Supporting MIPI in Embedded Vision System Designs Using Lattice CrossLink FPGAs

A Lattice Semiconductor White Paper.

April 2020

V. 1 WP0023

TABLE OF CONTENTS

The past few years have seen a dramatic increase in the use of embedded vision, with applications ranging from relatively simple vision-enabled doorbells to sophisticated industrial robots performing random pick-and-place to autonomous mobile robots (AMRs) navigating their way around an uncontrolled environment in which the landscape may be constantly changing. Industries that are rapidly adopting embedded vision include automotive, consumer electronics, medical, robotics, and security/surveillance, along with a wide range of industrial applications.

Today, embedded vision is increasingly being used as part of artificial intelligence (AI) and machine learning (ML) systems to analyze images and video streams, to detect and identify objects and people, and to derive actionable information from what they see (see also What the FAQ are AI, ANNs, ML, DL, and DNNs?[1]).

This paper first introduces some of the current trends in embedded vision design. Next, the use of field-programmable gate arrays (FPGAs) in embedded vision systems is considered. Finally, creating embedded vision systems using CrossLink™ FPGAs from Lattice Semiconductor is discussed.

Trends in Embedded Vision Design

Some of the main trends that are strongly driving embedded vision designs are the continual demand for increasing resolution and bandwidth, an increasing number of vision sensors and/or displays per system, and the ongoing mandate to reduce size and power consumption. Just to add a little spice to the mix, there‘s also an increasing demand to add AI/ML capabilities into embedded systems, especially close to the vision sensors themselves.

Another trend in vision systems is the continued expansion of the number and types of interface standards. Table 1 shows a subset of the most common interfaces, where standards that are used “inside the box” (i.e., inside a device) are typically required to drive only a few centimeters to tens of centimeters, while “box-to-box” interfaces may be required to drive vision data from a few meters to a hundred meters or more.

Embedded (inside the box) ⟷ Box-to-Box

Interface	RGB	OpenLDI	MIPI	eDP/DP	HDMI	USB	GigE
Protocol	Parallel	OpenLDI	CSI-2/DSI	DP	HDMI	USB	GigE
PHY	CMOS	LVDS	D-PHY	8b10b SERDES	8b10b TMDS	8b10b SERDES	8b10b SERDES

Table 1. Common video interfaces

Note that the term PHY, which is an abbreviation for “physical layer,” is an electronic circuit that is usually implemented on a chip or as a chip (integrated circuit). The PHY is required to implement the physical layer functions of the Open Systems Interconnection (OSI) model, which characterizes and standardizes the communication functions of a computing or telecommunications system without regard to its underlying internal structure and technology.

Introducing MIPI

Of particular interest to embedded vision applications is the growing use of MIPI, especially with regard to the CSI-2 (camera/sensor) and the DSI (display) protocols, both of which employ a PHY called the D-PHY. In terms of bandwidth and interface length, MIPI conceptually sits between OpenLDI and eDP/DP (embedded Display Port and Display Port).

The MIPI Alliance is a global organization that has more than 250 member companies worldwide. When the MIPI Alliance was originally founded by ARM, Intel, Nokia, Samsung, STMicroelectronics and Texas Instruments in 2003, "MIPI was an acronym for Mobile Industry Processor Interface." However, since the organization's specifications today address not only processor connectivity, but the full range of interface needs in a device, MIPI is no longer used as an acronym, but as a standalone name in its own right.

In order to provide some context with respect to the growing popularity of MIPI, we might think back to the mid-1990s when personal computers (PCs) were starting to become popular (Figure 1). The interfaces used by PCs at that time were the Peripheral Component Interconnect (PCI) and the Universal Serial Bus (USB). These low-cost PCI and USB technologies were subsequently adopted by a wide variety of different product classes

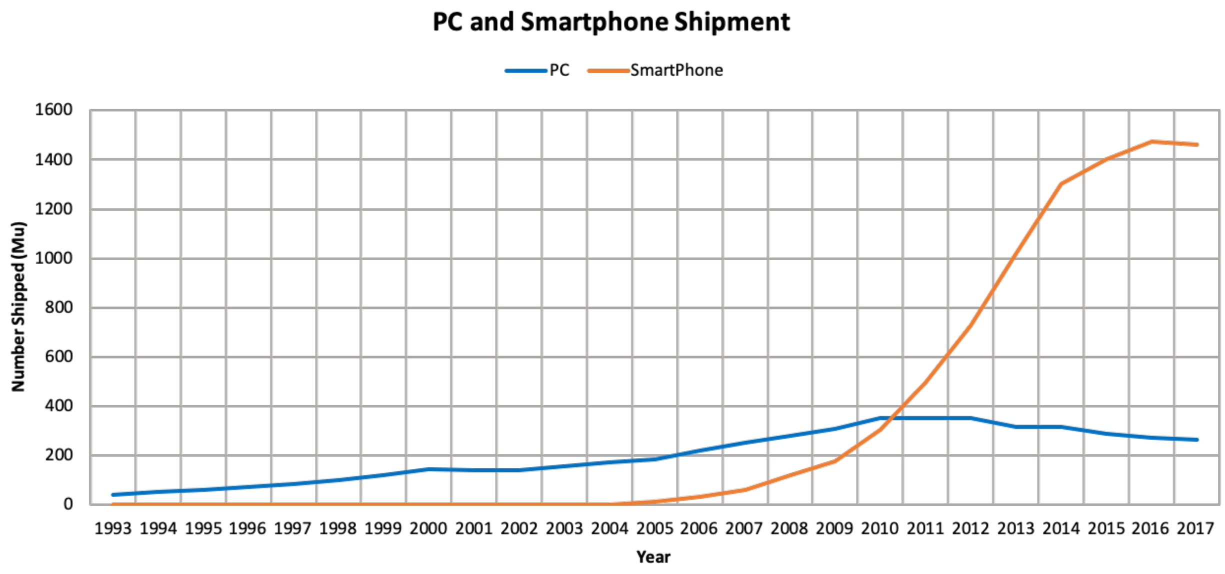


Figure 1. PC and smartphone shipments (Source: Lattice Semiconductor and Industry Analysts)

Much the same thing is happening with the MIPI CSI-2 and DSI-2 protocols, which were originally created with smartphones as the primary use case. Since the launch of the first iPhone in 2007, the growth of smartphones has been substantive, far outstripping that of PCs. This growth is driving economies of scale, resulting in low-cost camera, display, and processor components, where these components are now being adopted in a wide variety of other (non-smartphone) applications and markets. (In this context, the term "processor" may refer to System-on-Chip (SoC), application-specific standard part (ASSP), and application processor (AP) devices.)

The D-PHY interface employed by MIPI (Figure 2) employs one differential clock and between one and four differential data lanes, where the data lanes support speeds from 80 megabits-per-second (Mbps) to 2.5 gigabits-per-second (Gbps).

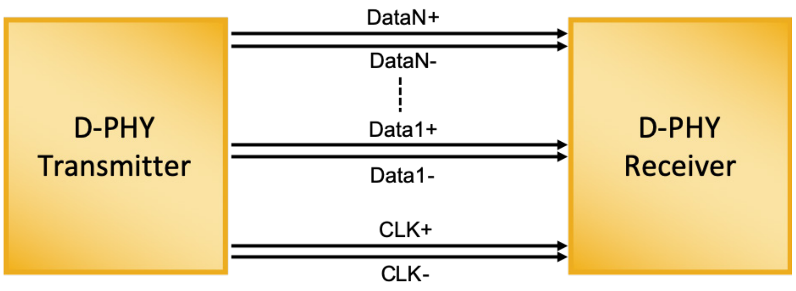


Figure 2. The MIPI D-PHY interface

The MIPI interface supports two modes of operation: high speed (HS) and low power (LP). The interface is primarily unidirectional between the transmitter and the receiver, although some low-speed communication is possible from the receiver back to the transmitter.

Using FPGAs in Embedded Vision Designs

One of the predominant characteristics of embedded vision designs is that the architecture oftentimes needs to dynamically change to respond to evolving specifications, which results in opportunities for FPGAs as the implementation technology of choice.

A big consideration is the relative difference in development cycles between ASIC/ASSP and FPGA designs, which results in a faster time to market (and time to profit) when using an FPGA implementation (Figure 3).

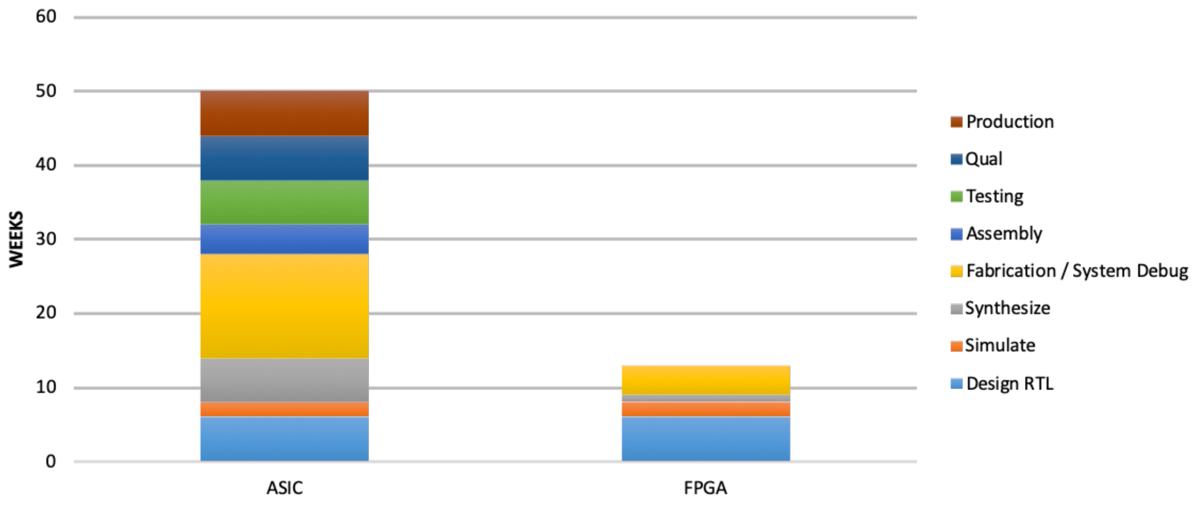


Figure 3. Faster time to market (and time to profit) using FPGAs

The development cycle for a medium-sized ASIC/ASSP takes almost a year from design to mass production. In addition to the fact that there is a lot of risk due to ongoing changes in the design specification (this is a common problem in the case of embedded vision designs), a lot of the ASIC/ ASSP design processes are intertwined, which means that a delay in one process can have a knock-on effect in others.

In the case of an FPGA implementation, the designers are already working with market-proven hardware that can be reconfigured on-the-fly to accommodate any changes in the design specification.

From the designers' standpoint, the initial stage of the development cycle – capturing the design at the register transfer level (RTL) of abstraction – is the same, while the remaining steps are either similar, require less time and resources, or are non-existent. The overall result of using an FPGA is to have a fully working device in approximately one third of the time.

For historical reasons, some embedded system designers think of FPGAs as being physically large, power-guzzling components intended only for use in large applications like data centers, communications hubs, medical imaging, and military applications.

These traditional FPGAs – with hundreds to thousands of LUTs consuming 50 to 100 watts or more presented in large 55 x 55 mm packages with heatsinks – certainly do exist. However, there are also more modern families of function-driven FPGAs that are focused on applications that demand small form factors and low power consumption. Lattice Semiconductor, for example offers function-driven FPGAs that range from "small" (10 x 10 mm consuming ~1 W) to "tiny" (1.4 x 1.4 mm consuming ~1 mW) that address the needs of applications where minimum size and maximum power efficiency are of critical importance.

While comparable to small ASIC/ASSPs, these function-driven FPGAs are easier and faster to develop and much more flexible to use. Furthermore, since most general-purpose FPGAs are intended to be deployed in multiple markets, including industrial and automotive, they are typically available with support for commercial and industrial temperature environments. By comparison, ASIC/ASSP support for extended temperature environments is less common because these devices are often developed with deployment in consumer products in mind.

Introducing CrossLink FPGAs

Lattice Semiconductor offers four main families of FPGAs: ECP™, MachXO™, iCE™, and CrossLink. The ECP family comprises what many designers would consider to be "traditional" FPGAs – general-purpose devices targeted at connectivity and acceleration applications. With hundreds of programmable input/outputs (I/Os), MachXO FPGAs are perfect for a wide range of applications that require GPIO expansion, interface bridging, and power-up management functions. Meanwhile, iCE are the smallest, ultra-low-power FPGAs available, with the smallest member of the family offering 18 I/Os in a 1.4 mm x 1.4 mm package.

Of particular interest to us here are the CrossLink FPGAs, which are optimized for high-speed video and sensor applications. Augmenting their traditional programmable fabric with hardened PHYs, CrossLink FPGAs provide the industry's fastest MIPI D-PHY bridging solution supporting 4K UHD resolution at speeds up to 12 Gbps (Figure 4). Furthermore, CrossLink devices are available in amazingly small 2.46 x 2.46 mm WLCSP packages and BGA packages with 0.4 mm, 0.5 mm, and 0.65 mm pitches.

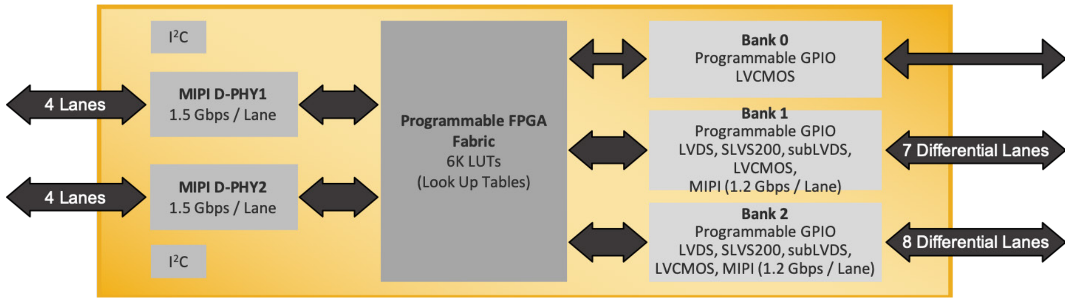


Figure 4. CrossLink FPGAs for embedded video include hardened MPIP D-PHYs for performance and design optimization

In addition to two 4-lane MIPI D-PHY transceivers offering 6 Gbps per PHY, CrossLink FPGAs also provide 15 programmable source synchronous I/O pairs for camera and display interfacing using the LVDS, SLVS200, subLVDS, LVCMOS, and OpenLDI (OLDI) interface standards. Furthermore, these programmable I/Os can be used to implement "soft MIPI" interfaces running at up to 1.2 Gbps-per-lane, while LVCMOS can be used to natively implement parallel/RGB interfaces.

Example Applications

Example applications for CrossLink FPGAS include sensor bridging, sensor duplication, sensor aggregation, display bridging, and display splitting, where the term "bridging" refers to converting video signals from one interface standard to another.

Using Legacy Controllers with MIPI Sensors and Displays: A typical case might involve a legacy system whose SoC, ASSP, or AP doesn't support MIPI, but whose designers wish to keep the existing processor (and its code) while upgrading the rest of the system with new, more efficient, lower-power, MIPI enabled sensors and/or displays (Figure 5).

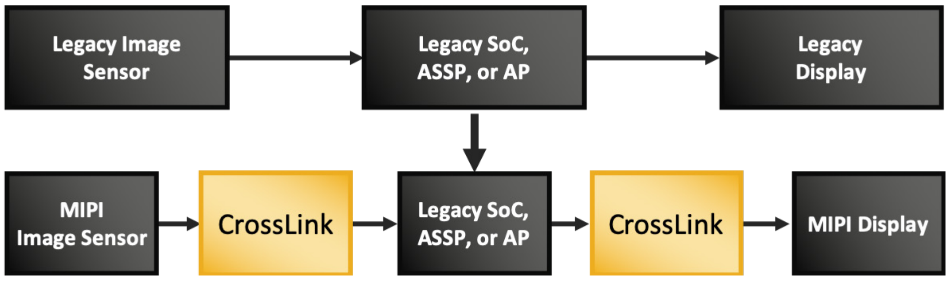


Figure 5. Using CrossLink FPGAs to allow a legacy SoC, ASSP, or AP to work with MIPI sensors and/or displays

Using MIPI Controllers with Legacy Sensors and Displays: A corresponding problem occurs when designers have an SoC, ASSP, or AP that supports MIPI, but they wish to use existing ("legacy") sensor and display subsystems that feature non-MIPI components. For example, many image sensors and displays used in industrial markets employ LVDS, SUBLVDS, or parallel interfaces. Furthermore, many of these older sensors employ a global shutter rather than a rolling shutter, all of which drives the need for a sophisticated bridging solution (Figure 6).

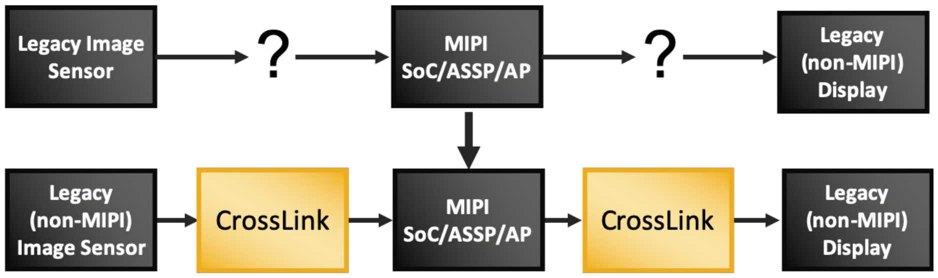

Legacy Image Sensor
?
MIPI SoC/ASSP/AP
?
Legacy (non-MIPI) Display
Legacy (non-MIPI) Image Sensor
CrossLink
MIPI SoC/ASSP/AP
CrossLink
Legacy (non-MIPI) Display

Figure 6. Using CrossLink FPGAs to allow a MIPI-enabled SoC, ASSP, or AP to work with legacy (non-MIPI) sensors and/or displays

Sensor Duplication: One example of the requirement for sensor duplication is provided by safety-critical applications that require redundant data, like a smart car, where the video feed from a sensor might be split into two streams that are fed to two separate processors (like many of the other interface standards, MIPI is point-to-point, so it's not possible to directly connect a single sensor to multiple processors).

The idea is that, should one of the processors fail, there has to be a backup. Furthermore, if there are multiple sensors, then the stream from each sensor may be split and fed into multiple processors. Of course, sensor duplication may be combined with a bridging function. One possible configuration is illustrated in Figure 7.

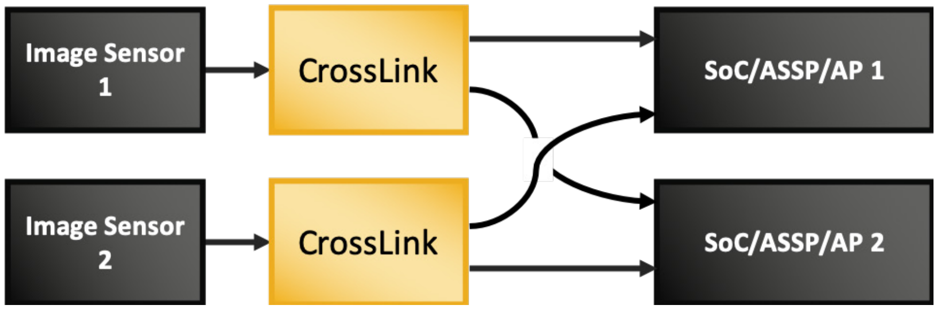


Figure 7. Using CrossLink FPGAs to implement sensor duplication in safety-critical systems

Display Splitting: A corresponding situation called display splitting occurs when it is required to take a video signal being generated by the system's processor and split this signal so as to feed multiple displays. As for sensor duplication, display splitting may be combined with a bridging function. One possible configuration is illustrated in Figure 8.

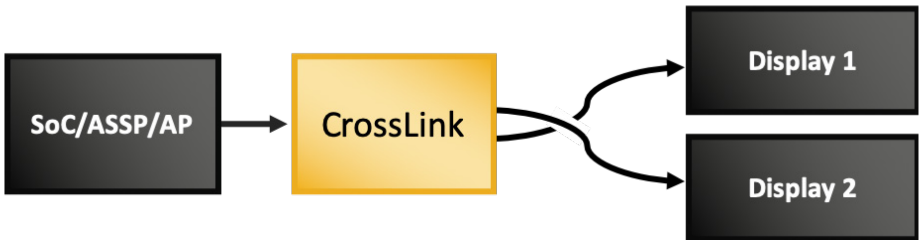


Figure 8. Using CrossLink FPGAs to implement sensor duplication in safety-critical systems

Sensor Aggregation: Finally, we come to sensor aggregation, which -- as was discussed earlier in this paper – may be driven by the trend to add more and more image sensors into systems. The issue here is that some processors have a limited number of sensor inputs, so we need some way to aggregate the data from multiple sensors.

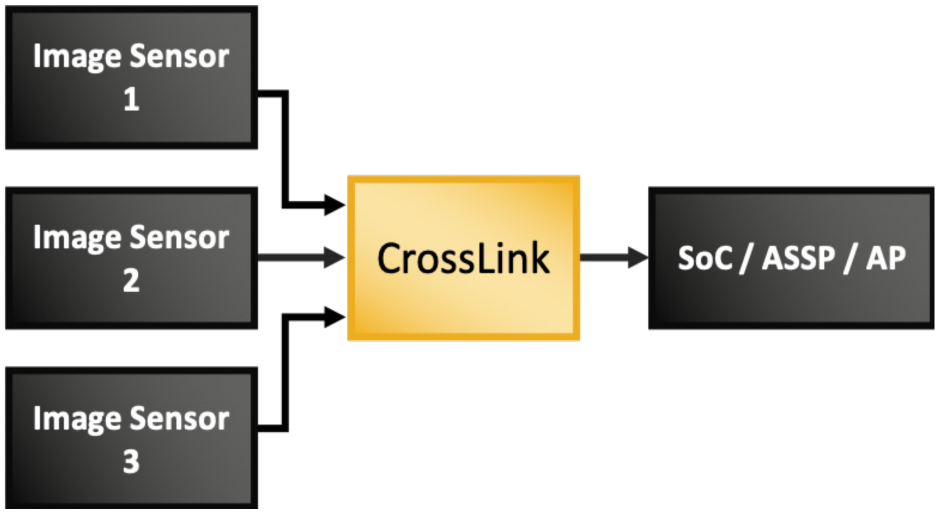


Figure 9. Using a CrossLink FPGA to implement sensor aggregation

And, of course, as with sensor duplication and display splitting, sensor aggregation may also be combined with bridging functionality.

Designing with CrossLink FPGAs

When you are designing with CrossLink FPGAs, the starting point is to ask some basic questions and use the answers to make some fundamental architectural decisions. For example, what PHYs and protocols are you going to use to input and output the video signals (Figure 10)?

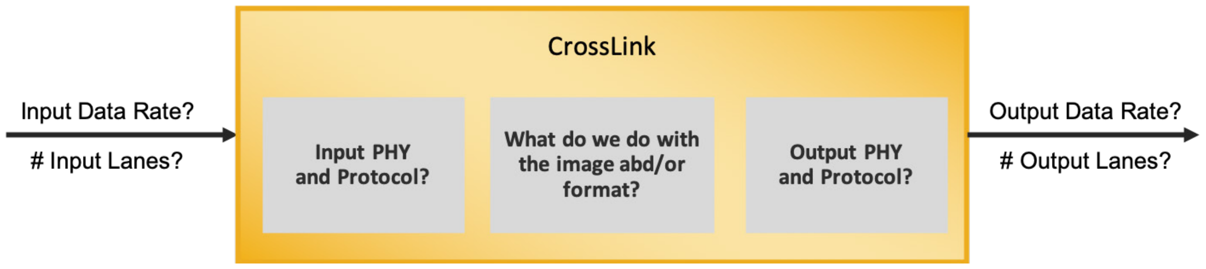


Figure 10. Making architectural decisions

Many of these decisions are driven by the types of sensors and displays you choose (or are obliged to use because they‘ve been chosen for you). These decisions may also be influenced by the type of SoC/ASSP/AP processor you are going to be working with. In turn, the answers to these questions will lead you to ask how you wish to manipulate the image and/or its format, including whether you are going to be performing duplication, aggregation, or splitting. Furthermore, once you‘ve determined the input and output data rates, you can combine this information with the PHY and protocol decisions to resolve how many input and output lanes your design is going to require.

Once you‘ve made your initial architectural decisions, you can use these to calculate the data rate. For example, let‘s assume that you are working with a Full High Definition (FHD) signal with a frame rate of 60 Hz and a color depth of 10 bits (which is also known as “RAW10”), as illustrated in Table 2.

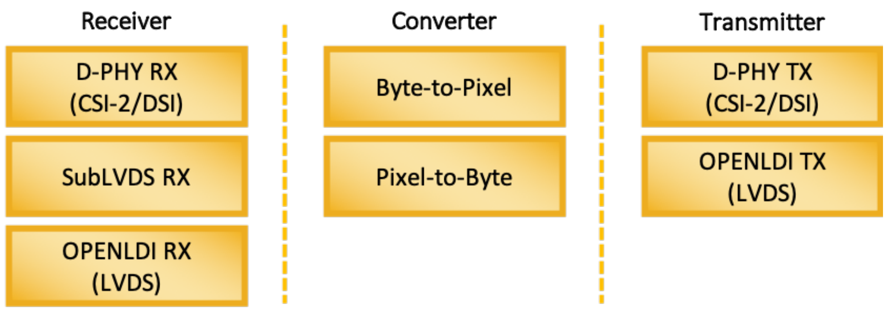
Resolution	Frame Rate (Hz)	Pixel Clock (MHz)	Color Depth (Bits)	Total Data Rate (Mbps)	# Lanes	Line Rate* (Mbps)
HD (1280 x 720p) (1650 x 750)	60	74.25	10	742.50	1	742.50
			12	891	1	891
FHD (1920 x 1080p) (2200 x 1125)			8	1188	1	1188
	60	148.50	10	1485	2	742.50
			12	1782	2	891
UHD (3840 x 2160p) (4400 x 2250)	30	297	8	2376	2	1188
			10	2970	4	742.50
			12	3564	4	891
	60	594	8	4752	4	1188

**The maximum bandwidth for the programmable I/O is 1.2 Gbps.*

Table 2. Calculating data rates

We typically think of FHD as having a resolution of 1920 x 1080 pixels, but the actual resolution is 2200 x 1125 because we have to add in blanking periods between lines and frames. The formula for calculating the data rate is total = horizontal clocks * vertical lines * frame rate * bits-per-pixel. In the case of our example, this means our total data rate is 2200 x 1125 x 60 x 10 = 1485 Mbps (1.485 Gbps). Since the maximum bandwidth for the CrossLink‘s programmable I/Os is 1.2 Gbps, this means we will need to use two lanes, each with a line rate of 742.5 Mbps.

CrossLink FPGAs are supported by a library of Video Modular IP. These IP modules, which are provided royalty free, are focused on receiving video data (Rx), transmitting video data (Tx), and clock domain conversion (Figure 11).



Parallel interfaces are supported natively

Figure 11. CrossLink Video Modular IP

Most designs use multiple IP modules as building blocks, where these modules are augmented by the designer's own "secret squirrel sauce" implemented in RTL.

One question some designers have when they see this image is why there's MIPI CSI-2 transmitter when they think of CSI-2 as being received from a camera sensor. Similarly, they may question why there's a MIPI DSI receiver when they think of DSI as being transmitted to a display. The answer, of course, is that it may be necessary to perform duplicating, splitting, and/or bridging functions.

Let's start with a simple SubLVDS to MIPI CSI-2 bridging example. In this case, we will employ the SubLVDS receiver module, the pixel-to-byte converter module, and the MIPI CSI-2 transmitter module (Figure 12).

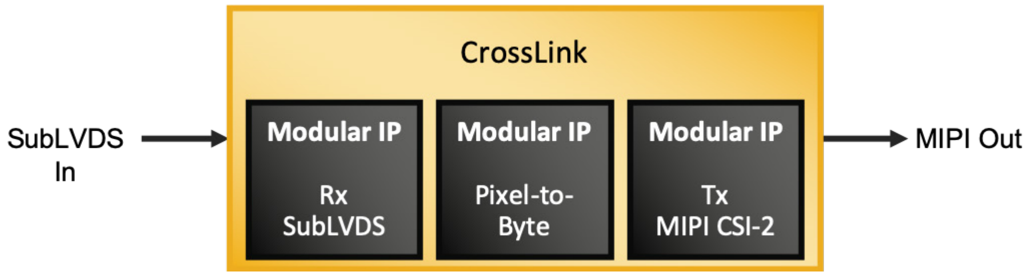


Figure 12. Using CrossLink to implement a SubLVDS to MIPI CSI-2 bridge

The reason for using the pixel-to-byte converter is that the LVDS and MIPI formats are different. Let's assume our SubLVDS input has a 10-bit (RAW10) color depth. The problem is that the MIPI format is based on 8-bit packets. This means we need to determine a common bit-length that uses the lowest common denominator, which will be 40 bits in this case (Figure 13).

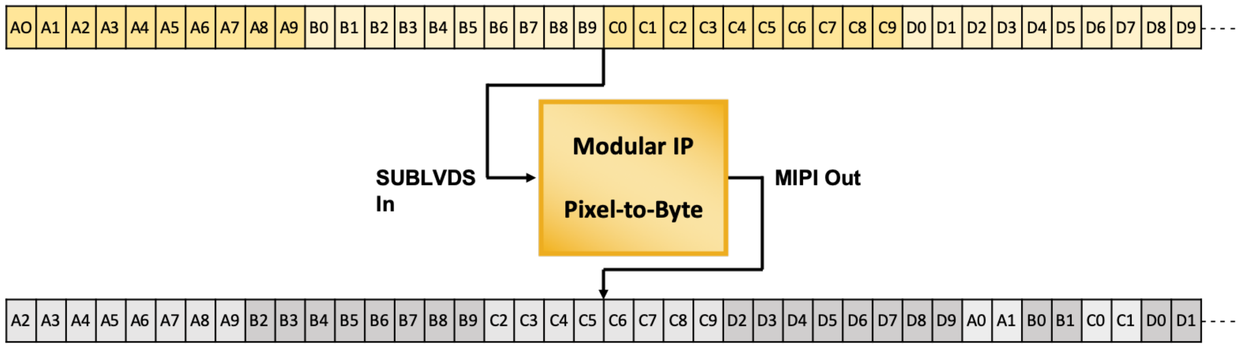

AO A1 A2 A3 A4 A5 A6 A7 A8 A9 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
SUBLVDS In
Modular IP
Pixel-to-Byte
MIPI Out
A2 A3 A4 A5 A6 A7 A8 A9 B2 B3 B4 B5 B6 B7 B8 B9 C2 C3 C4 C5 C6 C7 C8 C9 D2 D3 D4 D5 D6 D7 D8 D9 A0 A1 B0 B1 C0 C1 D0 D1

Figure 13. Using CrossLink's pixel-to-byte IP module to convert RAW10 to MIPI CSI-2

Next, let's consider a MIPI sensor duplicator example. In this case, we have a MIPI CSI-2 receiver and a MIPI CSI-2 transmitter. In between, we need to create some custom RTL to duplicate the MIPI frames as they pass through the CrossLink device. If more outputs are required, multiple CrossLink FPGAs can be cascaded together (Figure 14).

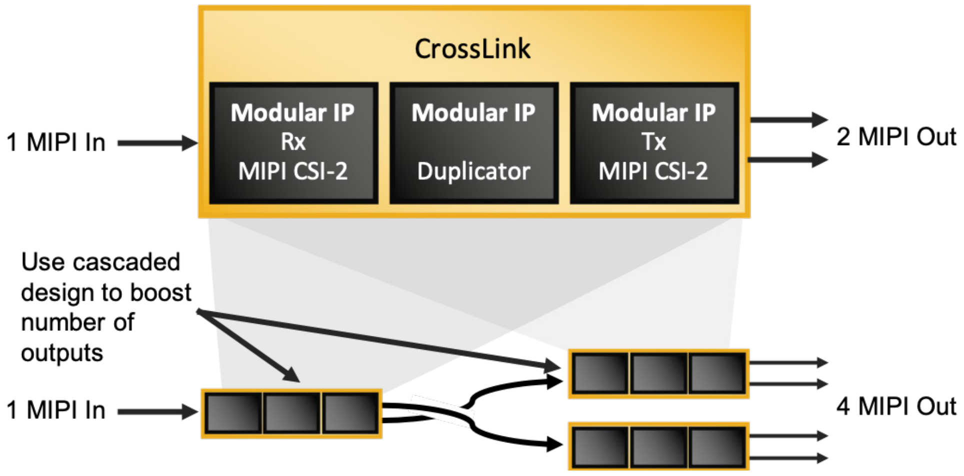


Figure 14. MIPI sensor duplicator example

Another common design requirement is to create an aggregator, in which multiple video sensor streams are combined. Figure 15 illustrates two scenarios – a side-by-side merge and a virtual channel.

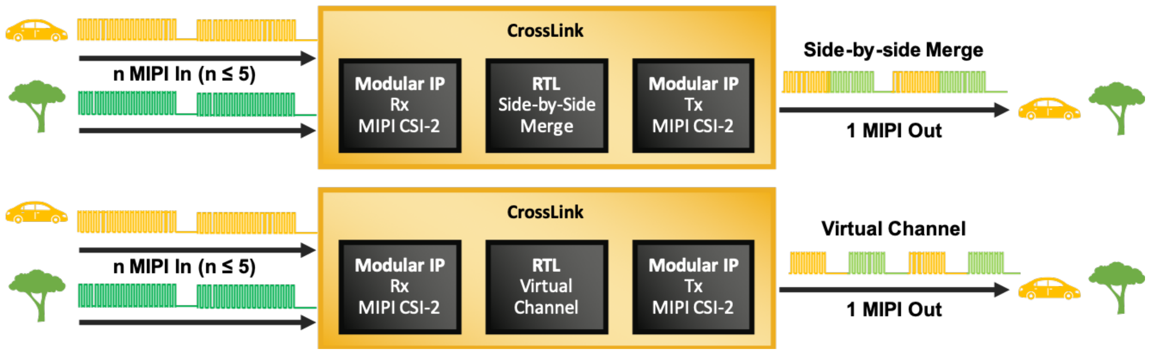


Figure 15. MIPI sensor aggregation examples

Both of these examples show two MIPI inputs and a single MIPI output. In fact, the Crosslink FPGA can support ‘n’ MIPI inputs, where n ≤ 5. When using two inputs, the output data rate has to be twice the speed of the input data rates.

In the case of the side-by-side merge, the output is a combined (merged) image, which is created on a line-by-line basis.

The concept of the virtual channel is something that is defined in the MIPI standard. MIPI is packet-based protocol, which means each packet contains a header and payload (data). In the case of the virtual channel, we tag each of the output packets and let the downstream SoC/ASSP/AP processor sort things out. At the time of this writing, most processors cannot handle virtual channels, but this approach is expected to be the trend moving forward and to proliferate in the not-so-distant future.

Another variation on sensor aggregation is to perform a top-and-bottom merge. In this case, from a design perspective, we can no longer simply pass the data through. Instead, we have to store at least one frame of the overall image before we can output anything via the MIPI transmitter, so selecting the number of frames to store and size of the external buffer are additional design considerations.

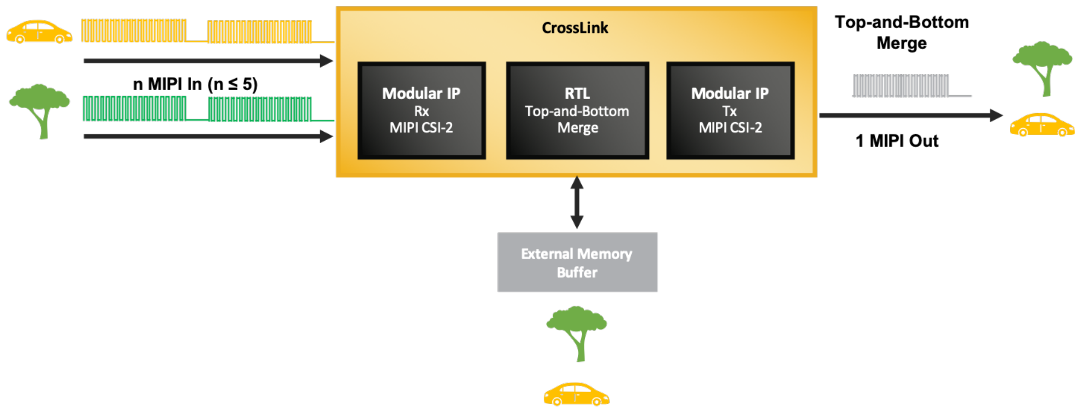


Figure 16. Example of MIPI sensor aggregation featuring top-and-bottom merge

Of particular interest here is that, once we've added an external buffer, in addition to performing top-and-bottom merge, we now have the have the capability to perform other image manipulation tasks, such as rotation and/or mirror-imaging.

The Design Process

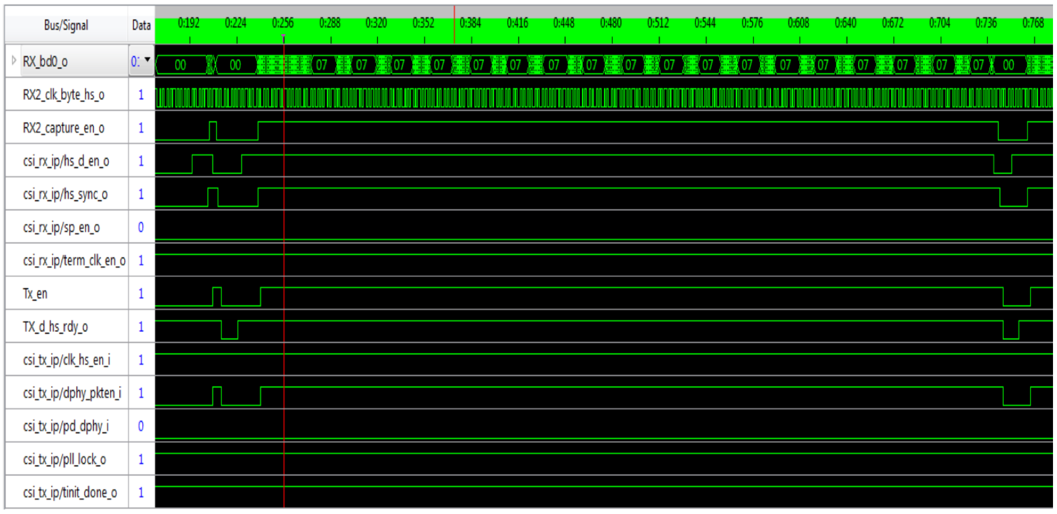
A high-level view of the elements forming the design process is as follows (the numbers associated with each step correspond to the references provided at the end of this paper):

- RTL Design 2
- IP Library 3
- Simulation 2
- Synthesis 2
- Hardware Development Platforms 4,5
- System Debug 2
- Additional Resources 6

Lattice Diamond is where you capture your RTL code, simulate, and synthesize your design. You will start your project by selecting the appropriate CrossLink device. Lattice Clarity is the IP library where you will find all of the CrossLink Video Modular IP we discussed earlier. You use the Clarity interface to select the IP blocks you wish to use and drop them into your custom design.

Lattice Reveal facilitates the debugging of your designs. Reveal has two aspects – the Reveal Inserter and the Reveal Analyzer. The Inserter allows you to define debug signal generators, while the Analyzer allows you to embed miniature logic analyzers into your design, where you can specify the trigger conditions and signals to be monitored.

These signal generators and analyzers will be included in the configuration bitstream that gets loaded into your CrossLink device. The results can be displayed using the integrated signal analyzer (Figure 17).



Bus/Signal	Data
RX_bd0_o	0:
RX2_clk_byte_hs_o	1
RX2_capture_en_o	1
csi_rx_ip/hs_d_en_o	1
csi_rx_ip/hs_sync_o	1
csi_rx_ip/sp_en_o	0
csi_rx_ip/term_clk_en_o	1
Tx_en	1
TX_d_hs_rdy_o	1
csi_tx_ip/clk_hs_en_i	1
csi_tx_ip/dphy_pkten_i	1
csi_tx_ip/pd_dphy_i	0
csi_tx_ip/pll_lock_o	1
csi_tx_ip/tinit_done_o	1

Figure 17. Example screenshot of the integrated debug signal analyzer in Lattice Reveal

Lattice also offers a suite of hardware development boards, including the LIF-MD6000 Master Link Board, which carries a CrossLink chip along with connectors that allow you to connect to various sensors, displays, and SoC/ASSP/AP devices.

There’s also the Video Interface Platform (VIP), which consists of three boards: The Crosslink VIP Input Bridge Board with two image sensors, an ECP5 VIP Board that acts as the image signal processor, and an HDMI VIP Output Board that can be used to display the results (Figure 18).

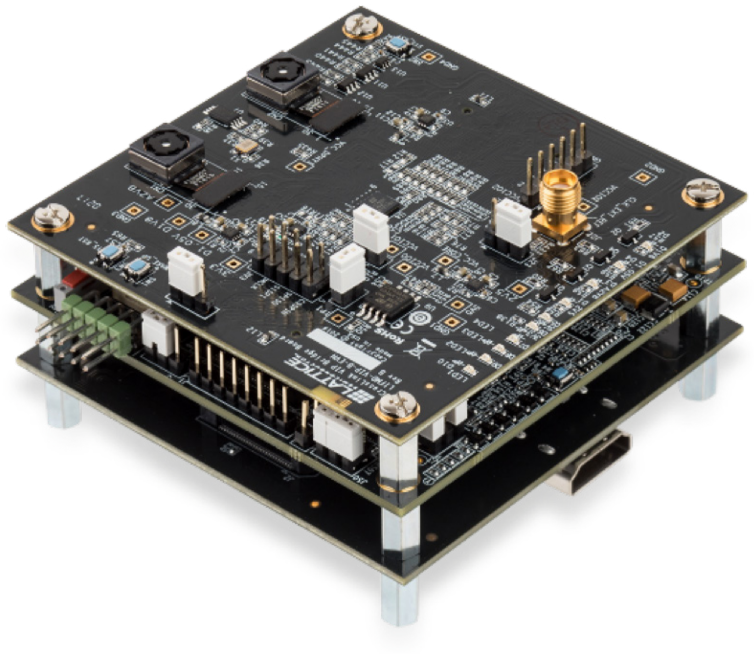


Figure 18. Lattice Video Interface Platform (VIP)

There is also a suite of additional boards you can "add to the stack," including an HDMI VIP Input Board, a DisplayPort VIP Input Board, a DisplayPort VIP Output Board, and a USB3-GbE VIP I/O Board. Together, these give you a complete set of boards to explore and prototype your embedded vision applications based on CrossLink FPGAs.

Last but certainly not least, there are three reference designs available that illustrate various facets of the following applications:

- N to 1 MIPI CSI-2 Virtual Channel Aggregation7
- MIPI DSI/CSI-2 to OpenLDI LVDS Interface Bridge8
- SubLVDS to MIPI CSI-2 Image Sensor Bridge9

We will be adding additional reference designs in the future, so keep on visiting the Lattice website.

Summary

The past few years have seen a dramatic increase in the use of embedded vision. The challenges facing the designers of embedded vision systems include the continual demand for increasing resolution and bandwidth, the trend for systems to use more vision sensors and displays, and the ongoing demand to reduce size and power consumption. Also, there's an increasing demand to add AI/ML capabilities into embedded systems, especially close to the vision sensors themselves.

Low-power CrossLink FPGAs from Lattice Semiconductor are optimized for high-speed video and sensor applications. Augmenting their traditional programmable fabric with hardened PHYs, CrossLink FPGAs provide the industry's fastest MIPI D-PHY bridging solution.

Lattice's product portfolio – which includes chipsets, IP, reference designs, development kits, and software tools – offers flexible solutions to address today's embedded vision designers' needs, such as evolving interface requirements, energy-efficient image signal processing, and hardware acceleration.

References

[1]https://www.clivemaxfield.com/fundamentals-ai-anns-ml-dl-and-dnns/
[2]http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/FPGAandLDS/LatticeDiamond
[3]http://www.latticesemi.com/view_document?document_id=52211
[4]http://www.latticesemi.com/en/Products/DevelopmentBoardsAndKits/EmbeddedVisionDevelopmentKit
[5]http://www.latticesemi.com/en/Products/DevelopmentBoardsAndKits/CrossLinkLIFMD6000MasterLinkBoard.aspx
[6]http://www.latticesemi.com/en/Products/FPGAandCPLD/CrossLink
[7]http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/ReferenceDesigns/ReferenceDesign04/Nto1
[8]http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/ReferenceDesigns/ReferenceDesign03/MIPIDSICSI2OpenLDILVDS
[9]http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/ReferenceDesigns/ReferenceDesign04/SubLVDStoMIPICSI2ImageSensorBridge

Learn more:

www.latticesemi.com

Contact us online:

www.latticesemi.com/contact
www.latticesemi.com/buy