EtherWaves

Automotive SDR over multi-core Linux

‘Digital Radio Connecting the Car’
Berlin, Germany

Nov, 2012

EtherWaves

Scope of discussion

- Automotive Linux
  - New and exciting tools for developers
  - Implications on DMB system partitioning
- The mission : DMB as SDR over Linux
  - Motivations
  - SDR feasibility and performance
- DMB stack done the Linux way
  - From monolithic to modular design
  - Potential area for wider standardization

The GENIVI platform

- The offering of Linux
  - De-facto standard GNU tools
  - Mature OS and libraries enable setting focus on the application rather than infrastructure
  - Open-source, documentation and community support - better ROI on our learning curve
- Technology push from the Smartphone industry
  - Vendor optimized algorithms for handling popular multimedia offloads s/w performance
  - Cutting edge computer-on-chip
    - ATOM and ARMv7 (NEON) over 1Ghz speeds
    - Larger memory space not present before in DMB receiver ICs
    - Dual and quad multi-core

Re-Partitioning of the DMB receiver

- Fixed design constraints
  - Physical layer RF & modem properties
  - Different continents = different standards
- What does change ?
  - DMB becomes on source out of many IVI platform
  - Platform GUI composed of multiple visual output.
  - Data applications keep adding to the broadcast stack.
  - Multichannel audio: back, front, surround, delayed playback , mixing
  - Variety of radio handover tactics between systems FM/AM-DMB , dual DMB, diversity, Internet radio
  - Different continent = different software

Mission

- Software defined receiver under Linux
  - DMB
  - Dual DAB or diversity
  - DAB-FM
  - Dual FM or diversity

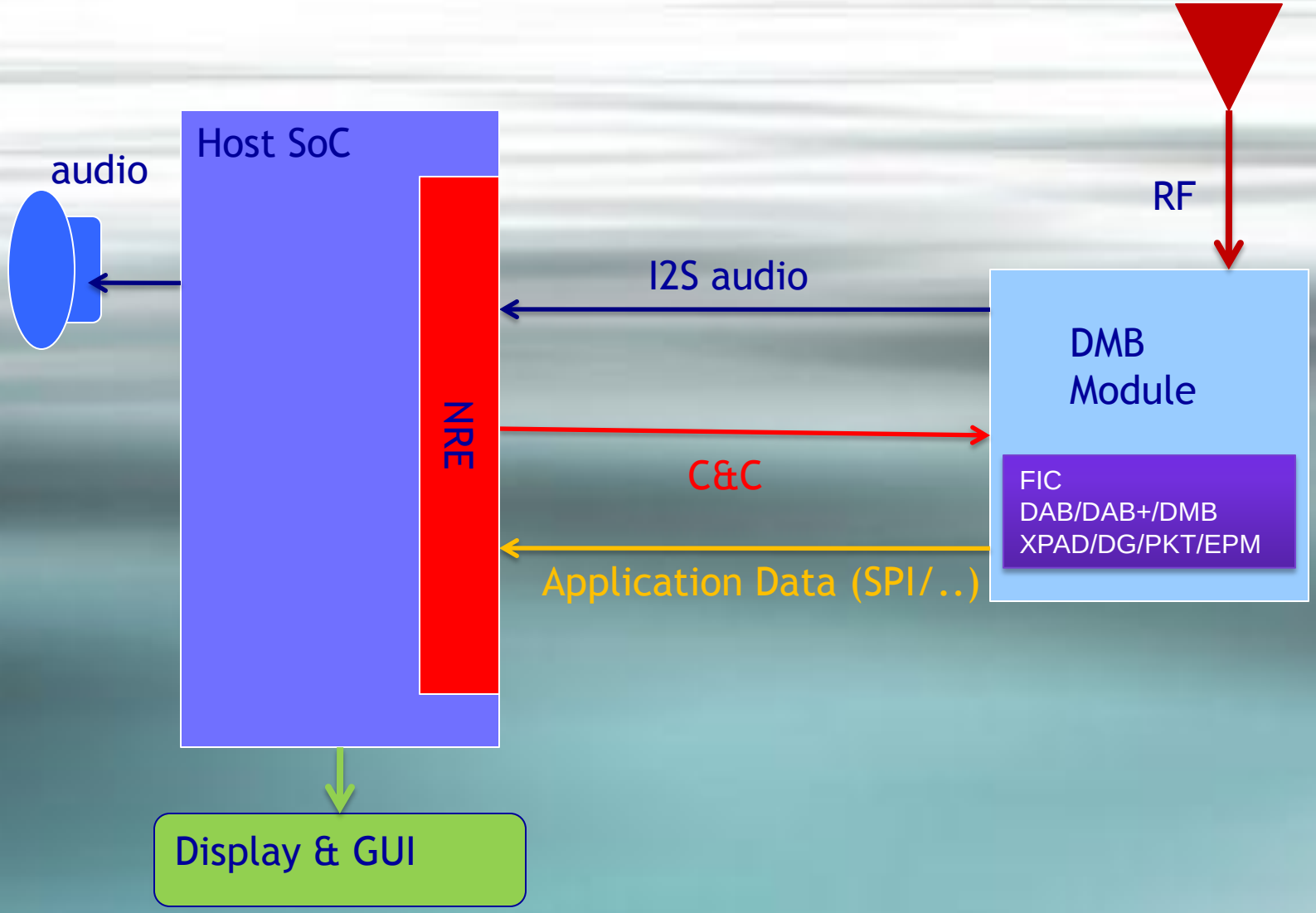
Why SDR? (..reminders)

- Sharing of hardware resources
  - Cost of MIPS & memory already attributed to ‘other’ applications
  - Excessive SoC resources enable innovative features that are not pre-designed into custom IC (e.g. diversity tactics)
- Reducing NRE on H/W qualification
  - Single hardware, multiple standards
  - Hardware qualification perimeter reduced
- Flexible interfaces between modules
  - API and datasheet not ‘hardwired’ to H/W module
  - Optimize data distribution between modules (queues, shared memory, data trees, etc)
- Long term investment
  - Capabilities and test results preserved as long term intellectual properly
  - Moor’s law

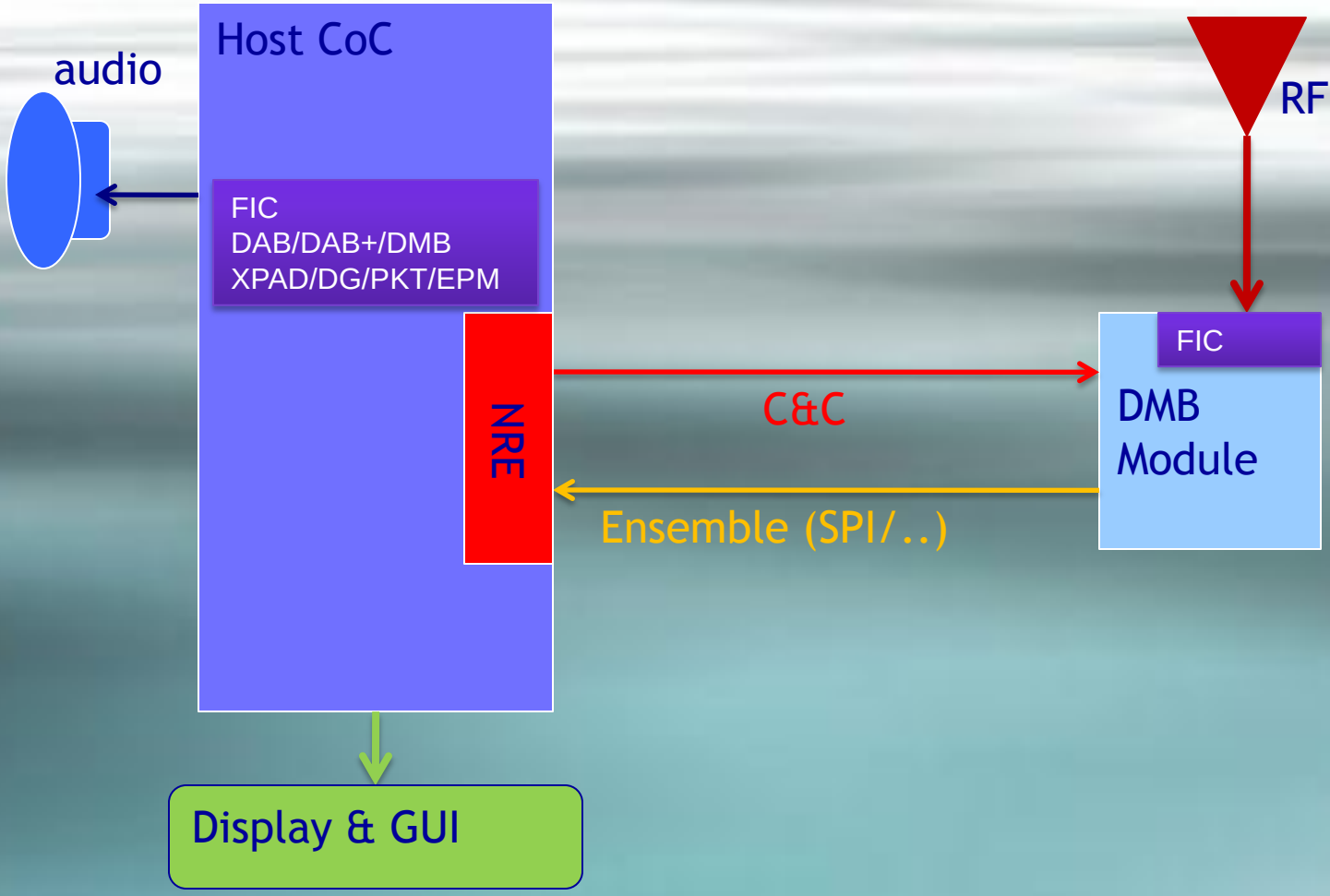
SDR also has cons

- Power consumption
  - Negligible effect within automotive IVI platform
- Software integration challenges
  - Predicting final CPU load balancing for a set of applications running concurrently
  - Faster change rate compared to h/w methodology requires repeated testing
  - Tendency to change the spec and push the envelope
  - Need inter process protection (..Linux to the rescue)

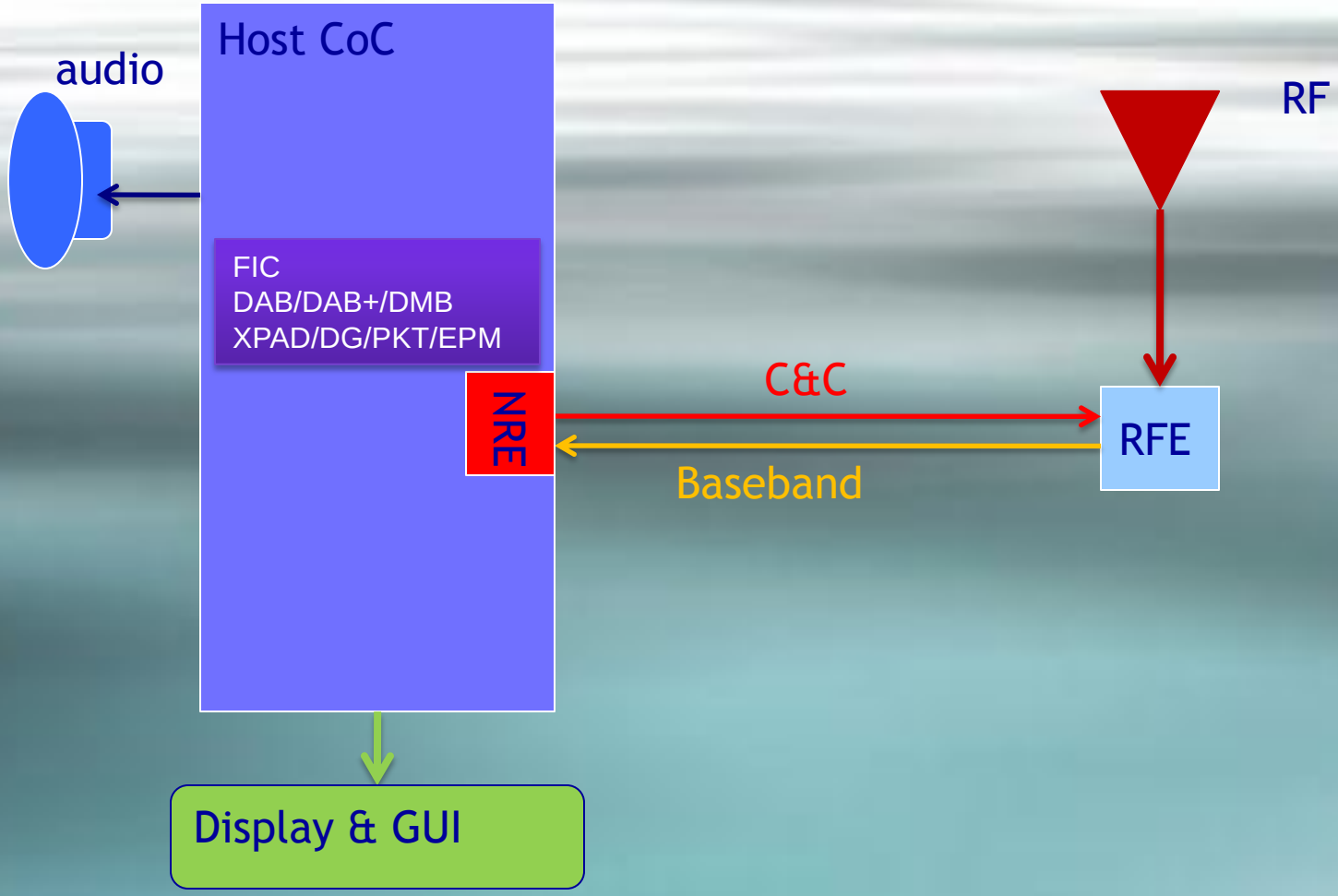
Legacy system



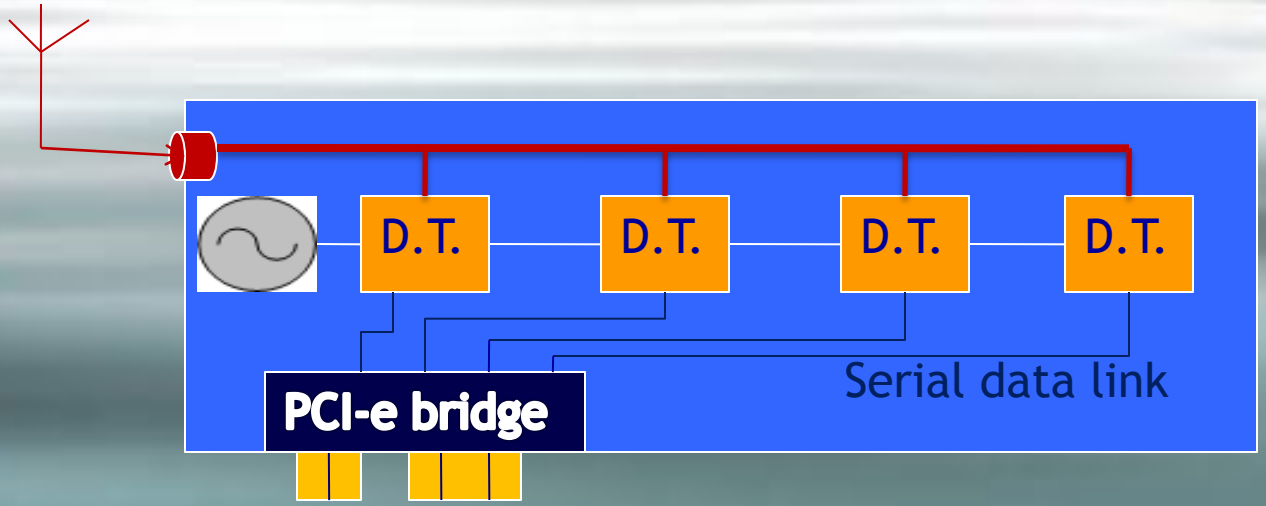
DMB/Linux system



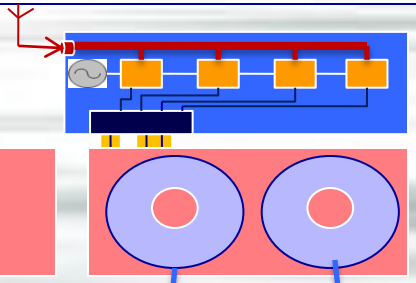
DMB/SDR/Linux system



SDR radio front-end



Simplified SDR stack



Kernel space

Kernel mode drivers

Read()

User space

DAB demod process

Ensemble | (or sub-selection)

EU147 demux

FIC | MSC

Transport

MP2 | DAB+ | DMB demux | PKT data

Media decoders

Mp2 | AAC | MOT | TPEG | H.264

Audio pre-process

Graphic rendering

User interface

External sources

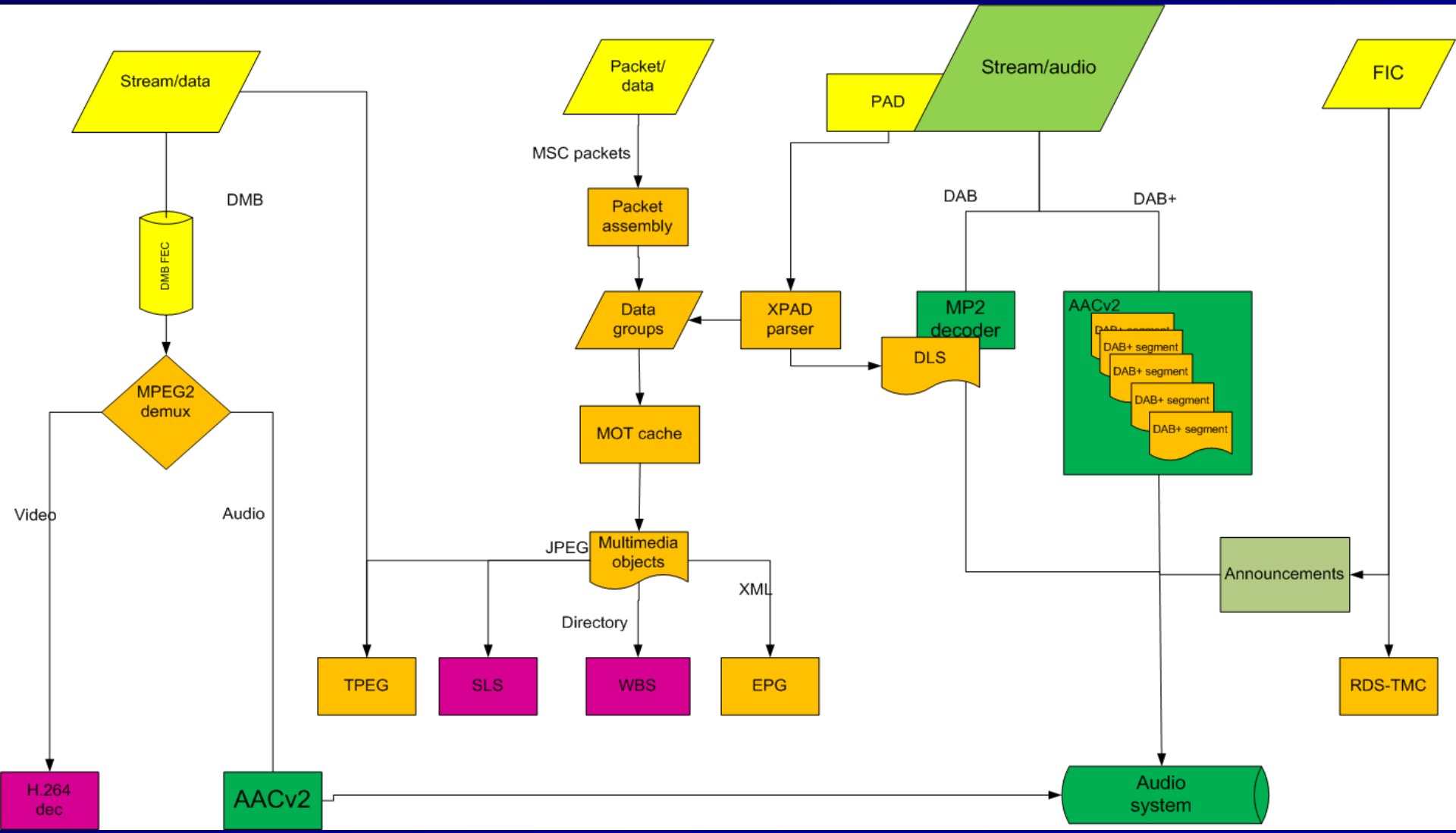
ARMv7 as baseband DSP

- Demodulation with ARMv7
  - PCI-e interface acting as bus master
  - FFT and Viterbi optimized to NEON instructions
- Performance
  - Clock rate: 940 MHz
  - Profile : mp2 Audio channel of 192kbps
  - 310 MHz on single core from baseband to audio
  - Audio decoding & playback alone 47 MIPS
    - Likely to execute on another core
- Conclusion :
  - Maybe not the best DSP in class, but..
  - Enough MIPS on single core for dual tuner receiver.

EtherWaves

The DMB stack

DMB media flows (and they keep adding..)



Gstreamer ‘magic’

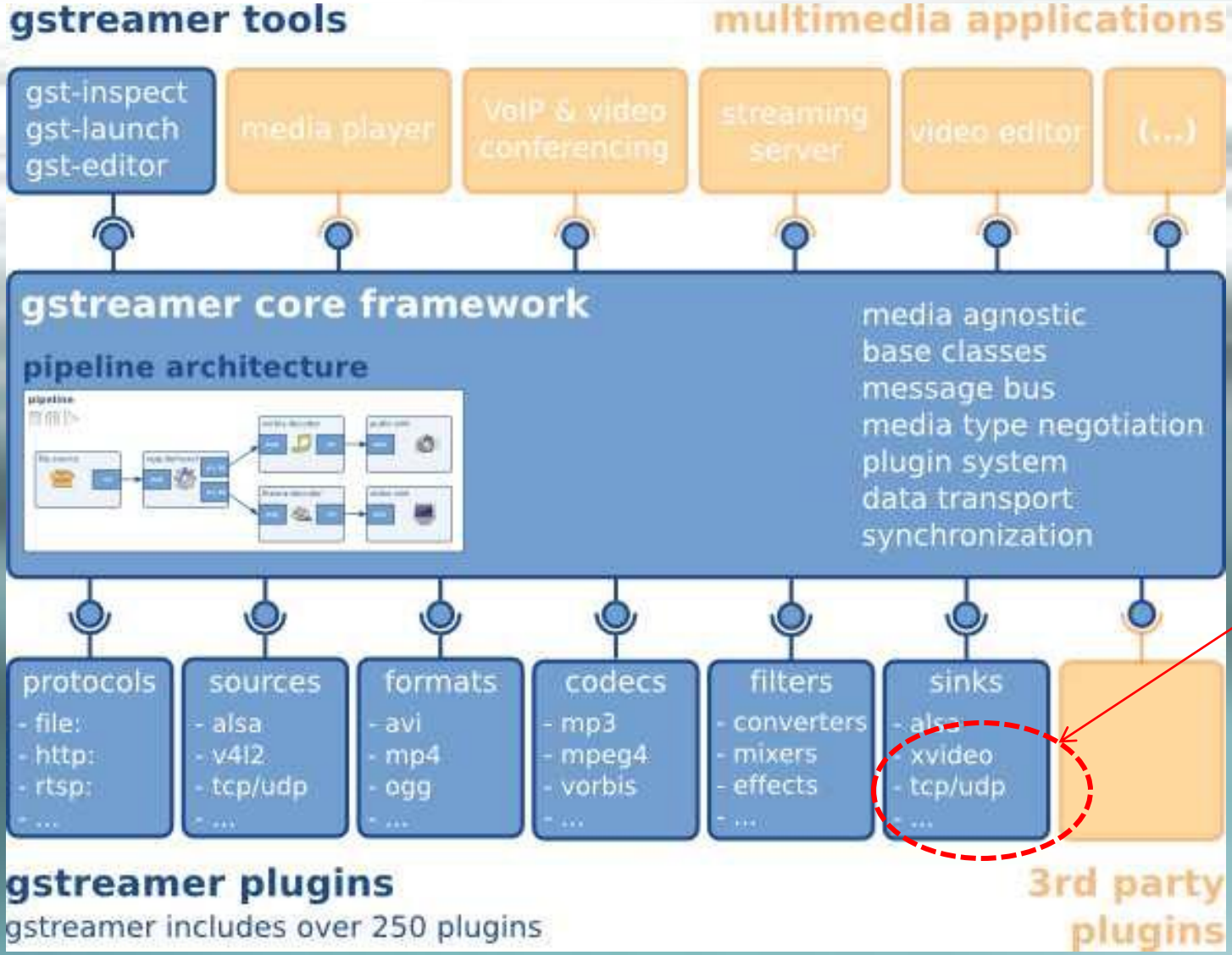
```
au_stream =
  popen(
    “gst-launch fdsrc fd=0 ! Decodebin ! Alsasink”, “w”
  );



  fwrite(data,sizeof(uint8_t),data_len,au_stream);
```

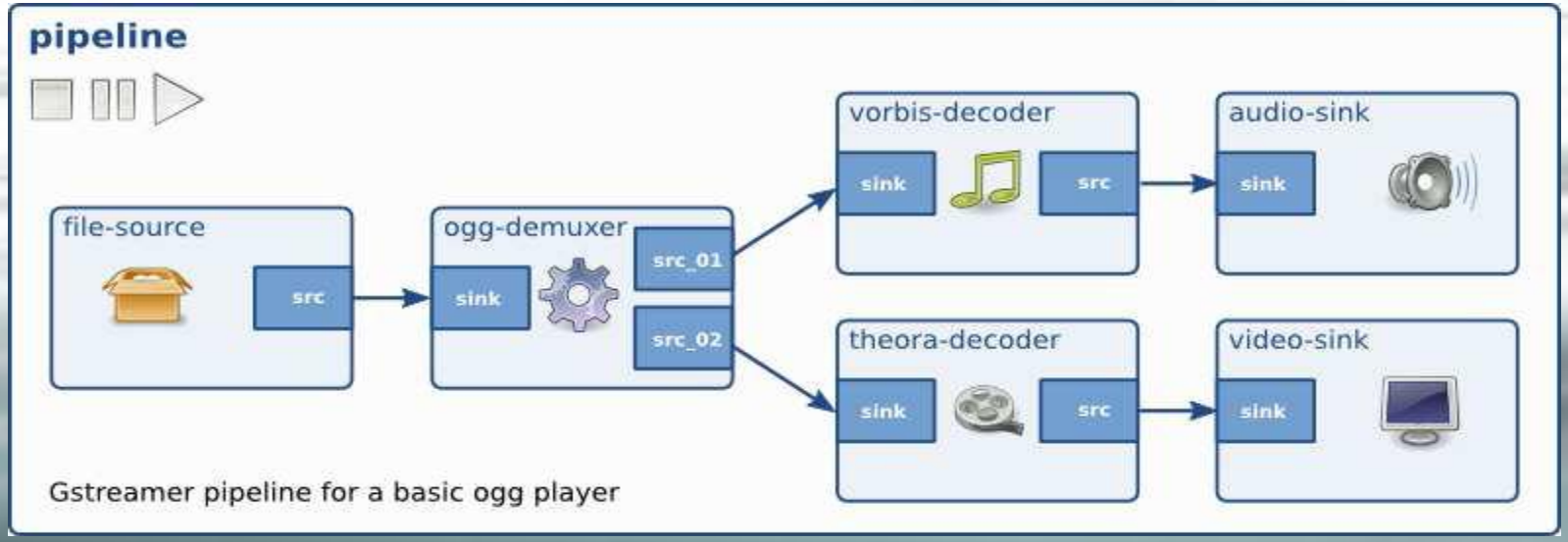
EtherWaves

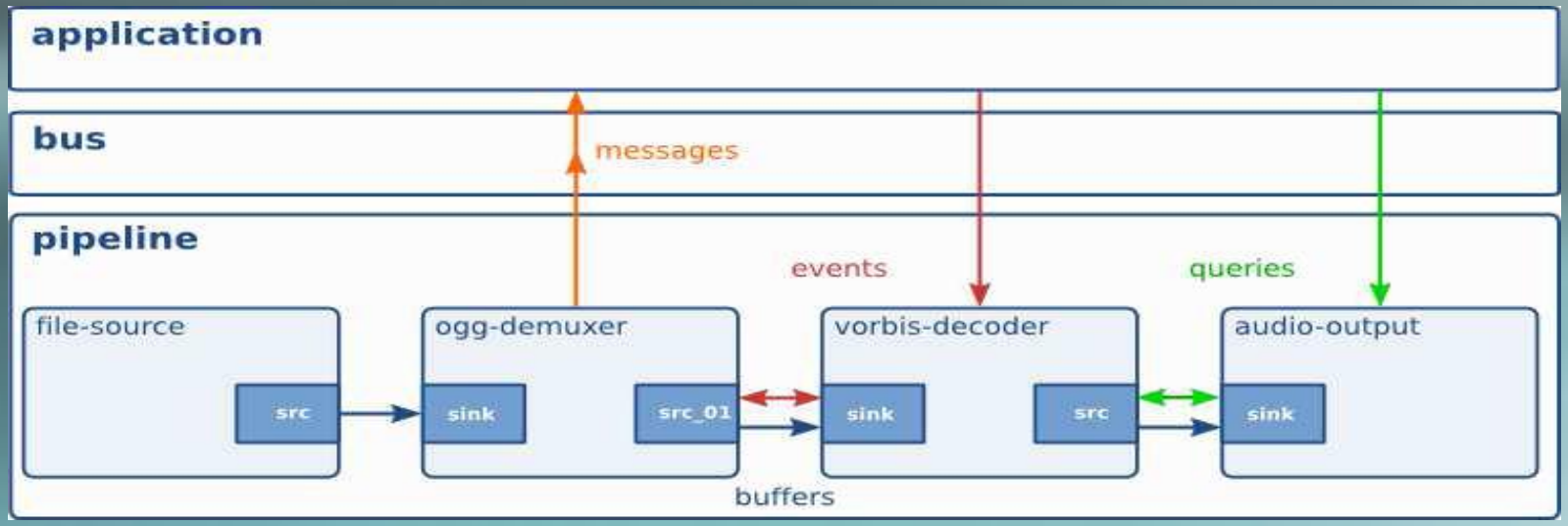
Gstreamer framework



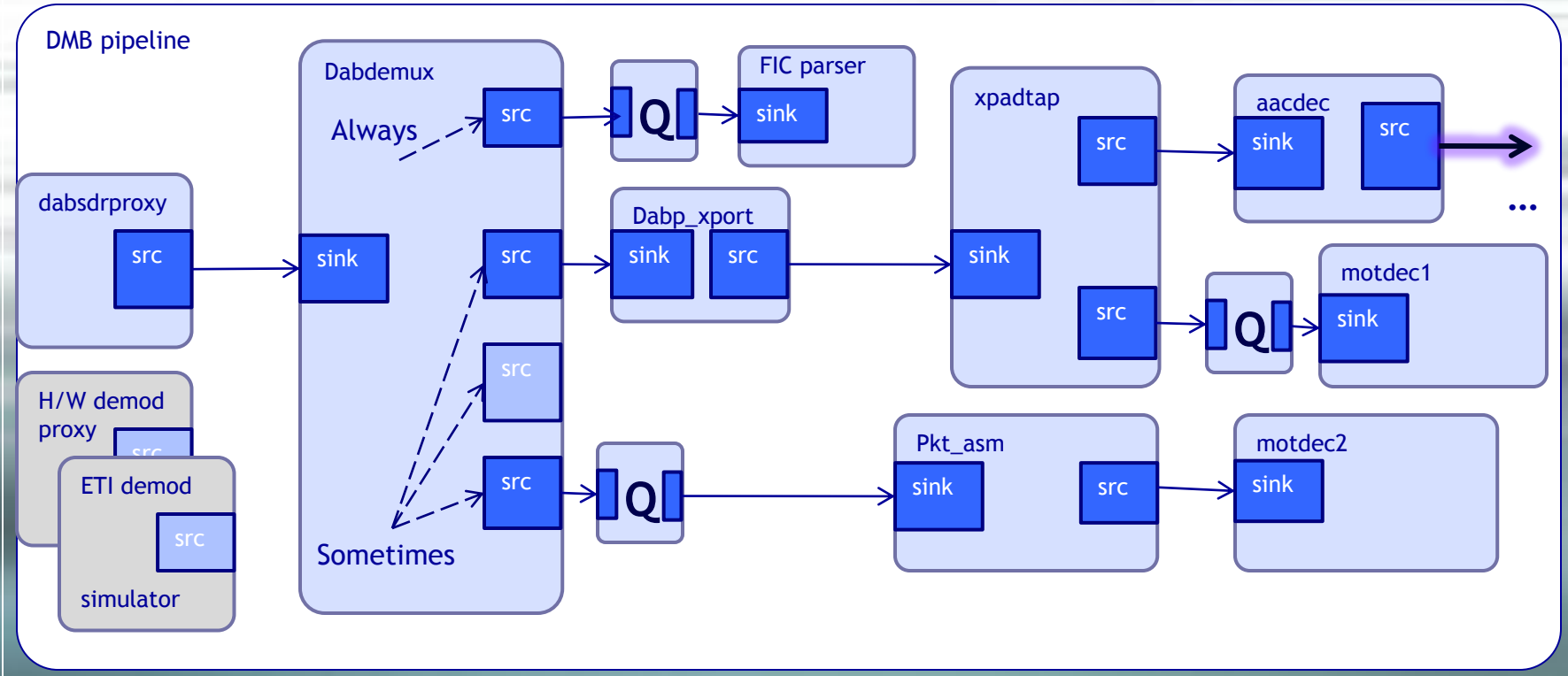
EtherWaves

Gstreamer concepts





DAB+ example pipe



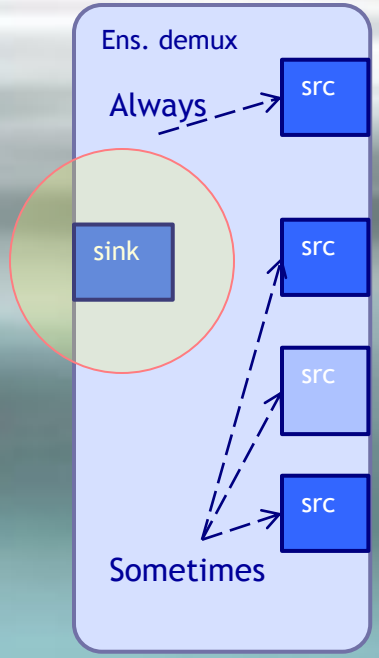

```
gst-launch dabsdrproxy ! Dabdemux name=demux
demux.FIC       ! queue ! FICparser
demux.msc00  ! dabp_xport | xpadtap name=Xpad
demux.msc01  ! queue ! Pkt_asm ! Motdec name=motdec1
Xpad.audioout ! aacdec  ! audioconvert ! audioresample ! osssink
Xpad.dataout   ! queue  ! Motdec name=motdec2
```

DMB stack made of plugins

- The obvious: Hardware agnostic stack
- Match feature set to platform capabilities
- Chain plugins from multiple s/w vendors
  - Plugin code can remain binary and private
  - Interfaces turn open
  - De-facto standard between developers
- Framework for unit testing
  - Example: ETI source replacing H/W demod
  - GOBJECT Introspection and scripting

Key junction ★
DAB demux sink interface

- Sink port of DAB ensemble demux
  - Signal Telemetry
  - Timing
  - Tuners configuration
  - UI ‘cookies’



Can it become standard ?

EtherWaves

Summary

- SDR is possible on modern ARMv7 platforms
- External H/W module can reduce to an array of digital tuners
- Modular DMB stack over Gstreamer framework offers important advantages