Building a Zero-Copy DSP Offload Framework in Linux Using RPMsg

Vishnu Pratap Singh

Paresh Bhagat

About us: TI Processors and Open source

Decades of contribution and collaboration

Ingrained culture to give back to the community

Upstream FIRST!

Focus on long term, sustainable and quality products

Upstream and opensource ecosystem in device architecture

Upstream FIRST mentality!

Overview

- Problem Statement & Goal
- Proposed Solution
- Software Architecture
- Linux-Controlled Multi-channel Audio Offload Example & Demo
- Linux-Controlled Multi-channel Audio Pipeline & Demo
- Performance Benchmarks
- How this is upstream friendly?
- Current Status and Plan Ahead
- Q&A

Disclaimers

- This is a technology presentation, not product-readiness or roadmap commitment
- Opinions presented here are that of the speakers and may not reflect that of Texas Instruments Inc.
- Some Illustrative visuals generated using AI tools; Performance data is based on measured results.

Problem Statement

Performance

General-purpose CPUs are inefficient for multi-channel real-time DSP

Integration

Linux ↔ DSP integration is complex

Inefficient data movement

Multiple memory copies → high latency & CPU overhead

Poor Scalability

Systems don't scale well with channels/workloads

Underutilization

DSPs remain underutilized

Problem is not compute - It is integration into Linux

Goal

Enable DSP usage using standard Linux components only

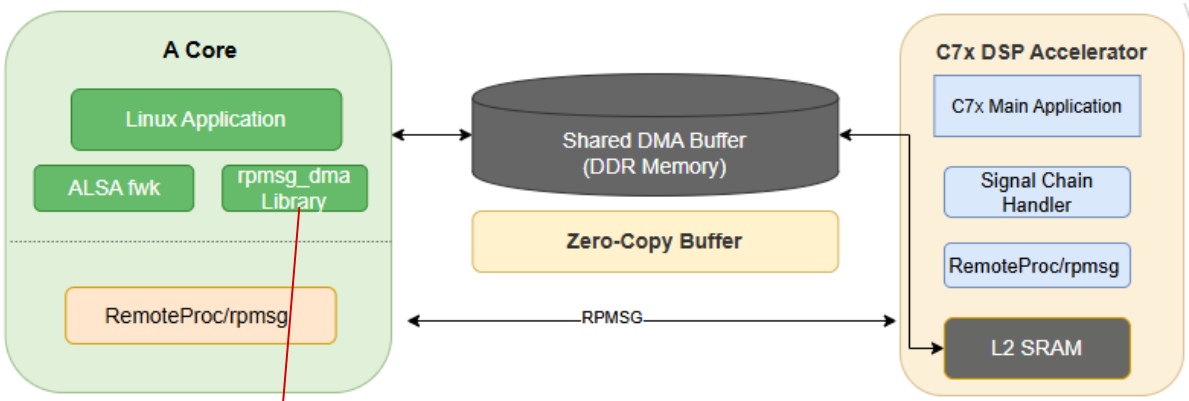
Proposed Solution: Linux Controls, DSP accelerates

Core Idea

- Keep Linux as the system orchestrator
- Use DSP as a compute accelerator
- Separate control plans(Linux) and data plane (shared memory)

Key Advantage

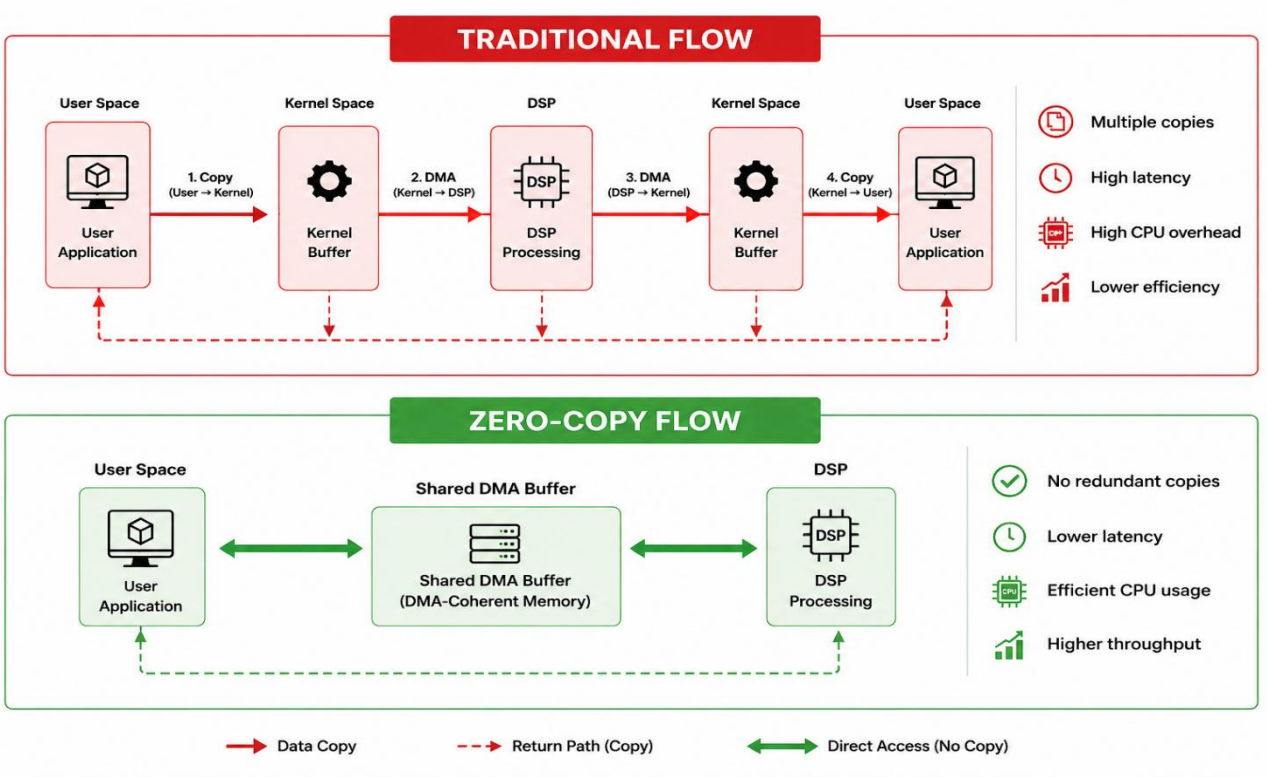
- No need to move application logic from Linux
- Reuse pre-built DSP kernels
- Zero-copy buffer model (no redundant ARM+DSP copies)
- Deterministic, low-latency processing
- Clean, reusable heterogeneous compute pattern



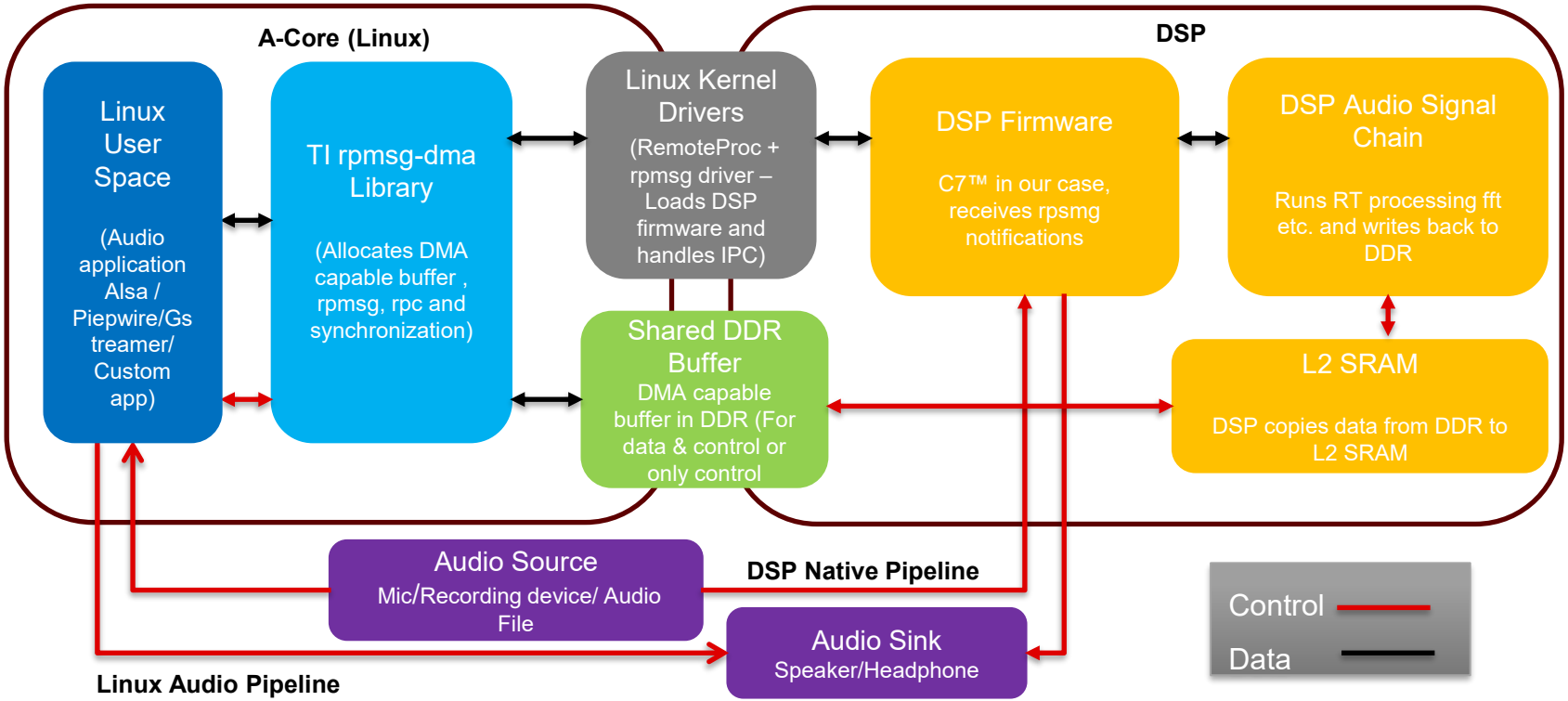
- Manages ARM → DSP communication (RPMsg/RPC)
- Allocates and manages shared DMA buffers (DDR)
- Handles synchronization & low-latency messaging

- Linux writes data → shared DMA buffer (DDR)
- Notifies DSP via RPMsg
- DSP copies data → L2 SRAM (Linux Audio Pipeline only)
- DSP executes signal chain
- Results/KPI written back → shared buffer
- DSP notifies Linux

Zero-Copy as Design Principle



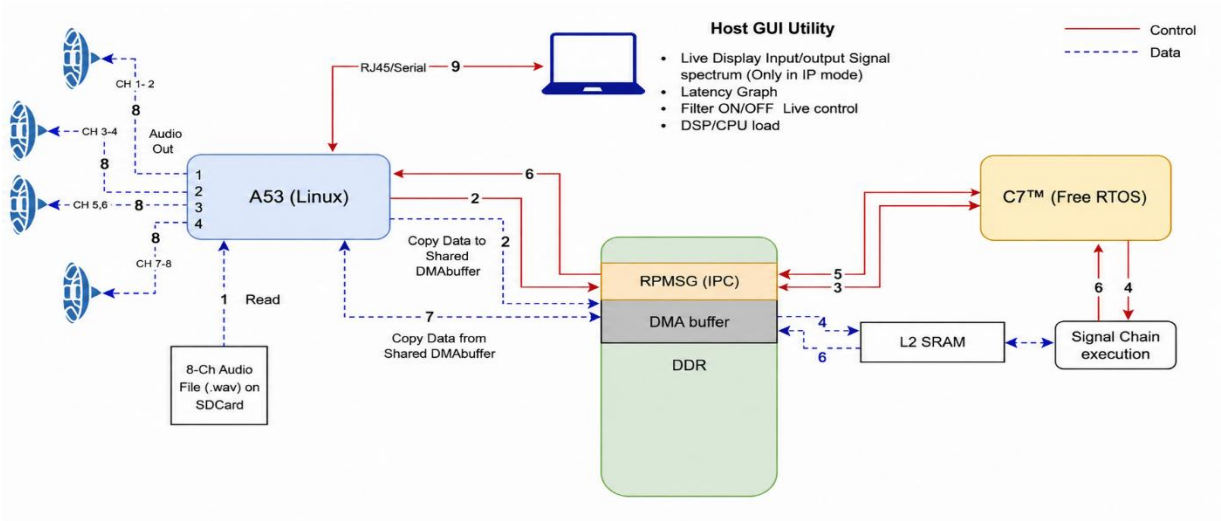
Software Architecture



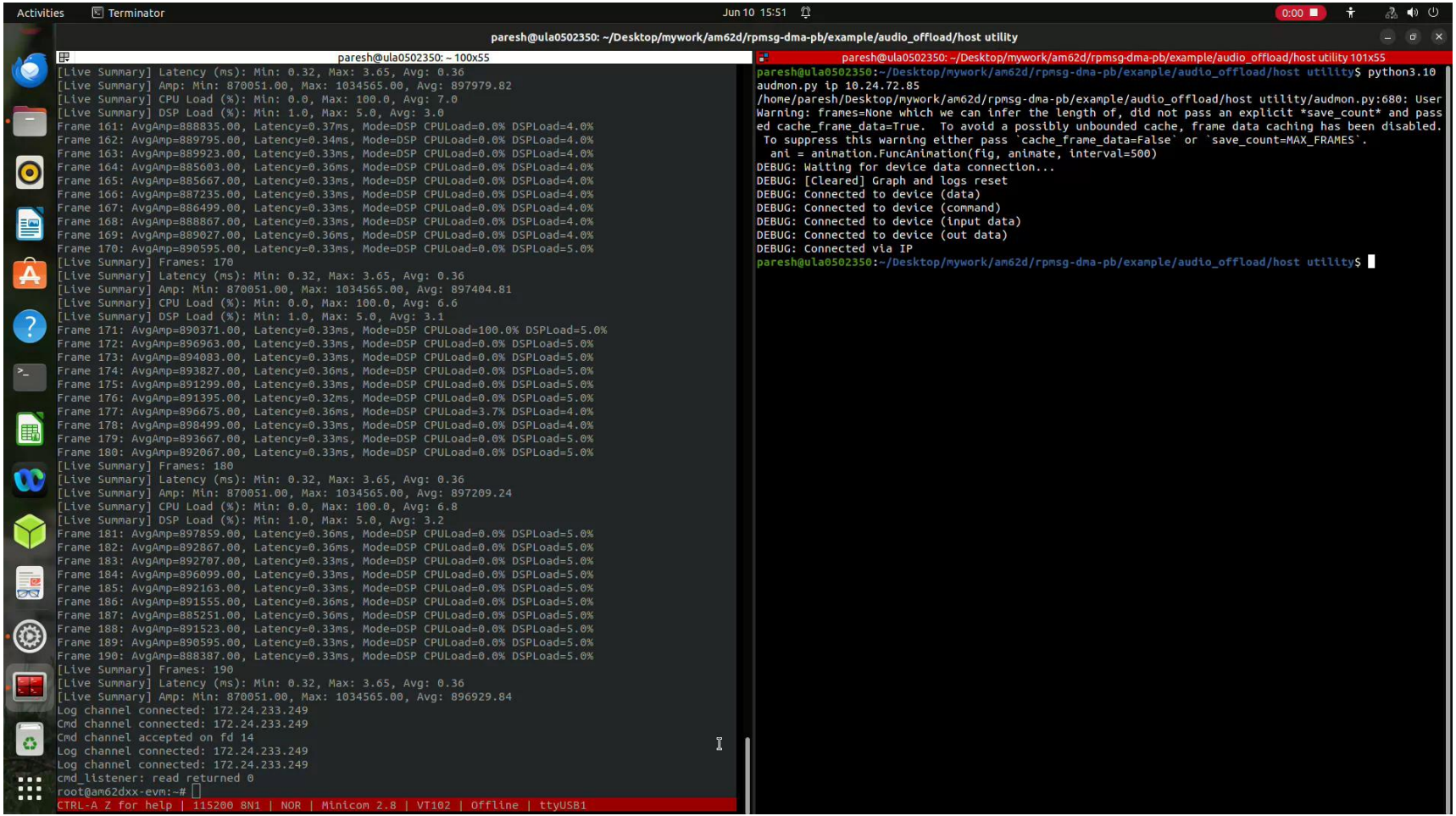
Linux-Controlled Multi-channel Audio Offload Example

What is AM62D?

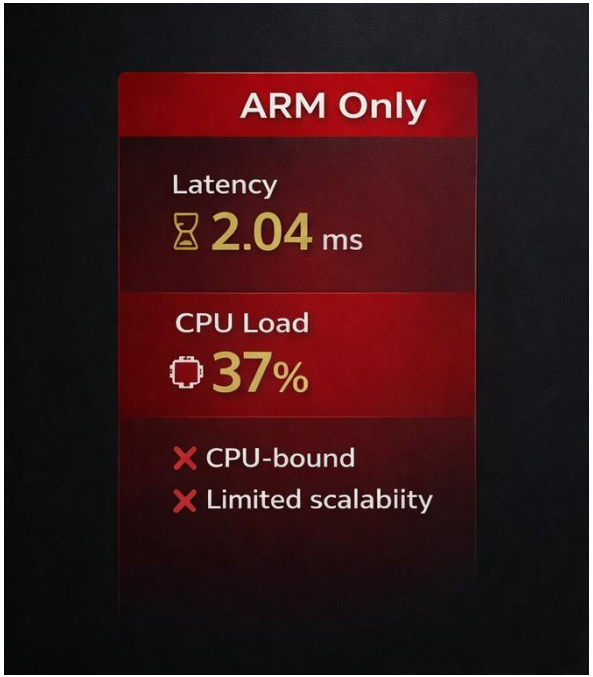
- Sitara AM62D Audio & Signal Processing SoC
- Expandable hardware for premium audio
- Quad Cortex-A53 running Linux
- C7™ vector DSP for signal processing
- R5F cores for real-time control
- McASP interfaces for multichannel audio

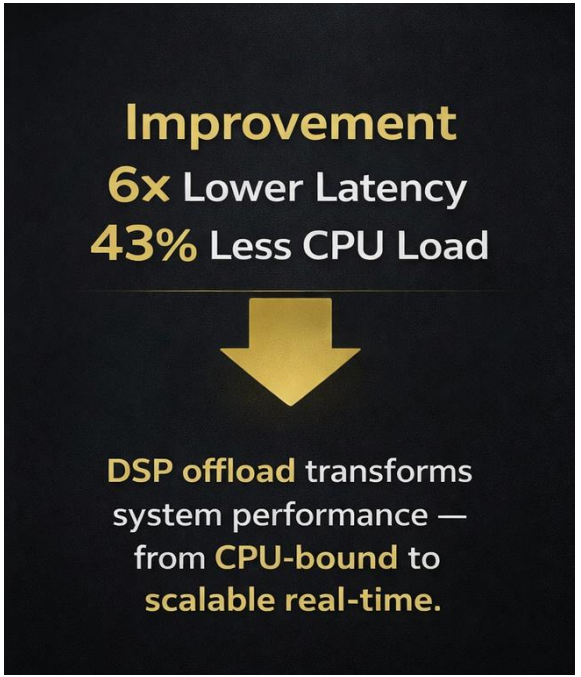


Signal chain execution on DSP


Activities
Terminator
Jun 10 15:51
0:00
paresh@ula0502350: ~/Desktop/mywork/am62d/rpmsg-dma-pb/example/audio_offload/host utility
paresh@ula0502350: ~ 100x55
[Live Summary] Latency (ms): Min: 0.32, Max: 3.65, Avg: 0.36
[Live Summary] Amp: Min: 870051.00, Max: 1034565.00, Avg: 897979.82
[Live Summary] CPU Load (%): Min: 0.0, Max: 100.0, Avg: 7.0
[Live Summary] DSP Load (%): Min: 1.0, Max: 5.0, Avg: 3.0
Frame 161: AvgAmp=888835.00, Latency=0.37ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 162: AvgAmp=889795.00, Latency=0.34ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 163: AvgAmp=889923.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 164: AvgAmp=885603.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 165: AvgAmp=885667.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 166: AvgAmp=887235.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 167: AvgAmp=886499.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 168: AvgAmp=888867.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 169: AvgAmp=889027.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 170: AvgAmp=890595.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
[Live Summary] Frames: 170
[Live Summary] Latency (ms): Min: 0.32, Max: 3.65, Avg: 0.36
[Live Summary] Amp: Min: 870051.00, Max: 1034565.00, Avg: 897404.81
[Live Summary] CPU Load (%): Min: 0.0, Max: 100.0, Avg: 6.6
[Live Summary] DSP Load (%): Min: 1.0, Max: 5.0, Avg: 3.1
Frame 171: AvgAmp=890371.00, Latency=0.33ms, Mode=DSP CPULoad=100.0% DSPLoad=5.0%
Frame 172: AvgAmp=896963.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 173: AvgAmp=894083.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 174: AvgAmp=893827.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 175: AvgAmp=891299.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 176: AvgAmp=891395.00, Latency=0.32ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 177: AvgAmp=896675.00, Latency=0.36ms, Mode=DSP CPULoad=3.7% DSPLoad=4.0%
Frame 178: AvgAmp=898499.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=4.0%
Frame 179: AvgAmp=893667.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 180: AvgAmp=892067.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
[Live Summary] Frames: 180
[Live Summary] Latency (ms): Min: 0.32, Max: 3.65, Avg: 0.36
[Live Summary] Amp: Min: 870051.00, Max: 1034565.00, Avg: 897209.24
[Live Summary] CPU Load (%): Min: 0.0, Max: 100.0, Avg: 6.8
[Live Summary] DSP Load (%): Min: 1.0, Max: 5.0, Avg: 3.2
Frame 181: AvgAmp=897859.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 182: AvgAmp=892867.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 183: AvgAmp=892707.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 184: AvgAmp=896099.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 185: AvgAmp=892163.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 186: AvgAmp=891555.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 187: AvgAmp=885251.00, Latency=0.36ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 188: AvgAmp=891523.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 189: AvgAmp=890595.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
Frame 190: AvgAmp=888387.00, Latency=0.33ms, Mode=DSP CPULoad=0.0% DSPLoad=5.0%
[Live Summary] Frames: 190
[Live Summary] Latency (ms): Min: 0.32, Max: 3.65, Avg: 0.36
[Live Summary] Amp: Min: 870051.00, Max: 1034565.00, Avg: 896929.84
Log channel connected: 172.24.233.249
Cmd channel connected: 172.24.233.249
Cmd channel accepted on fd 14
Log channel connected: 172.24.233.249
Log channel connected: 172.24.233.249
cmd_listener: read returned 0
root@am62dxx-evm:~#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.8 | VT102 | Offline | ttyUSB1
paresh@ula0502350: ~/Desktop/mywork/am62d/rpmsg-dma-pb/example/audio_offload/host utility 101x55
paresh@ula0502350:~/Desktop/mywork/am62d/rpmsg-dma-pb/example/audio_offload/host utility$ python3.10 audmon.py ip 10.24.72.85
/home/paresh/Desktop/mywork/am62d/rpmsg-dma-pb/example/audio_offload/host utility/audmon.py:680: UserWarning: frames=None which we can infer the length of, did not pass an explicit *save_count* and passed cache_frame_data=True. To avoid a possibly unbounded cache, frame data caching has been disabled. To suppress this warning either pass `cache_frame_data=False` or `save_count=MAX_FRAMES`.
ani = animation.FuncAnimation(fig, animate, interval=500)
DEBUG: Waiting for device data connection...
DEBUG: [Cleared] Graph and logs reset
DEBUG: Connected to device (data)
DEBUG: Connected to device (command)
DEBUG: Connected to device (input data)
DEBUG: Connected to device (out data)
DEBUG: Connected via IP
paresh@ula0502350:~/Desktop/mywork/am62d/rpmsg-dma-pb/example/audio_offload/host utility$

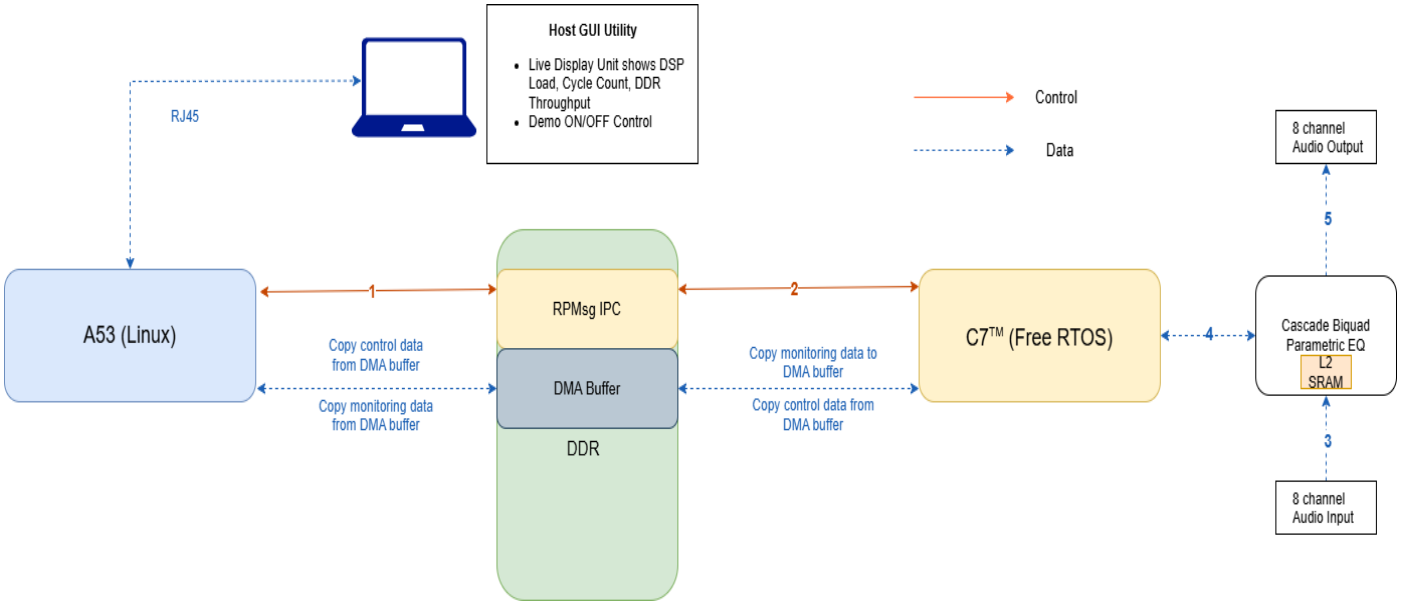
ARM vs C7x + ARM: Real-Time Performance Comparison



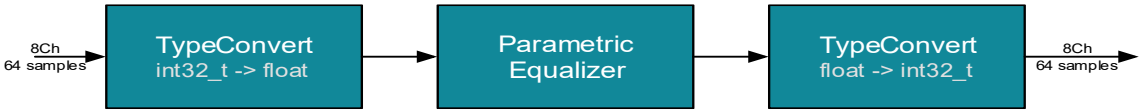


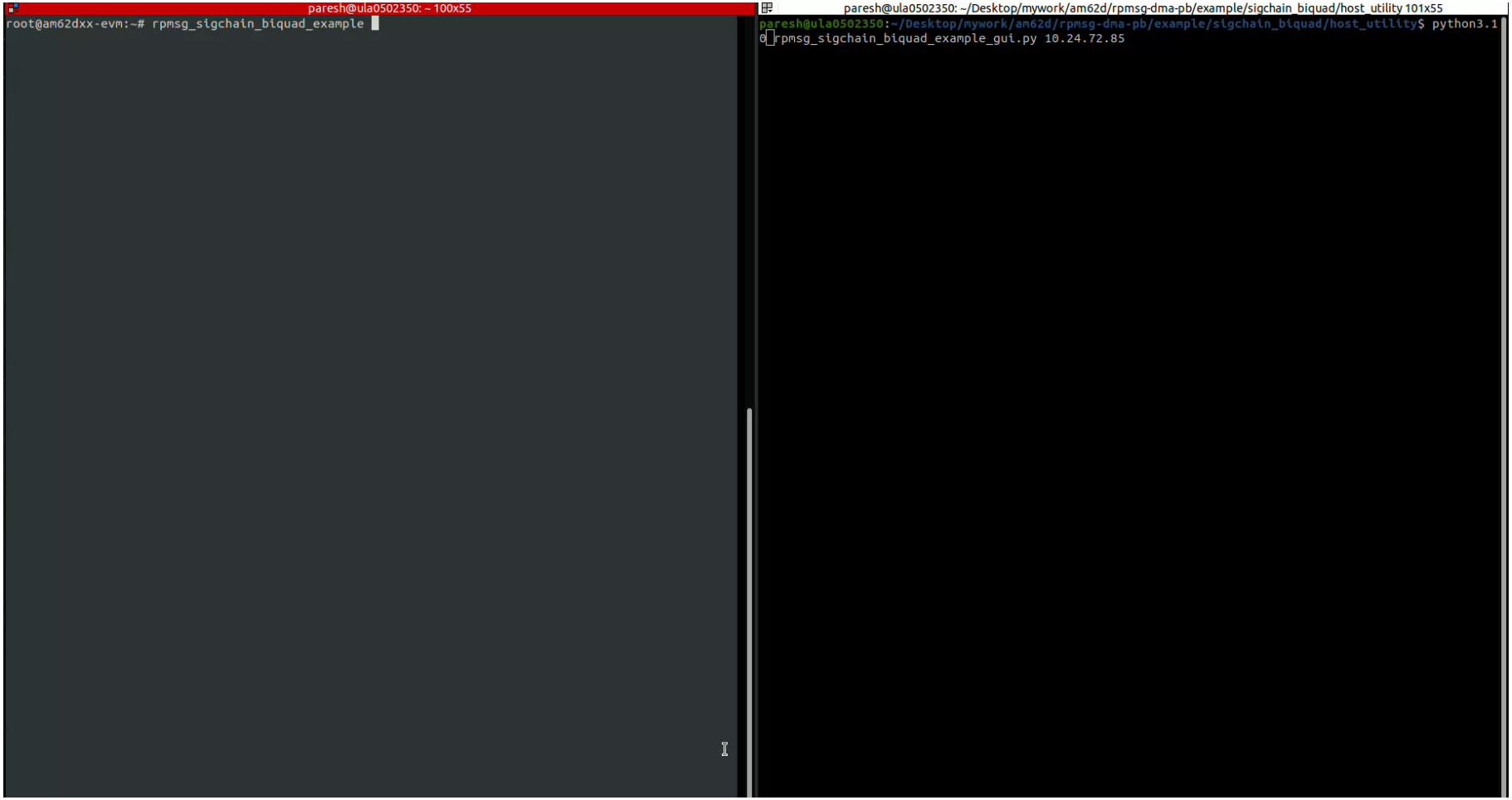
Signal chains are not strictly identical; results reflect practical system-level improvements with DSP offload.
Illustrative visuals generated using AI tools; performance data is based on measured results.

Linux-Controlled Multichannel DSP Audio Pipeline Example



Signal chain execution on DSP




paresh@ula0502350: ~ 100x55
root@am62dxx-evm:~# rpmsg_sigchain_biquad_example
paresh@ula0502350: ~/Desktop/mywork/am62d/rpmsg-dma-pb/example/sigchain_biquad/host_utility 101x55
paresh@ula0502350:~/Desktop/mywork/am62d/rpmsg-dma-pb/example/sigchain_biquad/host_utility$ python3.1
0 rpmsg_sigchain_biquad_example_gui.py 10.24.72.85

How this is upstream friendly?

No Kernel Patches

No Changes to Kernel Core, Standard APIs, Portable across SoCs.

No Proprietary Frameworks

Built on Remote Proc, ALSA, RPMsg etc.

No tight DSP coupling

DSP can run baremetal, freeRTOS etc.

Enables community reuse and evolution

Current Status and Plan Ahead

Current Status

- ❑ Working Linux↔DSP offload solution with working examples
- ❑ Real-time Streaming from Linux as well as DSP
- ❑ Zero copy buffer handling – Shared DDR buffer
- ❑ Stable RPMsg-based communication
- ❑ ALSA integration

Plan Ahead

- ❑ Integrate with PipeWire & GStreamer
- ❑ Extend to DSP + AI hybrid pipelines
- ❑ Enhance scalability (higher channels, bandwidth)
- ❑ Multi-buffering, scheduling & pipelining.
- ❑ Add Profiling, tracing, & latency measurement tools

References

- AUDIO-AM62D-EVM
- Processor SDK Linux Doc - AM62D
- Processor SDK Linux AM62D Demos
- Source Code : TI rpmsg-dma
- Source Code : MCU + SDK
- Source Code : Audio_filter_offload
- Source Code: fft2d_linux_dsp_offload
- Source Code: Sigchain_biquad_cascade
- MCU + SDK Doc : Audio_offload
- MCU + SDK Doc : FFT2D_LINUX_DSP_OFFLOAD
- MCU + SDK Doc : Cascade_Biquad_Parametric_EQ

Credits and Acknowledgement

- Barath Ramesh (TI C7™ DSP Team)
- Anoop Krishnan (MCU+SDK Team)
- Texas Instruments Inc.
- The Linux Foundation

Q&A

- Contact Information:
  - Vishnu Pratap Singh v-singh1@ti.com
  - Paresh Bhagat p-bhagat@ti.com
- Also, on IRC @ libera.chat #linux-ti

Learn more about TI products

- https://www.ti.com/linux
- https://www.ti.com/processors
- https://www.ti.com/edgeai

Why choose TI MCUs and processors?

Scalability

Our products offer scalable performance that can adapt and grow as the needs of your customers evolve.

Efficiency

We design products that extend battery life, maximize performance for every watt expended, and unlock the highest levels of system efficiency.

Affordability

We strive to make innovation accessible to all by creating cost-effective products that feature state-of-the-art technology and package designs.

Availability

Our investment in internal manufacturing capacity provides greater assurance of supply, supporting your growth for decades to come.