Application Note

*在 **AM6x** 上开发多摄像头应用*

Jianxhong Xu, Qutaiba Saleh

摘要

本报告介绍了在 AM6x 系列器件上使用多个 CSI-2 摄像头进行的应用开发。文中介绍了在 AM62A SoC 上使用 4 个摄像头通过深度学习进行物体检测的参考设计以及相应的性能分析。该设计的一般原理适用于具有 CSI-2 接口的其他 SoC，例如 AM62x 和 AM62P。

内容

插图清单

表格清单

商标

1 引言

嵌入式摄像头在现代视觉系统中发挥着重要作用。在一个系统中使用多个摄像头可以扩展这些系统的能力，并实现单个摄像头无法实现的功能。以下是一些使用多个嵌入式摄像头的应用示例：

安全监控：有策略地布置多个摄像头可以提供全方位的监控覆盖。这些摄像头可以实现全景视图，减少盲区，并提高物体跟踪和识别的准确性，从而改善整体安全措施。

环视：使用多个摄像头可以搭建立体视觉系统，从而提供三维信息和深度估算。对于自动驾驶车辆中的障碍物检测、机器人中的精确对象操作以及增强现实体验的逼真度等任务，此功能至关重要。

车厢记录仪和摄像头后视镜系统：具有多个摄像头的车厢记录仪可以利用单个处理器提供更全面的监控。同样，具有两个或更多摄像头的摄像头后视镜系统可以扩大驾驶员的视野，消除汽车四周的盲区。

医疗成像：医疗成像应用可以使用多个摄像头执行手术导航等任务，为外科医生提供多个视角，以提高手术精度。在内窥镜检查中，利用多个摄像头可以实现对内部器官的全面检查。

无人机和航空成像：无人机通常配备多个摄像头，以便从不同的角度拍摄高分辨率图像或视频。这在航空摄影、农业监控和土地测量等应用中非常有用。

随着微处理器的发展，可将多个摄像头集成到单个片上系统 (SoC) 中，从而提供紧凑高效的设计。AM62Ax SoC 具有高性能视频和视觉处理以及深度学习加速功能，非常适合上述用例。另一款 AM6x 器件（即 AM62P）专为高性能嵌入式 3D 显示应用而构建。AM62P 配备 3D 图形加速功能，可以轻松地将来自多个摄像头的图像拼接在一起，形成高分辨率的全景图。AM62A/AM62P SoC 的出色功能在各种出版物中都有介绍，比如 [4]、[5]、[6] 等。本应用手册不再重复介绍这些特性，而是重点介绍如何将多个 CSI-2 摄像头集成到 AM62A/AM62P 上的嵌入式视觉应用中。

表 1-1 展示了 AM62A 和 AM62P 在图像处理方面的主要差异。

表 1-1. AM62A 和 AM62P 在图像处理方面的差异

SoC	AM62A	AM62P
支持的摄像头类型	具有或不具内置 ISP	具有内置 ISP
摄像头输出数据	原始、YUV、RGB	YUV、RGB
ISP HWA	是	否
深度学习 HWA	是	否
3D 图形 HWA	否	是

2 将多个 CSI-2 摄像头连接到 SoC

如图 2-1 所示，AM6x SoC 上的摄像头子系统包含以下元件：

- MIPI D-PHY 接收器：从外部摄像头接收视频流，4 个数据通道每个通道最高支持 1.5Gbps。
- CSI-2 接收器 (RX)：从 D-PHY 接收器接收视频流，然后直接将数据流发送到 ISP 或将数据转储到 DDR 存储器。该模块支持多达 16 个虚拟通道。
- SHIM：支持通过 DMA 将捕获的数据流发送到存储器的 DMA 包装器。该包装器可以创建多个 DMA 上下文，每个上下文都对应于 CSI-2 接收器的一个虚拟通道。

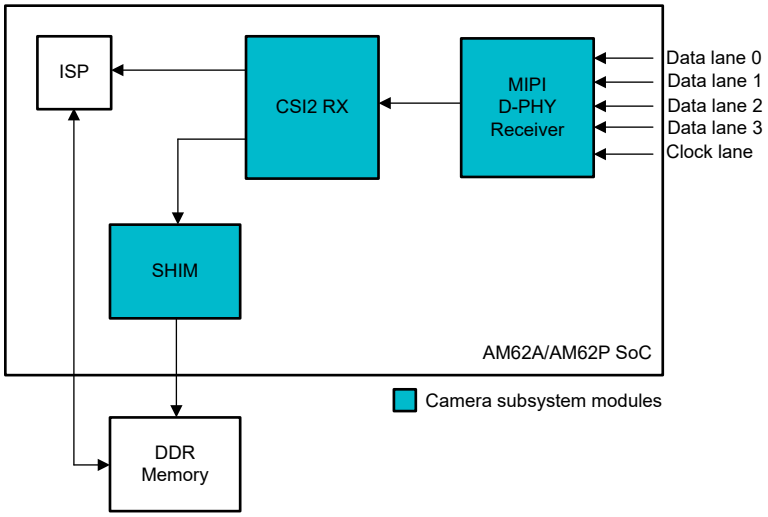

图 2-1. AM62A/AM62P SoC 上摄像头子系统的简要方框图

即使 SoC 上只有一个 CSI-2 RX 接口，也可以通过使用 CSI-2 RX 的虚拟通道在 AM6x 上支持多个摄像头（相同或不同）。需要一个外部 CSI-2 聚合元件来整合多个摄像头数据流并发送到单个 SoC。有两种类型的 CSI-2 聚合设计可供使用，具体如以下各节所述。

2.1 使用 SerDes 的 CSI-2 聚合器

要将多个摄像头流进行组合，一种方法是使用串行和解串设计。来自每个摄像头的 CSI-2 数据由串行器转换，然后通过电缆进行传输。解串器在收到通过电缆（每个摄像头一根电缆）传输的所有串行化数据后，将流转换回 CSI-2 数据，然后将交错 CSI-2 流发送到 SoC 上的单个 CSI-2 RX 接口。每个摄像头流由一个唯一的虚拟通道标识。该聚合设计提供了一个额外的优势，即允许在摄像头到 SoC 之间进行长达 15m 的长距离连接。

FPD-Link 或 V3-Link 串行器和解串器（在 AM6x Linux SDK 中受支持）是此类 CSI-2 聚合设计中的常用技术。FPD-Link 和 V3-Link 解串器都有反向通道，该通道可用于发送帧同步信号来同步所有摄像头，如 [7] 中所述。

图 2-2 展示了使用串行器和解串器将多个摄像头连接到单个 AM6x SoC 的示例。

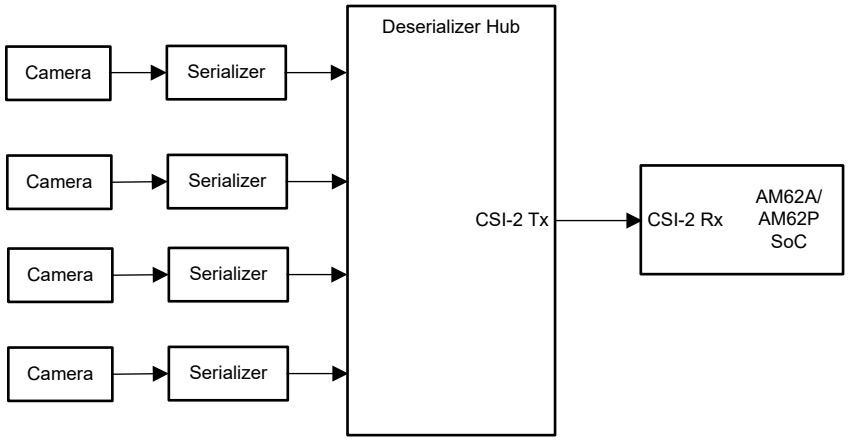

图 2-2. 使用 SerDes 连接多个摄像头

在 Arducam V3Link 摄像头解决方案套件中可以找到这种聚合设计的示例。该套件具有一个可聚合 4 个 CSI-2 摄像头流的解串器集线器，以及 4 对 V3link 串行器和 IMX219 摄像头，其中包括 FAKRA 同轴电缆和 22 引脚 FPC 电缆。后面讨论的参考设计就是在该套件上构建的。

2.2 不使用 SerDes 的 CSI-2 聚合器

此类聚合器可直接与多个 MIPI CSI-2 摄像头连接，并将来自所有摄像头的数据聚合到单个 CSI-2 输出流。

图 2-3 展示了此类系统的一个示例。此类聚合设计不使用任何串行器或解串器，但受到 CSI-2 数据传输最大距离（最远 30cm）的限制。AM6x Linux SDK 不支持此类 CSI-2 聚合器。

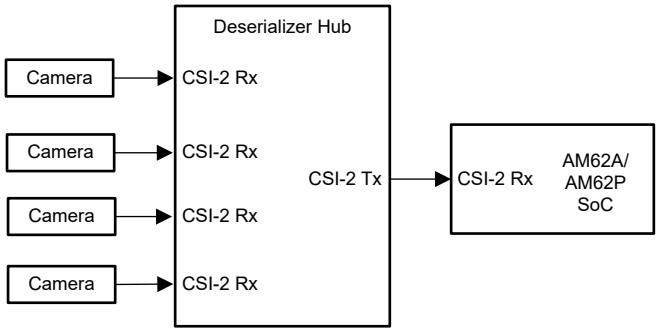

图 2-3. 使用 CSI-2 聚合器连接多个摄像头

2.3 支持的摄像头数据吞吐量

所有已连接摄像头支持的总数据吞吐量取决于 CSI-2 带宽和 ISP 带宽。

- 对于没有 ISP 的 AM6x 器件（例如 AM62P），CSI-2 带宽（每通道 1.5Gbps 或 4 通道 6Gbps）是限制因素。只要所有摄像头的总数据带宽低于 6Gbps，即受支持。
- 对于具有 ISP（如 AM62A）的 AM6x 器件，ISP 带宽是限制因素。AM62A 的 ISP 带宽为 315MPixel/s。只要所有摄像头的总数据速率低于 315MPixel/s（无论原始数据位深如何），即受支持。支持的摄像头数据总吞吐量还取决于系统负载，例如 DDR 带宽。*性能分析* 小节介绍了特定示例的 DDR 使用情况，供您参考。

3 在软件中启用多个摄像头

3.1 摄像头子系统软件架构

图 3-1 展示了 AM62A/AM62P Linux SDK 中摄像头捕捉系统软件的简要方框图，其对应于图 2-2 中的硬件系统。

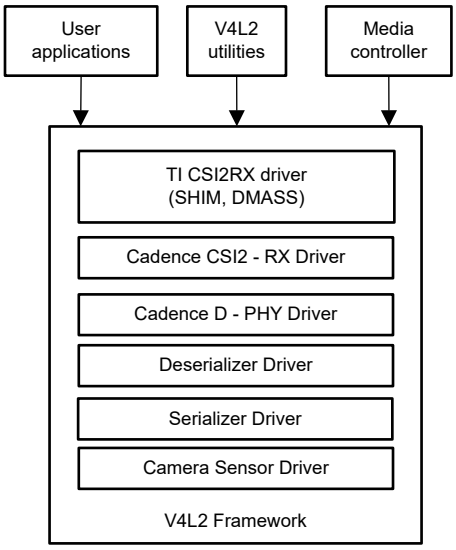

图 3-1. 使用 SerDes 的摄像头捕捉系统简要方框图

此软件架构使 SoC 能够使用串行器和解串器接收多个摄像头流，如图 2-2 所示。FPD-Link/V3-Link SerDes 为每个摄像头分配一个唯一的 I2C 地址和虚拟通道。需使用每个摄像头的唯一 I2C 地址创建唯一的器件树覆盖层。CSI-2 RX 驱动程序使用唯一的虚拟通道编号来识别每个摄像头，并为每个摄像头流创建一个 DMA 上下文。针对每个 DMA 上下文会创建一个视频节点。然后，使用 DMA 接收来自每个摄像头的数据，并将其相应地存储到存储器中。用户空间应用程序使用与每个摄像头相对应的视频节点来访问摄像头数据。第 4 章“参考设计”中提供了一些使用此软件架构的示例。

任何与 V4L2 框架兼容的特定传感器驱动程序都可以在此架构中即插即用。请参阅 [8] 来了解如何将新的传感器驱动程序集成到 Linux SDK 中。

3.2 图像流水线软件架构

AM6x Linux SDK 提供了 GStreamer (GST) 框架，该框架可用于用户空间，以便集成各种应用的图像处理组件。SoC 上的硬件加速器 (HWA)，例如视觉预处理加速器 (VPAC) 或 ISP、视频编码器和解码器以及深度学习计算引擎，可通过 GST 插件进行访问。VPAC (ISP) 具有多个模块，包括视觉成像子系统 (VISS)、镜头失真校正 (LDC) 和多标量 (MSC)，每个模块对应一个 GST 插件。

图 3-2 展示了从摄像头到 AM62A 上编码或深度学习应用的典型图像流水线方框图。有关端到端数据流的更多详细信息，请参阅 EdgeAI SDK 文档。

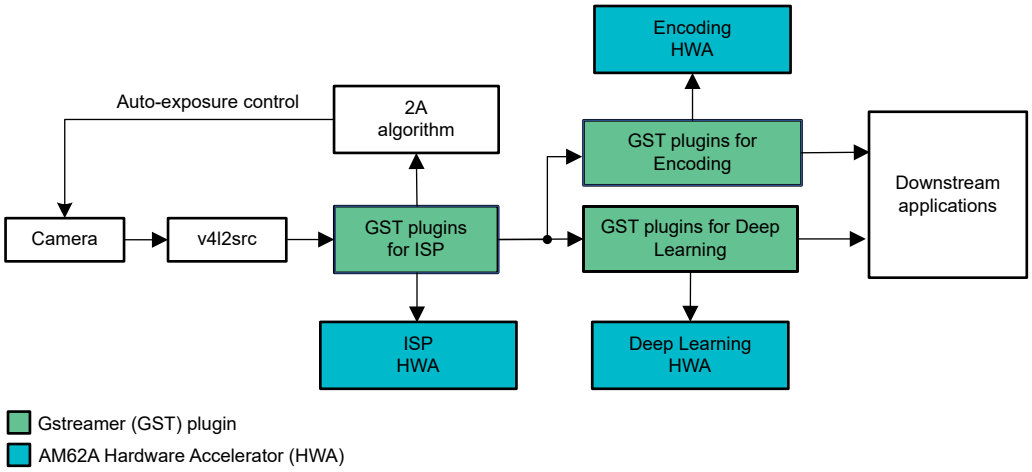

图 3-2. 使用 GStreamer 的典型 AM62A 图像流水线

对于 AM62P，图像流水线更简单，因为 AM62P 上没有 ISP。

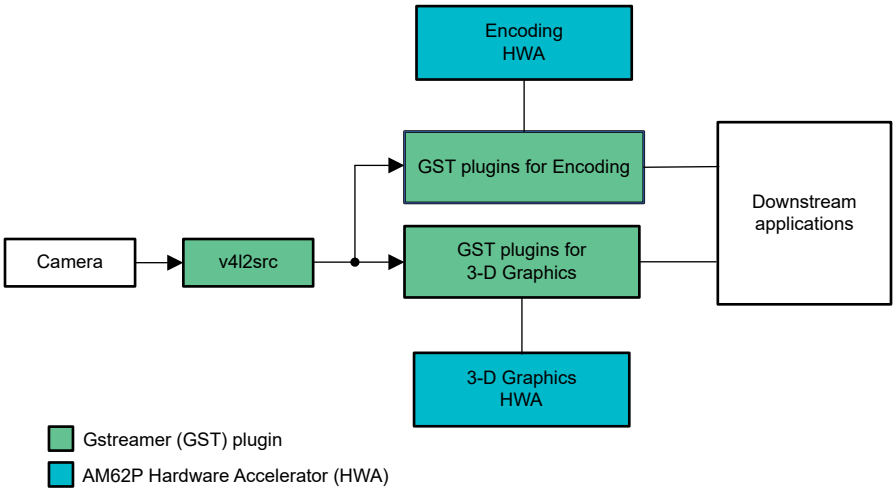

图 3-3. 使用 GStreamer 的典型 AM62P 图像流水线

借助为每个摄像头创建的视频节点，基于 GStreamer 的图像流水线允许同时处理多个摄像头输入。多个摄像头可以相同，也可以不同。对于 AM62A，ISP 会逐帧重新配置以处理来自每个已连接摄像头的图像。下一章将介绍使用 GStreamer 打造多摄像头应用的参考设计。

4 参考设计

本章介绍了在 AM62A EVM 上运行多摄像头应用的参考设计，其中使用 Arducam V3Link 摄像头解决方案套件 将 4 个 CSI-2 摄像头连接到 AM62A 并为所有 4 个摄像头运行物体检测。

4.1 支持的摄像头

Arducam V3Link 套件可与基于 FPD-Link 和 V3-Link 的摄像头以及与 Raspberry Pi 兼容的 CSI-2 摄像头配合使用。以下摄像头已经过测试：

- D3 Engineering D3RCM-IMX390-953
- Leopard Imaging LI-OV2312-FPDLINKIII-110H
- Arducam V3Link 摄像头解决方案套件中的 IMX219 摄像头
- OV5640 摄像头使用 V3Link 套件中相同的串行器

相同或不同的摄像头可通过 V3Link 板连接并同时运行。例如，以下组合已经过测试：

- 4 个 IMX219（在 AM62A 上）
- 4 个 OV5640（在 AM62A 和 AM62P 上）
- 2 个 IMX390 + 1 个 OV2312（在 AM62A 上）
- 2 个 IMX390（在 AM62A 上）
- 2 个 OV2312（在 AM62A 上）

本报告仅将第一种组合作为实施参考。

4.2 设置四个 IMX219 摄像头

按照 AM62A 入门套件 EVM 快速入门指南 中提供的说明来设置 SK-AM62A-LP EVM (AM62A SK) 并按照 ArduCam V3Link 摄像头解决方案快速入门指南 来通过 V3Link 套件将摄像头连接到 AM62A SK。确保柔性电缆、摄像头、V3Link 板和 AM62A SK 上的引脚均正确对齐。

图 4-1 展示了用于本报告中参考设计的设置。该设置中的主要组件包括：

- 1 个 SK-AM62A-LP EVM 电路板
- 1 个 Arducam V3Link 双通道适配器板
- 将 Arducam V3Link 连接到 SK-AM62A 的 FPC 电缆
- 4 个 V3Link 摄像头适配器（串行器）
- 4 根射频同轴电缆，用于将 V3Link 串行器连接到 V3Link 双通道套件
- 4 个 IMX219 摄像头
- 4 根 CSI-2 22 引脚电缆，用于将摄像头连接到串行器
- 电缆：HDMI™ 电缆、用于为 SK-AM62A-LP 供电的 USB-C® 以及用于 V3Link 双通道套件的 12V 电源
- 图 4-1 中未显示的其他组件：micro-SD 卡、用于访问 SK-AM62A-LP 的 micro-USB 电缆以及用于流式传输的 Ethernet®

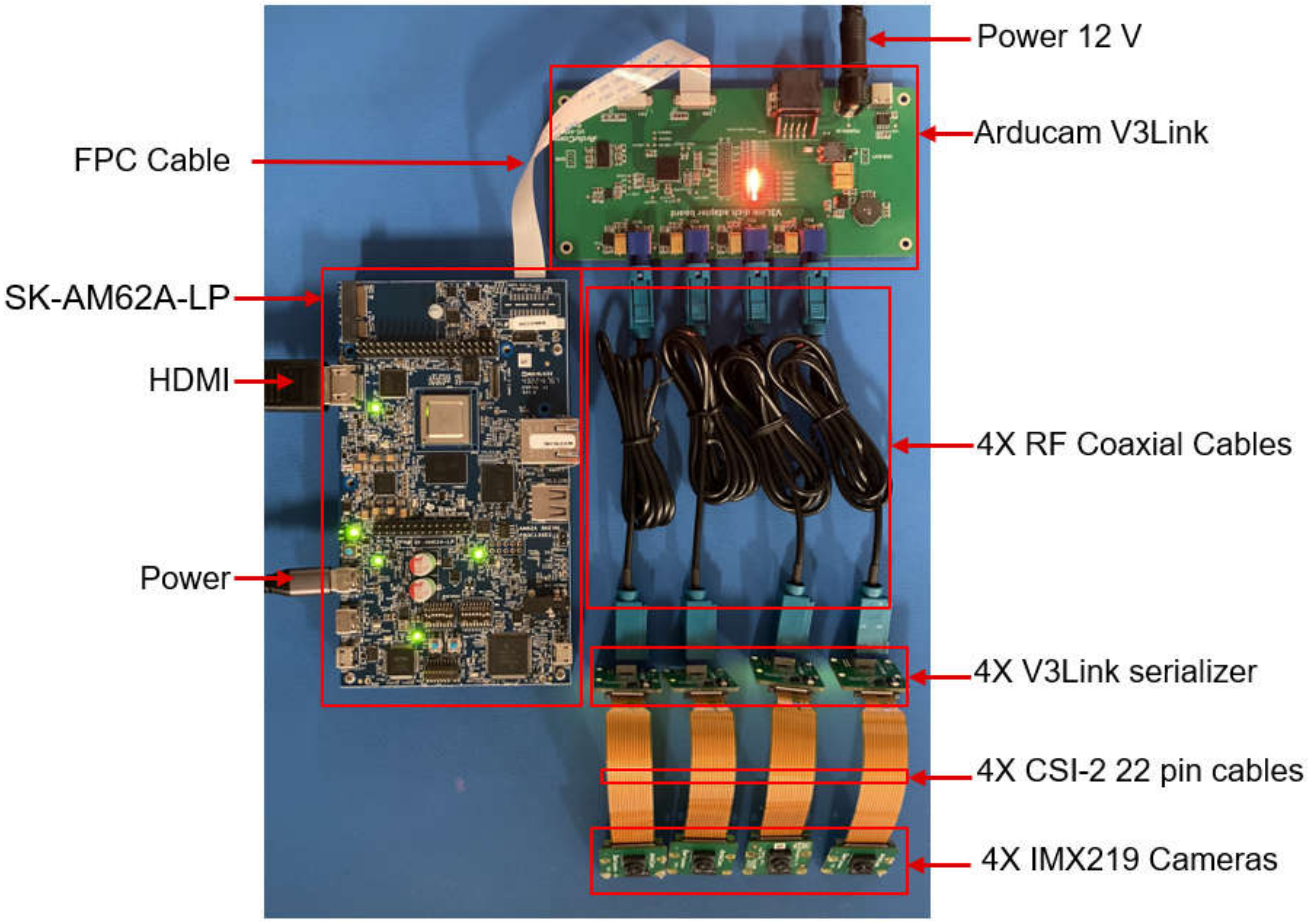

图 4-1. 参考设计中使用的 V3link 板和 4 摄像头设置

4.3 配置摄像头和 CSI-2 RX 接口

适用于 AM62A 的 Processor SDK Linux 支持 V3Link 板。要启用 4 个 IMX219 摄像头，请在 /run/media/BOOT-mmcblk1p1/uEnv.txt 中使用以下覆盖层 dtbo 文件：

```
k3-am62x-sk-csi2-v3link-fusion.dtbo
k3-v3link-imx219-0-0.dtbo
k3-v3link-imx219-0-1.dtbo
k3-v3link-imx219-0-2.dtbo
k3-v3link-imx219-0-3.dtbo
```

要配置摄像头和 CSI-2 Rx 接口，请运行脚本 /opt/edgeai-gst-apps/scripts/setup_cameras_v3link.sh。此脚本的运行结果如下所示：

```
root@am62axx-evm:/opt/edgeai-gst-apps# ./scripts/setup_cameras_v3link.sh
IMX219 Camera 0 detected
    device = /dev/video-imx219-cam0
    name = imx219
    format = [fmt:SRGGB8_1X8/1920x1080 field: none]
    subdev_id = /dev/v4l-imx219-subdev0
    isp_required = yes
    ldc_required = yes
IMX219 Camera 1 detected
    device = /dev/video-imx219-cam1
    name = imx219
    format = [fmt:SRGGB8_1X8/1920x1080 field: none]
    subdev_id = /dev/v4l-imx219-subdev1
    isp_required = yes
    ldc_required = yes
IMX219 Camera 2 detected
```

```
    device = /dev/video-imx219-cam2
    name = imx219
    format = [fmt:SRGGB8_1X8/1920x1080 field: none]
    subdev_id = /dev/v4l-imx219-subdev2
    isp_required = yes
    ldc_required = yes
IMX219 Camera 3 detected
    device = /dev/video-imx219-cam3
    name = imx219
    format = [fmt:SRGGB8_1X8/1920x1080 field: none]
    subdev_id = /dev/v4l-imx219-subdev3
    isp_required = yes
    ldc_required = yes
```

运行此脚本后，摄像头格式、CSI-2 RX 接口格式以及从每个摄像头到相应视频节点的路由都会得到正确配置。总共创建了 6 个视频节点。每个视频节点对应于 CSI2 RX 驱动器分配的一个 DMA 上下文。在 6 个视频节点中，4 个节点用于 4 个 IMX219 摄像头，如下面的媒体管线拓扑所示：

```
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]
```

如上所示，媒体实体 30102000.ticsi2rx 有 6 个 source pad，但仅使用前 4 个 source pad，每个 source pad 对应于一个 IMX219。媒体管线拓扑也可以用图形方式表示。运行以下命令以生成 dot 文件：

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

然后在 Linux 主机 PC 上运行以下命令以生成 png 文件：

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

图 4-2 是使用上述命令生成的图片。图 3-1 软件架构中的组件可在该图中找到。

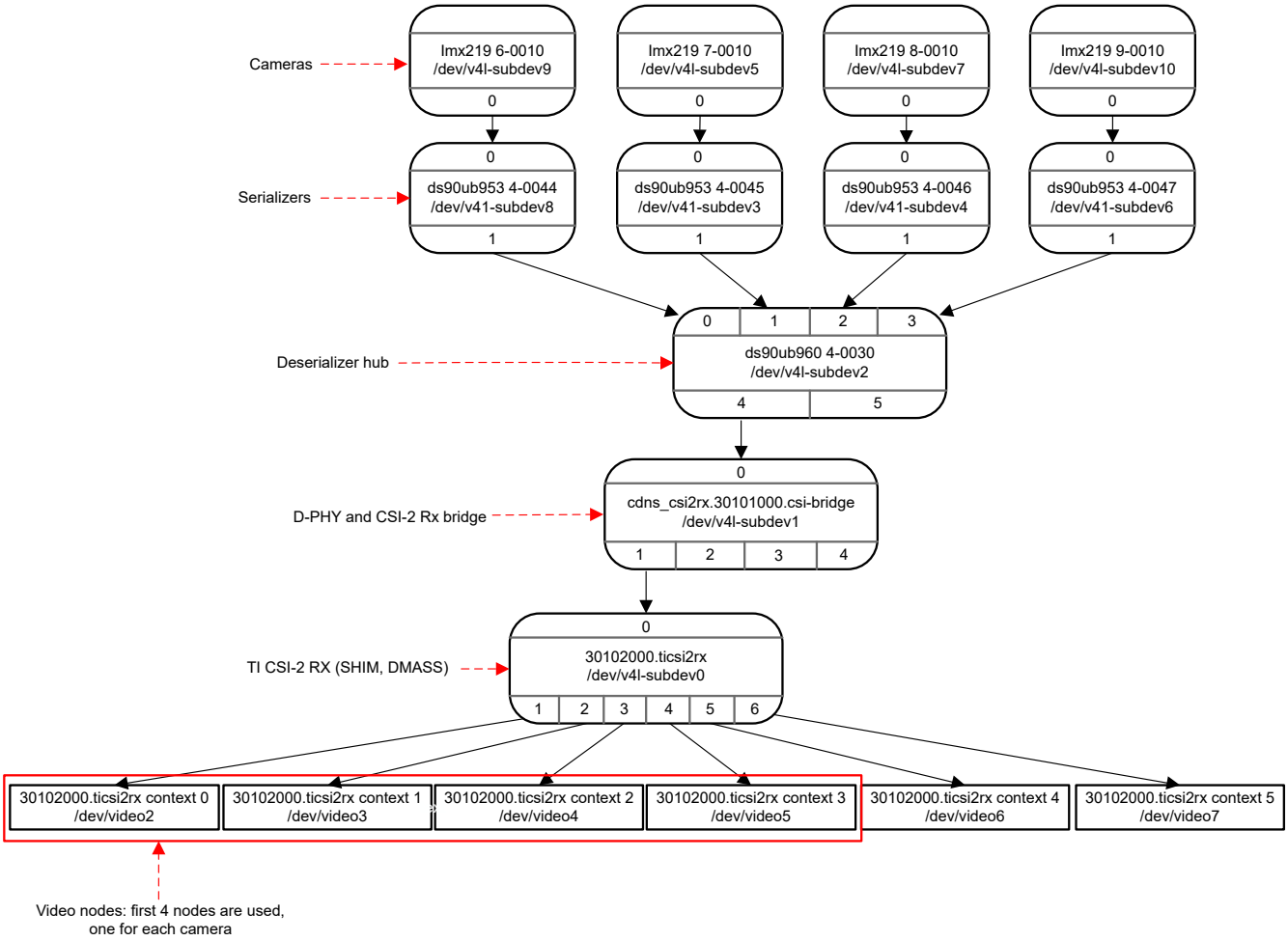

图 4-2. 多摄像头系统的媒体拓扑

4.4 从四个摄像头进行流式传输

正确设置硬件和软件后，可从用户空间运行多摄像头应用。对于 AM62A，必须调整 ISP 才能获得良好的图像质量。如需了解如何执行 ISP 调优，请参阅 AM6xA ISP 调优指南。以下各节介绍了将摄像头数据流式传输到显示器、将摄像头数据流式传输到网络以及将摄像头数据存储到文件的示例。

4.4.1 将摄像头数据流化传输到显示器

该多摄像头系统的一个基本应用是将所有摄像头的视频流式传输到连接到同一 SoC 的显示器。以下是将四个 IMX219 数据流式传输到显示器的 GStreamer 流水线示例。

```
gst-launch-1.0 \
v4l2src device=/dev/video-imx219-cam0 io-mode=5 ! queue max-size-buffers=1 leaky=2 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! \
tiovxisp sink_0::device=/dev/v4l-imx219-subdev0 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-
file=/opt/imaging/imx219/linear/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a_1920x1080.bin format-msb=7 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_0 \
v4l2src device=/dev/video-imx219-cam1 io-mode=5 ! queue max-size-buffers=1 leaky=2 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! \
tiovxisp sink_0::device=/dev/v4l-imx219-subdev1 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-
file=/opt/imaging/imx219/linear/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a_1920x1080.bin format-msb=7 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_1 \
v4l2src device=/dev/video-imx219-cam2 io-mode=5 ! queue max-size-buffers=1 leaky=2 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! \
tiovxisp sink_0::device=/dev/v4l-imx219-subdev2 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-
file=/opt/imaging/imx219/linear/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a_1920x1080.bin format-msb=7 ! \
video/x-raw,format=NV12, width=1920,height=1080 ! queue ! mosaic.sink_2 \
v4l2src device=/dev/video-imx219-cam3 io-mode=5 ! queue max-size-buffers=1 leaky=2 ! video/x-
bayer,width=1920,height=1080,framerate=30/1,format=bggr ! \
tiovxisp sink_0::device=/dev/v4l-imx219-subdev3 sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-
file=/opt/imaging/imx219/linear/dcc_viss_1920x1080.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a_1920x1080.bin format-msb=7 ! \
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 ! queue ! kmssink driver-name=tidss sync=false force-
modesetting=true
```

4.4.2 通过以太网流式传输摄像头数据

摄像头数据还可以通过以太网流式传输，而不是流式传输到连接到同一 SoC 的显示器。接收端可以是另一个 AM62A/AM62P 处理器，也可以是主机 PC。以下是通过以太网流式传输摄像头数据的示例（为简单起见，示例中使用了两个摄像头）（请注意流水线中使用的编码器插件）：

```
gst-launch-1.0 \
v4l2src device=/dev/video-imx219-cam0 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-imx219-
subdev0 ! 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/video-imx219-cam1 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-imx219-
subdev1 ! queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc ! rtph264pay ! udpsink
port=5001 host=<receiving IP address>
```

以下是接收摄像头数据并流式传输到另一 AM62A/AM62P 处理器上的显示器的示例：

```
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 \
tiovxmosaic 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 ! kmssink driver-name=tidss sync=false
```

4.4.3 将摄像头数据存储到文件

摄像头数据可以存储在本地文件中，而不是流式传输到显示器或网络。下面的流水线会将每个摄像头的数据存储到一个文件中（为简单起见，这里以两个摄像头为例）。

```
gst-launch-1.0 \
v4l2src device=/dev/video-imx219-cam0 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-imx219-
subdev0 ! queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc ! filesink location=cam-
cap-1.mp4 \
\
v4l2src device=/dev/video-imx219-cam1 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-imx219-
subdev1 ! queue ! \
video/x-raw,format=NV12,width=1920,height=1080,framerate=30/1 ! v4l2h264enc !  filesink
location=cam-cap-2.mp4
```

4.5 多摄像头深度学习推理

AM62A 配备了一个高达 2TOPS 的深度学习加速器 (C7x-MMA)，能够运行各种类型的深度学习模型，以进行分类、物体检测或语义分割等。本节介绍了 AM62A 如何在四个不同的摄像头源上同时运行四个深度学习模型。

4.5.1 模型选择

TI 的 EdgeAI-ModelZoo 提供了数百个先进的模型，这些模型从原始训练框架转换或导出为嵌入式适用格式，因此可卸载到 C7x-MMA 深度学习加速器。基于云的 Edge AI Studio 提供了一款易于使用的*模型选择*工具。它会进行动态更新以包含 TI EdgeAI-ModelZoo 支持的所有模型。该工具无需任何经验，并提供了一个易于使用的界面，用于输入期望模型中所需的特性。

为这个多摄像头深度学习实验选择了 TFL-OD-2000-ssd-mobV1-coco-mlperf。该多物体检测模型是在 Tensor Flow 框架中开发的，输入分辨率为 300x300。表 4-1 展示了在含有大约 80 个不同类的 COCO 数据集上进行训练时该模型的重要功能。

表 4-1. TFL-OD-2000-ssd-mobV1-coco-mlperf 模型的重要特性。

型号	任务	分辨率	FPS	mAP 在 COCO 上的精度达 50%	延迟/帧 (ms)	DDR 带宽利用率（MB/帧）
TFL-OD-2000-ssd-mobV1-coco-mlperf	多物体检测	300x300	约 152	15.9	6.5	18.839

4.5.2 流水线设置

图 4-3 展示了 4 摄像头深度学习 GStreamer 流水线。TI 提供了一套 GStreamer 插件，可将部分媒体处理和深度学习推理负载分流到硬件加速器。这些插件的一些示例包括 tiovxisp、tiovxmultiscaler、tiovxmosaic 和 tidlinferer。图 4-3 中的流水线包括多路径 GStreamer 流水线的所有必需插件，用于 4 个摄像头输入，每个都带有媒体预处理、深度学习推理和后处理功能。每个摄像头路径的重复插件均在图形中堆叠在一起，以便于演示。

可用的硬件资源均匀地分布在四个摄像头路径中。例如，AM62A 包含两个图像多标量：MSC0 和 MSC1。该流水线明确指定了 MSC0 来处理摄像头 1 和摄像头 2 路径，而 MSC1 专用于摄像头 3 和摄像头 4。

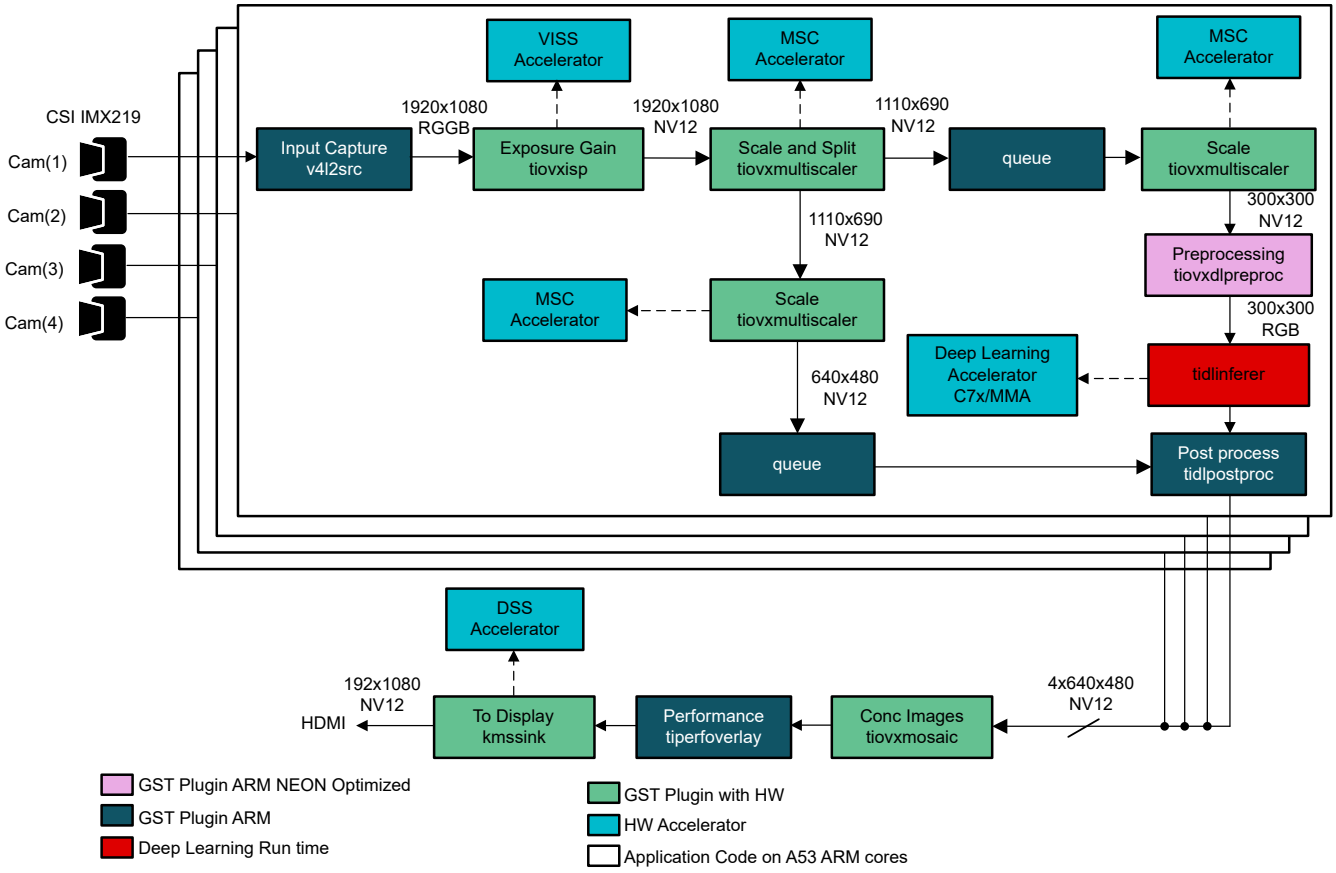

图 4-3. AM62A 上四路 CSI IMX219 摄像头深度学习推理的 GStreamer 流水线

四个摄像头流水线的输出通过 tiovxmosaic 插件缩小并连接在一起。输出显示在单个屏幕上。图 4-4 展示了深度学习模型运行对象检测时四个摄像头的输出。每个流水线（摄像头）以 30FPS 和总计 120FPS 的速度运行。点击此链接可查看展示此演示的录屏视频。

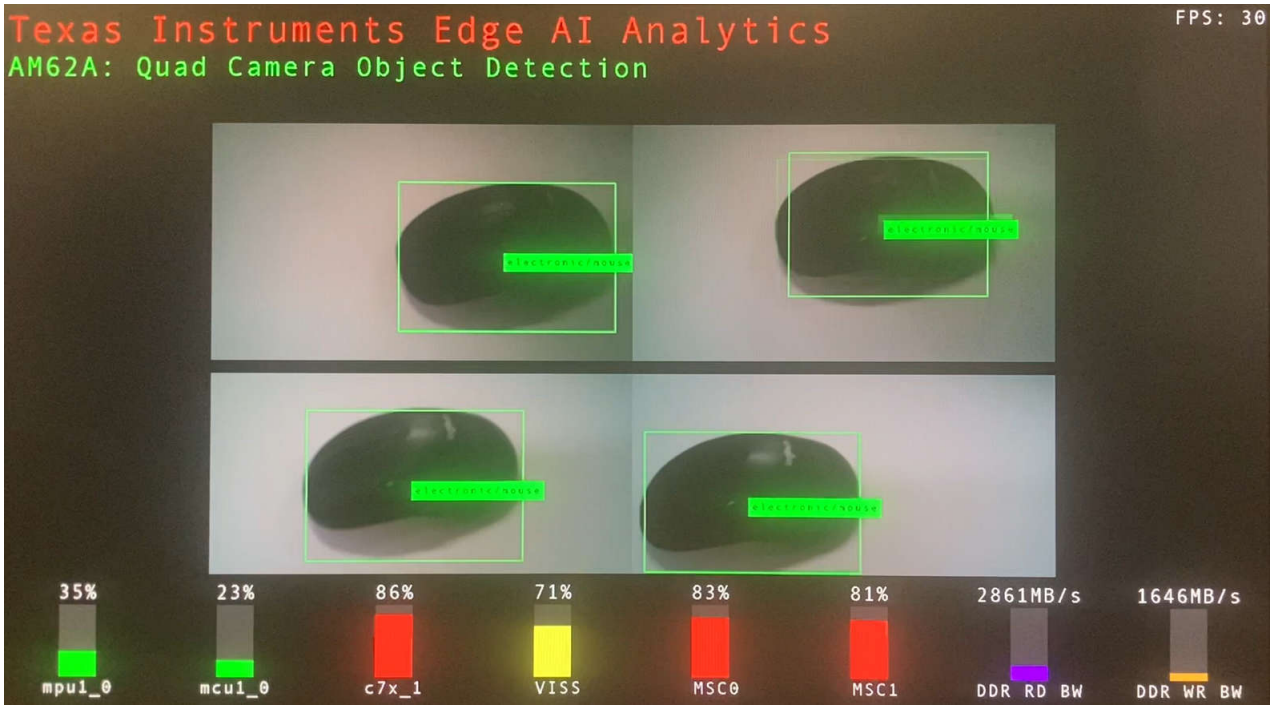

图 4-4. 使用 AM62A 且具有图形性能叠层的四摄像头物体检测深度学习推理屏幕截图

接下来是多摄像头深度学习用例的完整流水线脚本，如图 4-3 所示。

```
gst-launch-1.0 -v \
v4l2src device=/dev/video-imx219-cam0 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-imx219-subdev0 ! 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/video-imx219-cam1 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-imx219-subdev1 ! 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/video-imx219-cam2 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-imx219-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/video-imx219-cam3 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-imx219-subdev3 ! 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
```

5 性能分析

我们在各种应用场景中对使用 V3Link 板和 AM62A SK 的四摄像头设置进行了测试，包括直接在屏幕上显示、以太网流式传输（四个 UDP 通道）、录制到 4 个单独的文件以及使用深度学习推理。在每个实验中，我们都会监控帧速率和 CPU 内核的利用率，从而探索整个系统的功能。

如前面的图 4-4 所示，深度学习流水线使用 tiperfoverlay GStreamer 插件在屏幕底部以条形图的形式显示 CPU 内核负载。默认情况下，该图每两秒更新一次，以利用率百分比形式显示负载。除了 tiperfoverlay GStreamer 插件之外，第二个可直接在终端上显示内核性能的选项是 perf_stats 工具（具有文件保存选项）。此工具相比 tiperfoverlay 更精确，tiperfoverlay 会给 Arm 内核和 DDR 带来额外负载，以绘制图形并在屏幕上叠加。perf_stats 工具主要用于收集本文档中所述所有测试用例中的硬件利用率结果。这些测试中研究的一些重要处理内核和加速器包括主处理器（四个 A53 Arm 内核，频率为 1.25GHz）、深度学习加速器 (C7x-MMA，频率为 850MHz)、具有 VISS 和多标量（MSC0 和 MSC1）的 VPAC (ISP) 以及 DDR 操作。

表 5-1 展示了在三种用例中将 AM62A 与四个摄像头一起使用时的性能和资源利用率，包括将四个摄像头的视频流式传输到显示器、通过以太网进行流式传输以及录制到四个不同的文件。每个用例中都会执行两项测试：仅使用摄像头和使用深度学习推理。此外，表 5-1 中的第一行显示了只有操作系统在 AM62A 上运行而没有任何用户应用程序时的硬件利用率。在评估其他测试用例的硬件利用率时，该值用作比较的基准。如表所示，具有深度学习和屏幕显示功能的四个摄像头以 30FPS 的速率运行，四个摄像头的总速率为 120FPS。这种高帧速率只需深度学习加速器 (C7x-MMA) 满容量的 86% 即可实现。此外，需要注意的是，在这些实验中，深度学习加速器的时钟频率为 850MHz，而不是 1000MHz，后者仅为最高性能的 85% 左右。

表 5-1. 与 4 个 IMX219 摄像头配合使用来实现屏幕显示、以太网流式传输、录制到文件和执行深度学习推理时 AM62A 的性能 (FPS) 和资源利用率

应用	流水线（操作）	输出	FPS 平均流水线	FPS 总计	MPU A53（1.25 GHz 时）[%]	MCU R5 [%]	DLA (C7x-MMA)（850MHz 时）[%]	VISS [%]	MSC0 [%]	MSC1 [%]	DDR 读取 [MB/s]	DDR 写入 [MB/s]	DDR 总计 [MB/s]
无应用	**基线无操作**	不适用	不适用	不适用	1.87	1	0	0	0	0	560	19	579
仅限摄像头	**流式传输到屏幕**	屏幕	30	120	12	12	0	70	61	60	1015	757	1782
	以太网流式传输	UDP：4 个端口 1920x1080	30	120	23	6	0	70	0	0	2071	1390	3461
	录制到文件	4 个文件 1920x1080	30	120	25	3	0	70	0	0	2100	1403	3503
具有深度学习功能的摄像头	**深度学习：对象检测 MobV1-coco**	屏幕	30	120	38	25	86	71	85	82	2926	1676	4602
	深度学习：对象检测 MobV1-coco 和以太网流式传输	UDP：4 个端口 1920x1080	28	112	84	20	99	66	65	72	4157	2563	6720
	深度学习：对象检测 MobV1-coco 和录制到文件	4 个文件 1920x1080	28	112	87	22	98	75	82	61	2024	2458	6482

6 结语

本应用手册介绍了如何在 AM6x 系列器件上实现多摄像头应用。本文提供了基于 Arducam V3Link 摄像头解决方案套件和 AM62A SK EVM 的参考设计，其中包含多个使用四个 IMX219 摄像头的摄像头应用，例如流式传输和物体检测。建议用户从 Arducam 获取 V3Link 摄像头解决方案套件并复制这些示例。该报告还提供了在各种配置下使用四个摄像头时 AM62A 性能的详细分析，包括显示在屏幕上、以太网流式传输和录制到文件。此外，还展示了 AM62A 对四个单独的摄像头流并行执行深度学习推理的能力。如果您对运行这些示例有任何疑问，请在 TI E2E 论坛上提交咨询。

7 参考资料

1. 德州仪器 (TI)，AM62A 入门套件 EVM 快速入门指南
2. ArduCam V3Link 摄像头解决方案快速入门指南
3. 适用于 AM62A 的边缘 AI SDK 文档
4. 德州仪器 (TI)，使用高能效 AM62A 处理器的边缘 AI 智能摄像头技术白皮书
5. 德州仪器 (TI)，AM62A 上的摄像头后视镜系统技术白皮书
6. 德州仪器 (TI)，AM62A 上的驾驶员和乘客监控系统技术白皮书
7. 德州仪器 (TI)，用于环视和 CMS 摄像头系统的四通道摄像头应用应用报告
8. 德州仪器 (TI)，有关启用 CIS-2 传感器的 AM62Ax Linux Academy
9. 德州仪器 (TI)，Edge AI ModelZoo 网站
10. 德州仪器 (TI)，Edge AI Studio 网站
11. 德州仪器 (TI)，Perf_stats 工具网站

本应用手册中提到的 TI 器件：

- AM62A7
- AM62A7-Q1
- AM62A3
- AM62A3-Q1
- AM62P
- AM62P-Q1
- DS90UB960-Q1
- DS90UB953-Q1
- TDES960
- TSER953

8 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (February 2024) to Revision A (November 2024)	Page
• 添加了*支持的摄像头数据吞吐量*小节以介绍支持的摄像头数据吞吐量	4
• 向*支持的摄像头*小节中添加经过测试的多个摄像头组合列表	7
• 更新了*配置摄像头和 CSI-2 RX 接口*小节中的软件设置	8
• 更新了*从四个摄像头进行流式传输*小节中“多摄像头深度学习推理”的 GStreamer 流水线	10
• 更新了*多摄像头深度学习推理*小节中的 GStreamer 流水线	12

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265