



# Building Accelerated Gstreamer Applications for Video and Audio AI

Abdo Babukr






Accelerated Computing Consultant

Wave Spectrum, Inc.



**WAVE SPECTRUM**  
EMBEDDED COMPUTING

# Agenda

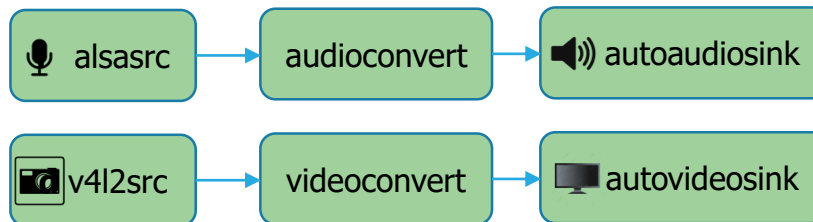
-  What is Gstreamer?
-  Why use Gstreamer?
-  How to use Gstreamer?
-  What are the AI extensions to Gstreamer?
-  Example AI pipeline

# What Is Gstreamer?

- Gstreamer is an open-source multimedia framework used for developing video and audio streaming applications.
- It is a highly modularized framework where pipelines are constructed in a plug-and-play fashion using building blocks called plugins.
- Examples of streaming applications include:
  - Network streaming (rtsp, http, rtp, etc.).
  - Video and audio transcoding (encoders, decoders, parsers, etc.).
  - Video and audio processing (audio/video filters and converters).

# What Is Gstreamer? (cont.)

- There are three main types of plugins:
  - Source plugins – provide input to the pipeline.
    - Cams/mics, network streams, multimedia files, app sources, etc.
  - Transform plugins – convert input buffer to output buffer.
    - Converters, muxers/demuxers, codecs, overlay operations, etc.
  - Sink plugins – are the outputs of the pipeline.
    - Displays/speakers, networks streams, multimedia files, app sinks, etc.

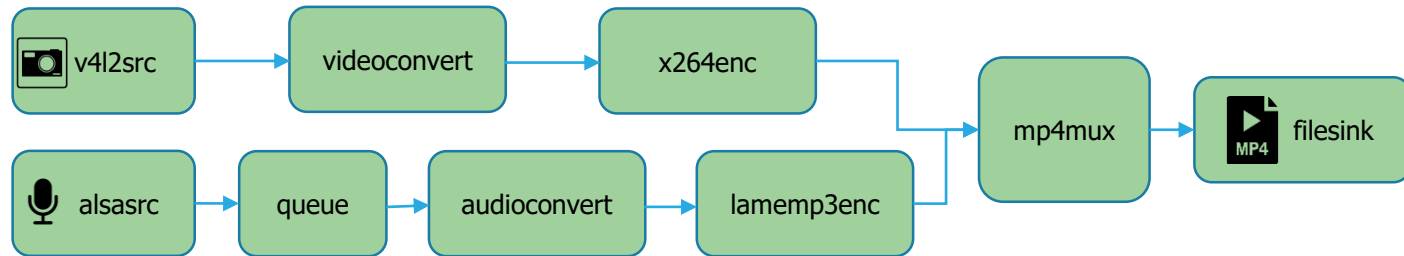


# Why Use Gstreamer?

- Gstreamer saves a considerable amount of development time.
  - It includes hundreds of plugins implementing functions a developer would not need to redevelop.
    - Codecs, converters, parsers, network protocols, etc.
  - It uses a lightweight mechanism for passing buffers between plugins, making it efficient for use in embedded systems.
  - It supports hardware buffers and plugins which use coprocessors to accelerate signal and image processing tasks including codecs, filtering and inferencing.
  - It is extremely flexible and can encompass either a small portion of the application or the entire application.

# Why Use Gstreamer? (command line example)

- Suppose you want to create a recorder; you can construct this pipeline:



- The queue in between the audio source plugin and the audioconvert plugin creates CPU thread separate from the video branch.
- The pipeline containing the nine plugins is constructed in a command line this way:

```
gst-launch-1.0 -e v4l2src ! videoconvert ! x264enc ! mp4mux name=m \  
alsasrc ! queue ! audioconvert ! lamemp3enc ! m. ! \  
filesink location=output.mp4
```

# Why Use Gstreamer? (python example)

```
import gi
gi.require_version('Gst', '1.0')
from gi.repository import Gst, GObject

# Initialize GStreamer
Gst.init(None)

# Create the pipeline
pipeline = Gst.Pipeline()

# Create the elements
src = Gst.ElementFactory.make("v4l2src", None)
..

# Set properties
src.set_property("device", "/dev/video0")
sink.set_property("location", "output.mp4")
```

```
# Add the elements to the pipeline
pipeline.add(src)
..

# Link the elements
src.link(vidconvert)
..

# Start playing the pipeline
pipeline.set_state(Gst.State.PLAYING)

# Wait until error or EOS
bus = pipeline.get_bus()
msg =
bus.timed_pop_filtered(Gst.CLOCK_TIME_NONE,
Gst.MessageType.ERROR | Gst.MessageType.EOS)

# Free resources
pipeline.set_state(Gst.State.NULL)
```

# Why Use GStreamer? (C++ example)

```

#include <gst/gst.h>

int main(int argc, char *argv[]) {
    GstElement *pipeline, *src ..;
    GstBus *bus;
    GstMessage *msg;
    GstStateChangeReturn ret;

    // Initialize GStreamer
    gst_init(&argc, &argv);

    // Create the pipeline
    pipeline = gst_pipeline_new("mypipeline");

    // Create the elements
    src = gst_element_factory_make("v4l2src", "src");
    ..
    // Set properties
    g_object_set(src, "device", "/dev/video0", NULL);
    g_object_set(sink, "location", "output.mp4", NULL);

    // Add the elements to the pipeline
    gst_bin_add_many(GST_BIN(pipeline), src, .., NULL);

    // Link the elements
    gst_element_link_many(src, .., NULL);
    gst_element_link_many(alsasrc, .., NULL);

    // Start playing the pipeline
    ret = gst_element_set_state(pipeline, GST_STATE_PLAYING);

    // Check for failure
    ..

    // Wait until error or EOS
    ..

    // Free resources
    ..

    return 0;
}

```



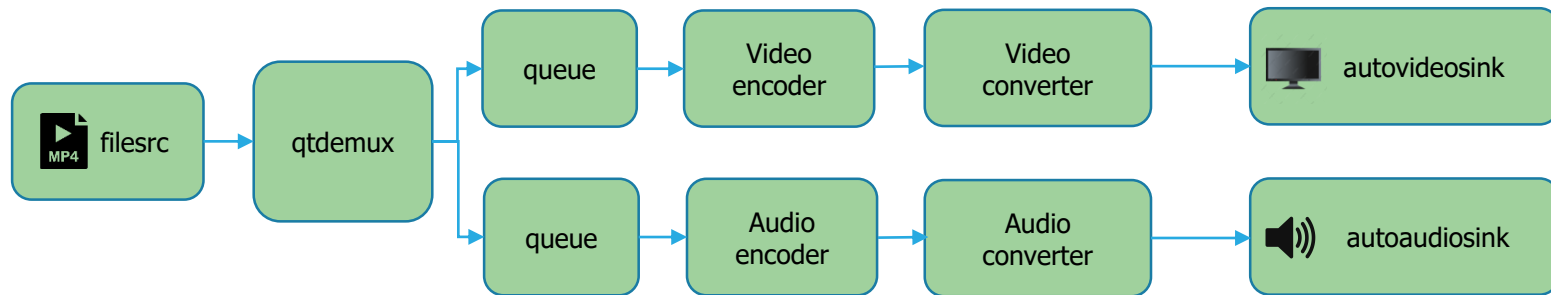


# How to Use Gstreamer?

- Gstreamer can be installed on popular operating systems:
  - iOS, MacOS, Android, Windows, and Linux.
- On Debian-based embedded or desktop Linux it is installed with:
 

```
apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev \
  gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad \
  gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-tools gstreamer1.0-x \
  gstreamer1.0-alsa gstreamer1.0-gi gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio
```
- For desktop hardware accelerated plugins you will need either an integrated or discrete GPU and the Video Acceleration API.
- For embedded hardware accelerated plugins you will need a DSP, a GPU or an ISP; these accelerators support Open Media Acceleration (OpenMAX).

# How to Use Gstreamer? (Accelerating)



```

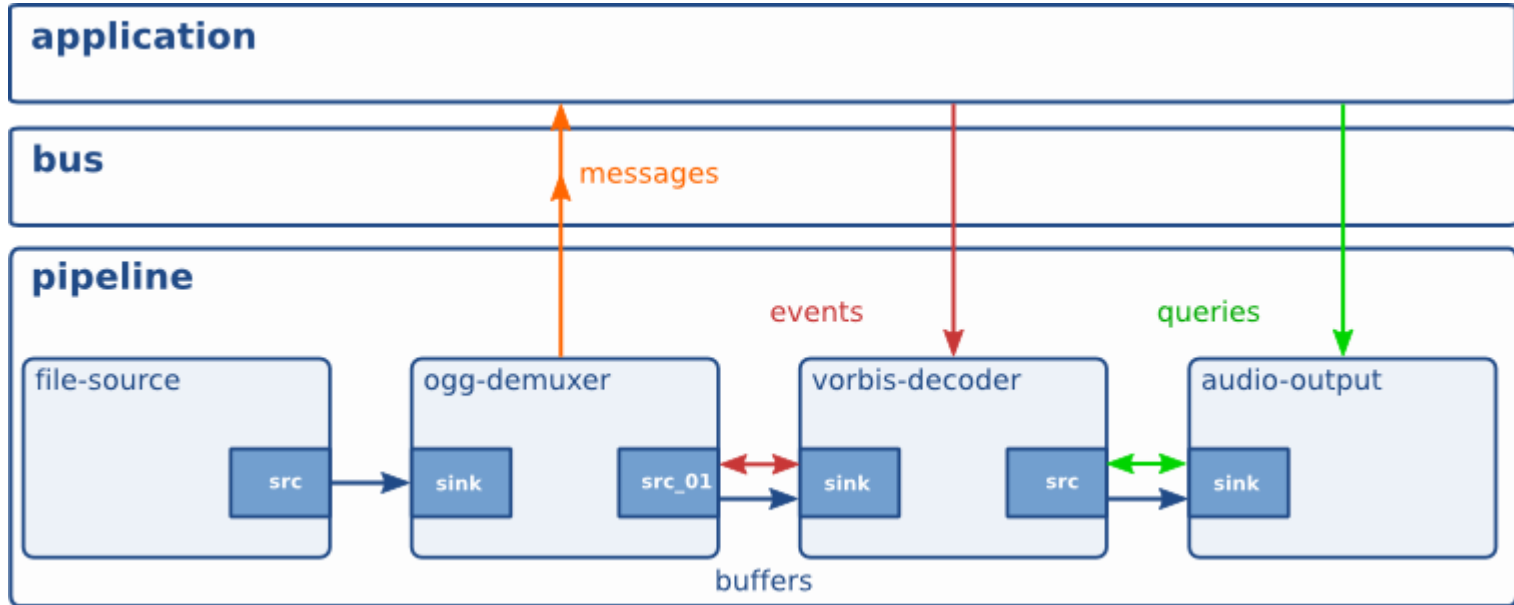
gst-launch-1.0 filesrc location=output.mp4 ! qtdemux name=demux \
  demux.video_0 ! queue ! avdec_h264 ! videoconvert ! autovideosink \
  demux.audio_0 ! queue2 ! avdec_acc ! audioconvert ! autoaudiosink
  
```

`avdec_h264` is a software plugin performing the video encoding, for hardware acceleration it can be replaced with:

*vaapicodec*, *omxh264dec* or *nvv4l2decoder*.

Likewise, *videoconvert* is a software plugin performing video conversion, for hardware acceleration it can be replaced, for example, by *nvvideoconvert*.

# How to Use Gstreamer? (Internals)



- **NNstreamer-** A set of plugins used to run neural network models and managing neural network pipelines easily and efficiently.
- **Deepstream-** Nvidia-based streaming analytics toolkit based on Gstreamer. Includes plugins for Video and Audio AI and IoT.
- **Vitis Video SDK-** Complete Software stack to build AI-powered intelligent video analytics solutions on AMD platforms.

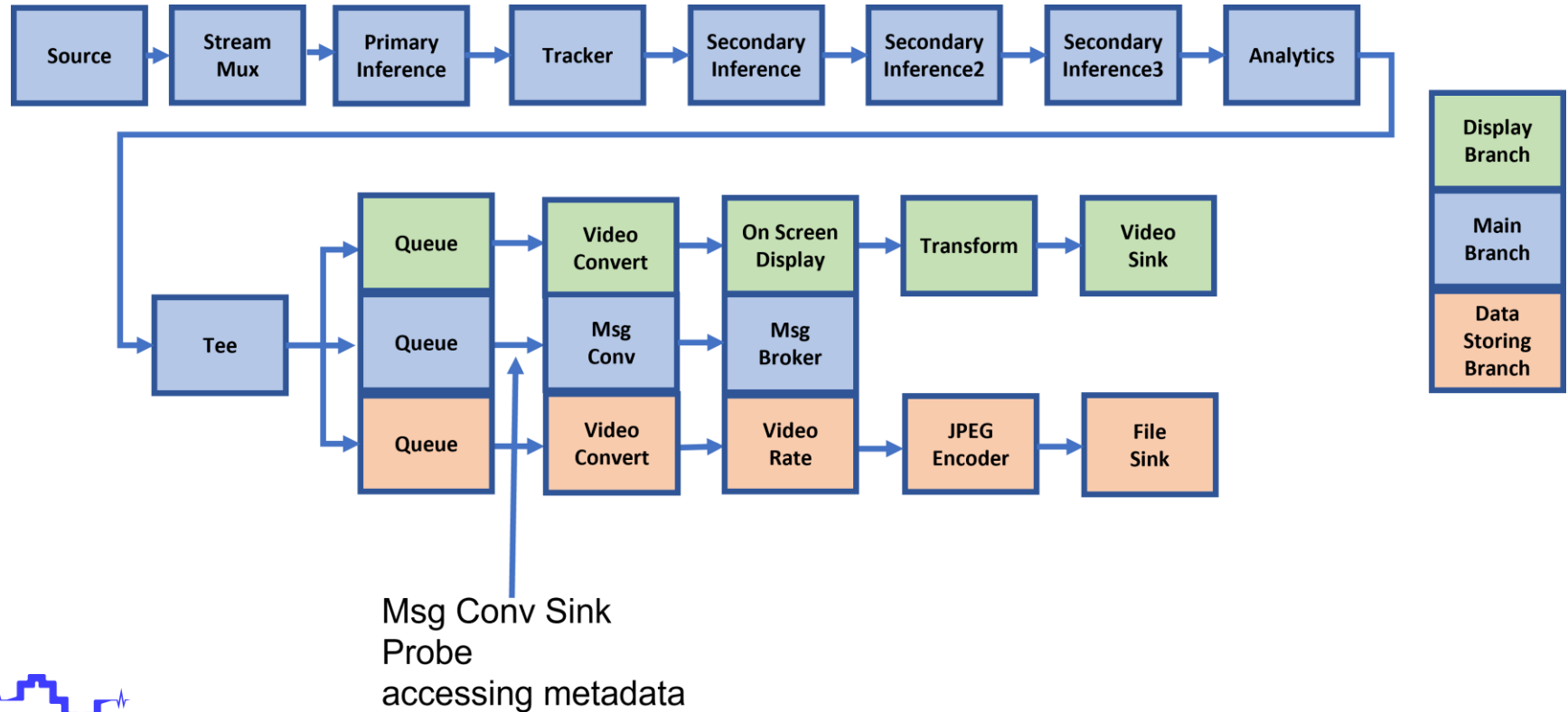
# Gstreamer AI Extensions

- Any model type can be supported.
- Additional plugins are available for specific model types including:
  - Classification
  - Detection
  - Segmentation

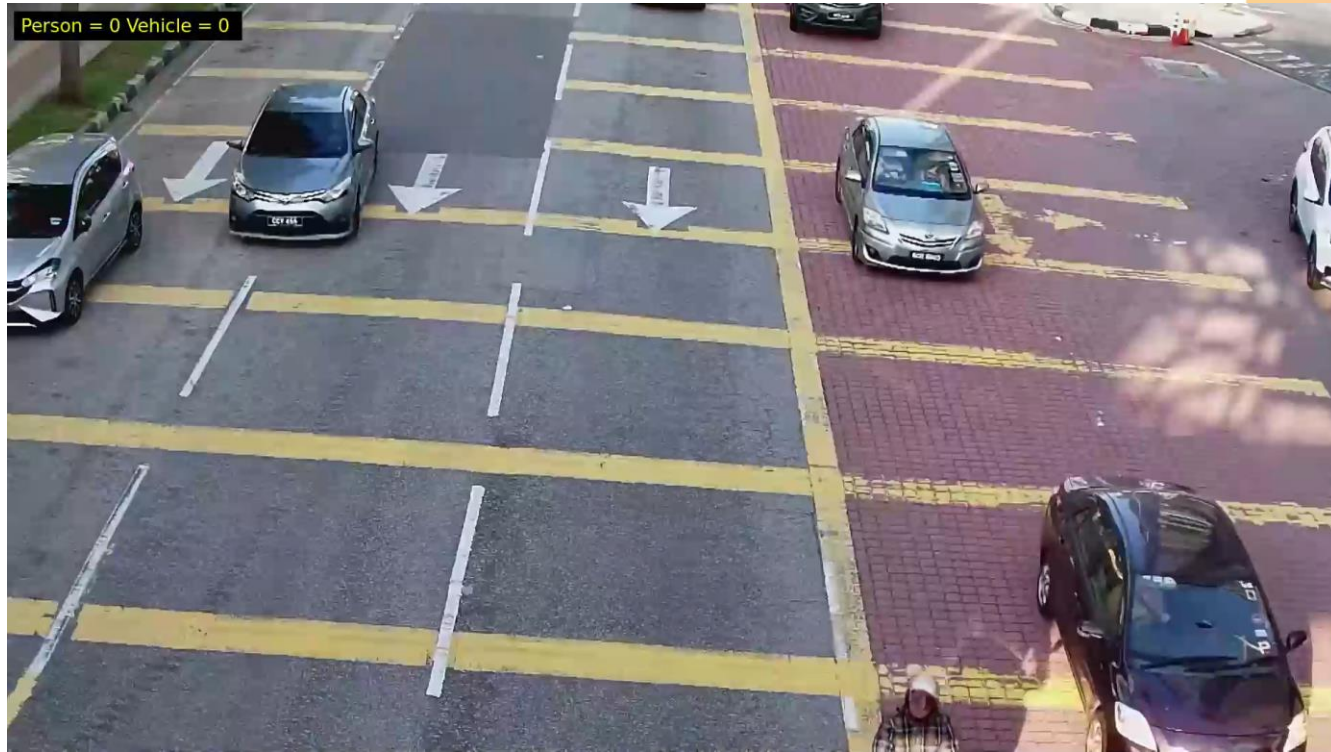
# Example AI Pipeline

- We construct a pipeline which counts vehicles passing by, gathers live analytics and sends the data to the cloud.
- Vehicles are detected, and classified by type, make and color.
- A data storage branch is included to collect more training images for continuous AI model improvements.
- A display branch is included for viewing the display in real-time.

# Example AI Pipeline



# Example AI Pipeline



<https://www.youtube.com/watch?v=Q8pqb8KVyzE>



# Conclusion

- Gstreamer and Gstreamer-based AI SDKs can be used to rapidly construct Video and AI processing pipelines.
- Gstreamer supports the use of coprocessors to accelerate compute intensive tasks within the pipeline.
- Gstreamer can consist of either a small portion or most of the application.

## Gstreamer and Accelerated Gstreamer

Accelerated Gstreamer user guide

[https://developer.download.nvidia.com/embedded/L4T/r32\\_Release\\_v1.0/Docs/Accelerated\\_GStreamer\\_User\\_Guide.pdf](https://developer.download.nvidia.com/embedded/L4T/r32_Release_v1.0/Docs/Accelerated_GStreamer_User_Guide.pdf)

Hardware accelerated video processing on Intel graphics

<https://github.com/GStreamer/gstreamer-vaapi>

Open-source multimedia framework

<https://gstreamer.freedesktop.org/>



## AI Extensions to Gstreamer

NNStreamer

<https://nnstreamer.ai/>

Nvidia Deepstream

<https://developer.nvidia.com/deepstream-sdk>

Vitis video SDK migration

<https://www.xilinx.com/developer/articles/vvas-migration-for-deepstream-users.html>