Application Note

Multimedia Applications on AM62A

Suren Porwar, Tarkesh Pande and Devarsh Thakkar

ABSTRACT

In this application note, various benchmarks are provided for the multimedia performance when using the AM62A processor [1]. Multimedia in this context refers to the ability to encode and decode video streams as well as encode still images. The hardware accelerators in AM62A are capable of H.264/265 encode/decode and MJPEG encoding. Example use cases are described along with a systems analysis of typical use case loadings.

Table of Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

In today's rapidly, evolving digital landscape there is a convergence of multiple technologies at the edge for Internet of Things Applications. Camera and Video based applications can now perform intelligent edge processing such as detecting people, objects of interests, or events with significantly reduced compute and power. This processed data can then be seamlessly streamed to a central point for example in surveillance applications like video door bells, drones, and security cameras. For these types of applications, it is important to have state of the art encode/decode capability in terms of performance along with the ability to support multiple configurations like multi-stream and multi-resolution encode/decode. This application note shows how the AM62A processor is targeted to solve these use cases. Its H.264 and H.265 encoding and decoding capabilities enables seamless compression and decompression of high-definition video content, ensuring efficient data handling for applications like live streaming and video surveillance. All of the above needs to be done at a low power foot print to maximize battery life. This application note primarily focuses on the multimedia capability of the AM62A device [2]. A brief introduction is provided to AM62A and describes the software framework for accessing the multimedia accelerators. Next, typical benchmarks that system designers care about like latency and DDR bandwidth consumption are provided. Finally, two system examples use cases show what a system flow looks like along with the resource loadings. These example applications have been tested on the AM62A starter kit [3].

2 AM62A Processor

The AM62A Edge AI Microprocessor is shown in Figure 2-1. AM62A Simplified Block Diagram, is a heterogeneous processor designed for analytics applications. There are different hardware accelerators optimized for different tasks thus enabling an optimized power and cost footprint for different vision and multimedia applications

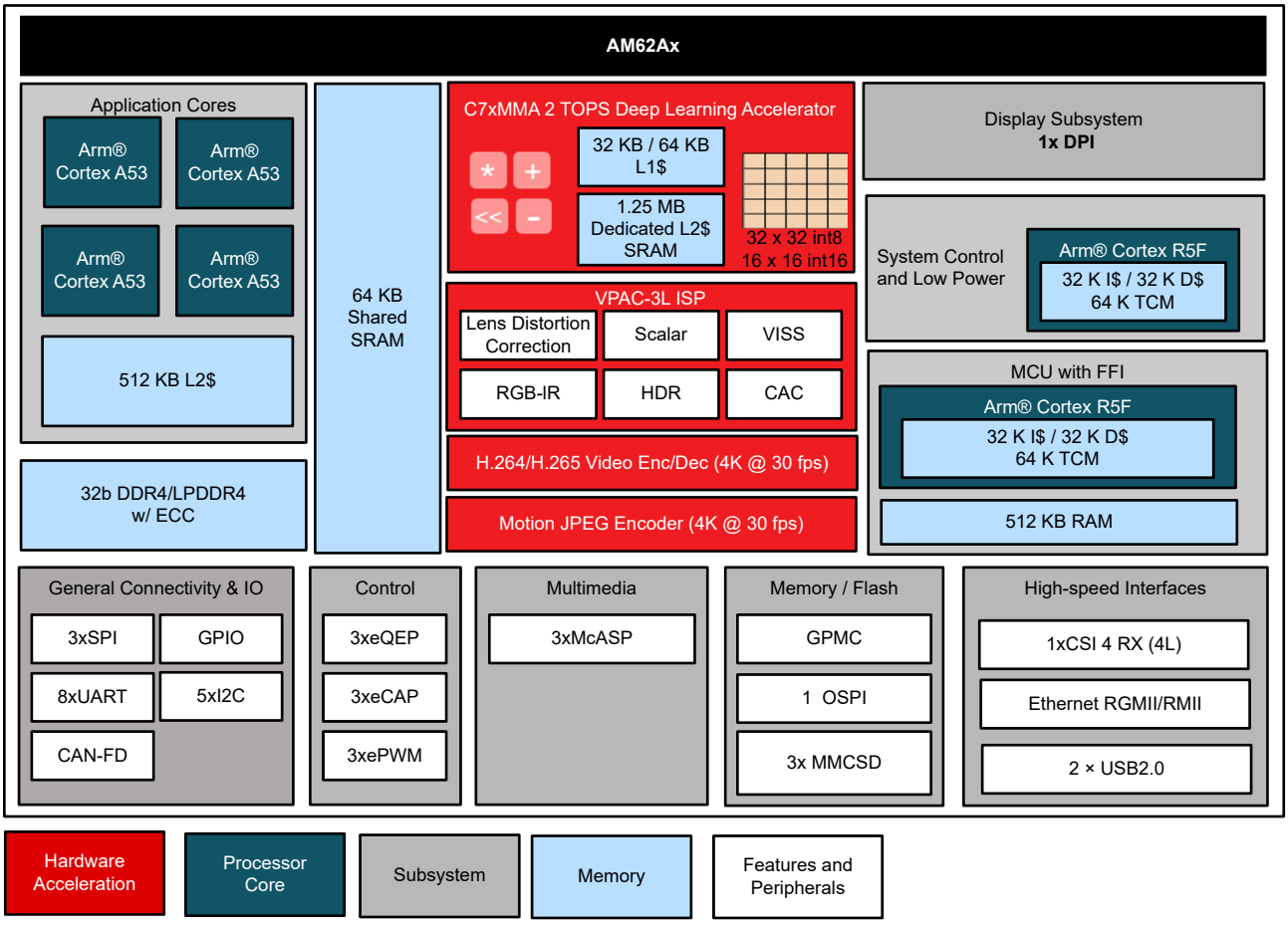

Figure 2-1. AM62A Simplified Block Diagram

The main processing and compute subsystems from a multimedia context in AM62A are as follows:

- Quad-A53 ARM cores: These can run up to 1.4 GHz and provide up to 16.8k Dhrystone Million Instructions Per Second (DMIPS) of performance.
- C7 Digital Signal Processor (DSP) and Matrix Multiplication Accelerator (MMA): TI's deep learning accelerator on AM62A is capable of two TOPs operations when clocked at 1 GHz.
- H.264/265 Encoder and Decoder: This has a total encode/decode measured capability of 240MP/s and can support up to 32 channels concurrently.
- MJPEG Encoder: This has a measured encode capability of 416MP/s and can easily support 4K30 applications.
- Vision Processing Accelerator (VPAC3L): The latest generation in TI ISP technology for performing image operations some examples of which are color conversions, chromatic aberration correction, pyramid scaling and lens distortion correction. It has a total throughput of up to 300MP/s with support for RGBIR cameras.

3 AM62A Video Processing Unit (VPU) Capability

The codec hardware IP block in AM62A supports both H.264 and H2.65 encoding/decoding capability. Table 3-1 illustrates some of the high-level specifications of the encoder/decoder block.

Table 3-1. Encoder/Decoder Specifications

Maximum Resolution	**8192x 8192**
Minimum Resolution	256x128
Clock Rate	400 MHz
Constraints	Picture width and height are both multiples of 8
Concurrent Encode/Decode Streams	Dependent on Resolution and Frame Rate 4 Encode + 4 Decode Streams of 1920x1080 @ 30fps
H.264 Encoding	H.264 Baseline/Constrained Baseline/ Main/High Profiles Level @ L5.2
H.265 Encoding	H.265 Main and Main Still Picture Profile @L5.1 High Tier
H.264 Decoding	H.264 Baseline/Constrained Baseline/ Main/High Profiles Level @ L5.2
H.265 Decoding	H.265 Main and Main Still Picture Profile @L5.1 High Tier

4 AM62A JPEG Encode Capability

The JPEG hardware IP block in AM62A is a stateful and scalable performance still image encoder. It is capable of real time encoding of YUV420/YUV422 raw picture data to fully compressed image (jpeg) or a sequence of compressed images (MJPEG).

Table 4-1 illustrates some of the high-level specifications of the MJPEG encoder.

Table 4-1. JPEG Specifications

Maximum Resolution	8k x 8k
Clock Rate	250MHz
Encode Rate	4bytes/cycle (2pixels/cycle in YUV422, 2.66 pixels/cycle in YUV420)
Feature Support	JPEG baseline profile Configuration compression ratio Two Quantization tables

Table 4-2 lists the theoretical throughput. Measured throughput is typically around the 80% mark and for YUV 4:2:2 format this was measured to be 416MP/s.

Table 4-2. Theoretical Throughput for the JPEG Encoder

Color Subsampling	**Throughput**
YUV 4:2:0	250x2.66= 666 MP/s
YUV 4:2:2	250x2 = 500MP/s

The hardware and software latency (which considers generating headers, bitstream) for JPEG encoding a single stream 1080P@30FPS is measured to be ~8.04 ms.

Detailed information on the JPEG encoder can be found in [6].

4.1 Open Source GStreamer Overview

Gstreamer is an open source framework that simplifies the development of multimedia applications, such as media players and capture encoders. It encapsulates existing multimedia software components such as codecs and hardware accelerators by using a standard interface and provides a uniform framework across applications. It is worthy to note that new GStreamer features are continuously being added and the core libraries are actively supported by members of the GStreamer community. Additional information about the GStreamer framework is available on the GStreamer project site [4].

The modular nature of GStreamer facilitates the addition of new functionality and allows for flexibility in application development and testing. Processing nodes are implemented via GStreamer plugins with several sink and/or source pads. Many plugins typically run as ARM software implementation, but for more complex SOCs, certain functions are better executed on hardware-accelerated IPs. The GStreamer plugins used in vision processing on AM62A can be found [5].

4.2 TI-Provided V4L2 Drivers for Multimedia

Video4Linux version 2 (V4L2) is an open source framework that provides a media interface to all Linux-based applications. V4L2 is a collection of device drivers and an API for supporting real-time video capture and video memory-to-memory operations on Linux systems.

Video encode and decode using the encoder and decoder hardware, respectively, are enabled as V4L2 drivers. The V4L2 is integrated with the encoder and decoder drivers by a thin layer that implements the V4L2 node ioctls and translates the V4L2 data structures to those understood by the encoder/decoder.

4.3 Hardware-Accelerated GStreamer Plugins

One benefit of using GStreamer as a multimedia framework is that the core libraries are already built and run on ARM Linux. A GStreamer plugin is the only component required to enable additional hardware features on TI's embedded processors with both ARM and hardware accelerators for multimedia. The open source GStreamer plugins provide elements for GStreamer pipelines that enable the use of hardware-accelerated video decoding through the V4L2 GStreamer plugin. The Gstreamer plugin and bitstream format support in AM62A processor SDK are given in Table 4-3.

Table 4-3. GStreamer Plugin and Format Support

	Gstreamer Plugin Support	Bit Stream Format Supported
Encoder	v4l2h264enc v4l2h265enc	V4L2_PIX_FMT_H264 V4L2_PIX_FMT_HEVC
Decoder	v4l2h264dec v4l2h265dec	V4L2_PIX_FMT_H264 V4L2_PIX_FMT_HEVC

5 Software Driver Architecture

As discussed in the previous section, At the user space level, TI uses GStreamer-an opensource framework for multimedia applications. The Gstreamer library in-turn interfaces with a Video for Linux (V4L2 plugin) which handles all the details specific to the underlying hardware accelerators.

Specifically, for the VPU the GStreamer based V4L2 plugin interfaces with the V4L2 codec kernel driver interface. The V4L2 driver in turn communicates with the firmware running on the encoder/decoder via inter-processor communication (IPC). Similarly, the JPEG driver is also based on the Video4Linux 2 (V4L2) API and is responsible for configuring both the JPEG encoder hardware and for generation of JPEG image headers once compressed image is generated. The driver is implemented using V4L2's M2M framework which is a framework for memory-to-memory devices that take data from memory, do some processing and write out processed data back to memory. The overall software driver architecture for leveraging encode/decode and JPEG encode functionality is shown in Figure 5-1.

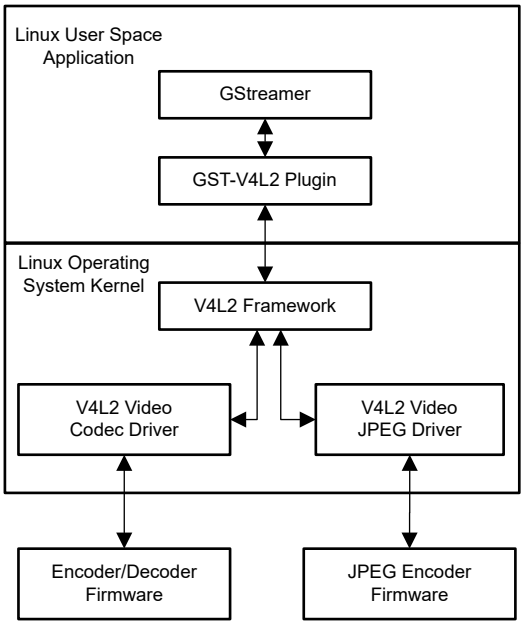

Figure 5-1. Software Driver Architecture Overview

6 Performance Measurements

This section provides memory requirements, latency and DDR bandwidth measurements for both H.264 and H.265 encode/decode on AM62A for typical configurations from 480P to 8MP. The measurements were made on the AM62Ax starter kit [3] using Linux SDK 9.0. The tool used to measure the DDR bandwidth and latency is the perf_stats tool. Details for replicating the performance benchmarks and relevant gstreamer commands can be found here [9].

Table 6-1. Single Instance Performance Results for Encoding With H.264/H.265

	8MP@15 FPS	5MP@30FPS	1080P@30FPS	480P@30FPS
	IMX219	OV5647	IMX219	IMX219
Memory Requirement (MB)	257.7 MB	76.16MB	43.2 MB	23.6 MB
Latency (ms)	33.3 ms	29ms	8.3 ms	4 ms
DDR BW (MB/s)	1271 MB/s	1200 MB/s	547 MB/s	137 MB/s

Table 6-2. Single Instance Performance Results for Decoding With H.264/H.265

	1080P@30FPS	480P@30FPS
	IMX219	IMX219
Memory Requirement (MB)	62.8 MB	29.5 MB
Latency (ms)	77 ms	36 ms
DDR BW (MB/s)	796 MB/s	75 MB/s

7 Multimedia Use Cases

7.1 Smart AI Box

Smart AI Box is a cost-effective way of adding intelligence to existing non-analytics-based cameras present in retail stores, traffic roads, factories, and buildings. Such a system receives live video streams from multiple cameras, decodes them, and does intelligent video analytics at the edge relieving the burden of transferring large video streams back to the cloud for analysis. The applications of AI Box include security surveillance system with anomaly or event detection, workplace safety systems that verifies workers wear personal protective equipment (PPE) such as goggles, safety vests, and hard hats before entering a hazardous zone.

Figure 7-1 illustrates the data flow for a system where there are two camera inputs coming in via ethernet which need to be depacketized and then decoded. Smart Analytics via deep learning is run on the decoded image stream. TI provides a model analyzer tool [7], where AI analytics designers can select the deep learning model of their choice based on accuracy, inference time and DDR bandwidth consumption. An example of one such utilization is given for SSDLite-MobDet-EdgeTPU-coco network in Table 7-1.

The targeted objects that are detected have bounding boxes drawn around them and the images are then re-encoded. This type of setup is used to bring analytics to legacy FHD cameras which do not have an analytics processor attached to them.

Table 7-2 provides the final resource utilization table that a systems engineer can construct. Power Estimation is done via the PET tool provided in [8].

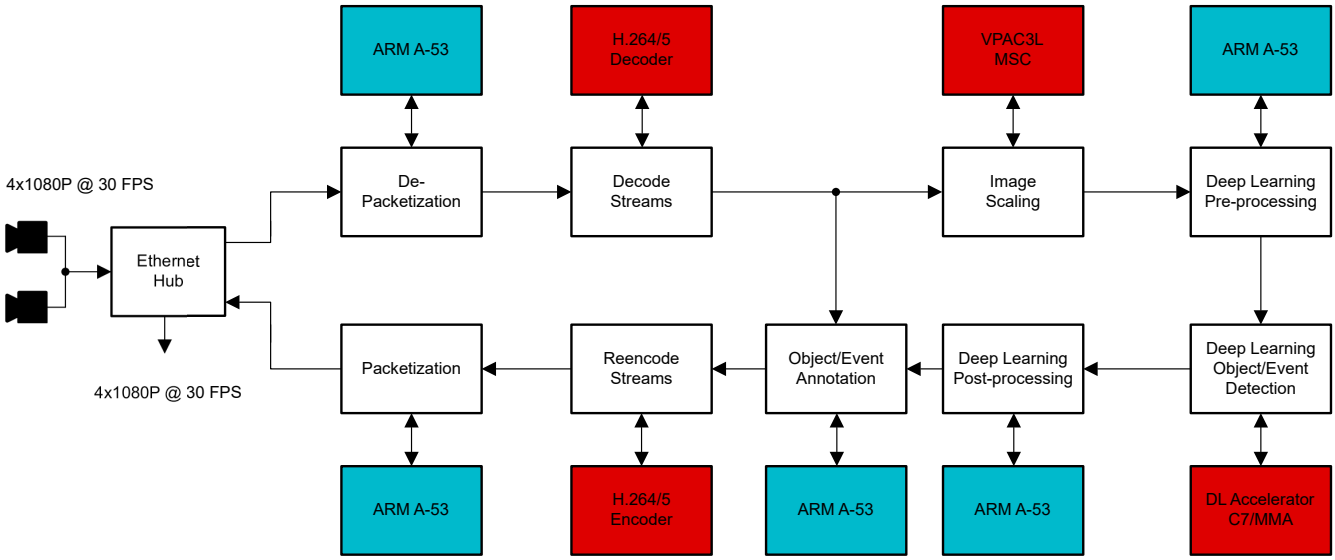

Figure 7-1. Smart AI Box

Table 7-1. Example Deep Learning Performance on AM62A With C7/MMA@850 MHz

Model	Resolution	Target FPS	MAP Accuracy On CoCo Data Set	Latency (ms)	Deep Learning Utilization	DDR BW Utilization
SSDLite-MobDet-. EdgeTPU-coco	320x320	30	29.7	8.35	25%	543 MB/s

Table 7-2. Example Resource Utilization Table for Smart AI Box Applications

Main IP	Loading
Decoder+ Encoder	4 x2MP@ 30 fps = 240MP/s
VPAC3L (ISP)	4x2MP@ 30 fps = 240MP/s
ARM loading @ 1.25GHz	11.4%
Deep Learning C7/MMA @850MHz	50%
DDR BW	2138 MB/s (2522- 384(Display))
Power Consumption Est (85c) using PET	~1.8W

In order to measure the DDR bandwidth a gstreamer based pipeline was constructed that emulated the data flow in Figure 7-1. From the server side 4 encoded streams where put on the network and the client side is used to decode the streams, go through the data flow and re-encode

- **Server side Gstreamer pipeline:**

```
gst-launch-1.0 -v v4l2src device=/dev/video-rpi-cam0 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=60/1, format=rggb10 ! \
tiovxisp sink_0::device=/dev/v4l-rpi-subdev0 sensor-name="SENSOR_SONY_IMX219_RPI" dcc-isp-
file=/opt/imaging/imx219/linear/dcc_viss_10b.bin sink_0::dcc-2a-file=/opt/imaging/imx219/linear/
dcc_2a_10b.bin format-msb=9 ! \
video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! \
tiovxmultiscaler name=msc \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! v4l2h264enc !
rtph264pay ! udpsink host=192.168.1.134 port=6001  \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! v4l2h264enc !
rtph264pay ! udpsink host=192.168.1.134 port=6002  \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! v4l2h264enc !
rtph264pay ! udpsink host=192.168.1.134 port=6003  \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! v4l2h264enc !
rtph264pay ! udpsink host=192.168.1.134 port=6004
```

- **Client side Gstreamer pipeline:**

```
gst-launch-1.0 -v udpsrc port=6001 ! 'application/x-rtp, media=(string)video, clock-
rate=(int)90000, encoding-name=(string)H264, payload=(int)96' ! \
rtph264depay ! h264parse ! v4l2h264dec capture-io-mode=4 ! tiovxmultiscaler name=msc \
msc. ! queue ! video/x-raw, width=640, height=480, format=NV12 ! kmssink driver-name=tidss \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080 ! v4l2h264enc output-
io-mode=5 extra-controls="controls,h264_i_frame_period=60,video_gop_size=60"  ! rtph264pay !
udpsink host=128.247.75.190 port=6001 &
gst-launch-1.0 -v udpsrc port=6002 ! 'application/x-rtp, media=(string)video, clock-
rate=(int)90000, encoding-name=(string)H264, payload=(int)96' ! \
rtph264depay ! h264parse ! v4l2h264dec capture-io-mode=4 ! queue ! v4l2h264enc output-
io-mode=dmabuf-import extra-controls="controls,h264_i_frame_period=60,video_gop_size=60"   !
rtph264pay ! udpsink host=128.247.75.190 port=6002 &
gst-launch-1.0 -v udpsrc port=6003 ! 'application/x-rtp, media=(string)video, clock-
rate=(int)90000, encoding-name=(string)H264, payload=(int)96' ! \
rtph264depay ! h264parse ! v4l2h264dec capture-io-mode=4 ! queue ! v4l2h264enc output-
io-mode=dmabuf-import extra-controls="controls,h264_i_frame_period=60,video_gop_size=60"  !
rtph264pay ! udpsink host=128.247.75.190 port=6003 &
gst-launch-1.0 -v udpsrc port=6004 ! 'application/x-rtp, media=(string)video, clock-
rate=(int)90000, encoding-name=(string)H264, payload=(int)96' ! \
rtph264depay ! h264parse ! v4l2h264dec capture-io-mode=4 ! queue ! v4l2h264enc output-
io-mode=dmabuf-import extra-controls="controls,h264_i_frame_period=60,video_gop_size=60"   !
rtph264pay ! udpsink host=128.247.75.190 port=6004 &
```

7.2 Surveillance

In many surveillance applications like traffic monitoring, event recognition systems, and drones it is desired to process a single camera stream at 8MP. AM62A's ISP has a 4k linewidth and is therefore able to support high-resolution camera sensors. The internal ISP then downscales the image to lower resolutions and corrects for camera lens distortion effects before processing by the deep learning accelerator. This is illustrated in Figure 7-2.

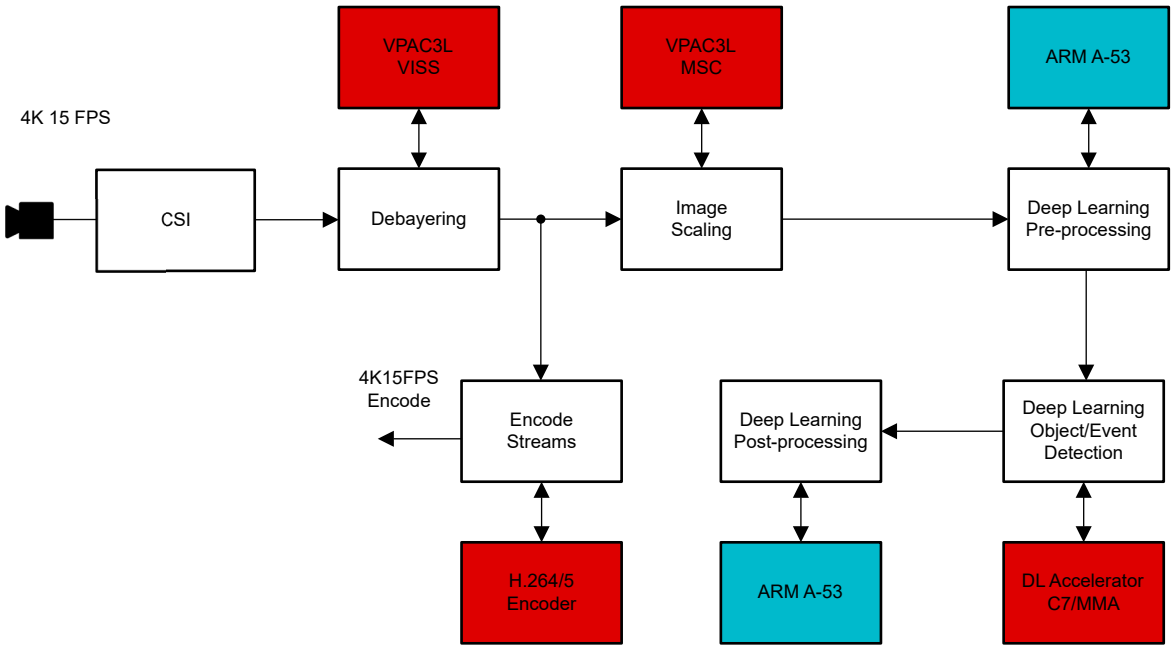

Figure 7-2. Smart Surveillance Single Encode Use Case

Table 7-3. Resource Utilization for 8MP Surveillance Camera With Encode and Analytics Running

Main IP	Loading
Encoder	8MP@ 15 fps = 120MP/s
VPAC3L (ISP)	8MP@ 15fps = 120MP/s
ARM loading @ 1.25GHz	16%
Deep Learning C7/MMA @850MHz	25%
DDR BW	1860 MB/s (2410 – 550(Display)) MB/s
Power Consumption Est (85c) using PET	~1.5W

- **Gstreamer Pipeline used:**

```
gst-launch-1.0 -v v4l2src device=/dev/video3  io-mode=dmabuf-import ! \
video/x-bayer, width=3280, height=2464, framerate=15/1, format=rggb ! \
tiovxisp sink_0::device=/dev/v4l-subdev2 sensor-name="SENSOR_SONY_IMX219_RPI" dcc-isp-file=/opt/
imaging/imx219/linear/dcc_viss.bin sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a.bin
format-msb=7 ! \
video/x-raw, format=NV12, width=3280, height=2464, framerate=15/1 ! \
tiovxmultiscaler  name=msc \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080 ! kmssink driver-name=tidss  \
msc. ! queue ! video/x-raw, format=NV12, width=3280, height=2464, framerate=15/1 ! v4l2h264enc !
\
rtph264pay ! udpsink host=192.168.65.187 port=5002
```

In many examples where the surveillance camera does not have a physical network connection like ethernet, the video stream needs to be encoded and sent through Wi-Fi. In this case, some applications need to encode the stream simultaneously at different resolutions and select the encode stream for Wi-Fi transmission based on what the network traffic is capable of supporting. This is illustrated in Figure 7-3.

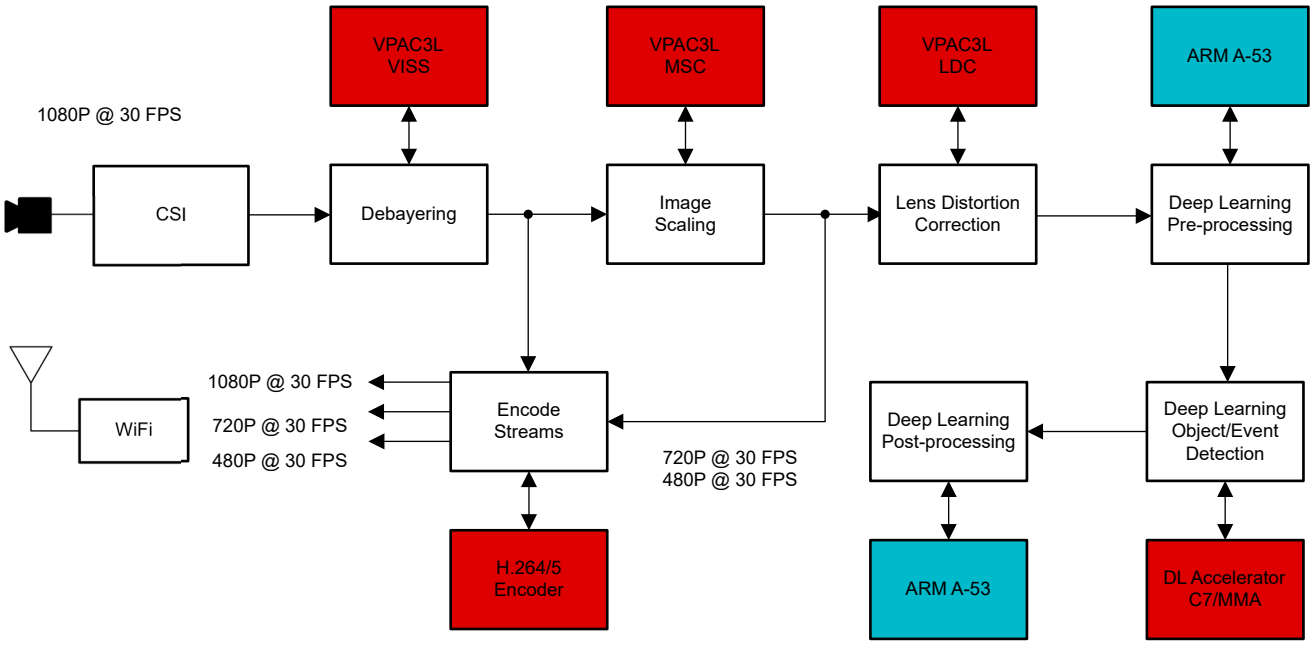

Figure 7-3. Smart Surveillance Multiple Encode Use Case

Table 7-4. Smart Surveillance Multiple Encode Use Case Resource Utilization

Main IP	Loading
Decoder+ Encoder	(2MP+0.9M+0.3MP)@ 30 fps = 96MP/s
VPAC3L (ISP)	2MP@ 30 fps = 60MP/s
ARM loading @ 1.25GHz	34.5%
Deep Learning C7/MMA @850MHz	25%
DDR BW	1310 MB/s (1860-550(Display))
Power Consumption Est (85c) using PET	

- **Gstreamer Pipeline used:**

```
gst-launch-1.0 -v v4l2src device=/dev/video-rpi-cam0 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
tiovxisp sink_0::device=/dev/v4l-rpi-subdev0 sensor-name="SENSOR_SONY_IMX219_RPI" \
dcc-isp-file=/opt/imaging/imx219/linear/dcc_viss_10b.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a_10b.bin format-msb=9 ! \
video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! tiovxmultiscaler name=msc \
msc. ! queue ! video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! \
v4l2h264enc output-io-mode=dmabuf-import extra-
controls="controls,h264_i_frame_period=60,video_gop_size=60"! \
rtph264pay ! udpsink host=192.168.65.187 port=5001 \
msc. ! queue ! video/x-raw, format=NV12, width=1280, height=720, framerate=30/1 ! \
v4l2h264enc output-io-mode=dmabuf-import extra-
controls="controls,h264_i_frame_period=60,video_gop_size=60"! \
rtph264pay ! udpsink host=192.168.65.187 port=5002 \
msc. ! queue ! video/x-raw, format=NV12, width=640, height=480, framerate=30/1 ! tee name=t1 \
t1. ! queue ! v4l2h264enc output-io-mode=dmabuf-import extra-
controls="controls,h264_i_frame_period=60,video_gop_size=60"! \
rtph264pay ! udpsink host=192.168.65.187 port=5003 \
t1. ! queue ! kmssink driver-name=tidss
```

8 Summary

This application note describes the multimedia capability of the AM62A device. Typical benchmarks and two different examples are presented on how multimedia accelerators are used.

9 References

1. AM62A Product
2. AM62A Multimedia Academy
3. AM62A Starter Kit
4. Gstreamer Project Site
5. Gstreamer plugins for Vision
6. JPEG Encoder
7. TI's Model Analyzer
8. Texas Instruments: *AAM62A Power Estimation Tool*
9. Example Gstreamer Pipelines

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265