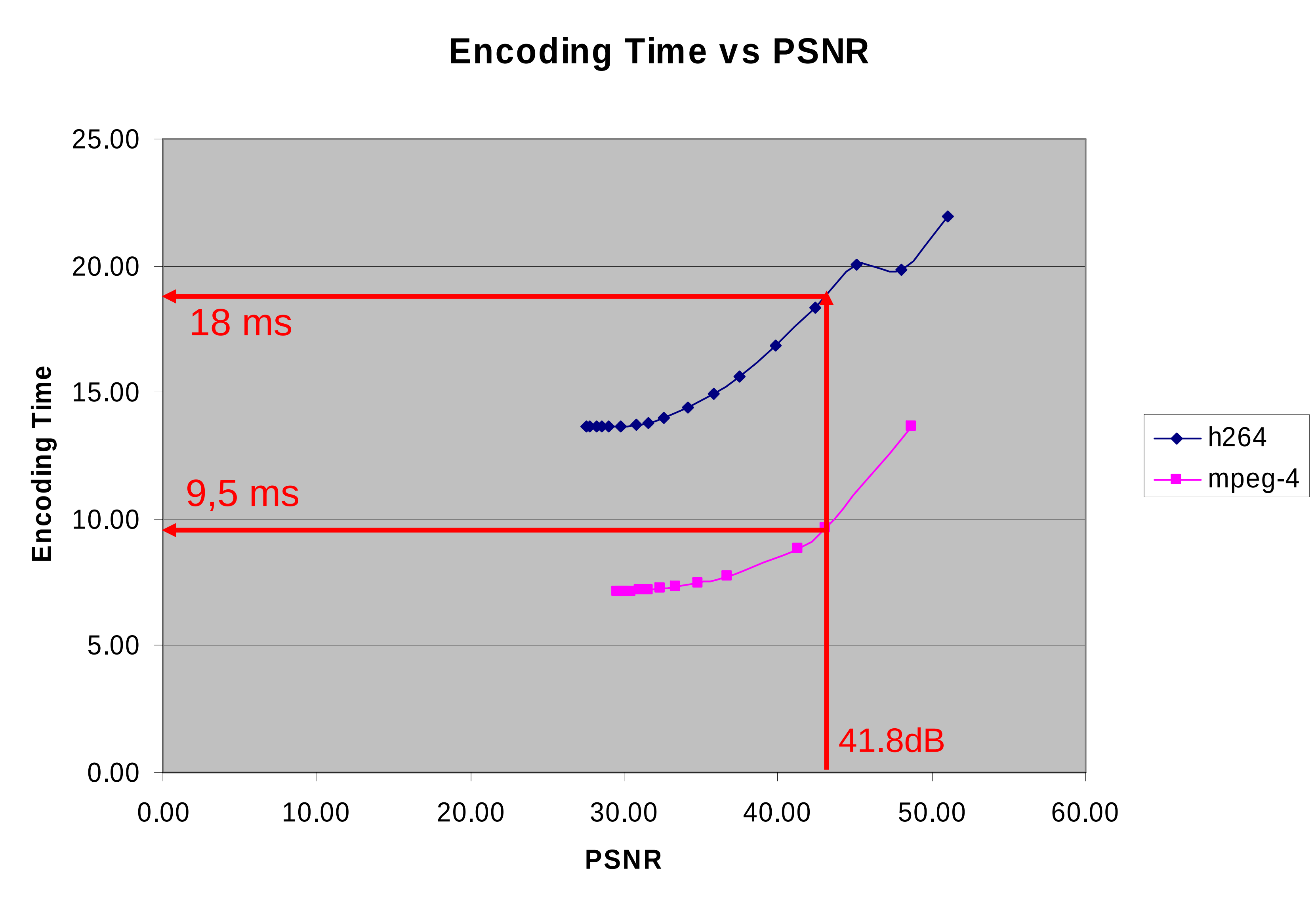
- De-interlace filter
- Resize filter

MPEG-4 / H.264 Comparison



20% gain in efficiency

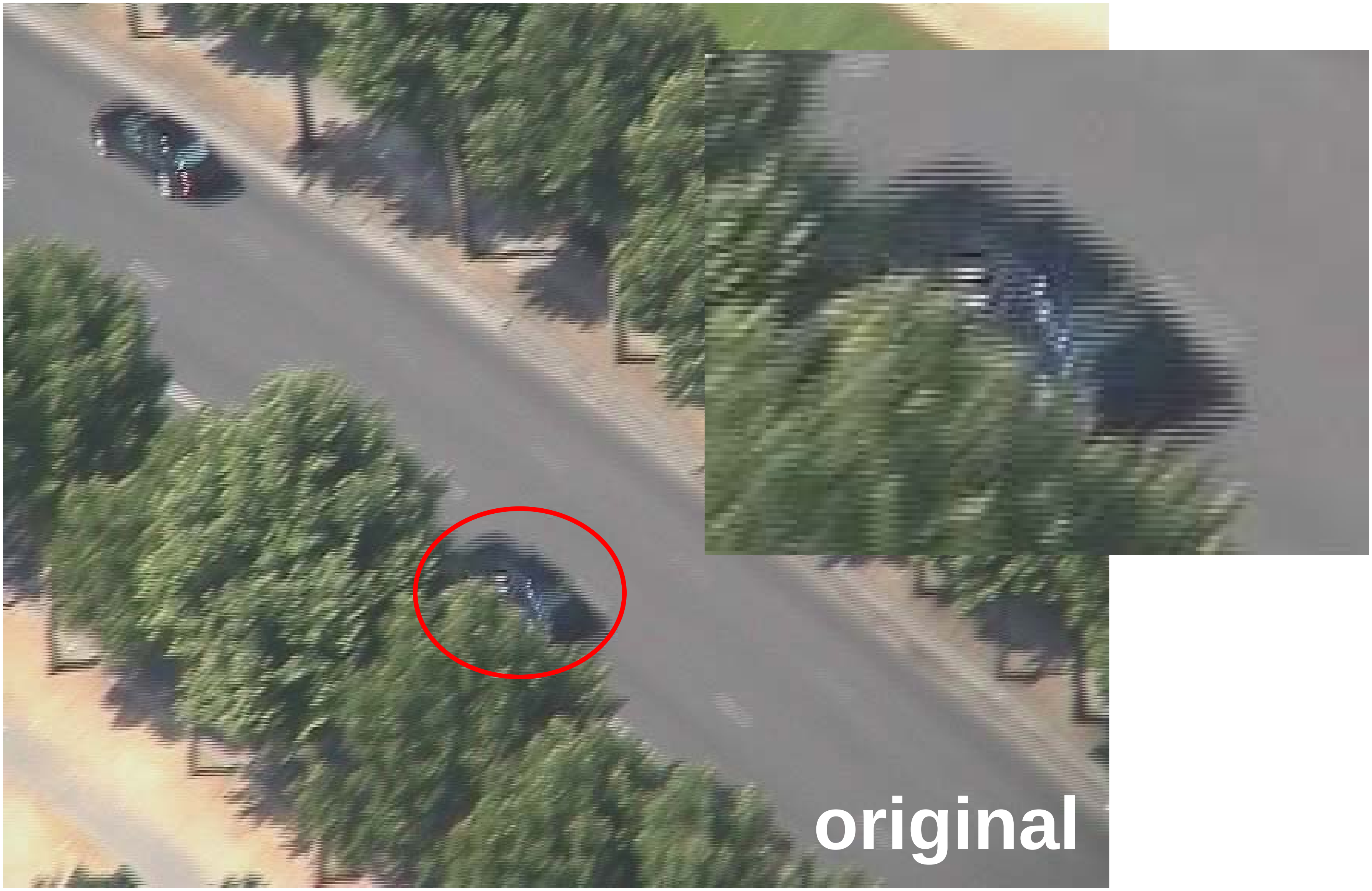
MPEG-4 / H.264 Comparison



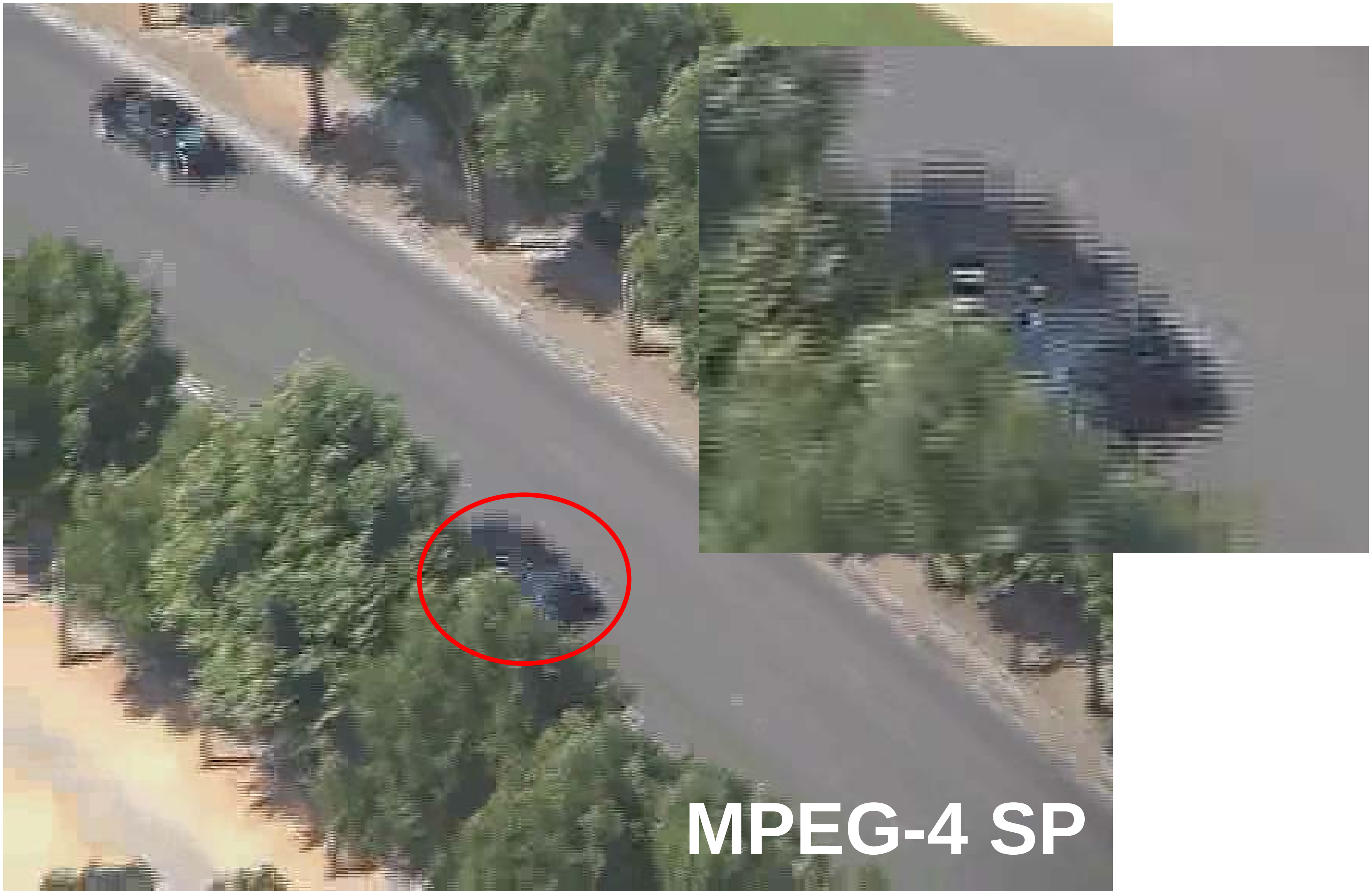
Codec Choice: Warnings

- **Not all implementations are equivalent**
- **Performance is « easy » to have**
  - *What has to be sacrificed then?*
- **Decoder**
  - Compliance
- **Encoder**
  - Quality (motion estimation, decision, regulation is heart of know-how)

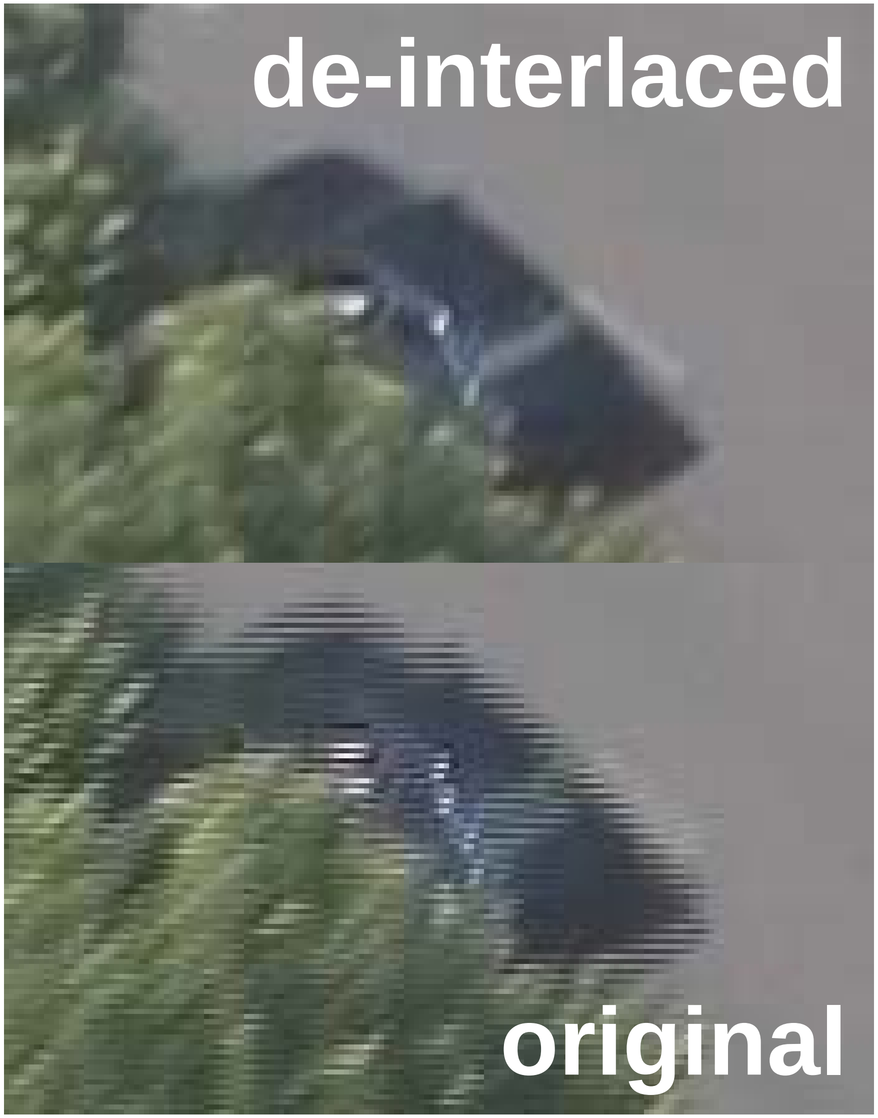
Interlace issues

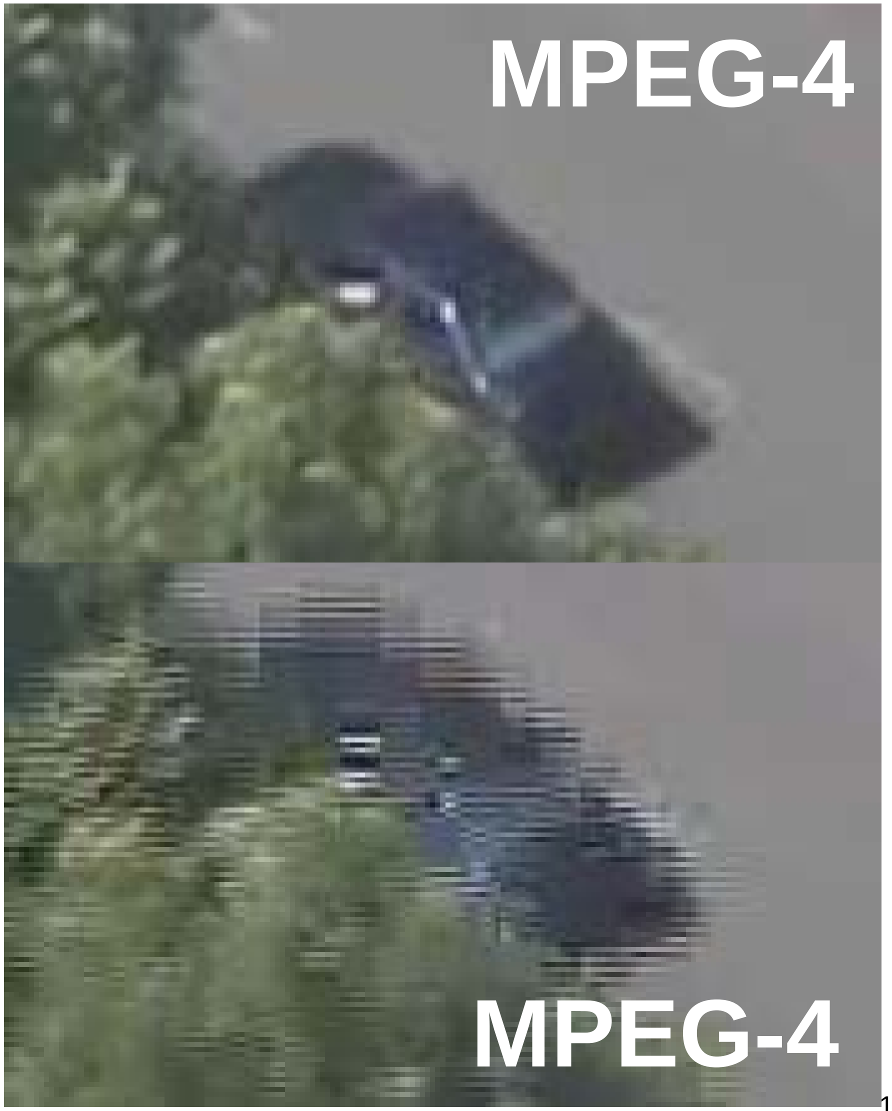


Interlace issues

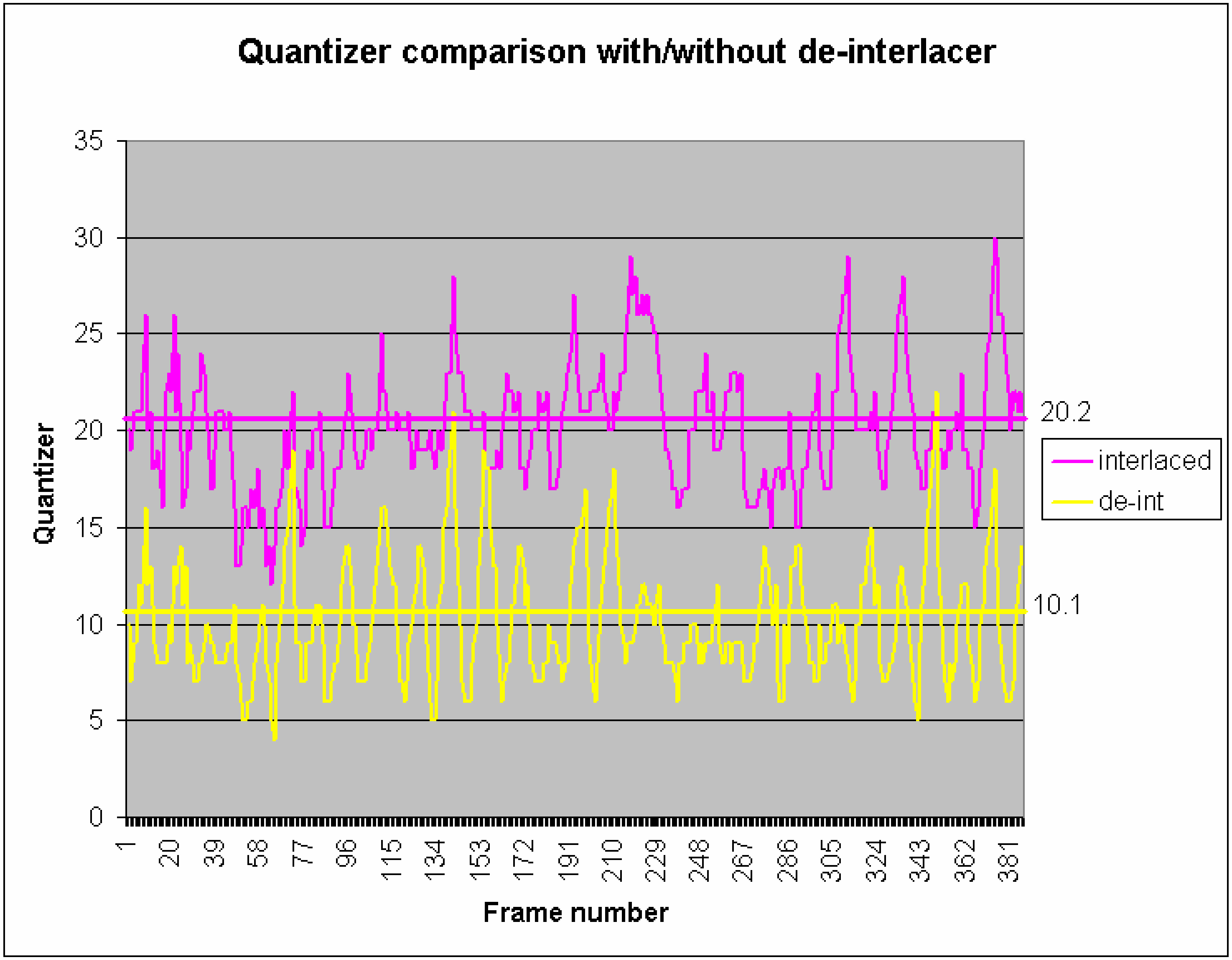


Interlace Issues





Interlace Issues



Video

Do I Need to De-Interlace?

- **Display is 90% PC → Progressive display**
  - Very disturbing artifact (comb artifact)
- **on PC: overload PC (lot of channels)**
- **at Encoder: improve efficiency (MPEG-4)**

	Display on PC	Display on TV
Encode in MPEG-4 SP	Encoder improve coding	Encoder improve coding
Encode in H.264	PC	No need

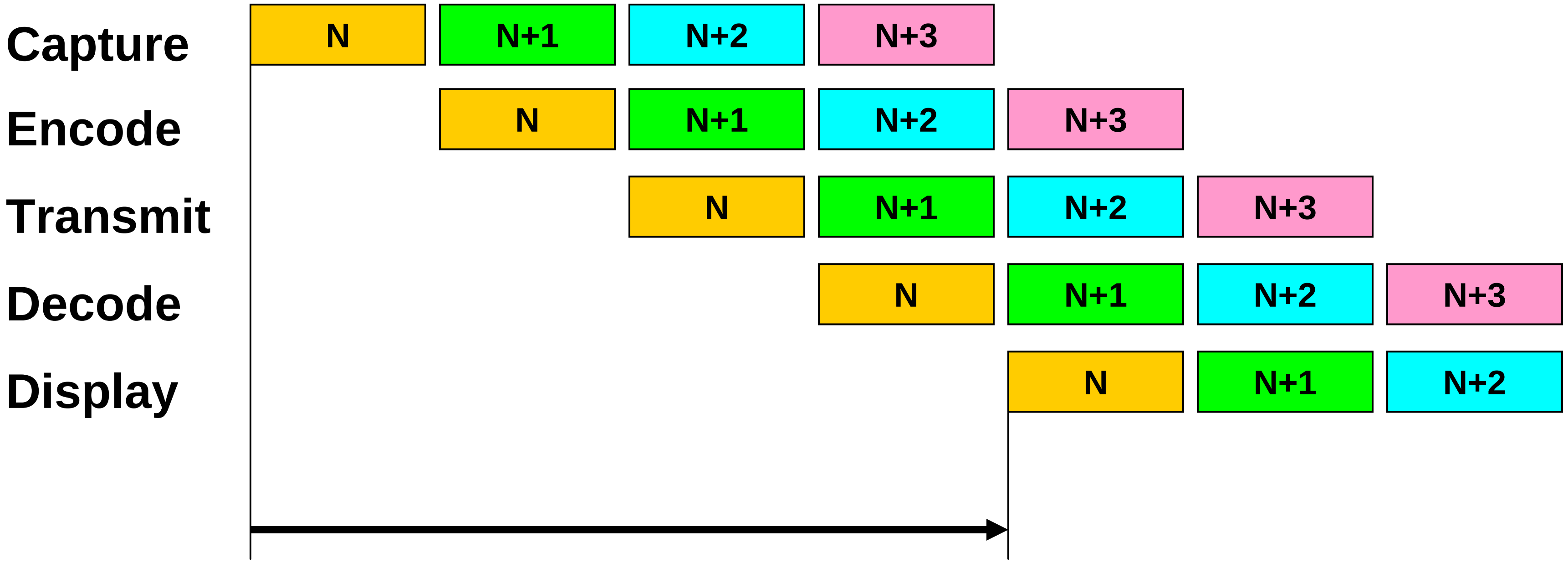
Noise Filtering

- **Encoder will try to encode noise**
- **More data to go through same bandwidth**
- **Quantize more ➔ lower quality**
- **Remove noise ➔ enhance coding efficiency**
- **Simple 3x3/5x5   not much efficient**
- **3D = spatial + temporal**
    - Remove noise
    - Remove flicker

Video

Causes for Latency

- **Pipeline unit is frame**

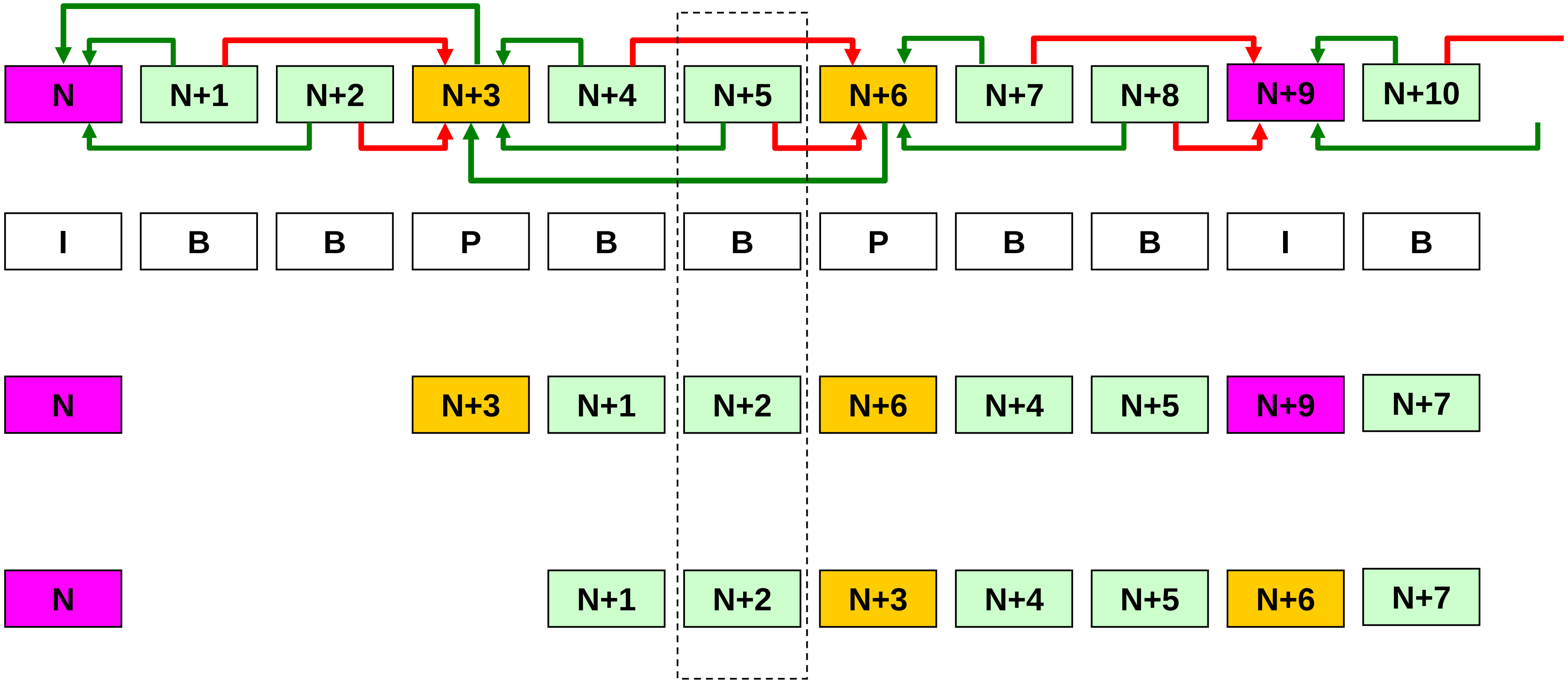


Capture	N	N+1	N+2	N+3			
Encode		N	N+1	N+2	N+3		
Transmit			N	N+1	N+2	N+3	
Decode				N	N+1	N+2	N+3
Display					N	N+1	N+2

Total latency = few frames

Causes for Latency

- B-frames encoding is non-causal



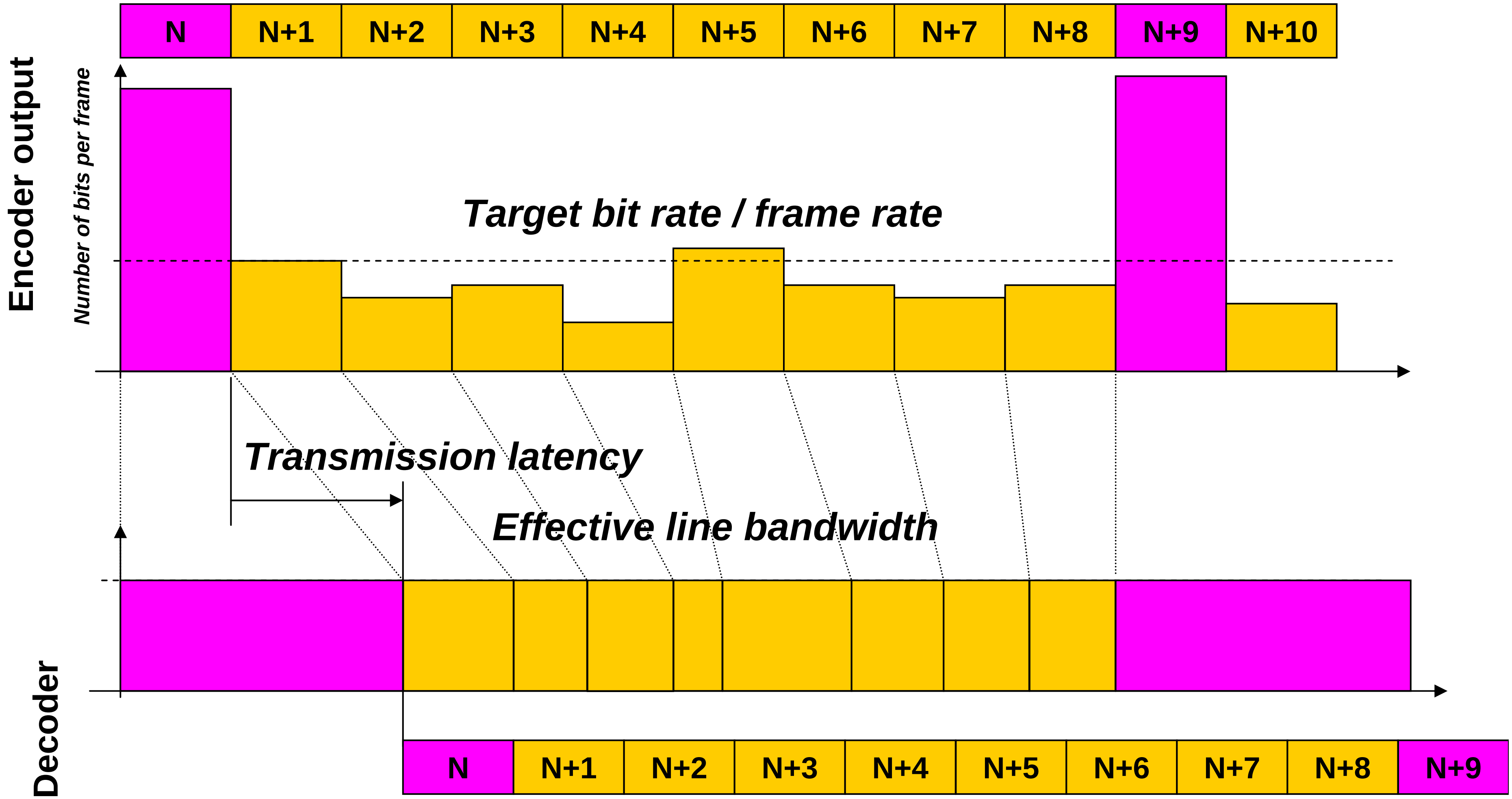
N	N+1	N+2	N+3	N+4	N+5	N+6	N+7	N+8	N+9	N+10
I	B	B	P	B	B	P	B	B	I	B
N			N+3	N+1	N+2	N+6	N+4	N+5	N+9	N+7
N				N+1	N+2	N+3	N+4	N+5	N+6	N+7

Total latency = 3 frames

TI Developer Conference

Causes for Latency

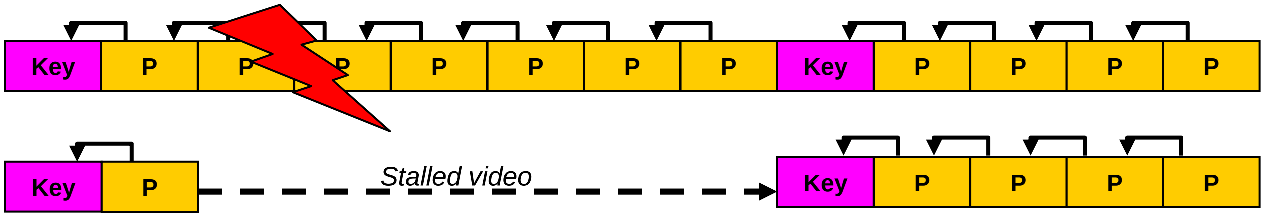
- I frames are bigger than P frames



Error Resilience

For how long can you stop watching?

- **Causes**
  - Network Loose packets (UDP)
  - Bit errors (radio)

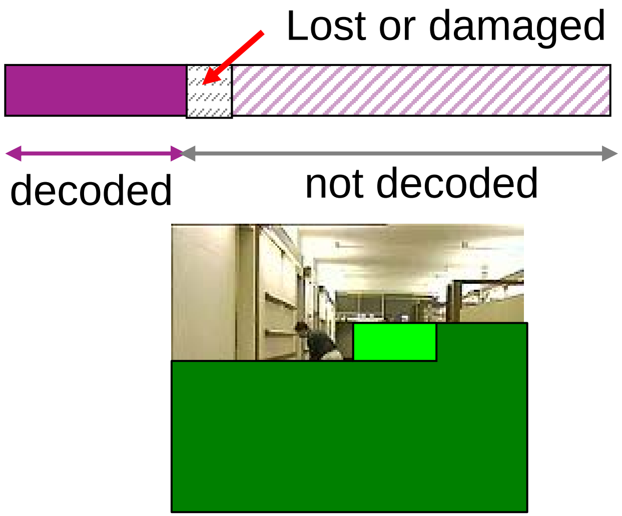


- **Result**
  - Corrupted image
  - Stalled video
- **Wait for I to reconstruct**
- **I period trades-off: better resilience, less efficiency**

Error Resilience Tools

- **MPEG-4**
  - RVLC: Reverse VLC
  - Data partitioning
  - Resync markers & Video Packet
- **H.264**
  - Slices

Normal encoded frame

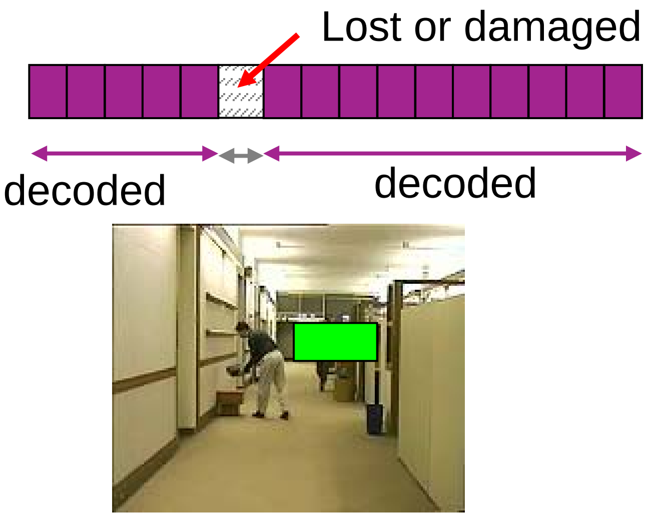


Lost or damaged

decoded

not decoded

Slice/packets encoded frame



Lost or damaged

decoded

decoded

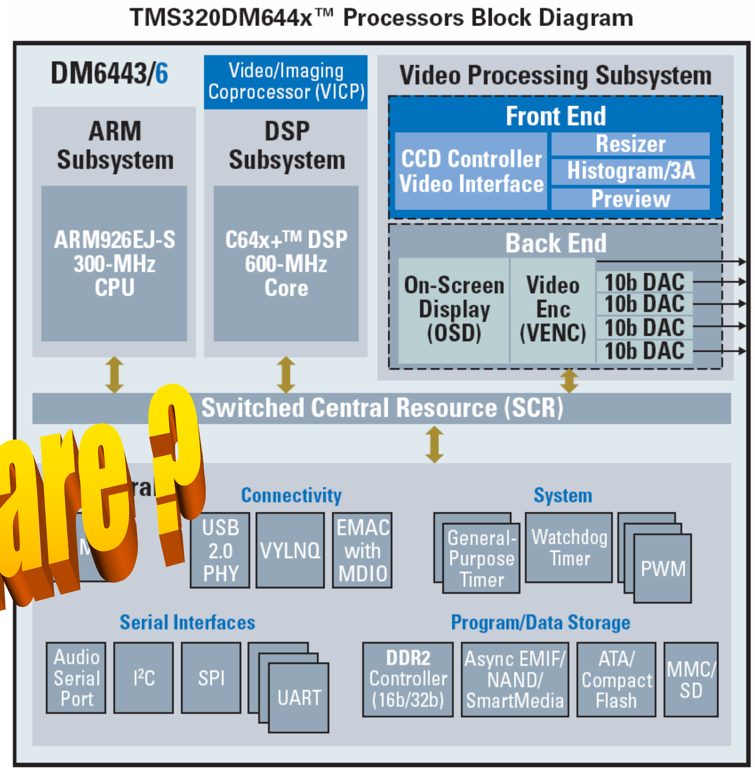
Video

Agenda

- Introduction
- DaVinci™ and NAVI Requirements
- Software design for NAVI on DaVinci
- Conclusion

Software Design

- Dual core
- Shared memory
- Shared peripherals



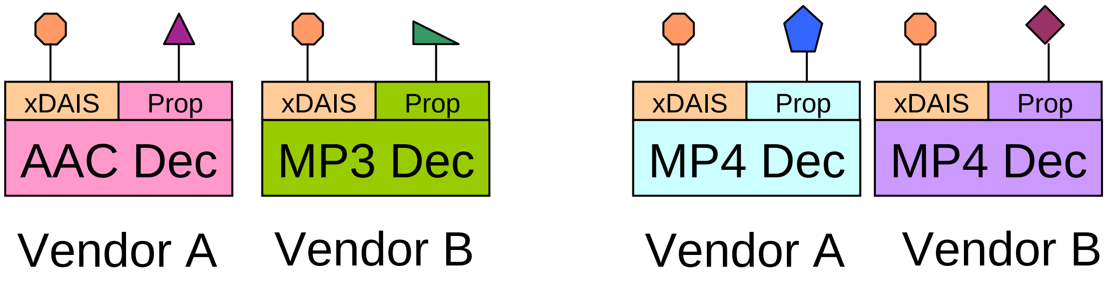
Programmer's nightmare ?

Software Design

- **Codec Engine**
  DaVinci made easy for everyone
- **Assumption #1**
  Don’t touch the DSP if you don’t need
- **Stay on Linux**
  All peripherals use standard drivers: sockets, file system, serial, V4L2 …
- **DSP is a “black-box” coprocessor**
  Forget about it, call API
- **Use open framework for A/V application**
  Platform portability, reference applications

xDAIS Algorithm Standard

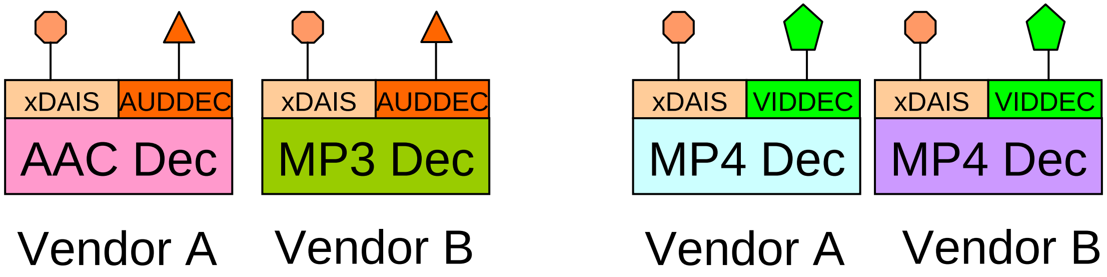
- **Standard API for resources management**
  - algNumAlloc : how many resources?
  - algAlloc : describe resources
  - algInit : init from those resources
- **Proprietary API for processing**
  - Proprietary configuration and parameters
  - Proprietary function call



- **Do not ease switching from MPEG-4 to H.264**
- **Do not ease to switch from VENDA to VENDB**

xDM – Digital Media Extension

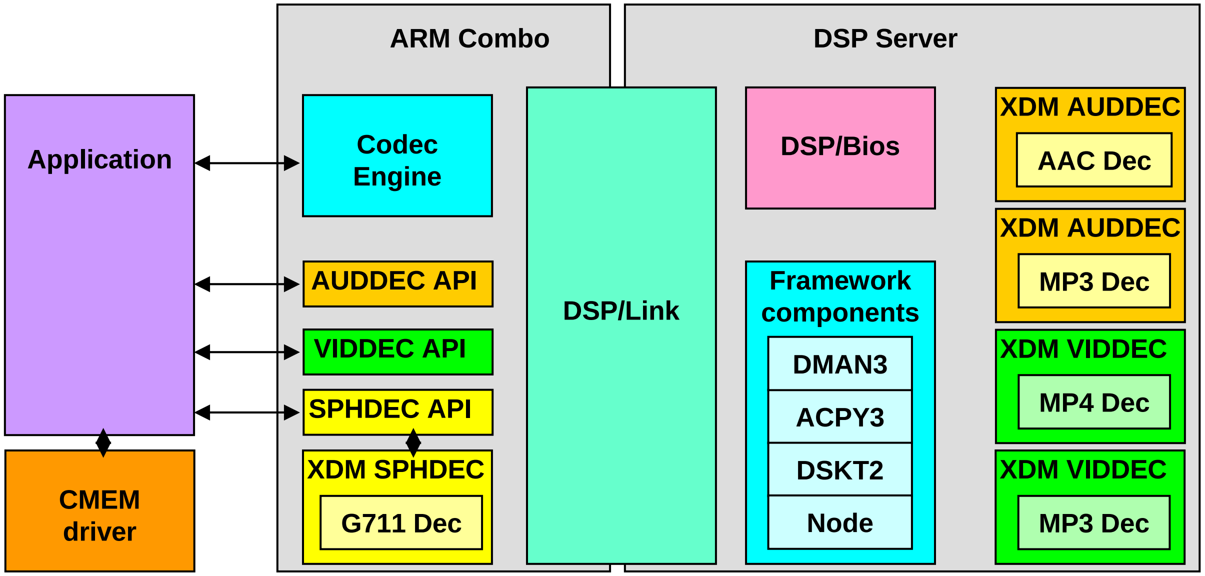
- **Define 8 standard classes of algos:**
  - Video – Imaging – Speech – Audio: VISA
  - Encoders and decoders
- **For each class, defines:**
  - Configuration structure
  - Function calls with full prototype



- **Application writer benefits:**
  - Plug'n'Play architecture
  - Each change of codec/provider

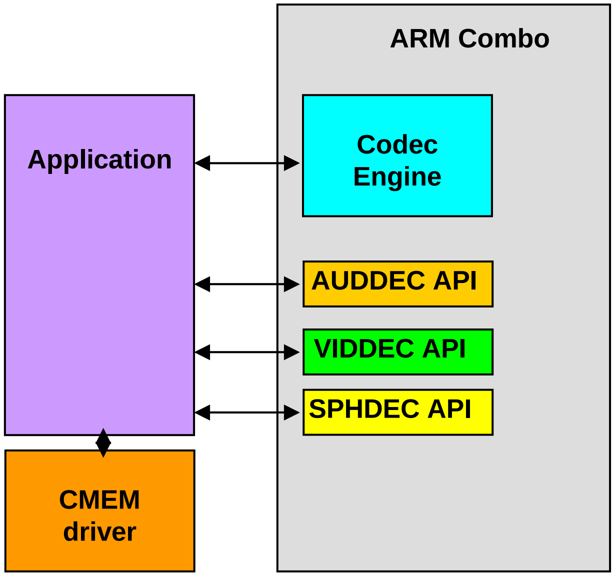
Codec Engine

- Pre-built codec combo/server
- Use DSP algs from ARM at no pain
- Transparent Remote Procedure Call
- Transparent mix of host (ARM) or DSP algs
- Deal only with Linux programming



Codec Engine

- All you have to remember is



Using Codec Engine

```
CERuntime_init();
myCE = Engine_open( “myengine”,…)
myEnc = VIDENC_create(myCE, “mpeg4”,…)
allocate_buffers()
do {
  capture_frame( &frame );
  VIDENC_process( myEnc, &frame, &bits…)
  send_stream( &bits )
} while (!end);
VIDENC_delete( myEnc );
Engine_close( myCE );
```

See Live Code in Action

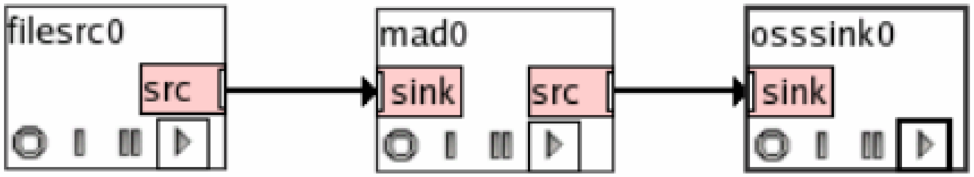
- Browse a sample application
- Run the demo
- Change MPEG-4 to H.264
- Rebuild and run again

Need for an A/V Framework

- **Codec Engine handles DSP calls**
- **Need a solution for**
  - A/V Sync
  - Streaming/Container handling
- **Linux on ARM give access to a lot of solutions**
  - OpenML http://www.khronos.org/openml/
  - GStreamer http://gstreamer.net

What is GStreamer?

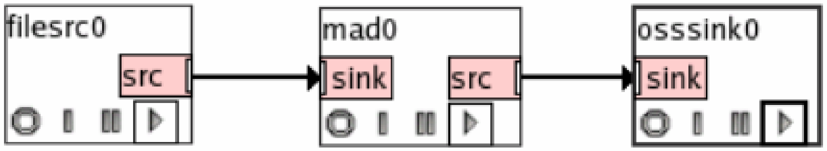
- **Pipeline oriented framework**
  - Graph based design: connect boxes



  - Content agnostic
  - GStreamer core is the engine
  - Plug-ins handles the work
  - Runs on desktop or embedded Linux
  - Only depends on gLibc
- **Already used by a lot of desktop applications**

How Does It Work?

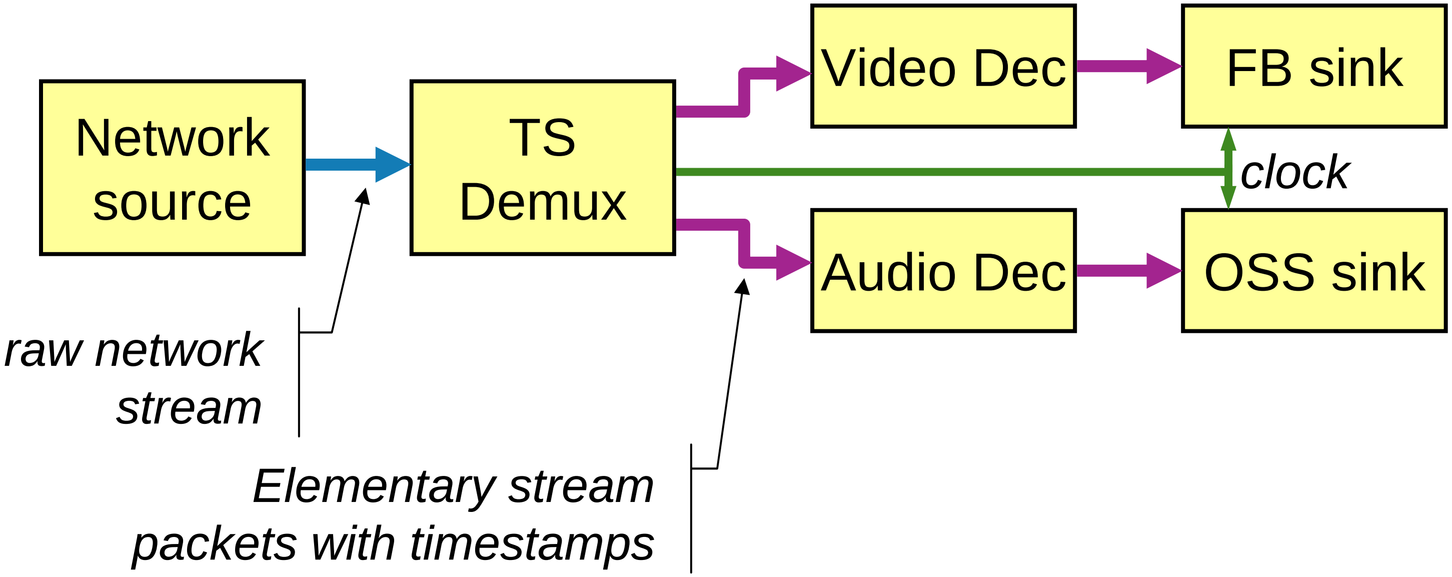
- **Source filters get data from the real world**
  - File parsers: AVI, MP4, ASF readers
  - Network client: UDP, TCP, RTP, RTSP
  - Capture: V4L2, OSS
- **Sink push data to the real world**
  - File writers
  - Network server/streamer
  - Rendering: Frame buffer, OSS



- **Nodes process data**
- **Elements are connected through pads**
- **Data types are negotiated**
- **Buffers are transported through pointers**
- **Copy is avoided as much as possible**

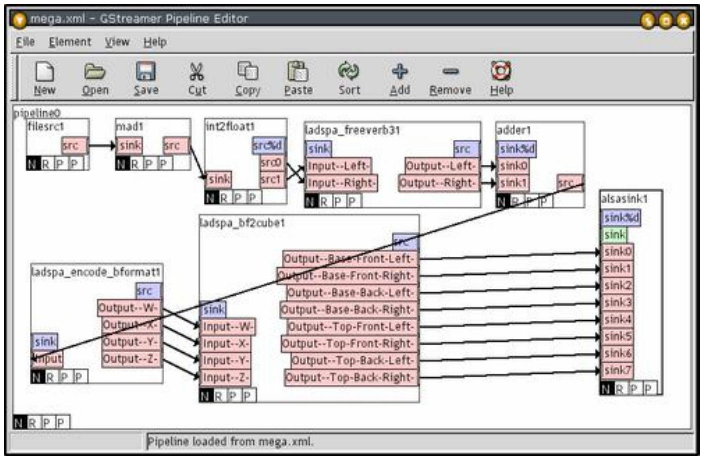
A/V Sync

- **Flexible clock scheme**
- **Sink and Source can be master or slave**
- **Example: network player**
  - Demux is master
  - Sinks are slaves



Tools and Utilities

- **gst-editor**
  - Graphical graph editor
  - Export design in XML
  - Play graph in GST-launch



Pipeline loaded from mega.xml.

- **gst-launch**
  - Command line utility
  - Create and play a graph by command line or XML

```
gst-launch “mysong.mp3” ! mp3dec ! osssink
```

- **gst-inspect**
  - List installed plug-ins

- **gst-register**
  - Register plug-ins capabilities

Agenda

- Introduction
- DaVinci™ and NAVI Requirements
- Software design for NAVI on DaVinci
- Conclusion

Conclusion

- **Codec Engine makes using DaVinci easier**
- **Framework helps for fast track to market**
- **Skilled 3P will help you go though**
- **ATEME specifics for DaVinci**
  - HW and SW expertise
  - Own enhanced codec or complementary IPs
    - H.264 Main Profile Encoder
    - MPEG-4 ASP Encoder
    - MPEG-4 ASP & DivX Decoder
    - Streaming protocols
  - A/V application and GStreamer expertise

NAVI Applications and DaVinci™ Technology

Jean-Michel Mercier

Application Manager
ATEME
jm.mercier@ateme.fr

SEE THE FUTURE

CREATE YOUR OWN