

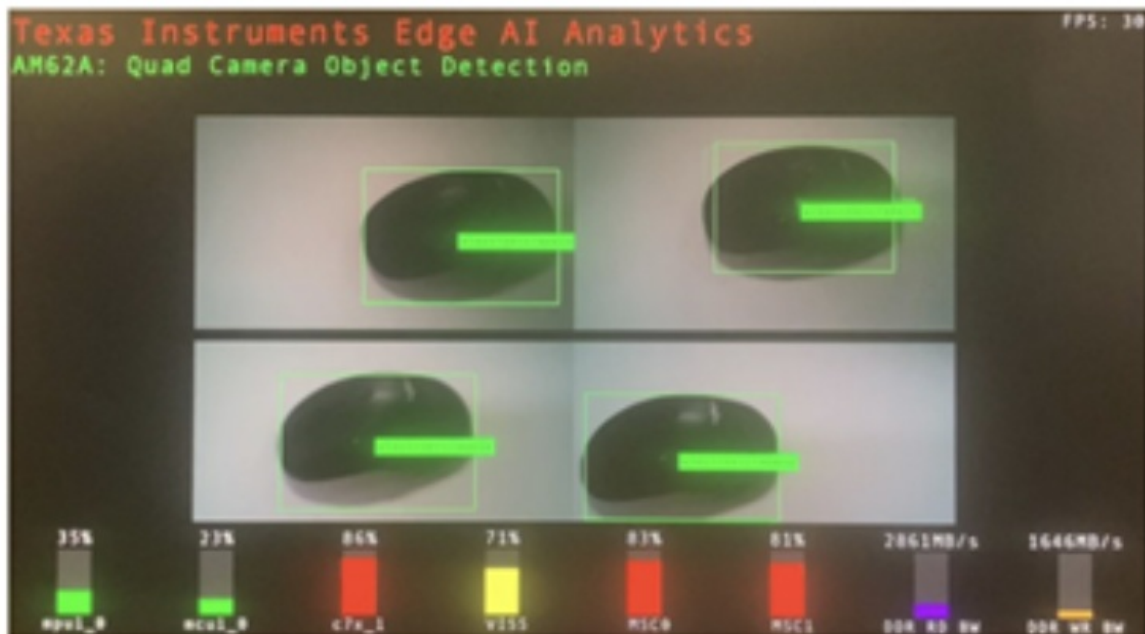


Contents [[hide](#)]

- [1 Texas Instruments AM6x Developing Multiple Cameras](#)
- [2 Specifications](#)
- [3 Introduction](#)
- [4 Enabling Multiple Cameras in Software](#)
- [5 Reference Design](#)
- [6 Multicamera Deep Learning Inference](#)
- [7 Performance Analysis](#)
- [8 IMPORTANT NOTICE AND DISCLAIMER](#)
- [9 Frequently Asked Questions](#)
- [10 Documents / Resources](#)
 - [10.1 References](#)



Texas Instruments AM6x Developing Multiple Cameras



Specifications

- Product Name: AM6x family of devices
- Supported Camera Type: AM62A (With or without built-in ISP), AM62P (With Built-in ISP)
- Camera Output Data: AM62A (Raw/YUV/RGB), AM62P (YUV/RGB)
- ISP HWA: AM62A (Yes), AM62P (No)
- Deep Learning HWA: AM62A (Yes), AM62P (No)
- 3-D Graphics HWA: AM62A (No), AM62P (Yes)

Introduction to Multiple-Camera Applications on AM6x:

- Embedded cameras play a crucial role in modern vision systems.
- Utilizing multiple cameras in a system enhances capabilities and enables tasks not achievable with a single camera.

Applications Using Multiple Cameras:

- **Security Surveillance:** Enhances surveillance coverage, object tracking, and recognition accuracy.
- **Surround View:** Enables stereo vision for tasks like obstacle detection and object manipulation.
- **Cabin Recorder and Camera Mirror System:** Provides extended coverage and

eliminates blind spots.

- **Medical Imaging:** Offers enhanced precision in surgical navigation and endoscopy.
- **Drones and Aerial Imaging:** Capture high-resolution images from different angles for various applications.

Connecting Multiple CSI-2 Cameras to the SoC:

To connect multiple CSI-2 cameras to the SoC, follow the guidelines provided in the user manual. Ensure proper alignment and connection of each camera to the designated ports on the SoC.

Application Note

Developing Multiple-Camera Applications on AM6x

Jianzhong Xu, Qutaiba Saleh

ABSTRACT

This report describes application development using multiple CSI-2 cameras on the AM6x family of devices. A reference design of object detection with deep learning on 4 cameras on the AM62A SoC is presented with performance analysis. General principles of the design apply to other SoCs with a CSI-2 interface, such as AM62x and AM62P.

Introduction

Embedded cameras play an important role in modern vision systems. Using multiple cameras in a system expands the capabilities of these systems and enables capabilities that are not possible with a single camera. Below are some examples of applications using multiple embedded cameras:

- **Security Surveillance:** Multiple cameras placed strategically provide comprehensive surveillance coverage. They enable panoramic views, reduce blind spots, and enhance the accuracy of object tracking and recognition, improving overall security measures.
- **Surround View:** Multiple cameras are used to create a stereo vision setup, enabling three-dimensional information and the estimation of depth. This is crucial for tasks such as obstacle detection in autonomous vehicles, precise object manipulation in

robotics, and enhanced realism of augmented reality experiences.

- **Cabin Recorder and Camera Mirror System:** A car cabin recorder with multiple cameras can provide more coverage using a single processor. Similarly, a camera mirror system with two or more cameras can expand the driver's field of view and eliminate blind spots from all sides of a car.
- **Medical Imaging:** Multiple cameras can be used in medical imaging for tasks like surgical navigation, providing surgeons with multiple perspectives for enhanced precision. In endoscopy, multiple cameras enable a thorough examination of internal organs.
- **Drones and Aerial Imaging:** Drones often come equipped with multiple cameras to capture high-resolution images or videos from different angles. This is useful in applications like aerial photography, agriculture monitoring, and land surveying.
- With the advancement of microprocessors, multiple cameras can be integrated into a single System-on-Chip.

(SoC) to provide compact and efficient solutions. The AM62Ax SoC, with high-performance video/vision processing and deep learning acceleration, is an ideal device for the above-mentioned use cases. Another AM6x device, the AM62P, is built for high-performance embedded 3D display applications. Equipped with 3D graphics acceleration, the AM62P can easily stitch together the images from multiple cameras and produce a high-resolution panoramic view. The innovative features of the AM62A/AM62P SoC have been presented in various publications, such as [4], [5], [6], etc. This application note will not repeat those feature descriptions but instead focuses on integrating multiple CSI-2 cameras into embedded vision applications on AM62A/AM62P.

- Table 1-1 shows the main differences between AM62A and AM62P as far as image processing is concerned.

Table 1-1. Differences Between AM62A and AM62P in Image Processing

SoC	AM62A	AM62P
Supported Camera Type	With or without a built-in ISP	With Built-in ISP

Camera Output Data	Raw/YUV/RGB	YUV/RGB
ISP HWA	Yes	No
Deep Learning HWA	Yes	No
3-D Graphics HWA	No	Yes

Connecting Multiple CSI-2 Cameras to the SoC

The Camera Subsystem on the AM6x SoC contains the following components, as shown in Figure 2-1:

- MIPI D-PHY Receiver: receives video streams from external cameras, supporting up to 1.5 Gbps per data lane for 4 lanes.
- CSI-2 Receiver (RX): receives video streams from the D-PHY receiver and either directly sends the streams to the ISP or dumps the data to DDR memory. This module supports up to 16 virtual channels.
- SHIM: a DMA wrapper that enables sending the captured streams to memory over DMA. Multiple DMA contexts can be created by this wrapper, with each context corresponding to a virtual channel of the CSI-2 Receiver.

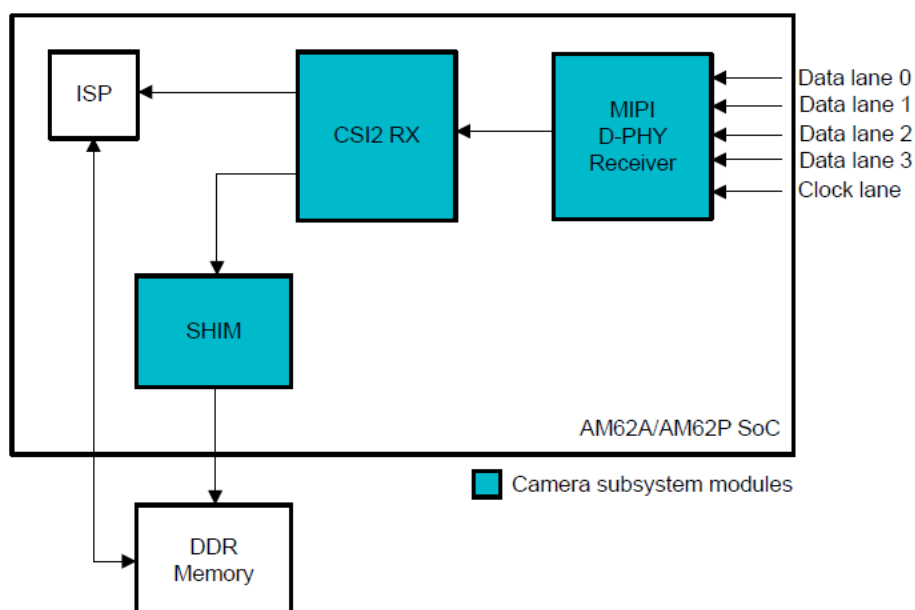


Figure 2-1. High-Level Block Diagram of the Camera Subsystem on AM62A/AM62P SoC

Multiple cameras can be supported on the AM6x through the use of virtual channels of CSI-2 RX, even though there is only one CSI-2 RX interface on the SoC. An external

CSI-2 aggregating component is needed to combine multiple camera streams and send them to a single SoC. Two types of CSI-2 aggregating solutions can be used, described in the following sections.

CSI-2 Aggregator Using SerDes

One way of combining multiple camera streams is to use a serializing and deserializing (SerDes) solution. The CSI-2 data from each camera is converted by a serializer and transferred through a cable. The deserializer receives all serialized data transferred from the cables (one cable per camera), converts the streams back to CSI-2 data, and then sends out an interleaved CSI-2 stream to the single CSI-2 RX interface on the SoC. Each camera stream is identified by a unique virtual channel. This aggregating solution offers the additional benefit of allowing long-distance connection of up to 15m from the cameras to the SoC.

The FPD-Link or V3-Link serializers and deserializers (SerDes), supported in the AM6x Linux SDK, are the most popular technologies for this type of CSI-2 aggregating solution. Both the FPD-Link and V3-Link deserializers have back channels that can be used to send frame sync signals to synchronize all the cameras, as explained in [7]. Figure 2-2 shows an example of using the SerDes to connect multiple cameras to a single AM6x SoC.

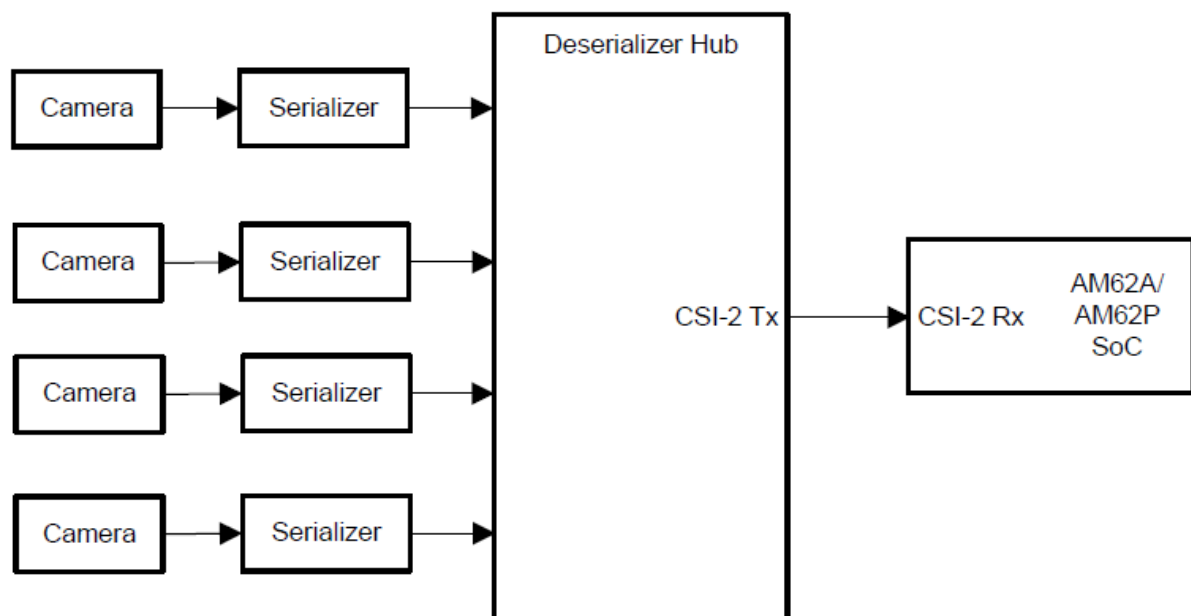


Figure 2-2. Connecting Multiple Cameras Using SerDes

An example of this aggregating solution can be found in the Arducam V3Link Camera

Solution Kit. This kit has a deserializer hub which aggregates 4 CSI-2 camera streams, as well as 4 pairs of V3link serializers and IMX219 cameras, including FAKRA coaxial cables and 22-pin FPC cables. The reference design discussed later is built on this kit.

CSI-2 Aggregator without Using SerDes

This type of aggregator can directly interface with multiple MIPI CSI-2 cameras and aggregate the data from all cameras to a single CSI-2 output stream.

Figure 2-3 shows an example of such a system. This type of aggregating solution does not use any serializer/deserializer but is limited by the maximum distance of CSI-2 data transfer, which is up to 30cm. The AM6x Linux SDK does not support this type of CSI-2 aggregator

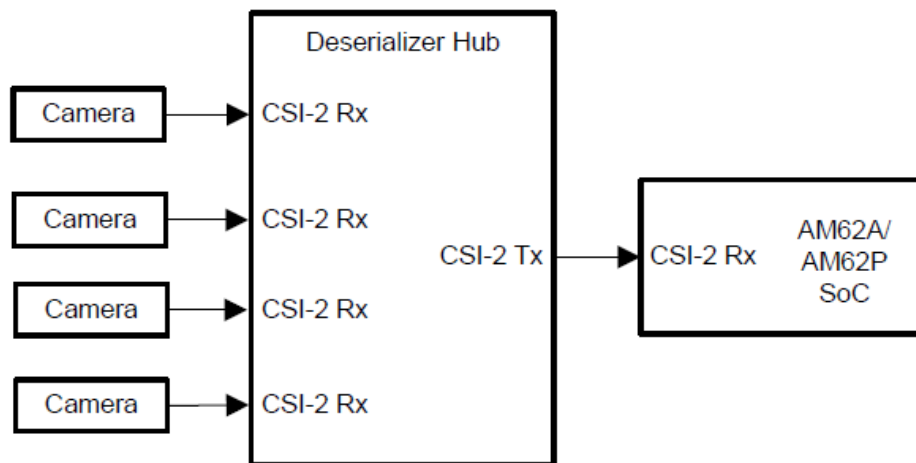


Figure 2-3. Connecting Multiple Cameras Using CSI-2 Aggregator

Enabling Multiple Cameras in Software

Camera Subsystem Software Architecture

Figure 3-1 shows a high-level block diagram of the camera capture system software in AM62A/AM62P Linux SDK, corresponding to the HW system in Figure 2-2.

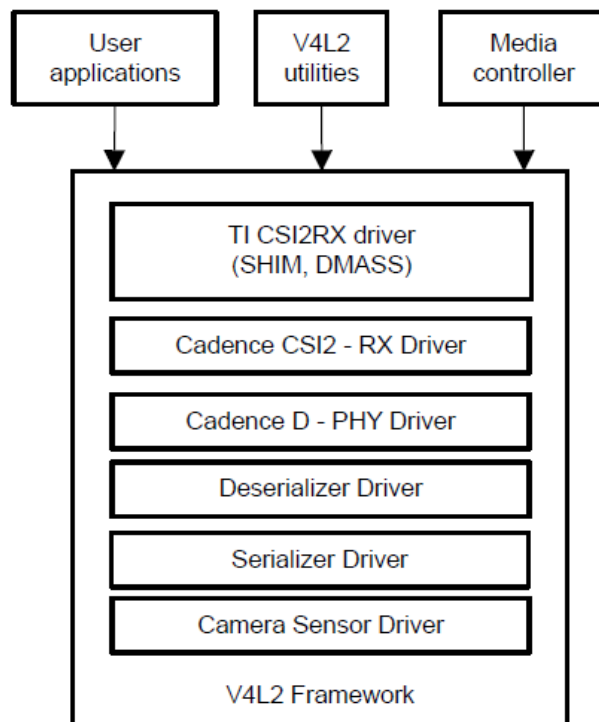


Figure 3-1. High-Level Block Diagram of Camera Capture System Using SerDes

- This software architecture enables the SoC to receive multiple camera streams with the use of SerDes, as shown in Figure 2-2. The FPD-Link/V3-Link SerDes assigns a unique I2C address and virtual channel to each camera. A unique device tree overlay should be created with the unique I2C address for every camera. The CSI-2 RX driver recognizes each camera using the unique virtual channel number and creates a DMA context per camera stream. A video node is created for every DMA context. Data from each camera is then received and stored using DMA to the memory accordingly. User space applications use the video nodes corresponding to each camera to access the camera data. Examples of using this software architecture are given in Chapter 4 – Reference Design.
- Any specific sensor driver that is compliant with the V4L2 framework can plug and play in this architecture. Refer to [8] regarding how to integrate a new sensor driver into the Linux SDK.

Image Pipeline Software Architecture

- The AM6x Linux SDK provides the GStreamer (GST) framework, which can be used in the ser space to integrate the image processing components for various applications. The Hardware Accelerators (HWA) on the SoC, such as the Vision Pre-processing Accelerator (VPAC) or ISP, video encoder/decoder, and deep learning compute

engine, are accessed through GST plugins. The VPAC (ISP) itself has multiple blocks, including Vision Imaging Sub-System (VISS), Lens Distortion Correction (LDC), and Multiscalar (MSC), each corresponding to a GST plugin.

- Figure 3-2 shows the block diagram of a typical image pipeline from the camera to encoding or deep learning applications on AM62A. For more details about the end-to-end data flow, refer to the EdgeAI SDK documentation.

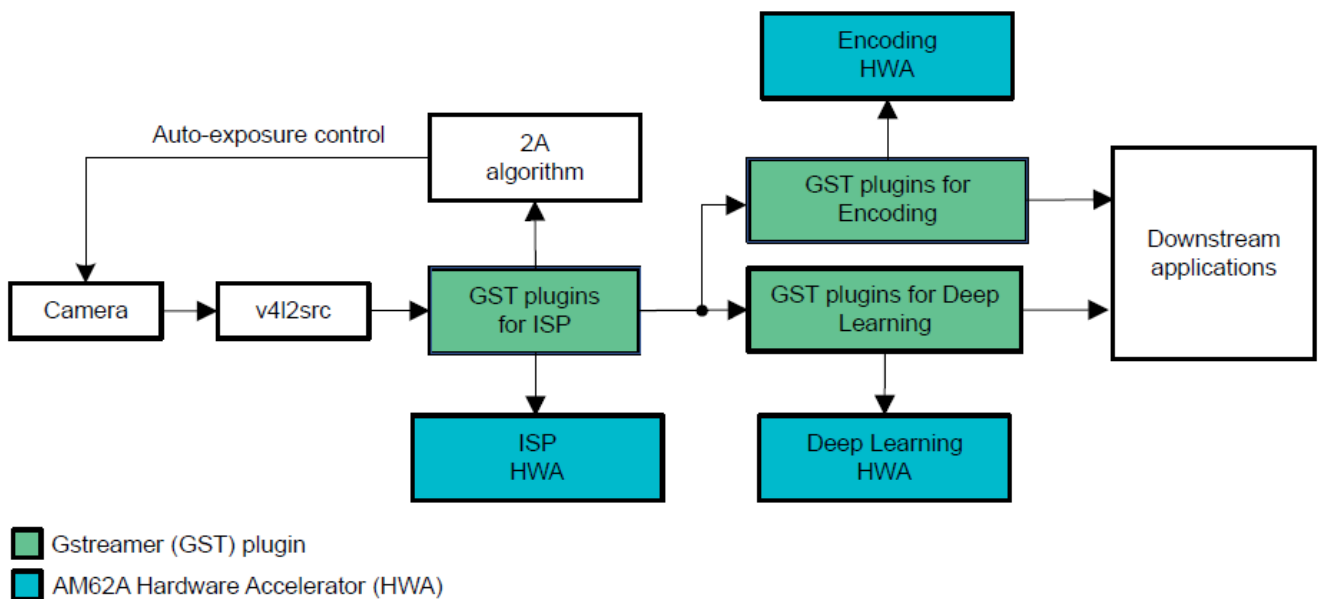


Figure 3-2. A Typical AM62A Image Pipeline Using GStreamer

For AM62P, the image pipeline is simpler because there is no ISP on AM62P.

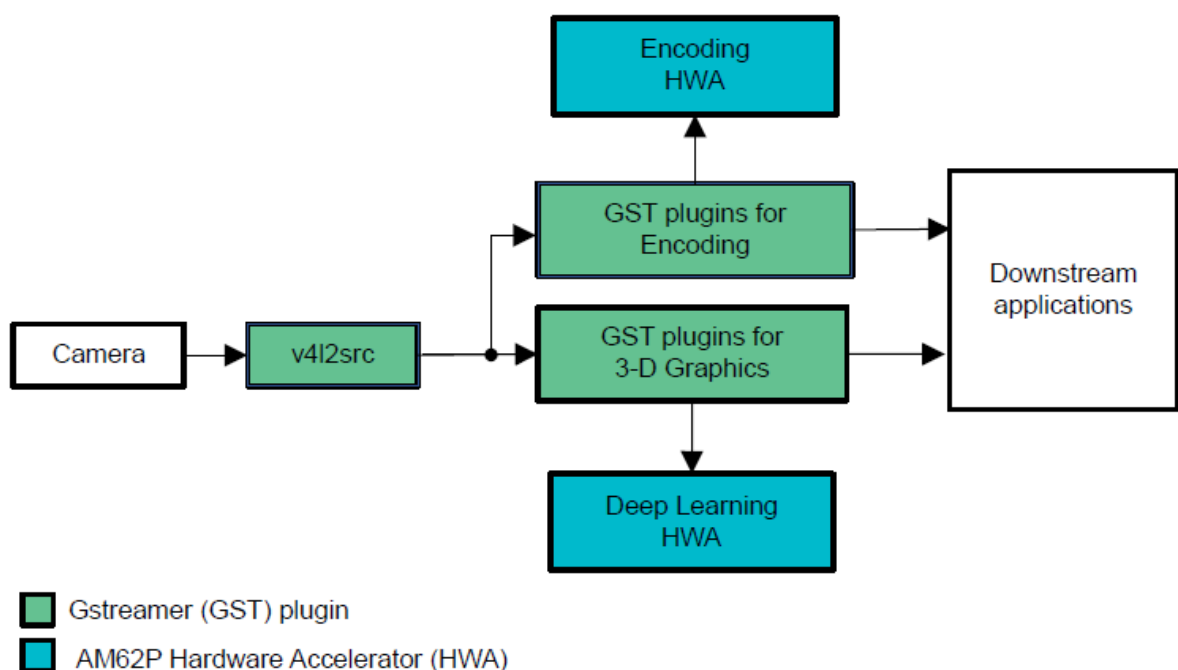


Figure 3-3. A Typical AM62P Image Pipeline Using GStreamer

With a video node created for each of the cameras, the GStreamer-based image pipeline allows the processing of multiple camera inputs (connected through the same CSI-2 RX interface) simultaneously. A reference design using GStreamer for multi-camera applications is given in the next chapter.

Reference Design

This chapter presents a reference design of running multiple-camera applications on AM62A EVM, using the Arducam V3Link Camera Solution Kit to connect 4 CSI-2 cameras to AM62A and running object detection for all 4 cameras.

Supported Cameras

The Arducam V3Link kit works with both FPD-Link/V3-Link-based cameras and Raspberry Pi-compatible CSI-2 cameras. The following cameras have been tested:

- D3 Engineering D3RCM-IMX390-953
- Leopard Imaging LI-OV2312-FPDLINKIII-110H
- IMX219 cameras in the Arducam V3Link Camera Solution Kit

Setting up Four IMX219 Cameras

Follow the instructions provided in the AM62A Starter Kit EVM Quick Start Guide to set up the SK-AM62A-LP EVM (AM62A SK) and ArduCam V3Link Camera Solution Quick Start Guide to connect the cameras to AM62A SK through the V3Link kit. Make sure the pins on the flex cables, cameras, V3Link board, and AM62A SK are all aligned properly.

Figure 4-1 shows the setup used for the reference design in this report. The main components in the setup include:

- 1X SK-AM62A-LP EVM board
- 1X Arducam V3Link d-ch adapter board
- FPC cable connecting Arducam V3Link to SK-AM62A
- 4X V3Link camera adapters (serializers)
- 4X RF coaxial cables to connect V3Link serializers to V3Link d-ch kit
- 4X IMX219 Cameras
- 4X CSI-2 22-pin cables to connect cameras to serializers

- Cables: HDMI cable, USB-C to power SK-AM62A-LP and 12V power sourced for V3Link d-ch kit)
- Other components not shown in Figure 4-1: micro-SD card, micro-USB cable to access SK-AM62A-LP, and Ethernet for streaming

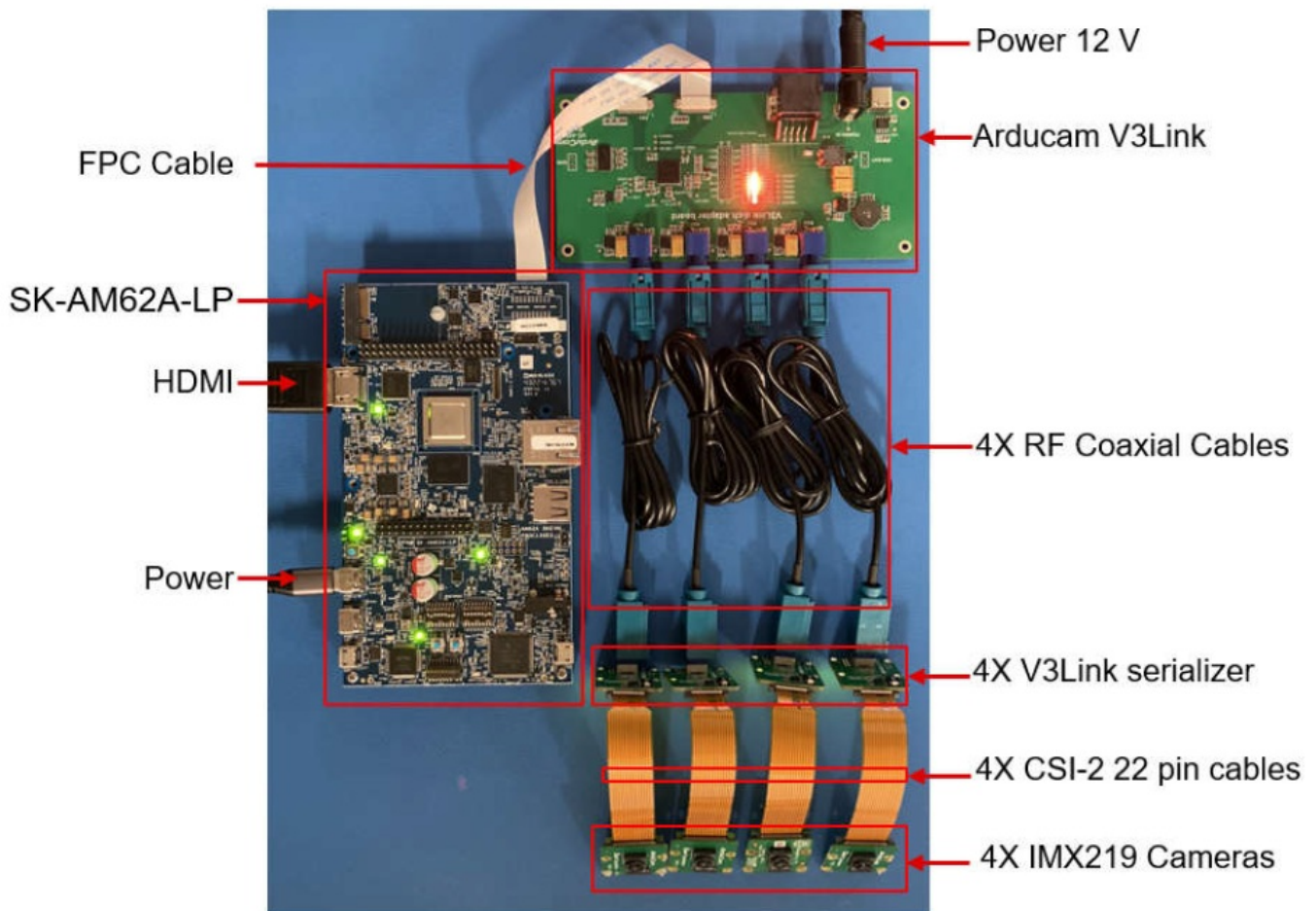


Figure 4-1. V3link Board and 4 Cameras Setup Used in the Reference Design

Configuring Cameras and CSI-2 RX Interface

Set up the software according to the instructions provided in the Arducam V3Link Quick Start Guide. After running the camera setup script, `setup-imx219.sh`, the camera's format, the CSI-2 RX interface format, and the routes from each camera to the corresponding video node will be configured properly. Four video nodes are created for the four IMX219 cameras. Command "`v4l2-ctl --list-devices`" displays all the V4L2 video devices, as shown below:

```

root@am62axx-evm:~# v4l2-ctl --list-devices
j721e-csi2rx (platform:30102000.ticsi2rx):
    /dev/video2
    /dev/video3
    /dev/video4
    /dev/video5
    /dev/video6
    /dev/video7
    /dev/media0

wave5-dec (platform:wave5-dec):
    /dev/video0

wave5-enc (platform:wave5-enc):
    /dev/video1

```

There are 6 video nodes and 1 media node under ticsi2rx. Each video node corresponds to a DMA context allocated by the CSI2 RX driver. Out of the 6 video nodes, 4 are used for the 4 IMX219 cameras, as shown in the media pipe topology below:

```

root@am62axx-evm:~# media-ctl -p

Device topology
- entity 1: 30102000.ticsi2rx (7 pads, 7 links, 4 routes)
  type V4L2 subdev subtype Unknown flags 0
  device node name /dev/v4l-subdev0
  routes:
    0/0 -> 1/0 [ACTIVE]
    0/1 -> 2/0 [ACTIVE]
    0/2 -> 3/0 [ACTIVE]
    0/3 -> 4/0 [ACTIVE]
  pad0: Sink
    [stream:0 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    [stream:1 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    [stream:2 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    [stream:3 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    <- "cdns_csi2rx.30101000.csi-bridge":1 [ENABLED,IMMUTABLE]
  pad1: Source
    [stream:0 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    -> "30102000.ticsi2rx context 0":0 [ENABLED,IMMUTABLE]
  pad2: Source
    [stream:0 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    -> "30102000.ticsi2rx context 1":0 [ENABLED,IMMUTABLE]
  pad3: Source
    [stream:0 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    -> "30102000.ticsi2rx context 2":0 [ENABLED,IMMUTABLE]
  pad4: Source
    [stream:0 fmt:UYVY8_1X16/640x480 field:none colorspace:srgb xfer:srgb ycbcr:601
quantization:lim-range]
    -> "30102000.ticsi2rx context 3":0 [ENABLED,IMMUTABLE]
  pad5: Source
    -> "30102000.ticsi2rx context 4":0 [ENABLED,IMMUTABLE]
  pad6: Source
    -> "30102000.ticsi2rx context 5":0 [ENABLED,IMMUTABLE]

```

As shown above, media entity 30102000.ticsi2rx has 6 source pads, but only the first 4 are used, each for one IMX219. The media pipe topology can also be illustrated graphically. Run the following command to generate a dot file:

```

root@am62axx-evm:~# media-ctl --print-dot > media.dot

```

Then run the command below on a Linux host PC to generate a PNG file:

```
$ dot -Tpng media-top.dot -o media-top.png
```

Figure 4-2 is a picture generated using the commands given above. The components in the software architecture of Figure 3-1 can be found in this graph.

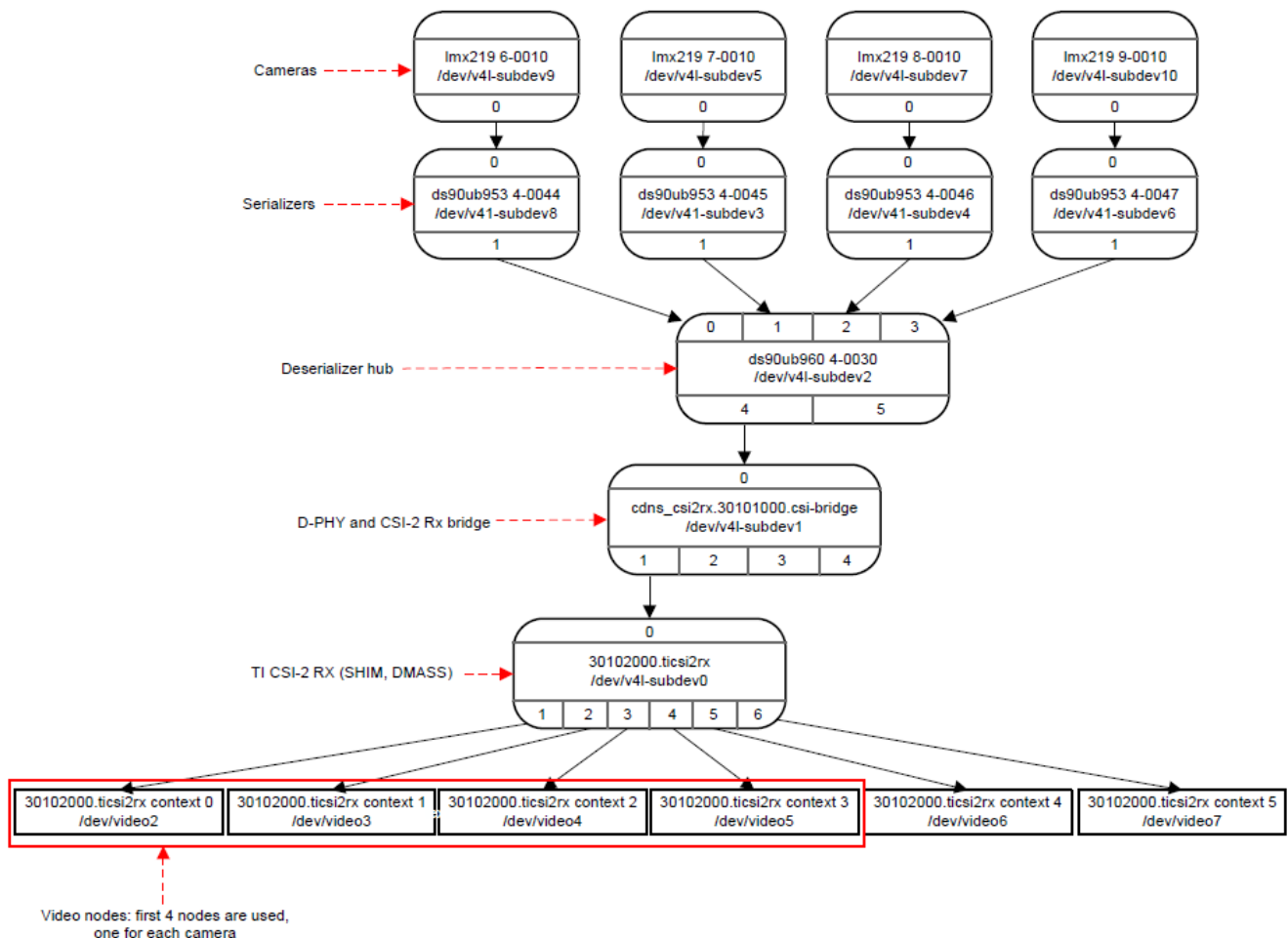


Figure 4-2. Media Topology of a Multi-Camera System

Streaming from Four Cameras

With both hardware and software being set up properly, multiple-camera applications can run from the user space. For AM62A, the ISP must be tuned to produce good image quality. Refer to the AM6xA ISP Tuning Guide for how to perform ISP tuning. The following sections present examples of streaming camera data to a display, streaming camera data to a network, and storing the camera data to files.

Streaming Camera Data to Display

A basic application of this multi-camera system is to stream the videos from all cameras to a display connected to the same SoC. The following is a GStreamer pipeline example of streaming four IMX219 to a display (the video node numbers and v4l-subdev numbers

in the pipeline will likely change from reboot to reboot).

```
gst-launch-1.0 \
v4l2src device=/dev/video2 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovxisp sink_0::device=/dev/v4l-subdev9 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/
imaging/imx219/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin format-msb=7 sink_0::pool-size=8
src::pool-size=8 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_0 \
v4l2src device=/dev/video3 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovxisp sink_0::device=/dev/v4l-subdev5 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/
imaging/imx219/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin format-msb=7 sink_0::pool-size=8
src::pool-size=8 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_1 \
v4l2src device=/dev/video4 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovxisp sink_0::device=/dev/v4l-subdev7 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/
imaging/imx219/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin format-msb=7 sink_0::pool-size=8
src::pool-size=8 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_2 \
v4l2src device=/dev/video5 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovxisp sink_0::device=/dev/v4l-subdev10 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/
imaging/imx219/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin format-msb=7 sink_0::pool-size=8
src::pool-size=8 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_3 \
tiovxmosaic name=mosaic \
sink_0::startx="<0>" sink_0::starty="<0>" sink_0::widths="<640>" sink_0::heights="<480>" \
sink_1::startx="<0>" sink_1::starty="<480>" sink_1::widths="<640>" sink_1::heights="<480>" \
sink_2::startx="<640>" sink_2::starty="<0>" sink_2::widths="<640>" sink_2::heights="<480>" \
sink_3::startx="<640>" sink_3::starty="<480>" sink_3::widths="<640>" sink_3::heights="<480>" ! \
queue ! video/x-raw, width=1920, height=1080 ! kmssink driver-name=tidss sync=false force-
modesetting=true
```

Streaming Camera Data through Ethernet

Instead of streaming to a display connected to the same SoC, the camera data can also be streamed through the Ethernet. The receiving side can be either another AM62A/AM62P processor or a host PC. The following is an example of streaming the camera data through the Ethernet (using two cameras for simplicity) (note the encoder plugin used in the pipeline):

```
gst-launch-1.0 \
v4l2src device=/dev/video2 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/dcc_viss_1920x1080.bin
\
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin sink_0::device=/dev/v4l-subdev9 !
queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc ! rtph264pay ! udpsink
port=5000 host=<receiving IP address> \
v4l2src device=/dev/video3 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/dcc_viss_1920x1080.bin
\
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin sink_0::device=/dev/v4l-subdev5 !
queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc ! rtph264pay ! udpsink
port=5001 host=<receiving IP address>
```

The following is an example of receiving the camera data and streaming to a display on another AM62A/AM62P processor:

```
gst-launch-1.0 -v \
udpsrc port=5000 ! 'application/x-rtp, encoding-name=H264, payload=96' ! rtpH264depay !
avdec_h264 ! queue ! videoconvert ! queue ! \
video/x-raw,format=NV12,width=1920,height=1080 ! queue ! mosaic.sink_0 \
udpsrc port=5001 ! 'application/x-rtp, encoding-name=H264, payload=96' ! rtpH264depay !
avdec_h264 ! queue ! videoconvert ! queue ! \
video/x-raw,format=NV12,width=1920,height=1080 ! queue ! mosaic.sink_1 \
tiovmosaic name=mosaic \
sink_0::startx="<0>" sink_0::starty="<0>" sink_0::widths="<960>" sink_0::heights="<540>" \
sink_1::startx="<960>" sink_1::starty="<540>" sink_1::widths="<960>" sink_1::heights="<540>" ! \
queue ! kmsink driver-name=tidss sync=false
```

Storing Camera Data to Files

Instead of streaming to a display or through a network, the camera data can be stored in local files. The pipeline below stores each camera's data to a file (using two cameras as an example for simplicity).

```
gst-launch-1.0 \
v4l2src device=/dev/video2 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovmxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin sink_0::device=/dev/v4l-subdev9 !
queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc ! filesink location=cam-
cap-1.mp4 \
v4l2src device=/dev/video3 io-mode=5 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! queue leaky=2 ! \
tiovmxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_1920x1080.bin sink_0::device=/dev/v4l-subdev5 !
queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc ! filesink
location=cam-cap-2.mp4
```

Multicamera Deep Learning Inference

AM62A is equipped with a deep learning accelerator (C7x-MMA) with up to two TOPS, which are capable of running various types of deep learning models for classification, object detection, semantic segmentation, and more. This section shows how AM62A can simultaneously run four deep learning models on four different camera feeds.

Model Selection

The TI's EdgeAI-ModelZoo provides hundreds of state-of-the-art models, which are converted/exported from their original training frameworks to an embedded-friendly format so that they can be offloaded to the C7x-MMA deep learning accelerator. The cloud-based Edge AI Studio Model Analyzer provides an easy-to-use "Model Selection" tool. It is dynamically updated to include all models supported in TI EdgeAI-ModelZoo. The tool requires no previous experience and provides an easy-to-use interface to enter the features required in the desired model.

The TFL-OD-2000-ssd-mobV1-coco-mlperf was selected for this multi-camera deep learning experiment. This multi-object detection model is developed in the TensorFlow framework with a 300×300 input resolution. Table 4-1 shows the important features of this model when trained on the cCOCO dataset with about 80 different classes.

Table 4-1. Highlight Features of the Model TFL-OD-2000-ssd-mobV1-coco-mlperf.

Model	Task	Resolut ion	FPS	mAP 50 % Accuracy On COC O	Latency/F rame (ms)	DDR BW Utilizatio n (MB/ Fr ame)
TFL-OD-2000 -ssd- mobV1-coco- mlperf	Multi O bject D etection	300×30 0	~152	15.9	6.5	18.839

Pipeline Setup

Figure 4-3 shows the 4-camera deep learning GStreamer pipeline. TI provides a suite of GStreamer plugins that allow offloading some of the media processing and the deep learning inference to the hardware accelerators. Some examples of these plugins include tiovxisp, tiovxmultiscaler, tiovxmosaic, and tidlinferer. The pipeline in Figure 4-3 includes all required plugins for a multipath GStreamer pipeline for 4-camera inputs, each with media preprocess, deep learning inference, and postprocess. The duplicated plugins for each of the camera paths are stacked in the graph for easier demonstration. The available hardware resources are evenly distributed among the four camera paths. For instance, AM62A contains two image multiscalers: MSC0 and MSC1. The pipeline explicitly dedicates MSC0 to process camera 1 and camera 2 paths, while MSC1 is dedicated to camera 3 and camera 4.

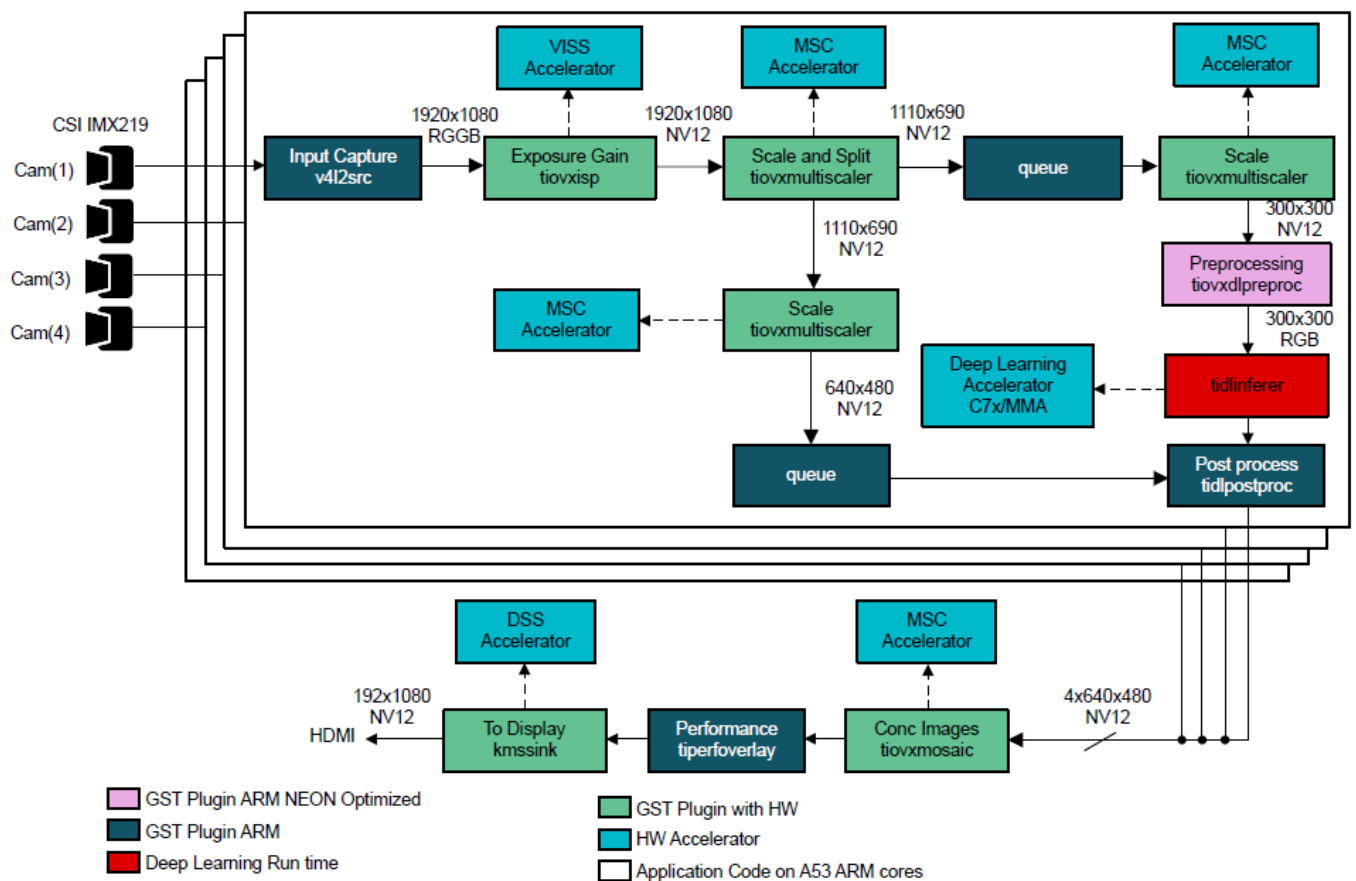


Figure 4-3. GStreamer Pipeline for Quad CSI IMX219 Camera Deep Learning Inference on AM62A

The output of the four camera pipelines is scaled down and concatenated together using the tiovxmosaic plugin. The output is displayed on a single screen. Figure 4-4 shows the output of the four cameras with a deep learning model running object detection. Each pipeline (camera) is running at 30 FPS and a total of 120 FPS.

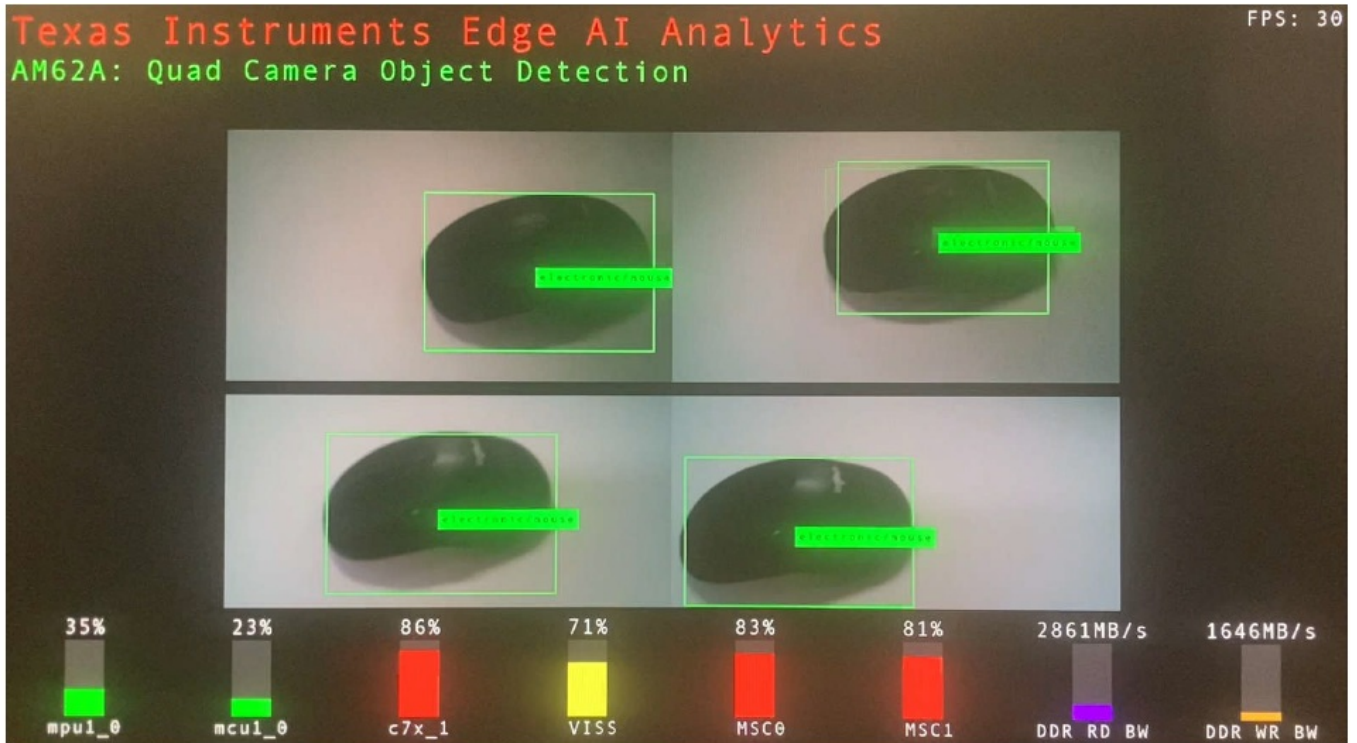


Figure 4-4. Screenshot of Quad Camera Object Detection Deep Learning Inference using AM62A with Graphical Performance Overlay

Next is the full pipeline script for the multicamera deep learning use case shown in Figure 4-3.

```
gst-launch-1.0 -v \
v4l2src device=/dev/video2 io-mode=5 ! queue leaky=2 ! video/x-bayer, width=1920, height=1080,
format=rggb ! tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/
dcc_viss.bin format-msb=7 sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a.bin sink_0::device=/dev/
v4l-subdev2 ! video/x-raw, format=Nv12 ! \
tiovxmultiscaler target=0 name=split_01 \
split_01. ! queue ! video/x-raw, width=1110, height=690 ! tiovxmultiscaler target=0 ! video/x-
raw, width=300, height=300 ! tiovxdlpreproc data-type=3 channel-order=1 tensor-format=rgb out-
pool-size=4 ! application/x-tensor-tiovx ! tidlinferer target=1 model=/opt/model_zoo/TFL-OD-2000-
ssd-mobV1-coco-mlperf-300x300 ! post_0.tensor \
split_01. ! queue ! video/x-raw, width=640, height=360 ! post_0.sink \
tidlpostproc name=post_0 model=/opt/model_zoo/TFL-OD-2000-ssd-mobV1-coco-mlperf-300x300
alpha=0.400000 viz-threshold=0.600000 top-N=5 ! queue ! mosaic_0. \
\
v4l2src device=/dev/video3 io-mode=5 ! queue leaky=2 ! video/x-bayer, width=1920, height=1080,
format=rggb ! tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/
dcc_viss.bin format-msb=7 sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a.bin sink_0::device=/dev/
v4l-subdev2 ! video/x-raw, format=Nv12 ! \
tiovxmultiscaler target=0 name=split_11 \
split_11. ! queue ! video/x-raw, width=1110, height=690 ! tiovxmultiscaler target=0 ! video/x-
raw, width=300, height=300 ! tiovxdlpreproc data-type=3 channel-order=1 tensor-format=rgb out-
pool-size=4 ! application/x-tensor-tiovx ! tidlinferer target=1 model=/opt/model_zoo/TFL-OD-2000-
ssd-mobV1-coco-mlperf-300x300 ! post_1.tensor \
split_11. ! queue ! video/x-raw, width=640, height=360 ! post_1.sink \
tidlpostproc name=post_1 model=/opt/model_zoo/TFL-OD-2000-ssd-mobV1-coco-mlperf-300x300
alpha=0.400000 viz-threshold=0.600000 top-N=5 ! queue ! mosaic_0. \
\
v4l2src device=/dev/video4 io-mode=5 ! queue leaky=2 ! video/x-bayer, width=1920, height=1080,
format=rggb ! tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/
dcc_viss.bin format-msb=7 sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a.bin sink_0::device=/dev/
v4l-subdev2 ! video/x-raw, format=Nv12 ! \
tiovxmultiscaler target=1 name=split_21 \
split_21. ! queue ! video/x-raw, width=1110, height=690 ! tiovxmultiscaler target=1 ! video/x-
raw, width=300, height=300 ! tiovxdlpreproc data-type=3 channel-order=1 tensor-format=rgb out-
pool-size=4 ! application/x-tensor-tiovx ! tidlinferer target=1 model=/opt/model_zoo/TFL-OD-2000-
ssd-mobV1-coco-mlperf-300x300 ! post_2.tensor \
split_21. ! queue ! video/x-raw, width=640, height=360 ! post_2.sink \
tidlpostproc name=post_2 model=/opt/model_zoo/TFL-OD-2000-ssd-mobV1-coco-mlperf-300x300
alpha=0.400000 viz-threshold=0.600000 top-N=5 ! queue ! mosaic_0. \
```

```
\
v4l2src device=/dev/video5 io-mode=5 ! queue leaky=2 ! video/x-bayer, width=1920, height=1080,
format=rggb ! tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/
dcc_viss.bin format-msb=7 sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a.bin sink_0::device=/dev/
v4l-subdev2 ! video/x-raw, format=NV12 ! \
tiovxmultiscaler target=1 name=split_31 \
split_31. ! queue ! video/x-raw, width=1110, height=690 ! tiovxmultiscaler target=1 ! video/x-
raw, width=300, height=300 ! tiovxdlpreproc data-type=3 channel-order=1 tensor-format=rgb out-
pool-size=4 ! application/x-tensor-tiovx ! tidlinferer target=1 model=/opt/model_zoo/TFL-OD-2000-
ssd-mobV1-coco-mlperf-300x300 ! post_3.tensor \
split_31. ! queue ! video/x-raw, width=640, height=360 ! post_3.sink \
tidlpostproc name=post_3 model=/opt/model_zoo/TFL-OD-2000-ssd-mobV1-coco-mlperf-300x300
alpha=0.400000 viz-threshold=0.600000 top-N=5 ! queue ! mosaic_0. \
\
tiovxmosaic src::pool-size=3 name=mosaic_0 \
sink_0::startx="<320>" sink_0::starty="<180>" sink_0::widths="<640>" sink_0::heights="<360>" \
sink_1::startx="<960>" sink_1::starty="<180>" sink_1::widths="<640>" sink_1::heights="<360>" \
sink_2::startx="<320>" sink_2::starty="<560>" sink_2::widths="<640>" sink_2::heights="<360>" \
sink_3::startx="<960>" sink_3::starty="<560>" sink_3::widths="<640>" sink_3::heights="<360>" \
! video/x-raw, format=NV12, width=1920, height=1080 ! queue ! tiperfoverlay title="AM62A: Quad
Camera Object Detection" ! kmssink sync=false driver-name=tidss force-modesetting=true
```

Performance Analysis

The setup with four cameras using the V3Link board and the AM62A SK was tested in various application scenarios, including directly displaying on a screen, streaming over Ethernet (four UDP channels), recording to 4 separate files, and with deep learning inference. In each experiment, we monitored the frame rate and the utilization of CPU cores to explore the whole system's capabilities.

As previously shown in Figure 4-4, the deep learning pipeline uses the tiperfoverlay GStreamer plugin to show CPU core loads as a bar graph at the bottom of the screen. By default, the graph is updated every two seconds to show the loads as a utilization percentage. In addition to the tiperfoverlay GStreamer plugin, the perf_stats tool is a second option to show core performance directly on the terminal with an option for saving to a file. This tool is more accurate compared to the tTiperfoverlay as the latter adds extra load on the ARMm cores and the DDR to draw the graph and overlay it on the screen. The perf_stats tool is mainly used to collect hardware utilization results in all of the test cases shown in this document. Some of the important processing cores and accelerators studied in these tests include the main processors (four A53 Arm cores @ 1.25GHz), the deep learning accelerator (C7x-MMA @ 850MHz), the VPAC (ISP) with VISS and multiscalers (MSC0 and MSC1), and DDR operations.

Table 5-1 shows the performance and resource utilization when using AM62A with four cameras for three use cases, including streaming four cameras to a display, streaming over Ethernet, and recording to four separate files. Two tests are implemented in each use case: with the camera only and with deep learning inference. In addition, the first

row in Table 5-1 shows hardware utilizations when only the operating system was running on AM62A without any user applications. This is used as a baseline to compare against when evaluating hardware utilizations of the other test cases. As shown in the table, the four cameras with deep learning and screen display operated at 30 FPS each ,with a total of 120 FPS for the four cameras. This high frame rate is achieved with only 86% of the deep learning accelerator (C7x-MMA) full capacity. In addition, it is important to note that the deep learning accelerator was clocked at 850MHz instead of 1000MHz in these experiments, which is about only 85% of its maximum performance.

Table 5-1. Performance (FPS) and Resource Utilization of AM62A when used with 4 IMX219 Cameras for Screen Display, Ethernet Stream, Record to Files, and Performing Deep Learning Inferencing

Appli catio n	Pipel ine (oper ation)	Outp ut	FP S a vg pip elin e s	FP S tot al	MP Us A5 3s @ 1.2 5 GH z [%]	M CU R5 [%]	DL A (C7 x- M MA) @ 85 0 MH z [%]	VI SS [%]	MS C0 [%]	MS C1 [%]	DD R Rd [M B/ s]	DD R Wr [M B/s]	DD R Tot al [MB /s]
No A pp	Base line No o perat ion	NA	NA	NA	1.8 7	1	0	0	0	0	56 0	19	57 9

Cam with Deep learning	Deep learning: Object detection MobV1 - coco and Stream over Ethernet	UDP: 4 ports 1920 ×1080	28	112	84	20	99	66	65	72	4157	2563	6720
	Deep learning: Object detection MobV1 - coco and record to files	4 files 1920 ×1080	28	112	87	22	98	75	82	61	2024	2458	6482

Summary

This application report describes how to implement multi-camera applications on the

AM6x family of devices. A reference design based on Arducam's V3Link Camera Solution Kit and AM62A SK EVM is provided in the report, with several camera applications using four IMX219 cameras, such as streaming and object detection. Users are encouraged to acquire the V3Link Camera Solution Kit from Arducam and replicate these examples. The report also provides a detailed analysis of the performance of AM62A while using four cameras under various configurations, including displaying to a screen, streaming over Ethernet, and recording to files. It also shows AM62A's capability of performing deep learning inference on four separate camera streams in parallel. If there are any questions about running these examples, submit an inquiry at the TI E2E forum.

References

1. AM62A Starter Kit EVM Quick Start Guide
2. ArduCam V3Link Camera Solution Quick Start Guide
3. Edge AI SDK documentation for AM62A
4. Edge AI Smart Cameras Using Energy-Efficient AM62A Processor
5. Camera Mirror Systems on AM62A
6. Driver and Occupancy Monitoring Systems on AM62A
7. Quad Channel Camera Application for Surround View and CMS Camera Systems
8. AM62Ax Linux Academy on Enabling CIS-2 Sensor
9. Edge AI ModelZoo
10. Edge AI Studio
11. Perf_stats tool

TI Parts Referred in This Application Note:

- <https://www.ti.com/product/AM62A7>
- <https://www.ti.com/product/AM62A7-Q1>
- <https://www.ti.com/product/AM62A3>
- <https://www.ti.com/product/AM62A3-Q1>
- <https://www.ti.com/product/AM62P>
- <https://www.ti.com/product/AM62P-Q1>
- <https://www.ti.com/product/DS90UB960-Q1>
- <https://www.ti.com/product/DS90UB953-Q1>

- <https://www.ti.com/product/TDES960>
- <https://www.ti.com/product/TSER953>

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 permits you to use these resources only for the 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](https://www.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.

IMPORTANT NOTICE

- Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
- Copyright © 2024, Texas Instruments Incorporated

Frequently Asked Questions

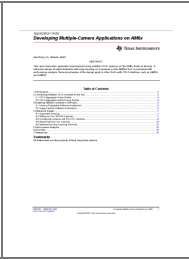
Q: Can I use any type of camera with the AM6x family of devices?

The AM6x family supports different camera types, including those with or without built-in ISP. Refer to the specifications for more details on supported camera types.

: What are the main differences between AM62A and AM62P in image processing?

The key variations include supported camera types, camera output data, presence of ISP HWA, Deep Learning HWA, and 3-D Graphics HWA. Refer to the specifications section for a detailed comparison.

Documents / Resources

	Texas Instruments AM6x Developing Multiple Camera [pdf] User Guide AM62A, AM62P, AM6x Developing Multiple Camera, AM6x, Developing Multiple Camera, Multiple Camera, Camera
---	---

References

- [User Manual](#)

Leave a comment

Your email address will not be published. Required fields are marked *

Comment *

Name

Email

Website

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

Search:

e.g. whirlpool wrf535swhz

Search

[Manuals+](#) | [Upload](#) | [Deep Search](#) | [Privacy Policy](#) | [@manuals.plus](#) | [YouTube](#)

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.