WHITE PAPER

Kedar Chitnis,
SW Architect, ADAS Systems

Roman Staszewski,
SW Architect, ADAS Systems

Gaurav Agarwal,
ADAS Marketing Manager

Texas Instruments

TI Vision SDK, Optimized Vision Libraries for ADAS Systems

Introduction

There were 1.2 million global traffic deaths in 2010[1]. 93 percent of traffic accidents in the US are due to human error, typically due to inattention[2]. ADAS (Advanced Driver Assistance Systems) applications such as Lane Departure Warning, Forward Collision Warning, Pedestrian Detection, Traffic Sign Recognition, High Beam Assist, to name a few, can help avoid these tragedies. These applications are part of technologies such as Front Camera, Park Assist (Surround View/Rear Camera) and Fusion.

Cars with some of these ADAS functionalities are currently available in the market and several OEMs (Audi, BMW, Toyota and Nissan)[3] have already announced advanced ADAS functionalities and driverless car programs, the future of ADAS technology.

ADAS systems use a range of sensors (RGB, Radar, Ultrasonic, see Figure 1) to capture information about the surroundings and then process this information to implement the ADAS functionalities. Vision sensor-based (e.g., RGB sensor) processing also known as vision analytics processing forms an integral part of ADAS systems and provide the "eyes" to these systems. Advanced System-on-Chips (SoCs)/Application Processors (AP) and embedded Software (SW) is required to build these ADAS systems and functionalities.

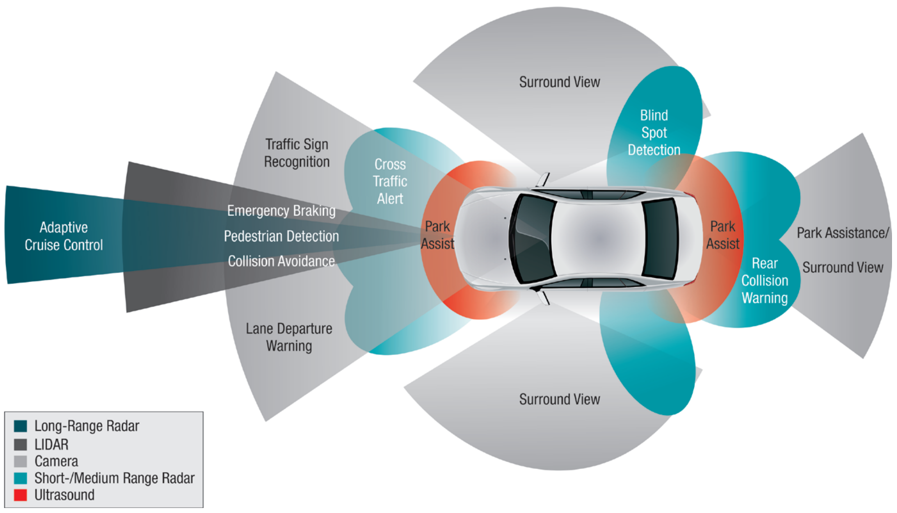


Figure 1: Example of ADAS vision system

Texas Instruments (TI) has been investing in this space for several years and has recently introduced a scalable new product family, the TDA2x[4], offers unprecedented performance at a low-power footprint and enables processing for Front Camera, Park Assist (Surround View/ Rear Camera) and Fusion/Radar applications on a single **System-on-Chip (SoC)**. This SoC enables easier system integration, reduced time to market and reduced cost. A new programmable hardware accelerator architecture, the Vision AccelerationPac, with multiple EVEs (Embedded Vision Engines) was also introduced which offers more than 8× the performance than existing ADAS systems at the same power levels[5].

With its heavy reliance on cameras and other imaging sensors, assisted or autonomous driving requires a great deal of high-performance vision processing, which, by nature, is heterogeneous. (Figure 2).

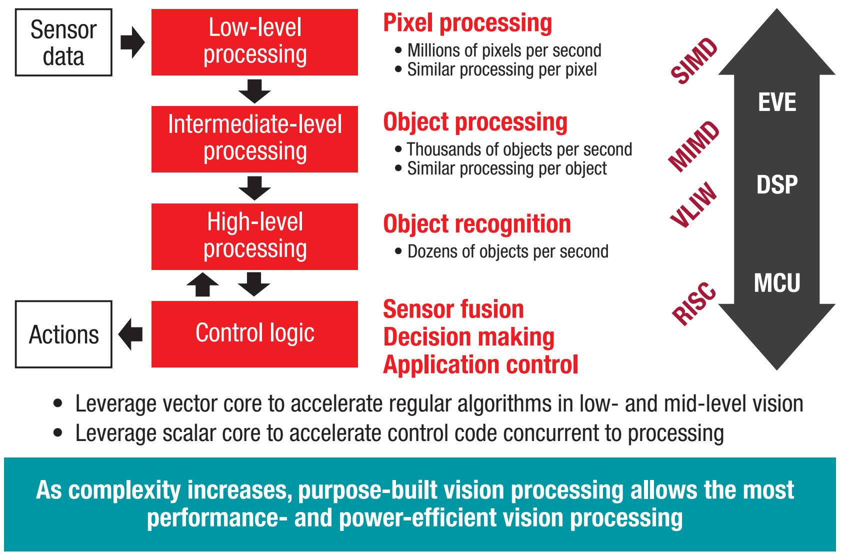

Sensor data
Low-level processing
Intermediate-level processing
High-level processing
Control logic
Actions
Pixel processing
• Millions of pixels per second
• Similar processing per pixel
Object processing
• Thousands of objects per second
• Similar processing per object
Object recognition
• Dozens of objects per second
Sensor fusion
Decision making
Application control
SIMD
MIMD
VLIW
RISC
EVE
DSP
MCU
• Leverage vector core to accelerate regular algorithms in low- and mid-level vision
• Leverage scalar core to accelerate control code concurrent to processing
As complexity increases, purpose-built vision processing allows the most performance- and power-efficient vision processing

Figure 2: Heterogeneous SoC concept for highest processing performance and power efficiency

Low-level processing concentrates on pixel data from an image sensor to create useful images for further processing. For example, low-level processing provides a video stream of detected edges and gradients on the road. Low-level processing tends to use relatively simple, repetitive algorithms that operate in parallel to handle massive amounts of input data. For such tasks, an SIMD (Single Instruction, Multiple Data) processor architecture is the best choice. TI's recently announced, purpose-built programmable vector accelerator, EVE, excels in SIMD processing.

Intermediate-level processing identifies sections of images where important objects could be located, such as shapes resembling pedestrians. The amount of data to process is much less than for low-level processing, but algorithms are more complex. This processing is efficiently addressed by SIMD or MIMD processor architectures and is well suited for EVE or Digital Signal Processors (DSPs).

High-level processing uses the resulting information from intermediate-level processing to recognize what types of objects these are – whether they are other vehicles, people, animals, or traffic signs, for instance. High-level processing has comparatively little data but very complicated algorithms and benefits from VLIW (Very Long Instruction Word) or RISC (Reduced Instruction Set Computing) processor architectures. DSPs and ARM® processors are well suited for such processing.

Finally, the microcontroller decides what the system should do – whether to stop, go, or wait until a pedestrian crosses the road, the light changes, or a nearby car passes. Such control logic is best implemented on a RISC processor with safety features such as ARM.

Software in ADAS systems

Software is an integral part of and is critical in building ADAS systems that meets overall embedded product requirements:

- Leading-edge processing performance within tight thermal budget limits
- Realtime response
- High reliability and functional safety requirements
- Small code size footprint
- Low cost
- Scalability and portability across platforms
- Inter-operability between components
- Ease of use (push towards open standards)
- And increasingly, security against hacking and reverse-engineering

In addition, the heterogeneous processor architecture, optimized for extreme performance at very low power has been a known hard programming challenge.

In addition, there are multiple levels of software ownership within an ADAS system:

- Semiconductor vendors supplying vision SoCs
- Various third parties supplying algorithms and SW subcomponents (aka Tier 2s)
- ADAS Electronic Control Unit (ECU) integrators and suppliers, delivering major component: Hardware (HW) + SW (aka Tier 1s)
- Auto-makers (aka OEMs)

While the Tier 1s (sometimes along with OEMs) are responsible for the overall ADAS ECU module, semi-conductor vendors play a key role in defining and implementing low-level SW architecture that allows to best leverage the underlying increasingly more complex SoC hardware. This paper focuses on the low-level SW architecture challenges and solutions provided by TI, as an ADAS SoC vendor.

ADAS SW hierarchy is composed of the following SW components (as shown in Figure 3):

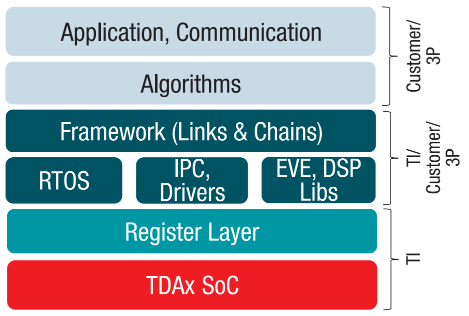


Figure 3: ADAS SW hierarchy

- Device drivers which provide low-level abstraction of the SoC hardware resources
- RTOS (Real-Time Operating System) providing OS services (e.g., memory management, SW thread support) with very low switching latency overheads required for real-time system response. Many of the RTOS used in the ADAS systems are proprietary, but there are several commercial ones, such as GHS's Integrity®, QNX® Neutrino® RTOS, Wind River's VxWorks®, or SYSGO's PikeOS among others which are certified for safety-critical applications.
- Middle-ware providing additional various higher-level services above low-level drivers (e.g., networking stacks, video codecs)
- Frameworks for easily implementing application data-flow management and performing computations
- Vision acceleration libraries which speed up implementation of vision algorithms by providing computational building blocks (i.e., vision kernels) that are highly optimized for hardware accelerators or specialized processors
- In-car networking with other car ECU modules, typically compliant with AUTOSAR, but other protocols also exist.

TI software offerings

TI's SW offering includes the Vision Software Development Kit (TI Vision SDK), DSP and EVE vision libraries and sample applications.

TI Vision SDK and sample applications enable:

- Rapid prototyping of algorithms on DSP/EVE and creation of data flows in full system context
- Consistent APIs for creating new and customizable system data flows
- Optimization and instrumentation for power, latency, performance, load
- Reference software framework and components for use in production systems software framework and components in production systems
- Little or no change in underlying baseline software components being delivered by TI.

EVE and DSP libraries offer optimized production-worthy low-level and mid-level functionalities which can be used in the application algorithm development and implementation.

TI Vision SDK

The TI Vision SDK is a multi-processor software development platform for TI's family of ADAS SoCs. The software framework allows users to create different ADAS application data flows involving video capture, video pre-processing, video analytics algorithms and video display. The SDK has sample ADAS data flows which exercise different CPUs and HW accelerators in the ADAS SoC and show customers how to effectively use different SoC sub-systems. The Vision SDK will be based on a framework named as the "Links and Chains" framework and the user API to this framework is called "Link API". The SDK installer package includes all

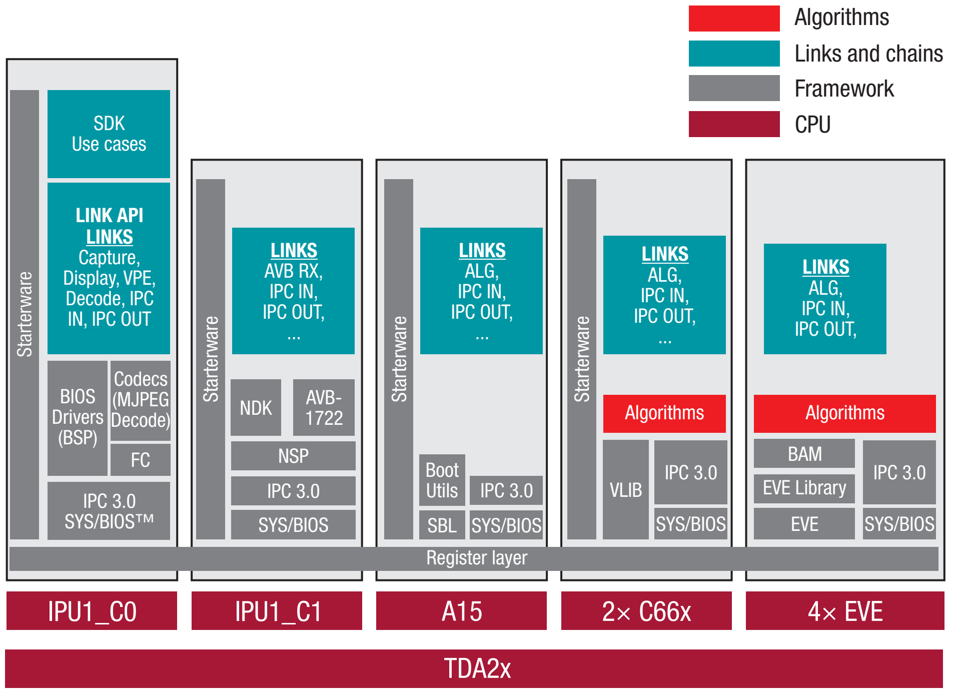


Figure 4: TI Vision SDK software stack

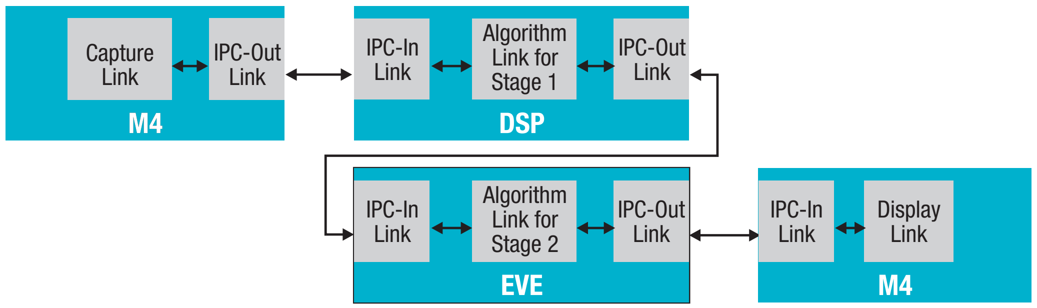
tools and components necessary to build such applications, including code gen tools, BIOS, IPC, starterware, BSP drivers, networking stacks, codecs and algorithm kernels (see Figure 4).

Inter Processor Communication (IPC)

Interprocessor communication (IPC) is used to collaborate between different tasks running on different CPUs on the SoC to build a data flow or a use-case. The choice of IPC affects system performance and processing latency. Hence it's important to use an IPC which offers lowest CPU overhead and fastest communication method in the real-time processing path. At the same time, most users are comfortable programming on single processor systems. So it's important to hide IPC details from users so these users still get the same convenience as they do when programming on a single processor.

A user integrates an algorithm or a processing element ("link" in Vision SDK) as if they would use it to talk to other elements or "links" on the same CPU. These "links" directly exchange information with each other at a high rate (ex: 60fps or 120fps) to establish a data flow, i.e., it's a peer-to-peer communication method with no intervention of any control task. By using an element called IPC link, the communication can be extended across different processors. The low-level IPC mechanism uses a non-locking shared region queue for lowest CPU overhead communication and a HW mailbox-based interrupt mechanism for lowest latency communication.

Example, a two-stage algorithm processing between video capture and video display is shown on the following page.



The TI Vision SDK also provides another means of IPC to pass infrequent control messages to different tasks on different CPUs. The user who sends this message simply calls a function on the local CPU using "MessageQ"-based IPC mechanism. The function to be invoked, along with user-supplied parameters, gets executed on a different CPU. This acts like a convenient "remote procedure call (RPC)" mechanism which is used to create, control, and delete "links" on other processors.

Thus IPC mechanism in TI Vision SDK provides a low overhead, low latency method to exchange real-time data among different processing elements. And at the same time, provides users with convenient "RPC" mechanisms to invoke asynchronous control functions on different CPUs.

Resource allocation (Memory/EDMA)

The TI Vision SDK supports efficient and flexible resource allocation of important resources in the system like external DDR memory, EDMA channels, and internal on-chip memory.

By default the TI Vision SDK is configured to use 256-MB DDR memory, but via configuration changes the "memory map" can be changed to use less or more memory. This is achieved by dividing the DDR memory into different segments which will be used for code/data for different CPUs. The sizes of these segments can be reduced or increased depending on the algorithms that need to be integrated into the system and the memory space they need for execution.

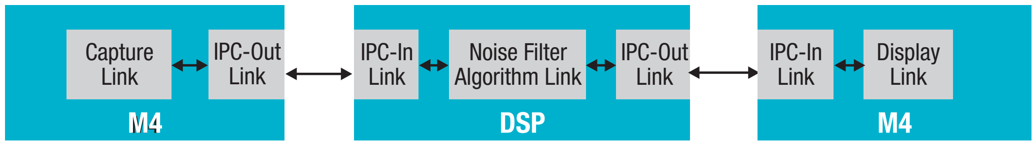
Frame buffer memory is a separate memory segment which is used to allocate memory for video and analytics information buffers. All buffer memory for frame buffers/vision analytics algorithm buffers would be allocated by the links at "create" time. Once a link is in run phase no dynamic memory will be allocated due to functional safety reasons. Thus the memory allocation for a given use case would be predictable and any memory allocation issues can be caught at development phase itself. A shared region in IPC is used as buffer heap. This heap is a multi-processor heap, i.e., different processors can allocate memory from this heap using IPC APIs. IPC will take care of interprocessor mutual exclusion during memory allocation.

The TDA2x SoC has multiple EDMA controllers for improving performance of streaming vision processing operations. The system EDMA controller can be used by all CPUs in the system. In addition each DSP and EVE has its own "local" EDMA controller. The system EDMA controller is made accessible to IPU1-M4-0, IPU1-M4-1, A15, 2×DSP via EDMA3 LLD APIs. The "local" EDMA controller is also made accessible to the local CPU (DSP or EVE) via the same EDMA3LLD APIs. A convenient means of configuring the channel allocation for each core is provided by the TI Vision SDK. EDMA3 LLD APIs is used for EDMA channel and PaRAM resource management. Utility APIs to copy / fill buffer are provided by TI Vision SDK for system EDMA controller. Both interrupt / polling methods are supported.

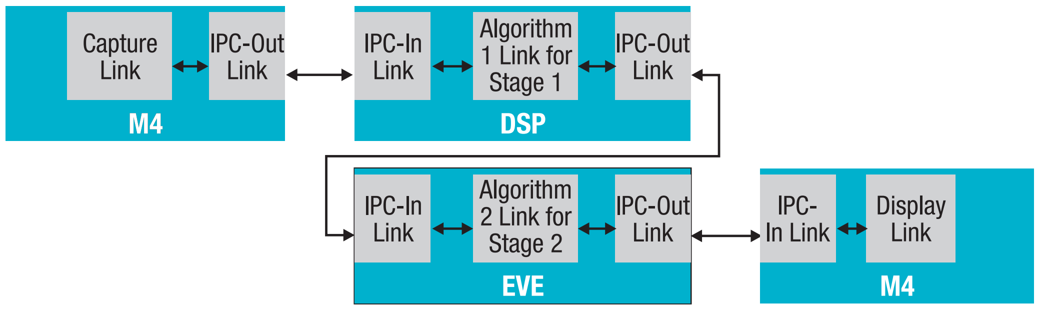
Algorithm – Easy plug-in support

TI Vision SDK provides a fast and convenient means of integrating various kinds of algorithms in the system. The means of integrating an algorithm in a system is via an “algorithm link”.

An algorithm link is like any other link in the system chain. While the capture or display link encapsulates the corresponding drivers in them, algorithm link encapsulates an algorithm in it. Algorithm links can be connected with other links in the system to form a chain. An example of a noise-filtering algorithm link on DSP, connected in the system chain, looks like below.



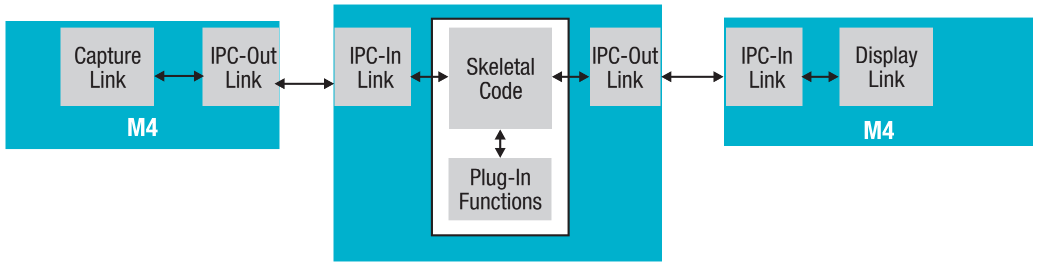
An algorithm encapsulated within the algorithm link can be a complete application or a stage of an application (for example, Pedestrian detection, Median filtering, etc.). Several stages of an application spread across several cores can be connected by forming a chain between multiple algorithm links as shown below.



If several stages of an application execute on a same core in a sequential manner, they can be encapsulated within a single algorithm link, by calling the stages in a sequential manner.

To enable easy and fast development of algorithm links, the link is designed to consist of two portions – skeletal and plug-in functions.

The skeletal part of algorithm link, comprises of portions of algorithm link implementation, which are common across algorithms. It takes care of generic aspects of link implementation like link creation, link state machine management and communication with other links.



Plug-In functions are comprised of functions which cater to algorithm dependent functionality. These need to be written by the user and are specific to the algorithm being integrated.

The skeletal portion of the code and plug-in functions communicate with each other via a predefined API. Skeletal code implementation and the communication API is kept the same, independent of the processing core (EVE/DSP/A15/M4). Skeletal code calls the plug-in functions based on the state of the algorithm link.

Plug-in functions have the implementation to create and use the actual algorithm functions (provided by the algorithm provider). Plug-in functions can interact with algorithm functions via TI XDAIS (TI TMS320 DSP algorithm API standard) or any other custom interface.

AlgorithmLink_AlgPluginCreate	Plug-in function which will perform algorithm instance creation.
AlgorithmLink_AlgPluginProcess	Plug-in function which will process new data. Internally it will call the process function of the algorithm.
AlgorithmLink_AlgPluginControl	Plug-in function which will perform Control (Configuration) of the algorithm. Internally it will call the control function of the algorithm.
AlgorithmLink_AlgPluginStop	Plug-in function which will perform all functionality which needs to be done at the end of algorithm. Example: If any buffers are locked inside the algorithm, they can be flushed in this function.
AlgorithmLink_AlgPluginDelete	Plug-in function which will perform algorithm instance deletion

Table 1: Algorithm plugin interface

Thus, algorithm link provides a means to integrate different algorithms by implementing few “plug-in” functions. The lower-level details like exchanging of frames with other algorithm stages or capture device and inter-processor communication are hidden from the algorithm integrator. This allows a fast and easy means to integrate algorithms within a system context.

EVE and DSP libraries

The TI Vision SDK is complete with more than 200 optimized functions for both EVE and DSP libraries, providing customers and third parties with the building blocks to jump start development and reduce the time to market. Additionally, both libraries are available for low-to-mid- and high-level vision processing. Integral image, gradient, morphological operation and histograms are examples of low-level image processing functionalities; HOG, rBRIEF, ORB, Harris and optical flow are key functions as a mid-level and Kalman filtering and Adaboost are high-level processing functions. At CES 2014, TI showcased a surround view demo and the industry's first 1080p60 Dense optical flow in real-time demo built upon the EVE and DSP libraries. Table 2 shows the categories and examples of EVE and DSP libraries.

Category	EVE and DSP Lib examples
Low-level processing	Integral image, Gradient, Morphological operation, Histograms
Mid-level processing	HoG, rBRIEF, ORB, Harris, Optical flow
High-level processing	Kalman filtering, Adaboost

Table 2: Categories of DSP / EVE vision kernels

Examples use cases

TI Vision SDK implements a few use cases with reference ADAS algorithms to show how to integrate and build realistic systems using TI Vision SDK.

Surround view

In the surround view use case, TI Vision SDK shows how to capture video data from multiple cameras, synchronize them and then "stitch" them together by passing the frames through a multi-stage surround view stitching algorithm. Currently the algorithm is implemented on a DSP and is spread across two DSPs as shown below in Figure 5.

The TI Vision SDK allows easy integration of the algorithm via algorithm link. It allows the user to test different system partitioning scenarios like 1xDSP vs. 2xDSP without having to rewrite code. Finally the configuration chosen for the reference algorithm implementation was a 2xDSP configuration. It allows algorithm integration without worrying about SoC details like number of CPUs in the system, i.e., the algorithm is integrated as if it is getting frames from a task on the same processor. Using IPC mechanism, the actual frames exchange happens between multiple CPUs.

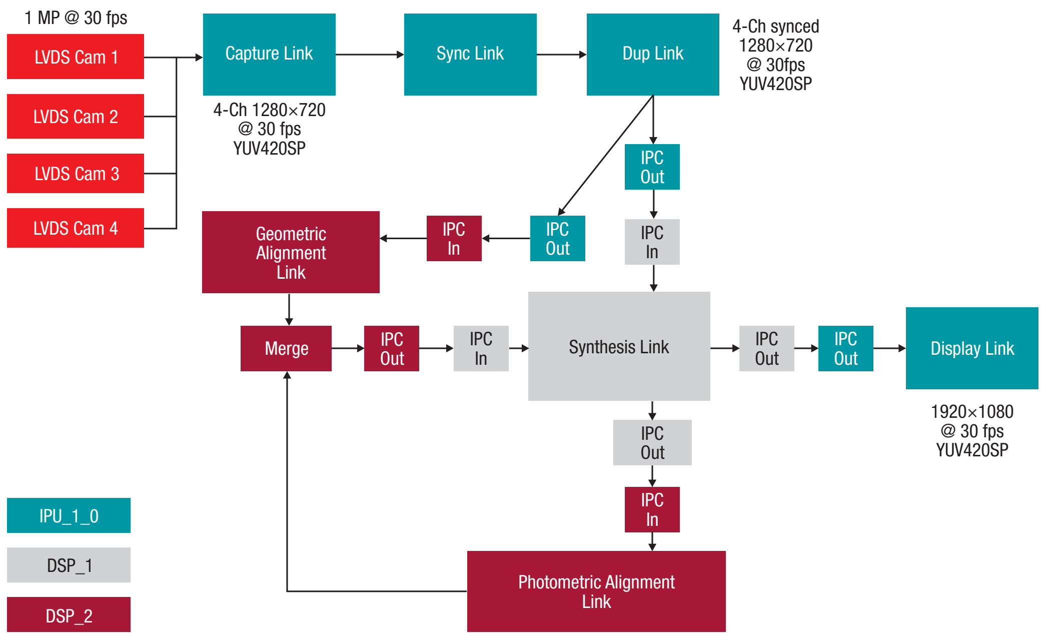


Figure 5: 4-ch LVDS surround view application – TI Vision SDK implementation

Optical flow

In the optical flow use case, a dense optical flow algorithm in integrated into the EVE subsystem. This use case shows how to split the same algorithm across different CPUs. Here 4xEVE collaborate to implement a 1080p 60 fps LK dense optical flow example. The TI Vision SDK allows the same frames to be processed by different EVEs, each operating on a different "region of interest" (ROI). Thus all EVEs work on the same frame at the same time but different portions of the frame to enable a high-performance, low-latency, optical flow system.

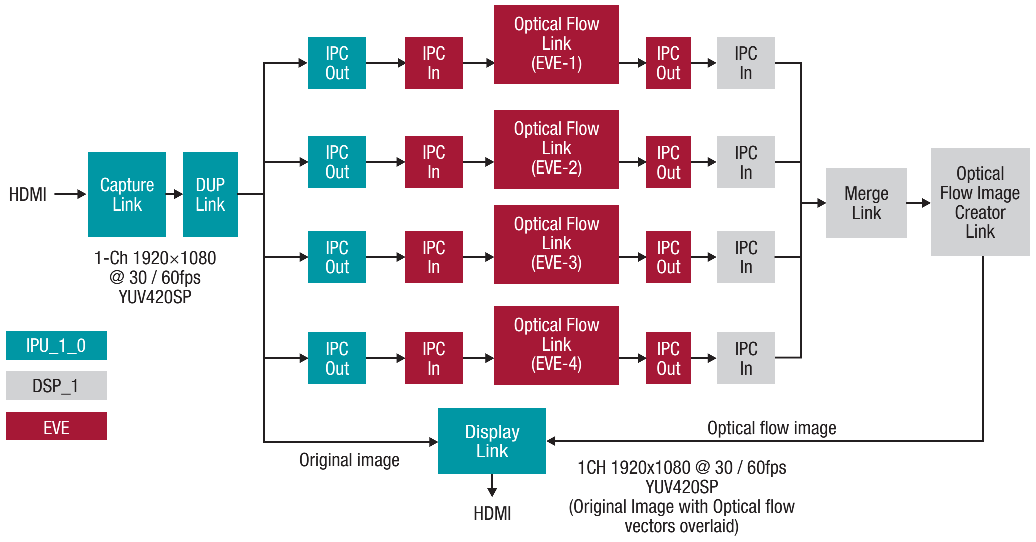

HDMI
Capture Link
DUP Link
1-Ch 1920×1080 @ 30 / 60fps YUV420SP
IPU_1_0
DSP_1
EVE
IPC Out
IPC In
Optical Flow Link (EVE-1)
Optical Flow Link (EVE-2)
Optical Flow Link (EVE-3)
Optical Flow Link (EVE-4)
IPC Out
IPC In
Merge Link
Optical Flow Image Creator Link
Optical flow image
Display Link
Original image
HDMI
1CH 1920x1080 @ 30 / 60fps YUV420SP (Original Image with Optical flow vectors overlaid)

Figure 6: Front camera dense optical flow application – TI Vision SDK implementation

New Open Software initiatives

With increasingly complex SW solutions for ADAS, the interoperability of different components, portability and maintenance become important. There are several important initiatives such as AUTOSAR and OpenVX which define system architecture guidelines to help meet these objectives. See Figure 7 on the following page.

AUTOSAR is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool vendors. Among many other objectives, it defines a communication network among a vehicle's many ECUs. Therefore, for vehicles implementing AUTOSAR, ADAS ECU has to comply, at minimum, with AUTOSAR communication protocols. While ADAS ECUs, with its stringent functional safety requirements, use dedicated safety MCUs for this purpose, ECUs with lesser safety requirements could optimize the cost, by utilizing available microcontroller cores inside Vision SoC. TDA2x SoC contains ARM Cortex®-M4 cores which could be used for running anything from full AUTOSAR software stack to a minimal CAN communication stack, compliant with AUTOSAR. TI is engaged with several partners, such as Vector, Electrobit and Mentor Graphics to provide such AUTOSAR software solutions.

OpenVX™ is a new Khronos open industry-wide standardization initiative that defines the computer framework and vision libraries for low-power, high-performance processing on embedded heterogeneous processors. An OpenVX application expresses vision processing as a graph of function nodes. An OpenVX

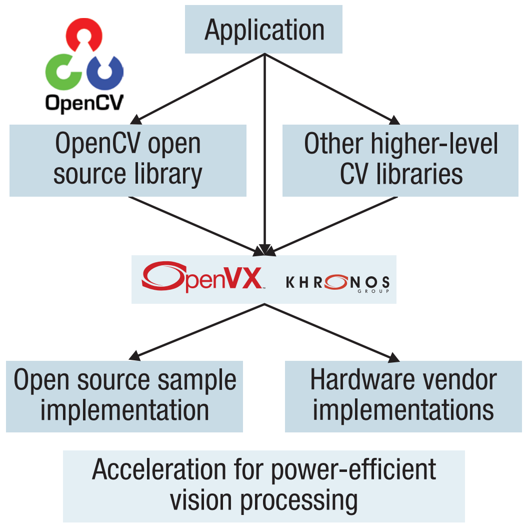

Application
OpenCV
OpenCV open source library
Other higher-level CV libraries
OpenVX
KHRONOS GROUP
Open source sample implementation
Hardware vendor implementations
Acceleration for power-efficient vision processing

Figure 7: Power efficient vision processing with OpenVX

implementer can optimize graph execution through a wide variety of techniques, such as acceleration of nodes on CPUs, DSPs or dedicated hardware accelerators. TI is a key contributor to the Khronos OpenVX standard.

Summary

It is apparent that the need for advanced ADAS systems in every car around the world is becoming important to making driving easier. TI has been investing in ADAS technology for years. TI's latest SoC, the TDA2x, enables easier system integration, reduced time to market and reduced cost. The Vision AccelerationPac, with multiple EVEs integrated offers more than 8× the performance of existing ADAS systems at the same power levels. It leverages the VisionSDK including its application SW, example code and DSP/EVE libraries to optimize and create ease-of-use in a complex heterogeneous SoC.

Resources

[1] **Global Traffic Accidents/Deaths Statistics**

[2] **Nissan Announces Unprecedented Autonomous Drive Benchmarks**

[3] **Audi Demonstrates Its Self-Driving Car at CES 2013**
BMW Self-Driving Car
Toyota Self-Driving Cars

[4] **TI Gives Sight to Vision-Enabled Automotive Technologies**

[5] **Empowering Automotive Vision with TI's Vision AccelerationPac**

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		